**VILNIUS UNIVERSITY**

**FACULTY OF MATHEMATICS AND INFORMATICS**

**SOFTWARE ENGINEERING STUDY PROGRAMME**

Laboratory work

# Software system design

## Programų sistemos projektavimo dizainas

Danielius Vičkačka, Paulius Navikas, Dovydas Girskas, Marius Česnauskas, Ruslanas Maksimkinas

Supervisor : Vasilij Savin

**Vilnius**
**2025**

# Contents

# 1 Introduction

## 1.1 Project description

This project aims to solve the problem of businesses using outdated technology and offer new and better ways to control orders in a food-service workplaces ( like restaurants ) or simple service workplaces ( like barbershops ).

In the next sections we aim to explain the general flow of the systems, the general management of the data and how these systems interact with each other.

# 2 Business flows and wireframes

## 2.1 Food-service business system

The food service business focuses on servicing the customers on the spot. For that there is a menu which the worker can choose items from for quicker ordering and calculation.

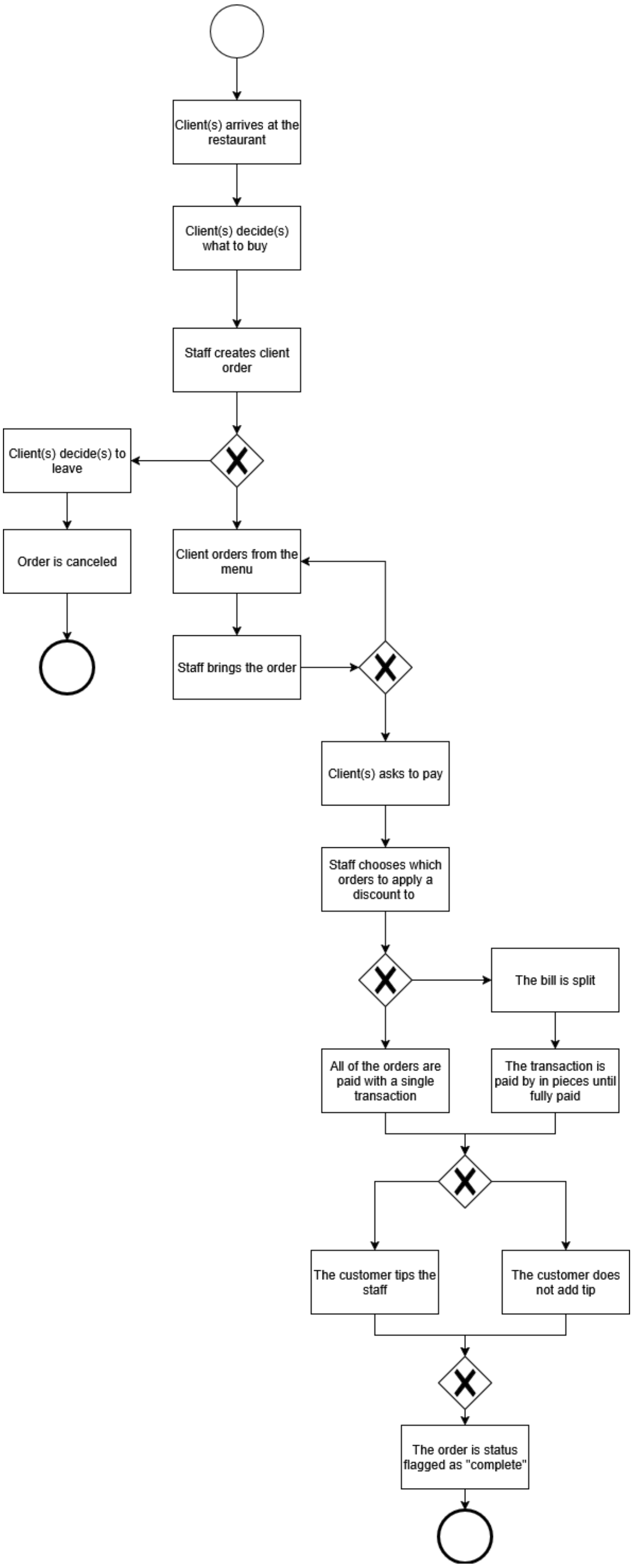
The owner has all of the functions a worker has with additional business management systems.

**The owner can:**

- Create a dish type or delete an existing dish
- Edit existing dish's name/price/description/discount
- Add new workers / delete existing
- Edit current worker's information
- Create new orders / cancel existing orders that are not yet paid for
- Edit currently ongoing orders
- See the history of previous orders and refund them

**The worker can:**

- Create new orders / cancel existing orders that are not yet paid for
- Edit currently ongoing orders
- See the history of previous orders and refund them

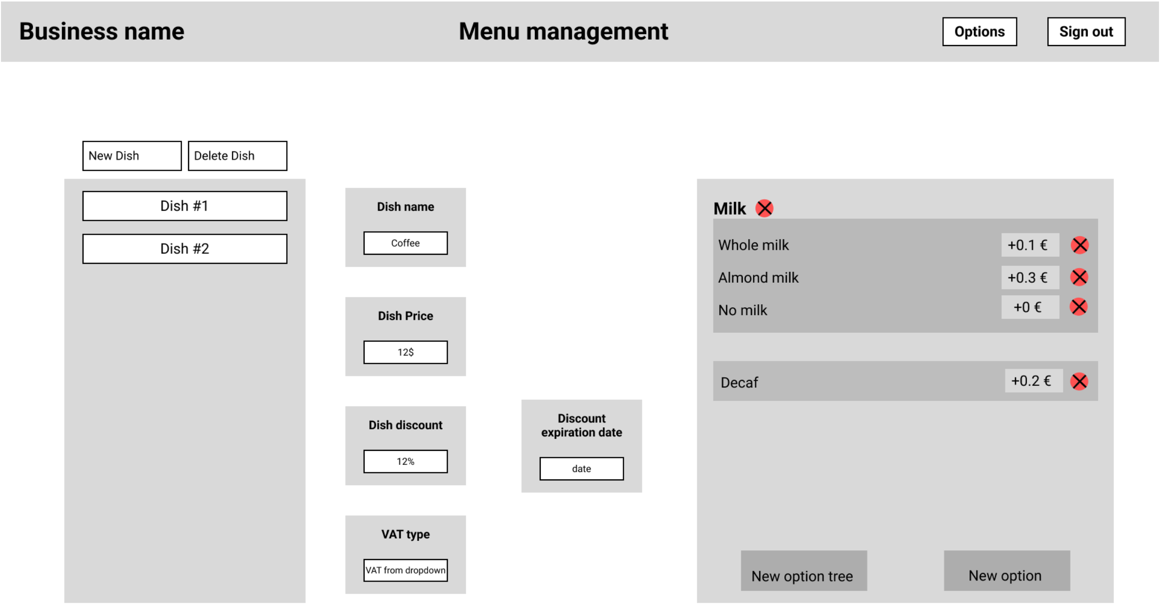

Client(s) arrives at the restaurant
Client(s) decide(s) what to buy
Staff creates client order
Client(s) decide(s) to leave
Order is canceled
Client orders from the menu
Staff brings the order
Client(s) asks to pay
Staff chooses which orders to apply a discount to
The bill is split
All of the orders are paid with a single transaction
The transaction is paid by in pieces until fully paid
The customer tips the staff
The customer does not add tip
The order is status flagged as "complete"

***1 figure.*** *Food service worker workflow*

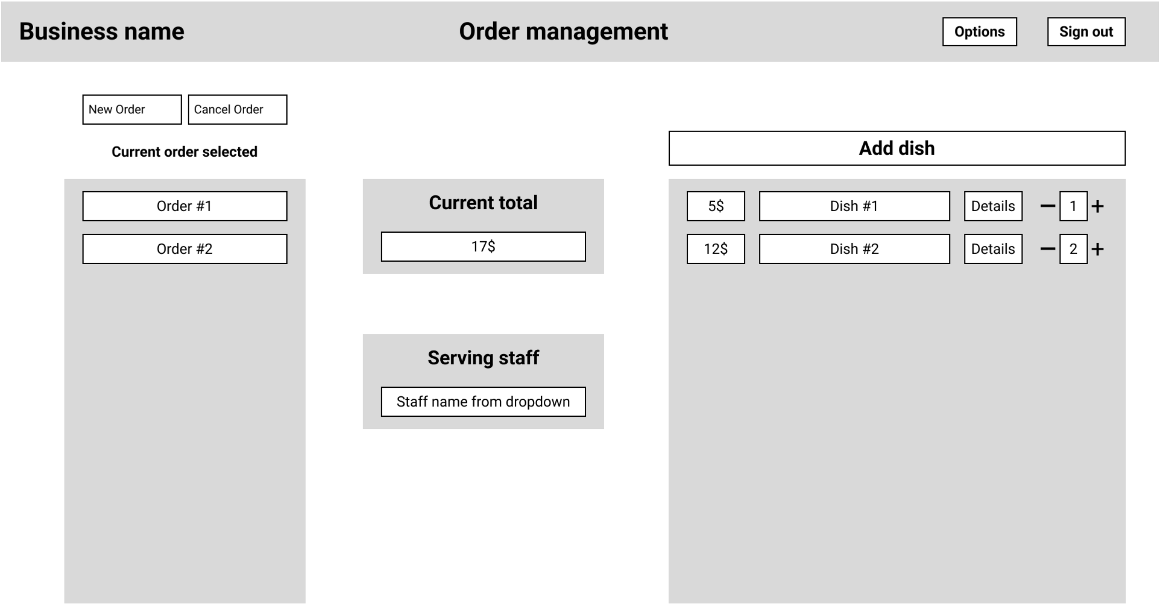
In 1 figure. we take a look at the general workflow the worker in a food-service business has when adding new orders or finalizing old ones.

The worker services the customer by first creating the client order, this way a new order is created in the system. After creating the order, at any point it can be edited to add more orders until the customer is ready to pay. When the transaction is complete the order is finalized and marked as ”complete”.

Order splitting is handled by selecting which items are to be paid for, until all of the items are all paid.

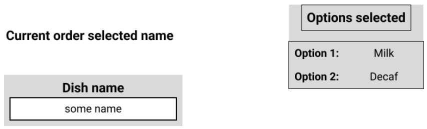


*__2 figure.__ Menu management wireframe*



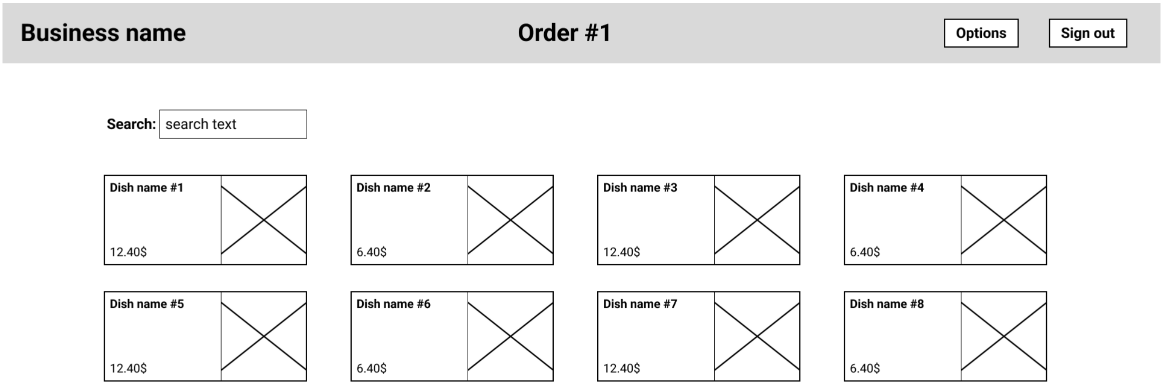
*__3 figure.__ Order management wireframe*

Order management is accessed window is accessed primarily by workers during work hours to create new orders.



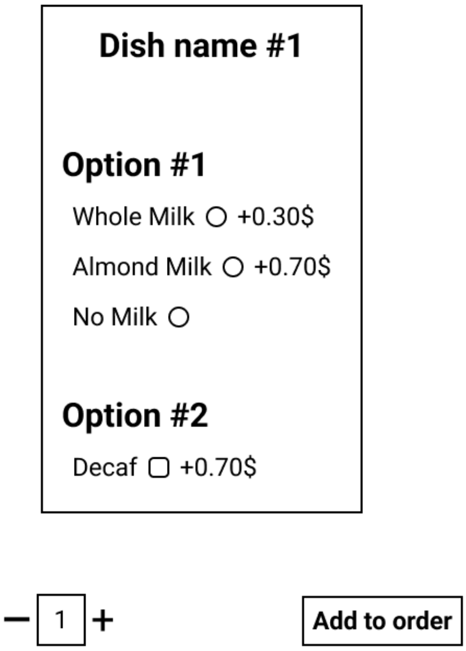
***4 figure.*** *Order management popup wireframe*

In 4 figure. we can see the popup which will provide additional information about the particular dish.



***5 figure.*** *Add order to the dish wireframe*

This menu will hold all the possible choices from the menu, by clicking on the specific dish the user is shown a popup for options.



***6 figure.*** *Wireframe for a popup when adding options to a dish in an order*

This is the wireframe with all the options a dish may have.

## 2.2 Appointment based business system

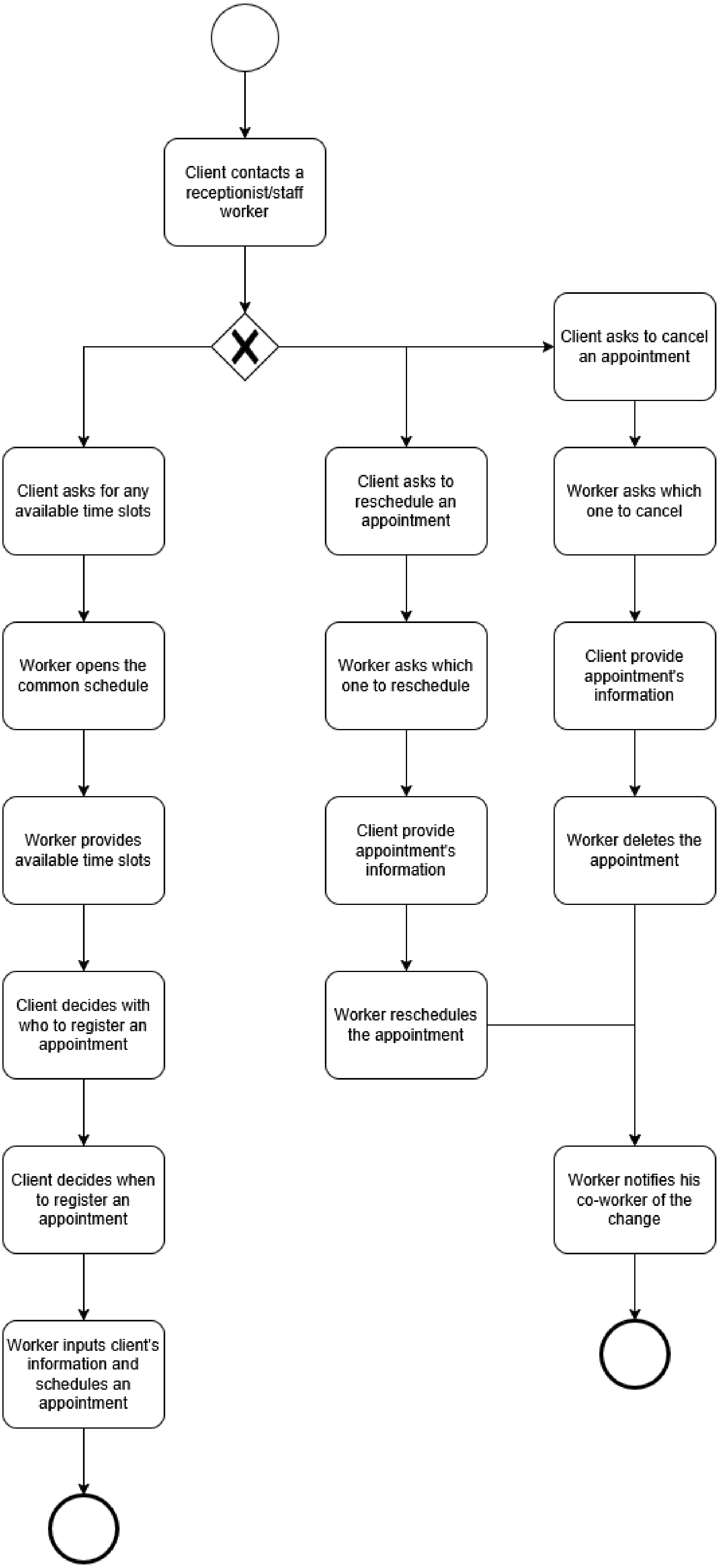
Similar to Food-Service business with a few distinctions, primarily the service business configures which services are available for appointments, while also having the ability to see and configure the schedule to add appointments for specific workers.

**The owner can:**

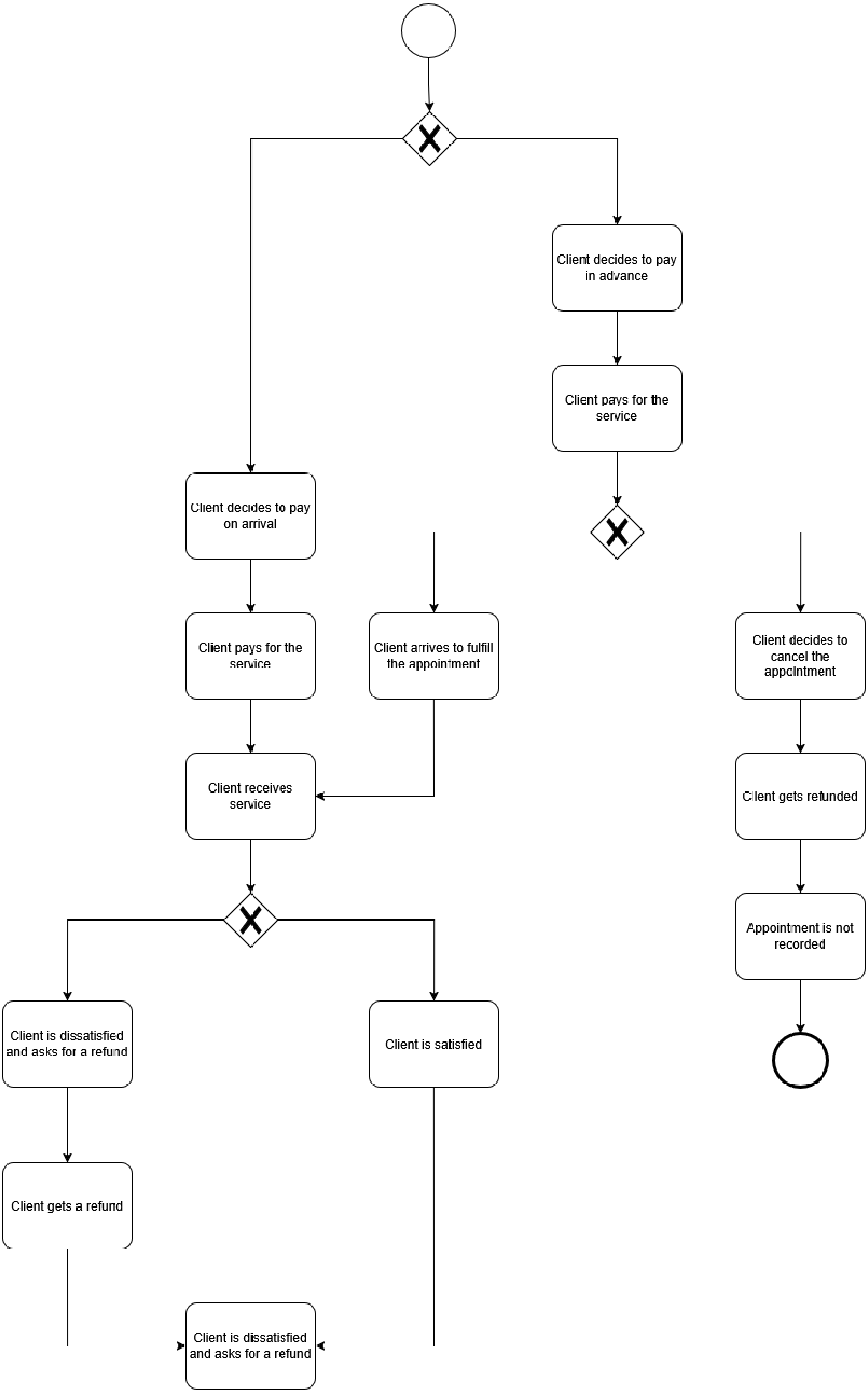
- Create a service type or delete an existing type
- Edit existing service's name/price/description/discount/duration
- Add new workers / delete existing
- Edit current worker's information
- Create new appointments / cancel existing appointments that are not yet paid for
- Edit currently scheduled appointments
- See the history of previous appointments and refund them

**The worker can:**

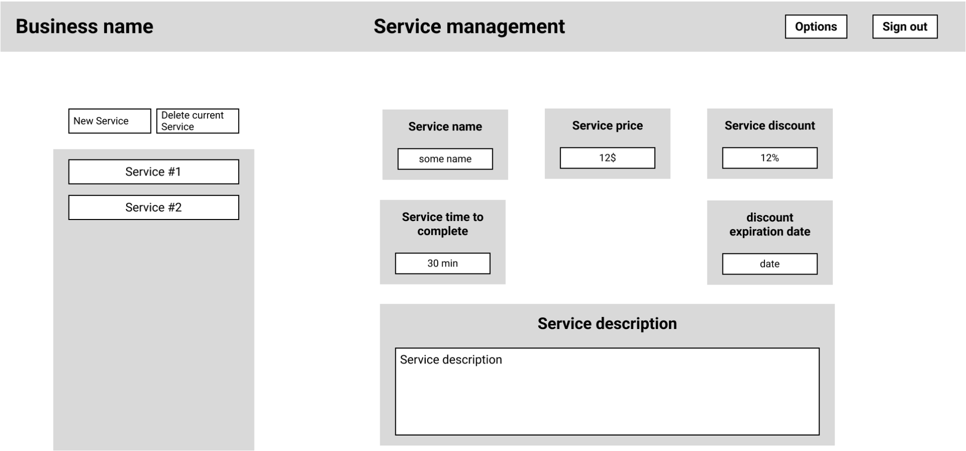
- Create new appointments / cancel existing appointments that are not yet paid for
- Edit currently scheduled appointments
- See the history of previous appointments and refund them


Client contacts a receptionist/staff worker
Client asks for any available time slots
Worker opens the common schedule
Worker provides available time slots
Client decides with who to register an appointment
Client decides when to register an appointment
Worker inputs client's information and schedules an appointment
Client asks to reschedule an appointment
Worker asks which one to reschedule
Client provide appointment's information
Worker reschedules the appointment
Client asks to cancel an appointment
Worker asks which one to cancel
Client provide appointment's information
Worker deletes the appointment
Worker notifies his co-worker of the change

***7 figure.*** *Appointment reception*


Client decides to pay in advance
Client pays for the service
Client decides to pay on arrival
Client pays for the service
Client arrives to fulfill the appointment
Client decides to cancel the appointment
Client receives service
Client gets refunded
Appointment is not recorded
Client is dissatisfied and asks for a refund
Client is satisfied
Client gets a refund
Client is dissatisfied and asks for a refund

***8 figure.*** *Appointment payment workflow*



Service name

some name

Service price

12$

Service discount

12%

Service time to complete

30 min

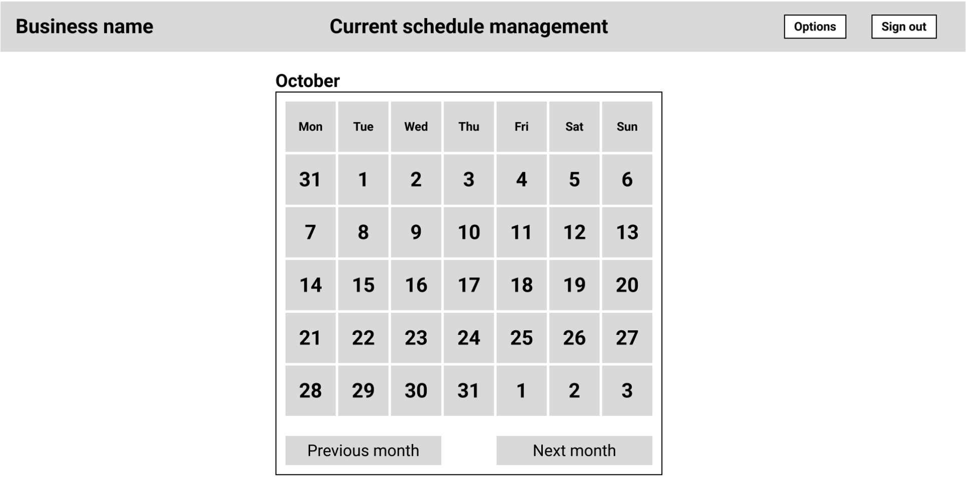
discount expiration date

date

Service description

Service description

*__9 figure.__ Service management wireframe*

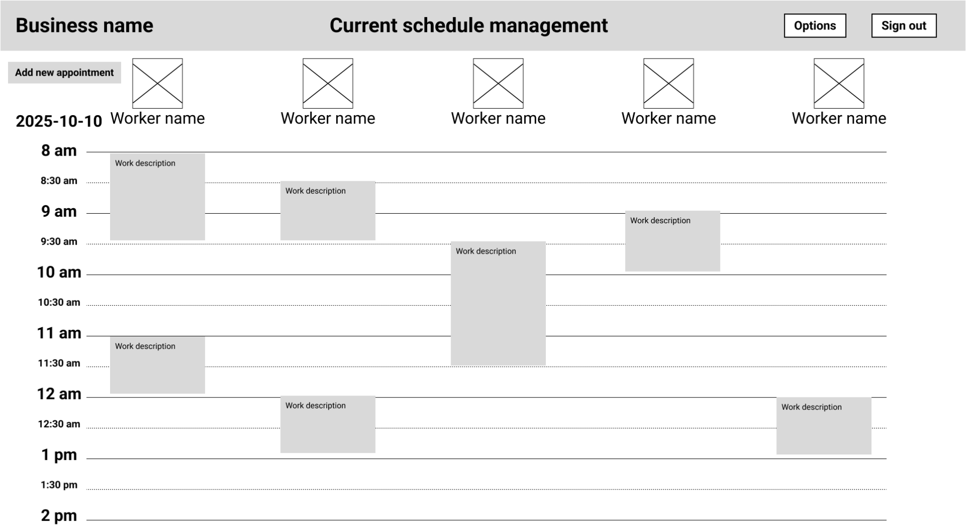


October

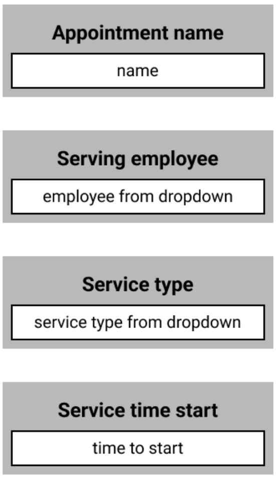
| Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|
| 31 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 | 1 | 2 | 3 |

*__10 figure.__ Schedule overview wireframe*

Here is the window both owner and the worker can see, the schedule is manipulated through a virtual calendar, by selecting a day both the worker and the owner can see that day’s scheduling for each worker.

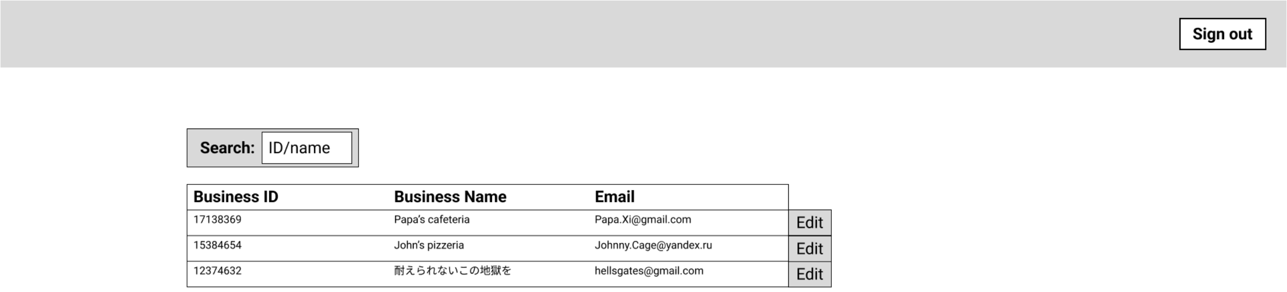


***11 figure.*** *Day scheduling wireframe*



***12 figure.*** *Add new appointment wireframe*

## 2.3 Super admin system

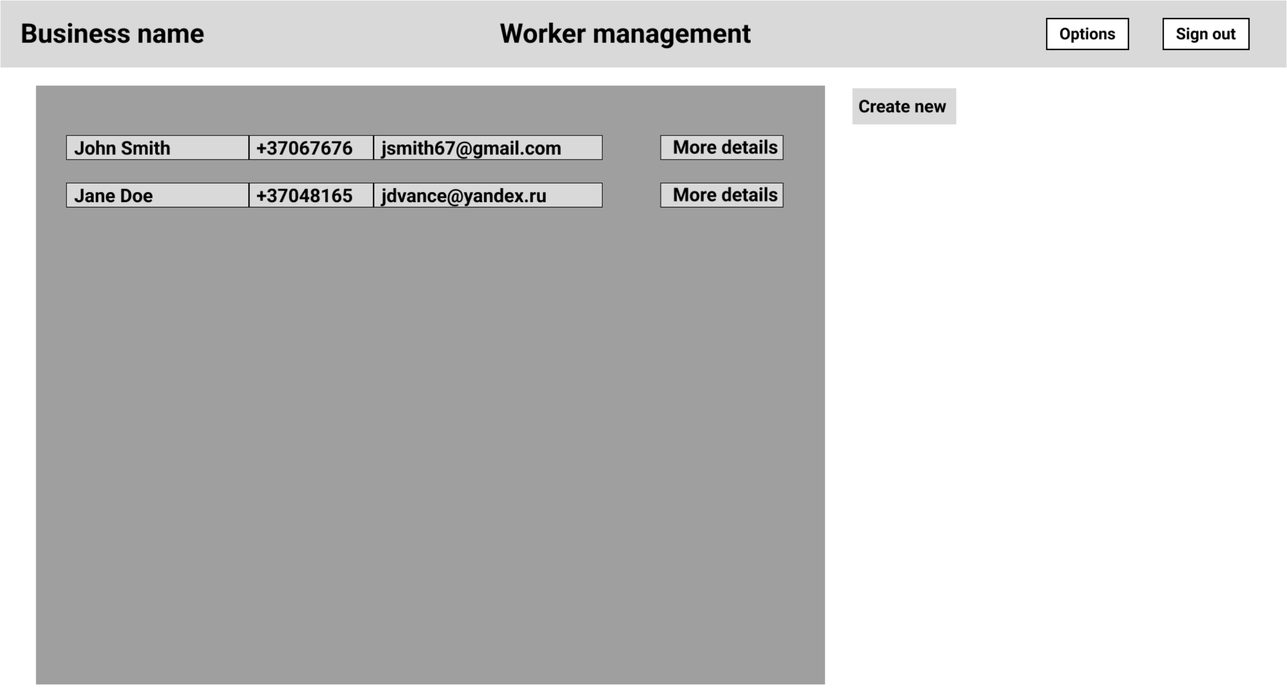


| Business ID | Business Name | Email | |
|---|---|---|---|
| 17138369 | Papa’s cafeteria | Papa.Xi@gmail.com | Edit |
| 15384654 | John’s pizzeria | Johnny.Cage@yandex.ru | Edit |
| 12374632 | 耐えられないこの地獄を | hellsgates@gmail.com | Edit |

***13 figure.*** *Wireframe of superadmin’s view of businesses*

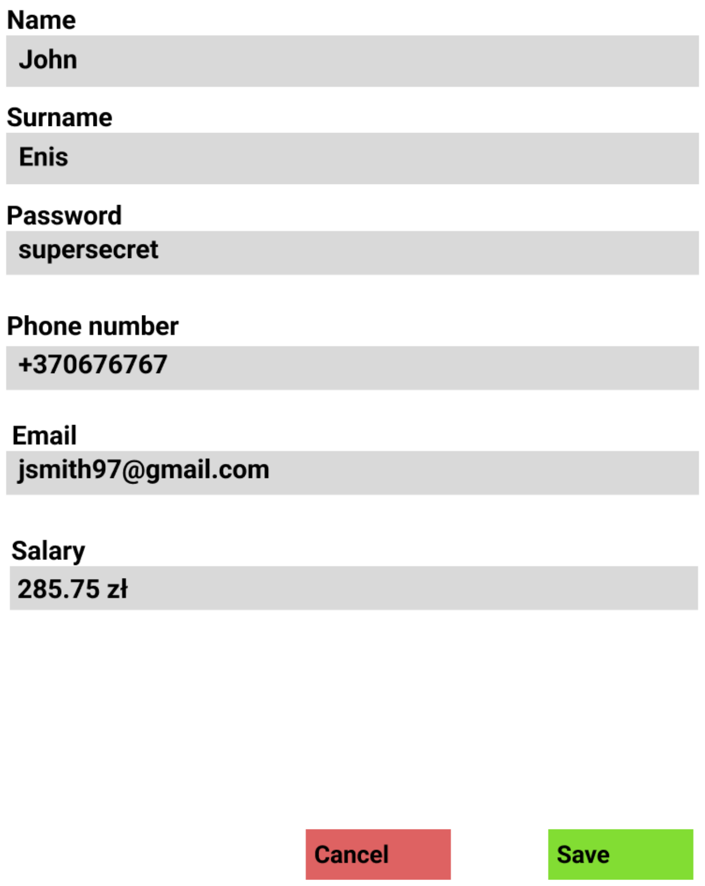
This is how a superadmin sees businesses’s table. He chooses among ones in the list to edit in case of a technical problem.

## 2.4 Worker Account management system



| | | |
|---|---|---|
| John Smith | +37067676 | jsmith67@gmail.com |
| Jane Doe | +37048165 | jdvance@yandex.ru |

***14 figure.*** *Worker management screen*



Name
John

Surname
Enis

Password
supersecret

Phone number
+370676767

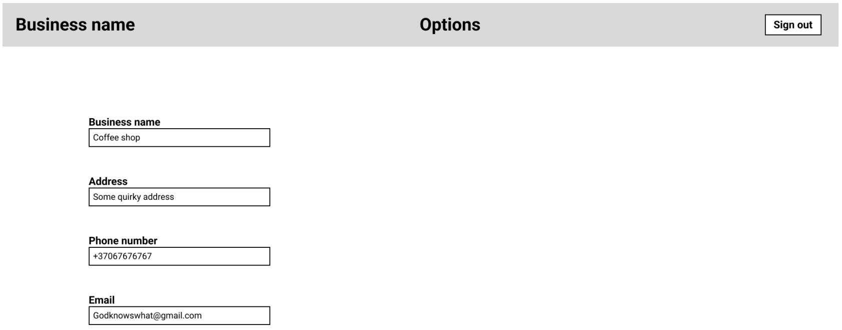
Email
jsmith97@gmail.com

Salary
285.75 zł

***15 figure.*** *Worker management pop-up*

Pop-up when clicking ’More details’ or ’Create new’. This pop-up offers basic CRUD options for workers in the business.

## 2.5 Worker Account management system



Business name
Coffee shop

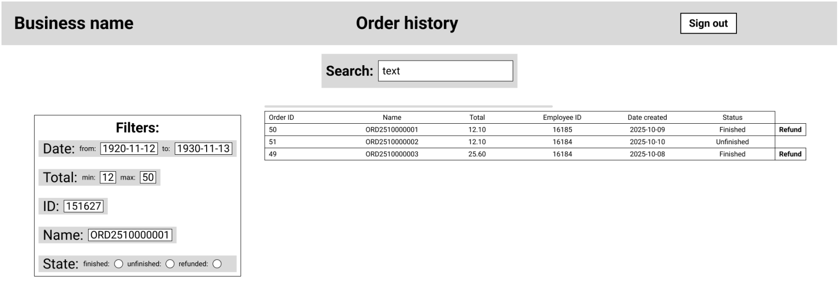
Address
Some quirky address

Phone number
+37067676767

Email
Godknowswhat@gmail.com

***16 figure.*** *Worker management pop-up*

## 2.6 Order/Appointment history preview



Search: text

Filters:

Date: from: 1920-11-12 to: 1930-11-13

Total: min: 12 max: 50

ID: 151627

Name: ORD2510000001

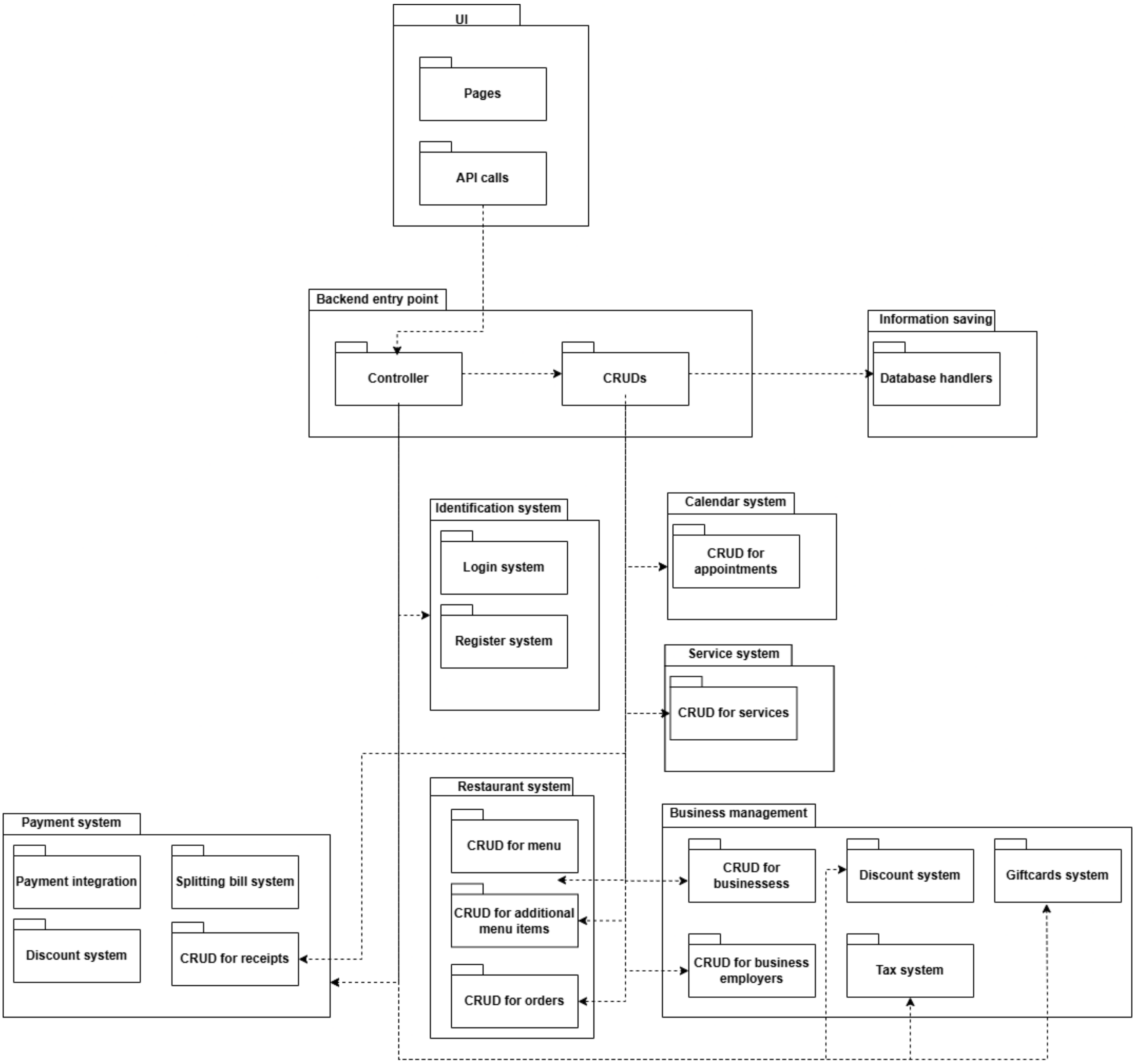
State: finished: ○ unfinished: ○ refunded: ○

| Order ID | Name | Total | Employee ID | Date created | Status | |
|---|---|---|---|---|---|---|
| 50 | ORD2510000001 | 12.10 | 16185 | 2025-10-09 | Finished | Refund |
| 51 | ORD2510000002 | 12.10 | 16184 | 2025-10-10 | Unfinished | |
| 49 | ORD2510000003 | 25.60 | 16184 | 2025-10-08 | Finished | Refund |

***17 figure.*** *History wireframe*

17 figure. shows how the history table for worker or owner looks like. The table can be navigated horizontally and vertically. The history is limited by default via paging that can be modified via API.

# 3 High level architecture



***18 figure.*** *High-level system package diagram*

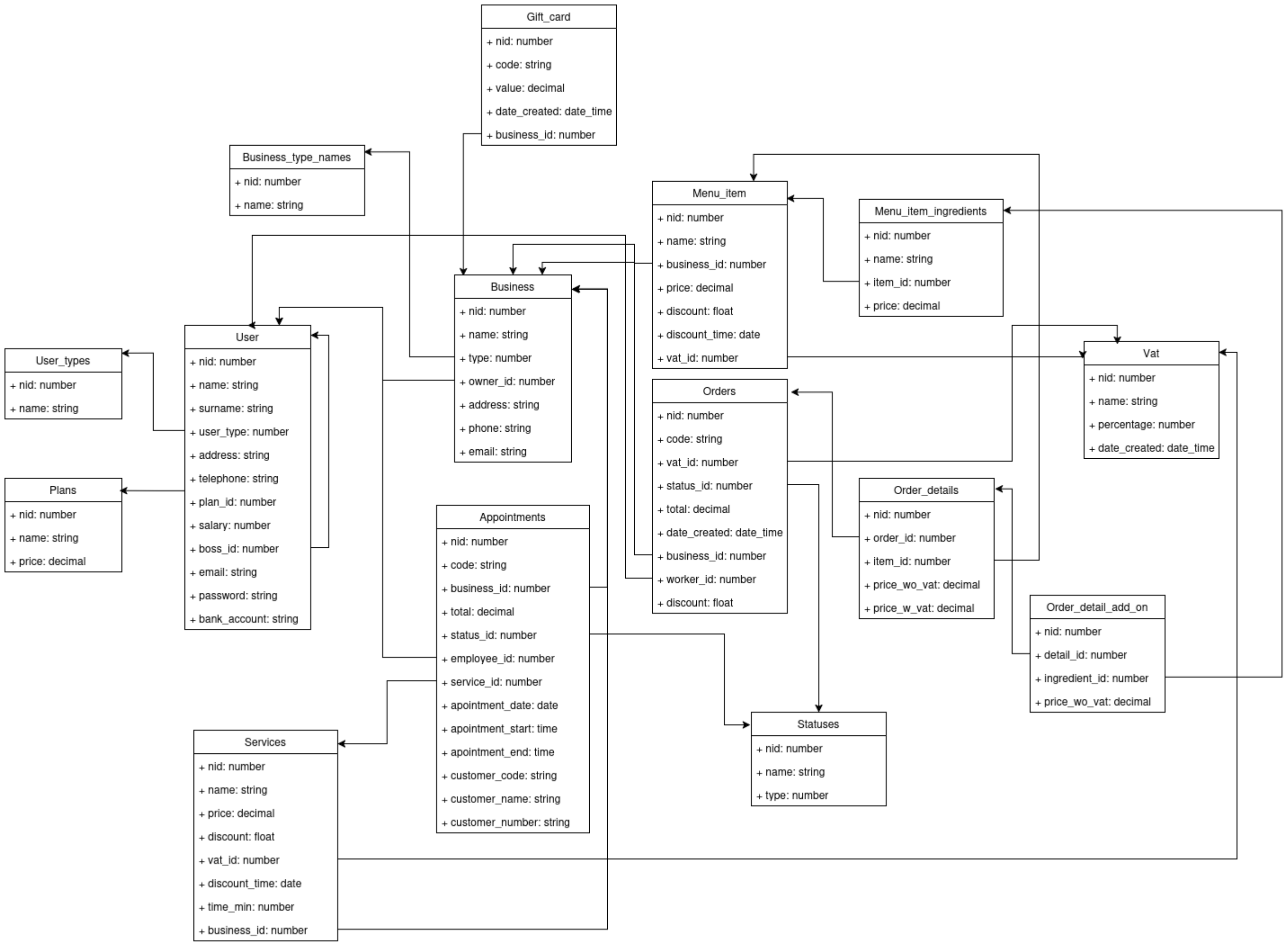
The system architecture is organized into several layers, each responsible for a specific part of the application's functionality. The goal of the project is to modernize how food-service and service-based businesses handle orders, payments, and scheduling.

- **User Interface (UI)** – includes pages for employees, managers, and administrators. Employees use it to handle orders, appointments, and payments. Managers and admins use it to view data, register businesses, and control settings. The UI communicates with the backend through API calls.
- **Backend Entry Point** – acts as the main connection between the UI and the backend logic. It contains the Controller, which handles incoming requests, and CRUD modules, which perform

operations on data. This layer ensures smooth communication between the UI and the internal systems.

- **Identification System** – manages user authentication. It includes a Login System and a Register System to handle secure access for employees and business owners.
- **Calendar System** – manages scheduling by providing a CRUD for appointments, allowing employers to create, view, edit, and delete appointment records.
- **Service System** – handles the services offered by the business, with a CRUD for services used to define and manage what the business provides (e.g. haircut types).
- **Restaurant System** – focuses on foodservice-specific needs, including CRUD for menu and CRUD for additional menu items, which allow restaurants to manage their offerings and extras.
- **Business Management** – provides tools for managing organizations. It includes CRUD for businesses and CRUD for business employers, plus modules for handling Discounts, Giftcards, and Taxes.
- **Payment System** – manages all payment-related features. It includes Payment Integration, Splitting Bill System, and a Discount System, which are connected to both restaurant and business management systems.
- **Information Saving** – represents the database and data handling part of the system. Database Handlers are responsible for saving and retrieving information used by the CRUDs and other backend modules.

# 4 Data model for entity used in the system



***19 figure.*** *Database class diagram*