



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ Информатика и системы управления (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

*НА ТЕМУ:*

База данных для транспортной системы завода

Студент

\_\_\_\_\_  
*подпись, дата*

Иванов В.А.

*фамилия, и.о.*

Руководитель курсовой работы

\_\_\_\_\_  
*подпись, дата*

Исаев А.Л.

*фамилия, и.о.*

2021 г.



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ  
Заведующий кафедрой ИУ7  
(Индекс)  
И.В. Рудаков  
(И.О.Фамилия)  
« \_\_\_\_ » \_\_\_\_\_ 2021 г.

## З А Д А Н И Е на выполнение курсового проекта

по дисциплине Базы данных

Студент группы ИУ7-62Б

Иванов Всеволод Алексеевич  
(Фамилия, имя, отчество)

Тема курсового проекта База данных для транспортной системы завода

Направленность КП (учебный, исследовательский, практический, производственный, др.)  
учебный

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

**Задание** Спроектировать и реализовать базу данных для транспортной системы завода. С помощью Web-приложения реализовать интерфейс для работы с информацией о заказах, записях о поездках, сотрудниках, дежурствах путём взаимодействия с базой данных. Также реализовать функционал для разных категорий пользователей

### Оформление курсового проекта:

Расчетно-пояснительная записка на 20-30 листах формата А4.

Расчетно-пояснительная записка должна содержать постановку задачи, введение, аналитическую, конструкторскую, технологическую части, заключение, список литературы.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.): на защиту проекта должна быть предоставлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграммы, интерфейс.

Дата выдачи задания « \_\_\_\_ » \_\_\_\_\_ 2021 г.

Руководитель курсового проекта

А.Л. Исаев  
(Подпись, дата) (И.О.Фамилия)

Студент

В.А. Иванов  
(Подпись, дата) (И.О.Фамилия)

## Реферат

Суть курсового проекта заключается в реализации базы данных для транспортной системы завода и web-приложения для предоставления интерфейса доступа к ней. В результате выполнения работы было получено программное обеспечение предоставляющее возможность планировать, отслеживать и регулировать работу службы доставки.

Приложение реализовано с использованием фреймворка Django на языке программирования Python 3. В качестве СУБД был использован PostgreSQL.

**Ключевые слова:** доставка, транспортная система, завод, web-приложение, PostgreSQL, Django.

РПЗ содержит 45 страниц, 12 таблиц, 18 иллюстраций, 10 ссылок на используемые источники.

# Оглавление

<b>Введение</b>	<b>6</b>
<b>1 Аналитическая часть</b>	<b>7</b>
1.1 Постановка задачи . . . . .	7
1.2 Формализация данных . . . . .	8
1.3 Формализация ролей . . . . .	9
1.4 Базы данных . . . . .	9
1.4.1 Модели данных . . . . .	10
1.4.2 СУБД . . . . .	11
<b>2 Конструкторская часть</b>	<b>12</b>
2.1 Диаграмма вариантов использования . . . . .	12
2.2 Проектирование базы данных . . . . .	13
2.2.1 Сущности базы данных . . . . .	13
2.2.2 Проектирование таблиц . . . . .	14
2.2.3 Нормализация схем отношений . . . . .	17
2.2.4 Схемы триггеров . . . . .	18
2.3 Архитектура приложения . . . . .	20
<b>3 Технологическая часть</b>	<b>22</b>
3.1 Выбор средств программной реализации . . . . .	22
3.1.1 Язык программирования . . . . .	22
3.1.2 СУБД и ORM . . . . .	22
3.1.3 Web-фреймворк . . . . .	23
3.2 UML-диаграммы компонентов приложения . . . . .	24
3.2.1 Компонент доступа к данным . . . . .	24
3.2.2 Компонент бизнес-логики . . . . .	25
3.2.3 Компонент представления . . . . .	26
3.2.4 Диаграмма приложения . . . . .	27

3.3	Реализация базы данных . . . . .	28
3.3.1	Создание таблиц . . . . .	28
3.3.2	Ролевая модель . . . . .	29
3.3.3	Триггеры . . . . .	31
3.3.4	Функции . . . . .	32
3.4	Интерфейс приложения . . . . .	33
	<b>Заключение</b>	<b>39</b>
	<b>Список литературы</b>	<b>40</b>
	<b>Приложение А. Код хранимых функций</b>	<b>42</b>

## Введение

Одним из ключевых аспектов эффективного ведения любого крупного или среднего бизнеса в настоящее время является хорошая связь между его сотрудниками[1]. Быстрый и удобный доступ к данным позволяет оперативно координировать всех работников и минимизировать временные издержки по передаче текущих задач. Поэтому, актуальным направлением в современном мире является разработка сервисов, реализующих данные возможности. Объектом разработки данного курсового проекта выбрана транспортная система завода, которая должна выполнять доставку заказов.

**Целью** данного курсовой работы является разработка базы данных для транспортной системы завода и web-приложения для доступа к ней.

Выделены следующие задачи курсового проекта:

- формализовать задание, определить необходимый функционал;
- для структурированного хранения информации спроектировать базу данных;
- проанализировать существующие СУБД и обосновать выбор одной из них;
- реализовать базу данных и интерфейс доступа к ней с использованием выбранной СУБД;
- обосновать выбор web-фреймворка;
- реализовать web-приложение для выделенного функционала.

## **1. Аналитическая часть**

### **1.1. Постановка задачи**

Необходимо разработать базу данных, которая будет хранить информацию о сотрудниках, заказах, записях, дежурствах и о проездах машин через контрольно-пропускные пункты, а также web-приложение для предоставления этих данных.

Также должны быть реализованы различные категории сотрудников, для которых будет предоставляться определённый функционал по работе с данными транспортной системы. Для этого должна существовать возможность регистрации аккаунтов и дальнейшего взаимодействия с ними.

## 1.2. Формализация данных

В соответствии с предметной областью, соответствующей описанному набором требований, база данных должна хранить сущности, описанные ER-диаграммой на рисунке 1.1

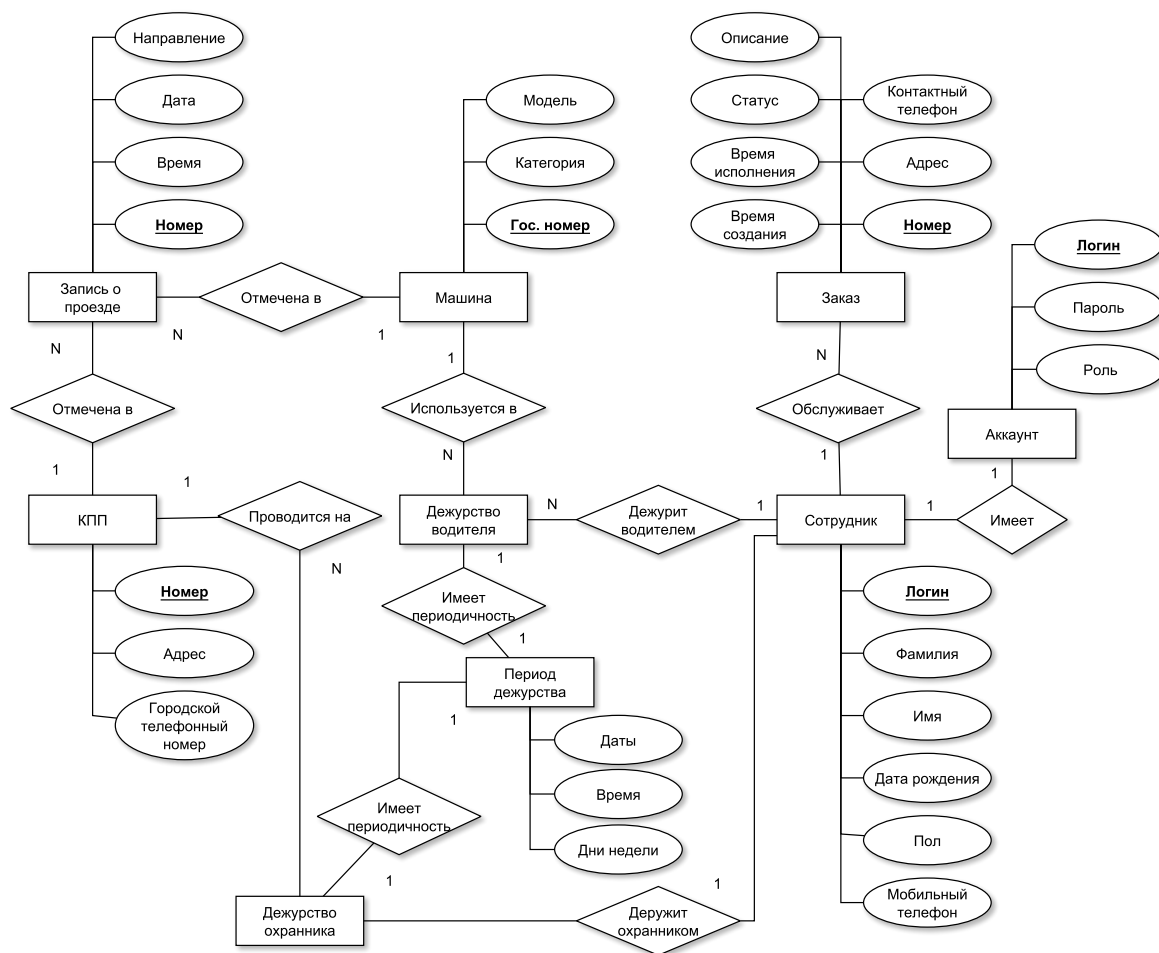


Рис. 1.1 — ER-диаграмма сущностей



### 1.3. Формализация ролей

Для функционирования транспортной системы были выделено несколько должностей сотрудников, названия и функционал которых приведены в таблице 1.1.

Таблица 1.1 — Роли сотрудников и их функционал

Должность	Функционал
Администратор	Назначение дежурств и заказов, рассмотрение заявок на регистрацию, создание новых сущностей: машин, КПП, заказов
Водитель	Выбор заказа, выполнение заказа, просмотр дежурств и личной информации
Охранник	Создание записей о проезде, просмотр дежурств и личной информации
Неподтверждённый сотрудник	Просмотр личной информации
Неавторизованный пользователь	Регистрация, авторизация

Для упрощения регистрации новых сотрудников в системе принято решение предоставить им возможность заполнения заявок, которые в дальнейшем могут быть подтверждены администратором.

### 1.4. Базы данных

База данных — это совокупность взаимосвязанных структурированных данных, относящихся к определенной предметной области и организованных так, чтобы обеспечить независимость данных от программ обработки. [2]

### 1.4.1. Модели данных

Модель данных - это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных. [3]

Существует три основные модели данных.

- Иерархическая. База данных представляется в виде древовидной структуры, состоящей из объектов различных уровней. Объекты находятся в отношении предка к потомку, причём объект может иметь только одного предка и любое количество потомков.
- Сетевая. Отличием от иерархической модели данных является отсутствие ограничения на количество предков. База данных в такой модели состоит из набора экземпляров записи и экземпляров связей между записями.
- Реляционная. Основной идеей является то, что все наборы данных представляются в виде множеств. База данных состоит из множества двумерных таблиц. Таблицы состоят из записей и полей, каждая запись имеет собственные значения для каждого поля. Иерархия элементов отсутствует.

Реляционная модель в настоящее время является наиболее гибкой и удобной в использовании, а также обладает наибольшим выбором в области систем управления базой данных. Помимо этого, она наилучшим образом подходит для описания выделенных связей между сущностями, поэтому принято решение о её применении в

данной работе.

### **1.4.2. СУБД**

Система управления базами данных (сокращённо СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных [2].

В процессе разработки было принято решение использовать СУБД для взаимодействия с базой данных, так как оно реализует множество функций, которые потребуются в работе с базой данных.

### **Вывод**

Результатом аналитического раздела стала постановка задачи, формализация данных программы и ролей пользователей, описание способов хранения данных и управления ими.



## 2.2. Проектирование базы данных

### 2.2.1. Сущности базы данных

На рисунке 2.2 представлена ER-диаграмма сущностей базы данных.

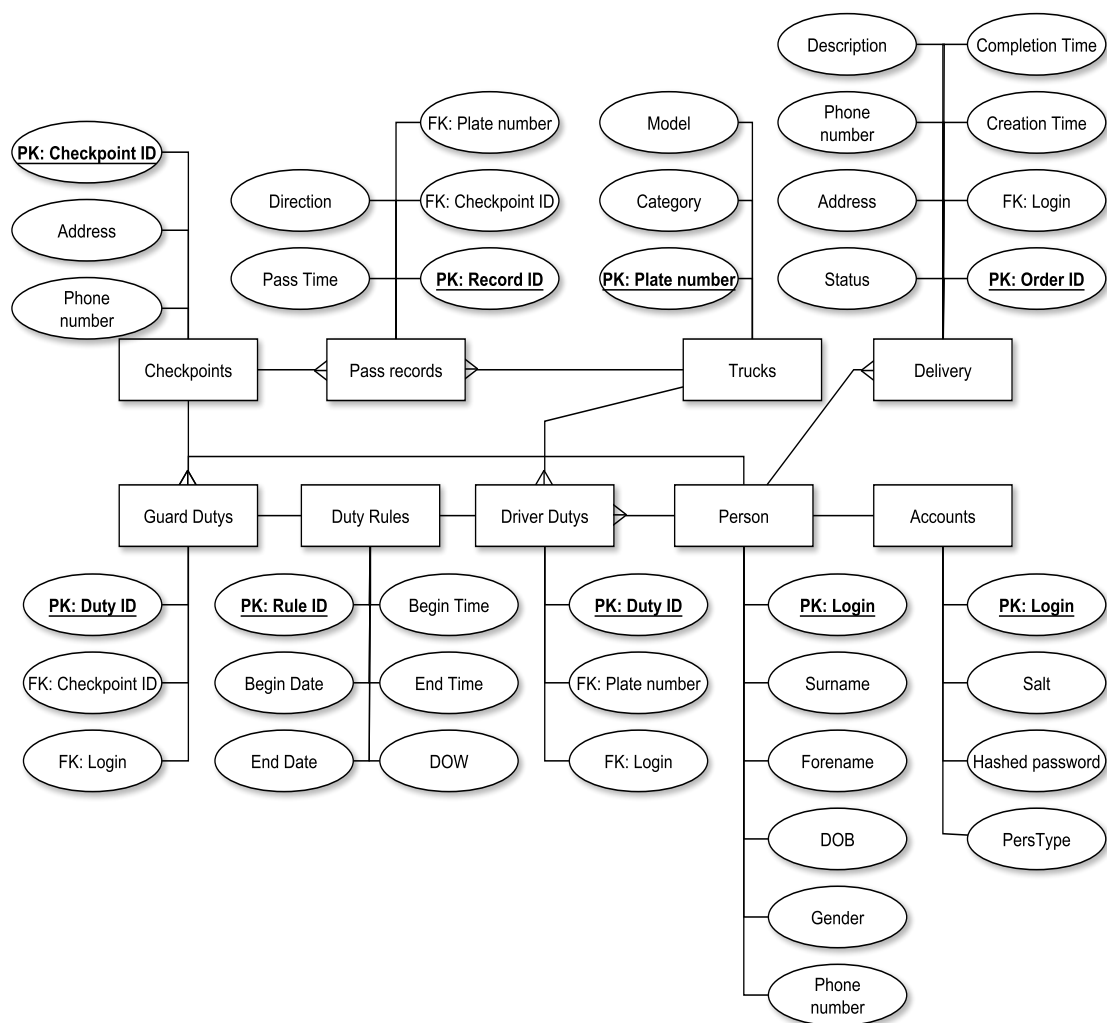


Рис. 2.2 — ER-диаграмма сущностей базы данных

## 2.2.2. Проектирование таблиц

На основании выделенных сущностей база данных должна содержать таблицы, описание которых приведено в таблицах 2.1-2.8

Для обеспечения приватности пароля к аккаунту было принято решение использовать хэширование. В качестве ролей используются значения admin, driver, guard (соотв. администратор, водитель, охранник).

Таблица 2.1 — Accounts (таблица аккаунтов)

Атрибут	Тип	Комментарий
Login	Строка	Логин аккаунта, РК
PersType	Строка	Роль аккаунта
Salt	Строка	"Соль" для хэширования пароля
HashedPassword	Строка	Хэшированный пароль

Таблица 2.2 — Person (таблица личной информации)

Атрибут	Тип	Комментарий
Login	Строка	Логин аккаунта, РК, FK
Surname	Строка	Фамилия сотрудника
Forename	Строка	Имя сотрудника
DOB	Дата	Дата рождения сотрудника
Gender	Строка	Пол сотрудника
PhoneNumber	Строка	Номер мобильного телефона сотрудника

В качестве статуса заказа используются значения not\_assigned, in\_transit, delivered (соотв. заказ не назначен водителю, заказ в процессе доставки и заказ доставлен).

Таблица 2.3 — Trucks (таблица машин)

Атрибут	Тип	Комментарий
PlateNumber	Строка	Гос. номер машины, РК
Category	Строка	Марка машины
Model	Строка	Модель машины

Таблица 2.4 — Delivery (таблица заказов)

Атрибут	Тип	Комментарий
OrderID	Целое число	Уникальный номер заказа, РК
Login	Строка	Логин водителя, доставляющего или доставившего заказ, FK
Address	Строка	Адрес заказчика
PhoneNumber	Строка	Контакты заказчика
Status	Строка	Статус заказа
Description	Строка	Содержание заказа
CreationTime	Дата, время	Дата и время создания заказа
CompletionTime	Дата, время	Дата и время завершения заказа

Таблица 2.5 — Checkpoints (таблица КПП)

Атрибут	Тип	Комментарий
CheckpointID	Целое число	Номер КПП, РК
Address	Строка	Адрес КПП
PhoneNumber	Строка	Номер телефона КПП

Дежурства как охранников, так и водителей имеют циклический характер по неделе. Поэтому рациональнее хранить не каждое дежурство отдельно, а правило, содержащее диапазон дат и дни недели, в которые сотрудник дежурит. Атрибут дней недели в таб-

лице хранит строку, определяющую дни по номеру (например строка 013 соответствует дежурству по понедельникам, вторникам и четвергам). Период (или расписание) дежурства выделено в отдельную таблицу.

Таблица 2.6 — DutyRules (таблица расписаний дежурств)

Атрибут	Тип	Комментарий
RuleID	Целое число	Номер периода дежурства, РК
BeginDate	Дата	Дата начала периода дежурства
EndDate	Дата	Дата окончания периода дежурства
BeginTime	Время	Время начала дежурства
EndTime	Время	Время окончания дежурства
DOW	Строка	Дни недели дежурства

Таблица 2.7 — GuardDutys (таблица дежурств охранников)

Атрибут	Тип	Комментарий
DutyID	Целое число	Номер дежурства, РК
Login	Строка	Логин дежурящего охранника, FK
CheckpointID	Целое число	Номер КПП, FK
RuleID	Целое число	Номер периода дежурства, FK

В качестве направления проезда используются значения in, out (соотв. въезд и выезд с территории предприятия).



Таблица 2.8 — DriverDutys (таблица дежурств водителей)

Атрибут	Тип	Комментарий
DutyID	Целое число	Номер дежурства, РК
Login	Строка	Логин дежурящего водителя, FK
PlateNumber	Строка	Гос. номер машины, FK
RuleID	Целое число	Номер периода дежурства, FK

Таблица 2.9 — PassRecords (таблица записей проездов)

Атрибут	Тип	Комментарий
RecordID	Целое число	Номер записи, РК
PlateNumber	Строка	Гос. номер машины, FK
CheckpointID	Целое число	Номер КПП, FK
PassTime	Дата, время	Дата и время проезда
Direction	Строка	Направление проезда

Также было принято создать таблицу для хранения информации о событиях в базе данных.

Таблица 2.10 — LogActions (таблица активности пользователей)

Атрибут	Тип	Комментарий
Actor	Строка	Роль, осуществившая действие
ActTime	Дата, время	Время действия
Description	Строка	Описание действия

### 2.2.3. Нормализация схем отношений

Распространённым подходом в проектировании баз данных является метод нормализации, служащий для устранения избыточ-

ности и упрощения контроля целостности. При проектировании данной базы данных была поставлена цель соответствия третьей нормальной форме, что является достаточным для выполнения описанных целей нормализации[4].

В разработанной модели данных переменная отношения находится в первой нормальной форме, так как выполняется условие атомарности. В каждом кортеже, соответствующем некоторой описанной таблице, каждый компонент является атомарным, то есть содержит только одно значение для каждого из атрибутов[5]. Данное утверждение основано на том, что каждый атрибут имеет простой тип данных.

Также созданная переменная отношения находится во второй нормальной форме, так как в каждой таблице любой не ключевой атрибут полностью зависит от первичного ключа. Для перехода в данную форму из таблиц дежурств охранников и водителей была отдельно выделена таблица расписаний дежурств.

Можно утверждать, что спроектированная переменная отношения находится в третьей нормальной форме. Во всех таблицах не ключевые атрибуты не связаны отношениями между собой, поэтому можно утверждать, что они зависят от первичного ключа нетранзитивно, что и является условием нахождения в третьей нормальной форме.

#### **2.2.4. Схемы триггеров**

В соответствии с описанной ранее схемой взаимодействия, существуют некоторые действия, которые могут выполнять сразу несколько ролей. Для сбора дополнительной информации о данных действиях требуется реализовать следующие триггеры.

- Триггер сохранения информации добавления записи о проезде. Данное действие может осуществлять как охранник, так и администратор, но таблица PassRecords не хранит информации об авторе, поэтому было решено создать данный триггер. Его схема представлена на рисунке 2.3

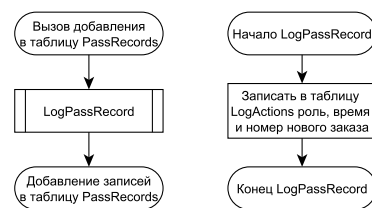


Рис. 2.3 — Схема алгоритма триггера на добавление в таблицу PassRecords

- Триггер сохранения информации обновления заказа. В данном случае действие могут совершать водитель и администратор путём изменения статуса заказа и установлением времени его доставки. Его схема представлена на рисунке 2.4

Также в системе нельзя допускать удаление аккаунтов действующих сотрудников, так как с ними связана некоторая информация. Например, для доставленных заказов обязательно требуется хранить информацию о его курьере. Поэтому также был создан триггер на удаление записей из таблицы Accounts, запрещающий удалять сотрудников с активным статусом. Схема данного триггера представлена на рисунке 2.5

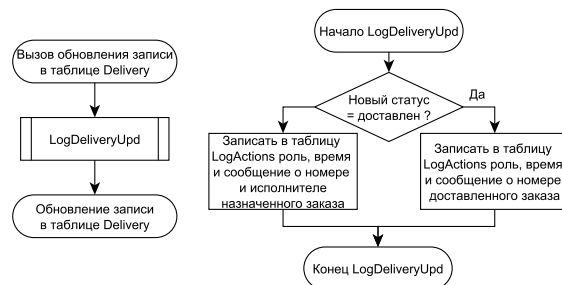


Рис. 2.4 — Схема алгоритма триггера на обновление в таблице Delivery

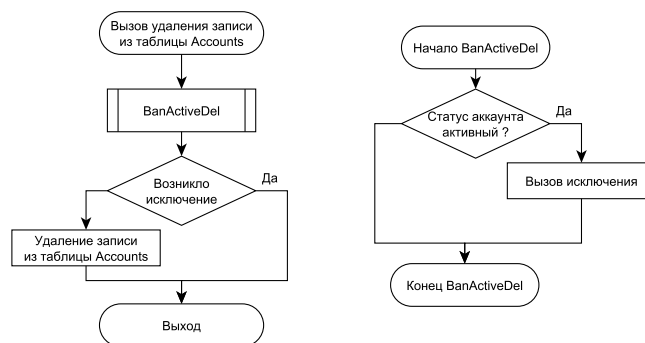


Рис. 2.5 — Схема алгоритма триггера на удаление из таблицы Accounts

## 2.3. Архитектура приложения

Наиболее известным подходом к проектированию архитектуры web-приложения является шаблон MVC. Данный паттерн предполагает разделение приложения на три описанные части.

- Model (модель) - компонент бизнес-логики, отвечает за взаимодействие с базой данных и основную обработку информации.
- View (представление) - компонент, отвечающий за отобра-

жения данных, полученных в результате работы модели.

- Controller (контроллер) - компонент, отвечающий за получение запроса от пользователя, передачу их модели для дальнейшего обновления представления.

В данной курсовой работе будет использован описанный подход, так как он чётко разграничивает зоны каждого компонента, что позволяет сделать их независимыми.

## **Вывод**

Результатом конструкторской части стала разработка сценариев использования приложения, его архитектуры и проектирование таблиц и триггеров базы данных.

## 3. Технологическая часть

### 3.1. Выбор средств программной реализации

#### 3.1.1. Язык программирования

В качестве языка программирования был выбран Python 3[8], по следующим причинам.

- Поддержка ООП, требуемого для архитектурного шаблона MVC.
- Наличие опыта работы с данным языком.
- Данный язык является популярным в разработке web-приложений, поэтому он обладает большим количеством web-фреймворков и библиотек для доступа к СУБД.

#### 3.1.2. СУБД и ORM

Наиболее популярными реляционными СУБД являются Oracle, MySQL и PostgreSQL[6]. Их сравнение[7] можно свести в таблицу 3.1

Таблица 3.1 — Сравнение реляционных СУБД

СУБД	Преимущества	Недостатки
Oracle	+ надёжность системы + поддержка современности функционала	- конечная стоимость СУБД - высокие системные требования
MySQL	+ прост в установке и использовании + производительность + широкий базовый функционал	- проблемы с надёжностью - не полная поддержка SQL

Продолжение на следующей странице

Таблица 3.1 – продолжение

СУБД	Преимущества	Недостатки
PostgreSQL	+ лёгкая масштабируемость + открытость ПО + поддержка текстовых форматов (в т.ч. json)	- низкая скорость выполнения пакетных операций - сложности поиска хостинга

Наиболее подходящим для небольших организаций, которые являются целевыми для разрабатываемой программы, является PostgreSQL[7]. Учитывая наличие опыта работы с данным средством, оно был выбран в качестве СУБД для данного курсового проекта.

В качестве ORM (технология объектно-реляционного преобразования) был выбран реeewe[9], так как она является простой в освоении и использовании, а также позволяет легко подменять СУБД.

### 3.1.3. Web-фреймворк

В качестве web-фреймворка был выбран Django[10], по следующим причинам.

- Django использует шаблон проектирования MVC.
- Лёгкая масштабируемость.
- Готовые решения для наиболее востребованных задач.
- Поддержка шаблонизации html страниц.

## 3.2. UML-диаграммы компонентов приложения

### 3.2.1. Компонент доступа к данным

В данной работе компонент доступа к данным реализован с использованием паттерна проектирования Repository. UML диаграмма компонента изображена представлена на рисунке 3.1

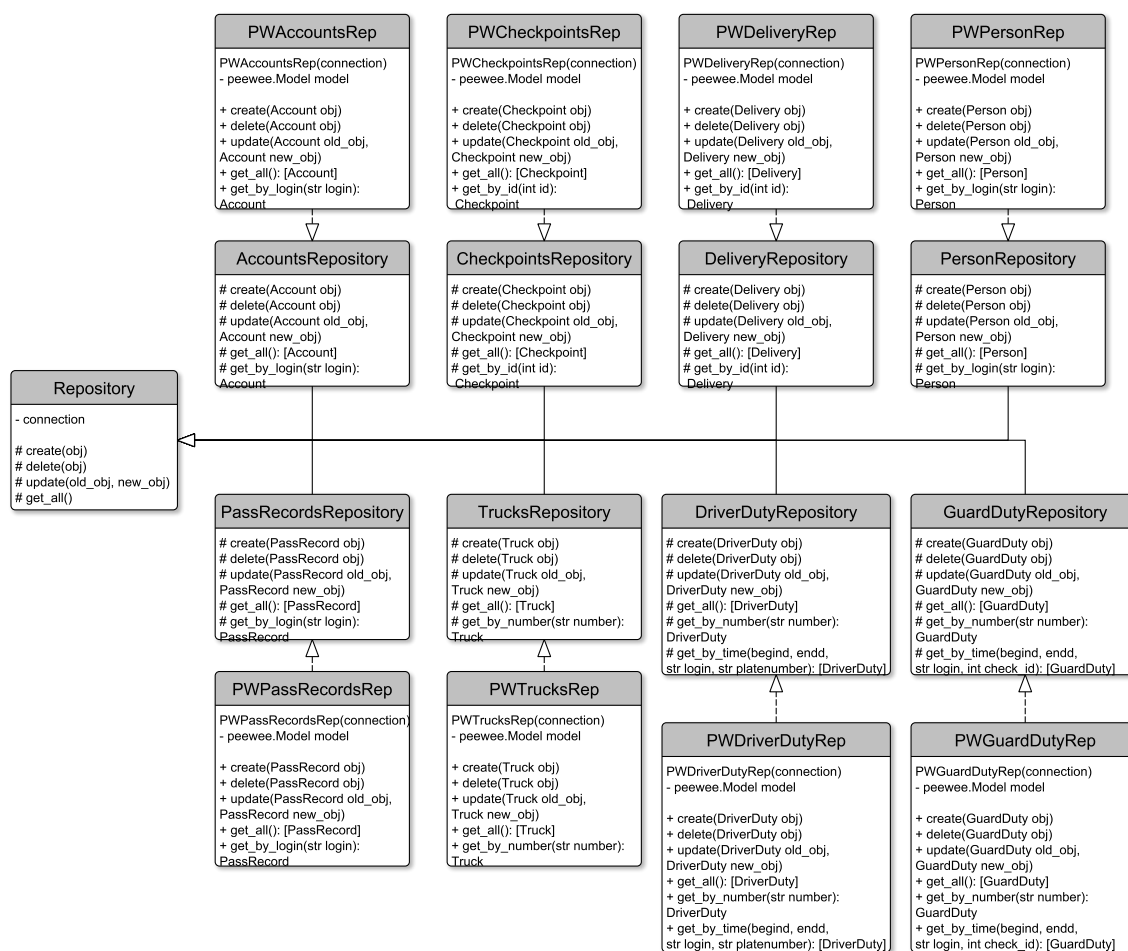


Рис. 3.1 — UML-диаграмма компонента доступа к данным



### 3.2.2. Компонент бизнес-логики

В соответствии с подходом MVC был создан компонент бизнес-логики, выполняющий основную обработку данных, UML диаграмма которого представлена на рисунке 3.2

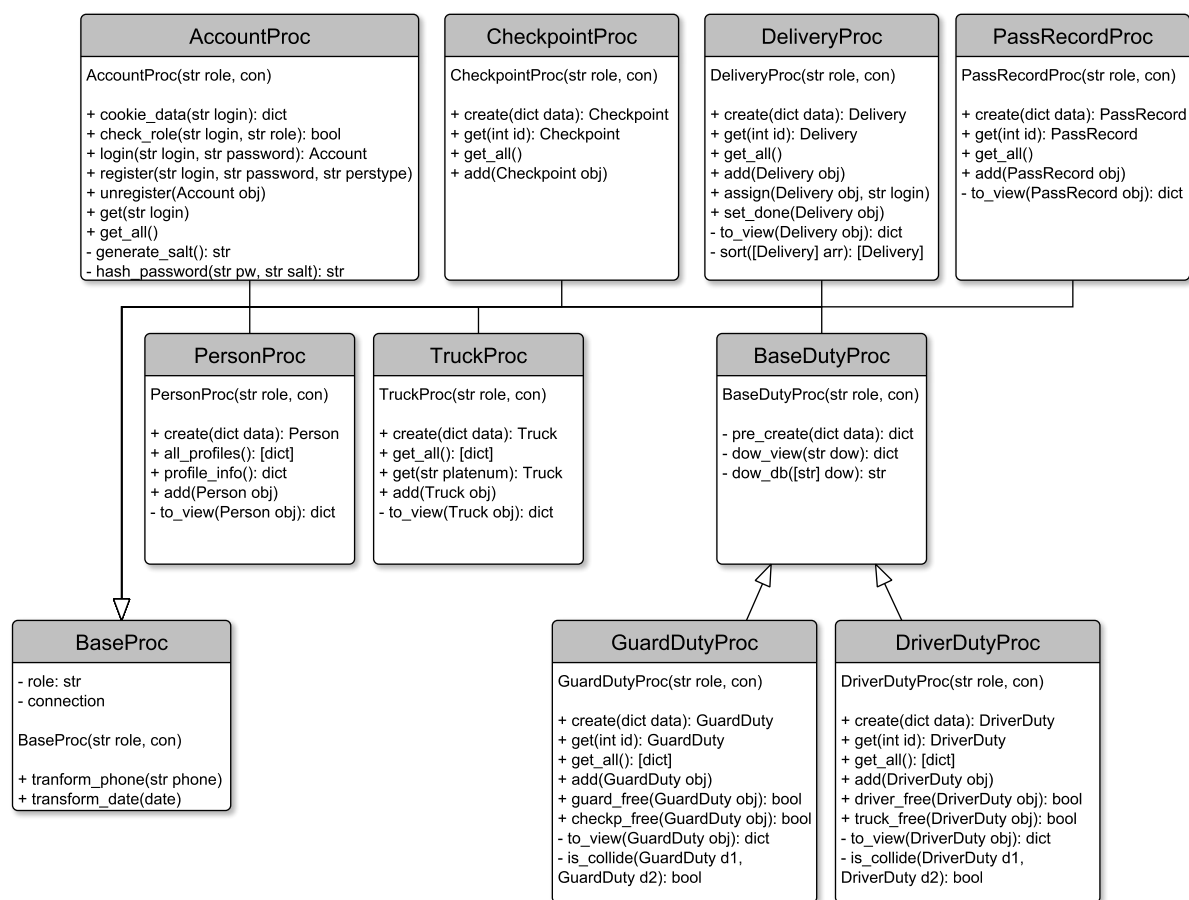


Рис. 3.2 — UML-диаграмма компонента бизнес-логики

### 3.2.3. Компонент представления

Также создан компонент представления, выполняющий отображение web-страниц в ответ на запросы пользователя, UML диаграмма которого представлена на рисунке 3.3. Помимо этого был реализован технический компонент представления, отображающий информацию в символьном виде, для возможности тестирования компонента бизнес-логики.

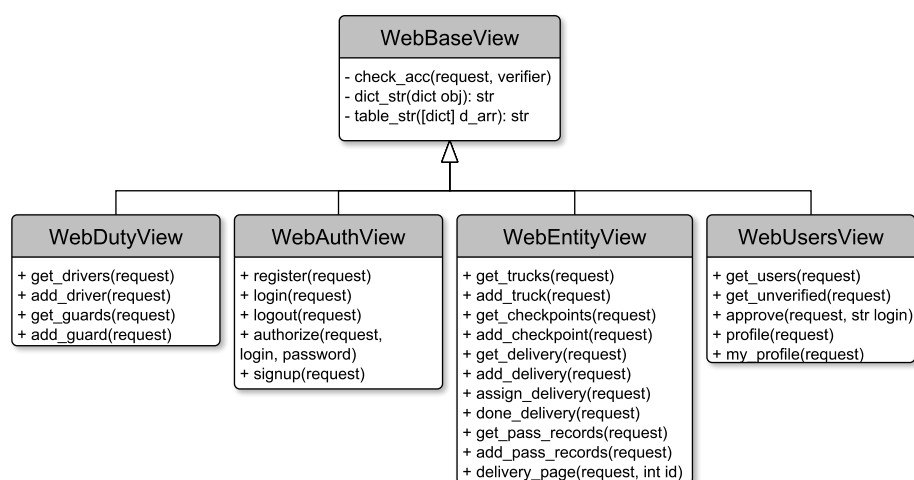


Рис. 3.3 — UML-диаграмма компонента web-представления

### 3.2.4. Диаграмма приложения

Все перечисленные компоненты можно объединить в одну UML-диаграмму всего приложения на рисунке 3.4

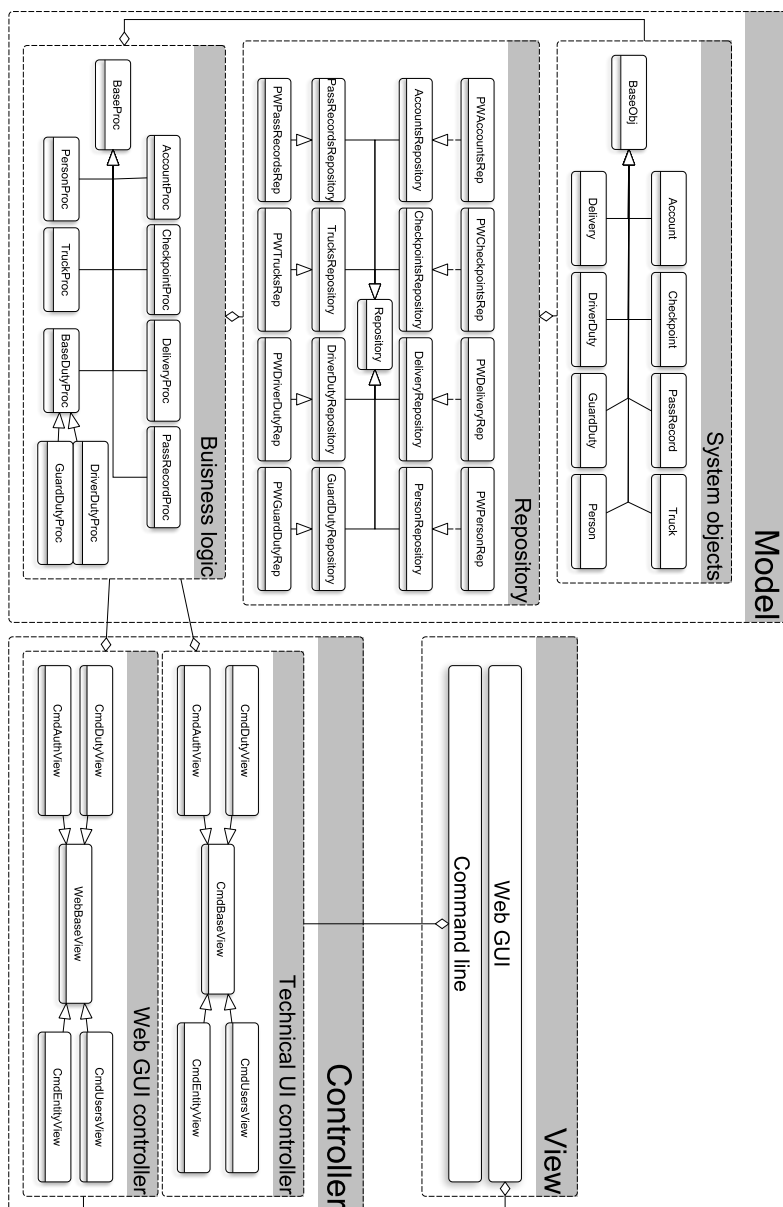


Рис. 3.4 — UML-диаграмма компонента web-представления

## 3.3. Реализация базы данных

### 3.3.1. Создание таблиц

В соответствии с ранее спроектированной моделью базы данных, были созданы 10 таблиц. Для примера в листинге 3.1 приведён код генерации таблиц, связанных с дежурствами: DutyRules, GuardDuty и DriverDuty.

Листинг 3.1 — Создание таблиц дежурств

```
1 CREATE TABLE IF NOT EXISTS DutyRules (  
2     RuleID      SERIAL      PRIMARY KEY,  
3     BeginDate   DATE        NOT NULL,  
4     EndDate     DATE,  
5     BeginTime   TIME        NOT NULL,  
6     EndTime     TIME        NOT NULL,  
7     DOW         VARCHAR(7)  NOT NULL  
8 );  
9 ALTER TABLE DutyRules ADD CONSTRAINT check_duty_dates  
10 CHECK (BeginDate <= EndDate);  
11 ALTER TABLE DutyRules ADD CONSTRAINT check_duty_times  
12 CHECK (BeginTime < EndTime);  
13 ALTER TABLE DutyRules ADD CONSTRAINT check_dow  
14 CHECK (DOW SIMILAR TO '[0|1|2|3|4|5|6]{1,7}');  
15  
16 CREATE TABLE IF NOT EXISTS GuardDuty (  
17     DutyID      SERIAL PRIMARY KEY,  
18     CheckpointID INT REFERENCES Checkpoints(CheckpointID) NOT NULL,  
19     Login        TEXT REFERENCES Person(Login) NOT NULL,  
20     RuleID       INT  REFERENCES DutyRules(RuleID) NOT NULL  
21 );  
22  
23 CREATE TABLE IF NOT EXISTS DriverDuty (  
24     DutyID      SERIAL PRIMARY KEY,  
25     PlateNumber TEXT REFERENCES Trucks(PlateNumber) NOT NULL,  
26     Login        TEXT REFERENCES Person(Login) NOT NULL,  
27     RuleID       INT  REFERENCES DutyRules(RuleID) NOT NULL  
28 );
```

В таблице DutyRules в качестве первичного ключа (сокр. ПК) указан атрибут RuleID. Созданы ограничения в значениях дат начала и завершения расписания дежурства и времени начала и заверше-

ния смены. Также реализованно ограничение содержимого атрибута DOW: в данной строке могут храниться только цифры от 0 до 6, соответствующие дням недели, в количестве от одной до семи.

В таблицах GuardDuty и DriverDuty в качестве ПК выступает атрибут DutyID. Также общим для двух таблиц является наличие внешних ключей Login и RuleID, ссылающихся на одноимённые поля в таблицах Person и DutyRules соответственно. Для GuardDuty также существует внешний ключ CheckpointID, ссылающийся на таблицу Checkpoints, для DriverDuty дополнительный внешний ключ это PlateNumber, связанный с таблицей Trucks. Все внешние ключи содержат ограничение NOT NULL, так как они являются обязательными в данных сущностях.

### 3.3.2. Ролевая модель

Для создания прав доступа выделенных ролей к различным данным была реализована ролевая модель на уровне базы данных. Выделено следующие четыре роли:

- admin - роль администратора;
- guard - роль охранника;
- driver - роль водителя;
- unverif - роль неподтверждённого сотрудника.

Код создания описанных ролей приведён в листингах 3.2–3.5

Листинг 3.2 — Релизация роли admin

```
1 CREATE ROLE admin WITH
2   LOGIN
3   SUPERUSER
4   PASSWORD 'admin';
```

### Листинг 3.3 — Релизация роли guard

```
1 CREATE ROLE guard WITH
2   LOGIN
3   PASSWORD 'guard';
4
5 GRANT SELECT, INSERT ON Passrecords TO guard;
6 GRANT SELECT ON Accounts TO guard;
7 GRANT SELECT ON Person TO guard;
8 GRANT SELECT ON GuardDutys TO guard;
9 GRANT SELECT, INSERT ON PassRecords TO guard;
10 GRANT ALL PRIVILEGES ON SEQUENCE passrecords_recordid_seq TO guard;
11 GRANT SELECT ON Checkpoints TO guard;
12 GRANT SELECT ON Trucks TO guard;
13 GRANT INSERT ON LogActions TO guard;
14
15 GRANT SELECT ON GuardDuty TO guard;
16 GRANT SELECT ON DutyRules TO guard;
```

### Листинг 3.4 — Релизация роли driver

```
1 CREATE ROLE driver WITH
2   LOGIN
3   PASSWORD 'driver';
4
5 GRANT SELECT, INSERT ON Passrecords TO driver;
6 GRANT SELECT ON Accounts TO driver;
7 GRANT SELECT ON Person TO driver;
8 GRANT SELECT, UPDATE ON Delivery TO driver;
9 GRANT SELECT ON DriverDutys TO driver;
10 GRANT INSERT ON LogActions TO driver;
11
12 GRANT SELECT ON DriverDuty TO driver;
13 GRANT SELECT ON DutyRules TO driver;
```

### Листинг 3.5 — Релизация роли unverif

```
1 CREATE ROLE unverif WITH
2   LOGIN
3   PASSWORD 'unverif';
4
5 GRANT SELECT ON Accounts TO unverif;
6 GRANT SELECT ON Person TO unverif;
7 GRANT INSERT ON LogActions TO unverif;
```

### 3.3.3. Триггеры

В базе данных реализованы триггеры соответствующие спроектированным схемам.

На листинге 3.6 приведена реализация триггера LogPassRecordIns, осуществляющего логирование информации добавления записи о проезде.

Листинг 3.6 — Реализация триггера LogPassRecordIns

```
1 CREATE OR REPLACE FUNCTION LogPassRecordIns()  
2 RETURNS TRIGGER AS $$ BEGIN  
3   INSERT INTO LogActions VALUES  
4     (current_user, NOW(), 'Created pass record №' || NEW.RecordID);  
5   RETURN NULL;  
6 END; $$ LANGUAGE plpgsql;  
7  
8 CREATE TRIGGER LogPassRecordIns BEFORE INSERT  
9 ON PassRecords FOR EACH ROW  
10 EXECUTE PROCEDURE LogPassRecordIns();
```

На листинге 3.7 приведена реализация триггера LogDeliveryUpd, осуществляющего логирование информации о добавлении заказа.

Листинг 3.7 — Реализация триггера LogDeliveryUpd

```
1 CREATE OR REPLACE FUNCTION LogDeliveryUpd()  
2 RETURNS TRIGGER AS $$ BEGIN  
3   IF NEW.Status = 'delivered' THEN  
4     INSERT INTO LogActions VALUES  
5       (current_user, NOW(), 'Set delivery №' || NEW.OrderID || ' status = delivered')  
6     ;  
7   ELSE  
8     INSERT INTO LogActions VALUES  
9       (current_user, NOW(), 'Assigned delivery №' || NEW.OrderID || ' to ' || NEW.  
10      Login);  
11   END IF;  
12 RETURN NULL;  
13 END; $$ LANGUAGE plpgsql;  
14  
15 CREATE TRIGGER LogDeliveryUpd BEFORE UPDATE  
16 ON Delivery FOR EACH ROW  
17 EXECUTE PROCEDURE LogDeliveryUpd();
```

На листинге 3.8 приведена реализация триггера BanActiveAccDel, осуществляющего запрет удаления подтверждённых сотрудников.

Листинг 3.8 — Релизация триггера BanActiveAccDel

```
1 CREATE OR REPLACE FUNCTION BanActiveAccDel()  
2 RETURNS TRIGGER AS $$ BEGIN  
3   IF (current_user = 'admin') AND (OLD.PersType NOT LIKE '~%')  
4   THEN  
5     RAISE EXCEPTION 'cannot delete active user';  
6   END IF;  
7   RETURN OLD;  
8 END; $$ LANGUAGE plpgsql;  
9  
10 CREATE TRIGGER BanActiveAccDel BEFORE DELETE  
11 ON Accounts FOR EACH ROW  
12 EXECUTE PROCEDURE BanActiveAccDel();
```

### 3.3.4. Функции

В работе программы особую сложность представляет работа с расписаниями дежурств, а также с самими дежурствами водителей. Пояснение реализованных функций для каждой таблицы приведено дальше. Все листинге расположены в приложении А, стр. 42.

- Таблица DutyRules. Требуется осуществлять выборку дежурств в определённом диапазоне дат, нахождение активных дежурств в определённый момент времени, в том числе и в текущий. Код функций приведён в листинге 4.1.
- Таблицы DriverRules и GuardRules. Для дежурств требуется осуществлять поиск по логину и номеру КПП в случае охранника или номеру машины в случае водителя. Также требуется учитывать диапазон дат, в котором осуществляется поиск или определённый момент времени. Код описанных функций приведён в листингах 4.2 и 4.3 для таблиц DriverRules и GuardRules соответственно.



### 3.4. Интерфейс приложения

Для авторизованного пользователя в заголовке каждой страницы содержится панель навигации, предоставляющая возможность перейти на все страницы, функционально соответствующие его роли. Данная панель изображена на примере администратора на рисунке 3.5. Некоторые пункты меню являются выпадающими, их можно открыть наведением мыши. Также в левом углу панели присутствует элемент информационного или ошибочного сообщения.

На рисунке 3.5 изображена страница профиля сотрудника, на котором он может посмотреть всю свою личную информацию. Просматривать свою страницу может пользователь с любой ролью. Для водителя на странице профиля также отображается история его заказов (на рисунке 3.6).

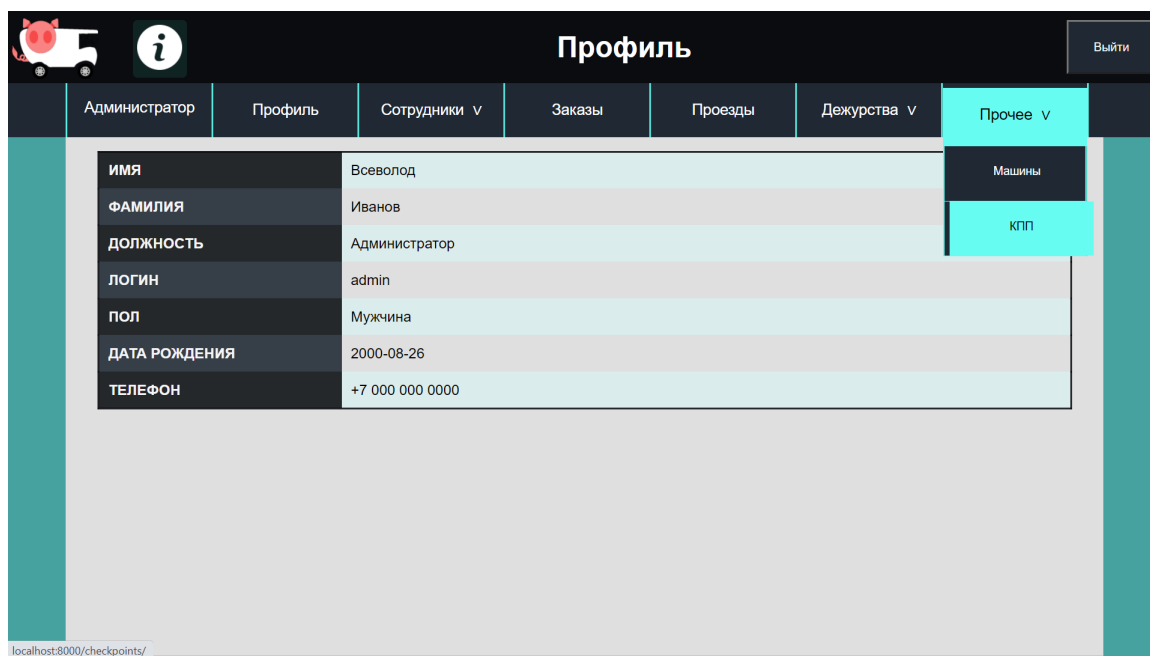


Рис. 3.5 — Страница профиля администратора

ТЕЛЕФОН

+7 999 123 4124

История моих заказов

НОМЕР	СТАТУС	АДРЕС	ДОСТАВЛЕН
5	В доставке	проезд Кукушкина, дом 21	-
3	Доставлен	ул. Хихишина, дом 123, корпус 4	03.06.2021 14:41:37
2	Доставлен	проезд Колотушкина, дом 21	01.06.2021 17:59:00

Рис. 3.6 — История заказов (для профиля водителя)

На рисунке 3.7 представлена страница просмотра всех сотрудников. С помощью клика на ряд таблицы можно перейти на страницу конкретного сотрудника, описанная выше. Данный функционал доступен только для администратора.


		<b>Сотрудники</b>					Выйти
Администратор	Профиль	Сотрудники v	Заказы	Проезды	Дежурства v	Прочее v	
ДОЛЖНОСТЬ	ИМЯ	ФАМИЛИЯ	ЛОГИН				
Администратор	Иванов	Всеволод	admin				
Водитель	Гослинг	Райан	driver				
Водитель	Техников	Павел	driver2				
Охранник	Золотова	Роза	guard				
Охранник	Боросов	Филипп	guard2002				

Рис. 3.7 — Страница просмотра всех сотрудников

На рисунке 3.8 представленна страница заявок на регистра-

цию. Заявки можно принять или отклонить кликом на соотв. кнопки. Данная страница также доступна только администратору.

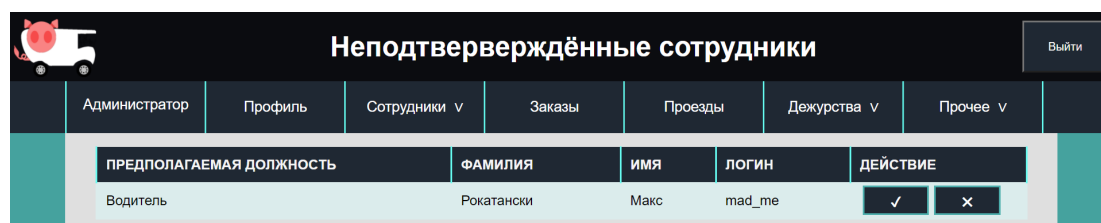


Рис. 3.8 — Страница просмотра заявок регистрации

На рисунке 3.9 изображена страница просмотра всех заказов. По клику на строку заказа можно перейти на страницу подробной информации о заказе. Просмотр этой страницы доступен для водителей и администраторов. Для последних также существует возможность создать заказ в всплывающем окне.

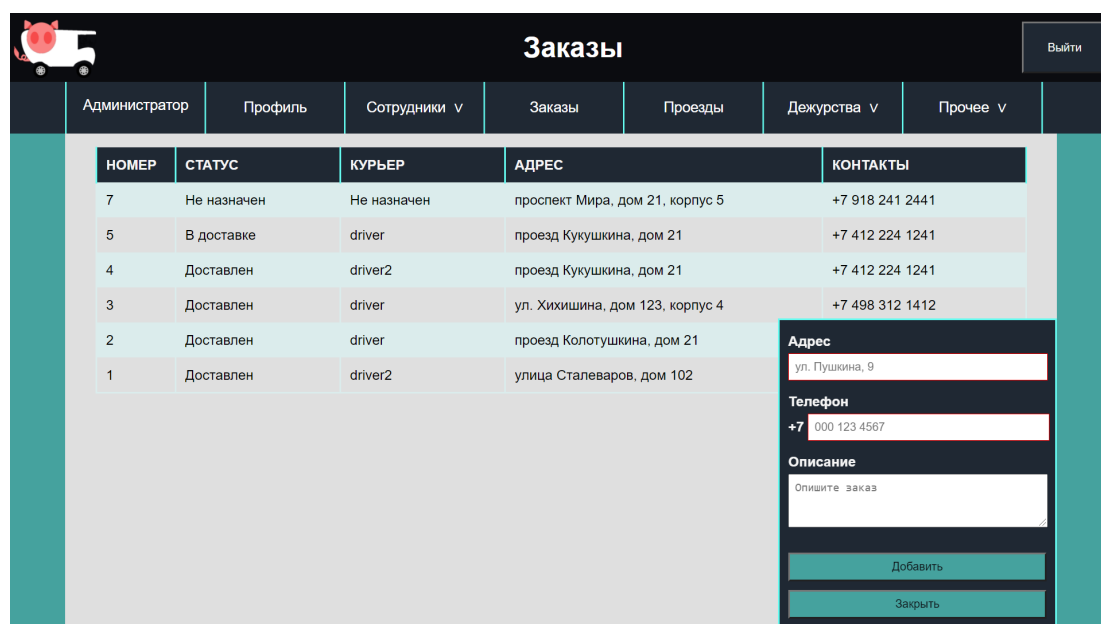


Рис. 3.9 — Страница просмотра и создания заказов

Вышеупомянутая страница подробной информации о заказе приведена на рисунке 3.10. Страница доступна для водителей и администраторов. Обе роли могут поручить заказ водителю и завершить заказ. Администратор назначает водителя указанием его логина в всплывающем окне. Водитель способен назначить заказ только себе нажатием кнопки «Выбрать заказ».

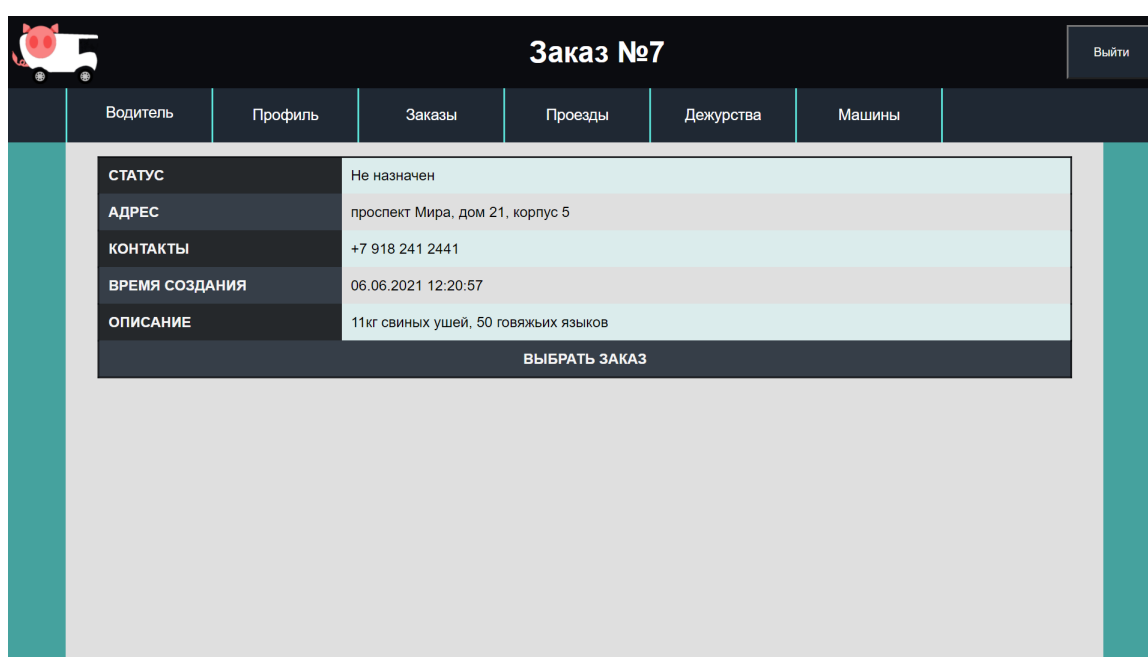
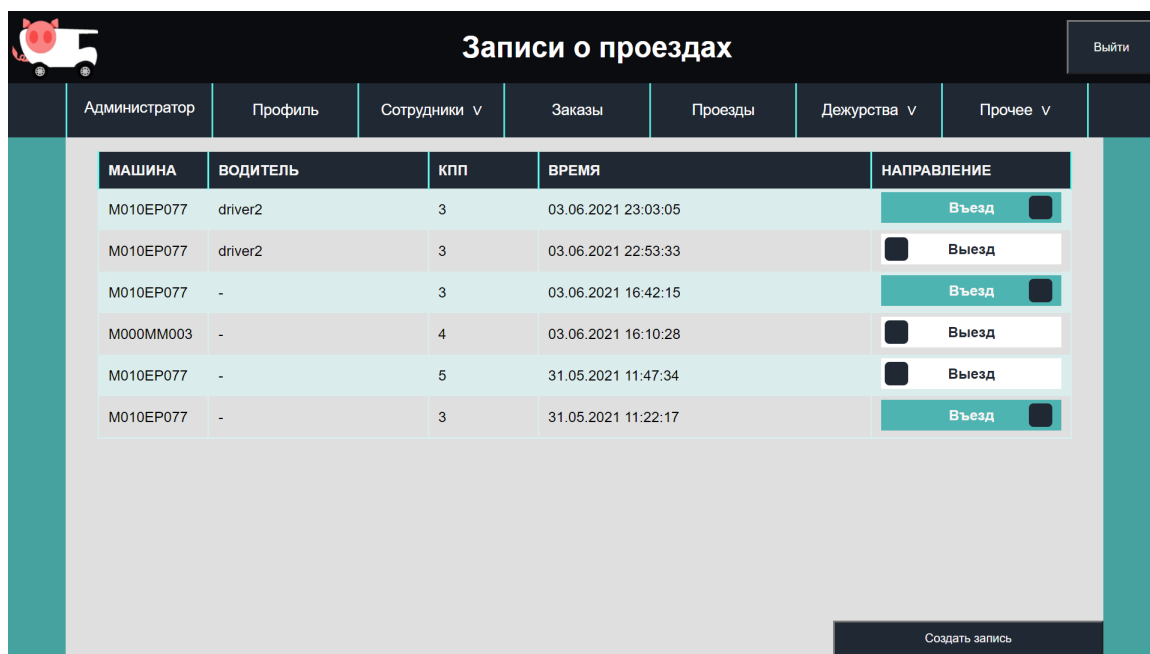


Рис. 3.10 — Страница просмотра и выбора заказа (для роли водителя)

На рисунке 3.11 изображена страница всех записей проездов. Для администратора на данной странице выводятся все записи в обратном хронологическом порядке и окно для добавления новой записи. Для охранника на данной странице видны записи о проезде в пределах его текущего дежурства (т.е. через определённое КПП в ограниченный временной диапазон). Охранник также может доба-

вить запись о проезде, но в отличие от администратора ему не нужно указывать номер КПП.



МАШИНА	ВОДИТЕЛЬ	КПП	ВРЕМЯ	НАПРАВЛЕНИЕ
M010EP077	driver2	3	03.06.2021 23:03:05	Въезд <input checked="" type="checkbox"/>
M010EP077	driver2	3	03.06.2021 22:53:33	<input checked="" type="checkbox"/> Выезд
M010EP077	-	3	03.06.2021 16:42:15	Въезд <input checked="" type="checkbox"/>
M000MM003	-	4	03.06.2021 16:10:28	<input checked="" type="checkbox"/> Выезд
M010EP077	-	5	31.05.2021 11:47:34	<input checked="" type="checkbox"/> Выезд
M010EP077	-	3	31.05.2021 11:22:17	Въезд <input checked="" type="checkbox"/>

Рис. 3.11 — Страница просмотра и регистрации записей о проездах

На следующем рисунке 3.12 представлена страница дежурств водителей. Доступ к ней имеют все подтверждённые сотрудники (охранники могут просматривать аналогичную страницу для их роли). Администратору выводятся все дежурства, отдельно отмечаются те, которые идут в данный момент. Также он может назначить новое дежурство в всплывающем окне. Для этого нужно указать логин сотрудника, номер КПП или машины (в зависимости от роли) и время его дежурства (задаётся диапазоном дат, времени и днями недели).

Водители и охранники на этой странице могут видеть все свои дежурства и информацию о ближайшем из них.

Также администратору доступны страницы просмотра и реги-

Дежурства водителей

Выйти

Администратор

Профиль

Сотрудники v

Заказы

Проезды

Дежурства v

Прочее v

МАШИНА	ВОДИТЕЛЬ	ПЕРИОД	ВРЕМЯ	ДНИ РАБОТЫ
Текущие дежурства				
M010EP077	driver	с 30.05.2021 по 30.06.2021	09:00 - 18:00	
Все дежурства				
M010EP077	driver2	с 09.07.2021 по 21.07.2021	09:00 - 18:00	
M000MM003	driver	с 23.08.2021	09:00 - 18:00	
M010EP077	driver2	с 03.06.2021 по 11.06.2021	17:30 - 23:55	
M010EP077	driver	с 30.05.2021 по 30.06.2021	09:00 - 18:00	

Номер машины

A0123BE777

Водитель

Логин

Период дежурства

06.06.2021 - 07.07.2021

Бессрочно

Время дежурства

09:00 - 18:00

Рабочие дни

ПН

ВТ

СР

ЧТ

ПТ

СБ

ВС

Добавить

Закрыть

Рис. 3.12 — Страница просмотра и назначения дежурств водителей

страции машин и КПП. На них представлен полный список данных объектов и всплывающее окно с формой добавления нового объекта.

Для неавторизованного пользователя доступны страницы входа в аккаунт и регистрации.

## Вывод

Результатом технологической части стал выбор средств программной реализации, реализация и описание структуры базы данных и приложения, визуальная демонстрация интерфейса приложения.

## Заключение

В ходе курсовой работы была достигнута поставленная цель: были разработаны база данных и web-приложение для транспортной системы завода. Выполнены все задачи проекта.

В ходе работы была формализована задача, определён и реализован необходимый функционал по взаимодействию с данными. Проведён анализ и выбор реляционной СУБД, с помощью которой была спроектирована и реализована база данных. Также были выбраны и использованы средства программной реализации web-приложения для предоставления графического интерфейса для работы с данными.

Реализованное приложение позволяет потенциальным сотрудникам завода удобно просматривать и изменять компоненты транспортной системы, в соответствии со своими обязанностями.

## Список литературы

1. Окулов С.А., Варзунов А.В. Информационные технологии в малом бизнесе // Актуальные проблемы гуманитарных и естественных наук. 2015. №1-1.
2. Коннолли Т., Бегг К. Базы данных: проектирование, реализация, сопровождение. Теория и практика, 3-е изд. : Пер. с англ. : Уч. пос. –М.: Изд. дом "Вильямс 2003. –1440 с.
3. Дейт К. Дж. Введение в системы баз данных. — 8-е изд. — М.: «Вильямс», 2006.
4. Чухраев И.В., Жукова И.В. Оптимизация работы с информацией в базах данных // Инновационная наука. 2016. №4-3 (16).
5. Информационные системы и базы данных: организация и проектирование: учеб. пособие. — СПб.: БХВ-Петербург, 2009. — 528 с.: ил. — (Учебная литература для вузов)
6. Тортика Алексей Сергеевич, Ершов Алексей Сергеевич ОБЗОР И СРАВНИТЕЛЬНЫЙ АНАЛИЗ СОВРЕМЕННЫХ СИСТЕМ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ // Вестник СГТУ. 2020. №4 (87)
7. Драч В.Е., Родионов А.В., Чухраева А.И. Выбор системы управления базами данных для информационной системы промышленного предприятия // Электромагнитные волны и электронные системы. 2018. Т. 23. № 3. С. 71-80.
8. Документация Python 3.9.5 [Электронный ресурс]. Режим доступа: <https://docs.python.org/3/>, свободный (дата обращения: 01.05.2021).



9. Документация ORM peewee [Электронный ресурс]. Режим доступа: <http://docs.peewee-orm.com/en/latest>, свободный (дата обращения: 01.05.2021).
10. Документация Web-фреймворка Django [Электронный ресурс]. Режим доступа: <https://docs.djangoproject.com/en/3.2/>, свободный (дата обращения: 01.05.2021).

## Приложение А. Код хранимых функций

Листинг 4.1 — Релизация функций для таблицы DutyDules

```
1 CREATE OR REPLACE FUNCTION DutyByRange(BeginD DATE, EndD DATE)
2 RETURNS TABLE (LIKE DutyRules) AS $$ BEGIN
3 RETURN QUERY (
4     SELECT *
5     FROM DutyRules
6     WHERE (
7         ((EndD IS NULL) AND
8          (
9              (EndDate IS NULL) OR
10             (EndDate >= BeginD)
11         )) OR
12         ((EndD IS NOT NULL) AND
13          (
14              ((EndDate IS NULL) AND (BeginDate <= EndD)) OR
15              (NOT (EndDate IS NULL) AND
16               (
17                   (BeginDate BETWEEN BeginD AND EndD) OR
18                   (EndDate BETWEEN BeginD AND EndD) OR
19                   (BeginD BETWEEN BeginDate AND EndDate)
20               )
21             )
22         ))
23     )
24 );
25 END; $$ LANGUAGE plpgsql;
26
27 CREATE OR REPLACE FUNCTION DutyAtMoment(Moment TIMESTAMP)
28 RETURNS TABLE (LIKE DutyRules)
29 AS
30 $$
31 DECLARE
32     NowDate DATE;
33     NowTime TIME;
34     NowDow TEXT;
35 BEGIN
36     SELECT *
37     FROM CAST(Moment AS DATE)
38     INTO NowDate;
39
40     SELECT *
41     FROM CAST(Moment AS TIME)
42     INTO NowTime;
43
44     SELECT CAST(Date_Part - 1 AS TEXT)
45     FROM EXTRACT(isodow FROM NowDate)
46     INTO NowDow;
47
48 RETURN QUERY (
49     SELECT *
```

```

50 FROM DutyByRange(NowDate, NowDate)
51 WHERE (
52     (POSITION(NowDow in Dow) != 0) AND
53     (NowTime BETWEEN BeginTime AND EndTime)
54 )
55 );
56 END; $$ LANGUAGE plpgsql;
57
58
59 CREATE OR REPLACE FUNCTION CurrentDuty()
60 RETURNS TABLE (LIKE DutyRules) AS
61 $$ BEGIN
62 RETURN QUERY (
63     SELECT *
64     FROM DutyAtMoment(CAST(NOW() AS TIMESTAMPTZ))
65 );
66 END; $$ LANGUAGE plpgsql;

```

#### Листинг 4.2 — Релизация функций для таблицы DriverDuty

```

1 CREATE TYPE DDutyJoin AS
2 (
3     DutyID          INT,
4     PlateNumber     TEXT,
5     Login           TEXT,
6
7     RuleID          INT,
8     BeginDate       DATE,
9     EndDate         DATE,
10    BeginTime       TIME,
11    EndTime         TIME,
12    DOW             VARCHAR(7)
13 );
14
15 CREATE OR REPLACE FUNCTION MomentDDuty(Moment TIMESTAMPTZ, QLogin TEXT, PlateN TEXT)
16 RETURNS SETOF DDutyJoin AS $$ BEGIN
17 RETURN QUERY (
18     SELECT DutyID, PlateNumber, Login, t1.RuleID,
19            BeginDate, EndDate, BeginTime, EndTime, DOW
20     FROM DriverDuty JOIN DutyAtMoment(Moment) AS "t1" ON DriverDuty.RuleID = t1.RuleID
21     WHERE (
22         ((QLogin IS NULL) OR (QLogin = Login)) AND
23         ((PlateN IS NULL) OR (PlateN = PlateNumber))
24     )
25 );
26 END;
27 $$
28 LANGUAGE plpgsql;
29

```

```

30 CREATE OR REPLACE FUNCTION CurrentDDuty(QLogin TEXT, PlateN TEXT)
31 RETURNS SETOF DDutyJoin AS $$ BEGIN
32 RETURN QUERY (
33     SELECT DutyID, PlateNumber, Login, t1.RuleID,
34           BeginDate, EndDate, BeginTime, EndTime, DOW
35     FROM DriverDuty JOIN CurrentDuty() AS "t1" ON DriverDuty.RuleID = t1.RuleID
36     WHERE (
37         ((QLogin IS NULL) OR (QLogin = Login)) AND
38         ((PlateN IS NULL) OR (PlateN = PlateNumber))
39     )
40 );
41 END; $$ LANGUAGE plpgsql;
42
43 CREATE OR REPLACE FUNCTION GetDDuty(BeginD DATE, EndD DATE, QLogin TEXT, PlateN TEXT
44 )
45 RETURNS SETOF DDutyJoin AS $$ BEGIN
46 RETURN QUERY (
47     SELECT DutyID, PlateNumber, Login, t1.RuleID,
48           BeginDate, EndDate, BeginTime, EndTime, DOW
49     FROM DriverDuty JOIN DutyByRange(BeginD, EndD) AS "t1" ON DriverDuty.RuleID = t1.
50           RuleID
51     WHERE (
52         ((QLogin IS NULL) OR (QLogin = Login)) AND
53         ((PlateN IS NULL) OR (PlateN = PlateNumber))
54     )
55 );
56 END; $$ LANGUAGE plpgsql;

```

#### Листинг 4.3 — Релизация функций для таблицы GuardDuty

```

1 CREATE TYPE GDutyJoin AS
2 (
3     DutyID          INT,
4     CheckpointID    INT,
5     Login           TEXT,
6
7     RuleID          INT,
8     BeginDate       DATE,
9     EndDate         DATE,
10    BeginTime       TIME,
11    EndTime         TIME,
12    DOW             VARCHAR(7)
13 );
14
15 CREATE OR REPLACE FUNCTION MomentGDuty(Moment TIMESTAMP, QLogin TEXT, CheckID INT)
16 RETURNS SETOF GDutyJoin AS
17 $$
18 BEGIN
19 RETURN QUERY (

```

```

20  SELECT DutyID , CheckpointID , Login , t1.RuleID ,
21  BeginDate , EndDate , BeginTime , EndTime , DOW
22  FROM GuardDuty JOIN DutyAtMoment(Moment) AS "t1" ON GuardDuty.RuleID = t1.RuleID
23  WHERE (
24      ((QLogin IS NULL) OR (QLogin = Login)) AND
25      ((CheckID IS NULL) OR (CheckID = CheckpointID))
26  )
27  );
28  END;
29  $$
30  LANGUAGE plpgsql;
31
32  CREATE OR REPLACE FUNCTION CurrentGDuty(QLogin TEXT, CheckID INT)
33  RETURNS SETOF GDutyJoin AS
34  $$
35  BEGIN
36  RETURN QUERY (
37      SELECT DutyID , CheckpointID , Login , t1.RuleID ,
38      BeginDate , EndDate , BeginTime , EndTime , DOW
39      FROM GuardDuty JOIN CurrentDuty() AS "t1" ON GuardDuty.RuleID = t1.RuleID
40      WHERE (
41          ((QLogin IS NULL) OR (QLogin = Login)) AND
42          ((CheckID IS NULL) OR (CheckID = CheckpointID))
43      )
44  );
45  END;
46  $$
47  LANGUAGE plpgsql;
48
49  CREATE OR REPLACE FUNCTION GetGDuty(BeginD DATE, EndD DATE, QLogin TEXT, CheckID INT
50  )
51  RETURNS SETOF GDutyJoin AS
52  $$
53  BEGIN
54  RETURN QUERY (
55      SELECT DutyID , CheckpointID , Login , t1.RuleID ,
56      BeginDate , EndDate , BeginTime , EndTime , DOW
57      FROM GuardDuty JOIN DutyByRange(BeginD , EndD) AS "t1" ON GuardDuty.RuleID = t1.
58      RuleID
59      WHERE (
60          ((QLogin IS NULL) OR (QLogin = Login)) AND
61          ((CheckID IS NULL) OR (CheckID = CheckpointID))
62      )
63  );
64  END;
65  $$
66  LANGUAGE plpgsql;

```