VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
SOFTWARE ENGINEERING

STAČIAKAMPIAI

Technical design

Author: Miglė Kirilovaitė, Arnas Variakojis, Džiugas Budreika,
Evelina Temnolonskytė, Domantas Pilipavičius

Vilnius – 2024

Contents

1 Introduction

In this assignment, we are developing a Point of Sale (PoS) system as part of a Software as a Service startup. This software is tailored for businesses in the catering (bars, cafes, restaurants) and beauty (barbershops, hairdressers, spas) sectors. The system is designed to be used exclusively by business employees and owners. Customers only provide their data to the business employees, specifically payment information and contact details. Only registered businesses are allowed to use the system. To ensure security, business owners must submit their business related details during registration via email to an admin for approval. Once verified, business owner can input the services or product they intend to sell as well as the service charges. Additionally, businesses can set up discounts or giftcards, if any are offered. The business owner is also responsible for creating employee accounts so that staff members can access and use the system. The PoS system will handle transactions, manage receipts, inventory and process orders and appointments, while also tracking other relevant business data.

1.1 Choices

a) The application will be designed as a mix between thin-client and fat-client platform. This architectural choice is driven by the necessity for rapid updates and streamlined maintenance, essential for adapting to the dynamic requirements of the PoS business market, but gives the ability to recover some details upon unexpected system shutdowns.

b) The operations within this framework will be characterized by their fine-grained mixed with coarse-grained nature, as the complexity of commands will be minimal, though some of them will be slightly more complex. Consequently, all interactions can be effectively managed through fundamental CRUD (Create, Read, Update, Delete) operations, ensuring straightforward and efficient data manipulation, with some of them having a combination for these calls (C+U for example).

c) When creating the system, choose whether the Super Admin can invoke all API calls or not.

2 Business flows

2.1 Appointment registration process

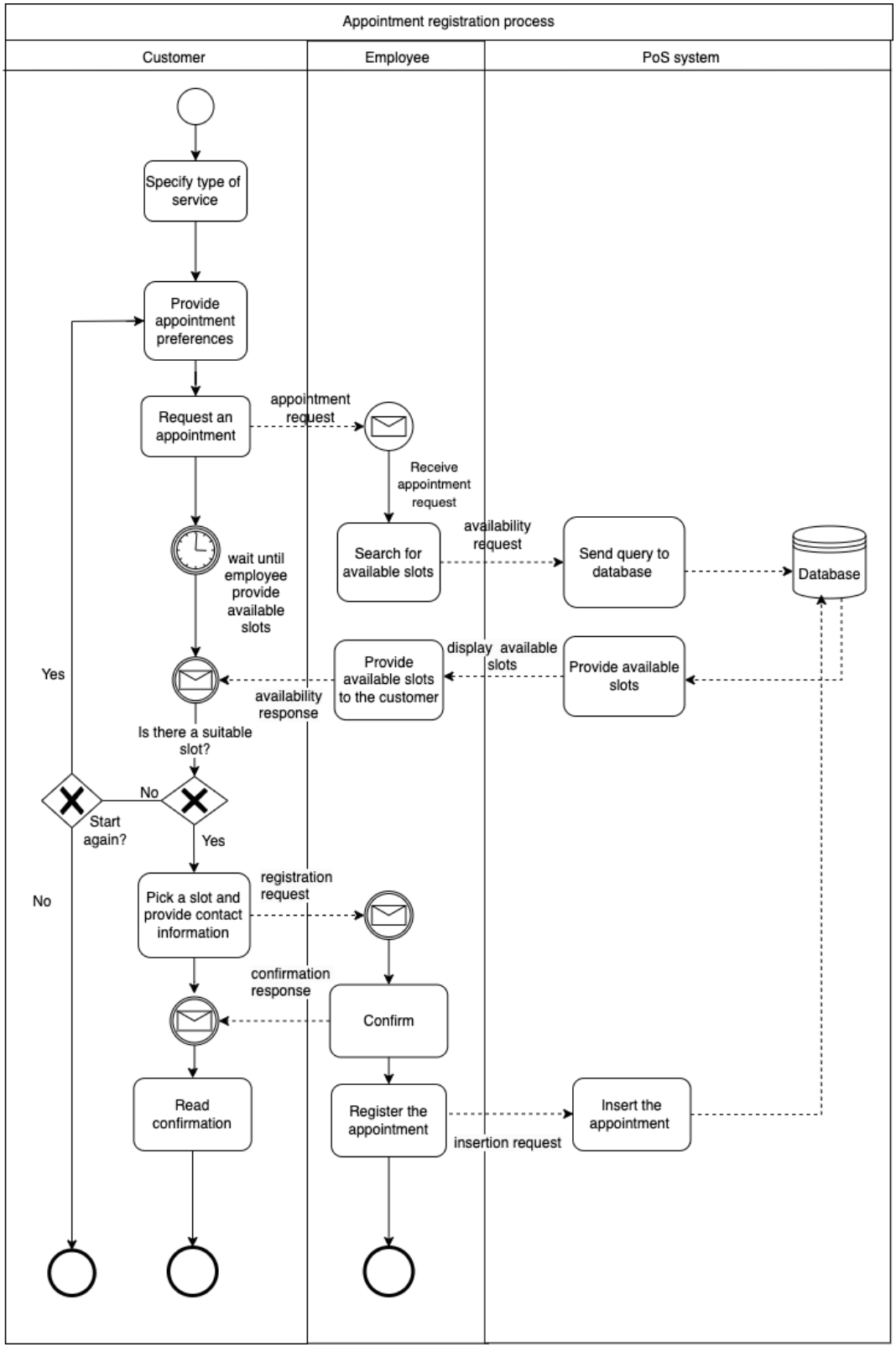

Figure 1. Appointment registration BPMN diagram

Figure 1 shows the appointment registration flow. The appointment registration process begins with the customer specifying the type of service and providing their appointment preferences, such as date, time or specific service employee. Once the customer submits the appointment request, they wait for the employee to check for available slots. The employee receives the request and sends an availability query to the Point of Sale (PoS) system, which validates the availability by checking the database to ensure there are no conflicting appointments. Afterwards, the PoS system sends the employee a list of available slots that do not have any conflicting appointments. Then those slots are provided to the customer by the employee. The customer then reviews the available slots and selects one if suitable. If no suitable slot is found, the customer can choose to provide another preferences and start the process again. Upon selecting a slot, the customer provides their contact information and sends a registration request. The employee confirms and registers the appointment, and the PoS system updates the database with the new booking. Finally, the customer receives a confirmation, completing the process.

2.2 Appointment cancellation process

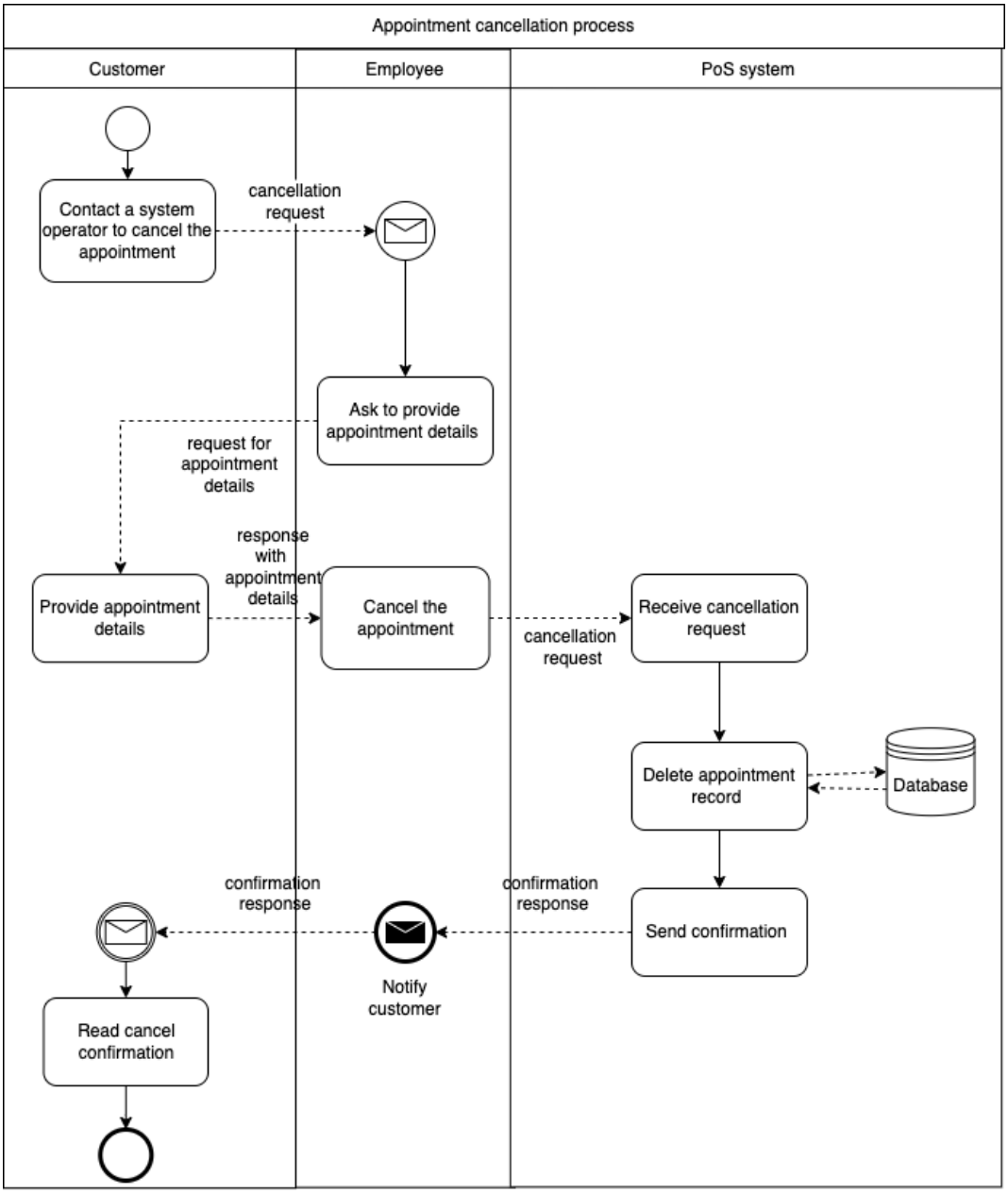

Figure 2. Appointment cancellation BPMN diagram

Figure 2 shows the appointment cancellation flow. The appointment cancellation process depicted in the flowchart involves three key parties: the customer, the employee, and the Point of Sale (PoS) system. The process begins when the customer contacts the employee to request the cancellation of an appointment. The employee, needing more information, asks the customer to provide the necessary appointment details. Once the details are received, the employee proceeds to cancel the appointment and sends a cancellation request to the PoS system. The PoS system deletes appointment record from the database and sends a confirmation back to the employee. Finally, the employee notifies the customer that the appointment has been successfully canceled, completing the process.

2.3 Create employee process

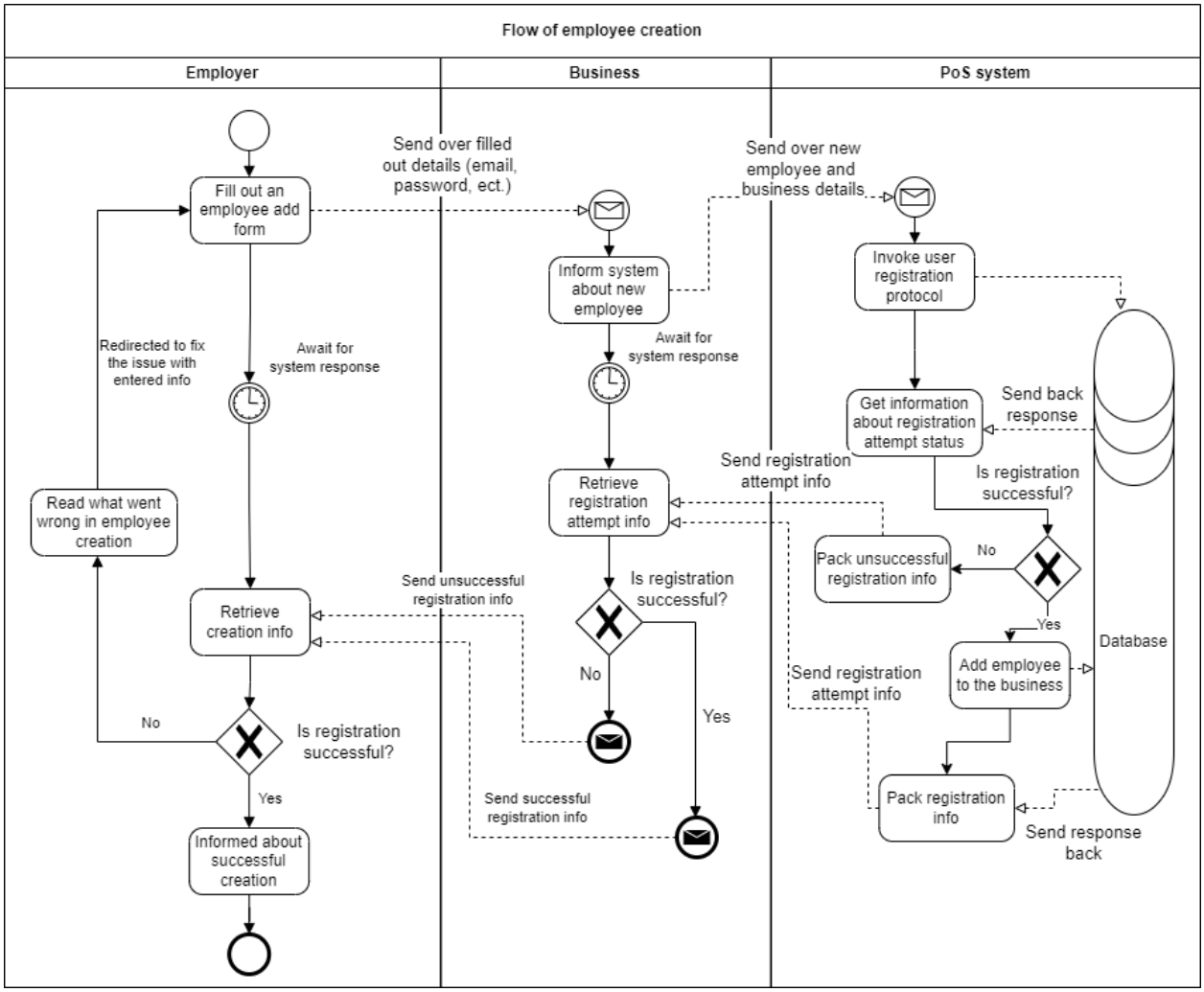

Figure 3. Create employee BPMN diagram

Figure 3 shows how employee accounts are created. A business employer can choose to create an employee account by filling out the standard registration survey with the ”Employee” tag. If the details are okay for creating the account, the system adds it to the business and informs about the successful creation of an employee account. If not, details are sent to the user why the account couldn’t be created. The employer can modify the entered detais without them dissapearing and the process restarts again on submition.

2.4 Create discount process

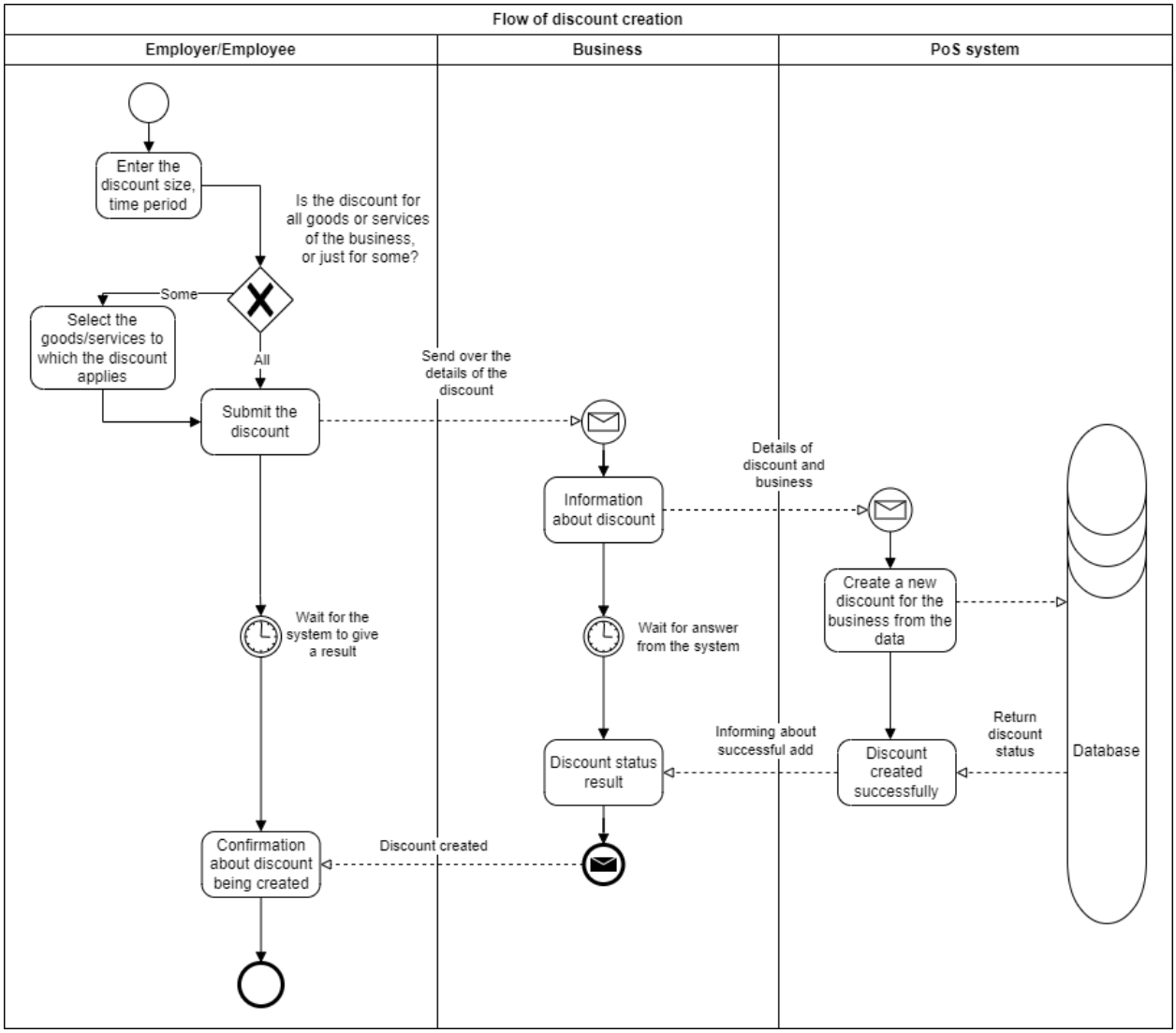

Figure 4. Create discount BPMN diagram

Figure 4 displays how a personel, working in the business, can create a discount. To start off, the user has to enter the details of the discount: the percentage, a description, ect. The employee/employer then can select if all goods/services get the discount, or if certain ones do. The discount is then checked with the system and if everything is okay, the user is informed.

2.5 Create service process

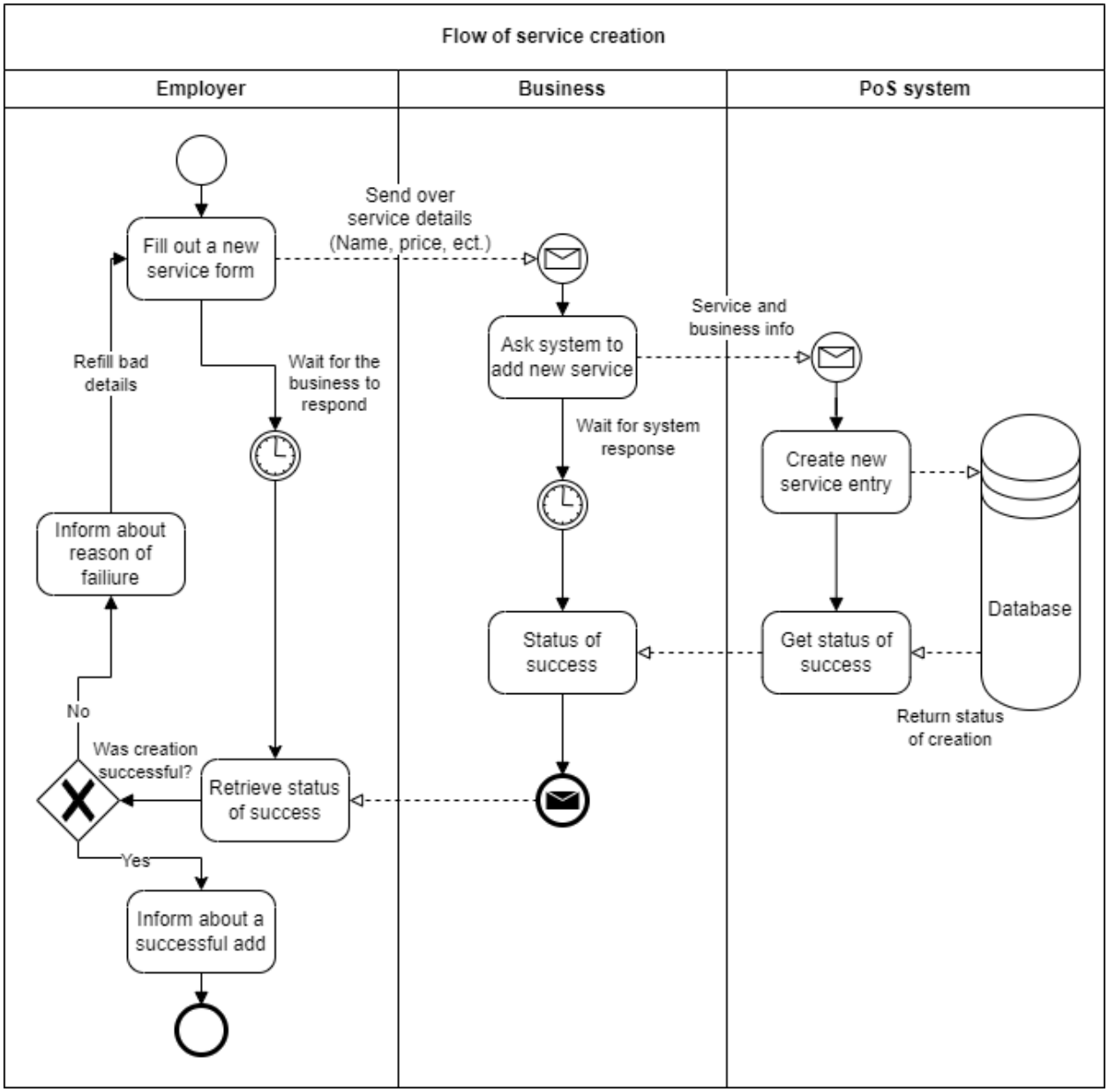

Figure 5. Create service BPMN diagram

Figure 5 shows how an employer can create a service for the business. The employer fills out details about the service, such as name, price, description, ect. This is sent to the system for saving. If it saves, the user is informed, if not - the user is informed about the error and can fix the mistake that occured. Then they can resubmit.

2.6 Create business process

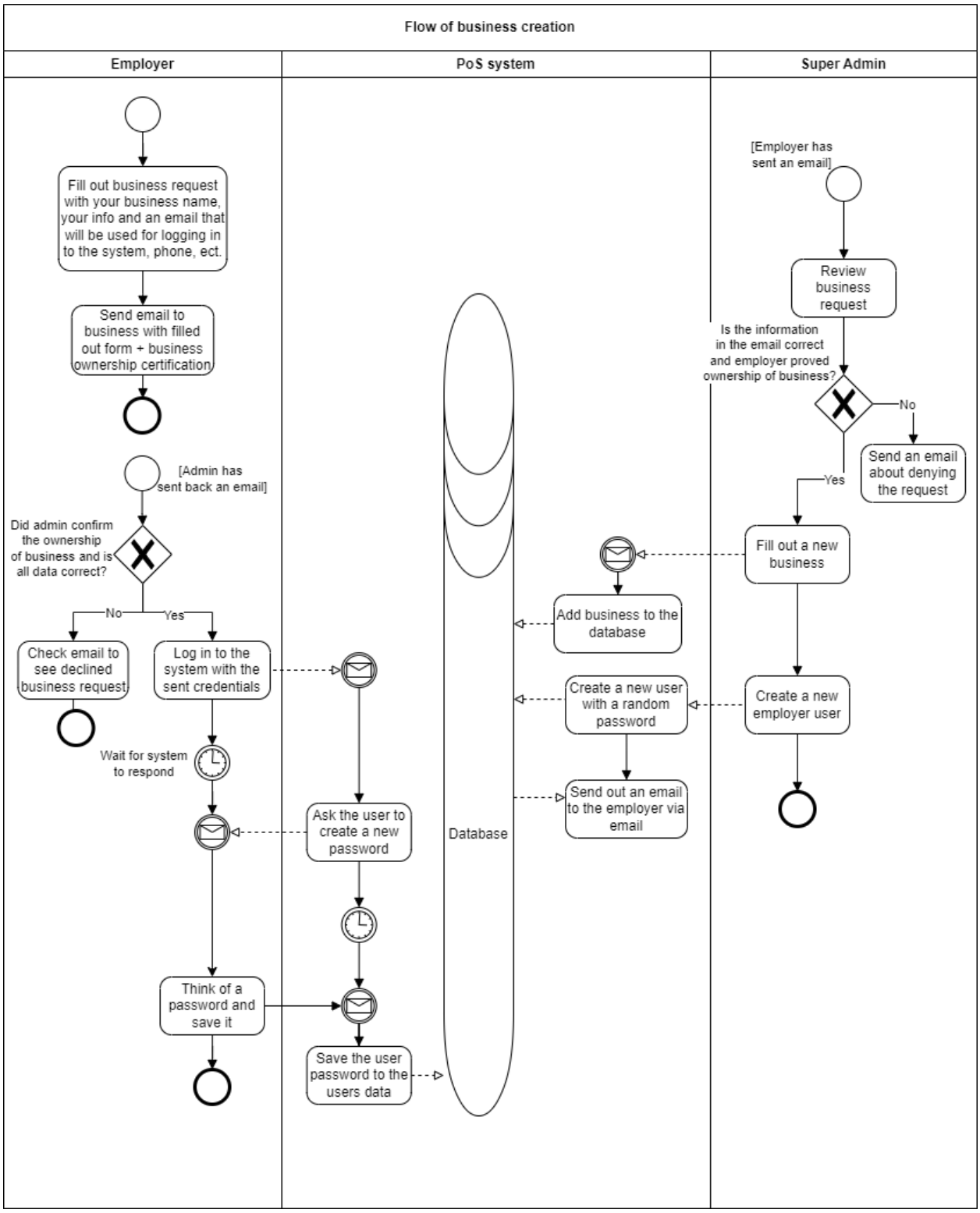

Figure 6. Create business BPMN diagram

Figure 6 depicts how the process of creating a business begins. An employer should fill out a business form to request a business registration with the details for their business, with proof of ownership, login email, phone number, first and last names. The owner sends all of this via an email to an admin for confirmation, because business are a delicate process, so moderation is

required. If the details aren't correct, the admin declines the request. If all details check out with the proof of ownership of the business, the admin creates a new business and an account for the business owner. The system then processes the account creation request, invokes a registration form with the provided details and generates a random password. The random password with the confirmation of registration is sent via email to the provided email address. When the user logs in for the first time, he should think of a new password.

2.7 Order management process

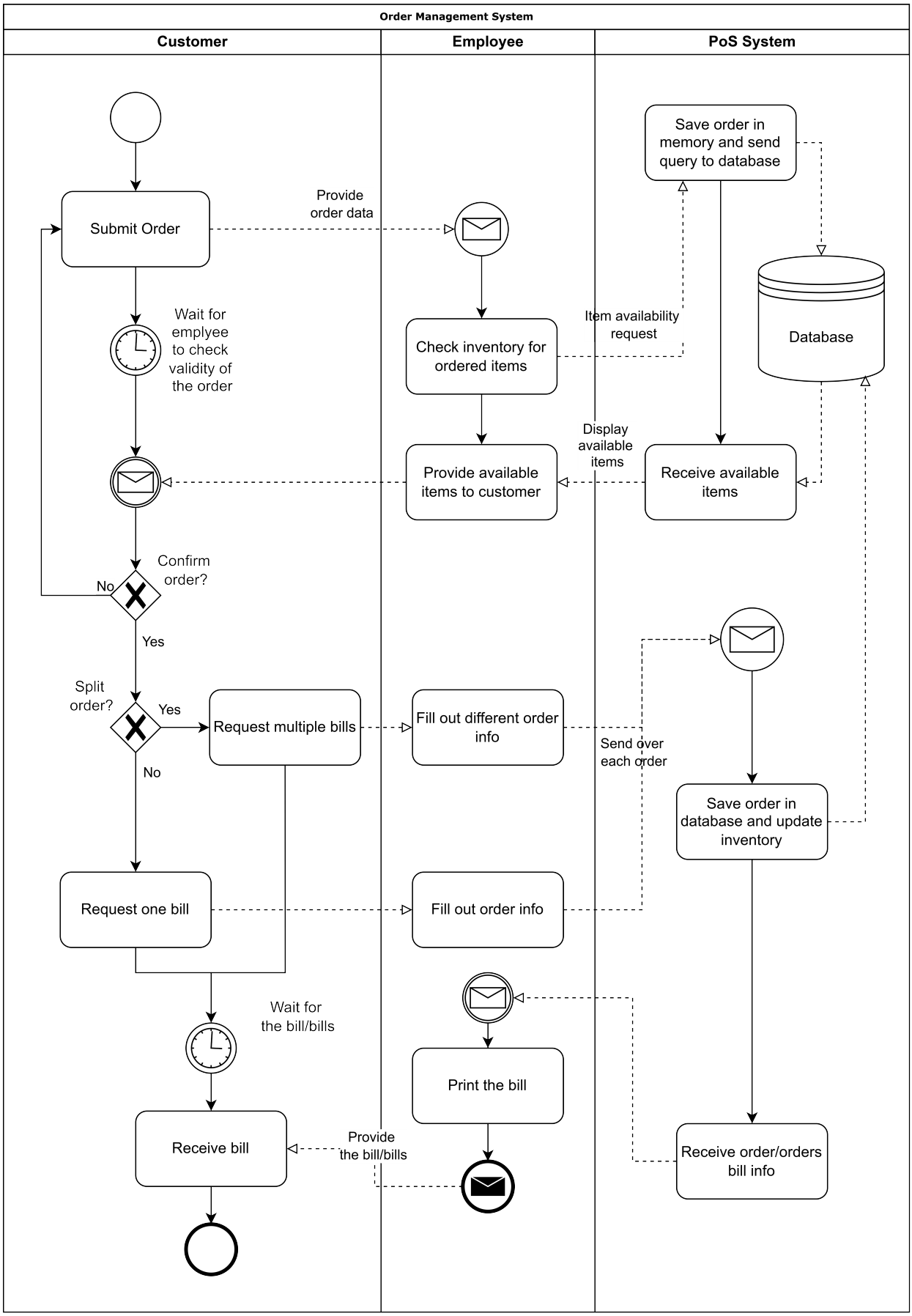

Figure 7. Order management BPMN diagram

Figure 7 shows the order management process. The customer initiates by submitting an order, which is checked by the employee for item availability. The employee consults the PoS system, which retrieves availability information from the database. If the customer confirms the order, they can request one or multiple bills. The employee fills out the required order info, prints the bill(s), and provides them to the customer. Simultaneously, the PoS system saves the order in the database, updating the inventory.

2.8 Order payment process

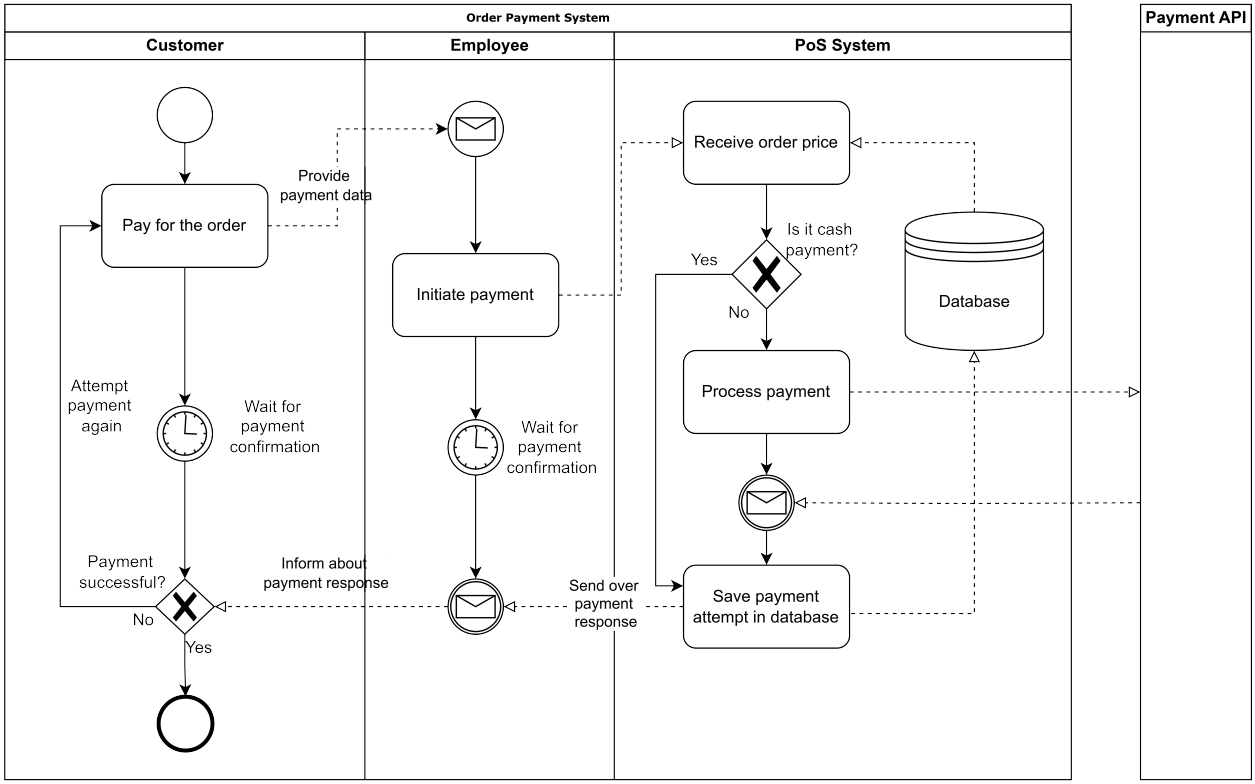

Figure 8. Order payment BPMN diagram

Figure 8 represents the order payment process. After receiving the bill, the customer proceeds to pay. The employee initiates the payment, retrieving the payment amount from the PoS system's database. If it is cash payment, it is only registered in database, as it is employees responsibility outside of the system to deal with the payment. If it is not a cash payment, the system processes the transaction. Payment confirmation is awaited, and once successful, it is saved in the database. If the payment fails, the customer is informed, and the process repeats until successful.

2.9 Order cancellation process

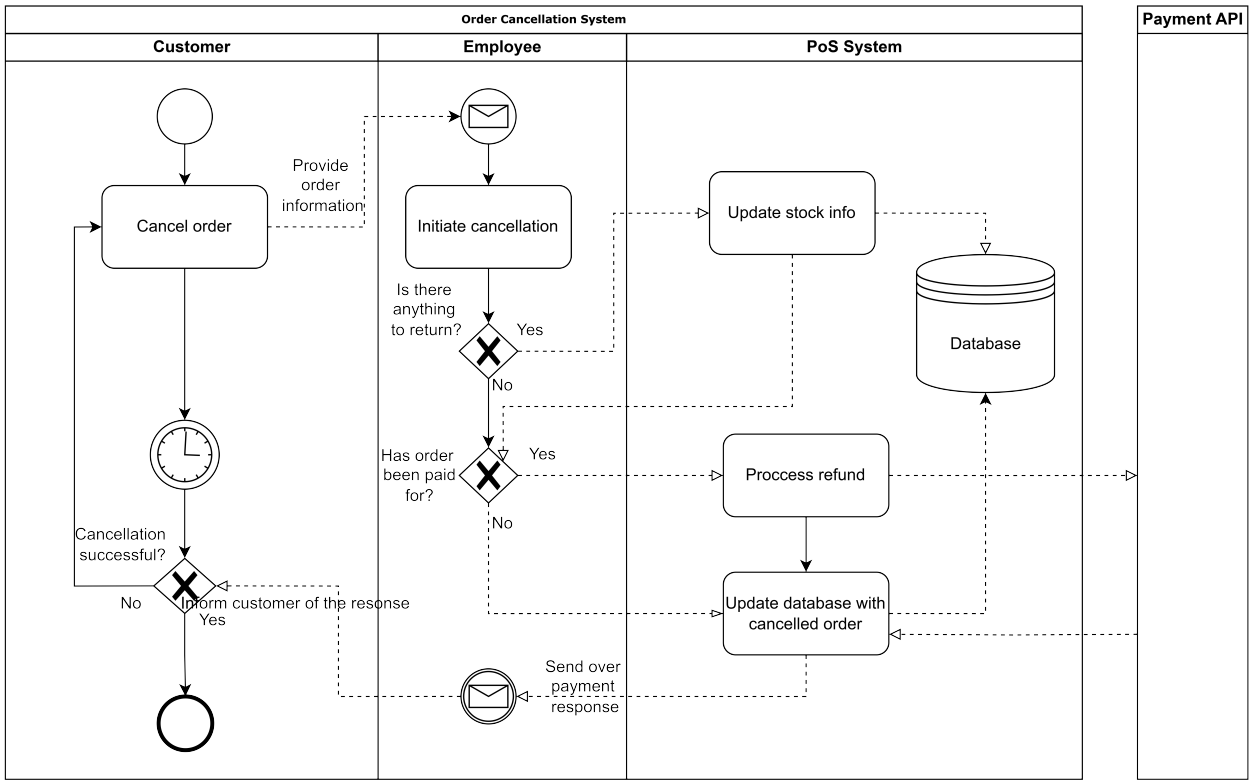

Figure 9. Order cancellation BPMN diagram

Figure 9 depicts the order cancellation flow. The customer requests an order cancellation, which is initiated by the employee. The employee determines if items need to be returned and whether payment was made. If paid, the system processes the refund and updates the stock and database. Upon completion, the customer is informed of the cancellation status, and the order is updated in the database.

2.10 Order saving process

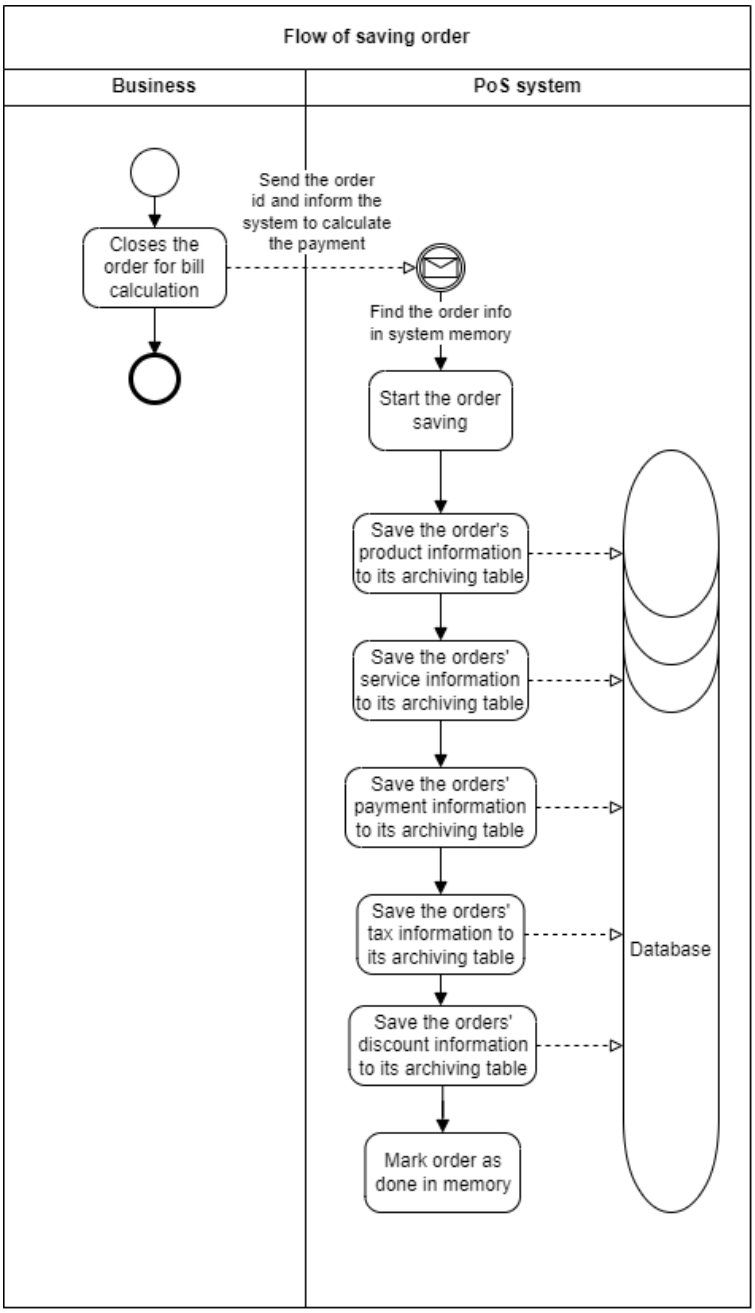

Figure 10. Calculating order BPMN diagram

Figure 10 shows how the system saves the order into the database after the payment is done. All of archives should have a reference to the order they come from. To archive the products in the database, the product and variation name are combined to one, the count of them is saved as well, and entered to the database for each of the product in the order. To archive service information, the name of the service and the price is saved for each of the service that was given in the order. To archive the payment info, the system saves how much money was paid and by what means was the payment made for each payment a person made. The tax archive holds its name and the percentage of tax that was applied at the time for the order. Discounts are also archived with the product they were applied to full name and the price that was cut off from the product's price.

3 Data model

3.1 Choices

a) A serialized primary key for each table was chosen, due to its simplicity, and due to manually creating primary keys sometimes can result in slower speeds and more errors, especially when trying to deal with CRUD operations of tables and following what ids were deleted and where.

b) VarChar() data type is used for string values, due to its ability to take up only as much memory as needed to store its value. Additionally, wherever possible Enums where used.

c) Open/Cancelled orders, will never be stored inside the database. Orders up until they are closed/paid for, should be cached inside the back end of the system, and should only be recorded in the database when they have been finalized. Also they should be cached in the front end, to ensure data persistency. I.e. if the electricity or the internet goes out for a business, or for our system servers, the open orders will not be lost.

d) Taxes and Currency are not present in the database (only inside Payment and TaxArchive tables for archiving purposes) and should be stored in property files with their values (name and percentage for tax and name for currency). This is chosen because each country has default taxes with different values such as VAT, excise duties, customs, etc. And the same is for currencies. For country differentiation can use ISO 3166-1 Alpha-3 Code (3-letter code) (e.g.: USA, JPN, etc.) The layout could look something like this:

```
country: char(3)
    currency: char(3)
    nameOfTax: string
        taxAmount: decimal(10,2)
        isPercentage: boolean
    nameOfTax: string
        taxAmount: decimal(10,2)
        isPercentage: boolean
```

e) No calculatable fields were used. Only field itemQuantity from the ProductVariation table should be constantly adjusted to correctly represent how many items are left of the specific variation and balance field from the Giftcard entity, should be changed on each payment (this can be done via triggers or inside the business logic). Therefore, all of the calculations should be done inside the business logic and shouldn't be stored anywhere.

f) For all email, phone fields a check constraint: check(email LIKE '%_@_%._%') and check(phone LIKE '+%') should be present. For decimal values if they are (10,2): check(value >= 0 and value <= 99999999.99), if they are (4,2): check(value >= 0 and value <= 99.99). Lastly if an entity has a start and end date, a check constraint for start and end date: check(startDate <= endDate)

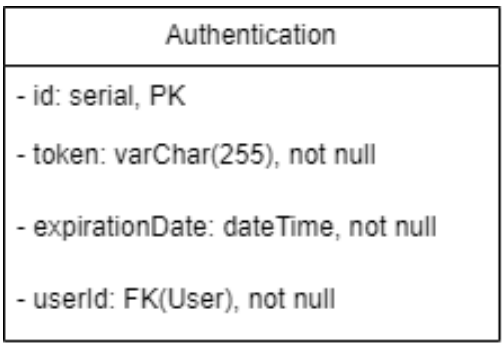

Figure 11. Authentication entity class diagram

g) Due to us not covering the authentication table in any of the components, we will disclose it here. Figure 11 shows the authentication entity, which will be responsible for holding all of the users' authentication tokens (they will be used to allow them to make api calls to the back-end of the system). One user will be able to have multiple tokens, and the entitiy will also be able to hold their expiration date.

3.2 Full database class diagram

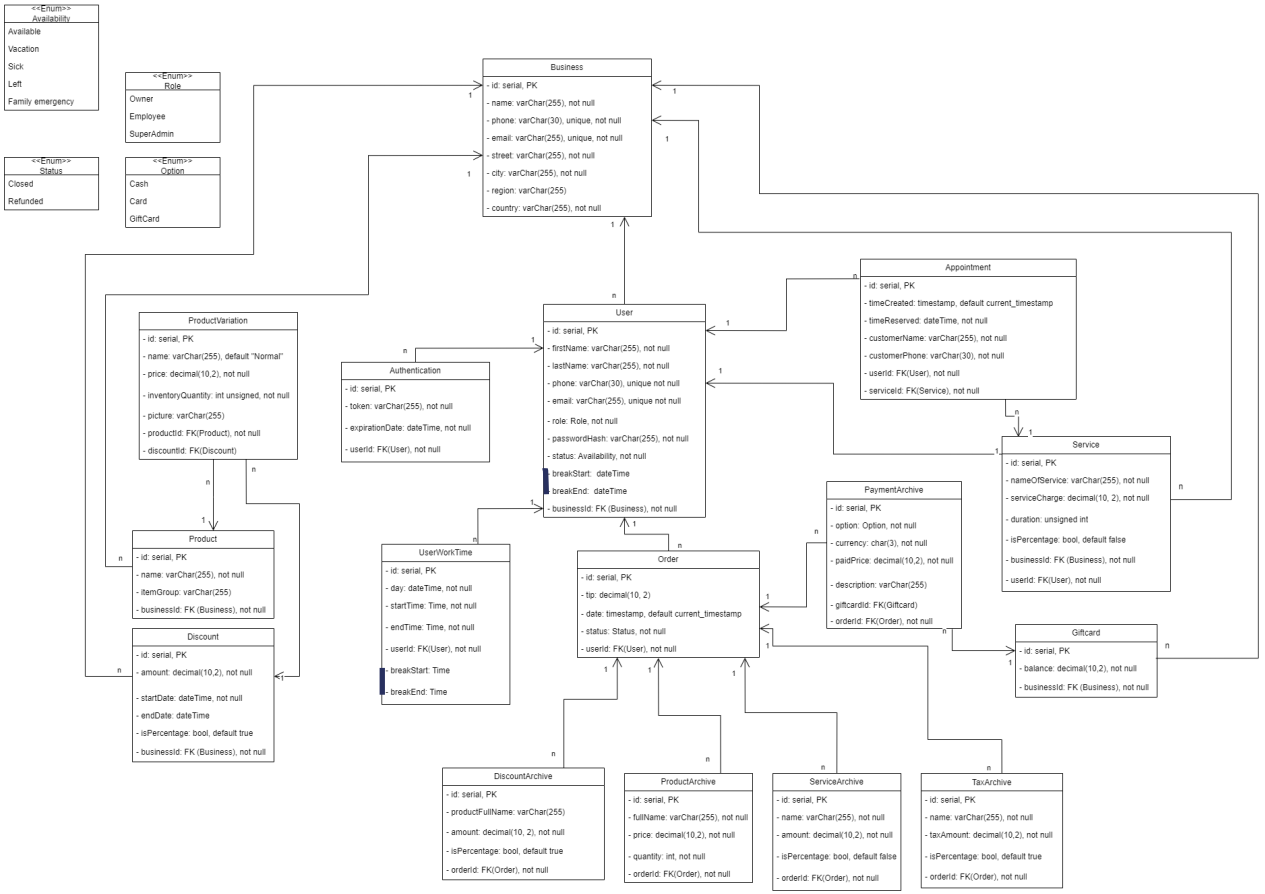

Figure 12. Data model class diagram

3.3 Orders and Payments component

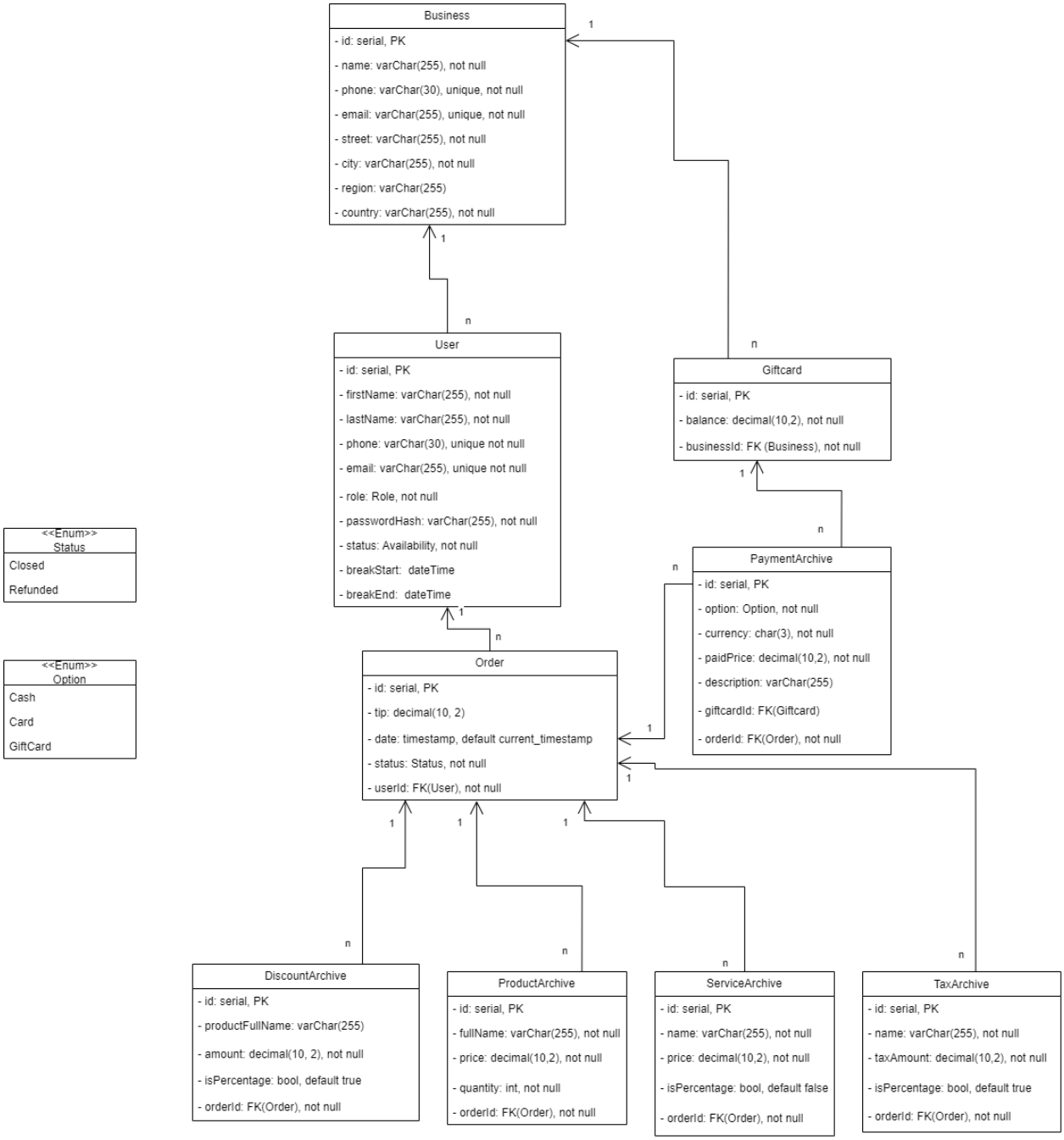

Figure 13. Data model of Orders and Payments component class diagram

3.3.1 Function of the component

Figure 13 shows the orders and payments component. This component will be responsible for creating/tracking and archiving orders and the products/discounts/services/taxes and payments which were used for that order indefinitely. As mentioned before all open/on going orders will be cahched inside the system up until they are paid for. After the order is paid for the system should insert an order into the orders table, stating what tip was left for it, when the order was finished, and which employee was responsible for the order. It should also insert all the payment information, what products and services were bought for the order, and also what discounts and

taxes where applied. Lastly, this component is responsible for storing all of the giftcards of the business and tracking each of their balances. If the order gets refunded, its status should be changed to refunded.

3.3.2 Entities of the component

Orders: responsible for stoing closed and refunded orders. Stores tips left from the customers, the employee which created and carried out the order and the date for when the order was created. Closed orders shouldn't be modified nor deleted to ensure archiving and data integrity.

Discount/Product/Service archive: these tables are used to store the snapshot of all the products/services/discounts used for when the order was created and carried out. Each of them hold relavant information about each product/service/discount that was used during the order. Entries in these tables just like the orders should not be modified or deleted after their creation, to ensure archiving of orders indefinitely. For discounts archive, if the discount was for the whole order, the productFullName field should be left null.

TaxArchive: similiar to all the archives, besides that Tax doesn't have its own entity in the database. It will be stored inside the properties file and its structure inside the file should be the same as the TaxArhive entity. Whenever the order is paid for and the check is printed the system should record all the taxes it used and store them inside the database assigning to that specific order.

PaymentArchive: again similiar to the other archives but has different fields. Currency field, should record with what currency the payment was carried out (as mentioned before currency is also stored inside the property file). Option and paidPrice fields work in tandem with one another. The option field tells with what option the order was paid for, and the paidPrice field will tell how much of the price was paid with that option. This way we will be able to allow customers to pay with a couple of different options for the same order.

Giftcard: used to store all of the business's giftcards it has created. These cards will be one of the options for paying for the order, and have a balance field, to show how much money is left inside of each of the cards. Each payment the balance should be decreased to keep data integrity, and not allow people abuse giftcards.

3.4 Products and Discounts component

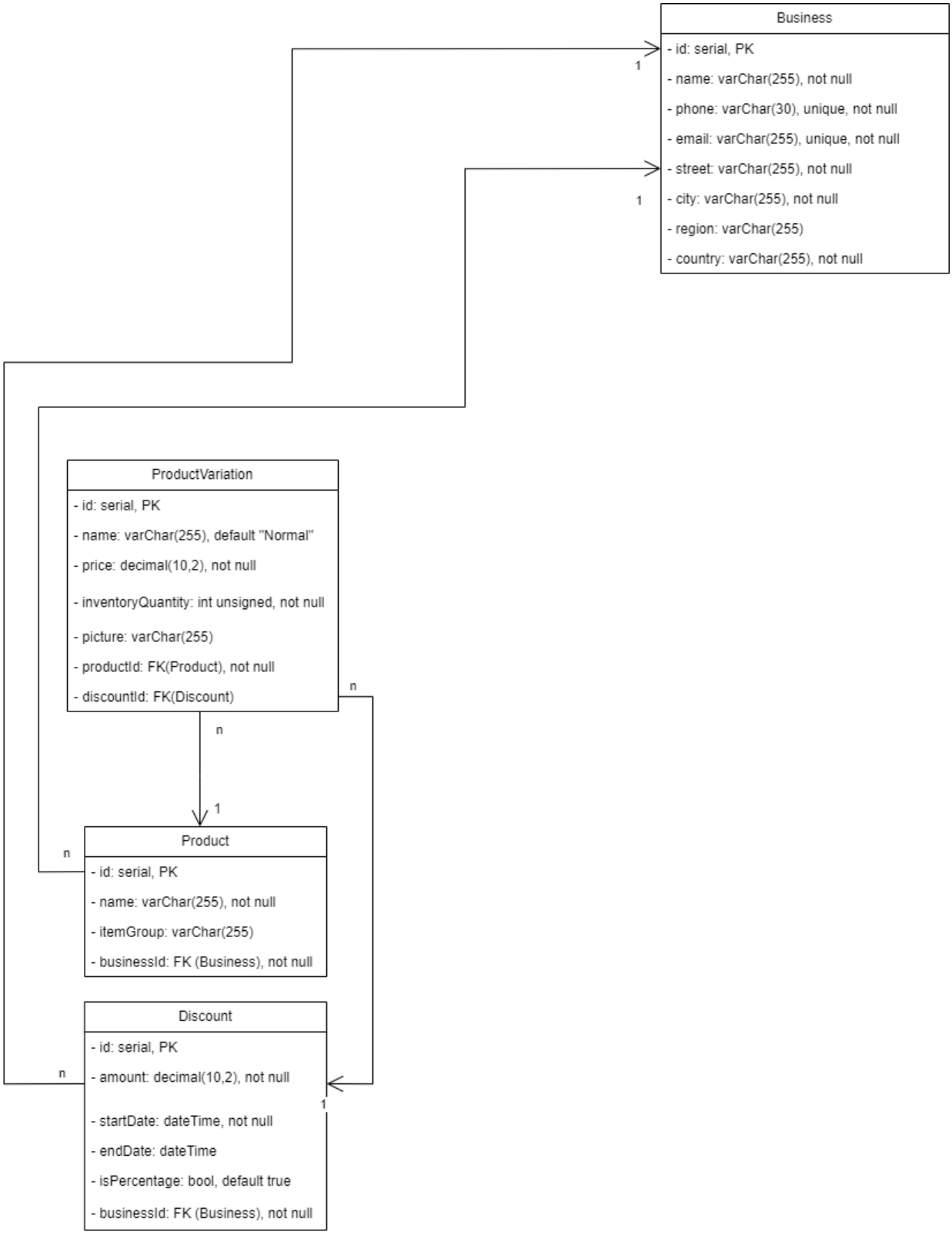

Figure 14. Data model of Products and Discounts component class diagram

3.4.1 Function of the component

Figure 14 shows the products and discounts component. This component will be responsible for storing all of the business products and discounts and ensuring each product can have many variations attributed to it simultaneously and one discount applied to each variation. With the help of the product table, which stores the name of the product and optionally its itemGroup, discounts can get applied to whole products or itemGroups and not just specific variations.

3.4.2 Entities of the component

Product/ProductVariation: these two entities will hold all of the information about products and their variations. As mentioned before a product has a name and an itemGroup (e.g.: ”cappucino” and ”coffee”). The ProductVariation table holds the actual price, picture, description and name of each variation. A default constraint is used as ”Normal”, which will be the actual product (e.g.: ”cappucino”), without any variations and its picture and price. Also inventoryQuantity is used to ensure resource management for each variation, and after an order is executed this field should be decremented, to ensure data integrity. Each time an employee will want to add a cappucino, decaf, with vegan milk, and with cinamon, he will be able to add the ”Normal” variation of the cappucino, and also the variations ”with vegan milk”, ”decaf” and ”with cinamon”, which all will have their price which should be added to the total of the order.

Discount: this table will be responsible for storing all of the discounts of the business. It has its start and end dates, when the discount is valid, and the amount, which will show how much the price needs to be decreased of a product/order. The endDate is optional, to give the possibility that the discounts could be used for a first time purchase (in this case another table should be added to the user entitiy which would hold, what discounts has the user claimed). isPercentage flag is used to let the user choose if he wants the discount to be a flat amount or a percentage. Each product variation can only have one discount attributed to it. Therefore if a discount is applied to the ProductVariation, it should override the last discountId, if it had one.

3.5 Users, Services and Appointments component

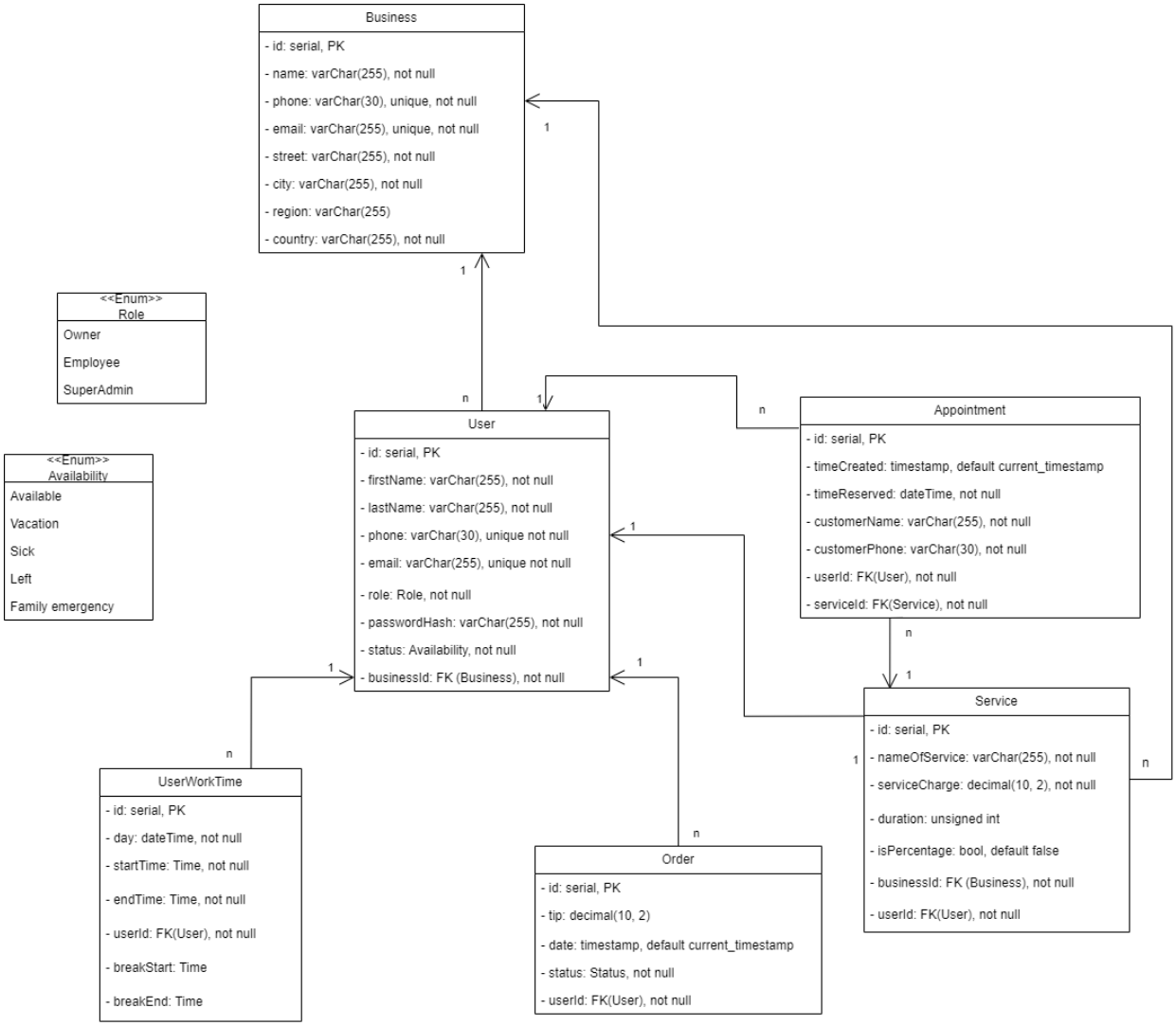

Figure 15. Data model of Users, Services and Appointments component class diagram

3.5.1 Function of the component

Figure 15 shows users, services and appointments component of the system. We chose to include users, services, and appointments into one component becasue they are all tied with one another through availability for services, when rezerving times for customers. This component will be responsible for tracking each employee of the system, his authentication tokens and availability. Additionally, it will hold all of the services the business provides and also its appointments/rezervations. Whenever a rezervation is needed for a specific service, the system will be able to check which users can perform the service, are they working right now, if not when will they be available. By this, in tandem with checking which times are already rezerved, the system will be able to give the earliest time when the rezervation is allowed to be created.

3.5.2 Entities of the component

User: is responsible for storing all of the businesses' employees/owners, and also the system's itself SuperAdmins. It will store their names, contact information, a hashed passwords. Additionally, for employees of a business, it will hold their status, which will be an enum, providing information if the employee is available, and if not for what reason.

UserWorkTime: this table is used to store when and what employee is working and for how long. Each day will have as many entries inside the table as there are employees, and for each employee for each day there will be a starting and ending time. This will help account for shifting work schedules. As an example if we had two employees: John and Tom, each of them would get an entry for day: 2024/10/15, and both of their starting and ending work times. Lastly, if the employee becomes unavailable, due to vacation, sickness, etc., then his WorkTimes should be removed from the table on the days he will be absent. This is all done again to help the system calculate when is the earliest rezervation time possible, and to know when the employee will be coming back to work. Also each employees lunch break's start and end time will be stored, to be more accurate when showing which times are available for rezervations.

Service: will hold all of the services the business provides (e.g.: manicure, pedicure, haircut, etc.). Each of the service, will have an employee associated with it which can perform the service. However, this can also be used solely as a service charge for a restaurant for example and it can be applied to just all orders of the restaurant, therefore, an isPercentage flag has been added to allow the serviceCharge be a flat price or a percentage. Lastly it has a duration (which is optional), to account for how long a service usually takes place, to help calculate when is the earliest possible rezervation time, an unsigned integer is used to now allow negative values (i.e. minutes).

Appointments: will hold the customer's contact information, when was the rezervation made, the employee which will be executing the service, and what service will be done.

4 Architecture

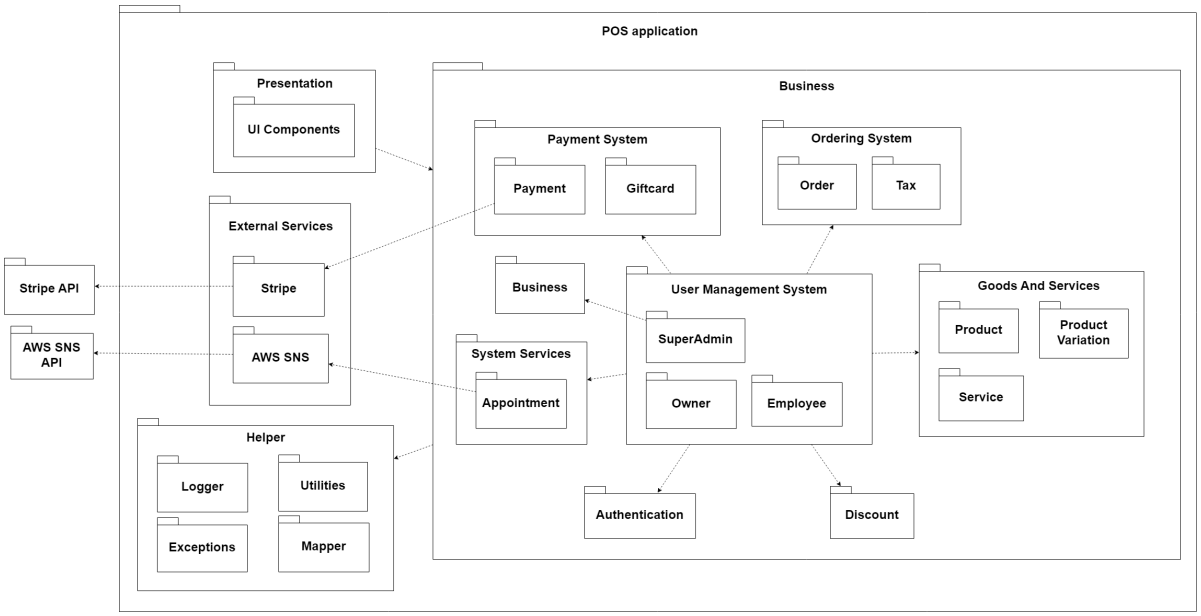

Figure 16. Component-based architecture of the POS system package diagram

The application will be composed of the following systems:

4.1 Presentation Layer

The **Presentation Layer** handles the user interface (UI) of the application. It includes all the visual elements and interactions that users engage with. This component is responsible for receiving user input and displaying data from the business logic.

4.2 Business Layer

The **Business Layer** contains the main logic of the system. It is divided into several systems and components, each responsible for a specific opperation of the application:

- **Ordering System**: handles everything related to customer orders, including order creation, updates, and applying taxes.
- **Payment System**: Manages all payment-related activities, such as processing payments and managing gift cards. It connects with external services such as **Stripe** to handle secure transactions.
- **User Management System**: Responsible for managing different types of users: **Super-Admins**, **Owners**, and **Employees**. It controls user roles, permissions, and access to various parts of the system.
- **Goods and Services**: Manages the products, product variations, and services offered by the business. It ensures that everything being sold or offered is tracked and organized.
- **System Services**: Deals with services offered by the application, in this case - appointment service.

4.3 User Management System

This component ensures that users are authenticated, have appropriate access levels, and can interact with the business systems based on their role. It also manages user data, permissions, and access to specific features of the application. The roles include:

- **SuperAdmin**: Has the highest level of control over the system and is able to manage businesses enrolled with the system.
- **Owner**: Manages business operations and employees.
- **Employee**: Accesses the main functionality offered by the application, such as managing orders.

4.4 Payment System

The **Payment System** is responsible for handling all transactions whithin the application. It processes payments through external services like **Stipe** to ensure secure and reliable payment handling. It also manages **Gift Cards**, allowing customers to purchase and use gift cards in exchange for goods and services withing the application. This component communicates with the **Ordering System** to ensure that payments are processed once an order is confirmed.

4.5 Goods and Services Component

The **Goods and Services Component** manages the products and services offered by the business. It handles the creation, updating, and archiving of products and services, ensuring that the information remains consistent across the system. This component interacts mainly with the **User Management System**, ensuring that authorized users (such as employees or owners) can manage and access the product and service data.

- **Product Component**: Responsible for managing the catalog of products, including adding, updating, and archiving items. Authorized users can access and modify product information as needed.
- **Product Variation Component**: Manages different variations of products and ensures that the system keeps track of these variations independently, which allows employees to keep stock levels accurate.
- **Service Component**: Organizes and schedules the services offered by the business. It works closely with the **User Management System** to ensure that employees can book, manage, and track service appointments in real-time.

4.6 External Services Component

This component manages interactions with external systems and APIs. It ensures smooth communication with third-party services that are integral to the system's operation:

- **Stripe**: Handles secure payment processing by connecting with the Stripe API to process customer transactions.

- **AWS SNS**: Integrates with Amazon Web Services (AWS) Simple Notification Service (SNS) to send notifications (such as appointment reminders) via SMS or email.

4.7 Helper Component

The **Helper Component** contains utility functions and common services that are used across various parts of the system. It includes:

- **Logger**: Keeps a record of system events, such as errors, warnings, and important actions, which is useful for debugging.
- **Utilities**: Common functions or operations that can be reused across different components of the application.
- **Exceptions**: Manages system errors and exceptions, ensuring that errors are handled without crashing the application.
- **Mapper**: Handles the conversion or transformation of data between different parts of the system, making sure that components can communicate effectively with each other.

4.8 Authentication Component

The **Authentication Component** ensures that only authorized users can access the system. It handles user login, session management, and secure access to different parts of the application. This component interacts closely with the **User Management System**, validating user credentials and ensuring that users have the appropriate roles and permissions before granting access to application's features or data.

4.9 Ordering System

The **Ordering System** manages all aspects of processing customer orders. It is designed to interact primarily with the **User Management System** to ensure that order processing is handled by authorized users only. This system includes the following components:

- **Order Component**: This component handles the lifecycle of orders within the system, from creation to finalization. It allows authorized users to create orders, update order details, and finalize transactions. By interacting with the **User Management System**, it ensures that only users with the appropriate permissions can perform these actions.
- **Tax Component**: This component is responsible for calculating the correct taxes for each order. It applies tax rates based on the products being purchased and to an entire order if eligible. The **User Management System** ensures that tax calculations and modificaitons can only be performed by users with the right level of access.

4.10 Discount Component

The **Discount Component** manages discounts that can be applied to both products and orders. Employees from the **User Management System** can create, update, and manage discounts as necessary.