

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Мобильное приложение для регистрации
сотрудников компании перевозок в отеле**

Студент _____ Овчаренко Владимир Павлович
(подпись, дата) (ФИО полностью)

Руководитель _____ Макарычев П.П.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»

Заведующий кафедрой

_____ А.Ю. Козлов

«__» _____ 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Овчаренко Владимир Павлович гр. 19ВП2 факультета вычислительной
<Фамилия, имя, отчество полностью>

техники

2. Тема работы Мобильное приложение для регистрации сотрудников компании
перевозок в отеле

Тема утверждена приказом ПГУ № _____ от "___" апреля 2023 года

2. Руководитель работы Макарычев Петр Петрович

3. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: мобильное приложение для сотрудников придорожной инфраструк-
туры, оптимизирующее и ускоряющее процесс обработки клиентов компаний

2) Среда выполнения: ЭВМ с операционной системой Windows 10, смартфон с опе-
рационной системой Adnroid 8 версии и выше.

Языковые средства разработки: Kotlin, xml, Python, SQL.

Инструментальные средства: Android Studio, Postgre SQL, PyCharm Community,
Draw.io

Технологии разработки: ООП, шаблоны проектирования, Retrofit2, MlKit,
NavGraph, Django, REST API.

3) Основные функции приложения:

а) регистрация прибывшего клиента;

б) отметка выезжающего клиента;

в) просмотр истории регистраций;

- г) авторизация в приложение;
д) управление администратора.

4. Перечень подлежащих разработке вопросов:

- 1) Анализ предметной области и постановка задачи;
2) Проектирование;
3) Реализация;
4) Контроль качества программного обеспечения;
5) План разработки бюджета;
6) Оформление пояснительной записки.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	13.04.2023 – 26.04.2023	
2	Проектирование программных средств	20%	27.04.2023 – 10.05.2023	
3	Реализация программных средств	30%	11.05.2023 – 30.05.2023	
4	Тестирование программных средств	20%	31.05.2023 – 03.06.2023	
5	Оформление пояснительной записки	20%	04.06.2023 – 18.06.2023	

Дата выдачи задания «22» декабря 2022 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 107 листов, 33 рисунков, 14 таблиц, 13 использованных источников и 3 приложения.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, РЕГИСТРАЦИЯ КЛИЕНТОВ, РАСПОЗНАВАНИЕ ТЕКСТА, КАМЕРА, ANDROID, КЛИЕНТ, СЕРВЕР, REST API, RETROFIT, MLKIT, DJANGO, СУБД, POSTGRESQL, СЕРВЕР БАЗЫ ДАННЫХ, UML.

Объект исследования – средства автоматизации рабочих процессов работников отелей и другой придорожной инфраструктуры.

Цель выпускной квалификационной работы – разработать мобильное приложение для сотрудников отелей и придорожной инфраструктуры, позволяющее быстро и эффективно производить регистрацию клиентов из компаний, с которыми заключён договор обслуживания.

Технологии разработки – разработка сервера проводилась на языке программирования Python с использованием Django REST Framework в среде разработки PyCharm Community Edition. Разработка клиентского приложения проводилась на языке Kotlin с использованием фреймворков Retrofit2 для реализации REST взаимодействия и MLKit для распознавания текста. Для реализации базы данных и взаимодействия с ней использовалась СУБД PostgreSQL.

Среда исполнения – смартфон с операционной системой Android версии 8 и выше.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная часть мобильного приложения регистрации сотрудников. Все требования к разработке приложения соблюдены.

					ПГУ 09.03.04– 8БР192.15 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Мобильное приложение для регистрации сотрудников компании перевозок в отеле. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Овчаренко В.П.						
Пров.		Макарычев П.П.					4	107
						Группа 19ВП2		
Н. Контр.		Такташкин Д.В						
Утв.		Козлов А.Ю.						

Содержание

Введение	7
1 Анализ области и выделение требований.....	9
1.1 Анализ предметной области и постановка задачи	9
1.2 Анализ программного обеспечения аналогичного назначения.....	11
1.3 Анализ функциональных требований	17
1.4 Анализ нефункциональных требований	20
2 Проектирование мобильного приложения регистрации сотрудников компании перевозок в отеле	21
2.1 Архитектура приложения	21
2.2 Проектирование структур данных	22
2.2.1 Концептуальная модель базы данных приложения	23
2.2.2 Логическая модель базы данных и описание атрибутов	25
2.2.3 Физическая модель базы данных	29
2.3 Проектирование программных средств	30
2.4 Проектирование диаграммы последовательности	32
2.5 Проектирование пользовательского интерфейса	33
3 Реализация приложения.....	42
3.1 Реализация серверной части приложения	42
3.2 Реализация базы данных.....	45
3.3 Реализация клиентской части приложения	45
4 Контроль качества программного обеспечения	48
4.1 Функциональное тестирование	48
4.2 Тестирование серверной части приложения	48
4.3 Тестирование клиентской части приложения	49
4.4 Анализ качества кода.....	50
4.4.1 Измерение устойчивости и абстрактности кода	50
4.4.2 Главная последовательность	52
4.5 Планирование разработки и оценка бюджета	54

Заключение	57
Список использованных источников	58
Приложение А. Глоссарий.....	60
Приложение Б. Код программы.....	63
Приложение В. Результаты тестирования	101

Введение

Мобильные приложения – программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах [1], в современном мире глубоко вошли в жизнь людей. Они используются во многих сферах деятельности, так как позволяют получить удобный, быстрый и независимый доступ к необходимым функциям практически в любом месте. Поэтому всё больше людей для решения задач отдают предпочтение мобильным приложениям.

Исходя из этого, в последнее время многие компании в ситуациях, когда возможно использовать мобильное приложение, используют именно их.

При анализе возможности внедрения такого приложения в систему отношений компаний по грузоперевозкам и владельцев мест отдыха и стоянки выявлена актуальность проблемы нехватки подобного решения на рынке. Такие компании часто договариваются с владельцами мест отдыха для своих работников напрямую, чтобы оптимизировать работу, сэкономить материальные средства, организовать расходы и уменьшить количество разовых оплат. Поэтому, у владельцев мест отдыха возникает достаточно большой объём данных, возможно по каждой из фирм, а также потребность в реализации быстрого и точного способа регистрации сотрудников таких грузовых компаний на стоянке или месте отдыха. Для данной задачи наиболее подходящим решением является реализация мобильного приложения для сотрудника места отдыха вместе с базой данных по сотрудникам и компаниям, а регистрацию быстрее всего возможно произвести по номерам прибывающих автомобилей компаний, которые возможно быстро и точно считать с помощью камеры смартфона.

Анализ рынка подобных приложений говорит о том, что в текущий момент подобные решения есть лишь у малого числа частных компаний-посредников и они предоставляются только при обслуживании их клиентов, что вызывает в случае желания автоматизации процесса ограничения для владельцев придорожной инфраструктуры в клиентах и возможностях.

Таким образом, можно сделать вывод, что существует потребность в приложении для регистрации сотрудников компаний перевозок в том или ином отеле, реализованном в виде мобильного приложения, позволяющего быстро и точно считать информацию и зарегистрировать клиента.

Целью выпускной квалификационной работы является разработка мобильного приложения для регистрации сотрудников компании перевозок в отеле. Разработка включает в себя анализ требований, проектирование, реализацию и тестирование программного обеспечения.

1 Анализ области и выделение требований

1.1 Анализ предметной области и постановка задачи

На сегодняшний день существует множество компаний грузоперевозок, пользующихся услугами мест отдыха и стоянки для своих сотрудников. Для экономии средств и организации процесса использования услуг, предоставляемых местами отдыха, а также уменьшения количества разовых оплат, компании часто заключают договор о предоставлении услуг (документ «Акт об указании услуг»), из-за чего возникает потребность в быстром и точном способе регистрации сотрудников и отправки чеков для оплаты данным компаниям.

У владельцев таких мест отдыха может быть большое количество договоров, так как компании могут быть небольшими. Из-за этого, может возникать большой объём документов и информации по каждой компании в отдельности и всем вместе. Также, при достаточно большом потоке клиентов, возрастает нагрузка на работника регистрации отеля и повышается риск ошибок и неточностей. Для минимизации ошибок и увеличения скорости обработки заявок, а также уменьшения работы с данными компаний, необходимо разработать мобильное приложение, позволяющее производить быструю и автоматическую в некоторых аспектах регистрацию клиентов.

Для того, чтобы определить требования к такому мобильному приложению, нужно провести анализ наиболее вероятных функций и вариантов использования.

Для эффективной, быстрой и точной регистрации сотрудников компании, необходимо знать основную информацию для регистрации, а именно:

- номер автомобиля;
- фамилия, имя, отчество клиента;
- документы клиента;
- наименование фирмы;

- необходимый перечень услуг (стоянка или ночёвка, душ, питание и т. д.);
- количество предоставляемых услуг (дни, часы, приёмы пищи и др.);
- дата и время заезда и выезда автомобиля.

Для оптимизации времени обработки и уменьшения вероятности ошибок проверки этих данных при регистрации, хранение данных должно производиться с помощью базы данных.

Наиболее быстрый и надёжный способ провести поиск по базе данных информации о прибывшем сотруднике компании в данной ситуации – сверить номер автомобиля, а далее – полученные по номеру автомобиля данные о сотруднике с его фактическими данными.

Чтобы исключить ошибки при проверке, можно добавить дополнительный модуль считывания данных с помощью камеры, благодаря чему не потребуется их вводить вручную. Следовательно, основная функция, выполняемая приложением, это регистрация прибывшего клиента, начинающаяся с поиска в базе данных данного сотрудника по номеру автомобиля, который выполняется вводом номера вручную или с помощью камеры телефона и сверки полученных данных с документом сотрудника. Далее, отмечаются предоставляемые услуги, после чего вахтёр (сторож) ставит дату заезда и отмечает автомобиль, а данные по предоставляемым услугам отправляются на сервер для дальнейшей сборки и отправки для оплаты компании. После отбытия автомобиля, вахтёр (сторож) ставит в приложении отметку об выезде.

Для отслеживания работы регистрирующего работника и настройки приложения отдельным видом пользователя (администратором), должна быть реализована функция входа в приложение по логину и паролю или возможность настройки по вводу определённого ключа. Также, в личном кабинете пользователя необходимо реализовать возможность просмотра

локальной истории регистраций для сверки зарегистрированных клиентов и добавить краткую инструкцию по использованию приложения.

Необходимо добавить возможность настройки меню регистрации, возможность редактирования предоставляемых компаниям услуг на случай, если в текущий момент место отдыха не имеет возможности предоставлять ту или иную услугу или если необходимо изменить предоставляемые услуги согласно заключенному договору.

Компании грузоперевозок чаще всего оплачивают только место стоянки, которое возможно отметить прямо на месте, не отходя от автомобиля, поэтому мобильное приложение хорошо подходит для реализации способа регистрации сотрудников компании, позволяя проводить её удобно как для сотрудников компании грузоперевозок, так и для работников места отдыха.

Так как подобные приложения чаще всего являются собственностью организаций и находятся в закрытом доступе, будет правильным решением сделать данное приложение бесплатным для использования, позволив многим компаниям получить удобный способ организации работы. Также нужно оставить исходный код открытым для возможной доработки приложения под те или иные условия.

Исходя из описанного выше, для реализации обращения приложения к базе данных, делается вывод о создании программного продукта на основе архитектуры из серверного приложения, управляемой им базы данных и клиентского мобильного приложения.

1.2 Анализ программного обеспечения аналогичного назначения

В результате анализа программных продуктов, используемых для поиска и сверки информации, отправки данных, сканирования информации с камеры и прочих требуемых для реализации функций, определены приложения, которые обладают приближенными аналогами разрабатываемого мобильного приложения.

Для проведения анализа в качестве критериев сравнения выбраны следующие ключевые возможности программных продуктов:

а) функциональные возможности:

- 1) поиск информации для сверки и регистрация сотрудника;
- 2) выбор предоставляемых услуг;
- 3) просмотр истории регистрации;
- 4) просмотр справочной информации;
- 5) настройка предоставляемых услуг.

б) нефункциональные возможности:

- 1) бесплатность продукта;
- 2) открытость программного кода.

Для проведения сравнительного анализа аналогичных программных продуктов на основе исследования бизнес-процессов, возникающих при использовании приложений, были определены коэффициенты значимости для данных критериев. Полученные коэффициенты значимости представлены в таблице 1.

Таблица 1 – Критерии оценки приложений регистрации клиентов

№	Наименование	Коэффициент значимости
	Функциональные возможности	
1	Поиск информации для сверки и регистрация сотрудника	1
2	Выбор предоставляемых услуг	0.75
3	Просмотр истории регистрации	0.75
4	Просмотр справочной информация	1
5	Настройка предоставляемых услуг	0.75
	Нефункциональные требования	
1	Бесплатность продукта	1
2	Открытость программного кода	0.75

Для оценки описанных выше критериев используется пятибалльная шкала соответствия, от 0 – «полное несоответствие», до 4 – «полное

соответствие» требованиям. Описание и характеристика оценки описанных критериев представлена в таблице 2.

Таблица 2 – Шкала баллов

Балл	Характеристика оценки
0	функция (качество) отсутствует, реализовать ее невозможно
1	функция (качество) отсутствует, но ее возможно реализовать путем доработки системы
2	похожие, но отличные от требуемых, функции или качества системы находятся в стадии разработки
3	реализованы похожие, но несколько отличные от требуемых, функции или качества системы
4	функция или качество безоговорочно реализовано

Для анализа программного обеспечения аналогичного значения были выбраны приложения следующих систем:

- приложение системы «Дорожная сеть»;
- приложение системы «MTS».

«Дорожная сеть» – цифровой агрегатор инфраструктуры для автомобильных грузоперевозок [2]. Данная системы выступает в качестве посредника между грузовыми компаниями и владельцами мест для стоянки и прочей дорожной инфраструктурой, но для отелей и владельцев стоянок выступает в качества грузовой компании и заключает с ними договор.

У данной системы есть приложение для регистрации автомобилей на стоянках. Приложение данного сервиса позволяет только устанавливать автомобиль сотрудника грузовой компании на стоянку.

Пример страницы приложения от системы «Дорожная сеть» приведён на рисунке 1.

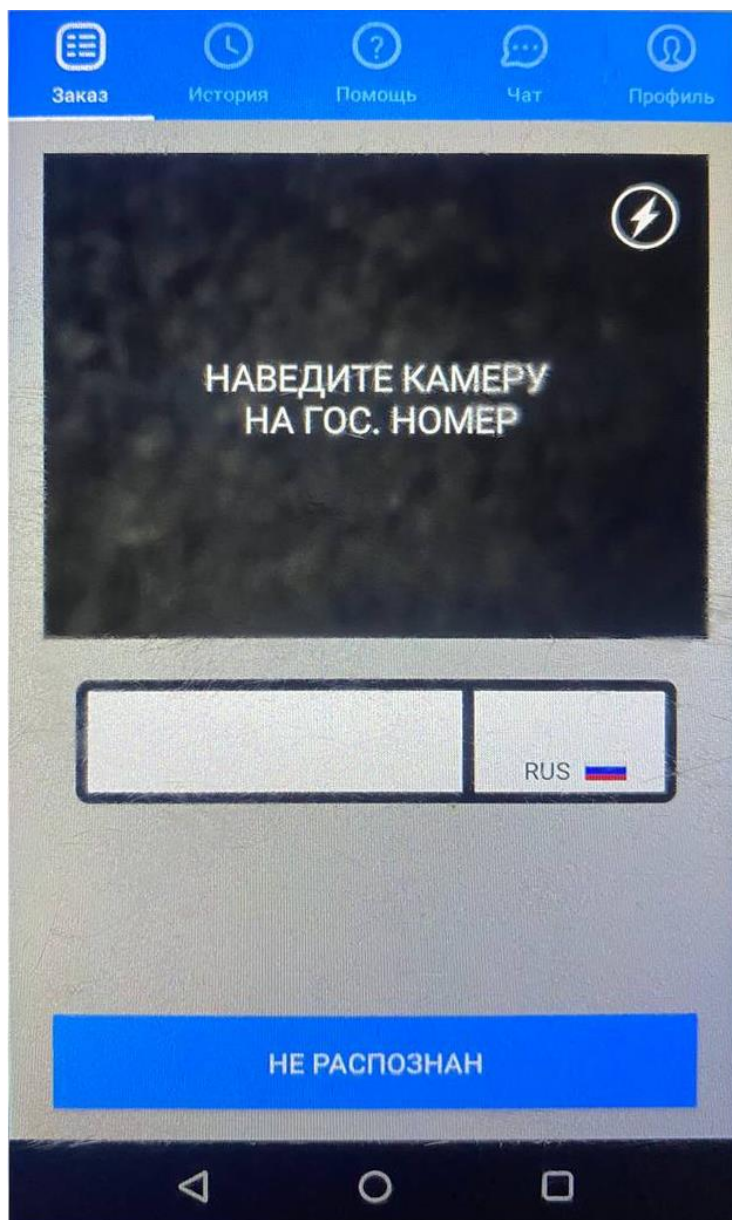


Рисунок 1 – Интерфейс приложения регистрации заказа от системы
«Дорожная сеть»

«MTS» (Международный Транспортный Сервис) – сервис, который также является посредником между грузовыми компаниями и дорожной инфраструктурой и имеет своё приложения для лиц, регистрирующих автомобили грузовых компаний на стоянках. Данное приложения, как и сервис, больше предназначены для очень маленьких компаний, так как приложение предлагает сразу все возможности взаимодействия с сервисом, показывает доходы, отзывы и прочую информацию, необходимую

администратору, но излишнюю, если регистрацией занимается рядовой сотрудник отеля или места отдыха.

Интерфейс данного приложения приведён на рисунке 2.

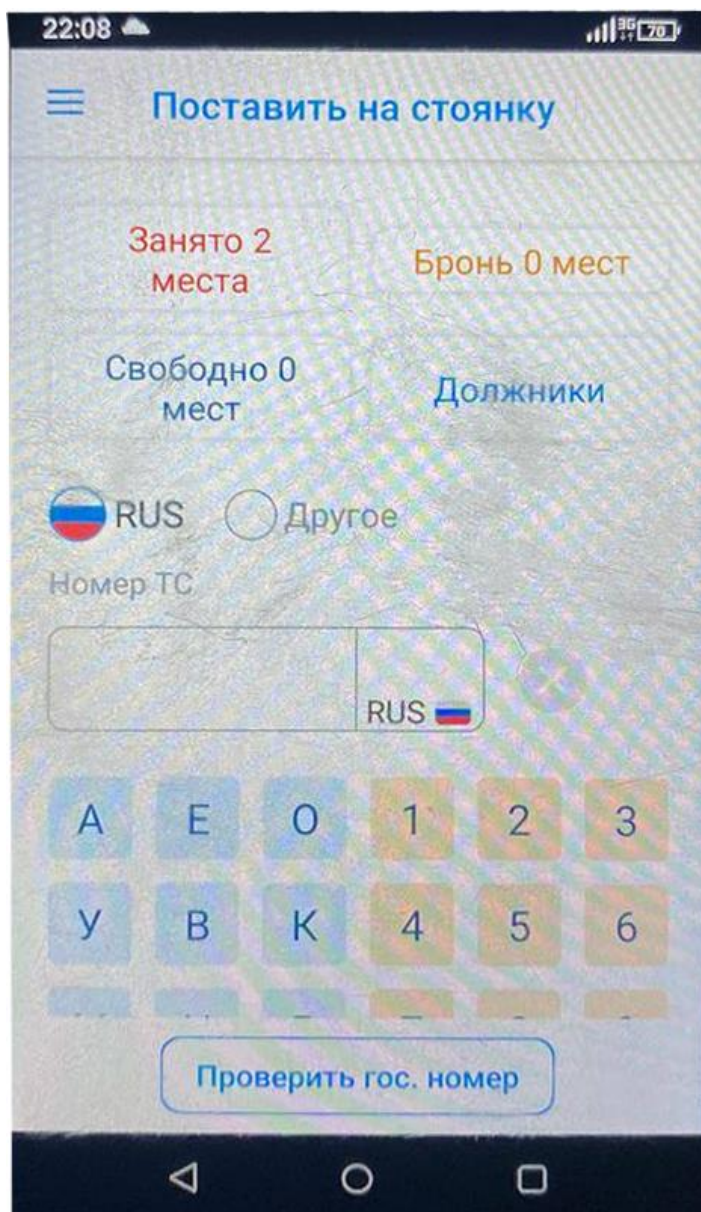


Рисунок 2 – Интерфейс приложения регистрации заказа от системы «MTS»

Оценим реализованный функционал описанных выше приложений, после чего произведем расчет аддитивной суммы интегральной оценки для каждого сравниваемого продукта. Формула для расчёта представлена ниже:

$$O_{\text{пп}} = \sum_{i=1}^n (Z_i * K_i),$$

где $O_{\text{пп}}$ – интегральная оценка программного продукта;

n – количество критериев сравнения заявленных продуктов согласно таблице критериев;

Z_i – значение выполнения критерия (1 – выполнен, 0 – не выполнен);

K_i – коэффициент значимости критерия сравнения (от 0 до 1).

Оценка реализованного функционала, полученная в результате анализа, приведена в таблице 3.

Таблица 3 – Критериальная оценка программных продуктов

№	Критерий	Программный продукт №	
		1	2
1	Поиск информации для сверки и регистрация сотрудника	4	3
2	Выбор предоставляемых услуг	1	1
3	Просмотр истории регистрации	4	4
4	Просмотр справочной информации	4	4
5	Настройка предоставляемых услуг	1	1
6	Адаптивность к размерам экрана устройств	4	4
7	Интуитивно-понятный интерфейс	4	4
8	Бесплатность продукта	4	4
9	Открытость программного кода	1	1
	Интегральная оценка	25,25	24,25

Подводя итог сравнения, можно сделать вывод, что приложения обеих систем имеют примерно равную интегральную оценку (во втором не реализована возможность ввода номера с помощью камеры, из-за чего оценка на один балл снижена), но при этом, так как данные приложения предоставляются со стороны компаний-посредников и разработаны лишь для предоставления стоянки, они не подразумевают возможности предоставления других услуг либо работы с ИНЫМИ компаниями. Также, данные приложения являются собственностью компаний, следовательно, не подразумевают открытого исходного кода и использования в других обстоятельствах.

1.3 Анализ функциональных требований

Согласно выделенным функциональным требованиям, была определена диаграмма вариантов использования. Данная диаграмма представлена на рисунке 3.

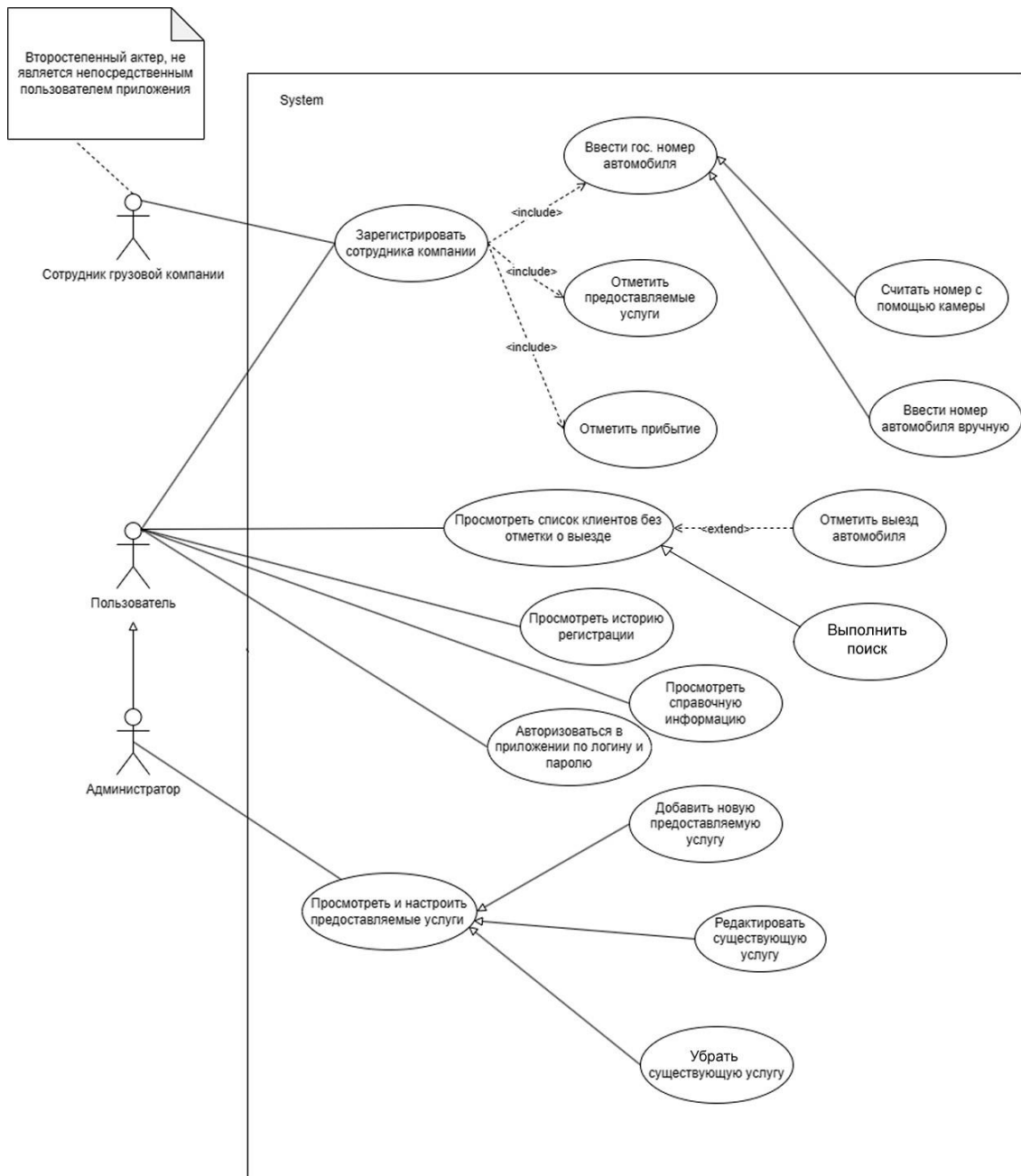


Рисунок 3 – Диаграмма вариантов использования

На данной схеме определены три пользователя – сотрудник грузовой компании, администратор и пользователь, являющийся сотрудником, регистрирующим прибытие. Администратор и пользователь приложения

имеют доступ к основным функциям приложения, но администратор, помимо этого, также может настраивать предоставляемые услуги, которые можно отметить с помощью приложения. Сотрудник грузовой компании выступает в виде лица, предоставляющего приложению и его пользователю информацию для сверки данных и регистрации.

Сценарии вариантов использования, требующих подробного пояснения, представлены на рисунках 4 – 6 в виде спецификаций.

На рисунке 4 приведено описание выполнения ролью «Пользователь» при использовании информации от роли «Сотрудник грузовой компании» варианта использования «Зарегистрировать сотрудника компании». Также, показан альтернативный поток с использованием ручного способа ввода номера автомобиля.

Наименование: Зарегистрировать сотрудника компании
ID: 1
Краткое описание: пользователь регистрирует прибывшего сотрудника грузовой компании в месте отдыха.
Главные актеры: Пользователь
Второстепенные актеры: Сотрудник грузовой компании
Предусловие: пользователь открыл приложение и находится в меню регистрации клиента, рядом находится автомобиль сотрудника грузовой компании
Основной поток: 1. Пользователь нажимает кнопку «Зарегистрировать сотрудника компании» 2. Пользователь выбирает ввод с помощью камеры смартфона 3. Пользователь наводит камеру на номер автомобиля до полного распознавания. 4. Пользователь сверяет полученные из базы данным сведения со сведениями, предоставляемыми сотрудником компании. 5. Пользователь отмечает предоставляемые согласно договору с компанией услуги 6. Пользователь отмечает прибытие сотрудника грузовой компании
Постусловие: Прибывший сотрудник компании успешно зарегистрирован и имеет право получить предоставляемые услуги.
Альтернативные потоки: 2б. Пользователь выбирает ввод номера автомобиля вручную. 3б. Пользователь вручную вводит номер автомобиля.

Рисунок 4 – Спецификация для варианта использования «Зарегистрировать сотрудника компании»

На рисунке 5 приведена спецификация варианта использования «Добавить новую предоставляемую услугу».

Наименование: Добавить определённой компании новую предоставляемую услугу
ID: 2
Краткое описание: администратор добавляет услугу в список услуг, которые возможно отметить при регистрации клиента из данной компании.
Главные актеры: Администратор
Второстепенные актеры: Нет
Предусловие: Администратор авторизовался в приложении.
Основной поток: 1. Администратор открывает список компаний и выбирает необходимую компанию 2. Администратор отмечает услугу как активную. 3. Администратор отмечает ограничение предоставления данной услуги сотруднику компании за одно прибытие.
Постусловие: Услуга добавлена в список предоставляемых услуг.
Альтернативные потоки: Нет.

Рисунок 5 – Спецификация для варианта использования «Добавить определённой компании новую предоставляемую услугу»

На рисунке 6 приведено описание основного порядка выполнения ролью «Пользователь» варианта использования «Отметить выезд автомобиля».

Наименование: Отметить выезд автомобиля
ID: 3
Краткое описание: Пользователь отмечает выезд автомобиля.
Главные актеры: Пользователь.
Второстепенные актеры: Нет
Предусловие: Пользователь вошёл в приложение, есть успешно зарегистрированный сотрудник грузовой компании, получивший необходимые услуги и намеревающийся покинуть зону стоянки.
Основной поток: 1. Пользователь открывает список клиентов без отметки о выезде. 2. Пользователь находит выезжающего клиента и ставит отметку о выезде автомобиля.
Постусловие: Обслуживание клиента завершено, запись о предоставлении услуг клиенту появилась в истории регистраций и удалена запись из списка клиентов без отметки о выезде.
Альтернативные потоки: Нет.

Рисунок 6 – Спецификация для варианта использования «Просмотреть атрибутивную информацию спортобъекта»

Как итог, были смоделированы бизнес-процессы, позволяющие пользователю и администратору проводить различные операции. В результате были смоделированы бизнес-процессы, позволяющие пользователю производить анализ размещения спортивных объектов.

1.4 Анализ нефункциональных требований

Серверная часть приложения должна быть разработана в среде разработки PyCharm на языке программирования Python при использовании фреймворка Django и Rest API framework. Выбран архитектурный подход REST API.

Клиентская часть приложения должна быть написана на языке kotlin в среде разработки Android Studio. В качестве СУБД выбрана PostgreSQL. Взаимодействие с СУБД необходимо организовать с помощью ORM, встроенной в фреймворк Django. Приложение должно работать на устройствах с операционной системой Android версии 8.0 и выше.

2 Проектирование мобильного приложения регистрации сотрудников компании перевозок в отеле

2.1 Архитектура приложения

Данное мобильное приложение будет написано с использованием трёхуровневой архитектуры клиент-серверного приложения. Схема данной архитектуры представлена на рисунке 7.

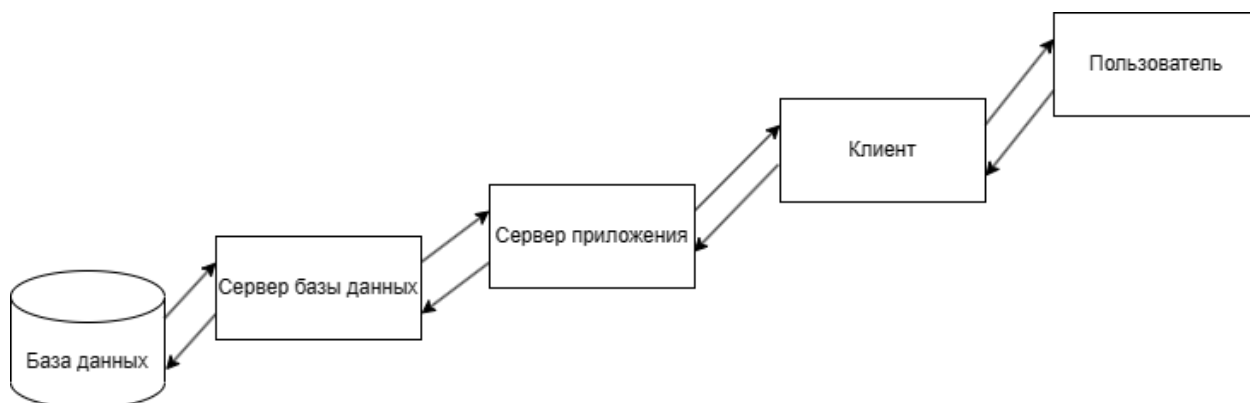


Рисунок 7 – Трёхуровневая клиент-серверная архитектура приложения

В трёхуровневой архитектуре представлены следующие уровни:

- уровень представления;
- уровень предметной области (элементы системы, выполняющие основные функции);
- уровень источника данных.

В трёхуровневой архитектуре сервер баз данных, файловый сервер и другие представляют собой отдельный уровень, результаты работы которого использует сервер приложений [3]. Логика данных и бизнес-логика находятся в сервере приложений. Все обращения пользователей через клиентов к базе данных происходят через промежуточное программное обеспечение, которое находится на сервере приложений. Вследствие этого, повышается гибкость работы системы и её производительность. Преимущества трёхуровневой архитектуры:

- целостность данных;
- высокая безопасность;

- защищённость базы данных от несанкционированного проникновения;
- ограничения такой структуры;
- более сложная структура коммуникаций между клиентов и сервером.

Взаимодействие пользователя с клиентом будет производиться согласно архитектуре REST API. REST API это архитектурный подход, который устанавливает ограничения для API: как они должны быть устроены и какие функции поддерживать [4]. Это позволяет стандартизировать работу программных интерфейсов, сделать их более удобными и производительными.

Архитектурный стиль REST – один из самых распространённых подходов к проектированию. Данный стиль применяют в следующих случаях:

- пропускная способность соединения с сервером ограничена;
- нужно соединить мобильные приложения с серверными;
- проект разбит на микросервисы;
- сервис предоставляет свои возможности другим разработчикам;
- известно, что систему нужно будет масштабировать.

Из возможностей и преимуществ, которые даёт использование REST API, данный подход был выбран для данного проекта.

Таким образом, в данном проекте будет реализовано взаимодействие клиента и сервера приложения осуществляется с помощью REST API запросов по HTTP протоколу, а сервер взаимодействует с источником данных, представленным в виде СУБД PostgreSQL, которая непосредственно осуществляет запросы к источнику данных.

2.2 Проектирование структур данных

На сегодняшний момент реляционные базы данных – это средство, позволяющее наиболее эффективно структурировать данные, обеспечить наибольшую безопасность и упростить написание прикладного ПО [5].

Поэтому, в данном приложении для хранения информации необходимо использовать базы данных. Ниже представлено проектирование структуры и организации данных в приложении.

2.2.1 Концептуальная модель базы данных приложения

Цель концептуального проектирования – создание концептуальной схемы данных на основе представлений о предметной области каждого отдельного типа пользователей [6]. При разработке концептуальной модели, прежде всего, нужно определить сущности. С этой целью нужно сделать следующее [7]:

- необходимо понять, о чем должна храниться и обрабатываться информация и можно ли это определить как сущность;
- присвоить этой сущности имя;
- выявить атрибуты сущности и присвоить им имя.

Выявив сущности, необходимо определить, какие связи имеются между ними. При определении связей (естественно, рассматриваются только те связи, которые имеют отношение к решаемым задачам обработки данных) необходимо учитывать следующее:

- то, как экземпляр одной сущности связан с экземпляром другой сущности;
- то, как должны быть установлены связи, чтобы была возможность ответа на все запросы пользователей (исходя из информационных потребностей пользователей).

Реализация концептуальной схемы базы данных проведена в графической нотации IDEF0, позволяющей точнее определить зависимости между моделями.

Ниже представлена концептуальная схема базы данных, спроектированная для данного приложения (рисунок 8).

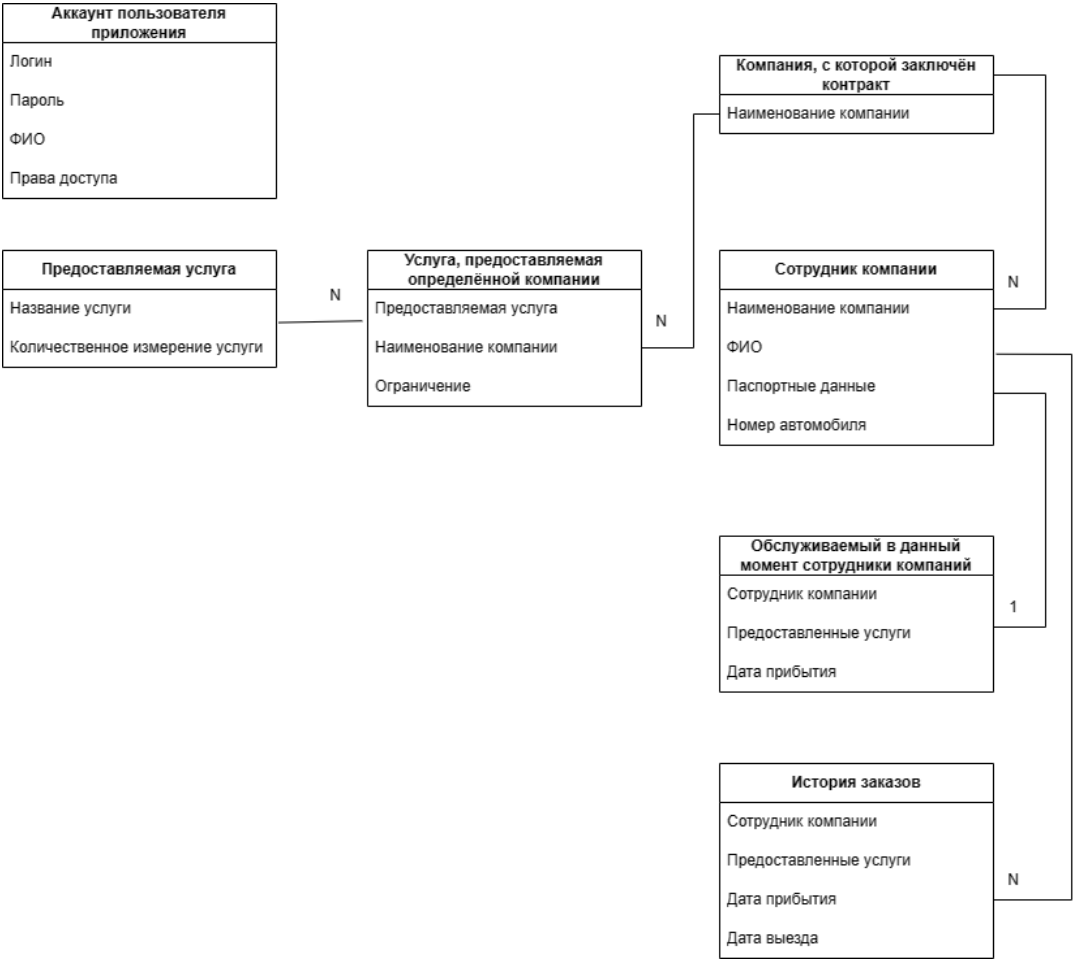


Рисунок 8 – Концептуальная модель базы данных приложения

В таблице 4 представлено описание сущностей, сформированное на основании информации, полученной при анализе данных бизнес-процессов.

Таблица 4 – Описание сущностей

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Аккаунт пользователя приложения	Аккаунт администратора или работника места отдыха, непосредственного пользователя приложения	Стержневая	50	В среднем 2 раза в год
2	Предоставляемая услуга	Услуга, которую может предоставить данное место отдыха	Стержневая	20	В среднем 2 раза в год
3	Компания, с которой заключён договор	Компания, на основании договора с которой предоставляются услуги по нестандартной процедуре и для которой будет использоваться данное приложение	Стержневая	40	В среднем 2 раза в год

Продолжение таблицы 4

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
4	Услуга, предоставляемая определённой компании	Услуга, которую пользователь приложения может отметить при регистрации сотрудника компании и ограничение по предоставлению данной услуги	Вспомогательная	100	В среднем 2 раза в месяц
5	Сотрудник компании	Сотрудник компаний и основная информация по нему,	Стержневая	200	В среднем 1 раз в месяц
6	Обслуживаемый сотрудник компании	Сотрудник компаний, который находится на территории места отдыха и пользуется услугами.	Стержневая	40	В среднем 20 раз в день
7	История заказов	Архив с информацией о завершённых предоставлениях услуг сотрудникам компаний.	Вспомогательная	5000	В среднем 10 раз в день

Таким образом, выяснилось, что для реализации данного приложения требуется семь сущностей.

2.2.2 Логическая модель базы данных и описание атрибутов

Логическая модель – это расширение концептуальной схемы базы данных. Она включает в себя все сущности, атрибуты, ключи и взаимосвязи, которыми определяются бизнес-правила [8].

Логическая модель базы данных представлена на рисунке 9.

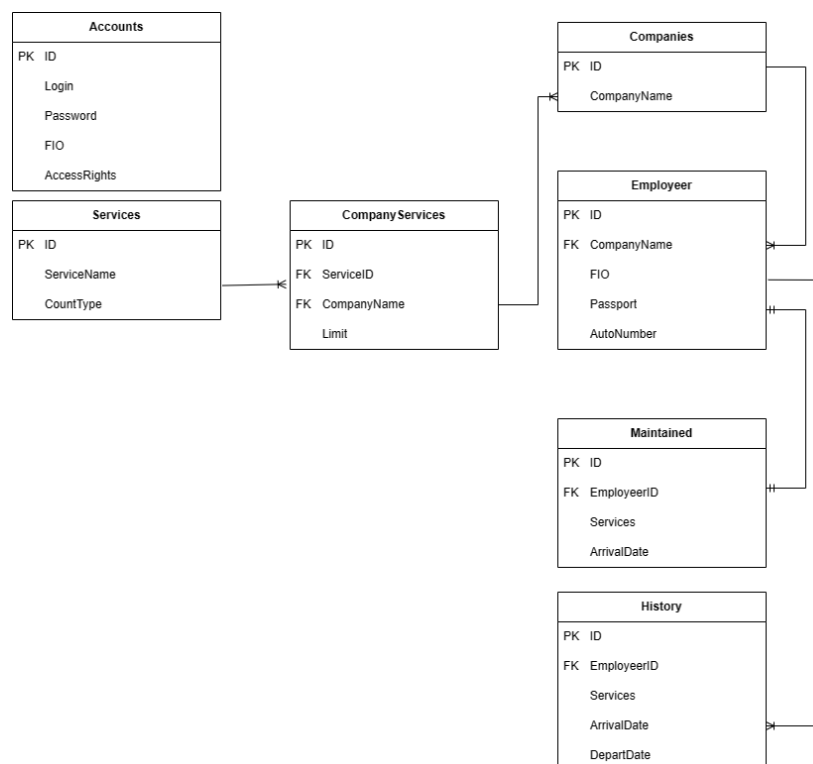


Рисунок 9 – Логическая модель базы данных

На основании построенной логической модели были выделены характеристики атрибутов каждой сущности. Атрибуты и их описание представлены в таблицах ниже.

В таблице 5 представлено описание атрибутов сущности «Accounts».

Таблица 5 – Описание атрибутов сущности «Accountst»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
ID	Уникальный номер актера в таблице	Целый	4	1 – 9999	Нет	PK	1553	–
Login	Логин от аккаунта	Символьный	30	Любые символы	Нет	–	admin1	–
Password	Пароль от аккаунта	Символьный	30	Любые символы	Нет	–	Password12345	–
FIO	Фамилия, имя и отчество	Символьный	100	Любые символы	Нет	–	Иванов Иван Иванович	Отчество в строке может отсутствовать
AccessRights	Права доступа аккаунта	Символьный	30	«User» или «Administrator»	Нет	–	Administrator	–

В таблице 6 представлено описание атрибутов сущности «Services».

Таблица 6 – Описание атрибутов сущности «Services»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
ID	Уникальный номер сервиса в таблице	Целый	4	1 – 9999	Нет	PK	1553	–
ServiceName	Название услуги	Символьный	100	Любые символы	Нет	–	Стоянка, Комната, приём пищи, душ	Используется при составлении JSON в других сущностях
CountType	Способ подсчёта объёма предоставляемой услуги	Символьный	30	Любые символы	Нет	–	День, час, шт.	Нужен для пояснения при регистрации и установке ограничения

В таблице 7 представлено описание атрибутов сущности «CompanyServices».

Таблица 7 – Описание атрибутов сущности «CompanyServices»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
ID	Уникальный номер предоставляемой компании услуги в таблице	Целый	4	1 – 9999	Нет	PK	1553	–
ServiceID	Идентификатор услуги	Целый	Целый	1 – 9999	Нет	FK	1255	–
Company Name	Имя компании, которой предоставляется услуга	Символьный	30	Любые символы	Нет	–	День, час, шт.	–
Limit	Ограничение по предоставлению услуги сотруднику за регистрацию	Целый	4	1 – 9999	Нет	–	120	–

В таблице 8 представлено описание атрибутов сущности «Companies».

Таблица 8 – Описание атрибутов сущности «Companies»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
ID	Уникальный номер актера в таблице	Целый	4	1 – 9999	Нет	PK	1553	–
Company Name	Наименование компании	Символьный	30	Любые символы	Нет	–	ООО «ДальГруз», ИП «Курников Д.У.»	–

В таблице 9 представлено описание атрибутов сущности «Employee».

Таблица 9 – Описание атрибутов сущности «Employee»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
ID	Уникальный номер сервиса в таблице	Целый	4	1 – 9999	Нет	PK	1553	–
Company Name	Наименование компании	Символьный	30	Любые символы	Нет	FK	ООО «ДальГруз», ИП «Курников Д.У.»	–

Продолжение таблицы 9

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
FIO	Фамилия, имя и отчество регистрирующегося клиента	Символьный	100	Любые символы	Нет	–	Иванов Иван Иванович	Отчество в строке может отсутствовать
Passport	Паспортные данные клиента	Символьный	10	Цифры 0-9	Нет	–	6320295984	–

В таблице 10 представлено описание атрибутов сущности «Maintained».

Таблица 10 – Описание атрибутов сущности «Maintained»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
ID	Уникальный номер актера в таблице	Целый	4	1 – 9999	Нет	PK	1553	–
EmployeeID	Уникальный номер обслуживаемого клиента	Целый	4	1 – 9999	Нет	FK	1553	–
Services	Строка формата JSON, содержащая информацию о полученных клиентом услугах	JSON	10000	Любые символы	Нет	–	[{"serviceName": "Стоянка", "serviceCount": 2}]	Нужен для последующей обработки для подсчёта оплаты компаний
ArrivalDate	Дата и время регистрации и клиента	Символьный	16	01.01.2023 - 01.01.2070	Нет	–	12.12.2023:30	–

В таблице 11 представлено описание атрибутов сущности «History».

Таблица 11 – Описание атрибутов сущности «History»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
ID	Уникальный номер актера в таблице	Целый	4	1 – 9999	Нет	PK	1553	–

Продолжение таблицы 11

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
EmployeeID	Уникальный номер обслуживаемого клиента	Целый	4	1 – 9999	Нет	FK	1553	–
Services	Строка JSON, содержащая информацию о полученных клиентом услугах	JSON	10000	Любые символы	Нет	–	[{"service Name": "Стойка", "serviceCount": 2}]	Нужен для обработки в других приложениях
ArrivalDate	Дата и время регистрации клиента	Символьный	15	01.01.2023 - 01.01.2070	Нет	–	12.12.2023	–
DepartDate	Дата и время отметки о выезде клиента	Символьный	15	01.01.2023 - 01.01.2070	Нет	–	12.12.2023	–

В итоге, были выявлены атрибуты сущностей, используемых в проекте.

2.2.3 Физическая модель базы данных

Физическая модель – логическая модель базы данных, выраженная в терминах языка описания конкретной СУБД.

Физическая модель базы данных представлена на рисунке 10.

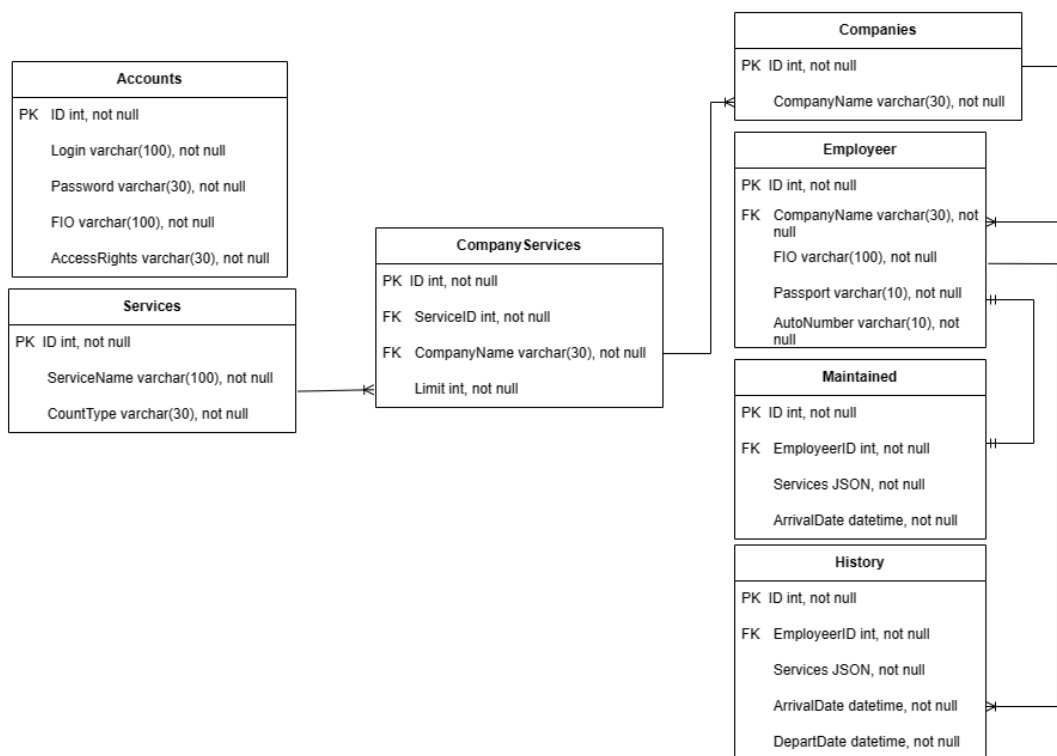


Рисунок 10 – Физическая модель базы данных

Как итог, в данной работе была представлена физическая модель базы данных представлена в терминах СУБД PostgreSQL.

2.3 Проектирование программных средств

На основе модели базы данных была спроектирована диаграмма классов для программной реализации сущностей в клиентском приложении.

Данная диаграмма также подходит как к клиентскому, так и к серверному приложению, так как при использовании REST API-приложения взаимосвязь будет проводиться аналогичными классами с подобными типами данных и идентичными названиями.

Полученная диаграмма классов показана ниже (рисунок 11).

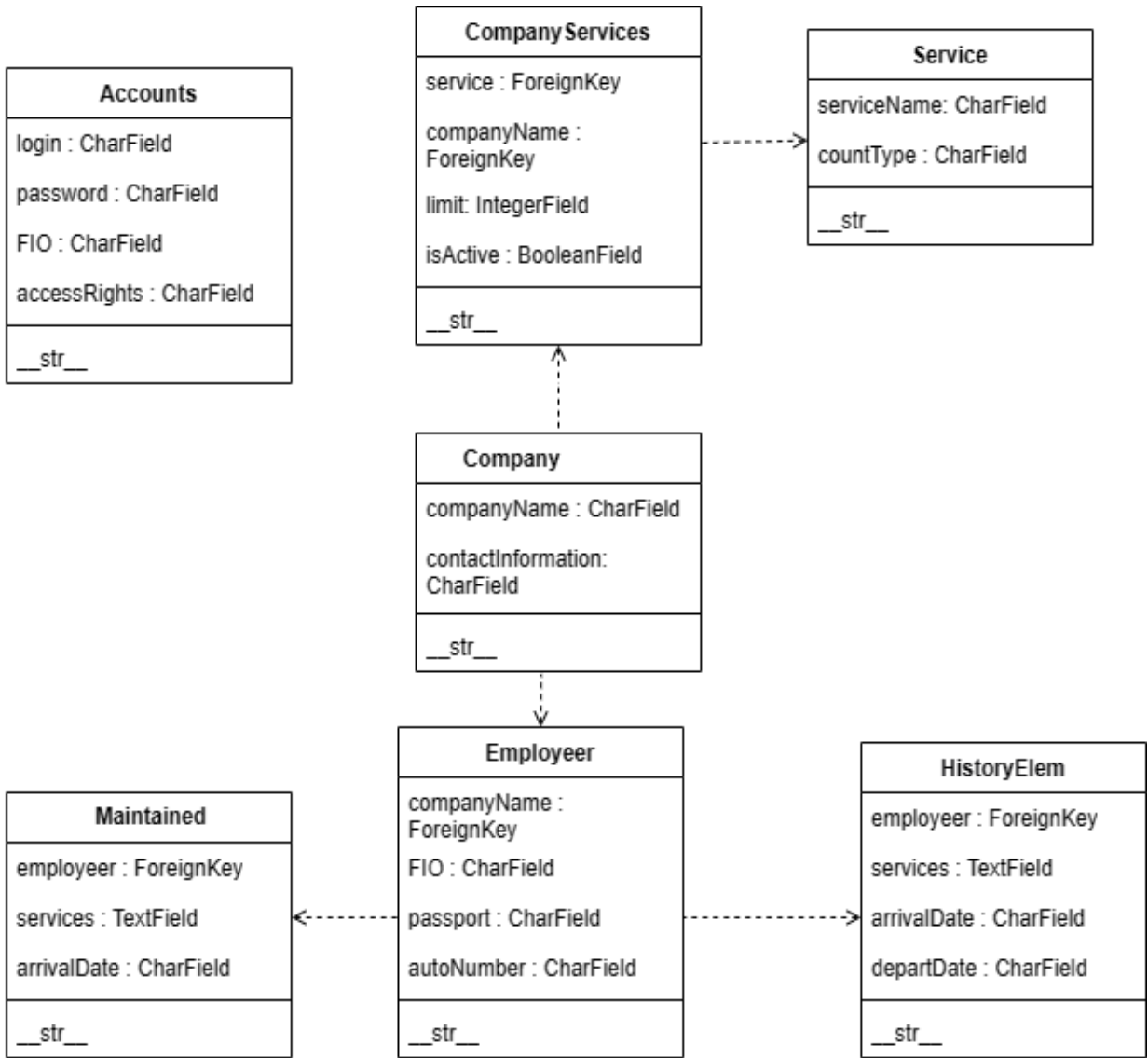


Рисунок 11 – Диаграмма классов приложения

Для уточнения функциональных требований была спроектирована диаграмма деятельности варианта использования «Зарегистрировать сотрудника компании». Данная диаграмма представлена на рисунке 12.

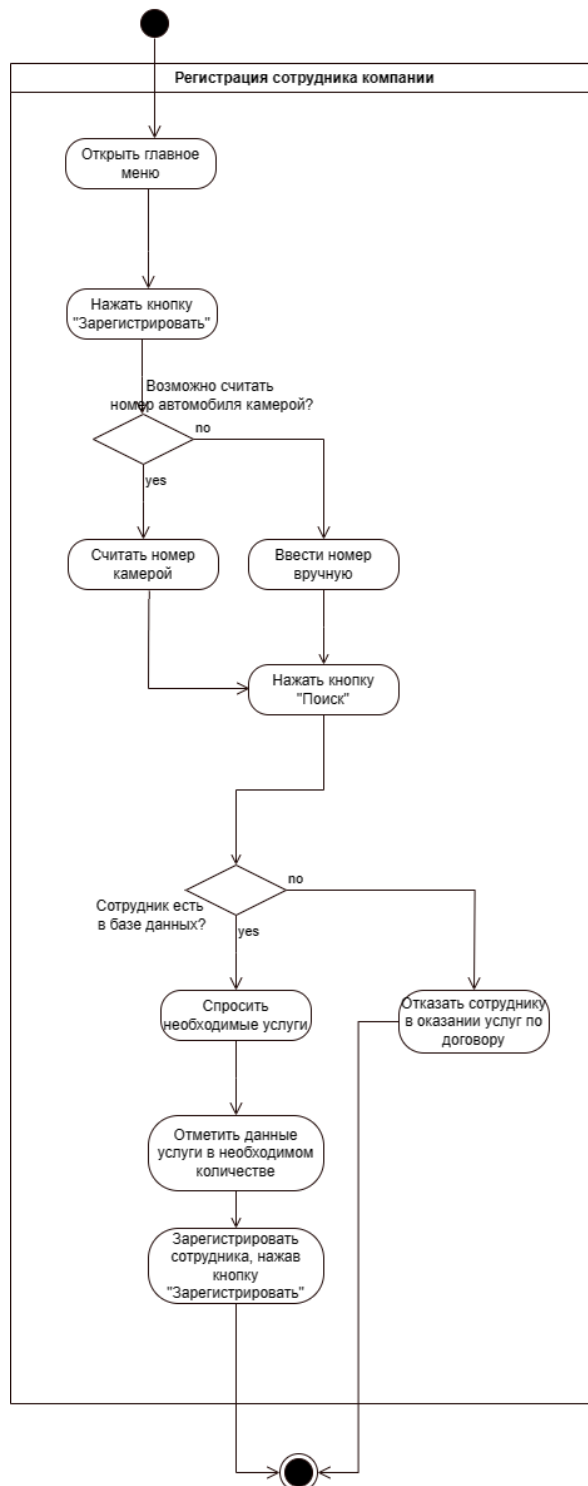


Рисунок 12 – Диаграмма деятельности для варианта использования «Зарегистрировать сотрудника компании»

Из диаграммы видно, что при регистрации сотрудников компаний, с которыми заключён договор, у пользователя приложения есть возможность ввести как при помощи автоматического распознавания текста с камеры, так и записать номер автомобиля вручную.

2.4 Проектирование диаграммы последовательности

Диаграммы взаимодействия (interaction diagrams) описывают взаимодействие групп объектов в различных условиях их поведения [9]. UML определяет диаграммы взаимодействия нескольких типов, из которых наиболее употребительными являются диаграммы последовательности.

Обычно диаграмма последовательности описывает один сценарий. На диаграмме показаны экземпляры объектов и сообщения, которыми обмениваются объекты в рамках одного прецедента.

Для анализа взаимодействия сервера и клиентского приложения была реализована диаграмма последовательности варианта использования «Редактировать существующую услугу», представленная на рисунке 13.

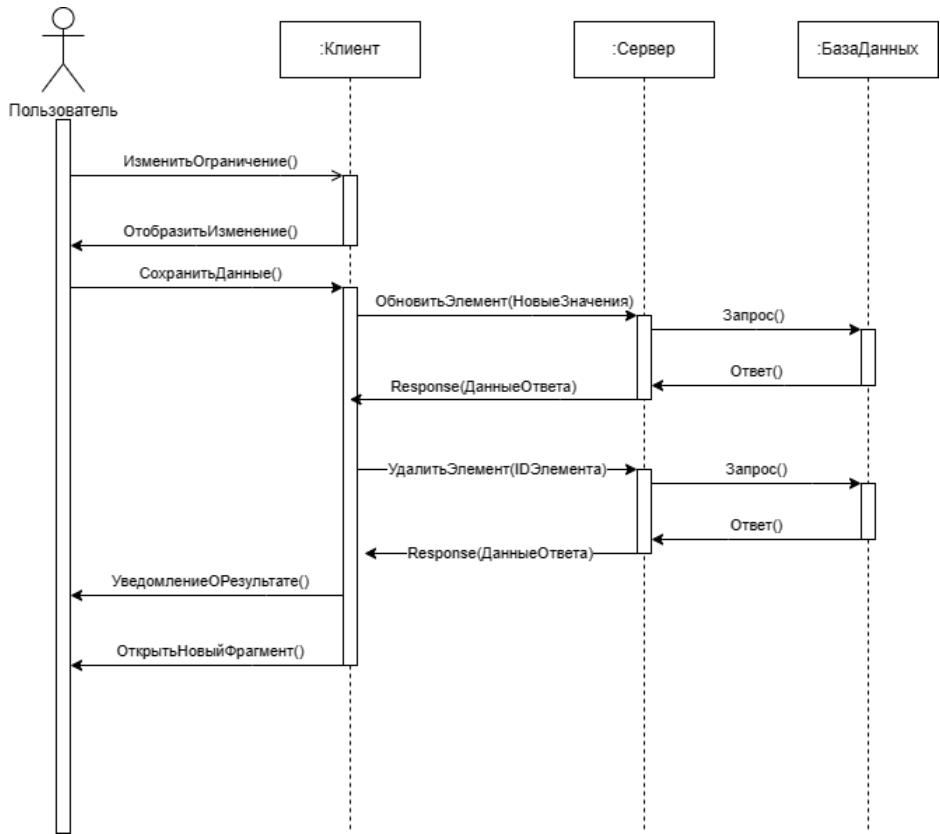


Рисунок 13 – Диаграмма последовательности

Как видно из диаграммы, основные объекты, выполняющие взаимодействие, это пользователь, клиент, сервер и база данных, и получение информации из базы данных происходит поэтапно, начиная с действий пользователя с клиентским приложением, которое делает запрос к серверу, которое, в свою очередь, уже отправляет запросы и получает ответы от базы данных.

2.5 Проектирование пользовательского интерфейса

На основе полученных данных о бизнес процессах и функциональных требованиях, был спроектирован интерфейс мобильного приложения. Пользовательский интерфейс является важной частью каждого приложения, он должен обеспечивать интуитивно понятное, быстрое и эффективное выполнение основных функций пользователей.

В данном случае, в приложении как минимум должны присутствовать следующие элементы интерфейса:

- «Панель управления»;
- «Страница ввода номера автомобиля сотрудника»;
- «Страница регистрации сотрудника»;
- «Страница отметки выезда сотрудника»;
- «Страница авторизации»;
- «Страница справки»;
- «Страницы просмотра списков данных, таких, как история оказания услуг, список компаний и список обслуживаемых сотрудников»;
- «Изменение услуг, предоставляемых компании».

Ниже представлены скриншоты макетов, созданных в встроенном дизайнера интерфейса Android Studio и описание элементов интерфейса.

На рисунке 14 показана панель управления. Здесь, при нажатии на элементы меню, возможно осуществить переход по основным разделам приложения. Меню выдвигается свайпом вправо либо нажатием на кнопку в

верхнем левом углу. Также, здесь есть возможность увидеть авторизованного в данный момент пользователя и сменить аккаунт.

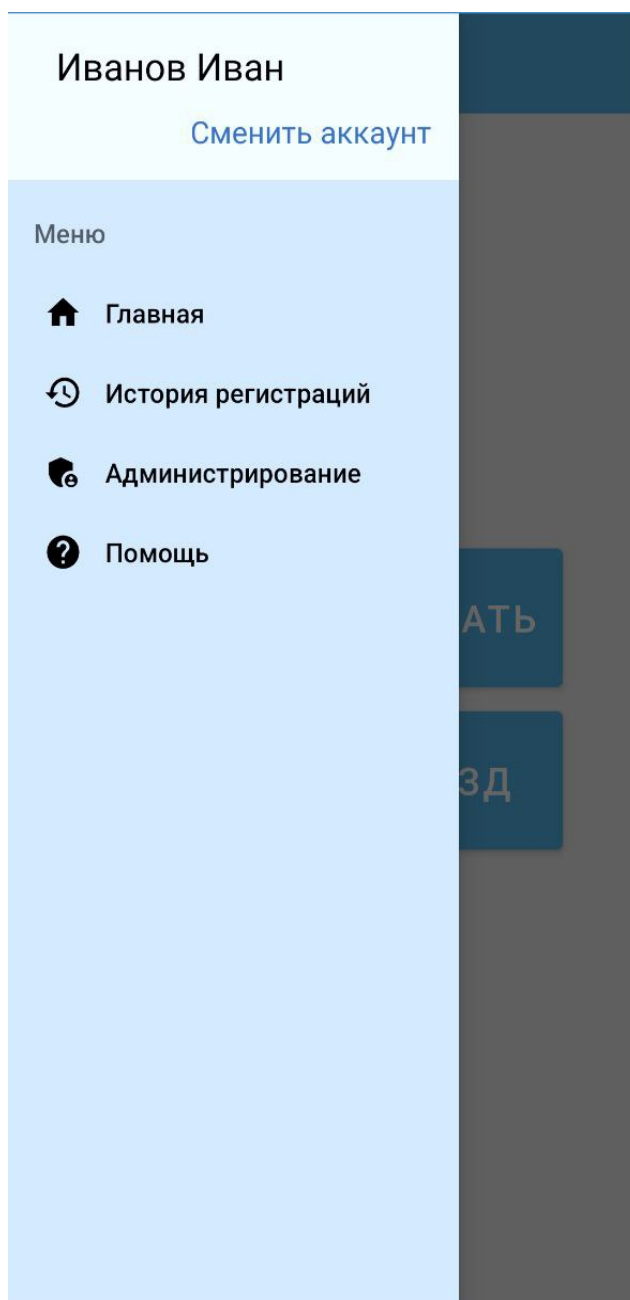


Рисунок 14 – Панель управления приложения

На странице, показанной на рисунке 15, осуществляется ввод номера автомобиля. Посередине находится квадрат с отображением того, что в текущий момент снимает камера, под ним – поле для ручного ввода номера. При нажатии кнопки «Ввести вручную» выдвигается сенсорная клавиатура для осуществления ввода в поле. При нажатии на кнопку «Поиск» осуществляется поиск по введённым данным. Если пользователь не

воспользовался ручным вводом, приложение автоматически распознает текст с камеры и начнёт поиск сотрудника с данным номером автомобиля.

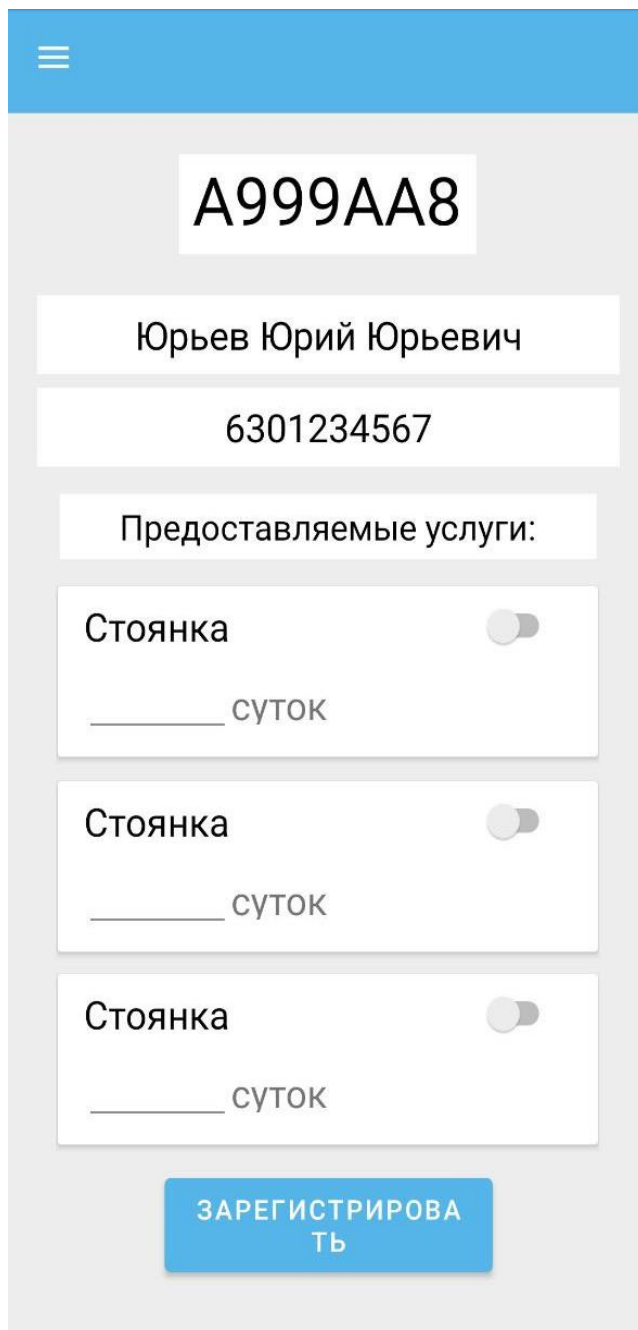


Рисунок 15 – Страница ввода номера автомобиля сотрудника

В случае, если сотрудник был успешно найден в базе данных, откроется страница, представленная на рисунке 16.

На данной странице регистрирующий работник сверяет собственные данные приехавшего работника, такие как фамилия, имя и отчество, а также

номер паспорта, с данными, полученными приложением с сервера, а также отмечает получаемые клиентом услуги и их количество. После завершения предварительной подготовки, пользователь приложения нажимает кнопку «Зарегистрировать», которая, в случае успеха, добавляет пользователя в список обслуживаемых.



The screenshot shows a mobile application interface for employee registration. At the top is a blue header with a white hamburger menu icon. Below the header, the registration details are displayed in white boxes on a light gray background:

- A large box containing the alphanumeric code **A999AA8**.
- A box containing the name **Юрьев Юрий Юрьевич**.
- A box containing the phone number **6301234567**.
- A box with the label **Предоставляемые услуги:** (Provided services:).
- Three identical service selection blocks, each containing:
 - The word **Стоянка** (Parking).
 - A horizontal line for input, followed by the word **СУТОК** (days).
 - A toggle switch on the right, currently in the 'off' position.
- A large blue button at the bottom with the text **ЗАРЕГИСТРИРОВАТЬ** (REGISTER) in white capital letters.

Рисунок 16 – Страница регистрации сотрудника.

Страница отметки выезда сотрудников и по совместительству страница просмотра обслуживаемых клиентов показана на рисунке 17.

На данной странице пользователь приложения находит в списке получающих услуги клиентов того, кто намеревается покинуть место отдыха и подтверждает установку отметки в всплывающем окне. При успехе операции, формируется и добавляется в базу данных запись в истории обслуживания, после чего запись об обслуживании клиента удаляется из списка и базы данных. Также, на данной странице есть поле для поиска необходимой записи по информации из неё.

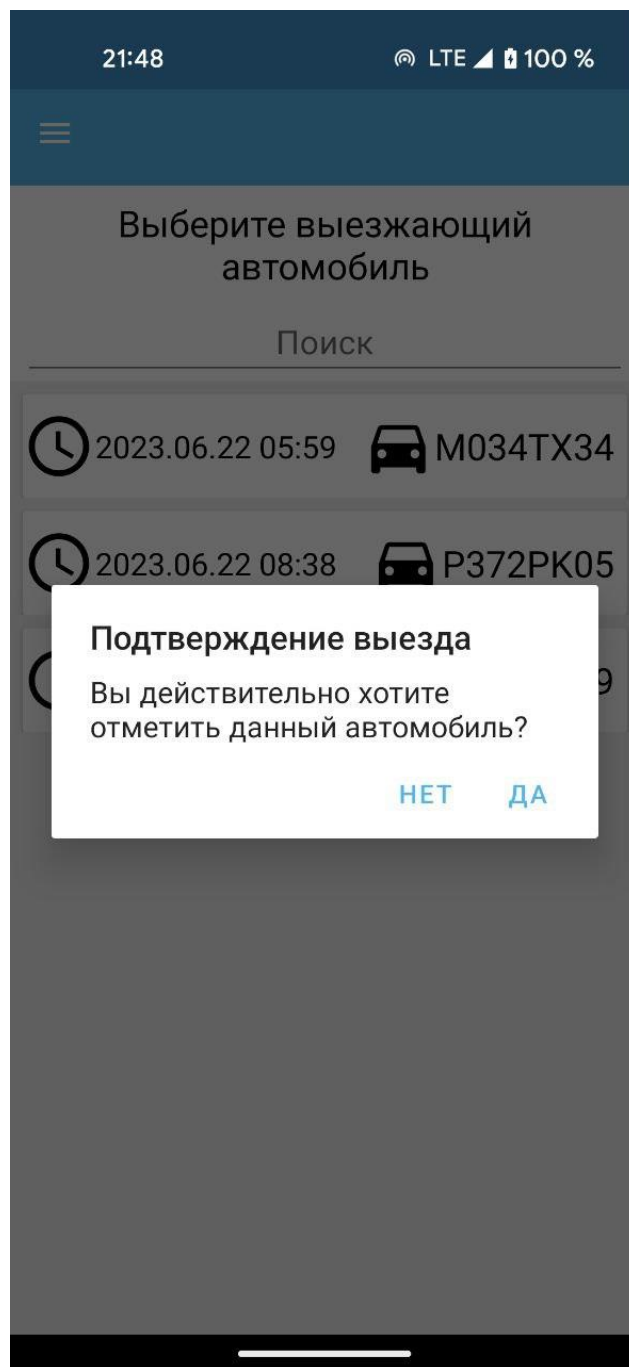
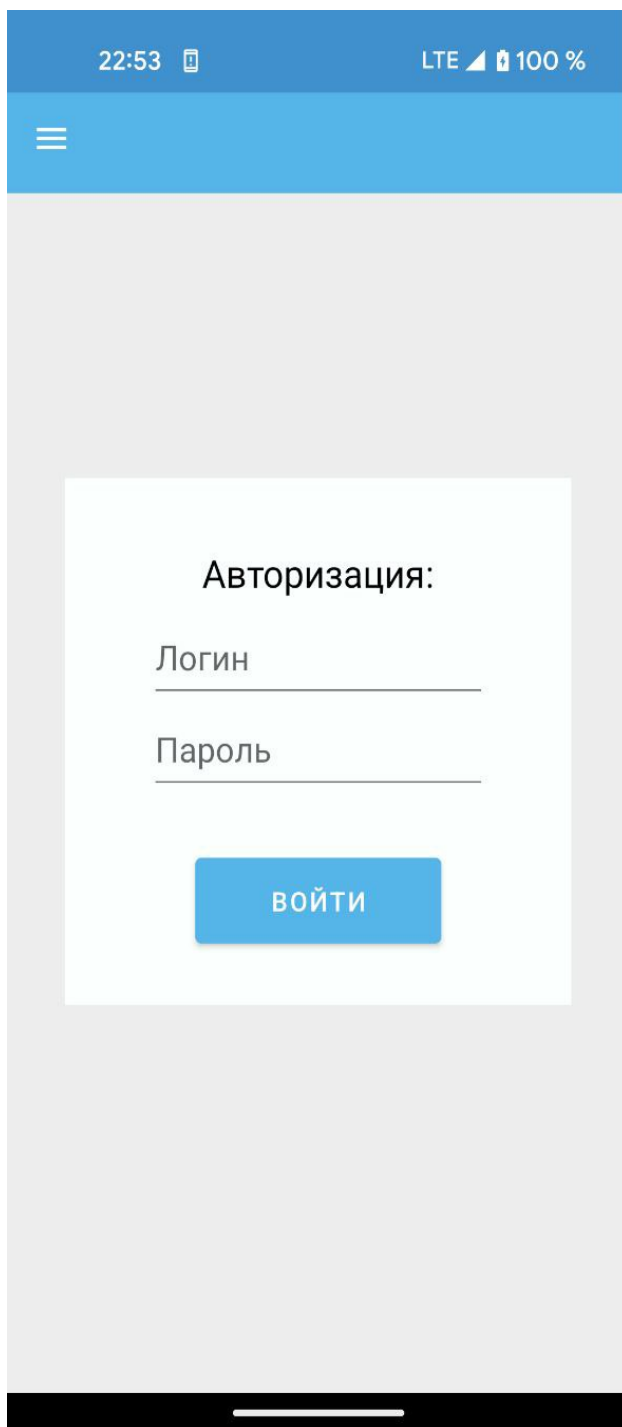


Рисунок 17 – Страница отметки выезда сотрудника

Страница авторизации пользователя приложения представлена на рисунке 18.

На данной странице производится авторизация пользователя приложения. Если пароль и(или) логин неверные, авторизация оканчивается неудачей, о чём выводится уведомление. При успешной авторизации, текущий пользователь сохраняется в памяти смартфона и отмечается как активный.



The image shows a mobile application interface for user authorization. At the top, there is a blue status bar with the time 22:53, a signal strength icon, the text 'LTE', and a battery icon with '100 %'. Below this is a blue header bar with a white hamburger menu icon. The main content area has a light gray background. In the center, there is a white rectangular box containing the text 'Авторизация:' in bold. Below this text are two input fields: the first is labeled 'Логин' and the second is labeled 'Пароль'. Both fields have a horizontal line underneath them. Below the input fields is a blue button with the white text 'ВОЙТИ'. At the very bottom of the screen, there is a black bar with a white horizontal line in the center, representing the home indicator on an iPhone.

Рисунок 18 – Страница авторизации пользователя приложения

кликабельными элементами, в которых будет находиться соответствующая информация.



Рисунок 20 – Страницы просмотра списков данных

Интерфейс страницы изменения услуг, предоставляемых компании, показан на рисунке 21.

На данной странице администратор может увидеть все услуги, предоставляемые компании по договору, а также оперативно добавить, удалить или изменить соответствующую услугу, включённую в список услуг конкретной компании в случае необходимости, переключив поля или изменив

поле ограничения. Для применения действий необходимо нажать кнопку «Сохранить». В данное меню есть, как и в просмотр списка компаний, может попасть только пользователь с административными возможностями.

The screenshot shows a mobile application interface for editing company services. At the top, there is a blue header bar with a white hamburger menu icon. Below the header, the company name "Фирма \"ДальГруз\"" is displayed in a large, bold, black font. Underneath the company name, the word "Услуги:" is written in a smaller, gray font. The main content area is a light gray rectangle. Inside this area, there is a white card with a thin gray border. The card contains the text "Стоянка" (Parking) in a bold black font, followed by "Активна" (Active) in a regular black font, and a green toggle switch that is currently turned on. Below this, the text "Ограничение" (Limit) is followed by a text input field containing the number "1", and then "(сутки)" (days) in parentheses. At the bottom of the card, there is a blue button with the white text "СОХРАНИТЬ" (SAVE).

Рисунок 21 – Изменение услуг, предоставляемых компании

В итоге, был спроектирован интерфейс клиентского приложения и определены действия, полностью подходящий под требуемые функциональные возможности.

3 Реализация приложения

Для представления реализации приложения была составлена диаграмма развертывания приложения. Данная диаграмма представлена на рисунке 22.

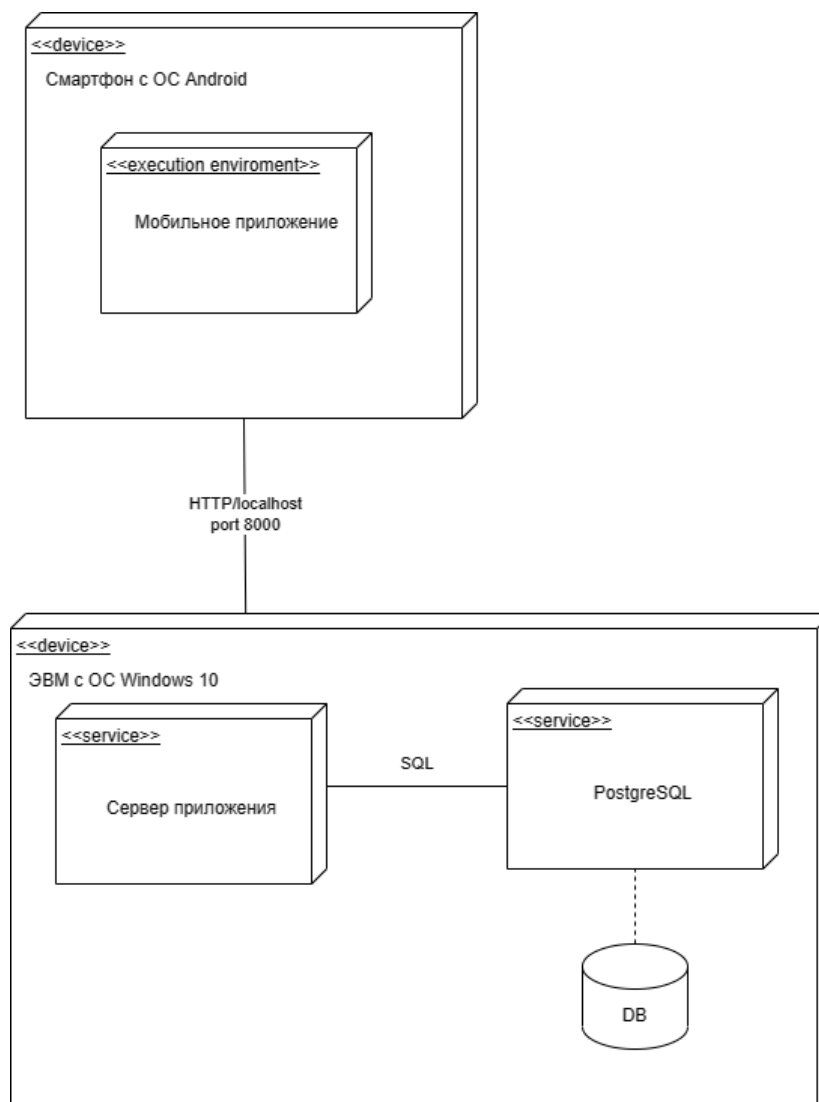


Рисунок 22 – Диаграмма развертывания

Как видно из реализации, всего используется два устройства – смартфон, на котором установлено клиентское приложение и компьютер, выполняющий роль сервера.

Код проекта представлен в приложении А.

3.1 Реализация серверной части приложения

Так как серверная часть проекта реализована на языке Python при помощи фреймворков Django и REST API, база данных создаётся на основе

model классов серверной части приложения. Ниже представлена диаграмма классов серверного приложения.

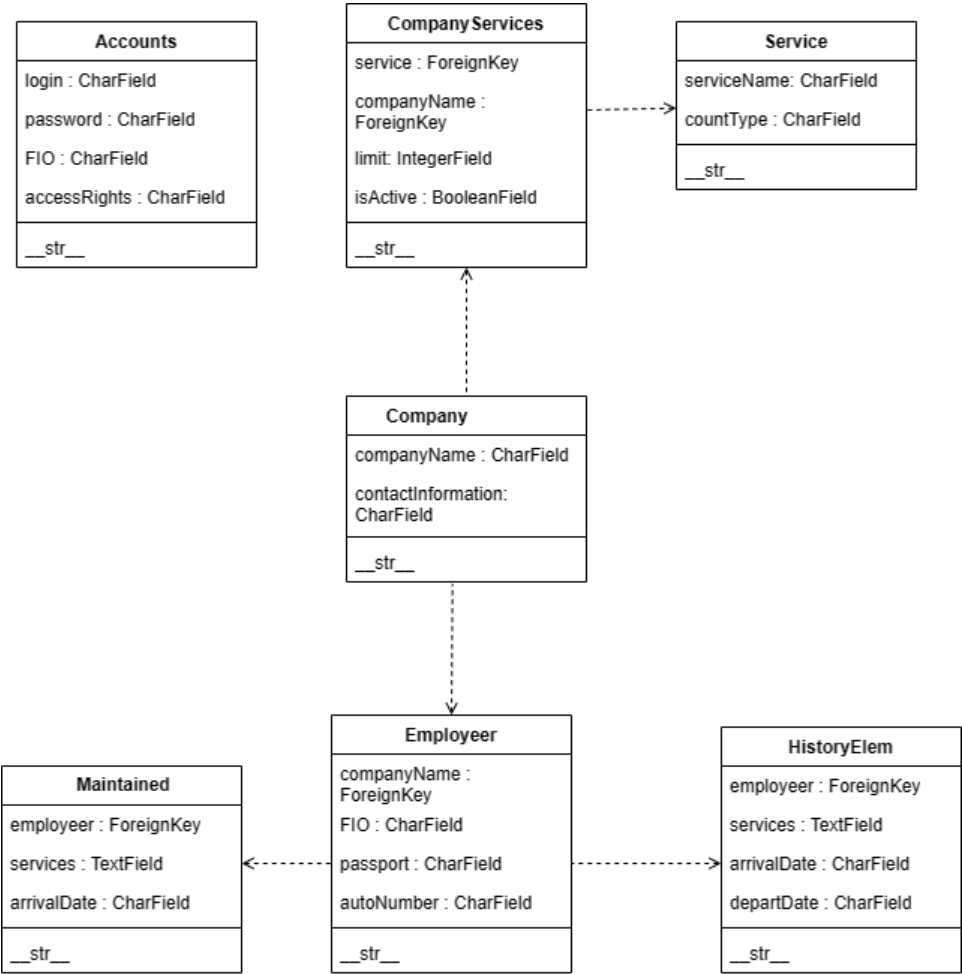


Рисунок 23 – Диаграмма классов реализованной серверной части приложения

Серверное приложение реализует архитектуру RST API, а значит, взаимодействие клиентских приложений с данными организуется с помощью обращений по URL. Предоставляемые методы показаны в таблице 12.

Таблица 12 – Спецификация REST API приложения

№	Назначение сервиса	HTTP метод	URI сервиса	Класс контроллера
1	Получение сервисов компании по имени компании	GET	api/v1/company_services_list /<str:receivedComName>	CompanyServices APIView
2	Получение одного сервиса компании по ID компании	GET	api/v1/company_service/<int:pk>	OneCompanyServiceAPIView

Продолжение таблицы 12

№	Назначение сервиса	HTTP метод	URI сервиса	Класс контроллера
3	Добавление сервиса компании	POST	api/v1/company_service/<int:pk>	OneCompanyServiceAPIView
4	Получение информации для проверки аккаунта	GET	api/v1/accounts_list	AccountsAPIView
5	Получение списка сервисов	GET	api/v1/services_list	ServiceAPIView
6	Получение списка компаний	GET	api/v1/companies_list	CompanyAPIView
7	Получение списка клиентов	GET	api/v1/employeers_list	EmployeerAPIView
8	Получение одного клиента	GET	api/v1/getemployeer/<str:receivedNumber>	OneEmployeerAPIView
9	Получение списка обслуживаемых клиентов	GET	api/v1/maintaineds_list	MaintainedAPIView
10	Добавление обслуживаемого клиента	POST	api/v1/maintaineds_list	MaintainedAPIView
11	Удаление обслуживаемого клиента	DELETE	api/v1/maintaineds_list/<int:id_del>	MaintainedAPIView
12	Получить список истории	POST	api/v1/history_list	HistoryElemAPIView

Для понимания структуры приложения реализована диаграмма компонентов сервера. Данная диаграмма представлена на рисунке 24.

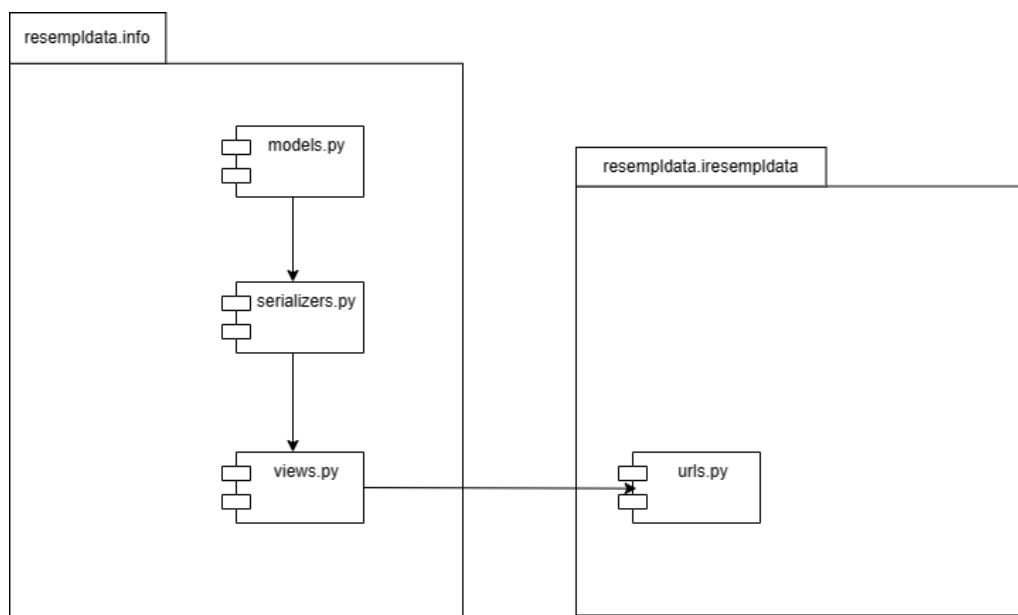


Рисунок 24 – Диаграмма компонентов

Для создания строки полученных услуг в классах Maintained_elem и history_elem был добавлен класс InvoiceService, позволяющий удобно организовать информацию и перевести её в JSON формат. Также, данный класс может потребоваться при дальнейшем расширении сервера или подключении клиентских приложений, выполняющих другие цели, например, подсчёт оплаты.

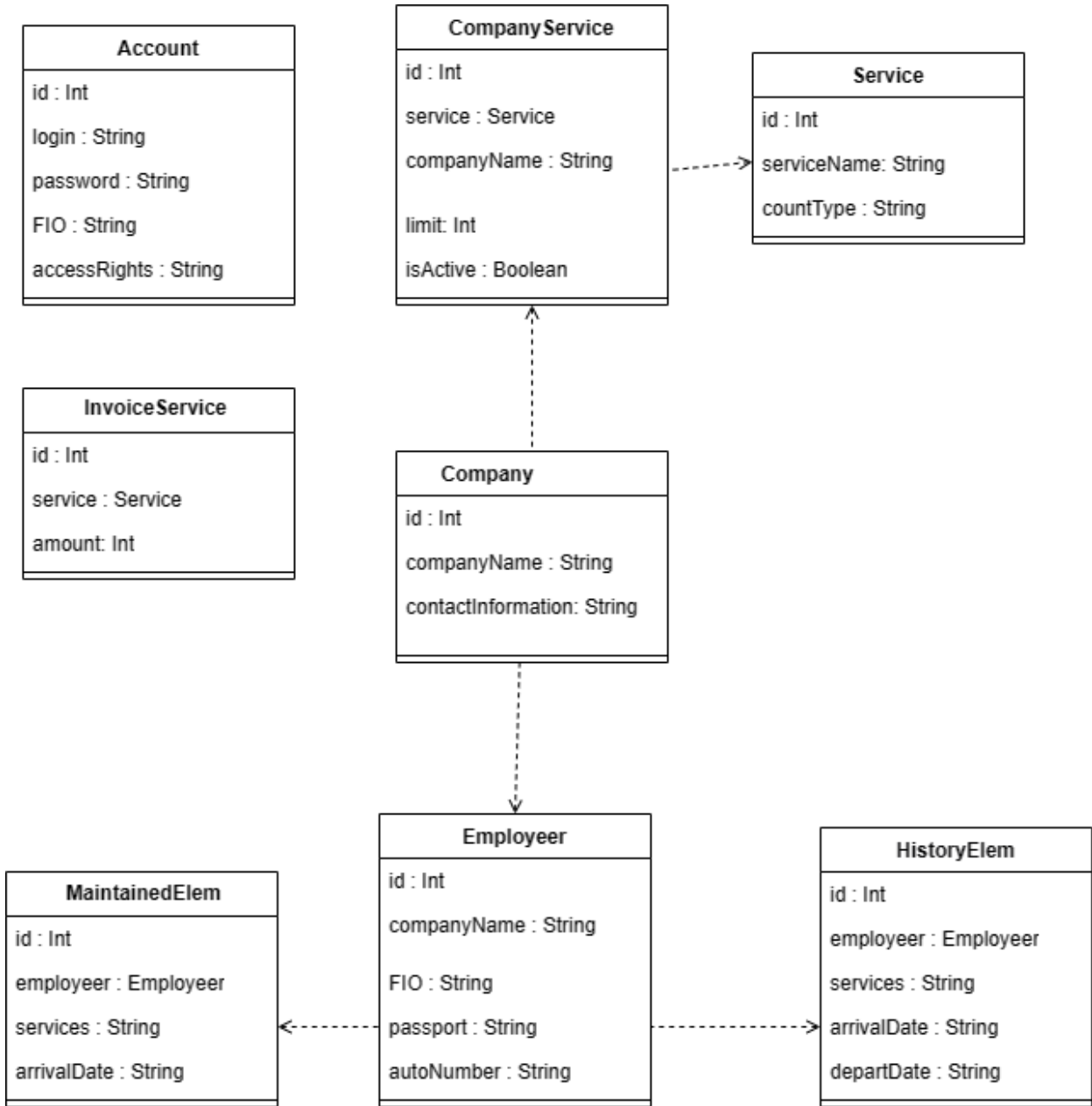


Рисунок 26 – Диаграмма классов реализованного клиентского приложения

Так как реализация переходов между экранами была проведена с помощью плагина NavGraph, была автоматически составлена схема переходов между экранами.

Данная схема представлена на рисунке 27.

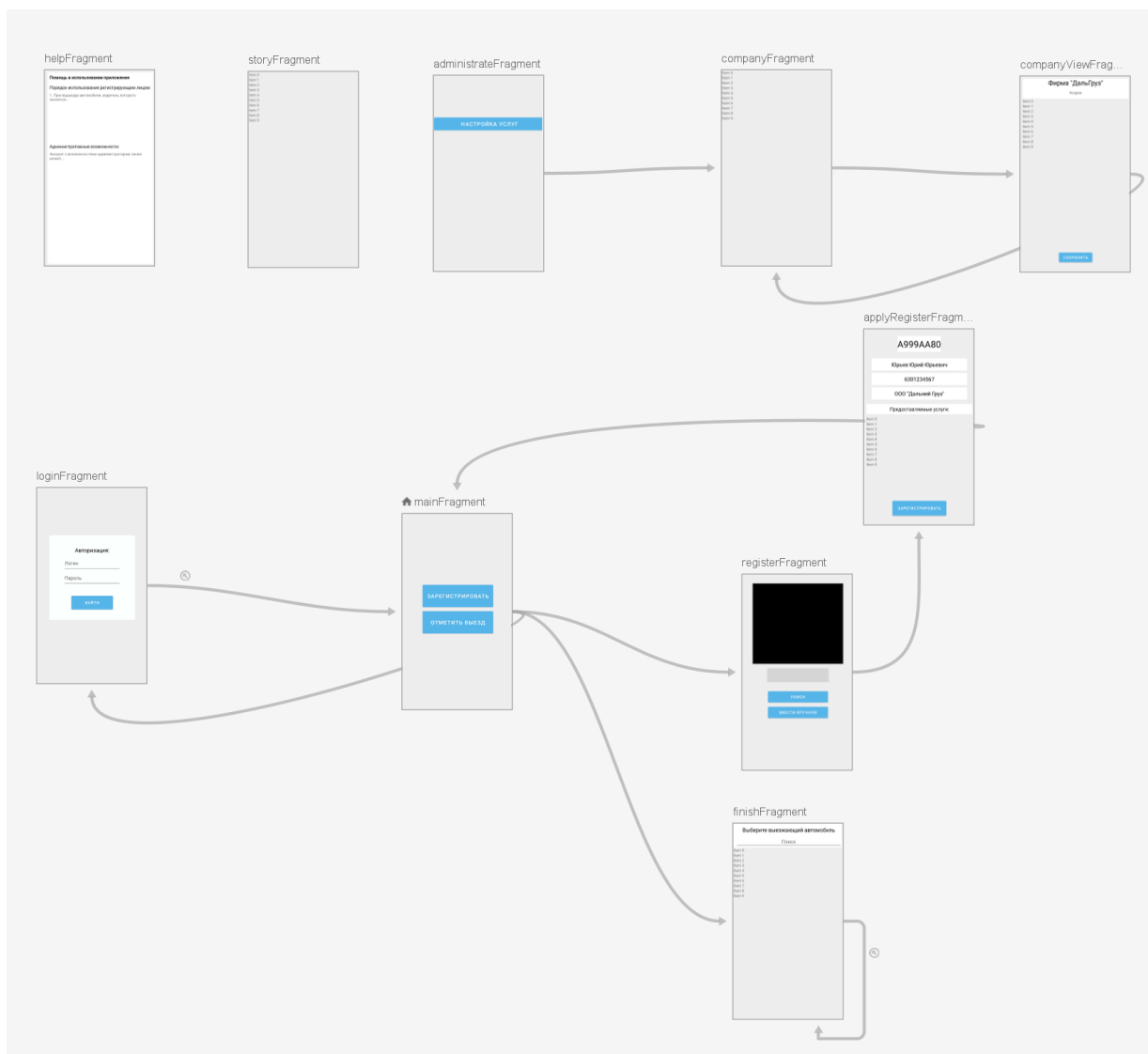


Рисунок 27 – Схема переходов между экранами nav_graph

На данной схеме можно увидеть переходы, которые реализованы в приложении при выполнении определённых действий и отмечен начальный экран. Перемещение между несоединёнными экранами происходит при помощи выдвижного меню слева. Как видно из схемы, всего в приложении реализовано 10 страниц.

4 Контроль качества программного обеспечения

Тестирование – важный технологический процесс, отвечающий за правильное выполнение функций той или иной системы. Обычно тестирование выполняется при анализе программного средства и сопутствующей документации как для выявления дефектов, так и для повышения качества продукта [10].

4.1 Функциональное тестирование

Функциональное тестирование методом черного ящика – это тестирование функциональных возможностей ПО, в условиях, когда внутренняя структура и реализация неизвестна [11]. Обычно, функциональное тестирование позволяет наиболее подобно повторить поведение конечного пользователя, отчего процесс тестирования становится наиболее эффективным. Поэтому, данный вид тестирования больше всего подходит для данной работы.

4.2 Тестирование серверной части приложения

Для серверной части приложения было проведено функциональное тестирование согласно основным вариантам использования. Таблица с результатами теста представлена в таблице 13. Результаты тестирования представлены в приложении В.

Таблица 13 – Функциональные тесты серверной части приложения

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Полученный результат
1	Проверка выдачи списка клиентов	По заданному URL отправляется GET-запрос	URL: http://localhost:8000/api/v1/employees_list	Список клиентов успешно получен	Пройден (рисунок В.1)
2	Добавление обслуживаемого клиента	По заданному URL отправляется POST-запрос	URL: http://192.168.35.156:8000/api/v1/maintaineds_list Данные: “employeeer”:6 “services”: {...} “arrivalDate”: “”	Элемент добавлен в базу данных	Пройден (рисунок В2)

Продолжение таблицы 13

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Полученный результат
3	Добавление уже существующего обслуживаемого клиента	По заданному URL отправляется POST-запрос	URL: http://192.168.35.156:8000/api/v1/maintaineds_list Данные: "employeeer":6 "services": {...} "arrivalDate": ""	Передаётся код выполнения 226, элемент не добавляется в базу данных	Пройден (рисунок В.3)
4	Удалить обслуживаемого клиента	По заданному URL отправляется DELETE-запрос	URL:http://192.168.35.156:8000/api/v1/maintaineds_list/30	Элемент успешно удалён	Пройден (рисунок В.4)
5	Проверка выдачи списка компаний	По заданному URL отправляется GET-запрос	URL: http://192.168.35.156:8000/api/v1/companies_list	Список успешно получен	Пройден (рисунок В.5)

По результатам проведения тестов можно увидеть, что сервер корректно обрабатывает запросы, в том числе при необходимости и в случае некорректного ввода.

4.3 Тестирование клиентской части приложения

Для основных функций, полученных из диаграммы вариантов использования, были проведены тесты. Их перечень представлен в таблице 14.

Таблица 14 – Функциональное тестирование клиентского приложения

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Актуальный результат
1	Проверка успешной авторизации в приложение	При авторизации вводится верный логин и пароль	Логин: admin Пароль: admin	Удаётся успешно войти в систему	Пройден (рисунок В.6)
2	Проверка неудачной авторизации в приложение	При авторизации вводится неправильный логин и пароль	Логин: admin Пароль: nimda	Уведомление об ошибке входа	Пройден (рисунок В.7)
3	Проверка сканирования автомобильного номера камерой	Камера наводится на номер авто и проходит несколько секунд, пока полученные данные не появятся в поле на экране	Фото номера авто: m999mm26	Номер успешно прочитан	Пройден (рисунок В.8)
4	Поиск по существующему номеру автомобиля	Нажимается кнопка «Поиск» при введённом номере клиента, добавленного в базу данных и не обслуживаемого в данный момент	Номер: M999MM26	Клиент успешно найден	Пройден (рисунок В.9)

Продолжение таблицы 14

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Актуальный результат
5	Отметить выезжающего клиента	Клиент выбирается в списке обслуживаемых клиентов и подтверждается намерение отметить его	Клиент: M999MM26	Клиент успешно отмечен, запись помещена в историю	Пройден (рисунок В.10)
6	Попытаться открыть меню администратора с аккаунта без прав доступа	С аккаунта без прав доступа нажимается элемент меню «Администрирование»	Аккаунт: user	Пользователю запрещён доступ к элементам	Пройден (рисунок В.11)
7	Проверка изменения сервисов, предоставляемых компании	В списке сервисов компании «Грузоперевозки России» отключается душ	Отсутствуют	Сервис «Душ» более не появляется у клиентов данной компании при регистрации	Пройден (рисунок В.12)

Исходя из результатов тестирования, можно сделать вывод, что при разработке успешно выполнены все необходимые функциональные требования, включая обработку исключительных ситуаций.

4.4 Анализ качества кода

4.4.1 Измерение устойчивости и абстрактности кода

Метрика устойчивости является мерой работы, которую нужно сделать для изменения других зависимых компонент.

Выделим компоненты серверного приложения, над которыми будет проводиться измерение:

- компонент логики обработки классов, хранящих информацию (Представлен файлом `serializers.py`);
- компонент обработки запросов к серверу (представлен файлом `views.py`).

Формула вычисления позиционной устойчивости представлена ниже:

$$I = \frac{Ce}{Ca + Ce'}$$

где I – мера неустойчивости в диапазоне от $[0; 1]$;

C_e – количество классов вне компонента, при этом зависящих от классов внутри этого компонента;

C_a – количество классов внутри данного компонента, зависящих от классов вне компонента.

Для каждой компоненты была просчитана величина I – метрика неустойчивости:

$$I_1 = \frac{9}{10+9} = 0,47;$$

$$I_2 = \frac{0}{9+0} = 0.$$

Как видно в расчётах, значение I_1 является средним, а благодаря тому, что у второй компоненты нет зависимых внешних классов, её метрика устойчивости оценивается как максимальная.

Для вычисления меры абстрактности используется формула, приведённая ниже:

$$A = \frac{Na}{Nc},$$

где A – мера абстрактности в диапазоне от 0 до 1, в котором 0 – полное отсутствие абстрактных классов в компоненте, а 1 – в компоненте только абстрактные классы;

Nc – количество классов в компоненте;

Na – количество абстрактных классов в компоненте.

Для всех компонентов приложения была вычислена мера абстрактности:

$$A_1 = \frac{0}{10} = 0;$$

$$A_2 = \frac{0}{9} = 0.$$

Таким образом, выявлены метрики, на основе которых можно построить главную последовательность.

4.4.2 Главная последовательность

После вычисления мер устойчивости и абстрактности компонент приложения можно построить график, вертикальная ось которого будет являться мерой абстрактности (A), а горизонтальная ось – мерой неустойчивости (I) (Рисунок 28).

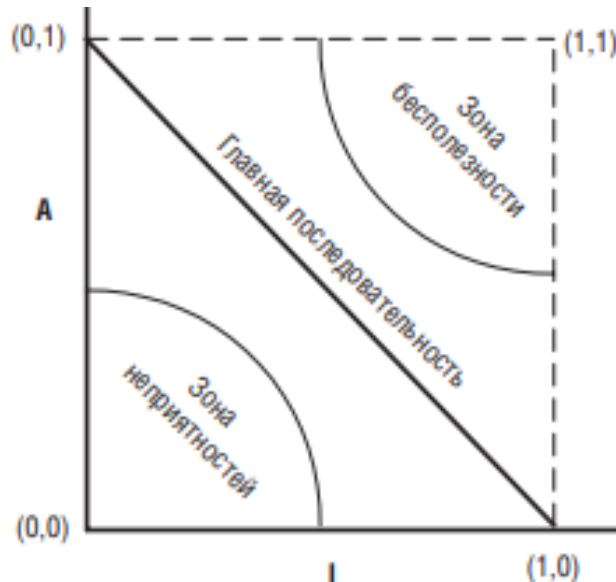


Рисунок 28 – График устойчивости и абстрактности

На данном рисунке отмечены 2 критические области:

Зона неприятностей – В этой зоне находятся компоненты очень устойчивые и конкретные. Это означает, что они трудно изменяются из-за большого количества зависимостей, и их нельзя легко расширить, потому что они не используют абстракции. Такие компоненты могут стать источником проблем при изменениях в системе [12].

Зона бесполезности – В этой зоне находятся компоненты с высокой абстрактностью и низким количеством входящих зависимостей. Обычно это устаревший и забытый код, который не используется активно в системе.

Оптимально, компоненты должны находиться на так называемой «главной последовательности» линии, проходящей от точки $(0,1)$ до точки $(1,0)$ на графике. Это означает, что они имеют сбалансированное соотношение абстрактности и устойчивости, что делает их хорошо спроектированными и адаптируемыми к изменениям.

Для вычисления количественной оценки используется формула вычисления расстояния до главной последовательности:

$$D = \left| \frac{(A + I - 1)}{\sqrt{2}} \right|,$$

где A – мера абстрактности;

I – мера устойчивости.

Было вычислено расстояние до главной последовательности:

$$D_1 = \left| \frac{0 + 0.47 - 1}{\sqrt{2}} \right| = 0.37,$$

$$D_2 = \left| \frac{0 + 0 - 1}{\sqrt{2}} \right| = 0.707.$$

Для полученных логических компонент был построен следующий график (рисунок 29).

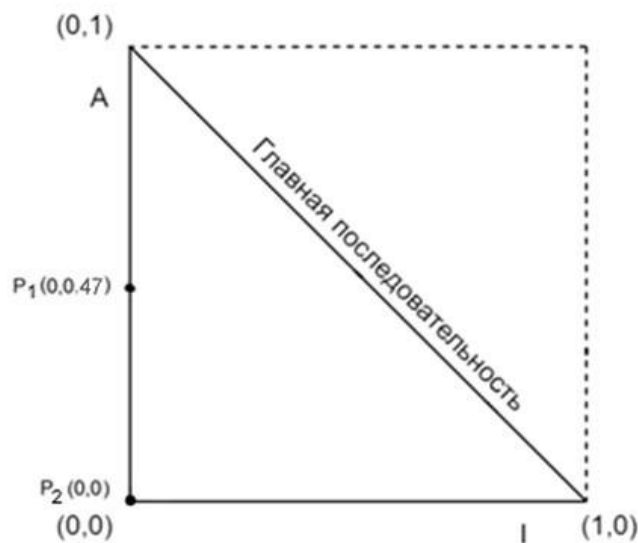


Рисунок 29 – Графическое отображение метрик на графике

Как видно по графику, первая компонента находится в оптимальном положении, а расстояние до главной невелико, что говорит о подходящем отношении абстрактности к устойчивости данной компоненты. Вторая же компонента оказалась удалена и попала в зону неприятностей, но это связано с тем, что в данной компоненте находятся все необходимые реализации

абстрактных классов предоставляемого API, из-за чего нет необходимости добавлять большую абстрактность или зависимые внешние файлы.

4.5 Планирование разработки и оценка бюджета

Управление проектами заключается в составлении плана и отслеживании выполнения работ по нему. Соответственно, чем лучше план проекта, чем более аккуратно он составлен, тем легче потом выполнять проектные работы и удачно завершить проект [13].

Планирование разработки приложения и оценка бюджета выполнялись в программе управления проектами Microsoft Office Project. Список задач, отмеченных при планировании, представлен на рисунке 30.

	Название задачи ▼	Длительность ▼	Начало ▼	Окончание ▼	Предшественники ▼
0	▲ Проект1	28 дней	Ср 19.04.23	Пт 26.05.23	
1	Начало проекта	0 дней	Ср 19.04.23	Ср 19.04.23	
2	▲ Анализ области и разработка требований	4 дней	Ср 19.04.23	Пн 24.04.23	
3	Изучение предметной области	1 день	Ср 19.04.23	Ср 19.04.23	1
4	Анализ аналогичных решений	1 день	Чт 20.04.23	Чт 20.04.23	3
5	Анализ функциональных требований	1 день	Пт 21.04.23	Пт 21.04.23	4
6	Анализ нефункциональных требований	1 день	Пн 24.04.23	Пн 24.04.23	5
7	Анализ области и разработка требований завершена	0 дней	Пн 24.04.23	Пн 24.04.23	6
8	▲ Проектирование приложения	7 дней	Вт 25.04.23	Ср 03.05.23	
9	Определения архитектуры приложения	1 день	Вт 25.04.23	Вт 25.04.23	7
10	▲ Проектирование структуры данных	3 дней	Ср 26.04.23	Пт 28.04.23	
11	Проектирование концептуальной модели базы данных	1 день	Ср 26.04.23	Ср 26.04.23	9
12	Проектирование логической модели базы данных	1 день	Чт 27.04.23	Чт 27.04.23	11
13	Проектирование физической модели базы данных	1 день	Пт 28.04.23	Пт 28.04.23	12
14	Проектирование структуры данных завершено	0 дней	Пт 28.04.23	Пт 28.04.23	13
15	Проектирование программных средств	1 день	Пн 01.05.23	Пн 01.05.23	14
16	Проектирование интерфейса	2 дней	Вт 02.05.23	Ср 03.05.23	15
17	Проектирование приложения завершено	0 дней	Пн 01.05.23	Пн 01.05.23	15
18	▲ Реализация приложения	16 дней	Вт 02.05.23	Вт 23.05.23	
19	Разработка и наполнение базы данных	2 дней	Вт 02.05.23	Ср 03.05.23	17
20	Разработка клиентской части приложения	6 дней	Чт 04.05.23	Чт 11.05.23	19
21	Разработка серверной части приложения	4 дней	Ср 17.05.23	Пн 22.05.23	20НН
22	Отладка приложения	1 день	Вт 23.05.23	Вт 23.05.23	21;20
23	Реализация приложения завершена	0 дней	Вт 23.05.23	Вт 23.05.23	22
24	▲ Тестирование приложения	3 дней	Ср 24.05.23	Пт 26.05.23	
25	Функциональное тестирование	1 день	Ср 24.05.23	Ср 24.05.23	23
26	Модульное тестирование	1 день	Чт 25.05.23	Чт 25.05.23	25
27	UI-тестирование	1 день	Пт 26.05.23	Пт 26.05.23	26
28	Тестирование приложения завершено	0 дней	Пт 26.05.23	Пт 26.05.23	27
29	Конец проекта	0 дней	Пт 26.05.23	Пт 26.05.23	28

Рисунок 30 – Список задач проектирования

Как видно из рисунка, всего было выделено 29 задач в проекте, а ожидаемое время на завершение проекта составляет 28 дней.

Для того чтобы успешно выполнить проект, необходимо использовать определенные ресурсы. Список ресурсов и их стоимость представлены на рисунке 31.

		Название ресурса ▾	Тип ▾	Стандартная ставка ▾	Ставка сверхурочных ▾
1		Аналитик	Трудовой	450,00 Р/ч	700,00 Р/ч
2		Дизайнер	Трудовой	350,00 Р/ч	600,00 Р/ч
3		Программист	Трудовой	500,00 Р/ч	700,00 Р/ч
4		Тестировщик	Трудовой	350,00 Р/ч	600,00 Р/ч
5		Интернет	Затраты		

Рисунок 31 – Список ресурсов проекта

На основе поставленных задач и определённых ресурсов, была построена диаграмма Ганта, показывающая связи между задачами, время выполнения задач и используемые ресурсы. Данная диаграмма представлена на рисунке 32.

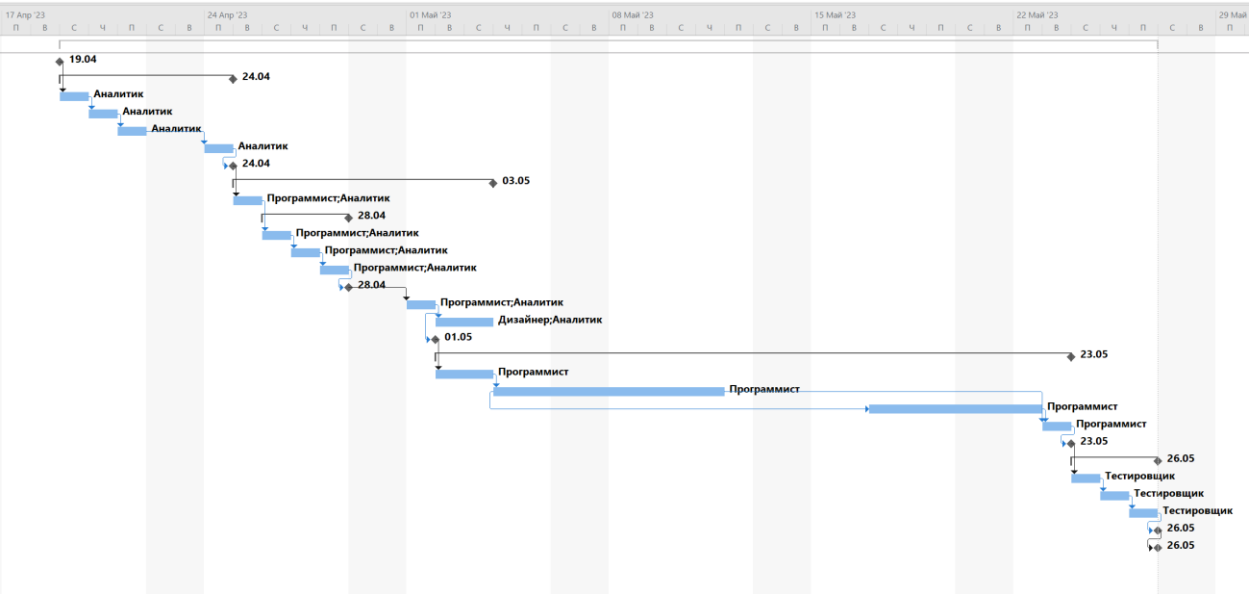
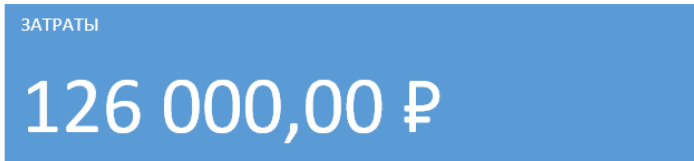


Рисунок 32 – Диаграмма Ганта

После составления проекта, определения дат и используемых ресурсов, был построен отчет, показывающий обзор затран проекта. Данный отчёт представлен на рисунке 33.

ОБЗОР ЗАТРАТ

СР 19.04.23 - ПТ 26.05.23



СОСТОЯНИЕ ЗАТРАТ

Состояние затрат для задач верхнего уровня.

Название	Фактически е затраты	Оставшиеся затраты	Базовые затраты	Затраты	Отклонение по стоимости
Начало проекта	0,00 ₹	400,00 ₹	0,00 ₹	400,00 ₹	400,00 ₹
Анализ области и разработка требований	0,00 ₹	14 400,00 ₹	0,00 ₹	14 400,00 ₹	14 400,00 ₹
Проектирование приложения	0,00 ₹	50 800,00 ₹	0,00 ₹	50 800,00 ₹	50 800,00 ₹
Реализация приложения	0,00 ₹	52 000,00 ₹	0,00 ₹	52 000,00 ₹	52 000,00 ₹
Тестирование приложения	0,00 ₹	8 400,00 ₹	0,00 ₹	8 400,00 ₹	8 400,00 ₹

Рисунок 33 – Отчет затрат

В данном отчёте видно, что предполагаемые затраты на данный проект составят 126000 рублей. На основании информации, полученной в результате планирования, можно сделать вывод о целесообразности разработки данного приложения.

Заключение

В результате выполнения выпускной квалификационной работы было разработано, протестировано и проанализировано мобильного приложение для регистрации сотрудников грузовой компании в отеле.

В процессе выполнения анализа требований была выявлена нехватка подобного приложения в открытом доступе и актуальность разрабатываемого приложения. Также, были выявлены функциональные и нефункциональные требования. Анализ приложений-аналогов позволил выяснить преимущества и недостатки подобных приложений и дать им оценку на основании данных критериев. Были определены следствия для реализации приложения.

На стадии проектирования приложения разработана концептуальная, логическая и физическая модели базы данных, а также спроектирована архитектура программного продукта. В итоге, было решено использовать клиент-серверное взаимодействие мобильного приложения и базы данных при использовании шаблона REST API. Также, были спроектированы макеты интерфейса.

В ходе выполнения этапа реализации была разработана серверная часть приложения, включающая в себя базу данных и взаимодействие с ней и клиентская часть приложения.

В ходе выполнения тестирования была выявлена функциональность и готовность продукта к использованию.

Для проекта осуществлено структурное и календарное планирование, а также оценена стоимость разработки приложения. Результатом стала диаграмма Ганта и отчет по проделанной работе.

В итоге, были выполнены все пункты поставленного задания. В результате выполнения работы реализовано приложения для регистрации сотрудников грузовых компаний в отеле согласно всем поставленным требованиям.

Список использованных источников

1. АО «Кодекс» Сравнительные испытания мобильных приложений для смартфонов // Электронный фонд правовых и нормативно-технических документов [Электронный ресурс]. URL: <https://docs.cntd.ru/document/1200159701> (дата обращения: 13.04.2023).
2. «Дорожная Сеть» «О нас» // Личный сайт компании [Электронный ресурс]. URL: <https://dornet.ru/content/about> (дата обращения 14.04.2023).
3. Фаулер, М. Архитектура корпоративных программных приложений / М. Фаулер – Москва. : «Вильямс», 2006. – 260 с.
4. «Skillbox» REST API: что это такое и как работает // Платформа образовательных услуг [Электронный ресурс]. URL: <https://skillbox.ru/media/code/rest-api-cto-eto-takoe-i-kak-rabotaet/> (дата обращения: 20.04.2023)
5. Ёсу, М. Т. Принципы организации распределенных баз данных / М. Т. Ёсу – М. : ДМК Пресс, 2021. – 672 с.
6. Дейт, К. Дж. Введение в системы баз данных, 8-е издание / К. Дж. Дейт – М.: «Вильямс», 2019. – 1328 с.
7. Швецов, В.И. Базы данных. Учебное пособие. / В.И. Швецов – Нижний Новгород: Изд-во ННГУ, 2004. – 217 с.
8. «Habr» Моделирование данных: обзор // Экосистема для развития людей, вовлеченных в IT [Электронный ресурс] [Электронный ресурс]. URL: <https://habr.com/ru/post/556790/> (дата обращения 31.05.2023).
9. Фаулер, М. UML. Основы, 3-е издание. / М. Фаулер – СПб: Символ Плюс, 2019. – 192 с.
10. Куликов, С. Тестирование программного обеспечения. Базовый курс, 3-е издание / С. Куликов – М.: Издательство МГТУ им. Н. Э. Баумана, 2020. – 301 с.
11. Бейзер, Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем / Б. Бейзер – СПб.: Питер, 2004 – 318 с.

12. «Habr» «Секретный ингредиент для оценки архитектуры кода» // Экосистема для развития людей, вовлеченных в IT [Электронный ресурс] <https://habr.com/ru/articles/726338/> (дата обращения: 13.06.2023)

13. Т. П. Брусенцова, Управление проектами в Microsoft Project: учебно-методическое пособие для студентов по направлению специальности «Информационные системы и технологии (издательско-полиграфический комплекс)» / Т. П. Брусенцова, В. В. Смелов. – Минск: БГТУ, 2011. – 160 с.

Приложение А
ГЛОССАРИЙ
(справочное)

API – механизмы, позволяющие двум программным компонентам взаимодействовать друг с другом, используя набор определений и протоколов. API упрощает процесс программирования при создании приложений, абстрагируя базовую реализацию и предоставляя только объекты или действия, необходимые разработчику.

REST API – архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы.

RETROFIT – REST клиент для Java и Android. Он позволяет легко получить и загрузить JSON (или другие структурированные данные) через веб-сервис на основе REST.

ML Kit – это SDK для мобильных устройств, который в настоящее время доступен на Android и iOS. Он использует возможности машинного обучения Google в мобильных приложениях для решения проблем из реальной жизни. ML Kit поможет добиться успеха во многих задачах.

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных предназначенный для передачи гипертекстовых документов.

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

СУБД – система управления базами данных.

PostgreSQL – свободная объектно-реляционная система управления базами данных, созданная на основе некоммерческой СУБД Postgres, разработанной как open-source проект в Калифорнийском университете в

Беркли. PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2011

JSON – JavaScript Object Notation, текстовый формат, независимый от языка реализации и используемый для взаимосвязи между разными программами и хранения данных.

Приложение Б
ЛИСТИНГ ПРОГРАММЫ
(рекомендуемое)

Серверная часть проекта:

Листинг Б.1 – Файл models.py

```

from django.db import models
from datetime import datetime

# Create your models here.
# class A(models.Model):

class Accounts(models.Model):
    login = models.CharField(max_length=30)
    password = models.CharField(max_length=30)
    FIO = models.CharField(max_length=100)
    accessRights = models.CharField(max_length=30)

    def __str__(self):
        return self.login

class Service(models.Model):
    serviceName = models.CharField(max_length=100)
    countType = models.CharField(max_length=30)

    def __str__(self):
        return self.serviceName

class Company(models.Model):
    companyName = models.CharField(max_length=30, unique=True, db_index=True)
    contactInformantion = models.CharField(max_length=100)

    def __str__(self):
        return self.companyName

class CompanyServices(models.Model):
    service = models.ForeignKey('Service', on_delete=models.PROTECT)
    companyName = models.ForeignKey('Company', to_field='companyName',
on_delete=models.CASCADE)
    limit = models.IntegerField(default=0)
    isActive = models.BooleanField(default=True)

    def __str__(self):
        return type(self.service).__str__(self.service) + ' ' +
type(self.companyName).__str__(self.companyName)

class Employeeer(models.Model):
    companyName = models.ForeignKey('Company', to_field='companyName',
on_delete=models.CASCADE)
    FIO = models.CharField(max_length=100)
    passport = models.CharField(max_length=10)
    autoNumber = models.CharField(max_length=10)

    def __str__(self):
        return self.FIO

class Maintained(models.Model):
    employeeer = models.OneToOneField('Employeeer', on_delete=models.PROTECT)
    services = models.TextField(default="dict")
    arrivalDate = models.CharField(default=datetime.now().strftime("%Y.%m.%d
%H:%M"), max_length=16)

```

```

    def __str__(self):
        return type(self.employer).__str__(self.employer) + ' ' +
self.arrivalDate.__str__()

class HistoryElem(models.Model):
    employer = models.ForeignKey('Employee', on_delete=models.PROTECT)
    services = models.TextField(default="dict")
    arrivalDate = models.CharField(max_length=16)
    departDate = models.CharField(default=datetime.now().strftime("%Y.%m.%d
%H:%M"),max_length=16)

    def __str__(self):
        return type(self.employer).__str__(self.employer) + ' ' +
self.arrivalDate + ' ' + self.departDate

class InvoiceService(models.Model):
    service = models.ForeignKey('Service', on_delete=models.DO_NOTHING)
    amount = models.IntegerField(default=1)

    def __str__(self):
        return type(self.service).__str__(self.service)

```

Листинг Б.2 – Файл serializers.py

```

from rest_framework import serializers
from .models import *

class AccountsSerializer(serializers.ModelSerializer):
    class Meta:
        model = Accounts
        fields = ('id', 'login', 'password', 'FIO', 'accessRights')

class ServiceSerializer(serializers.ModelSerializer):
    class Meta:
        model = Service
        fields = ('id', 'serviceName', 'countType')

class CompanySerializer(serializers.ModelSerializer):
    class Meta:
        model = Company
        fields = ('id', 'companyName', 'contactInformantion')

class EmployeeSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employee
        fields = ('id', 'companyName', 'FIO', 'passport', 'autoNumber')

class CompanyServicesSerializer(serializers.ModelSerializer):
    service = ServiceSerializer(read_only=True)

    class Meta:
        model = CompanyServices
        fields = ['id', 'service', 'companyName', 'limit', 'isActive']

```

```

class OneCompanyServiceSerializer(serializers.ModelSerializer):
    service = ServiceSerializer(read_only=True)

    class Meta:
        model = CompanyServices
        fields = ['id', 'service', 'companyName', 'limit', 'isActive']

class MaintainedSerializer(serializers.ModelSerializer):
    employeeer = EmployeeerSerializer(read_only=True)

    class Meta:
        model = Maintained
        fields = ('id', 'employeeer', 'services', 'arrivalDate')

class MaintainedPOSTSerializer(serializers.ModelSerializer):
    employeeer = EmployeeerSerializer(read_only=True)

    class Meta:
        model = Maintained
        fields = ('id', 'employeeer', 'services', 'arrivalDate')

class HistoryElemSerializer(serializers.ModelSerializer):
    employeeer = EmployeeerSerializer(read_only=True)

    class Meta:
        model = HistoryElem
        fields = ('id', 'employeeer', 'services', 'arrivalDate', 'departDate')

class ServicesSerializer(serializers.ModelSerializer):

    class Meta:
        model = Service
        fields = ('service', 'amount')

```

Листинг Б.3 – Файл views.py

```

from rest_framework import generics, viewsets, status
from rest_framework.response import Response
from rest_framework.views import APIView

from .serializers import *

# Create your views here.
"""
def index(request):
    return HttpResponse('Первая страница информации')

def apply_register_data(request):
    if (request.GET):
        print(request.GET)
    return HttpResponse('JSON файл списка предоставляемых услуг')

def companies_data(request):
    if (request.GET):
        print(request.GET)
    return HttpResponse('JSON файл списка компаний')

def company_view_data(request):
    if (request.GET):
        print(request.GET)
    return HttpResponse('JSON файл списка сервисов компании')

def finish_data(request):

```

```

    if (request.GET):
        print(request.GET)
    return HttpResponse('JSON файл списка обслуживаемых клиентов')

def story_data(request):
    if (request.GET):
        print(request.GET)
    return HttpResponse('JSON файл списка истории регистраций')
"""

# получает сервисы компании по имени компании
class CompanyServicesAPIView(APIView):
    def get(self, request, receivedComName):
        # list =
        CompanyServices.objects.get(companyName=getattr(Company.objects.get(id=fk), 'companyName'))
        list = CompanyServices.objects.filter(

companyName=getattr(Company.objects.get(companyName=receivedComName),
'companyName'))
        serializer = CompanyServicesSerializer(list, many=True)
        return Response(serializer.data)

# получает сервисы компании по id компании
class OneCompanyServiceAPIView(APIView):
    def get(self, request, pk):
        elem = CompanyServices.objects.get(id=pk)
        serializer = OneCompanyServiceSerializer(elem)
        # serializer_class = CompanyServicesSerializer
        return Response(serializer.data)

    def post(self, request, pk):
        elem = CompanyServices.objects.get(id=pk)
        elem.limit = request.data['limit']
        elem.isActive = request.data['isActive']
        elem.save(update_fields=["limit", "isActive"])
        return Response(status=status.HTTP_200_OK)

class AccountsAPIView(generics.ListAPIView):
    queryset = Accounts.objects.all()
    serializer_class = AccountsSerializer

class ServiceAPIView(generics.ListAPIView):
    queryset = Service.objects.all()
    serializer_class = ServiceSerializer

class CompanyAPIView(generics.ListAPIView):
    queryset = Company.objects.all()
    serializer_class = CompanySerializer

class EmployeeerAPIView(generics.ListAPIView):
    queryset = Employeeer.objects.all()
    serializer_class = EmployeeerSerializer

class OneEmployeeerAPIView(APIView):
    def get(self, request, receivedNumber):
        elem = Employeeer.objects.get(autoNumber=receivedNumber)

```

```

        serializer = EmployeeerSerializer(elem)
        # serializer_class = CompanyServicesSerializer
        return Response(serializer.data)

class MaintainedAPIView(APIView):
    def get(self, request):
        queryset = Maintained.objects.all()
        serializer = MaintainedSerializer(queryset, many=True)
        return Response(serializer.data)

    def post(self, request):
        if
Maintained.objects.filter(employeeer_id=int(request.data['employeeer'])).exists
():
            return Response(status=status.HTTP_226_IM_USED)
        post_new = Maintained.objects.create(
            employeeer_id=request.data['employeeer'],
            services=request.data['services'],
            arrivalDate=datetime.now().strftime("%Y.%m.%d %H:%M")
        )
        return Response(status=status.HTTP_200_OK)

    def delete(self, request, id_del):
        Maintained.objects.get(id=id_del).delete()
        return Response(status=status.HTTP_200_OK)

class HistoryElemAPIView(APIView):
    def get(self, request):
        queryset = HistoryElem.objects.all()
        serializer = HistoryElemSerializer(queryset, many=True)
        return Response(serializer.data)

    def post(self, request):
        post_new = HistoryElem.objects.create(
            employeeer_id=request.data['employeeer'],
            services=request.data['services'],
            arrivalDate=request.data['arrivalDate'],
            departDate=datetime.now().strftime("%Y.%m.%d %H:%M")
        )
        return Response(status=status.HTTP_200_OK)

```

Листинг Б.4 – Файл urls.py

```

from django.contrib import admin
from django.urls import path

from info.views import *

urlpatterns = [
    path('admin/', admin.site.urls),
    # path('', include('info.urls'))
    path('api/v1/company_services_list/<str:receivedComName>',
CompanyServicesAPIView.as_view()),
    path('api/v1/company_service/<int:pk>',
OneCompanyServiceAPIView.as_view()),
    path('api/v1/accounts_list', AccountsAPIView.as_view()),
    path('api/v1/services_list', ServiceAPIView.as_view()),
    path('api/v1/companies_list', CompanyAPIView.as_view()),
    path('api/v1/employeeers_list', EmployeeerAPIView.as_view()),
    path('api/v1/getemployeeer/<str:receivedNumber>',
OneEmployeeerAPIView.as_view()),
    path('api/v1/maintaineds_list', MaintainedAPIView.as_view()),

```

```

        path('api/v1/maintaineds_list/<int:id_del>',
MaintainedAPIView.as_view()),
        path('api/v1/history_list', HistoryElemAPIView.as_view())
]

```

Листинг Б.5 – Файл settings.py

```

from pathlib import Path

from django.conf.locale.es import formats as es_formats

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-
iu*=2j+9qzv4#c$m(hp9sij&&vu8+t8uu%vb(&c5un8h)5#3w#'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['*']

es_formats.DATETIME_FORMAT = "d M Y H:i:s"

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'info.apps.InfoConfig',
    'rest_framework',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'reempldata.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

```

```

]

WSGI_APPLICATION = 'resempldata.wsgi.application'

# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'regemp_base',
        'USER': 'postgres',
        'PASSWORD': 'admin',
        'HOST': 'localhost',
        'PORT': '5432'
    }
}

# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
        'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
        'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
        'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
        'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = 'ru'

TIME_ZONE = 'Europe/Moscow'

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

Клиентская часть проекта:

Листинг Б.6 – Файл MainApi.kt

```
package com.example.emp_reg_02.api

import com.example.emp_reg_02.entities.Account
import com.example.emp_reg_02.entities.Company
import com.example.emp_reg_02.entities.CompanyService
import com.example.emp_reg_02.entities.Employeeer
import com.example.emp_reg_02.entities.HistoryElem
import com.example.emp_reg_02.entities.MaintainedElem
import com.example.emp_reg_02.entities.Service
import okhttp3.RequestBody
import okhttp3.ResponseBody
import retrofit2.Response
import retrofit2.http.Body
import retrofit2.http.DELETE
import retrofit2.http.GET
import retrofit2.http.POST
import retrofit2.http.Path

interface MainApi {
    @GET("api/v1/accounts_list")
    suspend fun getAccounts(): ArrayList<Account>

    @GET("api/v1/services_list")
    suspend fun getServices(): ArrayList<Service>

    @GET("api/v1/company_services_list/{comName}")
    suspend fun getCompanyServicesListA(@Path("comName") comName: String):
    ArrayList<CompanyService>

    @GET("api/v1/company_services_list/{id}")
    suspend fun getCompanyServicesListB(@Path("id") id: Int):
    ArrayList<CompanyService>

    @GET("api/v1/company_service/{id}")
    suspend fun getOneCompanyService(@Path("id") id: Int): CompanyService

    @GET("api/v1/companies_list")
    suspend fun getCompaniesList(): ArrayList<Company>

    @GET("api/v1/history_list")
    suspend fun getHistoryList(): ArrayList<HistoryElem>

    @GET("api/v1/getemployeeer/{receivedNumber}")
    suspend fun getEmployeeerByNumber(@Path("receivedNumber") receivedNumber:
    String): Employeeer

    @GET("api/v1/maintaineds_list")
    suspend fun getMaintainedsList(): ArrayList<MaintainedElem>

    @POST("/api/v1/maintaineds_list")
    suspend fun createEmployee(@Body requestBody: RequestBody):
    Response<ResponseBody>

    @DELETE("/api/v1/maintaineds_list/{id}")
    suspend fun deleteMaintained(@Path("id") id: Int): Response<ResponseBody>

    @POST("api/v1/history_list")
    suspend fun addStoryElem(@Body historyElem: RequestBody):
    Response<ResponseBody>

    @POST("api/v1/company_service/{pk}")
```

```

        suspend fun UpdateCompanyService(@Body updateData: RequestBody,
@Path("pk") pk: Int): Response<ResponseBody>
    }

```

Листинг Б.7 – Файл Account.kt

```

package com.example.emp_reg_02.entities

data class Account(
    val id: Int,
    val login: String,
    val password: String,
    val FIO: String,
    val accessRights: String
)

```

Листинг Б.8 – Файл Company.kt

```

package com.example.emp_reg_02.entities

data class Company(
    val id: Int,
    val companyName: String,
    val contactInformation: String
)

```

Листинг Б.9 – Файл CompanyService.kt

```

package com.example.emp_reg_02.entities

data class CompanyService(

    var id: Int,
    var service: Service,
    var companyName: String,
    var limit: Int,
    var isActive: Boolean

)

```

Листинг Б.10 – Файл Employee.kt

```

package com.example.emp_reg_02.entities

import android.os.Parcelable
import kotlinx.parcelize.Parcelize

@Parcelize
data class Employee(
    val id: Int = -1,
    val companyName: String = "",
    val FIO: String = "",
    val passport: String = "",
    val autoNumber: String = ""
) : Parcelable

```

Листинг Б.11 – Файл HistoryElem.kt

```

package com.example.emp_reg_02.entities

data class HistoryElem(
    val id: Int,

```

```

        val employeeer: Employeeer,
        val services: String,
        val arrivalDate: String,
        val departDate: String,
    )

```

Листинг Б.12 – Файл InvoiceService.kt

```

package com.example.emp_reg_02.entities

data class InvoiceService(
    val id: Int,
    val service_id: Int,
    val company_id: Int,
    val amount: Int
)

```

Листинг Б.13 – Файл MaintainedElem

```

package com.example.emp_reg_02.entities

data class MaintainedElem(
    val id: Int,
    val employeeer: Employeeer,
    val services: String,
    val arrivalDate: String,
)

```

Листинг Б.14 – Файл Service

```

package com.example.emp_reg_02.entities

data class Service(

    var id: Int,
    var serviceName: String,
    var countType: String

)

```

Листинг Б.15 – Файл AdministrateFragment.kt

```

package com.example.emp_reg_02.view.fragments

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import com.example.emp_reg_02.R
import com.example.emp_reg_02.databinding.FragmentAdministrateBinding

class AdministrateFragment : Fragment() {

    private lateinit var mBinding: FragmentAdministrateBinding

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ) {

```

```

): View? {
    mBinding = FragmentAdministrativeBinding.inflate(layoutInflater)
    return mBinding.root
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    val nav_controller = findNavController()
    mBinding.buttonConfig.setOnClickListener {
        nav_controller.navigate(R.id.companyFragment)
    }
    activity?.title = "Меню администратора"
}
}

```

Листинг Б.16 – Файл ApplyRegisterFragment.kt

```

package com.example.emp_reg_02.view.fragments

import android.icu.text.SimpleDateFormat
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.widget.SwitchCompat
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView.ViewHolder
import com.example.emp_reg_02.R
import com.example.emp_reg_02.URLToLocale
import com.example.emp_reg_02.api.MainApi
import com.example.emp_reg_02.databinding.FragmentApplyRegisterBinding
import com.example.emp_reg_02.entities.Company
import com.example.emp_reg_02.entities.CompanyService
import com.example.emp_reg_02.entities.Employee
import com.example.emp_reg_02.entities.InvoiceService
import com.example.emp_reg_02.entities.Service
import com.example.emp_reg_02.view.RVadapters.ApplyRegisterRVAdapter
import com.google.gson.Gson
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.OkHttpClient
import okhttp3.RequestBody.Companion.toRequestBody
import okhttp3.logging.HttpLoggingInterceptor
import org.json.JSONObject
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import java.util.Date
import java.util.Locale

class ApplyRegisterFragment : Fragment() {

    private lateinit var mBinding: FragmentApplyRegisterBinding

```

```

private val adapter = ApplyRegisterRVAdapter()
private lateinit var employeeer: Employeeer
private var listCompanyServices = ArrayList<CompanyService>()
private lateinit var listCompanies: List<Company>
private var listInvoiceServices: ArrayList<InvoiceService> = ArrayList()
var flagSwitched = false

//тест
private var serList = listOf(
    CompanyService(1, Service(2, "Стоянка", "сутки"), "ООО \"ДальГруз\"",
1, true),
)

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    Log.d("mylog", "Работает ApplyRegisterFragment")

    arguments?.let { bundle ->
        employeeer = bundle.getParcelable<Employeeer>("thisEmployeeer")!!
    }
    val nav_controller = findNavController()
}

override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?, savedInstanceState:
Bundle?
): View? {
    mBinding = FragmentApplyRegisterBinding.inflate(layoutInflater)
    return mBinding.root
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    //activity?.title = "Регистрация"
    val interceptor = HttpLoggingInterceptor()
    interceptor.level = HttpLoggingInterceptor.Level.BODY
    val client =
OkHttpClient.Builder().addInterceptor(interceptor).build()

    val retrofit = Retrofit.Builder()
        //.baseUrl("http://10.0.2.2:8000").client(client) //для эмулятора
        .baseUrl(URLToLocale).client(client) //для подключенного
устройства
        .addConverterFactory(GsonConverterFactory.create()).build()
    val api = retrofit.create(MainApi::class.java)

    init()

    CoroutineScope(Dispatchers.IO).launch {
        //Найти сервисы компании по её названию
        listCompanyServices =
api.getCompanyServicesListA(employeeer.companyName)

        //убираем не активированные администратором сервисы
        for (i in listCompanyServices.indices) {
            if (!(listCompanyServices[i].isActive))
listCompanyServices.removeAt(i)
        }

        listCompanies = api.getCompaniesList()
        print("test")
        activity?.runOnUiThread {
            adapter.refreshList(listCompanyServices)
        }
    }
}

```

```

    }
}

mBinding.buttonApplyReg.setOnClickListener {
    val listInSer =
        setInvoiceServicesList() //Создаём список заказов для JSON
        строки MaintainedElem
    if (listInSer.size == 0) {

    } else {
        //val dateInString =
        Calendar.getInstance().time.toString("yyyy.MM.dd HH:mm")
        if (flagSwitched) {
            val jsonString = Gson().toJson(listInSer)

            val jsonObject = JSONObject()
            jsonObject.put("employeeer", employeeer.id.toString())
            jsonObject.put("services", jsonString)
            val requestBody =

jsonObject.toString().toRequestBody("application/json".toMediaTypeOrNull())

            CoroutineScope(Dispatchers.IO).launch {
                // Do the POST request and get response
                val response = api.createEmployee(requestBody)

                withContext(Dispatchers.Main) {
                    when (response.code()) {
                        200 -> {
                            Log.d("mylog", "Работает
ApplyRegisterFragment")

AlertDialog.Builder(requireContext()).setTitle("Уведомление")
                                .setMessage(
                                    "Клиент успешно добавлен"
                                ).setPositiveButton("Ок") { dialog, j
->

findNavController().navigate(R.id.action_applyRegisterFragment_to_mainFragmen
t)

                                }.show()
                            }

                        226 -> {
                            Log.d("mylog", "Работает
ApplyRegisterFragment")

AlertDialog.Builder(requireContext()).setTitle("Ошибка")
                                .setMessage(
                                    "Данный клиент уже есть в списке
обслуживаемых"
                                ).setPositiveButton("Ок") { dialog, j
-> }.show()
                            }

                        else -> { // Note the block
                            Log.e("RETROFIT_ERROR",
response.code().toString())

AlertDialog.Builder(requireContext()).setTitle("Ошибка")
                                .setMessage(
                                    "Ошибка при отправке данных, код
" + response.code()

                                    .toString()

```

```

                                ).setPositiveButton("Ок") { dialog, j
-> }.show()
                                }
                                }
                                }
                                } else {
AlertDialog.Builder(requireContext()).setTitle("Ошибка").setMessage(
    "При регистрации не отмечено ни одной услуги"
).setPositiveButton("Ок") { dialog, j -> }.show()
    }
    }/*
    {
        "employeeer": "7",
        "services": "[{\\"amount\\":2,\\"company_id\\":3,\\"id\\":-
1,\\"service_id\\":1}]"
    }
    */

    }

    override fun onDestroyView() {
        listCompanyServices.clear()
        listInvoiceServices.clear()
        super.onDestroyView()
    }

    fun Date.toString(format: String, locale: Locale = Locale.getDefault()):
String {
        val formatter = SimpleDateFormat(format, locale)
        return formatter.format(this)
    }

    private fun setInvoiceServicesList(): ArrayList<InvoiceService> {
        listInvoiceServices = ArrayList<InvoiceService>()
        for (i in 0 until mBinding.rcView.childCount) {
            val holder: ViewHolder =

mBinding.rcView.getChildViewHolder(mBinding.rcView.getChildAt(i))
            val txtName: TextView = holder.itemView.findViewById(R.id.name)
            val txtAmount: TextView =
holder.itemView.findViewById(R.id.amount)
            val itemSwitch: SwitchCompat =
holder.itemView.findViewById(R.id.switch_use)
            if (itemSwitch.isChecked) {
                flagSwitched = true
                Log.d("mylog", "txtAmount: " +
txtAmount.text.toString().toIntOrNull())

                if (txtAmount.text.toString().toIntOrNull() == null) {
                    Log.d("mylog", "txtAmountOrNull: " +
txtAmount.text.toString().toIntOrNull())

AlertDialog.Builder(requireContext()).setTitle("Ошибка").setMessage(
    "У услуги " +
listCompanyServices[i].service.serviceName + " количество не является числом
или не введено"
).setPositiveButton("Ок") { dialog, j -> }.show()
                return ArrayList<InvoiceService>()
            }

```

```

        if (txtAmount.text.toString().toInt() >
listCompanyServices[i].limit) {
            Log.d(
                "mylog", "txtAmountBiggerLimit: " +
txtAmount.text.toString().toIntOrNull()
            )

AlertDialog.Builder(requireContext()).setTitle("Ошибка").setMessage(
    "На услугу " +
listCompanyServices[i].service.serviceName + " ограничение: не более " +
listCompanyServices[i].limit
).setPositiveButton("Ок") { dialog, j -> }.show()
return ArrayList<InvoiceService>()
}
var companyId: Int = -1
for (item in listCompanies) {
    if
(item.companyName.equals(listCompanyServices[i].companyName)) {
        companyId = item.id
    }
}
listInvoiceServices.add(
    InvoiceService(
        id = -1,
        service_id = listCompanyServices[i].service.id,
        company_id = companyId,
        amount = txtAmount.text.toString().toInt()
    )
)
}
return listInvoiceServices
}

private fun init() = with(mBinding) {
    rcView.layoutManager = LinearLayoutManager(requireContext())
    rcView.adapter = adapter

    fieldFIO.text = employeeer.FIO
    fieldPassport.text = employeeer.passport
    fieldAutoNumber.text = employeeer.autoNumber
    fieldComName.text = employeeer.companyName
}
}

```

Листинг Б.17 – Файл CompaniesFragment

```

package com.example.emp_reg_02.view.fragments

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.emp_reg_02.URLToLocale
import com.example.emp_reg_02.api.MainApi
import com.example.emp_reg_02.databinding.FragmentCompaniesBinding
import com.example.emp_reg_02.entities.Company
import com.example.emp_reg_02.view.RVadapters.CompaniesRVAdapter
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers

```

```

import kotlinx.coroutines.launch
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class CompaniesFragment : Fragment() {

    private lateinit var mBinding: FragmentCompaniesBinding
    private val adapter = CompaniesRVAdapter()

    //тест
    private var comList = listOf(
        Company(1, "ООО \"ДальГруз\"", "контактная информация"),
        Company(2, "ИП Володчан С.В.", "контактная информация"),
        Company(3, "ООО \"МежГород\"", "контактная информация"),
        Company(4, "ООО \"Вдали\"", "контактная информация"),
        Company(5, "ИП Римов Г.П.", "контактная информация"),
    )

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        mBinding = FragmentCompaniesBinding.inflate(inflater)
        return mBinding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val nav_controller = findNavController()
        //nav_controller.navigate(R.id.companyViewFragment)
        activity?.title = "Услуги"
        rc_init()

        val interceptor = HttpLoggingInterceptor()
        interceptor.level = HttpLoggingInterceptor.Level.BODY
        val client = OkHttpClient.Builder()
            .addInterceptor(interceptor)
            .build()

        val retrofit = Retrofit.Builder()
            //.baseUrl("http://10.0.2.2:8000").client(client) //для эмулятора
            .baseUrl(URLToLocale).client(client) //для подключенного
            устройства
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        val api = retrofit.create(MainApi::class.java)

        CoroutineScope(Dispatchers.IO).launch {
            val list = api.getCompaniesList()
            activity?.runOnUiThread{
                adapter.refreshList(list)
            }
        }
    }

    private fun rc_init() = with(mBinding) {
        compRcView.layoutManager = LinearLayoutManager(requireContext())
        compRcView.adapter = adapter
    }
}

```

Листинг Б.18 – Файл CompanyViewFragment

```

package com.example.emp_reg_02.view.fragments

import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.EditText
import androidx.*
import com.example.emp_reg_02.*
import kotlinx.coroutines.*
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.OkHttpClient
import okhttp3.RequestBody.Companion.toRequestBody
import okhttp3.logging.HttpLoggingInterceptor
import org.json.JSONObject
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class CompanyViewFragment : Fragment() {
    private lateinit var mBinding: FragmentCompanyViewBinding
    private val adapter = CompanyViewRVAdapter()
    lateinit var serList: ArrayList<CompanyService>
    lateinit var name_company: String
    var list = ArrayList<CompanyService>()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        arguments?.let { bundle ->
            name_company = bundle.getString("name_company").orEmpty()
        }
    }
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        mBinding = FragmentCompanyViewBinding.inflate(layoutInflater)
        return mBinding.root
        //activity?.title = "Просмотр компании"
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        mBinding.coViComName.text = name_company
        activity?.title = "Услуги"
        init()
        val interceptor = HttpLoggingInterceptor()
        interceptor.level = HttpLoggingInterceptor.Level.BODY
        val client = OkHttpClient.Builder()
            .addInterceptor(interceptor)
            .build()
        val retrofit = Retrofit.Builder()
            //.baseUrl("http://10.0.2.2:8000").client(client) //для эмулятора
            .baseUrl(URLToLocale).client(client) //для подключенного
            устройства
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        val api = retrofit.create(MainApi::class.java)
        CoroutineScope(Dispatchers.IO).launch {
            list = api.getCompanyServicesListA(name_company)
            activity?.runOnUiThread {
                adapter.refreshList(list)
            }
        }
        mBinding.coViSave.setOnClickListener {
            var codeFlag = 200

```

```

        for (i in 0 until mBinding.comViewRcView.childCount) {
            val holder: RecyclerView.ViewHolder =

mBinding.comViewRcView.getChildViewHolder(mBinding.comViewRcView.getChildAt(i
))
            val switchCompat: SwitchCompat =
holder.itemView.findViewById(R.id.co_vi_switch)
            val textLimit: EditText =
holder.itemView.findViewById(R.id.co_vi_limit)
            if (textLimit.text.toString().all { it.isDigit() }) {
                val elemCoVi = list[i]
                val jsonObject = JSONObject()
                jsonObject.put("limit",
textLimit.text.toString().toInt())
                jsonObject.put("isActive", switchCompat.isChecked)
                val requestBody =
jsonObject.toString().toRequestBody("application/json".toMediaTypeOrNull())
                CoroutineScope(Dispatchers.IO).launch {
                    val response =
api.UpdateCompanyService(requestBody, list[i].id)
                    withContext(Dispatchers.Main) {
                        when (response.code()) {
                            200 -> {}
                            else -> { // Note the block
                                codeFlag = response.code()
                                this.cancel()
                                Log.e("RETROFIT_ERROR",
response.code().toString())}}}}
                        } else {
                            codeFlag = -1
                            AlertDialog.Builder(requireContext()).setTitle("Ошибка").setMessage(
                                "У услуги " + list[i].service.serviceName + " лимит
не является числом или не введен")
                                .setPositiveButton("Ок") { dialog, j -> }.show()
                                    break}}
                            when (codeFlag) {
                                -1 -> {}
                                200 -> {
                                    AlertDialog.Builder(requireContext()).setTitle("Уведомление")
                                        .setMessage("Данные успешно отправлены".toString())
                                        .setPositiveButton("Ок") { dialog, j ->

findNavController().navigate(R.id.action_companyViewFragment_to_companyFragme
nt)
                                }.show() }
                            else -> {
                                AlertDialog.Builder(requireContext()).setTitle("Ошибка")
                                    .setMessage(
                                        "Ошибка при отправке данных, код " + codeFlag
                                        .toString()
                                    ).setPositiveButton("Ок") { dialog, j -> }.show()}}}}
                            private fun init() = with(mBinding) {
                                comViewRcView.layoutManager = LinearLayoutManager(requireContext())
                                comViewRcView.adapter = adapter
                            }
                        }
                    }
                }
            }
        }
    }
}

```

Листинг Б.19 – Файл FinishFragment

```

package com.example.emp_reg_02.view.fragments
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup

```

```

import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.core.widget.doOnTextChanged
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.emp_reg_02.*
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.cancel
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.OkHttpClient
import okhttp3.RequestBody.Companion.toRequestBody
import okhttp3.logging.HttpLoggingInterceptor
import org.json.JSONObject
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class FinishFragment : Fragment(), FinishRVAdapter.Listener {
    private lateinit var mBinding: FragmentFinishBinding
    private val adapter = FinishRVAdapter(this)
    private lateinit var api: MainApi
    var list = ArrayList<MaintainedElem>()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?, savedInstanceState:
Bundle?
    ): View? {
        mBinding = FragmentFinishBinding.inflate(inflater)
        return mBinding.root
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val nav_controller = findNavController()
        rc_init()
        val interceptor = HttpLoggingInterceptor()
        interceptor.level = HttpLoggingInterceptor.Level.BODY
        val client =
OkHttpClient.Builder().addInterceptor(interceptor).build()
        val retrofit = Retrofit.Builder()
            //.baseUrl("http://10.0.2.2:8000").client(client) //для эмулятора
            .baseUrl(URLToLocale).client(client) //для подключенного
устройства
            .addConverterFactory(GsonConverterFactory.create()).build()
        api = retrofit.create(MainApi::class.java)
        CoroutineScope(Dispatchers.IO).launch {
            list = api.getMaintainedsList()
            Log.d("mylog", "лист: $list")
            activity?.runOnUiThread {
                adapter.refreshList(list)
            }
        }
    }
    override fun onDestroyView() {
        super.onDestroyView()
    }
    private fun rc_init() = with(mBinding) {
        finishRcView.layoutManager = LinearLayoutManager(requireContext())
        finishRcView.adapter = adapter
        searchField.doOnTextChanged { text, start, before, count ->
            if (text == null) {

```

```

        adapter.refreshList(list)
    } else {
        val perftext = Utils.replaceNumberWords(text.toString())
        val newList = ArrayList<MaintainedElem>()
        for (item in list) {
            val regex = Regex(perftext.toString())
            if (item.arrivalDate.contains(regex) ||
item.employeer.autoNumber.contains(regex)) {
                newList.add(item)
            }
        }
        adapter.refreshList(newList)
    }
}

override fun onClick(item: MaintainedElem) {
    super.onClick(item)
    //спросить подтверждение,
    AlertDialog.Builder(requireContext()).setTitle("Подтверждение
выезда")
        .setMessage("Вы действительно хотите отметить данный
автомобиль?")
        .setNegativeButton("Нет") { dialog, i ->
        }.setPositiveButton("Да") { dialog, i ->
            //выписать элемент из списка, добавить элемент в список
истории

            val jsonObject = JSONObject()
            jsonObject.put("employeer", item.employeer.id.toString())
            jsonObject.put("services", item.services)
            jsonObject.put("arrivalDate", item.arrivalDate)
            val requestBody =

jsonObject.toString().toRequestBody("application/json".toMediaTypeOrNull())
            CoroutineScope(Dispatchers.IO).launch {
                //удаление элемента
                val responsel = api.deleteMaintained(item.id)
                withContext(Dispatchers.Main) {
                    when (responsel.code()) {
                        200 -> {}
                        else -> { // Note the block
                            Log.e("RETROFIT_ERROR",
responsel.code().toString())
                            AlertDialog.Builder(requireContext()).setTitle("Ошибка")
                            .setMessage("Ошибка
при отправке данных, код " +
responsel.code().toString()).setPositiveButton("Ок") { dialog, j ->
                                }.show()this.cancel()}}}
                                //добавление элемента в историю
                                val response2 = api.addStoryElem(requestBody)
                                withContext(Dispatchers.Main) {
                                    when (response2.code()) {
                                        200 -> {}
                                        else -> { // Note the block
                                            Log.e("RETROFIT_ERROR",
response2.code().toString())
                                            AlertDialog.Builder(requireContext()).setTitle("О
шибка")
                                            .setMessage("Ошибка при добавлении в историю, код " +
response2.code().toString()).setPositiveButton("Ок") { dialog, j ->}.show()
                                                this.cancel()}}}
                                    val list = api.getMaintainedsList()
                                    activity?.runOnUiThread {
                                        adapter.refreshList(list)
                                        //toast об успехе удаления
                                        Toast.makeText(requireContext(), "Автомобиль
отмечен", Toast.LENGTH_LONG)
                                            .show()

```

```
findNavController().navigate(R.id.action_finishFragment_self)
                        adapter.refreshList(list)
                    })).show()
//holder.itemView.findNavController().navigate(R.id.action_companyFragment_to
_companyViewFragment,null)//list[position] в аргумент на следующий экран
    }}
```

Листинг Б.20 – Файл HelpFragment

```
package com.example.emp_reg_02.view.fragments
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.example.emp_reg_02.R
class HelpFragment : Fragment() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?, savedInstanceState:
Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_help, container, false)
        activity?.title = "Справка"
    }
}
```

Листинг Б.21 – Файл LoginFragment

```
package com.example.emp_reg_02.view.fragments
import android.annotation.SuppressLint
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.appcompat.app.AlertDialog
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import com.example.emp_reg_02.*
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
class LoginFragment : Fragment() {
    private var accList = ArrayList<Account>()
    private lateinit var mBinding: FragmentLoginBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?, savedInstanceState:
Bundle?
    ): View? {
        // Inflate the layout for this fragment
        activity?.title = ""
        mBinding = FragmentLoginBinding.inflate(inflater)
        return mBinding.root
    }
    @SuppressLint("CommitPrefEdits")
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
```

```

        super.onCreate(view, savedInstanceState)
        activity?.title = "Регистрация"
        val interceptor = HttpLoggingInterceptor()
        interceptor.level = HttpLoggingInterceptor.Level.BODY
        val client =
OkHttpClient.Builder().addInterceptor(interceptor).build()
        CoroutineScope(Dispatchers.IO).launch {
            val retrofit = Retrofit.Builder()
                //baseUrl("http://10.0.2.2:8000").client(client) //для
эмулятора
                .baseUrl(URLToLocale).client(client) //для подключенного
устройства
                .addConverterFactory(GsonConverterFactory.create()).build()
            val accApi = retrofit.create(MainApi::class.java)
            accList = accApi.getAccounts()
            mBinding.loginButton.setOnClickListener {
                var isFind = false
                for (item in accList) {
                    if (mBinding.login.text.toString() == item.login &&
mBinding.password.text.toString() == item.password) {
                        isFind = true
                        AlertDialog.Builder(requireContext())
                            .setMessage("Вы войдёте в систему как " + item.FIO)
                            .setNegativeButton("Отмена") { dialog, i ->
                                }.setPositiveButton("Подтвердить") { dialog, i ->
                                    val editor = pref.edit()
                                    editor.putString("login", item.login)
                                    editor.putString("password", item.password)
                                    editor.putString("FIO", item.FIO)
                                    editor.putString("access", item.accessRights)
                                    editor.apply()
                                    if (emptyLoginFlag == true) {
                                        emptyLoginFlag = false
                                        (activity as
MainActivity).init(item.accessRights, item.FIO)

findNavController().navigate(R.id.action_loginFragment_to_mainFragment)
                                    } else {
                                        val editor = pref.edit()
                                        editor.putString("login", item.login)
                                        editor.putString("password", item.password)
                                        editor.putString("FIO", item.FIO)
                                        editor.putString("access", item.accessRights)
                                        editor.apply()
                                        (activity as
MainActivity).init(item.accessRights, item.FIO)
                                        Log.d("mylog", "Изменение заголовка через
фрагмент")
findNavController().navigate(R.id.action_loginFragment_to_mainFragment)
                                    }
                                }.show()
                                    break
                                }
                            }
                        if (!isFind) {
                            AlertDialog.Builder(requireContext()).setMessage("Неверный
логин и(или) пароль")
                                .setNegativeButton("Ок") { dialog, i ->
                                    }.show()
                                }
                            }
                        override fun onDestroyView() {
                            accList.clear()
                            super.onDestroyView()
                        }
                    }

```

Листинг Б.22 – Файл MainFragment

```
package com.example.emp_reg_02.view.fragments
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import com.example.emp_reg_02.R
import com.example.emp_reg_02.databinding.FragmentMainBinding
class MainFragment : Fragment() {
    private lateinit var mBinding: FragmentMainBinding
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        mBinding = FragmentMainBinding.inflate(inflater)
        return mBinding.root
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        mBinding.buttonReg.setOnClickListener {

            findNavController().navigate(R.id.action_mainFragment_to_registerFragment,
            null)

        }
        mBinding.buttonCheckout.setOnClickListener {

            findNavController().navigate(R.id.action_mainFragment_to_finishFragment,
            null)

        }
        activity?.title = "Главная страница"
    }
}
```

Листинг Б.23 – Файл RegisterFragment

```
package com.example.emp_reg_02.view.fragments
import android.content.Context
import android.content.pm.PackageManager
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.inputmethod.InputMethodManager
import android.widget.Toast
import androidx.* import com.example.emp_reg_02.R
import com.example.emp_reg_02.URLToLocale
import com.example.emp_reg_02.Utils
import com.example.emp_reg_02.Utils.Companion.replaceNumberWords
import com.example.emp_reg_02.api.MainApi
import com.example.emp_reg_02.databinding.FragmentRegisterBinding
import com.example.emp_reg_02.entities.Employeeer
import com.google.mlkit.vision.text.*
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.HttpException
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
class RegisterFragment : Fragment() {
    private var _binding: FragmentRegisterBinding? = null
```

```

private val mBinding get() = _binding!!
private lateinit var textRecognizer: TextRecognizer
private var matchingArray: ArrayList<String> = ArrayList()
private var isOnceStart = true
private var isManually = false
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    _binding = FragmentRegisterBinding.inflate(layoutInflater)
    return mBinding.root
}
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val recognizer =
TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS)
    //val nubmer = scanCameraAutoNumber()
    //поиск по данным полученным с камеры
    if (permissionCameraGranted()) {
        startCamera()
    } else {
        requestCameraPermission()
    }
    mBinding.apply {
        buttonSearch.setOnClickListener {
            if (editAutoNumber.text.toString().isEmpty()) {
                //поиск по данным внутри поля ввода
                nextFragment(editAutoNumber.text.toString())
            } else {
            }}
        activity?.title = "Регистрация"
        buttonManInput.setOnClickListener {
            editAutoNumber.setText("")
            isManually = true
            editAutoNumber.showKeyboard()
        }}
    override fun onDestroyView() {
        mBinding.editAutoNumber.setText("")
        _binding = null
        textRecognizer.close()
        matchingArray.clear()
        isOnceStart = true
        isManually = false
        super.onDestroyView()
    }
    private fun startCamera() {
        val cameraController =
LifecycleCameraController(this.requireContext())
        val previewView: PreviewView = mBinding.previewView
        val options = TextRecognizerOptions.Builder()
            .build()
        textRecognizer = TextRecognition.getClient(options)
        cameraController.setImageAnalysisAnalyzer(
            ContextCompat.getMainExecutor(this.requireContext()),
            MlKitAnalyzer(
                listOf(textRecognizer),
                CameraController.COORDINATE_SYSTEM_VIEW_REFERENCED,
                ContextCompat.getMainExecutor(this.requireContext())
            ) { result: MlKitAnalyzer.Result? ->
                val textResult = result?.getValue(textRecognizer) ?:
return@MlKitAnalyzer
                val outTextBlocks = textResult.textBlocks
                //находим arrayMaxCapacity значений, похожих на номер авто,
                //выбираем наиболее часто встречающееся и отмечаем его как

```

верное

```

        //Останавливает поиск, если пользователь выбрал ручной ввод
        if (!isManually) {
            getMatchStrings(outTextBlocks)
        }
    }
)
cameraController.bindToLifecycle(this)
previewView.controller = cameraController
}
private fun getMatchStrings(outTextBlocks: List<Text.TextBlock>) {
    val arrayMaxCapacity = 6
    val regNumber =
        Regex("""^[АБЕКМНОРСТУХкопцых]\s?\d{3} (?<!000)\s?[АБЕКМНОРСТУХкопцых]{2}\s?\d{2,3}$""")

    for (block in outTextBlocks) {
        if (regNumber.matches(block.text)) {
            Log.d("mylog", "Есть, " + block.text)
            if (matchingArray.size < arrayMaxCapacity) {
                matchingArray.add(block.text.replace(" ", ""))
                Log.d("mylog", "Массив: $matchingArray")
            } else {
                //nextFragment(processProbableValue())
            }
        }
    }
    mBinding.editAutoNumber.setText(Utils.replaceNumberWords(processProbableValue()))
    matchingArray.clear()
}
} else {
    Log.d("mylog", "Нет, " + block.text)
}
}
}
private fun processProbableValue(): String {
    Log.d("mylog", "Обработка полученных значений:")
    var target: String = ""
    //получаем уникальные значения
    var existValues = matchingArray.distinct()
    //делаем массив частоты
    var valuesPop: ArrayList<Int> = ArrayList()
    for (item in existValues) {
        valuesPop.add(0)
    }
    //подсчитываем частоту для каждого
    for (item in matchingArray) {
        for (num in 0 until existValues.size) {
            if (item == existValues[num]) {
                valuesPop[num]++
            }
        }
    }
    Log.d("mylog", "Полученная популярность: ")
    Log.d("mylog", existValues.toString())
    Log.d("mylog", valuesPop.toString())
    //отправляем наиболее часто встречающееся в списке значение
    for (num in 0 until valuesPop.size) {
        if (valuesPop.max() == valuesPop[num]) {
            target = existValues[num]
        }
    }
    return target
}
}

```

```

private fun nextFragment(autoNumb: String) {
    if (isOnceStart) {
        val autoNumber = replaceNumberWords(autoNumb)
        var employeeer: Employeeer
        val interceptor = HttpLoggingInterceptor()
        interceptor.level = HttpLoggingInterceptor.Level.BODY
        val client = OkHttpClient.Builder()
            .addInterceptor(interceptor)
            .build()
        val retrofit = Retrofit.Builder()
            //baseUrl("http://10.0.2.2:8000").client(client) //для
эмулятора
            .baseUrl(URLToLocale) //для подключенного устройства
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        val empApi = retrofit.create(MainApi::class.java)
        CoroutineScope(Dispatchers.Main).launch {
            try {
                isOnceStart = false
                employeeer = empApi.getEmployeeerByNumber(autoNumber)
                Log.d("mylog", "Полученный с сервера клиент: $employeeer")
                val navBundle = bundleOf("thisEmployeeer" to employeeer)
                Log.d("mylog", "Бандл сделан, попытка навигации:")
                findNavController().navigate(

R.id.action_registerFragment_to_applyRegisterFragment,
                navBundle
            )
            } catch (e: HttpException) {
                Log.d("mylog", "Поймана ошибка, " + e.code())
                if (e.code() == 500) {

AlertDialog.Builder(requireContext()).setTitle("Ошибка")
                    .setMessage(
                        "Запись о клиенте с кодом автомобиля
\"$autoNumber\" не найдена." +
                        "При неверном определении камерой
воспользуйтесь ручным вводом."
                    )
                    .setPositiveButton("Ок") { dialog, i ->
                        matchingArray.clear()
                        isOnceStart = true
                    }
                    .setNegativeButton("Использовать ручной ввод") {
dialog, i ->
                        matchingArray.clear()
                        isOnceStart = true
                        isManually = true
                        mBinding.editAutoNumber.setText("")
                        mBinding.editAutoNumber.showKeyboard()
                    }
                    .show()
                } else {

AlertDialog.Builder(requireContext()).setTitle("Ошибка")
                    .setMessage("Произошла ошибка при связи с
сервером")
                    .setPositiveButton("Ок") { dialog, i ->
                        isOnceStart = true
                    }
                    .show()
                }
            }
        }
    }
}

```

```

        }
    } else {
        return
    }
}
@Deprecated("Deprecated in Java")
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<String>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
    if (requestCode == 0) {
        if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            startCamera()
        } else {
            Toast.makeText(
                this.requireContext(),
                "Использование камеры запрещено, работа с приложением
ограничена",
                Toast.LENGTH_SHORT
            )
                .show()
        }
    }
}
private fun View.showKeyboard() {
    this.requestFocus()
    val inputMethodManager =
        context.getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager
    inputMethodManager.showSoftInput(this,
InputMethodManager.SHOW_IMPLICIT)
}
private fun permissionCameraGranted() =
    ContextCompat.checkSelfPermission(
        this.requireContext(),
        android.Manifest.permission.CAMERA
    ) == PackageManager.PERMISSION_GRANTED
private fun requestCameraPermission() {
    ActivityCompat.requestPermissions(
        this.requireActivity(),
        arrayOf(android.Manifest.permission.CAMERA), 0
    )
}
}

```

Листинг Б.24 – Файл StoryFragment

```

package com.example.emp_reg_02.view.fragments
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.emp_reg_02.URLToLocale
import com.example.emp_reg_02.api.MainApi
import com.example.emp_reg_02.databinding.FragmentStoryBinding
import com.example.emp_reg_02.view.RVadapters.StoryRVAdapter
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch

```

```

import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
class StoryFragment : Fragment() {
    private lateinit var mBinding: FragmentStoryBinding
    private val adapter = StoryRVAdapter()
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        mBinding = FragmentStoryBinding.inflate(inflater)
        return mBinding.root
    }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val nav_controller = findNavController()
        activity?.title = "История"
        rc_init()
        val interceptor = HttpLoggingInterceptor()
        interceptor.level = HttpLoggingInterceptor.Level.BODY
        val client = OkHttpClient.Builder()
            .addInterceptor(interceptor)
            .build()
        val retrofit = Retrofit.Builder()
            //.baseUrl("http://10.0.2.2:8000").client(client) //для эмулятора
            .baseUrl(URLToLocale).client(client) //для подключенного
устройства
            .addConverterFactory(GsonConverterFactory.create())
            .build()
        val api = retrofit.create(MainApi::class.java)
        CoroutineScope(Dispatchers.IO).launch {
            val list = api.getHistoryList()
            activity?.runOnUiThread{
                adapter.refreshList(list)
            }
        }
    }
    private fun rc_init() = with(mBinding) {
        storyRcView.layoutManager = LinearLayoutManager(requireContext())
        storyRcView.adapter = adapter
    }
}

```

Листинг Б.25 – Файл ApplyRegisterRVAdapter

```

package com.example.emp_reg_02.view.RVadapters
import android.annotation.SuppressLint
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.emp_reg_02.R
import com.example.emp_reg_02.databinding.FragmentApplyRegisterRvItemBinding
import com.example.emp_reg_02.entities.CompanyService
import com.example.emp_reg_02.listApplyRegister
class ApplyRegisterRVAdapter :
    RecyclerView.Adapter<ApplyRegisterRVAdapter.ApplyRegisterHolder>() {
    class ApplyRegisterHolder(item: View) : RecyclerView.ViewHolder(item) {
        val binding = FragmentApplyRegisterRvItemBinding.bind(item)
        @SuppressLint("SetTextI18n")
        fun bind(service: CompanyService) = with(binding) {
            //нужна обработка ограничения администратора
            name.text = service.service.serviceName
        }
    }
}

```

```

        type.text = "/" + service.limit + " (" +
service.service.countType + ")"
        amount.setText(service.limit.toString())
    }
}
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ApplyRegisterHolder {
    val view = LayoutInflater.from(parent.context)
        .inflate(R.layout.fragment_apply_register_rv_item, parent, false)
    return ApplyRegisterHolder(view)
}
override fun getItemCount(): Int {
    return listApplyRegister.size
}
override fun onBindViewHolder(holder: ApplyRegisterHolder, position: Int)
{
    holder.bind(listApplyRegister[position])
}
fun addService(service: CompanyService) {
    listApplyRegister.add(service)
    //notifyItemInserted(serviceList.size)
    notifyDataSetChanged()
}
fun addServiceList(serList: List<CompanyService>) {
    serList.forEach { item -> addService(item) }
    notifyDataSetChanged()
}
fun refreshList(serList: List<CompanyService>) {
    listApplyRegister.clear()
    serList.forEach { item -> addService(item) }
    notifyDataSetChanged()
}
}
}

```

Листинг Б.26 – Файл CompaniesRVAdapter

```

package com.example.emp_reg_02.view.RVadapters
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.core.os.bundleOf
import androidx.recyclerview.widget.RecyclerView
import com.example.emp_reg_02.R
import com.example.emp_reg_02.databinding.FragmentCompaniesRvItemBinding
import com.example.emp_reg_02.entities.Company
import androidx.navigation.findNavController
import com.example.emp_reg_02.entities.CompanyService
import com.example.emp_reg_02.listApplyRegister
import com.example.emp_reg_02.listCompanies
class CompaniesRVAdapter:
RecyclerView.Adapter<CompaniesRVAdapter.CompaniesRVHolder>() {
    class CompaniesRVHolder(item: View): RecyclerView.ViewHolder(item) {
        val binding = FragmentCompaniesRvItemBinding.bind(item)
        fun bind(elem: Company) = with(binding) {
            comCompanyName.text = elem.companyName
        }
    }
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
CompaniesRVHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.fragment_companies_rv_it
em, parent, false)
        return CompaniesRVHolder(view)
    }
}

```

```

        override fun getItemCount(): Int {
            return listCompanies.size
        }
        override fun onBindViewHolder(holder: CompaniesRVHolder, position: Int) {
            holder.bind(listCompanies[position])
            val navBundle = bundleOf("name_company" to
listCompanies[position].companyName)
            holder.binding.compCard.setOnClickListener{

holder.itemView.findNavController().navigate(R.id.action_companyFragment_to_c
ompanyViewFragment,navBundle)
        }
    }
    private var navBundle: Bundle? = null
    fun addElem(elem: Company) {
        listCompanies.add(elem)
        //notifyItemInserted(serviceList.size)
        notifyDataSetChanged()
    }
    fun addList(newList: List<Company>){
        listCompanies.addAll(newList)
        notifyDataSetChanged()
    }
    fun refreshList(serList: List<Company>){
        listCompanies.clear()
        serList.forEach{item -> addElem(item)}
        notifyDataSetChanged()
    }
}

```

Листинг Б.27 – Файл CompanyViewRVAdapter

```

package com.example.emp_reg_02.view.RVadapters
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.emp_reg_02.R
import com.example.emp_reg_02.databinding.FragmentCompanyViewRvItemBinding
import com.example.emp_reg_02.entities.CompanyService
import com.example.emp_reg_02.listCompanyView
class CompanyViewRVAdapter :
RecyclerView.Adapter<CompanyViewRVAdapter.CompanyViewRVHolder>() {
    class CompanyViewRVHolder(item: View) : RecyclerView.ViewHolder(item) {
        val binding = FragmentCompanyViewRvItemBinding.bind(item)
        fun bind(elem: CompanyService) = with(binding) {
            coViName.text = elem.service?.serviceName
            coViType.text = "(" + elem.service?.countType + ")"
            coViLimit.setText(elem.limit.toString())
            coViSwitch.isChecked = elem.isActive
        }
    }
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
CompanyViewRVHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.fragment_company_view_rv_item, parent, false)
        return CompanyViewRVHolder(view)
    }
    override fun getItemCount(): Int {
        return listCompanyView.size
    }
    override fun onBindViewHolder(holder: CompanyViewRVHolder, position: Int)
{
        holder.bind(listCompanyView[position])
    }
}

```

```

fun addElem(elem: CompanyService) {
    listCompanyView.add(elem)
    //notifyItemInserted(serviceList.size)
    notifyDataSetChanged()
}
fun addList(newList: List<CompanyService>) {
    listCompanyView.addAll(newList)
    notifyDataSetChanged()
}
fun refreshList(serList: List<CompanyService>) {
    listCompanyView.clear()
    serList.forEach { item -> addElem(item) }
    notifyDataSetChanged()
}
}

```

Листинг Б.28 – Файл FinishRVAdapter

```

package com.example.emp_reg_02.view.RVadapters
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.emp_reg_02.R
import com.example.emp_reg_02.databinding.FragmentFinishRvItemBinding
import com.example.emp_reg_02.entities.MaintainedElem
import com.example.emp_reg_02.listFinish

class FinishRVAdapter(val listener: Listener) :
    RecyclerView.Adapter<FinishRVAdapter.FinishRVHolder>() {
    class FinishRVHolder(item: View) : RecyclerView.ViewHolder(item) {
        val binding = FragmentFinishRvItemBinding.bind(item)
        fun bind(elem: MaintainedElem, listener: Listener) = with(binding) {
            finDatetime.text = elem.arrivalDate
            number.text = elem.employeer.autoNumber
            finCard.setOnClickListener {
                listener.onClick(elem)
            }
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
    FinishRVHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.fragment_finish_rv_item, parent, false)
        return FinishRVHolder(view)
    }

    override fun getItemCount(): Int {
        return listFinish.size
    }

    override fun onBindViewHolder(holder: FinishRVHolder, position: Int) {
        holder.bind(listFinish[position], listener)
    }

    fun addElem(elem: MaintainedElem) {
        listFinish.add(elem)
        //notifyItemInserted(serviceList.size)
        notifyDataSetChanged()
    }

    fun addList(newList: List<MaintainedElem>) {
        listFinish.addAll(newList)
        notifyDataSetChanged()
    }

    fun refreshList(serList: List<MaintainedElem>) {
        listFinish.clear()
        serList.forEach { item -> addElem(item) }
    }
}

```

```

        notifyDataSetChanged()
    }
    fun notifyList() {
        notifyDataSetChanged()
    }
    interface Listener {
        fun onClick(item: MaintainedElem) {
        }
    }
}

```

Листинг Б.29 – Файл StoryRVAdapter

```

package com.example.emp_reg_02.view.RVadapters
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.emp_reg_02.R
import com.example.emp_reg_02.databinding.FragmentStoryRvItemBinding
import com.example.emp_reg_02.entities.HistoryElem
import com.example.emp_reg_02.listStory
class StoryRVAdapter: RecyclerView.Adapter<StoryRVAdapter.StoryRVHolder>() {
    class StoryRVHolder(item: View): RecyclerView.ViewHolder(item) {
        val binding = FragmentStoryRvItemBinding.bind(item)
        fun bind(elem: HistoryElem) = with(binding){
            arrivDatetime.text = elem.arrivalDate
            depDatetime.text = elem.departDate
            number.text = elem.employeeer.autoNumber
        }
    }
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
    StoryRVHolder {
        val view =
    LayoutInflater.from(parent.context).inflate(R.layout.fragment_story_rv_item,
    parent, false)
        return StoryRVHolder(view)
    }
    override fun getItemCount(): Int {
        return listStory.size
    }
    override fun onBindViewHolder(holder: StoryRVHolder, position: Int) {
        holder.bind(listStory[position])
    }
    fun addElem(elem: HistoryElem){
        listStory.add(elem)
        //notifyItemInserted(serviceList.size)
        notifyDataSetChanged()
    }
    fun addList(newList: List<HistoryElem>){
        listStory.addAll(newList)
        notifyDataSetChanged()
    }
    fun refreshList(serList: List<HistoryElem>){
        listStory.clear()
        addList(serList)
    }
}

```

Листинг Б.30 – Файл MainActivity

```

package com.example.emp_reg_02.view
import android.annotation.SuppressLint
import android.graphics.Color
import android.os.Bundle

```

```

import android.view.Menu
import android.view.View
import android.widget.TextView
import androidx.appcompat.app.ActionBarDrawerToggle
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.content.ContextCompat
import androidx.core.view.GravityCompat
import androidx.navigation.NavController
import androidx.navigation.fragment.NavHostFragment
import com.example.emp_reg_02.OnSwipeTouchListener
import com.example.emp_reg_02.R
import com.example.emp_reg_02.databinding.ActivityMainBinding
import com.example.emp_reg_02.emptyLoginFlag
import com.example.emp_reg_02.pref
class MainActivity : AppCompatActivity() {
    private lateinit var mBinding: ActivityMainBinding
    private lateinit var mDrawerHeaderView: View
    lateinit var navController: NavController
    @SuppressWarnings("ClickableViewAccessibility", "ResourceAsColor")
    override fun onCreate(savedInstanceState: Bundle?) {
        setTheme(R.style.Theme_Emp_reg_02);
        super.onCreate(savedInstanceState)
        mBinding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(mBinding.root)
        val navHostFragment =
            supportFragmentManager.findFragmentById(R.id.fragment_container_view)
        as NavHostFragment
        navController = navHostFragment.navController

        pref = getSharedPreferences("ACCOUNT",
AppCompatActivity.MODE_PRIVATE)
        val login = pref.getString("login", null)
        val password = pref.getString("password", null)
        val fio = pref.getString("FIO", null)
        val access = pref.getString("access", null)
        navController.navigate(R.id.mainFragment)
        if (login == null || password == null || fio == null || access ==
null) {
            emptyLoginFlag = true
            navController.navigate(R.id.action_mainFragment_to_loginFragment)
        } else {
            emptyLoginFlag = false
            init(access, fio)
        }
    }
    @SuppressWarnings("ClickableViewAccessibility")
    fun init(access: String, fio: String) = with(mBinding) {
        //выдвижение меню при свайпе вправо
        root.setOnTouchListener(object :
OnSwipeTouchListener(this@MainActivity) {
            override fun onSwipeRight(): Boolean {
                //Toast.makeText(context, "RIGHT", Toast.LENGTH_SHORT).show()
                drawer.openDrawer(GravityCompat.START)
                setMenuFIO(fio)
                return true
            }
        })

        //обработка нажатия кнопки "Сменить аккаунт"
        mDrawerHeaderView = leftDrawer.getHeaderView(0)

        mDrawerHeaderView.findViewById<TextView>(R.id.change_login).setOnClickListene
r {

```

```

        navController.navigate(R.id.loginFragment)
        drawer.closeDrawer(GravityCompat.START)
    }
    //обработка нажатий на элементы в выдвигаемом меню
    leftDrawer.setNavigationItemSelectedListener {
        when (it.itemId) {
            R.id.item1 -> {
                navController.navigate(R.id.mainFragment)
                drawer.closeDrawer(GravityCompat.START)
            }

            R.id.item2 -> {
                navController.navigate(R.id.storyFragment)
                drawer.closeDrawer(GravityCompat.START)
            }

            R.id.item3 -> {
                if (access == "admin") {
                    navController.navigate(R.id.administrateFragment)
                } else {
                    AlertDialog.Builder(this@MainActivity)
                        .setMessage("Необходимы права администратора")
                        .setNegativeButton("Ок") { dialog, i ->
                            }.show()
                }
                drawer.closeDrawer(GravityCompat.START)
            }

            R.id.item4 -> {
                navController.navigate(R.id.helpFragment)
                drawer.closeDrawer(GravityCompat.START)
            }
        }
        true
    }
}

//кнопка-гамбургер для открытия бокового меню
val toggle = ActionBarDrawerToggle(
    this@MainActivity, drawer, mainToolbar, 0, 0
)
drawer.addDrawerListener(toggle)
toggle.syncState()
toggle.drawerArrowDrawable.color =
ContextCompat.getColor(this@MainActivity, R.color.white)

//setSupportActionBar(mBinding.mainToolbar);
//mBinding.mainToolbar.subtitle = "Главная страница"
supportActionBar?.title = ""
mBinding.mainToolbar.setTitleTextColor(Color.WHITE)
//supportActionBar?.setDisplayHomeAsUpEnabled(false);
}

fun setMenuFIO(fio: String) {
    mBinding.drawer.findViewById<TextView>(R.id.account_name).text = fio
}

override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    return super.onCreateOptionsMenu(menu)
}
}

```

Листинг Б.31 – Файл Constant

```
package com.example.emp_reg_02

import android.annotation.SuppressLint
import android.content.Context.MODE_PRIVATE
import android.content.SharedPreferences
import android.content.ContextWrapper
import com.example.emp_reg_02.entities.Account
import com.example.emp_reg_02.entities.Company
import com.example.emp_reg_02.entities.CompanyService
import com.example.emp_reg_02.entities.HistoryElem
import com.example.emp_reg_02.entities.MaintainedElem

import com.example.emp_reg_02.view.MainActivity

@SuppressLint("StaticFieldLeak")

var loginList = ArrayList<Account>()

//списки recycleView
var listApplyRegister = ArrayList<CompanyService>()
var listCompanies = ArrayList<Company>()
var listCompanyView = ArrayList<CompanyService>()
var listFinish = ArrayList<MaintainedElem>()
var listStory = ArrayList<HistoryElem>()

//сохранённые данные об аккаунте
lateinit var pref: SharedPreferences
//флаг только что запущенного приложения (без аккаунта)
var emptyLoginFlag = false

//URL для соединения
var URLToLocale = "http://192.168.35.156:8000"
```

Листинг Б.32 – Файл Utils

```
package com.example.emp_reg_02

import android.util.Log
import android.content.Context
import android.view.GestureDetector
import android.view.MotionEvent
import android.view.View
import kotlin.math.abs

abstract class Utils {

    companion object {
        fun replaceNumberWords(autoNumb: String): String {
            //АВЕКМНОРСТУХ
            var inp: String =
                autoNumb
                    .uppercase()
                    .replace('А', 'A')
                    .replace('В', 'B')
                    .replace('Е', 'E')
                    .replace('К', 'K')
                    .replace('М', 'M')
                    .replace('Н', 'H')
                    .replace('О', 'O')
                    .replace('Р', 'P')
```

```

        .replace('C', 'C')
        .replace('T', 'T')
        .replace('Y', 'Y')
        .replace('X', 'X')

        Log.d("mylog", "Замена, $inp")
        return inp
    }
}

//Модуль обработки свайпов пользователя, в 4 направлениях.
open class OnSwipeTouchListener(val context: Context?) : View.OnTouchListener {
    {
        companion object {
            private const val SwipeThreshold = 100
            private const val SwipeVelocityThreshold = 100
        }

        private val gestureDetector = GestureDetector(context, GestureListener())

        override fun onTouch(v: View, event: MotionEvent): Boolean {
            return gestureDetector.onTouchEvent(event)
        }

        open fun onSwipeRight(): Boolean { return false }
        open fun onSwipeLeft(): Boolean { return false }
        open fun onSwipeUp(): Boolean { return false }
        open fun onSwipeDown(): Boolean { return false }

        private inner class GestureListener :
            GestureDetector.SimpleOnGestureListener() {
            override fun onDown(e: MotionEvent): Boolean {
                return true
            }

            override fun onFling(
                e1: MotionEvent,
                e2: MotionEvent,
                velocityX: Float,
                velocityY: Float,
            ): Boolean {
                try {
                    val diffY = e2.y - e1.y
                    val diffX = e2.x - e1.x

                    if (abs(diffX) > abs(diffY)) {
                        if (abs(diffX) > SwipeThreshold && abs(velocityX) >
                            SwipeVelocityThreshold) {
                            return when {
                                diffX > 0 -> onSwipeRight()
                                else -> onSwipeLeft()
                            }
                        }
                    } else if (abs(diffY) > SwipeThreshold && abs(velocityY) >
                        SwipeVelocityThreshold) {
                        return when {
                            diffY > 0 -> onSwipeDown()
                            else -> onSwipeUp()
                        }
                    }
                } catch (exception: Exception) {
                    exception.printStackTrace()
                }
            }
        }
    }
}

```

```
        }  
        return false  
    }  
}
```

Приложение В

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

(рекомендуемое)

```
GET /api/v1/employeers_list

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
  {
    "id": 6,
    "companyName": "ООО \"Грузоперевозки России\"",
    "FIO": "Сычев Иван Юринович",
    "passport": "4354166268",
    "autoNumber": "P372PK05"
  },
  {
    "id": 4,
    "companyName": "ООО \"ДальГруз\"",
    "FIO": "Тамахин Даниил Игнатьевич",
    "passport": "4943429823",
    "autoNumber": "T372PB05"
  },
  {
    "id": 2,
    "companyName": "ООО \"ДальГруз\"",
    "FIO": "Дальнов Алексей Алексеевич",
    "passport": "6320111222",
    "autoNumber": "A111AA77"
  }
],
```

Рисунок В.1 – Результат теста «Проверка выдачи списка клиентов»

Maintained Api

```
POST /api/v1/maintaineds_list
```

```
HTTP 200 OK
Allow: GET, POST, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
```

Рисунок В.2 – Результат теста «Добавление обслуживаемого клиента»

Maintained Api

```
POST /api/v1/maintaineds_list
```

```
HTTP 226 IM Used
Allow: GET, POST, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
```

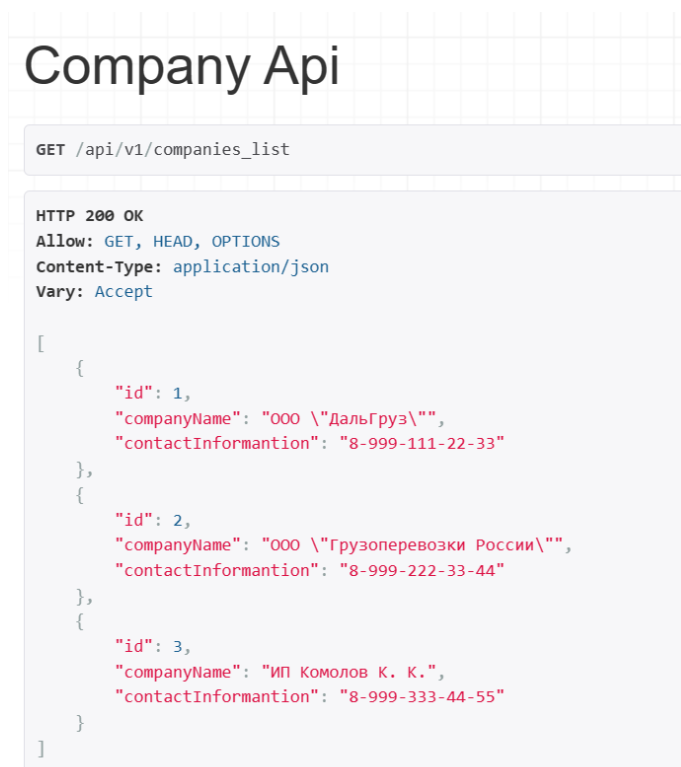
Рисунок В.3 – Результат теста «Добавление уже существующего обслуживаемого клиента»

```

D endAllActiveAnimators on 0xb400006e36b5a440 (RippleDrawable) with handle 0xb400006d46b71f40
I --> DELETE http://192.168.35.156:8000/api/v1/maintaineds_list/30
I --> END DELETE
I <-- 200 OK http://192.168.35.156:8000/api/v1/maintaineds_list/30 (60ms)
I Date: Thu, 22 Jun 2023 05:47:43 GMT
I Server: WSGIServer/0.2 CPython/3.9.13
I Vary: Accept, Cookie
I Allow: GET, POST, DELETE, HEAD, OPTIONS
I X-Frame-Options: DENY
I Content-Length: 0
I X-Content-Type-Options: nosniff
I Referrer-Policy: same-origin
I Cross-Origin-Opener-Policy: same-origin

```

Рисунок В.4 – Результат теста «Удалить обслуживаемого клиента»



The screenshot shows a REST client interface with a title bar 'Company Api'. Below the title bar, a text box contains the request 'GET /api/v1/companies_list'. The response area shows 'HTTP 200 OK' with headers: 'Allow: GET, HEAD, OPTIONS', 'Content-Type: application/json', and 'Vary: Accept'. The response body is a JSON array of three company objects.

```

GET /api/v1/companies_list

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
  {
    "id": 1,
    "companyName": "ООО \"ДальГруз\"",
    "contactInformantion": "8-999-111-22-33"
  },
  {
    "id": 2,
    "companyName": "ООО \"Грузоперевозки России\"",
    "contactInformantion": "8-999-222-33-44"
  },
  {
    "id": 3,
    "companyName": "ИП Комолов К. К.",
    "contactInformantion": "8-999-333-44-55"
  }
]

```

Рисунок В.5 – Результат теста «Проверка выдачи списка компаний»

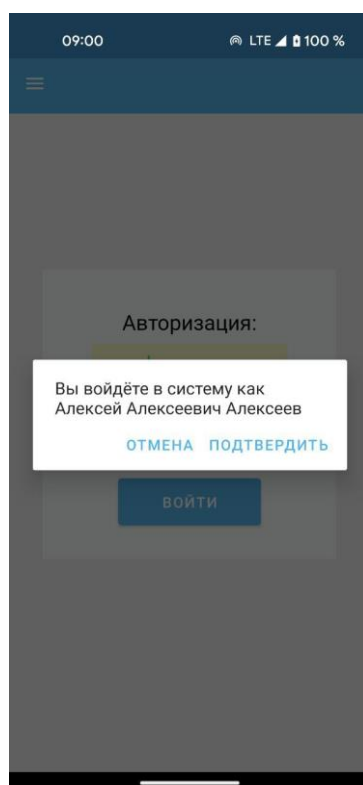


Рисунок В.6 – Результат теста «Проверка успешной авторизации в приложение»

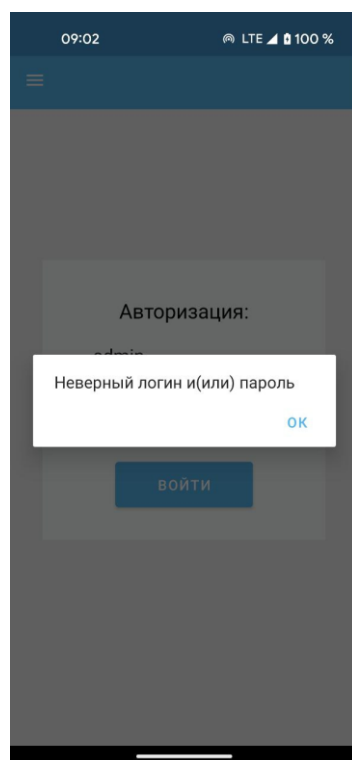


Рисунок В.7 – Результат теста «Проверка неудачной авторизации в приложение»

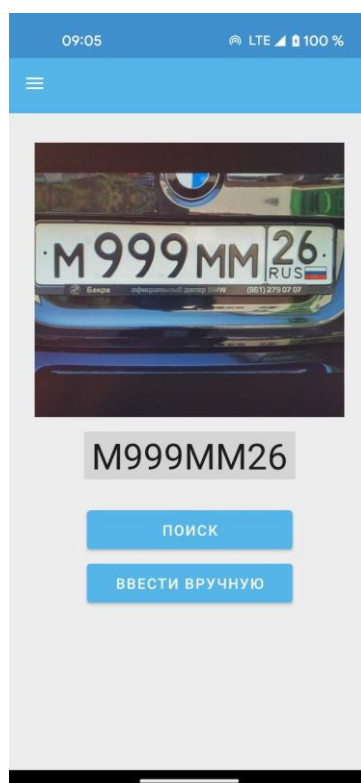


Рисунок В.8 – Результат теста «Проверка сканирования автомобильного номера камерой»

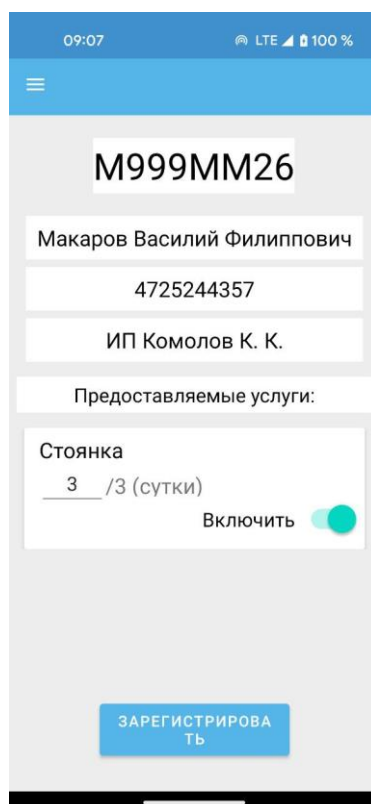


Рисунок В.9 – Результат теста «Поиск по существующему номеру автомобиля»

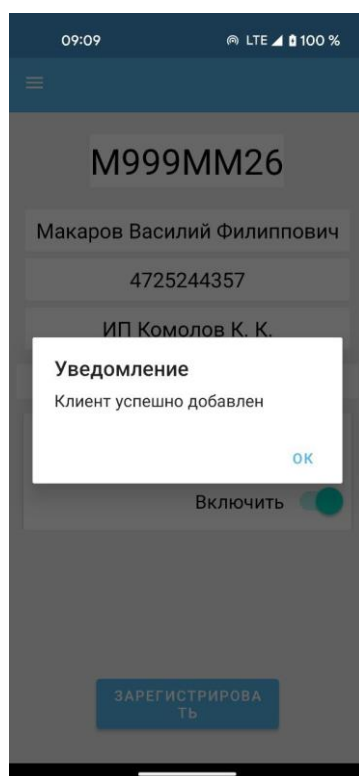


Рисунок В.10 – Результат теста «Отметить выезжающего клиента»

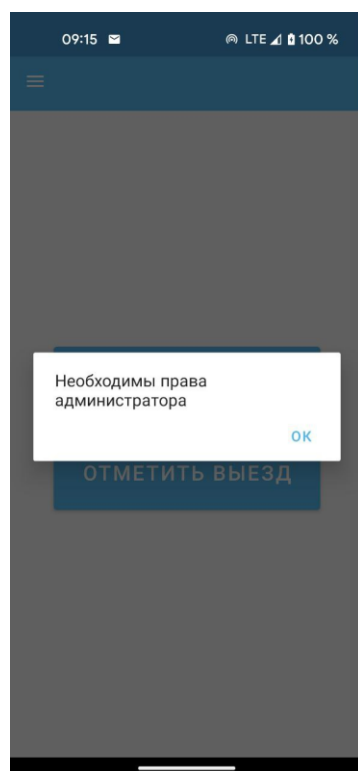


Рисунок В.11 – Результат теста «Попытаться открыть меню администратора с аккаунта без прав доступа»

The screenshot displays a mobile application interface with a blue header bar containing the time 09:17, LTE signal, and 100% battery. Below the header is a menu icon. The main content area is light gray and contains the following elements:

- A large text field with the value **M034TX34**.
- A text field with the name **Михайлов Давид Кириллович**.
- A text field with the number **4725244357**.
- A text field with the company name **ООО "Грузоперевозки России"**.
- A section titled **Предоставляемые услуги:** (Provided services:).
- Under this section, there are two service options:
 - Стойанка** (Parking): A text field with **2** and **/2 (сутки)** (per 2 days), followed by a toggle switch labeled **Включить** (Turn on) which is currently turned on.
 - Автомойка** (Car wash): A text field with **1** and **/1 (услуг)** (per 1 service), followed by a toggle switch labeled **Включить** (Turn on) which is currently turned on.
- At the bottom, there is a blue button with the text **ЗАРЕГИСТРИРОВАТЬ** (Register).

Рисунок В.12 – Результат теста «Проверка изменения сервисов, предоставляемых компании»