Student Research Project - Autumn Semester 2022/23

OSMyBiz

Company profile editor for OpenStreetMap

Version 1.0
Date: 2022-12-23

Authors: David Kalchofner
Dominic Ritz

Advisors: Prof. Stefan F. Keller
Joël Schwab

OST Eastern Switzerland University of Applied Sciences Campus Rapperswil-Jona

Contents

Part I

Abstract

Abstract

OpenStreetMap My Business (OSMyBiz) is an editor for the collaborative, open-mapping project OpenStreetMap (OSM). It focuses on enabling users to edit businesses on OSM without requiring knowledge about how OSM organises its data, specifically targeting new or inexperienced users. OSMyBiz got created in 2018 as a bachelor thesis project. Since then, it has only gotten minor updates and needs to catch up regarding technologies.

Since its creation, new version changes have been available for nearly all dependencies, and some are not even maintained. There have also been significant changes in the tooling used in modern web applications. OSMyBiz did not get optimised for mobile, and therefore, the usage on mobile devices was suboptimal. One of the external APIs used implemented rate-limiting, which caused constant error messages when moving around on the map. Additionally, the application has a growing list of issues and suggestions.

The application was brought up to date by migrating from Vue 2 to Vue 3, replacing Vuex with Pinia, switching from webpack to Vite and introducing TypeScript. The existing application got reworked to be responsive in a "mobile-first" approach. A caching mechanism was added to the application to reduce the load on the public overpass API instances. The implementation reduces unnecessary load on the shared infrastructure and improves user experience by reducing issues due to rate-limiting.

There are still multiple open issues, ideas and optimisation potential for OSMyBiz. This project has created a solid foundation to implement those further development options. All while ensuring better maintainability and making it simpler to modify the frontend code base.

Part II

Management Summary

Management Summary

Introduction

OpenStreetMap My Business (OSMyBiz) is an editor for the collaborative, open mapping project OpenStreetMap (OSM). It's focused on enabling users to edit businesses on OSM without requiring knowledge about how OSM organises its data, specifically targeting new or inexperienced users.

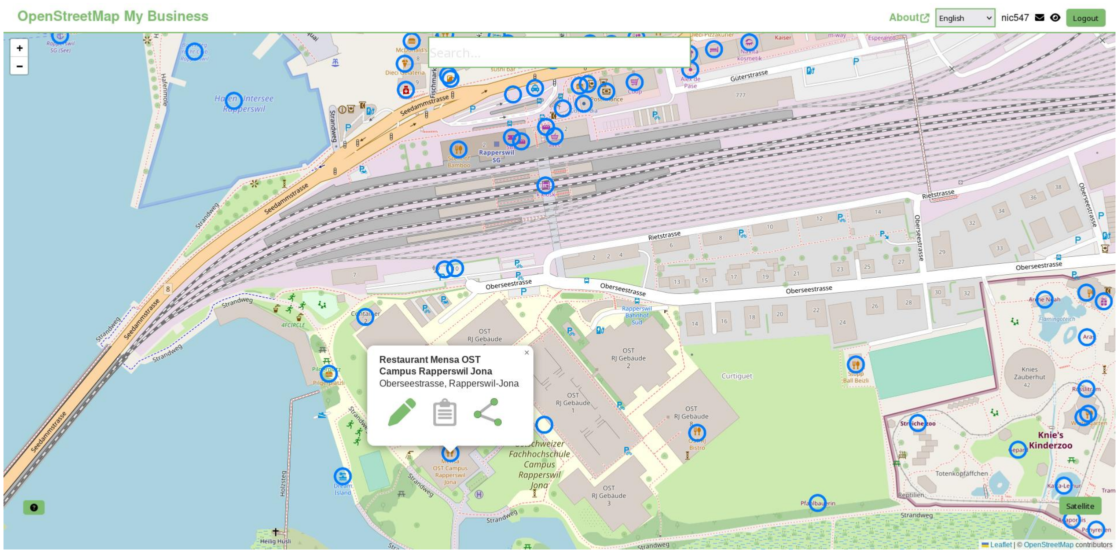

OpenStreetMap My Business
About
English
nic547
Logout
Search...
Restaurant Mensa OST Campus Rapperswil Jona
Oberseestrasse, Rapperswil-Jona
Satellite
Leaflet | © OpenStreetMap contributors

Figure 2: The desktop view of OSMyBiz. Editable businesses are loaded via Overpass API and displayed as circles on the map.

Project Goals

OSMyBiz was created in 2018 as a bachelor thesis project. Since then, it has only gotten minor updates and fallen behind in terms of technologies. Changes in an important external API lead to the application encountering significant issues due to rate-limiting. Using the application on mobile phones was out of the original project's scope, focusing instead on the classic desktop experience. Addressing this was sorely needed as traditional PCs are getting replaced more and more by mobile devices. Over the years a few other issues with the application have been reported, but not addressed so far. This project was an opportunity to address them.

Project Results

During this project, the application received a substantial overhaul. Technologically the application was brought up to date, migrating from Vue 2 to Vue 3, replacing Vuex with

Pinia, switching from webpack to Vite and many more things. Additionally, OSMyBiz now uses TypeScript rather than pure JavaScript to facilitate easier maintenance and future development. During this process a variety of small existing issues were fixed.
While a conscious decision was made not to create a new design from scratch, the existing application has been reworked to be responsive in a "mobile-first" approach, where such changes were needed.
To reduce load on the public overpass API instances, a caching mechanism was added to the application. This reduces unnecessary load on the shared infrastructure and improves user experience by reducing issues due to rate-limiting.

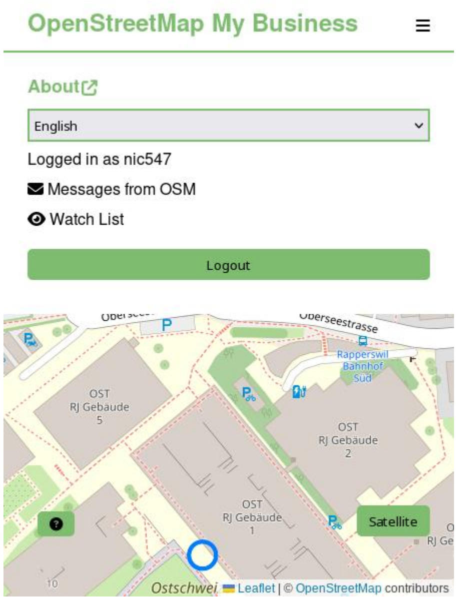
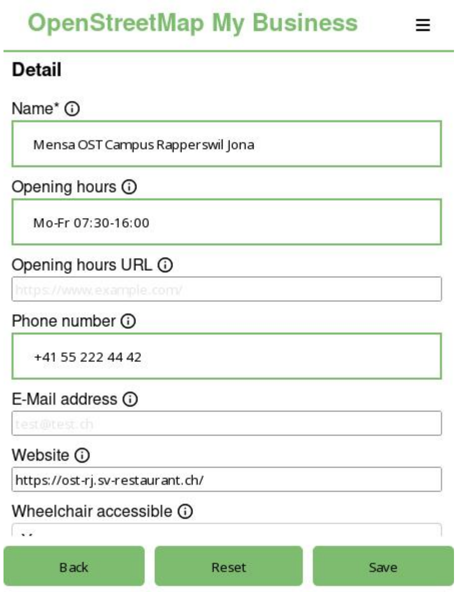


Figure 3: The new responsive header bar of OSMyBiz on a mobile device. Unlike before, all menu items are visible.



Detail

Name*	Mensa OST Campus Rapperswil Jona
Opening hours	Mo-Fr 07:30-16:00
Opening hours URL	https://www.example.com/
Phone number	+41 55 222 44 42
E-Mail address	test@test.ch
Website	https://ost-rj.sv-restaurant.ch/
Wheelchair accessible	

Figure 4: Editing a business on a mobile device with of OSMyBiz. Changes were made to improve the layout on mobile devices.

Outlook

The fundamental goals of the project have been achieved successfully. There are still many ideas for further enhancements, and we have built a solid foundation for further development that will hopefully happen as part of the follow-up bachelor's thesis. Features like a GUI editor for the opening_hours format and directly editing a business on OSM rather than just writing notes for other Mappers would further enhance the application.

Part III

Documentation

Chapter 1

Introduction

1.1 Vision

The OpenStreetMap project is a project that collects free geographical data, similar to proprietary services like Google Maps, Bing Maps or Apple Maps. Like Wikipedia, every user can contribute, and the resulting database gets licensed under an open license (The "Open Database License"). Most users contributing to the project are volunteers. The distribution of contributions follows the Pareto principle: A minority of users contributes the majority of data.

The goal of OSMyBiz is to enable people that are not regular contributors to OpenStreetMap to contribute data. The most important aspect of this is abstracting the specific schema used by the OSM community to describe something. Furthermore, OSMyBiz is not intended to be a general-purpose editor but instead focuses on business owners who want to administer the data of their businesses or businesses on behalf of others.

Time has taken its toll on OSMyBiz, and problems since its creation have made working with it difficult and, at times, impossible. Users should be able to work and enter new data on OSMyBiz without encountering constant error messages. Furthermore, the application should run on a modern tech stack again without using outdated packages and work methods.

1.2 Goals

The main goals of this project are:

- Upgrade the existing application to use up-to-date versions of tools, frameworks and libraries.
- Improve the Desktop-focused UI to work on mobile devices
- Resolve the issues with the Overpass API resulting in OSMyBiz being rate-limited and severely degrading the user experience.

Beyond those 3 main goals of the project, other improvements and features are a secondary goal.

1.3 Assignment

The web application OSMyBiz is now somewhat outdated and should therefore be renewed. It was decidedt to improve the old application through refactoring.

Refactoring

The refactoring of the old version includes:

- The web application should be responsive. Mobile-First should be used.
- The original application runs with some old dependencies. These should be upgraded to newest versions.
- As part of the refactoring, the design decisions of the old OSMyBiz should be checked and changed if necessary. This includes, for example, the architecture decision.
- Several issues have already been recorded in the current version of OSMyBiz. These should be checked and corrected if necessary.

Functionalities

The new version should continue to provide the following functionalities of the old version:

- As part of the work, a web application is to be developed that enables businesses to record all data themselves so that they are displayed in the OpenStreetMap. A note with all the data is created in the background and mapped to the OpenStreetMap.
- In addition, it should be intercepted if the address was not found correctly (e.g. if the user placed the GEO icon in front of the building instead of on top of the building) and assigned the correct address by overpass.
- The owner should also be able to manage or update existing information about their business.
- It should also be possible to log in to OSMyBiz with the OSM login so that you can display a history of the changes made.
- In addition, it should be checked for spam so that only serious companies are recorded and protection against bots should be set up. (best effort)
- It should be communicated to the creator as soon as his business is available in the OpenStreetMap (in the GUI).

Chapter 2

State of the Art

There are many editors for OSM, like JOSM, Merkaartor, Potlatch, Streetcomplete, and Maps.me. We have selected Google Business Profile as it is the original inspiration for OSMyBiz and the iD-Editor and onosm.org as we believe them to be the most relevant for our problem statement.

2.1 Google Business Profile

Formerly called "Google My Business", Google Business Profile allows verified Businesses to manage their data on Google Maps. Google Maps is currently one of the largest, if not the largest, web mapping sites.[5] Therefore, we can assume that most of our potential users have used this product before. Compared to OSM, the data on Google Maps is generally unavailable under an open license. For services relying on Google Maps APIs, OSM also has the advantage that the OpenStreetMap Foundation is located in the UK. In contrast, Google as a US company might cause issues with the EU's General Data Protection Regulation.

2.2 iD-Editor

The iD-Editor is a JavaScript-based editor for OpenStreetMap running in the browser, and it is the default editor on openstreetmap.org. It is a simple editor, allowing users to contribute without being aware of the agreed-on tagging scheme by the OSM community. The tags are not hidden away trough, and advanced users can edit them directly, for example, for more exotic tags that the iD-Editor is not aware of. The iD-Editor also allows editing any node or shape on OpenStreetMap and is not limited to businesses.

2.3 onosm.org

onosm.org is the tool most similar to OSMyBiz, focusing on businesses and hiding away the details about the tagging scheme used by the OSM Community. Unlike OSMyBiz, however, it is intended for creating new businesses on OpenStreetMap. Editing an existing business is not possible. Depending on the type of business, additional facts can get recorded about a business. For example, the tag "cuisine=italian" can describe a restaurant as an Italian restaurant, and such a key would not make sense in a doctor's office. onosm.org does not offer such context-dependent tags. OSMyBiz tries to offer appropriate options based on the type of business selected. There is also a difference in terms of UX. OSMyBiz has a much stronger focus on its map, whereas onosm.org requires explicitly searching an address before allowing a user to select the exact location on the map.

Chapter 3

Evaluation

3.1 Upgrading the Frontend to Modern Technologies

The original project has run on Node.js [11] (Node) version 8, which is EOL (End of Life, no longer supported) since 31.12.2019 [6] and has not had significant changes since its creation five years ago. As of November 2022, most current Node applications run on version 16 or higher, as those still receive security updates and are actively developed further. Multiple security risks of insecure and no longer maintained packages get displayed during its build. Furthermore, the main framework used, Vue is on version 2, but version 3 has existed for a while. This project has much extra overhead, with many unnecessary packages that bring little to the table.

The GitLab repository of the current implementation has a growing list of issues. Some of which are caused by outdated versions of used packages. The hope is that this upgrade will solve multiple problems already without having to invest too much time and then be able to focus on the remaining issues.

To ensure that the application does not have security risks from massively outdated dependencies and ensure better maintainability in the future, a more extensive version upgrade/redo was due.

3.2 Introduction of TypeScript

Although in the original thesis, a choice got made against TypeScript, it made sense to implement it in this project. Both Vite and Vue provide excellent support for TypeScript, and it can get included during the setup of the new application. Generally, using TypeScript makes sense, but for example, with OSMyBiz using multiple APIs, it is beneficial to be aware of the types.

Furthermore, some of the key benefits of using TypeScript include the following:

- TypeScript provides static type checking, which can help to catch errors and bugs early in the development process before the code is run or deployed.
- TypeScript allows developers to use modern JavaScript language features, such as classes, interfaces, and modules, making the code more organised, maintainable, and reusable.

Overall, TypeScript provides many benefits for developers who want to improve their projects' quality, maintainability, and scalability.

3.3 Reducing the Load on the Public Overpass API Instance

To find OSM nodes that represent a business that fit within the scope of OSMyBiz, the application relies on Overpass API. Overpass is a tool that allows to query the OpenStreetMap database for specific information. while Overpass can be used as a self-hosted solution, OSMyBiz relies on one of the main public instances of Overpass.

So far OSMyBiz has had a rather simple mechanism for loading the data from Overpass. After a user scrolled to a new area, a query with the current ViewBox was sent to Overpass. The result was then displayed on the map. This meant that potentially very similar areas where queried in short succession. This lead to a lot of unnecessary load on the Overpass servers and also meant that the user had to wait for the query to finish, which can be quite noticeable in urban areas with a lot of businesses.

The biggest issue with this simple approach is that the used overpass instance has pretty strict rate limiting to prevent problematic behavior.[1] This rate limiting was hit extremely quickly and easily by OSMyBiz, as it was sending a lot of queries when some scrolling was done while zoomed in far enough to display editable businesses. As this point a 15 second cool-down period is imposed on the client, which results in a error message and no editable businesses being presented to the user.

As this occurs very quickly, it effectively rendered OSMyBiz unusable in our opinion. Despite being aware of this limitation we struggled to avoid it ourselves.

Therefore it was pretty clear that reducing the amount of queries sent to Overpass API was extremely important.

Chapter 4

Concept

4.1 Frontend Upgrade

4.1.1 Analysis

An analysis was necessary to determine what packages would support the new Node version and the Vue dependencies. A lot of the implemented packages turned out to be no longer maintained and had to be replaced by alternative packages or implement the functionality itself.

4.1.2 Upgrade

The upgrade was split into multiple steps to have a better plan on how to proceed and get an overview of the progress:

Base Setup

This included setting up the base with the improved tech stack and Vite as the build tool instead of webpack.

Static File Migration

Copy all static files (i.e. locales, tags or images) that would not require changes to their new destination.

Store Replacement and Migration

Replace the Vuex store with the new Pinia store and implement it accordingly in the component and view files. Refactor and remove as much not needed code as possible.

Component and View Files

Migrate the existing component and views to the Vue 3 Composition API style and make changes when necessary.

Dependency Fixes and Replacements

Multiple dependencies have significant version changes, and thus behaviour also has changed. Therefore, those dependencies needed to be adjusted accordingly. It also included replacing no longer maintained packages with alternatives.

Fixes and Finalisation

The main parts of the system have now been changed and are ready. Now, everything needs to be pieced together and see where there are still any issues and solve them when they arise. It also includes fixing build and ESLint errors and warnings, and typing.

There are always obstacles that arise and are not planned for, which also get included in this step.

4.1.3 TypeScript

Typescript already gets included from the beginning of the setup. At first, the code gets migrated over from the previous setup in JavaScript and then converted to TypeScript. With the new store setup, it will need to get typed, and other code should get typed, if possible, at the moment. If unsure regarding the type, "any" can be used or create a local type. If a type gets used multiple times, it should get exported to a general type file. Multiple similar types can be combined and optimised into one type.

4.2 Reducing the Load on the Overpass API

4.2.1 Own Instance of the Overpass API

Overpass API is designed to run as a self-hosted service. Running an instance specifically for OSMyBiz would allow adjusting the rate limits based on the specific needs of OSMyBiz. While this could solve the issues with the public instances once and for all, it has some significant downsides.

Running our instance is a non-trivial task and would require a significant upfront effort and continued maintenance, especially after the end of this project. OSMyBiz is not heavily used, so its Overpass instance would be idle most of the time. Meanwhile, higher latency should be avoided, so adequately fast hardware would be desirable.

Considering these points, we decided that running our own instance of Overpass API is yet to be a viable option for OSMyBiz.

4.2.2 Caching

We assume that most users of OSMyBiz will not randomly jump around on the map but will stay within a specific area (where their business is located). Therefore it is likely that the same area will be loaded twice (temporal locality) or that areas will be loaded next to each other (spatial locality).

By keeping track of already loaded areas, we can avoid loading the same area multiple times, reducing the number of requests to Overpass API. We can also load slightly larger areas than the current viewport so that small movements on the map do not require a new request to Overpass API, further reducing the load.

Even if at a later point, the decision gets made to set up an Overpass instance for OSMyBiz, this would still reduce the load on the custom overpass instance.

Chapter 5

Results

5.1 Goals

The main goals defined in the introduction could get achieved.
The frontend has been overhauled and now uses up-to-date technologies and dependencies. The application works well on smaller screens. The likelihood of issues due to rate limiting got reduced massively. Additionally, many minor issues got fixed as well.

5.2 Outlook and Further Development

There is much potential in developing this application further. With this thesis, the application's main problems got addressed, ensuring it can be used in production again. Additionally, it lays the foundation for future development, making the development of new features simpler, and the code generally is more maintainable.

Following the most important points that could get addressed further

- OSMyBiz-specific Overpass API instance (see 11.3.1)
- Opening Hours GUI (see 11.3.2)
- Life Cycle Management - directly edit nodes (see 11.3.3)
- Change OSM API calls from XML to JSON (see 11.3.4)
- Upgrade to Node 18 (see 11.3.5)

Thanks

We want to express our deep gratitude to our advisors, Prof. Stefan Keller and Joël Schwab, for their invaluable guidance and support during the completion of this thesis. Their expertise and insights were instrumental in helping us to develop and refine our ideas, and we are truly grateful for the time and effort they dedicated to our project.

Part IV

Project Documentation

Chapter 6

Vision

As discussed in Section 1.1

Chapter 7

Requirements

Most requirements have stayed the same since the inception of OSMyBiz. Therefore most requirements have been translated from the original bachelor thesis [9] that created OSMyBiz. Significant changes are italic.

7.1 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. User leaves the site
Post-Condition	-
Alternative Scenarios	1a. User selects address by zooming 1b. User selects address via his current position 3a. User left-clicks the buisness 5a. User creates a OSM-Account

7.2 Create Business

Description	The user would like to add his business to OSM
Actor	User (logged in)
Pre-Condition	User has a 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. User saves changes by pressing "Save"
Post-Condition	OSM-Node was created
Alternative Scenarios	1a. User selects address by zooming 1b. User selects address via his current position *3a. The user "long touches" the building* 6a. The user doesn't want to save his changes and clicks "Back"

7.3 Edit Business

Description	The user would like to edit an existing business
Actor	User (logged in)
Pre-Condition	- User has a 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. User saves changes by pressing "Save"
Post-Condition	OSM-Note was created
Alternative Scenarios	1a. User selects address by zooming 1b. User selects address via his current position 6a. The user doesn't want to save his changes and clicks "Back"

7.4 Show Updates

Description	The user would like to be informed when someone else edits a buisness he created or edited.
Actor	User (logged in)
Pre-Condition	- The user has a 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 Scenarios	2a. User selects business by zooming 2b. The users check the edit on osm.org 2c. The user turns off notifications for this business

7.5 Show OSM Messages

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

7.6 Functional Requirements

Handle Newlines	*OSM generally discourages the usage of newlines in most fields. "description" and "note" fields are sometimes seen as exceptions. Some tools allow inserting newlines into these fields[23], but not all tools can actually properly display them. Osmose for example will mark them as possible errors.[20]* *Therefore the application should be able to deal with existing newlines, but not insert any tags with newlines in them.*

7.7 Nonfunctional Requirements

Responsive Design	The user interface should be responsive, *changes to the User Interface should be done with a mobile first approach. The Application should work on mobile devices as well as on desktop computers.*
Maintainability	The source code for the application should be easily extendable for new features
Usability	The application should be designed to be simple and usable without extensive documentation
Performance	Fast responses are not required, generally users should get a response within 5 seconds. Better reponse times should be aimed for if possible.
Availability	*External APIs should not be overloaded to prevent throttling.*
Reliability	Wrong search results should be avoided.

Chapter 8

Analysis

8.1 Architecture

Overall, the architecture of having the Vue frontend connected to the Python backend and Postgres database was a solid approach. Nevertheless, the frontend technologies needed to be updated. If the project continued on the previous setup, it would have left many security vulnerabilities of packages open and made further development difficult, as one has to use outdated libraries and development methods. The goal is not to change anything in the backend or the database. Therefore, webpack got replaced by Vite as the foundation and the build mechanism. This change reduced the build time for the frontend from 17s to 0.5s on average (average taken from 10 builds) without doing any further optimisations.

The previous state management tool, Vuex, was replaced by Pinia. Compared to Vuex, Pinia focuses on creating multiple smaller stores for specific application parts. Only the required store must be loaded when needed, not the entire store. This separation helps reduce the amount of data loaded for each view or component, as most have to access the store to some degree.

The application was built on Vue 2, as that was the current version at the time of development. Now, Vue 3 exists and has significant benefits and uses the newest versions of Node. Vue 2 uses the options API style of writing code, whereas the recommendation is to use the composition API with the setup script for Vue 3. The composition API allows the definition of component logic using a declarative, function-based syntax. This syntax can make writing clean, maintainable code easier and improve the readability of large components. Overall, Vue 3 offers improved performance, full TypeScript support, an improved reactive API, and better code organisation.

With the setup on Vite, there was the option to include this from the start. The problem was to migrate all the code to the new application foundation and type it. During this, many inconsistencies were found and appropriately resolved. Often, more data than needed got sent and handled, where one could normalise/validate beforehand and then not have to send so much overhead. Furthermore, typing does not have to happen at runtime, increasing the performance and guaranteeing that the code is type-safe.

While the original thesis explicitly focused on a web application oriented toward desktop PCs, the design had no fundamental issues preventing support for mobile devices. The only component that required significant adjustments was the header bar. Using the component system and CSS scoping that Vue provides has proven beneficial, as it made refactoring the design much more straightforward.

System Context

The OSMyBiz system uses multiple APIs (external services) and interacts with different users.

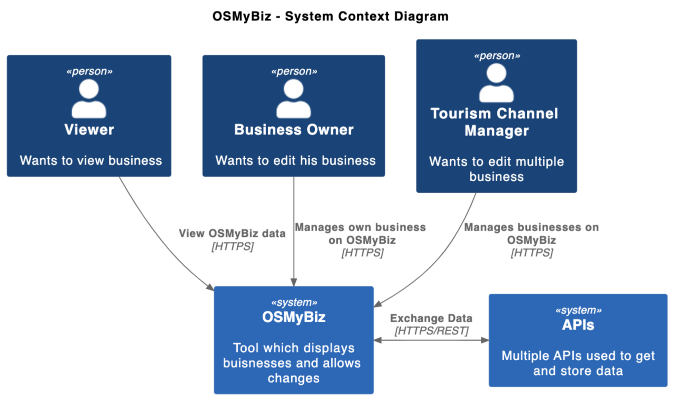

OSMyBiz - System Context Diagram
«person»
Viewer
Wants to view business
«person»
Business Owner
Wants to edit his business
«person»
Tourism Channel Manager
Wants to edit multiple business
View OSMyBiz data
[HTTPS]
Manages own business on OSMyBiz
[HTTPS]
Manages businesses on OSMyBiz
[HTTPS]
«system»
OSMyBiz
Tool which displays buisnesses and allows changes
Exchange Data
[HTTPS/REST]
«system»
APIs
Multiple APIs used to get and store data

Figure 8.1: C4 system context diagram

Container

The system gets split into three containers. Each container uses its own technology and could get exchanged if needed.

The *frontend* is a single-page application (SPA) which provides the user interface. A SPA provides improved user experience over server-rendered pages, as the entire page does not need to be updated when something changes. With a SPA, one is also more flexible for future development.

The *backend* provides multiple REST-API-Endpoints, which will get used by the *frontend*. It is the interface between the *frontend* and the *database*.

The *database* is a simple PostgreSQL database that stores the additional data. The database is part of the OSMyBiz system, so it gets installed, deployed, and migrated along with the other system components.

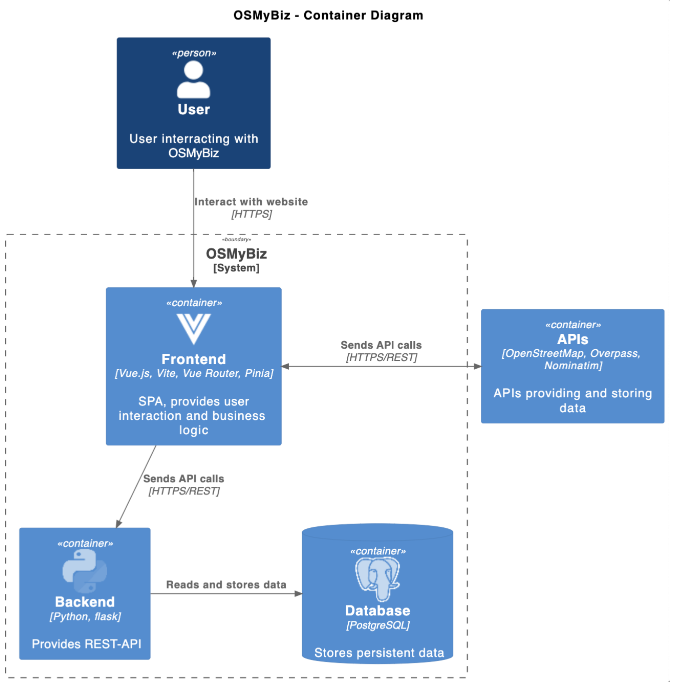

OSMyBiz - Container Diagram
«person»
User
User interracting with OSMyBiz
Interact with website [HTTPS]
«boundary»
OSMyBiz [System]
«container»
Frontend
[Vue.js, Vite, Vue Router, Pinia]
SPA, provides user interaction and business logic
Sends API calls [HTTPS/REST]
«container»
APIs
[OpenStreetMap, Overpass, Nominatim]
APIs providing and storing data
Sends API calls [HTTPS/REST]
«container»
Backend
[Python, flask]
Provides REST-API
Reads and stores data
«container»
Database
[PostgreSQL]
Stores persistent data

Figure 8.2: C4 container diagram

8.2 Components

8.2.1 Frontend

The frontend is a SPA written in Vue 3 and TypeScript. Vue *single-file components* get used, keeping all relevant parts of a component in a single file. TypeScript ensures more issues are caught at compile time, reducing testing overhead.

Store

Pinia gets used as the state store, which is the modern default for Vue 3. Using a central state ensures a clean separation of concerns. The business logic can be in a central, testable and re-usable place.

Views

Each route (*/*, */detail*) gets implemented as a view. These views are responsible for tying together various components and the state store. Relevant data is requested from the store and provided to components.

Components

Whenever any UI code should get re-used, it gets put inside a component, and these components get used across multiple views or inside of other components. A component has simple input and output making them flexible, and this separation guarantees re-usability without requiring refactoring first.

Services

Services (*api* directory in the frontend) provide a small backend abstraction layer. The Axios [3] dependency allows simple creation and handling of the requests.

8.2.2 Backend

Its job is to provide a REST API for the frontend to call, retrieve, and send data. The flask backend then either gets and returns the data from the database or receives the data and stores it in the database. It also encapsulates data migration and other database-specific tasks.

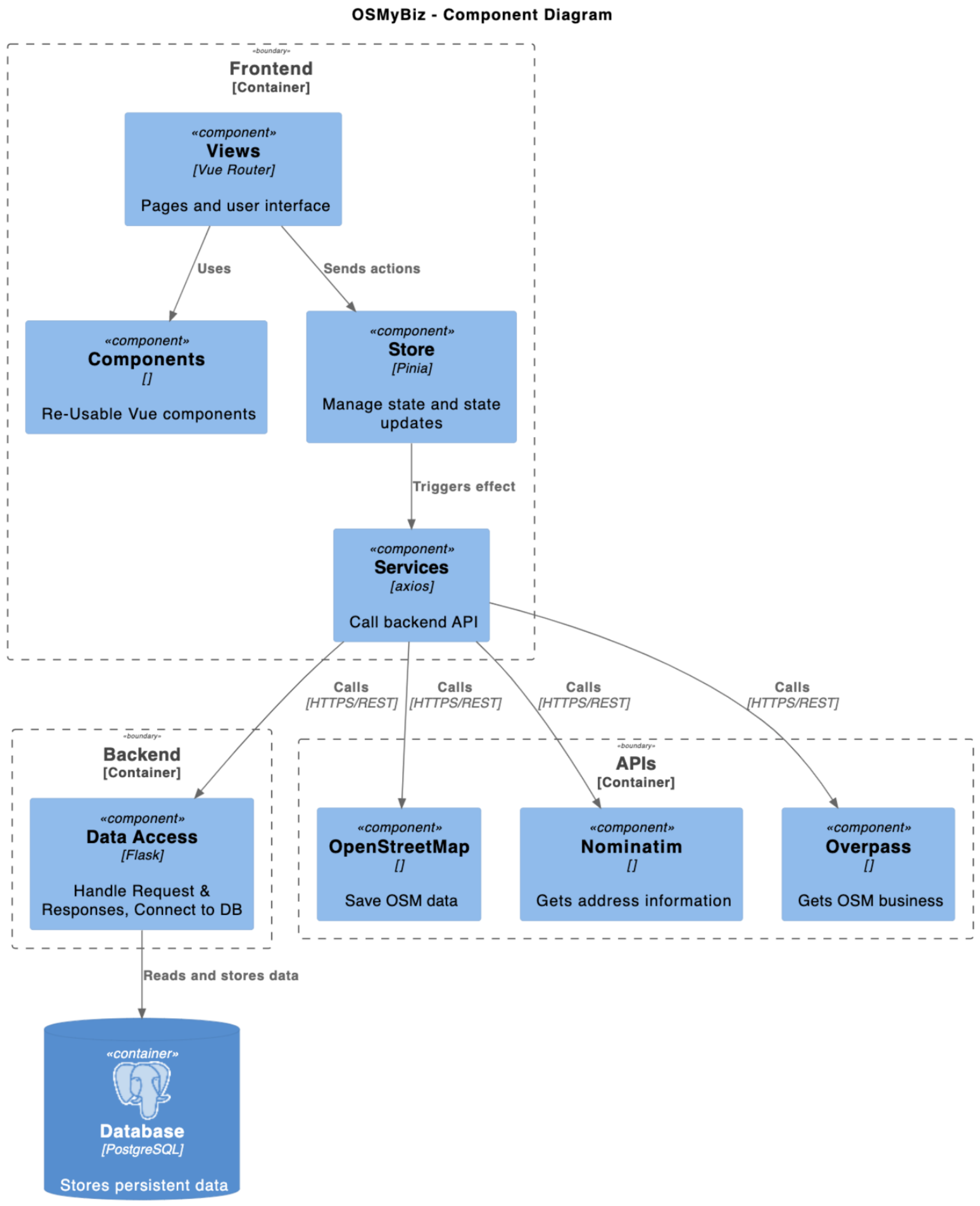


Figure 8.3: C4 component diagram

8.3 Tools & Frameworks

The technologies used got generally chosen with simplicity and maintainability in mind. Nevertheless, using technologies similar to the ones used in the previous thesis made much sense.

Frontend

Vue

Vue.js [33] is a JavaScript framework for building user interfaces focusing on building modern, interactive web applications. It is lightweight, easy to learn, and flexible, and it uses a template-based syntax to render dynamic content in the browser declaratively. Vue.js is popular for building SPAs and has strong community support and a large ecosystem of plugins and libraries.

Pinia

Pinia [12] is a store library for Vue. It allows sharing a state across components/views. Following are the key points of Pinia:

- Stores are as familiar as components. The API got designed to write well-organised stores.
- Types get inferred, meaning stores provide autocompletion in TypeScript.
- Pinia hooks into Vue dev tools to give an enhanced development experience.
- Build multiple stores and let the bundler code split them automatically.
- Lightweight, only 1.5kb

TypeScript

TypeScript [15] is a programming language that is a superset of JavaScript, with added features that make it easier to write and maintain large-scale applications. It supports static typing and the latest JavaScript features, such as classes and modules, to help write better code.

Vite

Vite [32] (French word for "quick", pronounced /vit/, like "veet") is a build tool that aims to provide a faster and leaner development experience for modern web projects. It consists of two major parts:

- A dev server that provides rich feature enhancements over native ES modules, for example extremely fast Hot Module Replacement (HMR).
- A build command that bundles your code with Rollup, pre-configured to output highly optimised static assets for production.

ESLint

ESLint [7] is a tool that identifies and reports patterns in JavaScript code to make it more consistent and avoid bugs. It can enforce specific coding styles and practices and integrate them into various development environments. Overall, it is a valuable tool for maintaining JavaScript code quality [4].

Packages

Further (main) dependencies used in OSMyBIZ with a short description.

- **fortawesome** provides a library of different icons [17]
- **deep-equal** Node's assert.deepEqual() algorithm as a standalone module [16]
- **jquery** is a fast, small, and feature-rich JavaScript library [18]
- **leaflet** is the leading open-source JavaScript library for mobile-friendly interactive map [19]
- **lodash** A modern JavaScript utility library delivering modularity, performance and extras [21]
- **moment** Library for manipulating and dislay of dates and times [22]
- **osm-auth** Easy authentication with OSM over OAuth 2.0 [24]
- **stream-browserify** the stream module from Node core, for browsers [25]
- **tiny-lru** Least Recently Used cache for Client or Server [26]
- **vue-i18n** is the state-of-the-art linter for TypeScript and JavaScript [28]
- **vue-router** official router for Vue [29]
- **vue-select** Vue Select is a feature rich select/dropdown/typeahead component [30]
- **vue3-cookies** A simple Vue.js 3 plugin for handling browser cookies [27]
- **xml-js** Convert XML text to Javascript object / JSON text (and vice versa) [31]

Backend

Flask

Flask [8] is a Python web framework used for creating web applications and APIs. It is lightweight and easy to use, with a simple development server and support for template rendering and custom error pages. Flask is often used for small projects, prototypes, and quick experiments, as well as for building APIs and microservices.

Database

Postgres

Postgres [13] is an open-source object-relational database management system (ORDBMS) for storing and managing data. It supports ACID transactions, advanced data types, and a powerful query language called SQL. Postgres is known for its reliability, performance, and scalability and gets often used for web, mobile, and big data applications. It also has robust security features and is widely used in various industries.

Chapter 9

Design

Design

From the original bachelor thesis, there was already a design created and implemented. In this thesis, the overall design got modified as little as possible. The goal was to improve the mobile usage of the web application without having to create a new UI, as generally, the website's design is well thought-through and well-styled. Furthermore, a new design might confuse the user that already knows and works with OSMyBiz, so there was a decision against redesigning the webpage.

The most significant design change was the header bar. The original did not scale well on small screens, and buttons could be offscreen and impossible to tap. We have decided to go with a so-called "Hamburger Menu." It is a typical design pattern on mobile phones. A button with three horizontal lines expands a vertical menu. This changed menu is only visible on smaller screens. Larger screens continue to use the old design. A third-party contributor had already started an implementation, which we finished up.

On small screens, the watch list is now an almost full-screen modal. It gained a close button as an obvious way to close it. This button is crucial when the watchlist button gets hidden under the hamburger.
Additionally, various other minor changes got made to fix overlapping elements and similar things.

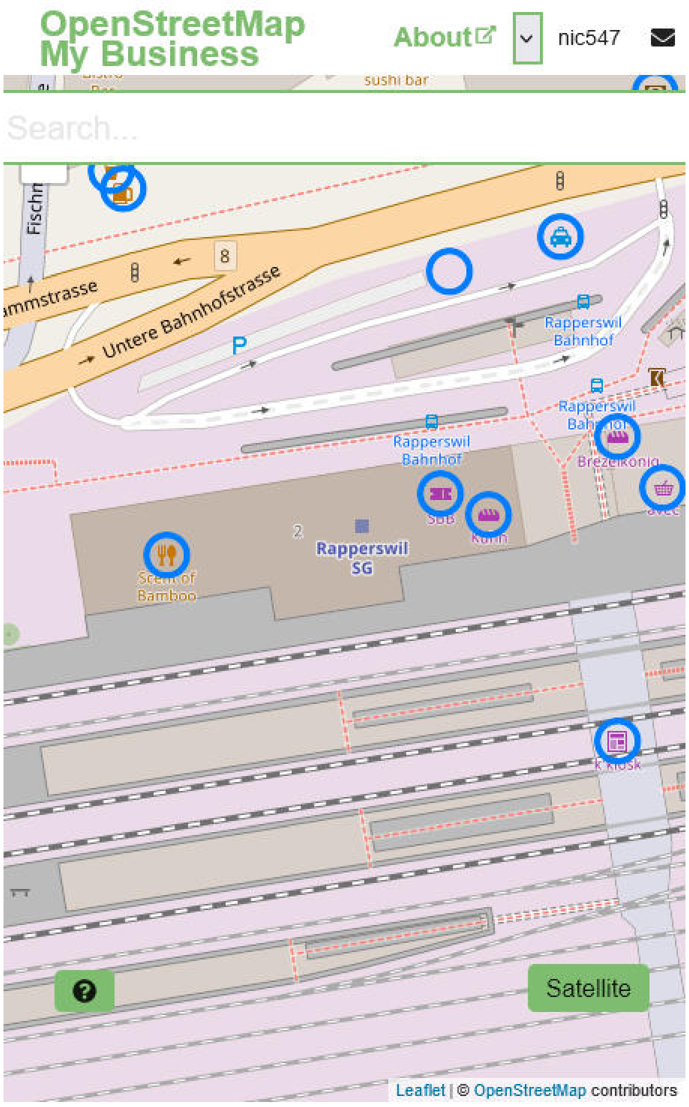


Figure 9.1: The old design on a small screen. Notice that the watch list and logout buttons are not visible onscreen.

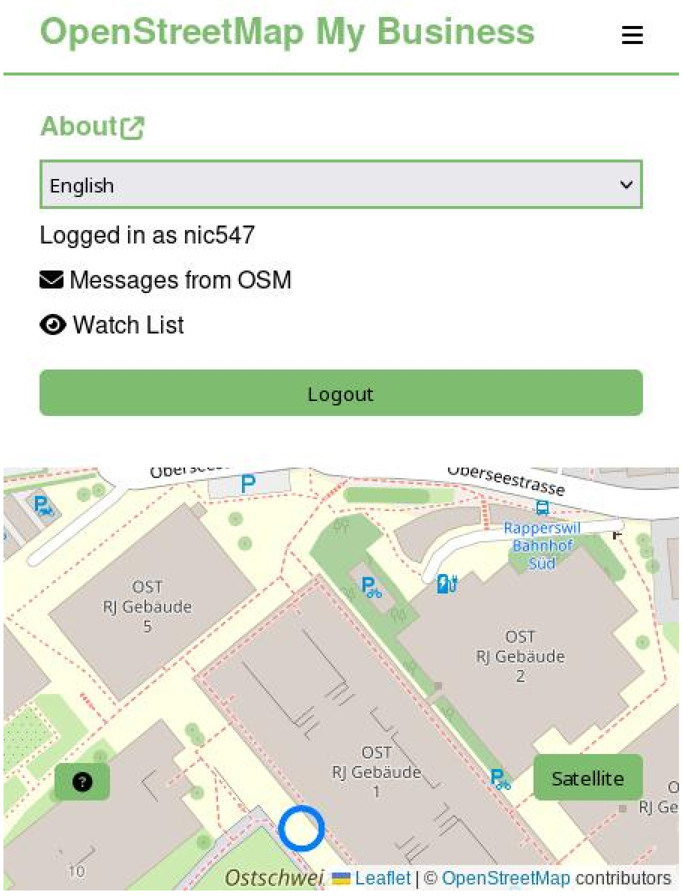


Figure 9.2: The new design on a small screen.

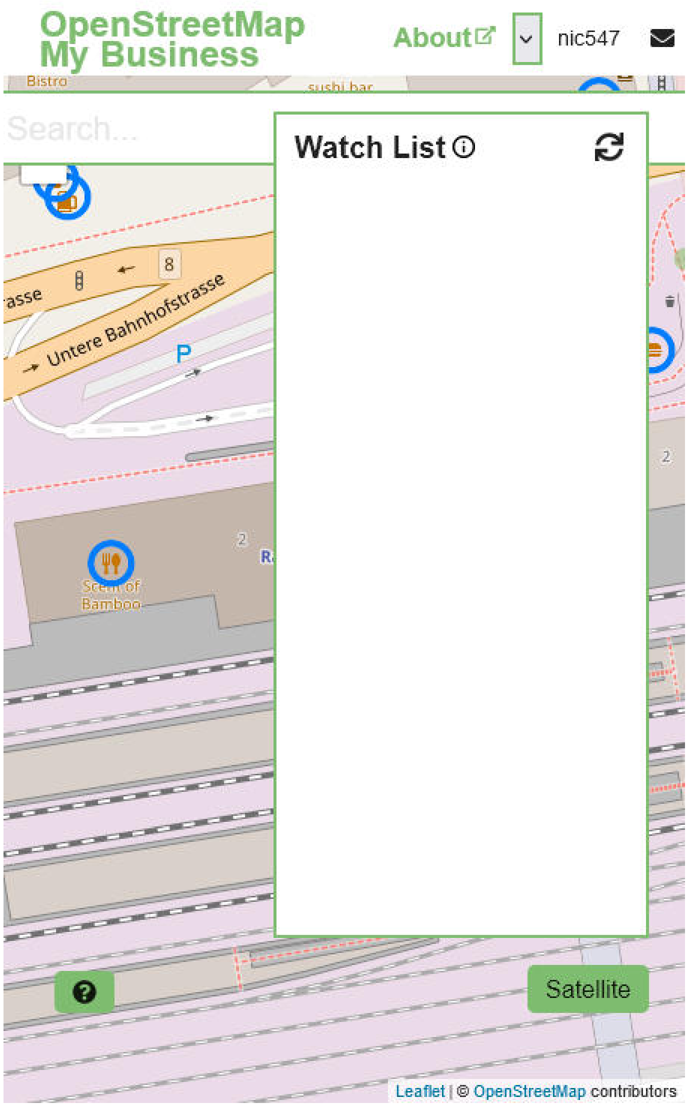


Figure 9.3: The old watch list.

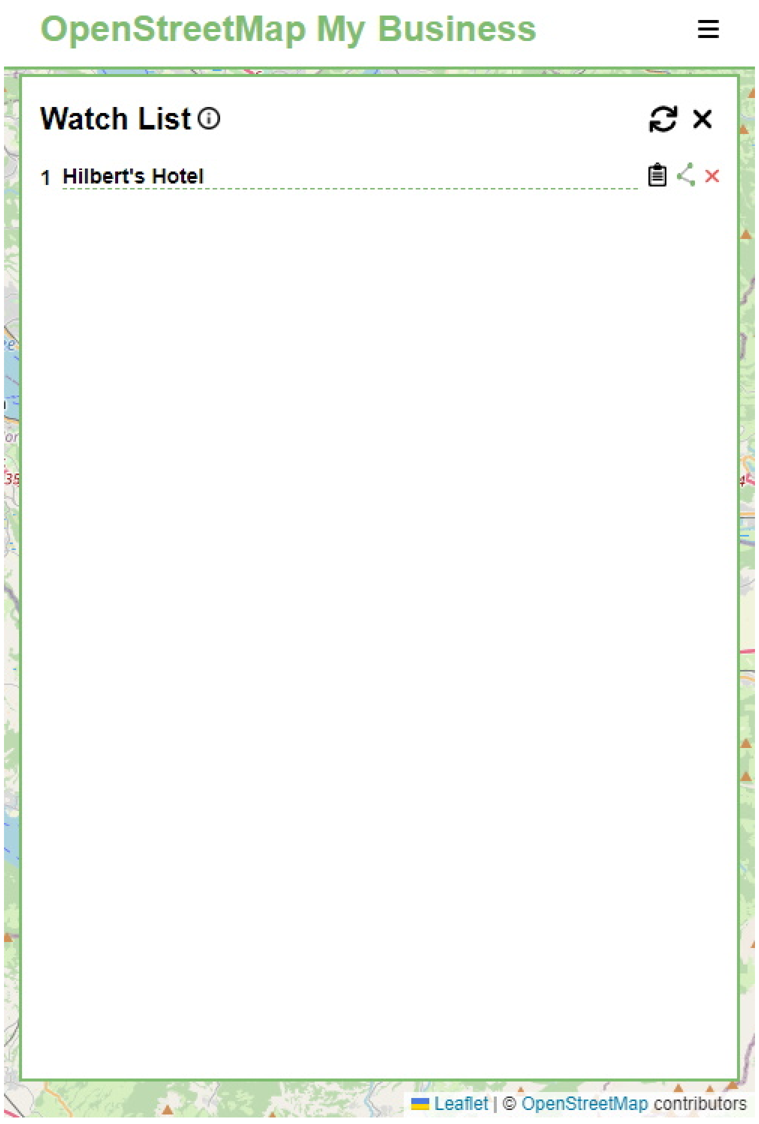

OpenStreetMap My Business
≡
Watch List ⓘ
1 Hilbert's Hotel
Leaflet | © OpenStreetMap contributors

Figure 9.4: The new watch list.

Chapter 10

Implementation and Testing

10.1 Frontend Upgrade

From the beginning, it was clear there was a need for a frontend upgrade. Either lifting the existing application to new versions or creating the application from scratch / new foundation and migrating the code. Especially since the application is five years old, the technologies have evolved far, and for most libraries, many new major versions exist and should get used. As on the build of the old application, multiple warnings/errors arose regarding security issues found in older packages and updating is recommended. Node version 8 was used instead of the current LTS version of Node 16.

There was an analysis conducted on the libraries used in the application. To identify which are not maintained or compatible with the newer Node version, a replacement must get found if it is not maintained or compatible with Node 16. With this analysis, multiple packages could get identified that could get reduced from standard functionalities in newer versions or from different libraries with the same functionalities.

Also, with the switch to Vite, Node 16, Vue 3 and TypeScript, the number of packages could be reduced. Also, many of the previous tools needed for development could get dropped. For example, webpack hot reload and all of the other webpack stuff, as this is all replaced by the one Vite dependency.

The first try was to update the current application and lift those packages accordingly. Convert everything to ES6 syntax and bump the packages to the newer versions. After bumping a few packages, the build was failing; one had to continue this "blindly" and hope that, in the end, everything would come together. After investing a day of work into this approach, it dropped, as it did not bear fruit. The problem is that one would have a new application with modern technologies. However, the foundation of an application got created five years ago, which is different from the standard of how applications get created. Also, packages could not get removed as hoped, as most were still required, just with a higher version.
Afterwards, the approach was to create an empty application based on the Vite, Vue 3, Pinia, TypeScript and ESLint tech stack that could be configured accordingly during the setup of the Vite application. Code had to be moved and rewritten from the old application into the new one with the ES6 syntax. TypeScript support and Vue files had to get migrated into the composition API style.

One of the main libraries, vue-leaflet, only had Vue 2 support. The vue3-leaflet gets maintained by a private person and is still in progress and does not provide full support and documentation yet. Although the key components were supported, a lot was studying the implementation and figuring out how to implement it to replace the Vue 2 implementation.

Another issue was the osm-auth library. It claims to have TypeScript support, but ESLint complained about no named export as described in the documentation. After spending multiple hours figuring out the problem and reading into the library source code, the issue is that the TypeScript definition file most likely has an error, and everything works as it should. Much time got wasted trying out different things until, finally, the authentication worked.

10.2 Caching

The big question from the start is how we can organise our cache to make it easy to check what is cached and what is not. Typical caches have a straightforward identifier for the stored data. A cache line is identified on a CPU cache by the address of the data it contains, and a browser cache can use the unique URL of an image to identify it. In our case, we are looking for data in a range of latitudes and longitudes. At the same time, we need to be able to tell whether an area is empty or not cached.

We decided to solve this by dividing the map into "Tiles". So instead of loading nodes based on a range of latitudes and longitudes, we load nodes based on a set of discreet tiles. Each tile has a size of 0.01 degrees latitude and longitude. The size was determined experimentally and hits a good trade-off between how often new tiles must be loaded and how much data gets loaded at once. These tiles get identified by their latitude and longitude; for example, the tile with the left corner at 51.52, 7.50 would be identified by the string "51.52/7.50". The cached data generally does not get stale very quickly, so we cache the data for a session. This means the cache gets cleared when the user closes the browser. On the other hand, we wanted to limit the amount of stored data. If the cache gets too large, we want to evict the least recently used tile.

We arbitrarily decided to cache 25 tiles simultaneously to prevent high memory usage. This limit could be increased, but we can assume there will not be thousands of entries. As the library will be part of the web app, we were looking for a small, simple library rather than a sizeable specialised library.

Based on this, we decided to use the "tiny-lru" library. This library is a simple LRU cache that stores the data in memory, and it has a fixed size and evicts the least recently used items when the cache is full.

When the map gets moved, we can check which tiles are currently visible and which are already cached. Because of how Overpass does its rate limiting, it makes more sense to load the tiles in a single request. Even if there is an overlap between the request and the tiles already cached, the missing tiles will be combined into one request for a rectangle containing all tiles.

Initially, we then displayed all of the affected tiles on the map, even if they were outside the browser's viewport. This meant that the markers could already be loaded when scrolling around the map. Unfortunately, we found this was not a good idea, as it would cause the map to be very slow to display, visibly adding them one after another over multiple seconds in very dense urban areas. Even after adding additional code filtering out the nodes that are not visible, the performance could have been better. Another issue we encountered was that when doing multiple small scrolling movements (for example, by navigating using the arrow keys), it was possible to send multiple requests for the same area. Because the request to

Overpass gets sent asynchronously, it is possible that the map was moved multiple times, the tiles still need to be cached, and therefore additional requests were sent. To solve this, we had to keep track of requests that were in flight. Instead of sending duplicate requests, further attempts to load the nodes in an area also await the first request.

10.3 Testing

We have done manual tests based on the requirements defined in Chapter 7. To enable us to test on actual smartphones, we have set up a test server to run OSMyBiz. In order to be able to test changes that were not yet integrated into the main development version, we decided to forgo integration into our CI/CD pipeline. Instead, the test server can access the git repo and pull changes from there.

The substantial effort to upgrade the frontend has also proven a hindrance to testing. Its many fundamental changes resulted in many issues with the application. One issue early in a scenario meant that further issues could only be found after the first issue was solved. It also simply ate a lot of the time budget.

Another issue we encountered while testing the application was the fact that there was a mix of live and test data. The map tiles from OpenStreetMap used live data, and Overpass API used live data. However, we used a development instance of OpenStreetMap to prevent modifying anything on the "real" version of OpenStreetMap. This generally required some workarounds.

There was a sizeable amount of feedback on the original GitLab repository of the project, which was incorporated where applicable. Some testing with potential users was started, but it encountered the abovementioned issues. Given the existing feedback from real users and a few fundamental UI changes, we decided to focus on other areas rather than conducting exhaustive user testing.

Chapter 11

Results and Further Development

11.1 Results

11.1.1 Frontend Upgrade

Due to the state of the OSMyBiz application before, it was clear that an upgrade had to take place in some form.

Vite Instead of Webpack

Vite and webpack are tools for building web applications, but their focus and features differ. Vite is fast and lightweight, using native ES modules to avoid the overhead of traditional bundlers. On the other hand, webpack is more fully-featured and can handle a broader range of use cases but it is also more complex.

Vite is the foundation of the new frontend and replaces webpack. As it is the current state-of-the-art way to set up new Vue 3 applications, it was the way to go. With the basic setup of Vite instead of webpack, around 15 dependencies could get removed. Without further configurations regarding the build, the build with Vite took 0.5s and with webpack around 17s. This number can be further reduced when more time is invested into optimising the build.

TypeScript

With the introduction of TypeScript, many issues became visible that before were just swept under the rug. Many inconsistencies with naming and types arose as well as bugs, precisely because of it. It makes the code safer as one knows what gets excepted and what gets sent over, and inherently makes the code more readable and easier to maintain.

Pinia

Allows the store's modularisation, and only the store and data needed get included and loaded instead of the complete store. Technological standard store of Vue 3 and users get advised to switch from Vuex, as it is no longer further developed.

Responsive

Thanks to fixing a few issues with the application's design on mobile devices, all features should work well on mobile phones and similar devices, keeping up with the trend towards replacing classic computers with mobile devices like smartphones.

11.1.2 Caching

By splitting the map into tiles and then loading and caching them accordingly, we have reduced the load on the Overpass API, lowering the likelihood of running into troubles with the rate limiting. While it is still possible to provoke such situations, most users should now be able to use OSMyBiz without regular error messages again.

11.2 GitLab Issues of OSMyBiz

The issue section on the GitLab repository `https://gitlab.com/geometalab/osmybiz/` was analysed and the following issues were solved:

Title	Issue
"Save button obscures part of form"	#253
"Add Persian for supported languages"	#252
"App doesn't cache Overpass API results"	#251
"Upload translations for Spanish and Catalan"	#247
"Help on smartphone is not accurate"	#243
"Impossible to save changes"	#242
"Impossible to type some characters"	#238
"Detail-Dialog: Add field "Opening Hours URL" (OSM key "opening_hours:url")"	#231
"Point out that the user needs to zoom right in (aka zoom level 18) to see markers"	#216
"Go to "Own Data" in map"	#189
"White Page after login to "master.apis.dev.openstreetmap.com""	#186

11.3 Further Develoment

As displayed in Chapter 5.2, multiple points can be improved or implemented. The most important ones are listed below with further information.

11.3.1 OSMyBiz-Specific Overpass API Instance

As mentioned in the concept for "Reducing load on Overpass API" (Section 4.2.1) the Overpass API is designed to be run as a self-hosted service. Running an instance specifically for OSMyBiz would allow adjusting the rate limits based on the specific needs of OSMyBiz, solving the rate limit issues with the public instances.

11.3.2 Opening Hours GUI Interface

Editing and adding opening hours is complex and not user-friendly at all. It requires a strict format which is not apparent to the user. A UI would simplify this task by creating the correct format string in the background to send to OSM.

11.3.3 Editing Nodes Directly

Currently, the application does not directly edit nodes; instead, it leaves notes for other mappers with the suggested changes. This is not a good workflow for our users or other mappers. Directly editing nodes would improve this. OSMyBiz could then use lifecycle prefixs to "delete" businesses.

11.3.4 JSON Responses Instead of XML

Since 2022, multiple OpenStreetMap API endpoints have been able to return data in JSON instead of only the XML format. Especially the '/node/nodeID' route, which is used multiple times in the OSMyBiz application, could benefit from this change alongside the other occurrences. The switch to JSON would simplify the parsing of the data into a usable format, as currently, the XML data gets parsed and formatted into a JSON object. JSON is lightweight and easy to parse, making it suitable for high-volume applications. XML is more flexible and allows for custom tags, making it suitable for complex data. Furthermore, two dependencies required for parsing and reading the XML data could get removed.

11.3.5 Node 18

Node 16 has its End of Life date of 11.09.2023. An upgrade to Node 18, where the EOL is 30.04.2025, would make sense and ensure security. As this is not a significant version jump like from 8 to 16, this should be fine. At the start of this thesis, some dependencies still needed to be compatible with Node 18. Therefore, it only got upgraded to Node 16. However, it most likely will not take long until all required packages are ready for Node 18.

Chapter 12

Project Management

12.1 Relevant Links

Following a list of tools referenced used for version control and issue tracking.

- GitLab Code: `https://gitlab.ost.ch/sa-osmybiz/osmybiz`
- GitLab Documentation: `https://gitlab.ost.ch/sa-osmybiz/documentation`
- Jira: `https://sa-osmybiz.atlassian.net/browse/MB`

12.2 Processes

We used Scrum [14] in our project and held regular planning and weekly meetings. We also used Jira [2] to create and manage issues and track our time. These processes helped us stay organised and focused and ensured that we were making progress towards our goals.

Meetings

Meeting	Schedule	Subject	Timebox
Weekly	Every Friday	Scrum Daily	15m
Planning	Every second Monday	Scrum Planning	45m
Project Weekly	Every other Monday	Discuss Progress and what's next	15-30min
Kick-Off	19.09.2022	Kick-Off - Project Idea	1.5h

12.3 Collaboration

We used Microsoft teams [10] for communication and OST GitLab for version control, allowing us to collaborate effectively and ensure everyone was on the same page.
For each bugfix, change or new feature that we worked on, we created a custom branch in GitLab. This allowed us to isolate our work and avoid conflicts with other team members. We reviewed each other's code before merging it into the main branch. This helped us catch any errors or bugs before they made it into the main codebase and ensured that our code was high quality. Here we followed the GitFlow process of collaboration and working with git.
Using teams and GitLab helped us collaborate effectively and produce high-quality code. It allowed us to work together efficiently, avoid conflicts, and ensure that our code was well-organised and maintainable.

12.4 Risk Management

In this section, the potential risks in our project get listed, and the plans how to react to them.

Risk	**Mitigation**	**Reaction**
Inexperience with a technology	Plan accordingly	Seek assistance as soon as problems arise
Dependencies have been deprecated	Acceptance	Replace as early as possible
Existing code has a fundamental issue	Acceptance	Rework existing code when necessary
OSM-API had backwards incompatible changes	Acceptance	Adjust project accordingly
OSM suffers a outage	Acceptance	Try to work around outage

12.5 Time and Issue Tracking

We used our Jira Board to track our time on different tasks in our project, which allowed us to stay organised and on track. Using Jira, we could create tasks and track the time spent on each. We could also assign tasks to different team members and set deadlines.

12.6 Time Tracking Report

Time tracking is done exclusively in Jira. Following an overview of the total time spend per person:

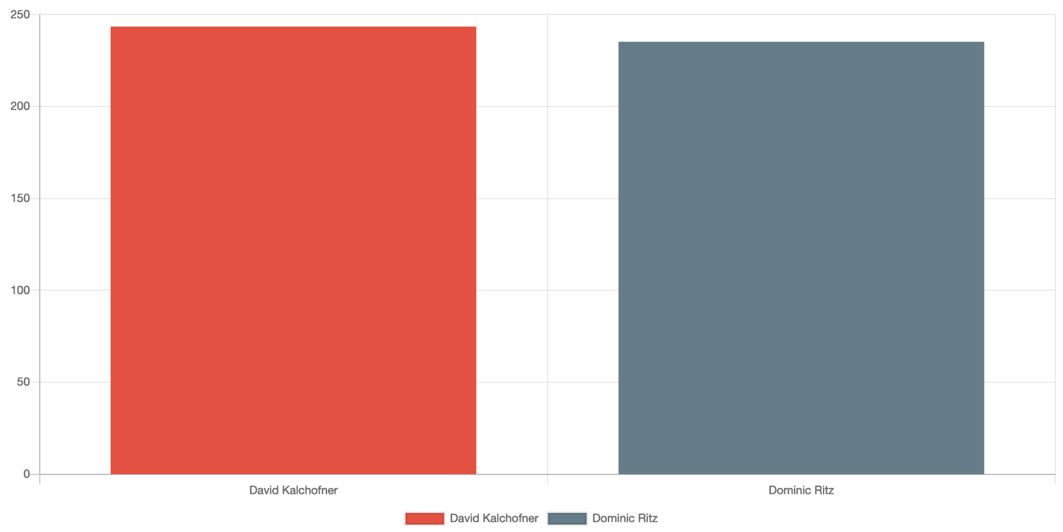


Figure 12.1: Time tracking export - David Kalchofner 243.5h, Dominic Ritz 235.25h

Part V

Appendix

Glossary

GitFlow Gitflow is a branching model for Git that provides a structured approach for managing source code. Gitflow defines rules for how these branches should be used and merged, helping teams manage their codebase consistently and predictably. .

lifecycle prefix Can be used to mark OSM objects as under construction, disused, demolished, etc..

Mapper Colloquial term for a contributor to the OSM map..

mobile first describes an approach to web design where the mobile version of a website is the baseline with the desktop version being built on top of that.

node A geographical point in OpenStreetMap that represents a specific location, such as a building, a street intersection, or a natural feature. .

note A user-generated comment or observation in OpenStreetMap that is associated with a specific location, and that can be viewed and commented on by other users. .

osmose Osmose is a quality assurance tool for OSM that detects possible issues with data on OSM.

responsive A website is responsive if it adapts to the screen size of the device it is displayed on.

tag A tag is a key-value pair that describes an object on OSM. For example, the tag "amenity=restaurant" describes that an object is a restaurant..

tagging scheme The tagging scheme is the set of tags that are used to describe the data on OSM. There is no formal definition, rather it's a set of conventions adopted by the OSM community. .

Bibliography

[1] Overpass API. Commons. `https://dev.overpass-api.de/overpass-doc/en/preface/commons.html`. Accessed on 2022-12-09.

[2] Atlassian. Atlassian jira. `https://www.atlassian.com/de/software/jira`. Accessed on 2022-12-20.

[3] Axios. Axios. `https://axios-http.com/`. Accessed on 2022-12-20.

[4] belindamarionk.hashnode.dev. What are eslint rules? `https://belindamarionk.hashnode.dev/what-are-eslint-rules`. Accessed on 2022-12-03.

[5] Datanyze. Google maps api market share and competitor report. `https://www.datanyze.com/market-share/mapping-and-gis--121/google-maps-api-market-share`. Accessed on 2022-10-22.

[6] EndOfLife.date. endoflife.date nodejs. `https://endoflife.date/nodejs`. Accessed on 2022-12-03.

[7] EsLint. Eslint. `https://eslint.org/`. Accessed on 2022-12-03.

[8] Flask. Flask. `https://flask.palletsprojects.com/en/2.2.x/`. Accessed on 2022-12-03.

[9] Max Lüthi and Simon Heller. Openstreetmap my business. Bachelor's thesis, Hochschule für Technik Rapperswil, Rapperswil, December 2017.

[10] Microsoft. Microsoft teams. `https://www.microsoft.com/de-ch/microsoft-teams/log-in`. Accessed on 2022-12-20.

[11] Node.js. Node.js. `https://nodejs.org/en/`. Accessed on 2022-12-20.

[12] Pinia. Pinia. `https://pinia.vuejs.org/`. Accessed on 2022-12-03.

[13] Postgres. Postgres. `https://www.postgresql.org/`. Accessed on 2022-12-03.

[14] Scrum. Scrum. `https://www.scrum.org/`. Accessed on 2022-12-20.

[15] TypeScript. Typescript. `https://www.typescriptlang.org/`. Accessed on 2022-12-03.

[16] Various. Deep equal. `https://github.com/inspect-js/node-deep-equal`. Accessed on 2022-12-03.

[17] Various. Fortawesome. `https://fortawesome.com/`. Accessed on 2022-12-03.

[18] Various. jquery. `https://jquery.com/`. Accessed on 2022-12-03.

[19] Various. Leaflet. `https://leafletjs.com/`. Accessed on 2022-12-03.

[20] Various. Line breaks in tag values. `https://github.com/osm-fr/osmose-backend/issues/1336`. Accessed on 2022-11-3.

[21] Various. Loadash. `https://lodash.com/`. Accessed on 2022-12-03.

[22] Various. Moment js. `https://momentjs.com/`. Accessed on 2022-12-20.

[23] Various. newline in fields. `https://github.com/openstreetmap/iD/issues/7249`. Accessed on 2022-11-3.

[24] Various. Osm auth. `https://github.com/osmlab/osm-auth`. Accessed on 2022-12-20.

[25] Various. Stream browserify. `https://github.com/browserify/stream-browserify`. Accessed on 2022-12-20.

[26] Various. tiny-lru. `https://github.com/avoidwork/tiny-lru`. Accessed on 2022-12-20.

[27] Various. Vue 3 cookies. `https://github.com/KanHarI/vue3-cookies`. Accessed on 2022-12-20.

[28] Various. Vue i18n. `https://kazupon.github.io/vue-i18n/`. Accessed on 2022-12-20.

[29] Various. Vue router. `https://router.vuejs.org/`. Accessed on 2022-12-20.

[30] Various. Vue select. `https://vue-select.org/`. Accessed on 2022-12-20.

[31] Various. Xml js. `https://github.com/nashwaan/xml-js`. Accessed on 2022-12-20.

[32] Vite. Vite. `https://vitejs.dev/`. Accessed on 2022-12-03.

[33] Vue.js. Vue.js. `https://vuejs.org/`. Accessed on 2022-12-03.

[34] Vuex. Vuex. `https://vuex.vuejs.org/`. Accessed on 2022-12-03.

List of Figures

Personal Reports

David Kalchofner

In this thesis project, I was excited to have the opportunity to upgrade OSMyBiz using the latest technologies and best practices. As someone familiar with some of the technologies to be used, I was looking forward to the challenges and opportunities the project would present. One of the key challenges I faced during the upgrade was the introduction of TypeScript to a codebase that had previously not used any type checking. The upgrade required much more careful analysis and debugging and took longer than I had anticipated. Despite these challenges, I am proud that the project succeeded. The upgraded frontend of the web application performed better and provided a better user experience than the previous version. It was great to see everything come together and work smoothly and to know that the application will be able to continue to meet the needs of its users in the future. Overall, this project was a valuable learning experience. It taught me the importance of thorough planning and preparation and allowed me to improve my knowledge of known and unknown technologies. I am grateful for the opportunity to work on this project, and I look forward to applying the lessons I learned to future projects.

Dominic Ritz

So far, my professional experience with software development has been mostly .NET-focused. Web technologies have been part of my studies at OST, and this project was a great look at how Vue and related technologies work in a larger project. I learned a lot from David, as he is very well-versed in this stack.
I anticipated that the dependency upgrade would take a significant amount of time. However, I still underestimated the effort it took. It continued throughout the project, and because it was a substantial change, it required quite some effort to prevent or solve conflicting changes. David mostly took care of the upgrade, whereas I took care of other things. I plan to split such work in future professional projects. In this case, it probably was impossible since our jobs and other educational obligations made "syncing up" somewhat hard.
Working with OSM and some community members has been a great experience, and I am happy that our work will hopefully help make OSM more accessible to "non-Mappers," keeping data up to date.

Meeting Minutes

Kickoff Meeting 19.09.2022

Goals

- Get an idea of SA
- Understand problems
- Understand Open Street Map
- Plan to-dos for the next two weeks
- Define and plan basic meetings

Open Street Map / Problem statement

Bi-Weekly Meeting 3.10.2022

Goals

- Plan further work

Results

- We will try and upgrade the existing Frontend.
- We will make sure that we can clearly show our own contribution.
- For every Bi-Weekly Meeting with our advisors, we should prepare a small statement describing what we have done, which obstacles we encountered, and what plans we have for the next iteration. This statement should be provided via teams or mail at least 24h before the meeting.
- Joël will provide a draft of the problem statement - we should review it afterward.

Bi-Weekly Meeting 17.10.2022

Goals

- Discuss work done with advisors
- Discuss planned work with advisors

Results

- We have a plan for upgrading the Frontend and will execute it this sprint
- We have a rough list of issues we want to solve to get a responsive design and will execute it this sprint
- Due to Dominic's injury, we are somewhat behind on documentation, but we want to catch up during this sprint

Bi-Weekly Meeting 31.10.2022

Goals

- Discuss work done with advisors
- Discuss planned work with advisors

Results

- We worked on both the dependency upgrade and the responsive design
- We came to the realisation that the dependency upgrade is one large task that is really hard to split up into smaller tasks.
- Documentation is a bit behind, we'll have to catch up on that. Dominic still has to catch up after his injury.

Bi-Weekly Meeting 14.11.2022

Goals

- Discuss work done with advisors
- Discuss planned work with advisors

Results

- Editing nodes directly rather then going via notes that might be resolved at some random point in the future would be an interesting feature.
- There is a SOSM-Meetup this week where we could ask Simon for advice about the node editing feature.
- The dependency upgrade is at a point where we can start merging other changes.
- Caching the Overpass-API results is planned for this Sprint.
- We have collected a bunch of issues that we want to analyse and fix.

Bi-Weekly Meeting 28.11.2022

Goals

- Discuss work done with advisors
- Discuss planned work with advisors

Results

- We still had do some work on the dependency upgrade.
- Caching is0 more or less done.
- Dominic was at the last SOSM-Meetup and was able to talk with Simon about the project.
  - Directly editing shouldn't be a problem as long as we're reasonably careful. Editing via notes was done out of historical carefulness.
  - Deleting directly isn't great, as it's hard to undo. We should rather use lifecycle prefixes. "Main" tags get prefixed, and minor tags (e.g. cuisine) get removed. That change is much easier to undo.
  - Some care needs to be taken for nodes that are two things simultaneously, e.g., a restaurant and a hotel. The gold-rim solution is to use a heuristic to determine which tags belong to which part and only operate on those. This is how vespucci handles this. For a start we should just detect this and fall back to a note or fail.
  - Using the review-needed tag on the changeset is probably not that useful. While the tag itself is a good idea, with how it's currently used for changesets means that there is a massive amount of changesets with this tag, but few people actually looking at them based on that. Therefore using this tag doesn't really do anything.
- David had less time than he wanted to work on the project because of a project at work.
- OSM-API was offline for a few days, which made it hard to test our changes.
- We have done some work on editing the nodes directly (rather than posting the edits as nodes). We'll try to do this feature, but if we encounter issues, we'll have to cut it relatively soon and rather focus on the core goals of the SA.
- Other than that, we'll work on the final polish and documentation.

Bi-Weekly Meeting 12.12.2022

Goals

- Discuss work done with advisors
- Discuss planned work with advisors

Results

- We have merged most large changes
- Unfortunately, we had to cut the "Editing nodes directly" feature due to time constrains
- There were some specific points raised about the documentation: The title page needs to be restructured. We should ensure that we're using a sans-serif font. Meeting minutes should be the last part of the documentation.