Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Application for ordering and management of waste pickup

Růžena Bednářová

Supervisor: Ing. Martin Klíma, Ph.D.
May 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bednářová** Jméno: **Růžena** Osobní číslo: **483553**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

Specializace: **Enterprise systémy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Aplikace pro objednání a správu svozu odpadu

Název bakalářské práce anglicky:

Application for ordering and management of waste pickup

Pokyny pro vypracování:

Navrhněte a implementujte aplikaci pro zákazníky dosud neexistující služby, pomocí které si budou moci objednávat a spravovat odvoz různých typů odpadu z jimi určených míst.
Aplikace bude mít klient-server architekturu, pro klientskou část použijte vue.js framework, pro serverovou část použijte Node.js technologii. Databázi vyberte podle svého uvážení.
Aplikace umožní určit typ odpadu, potřebné nádoby, lokace, časový rozvrh. Uvažujte i firemní klientelu. Ověřte použitelnost a funkčnost aplikace pomocí testování s uživateli formou zjednodušených testů použitelnosti.
Zjednodušení bude ve výběru cílové skupiny a to proto, že nábor a výběr skutečně relevantních testerů nebude z časových a organizačních důvodů možný.

Seznam doporučené literatury:

Vue.js online documentaiton
Node.js online sources
Nuxt framework https://nuxt.com/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Martin Klíma, Ph.D. Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **07.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

Ing. Martin Klíma, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studentky

Acknowledgements

I would like to thank my supervisor Martin Klíma for the guidance and patience. Special thanks to my supportive family and partner. Finally, I would like to thank the people who are developing the other applications that together with mine application make the whole waste management system, namely Damir Abdullayev and Egor Ulyanov, for their collaboration.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

I declare that I used AI tool Writefully to assist with rephrasing my work in a style more suitable for academic work.

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských prací.

Prohlašuji, že jsem použila AI nástroj Writefully k přeformulování mé práce do stylu vhodnějšího pro akademickou práci.

V Praze dne 24. května 2024

. . Růžena Bednářová

Abstract

The goal of this thesis is to design and develop a waste management tool. This project is part of multiple applications that together form a system for the entire waste disposal chain from the producer to the waste processor. This project focuses on the waste producer and creating a way for them to dispose of the waste. The main requirement for this tool is that the pickup time possibilities are flexible. This was achieved by analyzing the state-of-the-art requirements of a typical customer and partially developing an application in which the customer creates the order for the pickup of sorted waste. The project will likely be further developed after this thesis and will possibly be deployed to real customers. The result of this thesis is an analysis, the design, and the front-end part of the application that utilizes back-end services as well. The application was tested by multiple people for usability feedback.

Keywords: Waste management, Waste sorting, Environment, Ecology, Application, Pickup, Producer, Order, Schedule, Software, Frontend, Javascript, Typescript, Vue, Node

Supervisor: Ing. Martin Klíma, Ph.D.
E-321,
Resslova 9,
12000 Praha 2

Abstrakt

Cílem této bakalářské práce je navrhnout a vyvinout nástroj pro správu odpadu. Tento projekt je součástí několika aplikací, které společně propojují producenta odpadu se zpracovatelem odpadu. Projekt této bakalářské práce se zaměřuje na producenta odpadu a na vytvoření způsobu, kterým může producent odpad odvézt ke zpracovateli. Hlavním požadavkem tohoto nástroje bylo, aby možnosti času svozu byly flexibilní. Tohoto bylo dosaženo analýzou existujících řešení, požadavků typického zákazníka a vývojem části aplikace ve které si zákazník může objednat odvoz tříděného odpadu. Tento projekt bude pravděpodobně dál vyvíjen i po této bakalářské práci a bude nasazen pro reálné zákazníky. Výsledkem této práce je analýza, design a front-endová část aplikace která využívá servis na back-endu. Aplikace byla otestována několika lidmi, abych získala zpětnou vazbu k její uživatelské přívětivosti.

Klíčová slova: Správa odpadu, Třídění odpadu, Životní prostředí, Ekologie, Aplikace, Svoz, Producent, Objednávka, Rozvrh, Software, Frontend, Javascript, Typescript, Vue, Node

Překlad názvu: Aplikace pro objednání a správu svozu odpadu

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

In this age, it is important to try to preserve our planet and the environment in which we live as much as possible. This can be achieved, for example, by using sustainable resources, buying mindfully from ethical companies that follow ecological practices, and managing our own waste efficiently. However, waste management can be time-consuming and expensive. The motivation behind this project is to make this task easier for users to sort their waste and dispose of it correctly. We aim to develop an application that allows users to order waste pickups according to their specific needs. Existing waste management companies often operate on a fixed non-customizable schedule. However, this system can cause nearly empty containers to be picked up. Therefore, waste pickup cars consume an unnecessary amount of fuel, and their carbon footprint increases.

1.2 Goals

Popelka, as this application is named, is one of the multiple web applications that serve as an intermediary between waste producers and waste processors. The set of applications consists of:

- User application for waste producers
- Waste collector application for users who transport waste from waste producers to waste processors
- Application to collect and simplify all information from the two mentioned applications and that plans the upcoming pickup schedules

This work focuses on the user application. In this application, users can order waste pickups according to their needs. Users can set the time and place for waste collection and specify the type of waste that is reffered to as stream. The application then shares the data with the other applications, which manage all the orders and scheduling for the waste collectors. When the waste collection is completed, the application provides the user with the

confirmation of the waste disposal required by law. The user can also follow the guidelines provided by the application on how to sort and store waste accordingly. Waste collectors provide feedback on how well waste was sorted and whether the bin was empty, full, or overflowing. This feedback determines the cost of the user's future pickups.

Chapter 2

Analysis

2.1 State of the Art

We spoke with the general public spokesperson of Pražské technické služby to gather the information that helped us determine the features of the application. However, Pražské technické služby do not provide an application through which the customer can order a waste pickup. We explored existing solutions. Not many applications are focused on the same functionality, but there are several that are oriented towards similar topic, which is waste management and the circular economy. They include some functionalities present in our design. The applications are arranged according to their relevance to this project.

2.1.1 Seenons

Seenons is a Dutch company that facilitates connections between its customers and waste processors. Waste is not necessarily collected regularly, but rather on demand, depending on when the customer needs it. Different types of waste are classified as streams, for which customers can order containers from Seenons [1].

Features

- Tips & Tricks section on how to reduce and sort waste.
- Option of waste containers delivered within five days of the order.
- 24 hour call center available for customers to change scheduled pickup. However, a successful pickup is not guaranteed if the request was made less than two days before the pickup.
- The mobile application is implemented by adding a browser shortcut. Therefore, it is not available on Google Play or Apple Store. The desktop application has the same layout as the mobile application.

2.1.2 Cyrkl

The so-called "waste dating application" is a Czech start-up that operates as a waste marketplace. This start-up connects waste producers and potential buyers of this waste through machine learning and data analysis. This allows customers to save money on waste disposal as their waste will be utilized by another company, often without fees for removal [2].

Features

- Free and paid accounts with additional features.
- Initial scan during which Cyrkl visits a company and identifies different waste streams. They then calculate the optimal way to dispose of this waste, including estimating potential emissions and financial savings.
- They showcase examples of savings which their customers achieve by using their application. These examples are displayed on their website.
- Advisory services on new waste and disposal laws.
- A price comparison based on their trends, similar to commodities on the stock exchange.
- Verification of sellers.

2.1.3 reKáva

The company collects coffee grounds which are then used, for example, as compost in community gardens. The customer places the coffee grounds in the Smart Bin. When the bin is full, it automatically notifies the collectors. The collectors then come to empty the bin. They also offer Growkits. Growkits are flower pots that customers can use to grow oyster mushrooms by adding coffee grounds [3].

Features

- They transport the coffee grounds on electric bicycles that have a large bin attached to them.
- This service is currently only available in Prague.
- The ordering process is not automatic: The customer completes a form. The company then contacts the customer to make arrangements.

2.1.4 Waste Management - My WM

Pickups are not ordered through this application. However, it helps the user keep track of the waste they produce and how much waste pickup costs them [4].

Features

- Calculation of the user's next monthly payment according to their pick-ups.
- Alerts that inform the user about pickups.
- International advisory providing information on local laws related to waste.

2.1.5 Summary

There are applications that provide services on the topic of recycling and circular economy. A popular feature of applications seems to be some kind of advisory service, often powered by AI. The inspiration for this project comes from the Seenons application.

2.2 User

2.2.1 User persona

The typical user is a company or entrepreneur that produces various waste streams. They aim to lead their company in an environmentally and financially efficient manner, willing to sort waste. They are able to plan pickup schedules and estimate the quantity of waste. They have an overview of their company's waste production. They seek quality feedback, provided by the application in the form of a calculated carbon footprint and detailed statistics on how effectively they mediate waste. Their waste production is not uniform. They generate significantly more waste in some periods than in others.

2.2.2 User roles

Business - Global administrator

The registration for this role is the most comprehensive and requires agreement with contractual conditions. This role typically belongs to a user who needs an overview of all waste pickups in their subsidiaries and who needs to manage employee accounts.

Manager - Local administrator

This role is usually held by a subsidiary manager or someone responsible for its operation. The user may be responsible for the ordering of materials and pickups. They have all rights except for viewing data from other subsidiaries. They cannot cancel or modify a Business account.

Employee

This role is reserved for a typical subsidiary employee who does not plan pickups. However, they ensure that waste is sorted and collected by waste collectors. They have limited rights. They cannot create or modify pickups in any way. They cannot modify the Manager and the Business accounts.

Accountant

A person with an accountant role cannot change or create pickup orders. They can see all data for the registered subsidiaries of the company.

Primary contact - Global administrator

An account with this role cannot be deleted by anyone. They have all rights related to their company.

Administrator

The administrator role is reserved for the administrators and developers of the application.

Roles

Whitelist - User can whitelists new employees.
Pickups - User can order and edit pickups.
Status - User can see detail and status of a pickup order.
Statistics - User can see the statistics of the subsidiary they work in.
Sub - User can see data from multiple subsidiaries.
Business - User can see and manage data from multiple businesses.

2.2.3 User benefits

The user gains a better ecological image, making their business more attractive to environmentally conscious customers. The user has the option to view detailed records of pickups. The application allows them to connect to other aspects of their company, such as how much waste they produce and in which areas they can save money. Price optimization is implemented because waste is collected only when necessary, and not on a regular basis for almost empty bins. If the customer adheres to the proper waste disposal practices, they receive a discount on future pickup orders. With adequately filled bins and employees paying more attention to waste, the customer's premises become more hygienic. The customer can set the pickup time and adjust or cancel it two days prior to the pickup. They receive automatic confirmation of waste disposal according to legislative requirements. The customer does not have to adapt to the waste collector's schedule. Instead, they can customize pickup,

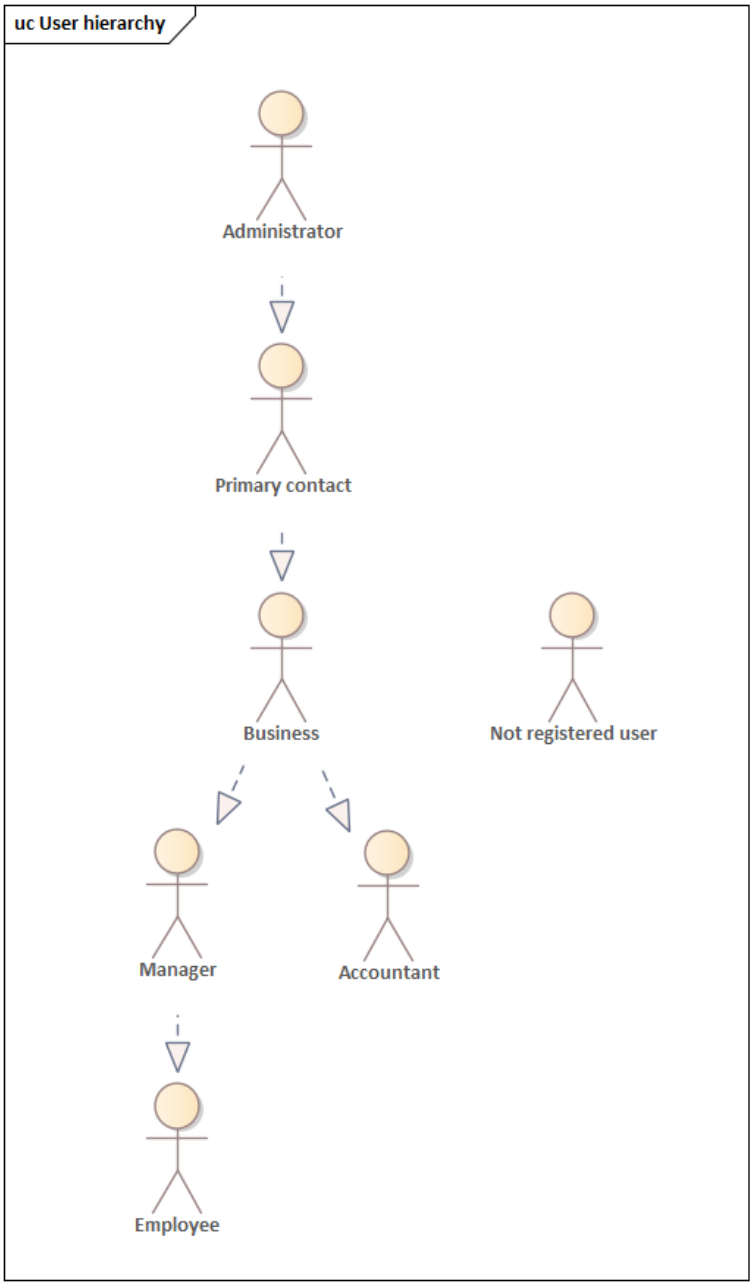

Figure 2.1: Users hierarchy

including access to the premises, communication with waste collectors, and more.

2.2.4 Actions needed on the user side

These actions should be taken by the user to ensure that the application works most effectively for them: The company should train its employees about waste sorting and using the application. Instead of one container, the

Role	Whitelist	Pickups	Status	Statistics	Sub	Business
Business	✓	✓	✓	✓	✓	✗
Manager	✓	✓	✓	✗	✓	✗
Employee	✗	✗	✓	✗	✗	✗
Accountant	✗	✗	✓	✓	✓	✗
Primary contact	✓	✓	✓	✓	✓	✗
Administrator	✓	✓	✓	✓	✓	✓

Table 2.1: User's rights depending on their role

company will have multiple containers, requiring more space. The customer needs to invest financially in these containers.

2.3 Requirements

2.3.1 Functional requirements

Business requirements

- **BRQ 001 - User Personalization**
  As a user, I need to have my settings and data in the application that no one else can access without my permission because I want to be able to customize the application to my needs.

- **BRQ 002 - Waste Pickup Schedule and Place**
  As a user, I need to be able to easily schedule pickup because I need my waste collected at a specific time and place.

- **BRQ 003 - Waste Pickup Records**
  As a user, I need information about my pickups to be recorded and accessible, as I want to monitor them and deduce important ecological data about my company's waste.

- **BRQ 004 - Existence of Waste Streams**
  As a user, there I need to choose from streams in the application, as I sort my waste.
- **BRQ 005 - Waste Disposal Confirmation**
  As a user, I need to obtain the confirmation of waste disposal because it is required by law.

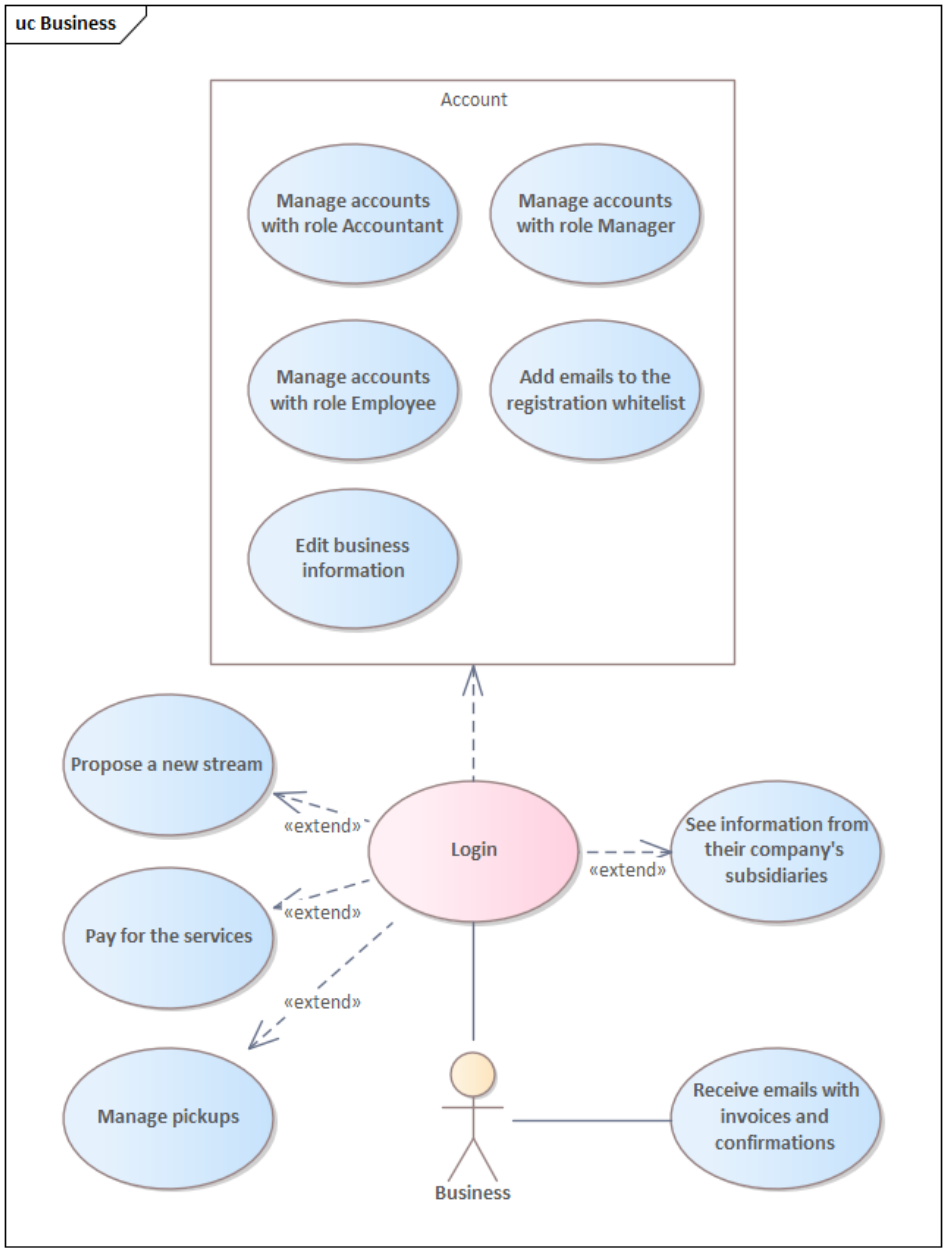

Figure 2.2: Business use cases

- **BRQ 006 - User Awareness of Pickup state**
  As a user, I need to know where the waste collector is, if the pickup was

successful, and the waste collector's feedback on my waste so that I can optimize these steps in the future.

- **BRQ 007 - Distribution of Role Rights**
  As a user, I need different employees of my company to have different rights so that certain actions are not performed by unauthorized employees.

- **BRQ 008 - Service Feedback**
  As an application administrator, I need to facilitate the ability for customers to rate waste collector's services so that I can optimize services for customers and prevent potential future issues.

- **BRQ 009 - Application Rating**
  As an application administrator, I need to facilitate the ability of customers to rate the application so that any issues can be documented and subsequently addressed.

- **BRQ 010 - Overview of the Funds Invested in Pickup**
  As a user, I need to have an overview of the finances invested in pickup because it will help me understand my finances and whether my employees manage waste efficiently.

- **BRQ 011 - Pickup customization**
  As a user, I need the option to customize the pickup because, for example, access to the location of the company's containers is complicated.

- **BRQ 012 - Carbon Footprint**
  As a user, I need an indicator of the carbon footprint so that I can improve the ecological performance of my company and that I can display it to my customers.

- **BRQ 013 - User Awareness of the Application**
  As an application administrator, I need my customers to be informed about the updates and features of the application so that they can use it consistently.

- **BRQ 014 - Application Security**
  As an application administrator, I need the application to be secure because it processes sensitive user data.

- **BRQ 015 - Legal Framework for the Application**
  As an application administrator, I need the customer to agree to the contractual terms because I need legal protection and clear rules need to be established.

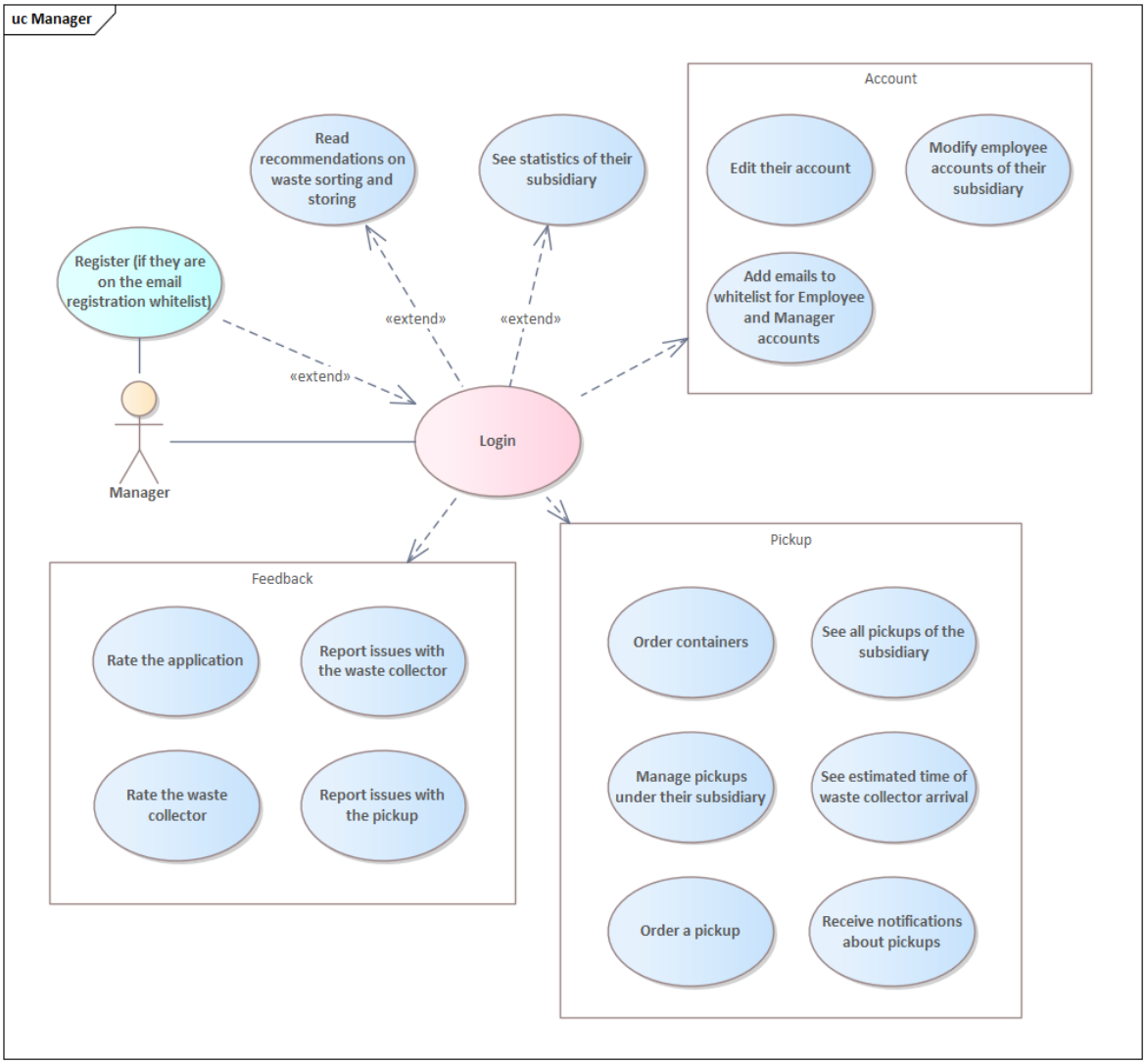

Figure 2.3: Manager use cases

- **BRQ 016 - User Payment**
  As an application administrator, I need users to have a way to pay for my services because I need to generate revenue.

- **BRQ 017 - User Penalization**
  As an application administrator, I need the user to be penalized for violating the terms and conditions (e.g., changing the scheduled pickup too late) because it leads to a complication on our side.

- **BRQ 018 - Communication Customization with Waste Collector**
  As an application administrator, I need the option for the customer to choose the method of communication with the waste collector because I am trying to customize the application as much as possible to the customer's preferences.

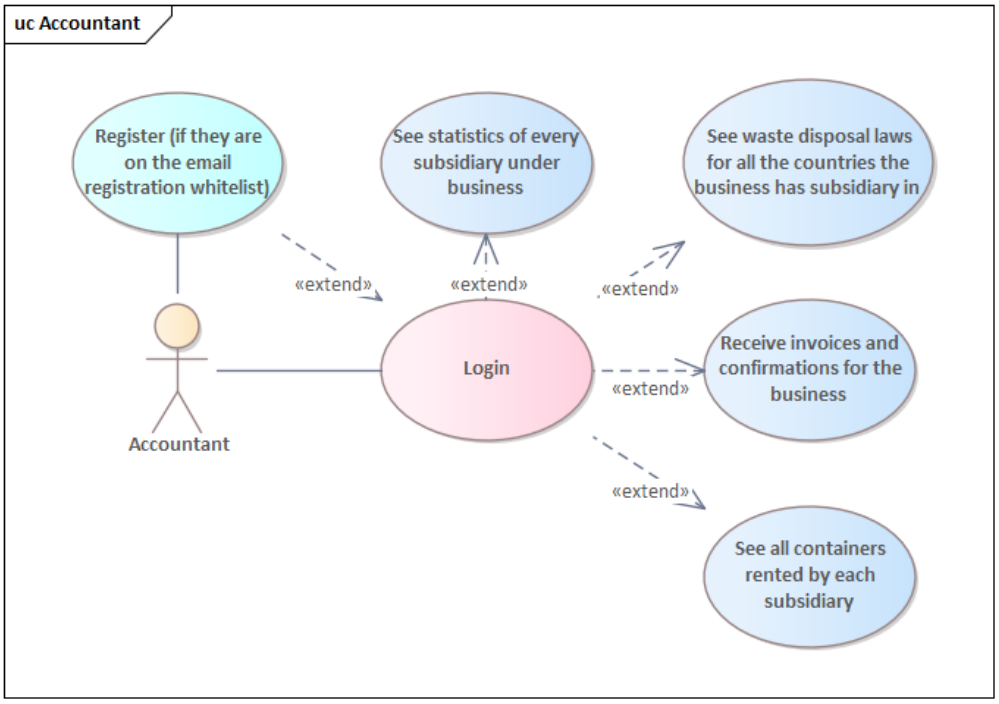

Figure 2.4: Accountant use cases

- **BRQ 019 - Updating Stream Selection**
  As an application administrator, I need to keep the stream selection up-to-date and tailored to my customers needs because the customers would choose a different waste management company otherwise.
- **BRQ 020 - Container Selection**
  As a user of the application, I need the company to provide a container for the waste because I do not know the requirements for the specific waste, nor do I have a container to put the waste into.

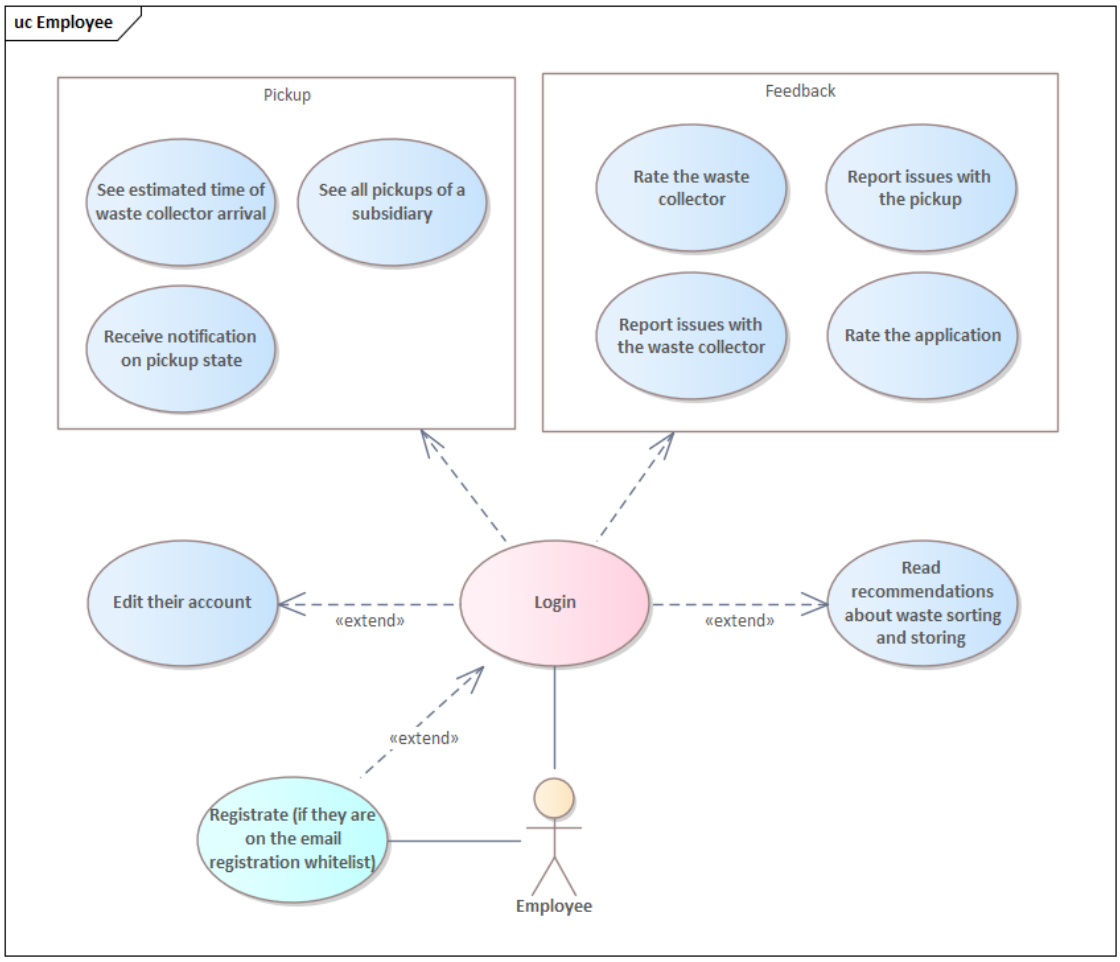

Figure 2.5: Employee use cases

System requirements

- **SRQ 001 - User Registration (BRQ 001, priority: high)**
  The system will allow the user to register. Users can register as a company via contacting the application's company. Subsequently, they can register their employees within the company account. When registering a company, all details must be filled in. When registering employees, only their name, surname, and identifier are required. Employees register with a maintainer who already has an account in the company. In the case of SSO login, missing information will be required after login.

  Attributes for business registration:

  - Company name
  - Business ID

- Company address
- Contact email
- Phone number
- Password
- Password confirmation
- Responsible person
- Name
- Surname
- Identifier within the company
- "Register the responsible person as the Manager" column - in case the company has only one employee who is also the manager
- Checkbox - subscribe to the newsletter

Attributes for the registration of subsidiary managers:

- Name
- Surname
- Password
- Password confirmation
- Email
- Phone number
- Address of the pickup point for which they are responsible
- Typical streams produced by the branch (optional)

Attributes for employee registration:

- Name
- Surname
- Password
- Password confirmation
- Phone number

- **SRQ 002 - User Login (BRQ 001, priority: high)**
  The system will allow the user to log into their account. Employee login does not require the Manager to be logged in. The application allows for the SSO log-in, such as through a Gmail account.
- **SRQ 003 - Selection of Streams Produced by the Company (BRQ 004, priority: high)**
  The system will allow the manager to select and save streams that are typically produced by their subsidiary. The manager can add a new stream from the menu or propose a new stream at any time. When creating a pickup, only the streams previously added by the manager are displayed using a dropdown.

- **SRQ 005 - Selection of Pickup Point Address (BRQ 002, priority: high)**
  The system will allow the manager to select a pickup point. The manager enters this location during registration. The manager can change the location. The manager enters the address into the input fields, and the system checks if the address exists. In case of an error, it warns the manager.

- **SRQ 006 - Selection of the Time Window for Pickup (BRQ 002, priority: high)**
  The system will allow the manager to select a pickup time window. The time window can be changed via the application no later than 48 hours prior to scheduled pickup.

- **SRQ 007 - Regularity of Pickup (BRQ 002, priority: high)**
  The system will allow the manager to set the regularity of the pickup.

  Such a regularity can be:

  - Once only
  - Weekly
  - Every two weeks
  - Every three weeks

- **SRQ 008 - Estimated Time of Arrival of the Waste Collector (BRQ 006, priority: medium)**
  The system will allow the user to have an estimate of the waste collector's arrival time. This estimate will be taken from the waste collector's application and automatically updated in the user's application. The estimate will be in minutes to an hour.

- **SRQ 009 - User Notification of Approaching Waste Collector (BRQ 006, priority: medium)**
  The system will allow the user to be notified of the arrival of the waste collector. There are three types of notifications:

  - The waste collector is on the way
  - The waste collector is few minutes from you
  - The waste collector is on-site

- **SRQ 010 - User Notification of Successful/Unsuccessful Pickup (BRQ 006, priority: high)**
  The system will allow the user to be notified of the pickup result. Both the manager and the currently logged-in employees of the subsidiary will be automatically notified about the success of the waste pickup.

- **SRQ 011 - Communication between Waste Collector and User in Case of a Complication (BRQ 018, priority: medium)**

The system will allow the user to communicate with the waste collector in case of a complication. This will be achieved through either a central application, an intermediary between the waste collector's and user's applications, or through direct communication between the user's and waste collector's applications. Chat functionality will not be implemented.

- **SRQ 012 - Summary of the Quality of Waste After Pickup (BRQ 006, priority: high)**
  The system will allow the user to be informed about the quality of the pickup. This will be found in the Pickups section, which will be sorted by the most recent pickups. The pickup that occurred a maximum of 20 minutes ago will be visually highlighted.

- **SRQ 013 - Rating of the Waste Collector's Service (BRQ 008, priority: medium)**
  The system will allow the user to rate the waste collector's service. The rating will include the following information:

  - The behavior of the waste collector.
  - The speed of service.
  - (Optional) a text field for comments.

  Rating is not mandatory. The user will be alerted that they can rate the waste collector, but the rating window will not be opened by default.

- **SRQ 014 - Pickup Creation (BRQ 002, priority: high)**
  The system will allow the user to schedule a pickup no later than 24 hours before the pickup.

- **SRQ 015 - Application Feedback Section (BRQ 009, priority: medium)**
  The system allows the user to provide feedback on the application. If there is an issue with the application, the user will complete a form on how to properly report the bug. If it is just a suggestion, the user will fill in a text field.

- **SRQ 016 - Overview of Pickups (BRQ 003, priority: medium)**
  The system allows the user to view all completed or scheduled pickups. Pickups can be filtered by:

  - Time when they were performed
  - Type of streams
  - Completed vs. scheduled

  Pickups are sorted from the most recent.

- **SRQ 017 - Billing Sent by Email (BRQ 010, priority: high)**
  The system allows the manager and the business to obtain the billing information for pickups for the month. This billing will be sent to the manager and the business by email every month.

- **SRQ 018 - Stream Proposal (BRQ 019, priority: low)** The system allows the user to suggest a stream to the application. The user proposes a stream not included in the application, but one they produce. The user describes the stream and adds a photo example of the stream.

- **SRQ 019 - Generation of Waste Disposal Confirmation (BRQ 005, priority: high)**
  The system allows the business to receive an automatic confirmation of waste disposal.

- **SRQ 020 - Financial Overview (BRQ 010, priority: medium)**
  The system enables the user to display a financial overview of investments in pickups for a specific period through statistics.

- **SRQ 021 - Special Requests for Pickup (BRQ 011, priority: high)**
  The system allows the manager to customize the execution of a pickup during its creation. Customizations may include special entry into the company's premises.

- **SRQ 022 - Calculation and Presentation of Carbon Footprint (BRQ 012, priority: medium)**
  The system allows users to view the carbon footprint of their branch and the overall carbon footprint of the business. The system updates the carbon footprint calculation after each pickup.

- **SRQ 023 - Container Ordering (BRQ 020, priority: high)**
  The system allows the manager to order containers. The manager can choose containers according to the type and volume of the stream if needed. The company invests financially in containers. Containers are delivered within 5 business days.

- **SRQ 024 - Generation of Pickup Reports (BRQ 003, priority: low)**
  The system allows the user to generate a report on all pickups for a given month or year in PDF format.

- **SRQ 025 - Sending Invoices by Email (BRQ 010, priority: high)**
  The system allows the system administrator to send the user an invoice by email.

- **SRQ 026 - Overview of Local Waste Laws (BRQ 015, priority: medium)**
  The system allows the user to view local waste laws that change in the application based on the country in which they are currently in.

- **SRQ 027 - Email Confirmation (BRQ 001, priority: high)**
  The system allows the user to confirm their email by sending and, if necessary, resending a six-digit confirmation code to the provided email.

The code is automatically inserted into the application field after being copied to the clipboard.

- **SRQ 028 - Roles in the application (BRQ 007, priority: high)**
  The system has different roles implemented for the potential users. The different roles have different rights depending on what are the users responsibilities in the company. They follow the rules that are defined in the section User roles. The roles are:
  - Business
  - Accountant
  - Manager
  - Employee
  - Primary contact
- **SRQ 029 - Application Updates and Notifications (BRQ 013, priority: low)**
  The system will notify the user every time the application is updated. This is so that the user's security within the application is maintained and that the user does not miss out on new features and fixes. This will be achieved through mobile phone notifications and emails.
- **SRQ 030 - Securing user interactions (BRQ 014, priority: high)**
  The system will ensure that each user action is authorized and that the unauthorized users do not have the permission to do such action. The user must be authorized at all times otherwise they need to log in. The application will be tested to minimize the unpredictable behavior and thus a space for mistakes or attacks.
- **SRQ 031 - Payment interface (BRQ 016, priority: medium)**
  The system will display monthly fees to the user and will provide a payment interface through which the user can pay for the services. Multiple payment methods will be available, and the payment process will follow standard online payment practices.
- **SRQ 032 - User penalization for late changes in pickup order (BRQ 017, priority: high)**
  The system will penalize users if they need to change the pickup order later than two days prior to the scheduled pickup. The penalty will be a change fee added to the upcoming monthly fee. If the user chooses to do this often, it will affect future prices for their pickups. The user will be able to see this in the application along with the reason for the fee.

2.3.2 Qualitative requirements

- **NFR 001 - Browser support**
  The application needs to be able to run on the two most recent versions of the Chrome and Firefox browsers.

- **NFR 002 - Application security**
  The application must authorize every user with every action so that all user data is protected.
- **NFR 003 - UI/UX**
  We need the application to have an effective UI so that users continue using our application. The risk of not achieving this can be minimized by designing and reviewing the UI design. The design must then be tested by potential users.
- **NFR 004 - Mean Time to System Recovery - 12 hours**
  The system must recover within 12 hours from a failure.
- **NFR 005 - Mean Time to System Failure - 30 days**
  The system must not fail more than once every 30 days.
- **NFR 006 - Logging**
  The logging system needs to be implemented so that important events and errors can be captured.
- **NFR 007 - Monitoring**
  The monitoring system needs to be integrated so that we have a tool to monitor the health of the application.

2.4 Technology stack

One of the first steps in the realization of this project was to choose the frameworks and tools that would provide the features we needed. We researched the available tools, compared them, and chose the ones that seemed the best according to our criteria.

2.4.1 Typescript

Typescript is a programming language that extends JavaScript. Typescript has static typing, which allows developers to define the types of variables. This ensures type security in programs. In some Vue components we use Javascript and in the others we use Typescript as a programming language in the script tags and on the back-end side of the application. The reason is that type security contributes to the debugging process because the variable origin is clearer than without it. Static typing also prevents developers from passing incorrect-type variables to functions [5].

2.4.2 Frontend

2.4.3 Vue.js

Vue.js is a framework based on Javascript that was first released in 2014. Vue is based on a component-like structure. Developers implement components

from which they can build larger components or pages. Vue component has three types of root tags [6]:

- <script> A section for Javascript/Typescript code and for Vue specific features such as defineProps. defineProps is an entry point for all input variables of the component.
- <template> A section for HTML code. Dynamic variables can be refferenced with ':' (e.g. :nameOfVariable). Then those variables are defined in the script tag section.
- <style> A section for CSS. This tag supports different variations of CSS (e.g. SCSS). The tag has an attribute which defines if the style should be loaded generally or only if the component is present on the current page.

Vue API

Vue developers implemented two types of API. They provide developers with different programming paradigms and features. When creating a component, developers choose the API in which they would like to program it. The two APIs are compatible with each other, meaning that the project can contain options API components and composition API components at the same time. A component can also consist of both APIs. However, this is only recommended when the project consists mostly of options API and some features that are not available in options API need to be used. APIs follow standard programming paradigms with some customizations [7, 8].

- **Options API**
  Options API was introduced in 2014 with the first Vue releases. Components that use the options API follow an object-oriented programming paradigm. This programming paradigm may be more familiar to some developers. However, the options API has limitations. The logic cannot be shared between multiple components. As a result, the code may repeat. This leads to a phenomenon where it is difficult to navigate through the code due to its complexity and volume.
- **Composition API**
  The composition API was released with Vue 3.0. The composition API is based on the functional programming paradigm.

  It addresses the issues of the options API. The logic can be shared between different components. Therefore, the code repetition is less common. Multiple features can be used with composition API such as: async/await, promises, third-party libraries, hooks, and dependency injection. Composition API provides a Reactivity API to make components reactive to changes. There is full support for Typescript in the composition API.

  The production bundle also tends to be smaller in composition API. The property access in options API uses "this", an instance proxy. However,

in composition API, the property access is direct. This makes the application more efficient.

In order to use composition API, developers need to use <script setup> tag instead of <script> tag.

Vue and React comparison

Vue and React are JavaScript-based frameworks. Both use reactivity with their components. These are some of the differences in React and Vue [9]:

- **React useState and Vue ref**
  Both of these features allow the variables to which they are attached to adapt automatically with change. When defining a variable with useState hook, we use three attributes: state variable, initial state, and a function through which we can update the state of the variable. When the function is called, React compares its virtual DOM with the previous version and re-renders only what has changed. On the other hand, in Vue we use the ref hook, which only needs the initial value. Vue ref then uses an observer pattern which re-renders the changed part when explicitly the value of the variable is changed.

- **React JSX and Vue template**
  HTML code is written differently in these two frameworks. React uses JSX, which allows the developer to write HTML-like code inside their Javascript code. Vue uses templates, which allow developers to write HTML-like code separately and provides ways to use dynamic variables, directives, and so on through different syntax.

- **Rendering**
  Both Vue and React use virtual DOM. Both use server-side and client-side rendering. To enable server-side rendering on Vue, we can use either an additional package or a framework, for example, Nuxt.js. React needs to use the Next.js library for server-side rendering.

CSS

An important part of the front-end is the language in which we style the components. We considered multiple options.

Bootstrap

Bootstrap is a CSS library with predefined classes and styled components. Instead of programming the styling ourselves, we can simply use some of the prepared styling on our components. However, this might not be the best option because it might be difficult to achieve a custom style we designed [10].

Tailwind CSS

Tailwind CSS is a library that contains predefined classes at the lower level. Instead of programming the style, we can use multiple classes to achieve the desired style. However, this might also not be the best option. The desired style is usually achieved with multiple classes, resulting in an unnecessary large amount of code in defining the class for one tag. The same style can be achieved with one class in the core CSS. However, it prevents the user from programming the same style multiple times and repeating the code [11].

SASS and SCSS

SASS and SCSS are CSS preprocessor scripting languages. In addition to the core CSS, they provide features such as functions, variables, and cycles. In this project, we used SCSS because of its flexibility in defining styles, the ability to provide some generalization, and because of the additional features [12].

Nuxt

Nuxt is a framework based on Vue.js. It provides server-side rendering, simplified project routing, and performance enhancements. The simplified routing is based on predefined names of directories in which developers place the corresponding files.

We can enhance the performance by server-side rendering, static site generation, asynchronous loading, and other configurations tailored to our application [13].

2.4.4 Backend

Vite

Vite is a build tool that aims to create a more efficient development environment by optimizing the build of the application and thus making it faster. It was created by the creator of Vue.js, Evan You. Another widely used tool among developers is Webpack [14].

Using Node.js as API framework

The API in this application is programmed with Node.js. We call the API endpoints with a Nuxt feature /server/routes. This exposes the paths in the file as endpoints. Each route then defines the controller as its event handler. When the endpoint is called, it uses the controller. The controller then utilizes authorization middleware and executes the desired action by calling the repository. The repository is programmed for each entity. It uses a query builder to work with the database.

2.4.5 Database

We decided to use a relational database. We know the structure of the data we plan to work with. The relational structure provides sufficient organization of the data. It is also horizontally scalable, which is important for this application [15].

PostgreSQL

The database we decided to use is PostgreSQL. PostgreSQL is an open source object relational database that has a wide variety of functions and a long history of releases. PostgreSQL has a large community of users and is continuously supported. It also supports transactions, so unsuccessful database attempts can be safely rolled back [16].

ORMs and query builders

ORM is an object-relational mapping tool. It provides developers with predefined database queries, making the development process easier. However, the performance may not be optimal and the predefined environment might be limiting for custom queries. Popular ORMs are, for example, TypeORM and Prisma.

Query builders provide lower-level predefined queries than ORMs which is useful for building custom queries. This might be more difficult for developers.

Both of these types of tools usually provide security features such as automatic parameterized queries, escaping, sanitization, validation, and type-checking. We decided to choose a query builder due to its flexibility.

We selected a tool based on these criteria which we conducted after searching on forums and in their respective documentations. The completion of these criteria was based on personal judgement [17]. These criteria are:

- ORM - Whether the tool is an ORM or a query builder. The latter may provide more flexibility for developers, whilst ORM may provide more comfortable coding.
- Support - Whether tool is regularly updated by its developers and the developers community is large and active. This can predict its survivability in the future.
- Paid - Whether the tool is a paid service.
- Reliable - Whether the tool is known for an unannounced update or a lot of bugs. If not, the tool is more reliable.
- Popular - Whether the tool is widely used amongst the general developer's community. This can predict the scale of difficulty of hiring a new developer to join the team.
- Typescript - Whether the tool has good support for Typescript.

- We also aimed for a tool that supports most types of databases in case we need to change the database in the future.

Tool	ORM	Support	Paid	Reliable	Popular	Typescript
Kysely	✗	✓	✗	✓	Rising	✓
TypeORM	✓	✗	✗	✗	✓	✗
Prisma	✓	✓	✓	✗	✓	✓
Sequelize	✓	✓	✗	✓	Declining	✓
Knex.js	✗	✗	✗	✓	✓	✗

Table 2.2: Comparison of ORMs and query builders based on our criteria

Kysely

Kysely is a query builder with support for Typescript. It is well maintained. It was created by a creator of Knex, another query builder. The developer defines tables for each entity and also the operations that can be performed on that entity. We can then ban some CRUD operations from an entity if we want to do so. The permission of operation is achieved by defining the table type with Kysely's generics such as Selectable<NameOfTable> which can be selected [18].

2.4.6 Validation

It is crucial to validate data on both the front-end and back-end. The key features which we were looking for in a validation framework are localization and cross-field validation. Localization allows us to validate data for different countries, which might also have different formats, such as the date of pickup. Cross-field validation allows us to validate multiple fields and their relation, for example, password and password confirmation, or whether the street number exists on a street.

VeeValidate

VeeValidate is a front-end validation framework for Vue.js. The reason we chose this framework is that it has continuous support and is compatible with

field validation frameworks. It has a feature through which we can validate whole forms and also emit values from nested components. VeeValidate has built-in support for localization validation [19].

Yup

Yup is a field validation framework. It provides simple functions for single fields such as max() or email(). It is possible to define a custom validation function, which is how we satisfy the cross-field validation requirement [20].

2.4.7 Security

Keycloak

Keycloack is an open source identity management tool. Keycloak is based on standard protocols and supports OpenID Connect, OAuth 2.0, and SAML. It provides GUI through which the administrator can set up authorization and different users, roles, and their rights. The application and Keycloak then exchange the Keycloak-created tokens to authorize users [21].

2.4.8 State management

Pinia

Pinia is a Vue-based state management framework to handle the state and general information available throughout the application. Pinia uses stores to store information. The developer can then get the store inside a component and use its data [22].

2.4.9 Documentation

API documentation

Swagger

Swagger is API documentation tool. It accepts a file defining all endpoints and data types. Then it generates an HTML page based on the data in that file. The user can try calling the endpoints through that page [23].

2.4.10 Deployment

Docker

Docker provides a suite of development tools, services, trusted content, and automations, used individually or together, to accelerate the delivery of secure applications. It enhances the ability for developers to easily share and deploy their applications. Through a Docker file, we specify all technologies, libraries, versions, and actions needed. On execution, everything needed is installed

and ready for the application to run. The usage of Docker in this app can be seen on a deployment diagram [24].

Nginx

Nginx is a reverse proxy. It manages the traffic to our server and redirects requests to the correct docker instances. Nginx also works as a load balancer and provides our application with performance enhancement and security features [25].

2.4.11 Development environment

Github

We use GitHub for version control and project coordination. We plan to use GitLab CI/CD instead of Github Actions for testing and deployment purposes.

IntelliJ IDEA

IntelliJ IDEA is an IDE. The support for Vue and Node.js applications is sufficient. Since I am familiar with Java and usually use this IDE, therefore, I am the most familiar with it and that is why I chose it.

2.5 Summary

We did an extensive analysis by which we determined the customers' requirements and the frameworks and languages needed to execute our goal. The analysis included speaking with a few people in the fields of waste management and logistics to obtain the latest information. In this way, we discovered new features that we did not know about before and which are needed to achieve our goal. The decisions we made were made so that it fits the other applications' needs as well.

Chapter 3

Design

3.1 Prototype

3.1.1 Figma

Figma is a design and prototyping tool that supports the creation of interaction points and the presentation of the designed prototype. Figma has support for real-time collaboration with other colleagues. The Figma community creates free designs that other users can use in their designs. Figma also provides templates for various management tools and techniques, such as the Gantt diagram or brainstorming sessions. This tool is one of the most popular tools among UI/UX designers, competing with AdobeXD.

3.1.2 First prototype

We started with a low-fidelity prototype drawn on a piece of paper. The high-fidelity prototype was made in Figma. This prototype was created and then changed multiple times in regular sessions. The application will be used by various people and is not age-specific, so there is no need to adapt the design accordingly. The color scheme of this prototype is mainly green. This is because green is usually associated with nature, ecology, and sustainability, which matches the purpose of this application. We used the Coolors.co [26] tool to help balance the color palette. We chose some colors that are different from the color scheme. These colors are minimally used and are meant to allow users to easily distinguish between the different streams. However, these colors will vary depending on the country in which the user is, because the colors that indicate the stream also vary depending on the country.

The main goal was not to make the design skeuomorphic. A skeuomorphic design utilizes many shapes, shadows, and colors, which can be overwhelming to the user. Then it is more difficult for the user to see important information [27].

The prototype was initially made in flat design, which means that the color palette is limited and that both the colors, shades, and shapes are minimalist. Eventually, some of the features were highlighted using shadows and different colors. The minimalist yet highlighting design can be seen in the Material

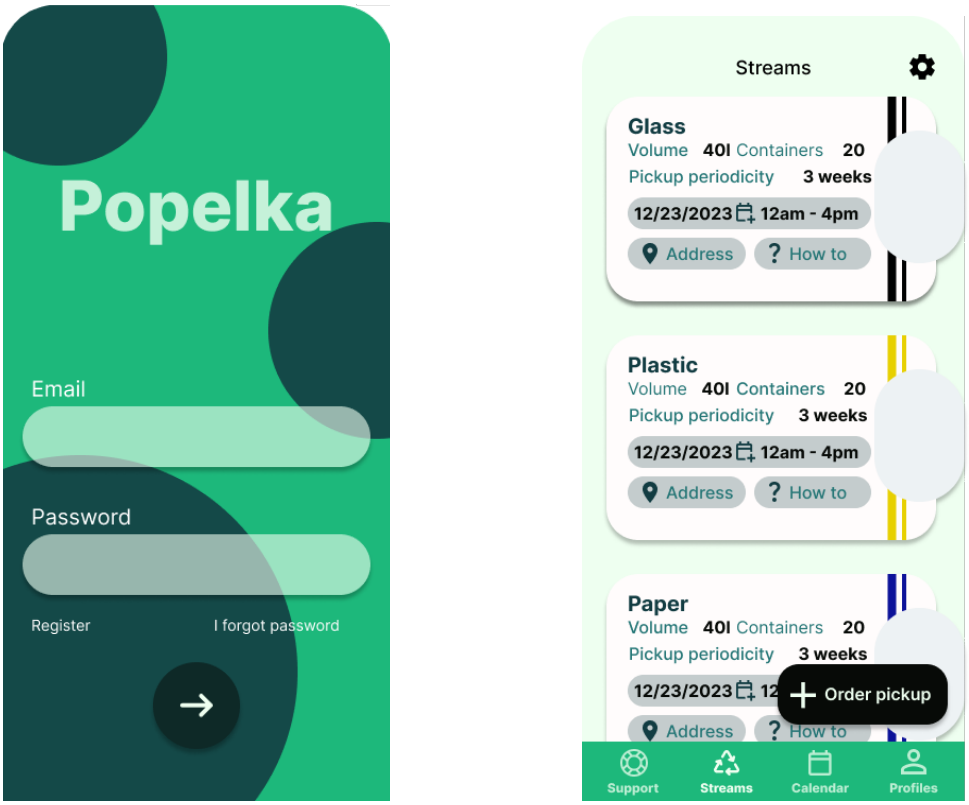

Figure 3.1: First prototype - Login and Pickups pages

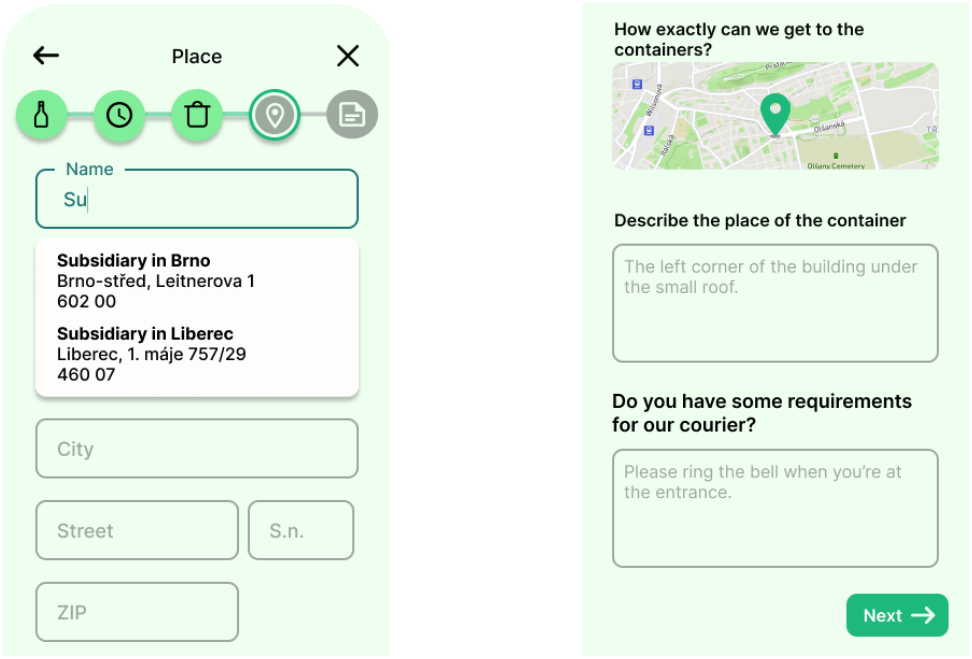

Figure 3.2: First prototype - Order place page

Design by Google.

We used the Iconify [28] plugin as a source for all icons. It is important to read what license the icons have, since it might be a paid license. We were focusing on using icons with the MIT license, which is free. As we were learning to use the tool, we started to design the pages right away. Later, we learned that it is best to design the individual components and make pages from those components since everything is unified.

3.1.3 Final prototype

We later decided that the application needs a clean design by a professional UI designer. This designer created a design that will eventually be used similarly across all applications in our system, making them look unified. The design features were taken from our first prototype and styled differently. Most of the development was done with the first design. We then implemented the front-end again according to the new design, which resulted in a time delay. Compared to the first design, the second one was made in a more flat design way, and its front-end effects were more simple; therefore, it was also easier to code.

Both of these prototypes can be found in the Attached files section as well as their links to Figma in the List of links section.

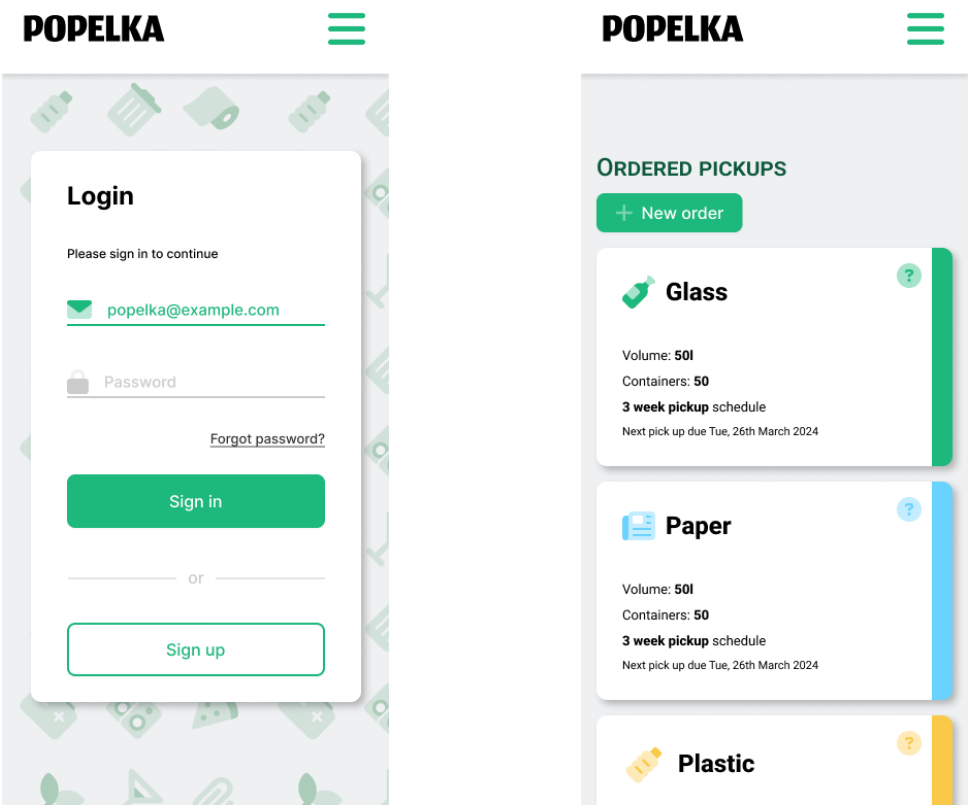

Figure 3.3: Final prototype - Login and Pickups pages

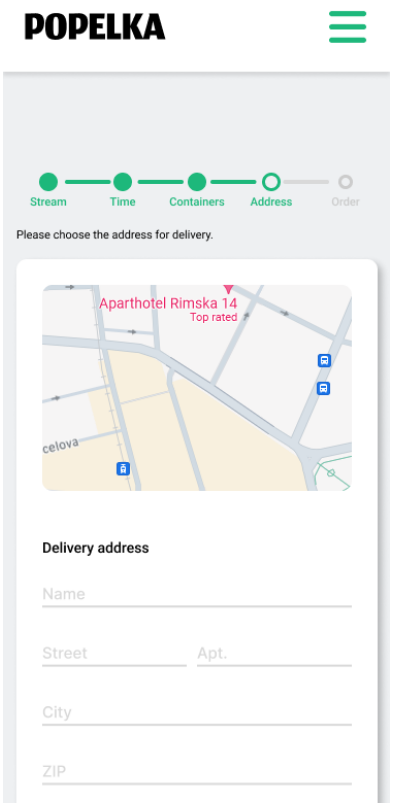

Figure 3.4: Final prototype - Order place page

3.2 Architecture

At the beginning of this project, the application was coded as a thick client - it contained both front-end and back-end components and it connected directly to a database. We managed to acquire a server on which our applications will run.

Later, it was decided to rewrite my application as a thin client. The back-end logic is then split between the services that run on the server, including the database. The thin client then authorizes with keycloak that runs separately on our server. The thin client sends a request to the server to get data from the database. Keycloak decides whether this client has the rights to do so. If it does, then the request proceeds to a service whose purpose is to retrieve those data from the server. The service retrieves those data and sends them back to the client. The distribution of requests that address different services is managed by a reverse proxy.

This division into the thin client and the services allows us to better distribute the workload between the services. It also provides better response time in case of heavy traffic, scalability, and separation of concerns.

The architecture and communication between the application and services are displayed in the deployment diagram and in the sequential diagram below.

3.3 Deployment diagram

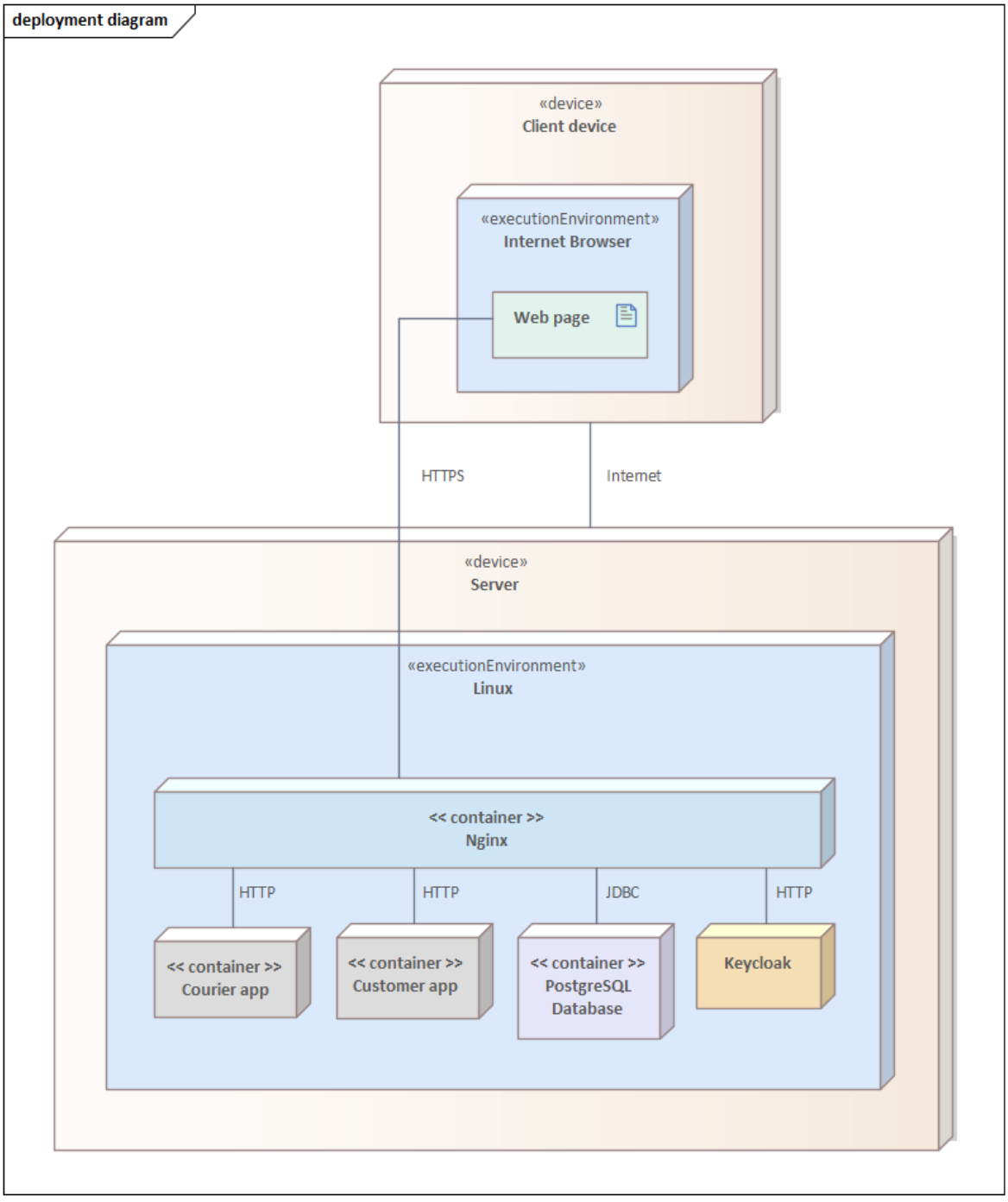

Figure 3.5: Deployment diagram

3.4 Sequential diagram

This diagram shows the communication between components when a user clicks on something that needs data from the database.

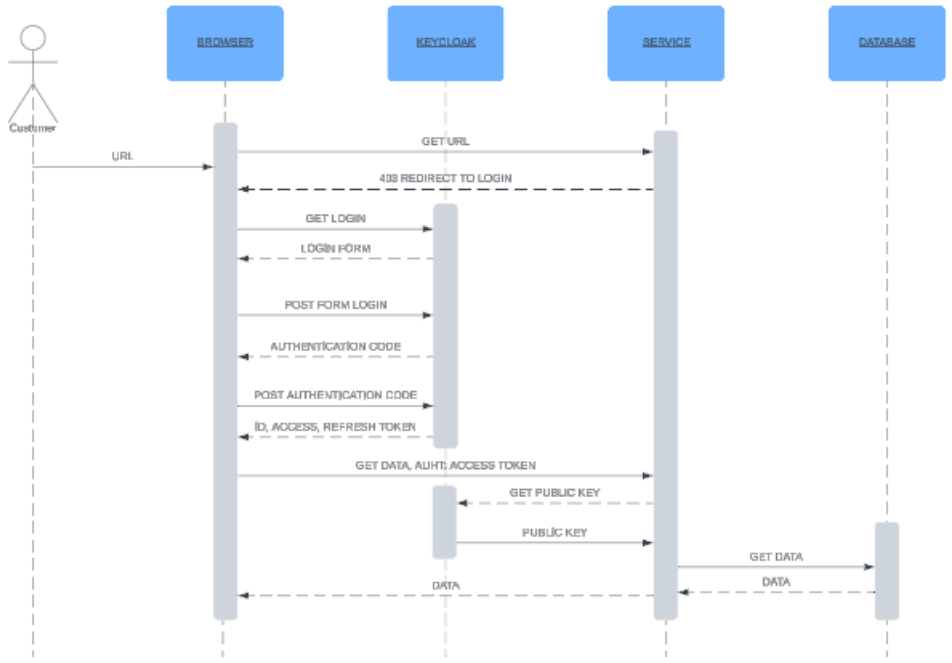

Figure 3.6: Sequence diagram

3.5 Class diagram

We designed the database that holds all the needed data. The design was changed many times as we gradually discovered different needs for desired features and possibilities of future feature development.

Enum tables are tables whose data will not change or be added to very often. It is important to keep records of changes to users' data in case there is a complication or miscommunication. That is why some entities also have a history version of their table. The history tables keep a deep copy of the original entity record, and the committed data in them cannot be changed. Whenever there is a change in the original entity data entry, the previous entry version is copied to the history table and is therefore no longer valid.

The email whitelist item contains emails that have been whitelisted for potential registration under the chosen company. The company has subsidiaries that have users that are called producers because they produce waste. The producer creates a pickup order that is connected to the order plan and place. Together, these three entities contain general order information. The

order plan then provides a template for the order implementation which is an instance of a pickup. For example, the customer orders a pickup for each Monday for the next month. The order implementation then copies this information and generates the record for each separate Monday in that month.

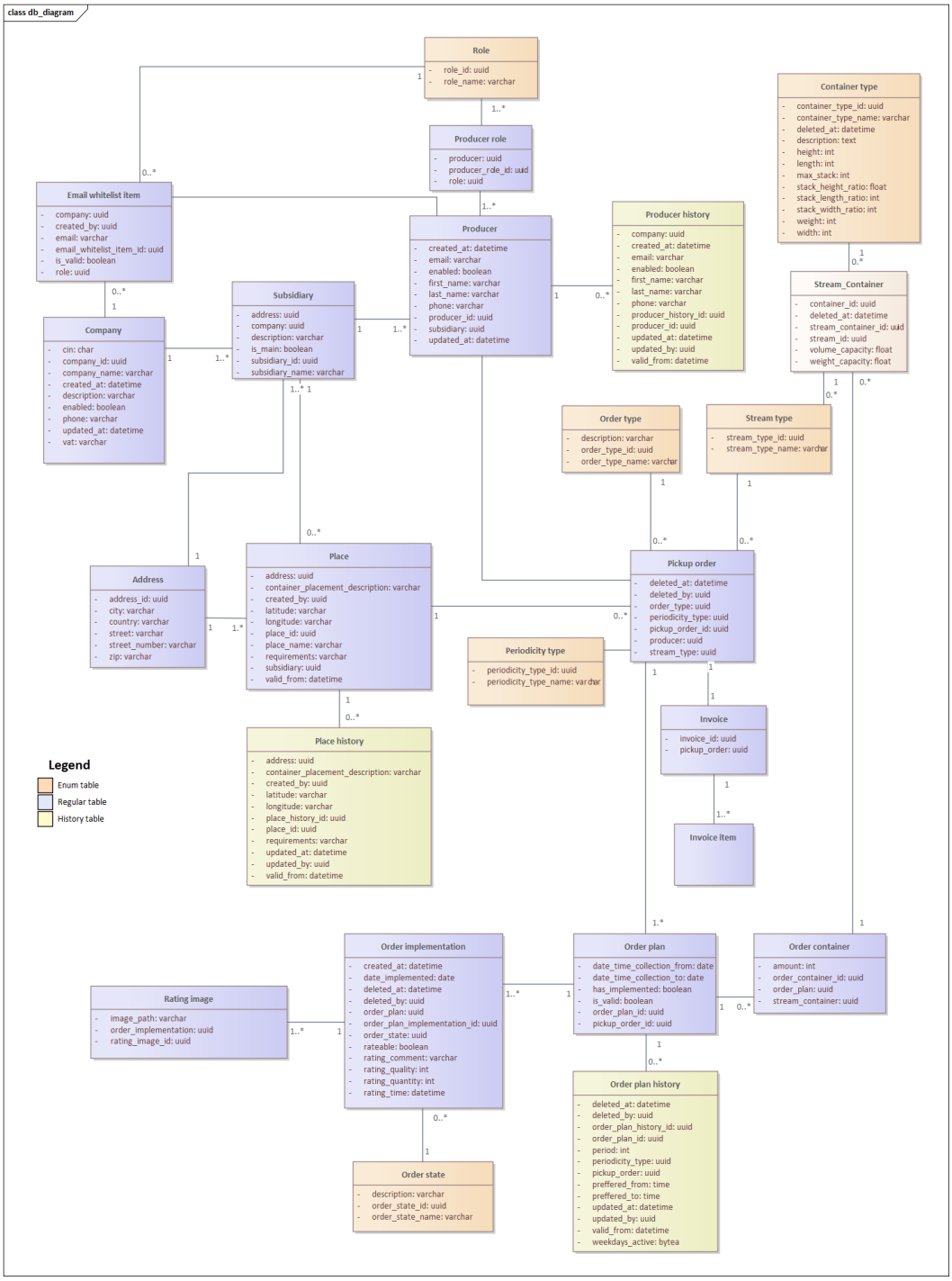

Figure 3.7: Class diagram

3.6 Summary

We met in numerous sessions to discuss the design of the application. We tried to follow the general guidelines for prototypes and eventually decided to change the prototype to a version made by a professional UI designer who used the original prototype as a list of features. The result of the prototype was a responsive design. The design theme will be used in all other applications.We designed the architecture always considering all the other applications and future potential extensions both to the project and to the developers' team that will continue to develop them. We created database on our server and populated it with mock data.

Chapter 4

Implementation

4.1 Code structure

The code structure utilizes the routing and organization of Vue and Nuxt. With these frameworks, we could simply name the folders the correct names and the routing would automatically work.

This is the code structure of the project. All folders except the language folder are recognized by Vue and Nuxt. Then these frameworks provide automatic routing for the files inside those directories. The following is an explanation of the contents of each folder.

Figure 4.1: Folders in project

- **components**
  The components folder contains Vue files that define all components. We can then take multiple components and make a page with them. This allows us to have clean code that does not repeat as the components are reusable.

- **language**
  The language folder contains json files in which are the translations for each text string in the application.

- **layouts**
  The layout is a template for a page. We can define what some pages have in common and place it in the layout file. We can then use the layout in our page by adding it to the page's meta. The default layout is refferenced to as default.vue and it does not need to be specified in page's meta.

- **pages**
  This is the directory that you place the websites pages into. The main page is index.vue. Nuxt allows developers to define a template page for an entity with a general id by defining a [id].vue. This page looks the same for entities with different id, but displays content specific to that id. Pages can be nested in directories, and the path follows as /directory/nestedDirectory.

- **plugins**
  This directory is used for storing plugin files, which are often from third-party libraries, for example, a localization library.

- **middleware**
  Middleware serves as a place for action or functions which occur between two actions. This is, for example, user authorization when we navigate from one page to another.

- **public**
  This directory usually contains style sheets, images, favicons and icons, these can be accessed from the browser. However, the icons in this project are in the components folder as stated above.

- **server**
  The server folder contains anything regarding database, entity types declarations and endpoints. This project will not need it as the back-end is being developed elsewhere right now. However, it was crucial at the beginning of this project to learn all the tools and also at the beginning there was no back-end outside of this application.

- **stores** We use Pinia for state management. Stores directory contains files that define files in which we can temporalily store values, e.g. values from creating order progress. These can be accessed everywhere in the application, but the values are not persistent.

The only files that are not coded by me are the files regarding Keycloak authorization which I received from a colleague who is working on his own application. I had to edit these files so that they work in my project.

The files in the root folder usually provide some kind of configuration. We used .env file to store global variables. The file is not pushed to Git, only its example without values as it would not be safe to do so. We wanted to use .gitlab-ci.yml for automatic testing and deployment; however, we decided that for now it will be done manually. The .yaml file describes the API and provides the necessary data for Swagger. However, the back-end was moved to the server so it is not needed anymore. Typescript allows developers to define a specific directive, a description of event, and what to do when that event occurs, e.g. clickOutsideDirective.ts file.

4.2 External APIs

Some of the parts that this application uses that are not standard libraries were developed by different developers. The following is a list of those projects.

4.2.1 Map

This plugin was used in the description of the place as part of the order creation process. The customer can select the exact coordinates of an entrance to the waste containers. On selection a marker shows up and the plugin emits a message event which contains the coordinate values. This event is then caught by the application, which displays this plugin in an iframe and validates these values separately from the rest of the form as the plugin is not a literal input.

4.2.2 Logr back-end

Logr back-end is a project to which the back end part of the thick client was transferred. It now acts as an API for the thin clients' front-end. The validation of requests in that project is still in progress. The project will likely be divided later to multiple services.

Links to these Github repositories are in the list of links in the appendix section.

4.3 Calendar

During the creation of the order, the customer must pick the date of pickup and the date of delivery of the containers. We used an existing component for this purpose. Its name is Vuedatepicker. It provides configuration as well as style options. It can also disable some dates and format them in a way the developer chooses [29].

4.4 Color-customizable icons

We needed a way to customize the color of the icons dynamically as there will be multiple themes in the future. The icons are usually imported as svg files and used in an img tag, however, this does not let us change the color dynamically. After different tries, we ended up creating each icon as a component which lets the user customize its attributes through the props field. There is still room for improvement as we did not find a way to program this without the code being repeated in each of these components. These icon components can be found in the components/icons folder.

4.5 Localization

The localization in this application is achieved with the i18n plugin. It lets the user define text in the application in many languages as well as format them in the usual language's format, e.g. date. The developer defines values in a separate file and then writes its key in the respective place in the component.

This goes hand in hand with the VeeValidate framework, which can display the validation warnings in different languages that were translated by their team. We can also create a specific validation rule that validates if the value is written in the correct format for each country [19, 30].

4.6 Deployment

The application is deployed on logr.dcgi.fel.cvut.cz. As of the time of this thesis submission, the application is deployed by cloning the code to the server and then executing the docker file. In the future, this process should be automated.

4.7 Summary

During implementation, we used the Vue and Nuxt tools to create the project. We ensured separation of concerns by changing the thick client to thin client and migrating the back-end to server as a separate application. That application was further developed by a different developer, and the application from this thesis retrieves the data using this back-end. The application also leverages two components that were created by either official vue developers and a fellow developer. Localization was implemented in the application. The application is deployed on a server provided by the supervisor's faculty.

Chapter 5

Testing

5.1 User testing

5.1.1 Tests execution

We needed to test the functionality and usability of our application. We chose user testing for this. Due to the nature of the application, we decided to create test tasks that the testers would have to complete without our guidance. The target group was people of all ages who work in some company and who are preferably not familiar with designing or implementing web applications. Such users would give us better feedback because using the application might be more difficult for them than for professional designers or developers. However, the tester's age might matter because certain generations might be more used to using applications. We tested the application with six testers. Each was asked several questions about the application. The following are their comments.

5.1.2 Results

Aesthetics

All testers said they liked the look of the design. The phrase they used the most was "clean design". The layout of the application was clear enough for them to understand it. One tester said that given the application's name "Popelka", they would expect a design with almost caricature images to imitate the fairy tale nature of the name.

Intuitiveness

The testers had no problem moving through the application. The main objective was to create an order and the testers did that without any complications. One tester expected that clicking on the order progress bar would take her to the respective page but it does not.

Comprehensibility

The comprehensibility of this design could be improved. The testers said they would like much more explanation in text regarding the different features and inputs. These are the ones they mentioned:

- Most testers thought that the place marker on the map in the order process locates the place where they live, however, it should locate the entrance to the containers.
- The monthly and weekly pickup periodicity should have an explanation text. Some testers were not sure how to properly pick the days for the monthly periodicity and what it would mean. The calendar should also start next month when it comes to the monthly subscription, or there can be just thirty days without naming the month.
- Some testers did not know what they should write in the additional requirements and description of the place. An example would be informative.
- According to two testers, the Thank you button at the end of the order creation does not clearly suggest that the order will be confirmed after clicking this button.
- There is no note on which units are the containers dimensions.
- One tester did not understand the difference between the start of the cycle and the date of the first pickup.

The testers understood how the rating from the courier works in the order detail. One tester mentioned that it looks like something for the user to click on and change. He suggested that the icons used for the rating are not the best option as they look too interactive.

Feature suggestions

The testers mentioned some features that could be implemented in the future and to which they are used to in most applications. These are the features:

- Automatic form values suggestions for palce based on where they put the marker.
- The country in place form should have predefined values from which customers can pick from.
- If the validation is not successful, the page should scroll to the input with the incorrect value.

5.2 Summary

We can conclude from the test results that the design of the application is aesthetically pleasing. However, there must be more explanation on what certain inputs mean in each of them. The users were confused when picking the parameters of the order. The navigation in the application is intuitive. The testers suggested many features that will improve the application and user experience.

Chapter 6
Conclusion

6.1 Summary

The assignment of this thesis was to design and implement an application through which users can order sorted waste pickup. The analysis and design were extensive and often were discussed with the team of developers who work on their own applications, so that the common parts, such as the database, would fit all needs. We decided on most of the frameworks used except for Vue.js. The application was first implemented as a thick client, discovering how all of these frameworks work together and what their possibilities are. The thick client was then divided into the thin client and the back-end on the server. The front end was first implemented following the first prototype, then reimplemented following the different prototype, and was deployed to the server. The applications function is not complete; however, this project defined core principles regarding analysis, requirements, technology stack, and design, which the other applications will follow and integrate into.

6.2 Future development

The goal of future development is to complete the application to make it a fully functional application. Then, this application will be exchanging data with other applications. We will add features to improve user experience and we will add application monitoring to better handle future errors and busy periods.

Bibliography

[1] "With your business towards zero waste? | How it works | Seenons — seenons.com," https://seenons.com/en/how-it-works/, [Accessed 23-05-2024].

[2] "Cyrkl - Digitální odpadové tržiště — cyrkl.com," https://www.cyrkl.com/cs, [Accessed 23-05-2024].

[3] "O NÁS | reKáva — rekava.cz," https://www.rekava.cz/o-nas, [Accessed 23-05-2024].

[4] "My WM - Apps on Google Play — play.google.com," https://play.google.com/store/apps/details?id=com.eBusiness&hl=en_US, [Accessed 23-05-2024].

[5] "JavaScript With Syntax For Types. — typescriptlang.org," https://www.typescriptlang.org/, [Accessed 23-05-2024].

[6] "Vue.js — vuejs.org," https://vuejs.org/guide/essentials/application.html, [Accessed 23-05-2024].

[7] "Getting started — vee-validate.logaretm.com," https://vee-validate.logaretm.com/v4/guide/composition-api/getting-started/, [Accessed 23-05-2024].

[8] "Options API vs Composition API - Vue School Articles — vueschool.io," https://vueschool.io/articles/vuejs-tutorials/options-api-vs-composition-api/, [Accessed 23-05-2024].

[9] "Comparison with Other Frameworks — Vue.js — v2.vuejs.org," https://v2.vuejs.org/v2/guide/comparison.html?redirect=true#React, [Accessed 23-05-2024].

[10] S. D. Smedt, "Bootstrap: Should I use it as a Developer? — simply_stef," https://medium.com/@simply_stef/bootstrap-should-i-use-it-as-a-developer-e7c7d0d26ff0, [Accessed 23-05-2024].

[11] T. Ryu, "Should You Use Tailwind CSS? — thomas.ryu," https://medium.com/@thomas.ryu/should-you-use-tailwind-css-83d519a29448, [Accessed 23-05-2024].

[12] "What's the difference between CSS, SASS, and SCSS? — dev.to," https://dev.to/mathlete/what-s-the-difference-between-css-sass-and-scss-g2b, [Accessed 23-05-2024].

[13] "Pages · Nuxt Kit — nuxt.com," https://nuxt.com/docs/api/kit/pages, [Accessed 23-05-2024].

[14] "Vite — vitejs.dev," https://vitejs.dev/guide/, [Accessed 23-05-2024].

[15] "Relational vs. Non-relational Database: The Difference Explained — coursera.org," https://www.coursera.org/articles/relational-vs-non-relational-database, [Accessed 23-05-2024].

[16] "PostgreSQL 16.3 Documentation — postgresql.org," https://www.postgresql.org/docs/16/index.html, [Accessed 23-05-2024].

[17] "SQL vs ORMs vs Query Builders | Compare | Prisma's Data Guide — prisma.io," https://www.prisma.io/dataguide/types/relational/comparing-sql-query-builders-and-orms, [Accessed 23-05-2024].

[18] "Introduction | Kysely — kysely.dev," https://kysely.dev/docs/intro, [Accessed 23-05-2024].

[19] "Overview — vee-validate.logaretm.com," https://vee-validate.logaretm.com/v4/guide/overview/, [Accessed 23-05-2024].

[20] "GitHub - jquense/yup: Dead simple Object schema validation — github.com," https://github.com/jquense/yup, [Accessed 23-05-2024].

[21] K. Team, "Keycloak — keycloak.org," https://www.keycloak.org/, [Accessed 23-05-2024].

[22] "State | Pinia — pinia.vuejs.org," https://pinia.vuejs.org/core-concepts/state.html, [Accessed 23-05-2024].

[23] "API Documentation & Design Tools for Teams | Swagger — swagger.io," https://swagger.io/, [Accessed 23-05-2024].

[24] "Home — docker.com," https://www.docker.com/, [Accessed 23-05-2024].

[25] "NGINX Documentation — docs.nginx.com," https://docs.nginx.com/, [Accessed 23-05-2024].

[26] "Coolors - The super fast color palettes generator! — coolors.co," coolors.co, [Accessed 23-05-2024].

[27] "Skeuomorphism — nngroup.com," https://www.nngroup.com/articles/skeuomorphism/, [Accessed 23-05-2024].

[28] I. OÜ, "Iconify — iconify.design," https://iconify.design/, [Accessed 23-05-2024].

[29] "Props - General configuration | Vue Datepicker — vue3datepicker.com," https://vue3datepicker.com/props/general-configuration/, [Accessed 24-05-2024].

[30] "Getting started | Vue I18n — vue-i18n.intlify.dev," https://vue-i18n.intlify.dev/guide/essentials/started.html, [Accessed 24-05-2024].

Appendix A

List of links

- First prototype
- Final prototype
- [Map]
- [Logr back-end]

Appendix B

List of attached files

- psasInterview.pdf
- firstPrototype.pdf
- finalPrototype.pdf
- Source code