Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

# АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА С УЧЕБНО-ИССЛЕДОВАТЕЛЬСКОЙ БАЗОЙ ДАННЫХ ДЛЯ УЧЁТА ПОСТАВОК И ПРОДАЖ В МАГАЗИНЕ

Курсовая работа
по дисциплине «Безопасность систем баз данных»

Пояснительная записка

Студент гр. 720-1

_____ С.А. Федосеев
(подпись)

_____
(дата)

Руководитель

Старший преподаватель кафедры КИБЭВС

_____ _____ Н.А. Новгородова
(оценка) (подпись)

_____
(дата)

Томск 2023

# Реферат

Курсовая работа содержит 38 страниц, 18 рисунков, 8 таблиц, 2 источника, 2 приложения.

БАЗА ДАННЫХ, REACT, TYPESCRIPT, SQL, POSTGRESQL, IDEF0, IDEF1X, NODEJS.

Объектом разработки является автоматизированная система для ведения учёта поставок и продаж.

Цель работы – разработать автоматизированную систему с базой данных для учёта поставок и продаж.

Результаты работы:

– определены необходимые технологии и проработана архитектура БД;

– разработано приложение для работы с БД.

В качестве инструментария для выполнения данной работы были использованы: язык программирования TypeScript, PostgreSQL, Visual Studio Code, фреймворк React.

Пояснительная записка к курсовой работе выполнена в текстовом редакторе Microsoft Word 2007 и была оформлена согласно ОС ТУСУР 01-2021. [1]

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ
Заведующий кафедрой КИБЭВС,
д-р техн. наук, профессор
_____ А.А. Шелупанов
“___”_____ 2022г.

**ЗАДАНИЕ**

на курсовую работу по дисциплине «Безопасность систем баз данных» студенту Федосееву Сергею Александровичу группы 720-1 факультета безопасности.

1 Тема работы: Автоматизированная информационная система с учебно-исследовательской базой данных: Учёт поставок и продаж в магазине;

2 Исходные данные к работе:

2.1 Реляционная СУБД PostgreSQL;

2.2 Данные по предметной области

Персонал, товар, поставки, продажи

3 Срок сдачи студентом законченной работы: _____

4 Содержание курсовой работы:

4.1 Проектирование инфологической модели данных:

– описание и структуризация предметной области (описание бизнес-процессов, диаграммы IDEF0);

– представление модели «Сущность-связь» (ER-модель);

– сценарий пользовательского интерфейса.

4.2 Проектирование даталогической (логической) модели данных:

– проектирование реляционной базы данных на основе принципов нормализации;

– проектирование концептуальной модели данных (использование методологии IDEF1X);

– составление глоссария модели.

4.3 Физическое проектирование БД:

– создание базы данных и ее необходимых элементов;

– описание ограничений на базу данных;

– сопоставление логических и физических имен.

4.4 Написание программы обработки и работы с данными:

– генерация программы меню, реализующей пользовательский интерфейс;

– режим просмотра данных с использованием экранных форм;

– использование режимов редактирования данных;

– процедуры поиска и манипулирования данными (сортировки, фильтры и пр.);

– использование SQL операторов (SQL запросы, операторы определения данных, операторы манипулирования данными);

– обеспечение безопасности данных.

5 Содержание пояснительной записки:

– титульный лист;

– реферат на русском языке;

– задание;

– содержание;

– введение;

– вопросы проектирования БД;

– обоснование выбора программных средств;

– руководство пользователя;

– описание прикладной программы;

– заключение;

– список использованных источников;

– приложения (экранные формы, листинг программы и др.).

Пояснительная записка должна быть оформлена в соответствии со стандартом ТУСУР.

В конверте на обложке приложить диск с БД, исходными текстами программы с соответствующими файлами, исполнительными файлами, пояснительной запиской, презентацией.

6 Дата выдачи задания:

« 14 » сентября 2022 г.

Задание согласовано:

Руководитель работы

Новгородова Н.А., старший преподаватель кафедры КИБЭВС

“ 14 ” сентября 2022 г. _____

Задание принято к исполнению

“ 14 ” сентября 2022 г. _____ _____

# Оглавление

# Введение

Цель работы – автоматизация процесса учёта продаж и поставок в магазине, путём проектирования и разработки приложения и базы данных. Также целью работы являются углубление знаний в области проектирования реляционных баз данных и получение практических навыков их разработки.

Объектом автоматизации является учёт поставок и продаж, а также обработка, хранение и накопление данных о них.

Автоматизация осуществляется благодаря взаимодействию пользователей с графическим интерфейсом и работой с БД.

# 1 КОНЦЕПТУАЛЬНОЕ (ИНФОЛОГИЧЕСКОЕ) ПРОЕКТИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ (ПО)

## 1.1 Неформальное описание ПО с использованием естественного языка

Продавец отвечает за продажи.

Менеджер отвечает за приём поставок.

Директор отвечает за инвентаризацию.

Данные обязанности можно назвать бизнес-процессами.

## 1.2 Описание бизнес-процессов ПО «на текущий момент» в методологии функционального моделирования IDEF0

В качестве участников бизнес-процессов могут быть:

- продавец;
- менеджер;
- директор.

Для исследования бизнес-процесса был выбран процесс учёта продаж и поставок.

Графически представленная функциональная модель ПО «Учёт продаж в и поставок в магазине» в методологии IDEF0 до внедрения автоматизированной системы описана на рисунках 1.1 и 1.2.

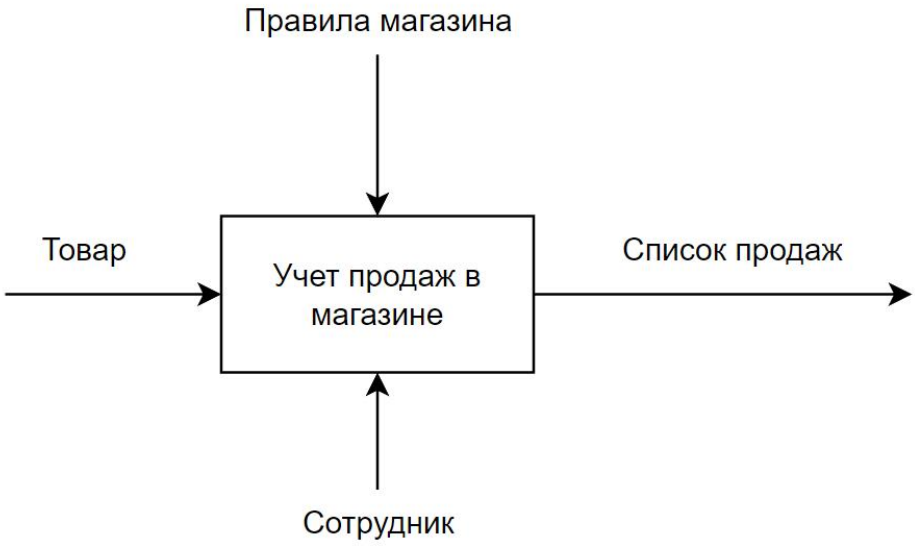



Рисунок 1.1 – Функциональная модель ПО «Учёт продаж и поставок в магазине»

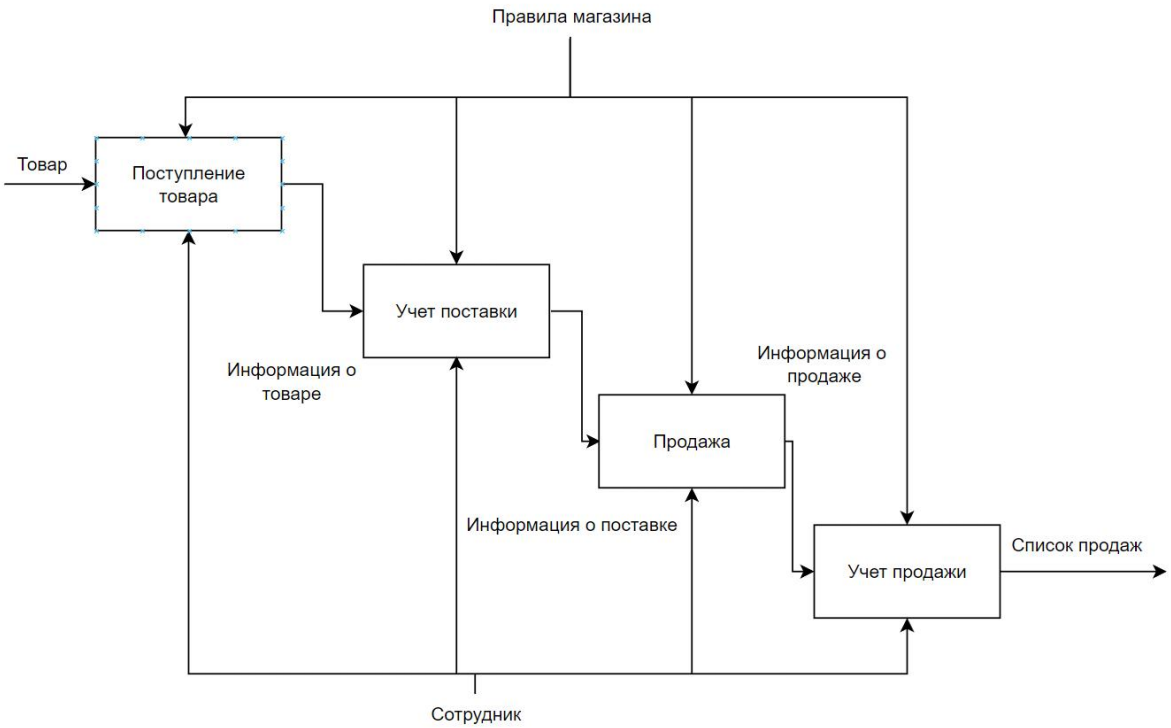



Рисунок 1.2 – Декомпозированная функциональная модель ПО «Учёт продаж и поставок в магазине»

## 1.3 Цель и актуальность внедрения автоматизации ПО

### 1.3.1 Выделение бизнес-процессов, подлежащих автоматизации

В качестве элементов, требующих автоматизации, были выбраны следующие: «Учёт поставок», «Учёт продаж».

Это необходимо для ускорения работы персонала, устранения ошибок при учёте поставок и продаж, а также наглядности документации в электронном виде.

В соответствии с предметной областью система строится с учётом следующих особенностей:

– за каждую поставку и продажу отвечает только один сотрудник;

– в каждой отдельной поставке и продаже идёт учёт только одного товара;

– у каждого товара должен быть уникальный идентификационный номер;

– у каждого сотрудника должен быть уникальный идентификационный номер;

– у каждой продажи должен быть уникальный идентификационный номер;

– у каждой поставкидолжен быть уникальный идентификационный номер;

– количество товара не может быть меньше 0;

– нельзя удалить строки из таблицы «Продажи»;

– нельзя удалить строки из таблицы «Поставки»;

### 1.3.2 Описание бизнес-процессов ПО в IDEF0 после внедрения автоматизированной информационной системы

Графически представленная функциональная модель ПО «Ведение учета поставок и продаж» в методологии IDEF0 после внедрения автоматизированной системы описана на рисунках 1.3 – 1.4.

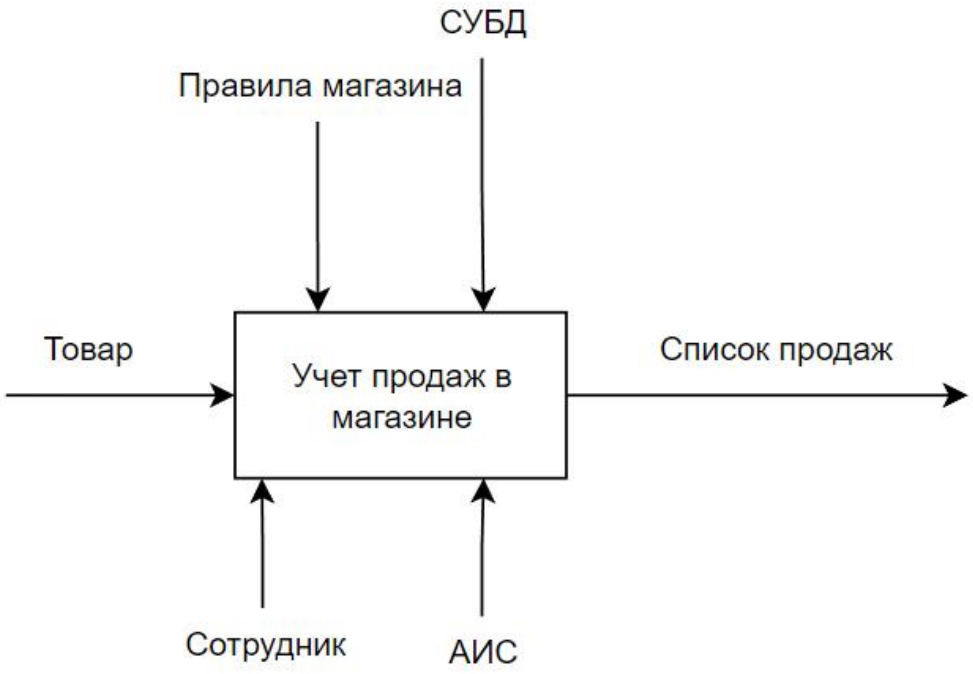



Рисунок 1.3 – Функциональная модель ПО «Учёт покупок и продаж в магазине» с АИС

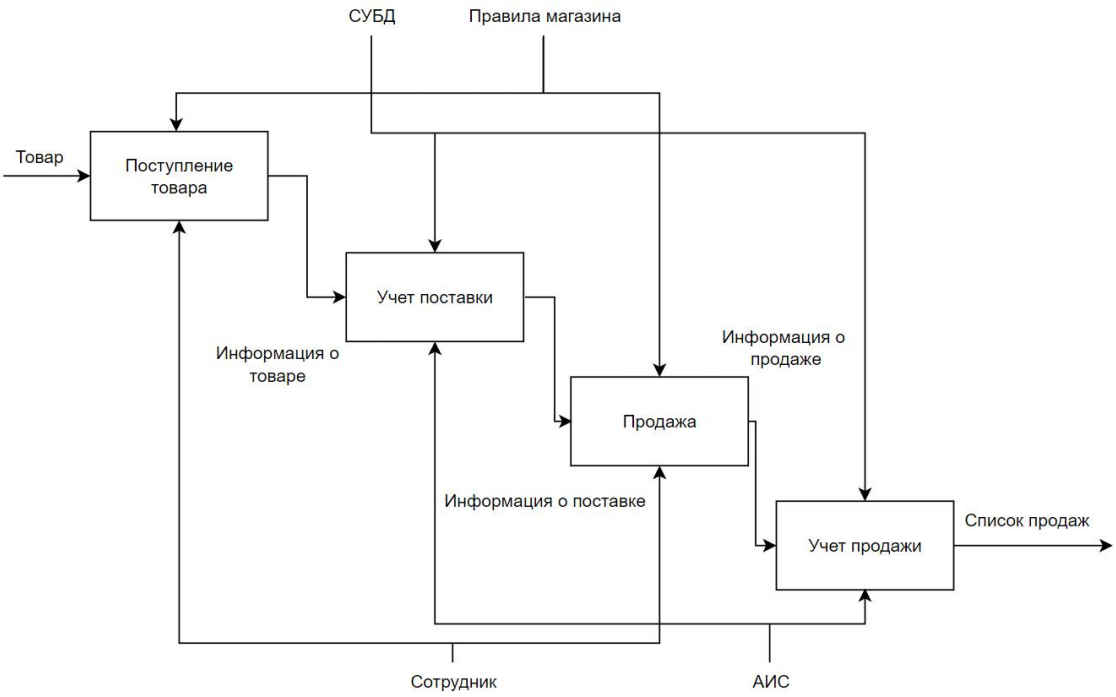



Рисунок 1.4 – Декомпозированная функциональная модель ПО «Учёт поставок и продаж» с АИС

## 1.4 Концептуальная информационная модель данных для ПО

### 1.4.1 Основные объекты ПО, информация о которых будет накапливаться в БД. Их характеристики и свойства (атрибуты)

В создаваемой базе данных основными объектами, информация о которых будет накапливаться в базе данных, будут являться сущности: «Сотрудник», «Товар», «Поставка», «Продажа».

Была рассмотрена каждая сущность отдельно, и выделены атрибуты каждой сущности.

Сущность «Сотрудник». Её атрибутами будут являться: id работника, пароль, ФИО, должность, пол, адрес.

Сущность «Товар». Её атрибутами будут являться: id товара, наименование товара, цена, описание, производитель, количество.

Сущность «Продажа». Её атрибутами будут являться: id продажи, id товара, id сотрудника, дата продажи, количество.

Сущность «Поставка». Её атрибутами будут являться: id поставки, id товара, id сотрудника, дата поставки, количество.

### 1.4.2 Определение связей между объектами ПО

Установим связи между сущностями.

Связь между сущностями «Сотрудник» и «Поставка» будет 1:M.

Связь между сущностями «Сотрудник» и «Продажа» будет 1:M.

Связь между сущностями «Товар» и «Поставка» будет 1:M.

Связь между сущностями «Товар» и «Продажа» будет 1:M.

### 1.4.3 Графическое представление концептуальной информационной модели данных

Графически представленная концептуальная информационная модель данных изображена на рисунке 1.5.

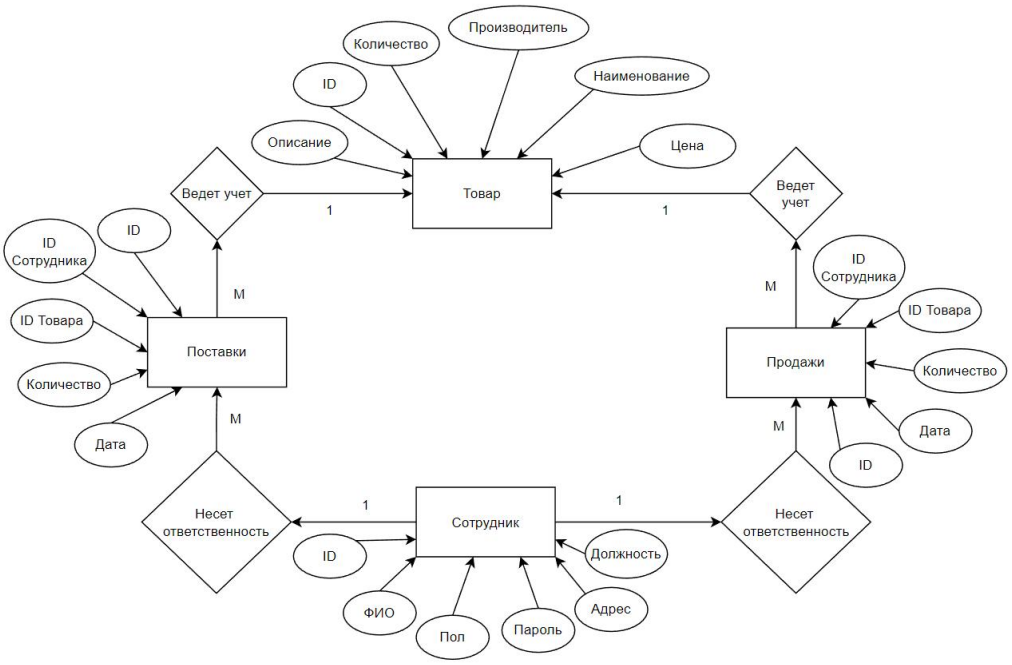



Рисунок 1.5 – Концептуальная информационная модель данных для ПО

## 1.5 Политика безопасности по работе с данными

### 1.5.1 Типы пользователей

Будет предусмотрено три типа пользователей: «Директор», «Менеджер», «Продавец».

Пользователи разных ролей будут ограничены в возможностях работы с таблицами из базы данных в целях сохранения безопасности информации.

Продавец:

– добавление новых строк в таблицу «Продажи»;

– просмотр таблицы «Товары»;

– ограниченный просмотр таблицы «Сотрудники» (Видны только продавцы).

Менеджер:

– добавление новых строк в таблицу «Поставки»;

– просмотр таблицы «Товары»;

– ограниченный просмотр таблицы «Сотрудники» (Видны только менеджеры).

Директор:

– просмотр информации в таблицах «Поставки» и «Продажи».

### 1.5.2 Ограничения пользователей по работе с объектами ПО

Пользователи разных ролей ограничены в возможностях работы с таблицами из базы данных в целях сохранения безопасности информации. Далее были приведены точные ограничения пользователей по работе с базой данных. Для удобства восприятия данные представлены в виде таблиц 1.1 – 1.3.

Таблица 1.1 – Разделение прав доступа для роли «Продавец»

| Права | «Товары» | «Сотрудники» | «Продажи» | «Поставки» |
|---|---|---|---|---|
| Просмотр | + | О продавцах | - | - |
| Обновление | - | - | - | - |
| Добавление | - | - | + | - |
| Удаление | - | - | - | - |

Таблица 1.2 – Разделение прав доступа для роли «Менеджер»

| Права | «Товары» | «Сотрудники» | «Продажи» | «Поставки» |
|---|---|---|---|---|
| Просмотр | + | О менеджерах | - | - |
| Обновление | - | - | - | - |
| Добавление | - | - | - | + |
| Удаление | - | - | - | - |

Таблица 1.3 – Разделение прав доступа для роли «Директор»

| Права | «Товары» | «Сотрудники» | «Продажи» | «Поставки» |
|---|---|---|---|---|
| Просмотр | - | - | + | + |
| Обновление | - | - | - | - |
| Добавление | - | - | - | - |
| Удаление | - | - | - | - |

# 2 ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ МОДЕЛИ БАЗЫ ДАННЫХ

## 2.1 Логическое (даталогическое) проектирование модели данных

### 2.1.1 Определение отношений и связей между отношениями на основе концептуальной информационной модели. Первичные и внешние ключи

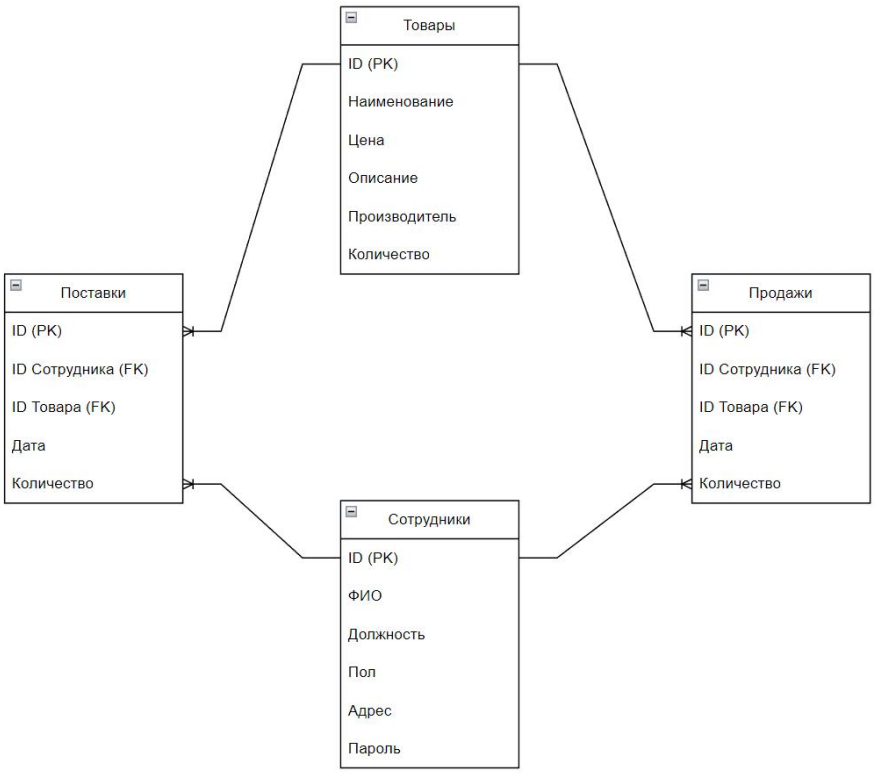
Для каждой таблицы был выделен первичный ключ. Для таблицы «Сотрудники» первичным ключом будет являться поле «id_сотрудника». Для таблицы «Товары» первичным ключом будет являться поле «id_товара». Для таблицы «Продажи» первичным ключом будет являться поле «id_продажи». Для таблицы «Поставки» первичным ключом будут являться поля «id_поставки».

Внешний ключ связывает таблицы, поэтому были выделены все внешние ключи. В таблице «Продажи» будет два внешних ключа «id_сотрудника» (связь с таблицей «Сотрудники»), «id_товара» (связь с таблицей «Товары»). В таблице «Поставки» будет два внешних ключа «id_сотрудника» (связь с таблицей «Сотрудники»), «id_товара» (связь с таблицей «Товары»).

### 2.1.2 Нормализация логической модели данных

Нормальная форма – свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Были определены отношения между сущностями, а также атрибуты, первичные и внешние ключи в отношениях. Полученные таблицы представлены на рисунке 2.1.




м

Рисунок 2.1 – Таблицы сущностей

Отношение находится в первой нормальной форме (сокращённо 1НФ), если все записи уникальны и атрибуты атомарны, то есть если ни один из его атрибутов нельзя разделить на более простые атрибуты, которые соответствуют каким-то другим свойствам описываемой сущности.

Первая нормальная форма представлена на рисунке 2.2.

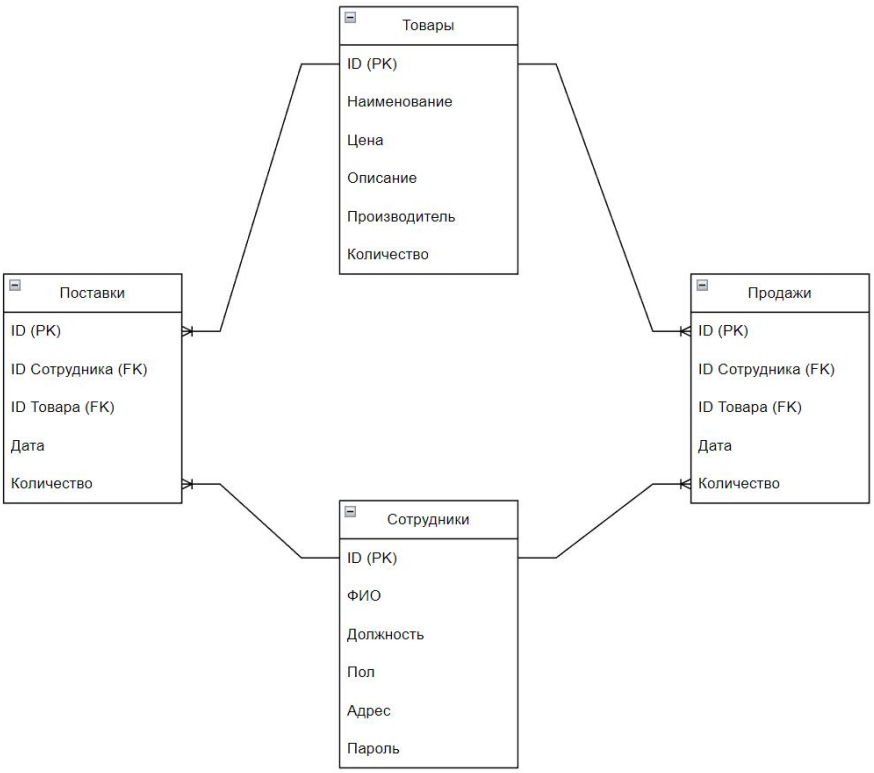



Рисунок 2.2 – Первая нормальная форма

Отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо зависит от (каждого) её потенциального ключа.

Для того чтобы привести таблицу к 2НФ необходимо найти атрибуты, частично зависящие от ключа и вынести их в новую сущность. Вторая нормальная форма представлена на рисунке 2.3.

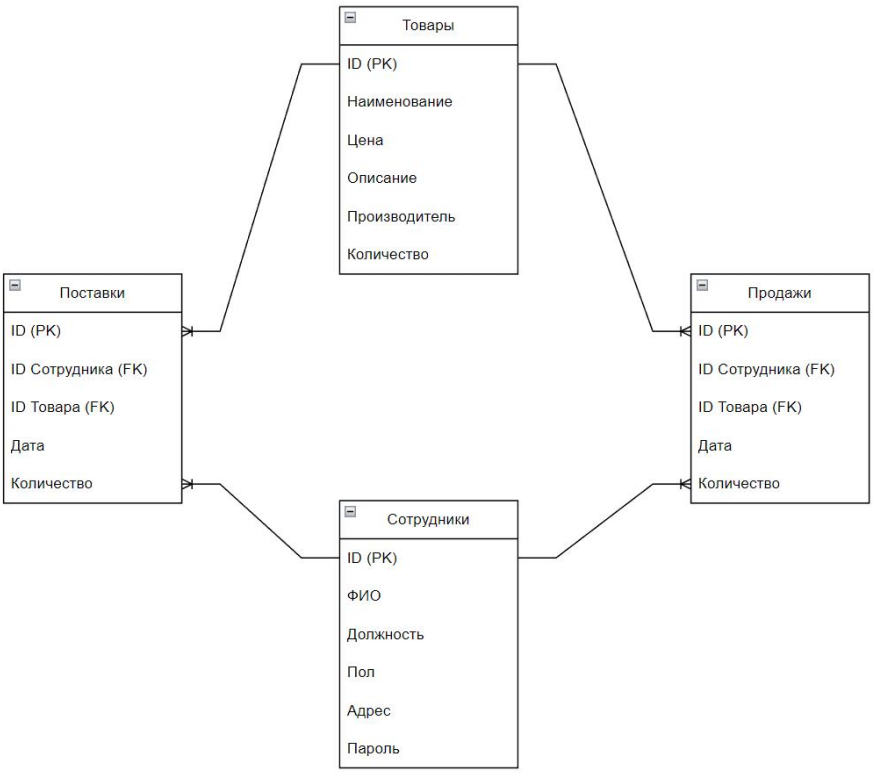



Рисунок 2.3 – Вторая нормальная форма

Отношение находится в третьей нормальной форме, когда находится во 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа. Если в отношении существует транзитивная зависимость между атрибутами, то транзитивно зависимые атрибуты удаляются из него и помещаются в новое отношение вместе с копией их детерминанта.

Третья нормальная форма представлена на рисунке 2.4.

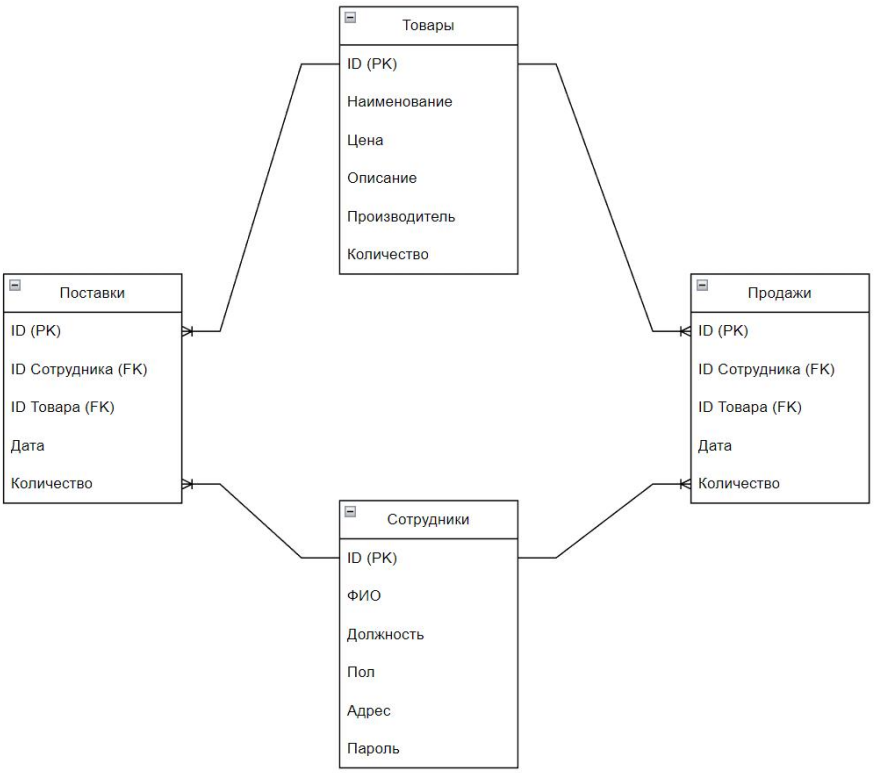



Рисунок 2.4 – Третья нормальная форма

### 2.1.3 Графическое представление логической модели данных в методологии IDEF1x

Реляционная модель данных в методологии IDEF1x представлена на рисунке 2.5.

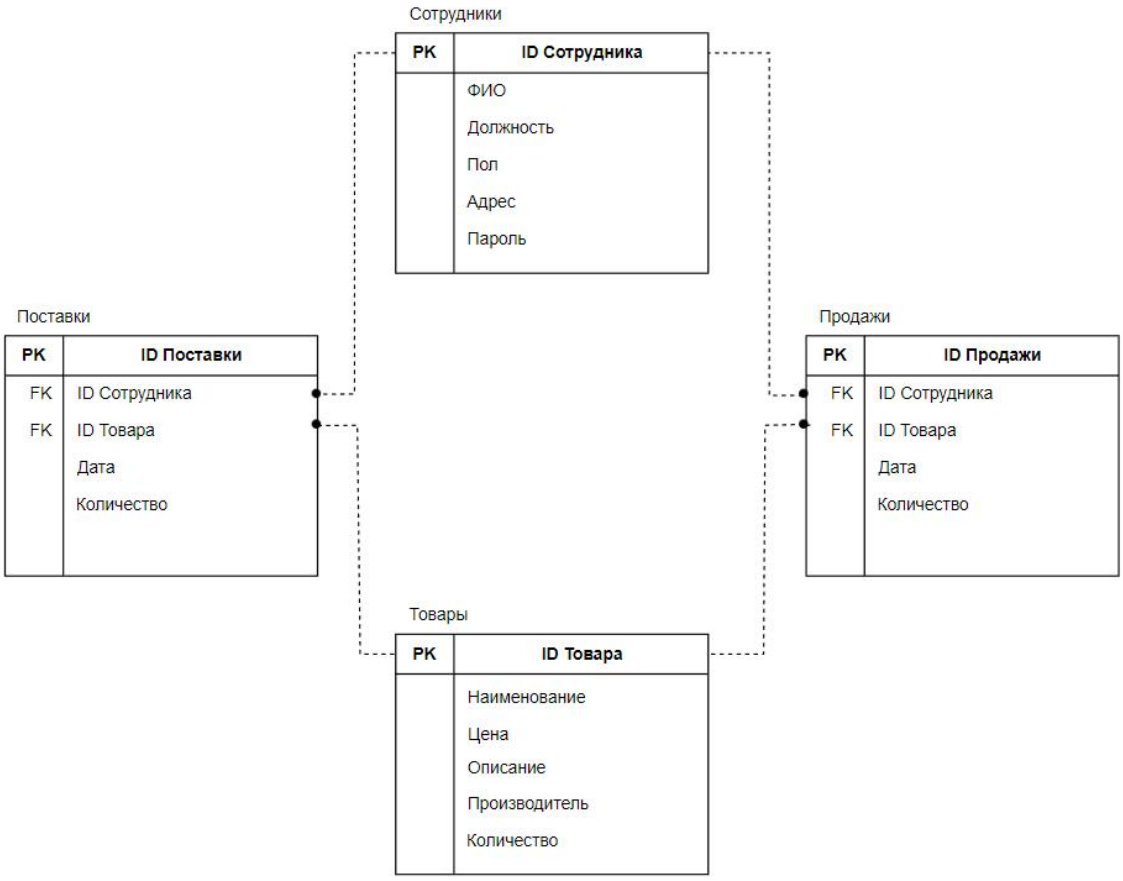



Рисунок 2.5 – Логическая модель данных IDEF1x

## 2.2 Физическое проектирование с учётом выбранной СУБД

PostgreSQL создана на основе некоммерческой СУБД Postgres, разработанной как open-source проект в Калифорнийском университете в Беркли. К разработке Postgres, начавшейся в 1986 году, имел непосредственное отношение Майкл Стоунбрейкер, руководитель более раннего проекта Ingres, на тот момент уже приобретённого компанией Computer Associates. Название

расшифровывалось как «Post Ingres», и при создании Postgres были применены многие ранние наработки.

### 2.2.1 Определение типов данных атрибутов отношений

Каждое реляционное отношение соответствует одной сущности и в него вносятся все атрибуты этой сущности. Отношения приведены в таблицах 2.1 – 2.5. Для каждого отношения указаны атрибуты с их внутренним названием, типом и условиями на значение.

Таблица 2.1 – Описание таблицы «Сотрудники»

| Имя столбца | Тип данных | Описание | Ограничения |
|---|---|---|---|
| id_сотрудника | Счётчик | Уникальный идентификатор сотрудника | Первичный ключ. Условие на значение: >0 |
| ФИО | Текстовый | Фамилия, имя, отчество | Обязательное поле. Размер поля 50. |
| Должность | Текстовый | Занимаемая должность | Обязательное поле. Размер поля 50. |
| Пол | Текстовый | Пол человека | Обязательное поле. Размер поля 50. |
| Адрес | Текстовый | Домашний адрес | Обязательное поле. Размер поля 11. |
| Пароль | Текстовый | Пароль для доступа к БД | Обязательное поле. |

Таблица 2.2 – Описание таблицы «Товары»

| Имя столбца | Тип данных | Описание | Ограничения |
|---|---|---|---|
| id_товара | Счётчик | Уникальный идентификатор сотрудника | Первичный ключ. Условие на значение: >0 |
| Наименование | Текстовый | Наименование товара | Обязательное поле. Размер поля 50. |
| Цена | Текстовый | Цена товара | Обязательное поле. Размер поля 50. |
| Описание | Текстовый | Описание товара | Обязательное поле. Размер поля 50. |
| Производитель | Текстовый | Название компании производителя | Обязательное поле. Размер поля 50. |
| Количество | Мало-целочисленный | Количество товара на складе | Обязательное поле. Не меньше 0 |

Таблица 2.3 – Описание таблицы «Продажи»

| Имя столбца | Тип данных | Описание | Ограничения |
|---|---|---|---|
| id_продажи | Счётчик | Номер продажи | Первичный ключ. Условие на значение: >0 |
| id_сотрудника | Мало-целочисленный | Идентификатор сотрудника ответственного за продажу | Внешний ключ. Обязательное поле. |
| id_сотрудника | Мало-целочисленный | Идентификатор продаваемого товара. | Внешний ключ. Обязательное поле. |
| Дата | Дата | Дата продажи товара | Размер поля 50. Обязательное поле. |
| Количество | Мало-целочисленный | Количество продаваемого товара | Обязательное поле. Размер поля 50. |

Таблица 2.4 – Описание таблицы «Поставки»

| Имя столбца | Тип данных | Описание | Ограничения |
|---|---|---|---|
| id_поставки | Счетчик | Номер поставки | Первичный ключ. Условие на значение: >0 |
| id_сотрудника | Мало-целочисленный | Идентификатор сотрудника ответственного за поставку | Внешний ключ. Обязательное поле. |
| id_сотрудника | Мало-целочисленный | Идентификатор поставляемого товара. | Внешний ключ. Обязательное поле. |
| Дата | Дата | Дата поставки товара | Размер поля 50. Обязательное поле. |
| Количество | Мало-целочисленный | Количество поставляемого товара | Обязательное поле. Размер поля 50. |

### 2.2.2 Графическое представление физической модели данных

Была построена диаграмму базы данных. Результат построения представлен на рисунке 2.6.

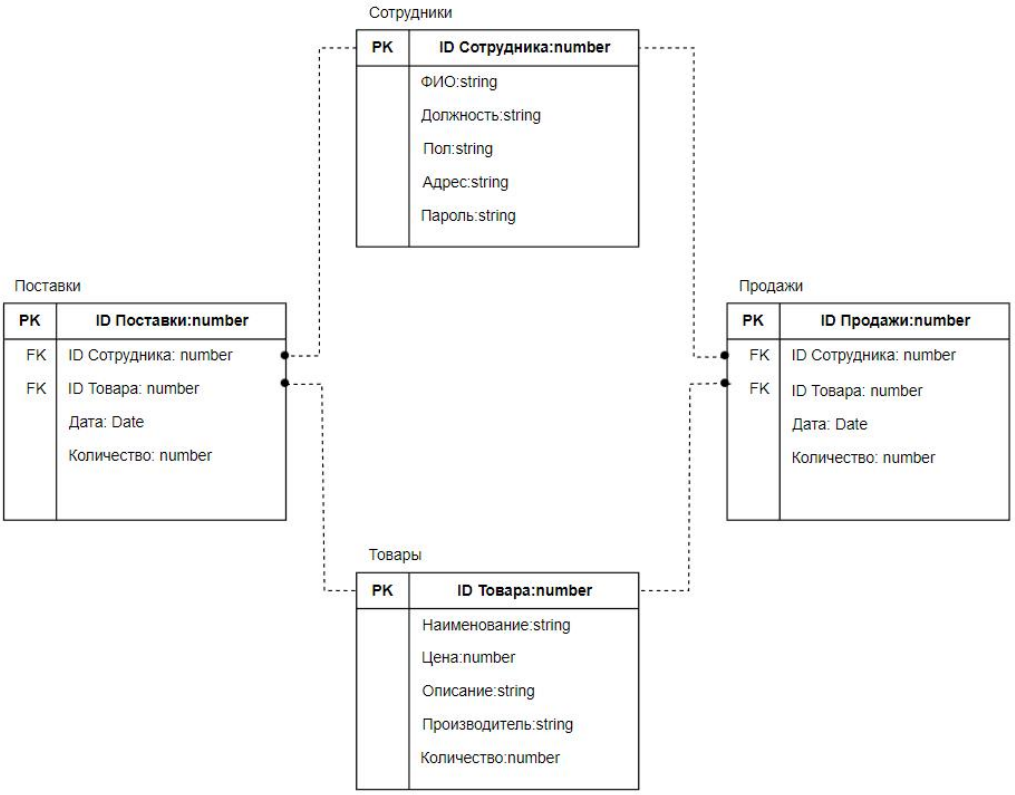



Рисунок 2.6 – Диаграмма базы данных

## 2.3 Представление физической модели данных на языке SQL

Физическая модель данных на языке SQL представлена в приложении А.

# 3 ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ РАБОТЫ С БД

## 3.1 Описание прикладной программы

### 3.1.1 Страница авторизации и страница выбора должности

Перед началом работы в приложении, необходимо выбрать должность пользователя. Главная страница 3.1. Она содержит 3 должности – «Менджер», «Продавец», «Директор». После выбора должности менеджера и продавца перенаправит на страницу выбора сотрудника (Рисунок 3.2). Директору же необходимо будет ввести пароль. После выбора сотрудника продавцам и менеджерам тоже будет необходимо ввести пароль (Рисунок 3.3).

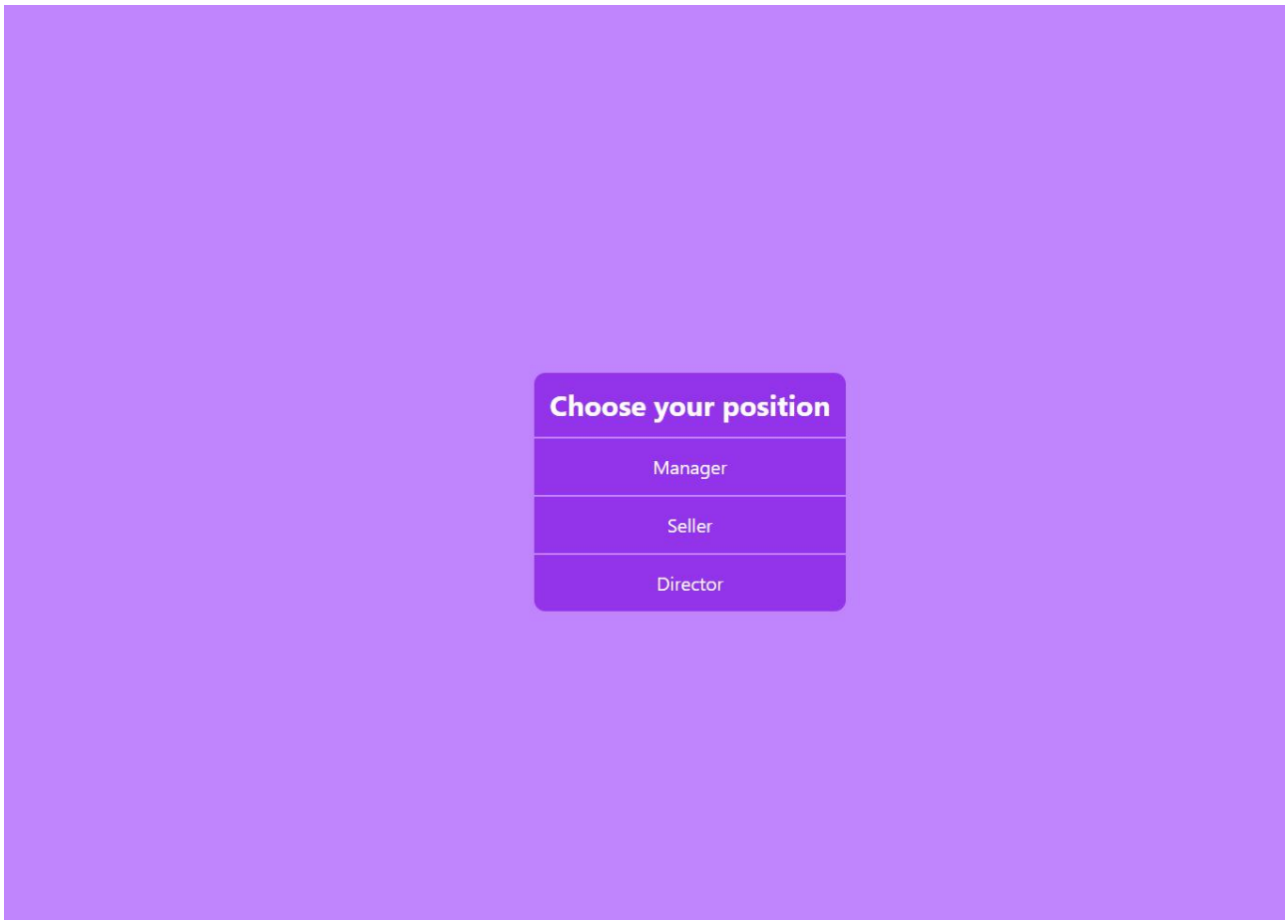



Рисунок 3.1 – Страница выбора должности

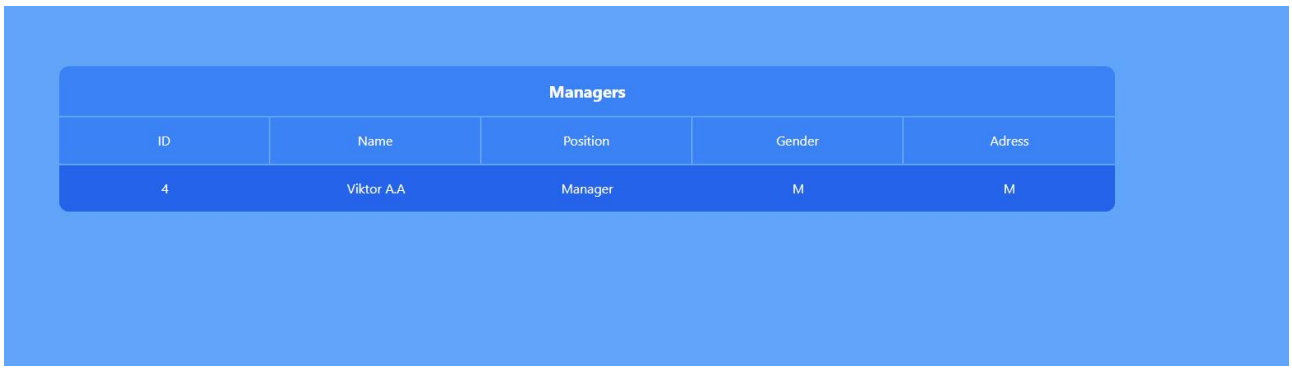



Рисунок 3.2 – Страница выбора сотрудника

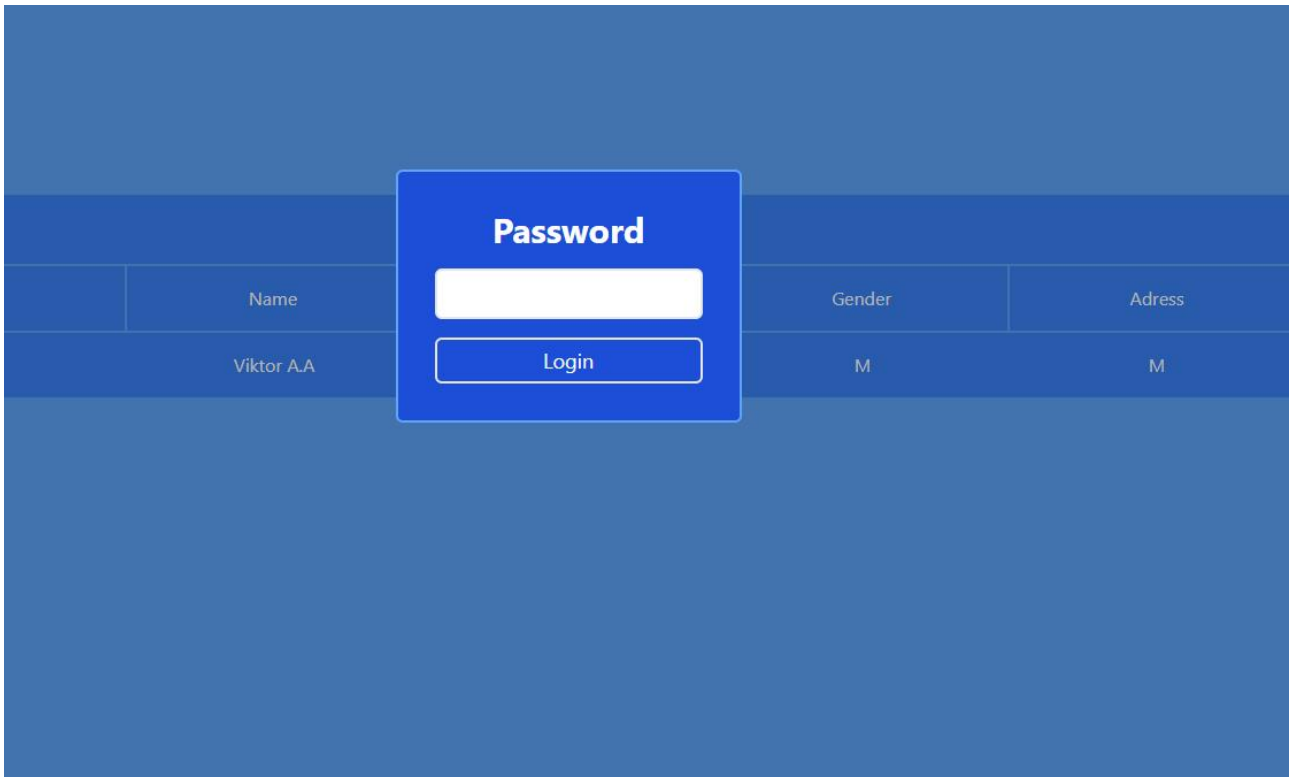



Рисунок 3.3 – Окно ввода пароля

### 3.1.2 Страница продавцов

Перед продавцом будет таблица со списком товаров, для продажи товара необходимо ввести количество продаваемого товара в ячейку и нажать «Sell».(Рисунок 3.4)

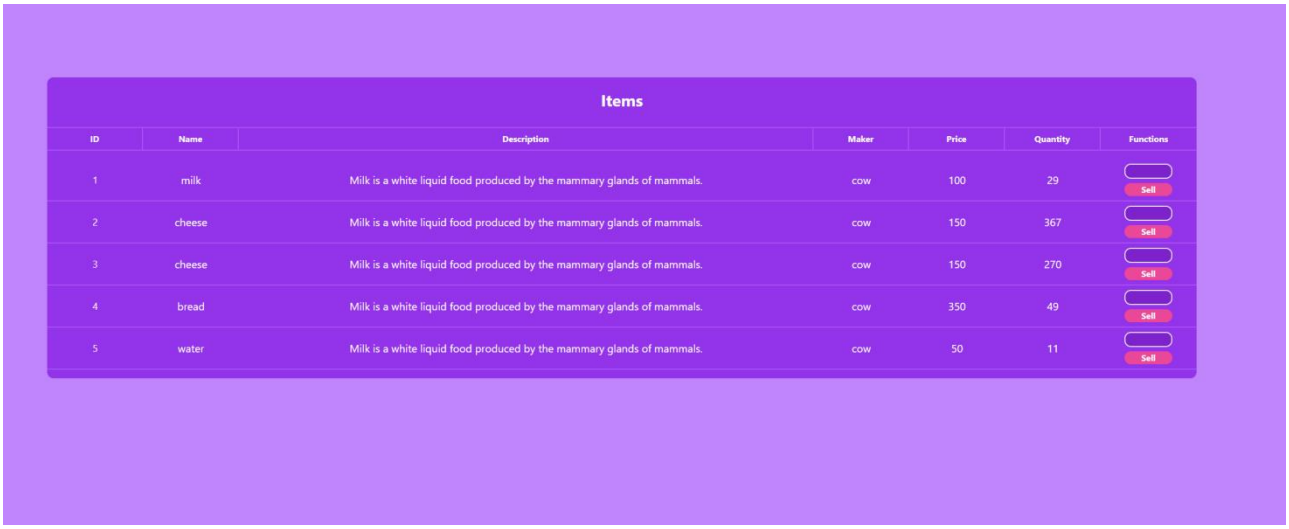



Рисунок 3.4 – Страница продавцов

### 3.1.3 Страница менеджеров

Перед менеджером будет таблица со списком товаров, для заказа товара необходимо ввести количество поставляемого товара в ячейку и нажать «Order»(Рисунок 3.5).

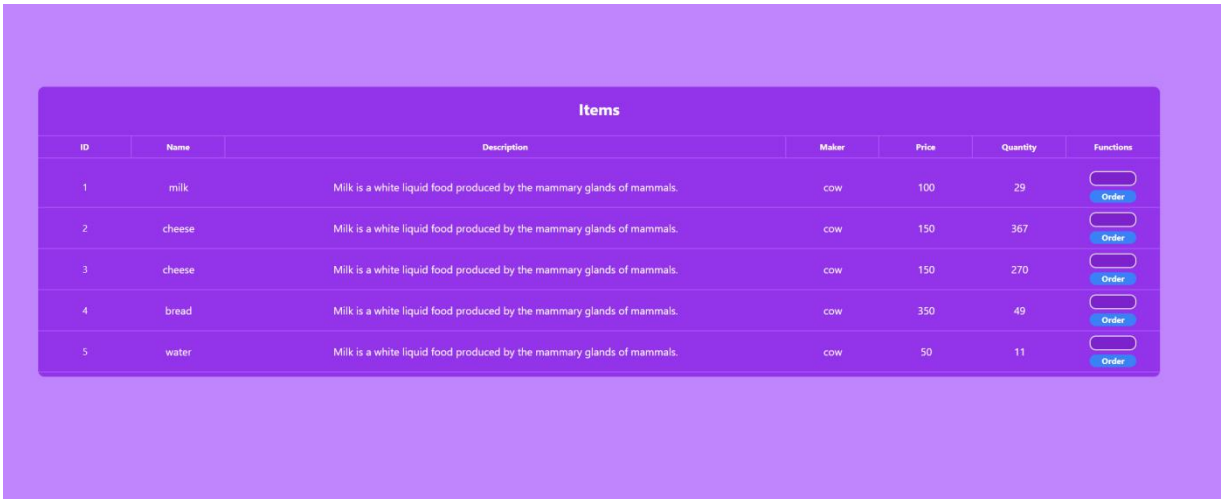



Рисунок 3.5 – Страница менеджеров

### 3.1.4 Страница директора

Перед директором будет таблица со списком поставок и продаж по которой можно будет вести учёт товаров (Рисунок 3.6).

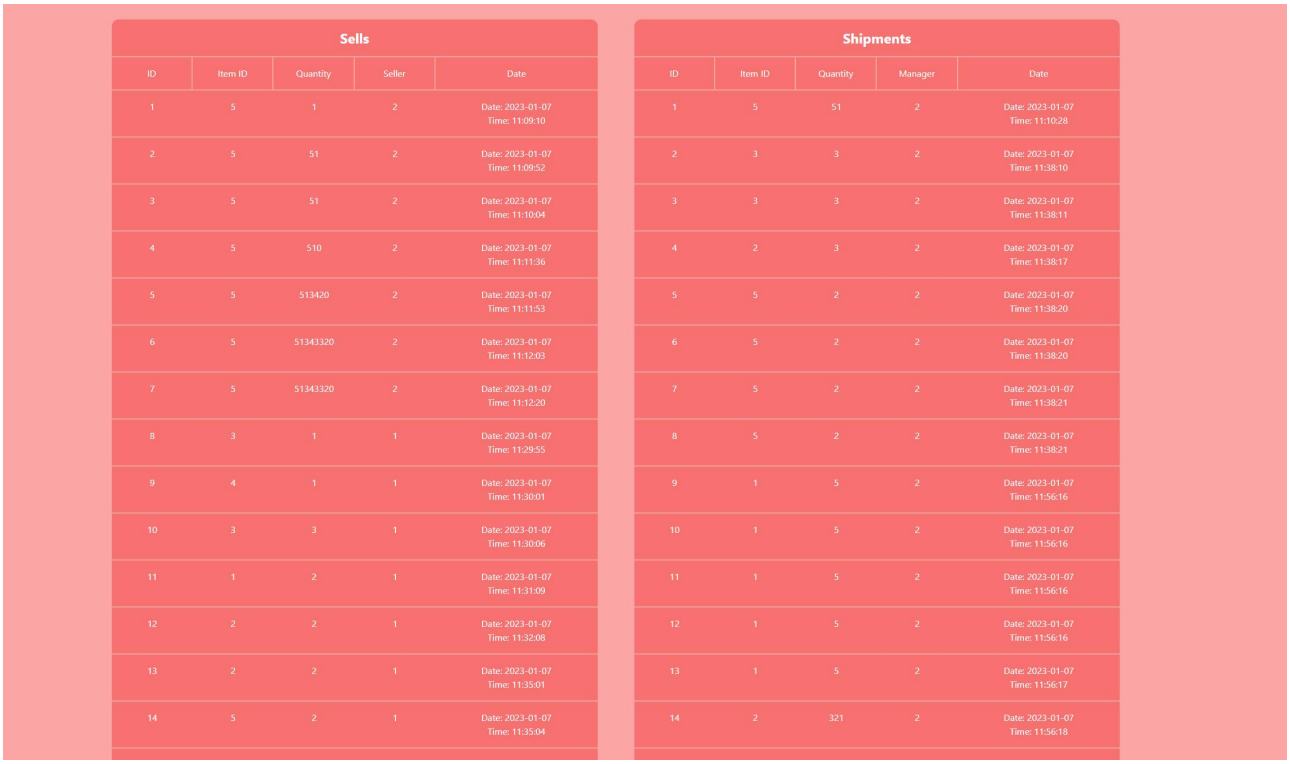



Рисунок 3.6 – Страница директора

## 3.2 Руководства администратора

Пользователям для работы с платформой необходимы базовые знания ПК.

Администратору необходимы базовые знания администрирования системы.

Для эксплуатации разрабатываемой информационной системы необходимы следующие условия:

– компьютер под управлением операционной системы Windows 7 и выше;

– питание устройства от сети или батареи;

– наличие дисплея или монитора;

– устройства ввода.

– браузер

Доступ к работе с интерфейсом системы имеют только авторизованные пользователи.

Администраторы систем не имеют технической возможности узнать пользовательские пароли (пароли хранятся в системе в зашифрованном виде).

Для функционирования системы необходимо техническое обеспечение с характеристиками, описанными в таблице 3.1.

Таблица 3.1 – Минимальные требования к техническому обеспечению клиентской части

| Компонент | Требования |
|---|---|
| Процессор | Тактовая частота 1 ГГц и выше. |
| Память | ОЗУ 128 Мб. |
| Свободное пространство | 40 Мб. |
| Дисплей и периферийные устройства | Разрешение экрана не менее 1280x720. |

Для функционирования системы на сервере должно быть установлено следующее программное обеспечение: NodeJS.

## 3.3 Руководство программиста

Проект состоит из компонентов, представленных на рисунке 3.7.

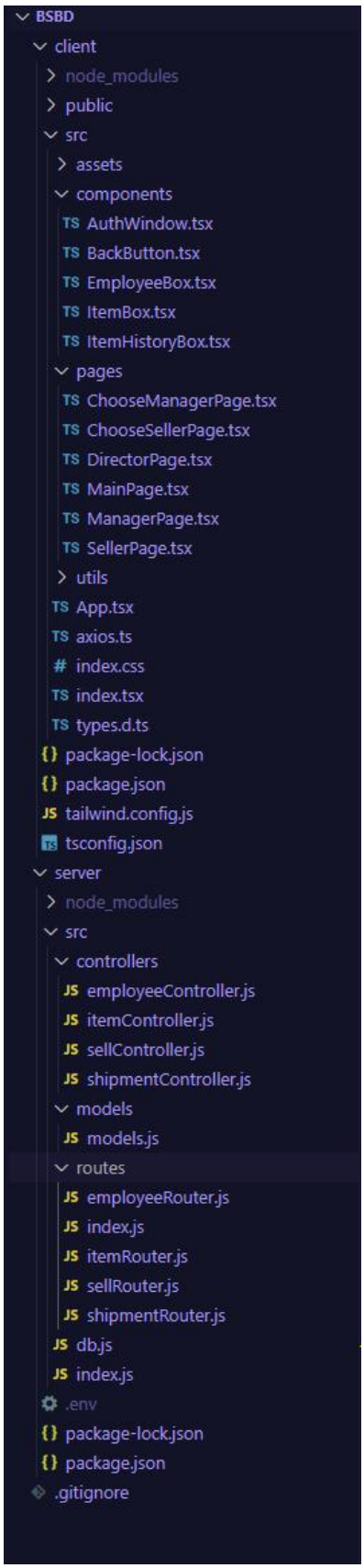



Рисунок 3.7 – Структура проекта

Пример листинга проекта приведён в приложении Б.

# Заключение

В результате выполнения курсовой работы, были получены навыки проектирования и разработки автоматизированных систем, а также:

– неформально описано ПО с использованием естественного языка;

– описаны бизнес-процессы ПО в методологии функционального моделирования IDEF0 до внедрения автоматизированной информационной системы;

– описаны бизнес-процессы ПО в методологии функционального моделирования IDEF0 после внедрения автоматизированной информационной системы;

– построена концептуальная информационная модель данных для ПО;

– определены типы пользователей и их ограничения по работе с объектами ПО;

– определены отношения и связи между отношениями на основе концептуальной информационной модели и выделены первичные и внешние ключи;

– составлена логическая модель данных в методологии IDEF1x;

– определены типы данных атрибутов отношений;

– составлена физическая модель данных;

– составлена физическая модель данных на языке SQL;

– разработан программный комплекс для работы с БД для учёта продаж и поставок в магазине;

– составлены руководства администратора, пользователя и программиста для программного комплекса.

# Список использованных источников

1 ОС ТУСУР 01-2021. Библиографическая ссылка. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления. – Томск: Томский гос. ун-т систем упр. и радиоэлектроники, 2021. – 52с.

2 Курс: Безопасность систем баз данных (1 и 2 семестры). [Электронный ресурс]: Система дистанционного образования ТУСУР. URL: https://sdo.tusur.ru/course/view.php?id=2121 (дата обращения 25.11.2022).

# Приложение А

(обязательное)

**Физическая модель данных на языке SQL**

```
CREATE DATABASE "BSBD"
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'Russian_Russia.1251'
    LC_CTYPE = 'Russian_Russia.1251'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1
    IS_TEMPLATE = False;

SELECT 1+1 AS result
 SELECT table_name FROM information_schema.tables WHERE table_schema = 'public' AND table_name = 'employees'
 CREATE TABLE IF NOT EXISTS "employees" ("id"  SERIAL , "name" VARCHAR(255) NOT NULL, "position" VARCHAR(255) NOT NULL, "gender" VARCHAR(255), "address" VARCHAR(255), "password" VARCHAR(255) NOT NULL, "createdAt" TIMESTAMP WITH TIME ZONE NOT NULL, "updatedAt" TIMESTAMP WITH TIME ZONE NOT NULL, PRIMARY KEY ("id"));

SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey, array_agg(a.attnum) as column_indexes, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM pg_class t,
pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND t.relkind = 'r' and t.relname = 'employees' GROUP BY i.relname, ix.indexrelid, ix.indisprimary, ix.indisunique, ix.indkey ORDER BY i.relname;

SELECT table_name FROM information_schema.tables WHERE table_schema = 'public' AND table_name = 'items'
CREATE TABLE IF NOT EXISTS "items" ("id"  SERIAL , "name" VARCHAR(255) NOT NULL, "price" INTEGER NOT NULL, "description" TEXT NOT NULL, "maker" VARCHAR(255) NOT NULL, "quantity" INTEGER NOT NULL, "createdAt" TIMESTAMP WITH TIME ZONE NOT NULL, "updatedAt" TIMESTAMP WITH TIME ZONE NOT NULL, PRIMARY KEY ("id"));

 SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey, array_agg(a.attnum) as column_indexes, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM pg_class t,
```

pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND t.relkind = 'r' and t.relname = 'items' GROUP BY i.relname, ix.indexrelid, ix.indisprimary, ix.indisunique, ix.indkey ORDER BY i.relname;

SELECT table_name FROM information_schema.tables WHERE table_schema = 'public' AND table_name = 'shipments'
CREATE TABLE IF NOT EXISTS "shipments" ("id"  SERIAL , "quantity" INTEGER NOT NULL, "createdAt" TIMESTAMP WITH TIME ZONE NOT NULL, "updatedAt" TIMESTAMP WITH TIME ZONE NOT NULL, "employeeId" INTEGER REFERENCES "employees" ("id") ON DELETE SET NULL ON UPDATE CASCADE, "itemId" INTEGER REFERENCES "items" ("id") ON DELETE SET NULL ON UPDATE CASCADE, PRIMARY KEY ("id"));

SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey, array_agg(a.attnum) as column_indexes, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM pg_class t, pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND t.relkind = 'r' and t.relname = 'shipments' GROUP BY i.relname, ix.indexrelid, ix.indisprimary, ix.indisunique, ix.indkey ORDER BY i.relname;

SELECT table_name FROM information_schema.tables WHERE table_schema = 'public' AND table_name = 'sells'
(default): CREATE TABLE IF NOT EXISTS "sells" ("id"  SERIAL , "quantity" INTEGER NOT NULL, "createdAt" TIMESTAMP WITH TIME ZONE NOT NULL, "updatedAt" TIMESTAMP WITH TIME ZONE NOT NULL, "employeeId" INTEGER REFERENCES "employees" ("id") ON DELETE SET NULL ON UPDATE CASCADE, "itemId" INTEGER REFERENCES "items" ("id") ON DELETE SET NULL ON UPDATE CASCADE, PRIMARY KEY ("id"));

SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey, array_agg(a.attnum) as column_indexes, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM pg_class t, pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND t.relkind = 'r' and t.relname = 'sells' GROUP BY i.relname, ix.indexrelid, ix.indisprimary, ix.indisunique, ix.indkey ORDER BY i.relname;

# Приложение Б

(обязательное)

## Листинг кода класса сотрудника

```
class EmployeeController {

	async getAll(req, res, next) {
		try {
			const employees = await Employee.findAll({ order: ["id"] });
			return res.status(200).json(employees);
		} catch (error) {
			console.log(error);
		}
	 }

	async create(req, res, next) {
		try {
			const { name, position, gender, address, password } = req.body;
			const hashPassword = await bcrypt.hash(password, 6);
			const user = await Employee.create({
			name,
			position,
			gender,
			address,
			password: hashPassword,
			});
			return res.status(200).json(user);
		} catch (error) {
			console.log(error);
		}
	}

	async auth(req, res, next) {
		try {
			const { id, password } = req.body;
			const employee = await Employee.findOne({ where: { id } });
			if (!employee) {
				throw "No such employee";
			}
			const result = await bcrypt.compare(password, employee.password);
			return res.status(200).json(result);
		} catch (error) {
		console.log(error);
		}
	}
}
```