The BioCatHub project: Research data management and digitalisation in biocatalysis

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften

Genehmigte Dissertation

Vorgelegt von

Stephan Malzacher M.Sc.

Aus Waldshut-Tiengen

Berichter:

Prof. Dr. rer. nat. Dörte Rother
Prof. Dr. rer. nat. Jürgen Pleiss
Prof. Dr. rer. nat. Anna Matuszyńska

Tag der mündlichen Prüfung: 13.12.2023

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar

Abstract

In the last decades, biocatalysis has evolved into an important part of white biotechnology. The synthesis of pharmaceuticals or their precursors or fine chemicals for the food and flavour industry are only some examples of products from enzyme-catalysed reactions in industrial processes. This development was mainly enabled by intense research in enzymology and biocatalysis, producing a large amount of data. This data is the result of experiments researchers conducted over the years and served as the basis for the conclusions that enabled an advance in biocatalytic research. Furthermore, published articles describing the experimental data are used as foundation by other researchers to reproduce the published experimental results to further develop biocatalytic or other processes. However, reproducing data, even from peer-reviewed articles, can be difficult to hardly possible, as well as time and resource-consuming. The situation even got described by a specific term: the reproducibility crisis. Reasons for the reproducibility crisis are numerous and diverse, but one important factor is the incomplete or incorrect documentation of experimental results. To counteract the reproducibility crisis, the FAIR (Findable, Accessible, Interoperable and Reusable) principles were developed in 2016 to improve the documentation, storage and publishing of scientific data and thereby enhance the reproducibility of experimental results. Nonetheless, the FAIR data principles are a set of demanding technical guidelines that are difficult to implement which often exceeds the capacity of scientists in the laboratory. In this thesis, a technical infrastructure was developed, realising the data acquisition, storage, publishing and processing of experimental data in biocatalysis based on the FAIR data principles and customised for experimenters in biocatalysis: the BioCatHub project.

In the center of the infrastructure is the BioCatHub-platform. This web application was conceptualised based on the FAIR data principles with a user interface designed for experimental procedures in biocatalysis in a wizard-like structure to make data acquisition as easy as possible for experimenters in biocatalysis. Interfaces to the public databases

BRENDA and SabioRK enable the import of already existing data describing enzymes, as well as enzyme-catalysed reactions, simplifying the data collection process for experimenters even further. Sharing of experimental data with other researchers in the biocatalysis community and beyond is possible with the use of EnzymeML (Enzyme markup language) as machine-readable data storage and exchange format. Furthermore, an interface implemented in the BioCatHub-platform to the public repository Zenodo enables the upload of experimental data to the Zenodo repository, where it can be published. Published EnzymeML documents in Zenodo are annotated with Digital Object Identifier (DOI), by which the collected experimental data is permanently available and unambiguously identifiable and retrievable.
To enhance the implementation of the BioCatHub-platform into the daily work life of experimenters in biocatalysis, the BioCatHub-data-processing (bch-data-processing) repository was developed, providing experimenters with a tool to comprehensibly process and analyse data, collected in the BioCatHub-platform. The bch-data-processing repository is customisable to the data processing requirements of the respective project and can be uploaded to a public repository to make not only the data of the project but also the data processing clearly publicly available.

Biocatalysis is a dynamic field with changing experimental procedures and applications. Therefore, the BioCatHub project was developed with the possibility of extending the data acquisition and integration based on the needs of experimental procedures and novel requirements in the future. With the RetroBioHub extension, experimenters can plan enzyme-catalysed retrosynthetic pathways in the web application RetroBioCat, and seamlessly submit the respective metadata to the BioCatHub-platform. In the BioCatHub-platform, the submitted data is usable for the planning or documentation of new or already existing enzyme-catalysed reactions.
Furthermore, as part of this thesis, a biocatalytic automation platform was developed in which the multistep enzymatic cascade from 3-hydroxy-benzaldehyde to tetrahydroisoquinoline is performed in a fully automated way. Included in the software of this platform

is the data acquisition in which primary data produced during the process is submitted seamlessly together with previously defined metadata to the BioCatHub-platform. With the primary and metadata jointly available in the BioCatHub-platform, the experimenters can review the documented reaction and share, store or publish the collected experimental data as EnzymeML document.

In total, with the BioCatHub project, a technical infrastructure was developed to collect, store, share, publish and process experimental data in biocatalysis based on the FAIR data principles. The individual parts of the BioCatHub project were developed to be as intuitively usable as possible for experimenters in biocatalysis, with the possibility of extending the BioCatHub project to the needs of the biocatalysis community.

Zusammenfassung

In den letzten Jahrzehnten hat sich die Biokatalyse zu einem wichtigen Bestandteil der weißen Biotechnologie entwickelt. Die Synthese von pharmazeutischen Inhaltsstoffen, deren Vorläufern, oder auch von Feinchemikalien für die Lebensmittel- und Aromenindustrie sind nur einige Beispiele für biokatalytisch hergestellte Produkte. Diese Entwicklung wurde vor allem durch die intensive Forschung im Bereich der Enzymologie und Biokatalyse ermöglicht, welche eine Vielzahl von Daten produziert haben. Diese sind das Ergebnis von Experimenten, die Forscher im Laufe der Jahre durchgeführt haben, und dienen als Grundlage für die Schlussfolgerungen, die einen Fortschritt in der biokatalytischen Forschung ermöglicht haben. Darüber hinaus werden die Publikationen, in denen die experimentellen Daten beschrieben werden, von anderen Forschern als Grundlage verwendet, um die veröffentlichten experimentellen Ergebnisse zu reproduzieren und biokatalytische oder andere Prozesse weiterzuentwickeln. Die Reproduktion von Daten, selbst aus Artikeln, welche ein Peer-review Verfahren durchlaufen haben, kann jedoch schwierig bis kaum möglich sowie zeit- und ressourcenaufwendig sein. Diese Beobachtung wurde sogar mit einem eigenen Begriff beschrieben: die Reproduzierbarkeitskrise. Die Gründe hierfür sind zahlreich und vielfältig, aber ein wichtiger Faktor ist die unvollständige oder fehlerhafte Dokumentation von experimentellen Ergebnissen. Um der Reproduzierbarkeitskrise entgegenzuwirken, wurden 2016 die 'FAIR data principles' (Findable, Accessible, Interoperable and Reusable) entwickelt, um die Dokumentation, Speicherung und Veröffentlichung wissenschaftlicher Daten zu verbessern und damit die Reproduzierbarkeit von Versuchsergebnissen zu erhöhen. Allerdings handelt es sich bei den FAIR data principles um eine Reihe anspruchsvoller technischer Richtlinien, welche oft die Kapazitäten der Wissenschaftler im Labor übersteigen. In dieser Dissertation wurde eine Infrastruktur entwickelt, welche die Datenerfassung, -speicherung -veröffentlichung und -verarbeitung von experimentellen Daten in der Biokatalyse, basierend auf den FAIR data principles realisiert und für Experimentatoren in der Biokatalyse angepasst ist: das BioCatHub Projekt.

Im Zentrum der Infrastruktur steht die 'BioCatHub-platform'. Diese Webanwendung wurde auf Basis der FAIR data principles mit einer auf experimentelle Abläufe in der Biokatalyse entwickelten Benutzeroberfläche in einer Wizard-ähnlichen Struktur konzipiert, um Experimentatoren in der Biokatalyse die Datenerfassung so einfach wie möglich zu machen. Schnittstellen zu den Datenbanken BRENDA und SabioRK ermöglichen den Import bereits vorhandener Daten, die sowohl Enzyme als auch enzymkatalysierte Reaktionen beschreiben, was die Datenerfassung für Experimentatoren noch weiter vereinfacht. Der Austausch von experimentellen Daten mit anderen Forschern in der Biokatalyse community und darüber hinaus ist durch die Verwendung von EnzymeML (Enzyme markup language) als maschinenlesbares Datenspeicher- und Austauschformat möglich. Darüber hinaus ermöglicht eine in der BioCatHub-Plattform implementierte Schnittstelle zum öffentlichen Repository Zenodo das Hochladen von experimentellen Daten in Zenodo, wo sie anschließend veröffentlicht werden können.
Die Biokatalyse ist ein dynamisches Gebiet mit sich ändernden experimentellen Verfahren und Anwendungen. Daher wurde das BioCatHub project mit der Möglichkeit entwickelt, die Datenerfassung und -integration auf der Grundlage der Bedürfnisse experimenteller Verfahren und neuer Anforderungen in der Zukunft zu erweitern. Mit der RetroBioHub-Erweiterung können Experimentatoren enzymkatalysierte retrosynthetische Reaktionen in RetroBioCat planen und die entsprechenden Metadaten nahtlos an die BioCatHub-Plattform übermitteln. In der BioCatHub-platform sind die übermittelten Daten für die Planung oder Dokumentation neuer oder bereits bestehender enzymkatalysierter Reaktionen nutzbar.
Darüber hinaus wurde im Rahmen dieser Arbeit eine biokatalytische Automatisierungsplattform entwickelt, in welcher der mehrschrittige enzymkatalysierte Prozess von 3-Hydroxybenzaldehyd zu Tetrahydroisochinolin automatisiert durchgeführt wird. In der Software dieser Plattform ist die Datenerfassung implementiert, bei der die während des Prozesses erzeugten Primärdaten zusammen mit zuvor definierten Metadaten an die BioCatHub-platform gesendet werden. Mit den Primär- und Metadaten, welche in der BioCatHub-

platform verfügbar sind, können die Experimentatoren dokumentierte Reaktion überprüfen und die gesammelten experimentellen Daten als EnzymeML-Dokument freigeben, speichern oder veröffentlichen.
Insgesamt wurde mit dem BioCatHub project eine technische Infrastruktur entwickelt, um experimentelle Daten in der Biokatalyse zu sammeln, zu speichern, zu teilen, zu veröffentlichen und zu verarbeiten, basierend auf den FAIR data principles. Die einzelnen Komponenten des BioCatHub projects wurden so entwickelt, dass sie für Experimentatoren in der Biokatalyse möglichst intuitiv nutzbar sind, wobei die Möglichkeit besteht, das BioCatHub project auf die Bedürfnisse der Biokatalyse community zu erweitern.

Declaration on oath

I, Stephan Malzacher, declare on oath that this thesis and its contents are my own and have been generated independently as a result of my own original research.
I hereby declare on oath:
1. This thesis has been produced in its entirety as a doctor of this faculty and university;
2. If any part of this thesis has previously been used for an academic degree or other qualification at this or any other institution, this has been clearly indicated;
3. If other own or public publications have been used, this has been clearly indicated;
4. If other own or public publications have been used, the source of this has been clearly indicated. This dissertation is entirely my own work, with the exception of such citations;
5. All relevant sources of support have been acknowledged;
6. If any part of this dissertation is based on the collaboration of others, I have clearly indicated what has been worked on by others and what has been worked on by myself;
7. Parts of this work have been previously published in:
From February 2019 until January 2023, this work was conducted at the Forschungszentrum Jülich at the Institute of Bio- and Geosciences – Biotechnology (IBG-1). Parts of this thesis have been published in peer-reviewed journals, at conferences and in supervised students' theses. Any thought, methodology, result and conclusion that is part of an existing own publication is not explicitly referred to, but considered as properly cited by the following list:

Peer-reviewed articles

Rother, D., Malzacher S., Computer-aided enzymatic retrosynthesis. **Nature catalysis**.(2021). Doi:10.1038/s41929-021-00582-5

Lauterbach, S., Dienhart, H.,Range, J., Malzacher, S., Spöring, J.-D., Rother, D., Pinto, M. F., Martins, P., Lagerman, C., Bommarius, A., Høst, A. V., Woodley, J., Ngubane, S., Kudanga, T., Bergmann, F. T., Rohwer, J. M., Iglezakis, D., Weidemann, A., Wittig, U., Kettner,C., Swainston, N., Schnell, S., Pleiss, J. EnzymeML at work: seamless data flow and modelling of enzymatic data. Doi:10.1038/s41592-022-01763-1

Articles in preparation

Malzacher, S., Warmelink, P., Neumann, A., Lohmann, J., Engelmann, F., Range, J., Pleiss, J., Rother, D. BioCatHub: Acquisition of experimental data in biocatalysis based on the FAIR data principles.

Weber, D., Meißner, D., Claaßen, C., Malzacher, S., Rother D., Automated process control of the Light-regulated tetra-hydroisoquinoline cascade with online-analytics via benchtop-NMR.

Students' thesis

Warmelink, Patrizia (2021) Entwicklung einer Plattform zur standardisierten Erfassung und dem Austausch von experimentellen Daten in der Biokatalyse. Master thesis. Technischen Hochschule Köln.

Nina Klos (2023) Investigation of factors influencing the enzymatic transamination towards chiral amino alcohols and resulting reaction engineering concepts. Master thesis.

RWTH Aachen University.

Conference posters

Malzacher, S., Bock, S., Vogel, C., Buchholz, P., Arnold, L., Pleiss, J., Wiechert, W., Pohl, M., Rother, D. (2019). Investigating the sequence-function relationship of thiamine diphosphate-dependent enzymes. BioTrans. Groningen, Netherlands, 2019-07-07 - 2019-07-11.

Malzacher, S., Halupczok, C., Buchholz, P., Pleiss, J., Rother, D. (2019). Setting standards for data aquisition in biocatalysis. COMBINE 2019. Heidelberg, Germany, 2019-07-15 - 2019-07-19.

Malzacher, S., Range, J., Halupczok, C., Buchholz, P., Pleiss, J., Rother, D. (2020). BioCatHub, an online platform for standardised data acquisition in biocatalysis. Amine Biocatalysis 4.0, Stuttgart, Germany, 2020-02-17 - 2020-02-20.

Malzacher, S., Range, J., Halupczok, C., Pleiss, J., Rother, D. (2020). BioCatHub, a graphical user interface for standardised data acquisition in biocatalysis. ProcessNet, virtual, Germany, 2020-09-21 - 2020-09-24.

Malzacher, S., Warmelink, P., Range, J., Lohmann, J., Pleiss, J., Rother, D. (2021). BioCatHub, a platform for FAIR data acquisition in biocatalysis. BioTrans, virtual, Austria, 2021-07-19 - 2021-07-22.

Malzacher, S., Warmelink, P., Range, J., Lohmann, J., Pleiss, J., Rother, D. (2021). BioCatHub, a research data management platform for biocatalysis based on the FAIR data principles. MECP (Multistep Enzyme Catalyzed Processes) 2020+1, virtual, Germany, 2021-09-13 - 2021-09-16.

Malzacher, S., Warmelink, P., Range, J., Lohmann, J., Engelmann, F., Pleiss, J., Rother, D. (2022). BioCatHub: Research data management and digitalisation in biocatalysis. Biocat, Hamburg, Germany, 2022-08-28 - 2022-09-101.

Malzacher, S., Warmelink, P., Range, J., Lohmann, J., Engelmann, F., Pleiss, J., Rother, D. (2022). BioCatHub: Research data management and digitalisation in biocatalysis. Amine Biocatalysis 5.0, Groningen, Netherlands, 2022-11-28 - 2022-11-30.

Conference talks

Malzacher S. (2020). Standardised data acquisition in biocatalysis according to the FAIR data principles. Workshop: Systematic profiling of multicopper oxidases, University of Stuttgart, virtual, Germany, 2020-10-06.

Malzacher S. (2021). BioCatHub, a data management platform for biocatalysis. ESAB (European Society of Applied Biocatalysis) webinar, virtual, Germany, 2021-02-22.

Rother, D., Malzacher S. (2021). Biocathub, a data management platform for biocatalysis. 2nd EnzymeML workshop, virtual, Germany, 2021-05-18.

Malzacher S. (2022). BioCatHub: Research data management and digitalisation in biocatalysis. Biocat, Hamburg, Germany, 2022-08-29.

Malzacher S. (2023). Research data management in biocatalysis. 3rd EnzymeML workshop, Rüdesheim, Germany, 2022-10-18.

Acknowledgement

First, I want to thank Prof. Dr. Dörte Rother and Prof. Dr. Jürgen Pleiss for their excellent supervision during my dissertation. Moreover, I want to thank Prof. Dr. Anna Matuszyńska for reviewing my disssertation. This project has taken me through a variety of disciplines and Prof. Dr. Rother and Prof. Dr. Pleiss always supported me during this time with words and deeds. Furthermore, I want to thank Prof. Dörte Rother who entrusted me with the project and hired me.
I also want to thank Prof. Dr. Wolfgang Wiechert and Prof. Dr. Julia Kowalski for their supervision in the HDSLEE graduate school. HDSLEE allowed me to broaden my horizons and complement my experimental and biocatalysis background with a broad spectrum in data science.

Many of my projects were developed in cooperation. One important partner in this regard is Jan Range, who always was an inspiration and help in terms of data management and research software engineering. Many thanks also to Hannah Dienhart and Mike Wunderlich, who were a great support in developing the modelling approaches in this thesis. I am very grateful to Alaric Prins, Dr. Eduardo Macedo De Melo, Nina Klos for providing and entrusting me with their experimental data.
Moreover, I want to thank Dr. William Finnigan for his great work during the development of the RetroBioHub extension. I am grateful to Dr. Christiane Claaßen, Dr. Douglas Weber, Dominik Meißner, Doris Hahn, Heike Offermann, Jürgen Paulzen, Florian Walkenbach, Prof. Dr. Christian Dick and Luca Fede. Together with them, I was could be part of the LightCas project.

Furthermore, I want to thank Particia Warmelink, Felix Engelmann, Jens Lohmann, Annika Neumann, Felix Döring, Sophie Herrmann and Beyza Bedel for the great work they performed on the development of the BioCatHub project.

Finally, I want to thank all the colleagues from the IBG-1 and the Biokats team. There were a lot of disciplines that melted together and in this project and I am very grateful for the ongoing support I experienced in this great institute.

Contents

1 Introduction 1

1.1 What is biocatalysis? . . . 1

1.1.1 Applied biocatalysis . . . 2

1.1.2 Enzymology and biocatalysis . . . 3

1.2 Reproducibility crisis . . . 3

1.3 Experimental data and data acquisition strategies . . . 4

1.4 FAIR data principles . . . 5

1.5 Machine readable data and EnzymeML . . . 8

1.5.1 Identifiers and resources . . . 10

1.5.2 EnzymeML . . . 11

1.6 The process of software development . . . 12

1.6.1 Programming languages . . . 13

1.6.1.1 Python and object oriented programming languages . . . 14

1.6.1.2 JavaScript . . . 14

1.6.1.3 HTML and CSS . . . 15

1.6.2 Frontend development . . . 16

1.6.2.1 Angular as web development framework . . . 18

1.6.3 Backend development and microservice architecture . . . 19

1.6.4 Communication with HTTP . . . 19

1.6.5 Encrypted communication . . . 21

1.6.6 Deployment and hosting . . . 22

1.6.6.1 Kubernetes . . . 24

1.7 Operational technology . . . 26
1.7.1 Nuclear magnetic resonance . . . 26
1.7.2 Arduino . . . 27
1.8 Aim of this thesis . . . 28

2 Development of the BioCatHub-platform 30
2.1 Requirements engineering . . . 31
2.1.1 The first level of requirements engineering: the vision . . . 31
2.1.2 The second level of requirements engineering: the use cases . . . 32
2.1.3 The second level of requirements engineering: the use cases: requirements specification . . . 33
2.1.4 Requirements regarding the FAIR data principles . . . 35
2.1.5 Requirements regarding usability for experimenters in biocatalysis . 38
2.1.6 Requirements regarding software development . . . 39
2.1.7 Requirements regarding the deployment and hosting . . . 40
2.2 Development of the BioCatHub-platform . . . 41
2.2.1 Development of the BioCatHub data model . . . 41
2.2.2 Frontend development . . . 45
2.2.2.1 Frontend design and usage . . . 45
2.2.3 Backend development . . . 52
2.2.3.1 Reading, writing of EnzymeML documents and the connection to Zenodo . . . 53
2.2.3.2 Querying databases with bch-database-query microservice 57
2.2.3.3 Visualisation of SMILES codes as chemical structures . . . 57
2.2.4 Deployment and hosting . . . 58
2.2.4.1 Docker container as deployment technology . . . 59
2.2.4.2 Kubernetes as container orchestrator . . . 60

2.3 Scenarios . . . 63
2.3.1 Scenario 1: Collection of data from enzyme function regulation studies using alternating magnetic fields . . . 63
2.3.1.1 Outline and target in the scenario . . . 64
2.3.1.2 Data collection and processing . . . 64
2.3.1.3 Collection of data from the Hotzymes project . . . 65
2.3.1.4 Visualisation and processing of data from the Hotzymes project . . . 66
2.3.2 Scenario 2:Reuse of already existing data for the application of a kinetic model . . . 73
2.3.2.1 Outline and target of the project . . . 73
2.3.2.2 Data acquisition and processing . . . 74
2.3.3 Scenario 3: Collection of large data sets . . . 80
2.3.3.1 Outline and target . . . 80
2.3.3.2 Data collection and processing . . . 82
2.4 Conclusion . . . 87

3 Extension: Integration of existing data from online available tool 94
3.1 RetroBioCat . . . 95
3.2 RetroBioHub requirements engineering . . . 98
3.2.1 Vision and use cases . . . 98
3.2.2 Requirements specification . . . 99
3.3 Development of RetroBioHub . . . 100
3.3.1 RetroBioCat and BioCatHub data model . . . 100
3.3.2 Bch-retrobiohub development . . . 101
3.3.3 Bch-frontend modifications . . . 104
3.4 RetroBioHub data exchange workflow . . . 105
3.5 Conclusion . . . 108

4 Implementation:LightCas 113
4.1 The LightCas project . . . 114
4.2 LightCas-implementation requirements engineering . . . 118
4.2.1 Vision and Use cases . . . 118
4.2.2 Requirements specification . . . 118
4.3 Hardware development for the automated cascade . . . 121
4.3.1 Concentration measurement during the automated process . . . 121
4.3.2 Liquid transport and light inactivation during the automated process 124
4.4 Software development of the automated process . . . 125
4.4.1 ABBA: Control center . . . 126
4.4.1.1 Implementation of the concentration measurement . . . 126
4.4.1.2 Implementation of the liquid transport . . . 127
4.4.2 ABBA: frontend . . . 128
4.4.2.1 Automated primary data acquisition and metadata annotation with the BioCatHub-platform . . . 129
4.5 The LightCas process . . . 130
4.5.1 The LigtCas process in a test environment . . . 132
4.6 Automated data collection from the LightCas process . . . 134
4.7 Experiments performed with the LightCas process . . . 135
4.8 LightCas implementation conclusion . . . 138

5 Conclusion 142

6 Outlook 145

7 Appendix 148
7.1 Appendix: Retrobiohub . . . 152
7.2 LightCas . . . 155

Bibliography A-173

List of abbreviations

°C	Degree celcius
3-OH-BA	3-Hydroxy-benzaldehyde
3-OH-PAC	3-Hydroxy-phenylacetylcarbinol
ABBA	The automated biocatalysis and benchtop NMR application
ABTS	2,2′-Azino-bis(3-ethylbenzthiazoline-6-sulphonic) acid
α-MBA	α-Methylbenzylamine
AMF	Alternating magnetic field
API	Applied pharmaceutical ingredient
API	Application programming interface
***Ap*PDC**	*Acetobacter pasteurianus* pyruvate decarboxylase
bch	BioCatHub
***Bm*ta**	*Bacillus megaterium* transaminase
BRENDA	Braunschweig enzyme database
CA	Certificate authorities
CO_2	Carbon dioxide

CPU	Central processing unit
CSS	Cascading style sheet
***Cv*2025**	*Chromobacterium violaceum* amine transaminase
DB	Database
DOI	Digital object identifier
DOM	Document object model
EC number	Enzyme commission number
EnzymeML	Enzyme markup language
eq	equation
FAIR	Findable, Accessible, Interoperable and Reusable
GB	Gigabyte
GPIO	General purpose input output
h	hours
HEPES	4-(2-Hydroxyethyl)-1-piperazineethanesulfonic acid
HIV	Human immunodeficiency virus
HTTP	Hyptertext transfer protocol
HyImPAct	Hybrid processes for Important Precursor and Active pharmaceutical ingredients
ID	Identifier
IDE	Intelligent development environment
IP	Internet protocol

IUPAC	International Union of Pure and Applied Chemistry
JSON	Javascript object notation
LED	Light emitting diode
$MgSO_4$	Magnesium sulfate
min	Minute
mL	Millilitre
mmol/L	Millimole per litre
NCS	Norcolaurine-synthase
NMR	Nuclear magnetic resonance
PLP	Pyridoxal phosphate
PWM	Pulse wide modulation
RAM	Random access memory
RDM	Research data management
REST	Representational state transfer
SBO	Systems biology ontology
SMILES	Simplified Molecular Input Line Entry Specification
SOPP3	Singulet oxygen photosensitizing protein 3
STRENDA	Standards for Reporting Enzymology Data
svg	Scalable vector graphics
ThDP	Thiamine pyrophosphate

THIQ Tetrahydroisoquinoline

TLS Transport layer security

TMSP Trimethylsilylpropanoic acid

URL Uniform resource locator

V Volt

XML Extensible markup language

YAML Y'aint another markup language

Glossary

Terms used in research data management

Primary data Data gained from measurements, for example absorption curves, from measuring enzyme kinetics. Primary data in biocatalysis are usually time-dependent concentration curves of substrates, products or side products

Metadata Information describing primary data generation. Metadata gives context, like reaction conditions, reactants or the biocatalyst used for the generation of the respective primary data

FAIR data principles FAIR (Findable, Accessible, Interoperable and Reusable) data principles, are a set of technical specification developed to establish sustainable management of research data

Negative data Experimental results that do not picture the expectation of the hypothesis of the experiment, e.g. when no product formation takes place in a biocatalytic process.

Positive data Experimental results that picture the expectation of the hypothesis of the experiment

Machine-readable format Structured data format, interpretable by machines (e.g. computers) without further processing required. Examples are XML (eXtensible Markup Language) or JSON (JavaScript Object Notation

Identifier Element (in most cases a combination of chars and/or numbers), for example a Digital Object Identifier (DOI) which unambiguously identifies a published article.

Resource Gerneric term for the description of an entity, that can be described by an identifier. For example, a published article that can be identified by a DOI, or a book which is reffered to by an International Standard Book Number (ISBN).

Repository Deposition possibility for the upload of digital assets. Examples are Zenodo, GitHub or GitLab

Terms used in computer science

Hardware Hardware describes all physical components of an electrical or mechanical system

Software Software defines instructions, data and logic used to control computers, networks or related devices

Web application A software, (e.g. build of HTML, CSS and/or JavaScript) available over a browser, examples are Google Chrome or Mozilla Firefox

Web service A software, available over a HTTP server

HTTP server Software, enabling the communication with the Hypertext Transfer Protocol (HTTP)

Webserver Physical computer, connected to the internet, providing computational resources to provide a web application or web service

Machine code Code in form of ones and zeros, which is understandable and processable by machines

Programming language A definition of commands and rules in a human readable language (e.g. English) which can be compiled (or translated) into machine code

Compiler A software, translating the source code of a programming language into machine code

Frontend The frontend is the graphical user interface of a web application

Backend The backend is the part of the web application which is not visible to the user. The backend processes the requests and inputs coming from the frontend. For example, database queries and the processing of measurement data

Browser Software applied to visualise and interact with web applications

Hypertext markup language (HTML) Markup language defining the structure of web application

Cascading style sheets(CSS) Scripting language defining the styling of a website

JavaScript Programming language used to create dynamic behaviour in web applications

TypeScript Extension to JavaScript enabling the type annotation of variables

Virtualisation Creation of a virtual entity. Often used for the creation of a virtual machine on a computer or server. By this, one server can provide two different operating systems on the same hardware

Container virtualisation A method to provide multiple operating systems in isolated instances (containers) on the same host system. The difference between a virtual machine is that in a virtual machine, the complete ressources of a computer have to be virtualised, which is not necessary for containers

Docker Software used to create and manage containerised environments

Kubernetes Tool used to organise and orchestrate containers in one entity, a so-called Kubernetes cluster

Helm Tool used to simplify the configuration of a Kubernetes cluster

Microcontroller Computer-based system, in which all required hardware components are located on one chip

Arduino Commercially available microcontroller systems

General purpose input-output (GPIO)-Pin Pins embedded in microcontrollers, used as communication interfaces with hardware devices

Terms used in biocatalysis

Chemical reaction Process, in which substances are transformed due to the breaking or reconnecting of chemical bonds

Substrate Initial substance of a chemical reaction

Product Resulting substance of a chemical reaction

Catalyst Structure, for example metals or enzymes, reducing the activation energy and increasing the speed of a chemical reaction

Enzyme Protein based catalyst

Cofactor Collective term for non-protein molecules, required for the catalytic activity of the enzyme. Examples are metal-ions which are permanently bound to the enzyme

Cosubstrate Molecule, required by an enzyme to catalyse a chemical reaction, which is not a permanent part of an enzyme

One-pot reaction Experimental procedure, that is performed in one reaction vessel

1 Introduction

1.1 What is biocatalysis?

In 2000 and 2001, two reviews [1, 2] described biocatalysis as potential and promising technology for industrial applications in the future. In particular, the stereoselective synthesis of applied pharmaceutical ingredients (API) was expected to be of great importance [1, 2]. 20 years later, biocatalysis is indeed established as an important part of the chemical industry in the production of APIs [3, 4]. Using biocatalytic reactions, anti HIV pharmaceuticals as islatravir [5], or metaraminol for hypotension treatment could be synthesised [6]. Metaraminol, can further be applied as precursor for the biocatalytic synthesis of Tetrahydroisoquinolines, which are again important components of pharamceuticals like anti-cancer drugs [7]. However, the predictions regarding the synthetic potential were even surpassed. Nowadays, biocatalysis not only found its way into API production as a standard synthesis catalyst but also into other fields like fine chemical precursors for the fragrance and flavour industry like nootkatone or sweeteners such as aspartame are synthesized using biocatalysis (figure 1.1) [3].
Current literature sees biocatalysis as an enabling technology for the treatment of wastewater, the establishment of closed-loop processes and the carbon neutral synthesis of chemicals [8]. This development of biocatalysis from a potential future technology for the chemical industry at the beginning of the century to an important part of industrial processes and now potential technology for the solution of environmental problems and the development of circular processes is only possible due to intense research and development in this field.

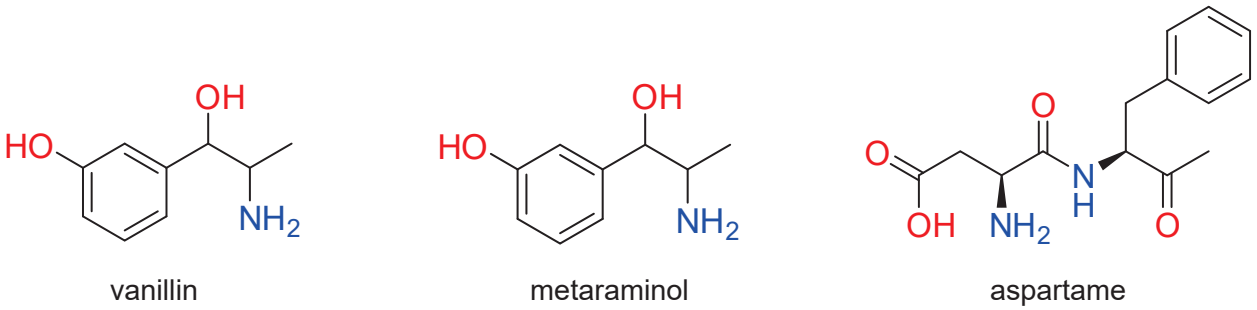

Figure 1.1: Representation of biocatalytic synthesized molecules.

1.1.1 Applied biocatalysis

Biocatalysis as a discipline is subordinated to biotechnology and more specific, white biotechnology [9]. White biotechnology uses living systems and enzymes for application in industrial processes [9]. Enzymes are catalysts made of proteins, reducing the activation energy needed for chemical reactions and increasing the speed of the reaction [10]. Therefore, enzymes in industrial applications provide tools to develop cost and waste-efficient processes [9]. Furthermore, enzymes can produce a desired three-dimensional structural identity of a formed product with high selectivity (stereoselectivity). This is especially important for producing APIs since small differences in the structure of pharmaceuticals can cause undesired side effects in the human body [11]. Moreover, the enzymes used in biocatalytic processes are, in most cases, nontoxic and nonhazardous, biodegradable and can be produced from abundant renewable resources [12, 13].

One important issue in process development is the cost of the process in contrast to the cost of the potential product [14]. When a product can be sold for a low price, the process has to adapt to this. Enzyme production is often one of the main cost drivers for a biocatalytic process. Fortunately, one characteristic of enzymes is a high versatility regarding their formulation [15]. Enzymes are usually produced in microorganisms and utilised from there in multiple forms. Dependent on the needs of the process and financial pressure, they can be purified, applied as supernatant of disintegrated cells or applied still in

the cells as whole-cell biocatalysts in a living, resting or dead state [15, 14]. Furthermore, biocatalysts in form of enzymes or whole-cell biocatalysts can be immobilised on surfaces or particles, where they can also be entrapped [16].

1.1.2 Enzymology and biocatalysis

Literature research about enzyme-catalysed reactions results in two terms, directly connected with one another: enzymology and biocatalysis. Enzymology was defined in a review from 2017 as a discipline focussing on the research of enzymes itself [17]. Biocatalysis on the other hand, aims to use enzymes for the synthesis or transformation of molecules [17]. Although the basis of both disciplines are enzymatic reactions, the research focus differs in some cases from one another. Enzymologists often focus on understanding the molecular mechanism of the enzyme and the performed enzymatic reaction [18, 17]. Therefore, in many cases, kinetic parameters from kinetic models (often the Michaelis Menten model) are estimated to analyse the reaction performed by an enzyme [19]. Researchers in biocatalysis concentrate on the reaction and synthesis or transformation of a target molecule [17]. Economically interesting metrics, like the reaction and space-time yield, the final product concentration or the conversion and selectivity, are also often of interest in biocatalysis [20]. Moreover, since biocatalytic reactions often focus on cost efficiency and industrial applicability, reactions can be performed in non-natural conditions and in different formulations [20, 21]. In contrast, analysis in enzymology rather tries to answer fundamental questions and is mainly done with purified catalysts to avoid possible background reactions.

1.2 Reproducibility crisis

The result yielded by experimental procedures is data. This data can be used as foundation for other researchers to reproduce the experiments in other laboratories to further develop projects, re-analyse data, develop new ideas or compare the data with other re-

sults. Nevertheless, this is of course only possible if the data can be reproduced by other researchers. In 2016 an article published by Monya Baker [22], showed that more than 70 % of researchers were not able to reproduce results published by other researchers, and more than 50 % were not able to repeat their own work. The pressure to publish and selective publishing of only positive results were further identified challenges [22]. Furthermore, a bias against negative data and higher retraction rates of not significant results by journals leads to the missing availability of data showing what is not working [23]. In addition to the missing negative data, lack of detail in the reported data concerning the used material and devices was also found to be of importance [24]. If a closer look is taken into published reports about data from enzyme-catalysed reactions, crucial information for the reproduction of the experiments is missing. An article from 2018 investigated 11 publications reporting data connected with enzymological experiments and concluded that none of those published articles had sufficient information to reproduce the experiments [25].
As possible solutions for the reproducibility crisis, in particular in biological sciences, improved documentation and standardised data acquisition [25, 24, 26] as well as more robust experimental design [22] could help to improve the reproducibility of data.

1.3 Experimental data and data acquisition strategies

When research data is collected, it can be separated into two different categories. First, the primary data, e.g. the product concentration of a reaction, which is produced during the experiments. To better understand the primary data, it is annotated with additional information, so-called metadata [27]. Metadata is, for example, the formulation and amino acid sequence of the catalyst or the applied reaction conditions. In particular biological data is characterised by a high level of heterogeneity due to many different experimental approaches and results [28]. Additionally, much of the metadata describing biological resources is distributed in public databases or laboratory journals. For example, information about chemicals can be found in PubChem [29], information concerning enzymes

and their reactions is stored in SabioRK [30, 31] and BRENDA [32]. Protein structure and sequence data are deposited in the BioCatNet database [33] or the Protein data bank [34], Zenodo and Dataverse provide a source for all kinds of research data [35, 36]. To annotate experiments, metadata from one or multiple sources must be identified, collected, and combined [37]. For this purpose, metadata should be organised and stored in a structured, comprehensive and reusable way [38].
Nevertheless, the situation is further complicated since much data is not publicly available due to restricted access or is available in not modifiable formats like PDF documents [39]. Publishers, institutes and funding agencies have recognised the problems of the reproducibility crisis and the potential of well-structured, organised and reusable scientific data [40, 41]. Therefore, research data management (RDM) plans are demanded in which the collection, storage and publishing of experimental data has to be defined by researchers before a project is started. The RDM plans are developed around a set of guidelines, named the FAIR (Findable, Accessible, Interoperable and Reusable) data principles [42].

1.4 FAIR data principles

In 2016, representatives from research, industry as well as scientific journals defined a set of directives: the FAIR (**F**indable, **A**ccessible, **I**nteroperable and **R**eusable). These were introduced with the goal to provide guidelines for researchers and publishers to collect, store and publish data as sustainably as possible [42]. In total 15 individual guidelines were defined and categorised in the principles **F**indable, **A**ccessible, **I**nteroperable, **R**eusable.

The principle 'Findable' defines guidelines on how primary- and metadata can be found and searched for. Therefore, every set of metadata has to be described with an identifier, pointing exactly and exclusively to this set of metadata (F1). If a set of primary- and/or metadata has been collected (F2) and published in a Zenodo repository [36], a digital

object identifier (DOI) is assigned, for example, 10.5281/zenodo.6090179 tracing back to this data set [43]. If the primary and metadata cannot be stored in the same repository or database, identifiers should be assigned for both and the identifier for the primary data has to be implemented in the metadata (F3). Once research data is published, it still needs to be findable. Therefore, the data set has to be indexed in search engines in order to be findable using search engines (F4).

F-1 (Meta)data are assigned a globally unique and persistent identifier

F-2 Data are described with rich metadata

F-3 Metadata clearly and explicitly include the identifier of the data they describe

F-4 (Meta)data are registered or indexed in a searchable resource

Once data is uploaded and published on a searchable resource, it should be retrievable from there. Therefore, it has to be possible to access the data as defined in the 'Accessible' guidelines with the previously defined identifier (chapter 1.5.1) by an open and free protocol, such as HTTP (chapter 1.6.4) (A1, A2). Nevertheless, some data require limited access, for example data used in industry. In these cases, it has to be possible to restrict access exclusively to authorised personnel by authentication procedures (A3). Finally, if primary and metadata is stored in different repositories or databases, the metadata should be available, even if the primary data cannot be retrieved any more. This can happen since the storage of especially large primary data sets is cost and resource intensive. Metadata usually requires only a small storage volume and can be published to public repositories, and retrieved even if the primary data is not existent any more (A4).

A-1 (Meta)data are retrievable by their identifier using a standardised communication protocol

A-2 The protocol is open, free, and universally implementable

A-3 The protocol allows for an authentication and authorisation procedure, where necessary

A-4 Metadata are accessible, even when the data are no longer available

The 'Findable' and 'Accessible' principles define where data has to be stored, found and retrieved. Another important variable is the format in which data and especially metadata is stored, defined by the 'Interoperable' guidelines. Machine readable formats like XML or JSON (chapter 1.5.2) store data and can be applied directly by machines to read, write or process (I1). Controlled vocabularies, such as the system biology ontology (SBO) can be used to further specify the collected metadata, and define relationships between the data (I2, I3) [44].

I-1 (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.

I-2 (Meta)data use vocabularies that follow FAIR principles

I-3 (Meta)data include qualified references to other (meta)data

The 'Reusable' principles define the requirements to reuse the data as best as possible. At first, data should be annotated with metadata, describing relevant information about how the primary data was generated and processed. For some disciplines, catalogues, e.g. the STRENDA-recommendations [26] are available giving guidelines on which metadata are relevant to annotate primary data (R1, R4). Once data is published, it can be used by researchers in academia or industry to develop new projects or for comparison with other data. Therefore, it should be declared for which purposes others can use the data (R2). Licences for example available through Creative Commons (https://creativecommons.org/)

define if the originator of the data has to be mentioned or if the data can be used for commercial purposes. Finally, it should be defined, if the data has been published before, how it was generated and collected and who was involved in the data processing and collection (R3).

R-1 (Meta)data are richly described with a plurality of accurate and relevant attributes

R-2 (Meta)data are released with a clear and accessible data usage license

R-3 (Meta)data are associated with detailed provenance

R-4 (Meta)data meet domain-relevant community standards

1.5 Machine readable data and EnzymeML

With more and more data available as digital assets, it is important that this data can be read, organised and processed by machines [38]. Therefore, it is beneficial to have structured data formats which can be interpreted without further processing or mappings. Examples are XML (eXtensible markup language) and JSON (JavaScript Object Notation). The structure of the formats is defined by a schema which defines the hierarchy and scope of the respective format.

XML is a text-based and structured format as shown in figure 1.2. It is organised in elements, each of which is defined by a starting (<>) and closing tag (</>), described by the tag names [45]. The element value is encapsulated between the starting and closing tag as string (text representation), number, boolean (true or false values) or one as well as multiple elements [45]. If elements are present as values, they have named child elements with the upper element as parent element. Annotation of elements is possible by the use of attributes, which are defined in the opening tags of the respective element [45]. The first line of every XML document is defined by the prologue. It specifies the version, as well as the encoding to be used. As described in chapter 1.6.1, machines are only able to interpret bits and bytes in form of ones and zeros. This means that also machine readable

formats like XML have to be translatable into machine code. The encoding defines how the XML can be transformed into machine code and back [45].
In more complex XML, it can happen that tag names appear more than once, which is why elements can be organised in namespaces (ns) (Figure 1.2). In this case, the elements are assigned to a namespace which again is identified by an xmlns attribute followed by an identifier unambiguously identifying the namespace in the document. The file extension for XML is .xml [45].

```xml
<?xml version="1.0" encoding="UTF-8"?>

<root>

    <element attribute="attribute 1">
        value
    </element>

    <parent>
        <child> value </child>
    </parent>


<parent xmlns="URL_of_namespace">
    <child> value </child>
</parent>

</root>
```

Figure 1.2: Example representation of the eXtensible Markup Language(XML)

JSON is a structured text based machine readable file format derived from the structure of JavaScript object literals (Figure 1.3) [46]. JSON stores data in a key – value based form, mainly developed for serialisation purposes. Data is stored in JSON in objects, with every object starting and ending with a curly bracket . Every object can contain none, one or multiple comma-separated key/value pairs. Values can be strings, numbers, booleans, objects or arrays. The file extensions for JSON is .json [46].

```
{
    "key":"value",
    "number": 1,
    "string": "value",
    "boolean": false,
    "object":{
                "key":"value"
            },
    "array":[1,2,3]
}
```

Figure 1.3: Example code representing an JavaScript Object Notation (JSON) object

.

1.5.1 Identifiers and resources

With data stored in machine-readable formats as described in chapter 1.5, it is possible to sustainably publish it on publicly accessible repositories like Zenodo. Nevertheless, once data is available on a repository, how are users and machines able to access exactly this data?

For this purpose, identifiers are used [47]. These are for example 'Uniform resource locators' (URLs) which can be used in a browser to access the files required to build a website (chapter 1.6.2). In general, identifiers are unambiguously connected to a resource. In many cases, these resources are available over the internet. 'Resources' is in general an abstract definition for something that can be identified by an identifier [47]. Images, machine-readable files or database entries, can all be representations for a resource.

A big challenge in this regard is besides the unambiguous identification of a resource also permanent identification. These are identifiers, which are created once to describe exactly one resource [43]. For example, digital object identifiers (DOI) are connected to digital resources, the International Standard Book Number (ISBN) unambiguously identifies books and the identifiers.org registry provides enduring URLs for the description of data

in life sciences [43, 27].

1.5.2 EnzymeML

EnzymeML (Enzyme markup language) is an XML-based machine-readable format, designed to store and exchange enzymology ad biocatalytic primary and metadata. The structure of EnzymeML is built upon SBML (Systems biology markup language) and incorporates the STRENDA-recommendations [48, 26, 49]. While the metadata is stored as XML, primary data is deposited in csv files as time dependent concentration or absorbance values. The individual files of EnzymeML are stored in a standardised zip compressed OMEX archive [50], named EnzymeML document (figure 1.4) [49]. The generation, as well as reading of EnzymeML documents, is possible due to a multitude of technologies. A RESTful-API accepts a JSON object based on the EnzymeML schema and converts it into an EnzymeML document. Furthermore, PyEnzyme provides a Python package for reading and writing EnzymeML documents [49]. Moreover, EnzymeML offers a number of application programming interfaces termed thin layers enabling databases and tools to utilise EnzymeML [51]. Data can be collected in EnzymeML, via the spreadsheet, PyEnzyme or the PyEnzyme RESTful-API or extracted from the STRENDA-DB as well as BioCatNet [49]. Modelling platforms like COPASI or PySces [52, 53] are able to utilise EnzymeML documents as input and output format. Kinetic models and estimated parameters can be collected in EnzymeML and uploaded to SabioRK and Dataverse for publication [49].

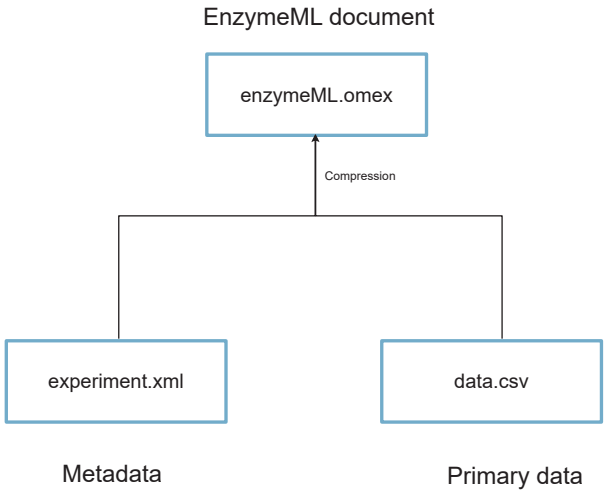

Figure 1.4: The compression workflow of the Enzyme markup language (EnzymeML)

1.6 The process of software development

In general, software can be developed and provided in two main categories: desktop and web based applications [54]. The main difference between both categories is the compartment in which they are running. The individual parts, or components required to provide a desktop application are contained in one computer, mobile phone or device. Web based applications have their components distributed over the internet [55, 54]. As in this work a web based application was developed, the main focus of the introduction will be on the description of those. In web applications, the frontend displays and specialises on the interaction with users by sending tasks to the backend. The backend handles tasks on the server not visible to users and processes data as requested by the frontend [56]. To develop and deliver web applications, the source code for the frontend and backend needs to be developed, deployed and finally hosted as shown in Figure 1.5. In the following chapters, first the software development for the frontend and backend (chapter 1.6.2, 1.6.3), the deployment in Docker (chapter 1.6.6) and finally the hosting in Kuberntetes (chapter 1.6.6.1) is described in detail, as it was performed in this dissertation.

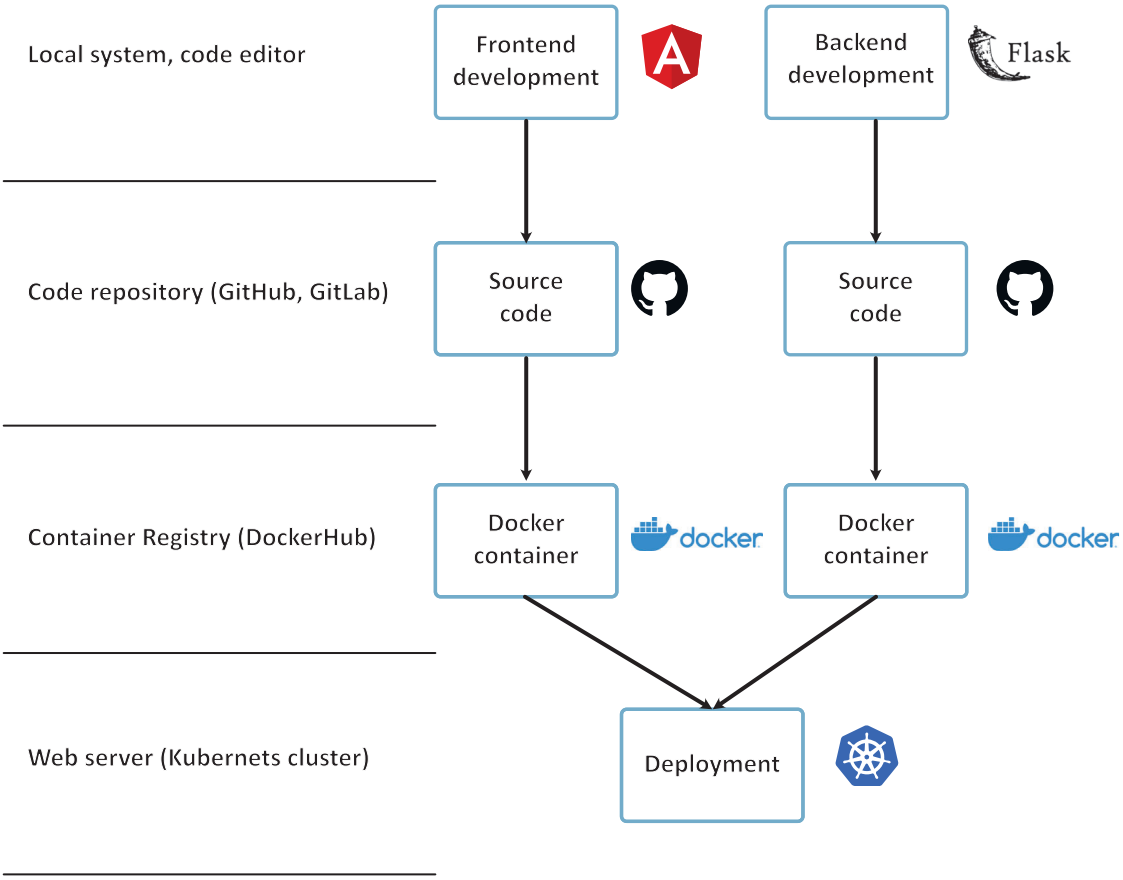

Figure 1.5: The individual steps of web bases software development

1.6.1 Programming languages

Nowadays, computers are present almost everywhere. They can be used for abstract tasks like calculations in the stock market or the weather forecast as well as for mechanical assignments like controlling a valve or a robotic arm and the development of a web application. Nevertheless, communication with a computer is only possible in an abstract binary code, consisting of ones and zeros: the machine code [57]. However, machine code is not readable for humans. Therefore, it needs a translation by a compiler from human readable language into machine code: programming languages [57]. These provide a set of commands and rules in human readable language, named 'syntax'. Using the syntax, programmers can write instruction sets (or programs) which are then translated (or compiled) into machine code [57].

1.6.1.1 Python and object oriented programming languages

Python is a general-purpose object-oriented programming (OOP) (see below) language. It is used for example in scientific data processing [58] and visualisation [59] as well as in web development [60]. Object orientation is an abstract paradigm in programming describing the architecture in form of 'classes' and 'objects' (Figure 1.6) [60]. Therefore, 'classes' are defined as blueprints comprising 'attributes' and 'methods'. 'Attributes' are variables available in the 'class' and 'methods' are defining the tasks that can be performed in the scope of a class. An object is present once a class is instantiated and the attributes are assigned with actual values. OOP provides the benefit that the code in the classes can be reused for multiple objects. Furthermore, the code present in 'classes' can be inherited to other classes, allowing advancement of the code in other classes without the need to modify the already existing code [60].

```
class Person():

    def __init__(self, first_name, last_name, job_description):
        self.first_name = first_name
        self.last_name = last_name
        self.job_description = job_description

    def describe_person(self):
        print(self.first_name+" "+self.last_name+" "+self.job_description)

Stephan = Person("Stephan", "Malzacher", "PhD-student")
Stephan.describ_person()
```

Figure 1.6: Example representation of Python source code in the object oriented programming paradigm.

1.6.1.2 JavaScript

Like python, JavaScript is an OOP yet in contrast to Python not used as general purpose programming language but specified mostly in web applications [57]. JavaScript is usually applied in combination with the hypertext markup language (HTML) and cascading style sheets (CSS) (chapter 1.6.1.3) and provides dynamic behaviour to the web application [57]. This is possible as JavaScript is executed in the browser and can interact, change, create or delete elements in the document object model (DOM) (chapter 1.6.2) [57].

For a long time, however, JavaScript had the disadvantage of only beeing processable in the browser. As consequence, web developers had to construct web application in HTML, CSS and JavaScript as well as another programming language to establish an HTTP-server. These servers store and provide HTML, CSS and JavaScript to the browser once a URL is navigated. With the development of Node.js, it is possible to use JavaScript for the development server-side tasks as well. This created the possibilities to develop web application on the browser and server side with JavaScript [57].

1.6.1.3 HTML and CSS

The hypertext markup language (HTML) is used especially in web development and describes the architecture and content of a website. The structure of HTML is similar to that of XML (chapter 1.5). However, one difference is that XML is mainly used for data transfer, meaning that the tags can be altered based on the data that must be exchanged. In HTML, tags are predefined since they have to be compatible with the browser [61]. The structure of HTML documents is separated into a head and a body (Figure 1.7). The head contains metadata about the HTML documents, like the title of the HTML document. The title in the header is not implemented on the website itself but displayed in the tab of the browser. Furthermore, the path of CSS scripts and JavaScript source code to be used can be defined in the head [61].
While HTML defines the structure of a website, CSS describes the styling. For example the color of displayed text, the font type or background color of the whole website or individual elements can be adapted using CSS (Figure 1.8)[62]. CSS code can be implemented directly into the body of HTML or defined as a separate file and imported in the head of the HTML document (Figure 1.7) [62].

```
<!DOCTYPE html>
<html>
<head>
    <title>My dissertation</title>
    <link rel="stylesheet" href="./dissertation_style.css">
    <script src="myscripts.js"></script>
</head>
<body>

  <p>
  Dissertation of Stephan Malzacher
  </p>

</body>
</html>
```

Figure 1.7: Example representation of a hypertext markup language (HTML) document.

```
p {
    color:blue
}
```

Figure 1.8: Example representation of a cascading style sheet (CSS).

1.6.2 Frontend development

Besides the backend, the frontend is the main unit of web-applications and is usually accessible over a browser [63]. Browsers like Google Chrome or Mozilla Firefox serve users as gateways to resources and functionalities available on the internet [64]. Nowadays, browsers are able to visualise static contents like the website of a company, serve as user interface for gaming or display media content [63]. The central element of a browser is its engine. It addresses URLs (uniform resource locators) to receive for example HTML (hypertext markup language), CSS (cascading style sheets) and visualise the submitted content to users. Browser engines use the document object model (DOM) to build and display the required contents (Figure 1.10) [63].

The DOM serves as structure for the browser engine, supplying information of how to display the respective contents. Using an URL, a resource is requested in form of HTML, CSS and JavaScript code by the browser (chapter 1.5.1). The received HTML is transformed

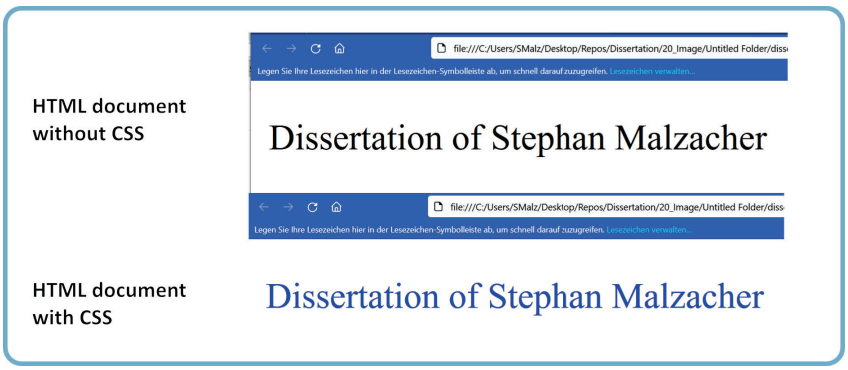

Figure 1.9: Representation of a browser window based on HTML and CSS. (a) shows the browser window if it's only based on HTML, (b) shows the same, but styled with CSS

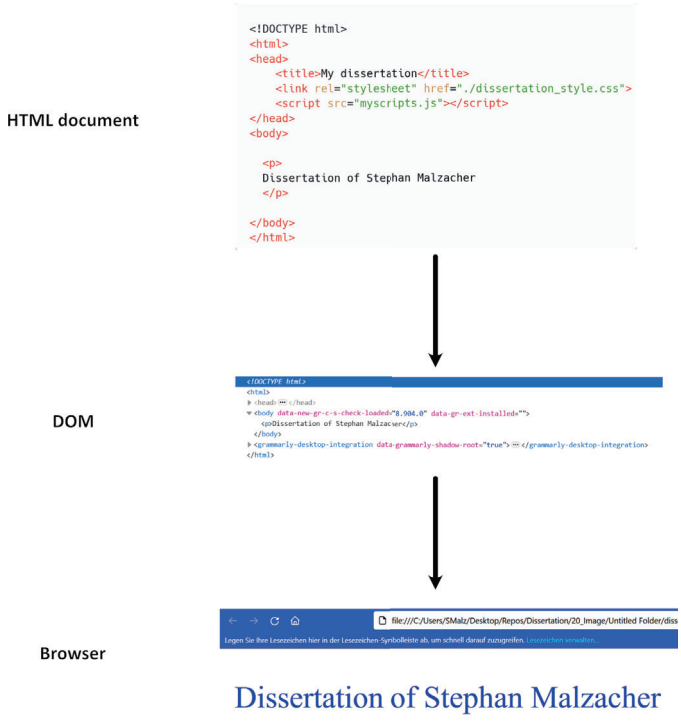

Figure 1.10: Visualisation process of a HTML document in a browser.

into the DOM and then further modified by Javascript and CSS. This DOM is then visualised and the final output is displayed in the browser [63]. Especially, if web applications grow larger, the amount of HTML, CSS and JavaScript code increases, making the development process more complex. Therefore, web development frameworks like Angular were established offering a structured and clear development environment (chapter 1.6.2.1).

1.6.2.1 Angular as web development framework

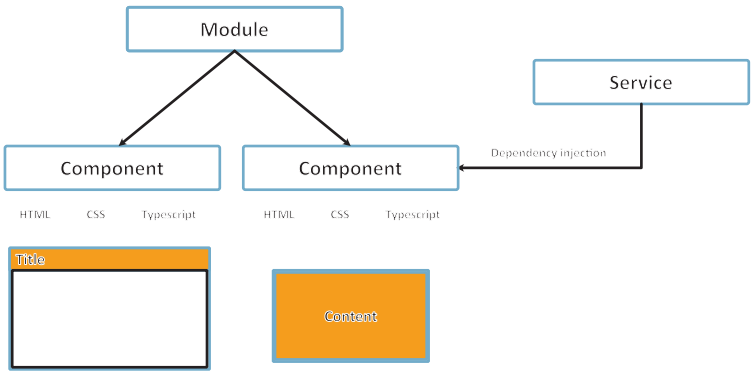

Figure 1.11: Architecture of an Angular project

Web applications in Angular are developed in a modular system called NgModules [65]. In the modules, the individual parts of the web application are again organised as components. Each component contains TypeScript code, connected with a HTML template and CSS resulting in a view. TypeScript is an extension to JavaScript in which the values and variables are annotated by their type (e.g. number) [66]. By this, the individual views can be developed and tested separately, creating a clear structure. For example in Angular it is possible to develop the title and content of a web application in two different components (Figure 1.11). In each component, the developer can concentrate on the specific needs, the title as well as the content component requires. Once the development is finished, both resulting views are combined in the module. In addition to components, services are provided by Angular to outsource tasks from components. These are often used for tasks needed by multiple components like requesting information from a web service (chapter 1.6.3).

1.6.3 Backend development and microservice architecture

With a frontend developed in a web development framework, the communication between the user and the system is created. However, modern web applications often require complex processes like data processing or authentication procedures which are handled by the backend. Backends in web applications are often developed as web services which realise a client-server interaction between machines and applications (e.g. frontend to backend) [67]. These can be established in two different architectures: monolithic and microservice-based. Monolithic architectures encapsulate all the functions in one service, while microservices separate individual tasks and provide individual services for individual tasks [68].

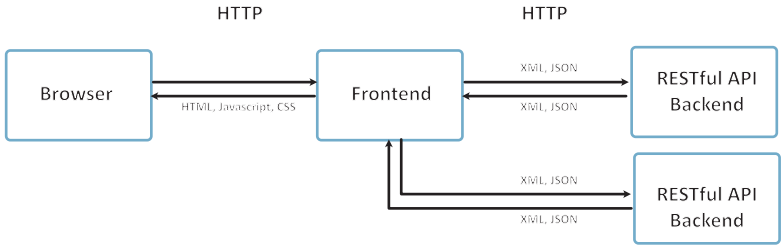

Figure 1.12: Communication schema in monolithic *versus* microservice based architectures.

One frequently used design pattern for web services is REST (representational state transfer). REST is not a technology itself, but a set of guidelines with the target to provide platform independent web services [67]. Nowadays, RESTful APIs are realised by implementing HTTP and unifying the communication between for example frontends and backends. These RESTful APIs can be addressed by URLs together with the HTTP methods and exchange data with machine readable formats like XML or JSON. Therefore, RESTful APIs can be developed and deployed independently of programming languages (e.g. Java, Python, C#) or operating systems, respectively [67].

1.6.4 Communication with HTTP

In a web application, the frontend and backend applications contain the functionalities required to fulfil the requirements specified to the software. One important part in these

applications is the communication between the frontend and backend, or the frontend and the browser. For this purpose, the HTTP (Hypertext transfer protocol) is utilised [69, 70]. HTTP is designed for data exchange in the client server model, meaning that a client sends a request and the servers responds to it [69, 70]. HTTP messages are used as data exchange format. Every message consist of a start line, a header, a blank line and a body [69, 70] (Figure 1.13).

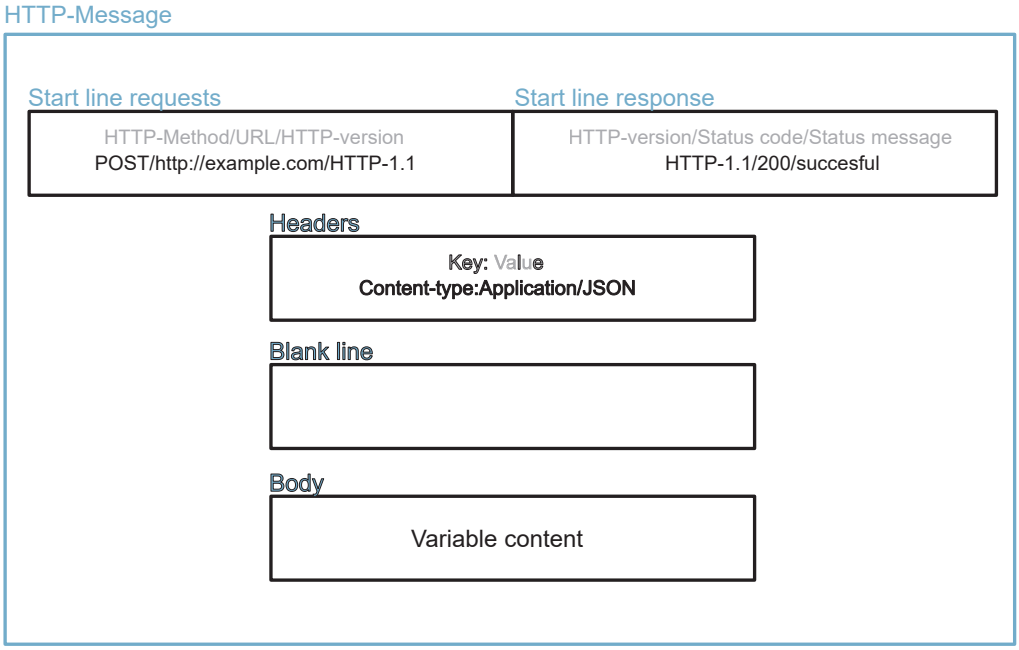

Figure 1.13: Visualisation of a HTTP message

In the case of client requests, the start line consists of three elements: First, a HTTP method, telling the server, what kind of task has to be performed (Table 1.1) [69, 70]. Second, the target of the request, provided as URL and the version of HTTP as third element. If a response is sent by the server, the start line consists of the HTTP version, a status code and a status message (Figure 1.13). Following the start line is the header which is defined as key:value pairs to annotate the message. The header can describe the content which is transported by the message, define the server type used for the request or annotate the size of the content [69, 70]. After the header follows a blank line, indicating that the header is finished and the body begins. HTTP bodies are used to transport multiple kinds of information, like HTML, JSON, XML or even files. Some HTTP methods like GET do

not require a body, since this method is mostly used to retrieve data or documents from a server [69, 70]. The messages are then exchanges on a technical level using the transport control protocol (TCP) together with IP addresses to identify the individual communication partners [69, 70].

Table 1.1: Method of HTTP requests

Method	**Function**
GET	Retrieve resource
POST	Create or process resource
PUT	Create or update resource
DELETE	Delete a resource

1.6.5 Encrypted communication

During the communication between a client (e.g. browser) and a server, sensitive information like bank account details, or medical history information can be exchanged. To ensure that the transferred data can only be read by the client and server, the exchanged messages are encrypted using Transport Layer Security (TLS) [71]. HTTP is named Hypertext transfer protocol secure (HTTPS) if the communication between client and server is additionally secured by TLS [71]. This protocol fulfils two tasks: the authentication of the server and the encryption of the message (Figure 1.14) [71]. As the first communication between the server and the client is completed, the client sends the 'hello-client message'. In this message the client shows the TLS version it is using.
Subsequently, the server sends its respective certificate in the 'hello-client message'. The server can purchase certificates from certificate authorities like let's encrypt [72]. Certificate authorities (CA) test the server based on different criteria defined beforehand by the CA, before the certificate is created. The client can verify if the certificate provided by the server was granted by a trustworthy CA and use certificate to encrypt a sequence of bytes which can only be decrypted by the server: the 'premaster key'. The encrypted 'premaster

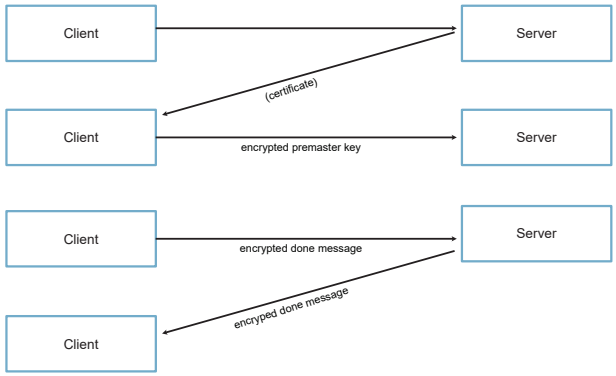

Figure 1.14: HTTP communication with transport layer security (TLS) .

key' is submitted to the server and the client as well as the server use the 'premaster key' to create a key which can be used to send encrypted communication: the 'session key'. Once the 'session key' is generated, a done message as check is sent from the client to the server, encrypted with the 'session key' and the other way around. With the session key generated in the client as well as the server, enrcypted messages can be submitted from the client to the server and *visa-versa*.

1.6.6 Deployment and hosting

As soon as software is developed and the source code for the frontend web application as well as the backend web services are available, hardware systems are needed, providing an operation system [73]. These hardware systems are commonly referred to as web-servers. With a microservice based backend architecture, numerous different software environments are required. If one microservice is coded in a Python 2 version and another microservice requires a Python 3 version, conflicts arise on the webserver since both Python versions would have to be installed. Therefore, the deployment of web applications and web services is often realised in containerised environments [73]. Each container is a customisable virtual environment in which the operation system as well as supporting software can be adapted as requested by the web service or web application. Containers are generated using a respective container technology an easy-to-use, well-documented and widely used container technology is docker [73].

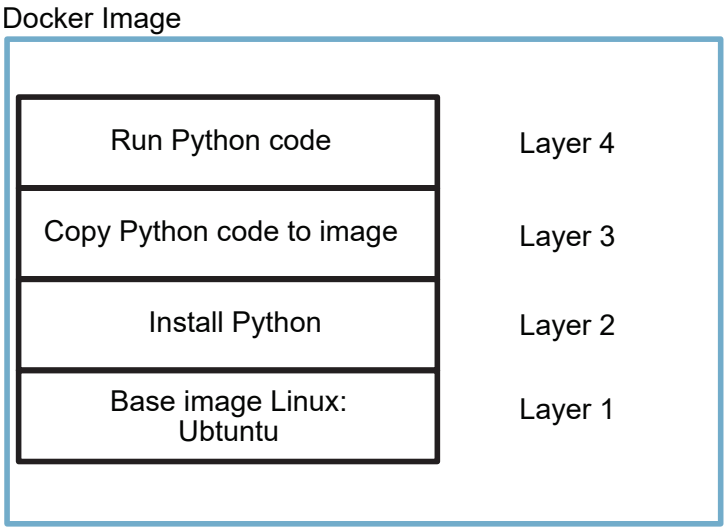

Figure 1.15: Architecture of a docker image.

To run a docker container, an image is required as blueprint for Docker to run the container. Images are files in which the architecture of the container is defined in layers built on top of each other. Docker images can be generated by supplying a machine readable YAML (Y'aint another markup language) [73] containing the commands required to build the image. An example of this is displayed in (Figure 1.15). The Docker image starts with an already existing image, the base image. This enables the re-use of existing images without the need to start from scratch each time [73].
One example for a base image is the 'ubuntu-image' providing functionalities of the ubuntu-linux operation system for the image [74]. Every command like the installation of Python, the copying of files and the execution of scrips is performed as a separate process in layers providing a docker image as final result [73]. A linux-based service which handles the building of images, the starting, stopping as well as the supervision of containers is the docker daemon (Figure 1.16).
The communication with the docker daemon is possible via HTTP over a RESTful API provided by the docker daemon. It can receive commands locally by a docker client on the server it is running on, or remotely over a network. Furthermore, it can be connected to a containter registry like DockerHub [75] to retrieve or upload images.

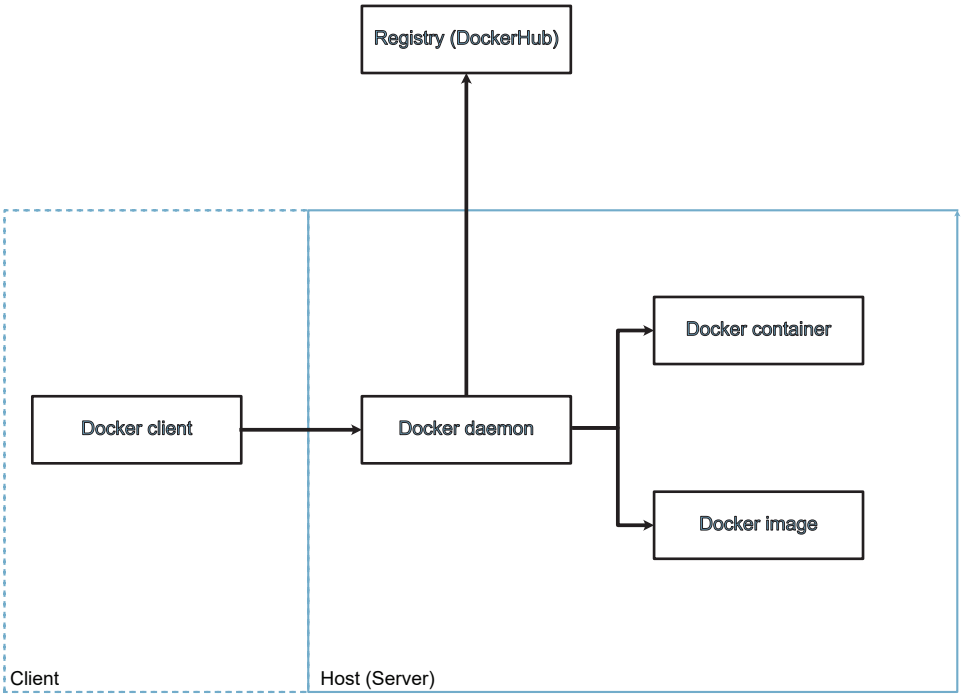

Figure 1.16: Functional components of docker.

1.6.6.1 Kubernetes

Once software is developed in a microservice-based architecture and containerised in docker containers, these can be deployed to the host system. They are physical web servers running the microservices and building the connection to the internet. However, with the scaling of a microservice-based architecture, also the number of containers grows, resulting in a larger effort for the provision and organisation of the web application. For that purpose, container orchestrators are available, handling the organisation and communication between the containers as well as the connection from external networks like the internet [76]. One popular tool providing these functionalities is Kubernetes [76]. An instance of Kubernetes is a Kubernetes cluster to which the container can be deployed and hosted[76].

A Kubernetes cluster is organised as an internal network in which containers are deployed in Pods (Figure 1.17) [76]. Every Pod can comprise one or multiple containers, each sharing one internal network IP address [76]. Since the Pods are organised in the Kubernetes

cluster in an internal network, an additional instance is needed to make the Pods available for an external network, for example the internet. For this purpose, an Ingress is applied. If a HTTP request is made from an external network to a container in the Kubernetes cluster, it is received by the Ingress and forwarded based on the URL to the respective Pod in which the container is applied. Since the Ingress is applied on the interface between the internal network of the Kubernetes cluster and external networks, is also an important element concerning the security of the Kubernetes cluster. Multiple security measures can be configured in the Ingress, for example the application of an encrypted communication using TLS [77] [76] (chapter 1.6.4). If Pods are newly created, updated or renewed, they are assigned a new internal IP address. This means the IP addresses of the pods can change regularly. Therefore, the Ingress does not communicate directly with the pod but communicates over an additional instance: the Service [76]. These are assigned to a specific Pod and serve as a connecting element. If a request for a containerised application is received by the Ingress, the Ingress further communicates with the respective Service, which then again communicates with the Pod in which the container is applied [76].

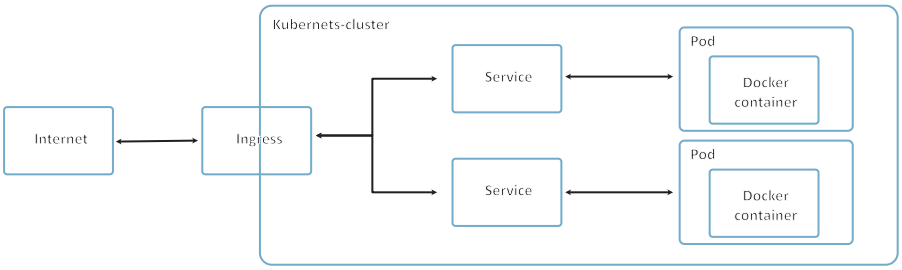

Figure 1.17: Schematic overview over a kubernetes cluster.

The configuration of a Kubernetes cluster is made by defining the individual elements, like Services, Ingress and Pods in configuration files with a predefined syntax by Kubernetes [76]. The submission of the configurations is possible over a RESTful API which is provided by the Kubernetes cluster. Once a Kubernetes cluster grows larger, a tool is available in which the configurations of the Kubernetes cluster can be bundled and applied with one command: helm [78]. This additional abstraction makes the configuration

of a Kubernetes cluster simpler, as the individual configurations are encapsulated and can be applied with one command, in particular if the Kubernetes cluster grows larger.

1.7 Operational technology

1.7.1 Nuclear magnetic resonance

One important part of every biocatalytic experiment is the qualitative and quantitative detection of reagents of interest. This can be performed with numerous methods, including chromatography, mass spectronometry or nuclear magnetic resonance (NMR) [79]. NMR has the advantageous property that the sample material is not destroyed and can be transferred back to the reaction vessel making it, especially in benchtop devices, suitable for the continuous monitoring of biocatalytic reactions [80]. NMR spectroscopy monitors molecules by measuring the nuclear spin frequency due to the exposition to an external homogeneous magnetic field. The principle is that molecules like ^{1}H, $^{1}3$C, $^{1}5$N or ^{31}P, which have a nuclear spin of unequal 0, can set their spin either parallel or antiparallel to the external homogeneous magnetic field [79, 81]. These spin states differ in their energy levels, meaning that one state has a higher energy level than the other. With the application of energy in form of electromagnetic radiation, the nuclei can be transferred from the lower to the higher energy level. The energy difference can be measured as the frequency of the applied radiation. The frequency of every nucleus depends on the chemical environment [79, 81]. If for example hydrogen (^{1}H) nuclei are measured, the frequency is different based on its bindings and surrounding atoms. Therefore, every measured frequency is processed with a reference substance yielding the chemical shift. By this, the environments each of the measured ^{1}H nucleus can be estimated, allowing an structural analysis of the measured compound [79, 81]. Moreover, besides the structural analysis of molecules, the quantitative analysis of reactants in solutions is possible. Therefore, if separated signals of the desired compounds can be identified, an NMR can also be used as online measurement device [80].

1.7.2 Arduino

Microcontrollers are computer systems containing the processor as well as peripheral components on one chip. These are often used in embedded systems for a multitude of tasks, for example to control devices like a peristaltic pump or to read out sensors like a thermometer. Arduino is an open source platform aiming to provide microcontrollers (Arduino boards), software and devices to build digitally controlled electronic infrastructures [82]. The communication with devices is possible due to the general purpose input output pins. These can realise the communication over digital, analogue as well as pulse wide modulation signals. The analogue output and input pins can measure or produce voltages between a low state (0V) and a high state (3,3 or 5 V). In contrast to the digital pins, the analogue pins can measure any value between the low and high state. To produce analogue signals, pulse wide modulation is applied to represent the digital values as analogue ones [83, 82]. Arduinos are programmable using the Arduino intelligent development environment (IDE) in which the required code can be scripted using C++. The IDE is built in functionalities to compile and upload the code to the respective Arduino board [82]. For continuous communication with the Arduino board, the firmata protocol enables communication with code running on a host computer, for example using an interface programmed in Python [84].

1.8 Aim of this thesis

Sustainable digitalisation of research data in biocatalysis is a challenge for researchers in the generations to come. As previously described, the reproducibility crisis, the increase in research data as well as the heterogeneity of data creates a complex mixture of challenges. The FAIR (**F**indable, **A**ccessible, **I**nteroperable, **R**eusable) data principles, data formats like EnzymeML (Enzyme markup language) and public repositories like Zenodo are available to improve the storage, publication and reusability of research data. Unfortunately, the implementation of the FAIR data principles in the everyday work life of experimenters is time as well as labour-intensive and therefore often not feasible, and no common standardisation tool for biocatalysis is available.

The goal of this thesis is to develop sustainable workflows to collect, store, publish and use experimental data in biocatalysis in an experimenter-friendly way: the BioCatHub-project. Therefore, three targets are aimed for. First, to develop the BioCatHub-platform, to collect, store and publish experimental data in the machine readable format EnzymeML. The BioCatHub-platform should support experimenters during the data processing with an intuitive user interface and data integration possibilities from publicly available databases. Second, the BioCatHub project provides reusable and comprehensible data processing possibilities. By this, data cannot only be collected and published, but experimenters can visualise and apply their output from the BioCatHub-platform to extract new conclusions from their experiments, also to a later point in time. Furthermore, the direct interface between the collection and usage of data simplifies the usage for experimenters. Third, the implementation of other data sources improves the quality of the collected data and further simplifies the data acquisition process. In particular, two sources are to be named. The first is the online tool RetroBioCat, containing information about enzyme-catalysed retrosynthesis pathways. With the seamless integration into the BioCatHub-platform an improved data acquisition process is aimed for, simplifying the usage and avoiding human copying errors. The second source is the integration of experimental data directly from the laboratory. As an example, a direct connection between the LightCas project, funded

by the European Commission, and the BioCatHub-platform reduces the necessary workload for data acquisition to a minimum and also reduces errors.

In total, the BioCatHub-project is an interdisciplinary research project combining research data management, biocatalysis as well as research software development. The overall goal is to create workflows for experimenters to easily collect and publish their positive as well as negative data in a standardised format, so that data scientists or other researchers can seamlessly reuse it.

2 Development of the BioCatHub-platform

Author's contribution

Stephan Malzacher performed the requirements engineering in chapter 2.1. The initial development of the *bch-frontend* web application (chapter 2.2.2), as well as the initial version of the *bch-database-query* microservice (chapter 2.2.3.2) was conceptualised by Stephan Malzacher and developed by Patricia Warmelink as part of her Master's thesis which was supervised by Stephan Malzacher. The *bch-database-query* microservice was developed by Stephan Malzacher, based on the work of Patricia Warmelink. The visualisation of primary data was further improved by Jens Lohmann in the *bch-frontend* web application under the supervision of Stephan Malzacher. The *bch-enzymeml* (chapter 2.2.3.1) microservice was developed by Stephan Malzacher. The *bch-smiles-to-structures* microservice (chapter 2.2.3.3) was developed by Felix Engelmann as part of his part-time work on the BioCatHub project under the supervision of Stephan Malzacher. The deployment and hosting (chapter 2.2.4) of the BioCatHub-platform was done by Stephan Malzacher. The *BioCatHub-data-processing* repository and the software development in Scenario 1-3 (chapter 2.3.1, 2.3.2, 2.3.3) was realised by Stephan Malzacher. In Scenario 2, advice was sought from Hannah Dienhart, Mike Wunderlich and Nina Klos. The data for Scenarios 1, 2 and 3 were provided by Nina Klos, Eduardo Macedo de Melo, and Alaric Prins, respectively.

2.1 Requirements engineering

The most critical measure of success in software engineering is how well the developed system fits the purpose it has to serve [85]. Therefore, it is essential to define the constraints and possibilities the provided system is supposed to comprise before the development starts. For example, in the case of a research data management platform software developers need to know which data has to be collected, how it has to be structured and where it must be stored. Furthermore, as experimenters and scientists interact with the system, custom ways must be identified, making it as user-friendly as possible. Therefore, experimenters and software developers need to specify the user interface's design and which functionalities have to be implemented behind the user interface. On the other hand the system's technical limitations, costs and maintenance need to be considered. Requirements engineering describes how scientists and software engineers identify, document, review and manage the individual requirements to realise the project.
For the BioCatHub-platform, a customised requirements engineering process was developed according to recommendations in [86].
In this, three levels were defined, each increasing the degree of detail. The first level covers the description of the vision. It represents the targets to be achieved without mentioning details. The second level is the use case description which describes how experimenters can interact with the system and which benefits the system provides. Based on the first two levels, the requirements can be specified in detail. Therefore, the requirements are categorised and specified in a tabular form in the third level. The in-detail process is presented below.

2.1.1 The first level of requirements engineering: the vision

The first task is the definition of the vision. As described in chapter 1.3, scientific data is often not publicly accessible and predominantly negative data is often not published at all. Moreover, available data is usually not present in a standardised and structured FAIR data principle-based way, limiting its reproducibility. Therefore, the reproduction

of published data costs time, resources and work. Not published negative data even cause repetitive reproduction of negative data. The vision behind the BioCatHub-platform is to create a tool for experimenters in the biocatalysis community to collect, store and publish their data in a structured, organised, comprehensive and machine-readable way based on the FAIR data principles. In addition, the BioCatHub-platform should be integrated into the daily work of the experimenters without generating a large additional workload.

2.1.2 The second level of requirements engineering: the use cases

Once the vision was defined, use cases could be defined in which the interaction and the benefit of users and experimenter with the system is declared (Figure 2.1).

Use case 1: The BioCatHub-platform should provide a user interface and make data acquisition based on the FAIR data principles as easy as possible. Furthermore, it should assist experimenters in the data collection process, striving for completeness.

Use case 2: Besides the collection of data, the BioCatHub-platform is required to provide the possibility to create reusable templates. By this, data can be reused once collected, preventing repetitive data collection and unnecessary time and work efforts.

Use case 3: With data collected in a standardised way, it has to be storable on the local file system or publishable to an online available repository. Once published, other experimenters can reuse it. For this purpose, the database needs to be retrievable by an identifier like a digital object identifier (DOI) (chapter 1.5.1). If data is not meant for publication, for example due to cooperation with industrial partners it should still be storable on the local file system.

Use case 4: Numerous public databases provide information, particularly metadata, about biocatalytic experiments. Therefore, it should be possible to integrate metadata from these databases seamlessly during the data collection.

Use case 5: The BioCatHub-platform addresses the biocatalysis community. Therefore, it should be possible to seamlessly exchange data between experimenters in different working groups or institutes and use the BioCatHub-platform independently from the respective in-house operating system.

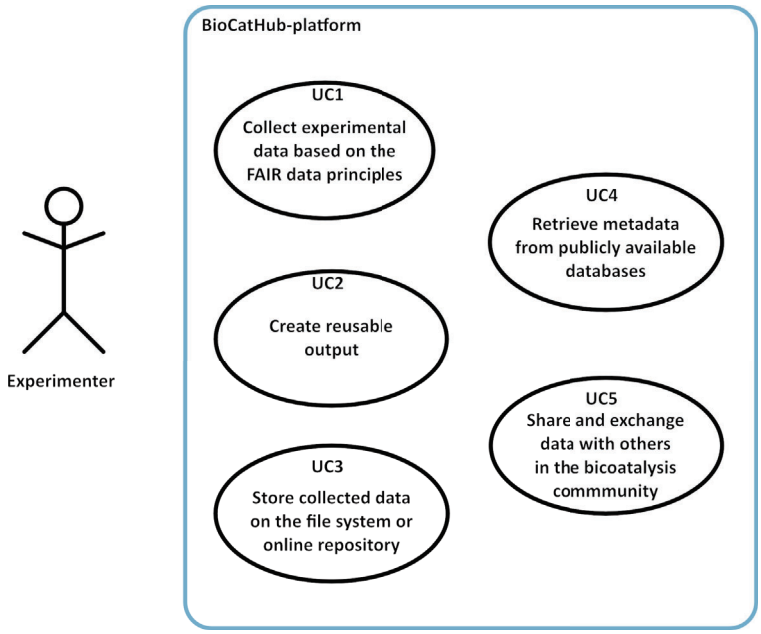

Figure 2.1: Use case diagram for the requirements of the BioCatHub-platform

2.1.3 The second level of requirements engineering: the use cases: requirements specification

The vision and the use cases provide a framework to define individual requirements in detail. In the case of the BioCatHub-platform four distinct categories of such requirements were identified (Figure 2.2). First, the FAIR data principles provide requirements regarding data acquisition, storage and sharing (**UC1, UC3, UC4, UC5**). Second, since the BioCatHub platform was conceptualised for data acquisition in biocatalysis, it should be developed to be user-friendly for experimenters in this specific field (**UC1, UC2, UC5**). Third, the developed software must be manageable and extendable within the restrictions

given by personal management in research. Last, once the software is created, it has to be deployed and hosted in a maintainable way.

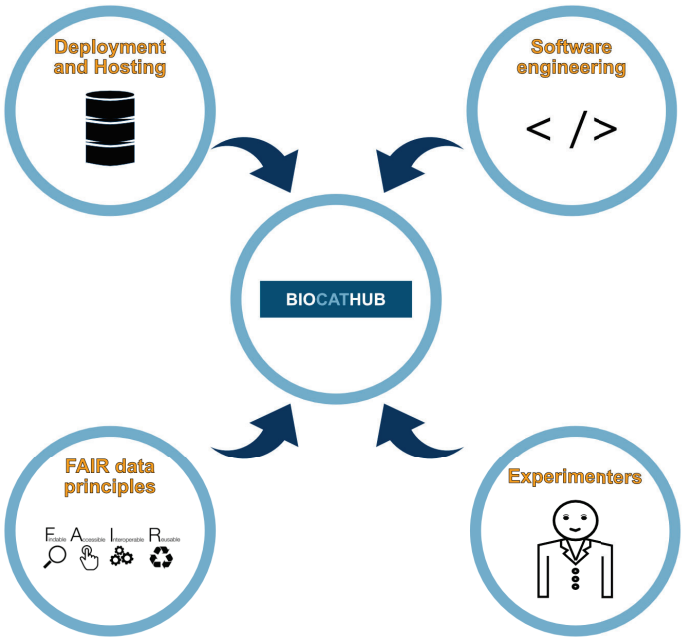

Figure 2.2: Identified requirement categories for the development of the BioCatHub-platform

2.1.4 Requirements regarding the FAIR data principles

Table 2.1: Requirements regarding the Findable guidelines of the FAIR data principles. The requirements were defined previously in [42].

Requirement	Specification
F-1 (Meta)data are assigned a globally unique and persistent identifier	Assigning datasets with a digital object identifier
F-2 Data are described with rich metadata	The BioCatHub data model describes the primary-and metadata to be collected for experiments in biocatalysis.
F-3 Metadata clearly and explicitly include the identifier of the data they describe	The primary data is stored in the same EnzymeML document as the metadata.
F-4 (Meta)data are registered or indexed in a searchable resource	Publishing EnzymeML documents in Zenodo

When research data is to be collected, stored and published, the FAIR data principles provide 15 guidelines as a state of the art foundation for conceptualising the research data management plans (chapter 1.3). Therefore, these guidelines were applied during the requirements engineering of the BioCatHub-platforms.

In tables 2.1, 2.2, 2.4, 2.3 the individual guidelines were used as requirements and specified for the BioCatHub-platform as much as possible. In detail, four technical resources were selected to realise the requirements of the FAIR data principles with existing technologies, formats and guidelines. At first, the Enzyme markup language (EnzymeML) (chapter 1.5.2) was specified as format to store and exchange data. It is a machine-readable file format designed to exchange data from enzyme-catalysed experiments [49]. Moreover, EnzymeML enables the combined storage of primary as well as metadata. Furthermore, Zenodo was selected as the second important resource. It is a publicly available repository recommended by funding agencies assigning published datasets with a DOI [36].

The HTTP protocol (chapter 1.6.4) was specified as the third resource and enables seamless upload and download of EnzymeML documents to Zenodo with the possibility to authenticate if necessary. Finally, the STRENDA recommendations were selected as the basis for defining a data model specific to biocatalytic reactions [26].

Table 2.2: Requirements regarding the Accessible guidelines of the FAIR data principles. The requirements were defined previously in [42].

Requirement	**Specification**
A-1 (Meta)data are retrievable by their identifier using a standardised communications protocol	EnzymeML documents in Zenodo can be retrieved by HTTP
A-2 The protocol is open, free, and universally implementable	HTTP is an open standard
A-3 The protocol allows for an authentication and authorisation procedure, where necessary	HTTP allows authentification procedures
A-4 Metadata are accessible, even when the data are no longer available	Primary as well as metadata is stored in the same EnzymeML document

Table 2.3: Requirements regarding the Interoperable guidelines of the FAIR data principles. The requirements were defined previously in [42].

Requirement	**Specification**
I-1 (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.	EnzymeML is used as machine readable file format
I-2 (Meta)data use vocabularies that follow FAIR principles	The already established systems biology ontology is used to annotate metadata with semantic identifiers
I-3 (Meta)data include qualified references to other (meta)data	Not implemented

Table 2.4: Requirements regarding the Reusable guidelines of the FAIR data principles. The requirements were defined previously in [42].

Requirement	**Specification**
R-1 (Meta)data are richly described with a plurality of accurate and relevant attributes	Metadata is defined by the BioCatHub data model based on the STRENDA recommendations
R-2 (Meta)data are released with a clear and accessible data usage license	Not implemented
R-3 (Meta)data are associated with detailed provenance	Not implemented
R-4 (Meta)data meet domain-relevant community standards	The STRENDA recommendations were used as basis for the data acquisition

2.1.5 Requirements regarding usability for experimenters in biocatalysis

To simplify the usage of the BioCatHub-platform for experimenters, four requirements needed to be fulfilled (Table 2.5). It should be operation system independent, available without installation, create reusable output and be as user-friendly as possible for experimenters in biocatalysis.

Table 2.5: Requirements regarding the usability for the usage of the BioCatHub-platform for experimenters.

Requirement	**Specification**
Operation system independent	Assigning datasets with a digital object identifier
Available without installation	The BioCatHub-platform is accessible by an internet connection, a browser and the respective URL (https://biocathub.net).
Generate reusable output	The primary data is stored in the same EnzymeML document as the metadata.
User interface should be intuitive to use for experimenters in biocatalysis	Wizard like structure, guiding experimenters through the individual steps of an experiment

For this purpose, the software needed to be available independently of the operating system as a web application over a web browser. This also enabled the usage of the software without the need to install it on the device. Further, the EnzymeML documents generated by the BioCatHub-platform needed to be reusable. By this, experimenters can document results or ideas, create EnzymeML documents and reuse them as templates for the documentation of further experiments or share their data with other scientists. The user interface of the BioCatHub-platform needed to be as intuitive as possible. For this purpose, the fundamental elements in every biocatalytic reaction were defined: the reaction ves-

sel, the biocatalysts, the reactants present in the experiment, the reaction condition and finally, the measured (or primary) data. To make the data collection process clearer, the user interface should not collect all the data at once but guide the experimenter through the elements of a biocatalytic reaction in a wizard-like structure.

2.1.6 Requirements regarding software development

Table 2.6: Requirements regarding the software development of the BioCatHub-platform

Requirement	**Specification**
Extensions and improvements need to be developable in projects between 6 weeks and 3 years	Microservice based architecture to encapsulate every functionality in one service
Uniform communication between microservices	Usage of the http protocol for communication between microservices.
Uniform development of Frontend and Backend microservices	Frontend development in Angular and Backend development in Python Flask

The BioCatHub-platform is intended to be applied in research. This demands that the development process needs to be possible in projects with durations of weeks up to a few years with no permanent positions available. Therefore, it is required that software aimed to improve or extend the BioCatHub-platform can be realised in individual projects, enabling software developers to only concentrate on the respective software to be developed without the need to understand the whole software infrastructure. For this purpose, the software architecture was defined microservice-based (chapter 1.12). As frontend, an Angular web application was selected for the interaction of experimenters (chapter 1.6.2.1). Functionalities, like database requests, upload of datasets to Zenodo or the generation of EnzymeML documents, individual microservices based on Python-flask-restx REST-APIs were defined for these tasks.

2.1.7 Requirements regarding the deployment and hosting

Table 2.7: Requirements regarding the deployment and hosting of the BioCatHub-platform

Requirement	**Specification**
Uniform deployment solution for every microservice	Deploy microservices as Docker-container
Make Microservices available	Publish docker container on DockerHub
Hosting of containers on a consitent platform, allowing the communication between container as well as creating availability by an internet connection	Hosting in Kubernetes cluster on a cloud computing platform

Once software is developed, it still needs to be deployed and hosted (chapter 1.6.6), the requirements and specifications for these tasks are listed in table 2.7. The first requirement was to provide the microservices and the web application for the deployment consistently to reduce complexity. Therefore, docker was chosen to containerise the individual microservices (chapter 1.6.6). Furthermore, a cloud computing platform was specified to host the individual containerised microservices and the web application, organising the containers using Kubernetes (chapter 1.6.6.1). By this, the dockerfiles for individual microservices can be generated as templates, reducing workload and complexity. Furthermore, the application of a cloud computing platform for hosting tasks outsources work, like maintaining and securing server hardware. Due to this architecture, more time and work are investable in software development and deployment.

2.2 Development of the BioCatHub-platform

As described in chapter 1.8 the overall abstract aim of the BioCatHub project can be separated into three different targets. First, to develop an infrastructure helping experimenters to collect, store and publish experimental data based on the FAIR data principles. Second, create customised and comprehensible data processing solutions once data is collected to visualise, analyse and share data with other scientists. Third, implement data from already existing public databases as well as online tools and integrate data directly from the laboratory. In the centre of these targets is the data acquisition platform: the BioCatHub-platform. In the concept, it serves as a connecting element for the other two targets.

The BioCatHub-platform was conceptualised as a web application (chapter 2.1.6). To realise this, a three-step workflow needed to be realised. At first, a data model had to be defined to serve as a basis for the communication between the experimenter and the BioCatHub-platform, as well as for software components with one another. Secondly, the frontend web application and backend microservices needed to be developed. Finally, the microservices had to be deployed and hosted to be available for experimenters in biocatalysis over an internet connection.

2.2.1 Development of the BioCatHub data model

A data model is a high-level description of data to visualise the structure of the data [87]. For the BioCatHub-platform, the data model was supposed to describe batch experiments in biocatalytic applications. For this purpose, the STRENDA recommendations [26] are available as a catalogue for metadata in enzyme-catalysed reactions. The STRENDA recommendations focus on the metadata documentation for kinetics and equilibrium experiments [26] which do not entirely match the metadata required for the description of experiments in biocatalysis (chapter 1.7). Therefore, metadata relevant for the description of biocatalytic experiments was selected from the STRENDA recommendations [26] and complemented with metadata specifically suited for biocatalytic procedures (chapter 7).

The resulting hierarchical data model is shown in Figure 2.3. It describes primary and metadata individually as key:value entries in elements. Every element is a collection of one or more data.

The root element defines metadata, describing the provenance of the experiment as the title (*title*), a customisable description of the experiment as free text (*description*), and if the data is publicly available, an identifier in the form of a DOI (*doi*). Details concerning the experimenter (*user*) are collected in another element. Here, the name (*firstName*, *lastName*), the current employer (*institute*), as well as contact information (*email*) are defined. The Vessel, Enzyme Condition and Measurement elements serve as the description of the experimental procedure. The BioCatHub-platform is focused on the description of batch experiments, it was prescribed that every experiment can be assigned to one vessel. Therefore, every Vessel element contains a description of the vessel (*type*), as well as the used volume (*volume, unit*). The enzyme element is built differently. Every experiment can include multiple enzymes, which again can catalyse multiple reactions. Enzymes are defined in elements with a name (*name*), the Enzyme Commission (*EC*) number (*ecnumber*). This identifier categorises enzymes based on their functions and describes them as a sequence of four numbers (e.g. EC 1.1.1.1 for an alcohol dehydrogenase) [88]. Further, the host organism (*organism*) specifies in which organism the enzyme occurs in nature. Moreover, it is defined if the enzyme is used as wild type or in a mutated form. In the case of mutations, the individual changes are described (*variant*) and complemented with the respective amino acid sequence (*sequence*). In case of a wild type enzyme, only the amino acid sequence is added (*sequence*) Moreover, the concentration (*concentration, unit*) and formulation (*formulation*) are described, together with the method used to measure the enzyme concentration (*method*). Reactions catalysed by the enzymes are defined as Reaction element, in which every reaction is specified by a name (*title*) and two element lists containing the Products (*products*) and Substrates (*substrates*) as elements as Reactants elements. Reactants are in this data model defined as the chemical substances which have a part in the described reaction. Every Substrate or Product element specifies the reactants

by their name (*name*), the SMILES code (*smiles*) and the molecular formula (*formula*). The SMILES code is an identifier describing molecules based on their chemical structure [89], and the molecular formula describes a molecule based on the number and kind of atoms. For example, the molecular formula of pyruvate is C3H3O3- and the SMILES code CC(=O)C(=O)[O-]. Every reactant's role is defined and can be expressed as substrate, product, cofactor or additive (*role*). Cofactors are molecules the enzyme needs for its catalytic functions, for example small metal ions. Any other added molecules are described as additives. In addition, information like the purity of the reactant (purity), the initial concentration (concentration, unit) as well a possible commercial origin (*supplier*) and publicly available images of the reactant (*imageURL*) are defined, too. Since every experiment can be defined with one vessel, one Conditions element is available in the Root element. It describes the reaction conditions as temperature (*temp*, *unit*) and the pH-value (*ph*). The solvent or buffer in which the reaction was performed is defined as Buffer element. Here, the name (*type*) and the buffer or solvent substance concentration are described (*concentration, unit*). To collect measurements during a biocatalytic reaction, the Measurement element is specified. This element can contain one or multiple measurements. The measured reactant is defined (*reagent*) as well as the units of the dependent (*y_unit*) and independent (*x_unit*) variables. Furthermore, if it is necessary to visualise the data in a graph, the names of the axes (*y_name, x_name*) as well as the style of the graph (*plotStyle*) are defined. The measured data are described as Replicates elements. In the replicates, one value of the independent variables (*x_value*) is collected with a variable number of measured values (*y_value*). In many cases, the provided data collection possibilities are insufficient to describe experimental procedures. Therefore, additional elements can be added (*others*) to describe customised data.

In this data model, a biocatalytic batch experiment is represented in primary and metadata. One important criterium in the development of this data model is the extendable. As this data model is designed for batch experiments, other experimental procedures like fed batch experiments are not in the scope of this data model. However, the modular structure of the data model makes it possible to add new elements and extend the data model

as needed.

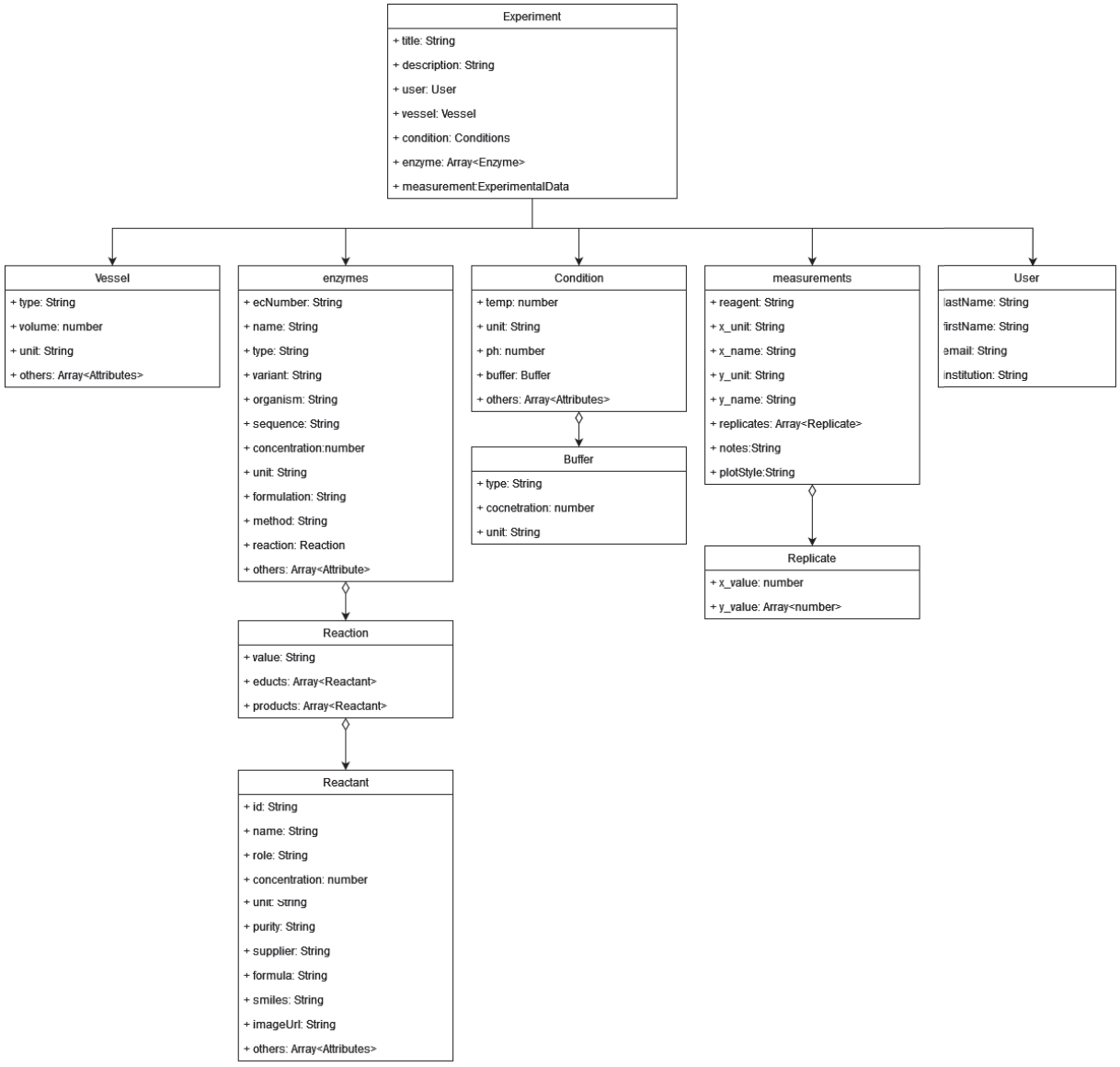

Figure 2.3: Data model for the BioCatHub-platform. Data is organised as a hierarchical structure in elements, with every element describing primary and metadata for a specific part of a batch experiment in biocatalysis. In the elements, data describing the individual experiment details are organised in key:value pairs. The model defines the individual data types for the values. String: text based input, number:numerical value, Array: represents a list in which further elements are contained.

2.2.2 Frontend development

With the data model defined, the development of the frontend as a web application could be realised. A frontend is the connecting element between software and the user or, in this case, the experimenter (chapter 1.6.2). The main goal of the frontend in the BioCatHub-platform is to collect the data specified in the data model described above. Moreover, the frontend should assist experimenters in this process as well as possible. The development of the frontend was realised in Angular (chapter 1.6.2.1), resulting in one web application named *BioCatHub-frontend* (*bch-frontend*) and the source code availabe on GitHub [90]. A detailed description of the frontend architecture is available in the master thesis of Patricia Warmelink [91].

2.2.2.1 Frontend design and usage

As described during the requirements engineering (chapter 2.1.5), the user interface was designed in a wizard-like structure to simplify the data collection. The individual steps in the wizard-like structure are named entries, and in the following chapter functions and the meaning of the entries are described. The entries collect primary and metadata based on the data model defined in (chapter 2.2.1).
The Landing Page is the first page available when navigating to the BioCatHub-platform (Figure 2.5). It is not an entry, since it does not collect primary or metadata. It is available as a navigation aid and allows the start of a new experiment or the usage of already existing experimental data by downloading previously published EnzymeML documents from Zenodo or uploading EnzymeML documents from the local file system.

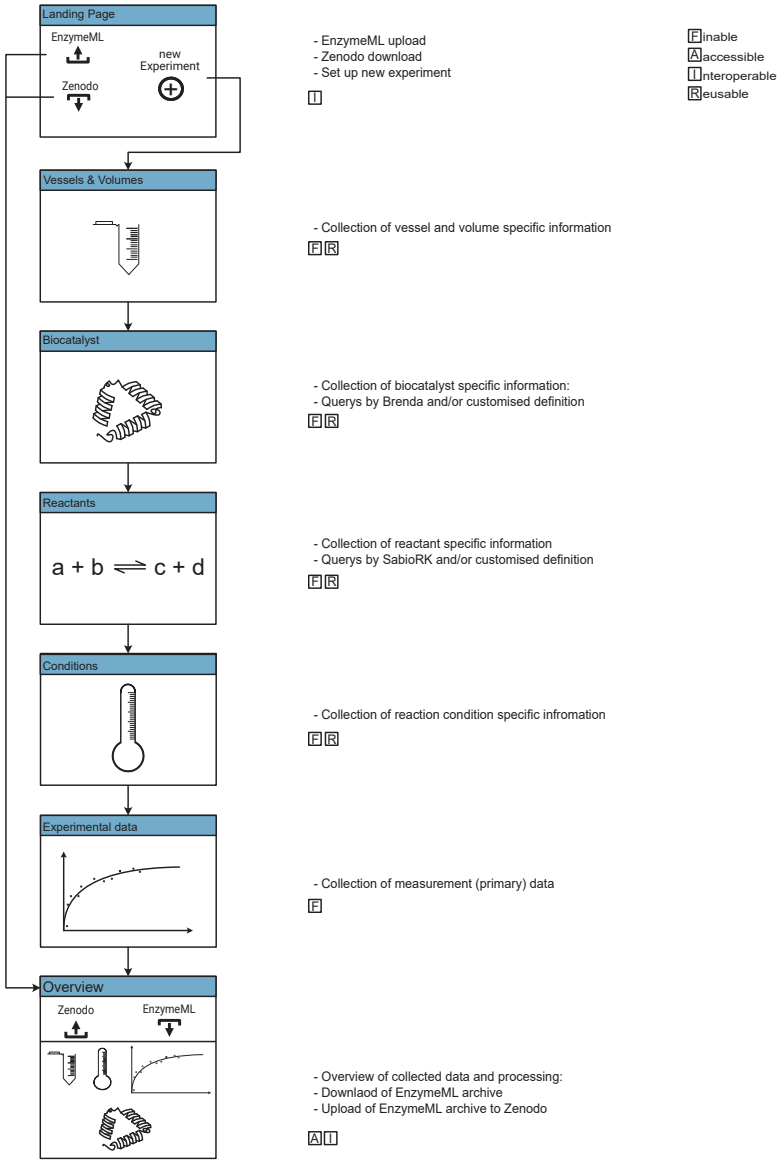

Figure 2.4: Entries available in the BioCatHub-platform.

The Overview entry is displayed when uploading or downloading an EnzymeML document from the local file system or Zenodo (Figure 2.11). This entry gives the experimenter an overview of the collected data, with the possibility to review or perform changes. Furthermore, it collects general information. Once changes or even new data are added, new EnzymeML documents can be downloaded to the file system or uploaded to Zenodo.

The Vessel and volumes entry is displayed if a new experiment is started and collects meta-

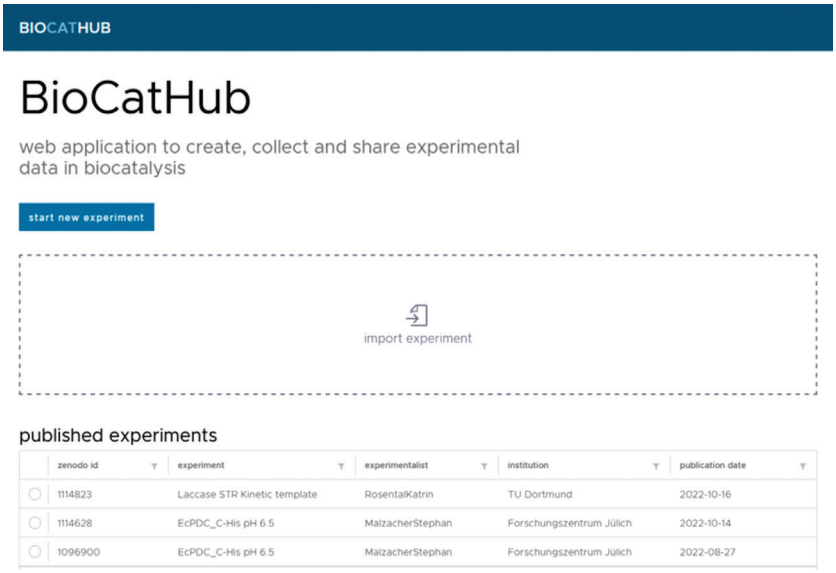

Figure 2.5: Screenshot displaying the Landing Page of the BioCatHub-platform.

data regarding the reaction vessel(Figure 2.6). This also starts the wizard-like structure described above, guiding the experimenter through the individual entries to collect data.

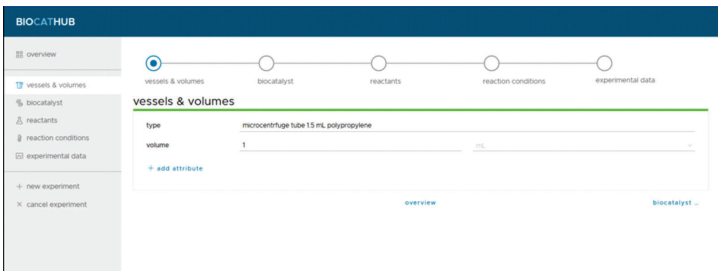

Figure 2.6: Screenshot displaying the Vessels and volumes entry of the BioCatHub-platform.

The following Biocatalyst entry describes metadata related to the applied enzymes. This entry has a speciality since it provides an input field connected to BRENDA [32] and returns a list of enzyme names and EC-numbers from which the suitable biocatalyst can be selected (Figure 2.7) (chapter 2.2.3.2).

Once the chosen biocatalyst is imported, the name and the EC-numbers are annotated automatically. Furthermore, a second request is performed automatically to SabioRK [30],

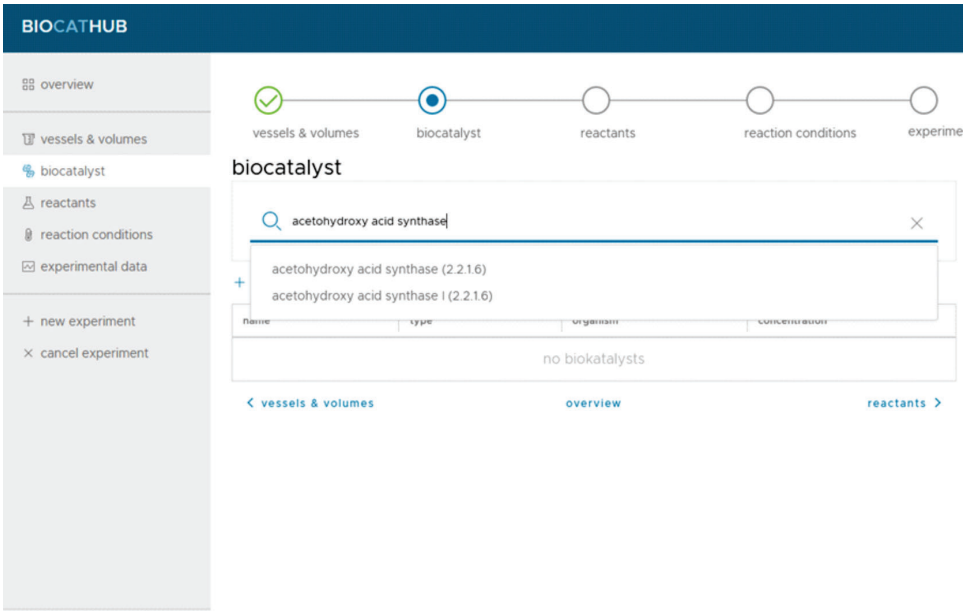

Figure 2.7: Screenshot displaying the Biocatalyst entry of the BioCatHub-platform.

querying available reactions based on the submitted EC-number. If a biocatalyst cannot be found in BRENDA, the description of the biocatalyst can be performed manually too. In this case, experimenters can manually add biocatalysts and describe the respective metadata.

As the experimenter proceeds with the reaction documentation, the Reactants entry is displayed. This entry serves to acquire metadata concerning the reactants involved in the reaction. If a BRENDA search was performed beforehand, a list of reactions from SabioRK is available here (Figure 2.8a) (chapter 2.2.3.2). Once a reaction is selected, the name, structure and SMILES code are added to the reactants entry (Figure 2.8b). Furthermore, the structure of the respective molecule is visualised automatically based on the submitted SMILES code (chapter 2.2.3.3). As with the Biocatalyst entry before, the database request to SabioRK aims to support the experimenter. If there are no reactants available from SabioRK, these can be annotated manually by the experimenter.

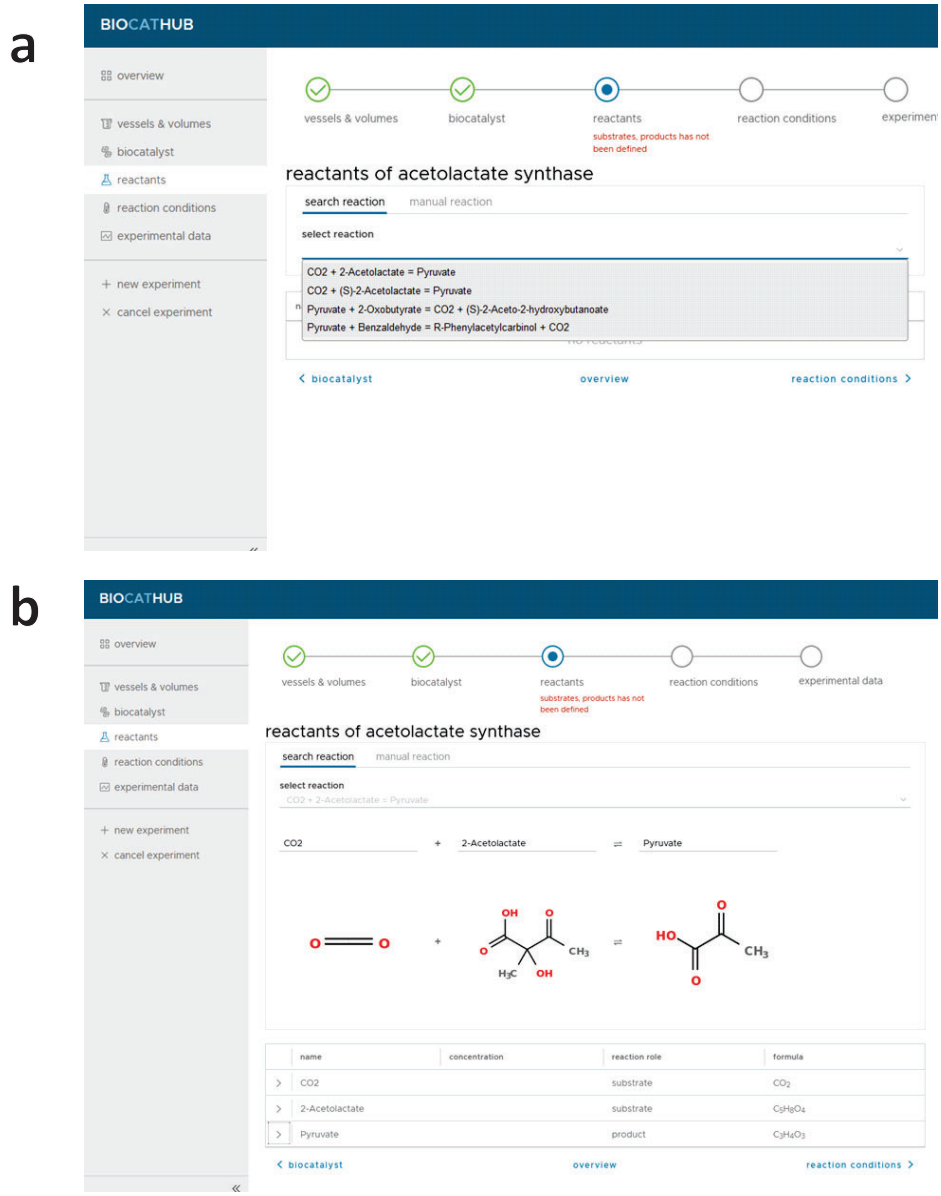

Figure 2.8: Screenshot displaying the reactants entry of the BioCatHub-platform. (a) Displays the reactions available from SabioRK, connected to a specific enzymatic reaction. (b) displays the reactants entry with collected metadata about the reactants.

The following Conditions entry (Figure 2.9) collects details about the reaction conditions during the experiment.

The Experimental data entry is used to add the primary data produced during the experiment (Figure 2.10). The current version of the BioCatHub-platform described in this thesis is designed to collect time dependent concentration or absorption measurements.

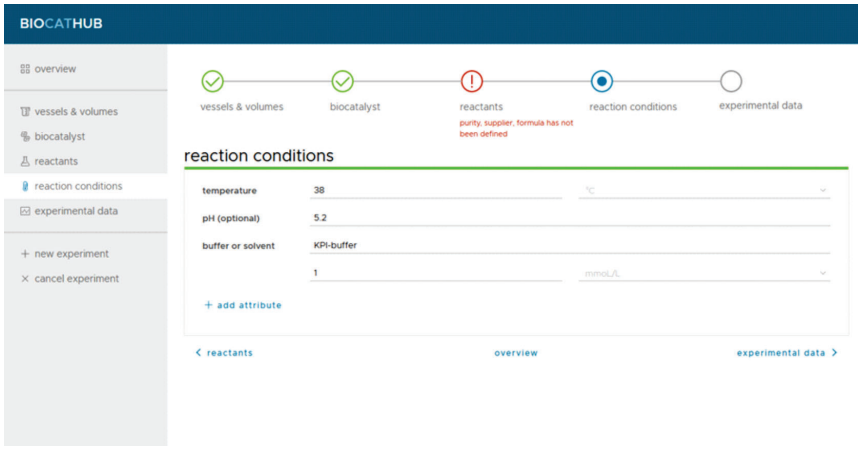

Figure 2.9: Screenshot displaying the Reaction conditions entry of the BioCatHub-platform.

It can be added by copy-pasting it from a spreadsheet into the experimental data entry or uploading a template with the primary data available on the Experimental data entry. Once added, the primary data is visualised as a line or dot plot.

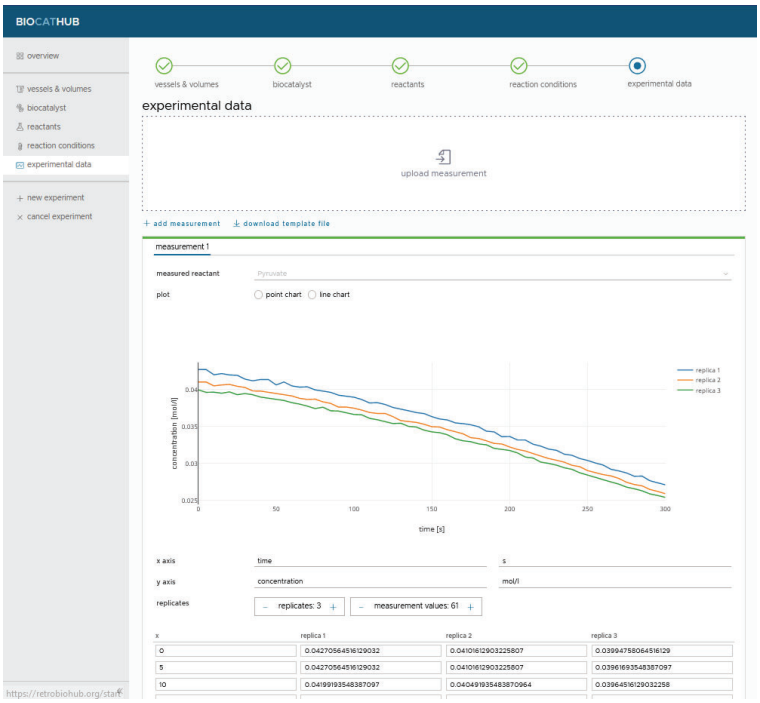

Figure 2.10: Screenshot displaying the Experiment entry of the BioCatHub-platform.

The final entry is the Overview (Figure 2.11). It provides an overview of the already collected meta – and primary data. Furthermore, metadata concerning the title, description

and contact details of the experimenter are added here. Once the experiment is described, it can be downloaded as an EnzymeML document into the local file system or uploaded to a connected repository in Zenodo.

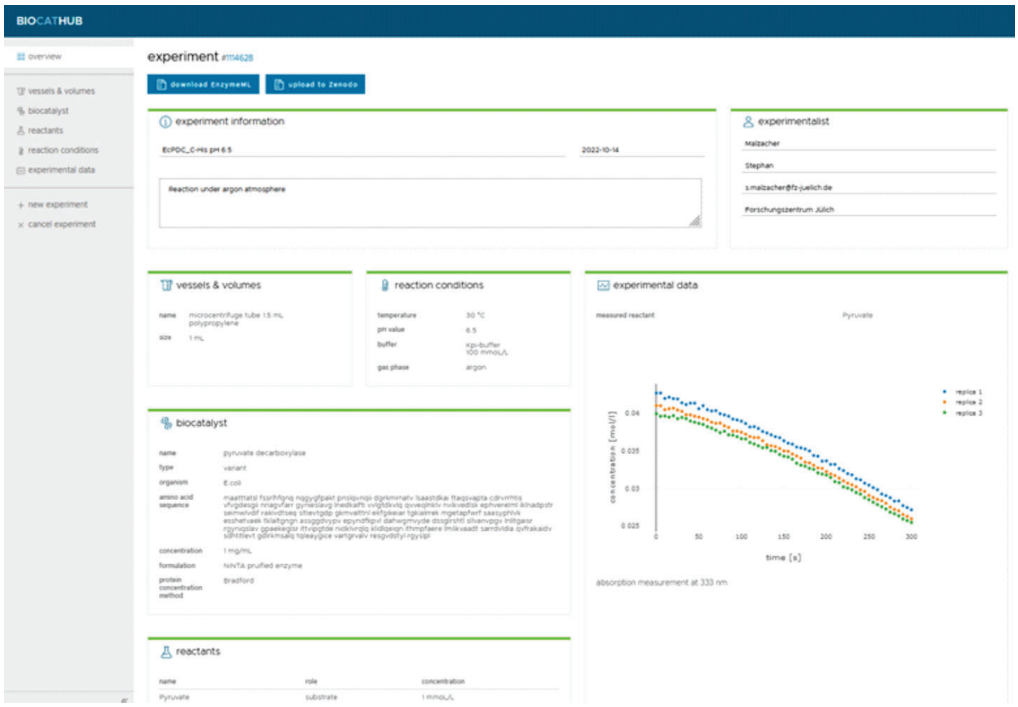

Figure 2.11: Screenshot displaying the Overview entry of the BioCatHub-platform.

With the *bch-frontend* web application developed, the requirements specified in (chapter 2.1.5) were realised. *Bch-frontend* collects the previously defined primary and metadata in the data model (chapter 2.3) in overall six entries, customised in particular for experimenters in biocatalysis and thereby serves as a gateway between researchers in biocatalysis and the BioCatHub-platform. Furthemore, it has functions implemented to reuse already collected data by uploading or downloading them from either the local file system or Zenodo in the Laning Page. Complementary, the Overview entry offers possibilities to generate EnzymeML documents by downloading them to the local file system or uploading them to Zenodo.

However, there are possible improvements to be made for future versions of the *bch-frontend*. For the description of reactants in the Reactants entry, it is possible to visualise the structure based on the submitted SMILES code, but it is not possible for experimenters

to draw the structure in an editor and retrieve the SMILES code from there. Since experimenters in biocatalysis often communicate over the drawn structure of molecules included in their experiments and not identifiers like the SMILES code, this would be beneficial. Furthermore, the experimental data can be uploaded in a predefined spreadsheet format or copy pasted from a spreadsheet file. It would be desirable for experimenters to have a spreadsheet interface available in *bch-frontend*, giving the possibility to easily update and reorganise the primary data. There are well established software packages for Angular and JavaScript available, realising the described functions. However, these packages are often a subject to charge and would exceed the provided budget. Free of charge packages are also available but require more time in the implementation and testing, which would have exceeded the time available for the development of bch-frontend. In the here presented version of the *bch-frontend*, the focus was on the realisation of the necessary requirements (chapter 2.1.5). Nevertheless, with the development and realisation of the *bch-frontend*, new developments can focus on the improved usability and implement the above-mentioned improvements.
With the *bch-frontend*, the web application for the BioCatHub-platform is realised. In the description, multiple functions are described, like the database query to BRENDA and SabioRK, the reading and writing of EnzymeML documents, the visualisation of reactants structures based on the SMILES code and the connection to Zenodo. These functions are triggered by the *bch-frontend* but not executed by it. Therefore, microservices in form of web services were developed (chapter 1.6.3). In the following chapters, the development and functions of the individual microservices are described.

2.2.3 Backend development

In web applications, the frontend provides all functionalities allowing the experimenter to interact with the software. The backend is the counterpart and fulfils tasks triggered by the frontend but not visible to experimenters. Furthermore, unlike the frontend, which is specialised in the best possible usability, the backend must focus on the fast and effective execution of tasks (chapter 1.6.3). In the following chapters, the individual microservices

required for the BioCatHub-platform are described. The development was realised in individual publicly available GitHub repositories providing the source code.

2.2.3.1 Reading, writing of EnzymeML documents and the connection to Zenodo

Once data is collected in the BioCatHub-frontend web application (*bch-frontend*), a format is needed to store the data in a machine-readable way as specified in (chapter 2.1.4). Therefore, EnzymeML was selected to serve as a data storage and exchange format. One microservice named BioCatHub-enzymeml (*bch-enzymeml*) [92] was developed with the tasks to perform the writing and reading of EnzymeML documents as well as the upload and download of EnzymeML documents from and to Zenodo.
Bch-enzymeml has two paths for handling the writing and reading of EnzymeML: *enzml/create_enzymeml/* and *enzml/read_enzymeml/*. The path *enzml/create_enzymeml/* is accessible by an HTTP request and performs the creation of an EnzymeML document based on the submitted BioCatHub-object. Two different objects are processed in this workflow. The BioCatHub-object and the EnzymeML-object are both JSON-objects (chapter 1.5) derived from the BioCatHub-model (chapter 2.2.1) and the EnzymeML-model, respectively. The EnzymeML data model is available at [51, 49]. The generation of EnzymeML documents is performed using the PyEnzyme RESTful API [51].

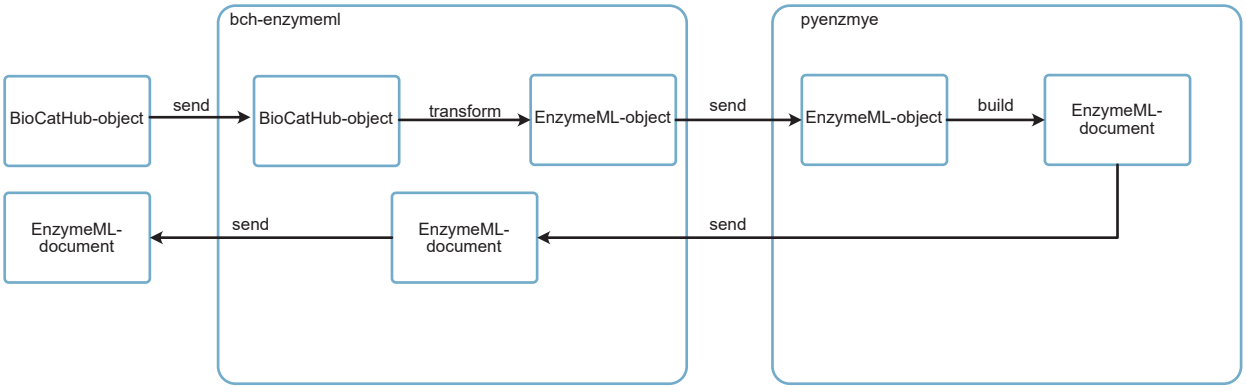

Figure 2.12: Workflow of the EnzymeML creation process in the bch-enzymeml microservice

The EnzymeML writing process is divided into three processes (Figure 2.12). Firstly, an

EnzymeML-object is generated using data from the submitted BioCatHub-object. For this, the EnzymeML-object was defined using the pydantic BaseModel class [93]. The resulting classes are named models in the pydantic documentation and referred to as pydantic-models in this thesis. The pydantic-models are used together with data from the BioCatHub-object to generate the EnzymeML-object (Figure 2.13).

```python
from pydantic import BaseModel
from typing import List, Optional, Dict

# Definition of the pydantic-model:

class CreatorDetail(BaseModel):
    given_name: str
    family_name: str
    mail: str
    id:str
    class Config:
            orm_mode = True


# Definition of a parsing class

class Creatorcls:
    def __init__(self, given_name: str,
                 family_name: str,
                 mail: str,
                 id: str):

        self.given_name = given_name
        self.family_name = family_name
        self.mail = mail
        self.id = id
```

Figure 2.13: Example code of the generation of a Pydantic class

The second process is creating the EnzymeML document, realised using the PyEnzyme RESTful-API. Therefore, an HTTP request is made using the requests python package [94] to the PyEnzyme RESTful API with the EnzymeML object attached. In response, an EnzymeML document is submitted. In the last process step, the BioCatHub-object is added to the EnzymeML document using the licombine package [95]. As this is finished, the workflow of the *enzml/create_enzymeml/* path ends as the EnzymeML document is sent back as a response. The EnzymeML document can then either be stored locally on the file system or as described below, uploaded to Zenodo. Network connectivity problems or errors in the transformation process from the BioCatHub-object to the EnzymeML-object can occur. For this purpose, a backdoor was implemented in the writing process of EnzymeML

documents. If an error occurs in the EnzymeML writing process, the BioCatHub-object is stored in an empty omex archive using the libcombine package and sent back as a response. To mark that these omex archives do not contain files specific to an EnzymeML document, the files are labelled with a (no EnzymeML) tag in the filename. The created omex archives can still be uploaded to the BioCatHub-platform, to generate EnzymeML documents, once the error or connectivity problem is solved. Furthermore, the data processing with the BioCatHub-data-processing repository, which is described in detail in chapter 2.3.1, is also possible with these omex archives.
The path *enzml/read_enzymeml/* is available by an HTTP request, extracts data from the attached EnzymeML document and writes it to the BioCatHub-object. This workflow contains two possible procedures. As the EnzymeML document is submitted, the libcomine package is used to examine if the BioCatHub-object is available in the document. If this is the case, the BioCatHub-object will be extracted, and the request finishes by sending the BioCatHub-object as a response. If the BioCatHub-object is unavailable, the EnzymeML document is submitted in an HTTP request to the PyEnzyme RESTful API, returning the EnzymeML-object. This is realised analogue to the process in the *enzml/create_enzymeml/* path, transforming from the EnzymeML-object to the BioCatHub-object. The request is finished by sending the BioCatHub-object as a response.

The path *enzml/zenodo/* queries a connected Zenodo repository for all previously stored EnzymeML documents. To establish a connection with a Zenodo repository, the repository owner has to create a specific authentication password, named *token* in the respective Zenodo repository. By implementing this token in an HTTP request to the Zenodo RESTful API [96], the API can retrieve a JSON-object containing metadata about all available EnzymeML documents with the id, title, creation date, experimenter name, affiliation and the link for every entry. The requests to the Zenodo RESTful API are performed similarly to the requests to the PyEnzyme RESTful API using the requests package. The Zenodo ID of an entry is submitted to the *enzml/zenodo/getexperiment* path and causes the download of the respective EnzymeML document. Therefore, an HTTP re-

quest is sent to the Zenodo RESTful API containing the ID of the EnzymeML document and the token for the individual repository. From the downloaded EnzymeML document, the BioCatHub-object is extracted as described above and sent as a response. The path *enzml/zenodo/publishexperiment* uploads an EnzymeML document to Zenodo based on a submitted BioCatHub-object. For this purpose, an EnzymeML document is created as in the *enzml/create_enzymeml/* path. Once the EnzymeML document is available, a new Zendo entry is created and annotated with the title, description and the experimenter's name, extracted from the BioCatHub-object. These descriptions are fields provided by Zenodo to improve the visibility of the uploaded data. This is performed by an HTTP request to the Zendo RESTful API. Furthermore, the upload type of Zenodo is annotated with a *dataset* and keywords *enzymes*, and *biocatalysis* are added. For this, an HTTP request is performed containing the data for the annotation of the entry as a JSON object. In response, the newly created entry's ID is sent. This ID is used for a follow-up HTTP request to the Zenodo RESTful-API, causing the upload of the EnzymeML document.

Visibility and security

The upload and publishing of experimental data to Zenodo is connected with large responsibilities. In particular in cooperation with industrial partners, it is essential to ensure that restricted data is not publicly available. In the BioCatHub-platform, the upload to Zenodo is realised due to usability reasons in a one-click action (chapter 2.2.2.1). However, due to this simplicity it can happen that data which is not meant to be publicly accessible is published to Zenodo by accident. Therefore, the connection to Zenodo was developed with two safety mechanisms. First, the current connection to Zenodo is realised into one account which is managed by the BioCatHub project developers in the Group of Prof. Dörte Rother at the Forschungszentrum Jülich. Meaning that every EnzymeML document is uploaded to one Zenodo account. Therefore, if any irregularities occur with the Zenodo, developers from the BioCatHub project have the possibility to intervene and help. For example, if an EnzymeML document not meant for publication is uploaded by accident, the experimenters can get in contact with the BioCatHub project developers and

the EnzymeML documents are removed.
The second safety mechanism focuses on the publication process of EnzymeML documents. Every EnzymeML document uploaded to Zenodo, is visible on the BioCatHub-platform and can be reviewed and used from there. Nevertheless, the publication of experimental data still has to be confirmed by the Zenodo account. For this purpose, the experimenters have to contact the BioCatHub project developers, which will then perform the publication. If an experimenter wishes not to publish data on Zenodo, the EnzymeML documents can still be downloaded and stored on the local file system.
These security measures create additional effort for experimenters and developers. Nevertheless, during the development of the *bch-enzymeml* microservice, it was decided that data protection must be of the highest priority. Of course, in the near future these mechanisms are desirable to be replaced enabling experimenters to connect their own Zenodo repositories with the BioCatHub-platform and manage their uploaded EnzymeML documents by themselves. Therefore, the *bch-enzymeml* microservice was designed in a way that these functionalities can be easily added.

2.2.3.2 Querying databases with bch-database-query microservice

Two databases were implemented for retrieving data concerning information about enzymes and reactions. The first database is BRENDA and the second is SabioRK. The microservice bch-database-query handles the communication with the databases. A detailed description of the communication with the databases is available in the master thesis of Patricia Warmelink [91].

2.2.3.3 Visualisation of SMILES codes as chemical structures

During the development of the bch-frontend web application, it was observed that visualising the molecular structures is helpful when describing the reagents in a biocatalytic reaction. The microservice *BioCatHub-smiles-to-structures* (*bch-smiles-to-structures*) [97] performs this task (Figure 2.14). It has one path */smiles* to which a SMILES code is provided in an HTTP request. The SMILES code is passed to the openbabel tool [98], which converts

the SMILES code to scalable vector graphic (svg) of the respective chemical structure. Following the svg is again converted to an image file and sent in response using the cairosvg [99].

```
def convert_smiles_to_molecule_image(smiles):

    # convert smiles to an svg


    os.system('obabel -:"{}" -O molecule.svg'.format(smiles))

    # transform the smiles to a PNG

    svg = open("molecule.svg", "rb")
    svg2png(file_obj=svg, write_to='molecule.png')
    return send_file('molecule.png', mimetype='image/png')
```

Figure 2.14: Example code of the bch-smiles-to-structures microservice

2.2.4 Deployment and hosting

With the software development of the BioCatHub-platform, three python-flask-restx microservices and one Angular web applications were developed, and the source code is available. However, these were not running in a production environment on a server (chapter 1.6.6) connected to the internet and therefore unavailable for experimenters. To address this challenge, two further developments needed to be made (chapter 2.2.4). First, the software had to be available in a container-based virtual environment in the production environment. Second, a possibility was needed to organise the available containers and connect them available over an internet connection. For this purpose, Kubernetes is used.

2.2.4.1 Docker container as deployment technology

For the containerisation of the microservices, docker was used (chapter 1.6.6). Therefore, one docker image was created and uploaded to the docker hub repository for every microservice and web application. Docker hub is a public repository for containerised applications [75]. This step was needed to make the docker image available for Kubernetes (see below). The *bch-frontend* microservice image was created in a two-step process, as displayed in Figure 2.15. In the first process, the node:12.7-alpine image [100] was used as the base image to build *bch-frontend*. Subsequently, the bch-frontend was built (chapter 1.6.2.1) and provided the folder dist (distribution) containing the files required for the browser to process the frontend.

```
# define base image
FROM node:12.7-alpine AS build

# define working directory

WORKDIR usr/src/app
COPY . .

# install required third party software

RUN npm install
RUN npm run build

# configure HTTP-server

FROM nginx:1.17.1-alpine
COPY nginx.conf /etc/nginx/nginx.conf
COPY --from=build /usr/src/app/dist/biocathub /usr/share/nginx/html
```

Figure 2.15: Example code displaying a dockerfile.

For the second process, 'nginx:1.17.1-alpine' [101] was used as the base image. This image provides the functionalities of an HTTP server (chapter 1.6.2). The configuration is done by copying the configuration file for the server to the respective folder in the 'nginx:1.17.1-alpine' image as the image is created. With the configuration set, the previously created

files in the first process need to be made available to the HTTP server by copying them to the 'usr/share/nginx/html' folder.
The backend microservices bch-enzymeml and bch-database-query are containerised by using the same dockerfile. The base image is the 'python:3.7-slim' [102] . First, a working directory is defined as '/app', and environment variables are set for the image. The 'FLASK_RUN_HOST' variable declares on which IP address the app is available. With the following steps, the configuration variables and required third-party software are installed to run the web service.
To deploy the bch-smiles-to-structures microservice, a separate dockerfile was required, although it is a flask-restx application. As in this image, openbabel is used, it needs to be installed on a 'Linux-ubuntu' system. Therefore the 'ubuntu_latest' image is used as the base image [74]. Subsequently, commands for installing the supporting software are added to the dockerfile. The rest of the description of the dockerfile follows the same logic as the dockerfile for bch-enzymeml and bch-database-query. As the individual dockerfiles were created, the dockerfiles were built and published on docker hub. After this workflow, the docker images are available for a hosting platform and can be used in a production environment like Kubernetes, described in the following chapter.

2.2.4.2 Kubernetes as container orchestrator

With the docker images available on docker hub, a server platform could be established hosting the microservices, orchestrating the individual containers, and making them available online (chapter 1.6.6.1). For that purpose, Kubernetes was used. Servers with a preconfigured Kubernetes cluster were rented from the cloud computing platform Linode. The application of a cloud computing platform was chosen to reduce the workload and improve security. If the deployment would have been realised in a self-hosted system, the server configuration including the Kubernetes instalment and security settings would have been carried out by the developers themselves. This work would have taken up a lot of time and human resources. Cloud computing platforms, therefore, provide already preconfigured Kubernetes instalments, which can be generated in less than five minutes.

Each server had the Linode-specific parameters of 2 CPU cores, 4 GB of RAM and 80 GB of file storage.

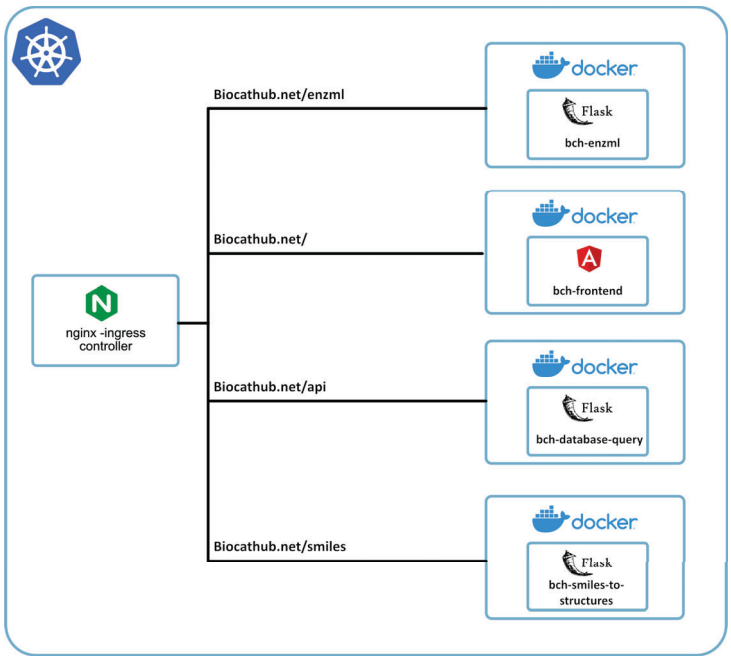

Figure 2.16: Schematic overview of the Kubernetes cluster.

Three helm charts were installed to realise the hosting of the BioCatHub-platform. First, an ingress controller was provided. This was done by installing the 'Nginx-ingress controller' [103]. The second helm chart to be installed is named 'biocathub' and was developed as part of this dissertation [104]. The third helm chart installed is the 'cert-manager', used to retrieve TLS certificates (chapter 1.14) [77].

The *bch-ingress* template describes how the Nginx-ingress-controller forwards requests incoming to the Kubernetes cluster (Figure 2.16) (table 2.8). The domain name was purchased by the domain registry joker.com. The domain name and the IP address are registered in Linode. If a request is made to the domain name, the registered IP address is used to forward the request to the Kubernetes cluster.

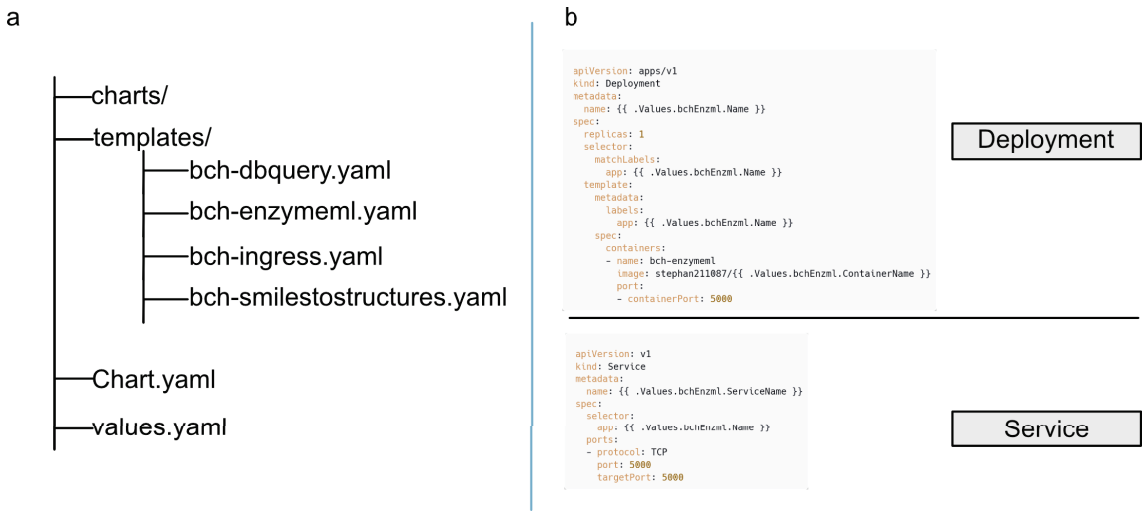

Figure 2.17: Schematic overview of the Kubernetes cluster.

Table 2.8: Paths available in the biocathub kubernetes cluster.

Path	**Microservice**
Biocathub.net/	*Bch-frontend*
Biocathub.net/smiles	*Bch-smiles-to-structures*
Biocathub.net/enzml	*Bch-enzymeml*
Biocathub.net/api	*bch-database-query*

Every deployment template consists of two parts (Figure 2.17): the deployment description and the service. The service is addressed by the ingress and forwards the request to the respective deployment. The last helm chart required was the cert-manager chart. The cert-manager provides the possibility to retrieve TLS certificates from the certificate authority *Let's encrypt* [72] and realises an encrypted communication.

With the source code of the bch-frontend web application and the backend microservies deployed as docker-containers and hosted on Linode in a Kubernetes cluster, the BioCatHub-platform is available over an internet connection and can serve as research data management platform for biocatalysis. Therewith, the specified requirements (chapter 2.1.3) were

realised successfully. The remaining question is how can experimenters implement the BioCatHub-platform in their daily work life. Therefore, three scenarios are described in the following sections, showing how the BioCatHub-platform can be applied.

2.3 Scenarios

Experimental procedures in biocatalysis as well as reaction performance values can vary strongly from one another (chapter 1.1.1). One possibility would be to implement data processing tools in the BioCatHub-platform. Although, implementing a multitude of data evaluation options would complicate the software development process and exceed the work capacities of individual developers. One of the goals of the BioCatHub project is to provide frequently used data processing possibilities for experimenters in biocatalysis (chapter 1.8). Therefore, data processing and publication workflows, inspired by [105], were established as described in the following scenarios. For this purpose, Jupyter-Notebooks [106] was used as a tool to develop customisable, comprehensible, shareable and extendable data processing workflows [107].

2.3.1 Scenario 1: Collection of data from enzyme function regulation studies using alternating magnetic fields

One major goal of scientific research is the publication of data produced in the laboratory in form of articles [108]. In many cases, this is true only for positive data (chapter 1.2). However, negative data often contain important information. For example, with the availability of negative data experimenters can save time, money and resources by not repeating non-working experimental procedures. Although, this is only possible if the negative data is publicly available to other researchers. Therefore, in this scenario a workflow for the collection, processing and publication of negative data is presented.

2.3.1.1 Outline and target in the scenario

Cross-reactivity in multistep enzymatic cascades or reactions is a challenging problem. Cross-reactivity means that enzymes in a cascade can accept the products, substrates or intermediates of other enzyme-catalysed reactions performed in the same cascade [109]. Therefore, it is of value if the enzyme activity in enzyme-catalysed multistep reactions can be separated in time or in space. If a one-pot experiment is chosen, meaning that every reaction takes place in only one reaction vessel, one after another, a regulation of each enzymatic step is advantageous. In the example of the 'EU-FETOPEN' project called Hotzymes [110], the biocatalysts were immobilised on magnetic silica nanoparticles. To regulate the enzyme activity, the reaction was performed at 4 °C and the immobilised enzymes were exposed to an alternating magnetic field (AMF), causing a molecular movement and thereby a heat gradient around the magnetic nanoparticles and increasing the enzyme activity. The target of this scenario is to document and publish the performed experiments as user-friendly as possible and, in addition, fulfil the requirements of an EU consortium. Moreover, in this scenario in-detail enzyme immobilisation procedures and the usage of the AMF needed to be documented. Therefore, it was aimed to document these metadata in the BioCatHub-platform, even if they were not implemented in the data model (chapter 2.3). As a final result, one link should be provided, presenting the collected data as self-explanatory as possible, enabling other researchers, especially in the 'EU-FETOPEN' project, to reuse the data. Therefore, the collected data should be processed with a short description to help experimenters to understand and reuse the data.

2.3.1.2 Data collection and processing

The data documentation, processing and publication process was divided into three steps. The first step concentrated on the data collection, and the second on the visualisation and processing of the collected primary data. The last step focussed on the publication of the collected data.

2.3.1.3 Collection of data from the Hotzymes project

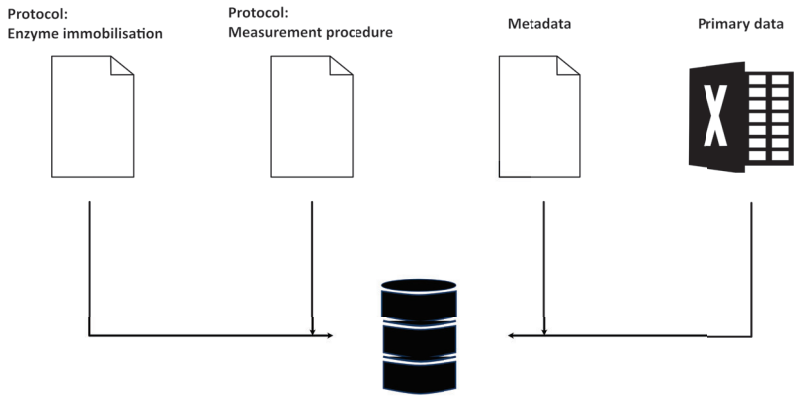

Figure 2.18: Protocols as well as primary and metadata to be collected in the Hotzymes project.

The documentation started by defining the metadata concerning the reaction and the enzyme. For this, RetroBioCat was used as described in chapter 3. RetroBioCat was in this case only used for documentation purposes, the outline and planning of the experiments were already defined beforehand. Following, the details about the biocatalyst, reagents, reaction conditions and reaction vessel were added. Subsequently, in addition to the metadata already defined by the BioCatHub-data model, protocols for the immobilisation, measurement procedure and usage of the AMF had to be added (Figure 2.19). This was done by uploading and publishing the protocols to Zenodo and making them available by a DOI. Therefore, new attributes were defined in the BioCatHub-platform (Figure 2.19b), containing the DOIs of the protocols. The resulting EnzymeML document was used as a template for collecting primary data.

Measurements in form of time dependent concentration courses needed to be collected of in total four different setups. The first measurement is an enzyme control experiment with the soluble enzyme at 25 °C (control *Bm*ta) without enzyme immobilisation. The second experimental measurement consisted of positive controls and the samples with the immobilised *Bm*ta in two different approaches. For the positive control, the reactions were performed at 25 °C without the influence of the AMF. The samples were measured at 4 °C with and without the influence of the AMF. The immobilisation application varied in the formulation in which the enzymes were applied. In the first approach (H2a), the enzymes

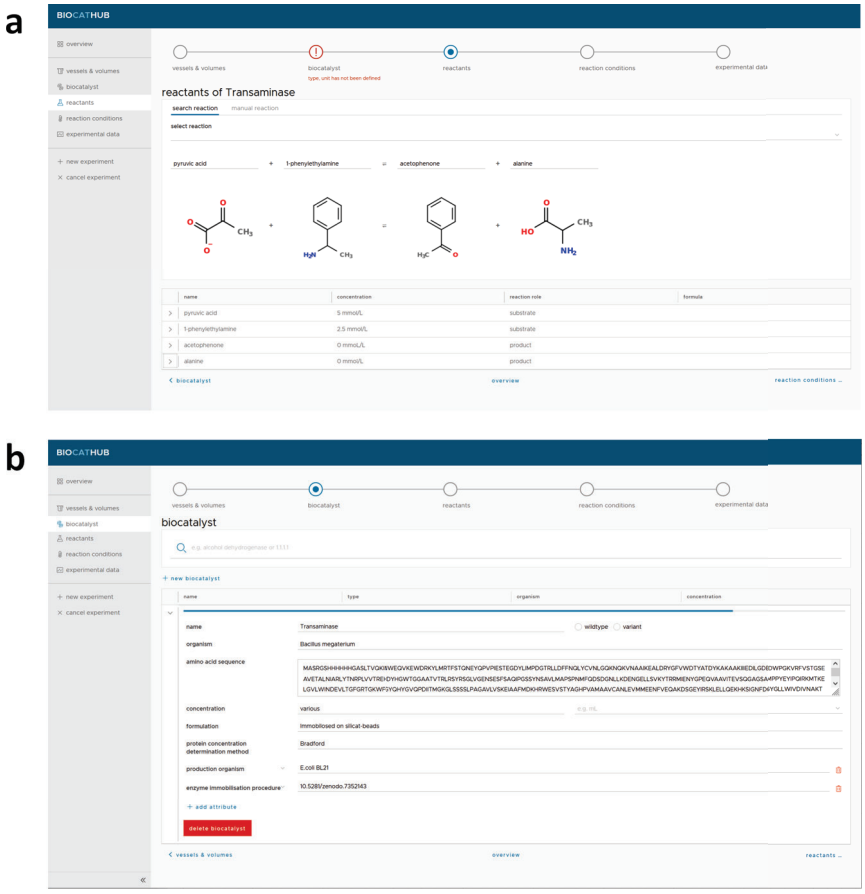

Figure 2.19

were solubilised in water before the immobilisation was performed. In the second (H2b), sodium phosphate buffer was used as solvent. More details about the experimental procedures can be found here: [111].

2.3.1.4 Visualisation and processing of data from the Hotzymes project

Data stored in EnzymeML documents has the advantage that metadata and primary data is collected in a standardised and machine-readable way, enabling experimenters and programmers to create reusable procedures for many data processing possibilities (chapter 1.5). In the Hotzymes scenario, data should be compared visually and numerically to support the following description during the publication step.

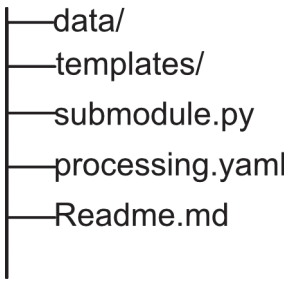

Figure 2.20: File structure of the BioCatHub-data-processing-repository (bch-data-processing.

As a template for the development, the *BioCatHub-data-processing* repository (*bch-data-processing*) was developed and is available on the GitHub organisation page of BioCatHub [112]. Its structure comprises two folders and three files in the root directory (Figure 2.20). The data folder contains the EnzymeML documents created with the BioCatHub-platform. In the templates folder, the experimental descriptions are stored as template EnzymeML documents (chapter 2.3.1.3). The submodule.py contains Python code, extracting primary- and metadata data from the EnzymeML documents, stored in the data folder. Processing.ipynb is a Jupyter-Notebook, in which the data visualisation and processing workflows are realised. In the last file, readme.md, the written text-based documentation can be provided by the experimenter following the markdown syntax and is automatically visualised once the repository is uploaded to GitHub or GitLab (Figure 2.25).
The just described process is the standard procedure when using a *bch-data-processing* repository. Nevertheless, since the respective source code for this process is implemented in the *bch-data-processing* repository, customised workflows can be developed in a copy of this repository if demanded by the project.
For this scenario, the *bch-data-processing* repository was downloaded. Subsequently, the with the BioCatHub-platform created EnzymeML documents were stored in the data folder. The EnzymeML document, which served as a template, was copied to the templates folder. The data processing started by extracting primary data from the EnzymeML documents in the data folder. Therefore, a specialised module (submodule.py) is available, providing the collected primary data as pandas DataFrames [113]. An in-detail description of DataFrames is available in the provided citation.

```python
import os
from submodule import importer
import pandas as pd

# list the EnzymeML documents in the data folder

print(os.listdir("./data"))

# import the primary data from the EnzymeML documents

control_H2a_25 = importer('data/2022-10-3Hotzymes-BmTA activity assay H2a control at
25degree(noEnzymeML).omex').build_data_frame()
control_H2b_25 = importer('data/2022-10-3Hotzymes-BmTA activity assay H2b control at
25degree(noEnzymeML).omex').build_data_frame()
control_BmTA = importer('data/2022-10-3Hotzymes-BmTA activity assay with purified enzyme control at
25degree(noEnzymeML).omex').build_data_frame()
H2a = importer('data/2022-10-3Hotzymes-BmTA activity assay H2a(noEnzymeML).omex').build_data_frame()
H2b = importer('data/2022-10-3Hotzymes-BmTA activity assay H2b(noEnzymeML).omex').build_data_frame()
```

Figure 2.21: Example code for the extraction of primary data from the EnzymeML documents in the data folder.

With the primary data available as pandas DataFrames, the next step was visualising the time-dependent concentration courses of the collected experimental data. This was implemented using the matplotlib and Seaborn Python packages [59, 114] (Figure 2.22). The resulting graphs are displayed in Figure 2.23. Every chart shows the *Bm*ta control, the respective positive control, and the sample measurements with and without exposure to the AMF. The positive controls as well as the *Bm*ta control show a comparable time-concentration course. These can also be observed, independently of the influence of the AMF, for the samples of H2a and H2b. Based on the hypothesis of the Hotzymes project, the time-dependent concentration courses should be steeper when the AMF is applied. These results suggest that there is no influence on the enzyme activity due to exposure to an AMF.

Besides the visual data comparison, the enzyme activities were calculated for a numerical comparison, based on eq 2.1. Data processing and visualisation are usually performed by experimenters in the Hotzymes project using spreadsheet software like Microsoft Excel and not Jupyter-Notebooks. For this purpose, the pandas DataFrame describing the previously collected primary data is exported into a csv file using the pandas package [113]. Csv files can be seamlessly imported by spreadsheet programs like Microsoft Excel, mak-

```
# matplotlib configurations

fig, (ax1, ax2) = plt.subplots(1,2)
plt.rcParams['figure.dpi'] = 95

# definition of the first plot wwith the data of H2a

ax1.plot(control_H2a_25["x_values"],control_H2a_25["y_values_0"], label="control 25°C", linestyle="none",
marker="s", color="black")
ax1.plot(H2a["x_values"],H2a["y_values_0"], label="AFM_off", linestyle="none", marker="o", color="darkgrey")
ax1.plot(H2a["x_values"],H2a["y_values_1"], label="AFM on", linestyle="none", marker="x", color="darkgrey")
ax1.plot(control_BmTA["x_values"],control_BmTA["y_values_0"], label="soluble enzyme", linestyle="none", marker="d",
color="lightgrey")

# annotate the plot with title and axis description

ax1.set_title("H2a")
ax1.set_xlabel("time in minutes")
plt.legend()
ax1.set_ylabel("concentration in mmol/L")



# definition of the first plot wwith the data of H2b

ax2.plot(control_H2b_25["x_values"],control_H2b_25["y_values_0"], label="control 25°C", linestyle="none",
marker="s", color="black")
ax2.plot(H2b["x_values"],H2b["y_values_0"], label="AFM_off",linestyle="none", marker="o", color="darkgrey")
ax2.plot(H2b["x_values"],H2b["y_values_1"], label="AFM_on",linestyle="none", marker="x", color="darkgrey")
ax2.plot(control_BmTA["x_values"],control_BmTA["y_values_0"], label="soluble enzyme", linestyle="none", marker="d",
color="lightgrey")

# annotate the plot with title and axis description

ax2.set_title("H2b")
ax2.set_xlabel("time in minutes")
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=0.1)
plt.legend()

# Store the plot as scaleabe vector graphics

plt.savefig("graph.svg", format="svg")
```

Figure 2.22: Code example for data visualisation in the Hotzymes project.

ing data processing possible in these programs.

$$specific\,enzyme\,activity in \frac{U}{mg} = \frac{\Delta c_{product}}{\Delta t} * \frac{1}{c_{enzyme}} \tag{2.1}$$

$c_{product}$: $concentration\ in\ mmol/L$

t : $time\ in\ minutes$

c_{enzyme} : $enzyme\ concentration\ in\ mg/ml$

The resulting graph is shown in Figure 2.24. The activities for the samples with and without the influence of the AMF at 4 °C show a range between 0.26 to 0.34 U/mg. The positive

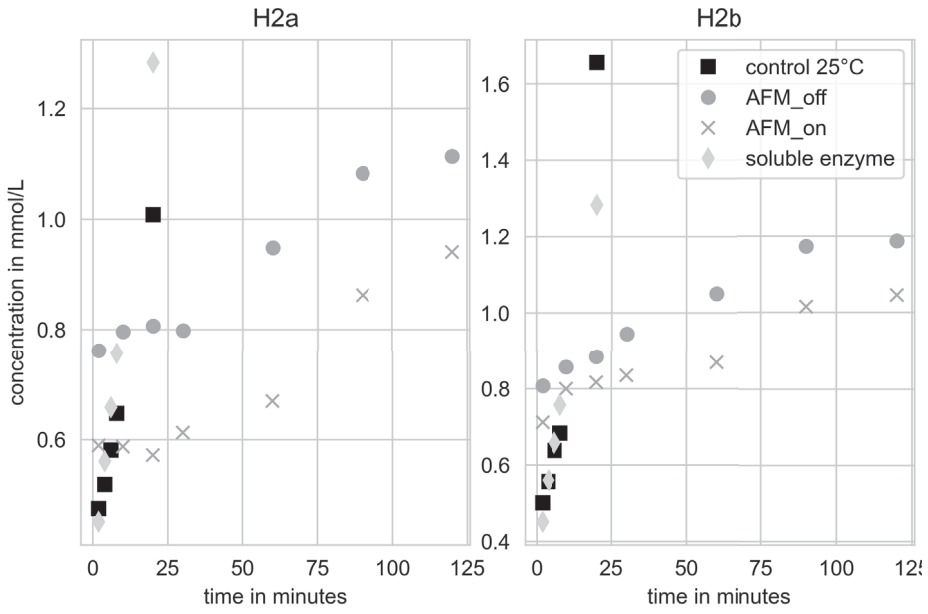

Figure 2.23: Time concentration values of the experiments performed in the Hotzymes project. The reaction solution contained 100 mmol/L HEPES (pH 7.5), 0.1 mmol/L PLP, 5 mmol/L Pyruvate, 2.5 mmol/L *a*-MBA. *Bm*ta concentrations were 0.1 mg/mL (H2a) and 0.2 mg/mL (H2b) (both immobilised) and 0.1 mg/mL (soluble enzyme). The control reaction was performed at 25 °C without the influence of an AMF, the other reactions were done at 4°C, with the AFM on (AFF_on, soluble enzyme) and the AFM off (AFM_off).

controls show an activity which is in the case of H2b at least 2.7 times greater than the respective measured samples. In the case of H2a, the difference is even greater, here the positive controls show an activity which was 6.1 times higher than the samples. Furthermore, no difference could be observed in the samples if the AMF was present or not. The enzyme activity comparisons match the conclusion made from the visualisation and suggest that the heat produced due to the exposure of the nanoparticles to the AMF is not sufficient to increase the activity of the enzymes bound to the nanoparticles under the used reaction conditions.

This data serves as a representation of negative data. As described in the outline, (chapter 2.3.1.1) the expected effect of the exposure of enzymes (H2a, H2b) bound to magnetic nanoparticles and exposed to the AMF was that an increase of enzyme activity is

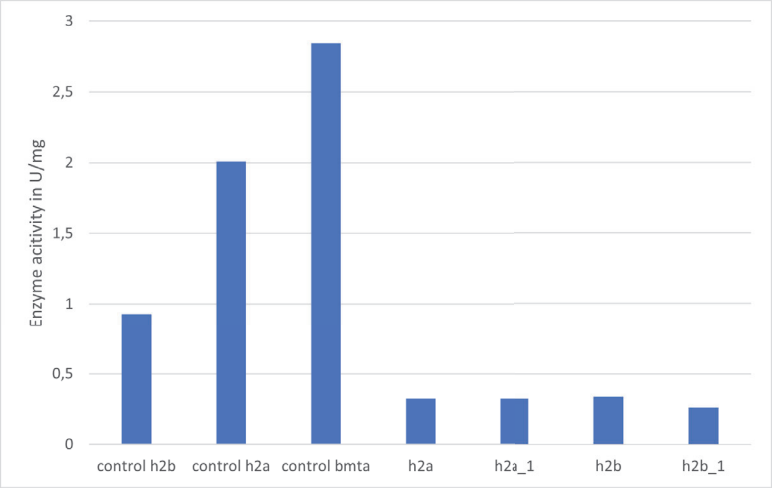

Figure 2.24: Enzyme activities in U/mg of the Hotzymes project. The reaction solution contained 100 mmol/L HEPES (pH 7.5), 0.1 mmol/L PLP, 5 mmol/L Pyruvate, 2.5 mmol/L *a*-MBA. *Bm*ta concentrations were 0.1 mg/mL (H2a) and 0.2 mg/mL (H2b) (both immobilised) and 0.1 mg/mL (soluble enzyme). The control reactions were performed at 25 °C without the influence of an AMF (control h2b, control h2a, control bmta), the other reactions were done at 4°C in duplicates (h2a_1, h2b_1) with the AFM on.

observed due to a local heat gradient around the nanoparticles. Since no significant increase in enzyme activity was seen, the hypothesis is not supported, making the data an example of negative data. It would be very likely that this data would not be published by experimenters in peer-reviewed articles or on repositories like Zenodo and thereby also not available for other researchers [108]. Nevertheless, it contains important information about a not working approach with a defined catalyst and defined reaction conditions. For this purpose, the data was published in two different repositories. First, the EnzymeML documents were individually uploaded and published to Zenodo, assigning the data sets with DOIs. This was performed as described in chapter 2.2.2. The EnzymeML documents can be downloaded from Zenodo at any time and reviewed in the BioCatHub-platform.

To make the collected primary and metadata as well as the conclusions available for other researchers, a repository was created on GitHub, and the bch-data-processing repository was uploaded [111]. The discussion and conclusions were added to the readme.md file,

which is visualised directly by GitHub (Figure 2.25).

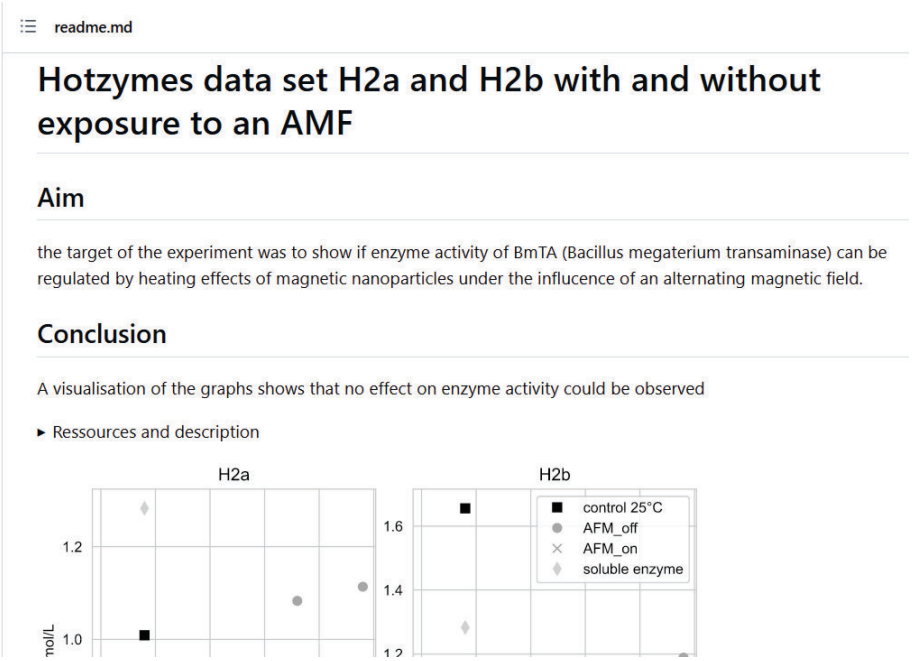

Figure 2.25: GitHub repository screenshot of the Hotzymes scenario.

With data uploaded to GitHub, only the link of the GitHub repository is required to access the data. This link can be added to posters, reports, publications or presentations, or shared with other experimenters as demanded by data management guidelines from e.g, the German Research Association (DFG) or the European Research Council (ERC) [41, 40].

This scenario shows how the BioCatHub-platform can be used for structured data acquisition and provide structured output usable by the *bch-data-processing* repository. By this, not only the experimental data is publicly available in a machine-readable format (in EnzymeML) on Zenodo but also the data processing and visualisation workflows are available over an internet connection on GitHub. Moreover, this scenario provides an example of how negative data can be published in a comprehensible and easily accessible way, without creating large additional efforts for experimenters. Thereby, other scientists can profit from this data by e.g., comparing their experimental setups with the ones from these ex-

periments to identify important differences or prevent the reproduction of negative data, saving time, money and resources.

2.3.2 Scenario 2:Reuse of already existing data for the application of a kinetic model

When experiments are performed in biocatalysis, data often need to be collected, organised, processed and stored. In scenario 1, it was described how this can be performed using the BioCatHub-platform and the *bch-data-processing* repository, where positive and negative data can be embedded. For example, in enzymology experiments the Michaelis Menten constants Km and Vmax are frequently calculated (Cornish-Bowden 2015, S. 7). In biocatalytic approaches, conversions, yields as well as final product concentrations are often of interest (chapter 1.1.2). The exact method in which the parameters and values are calculated can substantially impact the value itsef [115]. Therefore, it is important to have not only information about the collected data available but also about the data processing procedure. The following scenario shows how the BioCatHub-platform can be used to collect data from an existing biocatalytic experiment and the bch-data-processing repository to model the time-dependent course of substrate and product concentrations. Finally, the collected data as well as the precise data processing procedure are published.

2.3.2.1 Outline and target of the project

Chiral components are important structures and building blocks for pharma and other industries. With biocatalysis, the production of these molecules can be targeted in single and multistep biocatalysis [6]. The here described use case was part of the HyImPAct (Hybrid Process for Important Precursor and Active Pharmaceutical Ingredients) project of the Bioeconomy Science Center, targeting the production of metaraminol. [6, 116]. For this purpose, a two-step cascade was implemented, producing metaraminol from 3-hydroxy benzaldehyde (figure 2.26). In the first step, a carboligation is performed, catalysed by *Acetobacter pasteurianus* pyruvate decarboxylase (*Ap*PDC), producing 3-hydroxy-

phenylacetlycarbinol (3-OH-PAC) and carbon dioxide from 3-hydroxy-benzaldehyde (3-OH-BA) and pyruvate. For the second reaction, the *Chromobacterium violaceum* amine transaminase (*Cv*2025) catalyses the reaction of 3-hydroxy-PAC and isopropylamine (IPA) to metaraminol and acetone.

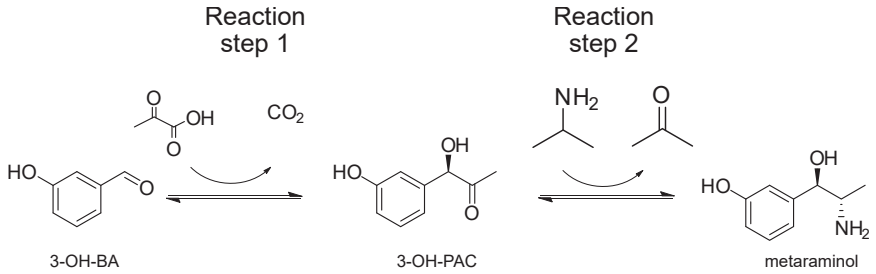

Figure 2.26: Biocatalytic cascade from 3-hydroxy-benzaldehyde to metaraminol

This scenario focussed on the collection and modelling of data from the second reaction step (Figure 2.26). In total, three different reaction scales were investigated. Every reaction was performed in a different vessel and volume. The first reaction was done in a volume of 1 mL in a shaken glass vessel, the second in 4 mL shaken glass vessel and the third in 20 mL in a stirred tank reactor. The goal of this scenario was to compare the scales with the help of a kinetic model and investigate differences and similarities between the reaction scales.

2.3.2.2 Data acquisition and processing

The data collection was carried out according to the same procedure as in scenario 1 (chapter 2.3.1.2). The experiments were monitored by measuring the substrate (3-OH-PAC) and product (metaraminol) over time with HPLC measurements. For the comparison of the reaction scales, it was aimed to describe a kinetic model based on the performed transformation. With this model, parameters were fitted on the data from the 1 mL reaction system. The modelled reaction course was then compared with the data from the other reaction scales.

Enzyme kinetics is described in most cases using the Michaelis-Menten kinetics [117]. Nevertheless, experiments were performed with lyophilised whole-cell biocatalysts in this

scenario, providing no information about the concentration and activity of the biocatalyst in the cell. Furthermore, the cell membrane is still present and acts as a transport barrier [118], although it might be damaged due to the lyophilisation process. Therefore, the processes in the reaction vessel and the whole cell biocatalyst were viewed as a "black box" approach, focussing only on the time-dependent concentration course of the substrate as well as the product. The reaction presented in Figure 2.26 is described using second-order kinetics [119] (eq 2.2). However, since the cosubstrate concentration was not measured, the second-order kinetics couldn't be used as is. Furthermore, the time-dependent concentration courses suggest that a back reaction took place, making the model even more complex. A closer look into the reaction setup shows that the cosubstrate was added in excess enabling the application of pseudo-first-order kinetics. These are kinetics which simplify second order kinetics by describing them as 'first-order reactions' [120]. By applying pseudo-first-order kinetics, two differential equations could be defined describing forward as well as back reaction (eq 2.3, 2.4).

$$\frac{dc_{(metaraminol)}}{dt} = kA * c_{(3-OH-PAC)} * c_{(IPA)} \tag{2.2}$$

$$\frac{dc_{(metaraminol)}}{dt} = kA * c_{(3-OH-PAC)} - kB * c_{(metaramionol)} \tag{2.3}$$

$$\frac{dc_{(3-OH-PAC)}}{dt} = kB * c_{(metaraminol)} - kA * c_{(3-OH-PAC)} \tag{2.4}$$

$\frac{dc}{dt}$: $reaction\ velocity$

c_{IPA} : $concentration\ of\ IPA$

$c_{3-OH-PAC}$: $concentration\ of\ 3-OH-PAC$

k : $reaction\ constant$

Using the measured data and the differential equations (eq 2.3, 2.4), parameters could be estimated with the measured substrate and product concentrations. Therefore, the differential equations were described (Figure 2.27a) and integrated using the scipy package

[58]. Lmfit was used as package, performing non-linear least squares optimisation (Figure 2.27b) [121]. The initial conditions were the substrate as well as product concentrations with 35 mml/L and 0 mmol/L, respectively.

a

```python
# differential equation for the substrate concentration

def dadt(kA, kP, A,P, t):
    dadt = -kA*A + kP*P
    return dadt

# differential equation for the product concentration

def dbdt(kA, kP, A, P, t):
        dbdt = kA*A - kP*P
        return dbdt

# differential equation system

def dglsys(init_cond_array, t, kA, kP):
    A = init_cond_array[0]
    P = init_cond_array[1]
    dAdt = dadt(kA, kP, A,P, t)
    dBdt = dbdt(kA, kP, A, P,t)
    dZdt = np.array([dAdt, dBdt])
    return dZdt
```

b

```python
# solver for the differential equation using SciPy

sol = odeint(dglsys, y0_array, time, args=(0.24262000, 0.39568405))

# parameter definition and configuration for lmfit

params = lmfit.Parameters()
params.add('kA', 2, min=0.00000001, max=3)
params.add('kP', 2, min=0.0000000001, max=3)

def residual(params, time, data_sets):
    kA = params['kA'].value
    kP = params['kP'].value

    ymodel=odeint(dglsys, y0_array, time, args=(kA, kP))
    substr = data_sets[0]-ymodel[:,0] # array 1
    product = data_sets[1]-ymodel[:,1] # array 2
    result = [substr, product] # result is returned as array
    return np.array(result)

# parameter fitting

solution = lmfit.minimize(residual, params, args=(np.array(time), data))
print(lmfit.report_fit(solution))
```

Figure 2.27: Example code for the parameter fitting using scipy and lmfit. (a) shows the definition of the differential equations. (b) depicts the integration of the differential equations and the parameter fitting.

The parameters for the reactions were estimated with $0.243 \pm 0.033\ \mathrm{min}^{-1}$ (kA) and $0.415 \pm 0.068\ \mathrm{min}^{-1}$ (kB) for the 1 mL reaction system. Multiple observations can be made if the differential equations are numerically integrated and visualised against the measured data (Figure 2.28a). The time-dependent concentration course in the 1 mL and 4 mL showed

that the fit of the measured concentration of substrates and products is rather good. With this data available, the previous assumption of the black box approach can be made, and the model is suitable to describe the reaction in the shaken system from 1 and 4 mL.

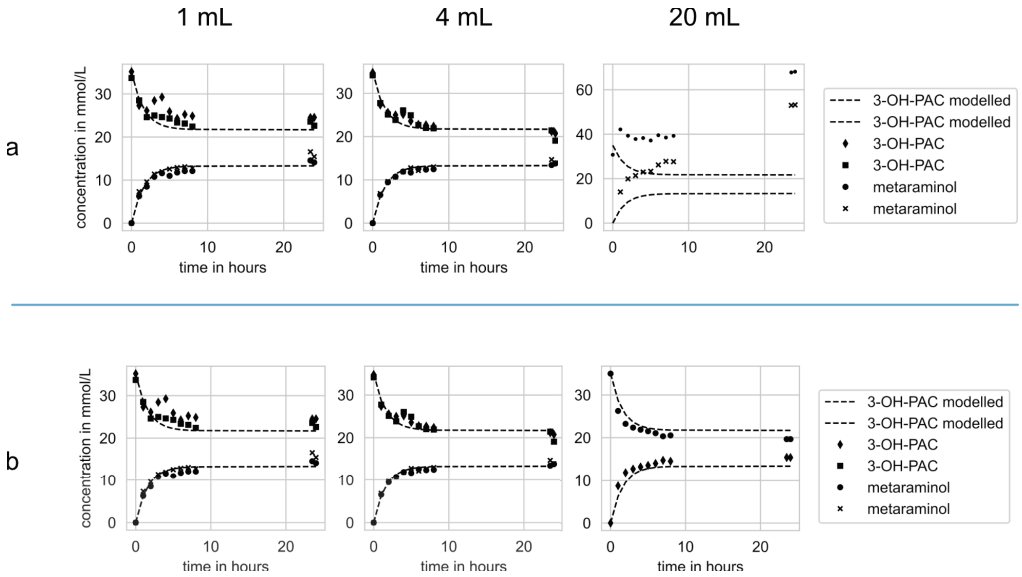

Figure 2.28: Measured and modelled time dependent concentration values of 3-OH-PAC and metaraminol. The reactions were performed at 30 °C in volumes of 1, 4 and 20 mL. The reactions in 1 and 4 mL were done as duplicates, the reaction in 20 mL as single experiment. The reaction solution consisted of 100 mmol/L KPi-buffer pH 7.5 with 40 mmol/L of 3-OH-BA. *Cv*2025 was added as lyophillised whole cell biocatalyst with a concentration of 10 mg/mL.

The reaction course in the 20 mL scale in the stirred scale did not correspond to the course predicted by the model. However, the concentration of the substrate was measured after 24 hours at 68.1 mmol/L, which is 1.9 times the concentration of the initially added substrate. As the boiling points of 1-phenylacetylcarbinol (as a comparison to 3-OH-PAC) and metaraminol are at 262 °C and 218 °C, respectively [122, 123] making it unlikely that the substrate or products evaporated. Therefore, the assumption was made that the reaction volume is decreased over time by the evaporation of water. To track the volume loss, the sum of the 3-OH-PAC and metaraminol concentrations have to result in a constant value of 35 mmol/L. To track this, the measured data from the 20 mL scale were pre-processed (Figure 2.29).

The result shows that the model fits the corrected data better, and the reaction in the 1 mL scale can be used to estimate the reaction course in a 20 mL scale. The volume loss was later confirmed in the master thesis of Nina Klos [124]. The collected EnzymeML documents, the data processing and a description were published as a GitHub repository to make the data publicly available analogue to scenario 1 (chapter 2.3.1).

```
# extracting time course data for the 20 mL scale

data_PAC = pd.to_numeric(df_20mL["1-hydroxy-1-(3-hydroxyphenyl)propan-2-one_0"])
data_metaraminol = pd.to_numeric(df_20mL["3-(2-amino-1-hydroxypropyl)phenol_0"])
data_PAC_corrected = data_PAC.fillna(35)
data_metaraminol_corrected = data_metaraminol.fillna(0)


PAC_corrected = []

# calculating the concentration sums and correction factor

result = data_PAC_corrected+data_metaraminol_corrected
factor = 35/result

# correcting the concentrations

for i,j in zip(data_PAC_corrected, factor):
    result = i*j
    PAC_corrected.append(result)

metaraminol_corrected = []

for i,j in zip(data_metaraminol_corrected, factor):
    result = i*j
    metaraminol_corrected.append(result)
```

Figure 2.29: Code example showing the concentration correction for the reaction performed in a 20 mL scale.

This scenario clearly shows how already produced data can be collected, organised, published and new knowledge extracted by analysing the data from a different perspective. With the 'black box' approach, a model was defined describing the course of the experimental data based on a pseudo-first-order kinetic. This model can be applied when performing further experiments with the above described reaction to mimic similarities and differences. Moreover, model application made the volume loss during the reaction clearly visible. Every reaction scale was performed as open system, meaning that in every reaction the evaporation of water was possible. The data indicates that the volume loss is

connected with the used reaction vessel and scale since in contrast to the 20 mL scale, no significant volume loss was discernible in the 1 and 4 mL scale. The volume correction presented above could even be implemented into the model approach to not only correct the reaction volume but also track it in dependency of e.g., different reaction vessels or temperatures. Furthermore, as the 'black box' approach is a very basic model, it can be extended, compared with new data and help to improve the understanding of the processes in the reaction vessel even further. Finally, with the publication of the data processing as Jupyter-Notebook and the data as EnzymeML documents in a GitHub repository, the collected data and the data processing is published comprehensibly. If other scientists are interested in using or understanding this approach, they can download the data, together with information about processing to follow the model definition, parameter fitting and visualisation. By this not only the data is publicly available in a comprehensible and structured way, the same holds true for the data processing.

This is useful, since multiple optimisations would be possible in the described modelling approach. First, in the presented approach parameters were estimated for the time dependent concentration courses of only one scale. The main focus in this scenario was to demonstrate a proof of concept of how the BioCatHub-platform and the bch-data-processing repository can be applied together to perform more complex data processing workflows using Jupyter-Notebooks as for example compared to scenario 1 (chapter 2.3.1). Therefore, the above described procedure was sufficient to realise this goal and it was even possible to extract new information regarding the volume loss from the experimental data. In future model based analysis, the parameters could be estimated and compared for every time dependent concentration course of the three scales. Furthermore, an application of models describing the experiment in more detail could help to extract more information from the collected data. Due to the availability of the data and the data processing details on GitHub, these approaches could easily be tested, in particular since besides the data itself, the Jupyter-Notebook is directly supplied as data processing tool.

2.3.3 Scenario 3: Collection of large data sets

In scenarios 1 and 2, data from biocatalytic experiments were collected, processed, and visualised. Nevertheless, how can the BioCatHub-platform be applied when at least ten individual experiments with at least three repeats and more than 20 measurements have to be documented? This was the challenge of the project of Alaric Prins. In this project, experiments were performed in 96 multiwell plates and measured over time in a multiwell plate reader. The resulting data was exported as text base file format and had to be extracted, sorted and organised manually. Therefore, in this scenario, the high flexibility of the *bch-data-processing* repository was used to extract and collect the primary data in an automated way to replace the time-consuming data extraction.

2.3.3.1 Outline and target

Waste treatment and the production of second-generation biofuels are important cornerstones for a sustainable economy in the future. To degrade toxic or hazardous substances like polyphenols, oxidation processes are required to make them accessible for biology based degradation processes [13]. Second-generation feedstock, e.g. lignocellulosic biomass is a highly abundant and underused source for the production of biofuels. Nevertheless, these substrates cannot be added directly into production processes but require harsh chemical and physical pre-treatment. Therefore, enzymatic processes would be a promising alternative to harsh chemical oxidation processes, since enzymes can work under mild and aqueous conditions [13]. Laccases are oxidoreductase enzymes, catalysing one-electron oxidation of various substrates, hazardous molecules, and lignin [13]. Therefore, these enzymes are in the research focus for developing sustainable and environmentally friendly alternatives. The project of Alaric Prins focuses on the identification, production and characterisation of Laccases from actinobacteria. Therefore, putative laccase sequences were identified, cloned and expressed in E.coli. In the described process, the focus was on the small laccase of *Streptomyces coelicolor* A3 (SLAC) [125, 126] The purified enzymes were characterised by estimating the Michaelis-Menten parameters km and

Vmax for the oxidation capabilities of 2,2′-azino-bis(3-ethylbenzthiazoline-6-sulphonic acid) (ABTS). For this purpose, laccase-mediated ABTS oxidation experiments were performed with a fixed enzyme and variable ABTS concentrations ranging from 0 to 175 µmol/L. Every experiment was conducted in triplicate with each samples and controls. The samples contained the laccase and ABTS. The controls only contained ABTS without the enzyme to monitor the autooxidation of ABTS (Figure 2.30).

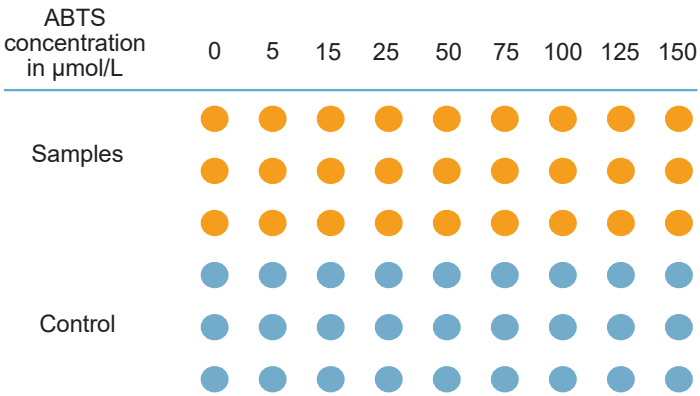

Figure 2.30: Layout of the 96-well plate used for the laccase activity experiments

The measurements were performed in a plate reader by collecting the absorption values at 420 nm every 1.5 minutes over a total time of 30 minutes. Finally, the measured absorptions were exported by the spectrometer into a text based file format. The challenge in this scenario was managing and processing the large amount of measurement data generated by the experiment. Every experiment produced 1200 individual measurement values. In the initial situation, every measurement value had to be extracted and reorganised manually by the experimenters from the text file to a spreadsheet format, creating a source of error and unnecessary work. Therefore, the goal of this use case was to automate the data extraction of the measured value, minimising the work of experiments. As a final result, only one EnzymeML document describing the metadata and one file with the output data from the plate reader should be provided, and a "one-click" solution should extract, organise and store the data sets in EnzymeML documents.

2.3.3.2 Data collection and processing

For the extraction and storage of data from the output format of the plate reader, a *bch-data-processing* repository was used and extended for this challenge. In addition to the base structure of the repository (Figure 2.20), a new folder was added, containing two python modules with functionalities for extracting and organising the primary data from the spectrometer output. Furthermore, writing EnzymeML documents was also performed by these modules. This process can be triggered by one function in the processing.ipynb notebook.

In scenarios 1 and 2, the primary data was available as a spreadsheet and added to the BioCatHub platform by the experimenter. In this scenario, this logic is reverted. The experimenter provided the plate reader's output as primary data in a text-based format, and *bch-data-processing* repository needed to build EnzymeML documents. Therefore, a three-step work process was performed. First, primary and metadata were required. Thus, the experimenter submitted an EnzymeML document as a template describing the metadata and the primary data as text-based plate reader output file. This output needed one prepossessing step for making it importable by a Python script. For this, the plate reader's output text file's first and last 20 lines were deleted, and the resulting document was stored as csv.
The second step concentrated on extracting and organising the primary data. For this purpose, the csv file was imported and transformed into a DataFrame consiting of time values as well as the sample and reference measurements (Figure 2.31).

The first Column (Column1) was selected to extract the measurement times. As shown in Figure 2.31, in between every measurement there are NaN (Not a number) values, which are empty cells in the csv files. These were removed to obtain a continuous list of measurement time values, which were formatted as dates. To make them easier processable, they were converted into seconds. The resulting array of the measurement times as seconds was converted into a DataFrame.

	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	1.0000	2.0000	3.0000	4.0000	5.0000	6.0000
2	00:00:00	25.0	0.0342	0.0893	0.1479	0.2082	0.3232	0.6293
3	NaN	NaN	0.0348	0.0880	0.1518	0.2030	0.3280	0.6418
4	NaN	NaN	0.0360	0.0893	0.1483	0.2074	0.3300	0.6483
..	...	...	...	...	...	...	...	...
123	NaN	NaN	0.0354	0.0549	0.0830	0.1096	0.1814	0.4121
124	NaN	NaN	0.0369	0.0546	0.0824	0.1114	0.1881	0.4200
125	NaN	NaN	0.0424	0.0899	0.1539	0.2267	0.3618	0.6903
126	NaN	NaN	0.0376	0.0917	0.1532	0.2294	0.3561	0.7065
127	NaN	NaN	0.0373	0.0899	0.1532	0.2282	0.3588	0.7031

Figure 2.31: DataFrame with the imported primary data from the plate reader output.

For the extraction of the absorption values, the data was imported in the same way as described above. Every column contains the absorption values for one ABTS absorption over time in triplicates. A schematic visualisation of the extraction and processing of the absortion values is available in Figure 2.32 As observed in Figure 2.31, the first two rows contain NaN values and needed to be deleted. Every column was transformed into individual DataFrames to separate the individual measured ABTS absorptions. Further, the DataFrames were converted to a NumPy array, resulting in a one-dimensional array with 126 elements. This was required since NumPy offers better data organisation functionalities than pandas. Information about NumPy is summarised in the following citation [127]. The sample and reference values are organised as sequential triplicates. To separate them, the one-dimensional array was reorganised into a 2-dimensional array with 42 elements of each 3 values. Further, the arrays were separated into samples as well as reference absorption measurements in two new 2-dimensional arrays. Finally, every triplicate in the 2-dimensional array was organised as a one-dimensional array with three values.
The last step concentrated on the creation of Combine archives by combining meta and primary data for experiments with a large number of individual measurements. To document the metadata, the previously submitted template EnzymeML document from the experimenter was used (Figure 2.33).
In this process, first, the sample and reference values are imported and annotated with

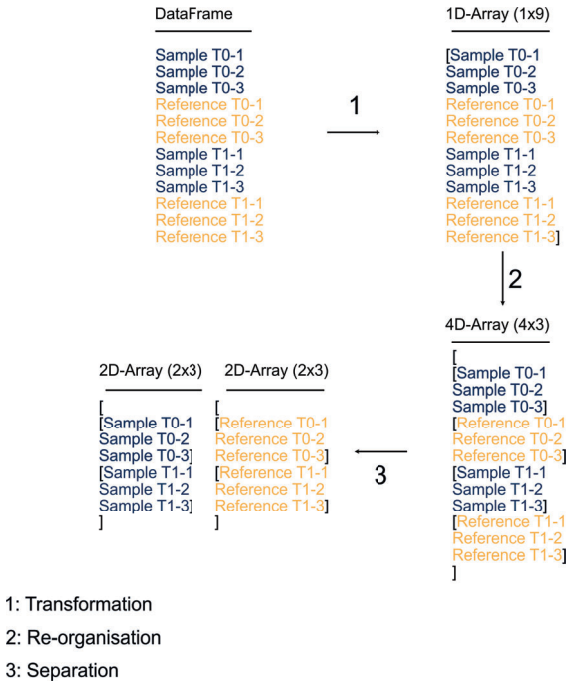

Figure 2.32: Reorganisation workflow schema for one column of the previous creates pandas DataFrame. The measured values are samples and references. The sample values contain ABTS and laccase, and the reference only the ABTS as autooxidation control (chapter 2.3.3.1). The measured values are declared starting with the value type, either sample or reference. The number of measurements is added as T and the number of the individual repeat.

```
def build_measurements_list(self):

    # extract measurement data from the readers output

    measurement_references = self.annotate_measurement(self.build_measurement("ref"), "samples")
    measurement_samples = self.annotate_measurement(self.build_measurement("sample"), "reference")
    measurements = {"measurements":[measurement_references, measurement_samples]}

    # import the previously collected metadata and define the measurement element

    json_template = open("biocathub-import.json",)
    template = json.load(json_template)
    template["enzymes"][0]["reaction"]["educts"][0]["concentration"] = self.concentration
    template["title"] = "test"
    template["vessel"] = {"type":"96 well plate", "volume":"100", "unit":"uL", "others":[]}
    template["experimentalData"] = measurements
    with open("biocathub.json", "w") as outfile:
        bch_model_json = json.dumps(template)
        outfile.write(bch_model_json)

    # write Omex document containing the extracted files

    archive=libcombine.CombineArchive()
    archive.addFile("./biocathub.json", "biocathub.json",libcombine.KnownFormats_lookupFormat("json"))
    archive.writeToFile("A540nm/AlaricnoEnzml"+str(self.concentration)+"mmolL.omex")
```

Figure 2.33: Code example showing the building of a measurement element.

already known metadata concerning the measurements (Figure 2.33). For this purpose, a BioCatHub-object is combined with the previously extracted and organised primary data. Subsequently, the BioCatHub-object is stored in a new Combine archive. Every Combine archive is annotated in the file name with the respective initial ABTS concentrations. As with the scenario 1 and 2 before (chapter 2.3.1, 2.3.2) the collected data was visualised in a similar process (Figure 2.34).

As this step is finished, the data was collected and visualised with almost no effort needed from experimenters. For this scenario one plate-readers output was available to develop the workflow and the extraction and structuring of data was realised successfully. However, as described in detail in the above, the successful data extraction, structuring and storage of the plate-readers output is highly connected to the layout of the output. It was not yet assessable, if the this output might change or other yet unpredictable challenges appear with further experiments, causing errors in the automated data extraction workflow. Therefore, one further check was added for this scenario. The experimenter has to reopen the Combine archives in the BioCatHub-platform and review the generated data and create the EnzymeML documents using the BioCatHub-platform. Once the Combine archive is uploaded to the BioCatHub-platform, the experimenter directly sees the overview page where the data can be reviewed. If the data extraction was performed correctly, the EnzymeML document can be downloaded from there and added to the respective *bch-data-processing* repository. This additional step creates only a small extra workload, but is necessary as an additional quality assurance. With more data collected and testing and eventual improvement of the implemented workflow, this additional check will probably be removable.

As in the scenarios before, the BioCatHub-data-processing repository can be uploaded to GitHub or Zenodo as is and finish the data acquisition process. The aim of this scenario was to realise an automated collection of data from the output of a 96-well plate reader and by this replace the manual time consuming data extraction which had to be performed

beforehand. Since the primary and metadata as well as the processing scripts are available on GitHub, the repository can be downloaded and the data processing extended, reused or improved. This is especially interesting if other device outputs are aimed to be extracted, structured and stored in EnzymeML documents.

This scenario showed the versatility of the BioCatHub project and the different possibilities how the BioCatHub-platform, EnzymeML and the bch-data-processing repository can interact to build completely new workflows to solve real life problems existent in the everydays worklife of experimenters.

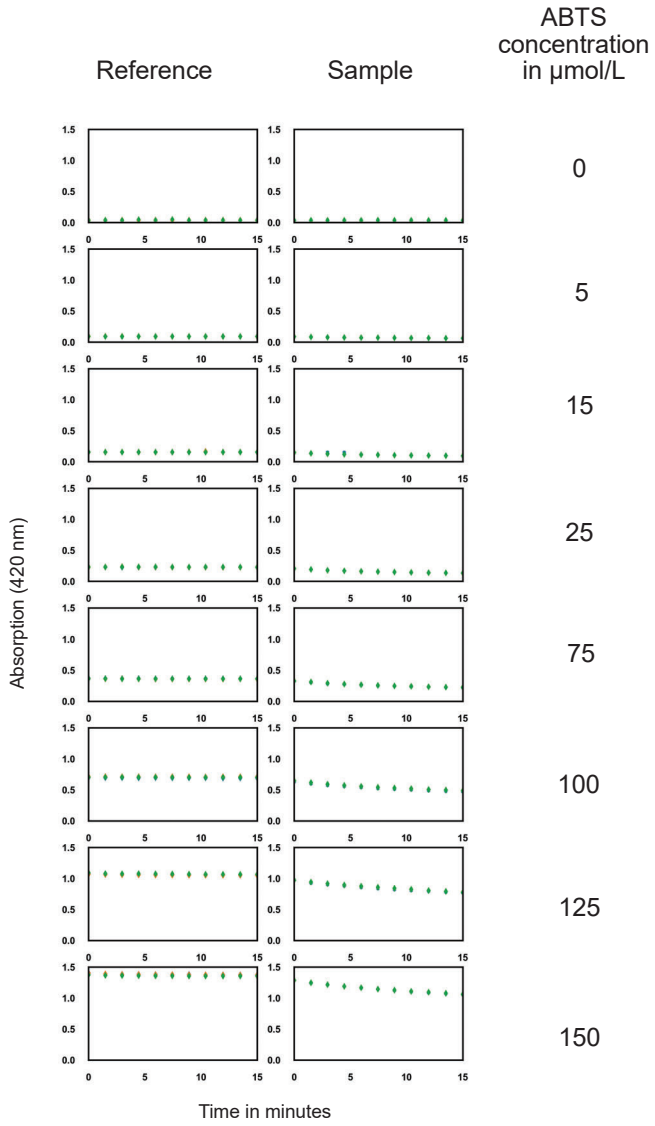

Figure 2.34: Time (min) dependent absorption values showing the laccase mediated oxidation of ABTS to ABTS+. The reaction solution consited of ABTS in concentrations between 0 and 200 µmol/L at pH3 with a laccase concentration of 0.085 µmol/L and the reactions were performed at 25 °C.

2.4 Conclusion

In the beginning of this thesis, an abstract aim was targeted: how can experimental data from biocatalysis be sustainably collected, stored and published, based on the FAIR data principles, meet the complex guidelines request by for example, funding agencies and

provide this in an experimenter friendly way (chapter 1.8). With the BioCatHub project, these goals could be realised (figure 2.36).
In the center of the BioCatHub project is the BioCatHub-platform. It provides an operating system independent web application for experimenters in biocatalysis to collect their primary- and metadata.



Figure 2.35: Overview of the data acquisition, storage, processing and publication possibilities in the BioCatHub project. Included databases are BRENDA [32], SabioRK [30, 31] and Zenodo [36]. As file format EnzymeML is used [49]. Modelling and Simulation tools show are COPASI [52] and PySCeS [53].

The implemented data acquisition process of the BioCatHub-platform is developed in a wizard-like structure, designed based on batch processes in biocatalysis, making the user interface as intuitive as possible for experimenters in biocatalysis. BRENDA and SabioRK interfaces are implemented to realise an optional and seamless integration of existing metadata, describing enzymes and reactants, further simplifying the data acquisition. This not only enhances the usability of the BioCatHub-platform but also increases the quality of the collected data. The heterogeneity of metadata from experimental procedures is a challenging task for research data management in the biological sciences.

Therefore, it is important to have an unambiguous possibility to identify metadata like the reactants or enzymes used in the experiments [128, 129]. In the BioCatHub-platform, once a database query to BRENDA and SabioRK is done, not only the name of the biocatalyst and the reactants are imported, but also the EC number to describe the catalysed reaction and the individual SMILES codes for the respective reactants are also integrated as unambiguous identifiers.
Once the data collection process is finished in the BioCatHubplatform, the collected primary and metadata can be exported as EnzymeML documents and re-uploaded to the BioCatHub-platform or shared with other scientists, creating the possibility of reusing once collected data and preventing duplication of work. Since EnzymeML is a widely applied file format in the biocatalysis and enzymology community, the output from the BioCatHub-platform can be used in combination with other tools. If modelling and simulation processes are desired to be performed with data collected in EnzymeML, tools like for example, PySCeS [53] or Copasi [52] can seamlessly make use of the EnzymeML documents. Moreover, EnzymeML documents produced by other tools or available on databases can seamlessly be applied as input for the BioCatHub-platform. With the usage of EnzymeML, the BioCatHub-platform does not apply a file format which can exclusively be used by the BioCatHub-platform. In the past, this was a major challenge since a multitude of different formats were available, often only readable by the web or software application which produced the respective format [44]. Once these web or software applications were not available any more, the collected data in the different formats got unusable, too [44]. Therewith, even if the BioCatHub-platform should not be present in the future, the collected experimental data is still applicable since it is stored in EnzymeML [49]. Finally, an interface to Zenodo enables the direct upload of EnzymeML documents from the BioCatHub-platform to Zenodo with the possibility of publishing and annotating the data unambiguously with a DOI.

The software development was planned in detail in an upstream requirements engineering process, defining the scope, limits and possibilities of the BioCatHub-platform. The

individual requirements were realised in an Angular web application as frontend, with a microservice-based backend architecture developed as Python-flask-restx RESTful APIs. With this approach, it is possible to modularise the development of the frontend and the backend in individual and manageable projects. In the backend, the microservice-based structure not only creates clarity over the backend structure by assigning individual tasks to the respective microservices but also enables the improvement and scaling of the backend. Therefore, existing microservices can be improved and connected to the bch-frontend using HTTP requests once the development has finished without impacting the usage of the BioCatHub-platform. Moreover, if new features or better approaches appear, for example, new public database interfaces or optimised packages for the interaction with Zenodo, new microservices can be developed and replace the previous obsolete ones. Furthermore, if future tasks require using other programming languages, microservices developed in the required programming language can be developed and implemented since the communication between the frontend and the backend microservices relies on the programming language-independent HTTP. The frontend development also benefits from this structure. Frontend developers can concentrate on the design and usability of the user interface since the backend tasks are outsourced to the backend microservices.

Besides the software development, the deployment was realised in docker-containers, providing a modular, standardised and yet flexible solution for deploying the web application and the microservices. Template dockerfiles were created for the frontend web application and the backend microservices. These reduce the required workload since the dockerfiles can be reused for further microservices and improve the quality of the resulting docker images, as improvements can be continuously implemented in the dockerfiles. Moreover, since the BioCatHub-platform was developed as an open source project, the deployment in docker containers enables other experimenters to run their own instances of the BioCatHub-platform. This is particularly of interest if data with restricted access, for example, in an industrial context, from biocatalytic experiments, have to be collected and stored without a direct internet connection.

To make the BioCatHub-platform available over an internet connection, the docker containers of the web application and the backend microservices were hosted in a Kubernetes cluster on a cloud computing platform. The Kubernetes cluster enables the clear and structured organisation of the docker containers required for the BioCatHub-platform. The connection to the Kubernetes cluster over the internet is realised with an Ingress as a gateway between the Kubernetes cluster and the internet. In terms of security, the Ingress provides two important safety aspects. First, the Ingress implements rules regulating the access of the docker containers in the Kubernetes cluster. By this, it is not possible to access the docker containers in the Kubernetes cluster directly, which would make them more vulnerable to attacks over an internet connection, but every request to the Kubernetes cluster is first processed by the Ingress. Secondly, a TLS certificate implemented in the Ingress of the Kubernetes cluster enables the encrypted communication between the experimenters and the BioCatHub-platform, protecting the exchanged data between the experimenter and the BioCatHub-platform from unauthorised access.
The software infrastructure is completely published in the BioCatHub GitHub organisation, enabling other software developers to cooperate and make improvements or fix problems that might occur in the software of the BioCatHub-platform.

With the BioCatHub-platform, it is possible to collect, store, publish and share experimental data from biocatalysis. However, experimenters not only need to store and collect their data, but also process, visualise and analyse it. Therefore, the *bch-data-processing* repository was developed, enabling the data processing of data stored in the EnzymeML documents created in the BioCatHub-platform. An instance of the repository can be copied from the BioCatHub GitHub organisation and includes, by default, code to extract the data previously generated by the BioCatHub-platform. Furthermore, a Jupyter-Notebook is implemented in this repository and provides a comprehensible and customisable way to process and visualise data. Moreover, the data collected with the BioCatHub-platform and the data processing can be stored comprehensibly in the respective instance of the BioCatHub-data-processing repository. A publication about the estimation of initial rates

from enzyme-catalysed reactions showed that the applied estimation procedure itself has a significant influence on the result [115]. With the *bch-data-processing* repository and the data processing in the Jupyter-Notebook, not only the collected data is stored comprehensibly but also the data processing itself is documented. Furthermore, once an instance of a *bch-data-processing* repository is published, it can be again copied and downloaded to investigate the data processing itself or even further improve the data processing.

With the development of the BioCatHub-platform, a web application is provided to collect, store, share and publish their data based on the FAIR data principles in the machine-readable file format EnzymeML. The BioCatHub-platform was designed to collect data from experimental batch processes in biocatalysis. Therefore, the user interface of the web application was designed based on experimental batch procedures in biocatalysis, supported with direct connections to the BRENDA and SabioRK databases to make the data acquisition process as intuitive and easy as possible for experimenters in biocatalysis. The BioCatHub-platform not only creates a sustainable solution for experimental data acquisition, storage and publication in biocatalysis based on the FAIR data principles but also meets the complex requirements of funding agencies like the German research association or the European research council regarding research data management.
The software development was realised as a web application with a modular and comprehensible microservice-based backend structure, simplifying the improvement and extension of the software in a clear structure. Docker containers were used for a comprehensible deployment of the web application and the microservices. Kubernetes was applied to realise the hosting of the docker containers in a clear and scaleable structure. The *bch-data-processing* repository was further developed to provide experimenters with the possibility to not only collect, store, publish and share experimental data with the BioCatHub-platform, but also visualise and process it in a comprehensible and reusable way using Jupyter-Notebooks.

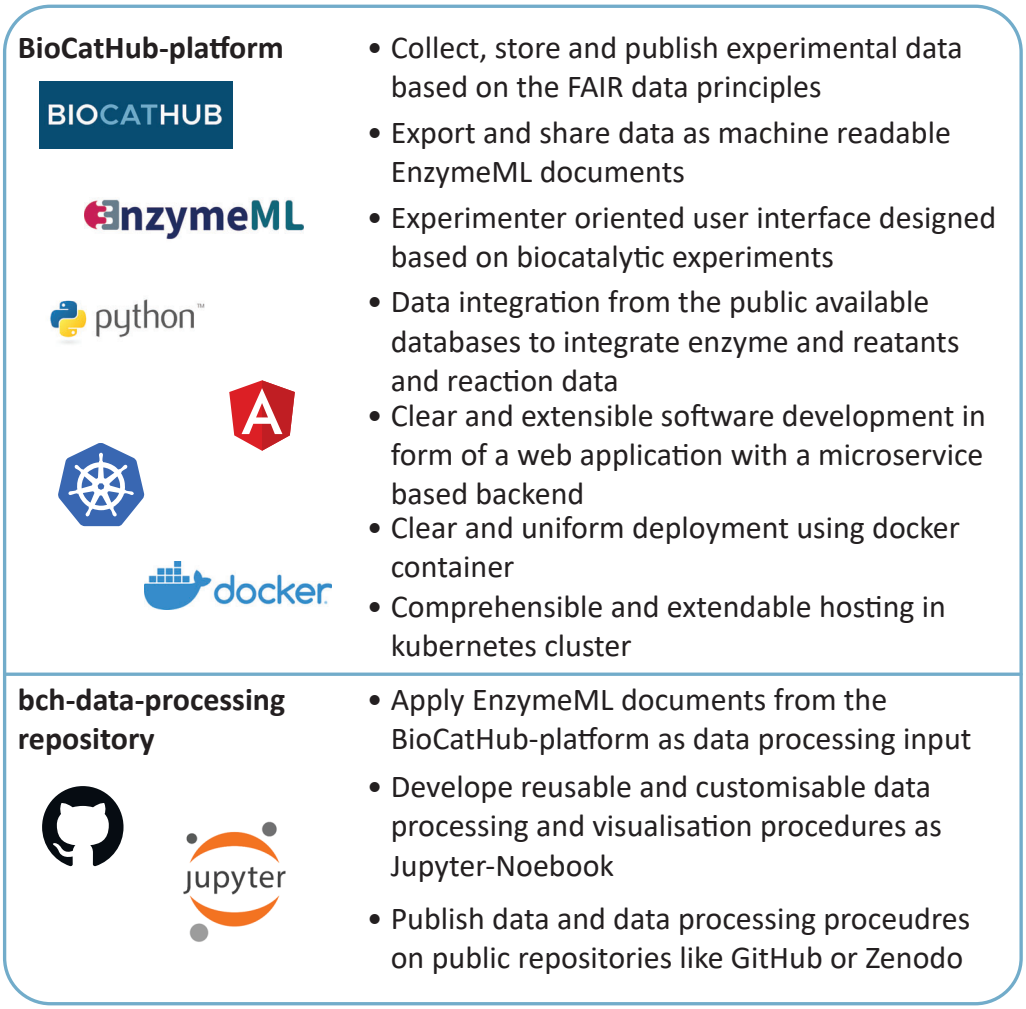

Figure 2.36: Conclusions of the BioCatHub-platform and the BioCatHub-data-processing repository. EnzymeML [49]

3 Extension: Integration of existing data from online available tool

Author's contribution

Stephan Malzacher, William Finnigan and Felix Engelmann performed the requirements engineering (chapter 3.2.1). Stephan Malzacher developed the *bch-retrobiohub* microservice (chapter 3.3) and implemented the modifications in the *bch-frontend* microservice (chapter 3.3.3). William Finnigan realised the required changes in RetroBioCat.

The BioCatHub project provides multiple possibilities to collect and integrate experimental data. The BioCatHub-platform implements a user interface for the collection of primary- and metadata from biocatalytic experiments. APIs support experimenters during the data acquisition by querying online databases like Brenda [32] or SabioRK [30]. However, this is only possible since data from public databases can be integrated by querying them with an individual enzyme name or EC number. Data integration would also be desirable for platforms like RetroBioCat, providing valuable metadata describing enzyme-catalysed retrosynthetic pathways. Nevertheless, RetroBioCat requires pre-processing steps, which cannot be implemented as easily as a database query. In order to still achieve seamless data integration, a cooperation between RetroBioCat and BioCatHub aimed to fulfil this goal.

3.1 RetroBioCat

One common target in biocatalysis is the synthesis of molecules of high economic interest using enzymes. The field of retrosynthesis provides the knowledge of how a target molecule can be synthesized [130]. Therefore, target molecules are deconstructed in order to gain information about possible precursors [131]. RetroBioCat is an online tool providing retrosynthetic pathways for enzyme-catalysed reactions [132]. Experimenters in biocatalysis can use RetroBioCat to investigate which precursors and enzymes can be applied to synthesise the desired reagent.

The usage of RetroBioCat requires structural knowledge about the target molecule. For the provision of a retrosynthetic network, the SMILES code, or the structure of the target molecule needs to be provided to the ‘network explorer´ configuration (Figure 3.1). As a result, multiple reaction pathways synthesising the required target molecule are presented in the reaction network (Figure 3.2). Besides the ‘network explorer´, RetroBioCat provides a ‘pathway explorer´ and the biocatalysis database (BioCatDB). Since for the description of RetroBioHub only the network explorer is required, the other resources should be mentioned but not discussed in depth.

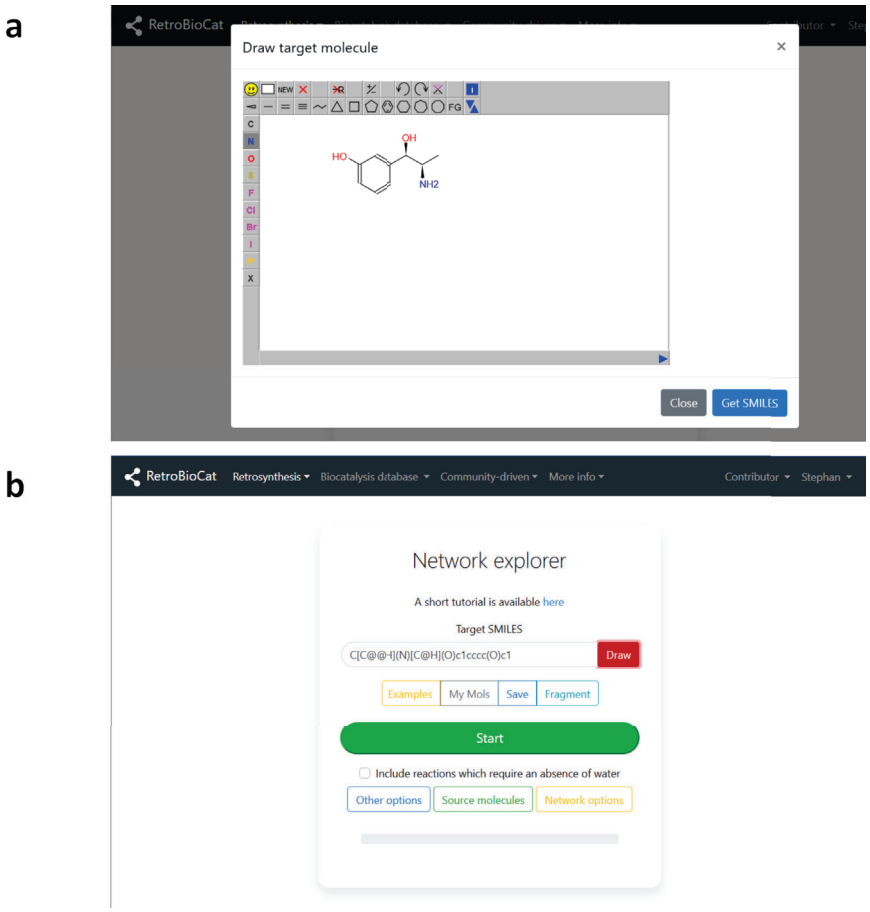

Figure 3.1: Screenshot displaying the Network explorer configuration.



Figure 3.2: Network explorer of RetroBioCat.

3.2 RetroBioHub requirements engineering

3.2.1 Vision and use cases

The requirements engineering was performed as described in chapter 2.1. The vision of the RetroBioHub extension was to provide a seamless communication between RetroBioCat and the BioCatHub-platform. This vision enables the usage of the retrosynthetic pathways available in RetroBioCat (chapter 3.1) and integrates them directly into the planning or documentation process performed in the BioCatHub-platform. The aim was to provide experimenters with a one or two-mouse-click solution to transfer data from RetroBioCat to the BioCatHub-platform. With this, it is aimed to reduce the workload of experimenters during the planning or documentation of reactions since they would not have to manually document their reactions but directly collect metadata about the reaction paths in a machine-readable way. Three use cases were identified for the RetroBioHub extension.

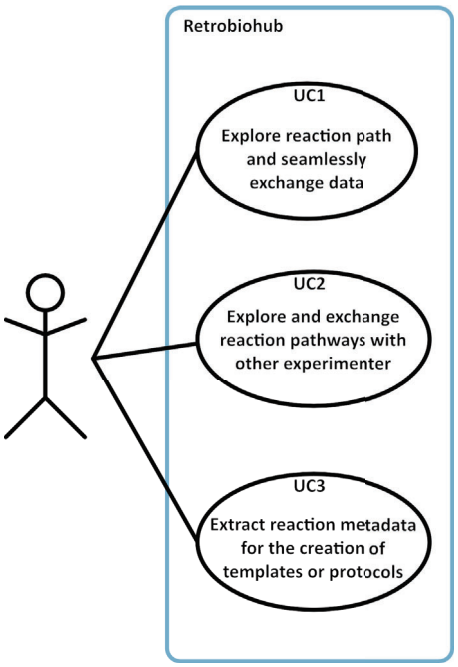

Figure 3.3: Use case diagram for the requirements of the RetroBioHub extension.

Use case 1: Explore and document reaction paths for a specific research interest. Using RetroBioCat, different routes for a target molecule can be investigated, and a comparison with the resources available in the laboratory can be made. With RetroBioHub, potential reaction paths can be extracted and stored without the risk of adding mistakes due to human errors (see also use case 3).

Use case 2: As many scientists work in intensive cooperations, with the RetroBioHub extension reaction paths from RetroBioCat can be extracted and shared as EnzymeML documents with other experimenters.

Use case 3: Many errors in documentation are made due to human errors. RetroBioHub can be used to create templates for the collection of experimental data, simplifying the creation of templates as well as protocols and reducing human errors to a minimum.

3.2.2 Requirements specification

Based on the vision and use cases, requirements could be defined and specified to realise the aimed communication between RetroBioCat and the BioCatHub-platform. First, the metadata exchange needed to be conceptualised (Table 7.5). Therefore, the definition of data models for RetroBioCat and BioCatHub was specified. Next, software was needed, transforming metadata from the RetroBioCat model to the BioCatHub model (Table 7.6). Therefore, a microservice (chapter 1.6.3) was required, addressable by RetroBioCat via HTTP requests to submit data from RetroBioCat to the microservice. Nevertheless, once the data is transformed into a representation of the BioCatHub model, it needed to be stored to be accessible for the *bch-frontend* web application. For this task, a MongoDB database was specified. Finally, once the data transformation and storage processes were defined, the experimenter needed be directed from RetroBioCat to the BioCatHub-platform where the exchanged data is presented (Table 7.7). For this process, a redirect to BioCatHub was specified, together with the database ID of the previously stored metadata, enabling BioCatHub to extract and process the metadata.

3.3 Development of RetroBioHub

Based on the requirements, the concept needed to realise the RetroBioHub extension was defined. The central element of the extension was the bch-retrobiohub microservice handling the data transformation. Modifications in RetroBioCat and the bch-frontend microservice (chapter 2.2.2.1) enabled the interaction with the bch-retrobiohub microservice and finally the communication between the platforms.

3.3.1 RetroBioCat and BioCatHub data model

To make the data provided by RetroBioCat available for the bch-frontend microservice, it has to be transformed into the BioCatHub-object. Therefore, the first step in development of the bch-retrobiohub microservice was the description of the data model (chapter 2.3). Metadata from RetroBioCat focuses on the description of reactions. Therefore, a model with two hierarchies was defined by William Finnigan, curator of RetroBioCat (Figure 3.4a). The first hierarchy (Experiment) defines the number of reactions present in the experiment. Every reaction element includes in the second hierarchy the cofactors, enzyme, reaction title as well as the substrates and products.
The enzyme and cofactors are described in the model by the available trivial names, whereas products and substrates are implemented as SMILES code. Since metadata provided by RetroBioCat focuses on the description of the enzyme-catalysed reaction, only a part of the BioCatHub-model could be mapped. A detailed description of the BioCatHub data model is available in chapter 2.2.1. Therefore, a reduced variant of the BioCatHub-model named BioCatHub-rbh-model was defined. This consists of the title and the enzyme element in the experiment root-element. For every enzyme element, the name of the biocatalyst as well as the reactions transformed are defined in the BioCatHub-model. In every reactant element, the name, SMILES code as well as the role of the reactant is specified.

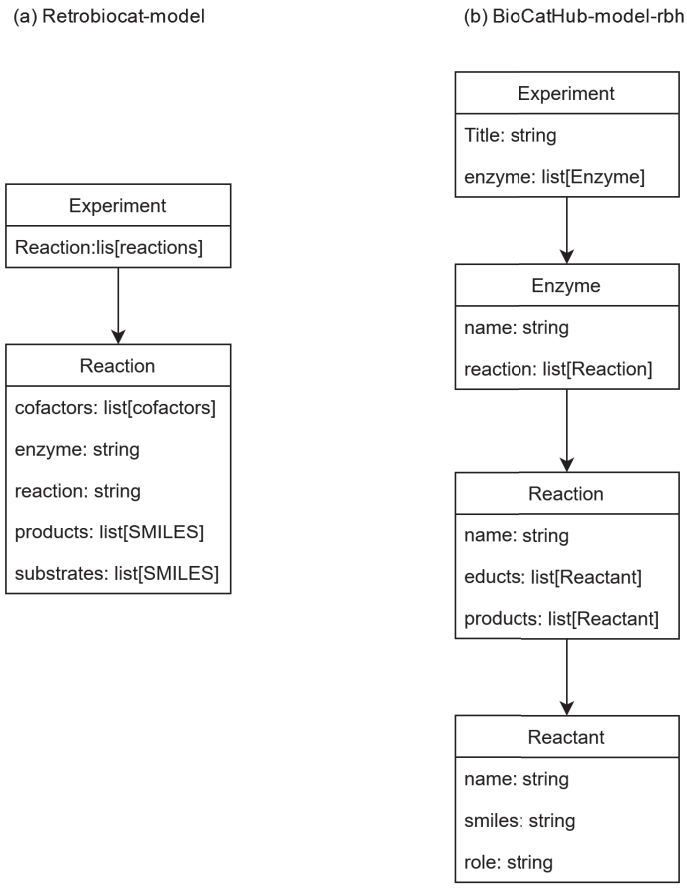

Figure 3.4: Data model of RetroBioCat (a), as well as the reduced data model for the BioCatHub platform (b).

3.3.2 Bch-retrobiohub development

For the process described in chapter 3.2.2, a microservice was specified, performing the data transformation and communication with the database. The microservice, named *bch-retrobiohub* was developed as Python flask RESTful API, along the lines of the APIs described in chapter 2.2.3.

Two paths are available in bch-retrobiohub, *retrobiohub/* and *retrobiohub/minimongo*. The *retrobiohub/* path performs the transformation of the 'RetroBioCat-object' to the 'BioCatHub-rbh-object' as well as the storage of the BioCatHub-rbh-object in the 'MiniMongo' database (described below). This process is separated into two sub-processes. Firstly, upon receiving the HTTP request, the attached JSON (RetroBioCat-object) is transformed into the

Table 3.1: Paths available in the bch-retrobiohub microservice.

path	**data**	**method**	**return**
retrobiohub/	RetroBioCat-object	POST	MiniMongo Id
retrobiohub/minimongo	MiniMongo Id	POST	biocathub-json

BioCatHub-rbh-object. The transformation was performed similarly as described in chapter 2.2.3.1. In the BioCatHub-rbh-model, the reagent name is listed, which is missing in the RetroBioCat-model (Figure 3.4). As the submitted RetroBioCat-object describes reagents based on the SMILES code, this can be utilised to retrieve the respective IUPAC name from PubChem [29, 133].

Therefore, the PubChemPy package was used to search for the IUPAC name based on the SMILES code in PubChem [134]. From the response, the IUPAC name is extracted, and the BioCatHub-rbh-object is instantiated together with the values from the RetroBioCat-object. The second subprocess exports the BioCatHub-rbh-object as JSON object and stores it on MiniMongo. Access to the database was implemented using the PyMongo [135] python package. For the connection to MiniMongo, an access token was created on the MongoDB instance, providing a username, password and connection string, which was used to connect to MiniMongo ((Figure 3.5a)). Every data set stored in MiniMongo can be retrieved using an unambiguous identifier ('MiniMongoID'), which is returned as the last part action of the request (Figure 3.5b).
The second available route, *retrobiohub/minimongo* can be addressed by HTTP request, submitting the 'MiniMongoID'. This is used to extract the respective BioCatHub-rbh-object using pymongo (Figure 6c) from 'MiniMongo'. Subsequently, the BioCatHub-rbh-model is returned, finishing the POST request.

a

```
# method to to build a connection to MiniMongo

  def mongo_connection(self):

        # define username and password

        pw = "username"
        un = "Password"

        # build connection to MiniMongo

        client = pymongo.MongoClient('mongodb+srv://{}:{}!!!@cluster0.cgs1w.mongodb.net
/myFirstDatabase?retryWrites=true&w=majority&authSource=admin'.format(un, pw))
        return client
```

b

```
# method to upload data to MiniMongo

def push_document(self, payload):

        # build connection to MiniMongo
        client = self.mongo_connection()
        db = client["Minimongo"]
        collection = db["retrobiohub"]

        # define the upload

        post = {"experiment":payload}
        upload = collection.insert_one(post)

        # return the ID of the uploaded data

        id = upload.inserted_id
        print(id)
        id_string = str(id)
        return id_string
```

c

```
def get_collection_by_id(self, id):

    # get object ID

    newId = ObjectId(id)

    # build connection to MiniMongo

    client = self.mongo_connection()
    db = client["Minimongo"]
    collection = db["retrobiohub"]

    # extract data from MiniMongo

    extract = collection.find_one({'_id':newId})
    return extract
```

Figure 3.5: (a) Displays example code for the connection to the MiniMongo database. (b) Example code for uploading data to MiniMongo. (c) Example code for querying the MiniMongo database.

The deplyoment and hosting of the *bch-retrobiohub* microservice was realised analogue to the microservices in the BioCatHub-platform (chapter 2.2.4)

3.3.3 Bch-frontend modifications

With the development of the *bch-retrobiohub* microservice as described above, the foundation for the data exchange between RetroBioCat and the BioCatHub-platform was made. Nevertheless, *bch-frontend* must communicate with *bch-retrobiohub*, retrieving the previously stored BioCatHub-rbh-object. Therefore, three modifications were made in the *bch-frontend* microservice (Figure 3.6). Firstly, the path */rbh* was added to the router module (chapter 2.2.2), specifying the start component. Secondly, in the start component, a function was implemented to extract the parameter name from the URL, which contains the MiniMongoID. Subsequently, MinoMongoID is used for an HTTP request to the */retrobiohub/minimongo* route of the bch-retrobiohub microservice (chapter 3.3.2). As response, the BioCatHub-object was returned and applied in *bch-frontend*. Finally, a redirect to the */biocatalyst* route of the bch-frontend microservice was performed.

```
private extractGETRequest(){
  this.loading = true

    # extract MiniMongoID and retrieve data from bch-retrobiohub

  this.routertype.queryParams.subscribe(params => {
      if (params){
      this.loading=true
      this.experimentService.retrobiocatDBCall(params).subscribe(payload => {
        if(payload["status"]=="not found"){
          this.loading=false;
          this.router.navigate(["/dashboard"])
          window.alert("There was en error during the import")
        }
        else{

    # implement the retieved data

          for (let i in payload){
            let others = payload[i]
            others["others"] = []
            newExp.push(others)
          }
          this.experiment.enzymes = newExp
          this.loading=false;

    # navigate to the biocatalyst entry

          this.router.navigate(["/biokatalyst"])}}

    })}
  })
}
```

Figure 3.6: Example code displaying the modifications in the bch-frontend web application.

3.4 RetroBioHub data exchange workflow

With the *bch-retrobiohub* (chapter 3.3.2) and the modifications in the *bch-frontend* (chapter 3.3.3), the communication between RetroBioCat and BioCatHub could be performed. The data exchange process is divided into two subprocesses. The first process consists of sending data to *bch-retrobiohub*, where it is transformed and stored in MiniMongo. In the second process, the experimenter is redirected to the BioCatHub-platform and the previously stored data is displayed. The following description is an example in which metaraminol is the product of interest from the scenario 2 (chapter 2.3.2).

The first process starts in RetroBioCat. As displayed in Figure 3.1, the experimenter has to select a target molecule to be synthesised in a biocatalytic process (metaraminol). With

metaraminol selected, the network explorer is displaying the possible biocatalytic reactions to synthesise metaraminol (Figure 3.2). As one of the reactions is selected for an experiment, a window shows the reaction and the 'send to BioCatHub' button (Figure 3.7). The button sends a POST request to the *bch-retrobiohub* path */retrobiohub*, to which the RetroBioCat-object is attached.

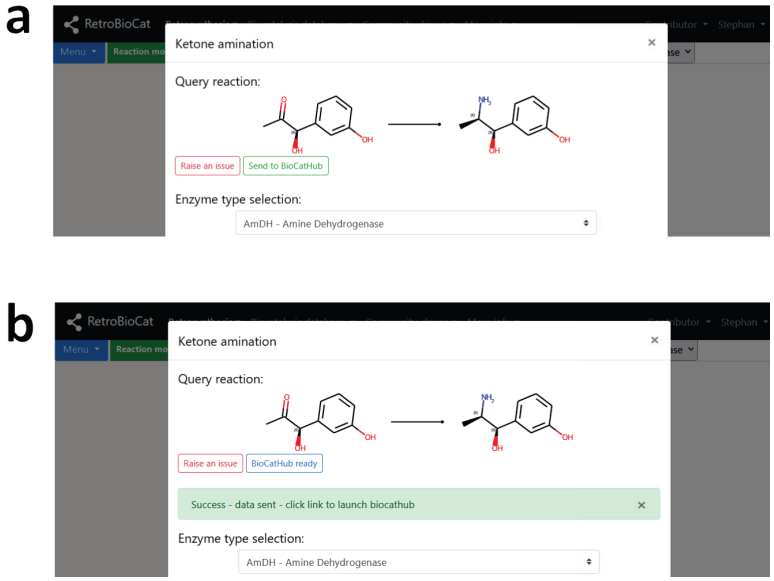

Figure 3.7: RetroBioCat screenshots of the selection of Network explorer nodes.

The request triggers the workflow described in chapter 3.3.2. The RetroBioCat-object is transformed into the 'BioCatHub-rbh-object' and stored on MiniMongo. Subsequently, the MiniMongoID is returned and available for RetroBioCat. The second process starts as soon as the MiniMongoID is available to RetroBioCat. A redirect is made from RetroBioCat to BioCatHub with the MiniMongoID provided as parameter in the URL. *Bch-frontend* extracts the MiniMongoID and sends an HTTP request to *bch-retrobiohub*, again with the 'MiniMongoID' as a parameter. *Bch-retrobiohub* retrieves the BioCatHub-rbh-object stored in MiniMongo and returns it to the *bch-frontend*, where it is processed and displayed.
The experimenter is seamlessly directed to the biocatalyst entry of the BioCatHub-platform (Figure 3.8a). In this entry, the experimenter finds the name of the biocatalyst prefilled in the name field. In the reactions entry, the IUPAC name and SMILES code of the reactants are defined (Figure 3.8b). From this point on, experimenters can utilise the collected data.

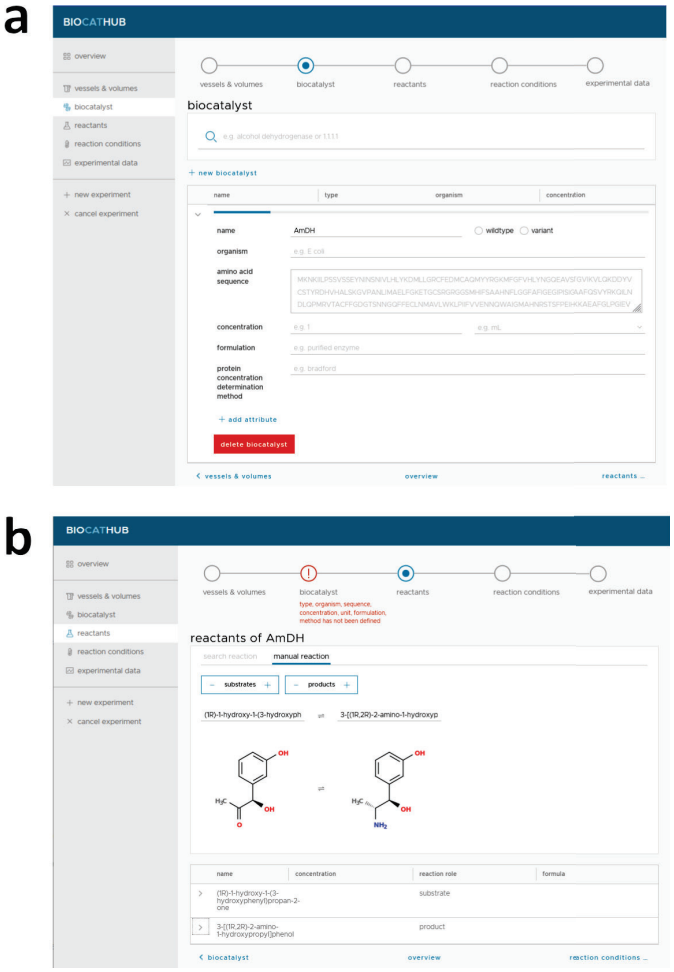

Figure 3.8: Screenshots of the BioCatHub-platform after the data submission process from RetroBioCat.

This approach was also used to document the reactions and biocatalysts in the previously described scenarios (chapter 2.3).

3.5 Conclusion

With the development of the BioCatHub-platform, a data acquisition platform based on the FAIR data principles for experimenters in biocatalysis is available, allowing experimenters to collect primary and metadata over the user interface of a web application. Implemented in the BioCatHub-platform is already the automated annotation of enzyme and reactants metadata from the publicly available databases BRENDA and SabioRK using simple database queries, implemented in the BioCatHub-platform. These queries to BRENDA and SabioRK could be easily implemented, but they quickly reach their limits, as they only allow to search for a reaction that is linked to an EC number. With the RetroBioHub extension, it is now possible to seamlessly integrate metadata about enzyme-catalysed retrosynthetic pathways from RetroBioCat, where the metadata acquisition cannot be realised by a simple HTTP request from the BioCatHub-platform since multiple manual pre-processing steps are required in RetroBioCat, before the metadata can be integrated. It is now possible for experimenters to search and explore RetroBioCat for enzyme-catalysed retrosynthetic pathways and integrate metadata about the reactants and enzymes of this pathway with only two mouse clicks into the BioCatHub-platform. With the integration of metadata about retrosynthetic pathways, the data acquisition possibilities of the BioCatHub project could be further extended with the online available tool RetroBioCat. (Figure 3.10).

The RetroBioHub extension creates, in particular two major benefits for experimenters in biocatalysis. First, enabled by the direct communication between RetroBioCat and the BioCatHub-platform, experimenters save time and prevent the duplication of documentation work. While planning or documenting reaction pathways in RetroBioCat, experimenters already invest time by describing their target molecule with a SMILES code or as structure in the structure editor of RetroBioCat and need to investigate the respective reac-

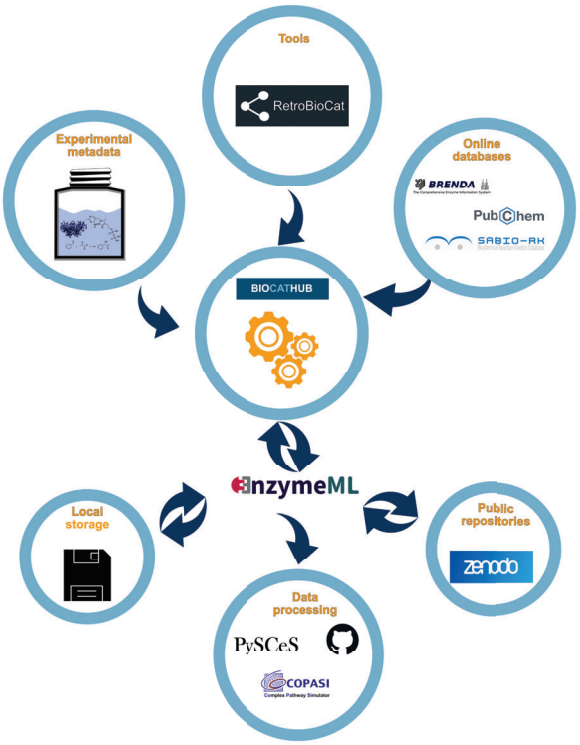

Figure 3.9: Overview of the data acquisition, storage, processing and publication possibilities in the BioCatHub project. Included databases are BRENDA [32], SabioRK [30, 31], Zenodo [36] and PubChem [29]. As file format EnzymeML is used [49]. As tools, COPASI [52] and PySCeS [53] and RetroBioCat [132] are shown.

tion pathways. Without the RetroBioHub extension, the metadata about the enzymes and reactants of a retrosythetic pathway investigated in RetroBioCat needed to be transferred manually from RetroBioCat to the BioCatHub-platform. With the RetroBioHub extension, this is performed automatically by two mouse clicks and in addition to reduction of work, errors due to manual copying can be avoided.

The second benefit is that especially with RetroBioCat a highly valuable information source for scientists in biocatalysis could be implemented. As described in chapter 1.1.1, biocatalyis often focuses on the development of enzyme catalysed processes, usually with the goal to synthesise a target molecule. This is exactly where RetroBioCat is of value. With a large, often supported by peer-reviewed articles, set of enzyme-catalysed retrosynthetic pathways, scientists in biocatalysis have access to a valuable information resource

in RetroBioCat, saving time through otherwise time-consuming literature research.
Examples of the usage of the RetroBioHub extension can be found in scenario 1, 2 and 3 (chapter 2.3.1, 2.3.2, 2.3.3). These show the application of the BioCatHub-platform and the BioCatHub-data-processing repository for different biocatalytic approaches. In each of these scenarios, it was required to describe one enzyme-catalysed reaction. The RetroBioHub extension was successfully applied in every scenario to collect metadata about the enzyme and the reactants participating in the respective reactions.
The RetroBioHub extension not only simplifies the data integration from RetroBioCat to the BioCatHub-platform, but it also improves the data quality. Data in biological research is characterised as highly heterogeneous with many metadata already available but distributed in public databases and repositories [28, 136]. Therefore, it is important to integrate publicly available metadata, and even more important, describe them with unambiguous identifiers to improve the machine-readablitiy and simplify further data integration [38, 27]. The RetroBioHub extension also was designed for this goal. Once data is integrated from RetroBioCat to the BioCatHub-platform, the SMILES codes of the reactants are automatically annotated, improving the unambiguous identification of the integrated reactants based on their structure. Using for example the SMILES code as identifier, it is further possible to query publicly available databases and retrieve metadata about the respective reactant. One example for this approach was implemented in the RetroBioHub extension. Using the SMILES code, the publicly available database PubChem is queried to retrieve the IUPAC name of the reactant [134]. If future projects require the annotation of specific metadata like for example the melting temperature, the SMILES code can be used to search and retrieve the melting temperature in public databases.

From a technical perspective, the RetroBioHub extension provides an easy to realise workflow for web and software applications to communicate with the BioCatHub-platform. To realise this, the infrastructure of the BioCatHub-platform was extended with one microservice (*bch-retrobiohub*) as RESTful API and one modification in the *bch-frontend*. *Bch-retrobiohub* implements one path for the communication with RetroBioCat. Data is sub-

mitted from RetroBioCat to *bch-retrobiohub* using a HTTP-request. In *bch-retrobiohub*, submitted data is transformed to the BioCatHub-object and stored in MiniMongo. Subsequently, the MiniMongoID send as response to the HTTP request. With a redirect to the BioCatHub-platform together with the MiniMongoID, the previously stored BioCatHub-object is extracted from MiniMongo and integrated into the BioCatHub-platform. This also creates a template workflow for other web or software applications to implement a seamless data integration into the BioCatHub-platform. Therefore, a new path can be created in *bch-retrobiohub* transforming the submitted data to the BioCatHub-object and storing it in MiniMongo. Since the communication between *bch-retrobiohub* and a potential other web or software application is realised in a HTTP-request, this is even possible independent of the programming language. Other web or software applications only need to implement one HTTP request, submitting the respective metadata to the newly created path in *bch-retrobiohub* and one redirect to the BioCatHub-platform, submitting the MiniMongoID.

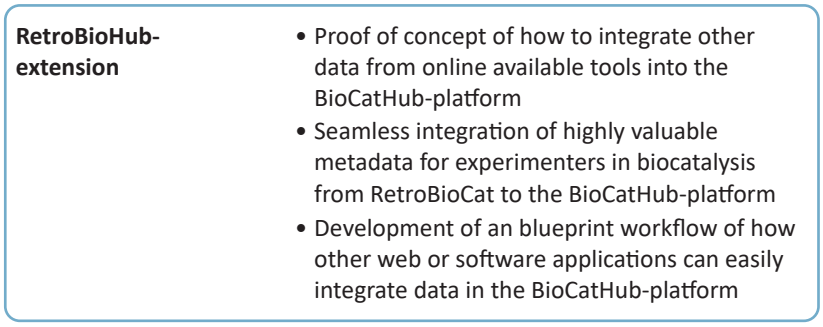

Figure 3.10: Conclusions of the RetroBioHub extension.

By the development and implementation of the RetroBioHub extension, in particular, three main goals were realised (Figure 3.10). First, it is now possible for experimenters in biocatalysis to seamlessly integrate enzyme-catalysed retrosynthetic pathway metadata from RetroBioCat. Second, it was possible to realise the communication with RetroBioCat, an online available tool, which requires manual preprocessing by scientists before it can be integrated. Third, from a technical perspective, an easily realisable technical workflow was realised, which can be used as template for the implementation of further primary

and or metadata from other web or software applications.

4 Implementation:LightCas

Author's contribution

Stephan Malzacher, Christiane Claaßen, Douglas Weber, Dominic Meißner, Doris Hahn and Dörte Rother performed the requirements engineering (chapter 4.2). The development of the Arduino module 1 was realised by Luca Fede under the supervision of Christian Dick from the Univeristy of applied sciences Köln. The Arduino Module 2 was conceptualised by Stephan Malzacher and assembled by Jürgen Paulzen and Florian Walkenbach from the electrical workshop of the Institute of Bio- and Geosciences 1 (Forschungszentrum Jülich GmbH). The software development of *ABBA* (chapter 4.4) and the automated data acquisition process (chapter 4.6) was performed by Stephan Malzacher. The software-based hardware control of the benchtop-NMR, the measurement pump, the dosage pumps and the LED-strip (chapter 4.3) was done by Stephan Malzacher. The laboratory work connected with the LightCas project was done by Douglas Weber, Dominik Meißner, Doris Hahn and Christiane Claaßen in a cooperative project. Data processing and visualisation were done by Stephan Malzacher (chapter 4.7).

With the development of the BioCatHub-platform (chapter 2), a web application is now available allowing experimenters in biocatalysis to collect their data based on the FAIR data principles. With the RetroBioHub extension, seamless data integration from the online available tool RetroBioCat to the BioCatHub-platform was realised. To make the data acquisition as coherent as possible, a further goal is the collection of data directly from where it is produced: the experimental procedure. Though, for this purpose, interaction with the software used in the laboratory during the experiment has to be possible to establish an interface to the BioCatHub-platform. However, modifying software used in biocatalytic experiments is often not possible or connected with a large effort. Therefore, it is a promising approach to implement a data acquisition strategy in application software during the development of this software.

In particular, this is the goal of the 'LightCas' implementation: developing a platform for biocatalytic experiments with the possibility to seamlessly collect the produced data once the experiment has finished and implementing this as part of the BioCatHub project.

4.1 The LightCas project

In the Hotzymes project described in scenario 2 (chapter 2.3.1), the enzyme activity is aimed to be increased due to the exposure to an alternating magnetic field, generating heat in magnetic nanoparticles and its release to the attached enzymes. The final aim was the prevention of cross-reactivities in multi-step enzyme cascades. The approach to avoid cross-reactivties of multi-enzyme catalysed processes in one pot is similar to the 'LightCas' project, with the difference that the enzyme activity is targeted to be stopped completely. For this purpose, photosensitizers were attached to the enzymes [137]. Through irradiation of light, photosensitizers produce reactive oxygen species like singlet oxygen and hydrogen peroxide which themselves cause the inactivation of the bound enzyme. Since the reactive oxygen species are short lived, other enzymes in the reaction are not inactivated [138, 137].

The overall target of the LightCas project is to produce tetrahydroisoquinolines (THIQ) in a three-step enzymatic cascade starting from 3-hydroxy-benzaldehyde (Figure 4.12a). In the first reaction step, 3-hydroxy-phenylacetylcarbonol (3-OH-BA) and carbon dioxide (CO_2) is produced from 3-OH-BA and pyruvate, catalysed by the *Acetobacter pasteurianus* pyruvate decarboxylase (*Ap*PDC). In the second reaction step, which is catalysed by the *Chromobacterium violaceum* amine transaminase (*Cv*2025), the 3-OH-PAC, produced in the first reaction step is applied together with α-methylbenzylamine (α-MBA) as cosubstrate to produce metaraminol and acetophenone. For the final reaction step, metaraminol, produced as product in the second reaction step, is applied together with an aldehyde to produce a THIQ and water in a reaction catalysed by a nocolaurine-synthase (NCS). However, mutiple side reactions are possible in this cascade when all three enzymes, cosubstrates and 3-OH-BA are present at the same time and in the same reaction vessel (Figure 4.12b). First, the (*Cv*2025) catalyses not only the reaction from 3-OH-PAC and α-MBA to metaramionol and acetophenone, but it is also possible to catalyse the reaction from 3-OH-BA with α-MBA to 3-hydroxy-benzylamine. Moreover, two more side reactions are possible with the *Cv*2025 and the *Ap*PDC and the cosubtrate (aldehyde) required for the third reaction step. In detail, the *Cv*2025 catalyses the reaction of the aldehyde required for the third reaction step, together with α-MBA to produce acetophenone and a primary amine. Furthermore, the *Ap*PDC can catalyse the formation of a α-hydroxy-ketone from the aldehyde required for the third reaction step and pyruvate. With these side reactions, undesired side products like 3-OH-benzylamine instead of 3-OH-PAC and an α-hydroxy-ketone as well as a primary amine instead of THIQ are formed, reducing the overall yield of the desired products [137, 109]. To avoid the just mentioned side reactions, the individual reaction steps need to be separated in time. Meaning that the individual enzyme-catalysed reactions are carried out one after another, with an inactivation of the enzymes once their respective step is finished. To realise the enzyme inactivation, *Ap*PDC and *Cv*2025 were fused to the photosensitizer SOPP3 resulting in *Cv*2025-SOPP3 and *Ap*PDC-SOPP3 [137]. No NCS was was fused to a photosensitizer. Since the NCS is the enzyme

required for the last of the three reaction steps, no more side reactions with a following reaction step are possible and therefore an inactivation step for this enzyme is not necessary.

As a result, the enzyme-catalysed cascade as shown in Figure 4.12a can be performed in sequential reaction steps, with the individual enzyme-catalysed reactions separated in time due to the light induced enzyme-inactivation of the *Cv*2025-SOPP3 and *Ap*PDC-SOPP3. However, the execution of the enzyme-catalysed three step cascade is highly time intensive. The concentrations of the substrates and products of the individual reaction steps need to be constantly measured manually by experimenters. Once a substrate for one of the reaction steps is consumed, the light inactivation needs to be carried out manually and the enzyme and cosubstrates for the subsequent reaction step added to the reaction vessel by an experimenter.

At this point, the LightCas implementation was planned to reduce the time consuming manual measurement. Therefore, most complex parts of the reaction, meaning the first and second reaction steps, which required the enzyme inactivation at the end should be realised as automated process, performing the online measurement of the substrate (3-OH-BA) and intermediate (3-OH-PAC) concentration, the light-inactivation of the enzymes and the feeding of the cosubstrate and enzyme for the second reaction step. In the following chapters it is described how the technical requirements were defined and developed to realise the just mentioned automation process and on top realise an automated data acquisition process with the BioCatHub-platform.

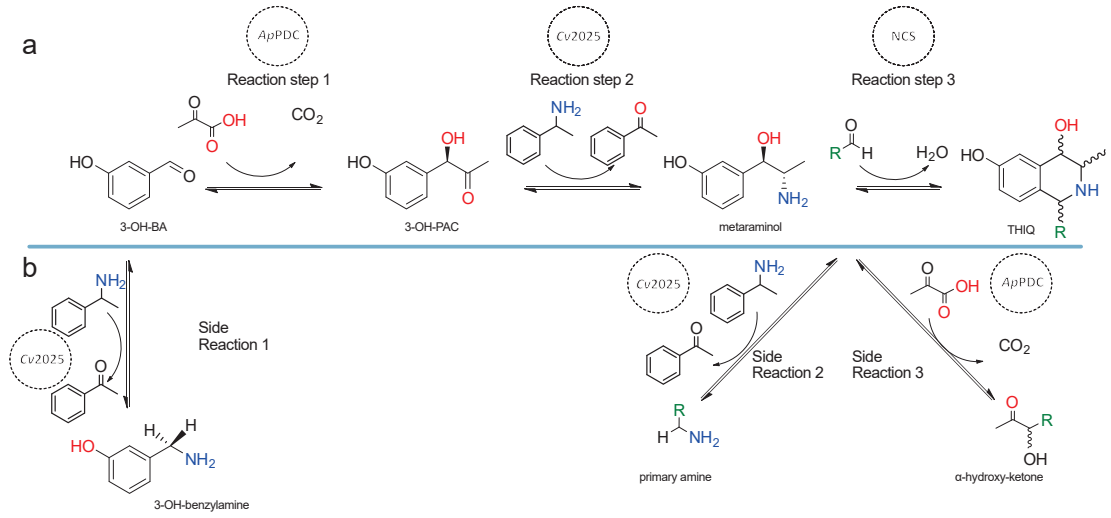

Figure 4.1: Reaction schema for the enzyme catalysed three step cascade of the 'LightCas' project. (a) shows the desired cascade from 3-hydroxy-benzaldehyde (3-OH-BA) to tetrahydroisquinoline (THIQ). The first reaction step (Reaction step 1) is catalysed by the *Acetobacter pasteurianus* pyruvate decarboxylase (*Ap*PDC), catalysing the reaction from 3-OH-BA with pyruvate as cosubstrate to 3-hydroxy-phenylacetylcarbol (3-OH-PAC) and carbon dioxide (CO_2). In the following step (Reaction step 2), the reaction of 3-OH-PAC with α-methylbenzylamine (α-MBA) as cosubstrate to metaraminol and acetophenone is catalysed by the *Chromobacterium violaceum* amine transaminase (*Cv*2025). In the last step (Reaction Step 3), the reaction of metaraminol and a primary ketone to THIQ and water is catalysed by a nocolaurine-synthase. (b) shows the side reaction possible in the cascade. The first side reaction (Side reaction 1) is catalysed by the *Cv*2025 and uses 3-OH-BA and α-MBA as cosubstrate to produce 3-hydroxy-benzylamine (3-OH-benzylamine). The second side reaction (Side reaction 2) is catalysed by the *Cv*2025 and produces a primary amine and acetophenone from the aldehyde cosubstrate required for the third reaction step (Reaction step 3) and α-MBA. The third side reaction (Side reaction 3) also uses the aldehyde which is used as cosubstrate for the third reaction step (Reaction step 3) and pyruvate to produce an α-hydroxy-ketone and (CO_2) in a reaction catalysed by the *Ap*PDC.

4.2 LightCas-implementation requirements engineering

4.2.1 Vision and Use cases

The requirements engineering for the LightCas-implementation was performed analogue to the process described in chapter 2.1. The vision of the LightCas-implementation was to develop a technical infrastructure, enabling experimenters to perform the cascade described in chapter 4 by the initial preparation of the required reactants, enzymes, and devices. The developed software and hardware infrastructure should perform all later steps, including the online measurement of the substrate and product concentrations, light inactivation and the dosage of enzyme and cosubstrate for the second reaction step after the completion of the first reaction step. At last, once the reaction is finished, an automated data transfer from the developed infrastructure to the BioCatHub-platform should be possible. With this vision defined, use cases were identified

Use case 1 The first and most important use case is automated sampling- and reaction control. Therefore, the LightCas-implementation should provide a user interface where experimenters can enter the most necessary settings (not defined here) and leave all remaining tasks to the LightCas-implementation.

Use case 2 Once the experiment is performed, the experimenter should be able to submit the collected data with a one-click action to the BioCatHub-platform.

4.2.2 Requirements specification

The individual requirements could be specified with the vision and use cases defined in chapter 4.2.1. Three categories were identified, in which the requirements were organised. The first category describes requirements for the operation technology used in the LightCas-implementation. The second category focusses on the direct interaction of the

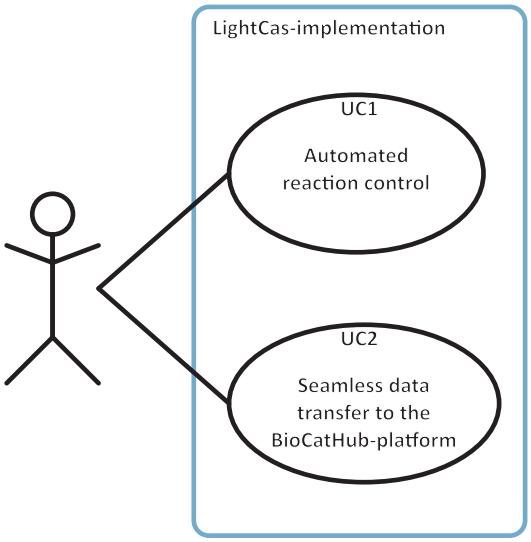

Figure 4.2: Use cases defined during the requirements engineering process of the LightCas-implementation

experimenters with the infrastructure, and the third on the requirements concerning software development.
The first question to be answered in this project was how a software-based interaction with lab devices can be established and which devices are available for developing the infrastructure (Table 4.1).

Concerning the measurement, a benchtop NMR was available, which could be used to measure the substrate and product concentrations required to realise the cascade. In previous work, it was possible to get separate signals, enabling the detection of 3-OH-BA, 3-OH-PAC and metaraminol [80]. The connection to the measurement device is possible due to a TCP/IP interface provided by the manufacturer (chaper 1.6.4). The reaction solution needs to be transported from the reaction vessel to the benchtop NMR to realise a measurement. For this purpose, a peristaltic pump from the manufacturer Longer was available. This was needed to transport the reaction solution to the NMR and back to the reaction vessel. Therefore, the pump needed to have adjustable pumping directions. Furthermore, the pumping speed had to be adjustable. To realise the control of this pump, an Arduino microcontroller should be applied (chapter 1.7.2). Furthermore, peristaltic

Table 4.1: Requirements regarding the operational technology development for the LightCas process.

Requirement	**Specification**
Controlling the dosage pumps for the enzyme and cosubstrate required for the second reaction step	Arduino based communication
Controlling the measurement pump for online NMR analytics	Arduino based communication
Controlling the LED strip	Arduino based communication
Controlling the benchtop NMR	TCP-IP (chapter 1.6.4) interface provided by the NMR

pumps were available for the dosage of cosubstrate and enzyme for the second reaction step. An Arduino microcontroller was also intended for the control of these pumps. Finally, the light inactivation had to be realised. For this purpose, an LED strip was available. The LED strip was also specified to be controlled over an Arduino microcontroller.
With the operational technology specified, it needed to be defined how experimenters can interact with the system (Table 4.2). For this purpose, a web application was specified. As the requirements for the operational technology and the user interactions were defined, it could be specified how the software engineering should be performed.
For the frontend development, an Angular app was intended. Since the BioCatHub-platform was developed with the same approach, knowledge in realising web applications is already present (chapter 2.2.2.1). Furthermore, with the bch-retrobiohub microservice a workflow was designed, realising the data exchange between online platforms (chapter 3). The same logic was identified to be applicable to the communication between the frontend in this project and the BioCatHub-platform. To realise the communication between the microcontrollers and the infrastructure, a backend in the form of a Python-Flask-restx RESTful API was defined. The PyFirmata python package enables seamless communication

Table 4.2: Requirements regarding the software development for the LightCas process.

Requirement	**Specification**
Usability for experimenters	Angular based frontend for experimenters to test the devices, visualise the measured data and start of a process, optimally automated
Software based communication with the hardware devices	Python flask-restx microservice
Data submission to the BioCatHub-platform	Usage of the RetroBioHub microservice to submit data
Controlling the benchtop NMR	TCP-IP interface provided by the NMR

between a python program and Arduino microcontrollers. For the TCP/IP communication with the NMR, the manufacturer suggests the usage of the python sockets package [139].

4.3 Hardware development for the automated cascade

As described in the requirements for the development of the LightCas system in chapter 4.2.2, the hardware needs to fulfil the tasks of concentration measurement, light inactivation and liquid transport. In total, five devices were necessary, as shown in Figure 4.3.

4.3.1 Concentration measurement during the automated process

Control variables during the automated LightCas project were the concentrations of starting material 3-OH-BA and the intermediate 3-OH-PAC. These were measured using nuclear magnetic resonance spectroscopy (chapter 1.7.1) with the Spinsolve 60 spectrometer from the manufacturer Magritek. The continuous measurement was possible due to a

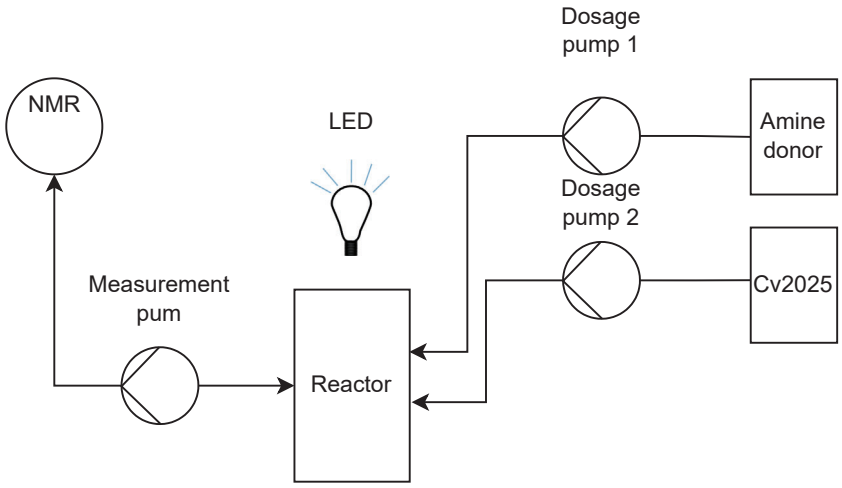

Figure 4.3: Devices applied in the automation platform of the LightCas project. The dosage pumps 1 and 2 were used to add the enzyme ($Cv2025$) and the cosubstrate (α-MBA) to the reaction vessel once the first reaction was finished. The measurement pump transported the reation solution from the reaction vessel to the benchtop NMR and back to the reaction vessel once the measurement was finished. The LED was implemented as LED strip and was switched on to inactivate the enzymes coupled with photosensitisers.

glass capillary inserted into the NMR spectrometer and connected by a silicon tube to the reaction vessel. The liquid transport was performed by a peristaltic pump (measurement pump) (chapter 4.3.2). The Spinsolve 60 benchtop-NMR spectrometer was described in detail in 2020 [80]. The control of the NMR-spectrometer is enabled by a TCP server implemented into the supplied manufacturer software (Spinsolve 2.1.5). The server is available by TCP/IP (chapter 1.6.4) and enables the remote control of the NMR spectrometer by sending XML commands predefined by the manufacturer. For the automated LightCas project, the following remote control commands were required:

1D-proton: method to run a one-dimensional proton scan.

1D-presat-multi: method starting a one-dimensional proton scan with the possibility to adjust parameters like the number of measurements or the acquisition time

Sample shim: method for shimming the NMR based on samples

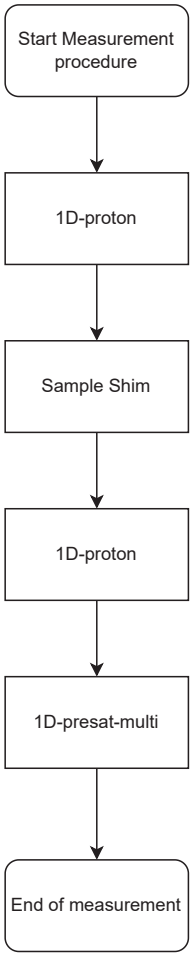

Figure 4.4: Procedure for the automated NMR measurement in the Spinsolve-60 Benchtop NMR.

Every measurement performed by the NMR spectrometer could be divided into two essential methods. The first one, the Sample shim method, was used to homogenise the magnetic field of the NMR spectrometer. The second essential method was the 'Presat-multi' method which performed the scan itself and integrated the signal in ranges which had to be predefined in the user interface of the Spinsolve software. During the measurements, it was observed that the 'Sample shim' method needed processing time after the shim was performed. Meanwhile, it was not possible to send further commands. Therefore, the '1D-proton' method was implemented to check if the system is ready. The '1D-proton' method was suitable for this task since it took only about 10 seconds to proceed and had no post-processing time. Subsequent to each scan, a folder on the local file system

was created, containing the integrals of the predefined signal ranges in a csv file, which served as the basis for data processing.

4.3.2 Liquid transport and light inactivation during the automated process

The liquid transport tasks in the automated LightCas project could be separated into two categories. The first one was the dosage of *Cv*2020 as enzyme and α-methylbenzylamine as cosubstrate for the second reaction step (dosage pumps), and the second task was the transport of the reaction mixture to the glass capillary in the NMR-spectrometer for the measurement (measurement pump) (chapter 4.3.2). The pumps were controlled by two Arduino microcontrollers: Arduino 1 and Arduino 2. An in-detail description of the architecture and development of the Arduino-based microcontroller can be found here [140].
For the 'measurement pump', a pumping procedure proposed by the company Magritek was implemented, named the reverse stop flow method. The reverse stop flow starts with filling the glass capillary with the 'measurement pump', and once the capillary is filled, the pump stops. Once the measurement is finished, the 'measurement pump' changes the direction and starts pumping again, transporting the reaction mixture back to the reaction vessel. Furthermore, the speed of the 'measurement pump' needed to be adjustable. During the filling of the glass capillary, bubble formation was observed when the speed of the pump was too fast in the first seconds.
The dosing for the second reaction step consisted of two peristaltic pumps, one supplying α-MBA and the other one the enzyme *cv2025*. It was neither necessary to adjust the speed nor the pumping direction for the 'dosing pumps'. The only parameter necessary is the on-time of the pump. For this the experimenter needs to measure the volume flow of every individual pump, which has to be determined before the reaction with the individual dosing medium. Both pumps were controlled by the Arduino 2 microcontroller.
The light inactivation of the enzymes is realised by the application of an LED strip applied to the reaction vessel. As control parameter for the LED strip, the on-time was required.

The LED strip is controlled by the Arduino 1 microcontroller.

4.4 Software development of the automated process

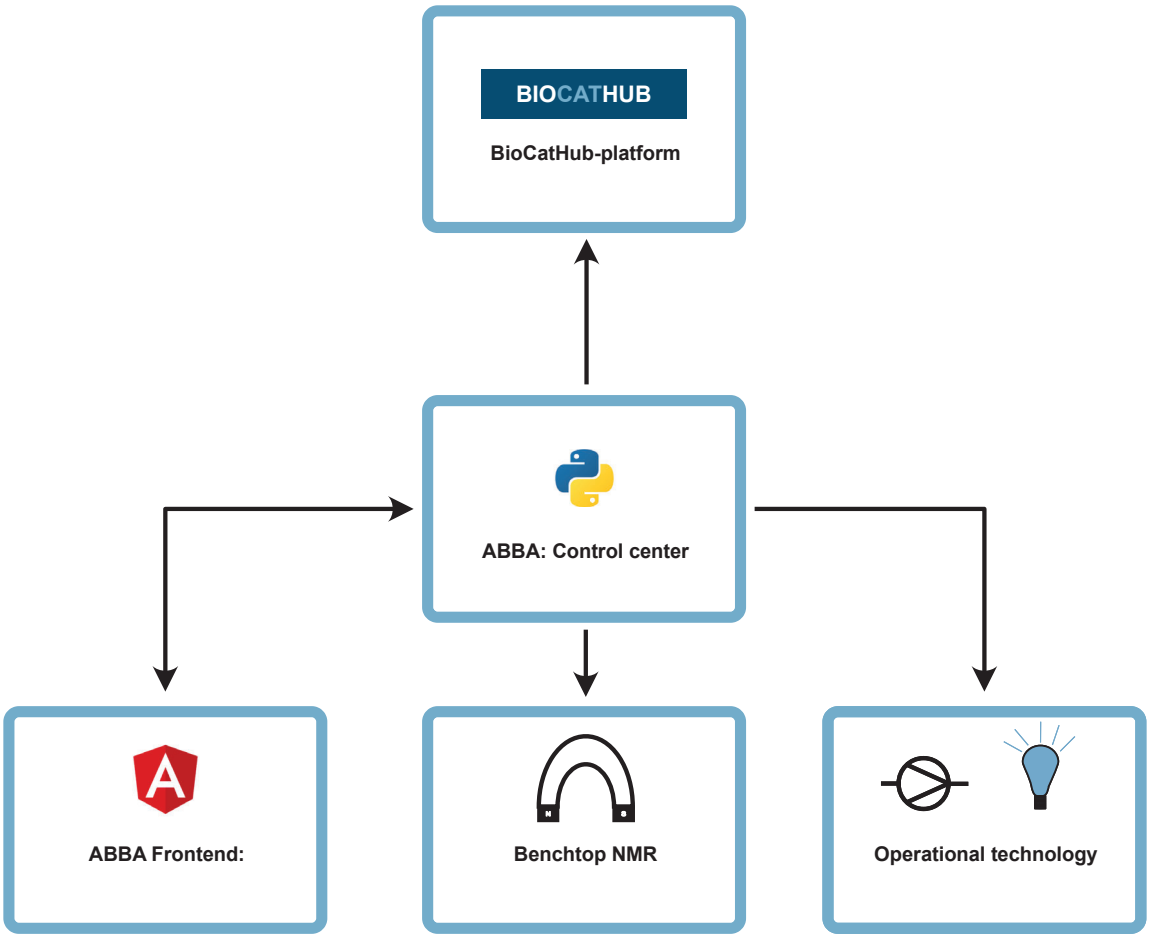

Figure 4.5: Requirements regarding the development of the BioCatHub platform

For the realisation of the automated system, (Figure 4.5) software is required, communicating with the devices and the BioCatHub-platform, processing data and handling the input from experimenters. For this purpose, a software platform was developed, named the automated biocatalysis and benchtop NMR application (*ABBA*) based on a Python Flask-restx RESTful API (*ABBA: Control center*) and an Angular-based frontend (*ABBA: Frontend*). The *ABBA: Control center* was developed and tasked with communicating with the devices and processing data. Interactions of experimenters with the platform were re-

alised in a browser-based user interface as an Angular web application (*ABBA: Frontend*).

4.4.1 ABBA: Control center

Figure 4.5 shows the individual tasks of the *ABBA:Control center*. It has to control the measurement pump, trigger the NMR measurements and administers the dosing pumps if the measured signal of the NMR crosses a certain threshold. This requires two different communication technologies. On the one hand, it must communicate with the NMR over a TCP/IP connection. On the other hand, the communication with the dosage and measurement pumps needs to be mediated over Arduino microcontrollers (chapter 4.3).

As mentioned, the *ABBA: control center* was developed as Python-Flask-restx RESTful API. The development was performed analogue to the RetroBioHub extension (chapter 3.3.2) as well as the microservices for the BioCatHub-platform (chapter 2.2.3). To communicate with the NMR or the Arduino microcontrollers, individual paths are available in the *ABBA: control center* and listed in Table 7.8.

4.4.1.1 Implementation of the concentration measurement

The control of the NMR-spectrometer is performed by sending XML-based messages (Table 7.9) to a TCP-server, supplied by the software provided by the manufacturer of the Benchtop NMR (Spinsolve 2.1.5) (chapter 4.3.1). The connection is realised as recommended by the manufacturer, using the Python socket package [139]. The measurement process is described in detail in chapter 4.3.1. Once a measurement is finished, the integral values are imported by the Pandas package [113] and the concentrations of the measured compounds are calculated as described in [141] (equation 4.1).

$$c_{analyte} = \frac{I_{analyte}}{I_{standard}} * \frac{N_{standard}}{N_{analyte}} * c_{standard} \tag{4.1}$$

$c : concentration$

$I : integralvalue$

$N : numberofprotons$

4.4.1.2 Implementation of the liquid transport

Three pumps and one LED strip must be controlled during the automated LightCas process. Two Arduino Microcontrollers administer the devices: Arduino 1, controlling the LED strip and the measurement pump and Arduino 2, controlling the two dosing pumps for the enzyme and cosubstrate required for the second reaction step. The communication between the *ABBA: control center* and the Arduino microcontrollers is enabled by the PyFirmata package [84]. For this purpose, a three-step process has to be realised and visualised based on the example of the digital output in Figure 4.6. First, the connection between the microcontroller and the *ABBA: control center* has to be established. Second, the respective GPIO pin must be selected to which the pump or LED strip is connected (chapter 1.7.2). Third, the GPIO pin is set to the desired mode (digital or a PWM signal). The LED strip and the dosage pumps are controlled by a digital signal. The pumping speed of the measurement pump is set by a PWM signal, whereas the pumping direction of the measurement pump is defined by a digital signal.

Start dosage pump

```
def start_pump_3(self):
    # connect to Arduino board
    board = pyfirmata.Arduino("COM4")
    # define pin to be used
    pump = board.get_pin('d:10:o')
    # set the pin on
    pump.write(1)
    # disconnect from board
    board.exit()
```

Stop dosage pump

```
def start_pump_3(self):
    # connect to Arduino board
    board = pyfirmata.Arduino("COM4")
    # define pin to be used
    pump = board.get_pin('d:10:o')
    # set the pin off
    pump.write(0)
    # disconnect from board
    board.exit()
```

Figure 4.6: Example code representing the starting and stopping of a dosage pump.

4.4.2 ABBA: frontend

With the *ABBA: control center* in chapter 4.4.1, the devices required for the LightCas process (chapter 4.3) could be controlled. To realise the LightCas process, the logic of the automation process, the user input and the communication with the BioCatHub-platform needed to be implemented. These tasks are handled by the *ABBA: frontend* as web application in Angular.

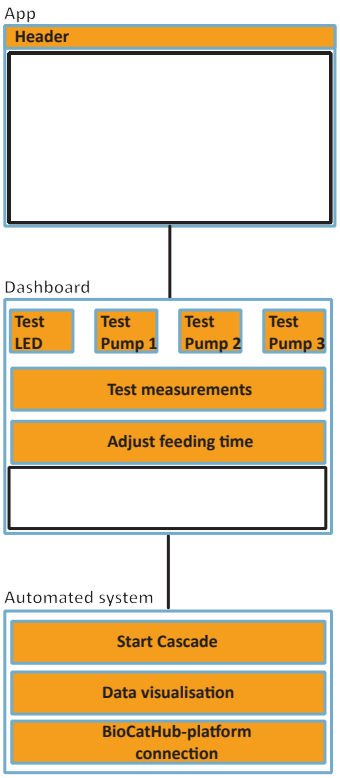

Figure 4.7: Component structure of the *ABBA: frontend* Angular web application.

The *ABBA: frontend* web application was developed analogue to the BioCatHub-platform described in chapter 2.2.2. The structure consists of three components (Figure 4.7). The app component is the root component, describes the title of the web application and implements the dashboard component. The dashboard components contains all functionalities to test and configure the LightCas project. First, four controls are available for experimenters, starting and stopping the LED stripe, the measurement pump as well as the

dosage pumps. With these, experimenters can test before the 'LigtCas' process starts if the devices can work as expected and if they can be controlled by the *ABBA: frontend*. The next control allows experimenters to perform a test measurement at the benchtop NMR. Experimenters can perform test measurements before the actual LightCas process starts. At last, the feeding time for the dosage pumps needs to be set before the LightCas process can start, which was also implemented in the *Dashboard* component.

The last component implemented is the *automated-system* component. This contains two control structures and one visualisation structure. The first control structure starts the LightCas process as described in chapter 4.5 and contains the checkbox to be activated once the enzyme for the first reaction step is added (chapter 4.5). For data visualisation, a plot is displayed showing the concentrations of 3-OH-BA and 3-OH-PAC during the reaction. Another visualisation shows the same concentrations as JSON object. This was implemented as a backup if an error with the plot visualisation occurs. The last control structure is the BioCatHub-platform connection. This control sends the collected data during the experiment to the BioCatHub-platform and allows automated and standardised data acquisition after the experiment has stopped.

4.4.2.1 Automated primary data acquisition and metadata annotation with the BioCatHub-platform

The data submission process is developed similarly to the RetroBioHub extension in chapter 3.4. For this purpose, a new path was created in the *bch-retrobiohub* microservice (*/retrobiohub/ligtCas*). Since the LightCas process is developed for exactly one reaction, the metadata for this reaction could already be predefined and described in an EnzymeML document, analogue to scenarios 1,2 and 3 (chapter 2.3.1, 2.3.2, 2.3.3). The BioCatHub-object was extracted manually from the EnzymeML-document and added to the *ABBA: frontend* Angular project. To submit the data after the lightCas process from the *ABBA: frontend* to the BioCatHub-platform, collected primary data are attached to the previously stored BioCatHub-object, based on the BioCatHub data model (chapter 2.2.1). Subsequently,

an HTTP request is made together with the BioCatHub-object to the newly defined path in the *bch-retrobiohub* microservice. There, the BioCatHub object is extracted and stored in the MiniMono database (chapter 3.4) and the MiniMongoID of the newly created database entry sent as response to the HTTP request. The MiniMongoID is then used analogue as in the RetroBioHub extension to navigate to the BioCatHub platform and display the data previously stored in the MiniMongo database (chapter 3.4). Once the automated data submission process is performed between the *ABBA: frontend* and the BioCatHub-platform, the extracted BioCatHub-object is annotated with the measured data and submitted to the BioCatHub-platform.

4.5 The LightCas process

For the realisation of the automated LightCas process, the developed software (chapter 4.4) and hardware (chapter 4.3) infrastructures were combined. The goal of the process is to perform the reaction from 3-OH-BA to metaraminol in an automated enzymatic two-step cascade in the form of an automated procedure. Therefore, the concentrations of the substrates for the first and second reaction step, 3-OH-BA and 3-OH-PAC, are measured to control the reaction process. Automated in this process means that the second reaction step of the cascade should start after the completion of the first step and the enzymes should be inactivated after completion of their corresponding step.

The LightCas process starts with two additional manual interactions to be made by the experimenter. Once the 'start cascade' button (Figure 7.2) is clicked, the process starts with a measurement to determine the concentrations of 3-OH-BA (starting substrate) and 3-OH-PAC (intermediate) in the initial reaction solution. This step is essential, as it allows the experimenter to check the reaction solution's composition before the automated process begins. For every NMR measurement, the glass capillary of the NMR spectrometer is filled by the measurement pump. Then the measurement proceeds as described in chapter 4.3.1, and the selected concentrations of substrates, intermediates and products are calculated as described in chapter 4.4.1.1. The measurement pump empties the glass

capillary and transports the reaction solution back to the reactor. After this first measure-

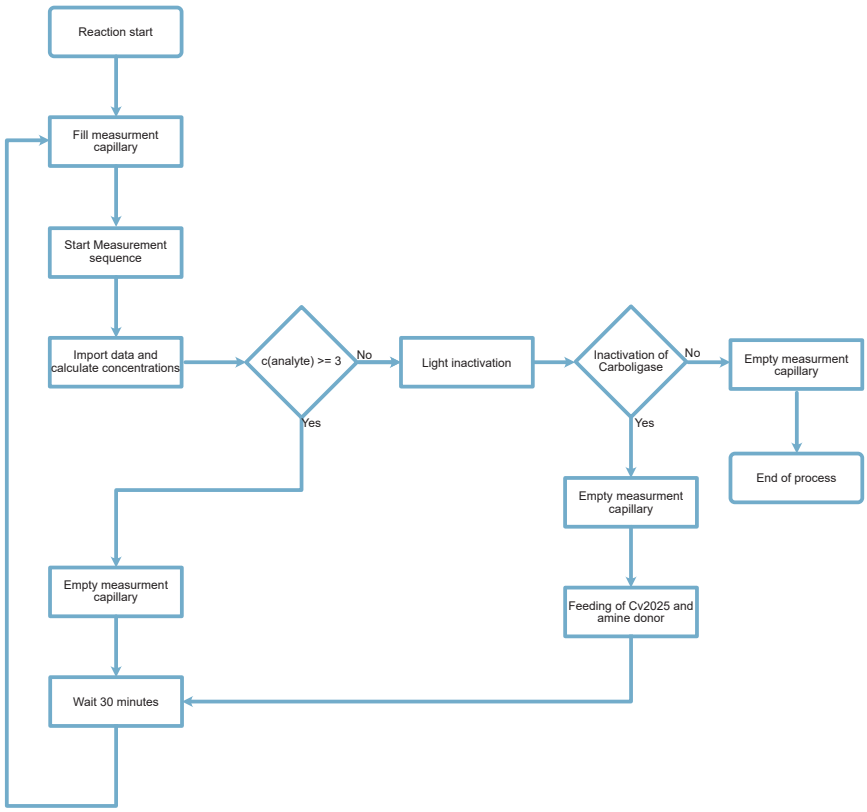

Figure 4.8: LightCas process visualised as process diagram.

ment, a two-minute period is available, enabling the second manual interaction, in which the *Ap*PDC enzyme is added manually to start the biocatalytic reaction of the first step. This has to be confirmed by the experimenter by clicking the 'enzyme added' checkbox (Figure 7.2). Once the 2-minute time has passed, a second measurement is performed to determine if adding the enzyme causes unwanted influences on the measurements.
After this, the period between measurements is set to 30 minutes, starting the automated process (Figure 4.8). Subsequently, the substrate concentrations are calculated after each measurement and used for the reaction control. The concentration of 3-OH-BA is observed for the first reaction step. The second reaction step is introduced if the 3-OH-BA concentration is below 3 mmol/L. When 3 mmol/L are undercut, the inactivation starts and the LED strip is switched on for 60 minutes. Afterwards, the *Cv*2025 and the cosubstrate α-methyl benzylamine are automatically added by the dosing pumps (chapter 4.3.2) to the reaction

vessel. For the second reaction step, the concentration of the intermediate 3-OH-PAC is monitored and the *Cv*2025 inactivated in the same way as in the first reaction step, once a concentration below a value of 3 mmol/L was measured, followed as in the first step by a 60-minute light-inactivation. With the completion of the second reaction step, the automated LightCas process was finished, too. As this was the case, no more measurements were performed, and the reaction solution can be applied for further applications.

4.5.1 The LigtCas process in a test environment

The LightCas process was performed multiple times in a testing environment. In this test environment, it should be reviewed if the process described in detail in chapter 4.5 is taking place as desired. In detail, it was aimed to test if the light-inactivation takes place after the first and second reaction step and if dosage pumps perform as required to feed the enzyme and cosubstrate for the second reaction step, once the light-inactivation for the first reaction step was finished. Furthermore, it was targeted to review if measurement with the benchtop NMR and the following concentration determination is performed correctly. For this purpose, two test samples were created ('Sample a' and 'Sample b'). Sample a (Table 4.3) represented the initial reaction state and contained the reference substance. Sample b (Table 4.4) was a reaction mixture taken after the second reaction step was completed in another reaction setup.
In this sample, the reference substance, metaraminol and small fractions of 3-OH-PAC were present but below a concentration of 3 mmol/L (Sample b). In the test environment, sample a was measured during the first two test measurements before the automated system started (described above). Then the sample was exchanged with Sample b. After the subsequent measurement, as expected, the LED strip lighted up, and the dosage pumps started. With the next measurement, due to the missing 3-OH-PAC, the completion of the second reaction was simulated, and the LED strip lighted up again, finishing the LightCas process. Carrying out the LightCas process in the test environment showed that all technical implementation were realised successfully as specified during the requirements engineering (chapter 4.2). The measurement of the targeted reactants 3-OH-BA and 3-

Table 4.3: Composition of Sample a. HEPES: 4-(2-hydroxyethyl)-1-piperazineethanesulfonic acid; ThDP: thiamine pyrophosphate; $MgSO_4$: magnesium sulfate; TMSP: Trimethylsilylpropanoic acid.

Compound	**Concentration**
HEPES pH 7.5	100 mmol/L
3-OH-BA	15 mmol/L
Sodium pyruvate	200 mmol/L
ThDP	0.1 mmol/L
MgSO4	2.5 mmol/L
TMSP	5 mmol/L
ApPDC-SOPP3	1.5 mg/mL

OH-BA and the concentration determination were performed and the calculated concentration values resulted in those as they were prepared in the samples previously (Sample a, Sample b). The light-inactivation was performed in reaction step one and two, once the resepective target concentration undercut the previously defined values for the respective reaction step (3-OH-BA and 3-OH-BA <3 mmol/L). The automated feeding of the enzyme and cosubstrate after the first reaction step was tested successfully, too.

Table 4.4: Composition of Sample b. HEPES: 4-(2-hydroxyethyl)-1-piperazineethanesulfonic acid; ThDP: thiamine pyrophosphate; $MgSO_4$: magnesium sulfate; α-MBA:α-methylbenzylamine; PLP: pyridoxal phosphate; TMSP: Trimethylsilylpropanoic acid

Compound	**Concentration**
HEPES pH 7.5	100 mmol/L
3-OH-BA	<3 mmol/L
3-OH-PAC	<3 mmol/L
Sodium pyruvate	<187 mmol/L
a-MBA	<8 mmol/L
PLP	0.3 mmol/L
ThDP	0.1 mmol/L
MgSO4	2.5 mmol/L
TMSP	5 mmol/L
ApPDC-SOPP3	1.5 mg/mL (inactivated)
Cv2025-SOPP3	1.5 mg/mL (inactivated)

4.6 Automated data collection from the LightCas process

With the realisation of the LightCas process in chapter 4.5, the technical requirements described during the requirements engineering in chapter 4.2 were realised. The last task to be realised is automated data collection. As described before, the functionalities for this are implemented in the *ABBA: frontend*.

Once the LightCas process is finished, the experimenter can see the collected data as a plot or JSON object representation (Figure 7.2). To submit the data to the BioCatHub-platform, the respective button has to be clicked (Figure 7.2). Subsequently, a new tab or window opens in the browser displaying the BioCatHub-platform with data collected during the LightCas process.

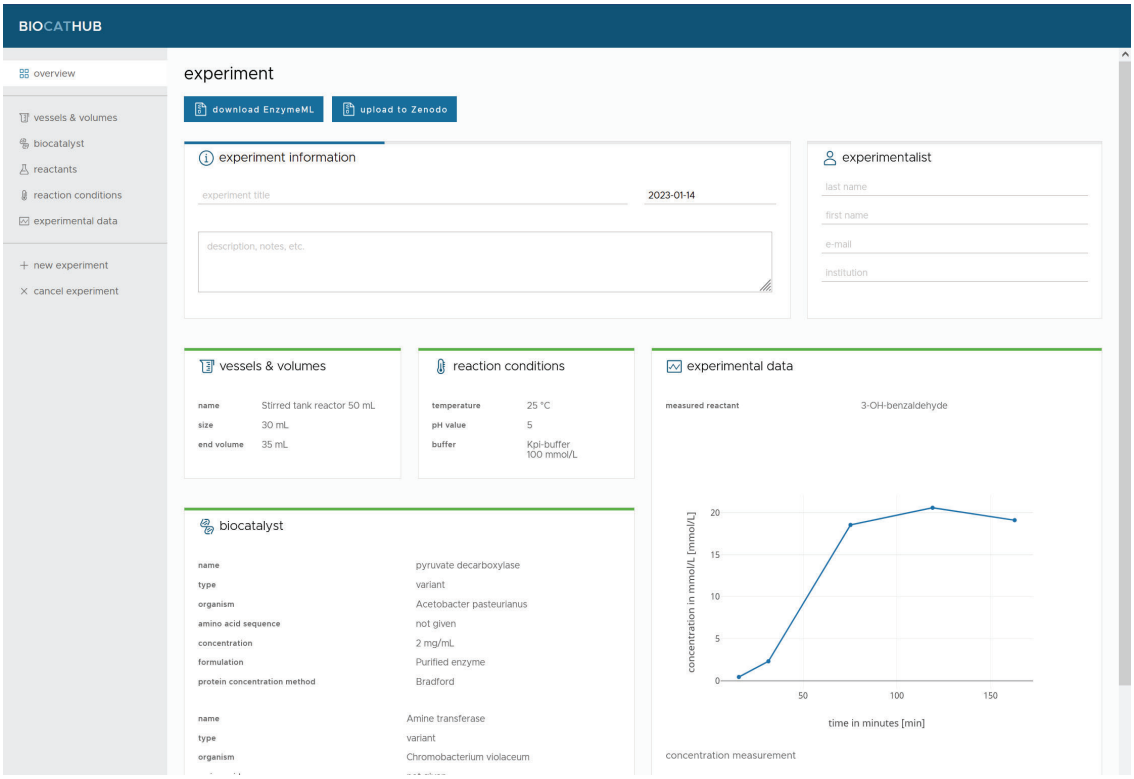

Figure 4.9: Screenshot of the BioCatHub-platform after the automated data collection process of *ABBA:frontend*.

4.7 Experiments performed with the LightCas process

Two experiments were performed in the described LightCas-platform, aiming to test the hardware infrastructure, ABBA, and the biocatalytic reaction. For this purpose, the reagents and the enzymes were prepared as described in Table 4.3, and the reactions done by the experimenters as described in chapter 4.5.

The results are displayed in Figure 4.10. The first experiment was performed over a total time of 20 hours and 41 minutes. The resulting graph (Figure 4.10a) shows the time-dependent concentration courses of 3-OH-BA and 3-OH-PAC. As expected, in an optimal scenario the initial concentration of 15 mmol/L of 3-OH-BA decreases the same way as the

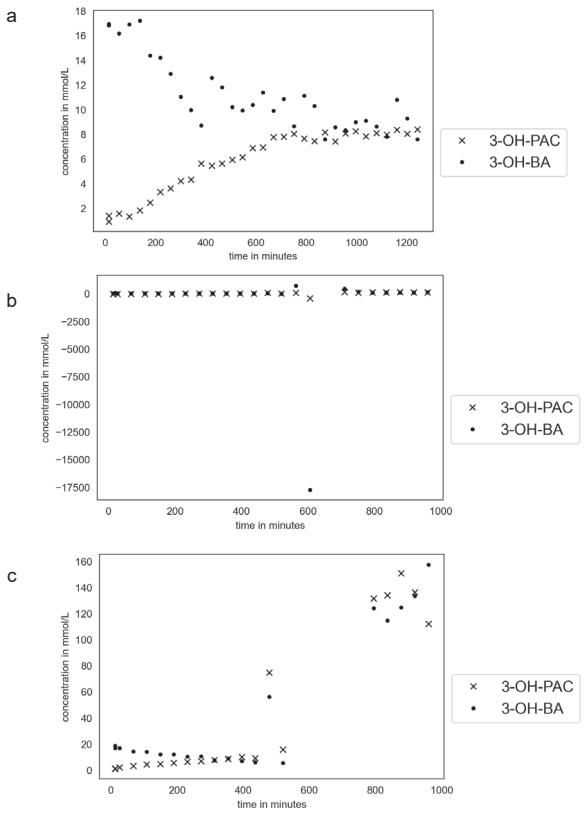

Figure 4.10: Automated LightCas experiments. The graphs show the time-dependent concentration courses of 3-OH-BA and 3-OH-PAC. A shows the first experiment, and b and c the second experiment. C shows the same time-dependent concentration measurements as b, with four measurements between 9 and 12 h deleted. The reactions were performed at 22 °C in a volume of 30 mL. The reaction solution consisted of 100 mmol/L HEPES pH 7.5, 15 mmol/L 3-OH-BA, 200 mmol/L sodium pyruvate, 0.1 mmol/L ThDP, 2.5 mmol/L $MgSO_4$, 5 mmol/L TMSP, 1.5 mg/mL *Ap*PDC-SOPP3.

concentration of 3-OH-PAC increases. After 20 h, the concentration of 3-OH-BA was measured at 7.6 mmol/L and 3-OH-PAC at 8.4 mmol/L. Since a concentration of <3 mmol/L of 3-OH-BA is required to initialise the second reaction step, neither the inactivation with the LED strip nor the induction of the second reaction step was performed. Thus, the experiment was stopped manually. Nevertheless, from a technical point of view, the time-dependent concentration courses of 3-OH-BA and 3-OH-PAC were automated and monitored successfully over 20 hours.

The second experiment (Figure 4.10b) was performed for 16 h, with the reaction setup

identical to the first reaction. In this experiment, the time-dependent reaction courses of 3-OH-BA and 3-OH-PAC show a different result than in the first experiment. During the first 7 hours, the time-dependent concentration courses of 3-OH-BA and 3-OH-PAC show a similar course (Figure 4.11) than in experiment 1. Afterwards, the concentration values reach between -17734.8 mmol/L and 453 mmol/L for 3-OH-BA and -399.63 mmol/L and 157.11 mmol/L for 3-OH-PAC. Since the initial concentration of 3-OH-BA was 15 mmol/L, there is no explanation for how the measured substrate concentrations could fluctuate between -17734.8 mmol/L and 453 mmol/L. The measured product concentrations were also not explainable by an initial substrate concentration of 15 mmol/L. Since one molecule of 3-OH-BA yields one molecule of 3-OH-PAC, the highest theoretical 3-OH-PAC concentration would be 15 mmol/L. Moreover, once the substrate is used up at a concentration of 0 mmol/L, no further reaction can theoretically occur, making negative concentrations not explainable. The most probable explanation is an error in the measurements. It was observed that if the benchtop NMR measures air instead of the reaction solution, values as the just described concentrations were observed. The reason why air could have been measured is that the *Ap*PDC-SOPP3 denatured and blocked tubing of the measurement pump, preventing the reaction solution from entering the capillary of the Benchtop NMR. The experimenters reported a white precipitate upon examining the automation platform after 16 hours. Therefore, the NMR likely did not measure the reaction solution but empty tubing, causing the offset in the measured concentrations.

The precipitation of the *Ap*PDC-SOPP3 might cause a problem for the LightCas process. The attachment of a filter to the tubing of the measurement pump could help to prevent the blockage in the tubing. Thereby, the entry of already denatured protein from the reaction vessel into the tubing of the measurement pump could be avoided, preventing a blockage in the tube. Nevertheless, in experiment 1 it was possible to monitor the reaction for 20 hours without blocking the tubing of the measurement pump. Furthermore, the continuous measurement with solubilised enzyme was successfully reported elsewhere, too [80].

In summary, the LightCas process was realised with hardware and software components,

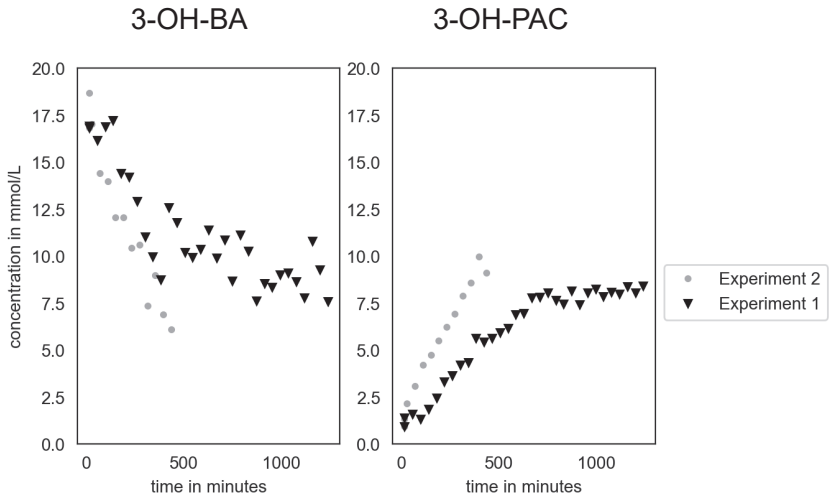

Figure 4.11: Measured time dependent concentration values of 3-OH-BA and 3-OH-PAC. 'Experiment 1' shows the concentration values of 3-OH-BA over 20 h. 'Experiment 2' also shows the 3-OH-PAC concentrations but only about 8 h. The reactions were performed at 22 °C in a volume of 30 mL. The reaction solution consisted of 100 mmol/L HEPES pH 7.5, 15 mmol/L 3-OH-BA, 200 mmol/L sodium pyruvate, 0.1 mmol/L ThDP, 2.5 mmol/L $MgSO_4$, 5 mmol/L TMSP, 1.5 mg/mL *Ap*PDC-SOPP3.

as defined during the requirements engineering (chapter 4.2). Furthermore, the automated inactivation and the induction of the second reaction step were successfully verified in a test environment. The automated reaction monitoring using a benchtop NMR was performed in two experiments. Here, optimisation is currently under construction to run both steps to completion. Finally, the automated data acquisition using the BioCatHub-platform could also be successfully implemented.

4.8 LightCas implementation conclusion

The aims for the LightCas implementation were the development of a biocatalytic platform in which the two-step cascade from 3-OH-BA to metaraminol can be performed in an automated process as described in chapter 4.1. In this cascade, the product formation of metaraminol (Figure 4.12a) is targeted to be performed in two reaction steps, with an automated light-induced inactivation of the enzyme after the first and second reac-

tion step. Furthermore, the automated feed of enzyme (*Cv*2025-SOPP3) and cosubstrate (α-MBA) for the second reaction step is realised, once the first reaction step is fully completed. Furthermore, regarding the BioCatHub project, an automated data acquisition strategy should be implemented to collect data directly from the experimental procedure in the laboratory. Both goals were successfully realised.

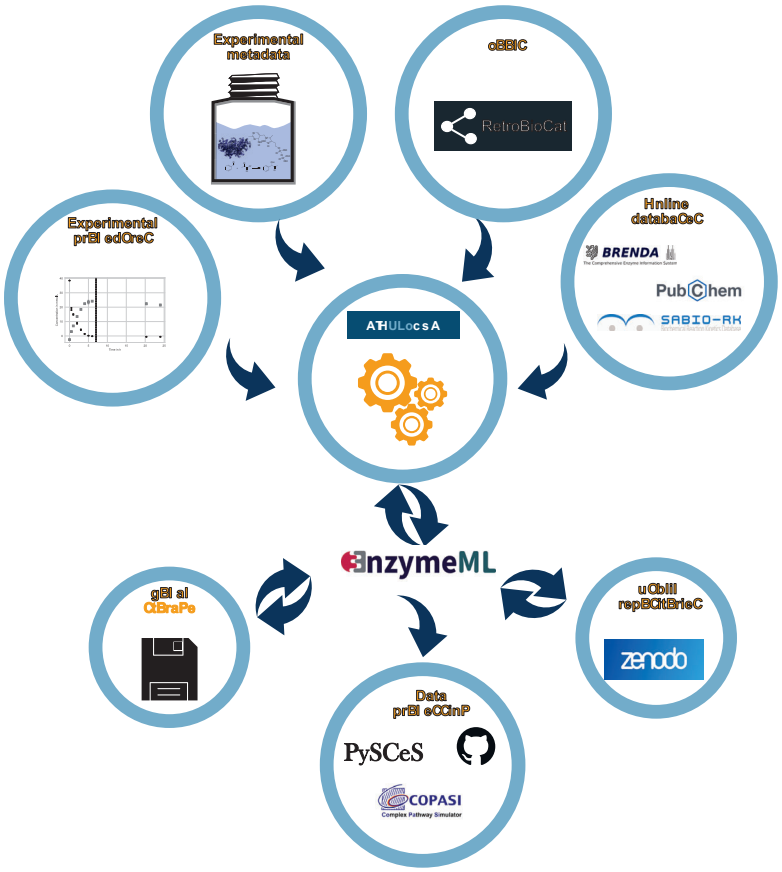

Figure 4.12: Overview of the data acquisition, storage, processing and publication possibilities in the BioCatHub project. Included databases are BRENDA [32], SabioRK [30, 31], Zenodo [36] and PubChem [29]. As file format EnzymeML is used [49]. As tools, COPASI [52] and PySCeS [53] and RetroBioCat [132] are shown.

With the *ABBA*, a software platform as central element, in total four different devices can be controlled, taking care of the enzyme inactivation, transport of the reaction solution for measurements to the benchtop NMR, as well as the inducing of the second reaction

step by feeding the respective enzyme and cosubstrate. Furthermore, with the control of a benchtop NMR, the reaction progress is monitored and controlled by *ABBA*. Finally, with the automated submission of data from *ABBA* to the BioCatHub-platform, an automated 'one click' data collection possibility is realised.

With the implementation of *ABBA*, it was possible to organise the complex control of three peristaltic pumps, one LED-strip and a benchtop NMR in one web service and provide a user interface for experimenters in the laboratory. If more lab devices are aimed to be integrated, it is further possible to develop standardised interfaces for the Arduino or TCP/IP controlled devices. With Stadardization in Lab Automation (SiLA), it is possible to develop standardised interfaces the software-mediated device communication [142, 143]. By this, the implementation of further lab devices in, for example, *ABBA* could be simplified even further. In this case, only the SiLA interface would need to be implemented and not the individual Arduino or TCP/IP interfaces, simplifying the software architecture in *ABBA* and making it more comprehensible. In addition, the software architecture of *ABBA* was already used before to develop the BioCatHub-platform (Angular web application as frontend, Python-flask-restx web service as backend).
The LightCas implementation not only improves the documentation of experimental data in biocatalysis but also generates a template workflow for the further implementation of devices in the laboratory. Using Arduino microcontrollers, it was possible to control pumps (measurement pump and dosing pumps) and a LED stripe. Using a TCP/IP interface, the benchtop NMR could be controlled. For future projects, lab devices like magnetic stirrers or heating units could be controlled as described here, using Arduino microcontrollers, implementing them based on the architecture applied here. For other measurement or other lab devices controllable over a TCP/IP interface, the same logic as with the benchtop NMR can be used.
Therefore, it was possible to concentrate more on the implementation and optimisation of the hardware control and the biocatalytic process development, since a blueprint for the software architecture was already present from the development of the BioCatHub-

platform.

Since the data transmission process from *ABBA* to the BioCatHub-platform is realised with one HTTP request to the *bch-retrobiohub* RESTful API, it was easily implemented in *ABBA*. As with the software architecture of *ABBA*, with the extension of the *bch-retrobiohub* microservice it was possible to reuse an already existing infrastructure of the BioCatHub project. In this workflow, the BioCatHub-object was submitted to the *bch-retrobiohub* microservice simplifying the data transfer.

With the realisation of the LightCas implementation, the last targeted overall goal for the BioCatHub project was realised. It allows experimenters to perform the biocatalytic reaction from 3-OH-BA to metaraminol in an automated setup and also collects the produced primary data with previously collected metadata seamlessly with one click by a connection to the BioCatHub-platform (Figure 4.13).

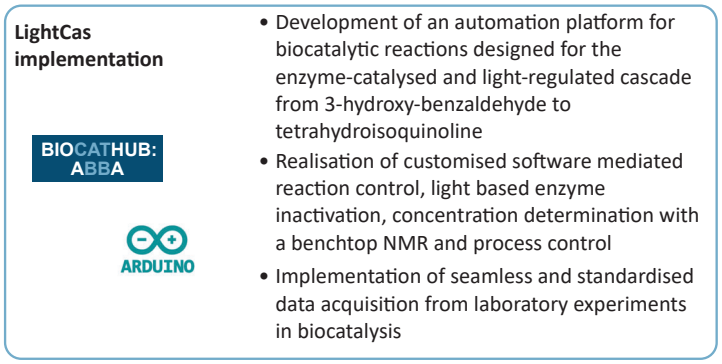

Figure 4.13: Conclusions from the LightCas implementation

5 Conclusion

In this dissertation, it was aimed to create a technical infrastructure for the data acquisition, storage and publication of experimental data from biocatalytic experiments based on the FAIR data principles to enhance the reproducibility and meet the demanding requirements that are specified by, for example, funding agencies like the DFG or the ERC [40, 41]. Moreover, besides the functional requirements, the technical realisation should be developed in an extendable, easily maintainable infrastructure associated with low costs. With the BioCatHub project, all these requirements could be fulfilled and even surpassed.

At the center of the BioCatHub project is the BioCatHub-platform. This web application was developed to realise the experimental data acquisition, storage and publication in biocatalysis based on the FAIR data principles and is intuitive to use for experimenters in biocatalysis. The intuitive usage of the BioCatHub-platform was achieved by designing the user interface in a wizard-like structure based on experimental batch procedures in biocatalysis, supported by database interfaces enabling the automated data integration from the public databases BRENDA and SabioRK. In the BioCatHub-platform, collected primary and metadata are exportable as EnzymeML documents, storing the exported data in a structured, machine-readable format which is also a commonly used format in biocatalysis and enzymology. Furthermore, it is possible to upload and later publish collected primary and metadata in an EnzymeML document on Zenodo, unambiguously annotating the collected data with a DOI once the data is published.
The software architecture was realised with an Angular web application as frontend and a microservice-based backend developed as Python-flask-restx RESTful APIs, creating a

comprehensible, interoperable and extendable software architecture. For deployment and hosting purposes, the web application and the RESTful APIs were deployed as Docker containers and then hosted in a Kubernetes cluster, creating a clear, extendable and well-administrable structure.

To enhance the implementation of the BioCatHub-platform in the daily work life of experimenters in biocatalysis, the *bch-data-processing* repository was developed as experimental data visualisation and processing repository to utilise the collected data from the BioCatHub-platform. The bch-data-processing repository applies the EnzymeML documents, exported from the BioCatHub-platform as input and allows comprehensible data processing using Jupyter-Notebooks.

With the RetroBioHub extension, the infrastructure of the BioCatHub-platform could be extended, providing experimenters with metadata about retrosynthetic pathways available on RetroBioCat. Scientists can explore the retrosynthetic pathways in RetoBioCat, and seamlessly submit metadata about the enzymes and reactants of the desired retrosynthetic pathway with only two mouse clicks to the BioCatHub-platform. By this, the documentation of reactions, reactants, and enzymes is simplified, and errors during the manual transfer from metadata can be avoided.

With the LightCas implementation, it was exemplarily shown how to realise seamless data acquisition directly from an experimental procedure in biocatalysis. For this purpose, an automation platform was developed to realise the enzyme-catalysed, light-regulated two-step cascade from 3-hydroxy-benzaldehyde to metaraminol in an automated way. Therefore, in total three pumps, one LED strip and one benchtop NMR were controlled by the automated biocatalysis and benchtop NMR application (*ABBA*), which was developed as part of the BioCatHub project. With an automated experiment finished, it is possible to submit the primary data, measured during the experiment by the benchtop NMR and a set of predefined metadata with one click to the BioCatHub-platform, realising the primary and metadata acquisition directly from the laboratory.

At the start, the BioCatHub project aimed to develop a platform for research data management in biocatalysis based on the FAIR data principles to improve the reproducibility of

experimental results. Nevertheless, with an intuitive user interface for experimenters in biocatalysis, extendable software development, deployment and hosting architecture and the application of the EnzymeML format, the BioCatHub project became more than the sum of its parts. The extendable software architecture enables with the RetroBioHub extension, the integration of metadata from the online available tool RetroBioCat and with the LightCas implementation it is possible to integrate data directly from experimental procedures. Moreover, besides the integration, with the RetroBioHub extension and the 'LightCas' implementation, template workflows could be developed, enabling a simple and comprehensible software solution for data integration to the BioCatHub-platform from other web applications or software used in the laboratory. Finally, with the *bch-data-processing* repository, experimenters have the possibility to seamlessly visualise or process the previously collected data in the BioCatHub-platform directly or at a later point in time.

6 Outlook

The BioCatHub project realises a research data management platform for experimental data in biocatalysis focusing on the collection, storage, publishing, sharing and processing of data from biocatalytic experiments. At the center of the BioCatHub project is the BioCatHub-platform, which enables the FAIR data principles based collection, storage, publishing and sharing of experimental data in biocatalysis. Now, a first version of the BioCatHub-platform is established. It focuses on batch experiments of single transformations. On the long run, the application should be extended, three possibilities were therefore identified to improve the BioCatHub-platform. First, with the current version it is possible to retrieve metadata describing reactants over the database query to SabioRK or add them manually to the user interface. However, scientists in biocatalysis are often used to identify their reactants by the molecular three-dimensional structure of the substances, which is drawn by the experimenters. This is currently not possible in the BioCatHub-platform. Nevertheless, due to the extendable Angular-based architecture of the BioCatHub-platform, an interface can be implemented, in the frontend for example, the 'ketcher' JavaScript package [144], allowing the experimenters to draw their reactants with subsequent and automated extraction of the SMILES code from the drawn structure. Second, the current version of the BioCatHub-platform allows the import and export of one EnzymeML document. Since the BioCatHub-platform was developed to describe batch experiments in biocatalysis, one EnzymeML document was sufficient for this scope. If multiple EnzymeML documents could be imported and exported, experimenters could upload and compare numerous experimental data stored in EnzymeML documents into the BioCatHub-platform and collect primary and metadata for multiple experimental pro-

cedures with one usage of the BioCatHub-platform. Third, concerning data publication, experimenters currently have the possibility to upload and publish experimental data to one Zenodo account, which is administered as part of the BioCatHub project. An interesting improvement would be to enable experimenters to connect their own Zenodo accounts with the BioCatHub-platform. By this, experimenters could control the upload and publication of EnzymeML documents by themselves. However, it should not go unmentioned that a connection of individual Zenodo accounts to the BioCatHub-platform requires careful conceptualisation, development and testing. Once this feature is implemented, the access tokens required for authentification on Zenodo of the connected Zenodo accounts are collected, stored and exchanged between the frontend and backend of the BioCatHub-platform. This has the potential to create major safety issues because if unauthorised persons access these tokens, they would have control over the Zenodo account and could upload, publish, alter or delete resources stored in the Zenodo repository without the permission of the repository owner.

With the *BioCatHub-data-processing* repository, an easy-to-use and comprehensible data processing and visualisation approach is available for experimenters in biocatalysis using the BioCatHub-platform. Moreover, once the experimental documentation is uploaded to a publicly available repository, the in-detail descriptions of the data processing workflows of experimenters are available. Publicly available and comprehensible data processing workflows could be investigated to identify data processing challenges and develop customised possibilities for these challenges. One possible example is shown in scenario 2 (chapter 2.3.2). The volume loss due to the evaporation of water or other solvents of biocatalytic reactions in open reaction vessels can create problems in calculating the exact concentration of the reactants involved in the reaction. With the approach in scenario 2, the concentrations of the reactants could be corrected, enabling a comparison of different reaction vessels with also different water evaporation rates. With more of these workflows available, it would even be possible to implement well-tested data processing workflows directly into the BioCatHub-platform.

The RetroBioHub extension is a highly valuable extension to the BioCatHub-platform, enabling the automated integration of enzyme-catalysed retrosynthetic pathways available in RetroBioCat. Still, it is in the current version only possible to retrieve data from RetroBioCat. Therefore, once collected, it would be interesting to submit data from the BioCatHub-platform to RetroBioCat. A microservice for this task is currently in the planning phase. One major challenge in this development is that experimental data, which is submitted from the BioCatHub-platform to RetroBioCat, has to be supported by a peer review process, which is required by RetroBioCat. Here, an easily applicable test would be required, making it possible to control if the submitted data is backed up by a peer-review process. A possible solution could be that every set of primary and metadata collected in the BioCatHub-platform and planned to be submitted to RetroBioCat, has to be annotated with the respective DOI of the published peer-reviewed article. This would make it possible to easily review the submitted data from the BioCatHub-platform based on the published article connected over the DOI.

With the LightCas implementation, it was possible to implement seamless data integration from experiments in the laboratory. The development of the LightCas implementation provides a blueprint for data integration from other experimental platforms or devices. Therefore, interfaces for other measurement devices could be created for the automated data integration to the BioCatHub-platform.

As already described, the BioCatHub-platform was developed to collect experimental data from batch experiments in biocatalysis. However, biocatalytic experiments are not solely performed as batch experiments but can also be carried out in a large number of other processes, for example, in a continuous mode or as fed-batch approach. Still, the description of these processes requires a more complex description with metadata. Currently, there is no sufficient metadata standard available to realise this. At the moment, this target is aimed for by an international consortium of scientists in biocatalysis, software engineering and enzymology, which formed around the STRENDA commission to define metadata besides the existing STRENDA recommendations describing in particular biocatalytic experiments.

7 Appendix

STRENDA metadata comparison

Table 7.1: STRENDA guidelines 1A [26]: description of the enzyme identity. The identity of the enzyme is defined by the name of the biocatalyst, the EC number and the reaction. Details such as the Isoenzyme, Tissue, Organelle, Localisation and Post translation modification were not implemented since this metadata is often not available in biocatalytic procedures.

STRENDA metadata	**BioCatHub data model**
Name of reaction catalyst	implemented
Fully balanced chemical reaction equation	The reaction is defined by the substrates and products in the reaction
EC number	implemented
Oligomeric state	Not implemented
Sequence or sequence accession number	implemented
Organism/species & strain	implemented
Isoenzyme	Not implemented
Tissue	Not implemented
Organelle	Not implemented
Localisation	Not implemented
Post translational modification	Not implemented

Table 7.2: STRENDA guidelines 1A [26]: description of the storage conditions. The storage conditions were not implemented in the BioCatHub data model, since the scope of the BioCatHub-platform was aimed to focus on the description of the enzyme catalysed reaction.

STRENDA recommendations	**BioCatHub-platform implementation**
Storage temperature	Not implemented
Atmosphere if not air	Not implemented
pH	Not implemented
At which temperature was the pH measured	Not implemented
pH-Buffer & concentrations (including counter ion)	Not implemented
Metal salt(s) & concentrations	Not implemented
Other components	Not implemented
Enzyme/portein concentration	Not implemented
Optional: Statement about observed loss of activity under the above conditions	Not implemented
Statement about the thawing process	Not implemented

Table 7.3: STRENDA guidelines 1A [26]: description of the assay conditions. The conditions in the BioCatHub model are described by the temperature, pH, buffer, and reactants in the reaction. The other metadata of the STRENDA recommendations regarding the assay conditions were implemented since these are often not part of experiments in biocatalysis.

STRENDA metadata	**BioCatHub data model**
Identity and purity of all assay components	Implemented
Measured reaction	Implemented
Assay temperature	Implemented
Assay pressure	Not implemented
Atmosphere if not air	Not implemented
Assay pH	Implemented
Buffer&concentrations	Implemented
Metal salts&concentrations	Not implemented
Other assay components	Not implemented
Coupled assay components	Not implemented
Substrate & concentration ranges	Not implemented
Enzyme/protein concentration	Not implemented
Varied components	Not implemented

Table 7.4: STRENDA guidelines 1A [26]: description of the activity. Biocatalytic experiments are often not analysed by calculating initial rates. Therefore, the BioCatHub data model collects the original measurement data instead of processed data.

STRENDA metadata	**BioCatHub-platform implementation**
Initial rates of the reaction measured	Not implemented
Proportianlity between initial velocity and enzyme concentration	Not implemented
Enzyme activity	Not implemented
Evidence that reaction reached equilibrium	Not implemented
State which of all reactants were measured directly	Not implemented
Range of starting material and product concentrations in the experiment	Not implemented
Complexing metal ions	Not implemented
Assay method	Not implemented
type of assay	Not implemented
Reaction stopping procedure	Not implemented
Direction of assay	Not implemented
Reactant determined	Not implemented
Free metal cation concentrations	Not implemented

STRENDA recommendations level 1 B

The STRENDA recommendations level 1 B [26] focuses on describing enzyme activity data. As described in chapter 1.1.1, biocatalytic procedures can differ. Therefore, the STRENDA recommendations level 1 B were not implemented, but the BioCatHub data model collects time-dependent concentration or absorption values of a biocatalytic reaction. The in-detail experimental analysis is then described in the BioCatHub-data-processing repository.

7.1 Appendix: Retrobiohub

Table 7.5: Requirements regarding the data models for RetroBioCat and the BioCatHub-platform.

Requirement	**Specification**
Represent data from Retrobiocat	Model visualisation for the data Retrobiocat data
Represent data for BioCatHub	Model visualisation of the BioCatHub data model, compatible to the Retrobiocat data

Table 7.6: Requirements regarding the functionalities of the microservice for the communication between RetroBioCat and the BioCatHub-platform

Requirement	**Specification**
Create a possibility for Retrobiocat and BioCatHub to communicate.	Python Flask RESTful API, allowing the http-protocol based communication.
Create instance to cache data	MongoDB instance
Implement functions allowing the exchange of data between Retrobiocat and BioCatHub	POST request endpoint, transforming Retrobiocat-object to the BioCatHub-object and storing it in the MongoDB instance
Extract the previously exchanged data and display it in on BioCatHub	POST request endpoint in bch-retrobiohub retrieving the previously stored data in the MongoDB instance.
Annotate Retrobiocat data with IUPAC name of reagents	Pubchem request to retrieve IUPAC name based on a SMILES code

Table 7.7: Add caption

Requirement	**Specification**
Enable the redirect from Retrobiocat to BioCatHub-platform	Path in the bch-frontend serving as endpoint
Submit the MiniMongoID from RetroBioCat to the BioCatHub-platform	Attach the MiniMongoID to the URL during the redirect
Retrieve the data stored in MiniMongo	Implement an HTTP-request with the submitted MiniMongoID

7.2 LightCas

Table 7.8: Paths available in the ABBA: control center microservice

Path	Task
light_up	Switches the LED strip on
light_down	Switches the LED strip off
start_pump	Starts the measurement pump
start_pump	Stops the measurement pump
stop_pump_2	Starts the enzyme dosing pump
start_pump_3	Starts the cosubstrate dosing pump
stop_pump_3	Stops the cosubstrate dosing pump
measure	Performs a NMR measurment
automated_system	Starts the autmated system
inactivate	Starts the inactivation of the enzymes
reaction_2	Starts the feeding procedure of the enzyme and cosubstrate for the second reaction step

Table 7.9: XML commands for controlling the NMR

XML	Function
<?xml version="1.0" encoding="utf-8"?> <Message><Start protocol="1D PROTON"> <Option name="Scan" value="QuickScan/> </Start></Message>	Message starting the quick scan method of the NMR-spectrometer
<?xml version="1.0" encoding="utf-8"?> <Message> <Start protocol='1D PRESAT MULTI'> <Option name='Number' value='8' /> <Option name='RepetitionTime' value='30' /> </Start> </Message>	Message starting the 1D extended + method
<?xml version="1.0" encoding="utf-8"?> <Message> <Start protocol='SHIM 1H SAMPLE'> <Option name='SampleReference' value='4.74' /> <Option name='Shim' value='QuickShim1' /> </Start> </Message>	Message starting the sample shim method of the NMR-specrometer

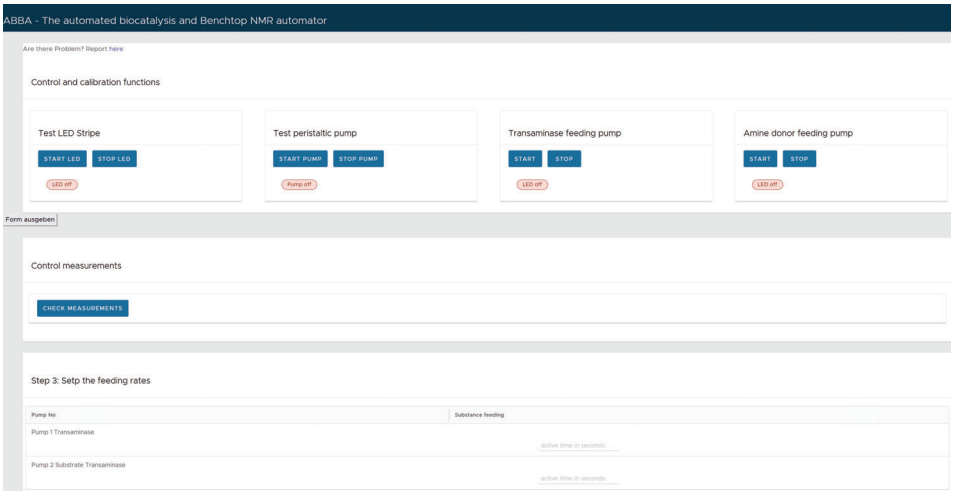

Figure 7.1: Screenshot of the *ABBA: frontend* showing the app and dashboard component

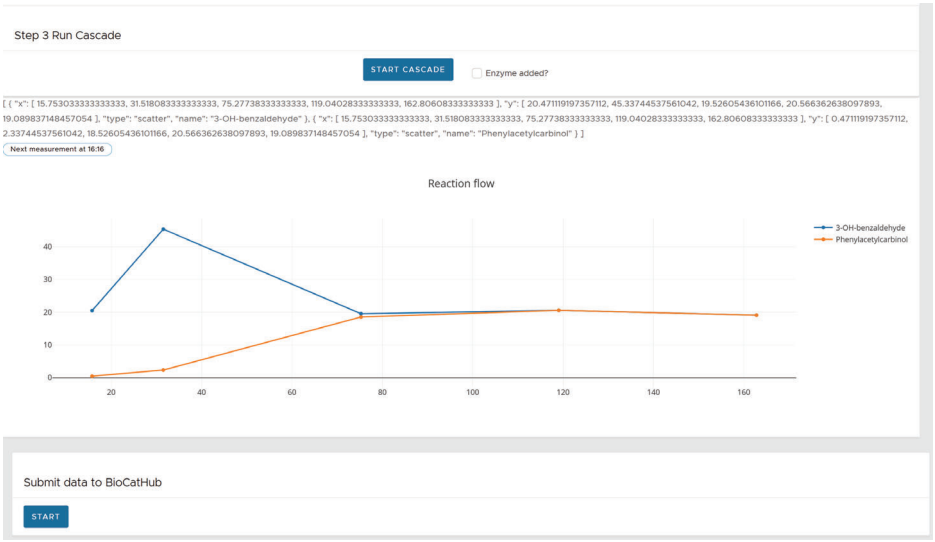

Figure 7.2: Screenshot of the *ABBA: frontend* showing the automated system component

Table 7.10: Devices used in the LightCas extension

Device	**Model**	**Manufacturer**
Peristaltic pump (feeding pump	ISMATEC Reglo Flex	ISMATEC SA, Glattbrugg, Switzerland
Peristaltic pump (measurement pump	BT100-2J	Halma, Amersham, Buckinghamshire, United Kingdom
LED strip	X105-0200	revoART GmbH Merkkleeberg, Germany
Microcontroller	Arduino nano	Arduino

Bibliography

[1] Wandrey, C., Liese, A. & Kihumbu, D. Industrial biocatalysis: Past, present, and future. *Organic Process Research & Development* **4**, 286–290 (2000).

[2] Schmid, A. *et al.* Industrial biocatalysis today and tomorrow. *Nature* **409**, 258–268 (2001). URL https://www.nature.com/articles/35051736.

[3] Bell, E. L. *et al.* Biocatalysis. *Nature Reviews Methods Primers* **1**, 1–21 (2021). URL https://www.nature.com/articles/s43586-021-00044-z.

[4] Wu, S., Snajdrova, R., Moore, J. C., Baldenius, K. & Bornscheuer, U. T. Biocatalysis: Enzymatic synthesis for industrial applications. *Angewandte Chemie International Edition* **60**, 88–119 (2021). URL https://onlinelibrary.wiley.com/doi/full/10.1002/anie.202006648.

[5] Huffman, M. A. *et al.* Design of an in vitro biocatalytic cascade for the manufacture of islatravir. *Science* (*New York, N.Y.*) **366**, 1255–1259 (2019).

[6] Kevin Mack, Moritz Doeker, Laura Grabowski, Andreas Jupke & Dörte Rother. Extractive in situ product removal for the application of naturally produced l-alanine as an amine donor in enzymatic metaraminol production. *Green Chemistry* **23**, 4892–4901 (2021). URL https://pubs.rsc.org/en/content/articlelanding/2021/GC/D1GC00852H.

[7] Erdmann, V. *et al.* Enzymatic and chemoenzymatic three-step cascades for the synthesis of stereochemically complementary trisubstituted tetrahydroisoquinolines. *Angewandte Chemie International Edition* **56**, 12503–12507 (2017).

[8] Sheldon, R. A. & Brady, D. Green chemistry, biocatalysis, and the chemical industry of the future. *ChemSusChem* **15**, e202102628 (2022). URL `https://chemistry-europe.onlinelibrary.wiley.com/doi/10.1002/cssc.202102628`.

[9] Frazzetto, G. White biotechnology. *EMBO reports* **4**, 835–837 (2003).

[10] Garske, A. L., Kapp, G. & McAuliffe, J. C. Industrial enzymes and biocatalysis. In Kent, J. A., Bommaraju, T. V. & Barnicki, S. D. (eds.) *Handbook of Industrial Chemistry and Biotechnology*, 1571–1638 (Springer International Publishing, Cham, 2017).

[11] Albarrán-Velo, J., González-Martínez, D. & Gotor-Fernández, V. Stereoselective biocatalysis: A mature technology for the asymmetric synthesis of pharmaceutical building blocks. *Biocatalysis and Biotransformation* **36**, 102–130 (2018).

[12] Siedentop, R., Claaßen, C., Rother, D., Lütz, S. & Rosenthal, K. Getting the most out of enzyme cascades: Strategies to optimize in vitro multi-enzymatic reactions. *Catalysts* **11**, 1183 (2021).

[13] Sheldon, R. A. & Woodley, J. M. Role of biocatalysis in sustainable chemistry. *Chemical reviews* **118**, 801–838 (2018).

[14] Tufvesson, P., Lima-Ramos, J., Nordblad, M. & Woodley, J. M. Guidelines and cost analysis for catalyst production in biocatalytic processes. *Organic Process Research & Development* **15**, 266–274 (2011).

[15] Rossino, G. *et al.* Biocatalysis: A smart and green tool for the preparation of chiral drugs. *Chirality* **34**, 1403–1418 (2022).

[16] van Schie, M. M. C. H., Spöring, J.-D., Bocola, M., Domínguez de María, P. & Rother, D. Applied biocatalysis beyond just buffers - from aqueous to unconventional media. options and guidelines. *Green Chemistry* **23**, 3191–3206 (2021).

[17] Agostini, F. *et al.* Biocatalysis with unnatural amino acids: Enzymology meets xenobiology. *Angewandte Chemie International Edition* **56**, 9680–9703 (2017).

[18] Grima, R., Walter, N. G. & Schnell, S. Single-molecule enzymology à la michaelis-menten. *The FEBS Journal* **281**, 518–530 (2014).

[19] Schnell, S. & Maini, P. *A century of enzyme kinetics. Should we believe in the Km and vmax estimates?* (2003). URL `https://ora.ox.ac.uk/objects/uuid:d75c660d-d149-4f8e-994d-bb5b5fc0cec1`.

[20] Erdem, E. & Woodley, J. M. Industrially useful enzymology: Translating biocatalysis from laboratory to process. *Chem Catalysis* **2**, 2499–2505 (2022). URL `https://www.sciencedirect.com/science/article/pii/S2667109322005292`.

[21] Gardossi, L. *et al.* Guidelines for reporting of biocatalytic reactions. *Trends in biotechnology* **28**, 171–180 (2010).

[22] Baker, M. 1,500 scientists lift the lid on reproducibility. *Nature* **533**, 452–454 (2016). URL `https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970`.

[23] Fanelli, D. Negative results are disappearing from most disciplines and countries. *Scientometrics* **90**, 891–904 (2012).

[24] Vasilevsky, N. A. *et al.* On the reproducibility of science: unique identification of research resources in the biomedical literature. *PeerJ* **1**, e148 (2013).

[25] Halling, P. *et al.* An empirical analysis of enzyme function reporting for experimental reproducibility: Missing/incomplete information in published papers. *Biophysical chemistry* **242**, 22–27 (2018).

[26] Tipton, K. F. *et al.* Standards for reporting enzyme data: The strenda consortium: What it aims to do and why it should be helpful: Perspectives in science, 1(1-6), 131-137. *Perspectives in Science* **1**, 131–137 (2014).

[27] Juty, N., Le Novère, N. & Laibe, C. Identifiers.org and miriam registry: community resources to provide persistent identification. *Nucleic acids research* **40**, D580–6 (2012).

[28] Marx, V. The big challenges of big data. *Nature* **498**, 255–260 (2013). URL `https://doi.org/10.1038/498255a`.

[29] Kim, S. *et al.* Pubchem substance and compound databases. *Nucleic acids research* **44**, D1202–13 (2016).

[30] Wittig, U. *et al.* Sabio-rk–database for biochemical reaction kinetics. *Nucleic acids research* **40**, D790–6 (2012).

[31] Wittig, U., Rey, M., Weidemann, A., Kania, R. & Müller, W. Sabio-rk: an updated resource for manually curated biochemical reaction kinetics. *Nucleic acids research* **46**, D656–D660 (2018).

[32] Schomburg, I., Hofmann, O., Baensch, C., Chang, A. & Schomburg, D. Enzyme data and metabolic information: Brenda, a resource for research in biology, biochemistry, and medicine. *Gene Function & Disease* **1**, 109–118 (2000).

[33] Buchholz, P. C. F. *et al.* Biocatnet: A database system for the integration of enzyme sequences and biocatalytic experiments. *Chembiochem : a European journal of chemical biology* **17**, 2093–2098 (2016).

[34] Berman, H. M. *et al.* The protein data bank. *Nucleic acids research* **28**, 235–242 (2000).

[35] Crosas, M. The dataverse network®: An open-source application for sharing, discovering and preserving data. *D-Lib Magazine* **17** (2011).

[36] European Organization For Nuclear Research, OpenAIRE. Zenodo (2013).

[37] Lapatas, V., Stefanidakis, M., Jimenez, R. C., Via, A. & Schneider, M. V. Data integration in biological research: an overview. *Journal of biological research (Thessalonike, Greece)* **22**, 9 (2015).

[38] Oliveira, A. L. Biotechnology, big data and artificial intelligence. *Biotechnology journal* **14**, e1800613 (2019).

[39] Molloy, J. C. The open knowledge foundation: open data means better science. *PLOS Biology* **9**, e1001195 (2011). URL https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001195.

[40] Deutsche Forschungsgemeinschaft. Guidelines for safeguarding good research practice. code of conduct (2019).

[41] European research council: Open research data and data management plans. URL:https://erc.europa.eu/sites/default/files/document/file/ERC_info_document-Open_Research_Data_and_Data_Management_Plans.pdf, last visited: 2023-01-17.

[42] Wilkinson, M. D. *et al.* The fair guiding principles for scientific data management and stewardship. *Scientific data* **3**, 18–27 (2016). URL https://doi.org/10.1038/sdata.2016.18.

[43] Paskin, N. Digital object identifier (doi®) system. *Encyclopedia of library and information sciences* **3**, 1586–1592 (2010).

[44] Goble, C. & Stevens, R. State of the nation in data integration for bioinformatics. *Journal of biomedical informatics* **41**, 687–693 (2008).

[45] Vonhoegen, H. *Einstieg in XML: Grundlagen, Praxis, Referenz; [für Anwendungsentwicklung und E-Publishing; Transformation, Formatierung; Schnittstellen; XML Schema, DTD, XSLT, CSS, XSL, XPath, DOM, SAX, SOAP, XQuery; XForms, HTML5, EPUB*. Galileo Computing (Galileo Press, Bonn, 2013), 7., aktualisierte und erw. aufl. edn. URL http://ebooks.ciando.com/book/index.cfm/bok_id/903055.

[46] Marrs, T. *JSON at work: Practical data integration for the web* (O'Reilly, Beijing and Boston and Farnham, 2017), first edition edn.

[47] Berners-Lee, T. Uniform resource identifier (uri): Generic syntax. URL https://www.rfc-editor.org/rfc/rfc3986.

[48] Hucka, M. *et al.* The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics (Oxford, England)* **19**, 524–531 (2003).

[49] Range, J. *et al.* Enzymeml - a data exchange format for biocatalysis and enzymology. *The FEBS Journal* (2021).

[50] Bergmann, F. T. *et al.* Combine archive and omex format: one file to share all information to reproduce a modeling project. *BMC Bioinformatics* **15**, 369 (2014). URL `https://pubmed.ncbi.nlm.nih.gov/25494900/`.

[51] Pyenzyme: Creates an enzymeml document based on a json body. URL:`https://enzymeml.sloppy.zone/redoc`, last visited: 2023-01-17.

[52] Hoops, S. *et al.* Copasi–a complex pathway simulator. *Bioinformatics (Oxford, England)* **22**, 3067–3074 (2006).

[53] Olivier, B. G., Rohwer, J. M. & Hofmeyr, J.-H. S. Modelling cellular systems with pysces. *Bioinformatics (Oxford, England)* **21**, 560–561 (2005).

[54] Nalaka Ruwan Dissanayake & GKA Dias. Web-based applications: Extending the general perspective of the service of web (2017). URL `https://www.researchgate.net/profile/nalaka-dissanayake/publication/319058851_web-based_applications_extending_the_general_perspective_of_the_service_of_web`.

[55] Chen, B., Hsu, H.-P. & Huang, Y.-L. Bringing desktop applications to the web. *IT Professional* **18**, 34–40 (2016).

[56] Zhang, D., Lin, S., Fu, Y. & Huang, S. The communication system between web application host computers and embedded systems based on node.js. In Li, Q. (ed.) *CISP-BMEI 2017*, 1–5 (IEEE, Piscataway, NJ, 2017).

[57] Ackermann, P. *JavaScript: Das umfassende Handbuch*. Rheinwerk Computing (Rheinwerk Verlag GmbH, Bonn, 2017), 1. auflage 2016, 1. korrigierter nachdruck edn.

[58] Virtanen, P. *et al.* Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods* **17**, 261–272 (2020).

[59] Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in Science& Engineering* 90–95 (2007).

[60] Ernesti, J. & Kaiser, P. *Python 3: Das umfassende Handbuch*. Rheinwerk Computing (Rheinwerk Verlag GmbH, Bonn, 2017), 4., aktualisierte und erweiterte auflage 2015, 3., korrigierter nachdruck edn.

[61] Münz, S. & Gull, C. *HTML5 Handbuch* (Franzis Verlag, Haar, Germany, 2014), online-ausg edn. URL `http://site.ebrary.com/lib/alltitles/Doc?id=10896699`.

[62] Müller, P. *Einstieg in CSS: Webseiten gestalten mit HTML und CSS ; [Grundlagen: Schriften, Farben und Box-Modelle ; Konzepte: Spezifität, Kaskade und Positionierung verständlich erklärt ; Gestaltung: Navigation, mehrspaltige Layouts und Media Queries ; aktuell zu HTML5 und CSS3*. Galileo Computing (Galileo Press, Bonn, 2015), 1. aufl., 1. korrigierter nachdr edn.

[63] Anderson, B. *et al.* Engineering the servo web browser engine using rust. In Dillon, L. (ed.) *Proceedings of the 38th International Conference on Software Engineering Companion*, ACM Digital Library, 81–89 (ACM, New York, NY, 2016).

[64] Tewari, N. & Datt, G. A study on the systematic review of security vulnerabilities of popular web browsers. In *Proceedings of International Conference on International Conference on Technological Advancements and Innovations (ICTAI - 2021)*, 314–318 (IEEE, Piscataway, NJ, 2021).

[65] Angular. URL:`https://github.com/angular/angular`, last visited: 2022-03-06.

[66] Typescript. URL:`https://github.com/microsoft/TypeScript`, last visited: 2023-01-17.

[67] Spichale, K. *API-Design: Praxishandbuch für Java- und Webservice-Entwickler* (dpunkt.verlag, Heidelberg, 2017), 1. auflage edn. URL https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5102100.

[68] Newman, S. *Monolith to Microservices* (Upfront Books and Safari, Erscheinungsort nicht ermittelbar and Boston, MA, 2021), 1st edition edn. URL https://learning.oreilly.com/library/view/-/1492047848/?ar.

[69] Ludin, S. & Garza, J. *Learning HTTP/2: A practical guide for beginners* (O'Reilly, Beijing and Boston and Farnham and Sebastopol and Tokyo, June 2017), first edition edn.

[70] Gourley, D. *HTTP*. Definitive Guides (O'Reilly Media, Sebastopol, 2002). URL http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10759109.

[71] Davies, J. A. *Implementing SSL/TLS using cryptography and PKI* (Wiley Pub, Indianapolis, Ind., 2011). URL https://learning.oreilly.com/library/view/-/9780470920411/?ar.

[72] Let's encrypt. URL:https://letsencrypt.org/, last visited: 2023-02-15.

[73] Mouat, A. *Docker: Software entwickeln und deployen mit Containern* (dpunkt.verlag, Heidelberg, 2016). URL http://nbn-resolving.org/urn:nbn:de:bsz:31-epflicht-1301513.

[74] Ubuntu docker image. URL:https://hub.docker.com/_/ubuntu, last visited: 2023-01-17.

[75] Docker hub. URL:https://hub.docker.com/, last visited: 2023-01-17.

[76] Saito, H. *DevOps with Kubernetes: Accelerating software delivery with container orchestrators* (Packt Publishing Limited, Birmingham, 2017), 1 edn. URL https://www.wiso-net.de/document/PKEB__9781788398008382.

[77] cert-manager (2022). URL:https://github.com/cert-manager/cert-manager, last visited: 2023-01-22.

[78] Helm: The package manager for kubernetes. URL:https://helm.sh/, last visited: 2023-02-01.

[79] Lottspeich, F. (ed.) *Bioanalytik* (Springer Spektrum, Berlin and Heidelberg, 2012), 3. aufl. edn.

[80] Claaßen, C., Mack, K. & Rother, D. Benchtop nmr for online reaction monitoring of the biocatalytic synthesis of aromatic amino alcohols. *ChemCatChem* **12**, 1190–1199 (2020).

[81] Schwedt, G. *Analytische Chemie: Grundlagen, Methoden und Praxis*. Wiley VCH Lehrbuchkollektion 1 (Wiley-VCH Verlag, Weinheim, 2008), 2., vollständig überarbeitete auflage edn. URL http://textbooks.wiley-vch.de/book/schwedt2061/#index.htm.

[82] Arduino documentation. URL:https://docs.arduino.cc/, last visited: 2022-03-06.

[83] Badamasi, Y. A. The working principle of an arduino. In Mahamat, M., Adeshina, S. A. & Arreytambe, T. (eds.) *Proceedings of the 11th International Conference on Electronics, Computer and Computation (ICECCO'14)*, 1–4 (IEEE, Piscataway, NJ, 2014).

[84] Pyfirmata: Python interface for the firmata (http://firmata.org/) protocol. URL:https://github.com/tino/pyFirmata, last visited: 2023-01-17.

[85] Nuseibeh, B. & Easterbrook, S. Requirements engineering. In Finkelstein, A. (ed.) *Proceedings of the Conference on The Future of Software Engineering*, ACM Conferences, 35–46 (ACM, New York, NY, 2000).

[86] Bergsmann, J. *Requirements Engineering für die agile Softwareentwicklung: Methoden, Techniken und Strategien. Unter Mitwirkung von Markus Unterauer* (dpunkt.verlag,

Heidelberg, 2018), 2. überarbeitete und aktualisierte auflage edn. URL http://nbn-resolving.org/urn:nbn:de:bsz:31-epflicht-1299942.

[87] Hull, R. & King, R. Semantic database modeling: survey, applications, and research issues. *ACM Computing Surveys* **19**, 201–260 (1987).

[88] Green, M. L. & Karp, P. D. Genome annotation errors in pathway databases due to semantic ambiguity in partial ec numbers. *Nucleic acids research* **33**, 4035–4039 (2005).

[89] Weininger, D. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences* **28**, 31–36 (1988).

[90] Bch-frontend: Frontend of the biocathub-platform. URL:https://github.com/BioCatHub/bch-frontend, last visited: 2023-01-17.

[91] Patricia Warmelink. Entwicklung einer plattform zur standardisierten erfassung und dem austausch von experimentellen daten in der biokatalyse.

[92] Bch-enzymeml: Biocathub-platform microservice, creating and reading enzymeml documents. URL:https://github.com/BioCatHub/bch_enzymeml, last visited: 2023-01-17.

[93] Pydantic: Data parsing and validation using python type hints. URL:https://github.com/pydantic/pydantic, last visited: 2023-01-17.

[94] Requests: A simple, yet elegant, http library. URL:https://github.com/psf/requests, last visited: 2023-01-17.

[95] Sarah M. Keating, Frank T. Bergmann. Libcombine: a c++ library for working with the combine archive format. URL:https://github.com/sbmlteam/libCombine, last visited: 2023-02-22.

[96] Zenodo restful api:. URL:https://developers.zenodo.org/, last visited: 2023-01-17.

[97] Bch-smiles-to-structures: Biocathub-platform microservice visualising the chamical structure based on a smiles code. URL:https://github.com/BioCatHub/bch_smiles-to-structure, last visited: 2023-01-17.

[98] Openbabel: Open Babel is a chemical toolbox designed to speak the many languages of chemical data. URL:https://github.com/openbabel/openbabel, last visited: 2023-01-17.

[99] CairoSVG: Convert your vector images. URL:https://github.com/Kozea/CairoSVG, last visited: 2023-01-17.

[100] Node.js is a javascript-based platform for server-side and networking applications. URL:https://hub.docker.com/_/node, last visited: 2023-01-17.

[101] NGINX docker image: Nginx is an open source reverse proxy server for HTTP, HTTPS, SMTP, POP3, and IMAP. URL:https://hub.docker.com/_/nginx, last visited: 2023-01-23.

[102] Python docker image. URL:https://hub.docker.com/_/python, last visited: 2023-01-17.

[103] Ingress-nginx controller for kubernetes (2022). URL https://github.com/kubernetes/ingress-nginx.

[104] Helm charts for the biocathub-platform. URL:https://github.com/BioCatHub/kubernetes, last visited: 2023-01-17.

[105] Lauterbach, S. *et al.* Enzymeml: seamless data flow and modeling of enzymatic data. *Nature methods* (2023).

[106] Jupyter. URL:https://jupyter.org/, last visited: 2023-01-15.

[107] Dienhart, H. Fair modelling of enzyme kinetics by enzymeml and jupyter notebooks (2022).

[108] Dirnagl, U. & Lauritzen, M. Fighting publication bias: introducing the negative results section. *Journal of cerebral blood flow and metabolism : official journal of the International Society of Cerebral Blood Flow and Metabolism* **30**, 1263–1264 (2010).

[109] Claaßen, C., Gerlach, T. & Rother, D. Stimulus-responsive regulation of enzyme activity for one-step and multi-step syntheses. *Advanced synthesis & catalysis* **361**, 2387–2401 (2019).

[110] Hotzymes. URL:`https://www.hotzymes.eu/about/`, last visited: 2023-01-15.

[111] Biocathub-data-processing scenario hotzymes. URL:`https://github.com/BioCatHub/bch_scenario_hotzymes`, last visited: 2023-01-17.

[112] The biocathub-github organisation. URL:`https://github.com/BioCatHub`, last visited: 2023-01-15.

[113] The pandas development team. pandas-dev/pandas: Pandas. URL:`https://github.com/pandas-dev/pandas`, last visited: 2023-01-17.

[114] Waskom, M. L. seaborn: statistical data visualization. *Journal of Open Source Software* **60** (2021). URL `https://joss.theoj.org/papers/10.21105/joss.03021`.

[115] Halling, P. J. Estimation of initial rate from discontinuous progress data. *Biocatalysis and Biotransformation* **38**, 325–342 (2020).

[116] Labib, M. *et al.* Toward the sustainable production of the active pharmaceutical ingredient metaraminol. *ACS Sustainable Chemistry & Engineering* **10**, 5117–5128 (2022).

[117] Cornish-Bowden, A. One hundred years of michaelis–menten kinetics. *Perspectives in Science* **4**, 3–9 (2015). URL `https://www.sciencedirect.com/science/article/pii/S2213020914000627`.

[118] Lin, B. & Tao, Y. Whole-cell biocatalysts by design. *Microbial Cell Factories* **16**, 106 (2017). URL https://microbialcellfactories.biomedcentral.com/articles/10.1186/s12934-017-0724-7.

[119] Mortimer, C. E. & Müller, U. *Chemie: Das Basiswissen der Chemie*. Thieme electronic book library (Thieme, Stuttgart, 2010), 10., überarb. aufl. edn. URL https://eref.thieme.de/ebooks/2108256.

[120] Corbett, J. F. Pseudo first-order kinetics. *Journal of Chemical Education* **49**, 663 (1972).

[121] Lmfit: Non-linear least squares minimization, with flexible parameter settings, based on scipy.optimize, and with many additional classes and methods for curve fitting. URL:https://github.com/lmfit/lmfit-py/, last visited: 2023-02-22.

[122] Chemspider entry of 1-Phenylacetylcarbinol. URL:http://www.chemspider.com/Chemical-Structure.9106838.html, last visited: 2023-03-01.

[123] Pubchem database entry of metaraminol. URL:https://pubchem.ncbi.nlm.nih.gov/compound/Metaraminol, last visited: 2023-03-01.

[124] Klos, N. Investigation of factors influencing the enzymatic transamination towards chiral amino alcohols and resulting reaction engineering concepts (2023).

[125] Machczynski, M. C., Vijgenboom, E., Samyn, B. & Canters, G. W. Characterization of slac: a small laccase from streptomyces coelicolor with unprecedented activity. *Protein science : a publication of the Protein Society* **13**, 2388–2397 (2004).

[126] Prins, A. *et al.* The effect of mutations near the t1 copper site on the biochemical characteristics of the small laccase from streptomyces coelicolor a3(2). *Enzyme and microbial technology* **68**, 23–32 (2015).

[127] Harris, C. R. *et al.* Array programming with numpy. *Nature* **585**, 357–362 (2020).

[128] Klinger, R., Kolárik, C., Fluck, J., Hofmann-Apitius, M. & Friedrich, C. M. Detection of iupac and iupac-like chemical names. *Bioinformatics (Oxford, England)* **24**, i268–76 (2008).

[129] Le Novère, N. *et al.* Minimum information requested in the annotation of biochemical models (miriam). *Nature Biotechnology* **23**, 1509–1515 (2005). URL `https://doi.org/10.1038/nbt1156`.

[130] de Souza, R. O. M. A., Miranda, L. S. M. & Bornscheuer, U. T. A retrosynthesis approach for biocatalysis in organic synthesis. *Chemistry (Weinheim an der Bergstrasse, Germany)* **23**, 12040–12063 (2017).

[131] Corey, E. J. The logic of chemical synthesis: Multistep synthesis of complex carbogenic molecules(nobel lecture). *Angewandte Chemie International Edition in English* **30**, 455–465 (1991).

[132] Finnigan, W., Hepworth, L. J., Flitsch, S. L. & Turner, N. J. Retrobiocat as a computer-aided synthesis planning tool for biocatalytic reactions and cascades. *Nature Catalysis* 1–7 (2021).

[133] Panico, R. (ed.) *A guide to IUPAC nomenclature of organic compounds: Recommendations 1993* (Blackwell Science, Oxford, 1998), reprinted. edn.

[134] Pubchempy: Python wrapper for the pubchem pug rest api. URL:`https://github.com/mcs07/PubChemPy`, last visited: 2023-01-19.

[135] Pymongo - the official mongodb python driver. URL `https://github.com/mongodb/mongo-python-driver`.

[136] Rigden, D. J. & Fernández, X. M. The 26th annual nucleic acids research database issue and molecular biology database collection. *Nucleic acids research* **47**, D1–D7 (2019).

[137] Gerlach, T. *et al.* Photo-regulation of enzyme activity: The inactivation of a carboligase with genetically encoded photosensitizer fusion tags. *Frontiers in Catalysis* **2** (2022).

[138] Endres, S. *et al.* An optogenetic toolbox of lov-based photosensitizers for light-driven killing of bacteria. *Scientific Reports* **8**, 15021 (2018). URL `https://www.nature.com/articles/s41598-018-33291-4`.

[139] socket — low-level networking interface. URL:`https://docs.python.org/3/library/socket.html`, last visited: 2022-03-10.

[140] Luca Fede. Industriepraktikum im studiengang elektrotechnik.

[141] Friebolin, H. *Ein- und zweidimensionale NMR-Spektroskopie: Eine Einführung* (VCH, Weinheim, 1988).

[142] Bär, H., Hochstrasser, R. & Papenfuß, B. Sila:basic standards for rapid integration in laboratory automation. *Journal of Laboratory Automation* **17**, 86–95 (2012).

[143] Sila. URL:`https://sila-standard.com/`, last visited: 2023-01-17.

[144] Ketcher: open-source web-based chemical structure editor. URL:`https://lifescience.opensource.epam.com/ketcher/`, last visited: 2023-02-15.