

**Министерство образования Иркутской области**

Государственное бюджетное профессиональное образовательное учреждение

Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

**ДП.09.02.03.23.191.02.ПЗ**

**УТВЕРЖДАЮ**

Зам. директора по УР, к.т.н.

\_\_\_\_\_ Е.А. Коробкова

**МОБИЛЬНОЕ ПРИЛОЖЕНИЕ**  
**«ТАКСОПАРК»**

Нормоконтролер:

\_\_\_\_\_  
(подпись, дата)

(В.А. Пролыгина)

Консультант по  
экономической части:

\_\_\_\_\_  
(подпись, дата)

(М.А. Рачкова)

Руководитель:

\_\_\_\_\_  
(подпись, дата)

(О.Н. Филимонова)

Студент:

\_\_\_\_\_  
(подпись, дата)

(А.В. Афонин)

Иркутск 2023

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	3
1 Предпроектное исследование .....	5
1.1 Описание предметной области.....	5
1.2 Анализ инструментальных средств реализации .....	6
2 Техническое задание на разработку программного продукта .....	12
3 Проектирование.....	13
3.1 Архитектура программного обеспечения.....	13
3.2 Функциональное проектирование.....	14
3.3 Проектирование базы данных .....	20
3.4 Проектирование пользовательского интерфейса.....	24
4. Реализация программного обеспечения.....	26
4.1 Кодирование программного обеспечения .....	26
4.2 Разработка базы данных .....	26
4.3 Разработка программного продукта.....	29
5 Документирование программного обеспечения.....	35
5.1 Руководство пользователя .....	35
6. Стоимость разработки и внедрения программного продукта .....	44
6.1 Организационно-экономическое обоснование проекта.....	44
6.2 Расчет затрат на разработку программного продукта .....	44
6.3 Расчет затрат на внедрение программного продукта .....	48
6.4 Основные выводы .....	48
ЗАКЛЮЧЕНИЕ .....	50
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	51
Приложение А – Авторизации .....	53
Приложение Б – Просмотра заказов .....	55

					ДП.09.02.03.23.191.02.ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Афонин А.В.			МОБИЛЬНОЕ ПРИЛОЖЕНИЕ «ТАКСОПАРК»  пояснительная записка	Лит.	Лист	Листов
Провер.		Филимонова О.Н.					2	57
Реценз.						ГБПОУИО «ИАТ» ПКС-19-1		
Н. Контр.		Пролыгина В.А.						
Утверд.		Коробкова Е.А.						

## ВВЕДЕНИЕ

Таксопарк – предприятие по организации перевозок пассажиров на такси, включающее в себя возможность быстрого и удобного заказа такси.

Программный продукт для данного предприятия, будет предназначен для упрощения и автоматизации работы организации «Таксопарк». Перевозка пассажиров из одной точки в другую с каждым годом становится все актуальней и актуальней. В больших городах растет численность населения, появляется большое количество новых жилых районов, расширяются границы городов. Все эти факторы делают тему перевозок людей актуальной.

Основным преимуществом приложения для заказа такси будет упрощение и удобство использования. С помощью приложения пользователь сможет быстро заказать такси в нужное ему место, сэкономяв время на общении с оператором, указывая нужный адрес напрямую в приложении. Удобство будет заключаться в том, что с помощью приложения можно будет просматривать рейтинг водителей и свой профиль, также в случае необходимости будет возможность отмены заказа удаленно. Также приложение будет предусматривать использование его водителем. Водителю же приложение позволит выбирать самому необходимый и более удобный для него заказ, отслеживать его актуальность, а также при необходимости отменять его.

Основной задачей мобильного приложения «Таксопарк» является автоматизация заказа такси и учет всех данных, необходимых для корректной работы таксопарка, среди таких данных будут: данные о клиента и водителях, их пароли и номера телефонов, сохраненные в базе вызовы, полная информация о автомобилях, включая дату последнего технического обслуживания и номера двигателей, тарифы и их стоимость, а также дополнительные услуги.

Целью дипломного проекта является разработка мобильного приложения «Таксопарк», основная задача которой, состоит в получении данных о заказе от клиентов, обработка этих данных и дальнейшая передача водителю.

					ДП.09.02.03.23.191.02.ПЗ	Лист
						3
Изм.	Лист	№ докум.	Подпись	Дата		

В ходе дипломного проектирования были поставлены следующие задачи:

1. Провести предпроектное исследование;
  - 1.1. Провести исследование предметной области;
  - 1.2. Проанализировать инструменты, используемые в разработке программного обеспечения;
  - 1.3. Обосновать выбор программных продуктов для разработки;
2. Составить техническое задание на разработку программного продукта;
3. Провести проектирование программного продукта;
  - 3.1. Представить архитектуру программного обеспечения;
  - 3.2. Провести функциональное проектирование;
  - 3.3. Спроектировать базу данных;
  - 3.4. Спроектировать пользовательский интерфейс программного продукта;
4. Реализовать программный продукт;
5. Разработать документы для программного продукта;
6. Рассчитать стоимость разработки и внедрения программного продукта.

					ДП.09.02.03.23.191.02.ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

# 1 Предпроектное исследование

## 1.1 Описание предметной области

Предметной областью дипломного проекта является предприятие «Таксопарк»

Клиент заказывает такси в приложении, информация и заносится в базу данных (Вызовы: дата, время, телефон, откуда, куда, код тарифа, код услуги, код водителя). Таким образом приложение закрепляет за клиентом автомобиль.

Также заносятся другие данные, а именно: дата заказа, время, когда нужно будет забрать клиента, номер телефона клиента для связи с ним, откуда забрать клиента, куда доставить клиента, по какому тарифу будет осуществляться перевозка клиента, код дополнительной услуги такой как грузовой автомобиль, минивэн или наличие детского кресла.

Дополнительные услуги представляют из себя список возможных дополнительных удобств, которые оплачиваются отдельно и имеют следующие параметры: код услуги, наименование, описание услуги, стоимость.

Тариф — это то, сколько будет стоить клиенту проехать 1км. Существуют разные тарифы, например, такие как Standart или Vip. Каждый тариф имеет: Код тарифа, название, краткое описание, стоимость.

К клиенту к нужному времени приезжает автомобиль. За автомобилем также закреплен сотрудник-водитель. У автомобиля широкий ряд требований и атрибутов: код автомобиля, код марки автомобиля, его регистрационный номер, номер кузова, номер двигателя, год выпуска автомобиля, пробег в километрах, дата последнего технического обслуживания, специальные отметки автомобиля.

На автомобиле к клиенту приезжает сотрудник-водитель. У каждого сотрудника есть своя должность, и, помимо этого, личные атрибуты, такие как: код сотрудника, ФИО, возраст, пол, адрес, телефон, паспортные данные, код должности.

					ДП.09.02.03.23.191.02.ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

В итоге «Таксопарк» работает по такому принципу: Клиент обращается к приложению, оно же в свою очередь заносит информацию в базу данных, далее эти данные передаются водителю машины, и тот в определенное время забирает клиента из одной точки и доставляет в другую.

В соответствии с предметной областью предприятия «Таксопарк» можно выделить базовые сущности проектируемого приложения.

Водители. Атрибуты водителя - Код водителя, ФИО, Возраст, Пол, Адрес, Телефон, Паспортные данные, Пароль.

Клиенты. Атрибуты клиента – Код клиента, ФИО, Телефон, Пароль.

Тарифы. Атрибуты Тарифа - Код тарифа, Наименование, Описание, Стоимость.

Дополнительные услуги. Атрибуты услуг - Код услуги, Наименование, Описание услуги, Стоимость.

Автомобили. Атрибуты Автомобиля - Код автомобиля, Марка авто, Регистрационный номер, Номер кузова, Номер двигателя, Год выпуска, Пробег, Код сотрудника-шофёра, Дата последнего ТО.

Вызовы. Атрибуты Вызова – Код вызова, Дата, Время, Телефон, Откуда, Куда, Код тарифа, Код услуги, Код водителя.

Мобильное приложение создается для обслуживания следующих групп пользователей:

- Клиент.
- Сотрудник-Водитель.
- Администратор

## 1.2 Анализ инструментальных средств реализации

Для разработки данного продукта было рассмотрено несколько инструментальных средств разработки программного обеспечения. Такие как: C#, Python, JavaScript.

					ДП.09.02.03.23.191.02.ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

Python – это высокоуровневый язык программирования общего назначения. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами.

JavaScript – динамический скриптовый язык программирования высокого уровня. Он отличается мультипарадигменностью. Чаще всего язык используется для создания интерактивных веб-страниц и приложений. Неизменно высокий интерес к JavaScript подтверждают специальные международные рейтинги.

C# – современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET.

Сравнение языков программирования представлено в таблице 1.

Таблица 1 – Сравнение языков программирования

Критерии	Python	JavaScript	C#
Наличие библиотек	+	+	+
Инструменты для работы с БД	+	+	+
Объектно-ориентированные возможности	+	+	+

Исходя из данной таблицы, было принято решение остановиться на языке программирования C# потому, что он достаточно прост и понятен, что поможет реализовать мне данный продукт.

Выбор среды разработки для C# встал между Visual Studio и JetBrains Rider.

Visual Studio – линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-

приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ,

JetBrains Rider – кроссплатформенная интегрированная среда разработки программного обеспечения для платформы .NET, разрабатываемая компанией JetBrains. Поддерживаются языки программирования C#.

Для выбора среды разработки была составлена таблица 2.

Таблица 2 – Сравнение сред разработки

Название IDE	Visual Studio	JetBrains Rider
Автосохранение	+	+
Автодополнение	+	+
Поиск по коду	+	+

После сравнения были сделаны выводы, что для выбранного дипломного проекта лучшее всего подойдет Visual Studio потому, что данная программа имеет очень хороший функционал, что поможет в реализации данного программного продукта. Также для создания мобильного приложения на C# был выбран фреймворк Xamarin обладающий необходимым функционалом для успешной разработки.

Для работы приложения необходимы базы данных, поэтому необходимо также провести анализ для выбора средств реализации баз данных.

MySQL достаточно легко устанавливается, а наличие множества плагинов и вспомогательных приложений упрощает работу с базами данных. Обширный функционал. Система MySQL обладает практически всем необходимым инструментарием, который может понадобиться в реализации практически любого проекта.

Преимущества:



- простота в использовании. MySQL достаточно легко устанавливается, а наличие множества плагинов и вспомогательных приложений упрощает работу с базами данных;
- обширный функционал. Система MySQL обладает практически всем необходимым инструментарием, который может понадобиться в реализации практически любого проекта;
- безопасность. Система изначально создана таким образом, что множество встроенных функций безопасности в ней работают по умолчанию;
- масштабируемость. Являясь весьма универсальной СУБД, MySQL в равной степени легко может быть использована для работы и с малыми, и с большими объемами данных;
- скорость. Высокая производительность системы обеспечивается за счет упрощения некоторых используемых в ней стандартов.

SQLite – это быстрая и легкая встраиваемая однофайловая СУБД на языке C, которая не имеет сервера и позволяет хранить всю базу локально на одном устройстве. Для работы SQLite не нужны сторонние библиотеки или службы.

Преимущества:

- файловая: вся база данных хранится в одном файле, что облегчает перемещение;
- стандартизированная: SQLite использует SQL; некоторые функции опущены (RIGHT OUTER JOIN или FOR EACH STATEMENT), однако, есть и некоторые новые;
- отлично подходит для разработки и даже тестирования: во время этапа разработки большинству требуется масштабируемое решение. SQLite, со своим богатым набором функций, может предоставить более чем достаточный функционал, при этом будучи достаточно простой для работы с одним файлом и связанной библиотекой.

PostgreSQL – это самая продвинутая РСУБД, ориентирующаяся в первую очередь на полное соответствие стандартам и расширяемость. PostgreSQL, или Postgres, пытается полностью соответствовать SQL-стандартам ANSI/ISO.

Преимущества:

- полная SQL-совместимость;
- сообщество: PostgreSQL поддерживается опытным сообществом 24/7;
- поддержка сторонними организациями: несмотря на очень продвинутые функции, PostgreSQL используется в многих инструментах, связанных с РСУБД;
- расширяемость: PostgreSQL можно программно расширить за счёт хранимых процедур;
- объектно-ориентированность: PostgreSQL – не только реляционная, но и объектно-ориентированная СУБД.

Сравним СУБД, чтобы определиться, какая больше подойдет для данной дипломной работы (таблица 3).

Таблица 3 – Сравнение средств реализации базы данных

Название СУБД	MySQL	SQLite	PostgreSQL
Большое количество типов данных	+	-	+
Популярность	+	+	-
Отказоустойчивость	-	-	+
Не требует удаленного сервера	-	+	-
Простота использования	+	+	-
Портативность	+	+	-

При сравнении СУБД для данного дипломного проекта была выбрана MySQL потому, что с ней приятно и удобно работать, что поможет лучше реализовать данный продукт.

					ДП.09.02.03.23.191.02.ПЗ	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

## 2 Техническое задание на разработку программного продукта

В начале разработки создавалось техническое задание, в котором указывались основные требования.

Для создания технического задания использовался стандарт ГОСТ 34.602-2020.

Согласно ГОСТ 34.602-2020 техническое задание должно включать следующие разделы:

1. общие сведения;
2. назначение и цели создания системы;
3. требования к системе в целом;
  - 3.1. требования к структуре и функционированию системы;
  - 3.2. требования к надежности;
  - 3.3. требования к безопасности;
  - 3.4. требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;
4. требования к документированию;
5. состав и содержание работ по созданию системы.

Техническое задание на разработку приложения представлено отдельным документом.

					ДП.09.02.03.23.191.02.ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

### 3 Проектирование

#### 3.1 Архитектура программного обеспечения

Архитектура программного обеспечения – это базовая организация системы, воплощенная в ее компонентах, их отношениях между собой и с окружением, а также принципы, определяющие проектирование и развитие системы.

Мобильное приложение «Таксопарк» использует клиент-серверную архитектуру (рисунок 1). Архитектура «клиент-сервер» предусматривает разделение процессов предоставления услуг и отправки запросов на них на разных компьютерах в сети, каждый из которых выполняет свои задачи независимо от других.

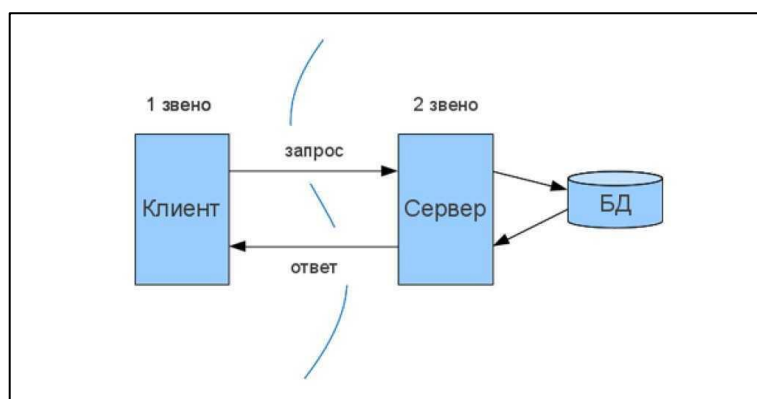


Рисунок 1 – Архитектура мобильного приложения «Таксопарк»

Сервер – специальное системное оборудование, которое предназначается для разрешения определенного круга задач по процессу выполнения программных кодов. Он выполняет работы сервисного обслуживания по клиентским запросам, предоставляет пользователям доступ к определенным системным ресурсам, сохраняет данные или БД.

Параметры, которые могут реализоваться на стороне сервера:

1. хранение, защита и доступ к данным;
2. работа с поступающими клиентскими запросами;

3. процесс отправки ответа клиенту.

Клиент – локальный компьютер на стороне виртуального пользователя, который выполняет отправку запроса к серверу для возможности предоставления данных или выполнения определенной группы системных действий.

Параметры, которые могут реализоваться на стороне клиента:

1. площадка по предоставлению пользовательского графического интерфейса;
2. формулировка запроса к серверу и его последующая отправка;
3. получение итогов запроса и отправка дополнительной группы команд (запросы на добавление, обновление информации, удаление группы данных).

Таким образом, в архитектуре «клиент-сервер» клиент посылает запрос на предоставление данных и получает только те данные, которые действительно были затребованы.

### 3.2 Функциональное проектирование

Проектирование мобильного приложения происходит при помощи CASE средств, которые помогают обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов.

На рисунке 2 изображена диаграмма прецедентов, которая показывает структурную схему приложения «Таксопарк» для пользователя «Водитель», «Клиент», «Администратор».

Данная схема отображает действия, выполняемые водителем, клиентом и администратором. «Водитель», «Клиент» и «Администратор» являются – актерами.

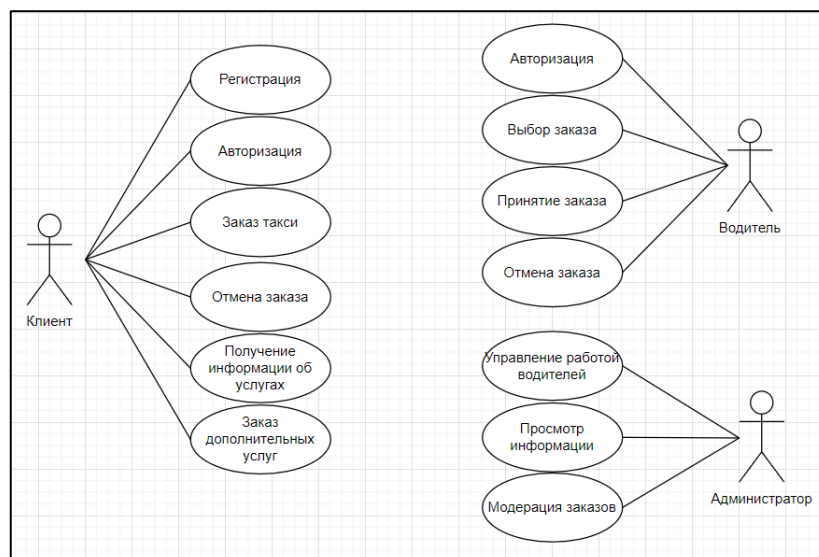


Рисунок 2 – Диаграмма прецедентов

Таким образом, представленные диаграммы демонстрируют взаимодействие основных объектов мобильного приложения и их действия.

Диаграмма последовательности является видом диаграмм взаимодействия языка UML, которые описывают отношения объектов в различных условиях. Условия взаимодействия задаются сценарием, полученным на этапе разработки диаграмм вариантов использования.

Сценарий заказа такси в приложении показан на рисунке 3.

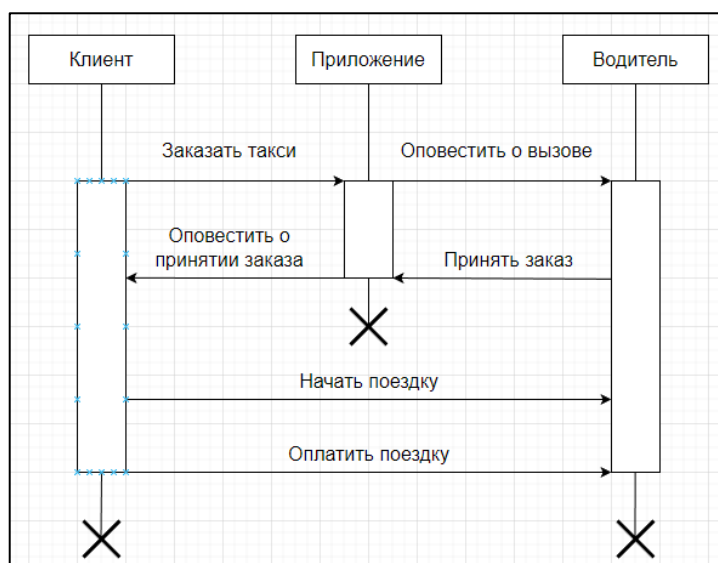


Рисунок 3 –Диаграмма последовательности

Диаграмма компонентов (рисунок 4) показывает разбиение программной системы на структурные компоненты и связи между компонентами.

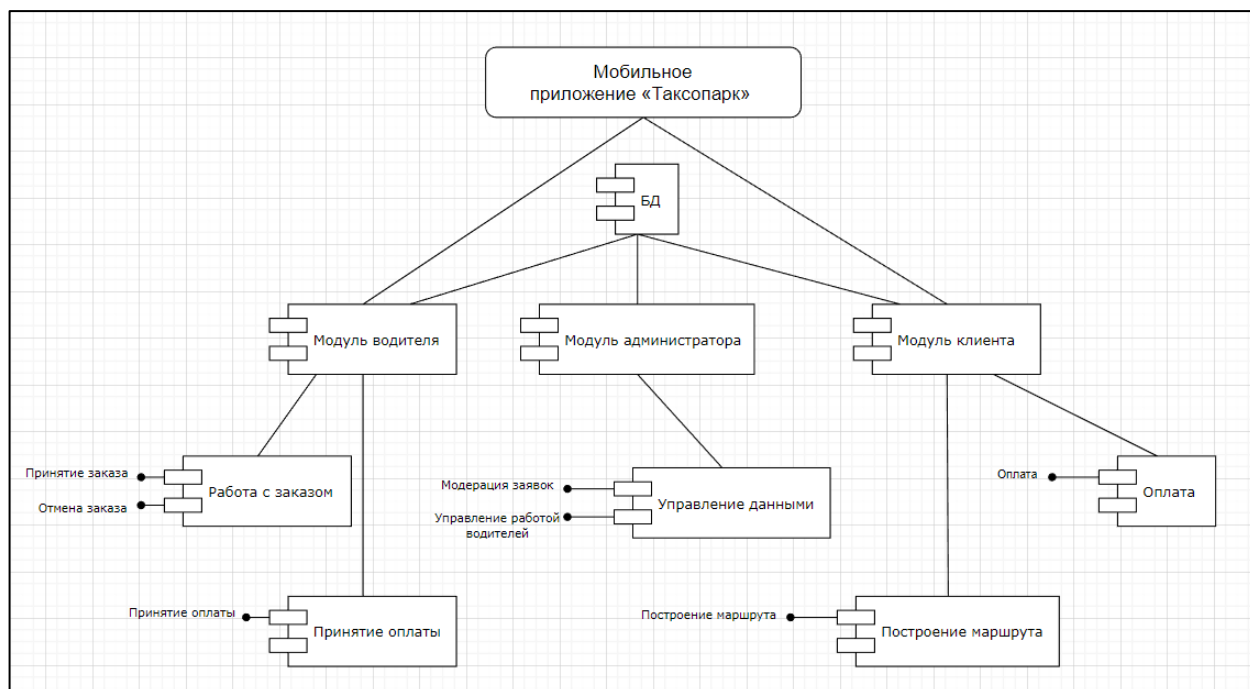


Рисунок 4 – Диаграмма компонентов

На рисунке 5 представлена диаграмма развертывания – это тип диаграммы, которая определяет физическое оборудование, на котором будет работать программная система. Он также определяет способ развертывания программного обеспечения на базовом оборудовании. Он отображает программные части системы на устройство, которое будет выполнять его.



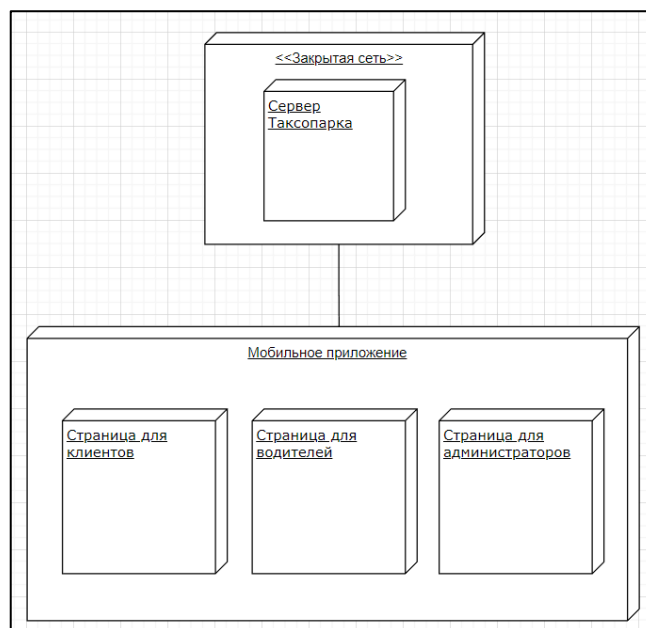


Рисунок 5 – Диаграмма развертывания

Контекстная диаграмма – это модель, представляющая систему как набор иерархических действий, в которой каждое действие преобразует некоторый объект или набор объектов.

На рисунке 6 изображена контекстная диаграмма, на которой показаны входные данные, управление, механизм, выходные данные и функция.

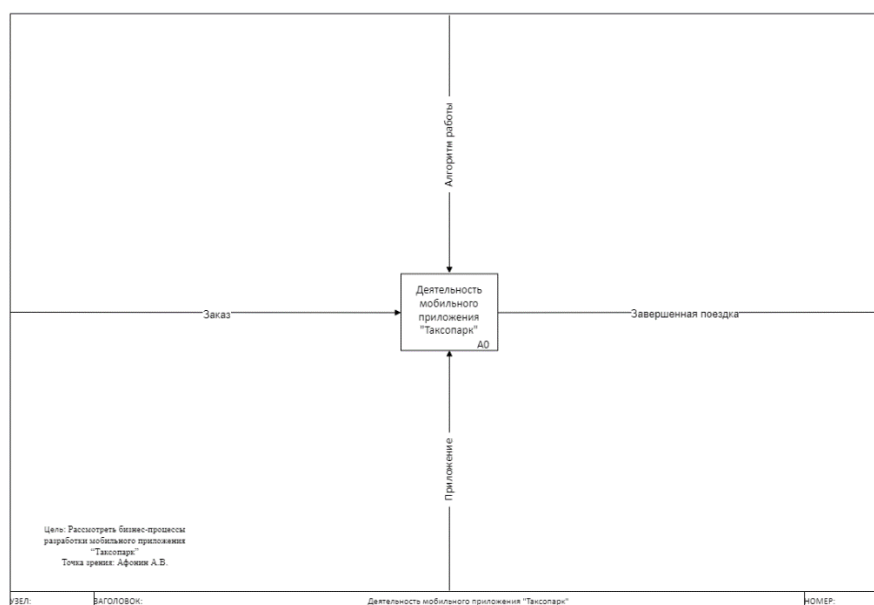


Рисунок 6 – Контекстная диаграмма

На рисунке 7 показана диаграмма декомпозиций, которая расписывает функции программы.

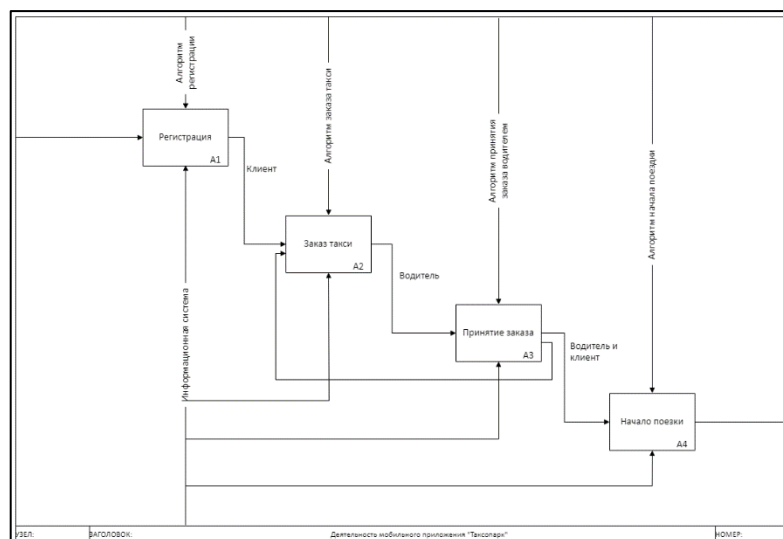


Рисунок 7 – Диаграмма декомпозиции составления расписания

Также в ходе работы была разработана диаграмма классов, приведенная на рисунке 8, которая отображает внутреннюю работу мобильного приложения.

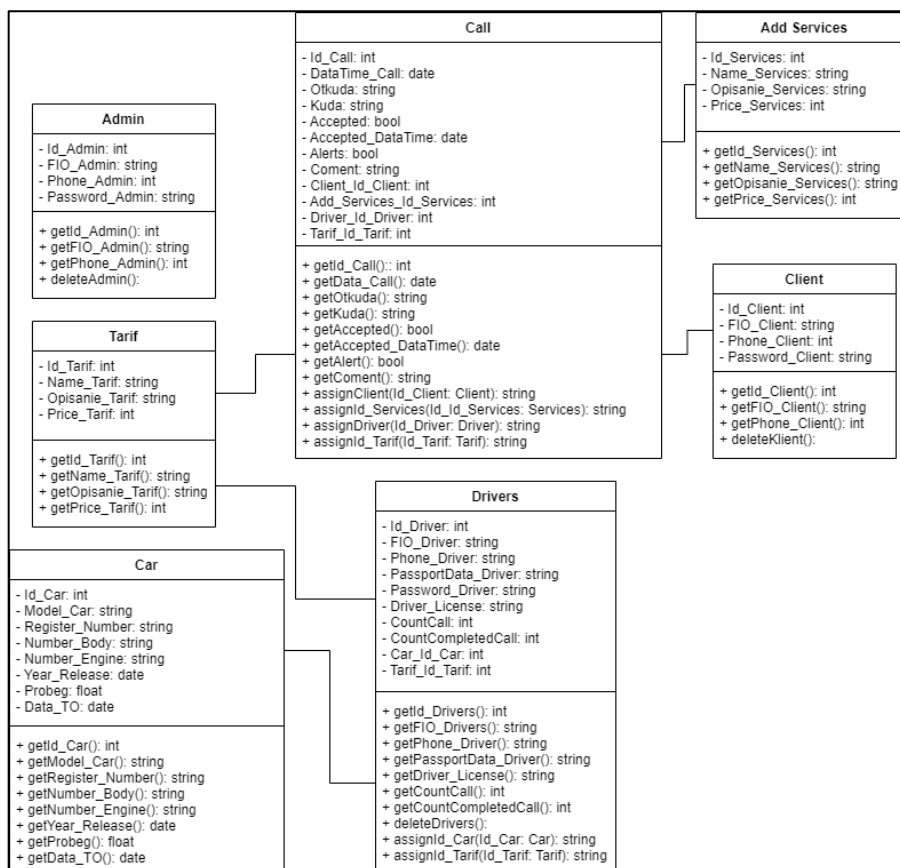


Рисунок 8 – Диаграмма классов

Диаграмма потоков данных, представленная на рисунке 9. Так называется методология графического структурного анализа, описывающая внешние по отношению к системе источники, и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.

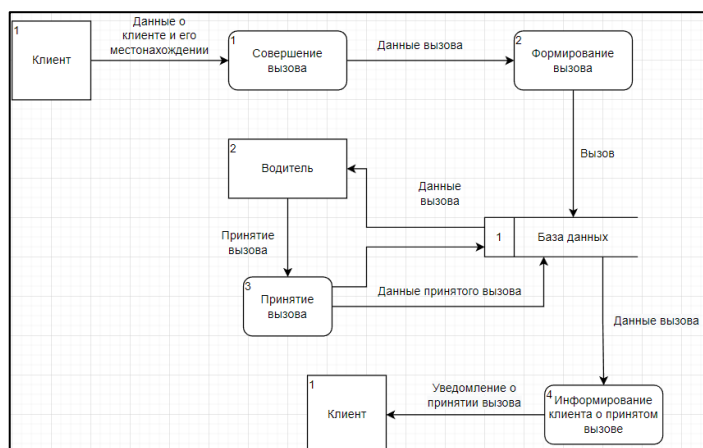


Рисунок 9 – Диаграмма потоков данных

Функциональное проектирование представляет наиболее общий подход к описанию систем. Определяются граничные условия и желательные входы, и выходы, составляется подробный перечень функций или операций, которые должны выполняться. При функциональном проектировании осуществляется синтез структуры и определяются основные параметры объекта и его составных частей (элементов), оцениваются показатели эффективности и качества процессов функционирования.

### 3.3 Проектирование базы данных

Перед разработкой базы данных есть необходимость в инфологическом моделировании. Результатом такого моделирования является инфологическая модель базы данных, изображенная на рисунке 10, которая наглядно показывает объект, его свойства и отношения между другими объектами.

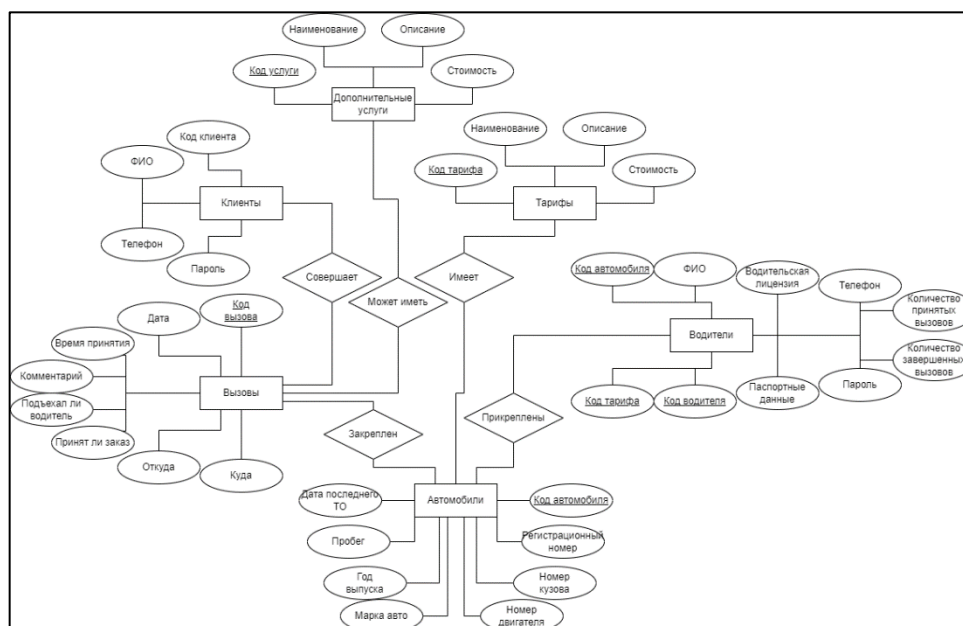


Рисунок 10 – Инфологическая модель БД

На рисунке 11 представлена даталогическая модель. Модель предполагает определение состава и взаимосвязей таблиц, отражающих содержание

информационных сущностей инфологической модели в терминах конкретной СУБД.

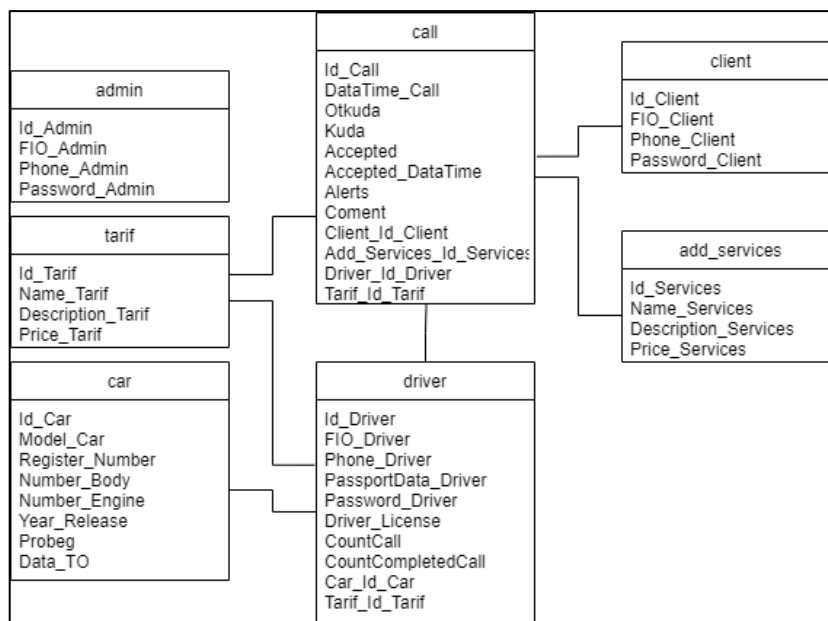


Рисунок 11 – Даталогическая модель

На рисунке 12 представлена ER-модель базы данных. Представлены таблицы, связи между ними и типы данных.

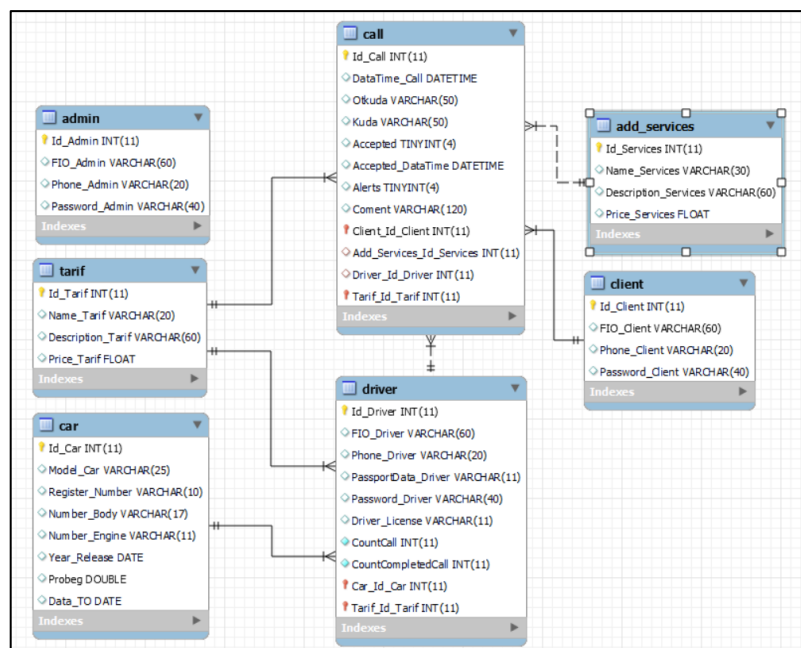


Рисунок 12 – ER-модель базы данных

Далее представим независимую сущность таблиц базы данных. Дальше будут отображены не все свойства сущностей, поскольку не являются необходимыми для функционирования дипломного проекта.

Таблица 4 – Таблица «client»

Поле	Тип данных	Описание
Id_Client	int	Код клиента
FIO_Client	Varchar(60)	ФИО
Phone_Client	Varchar(20)	Номер телефона
Password_Client	Varchar(40)	Пароль

Таблица 5 – Таблица «add\_services»

Поле	Тип данных	Описание
Id_Services	int	Код услуги
Name_Services	Varchar(30)	Название
Opisanie_Services	Varchar(60)	Описание
Price_Services	Float	Цена за доп. услугу

Таблица 6 – Таблица «call»

Поле	Тип данных	Описание
Id_Call	int	Код вызова
DataTime_Call	DateTime	Дата и время вызова
Otkuda	Varchar(30)	Место начала поездки
Kuda	Varchar(30)	Место окончания поездки
Accepted	tinyint(4)	Принят ли заказ
Accepted_DataTime	DateTime	Время принятия заказа
Alerts	tinyint(4)	Приехал ли водитель
Comment	Varchar(120)	Комментарий от клиента
Client_Id_Client	int	Код клиента
Add_Services_Id_Services	int	Код доп. услуги
Drivers_Id_Drivers	int	Код водителя
Tarif_Id_Tarif	int	Выбранный тариф

Таблица 7 – Таблица «car»

Поле	Тип данных	Описание
Id_Car	int	Код автомобиля
Model_Car	Varchar(20)	Марка автомобиля
Register_Number	Varchar(10)	Регистрационный номер
Number_Body	Varchar(17)	Номер кузова

Продолжение таблицы 7

Number_Engine	Varchar(11)	Номер двигателя автомобиля
Year_Release	Date	Год выпуска автомобиля
Probeg	Float	Пробег автомобиля
Data_TO	Date	Дата последнего ТО

Таблица 8 – Таблица «drivers»

Поле	Тип данных	Описание
Id_Drivers	int	Код водителя
FIO_Drivers	Varchar(40)	ФИО
Phone_Drivers	Varchar(20)	Телефон
Passport_Data_Drivers	int	Паспортные данные
Password_Drivers	Varchar(40)	Пароль
Driver_License	Varchar(11)	Водительское удостоверение
CountCall	int	Число взятых заказов
CountCompletedCall	int	Число завершённых заказов
Tarif_Id_Tarif	int	Код тарифа
Car_Id_Car	int	Код автомобиля

Таблица 9 – Таблица «tarif»

Поле	Тип данных	Описание
Id_Tarif	int	Код тарифа
Name_Tarif	Varchar(30)	Название
Opisanie_Tarif	Varchar(45)	Описание
Price_Tarif	Float	Цена за тариф

Таблица 10 – Таблица «admin»

Поле	Тип данных	Описание
Id_Admin	int	Код администратора
FIO_Admin	Varchar(60)	ФИО
Phone_Admin	Varchar(20)	Номер телефона
Password_Admin	Varchar(40)	Пароль

Для разработки ER-модели был использован MySQL Workbench 8.0 CE.

После завершения разработки прототипа базы данных и таблиц, находящихся в ней, можно перейти к проектированию интерфейса приложения.

### 3.4 Проектирование пользовательского интерфейса

Пользовательский интерфейс (UI) - это взаимодействие между пользователем и компьютерной системой, которое позволяет пользователю управлять функциональностью и настройками программного обеспечения с помощью графических элементов, таких как кнопки, текстовые поля, меню и диалоговые окна.

Разработка удобного пользовательского интерфейса – это один из важнейших этапов в процессе создания программного продукта.

На рисунках 13, 14, 15 представлен прототип интерфейса авторизации, регистрации и формы заказа такси в приложении таксопарка.

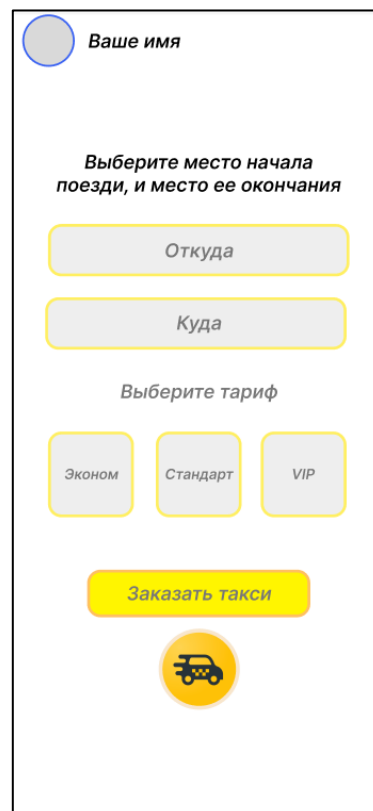
Рисунок 13 – Интерфейс авторизации





The registration interface features a yellow circular logo at the top with the word "TAXI" in black and a black car icon below it. Below the logo, the word "Регистрация" is written in bold. The form consists of several input fields: "Введите ваше ФИО", "Номер телефона", "Пол", "Возраст", "Придумайте пароль", and "Повторите пароль". A yellow "Регистрация" button is positioned below these fields. At the bottom, a link reads "Есть аккаунт? Просто войдите!".

Рисунок 14 – Интерфейс регистрации



The taxi ordering interface starts with a profile section labeled "Ваше имя" next to a blue circular placeholder. Below this, the instruction "Выберите место начала поездки, и место ее окончания" is followed by two input fields: "Откуда" and "Куда". Another instruction "Выберите тариф" is followed by three buttons: "Эконом", "Стандарт", and "VIP". A yellow "Заказать такси" button is located below the tariff selection. At the bottom, there is a yellow circular button with a black car icon.

Рисунок 15 – Интерфейс заказа такси

## 4. Реализация программного обеспечения

### 4.1 Кодирование программного обеспечения

Разработка удобного пользовательского интерфейса – это один из важнейших этапов в процессе создания приложения «Таксопарк».

В разрабатываемом мобильном приложении используется формат - \*.xaml и \*.cs. Файл с форматом \*.xaml содержит в себе дизайнерскую часть, а \*.cs – программную часть, написанную на языке C#.

Благодаря фреймворку Xamarin были созданы страницы для приложения (рисунок 16).

```
<StackLayout x:Name="MainStackLayout">
  <Grid>
    <Image x:Name="image1"
      Margin="0,20,0,0"
      HorizontalOptions="Center"
      HeightRequest="175"/>
    <ImageButton
      Margin="20,20,0,0"
      x:Name="buttonVodAvtoriz"
      Clicked="VoditelButton_Clicked"
      HeightRequest="60"
      BackgroundColor="White"
      HorizontalOptions="Start"
      VerticalOptions="Start"/>
    <ImageButton
      Margin="0,20,20,0"
      x:Name="buttonAdminAvtoriz"
      Clicked="AdminButton_Clicked"
      HeightRequest="60"
      BackgroundColor="White"
      HorizontalOptions="End"
      VerticalOptions="Start"/>
  </Grid>

  <Label Text="Авторизация" FontSize="30" Margin="0,0,0,0" HorizontalTextAlignment="Center" FontAttributes="Bold" TextColor="Black" />

  <Label Text="Введите ваш номер телефона" FontSize="20" Margin="0,20,0,0" HorizontalTextAlignment="Center"/>
  <Entry x:Name="nameInput1" HorizontalTextAlignment="Center" WidthRequest="350" HorizontalOptions="Center"/>

  <Label Text="Введите пароль" FontSize="20" Padding="30,0,30,0" HorizontalTextAlignment="Center"/>
  <Entry x:Name="nameInput2" HorizontalTextAlignment="Center" WidthRequest="350" HorizontalOptions="Center" IsPassword="True"/>
</StackLayout>
```

Рисунок 16 – Код разметки xaml

### 4.2 Разработка базы данных

Разработка базы данных мобильного приложения «Таксопарк» реализовывалась в СУБД MySQL База данных состоит из 7 таблиц (рисунок 17).

<input type="checkbox"/>	<b>add_services</b>						
<input type="checkbox"/>	<b>admin</b>						
<input type="checkbox"/>	<b>call</b>						
<input type="checkbox"/>	<b>car</b>						
<input type="checkbox"/>	<b>client</b>						
<input type="checkbox"/>	<b>driver</b>						
<input type="checkbox"/>	<b>tarif</b>						

Рисунок 17 – СУБД MySQL и БД

Структуры таблиц соответствуют схеме базы данных из пункта 4.3.  
Таблица «add\_services» на рисунке 18.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
<input type="checkbox"/> 1	<b><u>Id_Services</u></b>	int(11)			Нет	Нет	AUTO_INCREMENT
<input type="checkbox"/> 2	<b>Name_Services</b>	varchar(30)	utf8_general_ci		Да	NULL	
<input type="checkbox"/> 3	<b>Description_Services</b>	varchar(60)	utf8_general_ci		Да	NULL	
<input type="checkbox"/> 4	<b>Price_Services</b>	float			Да	NULL	

Рисунок 18 – Структура таблицы «add\_services»

Таблица «admin» на рисунке 19.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
<input type="checkbox"/> 1	<b><u>Id_Admin</u></b>	int(11)			Нет	Нет	AUTO_INCREMENT
<input type="checkbox"/> 2	<b>FIO_Admin</b>	varchar(60)	utf8_general_ci		Да	NULL	
<input type="checkbox"/> 3	<b>Phone_Admin</b>	varchar(20)	utf8_general_ci		Да	NULL	
<input type="checkbox"/> 4	<b>Password_Admin</b>	varchar(40)	utf8_general_ci		Да	NULL	

Рисунок 19 – Структура таблицы «admin»

Таблица «call» на рисунке 20.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
<input type="checkbox"/>	1 <u><b>Id_Call</b></u>	int(11)			Нет	<i>Нет</i>	AUTO_INCREMENT
<input type="checkbox"/>	2 <b>DataTime_Call</b>	datetime			Да	NULL	
<input type="checkbox"/>	3 <b>Otkuda</b>	varchar(50)	utf8_general_ci		Да	NULL	
<input type="checkbox"/>	4 <b>Kuda</b>	varchar(50)	utf8_general_ci		Да	NULL	
<input type="checkbox"/>	5 <b>Accepted</b>	tinyint(4)			Да	NULL	
<input type="checkbox"/>	6 <b>Accepted_DataTime</b>	datetime			Да	NULL	
<input type="checkbox"/>	7 <b>Alerts</b>	tinyint(4)			Да	NULL	
<input type="checkbox"/>	8 <b>Coment</b>	varchar(120)	utf8_general_ci		Да	NULL	
<input type="checkbox"/>	9 <u><b>Client_Id_Client</b></u>	int(11)			Нет	<i>Нет</i>	
<input type="checkbox"/>	10 <b>Add_Services_Id_Services</b>	int(11)			Да	NULL	
<input type="checkbox"/>	11 <b>Driver_Id_Driver</b>	int(11)			Да	NULL	
<input type="checkbox"/>	12 <u><b>Tarif_Id_Tarif</b></u>	int(11)			Нет	<i>Нет</i>	

Рисунок 20 – Структура таблицы «call»

Таблица «car» на рисунке 21.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
<input type="checkbox"/>	1 <u><b>Id_Car</b></u>	int(11)			Нет	<i>Нет</i>	AUTO_INCREMENT
<input type="checkbox"/>	2 <b>Model_Car</b>	varchar(25)	utf8_general_ci		Да	NULL	
<input type="checkbox"/>	3 <b>Register_Number</b>	varchar(10)	utf8_general_ci		Да	NULL	
<input type="checkbox"/>	4 <b>Number_Body</b>	varchar(17)	utf8_general_ci		Да	NULL	
<input type="checkbox"/>	5 <b>Number_Engine</b>	varchar(11)	utf8_general_ci		Да	NULL	
<input type="checkbox"/>	6 <b>Year_Release</b>	date			Да	NULL	
<input type="checkbox"/>	7 <b>Probeg</b>	double			Да	NULL	
<input type="checkbox"/>	8 <b>Data_TO</b>	date			Да	NULL	

Рисунок 21 – Структура таблицы «car»

Таблица «client» на рисунке 22.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
<input type="checkbox"/>	1 <u><b>Id_Client</b></u>	int(11)			Нет	<i>Нет</i>	AUTO_INCREMENT
<input type="checkbox"/>	2 <b>FIO_Client</b>	varchar(60)	utf8_general_ci		Да	NULL	
<input type="checkbox"/>	3 <b>Phone_Client</b>	varchar(20)	utf8_general_ci		Да	NULL	
<input type="checkbox"/>	4 <b>Password_Client</b>	varchar(40)	utf8_general_ci		Да	NULL	

Рисунок 22 – Структура таблицы «client»

Таблица «driver» на рисунке 23.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
<input type="checkbox"/> 1	<u><b>Id_Driver</b></u>	int(11)			Нет	Нет	AUTO_INCREMENT
<input type="checkbox"/> 2	<b>FIO_Driver</b>	varchar(60)	utf8_general_ci		Да	NULL	
<input type="checkbox"/> 3	<b>Phone_Driver</b>	varchar(20)	utf8_general_ci		Да	NULL	
<input type="checkbox"/> 4	<b>PassportData_Driver</b>	varchar(11)	utf8_general_ci		Да	NULL	
<input type="checkbox"/> 5	<b>Password_Driver</b>	varchar(40)	utf8_general_ci		Да	NULL	
<input type="checkbox"/> 6	<b>Driver_License</b>	varchar(11)	utf8_general_ci		Да	NULL	
<input type="checkbox"/> 7	<b>CountCall</b>	int(11)			Нет	Нет	
<input type="checkbox"/> 8	<b>CountCompletedCall</b>	int(11)			Нет	Нет	
<input type="checkbox"/> 9	<u><b>Car_Id_Car</b></u>	int(11)			Нет	Нет	
<input type="checkbox"/> 10	<u><b>Tarif_Id_Tarif</b></u>	int(11)			Нет	Нет	

Рисунок 23 – Структура таблицы «driver»

Таблица «tarif» на рисунке 24.

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно
<input type="checkbox"/> 1	<u><b>Id_Tarif</b></u>	int(11)			Нет	Нет	AUTO_INCREMENT
<input type="checkbox"/> 2	<b>Name_Tarif</b>	varchar(20)	utf8_general_ci		Да	NULL	
<input type="checkbox"/> 3	<b>Description_Tarif</b>	varchar(60)	utf8_general_ci		Да	NULL	
<input type="checkbox"/> 4	<b>Price_Tarif</b>	float			Да	NULL	

Рисунок 24 – Структура таблицы «tarif»

После создания всех таблиц и связей в базе данных, она готова к работе.

### 4.3 Разработка программного продукта

Для разработки мобильного приложения был использован Xamarin.Forms. Данный фреймворк позволил сделать удобный интерфейс для клиентов и водителей.

Вся основная работа построена на sql запросах в \*.cs:

- 1) Вывод данных текущих заказов клиента (рисунок 25).
- 2) Вывод данных всех активных заказов (рисунок 26).
- 3) Вывод данных о всех водителях (рисунок 27).

```

DB db = new DB();
db.openConnection();
MySQLCommand command = new MySQLCommand("SELECT * FROM 'call' WHERE 'Id_Call'=@id", db.getConnection());
command.Parameters.Add("@id", MySqlDbType.Int32).Value = Convert.ToInt32(call[0]);
MySQLDataReader reader = command.ExecuteReader();
call.Clear();
if (reader.HasRows)
{
    while (reader.Read())
    {
        for (int i = 0; i < reader.FieldCount; i++)
        {
            call.Add(reader[i].ToString());
        }
    }
}
else
{
    timer = false;
    OnTimerTick();
}

if (call[4] != "" && accepted == false)
{
    Profile.IsVisible = true;
    accepted = true;
    ProfileLabel.IsVisible = true;
    stateCall.Text = "Ваш заказ был принят";
    UpdateTable();
}
if (call[6] != "" && alert == false)
{
    Profile.IsVisible = true;
    ProfileLabel.IsVisible = true;
    alert = true;
    stateCall.Text = "Водитель ожидает вас";
}
db.closeConnection();
OnTimerTick();

```

Рисунок 25 – Код вывода текущего заказа клиента

```

DB db = new DB();
db.openConnection();
MySQLCommand command = new MySQLCommand("SELECT 'DataTime_Call', 'Otkuda', 'Kuda', 'FIO_Client', 'FIO_Driver', 'Name_Services' * * "
"FROM 'call', 'client', 'driver', 'add_services' * * "
"WHERE 'client_id_client'='Id_client' and 'Add_Services_Id_Services'='Id_Services' and 'Driver_Id_Driver'='Id_Driver' and 'Accepted' = 1", db.getConnection());
MySQLDataReader reader = command.ExecuteReader();

if (reader.HasRows)
{
    while (reader.Read())
    {
        Grid grid = new Grid
        {
            HeightRequest = 45,
            RowSpacing = -45,
            ColumnSpacing = 4,
            BackgroundColor = Color.DarkOrange,
            RowDefinitions =
            {
                new RowDefinition { Height = new GridLength(1, GridUnitType.Star) },
            },
            ColumnDefinitions =
            {
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) }
            }
        };
        for (int i = 0; i < 6; i++)
        {
            string textcall = reader[i].ToString();
            Label label1 = new Label
            {
                FontSize = 12,
                Text = textcall,
            };
            label1.VerticalTextAlignment = TextAlignment.Center;
            label1.HorizontalTextAlignment = TextAlignment.Center;
            label1.TextColor = Color.Black;
            label1.BackgroundColor = Color.LightGray;
            grid.Children.Add(label1, i, j);
        }
        j++;
        BoxView box = new BoxView
        {
            Margin = -5,
            HeightRequest = 2
        };
        box.BackgroundColor = Color.DarkOrange;
        callView.Children.Add(grid);
        callView.Children.Add(box);
    }
}
db.closeConnection();

```

Рисунок 26 – Код вывода активных заказов

```

DB db = new DB();
db.openConnection();
MySQLCommand command = new MySqlCommand("SELECT 'FIO_Driver', 'Phone_Driver', 'PassportData_Driver', 'Driver_License', 'Model_Car', 'Name_Tarif' " +
"FROM 'driver', 'tarif', 'car' WHERE 'Tarif_Id_Tarif'='Id_Tarif' and 'Car_Id_Car'='Id_Car'", db.getConnection());
MySQLDataReader reader = command.ExecuteReader();

if (reader.HasRows)
{
    while (reader.Read())
    {
        pickerVod.Items.Add(reader[0].ToString());

        Grid grid = new Grid
        {
            HeightRequest= 45,
            RowSpacing = -45,
            ColumnSpacing = 4,
            BackgroundColor = Color.DarkOrange,
            RowDefinitions =
            {
                new RowDefinition { Height = new GridLength(1, GridUnitType.Star) },
            },
            ColumnDefinitions =
            {
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
                new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) }
            }
        };
        for (int i = 0; i < 6; i++)
        {
            string textcall = reader[i].ToString();
            Label label1 = new Label
            {
                FontSize = 12,
                Text = textcall,
            };
            label1.VerticalTextAlignment = TextAlignment.Center;
            label1.HorizontalTextAlignment = TextAlignment.Center;
            label1.TextColor = Color.Black;
            label1.BackgroundColor = Color.LightGray;
            grid.Children.Add(label1, i, j);
        }
        j++;
    }
}

```

Рисунок 27 – Код вывода данных водителей

Реализация программного кода авторизации представлена в Приложение А. Вывод текущих заказов реализован с помощью С#. Данные сохраняются в базу данных. Программный код реализации просмотра заказов представлен в приложении Б

Таким образом, были реализованы следующие функции на соответствующих страницах:

1) кнопка – «Заказать такси», которая вызывает страницу, содержащую информацию о заказе и отправляет заказ водителю (рисунок 28);

```

DB db = new DB();
db.openConnection();
MySQLCommand command = new MySqlCommand("INSERT INTO 'call' +
    '(' + Id_Call + ', ' + DateTime_Call + ', ' + Otkuda + ', ' + Kuda + ', ' + Accepted + ', ' + Accepted_DateTime + ', ' + Alerts + ', ' + Coment + ', ' + Client_Id_Client + ', ' + Add_Services_Id_Services + ', ' + Driver_Id_Driver + ', ' + Tarif_Id_Tarif + ')" +
    " VALUES (default,@DateTime_Call,@Otkuda,@Kuda,null,null,null,@comment,@id,@Add_Services,null,@Tarif)", db.getConnection());
command.Parameters.Add("@DateTime_Call", MySqlDbType.DateTime).Value = dateTime;
command.Parameters.Add("@Otkuda", MySqlDbType.VarChar).Value = otkuda;
command.Parameters.Add("@Kuda", MySqlDbType.VarChar).Value = kuda;
command.Parameters.Add("@comment", MySqlDbType.VarChar).Value = result;
command.Parameters.Add("@id", MySqlDbType.Int32).Value = Convert.ToInt32(userData[0]);
if (addServices == 0)
{
    command.Parameters.Add("@Add_Services", MySqlDbType.Int32).Value = null;
}
else command.Parameters.Add("@Add_Services", MySqlDbType.Int32).Value = addServices;
command.Parameters.Add("@Tarif", MySqlDbType.Int32).Value = tarif;
if (command.ExecuteNonQuery() == 1)
{
    db.closeConnection();
    OpenTrackingPage();
    await Navigation.PushAsync(new TrackingPage(userData, call));
}

```

Рисунок 28 – Код заказа такси

2) кнопка – «Отменить заказ», которая при нажатии отправляет пользователя на заказ такси и отменяет текущий заказ (рисунок 29);

```

timer = false;
DB db = new DB();
await DisplayAlert("Оповещение", "Вы отменили свой заказ", "OK");
db.openConnection();
MySQLCommand command = new MySqlCommand("DELETE FROM 'call' WHERE 'Id_Call' = @id", db.getConnection());
command.Parameters.Add("@id", MySqlDbType.Int32).Value = Convert.ToInt32(call[0]);
command.ExecuteNonQuery();
db.closeConnection();
await Navigation.PopAsync();

```

Рисунок 29 – Код отмены заказа клиентом

3) кнопка – «Принять заказ», которая принимает заказ клиента и информирует его об этом (рисунок 30);

```

DateTime dateTime = new DateTime();
dateTime = DateTime.UtcNow;

db.openConnection();
MySQLCommand command2 = new MySqlCommand("UPDATE 'call' SET 'Accepted'= 1, 'Accepted_DateTime'=@DateTime_Call, 'Driver_Id_Driver'=@idDriver WHERE 'Id_Call' = @id", db.getConnection());
command2.Parameters.Add("@id", MySqlDbType.Int32).Value = Convert.ToInt32(acceptedCallData[0]);
command2.Parameters.Add("@DateTime_Call", MySqlDbType.DateTime).Value = dateTime;
command2.Parameters.Add("@idDriver", MySqlDbType.Int32).Value = Convert.ToInt32(voditelData[0]);
command2.ExecuteNonQuery();
DisplayAlert("Отлично", "Вы приняли заказ", "OK");
db.closeConnection();
Accepted.IsEnabled = false;
Alerts.IsEnabled = true;
CancelButton.IsEnabled = true;

Otkuda.Text = acceptedCallData[2];
Kuda.Text = acceptedCallData[3];
IncrementCall();

```

Рисунок 30 – Код принятия заказа клиентом

4) кнопка – «Оповестить о том, что вы подъехали», которая информирует клиента о подъехавшем такси (рисунок 31);



```

DB db = new DB();
db.openConnection();
MySQLCommand command = new MySqlCommand("UPDATE `call` SET `Alerts`= 1 WHERE `Id_Call` =@id", db.getConnection());
command.Parameters.Add("@id", MySqlDbType.Int32).Value = Convert.ToInt32(acceptedCallData[0]);
command.ExecuteNonQuery();
DisplayAlert("Отлично", "Оповещение отправленно", "Ок");
Alerts.IsEnabled = false;
Finished.IsEnabled = true;
db.closeConnection();

```

Рисунок 31 – Код оповещения клиента

5) кнопка – «Завершить заказ», которая отмечает окончание маршрута (рисунок 32);

```

DB db = new DB();
db.openConnection();
MySQLCommand command = new MySqlCommand("DELETE FROM `call` WHERE `Id_Call` = @id", db.getConnection());
command.Parameters.Add("@id", MySqlDbType.Int32).Value = Convert.ToInt32(acceptedCallData[0]);
command.ExecuteNonQuery();
db.closeConnection();
DisplayAlert("Отлично", "Заказ завершен", "Ок");
IncrementCompletedCall();

```

Рисунок 32 – Код завершения заказ

6) кнопка – «Отменить заказ», которая отменяет принятый заказ и информирует клиента об этом (рисунок 33);

```

Comment.IsEnabled = false;
Comment.IsVisible = false;
DB db = new DB();
db.openConnection();
MySQLCommand command = new MySqlCommand("UPDATE `call` SET `Accepted`=null,`Accepted_DataTime`=null,`Alerts`=null," +
"|`Finished`=null,`Driver_Id_Driver`=null WHERE `Id_Call` = @id", db.getConnection());
command.Parameters.Add("@id", MySqlDbType.Int32).Value = Convert.ToInt32(acceptedCallData[0]);
command.ExecuteNonQuery();
DisplayAlert("Оповещение", "Вы отменили свой заказ", "Ок");
db.closeConnection();
UpdateCallView();

```

Рисунок 33 – Код отмены заказа водителем

7) Кнопка «Профиль водителя», которая доступна только на странице принятия заказов при нажатии на которую откроется информация о водителе (рисунок 34).

```

DB db = new DB();
db.openConnection();
MySQLCommand command = new MySQLCommand("SELECT 'FIO_Driver', 'Phone_Driver', 'Model_Car', 'Register_Number', 'Year_Release', 'Data_T0', 'CountCall', " +
    = 'CountCompletedCall' FROM 'driver', 'car' WHERE 'Car_Id_Car' = 'Id_Car' and 'Id_Driver' = @id", db.getConnection());
command.Parameters.Add("@id", MySqlDbType.Int32).Value = id;
MySQLDataReader reader = command.ExecuteReader();

if (reader.HasRows)
{
    while (reader.Read())
    {
        Name.Text = reader[0].ToString();
        Phone.Text = reader[1].ToString();
        Marka.Text = reader[2].ToString();
        Number.Text = reader[3].ToString();
        Year.Text = reader[4].ToString();
        YearT0.Text = reader[5].ToString();
        CountCall.Text = reader[6].ToString();
        CountCompletedCall.Text = reader[7].ToString();
    }
}

```

Рисунок 34 – Страница личного кабинета

Приложение будет использоваться для заказа такси, легко в освоении и проста в использовании. Для работы приложения используется мобильный телефон на системе Android. Приложение разработано для использования в повседневной жизни.

## 5 Документирование программного обеспечения

Процесс написания документов программного обеспечения – важный этап в процессе создания и эксплуатации программного обеспечения. Именно с руководства пользователя начинается свое знакомство с программным продуктом.

### 5.1 Руководство пользователя

Для запуска приложения требуется предварительно его установить, скачав соответствующий файл установки, имеющий формат \*.apk. После установки приложение будет готово к работе, для его запуска потребуется кликнуть по иконке приложения.

Мобильное приложение «Таксопарк» имеет простой и понятный интерфейс, что позволяет пользователю легко использовать его.

При открытии приложения открывается страница авторизации (рисунок 35). Код авторизации представлен в приложении А. Чтобы получить доступ к функциям необходимо пройти авторизацию. Если аккаунта не существует, пользователь может сам зарегистрироваться клиент (рисунок 36).

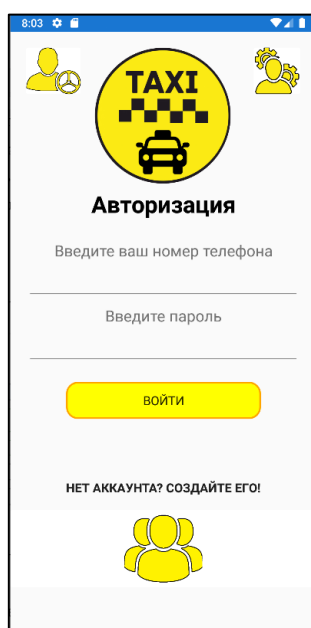


Рисунок 35 – Страница авторизации

					ДП.09.02.03.23.191.02.ПЗ	Лист
						35
Изм.	Лист	№ докум.	Подпись	Дата		

Рисунок 36 – Страница регистрации

После ввода данных авторизации пользователь попадает на страницу соответствующей ему роли. Если пользователь идентифицирован под ролью «клиента», то откроется страница заказа такси (рисунок 37), где необходимо выбрать: место отбытия, место прибытия, тариф и дополнительные услуги и по необходимости написать комментарий к заказу для водителя.

Рисунок 37 – Главная страница заказа такси

					ДП.09.02.03.23.191.02.ПЗ	Лист
						36
Изм.	Лист	№ докум.	Подпись	Дата		

После заказа такси, осуществляется переход на страницу с текущим заказом, где он может отслеживать статус поездки, либо отменить ее (рисунок 38).

При нажатии на отмену заказа, осуществляется переход на страницу заказа. Если заказ будет принят, то высветится информация о водителе, автомобиле и его номере (рисунок 39).

3:19

Пороник Антон

Информация о вашем заказе

Откуда:	Жукова 7
Куда:	Ямская 49
За вами приедет	Неизвестно
На автомобиле:	Неизвестно
Номер авто:	Неизвестно

Ваш заказ еще никто не принял

ОТМЕНИТЬ ЗАКАЗ

Рисунок 38 – Страница отслеживания заказа

3:15

Артем Афонин

Информация о вашем заказе

Откуда:	Лескова 6
Куда:	Ленина 10
За вами приедет	Луковников Иван
На автомобиле:	Renault
Номер авто:	K449YX31

Ваш заказ был принят

ОТМЕНИТЬ ЗАКАЗ

Информация о водителе

[Silhouette of a person]

Рисунок 39 – Страница с принятым заказом

Также если ваш заказ был принят появляется возможность перейти на форму с информацией о водителе, принявшем ваш заказ (рисунок 40). Эта страница содержит информацию не только о водителе, а также о его автомобиле и количестве выполненных им поездок.

3:24

Данные профиля водителя:  
 Луковников Иван

Номер телефона:  
 89041325016

Марка автомобиля:  
 Renault

Регистрационный номер автомобиля:  
 К449УХ31

Год выпуска автомобиля:  
 4/13/2021 12:00:00 AM

Дата последнего ТО автомобиля:  
 5/19/2023 12:00:00 AM

Количество принятых заказов	Количество завершенных заказов
15	9

Рисунок 40 – Страница с информацией о водителе

Если пользователь идентифицирован под ролью «водителя», пройдя авторизацию на соответствующей странице (рисунок 41), то откроется страница со всеми заказами клиентов соответствующего тарифа, где необходимо выбрать один из доступных заказов (рисунок 42). Также водитель может перейти к просмотру информации о себе, своем автомобиле и количестве завершенных им заказов.

После принятия заказа, водителю высвечивается информация о заказе, возможность отменить заказ или по приезду сообщить клиенту (рисунок 43).

При нажатии на отмену заказа, осуществляется переход на страницу с текущими заказами.

3:33

←

**TAXI**

**Авторизация водителя**

Введите ваш номер телефона

89041325016

Введите пароль

....

**ВОЙТИ**

Рисунок 41 – Страница авторизации водителя

3:33

Луковников Иван

**Доступные заказы**

Код заказа	Время заказа	Откуда	Куда	Доп. услуга
91	5/19/2023 2:32:20 AM	Sedova 8	Marata 2	Десткое кресло
92	5/19/2023 2:34:04 AM	Piskunova 22	Bezbovogo 18	Без Доп услуг
171	6/5/2023 3:14:49 AM	Лескова 6	Ленина 10	Десткое кресло
172	6/5/2023 3:19:07 AM	Жукова 7	Ямская 49	Десткое кресло

Выберите заказ, указав его код ниже

**ПРИНЯТЬ ЗАКАЗ**

Забрать из:

Доставить до:

ОПОВЕСТИТЬ О ТОМ ЧТО ВЫ ПОДЪЕХАЛИ

ЗАВЕРШИТЬ ЗАКАЗ

ОТМЕНИТЬ ЗАКАЗ

Рисунок 42 – Главная страница водителя

3:35

Луковников Иван

### Доступные заказы

Код заказа	Время заказа	Откуда	Куда	Доп. услуга
91	5/19/2023 2:32:20 AM	Sedova 8	Marata 2	Десткое кресло
92	5/19/2023 2:34:04 AM	Piskunova 22	Bezbovogo 18	Без Доп услуг
171	6/5/2023 3:14:49 AM	Лескова 6	Ленина 10	Десткое кресло
172	6/5/2023 3:19:07 AM	Жукова 7	Ямская 49	Десткое кресло

Выберите заказ, указав его код ниже

172

ПРИНЯТЬ ЗАКАЗ

Забрать из: Жукова 7

Доставить до: Ямская 49

ОПОВЕСТИТЬ О ТОМ ЧТО ВЫ ПОДЪЕХАЛИ

ЗАВЕРШИТЬ ЗАКАЗ

ОТМЕНИТЬ ЗАКАЗ

Рисунок 43 – Главная страница с принятым заказом

Если пользователь является администратором, то пройдя авторизацию на соответствующей странице (рисунок 44), откроется главная страница администраторов, на которой располагаются кнопки для перехода между страницами просмотра и модерирования данных (рисунок 45).

3:51

←

**TAXI**

**Авторизация администраторов**

Введите ваш номер телефона

Введите пароль

ВОЙТИ

Рисунок 44 – Страница авторизации администраторов





Рисунок 45 – Главная страница администраторов

При нажатии на кнопку «Добавить нового водителя» администратор перейдет на страницу с полями для заполнения данных о водителе (рисунок 46). Далее администратору нужно будет заполнить данные о его автомобиле, заполнив которые и нажав кнопку «Добавить водителя» в базу данных занесется информация о нем, после чего тот сможет авторизоваться в приложении как водитель (рисунок 47).

Рисунок 46 – Страница для заполнения данных о водителе

4:00

Заполните данные о его автомобиле

Марка автомобиля:

Регистрационный номер авто:

Номер кузова авто:

Номер двигателя авто:

Год выпуска:  
05/06/2023

Пробег авто:

Дата последнего технического обслуживания:  
05/06/2023

ДОБАВИТЬ ВОДИТЕЛЯ

Рисунок 47 – Страница добавление водителя

При нажатии на кнопку «Удаление водителей» администратор перейдет на страницу содержащую всех водителей и информацию о них, на этой странице он может выбрать водителя, после чего нажать на кнопку «Удалить водителя» для удаления его из базы данных (рисунок 48).

4:09

Список водителей

ФИО	Телефон	Паспорт	Лицензия	Авто	Тариф
Луковников Иван	89041325016	4194986407	9039362457	Renault	Эконом
Ясинский Дмитрий	89041325017	4847339832	1773230073	Honda	Стандарт
Широких Николай	89041325018	4671820352	6471174362	Fiat	VIP

Выберите водителя для редактирования

УДАЛИТЬ ВОДИТЕЛЯ

Рисунок 48 – Страница удаления водителей

Нажатие на кнопку «Модерация принятых заявок» откроется страница с возможностью просмотра всех активных на данный момент заказов, а также с возможностью внесения в них изменений (рисунок 49).

Активные заказы					
Время заказа	Откуда	Куда	Клиент	Водитель	Доп. услуга
6/5/2023 8:19:07 AM	Жукова 7	Ямская 49	Пороник Антон	Луковникова Иван	Детское кресло

Рисунок 49 – Страница модерации заказов

Кнопка «Просмотр информации» открывает страницу, позволяющую администратору просматривать всю информацию, переключаясь между соответствующими вкладками (рисунок 50).

5:14		
ДОП УСЛУГИ	ВЫЗОВЫ	АВТОМОБИЛИ
КЛИЕНТЫ	ВОДИТЕЛИ	ТАРИФЫ
Код клиента	ФИО	Номер телефона
10	Пороник Антон	89041325001
11	Винокуров Марк	89041325002
12	Дементьев Филипп	89041325003
17	Артем Афонин	89041325015

Рисунок 50 – Страница просмотра информации

## 6 Стоимость разработки и внедрения программного продукта

### 6.1 Организационно-экономическое обоснование проекта

Разрабатываемое мобильное приложение «Таксопарк» предназначено для заказа такси.

Средняя стоимость разработки приложения составляет (таблица 11).

Таблица 11 – стоимость аналогов мобильного приложения «Таксопарк»

Название сайта	Стоимость разработки аналога, руб.
Яндекс.Такси	От 4 798 800 руб.
Gett	От 3 999 000 руб.
Maxim	От 2 399 400 руб.

### 6.2 Расчет затрат на разработку программного продукта

Расчет полных затрат на разработку проектного решения ( $K_{РПР}$ ) осуществляется по формуле:

$$K_{РПР} = Z_{ОТР} + Z_{ЭВМ} + Z_{СПП} + Z_{ХОН} + E + A, \quad (6.1)$$

где  $Z_{ОТР}$  – сумма оплаты труда разработчика/разработчиков ПП;

$Z_{ЭВМ}$  – затраты, связанные с эксплуатацией техники;

$Z_{СПП}$  – затраты на специальные программные продукты, необходимые для разработки ПП;

$Z_{ХОН}$  – затраты на хозяйственно-операционные нужды (бумага, литература, носители информации и т.п.);

$E$  – затраты на электроэнергию, руб.;

$A$  – амортизация ПК, руб.

$$K_{РПР} = 38\,400 + 18\,240 + 0 + 0 + 218,11 + 1\,150 = 69\,528,11$$

Для подсчета фонда оплаты труда разработчика необходимо определить общее время разработки (таблица 12). Время, затрачиваемое на разработку проектного решения j-м разработчиком, определяется методом экспертных оценок или хронометража. Итоговое значение рассчитывается на основании приведенных исходных данных по формуле:

$$T_{РПРj} = \sum_{\beta=1}^n t_{\beta}, \quad (6.2)$$

где  $t_{\beta}$  – время  $\beta$ -го этапа разработки проектного решения, дн.

$$T_{РПРj} = 2 + 2 + 2 + 1 + 22 + 3 + 7 + 2 = 41$$

Таблица 12 - Затраты времени на создание программного продукта

Этап создания		Затраты времени (в днях)	Затраты времени (в часах)	Машинное время работы над ПП (в часах)
Разработка	Обследование объекта автоматизации	2	16	16
	Анализ и уточнение требований	2	16	16
	Разработка технического задания	2	16	16
	Проектирование структуры	1	8	8
	Программная реализация	22	176	176
	Тестирование программного продукта	3	24	24
	Отладка программного продукта	7	56	56
	Разработка описания	2	16	16
	ИТОГО	41	384	384

Рабочий день принимается равным 8 часам.

Сумму оплаты труда разработчика за время работы над программным продуктом рассчитаем исходя из часовой тарифной ставки и фонда фактического времени, затраченного на разработку программного продукта (по формуле 6.3):

$$Z_{отр} = C_{т1} \cdot \Phi_{вр}, \quad (6.3)$$

где  $C_{т1}$  - часовой тарифной ставки (принимается равной 100 руб./час);

$\Phi_{вр}$  – фонд фактического времени, затраченного на разработку программного продукта, час.

$$Z_{отр} = 100 \cdot 384 = 38\,400$$

Начисления на заработную плату рассчитываются в таблице 13.

Таблица 13 – Начисление на заработную плату

Начисление на заработную плату	Процент, %	Сумма, руб.
Пенсионный фонд (ПФ):		
- страховая часть	16	6 144
- накопительная часть	6	2 304
Фонд социального страхования (ФСС)	2,9	1 113,6
Федеральный фонд обязательного медицинского страхования (ФФОМС)	5,1	1 958,4
Итого	30	11 520

Затраты, связанные с использованием вычислительной и оргтехники (формула 6.4):

$$Z_{ЭВМ} = T_{МРПР} \cdot k_{Г} \cdot n \cdot C_{М-ч}, \quad (6.4)$$

где  $T_{МРПР}$  – машинное время работы над программным продуктом, час;

$k_{Г}$  – коэффициент готовности ЭВМ,  $k_{Г} = 0,95$ ;

$n$  – количество единиц техники, равно 1;

$C_{M-ч}$  – себестоимость машино-часа,  $C_{M-ч} = 50$  руб.

$$З_{ЭВМ} = 384 * 0,95 * 1 * 50 = 18\,240$$

Затраты на электроэнергию рассчитываются по следующей формуле 6.5:

$$E = W \times t \times T, \quad (6.5)$$

где  $W$  – мощность, потребляемая ПК, кВт/час;

$t$  – время работы ПК, час;

$T$  – тариф электроэнергии, руб.

$$E = 0,4 * 384 * 1,42 = 218,11$$

Тариф 1,42 рубля за киловатт.

Амортизация ПК рассчитывается по следующей формуле 6.6:

$$A = \frac{S \times q_{am}}{12}, \quad (6.6)$$

где  $S$  – первоначальная стоимость ПК, руб;

$q_{am}$  – процент амортизации в год.

$$A = \frac{46\,000 * 0,3}{12} = 1\,150$$

Для разработки программного продукта не произведено затрат на специальные программные продукты. Отсюда следует, что  $З_{СПП} = 0$ .

Поскольку для разработки также не требуются дополнительные вложений на хозяйственно-организационные нужды, то  $З_{ХОН} = 0$

Результаты выполненных расчетов сводятся в общей таблице (таблица 14).

					ДП.09.02.03.23.191.02.ПЗ	Лист
						47
Изм.	Лист	№ докум.	Подпись	Дата		

Таблица 14 – Затраты на разработку

Наименование затрат	Условное обозначение	Значение
Оплата труда разработчика программного продукта	З <sub>отр</sub>	49 920
Затраты, связанные с эксплуатацией техники	З <sub>эвм</sub>	18 240
Затраты на электроэнергию	Е	218,11
Затраты на амортизацию ПК	А	1 150
Затраты на хозяйственно-операционные нужды	З <sub>хон</sub>	0
Затраты на специальные программные продукты	З <sub>спп</sub>	0
Итого затрат на разработку	К <sub>рпп</sub>	69 528,11

### 6.3 Расчет затрат на внедрение программного продукта

Внедрение модуля в работу предприятия требует оплату хостинга в размере 100 рублей в месяц.

### 6.4 Основные выводы

Суть проекта заключается в разработке и внедрении мобильного приложения для вызова такси. Это позволит пользователям легко и удобно заказывать такси через мобильное устройство. Главная цель проекта - упростить и ускорить процесс заказа такси, предоставляя пользователям быстрый доступ к заказу такси и комфортную платформу для организации поездок.

Срок выполнения проекта составляет 41 день.

Итоговая стоимость затрат на разработку программного продукта, в которую включена оплата труда разработчика, затраты на эксплуатацию техники, в том числе амортизации ПК и затраты на электроэнергию, составила 56 472,11 рублей.



Ожидается, что проект будет окупаться в течение 6 месяцев. Это зависит от многих факторов, включая активность пользователей, тарифной политики и конкурентной среды. Однако, с учетом растущего спроса на услуги такси и возможности масштабирования приложения, ожидается достижение точки окупаемости в относительно короткие сроки.

					ДП.09.02.03.23.191.02.ПЗ	Лист
						49
Изм.	Лист	№ докум.	Подпись	Дата		

## ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта был проведен анализ существующих подобных систем, изучены требования и потребности пользователей, а также проведено проектирование и разработка приложения.

Целью работы было создание удобного и функционального инструмента, способного оптимизировать работу таксопарка, улучшить эффективность процессов управления и повысить удовлетворенность как клиентов, так и водителей.

Были реализованы следующие функциональные возможности: регистрация и авторизация пользователей, управление водителями и автомобилями, мониторинг текущего состояния автомобилей, учет заказов, а также аналитика данных.

Приложение создавалось с целью улучшить и упростить процесс заказа такси, предоставить пользователям удобный и надежный инструмент для вызова и оплаты услуг такси, а также повысить качество обслуживания и уровень безопасности в данной сфере

					ДП.09.02.03.23.191.02.ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 it-black.ru – Работа с базой данных в C#. – URL: [https://it-black.ru/rabota-s-bazoj-dannyh-v-ci\\_sharp/](https://it-black.ru/rabota-s-bazoj-dannyh-v-ci_sharp/) (дата обращения: 20.04.2023). – Текст: электронный.

2 metanit.com – Xamarin и кросс-платформенная разработка. – URL: <https://metanit.com/sharp/xamarin/1.1.php> (дата обращения: 14.03.2023). – Текст: электронный.

3 metanit.com – XAML. – URL: <https://metanit.com/sharp/xamarin/2.4.php> (дата обращения: 22.04.2023). – Текст: электронный.

4 metanit.com – Базы данных. – URL: <https://metanit.com/sharp/xamarin/7.2.php> (дата обращения: 25.04.2023). – Текст: электронный.

5 metanit.com – Графический интерфейс в Xamarin Forms. – URL: <https://metanit.com/sharp/xamarin/2.3.php> (дата обращения: 28.04.2023). – Текст: электронный.

6 metanit.com – Добавление форм. Взаимодействие между формами. – URL: <https://metanit.com/sharp/windowsforms/2.3.php> (дата обращения: 30.04.2023). – Текст: электронный.

7 metanit.com – Контейнер Grid – URL: <https://metanit.com/sharp/xamarin/3.6.php> (дата обращения: 22.04.2023). – Текст: электронный.

8 metanit.com – Руководство по программированию для Xamarin Forms. – URL: <https://metanit.com/sharp/xamarin/> (дата обращения: 30.04.2023). – Текст: электронный.

9 metanit.com – Текстовые поля. – URL: <https://metanit.com/sharp/xamarin/3.20.php> (дата обращения: 01.05.2023). – Текст: электронный.

10 microsoft.com – Xamarin.Forms Метки. – URL: <https://learn.microsoft.com/ru-ru/xamarin/xamarin-forms/user-interface/text/label> (дата обращения: 03.05.2023). – Текст: электронный.

					ДП.09.02.03.23.191.02.ПЗ	Лист
						51
Изм.	Лист	№ докум.	Подпись	Дата		

11 microsoft.com – Документация по Xamarin. – URL: <https://learn.microsoft.com/ru-ru/xamarin/> (дата обращения: 05.05.2023). – Текст: электронный.

12 microsoft.com – Настройка эмулятора Android. – URL: <https://learn.microsoft.com/ru-ru/xamarin/android/get-started/installation/android-emulator/> (дата обращения: 08.05.2023). – Текст: электронный.

13 xamarin.ru – Visual Studio Xamarin.Android для Windows. – URL: <https://xamarin.ru/knowledge-base/xamarin-android/xamarin-android-dlya-visual-studio-v-windows/> (дата обращения: 10.05.2023). – Текст: электронный.

14 xamarin.ru – Создание XAML и его структура. – URL: <https://xamarin.ru/knowledge-base/environment/sozdanie-xaml-i-ego-struktura/> (дата обращения: 12.05.2023). – Текст: электронный.

15 xamarin.ru – Элементы управления (UI) Xamarin. – URL: <https://xamarin.ru/knowledge-base/tools/ehlementy-upravleniya/> (дата обращения: 28.14.2023). – Текст: электронный.

## Приложение А – Авторизации

```

namespace TaxoparkMobile
{
    public partial class AvtorizKlientPage : ContentPage
    {
        List<string> userData = new List<string>();
        public AvtorizKlientPage()
        {
            InitializeComponent();
            image1.Source = ImageSource.FromResource("TaxoparkMobile.image.logo.png");
            imagebutton.Source = ImageSource.FromResource("TaxoparkMobile.image.reg.png");
            buttonVodAvtoriz.Source =
            ImageSource.FromResource("TaxoparkMobile.image.voditelLogo.png");
            buttonAdminAvtoriz.Source =
            ImageSource.FromResource("TaxoparkMobile.image.adminLogo.png");
        }
        private async void OpenReg_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new RegKlientPage());
        }
        private async void AdminButton_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new AvtorizAdmintPage());
        }
        private async void VoditelButton_Clicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new AvtorizVoditelPage());
        }
        private async void AvtorizButton_Clicked(object sender, EventArgs e)
        {
            string phoneUser = nameInput1.Text;
            string passwordUser = nameInput2.Text;
            try
            {
                DB db = new DB();
                db.openConnection();
                MySqlCommand command = new MySqlCommand("SELECT * FROM `client`
                WHERE `Phone_Client`=@phoneUser AND `Password_Client`=@passwordUser",
                db.getConnection());
                command.Parameters.Add("@phoneUser", MySqlDbType.VarChar).Value =
                phoneUser;
            }
            catch { }
        }
    }
}

```

```

command.Parameters.Add("@passwordUser", MySqlDbType.VarChar).Value =
GetHashMD5(passwordUser);
MySqlDataReader reader = command.ExecuteReader();
userData.Clear();
if (reader.HasRows)
{
while (reader.Read())
{
userData.Add(reader[0].ToString());
userData.Add(reader[1].ToString());
userData.Add(reader[2].ToString());
}
await Navigation.PushAsync(new ZakazKlientPage(userData));
}
else
{
await DisplayAlert("Ошибка", "Не верные данные", "OK");
}
db.closeConnection();
}
catch
{
await DisplayAlert("Ошибка", "Заполните все поля", "OK");
}
}
public string GetHashMD5(string input)
{
var md5 = MD5.Create();
var hash = md5.ComputeHash(Encoding.UTF8.GetBytes(input));
return Convert.ToBase64String(hash);
}
}
}

```

## Приложение Б – Просмотра заказов

```
private void UpdateCallView()
{
    updateButton.IsEnabled = true;
    Otkuda.Text = "";
    Kuda.Text = "";
    picker.IsEnabled = true;
    Accepted.IsEnabled = true;
    Alerts.IsEnabled = false;
    Finished.IsEnabled = false;
    CancelButton.IsEnabled = false;
    callView.Children.Clear();
    picker.Items.Clear();
    int j = 0;
    DB db = new DB();
    db.openConnection();
    MySqlCommand command = new MySqlCommand("SELECT
`Id_Call`,`DataTime_Call`,`Otkuda`,`Kuda`,`Add_Services_Id_Services` " +
"FROM `call` WHERE `Accepted` IS NULL and `Tarif_Id_Tarif`=@tarif",
db.getConnection());
    command.Parameters.Add("@tarif", MySqlDbType.Int32).Value =
    Convert.ToInt32(voditelData[9]);
    MySqlDataReader reader = command.ExecuteReader();
    if (reader.HasRows)
    {
        while (reader.Read())
        {
            picker.Items.Add(reader[0].ToString());
            string addservices;
            switch (reader[4].ToString())
            {
                case ("1"):
                    addservices = "Десткое кресло";
                    break;
                case ("2"):
                    addservices = "Зоотакси";
                    break;
                case ("3"):
                    addservices = "Пустой багажник";
                    break;
                default:
```

```

addservices = "Без Доп услуг";
break;
}
Grid grid = new Grid
{
    RowSpacing = -30,
    ColumnSpacing = 1,
    RowDefinitions =
    {
        new RowDefinition { Height = new GridLength(1, GridUnitType.Star) },
    },
    ColumnDefinitions =
    {
        new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
        new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
        new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
        new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) },
        new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) }
    }
};
for (int i = 0; i < 5; i++)
{
    string textcall = reader[i].ToString();
    if (i == 4)
    {
        textcall = addservices;
    }
    Label label1 = new Label
    {
        FontSize = 12,
        Text = textcall
    };
    label1.VerticalTextAlignment = TextAlignment.Center;
    label1.HorizontalTextAlignment = TextAlignment.Center;
    label1.TextColor = Color.Black;
    grid.Children.Add(label1, i, j);
}
j++;
BoxView box = new BoxView
{
    Margin = -5,
    HeightRequest = 2
};

```



```
box.BackgroundColor = Color.DarkOrange;  
callView.Children.Add(grid);  
callView.Children.Add(box);  
}  
}
```

					ДП.09.02.03.23.191.02.ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		