Bachelor Thesis
Documentation

OSMyBiz
Business Profile Editor for OpenStreetMap

Semester: Spring 2023

Version: 01.00
Date: 2023-06-15 11:44:24Z
Git Version: 9e1618a

Project Team: André Blöchlinger
Dominic Ritz
Khoa Tran

Project Advisor: Prof. Stefan F. Keller

Project Co-Advisor: Joël Schwab

School of Computer Science
OST Eastern Switzerland University of Applied Sciences
Campus Rapperswil-Jona

Abstract

Introduction

OpenStreetMap My Business (OSMyBiz) is an editor for the collaborative, open-mapping project OpenStreetMap (OSM). There are many editors available for OSM. However, none of them allow new or inexperienced users to manage their business (shop, restaurant, etc.) on OSM without knowledge of the OSM-specific data structure. OSMyBiz already allows users to create and update businesses on OSM. However, some understanding of the opening hours syntax was still required.

Goals

Expanding on the previous student research project, the goal of this work is to further improve the user experience and functionality. This should be achieved by introducing a new dialog for entering opening hours that can be used by novice users without any experience. Direct editing of nodes is another important goal. Further, OSMyBiz should record the date when the opening hours have last been checked. Existing functionality should be improved, like the watch list or the ”Unsaved Changes” dialog. In addition, the software maintenance effort for the application should be reduced.

Results

A new opening hours editor dialog has been introduced that does not require the user to know the opening hours syntax. OSMyBiz now recognizes the check_date:opening_hours tag, allowing consumers of OSM data to determine how up-to-date the hours are. A changed business no longer needs to be manually copied and saved to OSM by another mapper using another editor. The watch list has received some new features: it will notify users when the opening hours have not been checked for more than a year. Businesses can be added to the watch list without editing the business. In addition, the backend has now been migrated to Python 3.11 and its dependencies have been updated.

Several issues have been identified and fixed, such as the login process not handling errors correctly. Unsaved changes are now detected properly, and the unsaved changes dialog now appears when it was supposed to. The overall user interface style of OSMyBiz have been overhauled. The user preferences have been updated and other minor issues have been fixed.

Table of Content

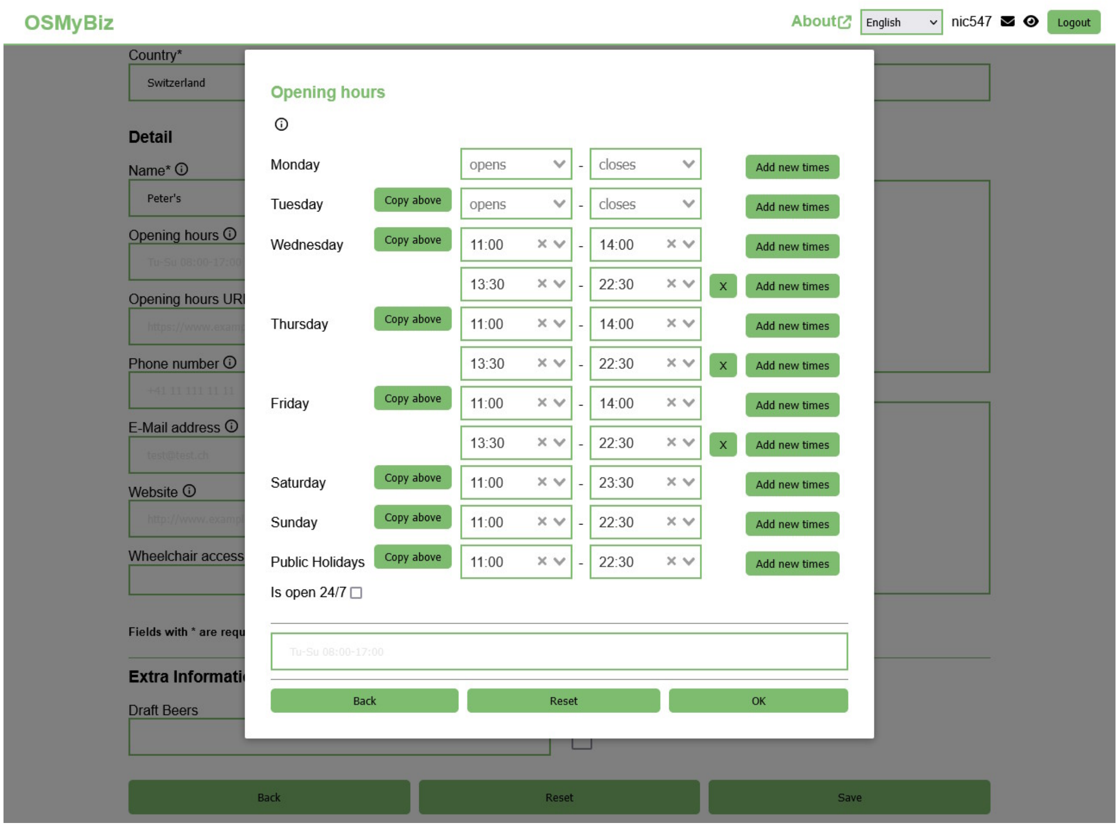
1. Management Summary

1.1 Background

OpenStreetMap My Business (OSMyBiz) is an editor for the collaborative, open-mapping project OpenStreetMap (OSM). There are many editors available for OSM. However, none of them allow new or inexperienced users to manage their business (shop, restaurant, etc.) on OSM without knowledge of the OSM-specific data structure. OSM uses a collection of key-value pairs (the so-called ”tags”) to describe elements on the map. By focusing on the use case of managing businesses, OSMyBiz enables new or inexperienced users to manage their businesses without knowing how these tags work. However, like many other simple editors, OSMyBiz lacks a simple way to edit opening hours, requiring that users learn the opening_hours syntax or find a specialized tool for this job.

1.2 Opening Hours Editor

During the project, a new opening hours editor dialog has been introduced that does not require the user to know the opening hours syntax. Rather than a text field, the user is presented with a dialog where they can select the opening and closing time (or multiple times when a business is not open during the afternoon, for example). Afterward, the data entered is converted into the OSM opening_hours format. In case more experienced users would like to enter irregular opening hours, a text field is still provided, enabling them to use the full power of the opening_hours syntax.


OSMyBiz

About English nic547 Logout

Country*

Switzerland

Detail

Name*

Peter's

Opening hours

Opening hours URL

Phone number

E-Mail address

Website

Wheelchair access

Fields with * are requ

Extra Informati

Draft Beers

Opening hours

Monday		opens	-	closes		Add new times
Tuesday	Copy above	opens	-	closes		Add new times
Wednesday	Copy above	11:00	-	14:00		Add new times
		13:30	-	22:30	X	Add new times
Thursday	Copy above	11:00	-	14:00		Add new times
		13:30	-	22:30	X	Add new times
Friday	Copy above	11:00	-	14:00		Add new times
		13:30	-	22:30	X	Add new times
Saturday	Copy above	11:00	-	23:30		Add new times
Sunday	Copy above	11:00	-	22:30		Add new times
Public Holidays	Copy above	11:00	-	22:30		Add new times

Is open 24/7 ☐

Tu-Su 08:00-17:00

Back Reset OK

Back Reset Save

Figure 1.1: OSMyBiz on a Desktop device, showing the new opening hours editor.

1.3 Direct edits

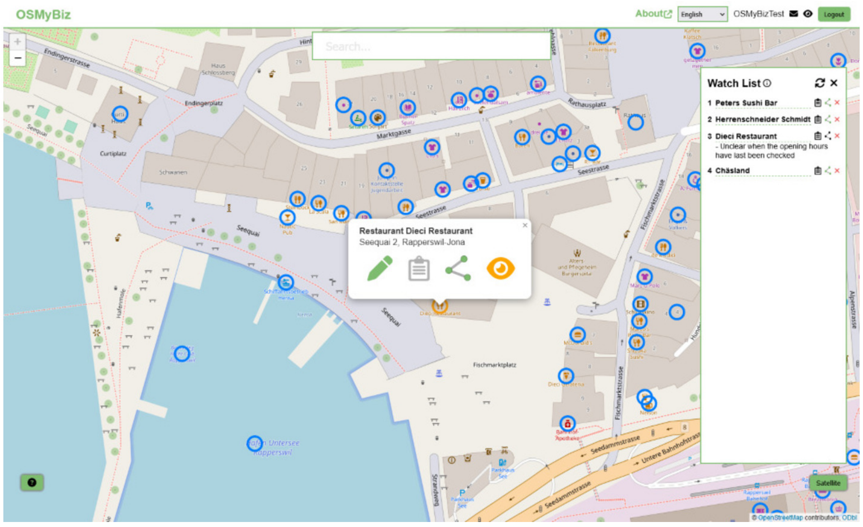
Previously, edits to a business were posted on OSM as a note. This meant that another contributor to OSM had to come across the note, make the edits (in another editor) according to the note, and then resolve the note to notify the user of the change. OSMyBiz can now directly edit things on OSM, eliminating the need for someone else to edit. It also reduces the time until an edit shows up on other applications using data from OSM and makes it obvious who provided data to OSM.

1.4 Check Date

OSMyBiz is now aware of the check_date:opening_hours tag, denoting the date when the opening hours were last checked, allowing consumers of OSM data to determine how likely it is that the opening_hours are still correct.

1.5 Watch List

The watch list has received some new features: it will tell users when the opening hours have not been checked for more than a year, based on the check_date:opening_hours tag. Businesses can now be added to the watch list without editing the business. In addition, the backend has now been migrated to Python 3.11 and its dependencies have been updated.



Watch List

1 Peters Sushi Bar

2 Herrenschneider Schmidt

3 Dieci Restaurant
- Unclear when the opening hours have last been checked

4 Chäsland

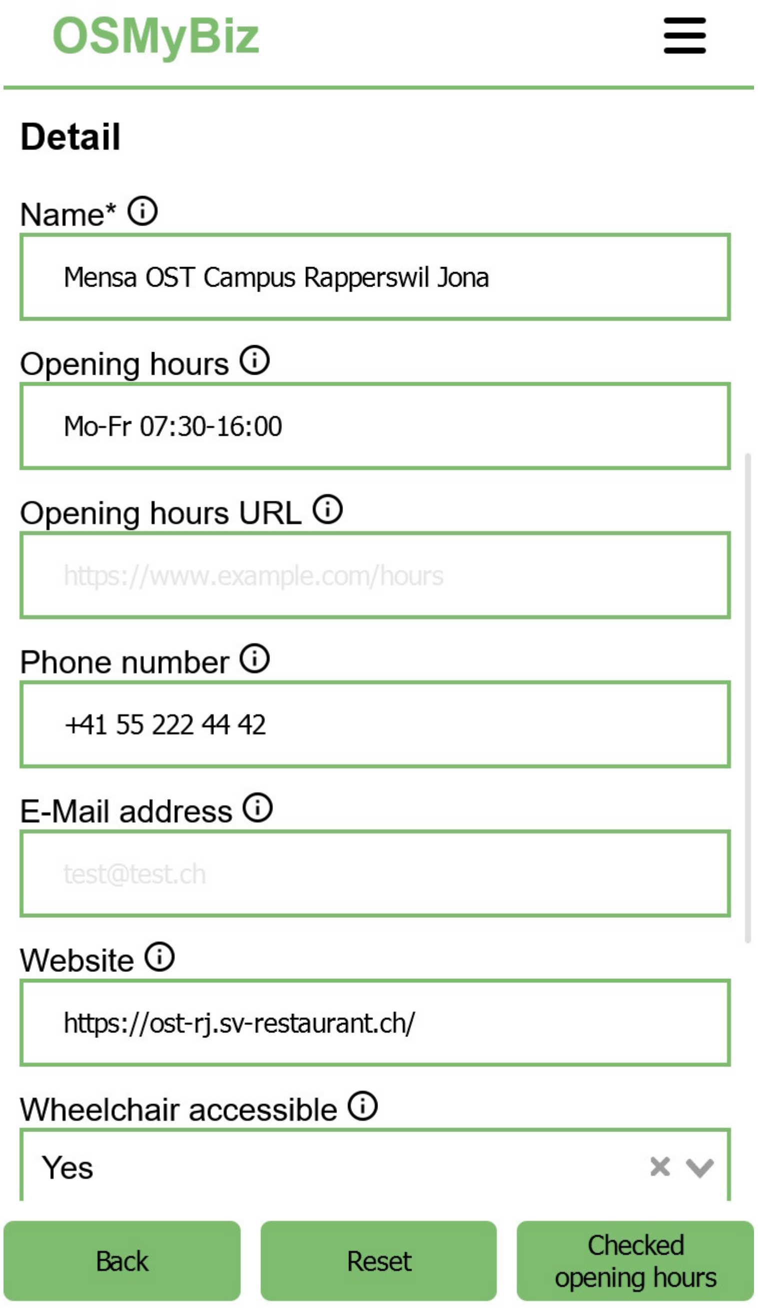
Restaurant Dieci Restaurant
Seequai 2, Rapperswil-Jona

Figure 1.2: OSMyBiz on a Desktop device, showing the new "add to watch list" button and the notifying the user about outdated opening hours.

1.6 Other changes

Several issues have been identified and fixed, such as the login process not handling errors correctly. Unsaved changes are now detected properly, and the unsaved changes dialog now appears when it was supposed to. The overall user interface style of OSMyBiz have been overhauled. The user preferences have been updated and other minor issues have been fixed.

The changes from the student research project were not upstreamed yet. Along with the changes made during this thesis, they were contributed to the original repo owned by the Geometa Lab of the Institute for Software. Along with this, issues with the deployment of OSMyBiz were identified and resolved, allowing the deployment of the current version of OSMyBiz to the staging environment (for now). Based on this, we have gathered some feedback from potential users and community members, already incorporating this feedback where possible.



OSMyBiz

Detail

Name* ⓘ

Mensa OST Campus Rapperswil Jona

Opening hours ⓘ

Mo-Fr 07:30-16:00

Opening hours URL ⓘ

https://www.example.com/hours

Phone number ⓘ

+41 55 222 44 42

E-Mail address ⓘ

test@test.ch

Website ⓘ

https://ost-rj.sv-restaurant.ch/

Wheelchair accessible ⓘ

Yes

Figure 1.3: OSMyBiz edit dialog showing the properties of a business on a mobile device

2. Assignment

Background

In the past we had the Yellow Pages, today we have Google Company Profiles - and tomorrow we will have OSMyBiz, because it is based on OpenStreetMap (OSM), free of charge and free of data snooping! The idea is that the business owners or tourism departments enter their own data into OSM. They will then appear on Qwant.com/maps and in navigation apps such as OrganicMaps, among others. OSMyBiz ("OpenStreetMap My Business") is a newly implemented responsive web application. A wide community, including the tourism industry, is using this tool. The goal is to develop OSMyBiz further to make it more widespread and easier to use.

Goals of the thesis

During the student research project, the old OSMyBiz website was improved by refactoring. Now further improvements and features should be added.

- Nodes should be editable directly, not indirectly via a note
- Object class templates (presets) should be available
- It should be easy to enter opening hours. An intuitive GUI should be used, but also generators like WebToOSMOH.
- The maintenance effort on the website should be optimized. (Well-defined non-functional requirements). For example: Sensible use of schema migration (e.g. with alembic or flask-migrate).

Further ideas:

- Capture addresses better as ID, e.g. with addr:place
- Scrape opening hours based on the given URL (cf. `https://osm-de.github.io/WebToOSMOH/`)
- Add an automatic tag check_date_opening_hours for opening hours
- Synchronization with presets from iD (JOSM/Vespucci) without recompilation
- Website validator/URL check in general (e.g. for all businesses someone "manages")
- Microdata scraping given URL, especially OH scraper: see JOSM plugin `https://gitlab.com/vucod/microdata-scraping`

Technologies

- Frontend: website with JavaScript, Leaflet Lib.
- Backend: Python, Flask Lib., PostgreSQL.

Part I

Technical Report

3. Introduction

The OSM project is a collaborative effort to collect geographic data and distribute it under the Open Database Licence. It is an alternative to proprietary map data providers such as Google Maps or Bing Maps. Contributions to the project follow the Pareto principle, meaning that a majority of the data is contributed by a minority of the contributors.

OSM uses a very flexible data model. POIs (generally ”ways” or ”nodes”) can be tagged with arbitrary key-value pairs. The possible keys and values (the so-called ”tagging-scheme”) are not defined by the project, but are rather based on conventions. Most tags are not exactly controversial and are used in a consistent way, documented by the OSM wiki and often supported by editors. However, there may be regional differences, and this tagging scheme can become extremely specific and complex.
Unfortunately this has led to a situation where most editors are very powerful and can be used to edit almost any aspect of the map, but require some knowledge of the inner workings of the OSM project.

In 2017, Max Lüthi and Simon Heller created the OSM editor ”OpenStreetMap My Business”[17] as part of their bachelor thesis at the University of Applied Sciences Rapperswil. Their goal was to provide a way for business owners to add their business to OSM without any knowledge of OSM by focusing on businesses and abstracting away from the OSM data model. After the initial release the project was maintained by some community members and employees of the University of Applied Sciences Rapperswil. Unfortunately the project has fallen behind in terms of dependencies.

During the autumn semester of 2022, OSMyBiz was reworked as part of a student research project [13]. This included a complete upgrade of all frontend dependencies, as well as fixing the design on mobile devices and improving the user experience by caching certain API requests. Unfortunately these changes have not yet been merged into the upstream repository of the project.

3.1 Vision

OSMyBiz successfully enables users to contribute to OSM without knowledge about the OSM-specific tagging scheme, thereby lowering the barrier to entry for contributions. OSMyBiz can be recommended to users who fit our target audience and improves the quality of data related to businesses on OSM, benefiting the OSM project and users of applications that use data from OSM.

3.2 Goals

The main goals we want to achieve are as follows:

- Improve the editing of ”opening_hours” in a way that allows users to edit them without knowing the specific format of the ”opening_hours” tag.
- Edit POIs directly, rather than just posting notes for other users to resolve.
- Update the backend to use the latest versions of its dependencies and python.

- Ensure that the changes made during the previous student research project and this bachelor thesis can be merged and deployed into the project's upstream repository.

3.3 Methodology

This thesis consists of two main parts: the technical report and the project documentation. The technical report mainly covers the introduction to the project, the methodology, the tools and technologies used as well as the results and conclusions. The project documentation follows a software engineering-specific approach, including requirement specifications, implementation details, results and options for further development.

At the beginning of this work, the focus was on familiarization with the existing project. The tools and programming languages already in use were also closely examined. At the same time, efforts were made to incorporate changes from the student assignment to the upstream from the original repository. The project initiation also included defining goals, analyzing risks, and finding solutions using existing tools and technologies.

During the elaboration phase, the project became more clearly defined. The development of OSMyBiz was in full swing. An initial prototype for the opening hours editor was created and tested. Further improvements were made to OSMyBiz.

The development team worked in sprints and communicated at least once a week. Regular meetings were also held with the advisors to discuss the current status, make decisions and get helpful advice.

4. State of the Art

This chapter covers the current state of the art in relation to OSM.

4.1 iD-Editor

The iD-Editor is a web-based editor for OSM. It is the default editor on openstreetmap.org and one of the most popular editors. It is relatively simple to use and allows users to edit elements without being aware of the OSM tagging scheme. It does not hide them however, so more advanced users are free to edit them in text form, allowing tags that iD might be unaware of. It supports editing all types of POIs and is not limited to Businesses, which can make it a bit overwhelming for new users. It does not include a solution for editing opening hours other than a basic text field. [25]

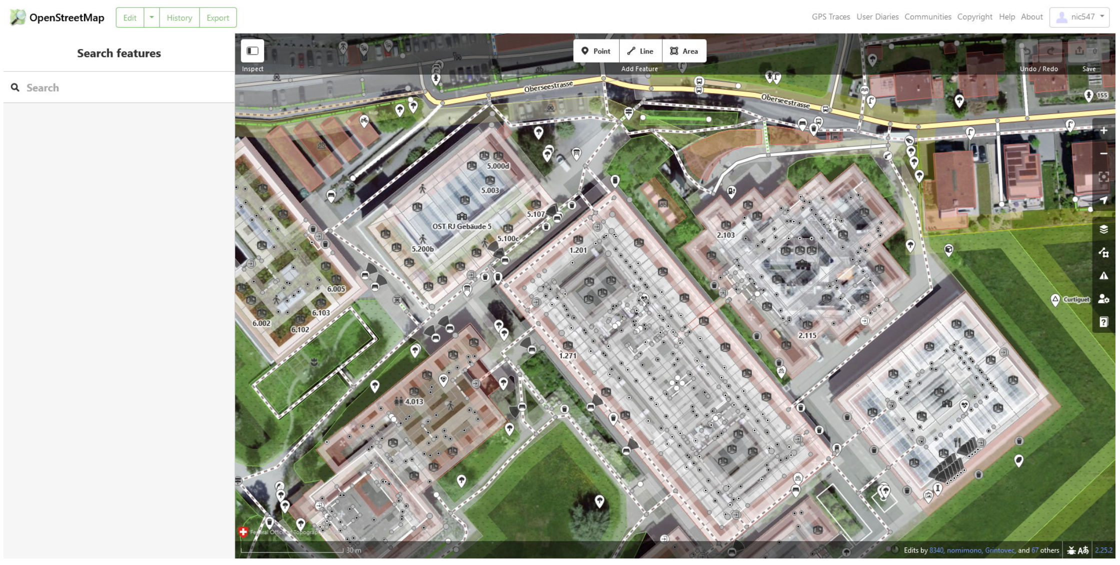
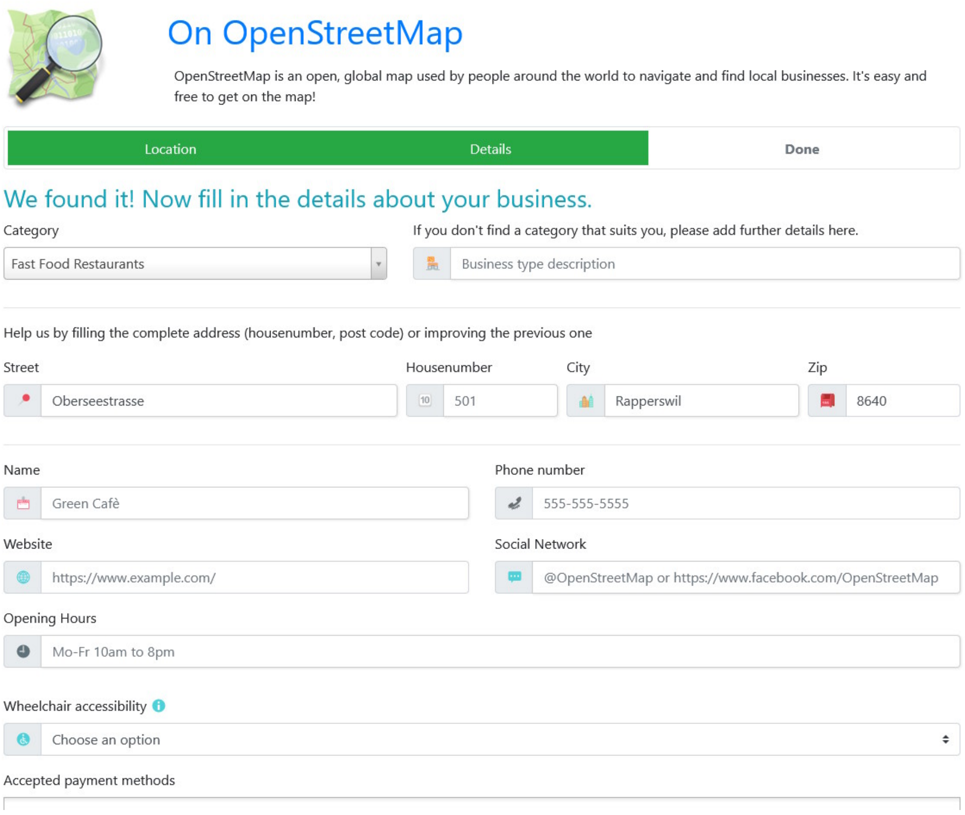


Figure 4.1: The iD-Editor

4.2 onosm.org

onosm.org is most similar to OSMyBiz. It is relatively limited however: businesses are located via their address; a map can only be used after entering an address to fine-tune the location. It does not allow editing businesses, only creating new ones. onosm.org does not support presets, instead only supporting a fixed set of tags. onosm.org does not provide an editor for opening_hours beyond a basic text field. [22]


On OpenStreetMap

OpenStreetMap is an open, global map used by people around the world to navigate and find local businesses. It's easy and free to get on the map!

Location | Details | Done

We found it! Now fill in the details about your business.

Category	If you don't find a category that suits you, please add further details here.
Fast Food Restaurants	Business type description

Help us by filling the complete address (housenumber, post code) or improving the previous one

Street	Housenumber	City	Zip
Oberseestrasse	501	Rapperswil	8640

Name	Phone number
Green Cafè	555-555-5555
Website	Social Network
https://www.example.com/	@OpenStreetMap or https://www.facebook.com/OpenStreetMap

Opening Hours: Mo-Fr 10am to 8pm

Wheelchair accessibility: Choose an option

Accepted payment methods

Figure 4.2: The details page of a business on onosm.org.

4.3 YoHours

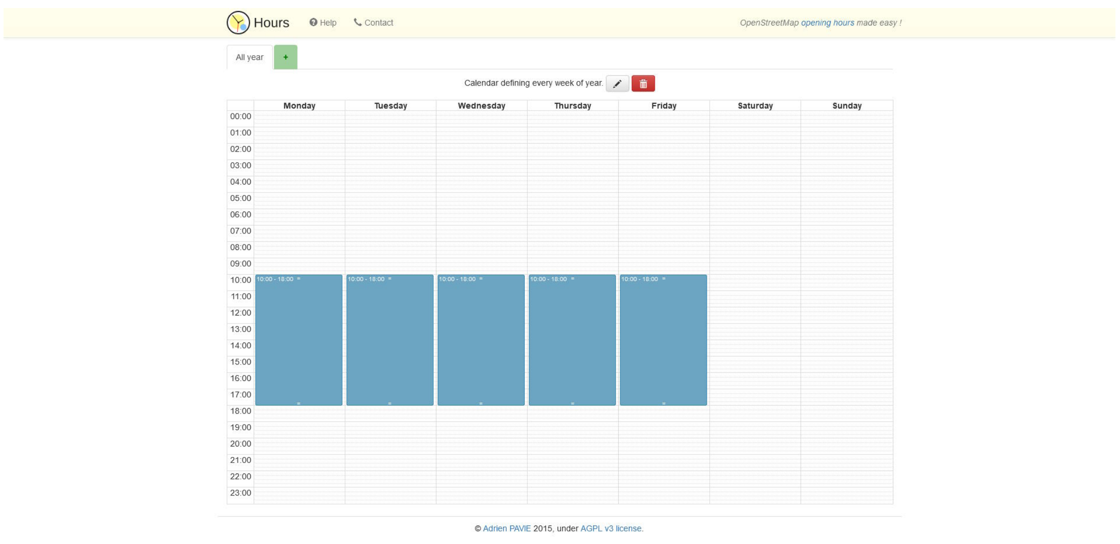


Figure 4.3: YoHours being used on a desktop computer.

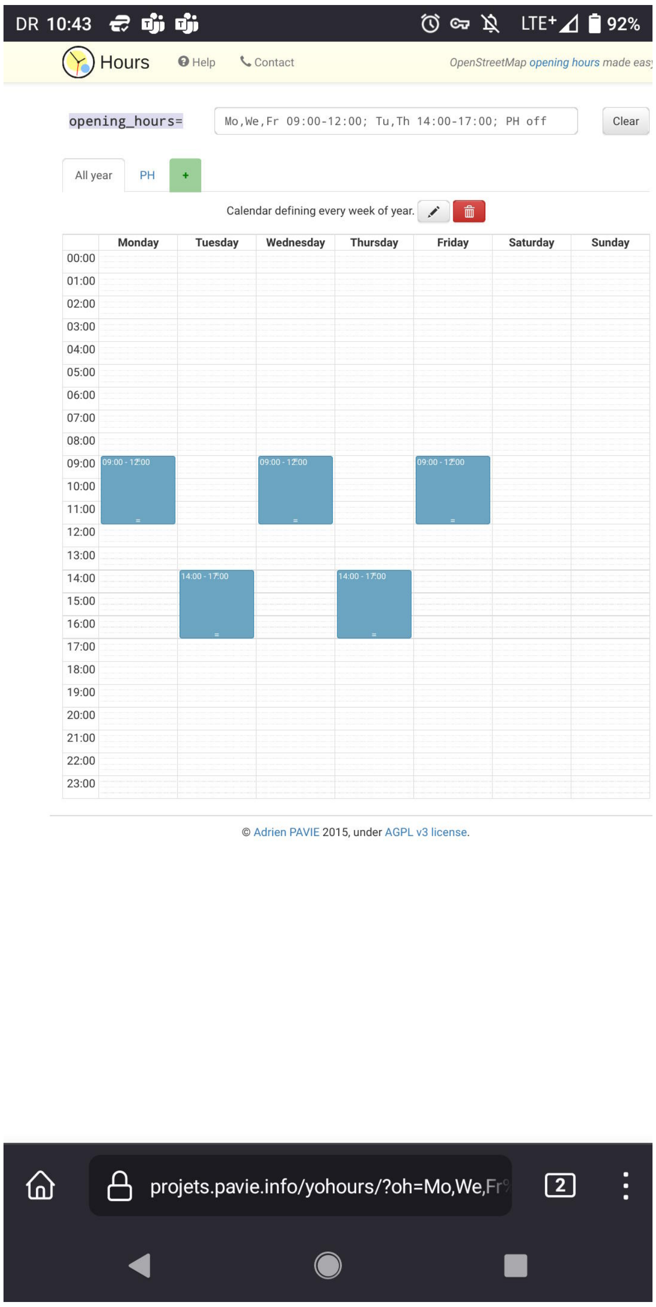
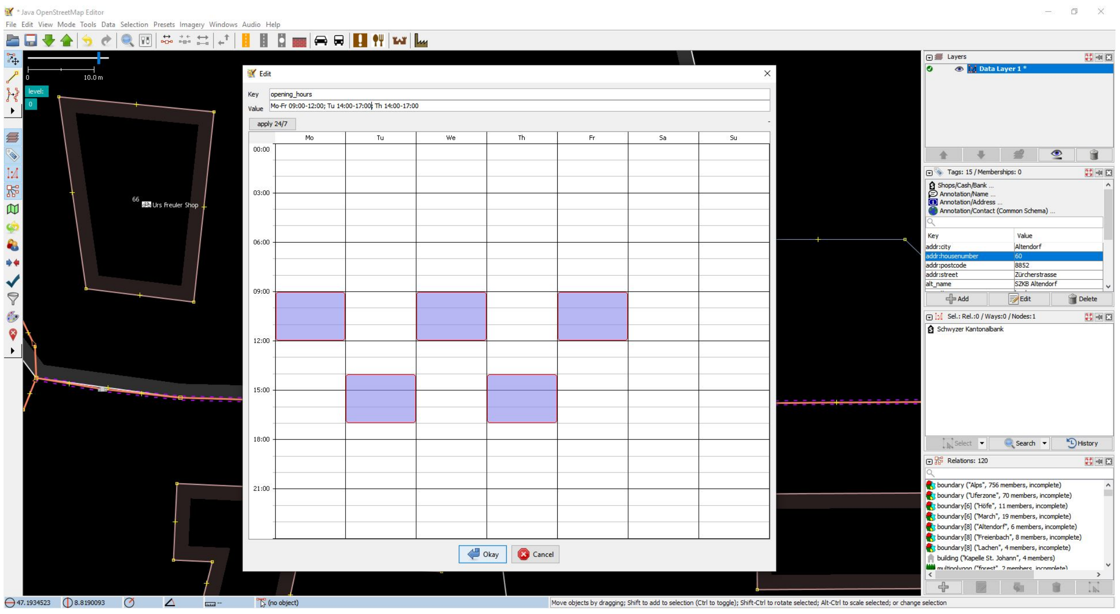

opening_hours= Mo,We,Fr 09:00-12:00; Tu,Th 14:00-17:00; PH off

Figure 4.4: YoHours being used on a mobile device. Zooming and panning is possible.

YoHours is a web-based editor for editing opening hours and does not include any other functionality. A week is represented by a table, where each column represents a day of the week and each row represents a 15 minute time interval. By clicking and dragging on the table, a user can create a cell that represents a time interval during which something is open. The tool then generates a string that represents the opening hours in the OSM format that a user can copy and paste into their editor of choice.

While this approach provides a great overview of the opening hours, we found it hard to accurately hit specific cells. This problem is even more evident on mobile devices. Additionally the table does not scale well on smartphone screens.

4.4 OpeningHoursEditor (JOSM-Plugin)



Key: opening_hours

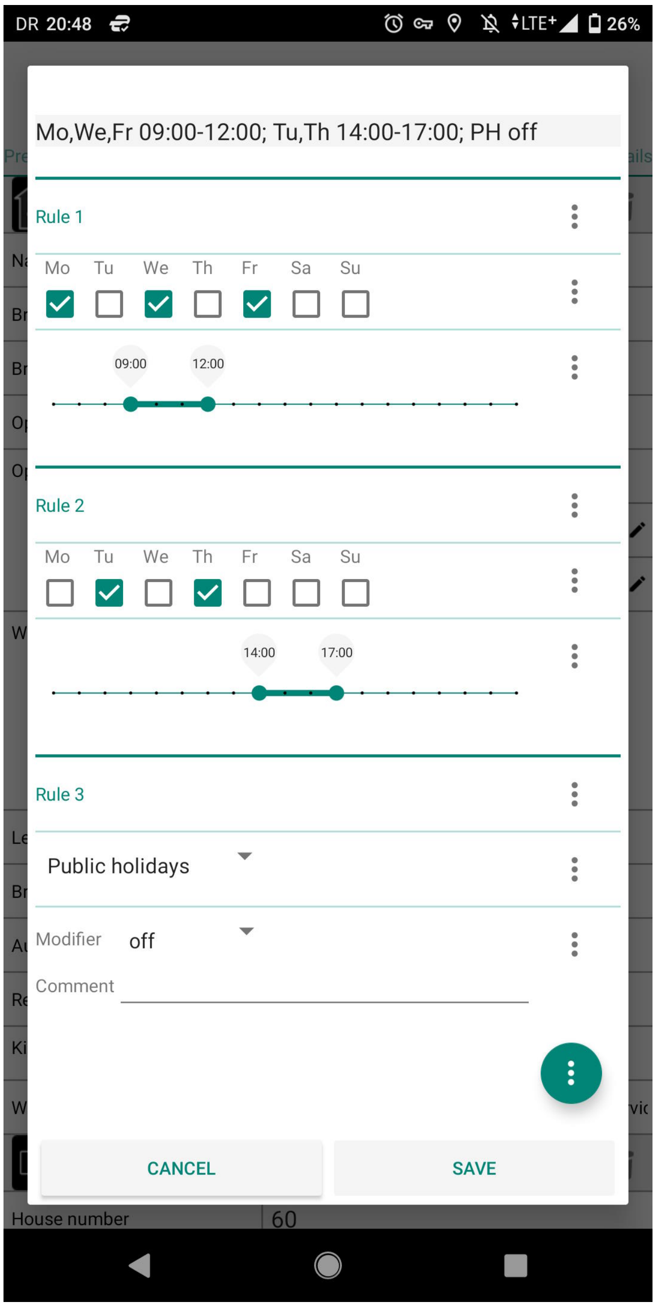
Value: Mo-Fr 09:00-12:00; Tu 14:00-17:00; Th 14:00-17:00

Key	Value
addr:city	Altendorf
addr:housenumber	60
addr:postcode	8852
addr:street	Zürcherstrasse
alt_name	SZKB Altendorf

Figure 4.5: OpeningHoursEditor

JOSM is a java-based desktop editor for OSM. It is powerful but fairly complex and probably not what our target audience expects. JSOM is extensible via plugins, for example ”OpeningHoursEditor” adds a Editor for opening hours to JOSM. Similar to YoHours, it uses a table to represent the opening hours. Compared to YoHours it allows to create a ”opening hours cell” that spans across multiple days. [26]

4.5 Vespucci


Mo,We,Fr 09:00-12:00; Tu,Th 14:00-17:00; PH off

Rule 1

Mo [✓] Tu [ ] We [✓] Th [ ] Fr [✓] Sa [ ] Su [ ]

09:00 12:00

Rule 2

Mo [ ] Tu [✓] We [ ] Th [✓] Fr [ ] Sa [ ] Su [ ]

14:00 17:00

Rule 3

Public holidays

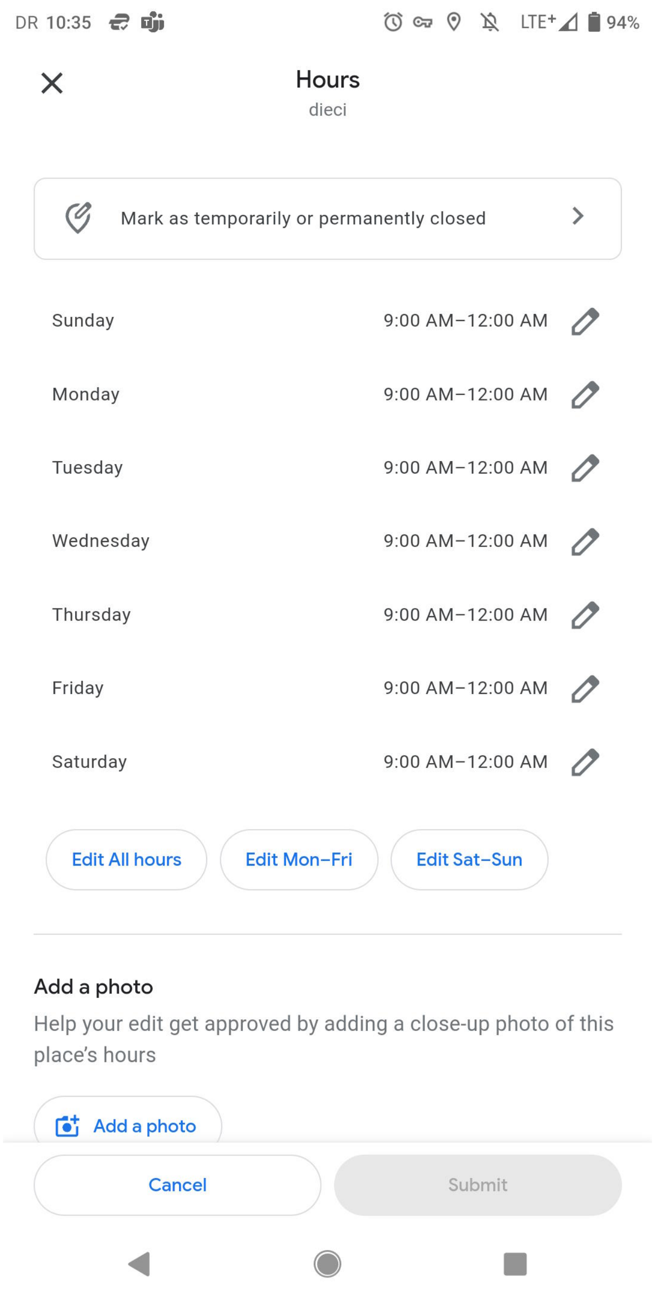
Modifier off

Comment

Figure 4.6: The opening_hours editing interface in Vespucci

Vespucci is a android-based editor for OSM. It also includes a editor for opening hours. The editor is based on a list of rules. A basic rule consists of one or more days (weekdays, specific dates, etc.) and one or more timespans. Vespucci supports many complex scenarios, for example opening hours that depend on sunrise or sunset, are open-ended. This complexity is probably unnecessary for 99% of all businesses and is quite confusing for users that are not familiar with OSM.

4.6 Google Maps/Google Business Profile



DR 10:35 LTE+ 94%

Hours
dieci

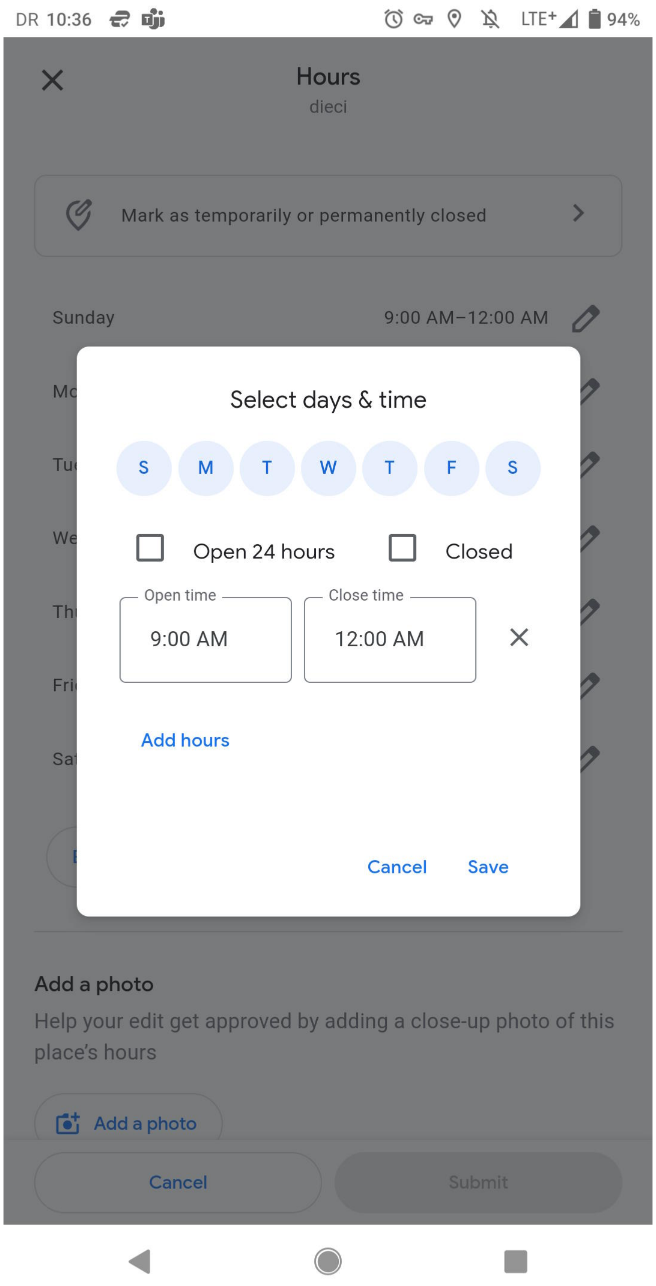
Mark as temporarily or permanently closed

Sunday	9:00 AM–12:00 AM
Monday	9:00 AM–12:00 AM
Tuesday	9:00 AM–12:00 AM
Wednesday	9:00 AM–12:00 AM
Thursday	9:00 AM–12:00 AM
Friday	9:00 AM–12:00 AM
Saturday	9:00 AM–12:00 AM

Add a photo

Help your edit get approved by adding a close-up photo of this place's hours

Figure 4.7: Overview over the opening hours in the google maps interface.


DR 10:36 LTE+ 94%

Hours
dieci

Mark as temporarily or permanently closed

Sunday 9:00 AM–12:00 AM

Select days & time

S M T W T F S

Open 24 hours Closed

Open time 9:00 AM
Close time 12:00 AM

Add hours

Cancel Save

Add a photo
Help your edit get approved by adding a close-up photo of this place's hours

Add a photo

Cancel Submit

Figure 4.8: Editing opening hours.

Google Maps/Google Business Profile is likely the most popular mapping tool [6] and most potential users of OSMyBiz are probably familiar with it.
For opening hours Google Maps uses a similar approach to Vespucci, but keeps it much simpler. For each weekday, one or more timespans can be specified during which a business is open.

5. Evaluation

In this chapter some features are further evaluated in terms of cost versus value.

5.1 Removing OAuth Secret from Frontend

The current implementation of the authentication uses the JavaScript library osm-auth [15], which provides an easy authentication with OSM over OAuth 2.0.

This is done in the Vue-based frontend and the authentication process with OSM requires OSMyBiz to provide its generated OAuth 2.0 client ID and client secret.

As this whole process is done in the frontend, the client ID and client secret that will be sent to OSM is exposed to third parties.

5.1.1 Potential damage with the Secret

The potential damage that can be done with the secret is rather low.
The client ID and the secret are only used to start the authentication process with OSM.

There is still the need to provide the correct login credentials of an OSM user and anything done during that session is accounted to that user.
So by obtaining this data, a malicious user cannot do anything besides what he can already do with his OSM user, like modifying any data records on OSM.

5.1.2 Analysis

There currently is no equivalent library for the Python backend to handle the OSM authentication.

A rather unknown library osm-login-python [31] exists, which only provides basic functionality for getting the needed login URL as well as the user data.
This is by far not enough to cover the current functionalities provided by osm-auth that are used in this project.

Another possibility is to change the backend from Python to a Node.js backend, which would allow the use of osm-auth.
This would take a bigger effort as it involves the switch to another technology.

5.1.3 Decision

Since the potential damage is low and the focus lies on the core functionalities, it is decided that the removal of the OAuth secret from the frontend will not be done during this project.

As both the possibilities to move the login functionality to the backend would be a rather time-consuming change and the benefit gained is almost non-existent, it is not worth the effort as there is not enough time during this project.

5.2 Synchronization with presets without recompilation

Currently OSMyBiz uses the presets from the iD editor [25] which are checked into git. The original presets are then modified and filtered to fit the needs of OSMyBiz. After building the frontend for deployment, they are part of a static set of files. These files get distributed as part of the osmybiz-nginx docker image. Updating them without a rebuild would increase complexity, as the container would need tooling to update the presets. Additionally, we found that the format of the iD presets has changed at some point. This required adapting the existing tooling to work with the new format. Keeping the presets (or at least their version) as part of the version-controlled project means that such breaking changes cannot occur. For this reason, we decided to focus on other features.

6. Technologies

As this project is based on the previous student research project[13], it uses the same technologies.
The technologies were overhauled over the course of the student research project, so the focus in this project lies on the features.

In the following sections, the key technologies will be listed.

6.1 Backend

Flask

Flask [27] is a Python web framework.
It is a microframework [14], resulting in its core being simple and extensible.

6.2 Frontend

Vue 3

Vue is a JavaScript framework for building user interfaces. Vue was selected in 2017 due to its relative simplicity and good documentation, making it easy to pick up for developers.[17] It is still a popular choice for JavaScript frontend frameworks and offers a solid ecosystem of libraries.

Vite

Vite is a build tool for JavaScript applications. It handles typescript transpilation, CSS pre-processing, minification, etc. During development, it operates as a web server and uses the JavasScript module functionality of modern web browsers to provide much faster builds than traditional bundlers. For deployment, Vite creates a static bundle of files.

Typescript

TypeScript is a superset of JavaScript, adding, amongst some other features, a compile-time static type system to try and offer better tooling to make development and maintenance easier.[18] At the inception of OSMyBiz, there was a deliberate decision against TypeScript, as its benefits were deemed too minor to justify the added effort required. [17] During the student research project, TypeScript was introduced to facilitate easier maintenance. [13]

Pinia

Pinia is the official state management library for Vue. It provides a simple way to manage application state via so-called stores, making it easier to share data across components. It is type-safe and supports static typing via TypeScript.

Leaflet

Leaflet is ”an open-source JavaScript library for mobile-friendly interactive maps.”[1] Leaflet integrates with Vue via the ”vue-leaflet” library, reducing the effort required to incorporate Vue’s approach to reactivity with the ”framework-agnostic” leaflet. [35]

6.3 Database

Postgres

Postgres [8] is an open-source object-relational database management system (ORDBMS).
It features transactions with ACID properties among others and is designed to handle a range workloads, from single machines to data warehouses or webservices with many concurrent users. Besides SQL it also supports JSON querying.
Postgres is a very reliable database and is used in many web applications as well as big data applications.

6.4 Testing

GitLab CI/CD

GitLab CI/CD is part of GitLab and allows developers to define pipelines for automating Integration, Testing and Deployment as .yaml files. A GitLab runner was set up on our Infrastructure to run the pipelines. [4]

Flake8

Flake8 is a tool to enforce code style guidelines in Python code and runs as part of the CI/CD pipeline. [5]

Pytest

OSMyBiz used Pytest as Test-Framework for the backend. However, only the URL-Check (whose inclusion into OSMyBiz is doubtful) has test coverage. [9]

ESlint

ESLint is a static analyzer for JavaScript/TypeScript, concerned with both code style and code quality. It runs as part of the CI/CD pipeline. [23]

vue-tsc

vue-tsc is a tool for type-checking applications built with TypeScript and Vue. It is also part of the CI/CD Pipeline of OSMyBiz. [12]

6.5 Tools

6.5.1 VS Code

VS Code is a freely available and extensible Editor/IDE for Windows, Linux, and MacOS. It offers extensions for Python, Vue, and Latex, solving the need for different language-specific IDEs. [19]

6.5.2 Docker

OSMyBiz used Docker for deployment, and Docker (along with Docker Compose) can also be used during development. Using Docker even during development simplifies the management of dependencies; developers do not need to install the correct version of Node or Python themselves, and there is no conflict with other dependencies already installed on a system. [10]

6.5.3 WSL2

The Windows Subsystem for Linux 2 allows users to run Linux applications on a Windows System. WSL uses some Hyper-V components but is better integrated into Windows. Running Docker on Windows requires using either the ”full” Hyper-V or WSL2; Docker recommends the latter, and unlike ”full” Hyper-V, it does not require specific Editions of Windows. [20]

6.5.4 Browsers

The team used a combination of Chrome/Chromium, Firefox and Edge for developing and testing the Application. Safari has only been used during some tests because not all team members possess hardware licensed to run MacOS or iOS.

6.5.5 Postman

Postman is a tool to develop and test APIs, simplifying the development process by enabling developers to test their APIs before implementing a feature in the frontend. [29]

6.5.6 Vue Devtools

The Vue-Devtools are available for Chrome, Firefox, and Edge and expand the browser-included dev tools with vue-specific functionality. [7]

6.5.7 Gitlab + Git

Git is a widely used version control system. The existing OSMyBiz application is in a Git repository hosted on gitlab.com and uses GitLab CI/CD Features.[4] [2]

6.5.8 Jira

A Jira instance was set up and used for project management and time tracking. [3]

7. Results

This chapter looks at the outcomes that have been researched and developed and how they compare with the initial stage of the project. It also discusses possible ideas for further development. Finally, there are the conclusions and some personal insights.

7.1 Goal Attainment

Earlier in this document (3.2) the main goals were set. These and other objectives will now be discussed.

A user-friendly editor was created that allows users to easily enter their opening hours. This then converts the input into an OSM-compliant opening hours output. In addition, the editor recognises the opening hours of existing businesses and converts them where possible. A text field has also been added for more complex entries such as seasonal opening times.

POIs can now be edited directly. Previously, only a note was created during an edit. This then had to be approved by other users.

Another goal that has been achieved, is the update of the backend. So OSMyBiz uses the latest versions of the dependencies used by the backend and of Python 3.11.

Since not everything in the previous work was merged into the upstream repository, this part fell into the current project. The pending merge request had to be split because it was very large and difficult to review.

In addition to the above-mentioned accomplishments, several bugs that occurred during the process were fixed. Feedback from the user experience tests was collected and analyzed, leading to the integration of certain aspects into the code. Further functions were implemented.

7.2 Advancement

There is still room for many new features and improvements in OSMyBiz.
This thesis' main focus point is on the user experience and as such, many confusing issues and fundamental bugs were fixed.
With the introduction of the opening hours editor, users can now create new businesses without having deeper knowledge about OSM and its data structures.

Following are the most important features or issues to be addressed further

- Usage of Vector Tiles for the Map Layer (see 12.2)
- Automated Scraping of website data (see 12.2)
- Replacing right clicking (see 12.2)

7.3 Personal Reports

There is a report from each team member describing their personal experience during the course of the project.

André Blöchlinger

Working together in this team has been wonderful. Even though we did not know each other before, we got along very well. Since Dominic was already involved in the study work, he was able to pass on a lot of knowledge to us, which was extremely helpful. Our supervisors were also very helpful throughout the project and it was a pleasure to work with them.
The whole project was very exciting and educational, but also exhausting. We were familiar with some of the technologies used, which made it easier to get started. The development went well for the most part, with a few problems here and there, the daily bread of a software engineer.
My most important findings from this bachelor thesis are, on the one hand, how much it helps to have a well-developed project management and, on the other hand, how much more fun it is to work in a well-functioning team. In addition, the experience and knowledge gained will stay with you. In the end, we were able to implement many important features, and I would have liked to implement many more. Fortunately, OSM is open source, in case I want to do something in the direction of OSM again in the future.

Dominic Ritz

I'm pleased I could continue working on OSMyBiz after the student research project. Having closer contact with OSM has fascinated me with the project and its community. I think it's commendable how a community of hobbyists, governmental actors, and commercial users contribute to one collaborative source of geodata under the ODbL, enabling them to do exciting or important things that probably would be much harder or impossible without OSM. I can tell from first-hand experience that, unfortunately, business-related data on OSM is, at least in Switzerland, only partially reliable. Google Maps is quite accurate thanks to the many contributions by users. Being well aware of available OpenStreetMaps-Editors, most editors are more suited to complex tasks and can't compete with the simplicity of the Google Maps editor. For this reason, OSMyBiz is something I'd like to have some success, both to use it myself and benefit from more accurate data on OSM.

Because the continuation of the project was a rather spontaneous decision, my original teammate already had other plans. While I'm also happy with my new teammates, this was a significant loss of knowledge. None of us three have much experience with Vue applications, generally having worked with different stacks. At the same time, I was the only one with significant experience with OSM and OSMyBiz. In quite a few situations, this has not been very pleasant. I've encountered multiple issues that could have been relatively simple for someone with experience with the respective technologies but wasted a good portion of time for us as inexperienced developers. For me, this reinforced the idea of ”institutional knowledge”, trying to keep experienced developers around and transferring know-how from more senior developers to junior developers.

Generally, I found it hard to justify explicit ”bug hunting” or adding automated tests over adding features. For features, it’s much easier to demonstrate the work done and the value they deliver, while it’s a lot harder for testing. However, this is certainly not limited to OSMyBiz, as I’ve heard this from other people working as software developers. It’s certainly something I will keep an eye on in future projects.

I’ve enjoyed getting some more experience with a stack I have not used professionally. Despite some difficulties, I’ve enjoyed working on OSMyBiz with my team, and I am happy with how our new features turned out. I’m looking forward to using these new features for actual changes myself.

Khoa Tran

At first, I was not sure how the project would pan out, because I was a complete novice regarding OSM. Although I had some experience before with a TypeScript project, I was new to Vue and had not worked with a Python backend before.

Luckily, Dominic was already well-versed with the project and the technologies, since he worked on OSMyBiz before during his student research project.
His knowledge and guidance helped André and me get through our initial phase of gathering knowledge. It took me some time to get accustomed to the OSMyBiz technology stack, so I was very slow to solve the issues at first.

I enjoyed working together with André and Dominic a lot. We didn’t know each other at first, but with the, often times online, meetings we held, we slowly got to know each other better and the team chemistry improved very quickly over the course of the project.

It was very helpful that André and Dominic were quick to respond on our communication channel, Microsoft Teams, also often being open for a quick call to discuss urgent issues.

During this project, I have learned that planning and execution of early tests is very important. As we have planned the more important tests on the later stages, we got under time pressure, because some feedback was negative and additional bugs were found. That led to us having to fix the issues in a very quick manner to finish the project in time.

Another issue is the lack of automated testing implemented by us. In hindsight, I would definitely plan a better testing concept next time to ensure there will be enough automated tests running.

The work on OSMyBiz has been my first experience with an open-source project. It was a very fun experience working on this project, considering it was hard to get into due to my lack of prior knowledge.
All in all I was very satisfied that we could deliver a project that we could proudly use ourselves.
After this project, I’m very excited to dive into many other interesting projects in the future.

Part II

Project Documentation

8. Requirements

This chapter defines the functional and non-functional requirements as well as the use cases.

8.1 Functional Requirements

The following functional requirements (FR) represent product characteristics or functions that are implemented so that the users can perform their tasks. Some of these FRs are summarized from the previous student research project [13]. Significant changes are marked in bold.

ID	Description
FR1	Users are able to browse the map.
FR2	Users can create new businesses.
FR3	Users can edit existing businesses **directly** on the node.
FR4	Users will be informed if another user sends them a direct message on osm.org.
FR5	Users are notified when a business they have created or edited has been edited by another user.
FR6	**Object class templates (presets) are available.**
FR7	OSMyBiz can handle newlines in the input fields.
FR8	**Maintenance effort on the website is optimized.**
FR9	**Users are able to easily enter opening hours. This can be done by an intuituve GUI, but also with generators like WebToOSMOH.**
FR10	**OSMyBiz should add the check_date tag for the opening_hours automatically.**
FR11	**OSMyBiz notifies users for entries on their watch list whose opening hours have not been checked recently.**
FR12	**Users are able to add businesses to their watch list manually.**
FR12	**Users can signalize that they have checked the opening hours of a business.**

Table 8.1: Funtional requirements

8.2 Use Cases

The use cases for OSMyBiz have not significantly changed since the student research project.[13] Besides the existing ones, the use cases for checking the opening hours as well as adding businesses to the watch list were added.

8.2.1 UC1: Browse Map

Description	The user would like to learn more about the application.
Actor	User (anonymous)
Pre-Condition	-
Main Success Scenario	1. The user enters an address into the search bar. 2. The map shows the location of the address. 3. The user right clicks the building and reads the information provided. 4. Steps 1-3 can be repeated. 5. The user leaves the site.
Post-Condition	-
Alternative Scenario	1a. The user selects address by zooming. 3a. The user left-clicks the business. 5a. The user creates an OSM-Account.

8.2.2 UC2: Create Business

Description	The user wants to add their business to OSM.
Actor	User (logged in)
Pre-Condition	The user has an OSM account and is logged in.
Main Success Scenario	1. The user enters an address into the search bar. 2. The map shows the location of the address. 3. The user right clicks the building. 4. The user checks the address and clicks ”Create”. 5. The user enters the details about the business. 6. The user saves changes by pressing ”Save”.
Post-Condition	OSM-Node was created
Alternative Scenario	1a. The user selects address by zooming. 3a. The user ”long touches” the building. 6a. The user does not want to save their changes and clicks ”Back”.

8.2.3 UC3: Edit Business

Description	The user would like to edit an existing business.
Actor	User (logged in)
Pre-Condition	- User has an OSM account and is logged in - Business has been created
Main Success Scenario	1. The user enters an address into the search bar. 2. The map shows the location of the address. 3. The user left clicks the existing business. 4. The user checks the address and clicks ”Edit”. 5. The user edits the details about the business. 6. The user saves changes by pressing ”Save”.
Post-Condition	*Node is updated directly on OSM*
Alternative Scenario	1a. The user selects address by zooming. 6a. The user does not want to save their changes and clicks ”Back”.

8.2.4 UC4: Show Updates

Description	The user would like to be informed when someone else edits a business they have created or edited.
Actor	User (logged in)
Pre-Condition	- The user has an OSM account and is logged in - Business has been created or edited by the user
Main Success Scenario	1. The user checks the edits. 2. The user dismisses the notification about the edit.
Post-Condition	The edit is no longer shown on the watch list.
Alternative Scenario	2a. The user selects business by zooming. 2b. The users check the edit on osm.org. 2c. The user turns off notifications for this business.

8.2.5 UC5: Show OSM Messages

Description	The user wants to be notified when someone sends them a direct message on osm.org.
Actor	User (logged in)
Pre-Condition	- The user has an OSM account and is logged in - The user has one or more unread messages
Main Success Scenario	1. The user can see that they have new messages. 2. The user clicks on the message button. 3. The user is redirected to osm.org where they can read the message.
Post-Condition	The unread message counter is set back to 0.
Alternative Scenario	-

8.2.6 UC6: Check opening hours

Description	The user wants to signalize that they have checked the opening hours of an existing business.
Actor	User (logged in)
Pre-Condition	- The user has an OSM account and is logged in
Main Success Scenario	1. The user selects an existing business. 2. The user clicks on the edit button. 3. The user clicks on the ”Checked opening hours” button.
Post-Condition	The tag ”check_date:opening_hours” will be set to the current date, signalizing that the opening hours have been checked.
Alternative Scenario	There are no opening hours defined for the selected business so there is only the ”Save” button instead.

8.2.7 UC7: Add business to Watch List manually

Description	The user wants to add a business to their watch list manually.
Actor	User (logged in)
Pre-Condition	- The user has an OSM account and is logged in
Main Success Scenario	1. The user selects an existing business. 2. The user clicks on the ”Add to watch list” button.
Post-Condition	The business is added to the watch list.
Alternative Scenario	There business is already on the watch list, in which case the button ”Add to watch list” is disabled.

8.3 Non-Functional Requirements

The following non-functional requirements (NFR) are defined during the elaboration phase of the project. They principally define how the product should be in the end. The list is based on the ISO 25010 standard [11]. The NFRs from the previous work [13] were taken into account and integrated into the following ones.

8.3.1 Performance Efficiency

ID:	NFR-1
Requirement:	Tile loading should take place in a reasonable amount of time.
Trigger:	Annoyed users due to long loading times.
Reaction:	Use of reliable and fast web hosting and optimized database.
Measure:	Tiles should load in no longer than 5 seconds.

Table 8.2: Non-functional requirement 1

ID:	NFR-2
Requirement:	Loading of blue bubbles for businesses should take place in a reasonable amount of time.
Trigger:	Annoyed users due to long loading times.
Reaction:	Use of reliable and fast web hosting and optimized database.
Measure:	Bubbles should load in no longer than 5 seconds after zooming in.

Table 8.3: Non-functional requirement 2

ID:	NFR-3
Requirement:	The database has enough capacity.
Trigger:	Database can no longer hold new data in case of overflow.
Reaction:	Define a database able to hold enough entries.
Measure:	The database should be able to manage up to 10’000 entries in the watch list.

Table 8.4: Non-functional requirement 3

8.3.2 Usability

ID:	NFR-4
Requirement:	The application should be user-friendly.
Trigger:	Applications with little user-friendliness are reluctantly used by the user.
Reaction:	Before deployment, the application should be tested for usability.
Measure:	Three out of four test users are asked to rate the application’s user interface (categories: layout, responsiveness, intuitiveness) with a cell phone with at least 7 out of 10 points, with 10 being the best rating.

Table 8.5: Non-functional requirement 4

ID:	NFR-5
Requirement:	Saving of a business should be efficient.
Trigger:	Too much time investment in saving has a negative effect on usability.
Reaction:	The process of saving should be optimized.
Measure:	The saving of a business should take no longer than 10 seconds until it is completed.

Table 8.6: Non-functional requirement 5

ID:	NFR-6
Requirement:	The application should be able to be used on the go.
Trigger:	Map applications have few benefits if you cannot use them on the move.
Reaction:	The application works optimally on mobile devices.
Measure:	Usability testing is done early so that feedback can be incorporated into implementation.

Table 8.7: Non-functional requirement 6

8.3.3 Reliability

ID:	NFR-7
Requirement:	Errors must not cause the application to crash.
Trigger:	Application crashes due to an error.
Reaction:	Implement error handling.
Measure:	Errors should not generate system errors, but show an error message and reset the system to the previous state.

Table 8.8: Non-functional requirement 7

ID:	NFR-8
Requirement:	Errors must be visible.
Trigger:	Errors occur.
Reaction:	Implement error logging.
Measure:	Every error should be logged in the system.

Table 8.9: Non-functional requirement 8

8.3.4 Maintainability

ID:	NFR-9
Requirement:	Backend accesses must be tested.
Trigger:	Certain functions do not work as desired.
Reaction:	Testing the backend API.
Measure:	The backend API should be tested by API-testing tools.

Table 8.10: Non-functional requirement 9

ID:	NFR-10
Requirement:	New implementations must be tested.
Trigger:	New implementations lead to errors.
Reaction:	Testing new implementations.
Measure:	New implementations should be tested by automated testing tools.

Table 8.11: Non-functional requirement 10

ID:	NFR-11
Requirement:	The application should be easy to develop further.
Trigger:	The implementation structure confuses developers and they need much more time for further development.
Reaction:	Modular extension of the implementation structure.
Measure:	Business logic in the backend should be implemented in a modular way so that it can be easily can be extended.

Table 8.12: Non-functional requirement 11

8.3.5 Portability

ID:	NFR-12
Requirement:	The application should run on multiple web browsers.
Trigger:	Main web browsers in the mobile space.
Reaction:	Use of a cross-platform framework.
Measure:	The application should run on Google Chrome, Firefox and Microsoft Edge.

Table 8.13: Non-functional requirement 12

9. Analysis

In this chapter the domain model is discussed.

9.1 Domain Model

The domain model was taken from the bachelor thesis [17]. Since there were no changes in the structure, it did not need to be adapted.

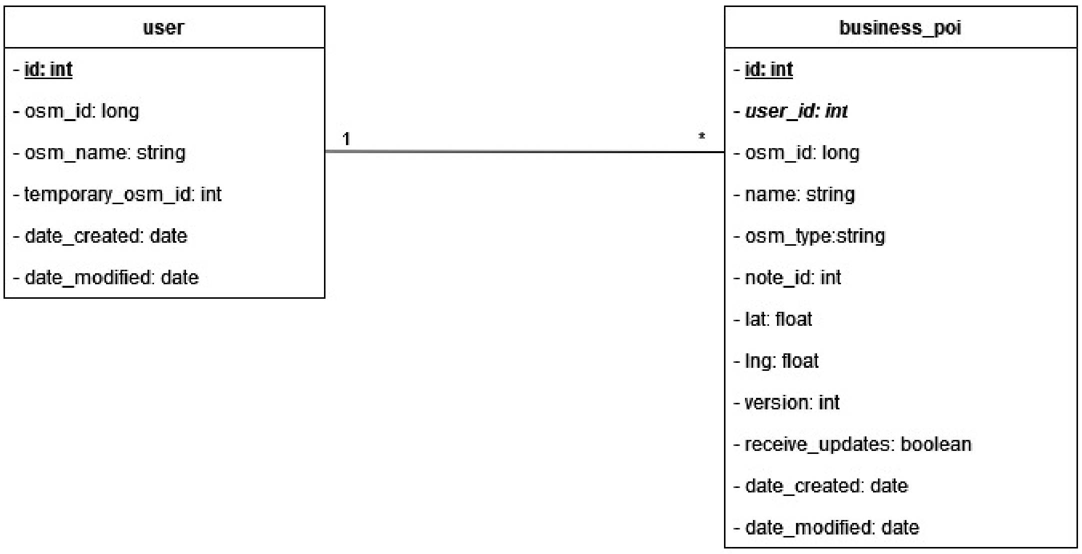


Figure 9.1: Domain model of OSMyBiz.

The backend is only used to save nodes created or edited by the user. In the user table, only the username and the OSM ID are persisted for identification.

Since the login is done via OSM OAuth, no attributes are needed for email addresses or passwords. To uniquely identify a node and associate it with the corresponding OSM data, the OSM ID is used as the primary identifier. Edited or newly created OSM nodes are stored in the node table.

10. Design

The software architecture is described below. In addition, there is a description of the process of creating the UI design.

10.1 Architecture

As an existing project, OSMyBiz also had an existing architecture. It generally follows the classic three-tier architecture. The presentation layer consists of its vue-based web app, the application layer of the Python/Flask-based backend, and the data layer consists of a PostgreSQL database.

The web app interfaces directly with OSM, Overpass, and Nominatim for most functionality. In this regard, OSMyBiz is more akin to a single layer above the APIs offered by the third-party services it uses.

The backend and database are used for the watch-list-related functionality of the project, where a client-side implementation would have significant issues due to limitations related to data storage and Cross-Origin HTTP Requests.

To simplify the setup for development and deployment, the application uses Docker containers to provide isolated environments for each part of the application.

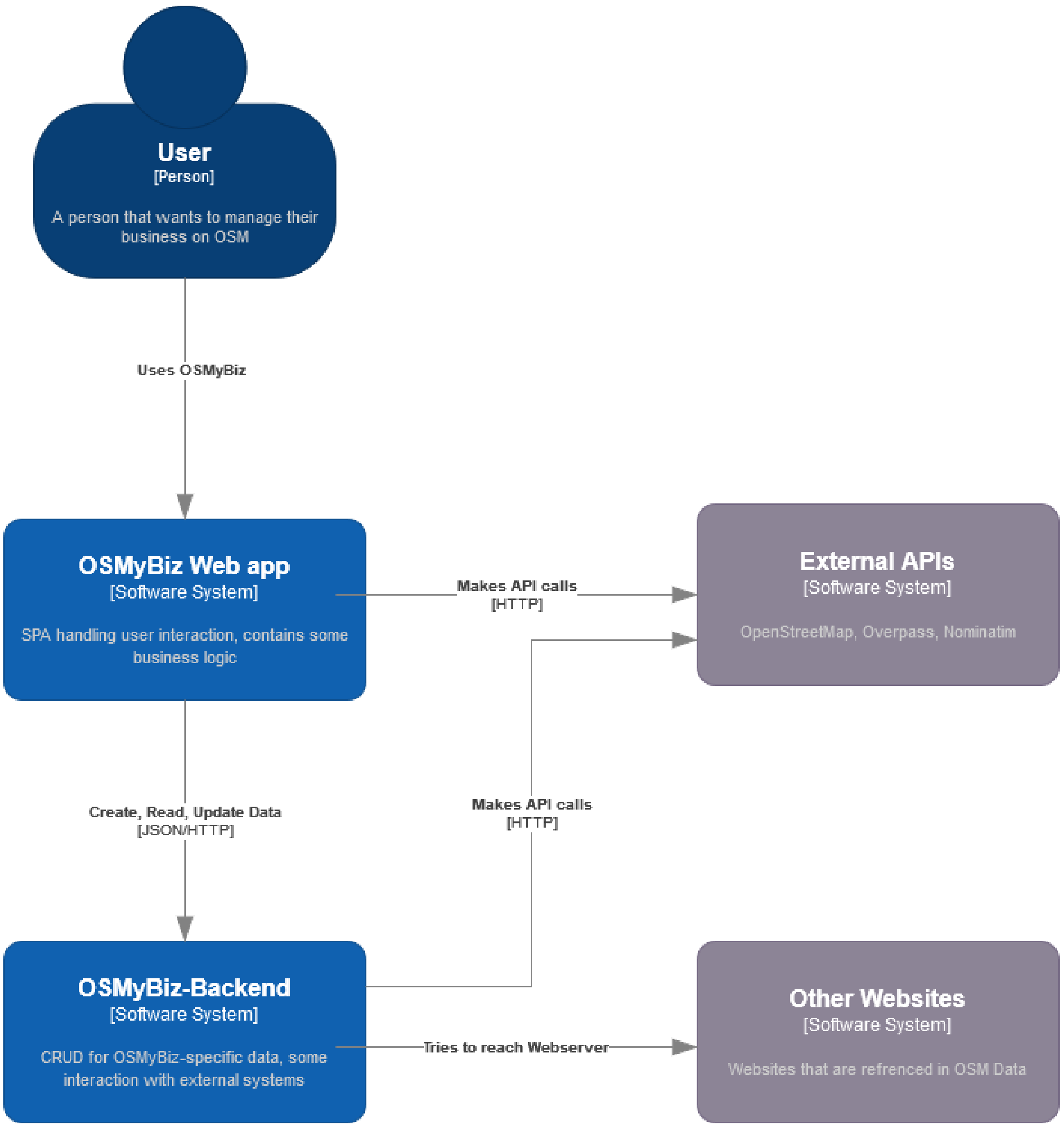

User
[Person]
A person that wants to manage their business on OSM
Uses OSMyBiz
OSMyBiz Web app
[Software System]
SPA handling user interaction, contains some business logic
Makes API calls
[HTTP]
External APIs
[Software System]
OpenStreetMap, Overpass, Nominatim
Create, Read, Update Data
[JSON/HTTP]
Makes API calls
[HTTP]
OSMyBiz-Backend
[Software System]
CRUD for OSMyBiz-specific data, some interaction with external systems
Tries to reach Webserver
Other Websites
[Software System]
Websites that are refrenced in OSM Data

Figure 10.1: C4 System Context Diagram of OSMyBiz

10.1.1 Frontend

The frontend is a Single Page Application written in TypeScript and uses the Vue 3 Framework. During development, Vite provides a web server for easy development. For deployment, Vite builds a static set of files that can be delivered to the client by any web server capable of serving static content.

Views The application consists of 3 views, each associated with a route. A view is responsible for combining components to create the page the user is presenting. OSMyBiz consists of three views: a ”Map View”, a view for business details, and a ”special” landing view used by OSMAuth for the authorization flow.

Components Using Vue’s Single File Components, the web application is split into separate reusable parts. A component contains UI content and logic that can be used in other components and views to compose the final User Interface.

Stores To manage the application state, Pinia is used to create various stores for the state, encompassing things like the currently selected locale, the current location of the map, the watch list, and others. Components can then import relevant stores to receive the data.

Services The services (located under ./api in the frontend) are an abstraction layer over the APIs used by OSMyBiz, for example handling the conversion from XML to JavaScript objects and back.

10.1.2 Backend

The backend is a simple Python/Flask server that provides a REST API for the frontend to create, read, update, and delete data from the database and implements some functionality that the client application cannot handle related to the reachability of websites. It is also responsible for database schema migrations.

10.1.3 Database

A simple PostgreSQL database is used for the application.

10.1.4 Deployment

An nginx container serves the web app files and acts as a reverse proxy for the OSMyBiz-API. It is also responsible for the SSL/TLS encryption.

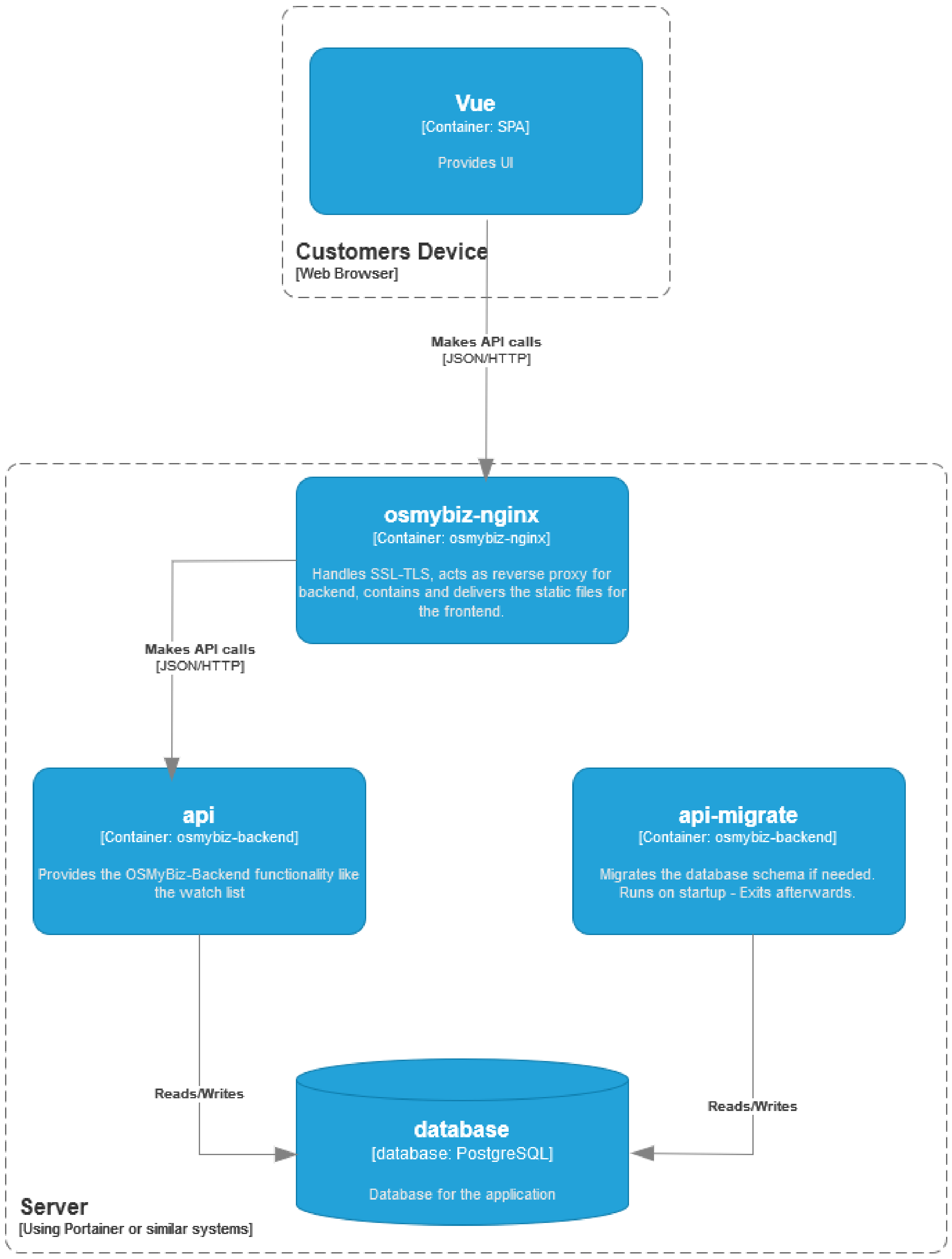

Vue
[Container: SPA]
Provides UI
Customers Device
[Web Browser]
Makes API calls
[JSON/HTTP]
osmybiz-nginx
[Container: osmybiz-nginx]
Handles SSL-TLS, acts as reverse proxy for backend, contains and delivers the static files for the frontend.
Makes API calls
[JSON/HTTP]
api
[Container: osmybiz-backend]
Provides the OSMyBiz-Backend functionality like the watch list
api-migrate
[Container: osmybiz-backend]
Migrates the database schema if needed. Runs on startup - Exits afterwards.
Reads/Writes
Reads/Writes
database
[database: PostgreSQL]
Database for the application
Server
[Using Portainer or similar systems]

Figure 10.2: C4 Deployment Diagram of OSMyBiz

10.2 UI Design

For the user interface design, research was done on existing solutions. Using that as inspiration, a wireframe was created and it was shown to several people to test and see if they understood it.

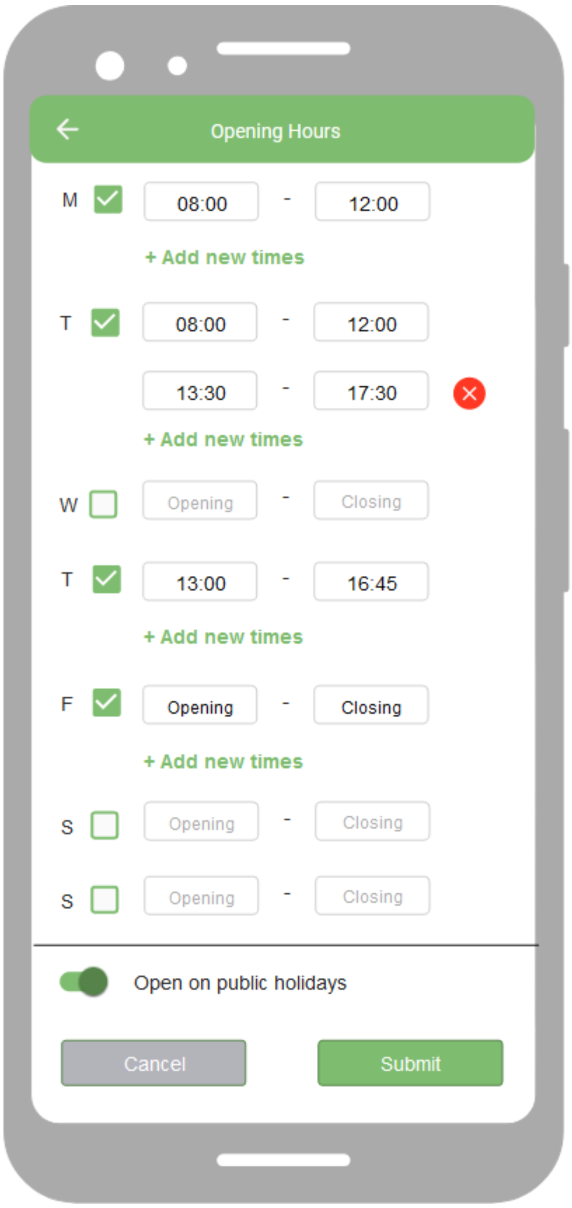
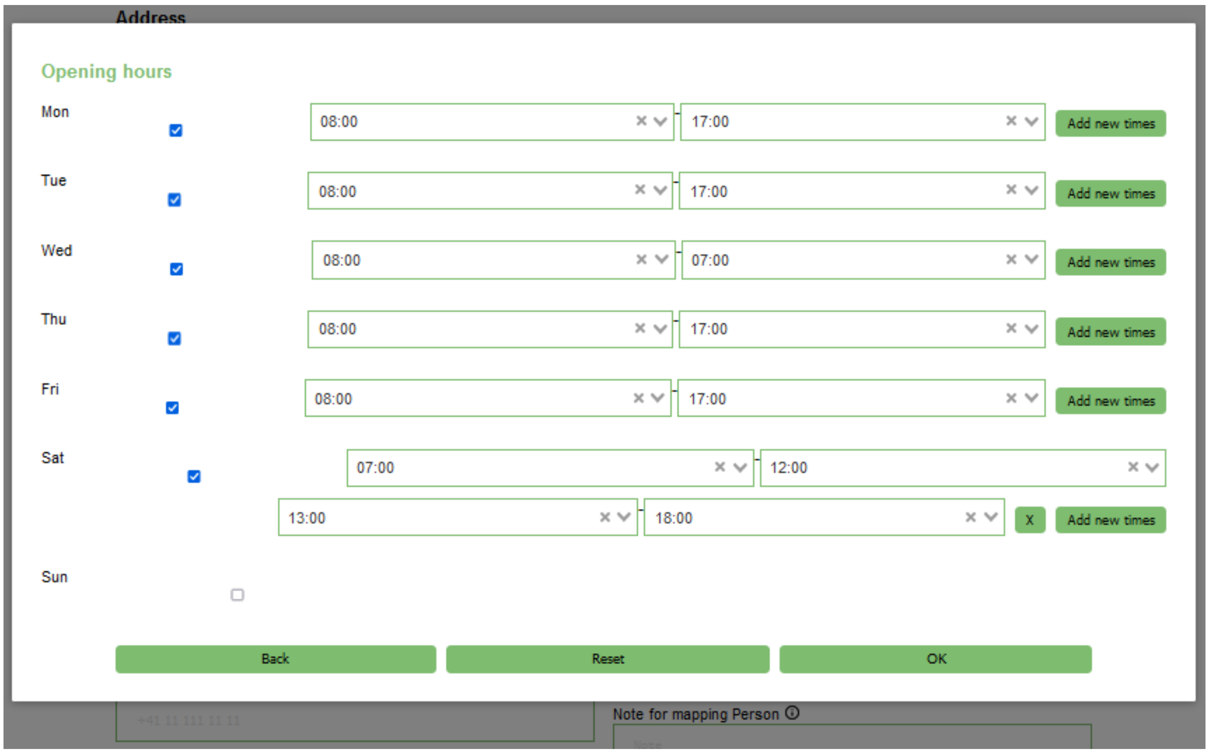


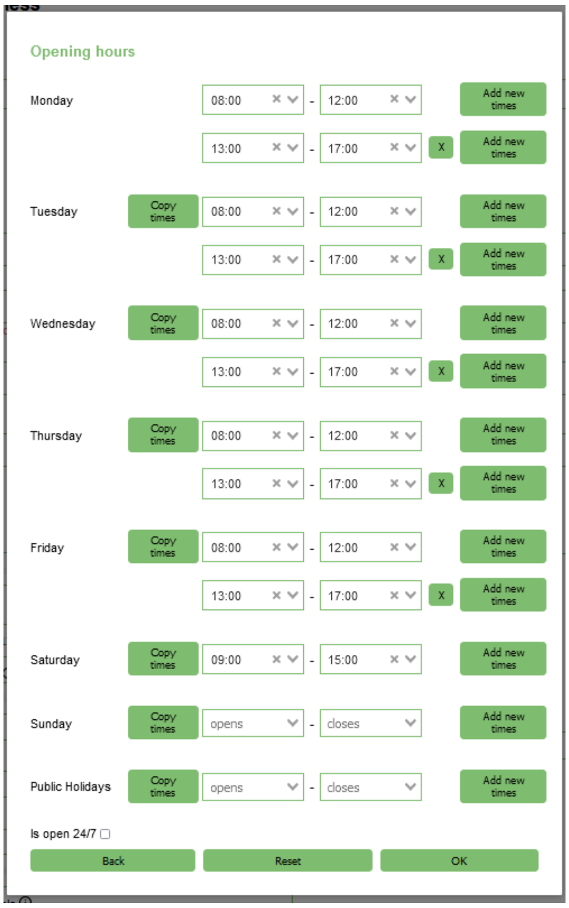
Figure 10.3: Wireframe of the opening hours editor in responsive view.

After the wireframe testing, the development on a first prototype in OSMyBiz has started. During the process loop between implementation and usability testing, the development team discussed the improvements from the usability tests and adapted them into the user interface.



Mon	☑	08:00	17:00
Tue	☑	08:00	17:00
Wed	☑	08:00	07:00
Thu	☑	08:00	17:00
Fri	☑	08:00	17:00
Sat	☑	07:00	12:00
		13:00	18:00
Sun	☐		

Figure 10.4: First prototype of the opening hours editor in OSMyBiz.

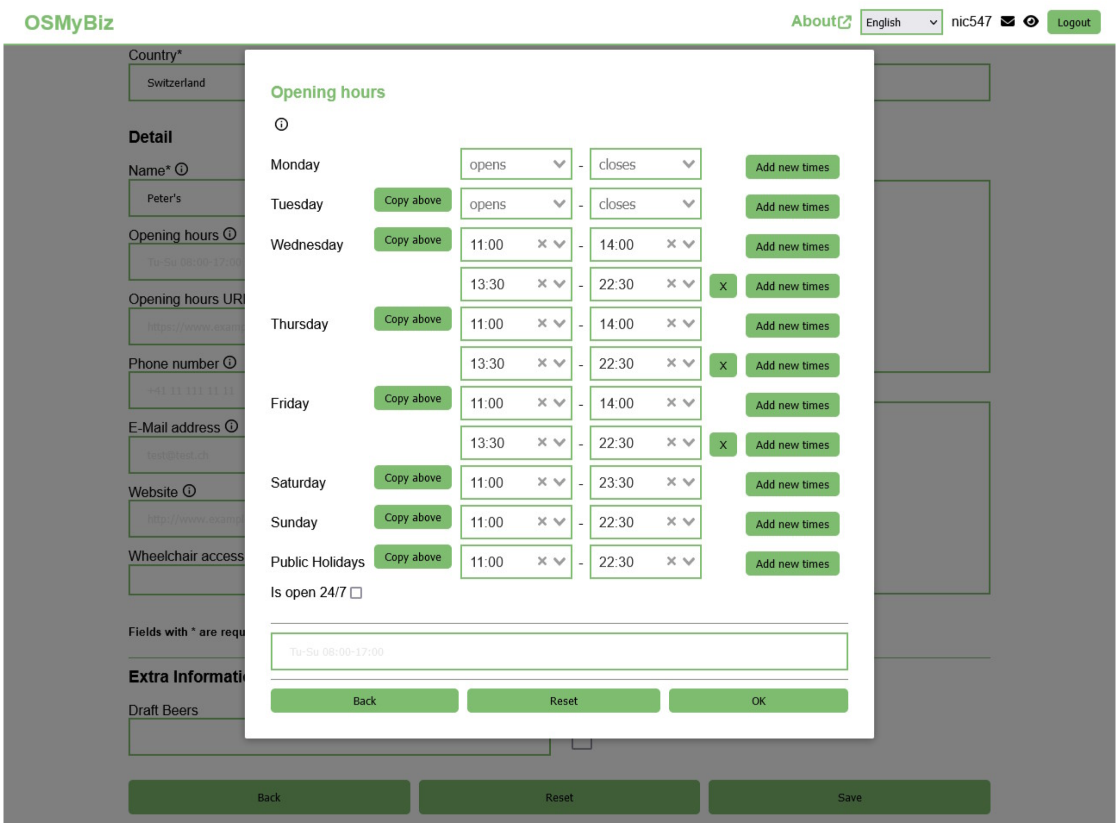


Monday	08:00	-	12:00
	13:00	-	17:00
Tuesday	08:00	-	12:00
	13:00	-	17:00
Wednesday	08:00	-	12:00
	13:00	-	17:00
Thursday	08:00	-	12:00
	13:00	-	17:00
Friday	08:00	-	12:00
	13:00	-	17:00
Saturday	09:00	-	15:00
Sunday	opens	-	closes
Public Holidays	opens	-	closes

Is open 24/7 ☐

Figure 10.5: Later version of the opening hours editor.

In the end, an opening hours editor could be created which is understood by most of the testers. It implements the most important functions so that the opening hours can be entered with little effort.



Opening hours

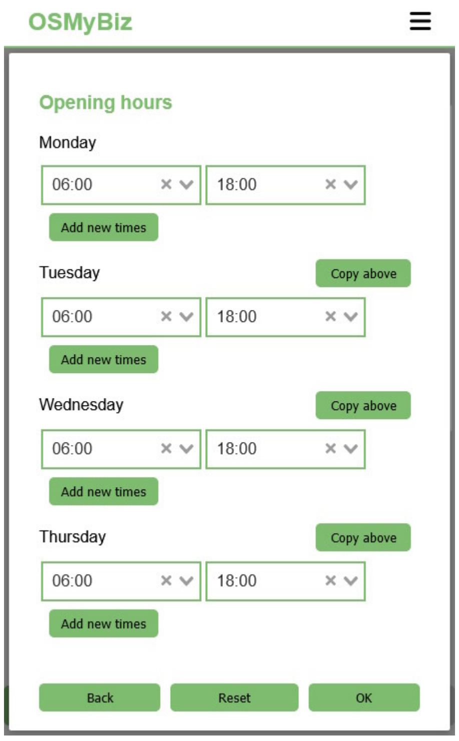
Monday		opens	-	closes	Add new times
Tuesday	Copy above	opens	-	closes	Add new times
Wednesday	Copy above	11:00	-	14:00	Add new times
		13:30	-	22:30	X Add new times
Thursday	Copy above	11:00	-	14:00	Add new times
		13:30	-	22:30	X Add new times
Friday	Copy above	11:00	-	14:00	Add new times
		13:30	-	22:30	X Add new times
Saturday	Copy above	11:00	-	23:30	Add new times
Sunday	Copy above	11:00	-	22:30	Add new times
Public Holidays	Copy above	11:00	-	22:30	Add new times

Is open 24/7 ☐

Tu-Su 08:00-17:00

Back Reset OK

Figure 10.6: Actual version of the opening hours editor on a desktop device.



Opening hours

Monday	06:00	18:00	Add new times
Tuesday (Copy above)	06:00	18:00	Add new times
Wednesday (Copy above)	06:00	18:00	Add new times
Thursday (Copy above)	06:00	18:00	Add new times

Back Reset OK

Figure 10.7: Actual version of the opening hours editor on a mobile device.

11. Implementation

This chapter is about the implementation. It covers the implemented features as well as bugs that occurred during development. It also covers enhancements, proof of quality, and the evaluation of non-functional requirements.

11.1 Features

11.1.1 Upstreaming and Deployment

Due to time constraints, the changes from the previous student research project for OSMyBiz had not been contributed to the upstream repository. As this bachelor thesis is a continuation of that project, the changes were upstreamed during this project. Following that, issues with the deployment pipeline were identified and fixed so that an up-to-date version of the application could be deployed to the staging environment.

11.1.2 Directly Edit Nodes

So far, OSMyBiz has generally not edited POIs directly on OSM, instead posting suggested changes via the notes functionality of OSM. Only newly created businesses with a known category were directly added to OSM. Another contributor to OSM needed to come across the note, make the changes suggested by the OSMyBiz user in another editor, and then resolve the note.

OSM uses Optimistic Locking to prevent conflicting modifications. For this, each element has a version attribute. When updating an existing element, this attribute has to be sent along. If the version no longer matches because another update has already incremented the version, the update will fail.[24]

Previously OSMyBiz used the data loaded from Overpass or from the watch list to edit a business. Data from Overpass does not contain the version attribute, so we need to load the POI from OSM. Both data loaded from the watch list and Overpass could lag behind, which is also no longer an issue when loading the POI from OSM.

OSMyBiz uses the OSMChange format to upload its changes. The existing code could simply be extended to support ”modify” operations. Elements that do not yet exist on OSM have an ID that is zero or negative, and existing elements have an ID that is a positive integer. This makes it easy to differentiate between the two cases, so until creating the OSMChange functions do not need to care about whether they are updating or creating an element.

11.1.3 URL-Check for Watch List

Checking if a given URL is reachable and returns some non-error content is a relatively simple task in most programming languages; the Same-Origin Policy of Web browsers makes this task rather complicated.
The usual solution to issues with the Same-Origin Policy (SOP) is to use the correct Cross-Origin-Resource Sharing (CORS) Headers. Unfortunately, this is not an option

in our case, as the URLs we like to check on are not on servers under our control.

The initial approach was to try and use embedding to get around the Same-Origin-Policy. Embedding cross-origin resources is generally allowed due to historical reasons and being relatively lower risk than regular reads. There are some approaches involving the image element and onLoad/onError combined with timeouts to determine whether or not a web server does respond to a particular URL. However, browser manufacturers seem to have ”tightened the hatches” so to speak, as solutions that seemed to have worked previously were reportedly no longer working.[34] As there did not seem to be a reliable method to check for a reply for an arbitrary URL, we have abandoned this approach after discussing this with the Product Owner.

The second approach was to check the URLs on the server. This eliminated all issues with the SOP and worked well. A request has to specify the OSM element and the URL. The URL is used to check a least-recently-used cache; if the URL has been checked in the last 24 hours, the cached result is returned. This was done to reduce the number of requests in case multiple POIs share the same website. On the other hand, for URLs not in the cache, the URL is loaded from OSM before actually trying to connect. Only a boolean is returned, indicating whether the backend got a 200 OK response. This is an attempt to address the significant issue with this approach: This approach is an SSRF vulnerability.[33] In addition to not taking the URL directly from the client, the application also only uses GET-Requests, does not use schemes other than HTTP/HTTPS, and does not use non-standard Ports. However none of these precautions can prevent a motivated attacker from abusing this feature.

11.1.4 Opening Hours Editor

The format of the opening_hours tag takes some getting used to. Therefore an opening hours editor has been developed, which allows to convert simple opening hours into an accepted string format. The editor can handle input with a maximum of 3 different opening hours per day. It also converts the opening_hours tag back to the editor’s format. So even existing businesses can easily change their opening hours. In case an existing opening hour string is too complex, the editor will keep the original string in the special string field. This ensures that existing opening hours will not be broken by the opening hours editor. The string can still be edited directly. Another useful feature is the ”Copy Above” button. This allows you to quickly copy the previous day’s entries. With the features of the Opening Hours Editor, the inhibition to create a business on OSM decreases a little more.

11.1.5 Automatic check_date tag for opening_hours and opening_hours:url

As the core of this project revolves around the management of the opening hours, it is important that the users have a general idea when the opening hours of a business have last been checked.

The tag ”check_date” [36] serves this purpose. Using this tag as a prefix, it is possible to document when a certain tag was last checked, in this case the opening_hours tag. As it is also possible for the user to only specify the opening_hours:url tag, the prefix can be used here as well.

This feature introduces the automatic addition of the tags ”check_date:opening_hours” and ”check_date:opening_hours:url” whenever a business is created or edited and there is a corresponding tag for the opening_hours or opening_hours:url set.
The value will be set to the date on which the business is created or edited.

It is not certain if the user has checked the opening hours during the edit of the business if there is no real change. Since it would not be user-friendly if the user has to specifically tell OSMyBiz that he has not checked the opening hours during each edit, it is better to just assume that the opening hours have been checked implicitly.

11.1.6 Check Opening Hours of existing businesses

With the addition of the feature for adding the automatic check_date tag 11.1.5, the user can see when a business' opening hours have last been checked.
However, it certainly does not look reliable if the opening hours have last been checked a few years ago.

The user needs to be able to just check the business' opening hours without having to make other changes to the business. As such, this feature introduces the possibility to check the opening hours via the regular edit functionality.
With this feature, the user can simply open the edit dialog, check the opening hours and then use the new button ”Checked opening hours”.
This will simply update the date on the check_date tags to the current date.

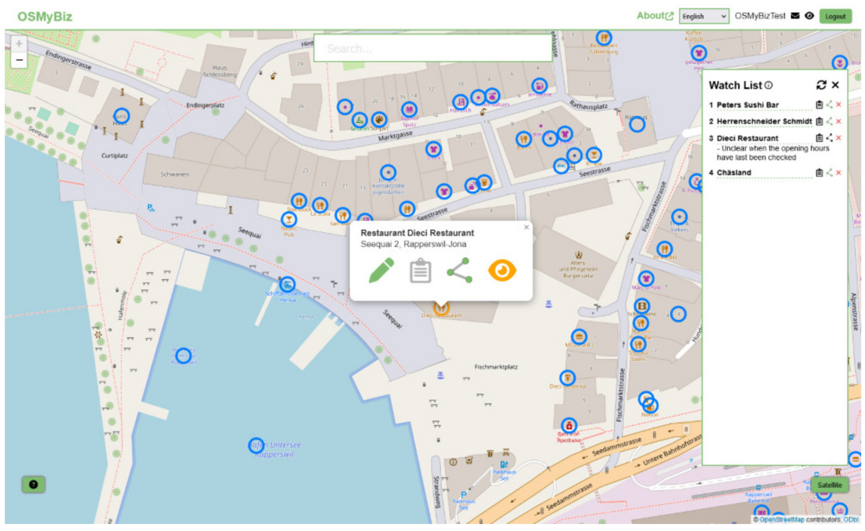


Figure 11.1: OSMyBiz on a Desktop device, notifying the user about possibly outdated opening hours.

11.1.7 Upgrade of Backend Dependencies

To improve the quality of the application as well as eliminate possible security threats, it is important to update the backend and its dependencies. During this project, the used Python version has been upgraded from 3.6 to the current 3.11.

Using the pip-compile command from the pip-tools library, the required dependencies are upgraded to the most recent versions that are compatible with the Python version

3.11. As OSMyBiz used the no longer supported MigrateCommand [32] from the flask-migrate library to handle its database migrations, this upgrade led to the database migrations no longer working.

It seems that with the newer Python version, it is now recommended to switch to the Flask CLI to handle migrations.
As a result, along with the upgrade of the backend dependencies, the backend migrations are also upgraded to now using the Flask CLI for the migrations.
After the switch, the database migrations worked correctly again.

11.2 Bug fixes

During this project, various bugs that have accumulated over the years were found. To ensure a good user experience and improve the software quality, any potentially prohibitive that were found are addressed.
There were no security-threatening bugs found, but any such bugs would have been addressed with higher priority.

Every bug that was found, is also tracked in Jira. Following is a list of the bugs with their Jira-generated task number which are fixed during this project.
This list does not contain bugs which are a result of this project itself.

- **OSMBIZ-42** - The login on the staging environment does not work
- **OSMBIZ-47** - Refreshing the Watch List results in an error
- **OSMBIZ-49** - Input fields of type ”check” are not shown with a check box
- **OSMBIZ-50** - The Watch List only loads data after refreshing after a login
- **OSMBIZ-51** - The business is not shown after creation
- **OSMBIZ-52** - The Watch List does not reload after adding new elements
- **OSMBIZ-54** - The data is not updated after the edit of a business
- **OSMBIZ-55** - Map zoom is set to default after using the ”Back” button on the edit dialog
- **OSMBIZ-66** - The Unsaved Changes dialog appears randomly

11.3 Improvements

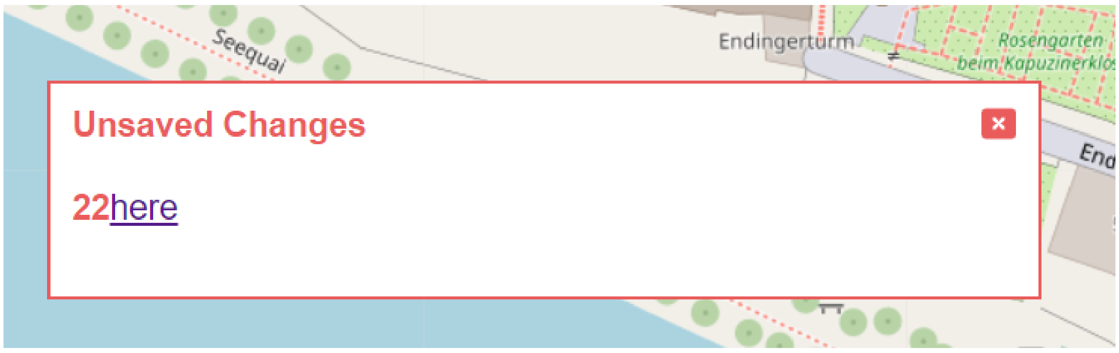
The usability testings as well as feature tests resulted not only in bugs being found, but also a variety of other design, performance and user experience issues.

Not all issues are tracked in Jira, as some are smaller improvements that were done on the go or addressed together with other issues. Following is a list of the more noteworthy improvements. These are tracked in Jira and are addressed during this project.
This list does not contain improvements on features being introduced by this project itself.

OSMBIZ-57 - Improve Styling for Unsaved Changes Dialog

The styling of the ”Unsaved Changes” dialog was initially not in line with the rest of OSMyBiz. This remained unnoticed however, as the dialog only under irregular circumstances and only was noticed after the bug OSMBIZ-66 11.2 was resolved.

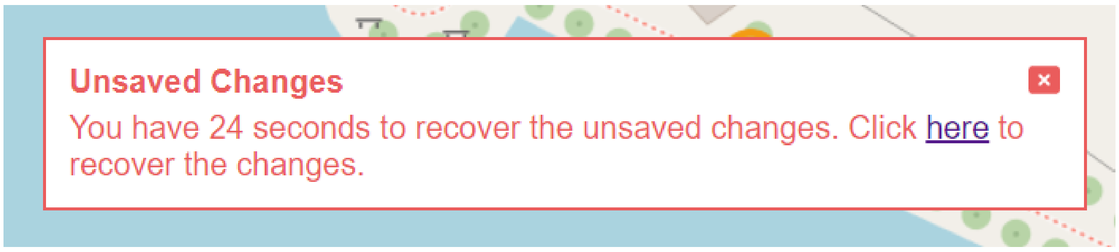
The dialog only showed the remaining time in seconds until the changes are lost and a link with the words ”here” which opened the edit dialog with the unsaved changes.


Unsaved Changes

22here

Figure 11.2: Unsaved Change Dialog before

This dialog has been changed to showing a more meaningful text for the user, still retaining the 30 seconds time to recover the unsaved changes.


Unsaved Changes

You have 24 seconds to recover the unsaved changes. Click here to recover the changes.

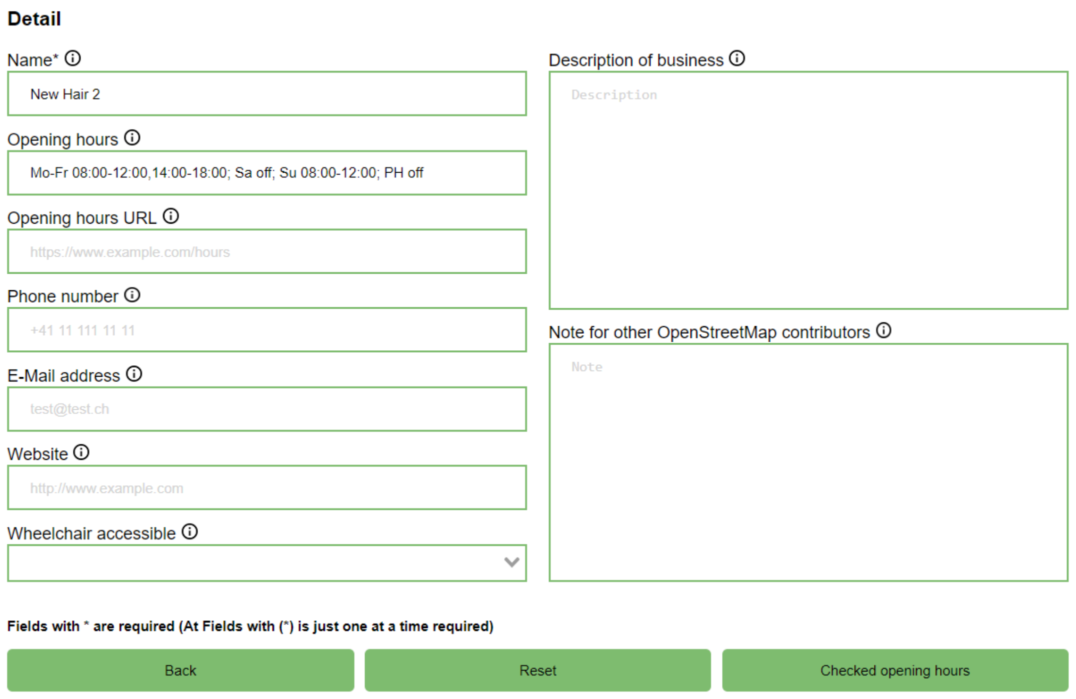
Figure 11.3: Unsaved Change Dialog after

OSMBIZ-59 - Moving of info about required fields below address

When a business is created or edited, the detail page is shown as a form to input the necessary data. Initially, there was a text label that informs the user about the required fields and that either the ”Street” or the ”Place” field have to be filled out.

This info was placed almost at the bottom of the form, after the user has gone through the form’s fields. This led to many confused users during the usability testing.

This info has been moved to the start of the form’s fields.



Detail

Name* ⓘ	New Hair 2
Opening hours ⓘ	Mo-Fr 08:00-12:00,14:00-18:00; Sa off; Su 08:00-12:00; PH off
Opening hours URL ⓘ	https://www.example.com/hours
Phone number ⓘ	+41 11 111 11 11
E-Mail address ⓘ	test@test.ch
Website ⓘ	http://www.example.com
Wheelchair accessible ⓘ	
Description of business ⓘ	Description
Note for other OpenStreetMap contributors ⓘ	Note

Fields with * are required (At Fields with (*) is just one at a time required)

Figure 11.4: Info label before

OSMBIZ-70 - Various feedback from Simon Poole

During the first testing of the newly upstreamed student research project changes, Simon Poole (developer of Vespucci 4.5 and president of the Swiss OSM Association) reported various smaller issues.

Following is the list of those issues which were collected and addressed in this improvement.

- White space is missing above ”Edit Business”
- Re-word ”Note for mapping Person”

OSMBIZ-73 - Change form info label text to more meaningful text

The info text in the form of the detail page as mentioned in the previous improvement 11.3 was confusing the users during the tests.
It was unclear what had to be filled in the fields with ”(*)”.

This info text has been changed to ”Either ’Street’ or ’Place’ has to be filled”.

11.4 Proof of Quality

To ensure a high quality, it is important that the work is often tested and reviewed. The use of state-of-the-art CI/CD tools like GitLab should guarantee that the produced work builds properly and automated tests run without failure.
Additionally, GitLab as a Git host also serves as a versioning tool so that the work is not lost.

Code reviews

Every feature requires its own branch and has to be approved by a second developer. This serves as a code review and makes sure that no unreviewed code is pushed to the develop branch.

Development guidelines

Certain guidelines are defined so that the project members are on the same page and can produce more maintainable and high-quality code. In the following section, these guidelines are explained further.

Feature branches and merge process

As mentioned in the section above 11.4, every feature requires its own branch.
The naming of the feature branch is always named "<Jira Issue number>_<Issue name>" using underscore casing for the issue name, e.g. "OSMBIZ-57_improve_unsaved_changes_styling".
Before the feature can be merged into the develop branch, it has to be reviewed by a second developer and merged into its own dedicated merge branch.
This merge branch will then result in a merge request into upstream, during which it will be reviewed again.

Coding conventions

In general, the best practices and clean code principles of each technology in the stack should be followed.
For TypeScript, the best practices guidance [30] from AWS is to be followed.
Code should be formatted and linted properly before committing.

11.4.1 Testing

The testing covered various aspects to ensure the quality, reliability and usability of the software.

Manual Testing

A key component of the testing process was manual testing. These tests were used to verify specific functionalities and user interactions. Manual testing allowed for the discovery of potential bugs or inconsistencies as there were no automated tests. Various test cases were defined and executed to ensure that the software met requirements and provided a smooth user experience. The test were performed according to the use cases in 8.2.

Usability Testing

Usability tests were conducted to evaluate the usability of the software solution. This involved inviting potential users to use the software in a real-world environment and provide feedback. The tests included evaluation of the user interface, navigation, and other important aspects. User feedback was collected, analyzed, and used to improve the software and enhance the user experience. The tests performed are listed in the appendix B. The tests were conducted in German, as all test subjects are proficient in the German language.

Code Reviews

Code reviews were another important part of the testing process. During these reviews, the development team took turns to review merge requests and identify potential bugs, poor coding practices, or opportunities for optimization. Regular code reviews ensured high code quality and identified potential issues early on.

Linting

In addition to the code reviews, linting tools were used on the frontend to perform automated code analysis and ensure that the code met the defined standards. No linter tool was used on the backend, as the backend was not drastically rebuilt in this project.

CI/CD

Another part of the testing process was the use of Continuous Integration (CI) and Continuous Deployment (CD). The CI/CD pipeline first builds the front-end project. If this was successful, it starts two test jobs. One is for linting the frontend project, the other is for code linting in the backend with Flake8 and run tests with pytest. Unfortunately, no tests were implemented. Because of the simple backend and other more important tasks, the development team decided to leave it as it was. Had there been more time, more consideration would have been given to using tests in the CI/CD pipeline.
Unit testing was not performed. This was due to lack of time resources. The focus was on the other types of testing to achieve the project goals.

This testing approach ensured that the developed software solution met a high quality standard. The combination of the above measures allowed the software to be tested for robustness, functionality and usability to ensure optimal performance and user satisfaction, but it still has more room for extension of the test process.

11.5 Evaluation of Non-Functional Requirements

NFR-1 - Tiles should load in no longer than 5 seconds

The loading time of the tiles was tested with the ”Performance insights” Dev Tool of the browser Chrome. It took somewhere around the one second mark to load the tiles on refresh.

This requirement is fulfilled.

NFR-2 - Bubbles should load in no longer than 5 seconds after zooming in.

The loading of the bubbles was tested with the ”Performance insights” Dev Tool of the browser Chrome. It took somewhere between one to two seconds to load the bubbles over Rapperswil once the necessary zoom was reached.

This requirement is fulfilled.

NFR-3 - The database should be able to manage up to 10’000 entries in the Watch List.

To test this requirement, an excel sheet was used to generate 10’000 insert statements for the businessPOI table, distributed over many users. The statements could then be executed without problems and the local environment still worked properly.

This requirement is fulfilled.

NFR-4 - Three out of four test users are asked to rate the application’s user interface (categories: layout, responsiveness, color, content) with a cell phone with at least 7 out of 10 points, with 10 being the best rating.

The usability tests in chapter 11.4.1 included ratings by the users. These were on average above the required 7 out of 10 points.

This requirement is fulfilled.

NFR-5 - The saving of a business should take no longer than 10 seconds until it is completed.

The saving of the business was tested with the ”Performance insights” Dev Tool of the browser Chrome. It took somewhere between one to two seconds to from clicking the ”Save” button until the success message was displayed.

This requirement is fulfilled.

NFR-6 - Usability testing is done early so that feedback can be incorporated into implementation.

The first usability testing was conducted on 15.05.2023, so there was still about a month to incorporate the feedbacks.

This requirement is fulfilled.

NFR-7 - Errors should not generate system errors, but show an error message and reset the system to the previous state.

Errors result in an error message on the frontend and do not generate system errors.

This requirement is fulfilled.

NFR-8 - Every error should be logged in the system.

There is no dedicated logging implemented for errors.

This requirement is **not** fulfilled.

NFR-9 - The backend API should be tested by API-testing tools.

There is dedicated testing of the backend API by API-testing tools implemented.

This requirement is **not** fulfilled.

NFR-10 - New implementations should be tested by automated testing tools.

New implementations are being tested manually according to the usability tests, which consist of the core functionalities.
There is however no dedicated automated testing implemented.

This requirement is **not** fulfilled.

NFR-11 - Business logic in the backend should be implemented in a modular way so that it can be easily can be extended.

The business logic in the backend is implemented in modules.

This requirement is fulfilled.

NFR-12 - The application should run on Google Chrome, Firefox and Microsoft Edge.

The creating and editing of businesses was tested on the most recent versions of Google Chrome, Firefox and Microsoft Edge, and were working correctly.

This requirement is fulfilled.

12. Results And Further Development

This chapter discusses the results obtained. There are also some suggestions for features to be added in the future.

12.1 Results

A dialog for opening hours, that is easily understood by users who are unaware of the OSM-specific format used, was implemented. This resolves the biggest hurdle for inexperienced users to use OSMyBiz.
That OSMyBiz no longer has to take a detour via notes and instead edits POIs directly is something most users of OSMyBiz probably will not notice. However, eliminating the need to copy suggested edits from a note into another editor is probably appreciated by other OSM contributors.

OSMyBiz now supports the check_date:opening_hours tag to help downstream consumers of OSM data to understand how likely it is that given data is up-to-date. On the other hand, the watch list will now tell someone managing businesses with OSMyBiz when they should check that opening_hours are still up-to-date.

While it is not entirely clear if the URL-Check for businesses on the watch list will be part of OSMyBiz at some point, it would be an exciting feature from a product standpoint. Having discovered this feature's technical and security aspects is a valuable result either way.

In addition to the more prominent features, various minor improvements and fixes were found and implemented. This includes tests with non-technical users.

All changes have been provided to the upstream project on gitlab.com, including the changes done during the previous student research project. [2]
The deployment process has been fixed so that the current version of OSMyBiz can be deployed to the staging environment.

12.2 Possibilities For Further Development

Vector Tiles for the Map Layer

Currently, the map layer of OSMyBiz uses raster tiles rendered by OSM and combines them with data from Overpass to create the interactive map. An interesting alternative is the usage of vector tiles. This would allow more control over the map rendering, allowing the replacement of the current "blue bubbles" denoting editable businesses and simply hide POIs that cannot be edited.

Automated Scraping

Currently a user has to copy data from a website by hand. Features that would help the user to automatically enter data by scraping the website of a business could help. Some businesses provide metadata on their website via the so called "Microdata" schema.

Another approach could be to try and use Machine Learning to extract information from a website.

Replacing Right Clicking

OSMyBiz uses a right click (or ”touch and hold” on a mobile device) to create a new business. During the tests with potential users, we found that this takes users some time to figure out, either by reading the provided help texts or simply trying different actions. Other approaches could be a button for creating businesses or creating businesses with left clicks that do not hit an existing business or are followed by map movement. Identifying a better approach would reduce the friction for new users and improve the first impression of OSMyBiz.

Other Features

There are various other ideas for features, for example supporting additional tags, adding more checks for items on the watch list, allowing users to sort the watch list according to different criteria or searching the watch list.

13. Project Management

This project follows the Scrum+ project model, which means that it will be divided into the four phases **Inception, Elaboration, Construction, and Transition** with an additional **End of Elaboration**.

Each sprint will have a duration of two weeks and at the end of each sprint there will be a sprint finish as well as sprint planning.
These usually take place every second Tuesday where each task will be estimated and assigned to a developer.

13.1 Milestones

The milestones are defined to make sure the project stays on track.
Following are the defined milestones of this project.

Name	Planned date	Goal
M1 - Project plan	08.03.2023	Create a complete project plan and risk analysis.
M2 - Setup development environment	12.03.2023	Every developer has setup the development environment and installed the necessary tool chain.
M3 - Requirements	22.03.2023	The GUI wireframes are created and the use cases as well as the requirements are defined.
M4 - End of elaboration	05.04.2023	The UI design should be done and the architecture should be up and running.
M5 - Alpha	03.05.2023	The first version with the core functionality for the opening hours editor should be implemented.
M6 - Beta	17.05.2023	The core functionality for the opening hours editor should work without major issues.
M7 - Abstract hand-in	31.05.2023	The brochure abstract should be handed in.
M8 - Final hand-in	15.06.2023	The final project should be handed in.

Table 13.1: Milestones

13.2 Project Organization

Product Owner / Advisor

- Stefan Keller, stefan.keller@ost.ch

Co-Advisor

- Joël Schwab, joel.schwab@ost.ch

Developers

- André Blöchlinger, andre.bloechlinger@ost.ch
- Dominic Ritz, dominic.ritz@ost.ch
- Khoa Tran, khoa.tran@ost.ch

13.3 Project Plan

The project time is divided into 15 weeks, which are distributed among the different phases. It takes the two week sprints into consideration so that ideally the phases start or end with corresponding to a sprint.

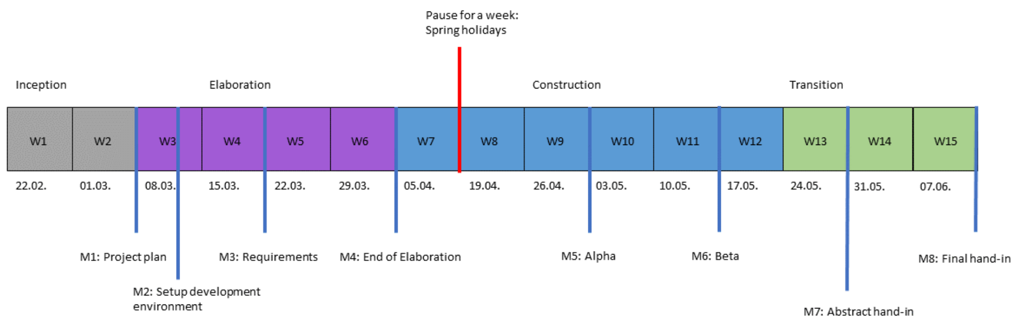


Figure 13.1: Project plan

13.4 Project Phases

13.4.1 Inception

The Inception phase starts with the first project meeting on the 22.02.2023 and ends on the 07.03.2023. In this phase, the goals and the vision of the project are defined. The first tasks are to be defined and the project plan as well as the risk analysis will be done.

13.4.2 Elaboration

This phase takes place from the 08.03.2023 until the 04.04.2023. In this phase, the requirements are defined and the design of the UI will be done.
The development environment should be ready and all the needed tool chain should be installed for development.

13.4.3 Construction

This phase takes place from the 05.04.2023 until the 23.05.2023. The development takes place in two week sprints during this time.
As a result of this phase, a working application without any major issues is expected.

13.4.4 Transition

This phase takes place from the 24.05.2023 until the final hand-in on the 16.06.2023. In this phase, final tests take place and if necessary, some smaller fixes and improvements will be done.
The application will be released on the production environment and the documentation has to be finalized.

13.5 Risks

The risks of the project as well as their estimated damage and probability of occurrence are listed in the following table.

ID	Description	Probability of occurrence (%)	Max. damage (h)	Weighting
R1	Inaccurate time estimations.	40	40	16
R2	Existing software architecture is not optimal.	10	40	4
R3	Loss of data.	10	8	0.8
R4	Domain model is not optimal.	10	20	2
R5	The selected technologies are not compatible.	20	20	4
R6	Existing application is incomprehensible for the user.	15	16	2.4
R7	The OH GUI is incomprehensible.	10	24	2.4
R8	The OSM Dev API is unavailable.	10	8	0.8
R9	The upstream is not content with the changes.	15	12	1.8
	Total		**188**	**34.2**

Table 13.2: Risk analysis

ID	Preventive measures	Measures on occurrence
R1	Additional time buffers.	Correct estimations or reduce functionality.
R2	Agile approach.	Evaluate current state, adjust architecture if necessary.
R3	Additional time buffers.	Use backups if available or worst case redo work.
R4	Agile approach.	Make adjustments to the domain model.
R5	Agile approach.	Reevaluate and choose another technology if necessary.
R6	-	Reevaluate and improve usability.
R7	Usability tests and agile approach.	Reevaluate and improve usability.
R8	Additional time buffers.	-
R9	Split tasks in smaller chunks and create Pull requests more often.	Review code and make necessary changes.

Table 13.3: Risk analysis measures

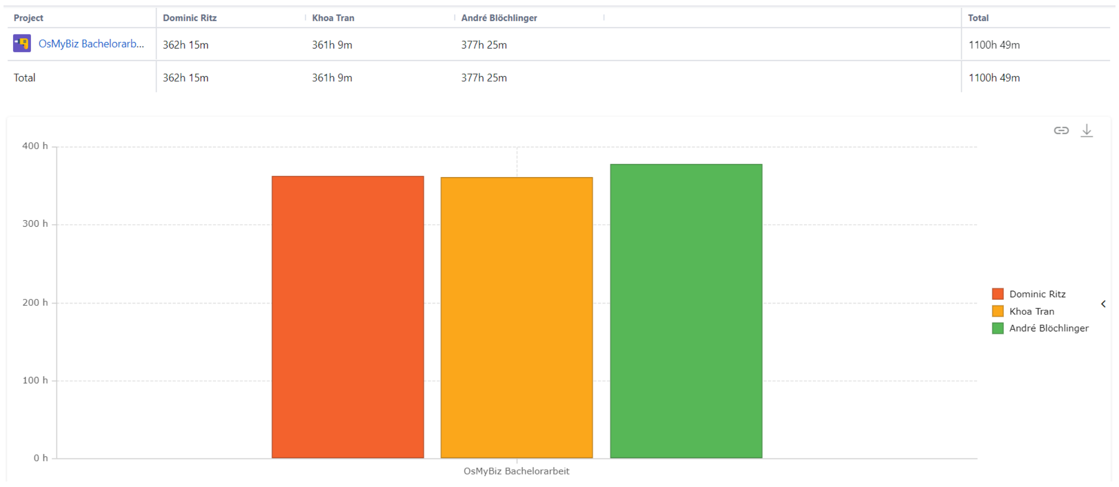
14. Project Monitoring

This short chapter provides an overview of the time spent during the project.

14.1 Time Tracking Report

The time tracking is done in Jira. According to the official ECTS formula of 1 ECTS = 30h, the total required work hours per student is 360h.

Below is an overview of the total time spent per student, generated using the ”Timesheet” app from Jira.



Project	Dominic Ritz	Khoa Tran	André Blöchlinger	Total
OsMyBiz Bachelorarb...	362h 15m	361h 9m	377h 25m	1100h 49m
Total	362h 15m	361h 9m	377h 25m	1100h 49m

Figure 14.1: Time Tracking Report

Part III

Software Documentation

15. Installation

The Documentation related to the installation of OSMyBiz is part of the repository containing the source code. [2]

16. User Manual

A separate user manual is not provided. OSMyBiz contains various helpful tooltips as well as some short introductions.

A. Directories

Glossary

- **Least-recently-used Cache** A LRU cache is a size-limited cache that discards the least recently used items first.
- **Microframework** Microframeworks usually refer to minimalistic web application frameworks. It lacks common functionalities like access control, input validation and sanitation, and so on.
- **Preset** A set of tags used to provide details about a specific type of business.
- **Same Origin Poliy** SOP is a security concept that restricts how one website from one origin can interact with other website from another origin.
- **Server Side Request Forgery** SSRF is a type of vulnerability where an attacker can cause a server to send request to a unexpected location.
- **System error** An error that leads to the crash of the application.
- **Tag** A tag is a key-value pair used by OSM to describe a feature.
- **Transpiler** A special type of compiler that does not compile to machine code but to another programming language. (e.g. TypeScript to JavaScript)
- **Upstream** Upstream is the term used to describe the original repository or maintainers of a project. Other developers can then ”fork” the project, make changes, and then ”merge” their changes back into the upstream repository. For this project the upstream repository is the repository of OSMyBiz on github.com owned by the Geometa Lab of the Institute for Software.

List Of Abbreviations

- **ACID** The four key properties that define a database transaction: Atomicity, Consistency, Isolation and Durability
- **API** Application Programming Interface
- **POI** Point of Interest
- **SOP** Same Origin Policy
- **SSRF** Server Side Request Forgery

List Of Figures

List Of Tables

Bibliography

[1] Vladimir Agafonkin. Leaflet - a javascript library for interactive maps. https://leafletjs.com/, 2023. [Online; accessed 06.06.2023].

[2] Geometa Lab at IFS. Geometa lab at ifs // osmybiz · gitlab. https://gitlab.com/geometalab/osmybiz, 2023. [Online; accessed 09.06.2023].

[3] Atlassian. Jira | issue & project tracking software | atlassian. https://www.atlassian.com/software/jira,, 2023. [Online; accessed 14.06.2023].

[4] GitLab B.V. The devsecops platform | gitlab. https://about.gitlab.com/, 2023. [Online; accessed 14.06.2023].

[5] Ian Stapleton Cordasco. Flake8: Your tool for style guide enforcement. https://flake8.pycqa.org/en/latest/, 2016. [Online; accessed 14.06.2023].

[6] datanyze. Google maps api market share and competitor report. https://www.datanyze.com/market-share/mapping-and-gis--121/google-maps-api-market-share, 2023. [Online; accessed 10.06.2023].

[7] Guillaume Chau Evan You. Home | vue devtools. https://devtools.vuejs.org/,, 2023. [Online; accessed 14.06.2023].

[8] Wikimedia Foundation. Postgres. https://de.wikipedia.org/wiki/PostgreSQL, 2023. [Online; accessed 10.06.2023].

[9] pytest-dev team holger krekel. pytest: helps you write better programs. https://docs.pytest.org/en/7.3.x/, 2015. [Online; accessed 14.06.2023].

[10] Docker Inc. Docker: Accelerated, containerized application development. https://www.docker.com/, 2023. [Online; accessed 14.06.2023].

[11] ISO25000. ISO/IEC 25010. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010, 2022. [Online; accessed 18.03.2023].

[12] johnsoncodehk. vue-tsc - npm. https://www.npmjs.com/package/vue-tsc, 2023. [Online; accessed 14.06.2023].

[13] David Kalchofner and Dominic Ritz. OSMyBiz - Company profile editor for OpenStreetMap. Student research project, OST Eastern Switzerland University of Applied Sciences Campus Rapperswil-Jona, 2022.

[14] Bruno Krebs and Juan Cruz Martinez. Developing RESTful APIs with Python and Flask. https://auth0.com/blog/developing-restful-apis-with-python-and-flask/#-span-id--why-python----span--Why-Python, 2023. [Online; accessed 10.06.2023].

[15] OSM Lab. Github - osmlab/osmauth: Authentication with openstreetmap. https://github.com/osmlab/osm-auth, 2023. [Online; accessed 14.06.2023].

[16] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[17] Max Lüthi and Simon Heller. Openstreetmap my business. Bachelor's thesis, University of Applied Science Rapperswil, 2017.

[18] Microsoft. Typescript: Javascript with syntax for types. https://www.typescriptlang.org/, 2023. [Online; accessed 06.06.2023].

[19] Microsoft. Visual studio code - code editing. redefined. https://code.visualstudio.com/, 2023. [Online; accessed 14.06.2023].

[20] Microsoft. What is windows subsystem for linux | microsoft learn. https://learn.microsoft.com/en-us/windows/wsl/about, 2023. [Online; accessed 14.06.2023].

[21] Eduardo San Martin Morote. Pinia | the intuitive store for vue.js. https://pinia.vuejs.org/, 2023. [Online; accessed 06.06.2023].

[22] OnOsm. On openstreetmap - a tool to add your business to the map. https://www.onosm.org/, 2023. [Online; accessed 10.06.2023].

[23] ESLint contributors OpenJS Foundation. Find and fix problems in your javascript code. https://eslint.org/, 2023. [Online; accessed 14.06.2023].

[24] OpenStreetMap. Api v0.6 - openstreetmap wiki. https://wiki.openstreetmap.org/wiki/API_v0.6#Version_numbers/optimistic_locking, 2023. [Online; accessed 11.06.2023].

[25] OpenStreetMap. id - friendly javascript editor for openstreetmap. https://github.com/openstreetmap/iD, 2023. [Online; accessed 10.06.2023].

[26] OpenStreetMap. Josm/plugins/openinghourseditor. https://wiki.openstreetmap.org/wiki/JOSM/Plugins/OpeningHoursEditor, 2023. [Online; accessed 10.06.2023].

[27] Pallets. Flask. https://flask.palletsprojects.com/en/2.3.x/, 2023. [Online; accessed 10.06.2023].

[28] Adiren Pavie. Yohours. https://projets.pavie.info/yohours/, 2015. [Online; accessed 27.04.2023].

[29] Inc. Postman. Postman api platform. https://www.postman.com/, 2023. [Online; accessed 14.06.2023].

[30] Amazon Web Services. Follow typescript best practices - aws prescriptive guidance. https://docs.aws.amazon.com/prescriptive-guidance/latest/best-practices-cdk-typescript-iac/typescript-best-practices.html, 2023. [Online; accessed 12.06.2023].

[31] Kshitij Raj Sharma. osm-login-python - pypi. https://pypi.org/project/osm-login-python/, 2022. [Online; accessed 14.06.2023].

[32] Spyromancer. flask - can't import migratecommand from... https://stackoverflow.com/questions/68527489/cant-import-migratecommand-from-flask-migrate, 2023. [Online; accessed 12.06.2023].

[33] OWASP Top 10 team. A10 server side request forgery (ssrf) - owasp top 10:2021. https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/, 2001. [Online; accessed 10.06.2023].

[34] Various. Is it possible to ping a server from javascript? https://stackoverflow.com/questions/4282151/is-it-possible-to-ping-a-server-from-javascript, 2013. [Online; accessed 11.06.2023].

[35] Vue-Leaflet. vue-leaflet/vue-leaflet: vue-leaflet compatible with vue3. https://github.com/vue-leaflet/vue-leaflet, 2023. [Online; accessed 06.06.2023].

[36] OpenStreetMap Wiki. Key:check date — openstreetmap wiki. https://wiki.openstreetmap.org/wiki/Key:check_date, 2023. [Online; accessed 12.06.2023].

[37] Evan You. Vue.js - the progressive javascript framework | vue.js. https://vuejs.org/, 2023. [Online; accessed 06.06.2023].