

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ  
ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий  
Кафедра Информационных систем и технологий  
Специальность 1-98 01 03 Программное обеспечение информационной безопасности  
мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к дипломному проекту:**

Веб-приложение «Служба доставки малогабаритных товаров»

Дипломник \_\_\_\_\_ Руденя П. А.

Руководитель проекта \_\_\_\_\_ Берников В. О., асс.

Заведующий кафедрой \_\_\_\_\_ Смелов В. В., к.т.н., доцент

Консультант \_\_\_\_\_ Соболевский А. С., ст. преп.

Нормоконтролер \_\_\_\_\_ Нистюк О. А., асс.

Дипломный проект защищен с оценкой \_\_\_\_\_

Председатель ГЭК \_\_\_\_\_ Дюбков В. К., к.т.н., доцент

Минск 2023



## Реферат

Пояснительная записка дипломного проекта содержит 77 страниц пояснительной записки, среди которых 11 таблиц, 14 формул, 39 иллюстраций, 17 источников литературы, а также 7 приложений.

КРОСПЛАТФОРМЕННОЕ ПРИЛОЖЕНИЕ, ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON, СИСТЕМА КОНТРОЛЯ ВЕРСИЙ GIT, ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ PYCHARM, СИСТЕМА УПРАВЛЕНИЯ РЕЛЯЦИОННЫМИ БАЗАМИ ДАННЫХ SQLITE

Пояснительная записка состоит из введения, семи разделов, заключения и списка используемой литературы.

В введении сформулирована цель и задачи проекта.

В первом разделе проводится постановка задачи, обзор аналогичных решений.

Во втором разделе описаны основные шаги проектирования веб-платформы.

В третьем разделе описаны детали и примеры реализации серверной и клиентской части разрабатываемого веб-приложения.

В четвертом разделе описаны основные шаги тестирования проекта.

В пятом разделе описано руководство пользователя, позволяющее понять принцип взаимодействия с интерфейсом клиентского приложения.

В шестом разделе приводится расчет экономических параметров и себестоимости программного средства, разработанного в рамках дипломного проекта.

В седьмом разделе описан анализ информационной безопасности приложения.

В заключении представлены итоги дипломного проектирования и задачи, которые были решены в ходе разработки проекта.

Объем графической части дипломного проекта составляет 1.125 листа A1.

					БГТУ 00.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Руденя П.А.			Реферат		Лист	Листов
Провер.		Берников В.О.					1	1
						74217088, 2023		
Н. контр.		Нистюк О.А.						
Утв.		Смелов В.В.						

## Abstract

The explanatory note of the diploma project contains 77 pages of explanatory note, 11 tables, 14 formulas, 39 illustrations, 17 sources used, 7 appendices.

CROSS-PLATFORM APPLICATION, PYTHON PROGRAMMING LANGUAGE, GIT VERSION CONTROL SYSTEM, PYCHARM INTEGRATED DEVELOPMENT ENVIRONMENT, SQLITE RELATIVE DATABASES MANAGEMENT SYSTEM

The explanatory note consists of an introduction, seven sections, a conclusion, and a list of references.

The introduction sets out the goal and objectives of the project.

In the first section, you will find a problem statement, a review of similar solutions, and a patent search.

The second section describes the main steps for designing a web platform.

The third section describes the details and examples of the implementation of the server and client parts of the web application being developed.

The fourth section describes the basic steps of project testing.

The fifth section describes the user's guide for understanding how to interact with the client application interface.

The sixth section provides a calculation of the economic parameters and cost of the software developed within the framework of the diploma project.

The seventh section describes the analysis of the information security of the app.

In conclusion, the results of the diploma design and the tasks that were solved during the development of the project are presented.

The volume of the graphic part of the diploma project is 1.125 sheets A1.

					БГТУ 00.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Руденя П.А.			Abstract		Лист	Листов
Провер.		Берников В.О.					1	1
						74217088, 2023		
Н. контр.		Нистюк О.А.						
УТВ.		Смелов В.В.						

## Содержание

Введение .....	7
1 Постановка задачи и обзор аналогичных решений.....	8
1.1 Постановка задачи.....	8
1.2 Аналогичные решения.....	8
1.2.1 Приложение Яндекс Доставка.....	8
1.2.2 Приложение CDEK .....	9
1.2.3 Приложение Delivery Club .....	10
1.3 Обзор средств разработки.....	12
1.3.1 Выбор языка программирования.....	12
1.3.2 Выбор средства программирования .....	13
1.3.3 Выбор платформы разработки.....	14
1.4 Выводы по разделу.....	14
2 Проектирование программного средства.....	15
2.1 Архитектура приложения.....	15
2.2. Диаграмма вариантов использования .....	15
2.3 Проектирование серверной части приложения .....	16
2.4 Проектирование клиентской части приложения .....	17
2.5 Структура базы данных клиентской части приложения.....	18
2.6 Выбор дизайнерского решения.....	22
2.7 Выводы по разделу.....	23
3 Реализация программного средства.....	24
3.1 Реализация серверной части приложения .....	24
3.2 Файловая структура .....	25
3.3 Интеграция Google Map API .....	26
3.4 Интеграция Stripe API.....	28
3.5 Интеграция PayPal API .....	28
3.6 Отображение текущего местоположения курьера.....	30
3.7 Реализация WebSocket.....	31
3.8 Реализация PWA.....	32
3.9 Разработка клиентской части приложения.....	33
3.10 Взаимодействие с базой данной .....	34
3.11 Реализация авторизации и регистрации.....	35
3.12 Выводы по разделу.....	36
4 Тестирование.....	37
4.1 Ручное тестирование.....	38
4.2 Тестирование web-сервера .....	39

					БГТУ 00.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Руденя П.А.			Содержание			
Провер.		Берников В.О.						
Н. контр.		Нистюк О.А.						
Утв.		Смелов В.В.						
							Лист	Листов
							1	2
						74217088, 2023		

	6
4.2.1 Доступ к защищенным ресурсам.....	39
4.2.2 Некорректность данных .....	40
4.3 Модульное тестирование.....	41
4.4 Выводы по разделу.....	42
5 Руководство пользователя .....	43
5.1 Роль «Администратор» .....	43
5.2 Роль «Пользователь» .....	46
5.3 Роль «Курьер».....	53
5.4 Выводы по разделу.....	58
6 Техничко-экономическое обоснование проекта.....	59
6.1 Общая характеристика разрабатываемого продукта .....	59
6.2 Исходные данные для проведения расчетов .....	60
6.3 Методика обоснования цены .....	62
6.3.1 Объем программного средства.....	63
6.3.2 Основная заработная плата.....	64
6.3.3 Дополнительная заработная плата.....	65
6.3.4 Отчисления в Фонд социальной защиты населения .....	65
6.3.5 Расходы на материалы.....	66
6.3.6 Расходы на оплату машинного времени.....	66
6.3.7 Прочие прямые затраты .....	66
6.3.8 Накладные расходы .....	67
6.3.9 Сумма расходов на разработку программного средства .....	67
6.3.10 Расходы на сопровождение и адаптацию.....	67
6.3.11 Полная себестоимость.....	68
6.3.12 Определение цены, оценка эффективности .....	68
6.4 Выводы по разделу.....	69
7 Анализ информационной безопасности проекта.....	70
7.1 Обзор безопасности базы данных SQLite.....	70
7.2 Подтверждение номера телефона.....	71
7.3 Безопасность фреймворка Django .....	72
7.3.1 Защита от CSRF в Django.....	72
7.3.2 Защита от XSS-атак в Django.....	73
7.3.3 Защита от Clickjacking в Django .....	73
7.3.4 Аутентификация и авторизация – Django .....	73
7.4 Безопасность использование платежной системы Stripe .....	74
7.5 Выводы по разделу.....	75
Заключение .....	76
Список использованных источников.....	77
ПРИЛОЖЕНИЕ А .....	78
ПРИЛОЖЕНИЕ Б .....	79
ПРИЛОЖЕНИЕ В .....	80
ПРИЛОЖЕНИЕ Г .....	81
ПРИЛОЖЕНИЕ Д.....	82
ПРИЛОЖЕНИЕ Е .....	83
ПРИЛОЖЕНИЕ Ж .....	84

## Введение

Доставка товаров – несомненно, удобный сервис, которым пользуются практически все категории граждан. На дом сейчас можно заказать любые виды товаров, но среди нынешнего многообразия служб доставки легко запутаться. Доставка товаров и продуктов на дом облегчает жизнь, особенно, если это жизнь в условиях самоизоляции или удалённой работы из дома. Для тех, у кого нет времени или просто желания ходить среди магазинных полок, предоставлена возможность заказать товары, не выходя из дома или работы. Клиенты могут оплатить покупки онлайн прямо на сайте или при получении наличными или банковской картой на выбор. Сервис доставки на дом ничем не уступает личному посещению магазина, а только обладает массой достоинств. Одно из самых главных, конечно, это экономия времени, которое можно посвятить более важным делами.

Курьерская доставка – это способ доставки товаров, документов и других грузов, когда между отправителем и получателем используется курьер. Курьер может забрать груз у отправителя и доставить его непосредственно до адресата. Курьерская доставка может быть международной или национальной, может осуществляться с использованием различных видов транспорта: автомобиль, мотоцикл или даже велосипед. Важная особенность курьерской доставки – это возможность контролировать местоположение груза в режиме онлайн.

Для отправки груза через курьерскую службу необходимо оформить заказ с указанием данных отправителя и получателя, а также характеристик груза и его веса. Затем курьер приедет и заберет груз у отправителя, после чего доставит его получателю. Курьерская доставка отличается высокой степенью надежности, быстротой и удобством. Она особенно востребована в эпоху интернет-магазинов, когда покупатель может заказать товар с доставкой прямо на дом или в офис.

На рынке приложений представлено огромное количество различных служб доставки, однако большая часть из них имеет ряд схожих недостатков, таких как:

- невозможность использования в Беларуси;
- кража личной информации с целью персонализации рекламы;
- высокая стоимость.

Таким образом, целью дипломного проекта является веб-приложение позволяющие в полной мере реализовать доставку товаров.

Реализовывать поставленную задачу я буду при помощи следующих технологий: Python 3.7 [1], HTML/CSS, JavaScript [2], фреймворк Django [3], шаблонизатор Jinja. В качестве базы данных использована компактная встраиваемая Система управления базами данных SQLite [4].

					БГТУ 00.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Руденя П.А.			Введение		Лист	Листов
Провер.		Берников В.О.					1	1
Н. контр.		Нистюк О.А.						
Утв.		Смелов В.В.					74217088, 2023	

# 1 Постановка задачи и обзор аналогичных решений

## 1.1 Постановка задачи

В современном мире многие привычные вещи как поход в магазин, работа, обучение и многие другие вещи переходят в цифровой формат. Людям не нужно гладить брюки на работу или носить тяжёлый портфель в школу.

Доставка товаров приносит значительную выгоду пользователям и пользуются большим спросом, разработанное приложение для доставки малогабаритных товаров должно быть конкурентоспособным, а значит соответствовать всем или большинству требованиям к такому приложению.

Главной задачей является разработка WEB-приложения, позволяющего пользователям создавать заказы, изменять, удалять, а также производить оплату за доставку. Так как все данные должны принадлежать какому-либо конкретному пользователю, то для входа в приложение регистрация и авторизация обязательны. Таким образом, целью дипломного проекта является веб-приложение позволяющие в полной мере реализовать доставку товаров. Для достижения поставленной цели сформулированы следующие задачи:

- спроектировать архитектуру и структуру базы данных;
- реализация трех ролей: курьер, пользователь и администратор;
- авторизация, регистрация, заполнение личной информации пользователя;
- создание заказов, ввод информации о них, указание точек доставки;
- просмотр свободных заказов на карте для курьера;
- отображение местоположения курьера на карте;
- получение фотографий от курьера в момент приема и доставки товара.

## 1.2 Аналогичные решения

### 1.2.1 Приложение Яндекс Доставка

Яндекс Доставка – это универсальное приложение с разнообразным функционалом, есть версии для iOS, Android. «Яндекс Доставка» – сервис доставки, который организует отправку посылок физическим и юридическим лицам.

Сервис предлагает услуги от экспресс-доставки по городу до междугородной доставки и магистральных перевозок большегрузным автотранспортом. Доставку осуществляют партнёрские курьерские службы и самозанятые курьеры – партнёры сервиса. Вызвать курьера можно как в личном кабинете на официальном сайте компании «Яндекс Доставки», так и при помощи мобильного приложения «Яндекс.Go», в котором отправитель и получатель могут отслеживать статус заказа. Также юридические лица могут встроить доставку в свою CRM- и CMS-систему и заказывать доставки оттуда.

					БГТУ 01.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Руденя П.А.			1 Постановка задачи и обзор аналогичных решений		Лист	Листов
Провер.		Берников В.О.					1	7
Н. контр.		Нистюк О.А.						
Утв.		Смелов В.В.				74217088, 2023		



Пользователи могут заказать доставку до двери или до пункта выдачи заказа. В приложении «Яндекс Go» можно выбрать доставку по городу или между городами. Чтобы отправить посылку, нужно перейти в один из тарифов «Доставки», заполнить данные и дожидаться курьера. В случае с межгородом нужно заполнить данные в приложении, а после самостоятельно дойти до ПВЗ и передать посылку в доставку. Корпоративные клиенты подключаются к сервису и управляют доставками через личный кабинет на сайте «Яндекс Доставки». Курьеры забирают заказы со склада или из магазина и доставляют до пункта выдачи или до двери получателя. Сервис подключает для доставки логистических партнёров и управляет распределением заказов корпоративных клиентов и пользователей приложения «Яндекс Go». Чтобы получать заказы, курьеры используют приложение «Яндекс Про» У пользователей будет два варианта использования приложения: веб-версия и мобильная версия.

Скриншоты приложения представлены на рисунке 1.1.

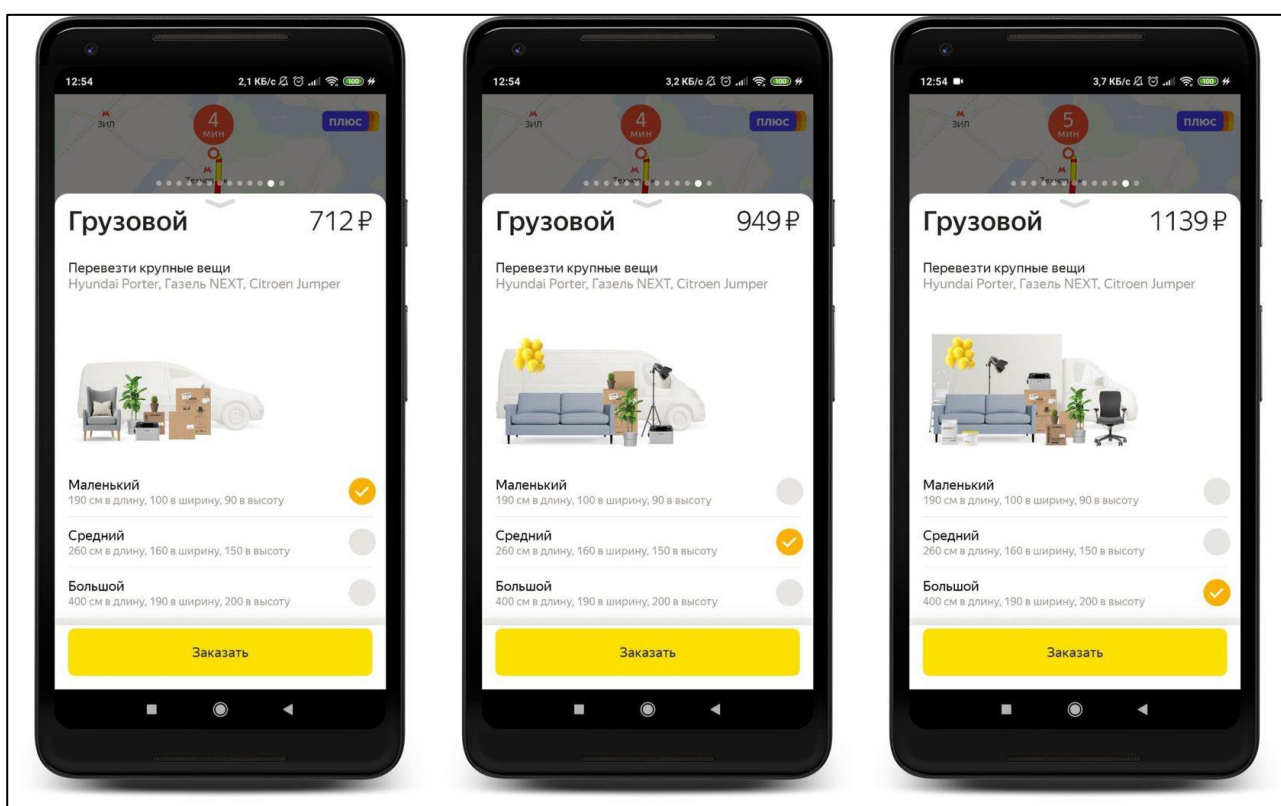


Рисунок 1.1 – Скриншоты приложения Яндекс Доставка

Недостатки приложения:

- высокая стоимость;
- возможность использования только в странах СНГ;
- присутствует реклама;
- приложение не имеет веб-версии для курьера.

### 1.2.2 Приложение CDEK

CDEK – это крупнейший логистический оператор, занимающийся доставкой грузов и посылок по всей стране и за ее пределами. Компания была основана в 2000 году, а в настоящее время она работает со свыше 200 странами мира.

CDEK предоставляет своим клиентам несколько способов доставки: курьерскую службу, самовывоз из пунктов выдачи и пункты выдачи в магазинах. Кроме того, компания предлагает услуги складирования и перевозки грузов.

Один из основных принципов работы CDEK – максимально удобные условия для клиентов. Она предоставляет своим заказчикам широкий спектр услуг, включая онлайн-отслеживание выполнения доставки, особые условия для интернет-магазинов и возможность быстрой доставки.

Эффективность деятельности CDEK доказывается огромной клиентской базой – компания обслуживает более 300 000 клиентов ежегодно.

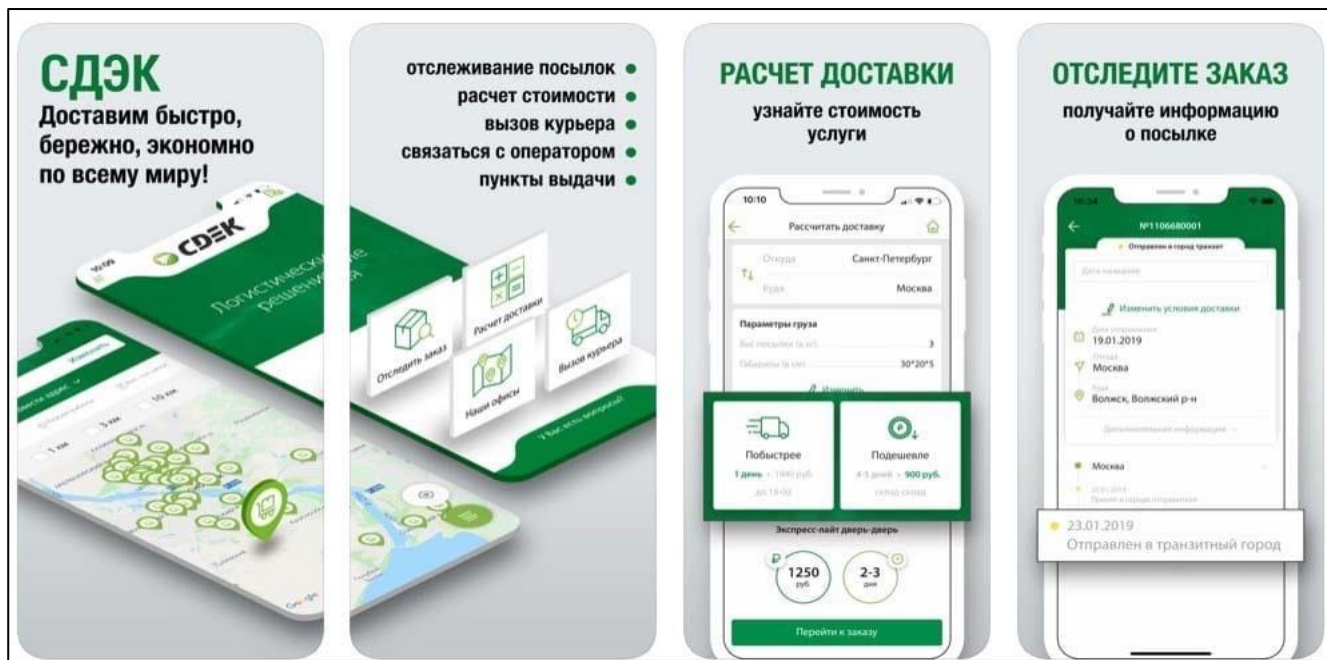


Рисунок 1.2 – Скриншоты приложения CDEK

Недостатки приложения:

- необходимо предоставить паспортные данные;
- перегруженный интерфейс;
- присутствует реклама.

### 1.2.3 Приложение Delivery Club

Delivery Club – это сервис онлайн-заказа еды и доставки на дом из ресторанов и кафе, работающий в более чем 300 городах России, Беларуси и Казахстана. Пользователи могут выбирать блюда из множества заведений, оформлять заказ и оплачивать его онлайн через сайт или мобильное приложение Delivery Club.

Заказы доставляются курьерами в указанное время и место. Сервис предоставляет возможность выбора из различных кухонь, предоставляет информацию о меню, ценах, отзывах и рейтинге заведений. Регистрация на сервисе бесплатна для пользователей. Сервис имеет большой ассортимент блюд, регулярно обновляющийся список заведений и кухонь.

Кроме того, система оформления заказа и оплаты удобная и безопасная, а быстрая и своевременная доставка блюд позволяет получить желаемое блюдо в удобное

время и место без лишних хлопот. Пользователи могут часто использовать акции, скидки и бонусную систему, что делает сервис лояльным как для одноразовых, так и для регулярных пользователей.

Благодаря сотрудничеству с множеством популярных ресторанов и кафе, сервис обеспечивает своих клиентов качественными и разнообразными блюдами, удовлетворяя самые взыскательные вкусы и предпочтения. Каждому заказу прилагаются подробные инструкции по его получению, а также информация о возможности отмены заказа в случае невозможности его выполнения. Сервис также предоставляет услуги по организации кейтеринга на мероприятиях любого масштаба или сложности.

Кроме того, возможность оставить отзывы о заведениях и блюдах помогает другим пользователям сделать правильный выбор. Скриншоты данного приложения представлены на рисунке 1.3.

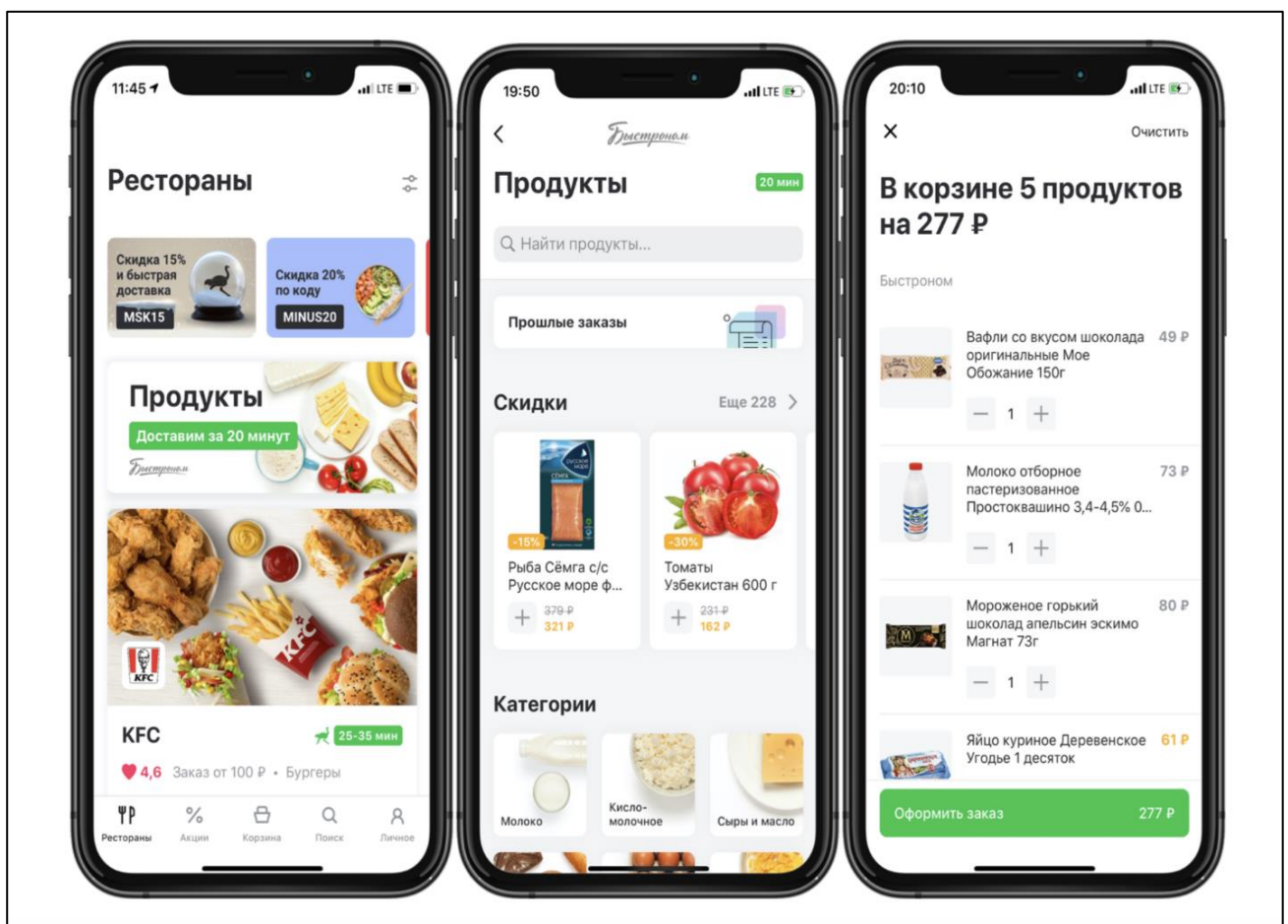


Рисунок 1.3 – Скриншоты приложения Delivery Club

К недостаткам приложения можно отнести:

- высокие цены на еду и доставку в некоторых ресторанах;
- заказы могут быть задержаны или перепутаны из-за высокой загруженности курьеров в пиковые часы;
- иногда приложение не обновляет данные о наличии блюд и часто отображает неправильную информацию о работе ресторанов;
- приложение может потреблять много ресурсов, а также занимать много места на устройстве пользователя.

### 1.3 Обзор средств разработки

В данном разделе будут рассмотрены различные библиотеки, технологии и подходы, которые будут использованы в разработке приложения.

#### 1.3.1 Выбор языка программирования

В качестве языка программирования серверной части был выбран Python и фреймворк для разработки веб-приложений Django.

На сегодняшний момент язык программирования Python один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей по всему миру.

Python – это объектно-ориентированный, интерпретируемый язык программирования с динамической типизацией. Он был разработан в конце 1980-х годов Гвидо ван Россумом в Нидерландах и с тех пор стал очень популярным языком программирования во всем мире.

Python имеет простой и понятный синтаксис, который делает его легко читаемым и понятным для начинающих разработчиков. Он также является мощным языком программирования с множеством инструментов и библиотек для работы с данными, создания веб-приложений и многое другое.

Python может использоваться для различных целей, таких как научные расчеты, создание программного обеспечения, разработка веб-приложений, обработка данных, автоматизация задач, создание игр и т.д. Он также широко используется в компаниях и университетах по всему миру.

Некоторые из достоинств Python:

- легковесный и эффективный;
- прост в изучении и использовании;
- обладает богатым экосистемой и множеством библиотек;
- широко применим в различных областях, включая науку, бизнес и разработку программного обеспечения.

Также для работы с Google Map API, Stripe and PayPal API будет использоваться язык программирования JavaScript.

JavaScript – это язык программирования, который используется для создания динамических интерактивных веб-сайтов, взаимодействия с клиентами и многими другими веб-приложениями, API.

Этот язык разработан на основе стандарта ECMAScript и обеспечивает возможность создания интерактивных пользовательских интерфейсов, управления поведением динамических веб-страниц и создания веб-приложений.

JavaScript можно использовать для различных веб-задач, таких как:

- динамическое создание и изменение HTML-элементов и стилей;
- обработка событий и управление сценариями на веб-страницах;
- взаимодействие с серверными приложениями через AJAX.

JavaScript стал одним из самых популярных языков программирования благодаря своей широте применения и легкому изучению. Он используется не только на

веб-страницах, но также в других средах, таких как серверное программное обеспечение и мобильные приложения.

Для хранения данных будет использоваться СУБД SQLite.

SQLite – это легковесная и самодостаточная встраиваемая база данных, которую можно использовать в приложениях на множестве платформ, включая мобильные устройства, веб-браузеры, настольные операционные системы и другие.

SQLite является сервером базы данных, который не требует отдельно установленного сервера баз данных и управляется из приложения. Он использует локальный файл для хранения данных, что делает его легким в использовании и хранении.

SQLite отлично подходит для небольших проектов, где не требуется высокая производительность и большой объем данных. Однако, это не означает, что SQLite не может работать с большими объемами данных. Он имеет возможность обрабатывать множество одновременных подключений к базе данных, что позволяет использовать его для некоторых типов приложений, таких как мобильные приложения или веб-приложения с небольшим трафиком.

Кроме того, SQLite легко интегрируется в многие языки программирования, включая Java, Python, Ruby, C# и многие другие, что делает его очень популярным среди разработчиков. Он также имеет множество инструментов для работы с базами данных, таких как командная строка, браузер базы данных SQLite и другие.

Общая высокая производительность, компактность и легковесность SQLite делают его излюбленным выбором для приложений, которые не требуют огромных объемов данных и не обязательно зависят от сервера базы данных.

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

### **1.3.2 Выбор средства программирования**

В качестве инструментария разработки использовался PyCharm. PyCharm – это интегрированная среда разработки (IDE) для языка программирования Python, созданная компанией JetBrains. Это мощное и удобное средство разработки, которое предоставляет множество инструментов для улучшения производительности и эффективности работы Python-разработчика.

PyCharm имеет множество функций и возможностей, включая автодополнение, контекстную помощь, отладчик, систему контроля версий, интеграцию с Django и другими фреймворками, удобный редактор кода, инструменты для управления зависимостями и многое другое.

Кроме того, PyCharm имеет удобный интерфейс и прост в использовании. Он также обеспечивает высокую продуктивность благодаря использованию быстрого веб-движка, правильной подсветке синтаксиса, автоматической генерации кода и быстрому рефакторингу. Это позволяет разработчикам быстро и удобно создавать, и поддерживать свои проекты на Python.

PyCharm также поддерживает многоплатформенность и может использоваться на таких платформах как Windows, Linux.

Базовая версия PyCharm доступна бесплатно, но есть и профессиональная версия со множеством дополнительных функций за платную подписку.

### 1.3.3 Выбор платформы разработки

Django – это высокоуровневый веб-фреймворк на языке программирования Python, который используется для быстрого и эффективного создания веб-приложений. Он основан на принципах модульного, переиспользуемого и масштабируемого кода и включает в себя множество инструментов для работы с базами данных, управления учетными записями пользователей и многое другое.

Django также обладает множеством расширений и пакетов, которые облегчают разработку веб-приложений на данной платформе. Например, есть пакеты для создания адаптивных интерфейсов, работы с веб-сокетами, создания REST API, и многих других полезных функций.

Кроме того, Django поддерживает различные базы данных, включая PostgreSQL, MySQL, Oracle, и SQLite, что позволяет разработчикам выбрать наиболее подходящее решение под их проект.

Для удобства разработки, Django включает в себя встроенный сервер разработки, который облегчает процесс тестирования и отладки приложений. Он также поддерживает использование шаблонов для создания пользовательского интерфейса.

Django имеет модульную архитектуру, которая позволяет разработчикам легко управлять всеми аспектами создания веб-приложения, включая взаимодействие с веб-сервером, работу с базами данных и управление HTTP-запросами и ответами.

Он также включает в себя мощный ORM, который позволяет разработчикам работать с базами данных, используя объектно-ориентированный подход.

Кроме того, Django имеет встроенную систему аутентификации и авторизации пользователей, так что разработчики могут быстро и легко добавлять функциональность учетных записей пользователей к своим приложениям.

В целом, Django является мощным инструментом для создания веб-приложений и благодаря своей гибкой архитектуре и большому сообществу разработчиков, она может быть использована для различных проектов, от простых блогов до сложных корпоративных приложений.

### 1.4 Выводы по разделу

В данном разделе были рассмотрены различные службы доставки, определены их сильные и слабые стороны. Было изучено какие средства можно использовать лучше и эффективнее, а какие нет. Каждый аналог был подробно описан, и было показано, для чего предназначено то или иное программное обеспечение.

Необходимо реализовать серверную и клиентскую часть приложения.

Клиентская часть приложения должна: позволять подключаться к серверу, обладать удобным интерфейсом, предоставлять актуальную информацию.

Серверная часть приложения должна: обеспечивать хранение данных, основываться на REST-архитектуре, обеспечивать легкую поддержку.

В качестве аналогов разрабатываемого веб-приложения были рассмотрены следующие приложения: «Яндекс Доставка», «CDEK», и «Delivery Club».

Для разработки серверной части был выбран язык программирования Python и среда разработки PyCharm. Для хранения данных будет использоваться встраиваемая система управления базами данных SQLite.



## 2 Проектирование программного средства

### 2.1 Архитектура приложения

Архитектура программного средства – это его строение как оно видно (или должно быть видно) извне его, т.е. представление программного средства как системы, состоящей из некоторой совокупности взаимодействующих подсистем. В качестве таких подсистем выступают обычно отдельные программы.

Разработка архитектуры является первым этапом борьбы со сложностью программного средства, на котором реализуется принцип выделения относительно независимых компонент [5].

Основные задачи разработки архитектуры программного средства (ПС):

- выделение программных подсистем и отображение на них внешних функций (заданных во внешнем описании) ПС;
- определение способов взаимодействия между специально выделенными программными подсистемами.

С учетом принимаемых на этом этапе решений производится дальнейшая конкретизация и функциональных спецификаций.

Архитектура разрабатываемого программного средства представлена на рисунке 2.1, а также в приложении А.

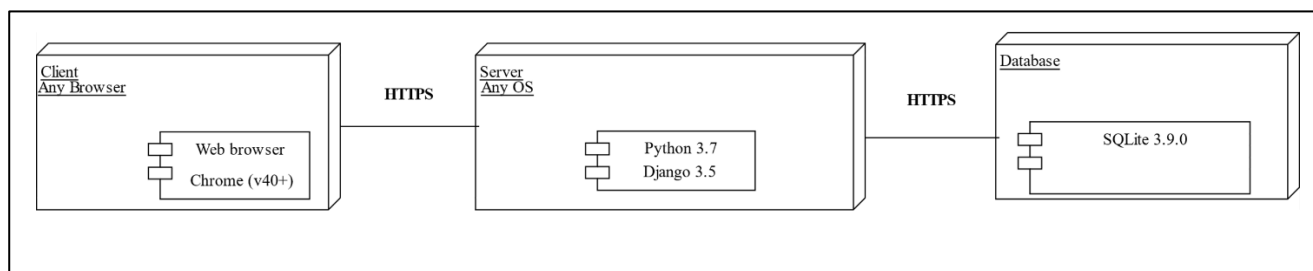


Рисунок 2.1 – Диаграмма архитектуры приложения

Само приложение будет работать по принципу клиент-серверного взаимодействия. Серверная часть приложения будет работать на фреймворке Django Core, преимущество которого заключается в своей кроссплатформенности.

### 2.2. Диаграмма вариантов использования

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования, а также на стадии разработки.

					БГТУ 02.00.ПЗ						
		ФИО	Подпись	Дата							
Разраб.		Руденя П.А.			2 Проектирование программного средства				Лист	Листов	
Провер.		Берников В.О.								1	9
Н. контр.		Нистюк О.А.									
Утв.		Смелов В.В.									
							74217088, 2023				

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования.

При этом актером называется любой объект, субъект или система, взаимодействующая с моделируемой системой с другой удалённой точки.

В свою очередь вариант использования – это спецификация сервисов (функций), которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

Диаграммы рисуют для визуализации. Основная цель диаграмм – визуализация разрабатываемой системы с разных точек зрения.

Для данного дипломного проекта была разработана диаграмма вариантов использования для пользователей с разными ролями. В качестве ролей выбрана роль администратора, пользователя и курьера.

Диаграмма вариантов использования представлена в приложении Б.

В ходе разработки программного средства было реализовано три типа пользователей: администратор, пользователь и курьер.

Функционально программное средство для *пользователя* должно предоставлять следующие возможности:

- создание нового заказа;
- отслеживание состояния своего заказа;
- привязка банковской карты для оплаты заказа;
- возможность отмены заказа, до момента принятия его курьером.

Функционально программное средство для *курьера* должно предоставлять следующие возможности:

- просматривать все возможные заказы на карте;
- привязка банковской карты для получения вознаграждения за доставку;
- возможность взять любой заказ;
- добавление фото в момент получения и передачи заказа;
- просмотр выполненных заказов;
- вывод заработанных средств на банковскую карту.

Функционально программное средство для *администратора* должно предоставлять следующие возможности:

- отменить заказ;
- просмотр истории заказа;
- создание категорий товаров;
- администрирование пользователей.

## 2.3 Проектирование серверной части приложения

В качестве архитектуры сервера была использована классическая трехуровневая архитектура, являющаяся наиболее используемой в производстве.

Данная архитектура подразумевает полное разделение выбранного веб-приложения на такие уровни как:

- уровень доступа к данным;



- уровень бизнес-логики;
- уровень представления.

В рамках такой архитектуры пользователи выполняют запросы через уровень представления, который взаимодействует только с уровнем бизнес-логики.

Уровень бизнес-логики может вызывать уровень доступа к данным для обработки запросов. Также данный уровень отвечает за построение всей логики приложения, за ликвидацию связей уровня представления и уровня данных за счёт использования паттерна проектирования DTO.

DTO – один из шаблонов проектирования, используется для передачи данных между подсистемами приложения.

Data Transfer Object, в отличие от business object или data access object не должен содержать какого-либо поведения.

Уровень представления не должен выполнять запросы напрямую к уровню доступа к данным и какими-либо другими способами напрямую взаимодействовать с функциями приложения.

Аналогичным образом, уровень бизнес-логики должен взаимодействовать с функциями только через уровень доступа к данным.

Таким образом, для каждого уровня четко определена своя обязанность.

## 2.4 Проектирование клиентской части приложения

Для создания клиентской части использовался фреймворк Django, дополнительно использовались шаблонизаторы и статические файлы.

Шаблонизаторы Django позволяют встраивать Python-код в HTML-шаблоны, чтобы создать динамические страницы.

Статические файлы (CSS, JavaScript, изображения) можно хранить в отдельной директории и подключать их в HTML-шаблонах.

Шаблонизатор Jinja – это инструмент для упрощения создания HTML-страниц и других типов файлов, используемых в веб-приложениях.

Он позволяет разделять логику и представление, что в свою очередь облегчает разработку веб-приложений и повышает их совершенство. Вот несколько способов, которыми Jinja упрощает работу разработчику:

- синтаксис шаблонов: Jinja позволяет использовать простой и интуитивный синтаксис в HTML-шаблонах, благодаря чему упрощается создание пользовательских интерфейсов и уменьшается объем кода.
- Оптимизация производительности: Jinja предоставляет функционал для кэширования шаблонов и компиляции их в байт-код, что позволяет снизить нагрузку на сервер и ускорить загрузку страниц.
- Мощная система управления контентом: с помощью Jinja можно использовать множество инструментов для управления контентом на веб-страницах. Он поддерживает условные выражения, циклы, фильтры и другие функции, которые позволяют создавать многократно используемый код и избежать дублирования контента на странице, тем самым уменьшив количество кода.
- Простота разработки: использование Jinja упрощает разработку веб-приложений, так как разделение логики и представления помогает держать код более организованным и читаемым для программистов.

– Поддержка на уровне фреймворка: Jinja является частью многих веб-фреймворков, таких как Flask, Bottle, Pyramid и другие. Это означает, что можно использовать его прямо из коробки без необходимости устанавливать и настраивать его самостоятельно.

Кроме того, Jinja обладает множеством других преимуществ, которые облегчают работу веб-разработчиков и повышают качество веб-приложений.

## 2.5 Структура базы данных клиентской части приложения

При проектировании приложений разного уровня сложности рано или поздно встает вопрос хранения данных. Поэтому одним из самых важных этапов разработки является также построение схемы базы данных.

Проектирование базы данных – процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования баз данных:

- обеспечение хранения в БД всей необходимой информации;
- обеспечение возможности получения любых данных по всем запросам, которые доступны пользователю;
- обеспечение целостности базы данных.

Схема данных – это структура базы данных, а, именно, структура или структуры основных таблиц базы данных.

В общей сложности база данных будет содержать 5 таблиц. Логическая схема базы данных представлена в приложении В.

Список таблиц, представленных в базе данных и их описание можно изучить в таблице 2.1, которая представлена ниже.

Таблица 2.1 – Таблицы базы данных

Таблица	Описание
Customer	Содержит данные о доступных пользователях
Courier	Содержит данные о доступных курьерах
Category	Содержит список категорий товаров
Job	Содержит информацию о заказах пользователей
Transaction	Содержит данные для совершения транзакций

Все таблицы связаны между собой первичными и вторичными ключами. В основе таблицы «Customer» и «Courier» лежит модель User. Объекты User являются ядром системы аутентификации. Данная модель является автоматически встроенной в фреймворк и позволяет с самого начала использовать ее в веб-приложении.

Она хранит такую информацию как имя и фамилия пользователя, его пароли от аккаунта в зашифрованном виде, а также email и его никнейм в веб-приложении.

Таблица «Customer» предназначена для хранения данных о зарегистрированных пользователях приложения.

Данная таблица содержит информацию о пользователе, его контактные данные и данные для оплаты заказа. Также в нее пользователь может добавлять свою аватарку и номер телефона, который он должен будет подтвердить с помощью двухфакторной аутентификации, которая отправит ему СМС с кодом для входа. Описание представлено в таблице 2.2.

Таблица 2.2 – Описание таблицы «Customer»

Название	Тип данных	Ограничения	Описание
user	Uniqueidentifier Object	Primary key	Первичный ключ, уникальный идентификатор пользователя.
avatar	Image	Default(null)	Фотография пользователя
phone_number	Char(50)	Default(null)	Номер телефона пользователя
stripe_customer_id	Char(255)	Default(null)	Уникальный идентификатор пользователя в системе Stripe
stripe_payment_method_id	Char(255)	Default(null)	Уникальный идентификатор карты пользователя в системе Stripe
stripe_card_last4	Char(255)	Default(null)	Последние 4 цифры карты пользователя

Таблица «Courier» предназначена для хранения данных о зарегистрированных курьерах приложения. Также данная таблица хранит информацию о местоположении курьера, если он выполняет заказ, а также поле с хранением данных с помощью которых он будет получать вознаграждение за доставку товаров. В дальнейшем данная таблица будет использоваться для взаимодействия с заказами, информация о курьере, который получил заказ будет передавать в таблицу «Job». Поля с информацией о долготе и широте курьера являются не обязательными и предназначены для отображения курьера на карте, если у курьера отключено определение местоположения, то он не появится на карте у пользователя. Описание представлено в таблице 2.3.

Таблица 2.3 – Описание таблицы «Courier»

Название	Тип данных	Ограничения	Описание
user	Unique identifier	Primary key	Первичный ключ, уникальный идентификатор пользователя.
lat	Float	Default(0)	Широта на которой находится курьер
lng	Float	Default(0)	Долгота на которой находится курьер
paypal_email	EmailField	Default(null)	Почта от кошелька PayPal курьера
fcm_token	TextField	Default(null)	Необходимый токен для проведения оплаты за доставку

Таблица «Category» предназначена для хранения категорий, созданных администратором. Все категории являются уникальными и могут быть созданы и удалены только администратором приложения на странице администрирования приложения. Описание представлено в таблице 2.4.

Таблица 2.4 – Описание таблицы «Category»

Название	Тип данных	Ограничения	Описание
name	Char(255)	Primary key Unique	Первичный ключ, уникальный идентификатор названия категории.

Таблица «Transaction» предназначена для хранения данных относящихся к выводу на карту PayPal курьера, а также информацию о выполненном заказе, время выполнения и статус получения вознаграждения. С помощью данных которые указал курьер в личном кабинете администратор будет отправлять вознаграждение за выполненные доставки при помощи админпанели. Описание таблицы для проведения оплаты курьеру представлена в таблице 2.5

Таблица 2.5 – Описание таблицы «Transaction»

Название	Тип данных	Ограничения	Описание
stripe_payment_intent_id	CharField	Unique	Уникальный идентификатор для каждой транзакции.
job	Object	ForeignKey	Объект содержащий информацию о выполненном заказе
amount	FloatField	Default(0)	Вознаграждение курьера за доставку товара
status	CharField(20)	Default=IN_STATUS	Статус выполнения оплаты на карту курьера
created_at	DateTimeField	Default=time-zone.now	Дата и время создания транзакции

Таблица «Job» предназначена для хранения данных о заказах информацией о доставке этих товаров, а также информация о курьере, выполняющем заказ и информация о ходе доставки, стоимость заказа. Также в этой таблице кроме фотографии товара при создании сохраняются фотки товара в момент получения заказа курьером и в момент доставки товара.

При создании товара ему присваивается статус, данный статус будет меняться по ходу выполнения заказа и сразу после выполнения появится в личном аккаунте пользователя во вкладке архивные заказы. Также при создании заказа необходимо указать размер товара из предложенных вариантов. При указании начальной и конечной точки заказа автоматически определяется долгота и ширина данного места и автоматически происходит демонстрация данных точек на карте, а затем уже построенный маршрут между этими двумя точками появится на кастомной карте в описании заказа. Стоимость за заказ рассчитывается автоматически перед публикацией заказа в зависимости от расстояния между точками.

Описание таблицы «Job» представлено в таблице 2.6.

Таблица 2.6 – Описание таблицы «Job»

Название	Тип данных	Ограничения	Описание
1	2	3	4
Id	Uniqueidentifier	Primary key	Первичный ключ, уникальный идентификатор заказа.
customer	Object	ForeignKey	Информация о пользователе создавшем заказ
courier	Object	ForeignKey	Информация о курьере взявшем заказ
name	CharField(255)	Not null	Название доставки
description	CharField(255)	Not null	Описание товара для доставки
category	Object	ForeignKey Not null	Категория товара для доставки
size	CharField(20)	Default=MEDIUM	Размер товара
quantity	Int	Default(1)	Количество товаров
photo	ImageField	not null	Изображение товара
status	CharField(20)	Default=CREATING_STATUS	Статус выполнения заказа
created_at	DatetimeField	default=timezone.now	Дата создания заказа
pickup_address	CharField(255)	Default(null)	Адрес с которого надо забрать товар
pickup_lat	FloatField	Default(0)	Долгота на которой находится товар
pickup_lng	FloatField	Default(0)	Ширина на которой находится товар
pickup_name	CharField(255)	Default(null)	Дополнительная информация про место получения заказа для курьера
delivery_address	CharField(255)	Default(null)	Адрес доставки
delivery_lat	FloatField	Default(0)	Долгота точки доставки товара
delivery_lng	FloatField	Default(0)	Ширина точки доставки товара

Продолжение таблицы 2.6

1	2	3	4
delivery_name	CharField(255)	Default(null)	Дополнительная информация про место доставки заказа для курьера
pickup_phone	CharField(50)	Default(null)	Номер телефона по которому нужно забрать заказ
duration	FloatField	Default(0)	Приблизительное время доставки
distance	FloatField	Default(0)	Расстояние между точками доставки
price	FloatField	Default(0)	Оплата за выполнение для курьера
pickup_photo	ImageField	Default(null)	Фото при получении товара курьером
pickedup_at	DateTimeField	Default(null)	Дата и время получение товара курьером
delivery_photo	ImageField	Default(null)	Фото при доставке товара курьером
delivered_at	DateTimeField	Default(null)	Дата и время доставки товара курьером

Данные из таблицы «Job» в дальнейшем используются для создания транзакции и оплаты курьеру за выполнение доставки.

## 2.6 Выбор дизайнерского решения

Дизайн веб-приложения играет очень важную роль, поскольку он может сильно влиять на восприятие пользователем качества, удобства и эффективности продукта.

Привлекательный и понятный дизайн сделает использование веб-приложения более приятным и интуитивно понятным, что повысит его привлекательность и популярность у пользователей приложения.

Дизайн должен быть адаптивным и подходящим для различных устройств, таких как компьютеры, смартфоны и планшеты. Дизайн веб-приложения также должен учитывать особенности и потребности конечных пользователей, особенности продукта и его бренд-идентичность.

Неверное использование цветов, шрифтов и элементов дизайна может оттолкнуть пользователей от продукта и снизить его эффективность. Поэтому важно уделять внимание дизайну и находить баланс между визуальной привлекательностью и функциональностью веб-приложения.

При рассмотрении аналогов, было выявлено, что на данный момент более привлекательным и лаконичным является стиль – минимализм.

В основе дизайна должна лежать простота и функциональность. Вместо придания объема используются простые цвета, формы, кнопки и иконки.

За счет такого дизайна будет легче донести до пользователя основные идеи и назначение нашего веб-приложения.

Приложения, использующие стиль минимализм, позволяют пользователю почувствовать себя владельцем чего-то современного, эффективного и элитного.

Цвета на веб-сайте должны гармонизировать между собой, необходимо выделить цвета для фона и для текста, которые будут создавать необходимый контраст для наибольшей удобочитаемости.

Кроме того, для достижения более высокой эффективности дизайна веб-приложения, необходимо учитывать принцип юзабилити. Он заключается в том, чтобы упростить работу пользователя с приложением и сделать взаимодействие максимально простым и понятным для него. Важно также уделить внимание скорости загрузки сайта, чтобы пользователь мог быстро получить доступ к необходимой для него информации.

Также стоит учитывать, что веб-приложение должно быть адаптировано под различные устройства и экраны, такие как мобильные телефоны, планшеты, ноутбуки и т.д. Это поможет обеспечить комфортный доступ к приложению в любой момент и из любой точки мира.

В целом, минималистичный стиль дизайна веб-приложения является универсальным и эффективным, т.к. делает его более понятным, удобным и привлекательным для пользователей. Однако, необходимо учитывать особенности целевой аудитории и адаптировать стиль дизайна под ее потребности и привычки.

В качестве основных цветов обычно выбираются 3-4 главных цвета, на основании которых и создается основная палитра сайта. Главными оттенками сайта были выбраны оттенки серого цвета, оранжевый и синий для навигационной части и кнопок, используемых в приложении.

## 2.7 Выводы по разделу

Программный продукт включает в себя следующие компоненты:

- сервер базы данных SQLite;
- сервер приложения на базе языка Python и фреймворка Django, представляющий REST API [6];
- клиентское приложение на Django, CSS/HTML при помощи шаблонизатора Jinja2 и статических файлов.

Была построена схема архитектуры серверной части и диаграмма вариантов использования. В ходе проектирования была получена блок-схема, демонстрирующая процесс определения характера человека по действиям в системе.

### 3 Реализация программного средства

#### 3.1 Реализация серверной части приложения

Django – это популярный фреймворк для веб-разработки на языке Python. Он предоставляет различные модули, инструменты и библиотеки для создания высококачественных, современных веб-приложений. Одним из таких инструментов является Django REST framework, который предоставляет возможность создавать API для взаимодействия с веб-приложением.

Серверная часть приложения представляет собой WebAPI [7] приложение, обеспечивающее общение с клиентом по средству передачи данных.

Разработку серверной части можно разделить на несколько частей:

- разработка уровня доступа к данным, который включает разработку контекста и сущностей базы данных;
- разработка уровня бизнес-логики, включающая разработку сервисов;
- разработка уровня представления, включающая разработку контроллеров и конечных точек приложения.

В проекте присутствует файл «setting.py», часть его реализации, а именно установленные библиотеки представлены на рисунке 3.1.

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'core.apps.CoreConfig',  
    'bootstrap4',  
    'social_django',  
    'channels',  
]
```

Рисунок 3.1 – Установленные библиотеки

На рисунке представлено список обязательных библиотек, используемых при разработке веб-приложении.

					БГТУ 03.00.ПЗ						
		ФИО	Подпись	Дата							
Разраб.	Руденя П.А.				3 Реализация программного средства				Лист	Листов	
Провер.	Берников В.О.								1	13	
Н. контр.	Нистюк О.А.							74217088, 2023			
Утв.	Смелов В.В.										



### 3.2 Файловая структура

Перед началом разработки приложения была продумана файловая структура проекта, в соответствии с выбранной архитектурой. Файловая структура программного средства представлена на рисунке 3.2.

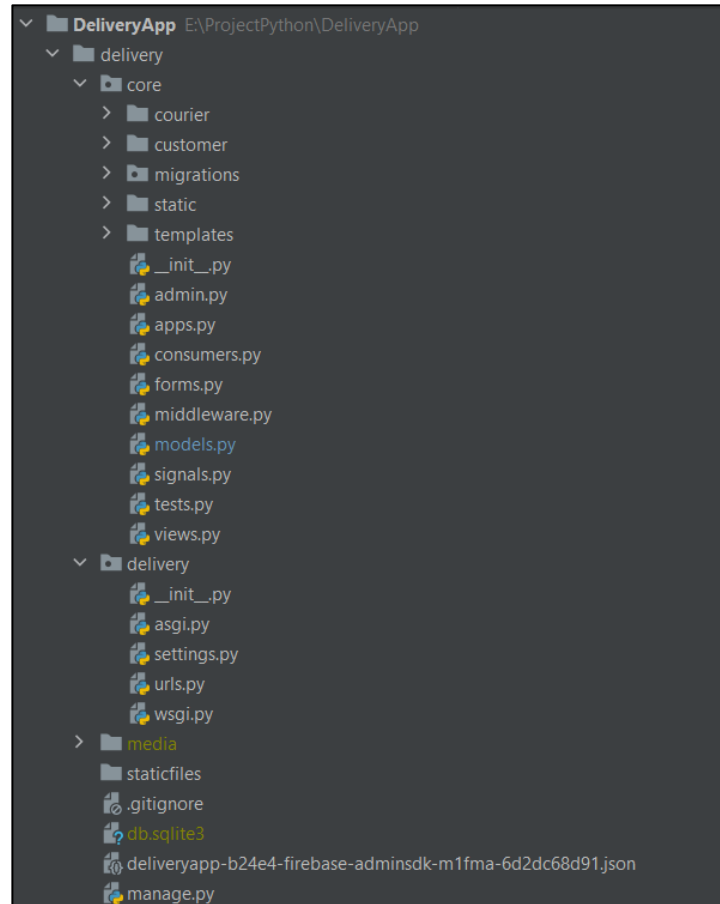


Рисунок 3.2 – Файловая структура

Проект состоит из следующих основных папок:

- media содержит изображения которые пользователи устанавливают в своем личном кабинете, а также изображения которые отправляет курьер при получении и доставке товара;
- core содержит файлы контроллеров для обработки действий на страницах, а также шаблоны страниц и статические файлы, используемые в приложении;
- delivery содержит файлы создаваемые при создании проекта, данные файлы являются обязательными и они полностью контролируют работу приложения.

Описание основных файлов, используемых в проекте:

- manage.py – файл, который позволяет управлять Django-приложением при помощи командной строки;
- \_\_init\_\_.py – файл, который говорит Python, что это пакет;
- settings.py – файл настроек проекта, здесь можно изменять настройки базы данных, временную зону, указать установленные приложения и прочее;
- urls.py – файл, содержащий маршруты проекта. Здесь настроены все маршруты URL для проекта;

- asgi.py – ASGI-сервер (Asynchronous Server Gateway Interface) для проекта;
- wsgi.py – WSGI-сервер (Web Server Gateway Interface) для проекта;

Каждое приложение обычно имеет свою структуру файлов и папок, включая файлы `views.py` (отвечает за обработку HTTP-запросов), `models.py` (определяет модели базы данных), `migrations/` (хранение миграций базы данных).

Кроме этого, можно создавать свои файлы и папки внутри приложения или проекта в зависимости от потребностей проекта.

### 3.3 Интеграция Google Map API

Google Maps API – это сервис, предоставляемый Google, который позволяет использовать карты Google в сторонних приложениях.

Google Maps API [8] предоставляет разработчикам доступ к мощным инструментам для работы с картами, такими как создание путевых точек, построение маршрутов, взаимодействие с местами и другими логическими элементами на карте. API работает на основе JavaScript, что позволяет легко интегрировать его в веб-приложения. API включает в себя несколько различных пакетов:

- Maps JavaScript API – пакет, который позволяет использовать карты на веб-страницах в ваших веб-приложениях.
- Roads API – пакет, который позволяет работать с дорожными данными, такими как номера дорог, ограничения скорости, полосы движения и т.д.
- Places API – пакет, который позволяет использовать информацию о местоположениях, такую как адреса, координаты, названия бизнесов и т.д.
- Static Maps API – пакет, который позволяет создавать изображения статических карт без необходимости использования JavaScript.
- Directions API – пакет, который позволяет строить маршруты для автомобилей, общественного транспорта, велосипедов и пешеходов.

При разработке дипломного проекта использовались такие пакеты как Maps JavaScript API и Directions API.

Использование Google Maps API позволяет создавать пользовательские карты, отображать на карте информацию о местоположениях, маршрутах, взаимодействовать с местами и многое другое.

Однако, для использования Google Maps API необходимо всегда иметь действующее разрешение от Google и соблюдать определенные правила использования данных и возможностей API. Также Google взимает плату за использование некоторых его функций в API в соответствии с тарифным планом.

На каждое приложение накладывается определенный лимит количества отправляемых запросов к API и в случае превышения лимита по истечению месяца будет выставлена оплата за пользование услуг Google.

Интеграция Google Maps API в Django фреймворке может быть достигнута при выполнении этих действий:

Во-первых, необходимо зарегистрировать приложение в личном аккаунте разработчика на сайте <https://developers.google.com>.

После этого получить API-ключ от веб-сайтов Google Maps и установить его в файле `settings.py`, в котором хранятся все ключи для работы с API.

Затем создаем представление, которое будет отображать карту Google. Можно использовать API-ключ в представлении напрямую или передать его в виде аргумента в шаблон. Пример метода для работы с картой в листинге 3.1.

```
elif request.POST.get('step') == '3':
    step3_form=forms.JobCreateStep3Form(request.POST,
    instance=creating_job)
    if step3_form.is_valid():
        creating_job = step3_form.save()
    try:
        r = requests.get(
        "https://maps.googleapis.com/maps/api/distancematrix/json?origins={{
        &destinations={{&mode=transit&key={}}".format(
        creating_job.pickup_address,
        creating_job.delivery_address,
        settings.GOOGLE_MAP_API_KEY,))
        print(r.json()['rows'])
        distance = r.json()['rows'][0]['elements'][0]['distance']['value']
        duration = r.json()['rows'][0]['elements'][0]['duration']['value']
```

Листинг 3.1 –Метод для работы с Google API

После этого мы создаем шаблон, на странице которого будет отображаться карта Google. Можно использовать JavaScript API для отображения карты Google. Пример шаблона для отображения карты в листинге 3.2.

```
<script
src="https://maps.googleapis.com/maps/api/js?key={{
GOOGLE_MAP_API_KEY}}&
callback=initMap&libraries=places&v=weekly"defer>
</script>
<script>
function initMap() {
const map = new google.maps.Map
(document.getElementById("map"), {
zoom: 13,
center: { lat: 53.9, lng: 27.5656 },});
var bounds = new google.maps.LatLngBounds();
fetch("
{% url 'courier:available_jobs_api' %}
")
then(response => response.json())
```

Листинг 3.2 – Добавление Google Maps в шаблон

После выполнения данных действий банковская карта полностью интегрирована в разрабатываемый дипломный проект.

Стоит отметить, что при разработке карт был установлен режим transit, который позволяет создать заказ при условии, что между точками есть общественный транспорт. Дополнительно в рамках дипломного проекта карта была модифицирована под нужды проекта, заменены иконки старта и конца маршрута, а также добавлено отображение текущего местоположения курьера.

### 3.4 Интеграция Stripe API

Stripe API – это набор инструментов, который позволяет разработчикам интегрировать платежи непосредственно в свои веб-сайты и приложения. API Stripe предоставляет богатые функциональные возможности для выполнения таких задач, как обработка кредитных карт, фактурирование, подписки, обмен сообщениями, аналитика и многое другое. Он является одним из самых популярных сервисов для онлайн-платежей и используется многими крупными онлайн-магазинами и платежными сервисами по всему миру.

После создания аккаунта пользователю для того чтобы разместить свой заказ на доставку необходимо привязать свою банковскую карту. После выбора конечной и начальной точки приложение рассчитывает стоимость доставки и если она устраивает пользователя, то он подтверждает свой заказ и с помощью Stripe API автоматически происходит оплата заказа. В листинге 3.3 представлен пример создания транзакции, которая и будет в дальнейшем использоваться для оплаты пользователем за заказ, данная транзакция создается автоматически.

```
if creating_job.price:
    try:
        payment_intent = stripe.PaymentIntent.create(
            amount=int(creating_job.price * 100),
            currency='usd',
            customer=current_customer.stripe_customer_id,
            payment_method=current_customer.stripe_payment_method_id,
            off_session=True,
            confirm=True,)
        Transaction.objects.create(
            stripe_payment_intent_id=payment_intent['id'],
            job=creating_job,
            amount=creating_job.price )
```

Листинг 3.3 – Создание транзакции Stripe API

Если на карте недостаточно денежных средств пользователю не удастся создать заказ, и он получит ошибку о недостаточном количестве денежных средств на карте. На банковской карте осуществляется резервация суммы и она будет окончательно списана после выполнения заказа, это сделано для того чтобы не было моментов, когда на момент выполнения заказа у пользователя не хватает денег для оплаты за заказ.

Также стоит отметить, что для взаимодействия со Stripe API [9] используется сразу два ключа, публичный и приватный. Это необходимо для обеспечения безопасности при проведении банковских платежей, данный момент будет подробно рассмотрен в разделе информационная безопасность.

### 3.5 Интеграция PayPal API

PayPal API – это набор программных интерфейсов, который позволяет разработчикам производить платежи, проверки платежей, возвраты и другие операции, связанные с платежами, с помощью интеграции с PayPal. PayPal API предоставляет

возможности для различных типов платежей, включая кредитные карты, банковские счета и электронные кошельки.

PayPal API [10] предоставляет два различных интерфейса – REST API и Classic API. REST API используется для создания приложений, которые работают с новыми версиями PayPal, а Classic API используется для взаимодействия с более ранними версиями PayPal. При разработке дипломного проекта было принято использовать при разработке REST API PayPal.

В данном проекте данная технология использовалась для оплаты вознаграждения курьеру за выполнение доставки. Курьеру необходимо в личном кабинете указать свои реквизиты, на которые он будет получать оплату.

В самом начале необходимо подключить библиотеку к нашему проекту, а также в `setting.py` указать уникальный идентификатор пользователя и секретный ключ, они предназначены для отправки вместе с данными для PayPal API и используются для подтверждения данных и успешного выполнения транзакций.

Также необходимо создать view для обработки платежей, который будет принимать POST запросы с данными о платеже. Одновременно в настройках проекта необходимо установить URL-адрес для получения уведомлений IPN, чтобы уведомления о платежах поступали напрямую в приложение. В листинге 3.4 представлен скрипт для отправки денежных средств курьеру за заказ.

```
def payout_to_courier
(modeladmin, request, queryset):
    payout_items = []
    transaction_querysets = []
    for courier in queryset:
        if courier.paypal_email:
            courier_in_transactions
            = Transaction.objects.filter(
            job__courier=courier,
            status=Transaction.IN_STATUS)
            if courier_in_transactions:
                transaction_querysets.append(courier_in_transactions)
            balance = sum(
            i.amount for i in courier_in_transactions)
            payout_items.append(
            {
            "recipient_type": "EMAIL",
            "amount": {
            "value": "{:.2f}".format(balance * 0.8),
            "currency": "BYN"},
            "receiver": courier.paypal_email,
            "note": "Thank you.",
            "sender_item_id": str(courier.id)
            }
            )
```

Листинг 3.4 – Скрипт работы с запросом от PayPal API

Если данные прошли полную валидацию, то денежные средства будут в полном объеме доставлены курьеру на его счет.

Для отправки денег на счет курьеру администратор приложения после выполнения заказа с помощью специальной формы производит оплату работы.

### 3.6 Отображение текущего местоположения курьера

Отображение местоположения курьера в службах доставки очень важно сразу по нескольким причинам, приведенным ниже.

Во-первых, это позволяет получателю узнать, когда точно ожидать курьера, чтобы не пропустить доставку.

Во-вторых, это помогает оптимизировать маршруты доставки, что позволяет сэкономить время и деньги на доставке.

В-третьих, это обеспечивает повышение безопасности и защиту от мошенничества, так как служба доставки может точно отслеживать местоположение курьера и распределять заказы между ними наиболее эффективно.

Исходя из перечисленных выше фактов было решено реализовать данную технологию в дипломном проекте.

Для отображения текущего местоположения курьера было решено использовать Google Maps JavaScript API [11], поскольку это простая технология.

Для получения местоположения необходимо чтобы курьер на своем мобильном устройстве разрешил использование геопозиции, которую запрашивает приложение, в противном случае его местоположение будет скрыто на карте.

В листинге 3.5 представлен скрипт для получения геопозиции курьера.

```
function updateCourierPosition(map) {
  const jobSocket = new WebSocket(
    "ws{% if request.get_host != 'localhost:8000' %}s{% endif %}://" +
    window.location.host + "/ws/jobs/{{ job.id }}/" );
  navigator.geolocation.watchPosition(      pos => {
    var courierPostion
    = new google.maps.LatLng(pos.coords.latitude, pos.coords.longitude);
    if (!window.courierMarker) {
      window.courierMarker = new google.maps.Marker({
        +position: courierPostion,
        icon: "{% static 'img/courier.png' %}" ); } else {
      window.courierMarker.setPosition(courierPostion); }
    map.panTo(courierPostion);
    try {jobSocket.send(JSON.stringify({
      job:
      {courier_lat: pos.coords.latitude,
      courier_lng: pos.coords.longitude,}))}
    }
    catch (error) {
      console.log(error); }
    pos => console.log(pos) ) }
```

Листинг 3.5 – Получение геопозиции курьера

Если пользователь предоставил свою геопозицию то он отображается на карте, а его широта и долгота сохраняется в базе данных для демонстрации его место нахождения или последнего места, где курьер демонстрировал свое местоположение.

В целом, отображение местоположения курьера в службах доставки позволяет улучшить качество обслуживания и повысить удовлетворенность клиентов.

### 3.7 Реализация WebSocket

WebSocket – это протокол двусторонней связи между клиентом и сервером, работающий поверх протокола HTTP. Этот протокол позволяет установить постоянное соединение между клиентом и сервером, которое может использоваться для обмена сообщениями в режиме реального времени.

WebSocket [12] вводит новый HTTP-признак "Upgrade" и использует его для того, чтобы переключить соединение на более "двухсторонний" режим. Это означает, что после установки соединения клиент и сервер могут обмениваться сообщениями без необходимости отправки новых запросов на каждое сообщение.

Для защиты пользователей и курьеров от халатности и возможного повреждения имущества, с целью защиты прав пользователей от недобросовестных курьеров было решено использовать технологию WebSocket, чтобы в режиме реального времени пользователь видел в каком состоянии курьер принимает заказ и в каком состоянии его передает. Это помогает защитить не только пользователя, но и курьера, ведь заказ изначально мог быть не качественным. В листинге 3.6 приведен пример кода для работы с WebSocket.

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'delivery.settings')
application = ProtocolTypeRouter(
{"http": get_asgi_application(),
"websocket": AuthMiddlewareStack(
URLRouter(websocket_urlpatterns))})
class JobConsumer(WebsocketConsumer):
def connect(self):
self.job_id = self.scope['url_route']['kwargs']['job_id']
self.job_group_name = 'job_%s' % self.job_id
async_to_sync(self.channel_layer.group_add)
(self.job_group_name,
self.channel_name)
self.accept()
def disconnect(self):
async_to_sync(self.channel_layer.group_discard) (
self.room_group_name,
self.channel_name)
def receive(self, text_data):
text_data_json = json.loads(text_data)
job = text_data_json['job']
print("Job", job)
```

Листинг 3.6 – Подключение и отключение от WebSocket

Также помимо моментального получение изображений в деталях заказа изменяется и статус заказа в зависимости от хода его выполнения. Данный протокол помогает пользователю не обновлять страницу каждый раз, чтобы отследить текущее местоположение своего заказа.

### 3.8 Реализация PWA

При разработке дипломного проекта одной из главных целей было использовать все новейшие технологии, чтобы приложение могло быть максимально современным и поддерживать все новинки. Одной из таких технологий является PWA [13]. Поскольку с большой долей вероятности курьеры будут пользоваться мобильной версией сайта, был сделан большой упор на разработку адаптации под мобильные устройства, для удобной работы с веб-приложением.

Прогрессивные веб-приложения (PWA) – это веб-приложения, призванные устранить границы между нативными приложениями и веб-страницами. PWA взаимодействуют с пользователем, как нативные приложения, но загружаются через браузер, как обычные веб-страницы.

Основные преимущества PWA:

- оффлайн работа. Многие веб-приложения теряют свою функциональность, если нет доступа к Интернету. PWA могут сохранять данные определенное время и работать в оффлайн-режиме.

- Быстрая загрузка. PWA загружаются моментально, в отличие от традиционных веб-страниц. Это повышает удобство использования и увеличивает скорости и количество пользователей.

- Не требует установки. PWA можно использовать прямо через браузер, не устанавливая приложение на устройство. Это сокращает количество технических препятствий для пользователей.

- Работают на любых устройствах. PWA могут работать на любых устройствах, включая смартфоны, планшеты, настольные компьютеры и ноутбуки.

Для реализации PWA в дипломном проекте необходимо было создать файл конфигураций, в котором хранилась небольшая информация о самом приложении., содержание файла в листинге 3.7

```
{
  "short_name": "DeliveryApp",
  "name": "Welcome to DeliveryApp",
  "description": "The best place for shipping",
  "icons":
  [
    {
      "src": "/static/img/logo.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": "/courier/",
  "display": "standalone"
}
```

Листинг 3.7 – Файл конфигураций для PWA

PWA используют современные веб-технологии, такие как Service Worker и Web App Manifest, для создания функциональности, которая раньше была доступна только на нативных приложениях. Это позволяет создавать веб-приложения, которые выглядят и работают, как нативные приложения, но с использованием веб-технологий.



После этого необходимо импортировать данный файл в базовый шаблон и можно пользоваться прогрессивным веб-приложением.

### 3.9 Разработка клиентской части приложения

HTML, CSS и JavaScript – это три ключевых языка для создания клиентской части сайта которые использовались в проекте.

HTML – это язык разметки, используемый для создания содержимого веб-страниц. С его помощью можно создавать различные компоненты.

CSS – это язык стилей, используемый для задания внешнего вида и оформления веб-страниц приложения. С его помощью можно изменять цвета, шрифты, расположение элементов и многое другое.

JavaScript – это язык программирования, используемый для создания клиентской логики и динамического взаимодействия с пользователем. Google картой и взаимодействием с другими API.

Все эти компоненты могут быть комбинированы и использованы для создания разнообразных интерфейсов веб-приложений.

Также стоит отметить, что для разработки дизайна использовался Bootstrap. Bootstrap – это бесплатный фреймворк для разработки пользовательского интерфейса (UI), который создан на основе HTML, CSS и JavaScript. Он содержит набор компонентов, стилей и скриптов, которые позволяют быстро создавать отзывчивые веб-страницы, адаптированные для различных типов устройств и разрешений экрана.

Bootstrap [14] также имеет преимущество для разработчиков, которые могут использовать готовые компоненты и стили, что существенно экономит время и упрощает процесс создания веб-страниц. В результате, создание веб-сайтов с помощью Bootstrap не только упрощает и ускоряет работу, но также обеспечивает более качественный и отзывчивый пользовательский интерфейс, поэтому он активно использовался в дипломном проекте.

Также для взаимодействия серверной части с клиентской использовался шаблонизатор Jinja2. Jinja2 – это библиотека шаблонизации для языка Python. Она позволяет использовать представления, которые можно написать на отдельном языке шаблонов, и затем подставлять в них данные. Jinja2 поддерживает условия, циклы, наследование шаблонов и другие функции. В общем, это удобный и гибкий инструмент для работы с шаблонами в веб-разработке с использованием Python.

Основными преимуществами использования Jinja2 с Django являются:

- удобство использования: Jinja2 может быть использован в Django без необходимости модификации существующего кода. Он поддерживает наследование шаблонов, условия, циклы, макросы, фильтры и другие функции, что заметно облегчает работу с шаблонами.

- Безопасность: Jinja2 защищает от CSRF (межсайтовой подделки запроса), предоставляя специальные функции, такие как `csrf_token()`, и обеспечивает безопасность пользовательских данных.

Для использования Jinja2 не надо ничего устанавливать, только необходимо в файле писать код python в фигурных скобках и интерпретатор распознает его как текст шаблонизатора и предоставит результат выполнения в виде html. Это заметно упрощает работу, ведь можно часть кода перенести сразу в тело страницы.

### 3.10 Взаимодействие с базой данных

Django предоставляет высокоуровневый интерфейс взаимодействия с базами данных (БД). Django поддерживает несколько систем управления базами данных, таких как PostgreSQL, MySQL, SQLite и других.

Django использует модель ORM [15] (Object-Relational Mapping, объектно-реляционное отображение), которое позволяет работать с БД, используя объекты Python. Модель ORM в Django позволяет определять схему БД с помощью классов Python, которые наследуются от базового класса Model.

По умолчанию Django в качестве базы данных использует SQLite. Она очень проста в использовании и не требует запущенного сервера. Все файлы базы данных могут легко переноситься с одного компьютера на другой.

Однако при необходимости мы можем использовать в Django большинство распространенных систем управления базами данных.

Для работы с базами данных в проекте Django в файле settings.py определен параметр DATABASES, который представлен в листинге 3.8.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Листинг 3.8 – Подключение к базе данных

Переменная DATABASES содержит набор конфигураций подключений к базам данных в виде словаря. Ключи в этом словаре – названия подключений. То есть мы можем определить кучу подключений.

Но как минимум одно из подключений должно быть определено в переменной DATABASES – подключение с именем default, которое представляет собой подключение по умолчанию.

Конфигурация каждого подключения может состоять из ряда параметров. По умолчанию указываются только два параметра.

Параметр ENGINE указывает на используемый движок для доступа к БД. В данном случае это встроенный пакет django.db.backends.sqlite3.

Второй параметр – NAME указывает на путь к базе данных. По умолчанию база данных называется db.sqlite3. Для установки пути используется каталог из переменной BASE\_DIR, которая задана в начале файла.

Модели в Django описывают структуру используемых данных. Используемые в программе данные хранятся в специальных базах данных, и с помощью моделей как раз осуществляется взаимодействие с нашей базой данных. При создании приложения по умолчанию в его каталог добавляется файл models.py, который применяется для определения моделей.

Модель представляет класс, унаследованный от django.db.models.Model. После того как класс для модели успешно создан необходимо при помощи командной строки необходимо сначала создать миграцию, а потом внедрить ее в проект.

Пример создание модели на примере таблицы Customer представлен в листинге 3.9. В данном листинге описывается процесс создания модели таблицы для хранения данных о пользователе приложения.

```
class Customer(models.Model):
    user = models.OneToOneField
    (User,on_delete=models.CASCADE)
    avatar=models.ImageField(upload_to='customer/avatars/',
    blank=True,null=True, verbose_name =Аватарка,help_text=" Аватарка ")
    phone_number = models.CharField(max_length=50,blank =True,
    verbose_name ='Номер телефона',help_text="Номер телефона")
    stripe_customer_id = models.CharField(max_length = 255,blank=True)
    stripe_payment_method_id =models.CharField(max_length255,blank=True)
    stripe_card_last4 = models.CharField(max_length =255,blank=True)
```

Листинг 3.9 – Создание модели базы данных

После того как база данных создана и у нас есть готовая модель необходимо осуществить миграцию, которая создаст данную таблицу в базе данных из модели. Затем мы можем в полном объеме взаимодействовать с базой данной, добавлять и изменять в ней информацию.

### 3.11 Реализация авторизации и регистрации

Для регистрации и авторизации было принято решение не создавать отдельные модели, а воспользоваться уже встроенным в фреймворк модулем.

UserCreationForm – это модель формы Django, позволяющая удобно создавать форму пользователя для создания новых пользователей в приложении. Эта форма предоставляет поля для имени пользователя, email и пароля, а также встроенную валидацию, чтобы убедиться, что выбранное имя пользователя не занято другим пользователем и что введенный пароль соответствует требованиям безопасности. Настройка модели в листинге 3.10.

```
class SignUpForm(UserCreationForm):
    email = forms.EmailField(max_length=50)
    first_name = forms.CharField(max_length=30)
    last_name = forms.CharField(max_length=30)
    class Meta:model = User
    fields =('email','first_name', 'last_name','password1','password2')
    def clean_email(self):
    email = self.cleaned_data
    ['email'].lower()
    if User.objects.filter(email = email):
    raise ValidationError(" Данная почта уже существует.")
    return email
```

Листинг 3.10 – Модуль для регистрации

Чтобы использовать UserCreationForm, необходимо импортировать его из модуля django.contrib.auth.forms и настроить его в вашем представлении/контроллере.

Затем нам остается только импортировать данную форму на страницу. Данная форма сразу готова к работе и в ней автоматически встроена валидации, которая позволяет не тратить лишнее время на разработку и заметно уменьшает стоимость разработки веб-приложения.

Если во время заполнения формы будет допущена ошибка, то фреймворк предупредит об ошибке моментально, без перезагрузки страницы, что позволит пользователям приложения быстрее проходить этап регистрации.

### **3.12 Выводы по разделу**

В данном разделе была описана программная реализация веб-приложения. Были подробно рассмотрены реализации серверной части и веб-интерфейса приложения, а также интеграция API.

Были даны примеры и определения данным, которые составляют серверную часть приложения. Описан принцип работы каждого представленного компонента и его краткое описание функционала.

В данной главе были описаны модели данных, что позволяет сопоставить результаты проектирования базы данных приложения в 2 главе с реализацией в серверной части приложения. Благодаря подробно рассмотренным элементам приложения можно представить общий процесс работы.

Серверная часть содержит около 12000 строк кода клиентская часть в общей сложности содержит порядка 18000 строк кода, не включая встроенные модули и библиотеки, весь код также поделен по компонентам и моделям, для сохранения существующей архитектуры приложения.

Итогом выполнения разработки программного средства стало веб-приложение, которое реализует доставку товаров в полном объеме.

Полученное веб-приложение не требует специфического программного обеспечения, для его запуска необходимо наличие установленного веб-браузера на компьютере пользователя и доступ в сеть интернет.

Интерфейс проекта является интуитивно понятным и довольно простым для использования. У пользователя не должно возникать никаких трудностей с использованием данного приложения.

В целом можно сказать, что разработанная система полностью соответствует требованиям, поставленным перед проектом, и является функциональным и мощным инструментом для управления процессами в различных сферах деятельности.

## 4 Тестирование

Тестирование программного обеспечения – процесс выявления ошибок в программном обеспечении (ПО). На сегодняшний день при помощи тестирования можно предотвратить выход продукт с дефектами.

Тестирование является важным процессом в разработке программного обеспечения. Тестирование помогает повышать уверенность в том, что приложение будет работать корректно в самых разных условиях и с разными типами данных. В целом, тестирование является одним из самых ключевых этапов в жизненном цикле разработки программного обеспечения.

В функциональных тестах основное внимание уделяется бизнес-требованиям к приложению. Они проверяют только результат некоторого действия и не проверяют промежуточные состояния системы при выполнении этого действия.

Иногда возникает путаница между понятиями интеграционных и функциональных тестов, так, как и те и другие требуют взаимодействия нескольких компонентов друг с другом. Разница в том, что интеграционный тест нужен просто чтобы убедиться, что вы можете отправлять запросы к базе данных, тогда как функциональный тест будет ожидать получения из базы данных определенного значения в соответствии с требованиями продукта.

Smoke-тесты представляют собой базовые тесты, которые проверяют основные функциональные возможности приложения. Они должны выполняться быстро, поскольку цель таких тестов – убедиться, что основные возможности системы работают так как было запланировано.

Smoke-тесты полезно запускать сразу после создания новой сборки или сразу после развертывания (чтобы убедиться, что приложение работает правильно в новой, только что развернутой среде).

Автоматические тесты, напротив, выполняются машиной, которая использует заранее написанный тестовый скрипт. Такие тесты могут значительно различаться по сложности, от проверки одного метода в классе до обеспечения условий, в которых выполнение последовательности сложных действий в пользовательском интерфейсе приводит к одинаковым результатам. Такой подход гораздо стабильнее и надежнее по сравнению с тестами, выполняемыми вручную. При этом качество автоматического тестирования будет зависеть от качества тестовых скриптов.

В рамках дипломного проекта будем проверять удовлетворяет ли проект требованиям, которые были представлены в листе задания, а также протестируем его при помощи Postman. Так же рассматривался вариант создания фреймворка для автоматизированного тестирования UI и API, но данный момент – это не целесообразно и экономически не обосновано, но при дальнейшем развитии функционала можно рассмотреть автоматизацию.

					БГТУ 04.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Руденя П.А.			4 Тестирование		Лист	Листов
Провер.		Берников В.О.					1	6
Н. контр.		Нистюк О.А.						
Утв.		Смелов В.В.					74217088, 2023	

## 4.1 Ручное тестирование

Для проведения ручного тестирования был составлен чек-лист основной функциональности приложения. Чек-лист с функциями и дополнительной информацией представлен в таблице 4.1.

Таблица 4.1 – Функциональные тест-кейсы

Описание теста	Ожидаемый результат	Роль	Статус
1	2	3	4
Вход в приложение	Успешный вход в приложение. Переход на главную страницу	Пользователь, администратор, курьер	Успешно
Выход из аккаунта пользователя	Успешный выход из своей учетной записи. Переход на страницу авторизации	Пользователь, администратор, курьер	Успешно
Регистрация нового пользователя	Приложение уведомляет, что поля обязательны для заполнения	Пользователь	Успешно
Регистрация курьера	Успешное создание курьера, переход на главную	Курьер	Успешно
Создание заказа	Успешное сохранение заказа и успешное появление в профиле	Пользователь	Успешно
Просмотр свободных заказов на карте	Успешное отображение заказов, доступных курьеру	Курьер	Успешно
Изменение информации профиля	Успешное изменение информации в профиле.	Пользователь, курьер	Успешно
Просмотр истории заказов	Успешный переход на страницу с историей заказов	Пользователь, курьер	Успешно
Изменение статуса пользователя	Успешное изменение статуса пользователя.	Администратор	Успешно
Привязка банковской карты	Успешное отображение привязанной банковской карты	Пользователь	Успешно
Подтверждение номера телефона	Успешное подтверждение номера телефона	Пользователь	Успешно
Отправка вознаграждения курьеру на карту	Успешное выполнение транзакции	Администратор	Успешно
Просмотр списка зарегистрированных пользователей и курьеров	Успешное отображение списка пользователей и курьеров	Администратор	Успешно

Продолжение таблицы 4.1

1	2	3	4
Просмотр текущего местоположения курьера на карте	Успешное отображение курьера на карте	Пользователь	Успешно
Отправка изображений с заказом от курьера пользователю	Успешная отправка изображений пользователю	Курьер	Успешно

Выполнив последовательность тест-кейсов, получили фактический результат, который полностью соответствует ожидаемому.

## 4.2 Тестирование web-сервера

### 4.2.1 Доступ к защищенным ресурсам

Web-сервер должен проверять разрешения на доступ к ресурсам. Для проверки разрешений используются куки, выдаваемые при входе в систему.

Если пользователь получил токен, то он может получить доступ к ресурсам, для которых он авторизован. Если же что-то пошло не так, и токен не был получен, пользователь останется на странице логина.

При обращении к защищенному ресурсу без токена в заголовке, сервер перенаправит на страницу с кодом ошибки 403.

Пример ответа сервера при доступе неавторизованного пользователя к карте с заказами представлен на рисунке 4.1.

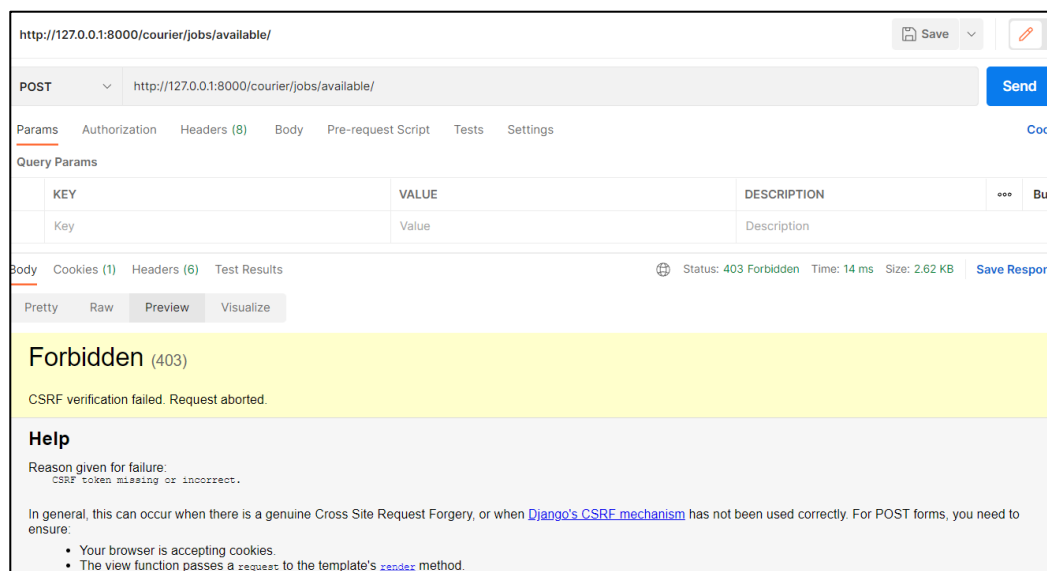


Рисунок 4.1 – Доступ к защищенному ресурсу

Выполнив последовательность действий на получение доступа к запрещенному функционалу, убедились, что на стороне сервера осуществляется проверка прав доступа у данного пользователя. Данная проверка показывает, что страницы с ограниченным функционалом полностью защищены от несанкционированного доступа и доступ к ним можно получить только с помощью токена.

### 4.2.2 Некорректность данных

Также в проекте реализована валидация на стороне клиента на случай неправильного поведения пользователя. На рисунке 4.2 приведен пример валидации на окне страницы регистрации в приложении.

На рисунке 4.2 представлено уведомление, которое пользователь получит, если не заполнит все поля. В данном дипломном проекте валидируются все поля ввода, на которых в случае некорректного ввода, может произойти ошибка или непредвиденное поведение программы. Визуально валидация представлена в виде уведомлений, которые появляются на странице при вводе данных, в которых найдена ошибка или какое-либо несоответствие.

The image shows a registration form titled "CUSTOMER". It contains several input fields: "Email" (filled with "leo@gmail.com"), "First name" (filled with "Архипова"), "Last name" (empty, with a placeholder "Last name"), "Password", and "Password confirmation". A yellow error message box with an exclamation mark icon is positioned over the "Password" field, displaying the text "Вы пропустили это поле." (You missed this field.). Below the "Password" field, there is a list of password requirements: "Your password can't be too similar to your other personal information.", "Your password must contain at least 8 characters.", "Your password can't be a commonly used password.", and "Your password can't be entirely numeric." The "Password confirmation" field is empty. At the bottom of the form is a yellow "Sign Up" button and a link "Already have an account? Sign In".

Рисунок 4.2 – Валидация на пустые поля в форме

Валидация данных – это процесс проверки входных данных на соответствие требованиям, заданным для конкретного приложения или системы. Это может включать проверку правильности формата данных, правильности использования символов, правильности длины и т.д.

Наличие валидации данных в программном обеспечении позволяет убедиться в том, что пользователь вводит корректные данные, что в свою очередь повышает качество и безопасность приложения или системы.



На рисунке 4.3 представлен пример сообщения, которое пользователь получит, если введет некорректные данные.

The image shows a registration form titled "CUSTOMER". It contains several input fields: "Email" with the value "leo@gmail@" and a red error message "Часть адреса после @ не может содержать символ «@»."; "Last name" with the value "Архипова"; "Password" with a masked value "....." and a list of password requirements; and "Password confirmation" with a masked value "....." and a note "Enter the same password as before, for verification.". A yellow "Sign Up" button is at the bottom.

**CUSTOMER**

Email

leo@gmail@

Часть адреса после @ не может содержать символ «@».

Архипова

Last name

Ольга Александровна

Password

.....

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation

.....

Enter the same password as before, for verification.

Sign Up

Рисунок 4.3 – Обработка некорректных данных.

Выполнив последовательность проверок основных полей форм внесения информации, определили, что валидация на клиенте приложения функционирует без каких-либо ожидаемых ошибок.

### 4.3 Модульное тестирование

Модульное тестирование Python – это процесс тестирования частей программного кода (модулей) или всего приложения для обеспечения их корректной работы и соответствия предполагаемым требованиям.

Для тестирования применяются специальные библиотеки, такие как unittest, pytest и doctest. Для модульного тестирования использовалась библиотека unittest и встроенный файл tests.py.

Unittest является встроенной библиотекой в Python, позволяющей создавать и запускать тесты, включающие в себя проверку результатов исходного кода. Существуют также сторонние библиотеки, такие как pytest, которые также обеспечивают мощные функции тестирования.

Автоматизированное модульное тестирование – это важный инструмент для обеспечения качества программного кода, ускорения разработки и снижения количества ошибок в продукте, проверку расширения файла, создание пользователя и проверка авторизация данных пользователей.

Для тестирования использовалась таблица категории из базы данных. Данная таблица прошла все проверки успешно. Скрипт выполнения данного тестирования функционала представлен в листинге 4.1.

```
from django.test import unittest
from django.contrib.auth import get_user_model
# Create your tests here.
from .models import Category
User = get_user_model()

class DeliveryTestCases(TestCase):
    def test_setUP(self) -> None:
    def test_fun(name, slug):
self.category = Category.object.create(name=name, slug=slug)
```

Листинг 4.1 – Листинг модульного тестирования

Результат проведенного тестирования можно увидеть на рисунке 4.4.

```
-----
Ran 1 test in 0.001s

OK
Destroying test database for alias 'default' ('file:memorydb_default?mode=memory&cache=shared')...
E:\ProjectPython\DeliveryApp\delivery>
```

Рисунок 4.4 – Результат модульного тестирования

Просмотрев результаты модульного тестирования можно прийти к выводу, что по итогу пройденных тестов уязвимостей и проблем в функциях не найдено.

#### 4.4 Выводы по разделу

В данном разделе подробно описывается процесс тестирования и исправления ошибок в программном обеспечении непосредственно во время разработки и после окончания процесса реализации системы.

Раздел включает в себя проверку сервера с помощью программного средства Postman, а также набор функциональных тест-кейсов для проведения пользовательского ручного тестирования.

## 5 Руководство пользователя

В данном разделе будет представлено руководство пользователя для ролей «Пользователь», «Курьер» и «Администратор».

### 5.1 Роль «Администратор»

При запуске веб-приложения и переходе на страницу админ, пользователь попадает на страницу авторизации, внешний вид которой представлен на рисунке 5.1.

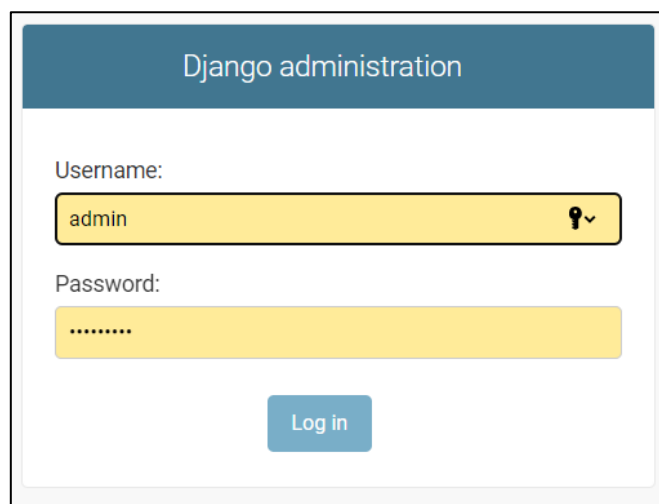


Рисунок 5.1 – Главная страница приложения

Администратору на данном моменте не доступен функционал приложения, для того, чтобы появились функциональные возможности приложения ему необходимо войти в систему, введя свой логин и пароль, который быть указан при создании приложения, как показано на рисунке 5.1.

После входа в систему администратору становится доступен полный функционал, предназначенный для него. Данный функционал предоставляется фреймворком и называется Django Administration.

Django Administration – это встроенный интерфейс Django для управления своими приложениями и данными в базе данных. Он предоставляет множество функций для просмотра, добавления, изменения и удаления записей из базы данных.

Кроме того, Django Administration позволяет создавать пользовательские модели и представления, настраивать доступ к админ-панели для разных пользователей и групп, а также добавлять различные расширения и плагины для улучшения функциональности. Благодаря этому, использование Django Administration делает управление приложениями на базе Django более удобным и быстрым процессом.

					БГТУ 05.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Руденя П.А.			5 Руководство пользователя		Лист	Листов
Провер.		Берников В.О.					1	16
							74217088, 2023	
Н. контр.		Нистюк О.А.						
Утв.		Смелов В.В.						

На рисунке 5.2 мы можем видеть главную страницу администратора, на которой отображается список всех классов в приложении с которыми он может взаимодействовать, а также список последних действий. По нажатию на любую из таблиц администратор может просмотреть информацию о содержимом таблицы и при необходимости добавить новые данные.

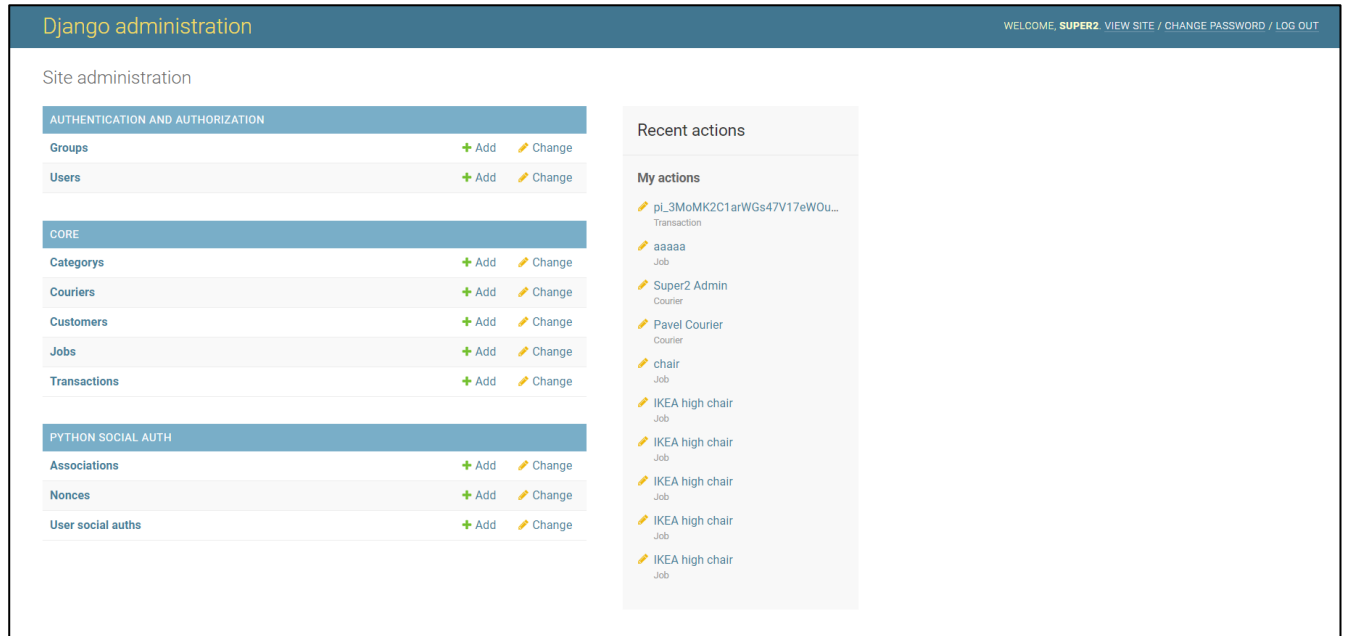


Рисунок 5.2 – Меню для администратора после входа

Для того чтобы добавить категории, необходимо нажать на кнопку добавить категорию. Откроется окно с полями, которые необходимо заполнить для добавления новой категории, как показано на рисунке 5.3. Также можно заполнять и другие таблицы базы данных, единственное отличие будет в количестве полей.

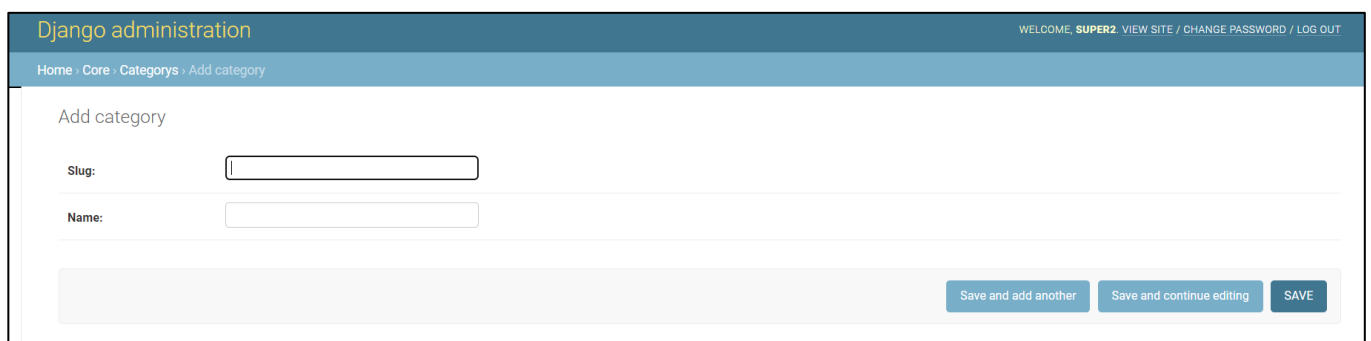


Рисунок 5.3 – Добавление категории

После добавления категории она сразу станет доступна для всех пользователей. На рисунке 5.4 показан пример отображения просмотра созданных заказов, в случае обнаружения ошибки или если информация не соответствует заказу, то администратор может деактивировать заказ. Данная страница полностью описывает заказ и содержит информацию о заказе, о пользователе и курьере, а также о ходе выполнения заказа включая текущее местоположение курьера. Некоторые поля данной таблицы имеют значения по умолчанию.

Это сделано потому что некоторые данные становятся известными после заполнения других полей. Ярким примером является поле стоимость, ведь только после ввода конечных точек приложение может рассчитать стоимость за выполнение заказа.

Рисунок 5.4 – Просмотр информации о заказе

После того как у нас появились заказы, которые полностью удовлетворяют требованиям и успешно выполнены курьером можно перейти к следующей функциональности, а именно отправки вознаграждения курьеру, который представлен на рисунке 5.5. Для начала стоит выбрать действие для отправки вознаграждение и выбрать курьера и после этого нажать кнопку выполнить.

USER FULL NAME	PAYPAL EMAIL	BALANCE
<input type="checkbox"/> Курьер Вторая комиссия	sb-pi9li19534339@personal.example.com	5.54
<input type="checkbox"/> dsdfft testov	sb-pi9li19534339@personal.example.com	0.0
<input type="checkbox"/> Super2 Admin	sb-pi9li19534339@personal.example.com	0.0

Рисунок 5.5 – Отправка вознаграждения курьеру

Если все выполнено успешно, то пользователь получит уведомление об успехе, в противном случае сообщение с указанием ошибки. На этом описание для роли администратора окончено. В целом, Django Administration представляет собой мощный инструмент для управления вашими приложениями и данными. Он может быть очень полезным в разработке и тестировании ваших приложений Django, а также для поддержки их в производственной среде.

На этом описание для роли пользователь окончено.

## 5.2 Роль «Пользователь»

В предыдущей части мы начинали с регистрации администратора, поэтому начнём с регистрации и в данном разделе. Пример регистрации можно увидеть на рисунке 5.6. Пользователю необходимо ввести свою почту, имя, фамилию и указать пароль. Если у пользователя уже есть аккаунт, то он должен нажать на кнопку внизу экрана с надписью войти в систему. После этого он будет перемещен на страницу авторизации. Данная страница похожа на страницу регистрации, главное отличие — это количество полей для записи. В ней необходимо ввести только почту и пароль, который был создан при регистрации. Авторизация более подробно продемонстрирована в руководстве пользователя для курьера.

The image shows a registration form titled "ПОЛЬЗОВАТЕЛЬ" (User). It contains the following fields and elements:

- Email:** A text input field containing "pavelrudenia@gmail.com".
- First name:** A text input field containing "Pavel".
- Last name:** A text input field containing "Rudenia3".
- Password:** A password input field (masked with dots) with a yellow background. Below it, there are four bullet points providing password requirements:
  - Your password can't be too similar to your other personal information.
  - Your password must contain at least 8 characters.
  - Your password can't be a commonly used password.
  - Your password can't be entirely numeric.
- Password confirmation:** A second password input field (masked with dots) with a yellow background.
- Verification text:** "Enter the same password as before, for verification."
- Registration button:** A large yellow button labeled "Зарегистрироваться" (Register).
- Login link:** A link labeled "Уже есть аккаунт? Войти в систему" (Already have an account? Log in).

Рисунок 5.6 – Регистрация пользователя

Если все данные введены корректно, то после регистрации пользователь будет перенаправлен в личный кабинет, который представлен на рисунке 5.7 и в Приложении Е. Там содержится вся информация, которую пользователь вводил при регистрации и есть возможность изменить и дополнить данные о себе.

Рисунок 5.7 – Страница профиля

Помимо изменения основной информации как имени, аватарки доступно изменение пароля и привязка номера телефона, как представлено на рисунке 5.8.

Рисунок 5.8 – Изменение пароля и подтверждение номера телефона

Перед тем как перейти к созданию заказа пользователю необходимо привязать банковскую карту, если он попытается создать заказ без карты его перекинет на страницу с привязкой карты. Данная страница представлена на рисунке 5.9, а также блок-схема работы в Приложении Г. Данные карты сразу проходят валидацию на корректность.

Рисунок 5.9 – Привязка банковской карты

Так как хранить данные карты в открытом виде небезопасно, ведь в случае взлома деньги могут быть украдены, то приложение не хранит данные о банковской карте, а лишь создает транзакцию для дальнейшей оплаты.

Данная технология является максимально безопасной и все данные пользователя остаются в безопасности.

Сохраняются лишь только 4 последние цифры карты для совершения транзакции и показываются в личном кабинете для того, чтобы пользователь видел какая банковская карта привязана.

Отображение привязанной карты в личном кабинете представлена на рисунке 5.10, также после успешной привязки банковской карты ее также можно удалить и привязать другую банковскую карту. Для удаления пользователю необходимо нажать красную кнопку удалить.

Рисунок 5.10 – Отображение привязанной карты

После того как карта успешно привязана можно переходить к странице создания заказа. На странице необходимо заполнить информацию о товаре и после этого нажать кнопку сохранения товара. Также обязательно стоит выбрать категорию товара и загрузить файл с изображением, а также указать размер товара в зависимости от характеристики и габаритов товара.

В описании товара стоит максимально подробно описать товар, чтобы курьеру было легче понять, что за заказ он будет доставлять. Чем подробнее будет описан заказ, тем больше шансов на скорейшее выполнение заказа, поскольку курьеру куда легче работать с полной информацией о товаре.

После ввода всех данных необходимо нажать кнопку сохранить. Скриншот данной страницы представлен на рисунке 5.11.



## Информация о заказе

Информация о заказе

Название

Описание

Категория

Размер

Количество

Фото  
 chair.jpg

Рисунок 5.11 – Заполнение информации о заказе

После того как данные о заказе введены, наступает второй этап создания заказа. В нем необходимо ввести данные откуда необходимо забрать товар и контактные данные. Стоит отметить что после ввода адреса происходит появление маркера на карте, как показано на рисунке 5.12.

Информация о заказе > **Получение** > Доставка > Оплата >

### Создать заказ

## Получение

Информация о месте получения заказа

Адрес получения посылки

Адрес получения посылки

Дополнительные данные

Дополнительные данные

Номер для связи

Номер для связи

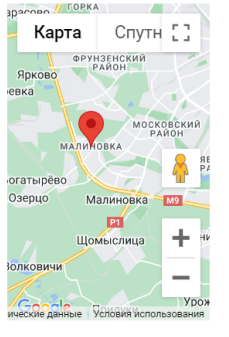


Рисунок 5.12 – Заполнение информации о стартовой точке доставки

Если все данные введены корректно, то можно переходить к заполнению информации о конечной точке доставки, как показано на рисунке 5.13. Для перехода дальше нажимаем кнопку сохранить.

Рисунок 5.13 – Заполнение информации о конечной точке доставки

Последним этапом создания заказа является страница, которая показывается стоимость за выполнение доставки, в зависимости от расстояния между точками, пример показан на рисунке 5.14.

Рисунок 5.14 – Страница с отображением стоимости за доставку

Следует отметить, что во время выполнения всех четырех этапов создания заказа в левом углу отображается краткая информация о пройденных этапах. Пример можно увидеть на рисунке 5.15.

КРАТКОЕ ОПИСАНИЕ ЗАКАЗА

Стул

1 Предмет

Размер упаковки не превышает 40x40x26см, вес не более 15 кг. Заказ

Место получения заказа

Павел

Малиновка, Минск, Беларусь

Место доставки товара

Николай


Стадион БГТУ, Минск, Беларусь

Рисунок 5.15 – Краткое описание заказа

Если пользователя устраивает сумма заказа, то он может успешно создать заказ. После этого данный заказ появляется в его личном кабинете, помимо активных заказов он может просмотреть архивные. Пример можно увидеть на рисунке 5.16.

Текущий заказ

Архивные заказы



Стул

Детский белый складной стул

Павел

Малиновка, Минск, Беларусь

Николай

Стадион БГТУ, Минск, Беларусь

Processing

BYN52.76

Рисунок 5.16 – Страница просмотра новостей

После того как заказ создан, наш пользователь может просмотреть информацию о ходе его выполнении и при необходимости может удалить заказ. Пример можно увидеть на рисунке 5.17. Данный заказ появляется в личном кабинете, и пользователь не сможет создать новый заказ до того момента пока у него будет в активном выполнении будет данный заказ.

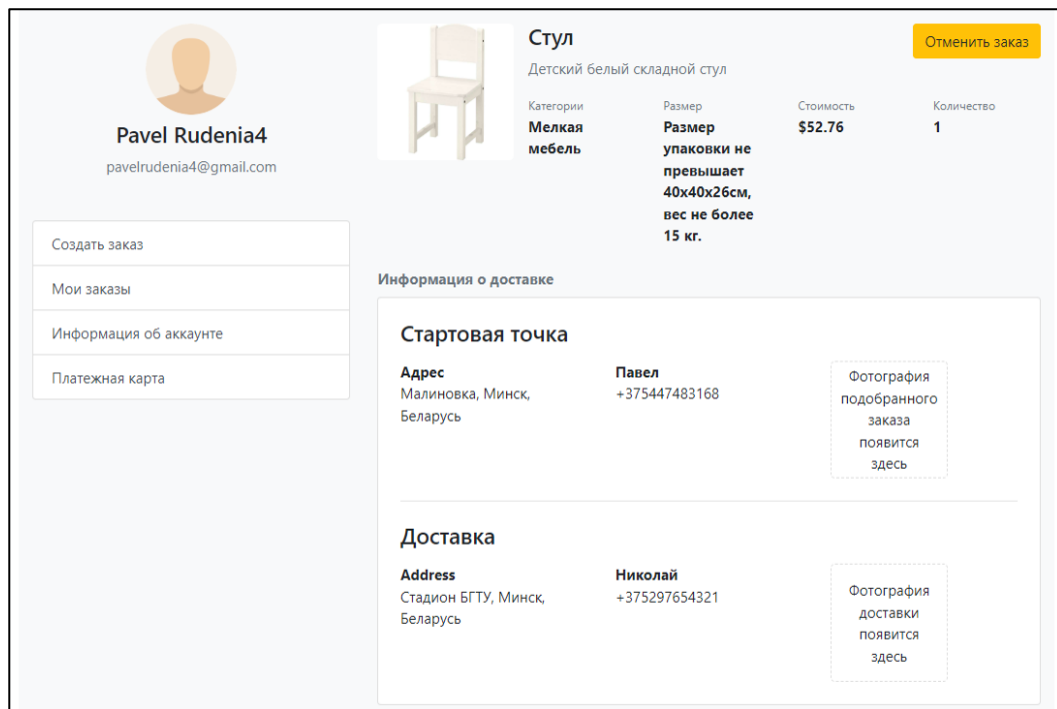


Рисунок 5.17 – Страница просмотра созданного заказа

На данной странице он будет получать фотографии от курьера в момент получения им заказа и в момент доставки. Также внизу экрана расположена карта которая показывает стартовую и конечную точку и текущее местоположение курьера, если он выполняет заказ. Пример можно увидеть на рисунке 5.18.

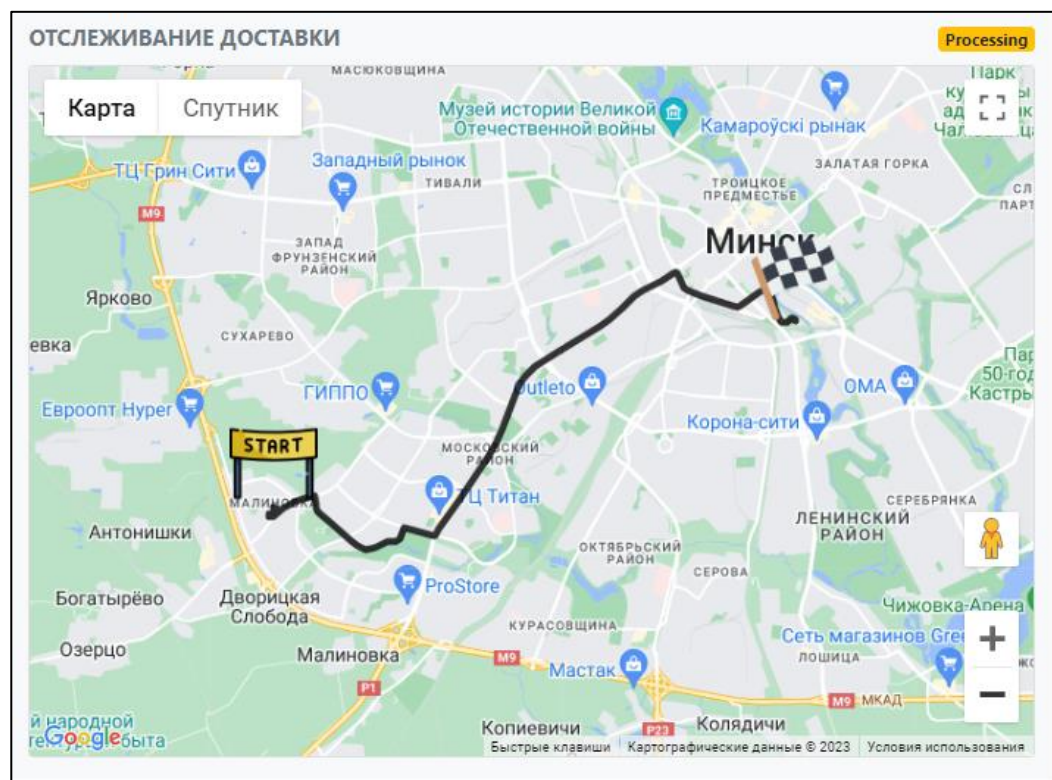


Рисунок 5.18 – Отслеживание выполнения заказа

На этом описание для роли пользователь окончено.

### 5.3 Роль «Курьер»

В предыдущей части мы создавали нового пользователя, поэтому начнём с входа в аккаунт курьера и здесь.

Пример входа в аккаунт курьера можно увидеть на рисунке 5.19.

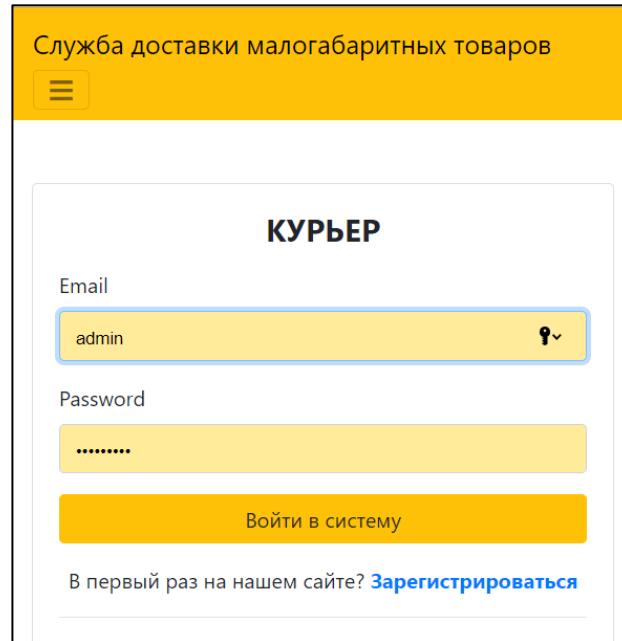


Рисунок 5.19 – Вход в аккаунт курьера

После успешного входа перед курьером открывается карта и отображаются свободные заказы. Пример представлен на рисунке 5.20.

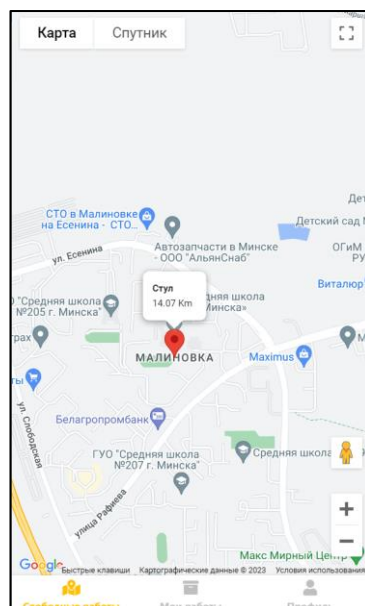


Рисунок 5.20 – Страница со свободными работами

Если курьер выбрал заказ на карте и хочет его просмотреть, то он должен кликнуть на маркер на экране. После этого откроется страница с информацией о работе. Пример представлен на рисунке 5.21.

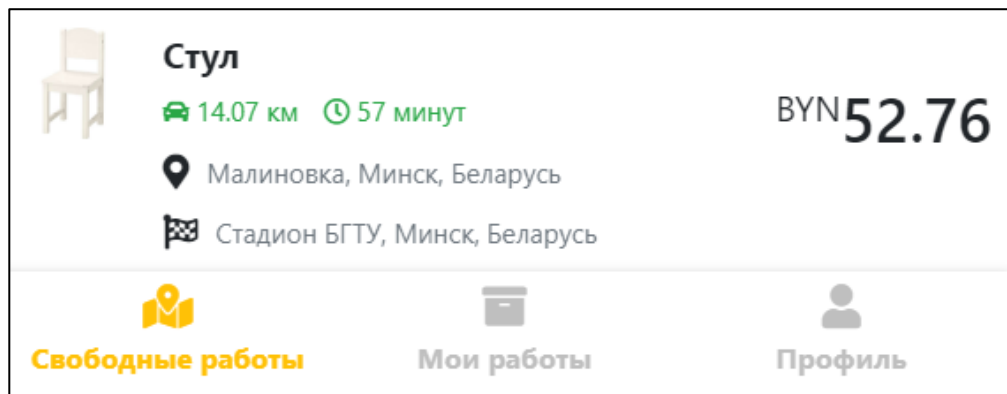


Рисунок 5.21 – Информация о работе

Если курьера заинтересовал данный заказ он должен нажать на заказ и откроется страница с его подтверждением. Пример показан на рисунке 5.22. Для детального изучения информации о заказе, курьеру достаточно просто нажать на сам заказ и он откроется в новом окне.

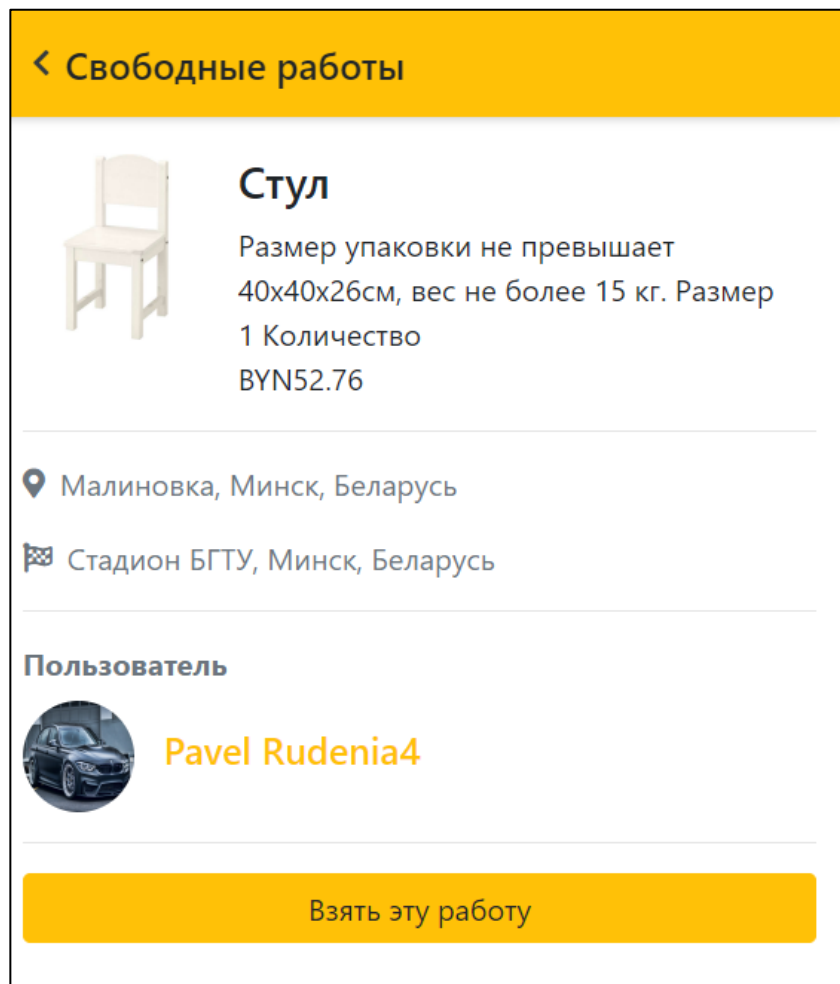


Рисунок 5.22 – Страница для подтверждения заказа

После того как он подтвердил заказ он может перейти на страницу со своей работой и начать выполнения. Перед ним будет карта с начальной и конечной точкой, его местоположения на карте и информация о заказе. Пример отображения страницы

представлен на рисунке 5.23. На данной карте используются специальные картинки, заменяющие начало и конец маршрута, а также иконка курьера.

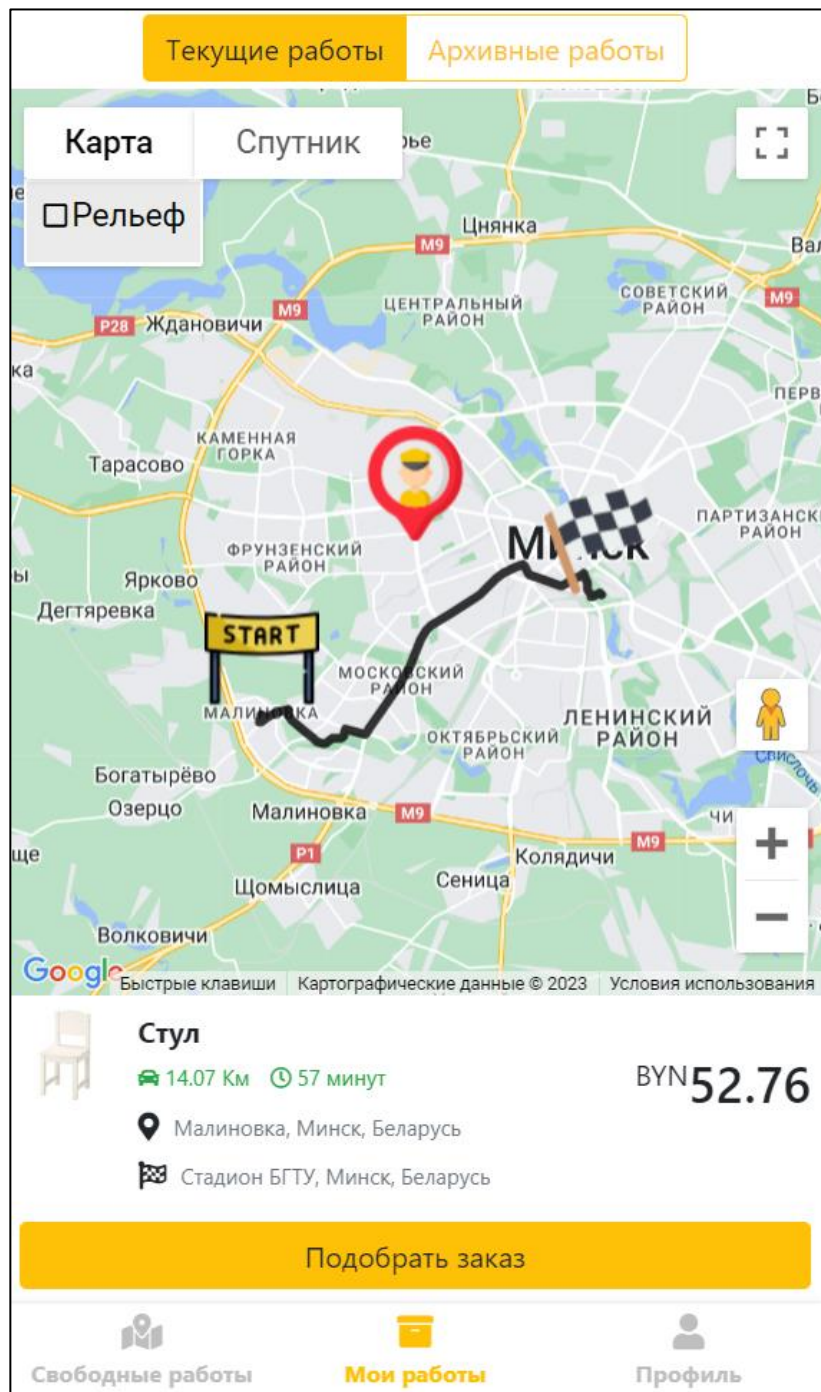


Рисунок 5.23 – Страница с текущей работой

После того как доберется до стартовой точки он должен нажать на кнопку подобрать заказ представленную на рисунке 5.23. У него откроется камера на телефоне, на которую он должен сфотографировать заказ при получении, данное действие является обязательным. Данное действие обязательно и фотография заказа моментально будет добавлена в личном кабинете пользователя. Также стоит отметить что местоположение курьера с заказом будет отображаться у пользователя в личном кабинете, если курьер на своем мобильном телефоне включил автоматическое определение местоположения. Пример на рисунке 5.24.





Рисунок 5.24 – Фото заказа при получении

Данную процедуру он должен проделать при доставке товара, она полностью идентична. Все изображения приходят пользователю в профиль с заказом.

После отправки изображения при доставке заказ будет завершен и открыта страница с успешной доставкой. Пример на рисунке 5.25

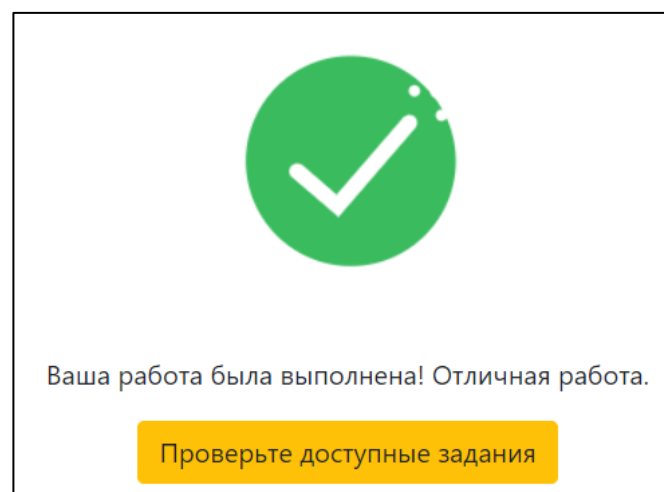


Рисунок 5.25 – Страница после выполнения заказа

После выполнения заказа курьер может зайти в личный кабинет и просмотреть информацию о выполненных заказах. Пример на рисунке 5.26.



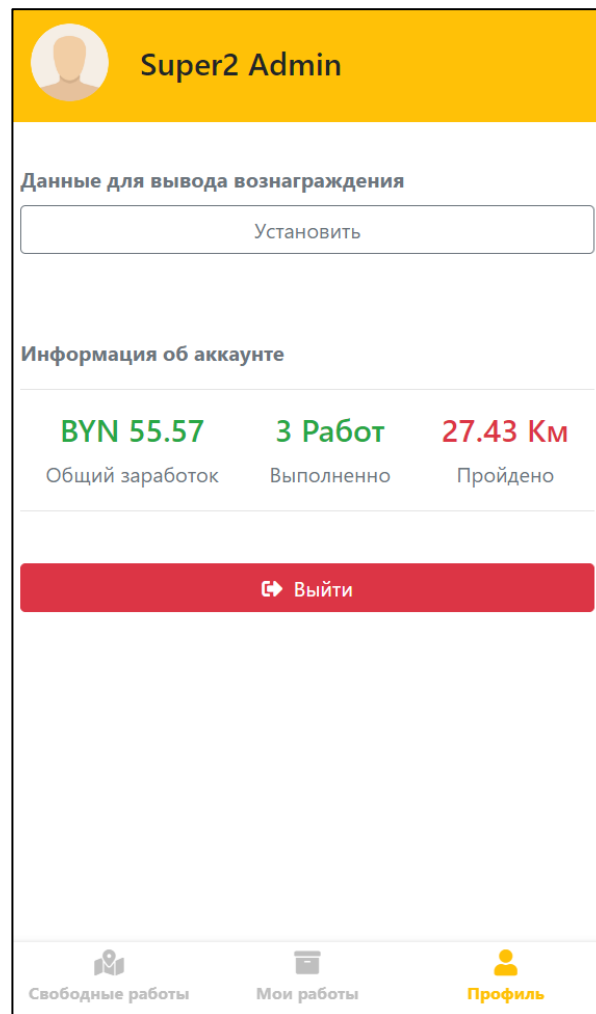


Рисунок 5.26 – Страница профиля курьера

Здесь отображается количество выполненных заказов, заработанная сумма денежных средств и суммарное расстояние между доставками. Следующей возможностью курьера является вывод вознаграждения за доставку. Для этого ему необходимо нажать на кнопку установить на рисунке 5.26 и ввести данные PayPal кошелька. Пример можно увидеть на рисунке 5.27.

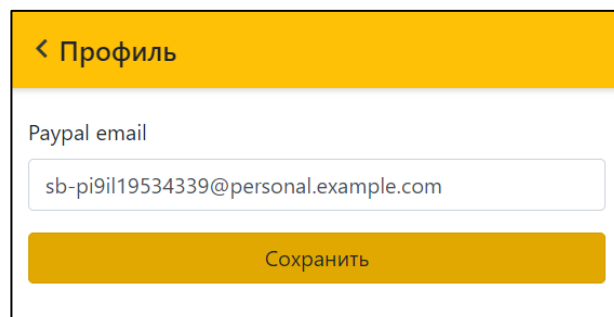


Рисунок 5.27 – Страница с вводом данных для вывода средств

За отправку вознаграждения отвечает администратор, который в любой момент может отправить вознаграждения за доставку каждому курьеру или только конкретному курьеру, который выполнил заказ.

Если пользователь не привязал данные для вывода средств, то администратор не сможет отправить денежные средства. После привязки данных на его кошелек могут поступать денежные средства, если у него уже есть выполненные заказы.

Подробнее о проведении оплаты написано в руководстве пользователя для администрации и в разделе разработка программного средства, а также в разделе анализ информационной безопасности приложения.

На этом описание для роли курьера окончено.

#### **5.4 Выводы по разделу**

Данный раздел представляет собой руководство пользователя, которое описывает все возможные моменты, которые могут возникнуть при работе с приложением для администратора, пользователя и курьера.

Цель этого раздела заключается в создании инструкции, которая позволит пользователям легко и быстро освоиться с функционалом приложения.

Для того, чтобы руководство было доступно и понятно, использовался более простой язык, который понятен людям, имеющим базовые знания в работе с компьютером и умеющим управлять мышью.

Для каждого доступного действия был разработан соответствующий снимок экрана с пояснением к нему, чтобы пользователи могли визуально увидеть каждый этап работы с приложением.

При проектировании и создании интерфейса приложения были изучены потребности и особенности каждой категории пользователей, что позволило создать дизайн, который интуитивно понятен и удобен в использовании. Скриншот работы приложения представлен в Приложении Д.

Функционал приложения ограничен небольшим количеством экранов, что значительно упрощает работу с приложением как для пользователя, так и для курьера. Отдельным преимуществом данного приложения является адаптивный интерфейс, который позволяет курьеру пользоваться веб-приложением с мобильного телефона что заметно облегчает работу и делает его более мобильным. Интерфейс на мобильном телефоне и компьютере полностью идентичен, что является очень важным, поскольку отсутствие некоторых возможностей на мобильном устройстве могло бы поставить под сомнение уровень приложения и как следствие приложение могло бы потерять потенциальных работников.

Благодаря созданию интуитивно понятного дизайна и разработке понятной инструкции, пользователи могут легко освоиться с приложением и использовать все его возможности для создания и выполнения заказов.

## 6 Технико-экономическое обоснование проекта

Особенностью современных бизнес-процессов в любой отрасли общественной деятельности является автоматизация сбора и обработка полученной информации для принятия управленческих решений.

Разработка проектов программного обеспечения требует затрат разнообразных и, нередко значительных объемов, ресурсов (трудовых, материальных, финансовых). В связи с этим, разработка и реализация каждого проекта должна быть обоснована, как технически, так и экономически.

Проект стоит разрабатывать, если он дает определенные преимущества по сравнению с известными передовыми аналогами или, в крайнем случае, по сравнению с существующей практикой. Поэтому, до того, как приступить к разработке проекта программного обеспечения, специалист должен, используя соответствующие методы, найти наиболее рациональное программное решение, обеспечивающее высокий технический уровень программы и дающее существенную экономию ресурсов, как при разработке проекта в научно-технической организации (у разработчика), так и при его реализации у пользователя (покупателя, заказчика).

Основной целью экономического раздела является экономическое обоснование целесообразности разработки программного средства (ПС), представленного в дипломном проекте. В этом разделе пояснительной записки проводится расчет затрат на всех стадиях разработки, а также анализ экономического эффекта в связи с использованием данного программного средства.

### 6.1 Общая характеристика разрабатываемого продукта

Основной целью экономического раздела является экономическое обоснование целесообразности разработки программного средства (ПС), представленного в дипломном проекте. В этом разделе пояснительной записки проводится расчет затрат на всех стадиях разработки, а также расчет экономии основных видов ресурсов в связи с использованием данного ПС.

В современных рыночных экономических условиях ПС выступает преимущественно в виде продукции организаций, представляющей собой функционально завершенные и имеющие товарный вид ПС ВТ, реализуемые покупателям по рыночным отпускным ценам.

Кроме того, в этом разделе проводится анализ рынка программных продуктов, определяются конкурентоспособность и уникальность предлагаемого программного средства, а также прогнозируется спрос на данное ПС. Результаты данных расчетов и анализов помогают принять обоснованное решение о целесообразности создания и внедрения программного продукта на рынок, а также о его окупаемости и прибыльности.

					БГТУ 06.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Руденя П.А.			6 Технико-экономическое обоснование проекта		Лист	Листов
Провер.		Берников В.О.					1	11
Консульт.		Соболевский А.С.					74217088, 2023	
Н. контр.		Нистюк О.А.						
Утв.		Смелов В.В.						

Важным элементом экономического раздела является также определение цены на продукт, которая может включать в себя затраты на разработку, производство, маркетинг и управление, а также прибыль, желаемую компанией разработчиком.

Общая цель экономического раздела дипломного проекта заключается в обосновании эффективности и жизнеспособности программного продукта на рынке, а также в определении потенциальных рисков и возможных стратегий по минимизации этих рисков.

Рассматриваемое программное средство представляет собой проект, позволяющий реализовать доставку товаров в полном объеме, от заказа пользователем до выполнения заказа курьером. Данное веб-приложение разработано для продажи данного программного средства и смежных прав на него заказчику.

## 6.2 Исходные данные для проведения расчетов

Исходные данные для расчета стоимости разработки программного продукта представлены в таблице 6.1.

Таблица 6.1 – Исходные данные для расчетов

Наименование показателя	Единица измерения	Условные обозначения	Норматив
Численность разработчиков	чел.	$Ч_p$	1
Норматив дополнительной заработной платы	%	$H_{дз}$	15
Ставка отчислений в Фонд социальной защиты населения	%	$H_{фсзн}$	34
Ставка отчислений в БРУСП «Белгосстрах»	%	$H_{бгс}$	0,6
Цена одного машино-часа	руб.	$C_{мч}$	0,06
Норматив прочих затрат	%	$H_{пз}$	18
Норматив накладных расходов	%	$H_{обп, обх}$	30
Норматив расходов на сопровождение и адаптацию	%	$H_{рса}$	10

Как было сказано выше, монетизация продукта будет осуществляться по принципу продажи прав на приложение. Таким образом, предполагается, что данным продуктом заинтересован определенный заказчик, которому необходим функционал, который предоставляет приложение, созданное в ходе дипломного проекта.

В ходе проведения маркетингового анализа, была выявлена стоимость разработки веб-приложения для разработки службы доставки. Минимальная цена разработки аналогичного продукта составляет 13 000 рублей.

Существует множество калькуляторов для расчета стоимости сайта. Они могут быть различных типов, в зависимости от того, какие параметры учитываются при расчете стоимости разработки сайта.

Для оценки стоимости разработки аналогичного продукта я использовал веб-ресурс [intid.ru](http://intid.ru), так как он позволяет учитывать не только основные факторы, такие как количество страниц, тип дизайна, функциональность, тип хостинга и т.д., а более

специфические факторы, такие как опциональные модули и плагины, интеграция со сторонними сервисами, настройка SEO и т.д.

Результаты расчетов стоимости разработки аналогичного приложения с помощью специального калькулятора приведен на рис 6.1.

The screenshot shows the INTRID website's cost calculator. The header includes the INTRID logo with the tagline 'Иновации в интернете' and navigation links: РАЗРАБОТКА, ПРОДВИЖЕНИЕ, ДИЗАЙН, ХОСТИНГ, and a button ЗАКАЗАТЬ САЙТ. The main heading is '<КАЛЬКУЛЯТОР СТОИМОСТИ САЙТА />'. There are two main sections: 'ОНЛАЙН-РАСЧЕТ СТОИМОСТИ САЙТА' and 'ОФОРМИТЬ БЫСТРЫЙ ПРЕДЗАКАЗ'. The first section describes the online calculator and has a button 'ПЕРЕЙТИ К РАСЧЕТУ'. The second section is for quick ordering and has a button 'БЫСТРЫЙ ЗАКАЗ'. Below these is a section titled '<РАСЧЕТ СТОИМОСТИ РАЗРАБОТКИ САЙТА ПО ИНДИВИДУАЛЬНЫМ ПАРАМЕТРАМ />'. It includes a dropdown for 'ПРЕДПОЛАГАЕМЫЙ ТИП САЙТА' with an option 'НЕТ' selected, labeled 'Сайт визитка или LandingPage'. At the bottom, it shows 'СТОИМОСТЬ САЙТА ПО ВЫБРАННЫМ ПАРАМЕТРАМ: 337500 Р' and a button 'ПОЛУЧИТЬ ПРОЕКТ САЙТА'. A WhatsApp icon is in the bottom right corner.

Рис 6.1 Расчет стоимости сайта intid.ru

Также был проведен расчёт стоимости разработки сайта на сайте <https://secretschool.ru/>. Результаты расчетов стоимости разработки аналогичного приложения на рисунке 6.2.

The screenshot shows the 'Секретная школа бизнеса' website. The header includes the company logo, name, and description: 'Бизнес-консалтинг, тренинги и курсы для достижения успеха, создание сайтов, ведение соцсетей.' It also lists contact information: info@secretschool.ru and +7-961-124-79-85. A navigation menu includes: Главная, Кратко о..., Услуги, Курсы и книги, Каталог товаров, Блог, Вакансии, and Контакты. The main heading is 'КАЛЬКУЛЯТОР СОЗДАНИЯ САЙТА'. Below it, it says 'Отметьте необходимые пункты, и получите автоматически сформированное КП на e-mail'. There is a section for 'РАЗРАБОТКА САЙТА' with an illustration of people working on a screen. The final result shown is 'Итого: 733320 Руб.'.

Рис 6.2 Расчет стоимости сайта secretschool.ru

Можно заметить, что после подсчета стоимости на разработку сайта было выявлено что цена, полученная в калькуляторах для расчета стоимости сайта, отличается в разы. Поэтому было провести еще один расчет на сайте и в дальнейшем определиться со стоимостью аналога.

Также был проведен расчёт стоимости разработки сайта на сайте <https://www.brevis-site.ru/calculator>. Результаты расчетов стоимости разработки аналогичного приложения на рис 6.3.

■ СКОЛЬКО БУДЕТ СТОИТЬ ТВОЙ САЙТ?

## КАЛЬКУЛЯТОР СТОИМОСТИ УСЛУГ

РАЗРАБОТКА САЙТА ⓘ

Тип сайта:

- ☐ Одностраничный landing page
- ☐ Сайт-визитка (до 15 страниц)
- ☐ Сайт с каталогом услуг

СТОИМОСТЬ

Разработка сайта: 286000 - 390000 руб. ▾

Итого: **286000 - 390000 руб.**

Рис 6.3 Расчет стоимости сайта brevis-site.ru

Минимальная цена разработки аналогичного продукта составляет 13 000 рублей. Таким образом, общая себестоимость разработки данного дипломного проекта будет сравниваться с указанной суммой для определения рентабельности.

### 6.3 Методика обоснования цены

В современных рыночных экономических условиях программное средство (ПС) выступает преимущественно в виде продукции организаций, представляющей собой функционально завершённые и имеющие товарный вид ПС, реализуемые покупателям по рыночным отпускным ценам. Все завершённые разработки ПС являются научно-технической продукцией.

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования ПС.

Выбор эффективных проектов программного средства связан с их экономической оценкой и расчётом экономического эффекта, который может определяться как у разработчика, так и у пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации ПС, остающейся в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счёт:

- снижения трудоёмкости расчётов и алгоритмизации программирования и проведение отладки программ;
- сокращения расходов на оплату машинного времени и других ресурсов на проведение отладки программ;
- снижения расходов на материалы;
- ускорение ввода в эксплуатацию новых систем;
- улучшения показателей основной деятельности проекта в результате использования программного средства.

Стоимостная оценка ПС у разработчиков предполагает определение затрат, что включает следующие статьи:

- заработная плата исполнителей – основная и дополнительная;
- отчисления в фонд социальной защиты населения;
- отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
- расходы на оплату машинного времени;
- прочие прямые затраты;
- накладные расходы.

На основании затрат рассчитывается себестоимость и отпускная цена программного средства, расчеты представлены ниже.

### 6.3.1 Объем программного средства

Для общей оценки объема программного средства, функции приложения оцениваются с помощью специальной классификационной таблицы, представленной в веб-приложении, в которой определяется объем каждой отдельной функции. Общий объем программного средства  $V_o$ , вычисляется как сумма объемов  $V_i$  каждой из  $n$  его функций формуле 6.1.

$$V_o = \sum_{i=1}^n V_i, \quad (6.1)$$

где  $V_i$  – объем  $i$ -ой функции ПС, условных машинных команд;  
 $n$  – общее число функций.

В ходе рассмотрения классификационной таблицы были выбраны функции, присутствующие в итоговом программном средстве. В таблице 6.2 представлены функции в условных машино-командах.

Таблица 6.2 – Содержание и объем функций в программном средстве

Номер функции	Содержание функции	Объем, условных машино-команд
101	Организация ввода информации	110
102	Контроль, предварительная обработка информации	550
111	Управление вводом/выводом	3900
204	Обработка наборов и записей базы данных	3375
301	Формирование последовательного файла	360
507	Обеспечение интерфейса между компонентами	850
703	Расчет показателей	600
707	Графический вывод результатов	410
	Итого:	10155

Исходя из данных таблицы 6.2, можно рассчитать объем программного средства, разработанного в процессе дипломного проектирования:

$$V_o = 110 + 550 + 3900 + 3375 + 360 + 850 + 600 + 410 = 10155 \text{ (условных машино-команд)}$$

Уточненный объем ПС  $V_o/$  равен произведению объема программного средства  $V_o$  на коэффициент изменения скорости обработки информации  $K_{ск}$ .

$$V_o/ = V_o \cdot K_{ск} \quad (6.2)$$

Исходя из вычисленного объема программного средства, можно определить уточненный объем программного средства:

$$V_o/ = 10155 \cdot 0,6 = 6093 \text{ (условных машино-команд)}.$$

Таким образом, объем программного средства, разработанного в ходе дипломного проекта, составил 10155 условных машинно-команд, а уточненный объем программного средства составил 6093 условных машино-команд.

### 6.3.2 Основная заработная плата

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в сфере веб-программирования на языке программирования Python.

Источником данных служили открытые веб-порталы, различные форумы, официальная отчетность, а также общий средний уровень заработка в сфере информационных технологий в Республике Беларусь.

В таблице 6.3 приведено количество дней потраченных на выполнение различных работ одним программистом.

Таблица 6.3 – Затраты рабочего времени на разработку ПС

Содержание работ	Затраты рабочего времени, дней
Инициализация клиентской и серверной частей	1
Инициализация базы данных	1
Инициализация WEB API	2
Написание логики для модуля регистрации	2
Написание логики для модуля авторизации	3
Написание логики для пользователя	6
Написание логики для курьера	6
Написание логики для модуля работы с картой	3
Тестирование приложения	6
Всего	30

Итогом изучения и анализа полученных данных, стала информация о том, что средняя месячная заработная плата для позиций junior составляет 2000 рублей. Проект разрабатывался одним человеком на протяжении одного месяца. Основная заработная плата рассчитывается по формуле (6.3):

$$C_{оз} = T_{раз} \cdot K_{раз} \cdot C_{зп} \quad (6.3)$$



где  $C_{оз}$  – основная заработная плата, руб.;  
 $T_{раз}$  – время разработки, месяцев;  
 $K_{раз}$  – количество разработчиков, человек;  
 $C_{зп}$  – средняя месячная заработная плата.

$$C_{оз} = (30/22) \cdot 1 \cdot 2000 = 2727,27 \text{ руб.}$$

Исходя из расчетов, мы получили, что основная заработная плата программиста при разработке программного средства составит 2727,27 руб, при разработке данного веб-приложения на протяжении 30 рабочих дней.

### 6.3.3 Дополнительная заработная плата

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле (6.4):

$$C_{дз} = \frac{C_{оз} \cdot H_{дз}}{100}, \quad (6.4)$$

где  $C_{оз}$  – основная заработная плата, руб.;  
 $H_{дз}$  – норматив дополнительной заработной платы, %.

$$C_{дз} = 2727,27 \cdot 15 / 100 = 409,1 \text{ (руб.)}.$$

Исходя из основной заработной платы, а также норматива дополнительной заработной платы, можно рассчитать сумму дополнительной заработной платы:

### 6.3.4 Отчисления в Фонд социальной защиты населения

Отчисления в Фонд социальной защиты населения (ФСЗН) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей.

Таким образом, отчисления в Фонд социальной защиты населения, вычисляются по следующей формуле (6.5):

$$C_{фсзн} = \frac{(C_{оз} + C_{дз}) \cdot H_{фсзн} + H_{стр}}{100}, \quad (6.5)$$

где  $C_{оз}$  – основная заработная плата, руб.;  
 $C_{дз}$  – дополнительная заработная плата на конкретное ПС, руб.;  
 $H_{фсзн}$  – норматив отчислений в Фонд социальной защиты населения, %.

$$C_{фсзн} = (2727,27 + 409,1) \cdot 34,6 / 100 = 1085,18 \text{ (руб.)}.$$

Таким образом, общие отчисления в фонд социальной защиты населения при разработке приложения составят 1085,18 рублей.

### 6.3.5 Расходы на материалы

Сумма расходов на материалы  $C_m$  определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода  $H_m$  на уточненный объем программного средства  $V_o'$  (формула 6.6).

$$C_m = H_m \cdot \frac{V_o'}{100}, \quad (6.6)$$

Учитывая, что норма расхода материалов в расчете на сто строк исходного кода равна 0,460 рублей, можно определить сумму расходов на материалы:

$$C_m = 0,46 \cdot 10155 / 100 = 46,713 \text{ (руб.)}.$$

Сумма расходов на материалы была вычислена на основе данных приведенных в таблице 6.1 данного дипломного проектирования.

### 6.3.6 Расходы на оплату машинного времени

Сумма расходов на оплату машинного времени  $C_{mv}$  определяется как произведение стоимости одного машино-часа  $C_{мч}$  на уточненный объем программного средства  $V_o'$  и на норматив расхода машинного времени на отладку ста строк исходного кода  $H_{mv}$  (формула 6.7).

$$C_{mv} = C_{мч} \cdot \frac{V_o'}{100} \cdot H_{mv}, \quad (6.7)$$

Учитывая, что норматив машинного времени на отладку ста строк исходного кода равен пятнадцати по условию расчетов, то можно определить сумму расходов на оплату машинного времени:

$$C_{mv} = 0,06 \cdot 10155 \cdot 15 / 100 = 91,4 \text{ (руб.)}.$$

### 6.3.7 Прочие прямые затраты

Сумма прочих затрат  $C_{пз}$  определяется как произведение основной заработной платы исполнителей на конкретное программное средство  $C_{оз}$  на норматив прочих затрат в целом по организации  $H_{пз}$  (формула 6.8).

$$C_{пз} = \frac{C_{оз} \cdot H_{пз}}{100}, \quad (6.8)$$

Все данные необходимые для вычисления у нас уже есть, поэтому можно определить сумму прочих затрат:

$$C_{пз} = 2727,27 \cdot 18 / 100 = 490,91 \text{ (руб.)}.$$

Таким образом, сумма прочих затрат при разработке данного программного средства составит 490,91 рублей.

### 6.3.8 Накладные расходы

Сумма накладных расходов  $C_{обп,обх}$  – произведение основной заработной платы исполнителей на конкретное программное средство  $C_{оз}$  на норматив накладных расходов в целом по организации  $H_{обп,обх}$  (формула 6.9).

$$C_{обп,обх} = \frac{C_{оз} \cdot H_{обп,обх}}{100}, \quad (6.9)$$

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов:

$$C_{обп,обх} = 2727,27 \cdot 30 / 100 = 818,18 \text{ (руб.)}.$$

Таким образом, сумма накладных расходов равно 818,18 рублей.

### 6.3.9 Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства  $C_p$  определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов (формула 6.10).

$$C_p = C_{оз} + C_{дз} + C_{фсзн} + C_m + C_{мв} + C_{пз} + C_{обп,обх} \quad (6.10)$$

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на разработку программного средства:

$$C_p = 2727,27 + 409,1 + 1085,18 + 46,713 + 91,4 + 490,91 + 818,18 = 5668,753 \text{ (руб.)}.$$

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе.

### 6.3.10 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства  $C_{рса}$  определяется как произведение суммы расходов на разработку  $C_p$  на расходы на сопровождение и адаптацию  $H_{рса}$ .

Таким образом, сумма расходов на сопровождение и адаптацию программного средства на сопровождение и адаптацию программного средства вычисляется по следующей формуле (6.11):

$$C_{рса} = \frac{C_p \cdot H_{рса}}{100}, \quad (6.11)$$

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на сопровождение и адаптацию программного средства:

$$C_{pca} = 5668,753 \cdot 10 / 100 = 566,88 \text{ (руб.)}.$$

Сумма расходов на сопровождение и адаптацию была вычислена на основе данных, рассчитанных ранее в данном разделе. Сумма расходов на сопровождение и адаптацию составила 566,88 рублей.

Все проведенные выше расчеты необходимы для вычисления полной себестоимости разработки веб-приложения.

### 6.3.11 Полная себестоимость

Общая сумма расходов (полная себестоимость)  $C_{\pi}$  определяется как сумма двух элементов: суммы расходов на разработку  $C_p$  и суммы расходов на сопровождение и адаптацию программного средства  $C_{pca}$  (формула 6.12).

$$C_{\pi} = C_p + C_{pca} \quad (6.12)$$

Все данные необходимые для вычисления у нас есть, поэтому можно определить общую сумму расходов:

$$C_{\pi} = 5668,753 + 566,88 = 6235,633 \text{ (руб.)}.$$

Таким образом, общая сумма расходов составляет 6235,633 рублей.

### 6.3.12 Определение цены, оценка эффективности

При рассмотрении аналогов было установлено, что данные продукты являются коммерческими и найти стоимость аналогичных продуктов найти сложно.

В своём представлении данный продукт представляет веб-приложение, поэтому в качестве среднее рыночной цены можно взять среднюю цену разработки веб-ресурсов при помощи калькулятора стоимости сайта.

Прибыль рассчитывается по формуле (6.13):

$$\Pi_{\pi c} = \frac{\Pi_p}{1.2} - C_{\pi} , \quad (6.13)$$

где  $\Pi_{\pi c}$  – прибыль от реализации программного средства, руб.;

$\Pi_p$  – средняя рыночная цена продукта, руб.;

$C_{\pi}$  – полная себестоимость программного средства, руб.

$$\Pi_{\pi c} = 13000 / 1,2 - 6235,633 = 4597,7 \text{ (руб.)}.$$

Уровень рентабельности разработанного программного средства определяется по формуле (6.14):

$$Y_p = \frac{\Pi_{\pi c}}{C_{\pi}} \cdot 100 , \quad (6.14)$$

где  $Y_{\text{рент}}$  – уровень рентабельности программного средства, %

$C_{\pi}$  – полная себестоимость программного средства, руб.;

$\Pi_{\text{пс}}$  – прибыль от реализации программного средства, руб.

$$Y_p = 4597,7/6235,633 \cdot 100 = 73,7 (\%).$$

Таким образом мы на практике смогли рассчитать прибыль от реализации программного средства и уровень рентабельности нашего программного продукта.

Прибыль от реализации программного средства составляет 4597,7 руб., а уровень рентабельности программного средства составляет 73,7%.

#### 6.4 Выводы по разделу

Разработка веб приложения служба доставки малогабаритных товаров, осуществляемая одним программистом на протяжении 30 рабочих дней, обойдется компании в 6235,633 рублей. В данную стоимость включена себестоимость разработки программного средства, а также расходы на сопровождение и адаптацию программного средства и накладные расходы.

Уровень рентабельности программного средства составит 73,7%, что является очень хорошим показателем и может говорить, что план по разработке программного средства экономически обоснован.

В таблице 6.4 и Приложении Е представлены результаты расчетов для основных показателей, посчитанных в данном разделе.

Таблица 6.4 – Результаты расчетов

Наименование показателя	Значение
Время разработки, мес.	1,36
Количество программистов, чел.	1,00
Зарплата с отчислениями, руб.	1642,09
Расходы на материалы, оплату машинного времени, прочие, руб.	629,023
Дополнительная заработная плата, руб.	490,1
Сумма расходов на материалы, руб.	46,713
Общие отчисления в фонд социальной защиты населения, руб.	1085,18
Сумма расходов на оплату машинного времени, руб.	91,4
Сумма прочих затрат, руб.	490,91
Накладные расходы, руб.	818,18
Себестоимость разработки программного средства, руб.	5668,753
Расходы на сопровождение и адаптацию, руб.	566,88
Полная себестоимость, руб.	6235,633
Цена аналога, руб.	13 000,00
Уровень рентабельности программного средства, %	73,7

Результаты проведенных расчетов показывают, что подобное веб-приложение для службы доставки малогабаритных товаров является рентабельным и экономически обоснованным программным продуктом, а отпускная цена, является приемлемой для потребителя, обеспечит конкурентоспособность на внутреннем рынке.

## 7 Анализ информационной безопасности проекта

Информационная безопасность – практика предотвращения несанкционированного доступа, использования, раскрытия, искажения, изменения, исследования, записи или уничтожения информации.

Анализ информационной безопасности (Information Security Analysis) – это процесс идентификации, оценки и управления уязвимостями и угрозами информационной системы или организации. Этот процесс позволяет определить риски и разработать стратегию защиты информации.

Основные этапы анализа информационной безопасности:

- идентификация угроз – этот этап включает оценку потенциальных и реальных угроз безопасности, которые могут навредить системе. Угрозы могут быть как внутренними, так и внешними (вирусы, хакеры и т.д.).

- Оценка рисков – этот этап идентифицирует уязвимости в системе и оценивает уровень рисков. Оценка рисков позволяет определить вероятность возникновения угрозы и потенциальный ущерб, который может быть нанесен системе.

- Разработка стратегии защиты – на этом этапе определяются механизмы, которые позволяют защитить систему или организацию от возможных угроз. Это может включать различные технические средства защиты, процедуры и правила безопасности, обучение персонала и т.д.

- Реализация защитных мер – на этом этапе реализуются механизмы защиты, определенные на предыдущем этапе. Реализация может включать установку программного и аппаратного обеспечения, настройку системы безопасности и т.д.

- Мониторинг и анализ – на этом этапе проводится постоянный мониторинг системы безопасности с целью выявления возможных угроз и уязвимостей в приложении. Также проводится анализ эффективности использования защитных мер и необходимость их дополнительной настройки.

Приложение, разработанное в рамках дипломного проекта, имеет две основные точки утечки информации – общедоступное API и база данных. И если с первым пунктом остается надеяться только на разработчиков, которые предусматривают меры по предотвращению посягательства посторонних лиц, то для базы данных есть определенные методы и настройки, позволяющие сохранить безопасность информации на должном уровне.

### 7.1 Обзор безопасности базы данных SQLite

База данных SQLite – это компактная, быстрая, удобная и открытая СУБД, которая используется самыми разными приложениями и системами. Обеспечение безопасности базы данных SQLite в рамках дипломного проекта выполнялось следующими способами:

					БГТУ 07.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Руденя П.А.			7 Анализ информационной безопасности проекта		Лист	Листов
Провер.		Берников В.О.					1	6
Н. контр.		Нистюк О.А.						
Утв.		Смелов В.В.					74217088, 2023	

- шифрование базы данных: SQLite поддерживает шифрование баз данных при помощи алгоритма AES-256. Шифрование позволяет защитить данные от несанкционированного доступа и перехвата данных в процессе их передачи преступниками.

- Использование пароля: Рекомендуется установить пароль на базу данных SQLite, чтобы ограничить доступ к ней. Можно установить такой пароль на всю базу данных SQLite или на конкретные таблицы в базе данных.

- Ограничение доступа: SQLite имеет встроенную поддержку прав и ролей пользователей. Это позволяет ограничить доступ к базе данных со стороны неквалифицированных пользователей и поддерживает управление правами доступа к базе данных.

- Использование PreparedStatement: PreparedStatement – это способ защиты более чувствительных данных от SQL-инъекций. Они защищают базу данных от ввода пользователей, содержащего вредоносный код.

- Резервное копирование БД: Регулярное резервное копирование базы данных SQLite является оптимальным методом защиты от случайного её повреждения, потери или удаления злоумышленниками.

- Обновление SQLite: Всегда следует использовать самую последнюю версию SQLite и его компонентов, которая исправляет обнаруженные ошибки безопасности и проводит другие улучшения.

- Аудит: Оцените производительность СУБД SQLite через регулярную проверку базы данных, чтобы убедиться, что настройки безопасности сохраняются и соответствуют требованиям безопасности.

## 7.2 Подтверждение номера телефона

Firebase предоставляет удобный и безопасный способ подтверждения номера телефона, используя Firebase Authentication. Для этого необходимо подключить библиотеку Firebase Auth SDK к проекту, а затем использовать методы и свойства, которые доступны в этой библиотеке.

Для подтверждения номера телефона можно использовать различные способы, в дипломном проекте используется отправка СМС. Кроме того, Firebase Authentication предоставляет возможность настройки множества параметров для более гибкой настройки процесса аутентификации, включая ограничение числа попыток подтверждения номера телефона, ограничение числа кодов, отправляемых на один номер и другие параметры.

Чтобы начать использовать Firebase для подтверждения номера телефона, нужно зарегистрировать приложение в Firebase Console и настроить проект. Затем нужно добавить Firebase Authentication в проект, чтобы получить доступ к методам подтверждения номера телефона. После того, как Firebase Authentication добавлен в ваш проект, вы можете использовать методы, такие как `verifyPhoneNumber`, чтобы отправить оповещение пользователю для подтверждения номера телефона. Когда пользователь вводит полученный код подтверждения, вы можете использовать метод `verifyPhoneNumberWithCode` для завершения процесса подтверждения.

Firebase Authentication также предоставляет возможность настройки текста СМС-сообщений, отправляемых для подтверждения номера телефона, а также управление устареванием отправленных кодов и другие функции для безопасности и настройки процесса аутентификации.

Перед отправкой кода для подтверждения номера телефона пользователь должен пройти защиту CAPTCHA. CAPTCHA – это тест на проверку, что пользователь является человеком, а не компьютерной программой или ботом. Обычно это набор символов или картинка, которую пользователь должен распознать и ввести. CAPTCHA используется для защиты от спама, взлома аккаунтов и других видов кибератак.

В целом, использование Firebase для подтверждения номера телефона является надежным и удобным способом, который может помочь обезопасить ваше приложение и защитить его пользователей от мошенничества и взлома.

### 7.3 Безопасность фреймворка Django

Django является одним из самых надежных фреймворков для создания веб-приложений. Django предлагает множество инструментов для обеспечения безопасности приложений, и разработчики должны использовать их все, чтобы создавать по-настоящему безопасные приложения.

Но важно понимать, что безопасность – это не только обязанность фреймворка, но и разработчика. Применение хороших практик безопасности в технологии и архитектуре приложений поможет уменьшить количество уязвимостей и сделать разрабатываемое приложение более надежным.

#### 7.3.1 Защита от CSRF в Django

Защита от CSRF (межсайтовой подделки запроса) в Django осуществляется с использованием токенов. Каждый раз, когда пользователь отправляет форму, создается токен с помощью тега `{% csrf_token %}`.

Этот токен добавляется в скрытое поле формы. Когда форма отправляется на сервер, Django проверяет токен на соответствие токenu, который был отправлен вместе с формой. Если токены не совпадают, то Django отклоняет запрос и отображает сообщение с указанием ошибки.

Для обеспечения защиты от CSRF [16] в фреймворке Django необходимо выполнить следующие шаги. Во-первых, необходимо включить Middleware для защиты от CSRF в настройках проекта и добавить `{% csrf_token %}` во все формы в нашем приложении. Данные действия представлены в листинге 7.1.

```
MIDDLEWARE =
[
    'django.middleware.csrf.CsrfViewMiddleware',
]
<form method="post">
    {% csrf_token %}
    <button type="submit">Отправить</button>
</form>
```

Листинг 7.1 – Защита от CSRF

В целом, использование токенов является эффективным способом защиты от CSRF в Django. Но необходимо помнить, что это не единственная мера безопасности, которую необходимо принять при создании веб-приложений.



Важно также следить за обновлениями программного обеспечения и использовать другие меры безопасности, такие как защита от XSS-атак, проверка на валидность данных и тестирование на проникновение.

### 7.3.2 Защита от XSS-атак в Django

XSS-атаки позволяют пользователю внедрять клиентские скрипты в браузеры других пользователей. Обычно это достигается путем хранения вредоносных сценариев в базе данных, откуда они будут извлечены и отображены для других пользователей, или путем побуждения пользователей щелкнуть ссылку, которая вызовет выполнение JavaScript злоумышленника в браузере пользователя.

Однако XSS-атаки [17] могут исходить из любого ненадежного источника данных, такого как файлы cookie или веб-службы, всякий раз, когда данные недостаточно очищены перед включением на страницу.

Использование шаблонов Django защищает от большинства XSS-атак. Однако важно понимать, какие средства защиты он обеспечивает и его ограничения.

Шаблоны Django избегают определенных символов, которые особенно опасны для HTML. Хотя это защищает пользователей от большинства злонамеренных действий, это не совсем надежно.

### 7.3.3 Защита от Clickjacking в Django

Clickjacking – это тип атаки, при которой вредоносный сайт помещает другой сайт во фрейм. Эта атака может привести к тому, что ничего не подозревающий пользователь будет обманом совершить непреднамеренные действия на целевом сайте.

Django содержит защиту ClickJacking в форме `X-Frame-Options` middleware, которая в поддерживающем браузере может предотвратить сайт от визуализации внутри фрейма. Можно отключить защиту для каждого просмотра или настроить точное значение отправляемого заголовка.

Промежуточное ПО настоятельно рекомендуется для любого сайта, для которого не требуется, чтобы его страницы были заключены во фрейм сторонними сайтами, или необходимо разрешить это только для небольшого раздела сайта.

### 7.3.4 Аутентификация и авторизация – Django

Django предоставляет функционал для создания из коробки системы аутентификации и авторизации пользователей. Система аутентификации Django позволяет пользователям регистрироваться, аутентифицироваться и выходить из системы. Это включает в себя механизмы проверки паролей, смены пароля и сброса пароля.

Django также содержит мощный механизм разграничения прав и доступов пользователей. Он позволяет создавать пользовательские группы и разрешения (полномочия) для управления доступом и правами пользователей в приложении.

Пользовательские группы могут быть настраиваемыми и содержать любое количество пользователей. Разрешения позволяют определить, какие разделы сайта могут быть просмотрены, отредактированы или удалены пользователями в зависимости, от полномочий которыми они обладают.

Django также позволяет настраивать электронную почту для уведомлений пользователей о событиях в приложении. Это может включать в себя уведомления о регистрации пользователя, сбросе пароля, изменении прав пользователей и других событиях происходящих в веб-приложении.

Django использует PBKDF2 (Password-Based Key Derivation Function 2) для хеширования паролей пользователей с солью. PBKDF2 – это функция, используемая для преобразования пароля в ключ шифрования симметричного алгоритма.

PBKDF2 при хешировании пароля, использует итеративный процесс хэширования, который повышает сложность обратного преобразования (восстановления) пароля из хеша и увеличивает время, необходимое для взлома хеша.

Еще один важный аспект безопасности PBKDF2 заключается в том, что он добавляет случайную соль к паролю перед хешированием, что обеспечивает дополнительный уровень безопасности.

Django позволяет настраивать параметры PBKDF2, такие как количество итераций, алгоритм хэширования, длина соли и другие параметры, для увеличения безопасности паролей пользователей. Кроме того, если используется библиотека `argon2_cffi`, Django может использовать специальный алгоритм хэширования паролей Argon2, который считается одним из наиболее надежных на сегодняшний день.

В целом, использование PBKDF2 в Django обеспечивает безопасное хеширование пользовательских паролей и увеличивает защиту приложения от взлома.

## **7.4 Безопасность использование платежной системы Stripe**

В дипломном проекте Stripe API имеет два ключа – Public Key и Secret Key.

Public Key используется для передачи публичной информации о учетной записи Stripe, такой как email, ваши доступные платежные методы и другие данные необходимые для выполнения транзакции. Этот ключ используется для улучшения пользовательского интерфейса, стилизации платежных форм и других функций, которые не требуют полномочий на обработку и изменение платежей.

Secret Key, с другой стороны, используется для авторизации платежей и доступа к транзакционной информации в Stripe. Это конфиденциальный ключ, который должен храниться в безопасности, так как его утечка может привести к нарушению безопасности и доступу к финансовой информации вашей учетной записи.

Secret Key используется для создания, подачи и обработки платежей, а также для доступа к другой конфиденциальной информации, такой как история транзакций и информация о клиентах.

Взаимодействие между ключами в Stripe API происходит следующим образом: сначала используется Public Key для конфигурирования и улучшения пользовательского интерфейса учетной записи, а затем используется Secret Key для обработки платежей и другой конфиденциальной информации.

Для взаимодействия с Stripe API необходимо использовать оба ключа. Public Key используется для передачи информации из приложения в Stripe.

Например, когда пользователь заполняет форму оплаты на сайте, Public Key используется для передачи информации о платеже в Stripe. Пользователь не имеет доступа к Secret Key, так как он не должен видеть или использовать этот конфиденциальный ключ для повышения безопасности.

Stripe API использует Secret Key для обработки и авторизации платежей. Приложение должно использовать Secret Key только для отправки запросов к Stripe, не передавая его дальше в приложение. Если Secret Key станет известен злоумышленникам, они могут использовать его для доступа к вашей учетной записи Stripe и осуществления незаконных операций.

В целом, Public Key и Secret Key – это обязательные компоненты Stripe API, которые взаимодействуют друг с другом, чтобы обеспечить безопасное и защищенное взаимодействие между вашим приложением и Stripe.

## 7.5 Выводы по разделу

В данном разделе были рассмотрены уязвимые точки приложения, разработанного в рамках дипломного проекта, а также способы их решения.

Безопасность информации является одним из важнейших аспектов в современном мире информационных технологий, поэтому была разработана политика безопасности для системы управления базами данных SQLite, которая защищает от несанкционированного доступа к информации.

В результате анализа уязвимых точек приложения было выявлено, что все действия, связанные с чтением и записью данных, требуют минимальной степени авторизации, для того чтобы повысить безопасность.

Кроме того, вся информация полностью контролируется СУБД, что гарантирует наличие высокой степени безопасности в приложении.

Таким образом, можно заключить, что информационная безопасность в приложении, разработанном в рамках дипломного проекта, находится на должном уровне и обеспечена за счет эффективной политики безопасности, реализованной в систему управления базами данных SQLite.

Также была рассмотрена безопасность при работе с платежной системой Stripe API. Был проведен анализ и полный разбор взаимодействия между собой Public Key и Secret Key. По итогу можно сделать вывод, что система оплаты на сайте организована максимально безопасно и данные пользователей являются максимально защищенными от несанкционированного доступа.

Кроме того, были рассмотрены возможности Django для создания безопасных систем аутентификации и авторизации пользователей. Django представляет собой мощный и гибкий фреймворк, который позволяет создавать безопасные и простые в использовании системы авторизации и аутентификации для пользователей.

## Заключение

В результате работы над дипломным проектом спроектировано и разработано веб-приложение для доставки малогабаритных товаров.

Обзор аналогов показал слабые и сильные стороны похожих приложений, что позволило учесть это при разработке.

Проведён аналитический обзор литературы и ПС, который показал средства и технологии, которые были использованы для решения поставленной задачи.

В ходе выполнения данного дипломного проекта были выполнены все поставленные задачи, такие как:

- обеспечить возможность регистрации и авторизации;
- создание заказов, ввод информации о них, указание точек доставки;
- вычисление суммы заказа в зависимости от расстояния между выбранными точками получения и доставки товара;
- просмотр свободных заказов на карте для курьера;
- отображение местоположения курьера на карте;
- получение фотографий от курьера в момент приема и доставки товара.
- проанализированы требования и информационные источники по теме данного дипломного проекта;
- разработаны функциональные требования и архитектура веб-приложения;
- составлено руководство пользователя;
- протестировано приложение.

Программное средство создано на основе разработанной базы данных и содержащейся в ней 5 таблиц, хранящих всю информацию о пользователях

В приложении присутствует разделение ролей, есть пользователи, курьеры и администраторы. Для каждого имеются свои возможности.

Авторизация производится при помощи модели UserCreationForm. Она представляет собой набор полей, которые могут использоваться при разработке страницы авторизации. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях.

Для создания программного средства использовались технологии, такие как: Python, JavaScript, Django, Bootstrap и Jinja2.

В качестве СУБД использовалась SQLite.

Таким образом, при создании программного средства были использованы все указанные технологии, выполнены все функциональные требования.

					БГТУ 00.00.ПЗ			
		ФИО	Подпись	Дата				
Разраб.		Руденя П.А.			Заключение		Лист	Листов
Провер.		Берников В.О.					1	1
Н. контр.		Нистюк О.А.						
Утв.		Смелов В.В.				74217088, 2023		

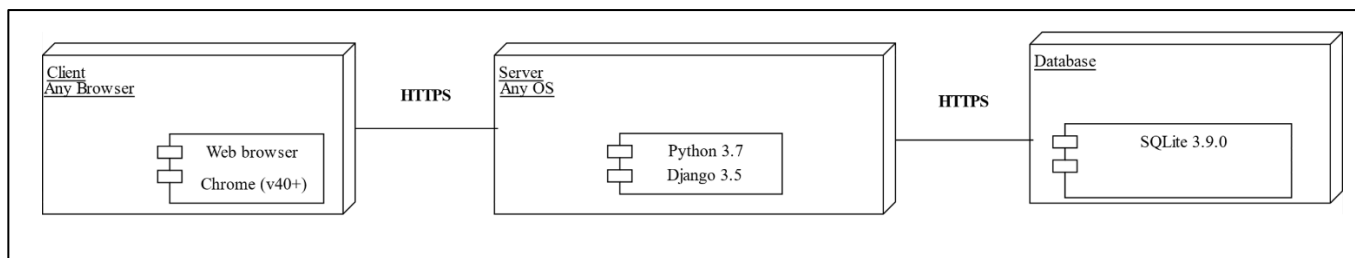
## Список использованных источников

- 1 Краткий обзор языка Python [Электронный ресурс] / [wiki.python.org](https://wiki.python.org/moin/BeginnersGuide) – 2023. – Режим доступа: <https://wiki.python.org/moin/BeginnersGuide> – Дата доступа: 14.04.2023.
- 2 Современный Javascript [Электронный ресурс] / [learn.javascript.ru](https://learn.javascript.ru/manuals-specifications) – 2023. – Режим доступа: <https://learn.javascript.ru/manuals-specifications> – Дата доступа: 14.04.2023.
- 3 Django [Электронный ресурс] / <https://www.djangoproject.com/> – 2023. – Режим доступа: <https://docs.djangoproject.com/en/4.2/> – Дата доступа: 14.04.2023.
- 4 SQLite [Электронный ресурс] / <https://www.sqlite.org/index.html> – 2023. – Режим доступа: <https://www.sqlite.org/docs.html> – Дата доступа: 14.04.2023.
- 5 Многоуровневая архитектура [Электронный ресурс] / [metanit.com](https://metanit.com/sharp/mvc5/23.5.php). – Режим доступа. – <https://metanit.com/sharp/mvc5/23.5.php>. – Дата доступа: 25.04.2023.
- 6 Введение в REST API [Электронный ресурс] / [habr.com](https://habr.com/ru/post/483202/). – Режим доступа. – <https://habr.com/ru/post/483202/> – Дата доступа: 20.04.2023.
- 7 Введение в web APIs [Электронный ресурс] / [https://developer.mozilla.org](https://developer.mozilla.org/ru/docs/Learn). – Режим доступа. – <https://developer.mozilla.org/ru/docs/Learn> – Дата доступа: 25.04.2023.
- 8 Все о Google картах [Электронный ресурс] / [https://developers.google.com](https://developers.google.com/maps?hl=ru/). – Режим доступа. – <https://developers.google.com/maps?hl=ru/> – Дата доступа: 28.04.2023.
- 9 Stripe API [Электронный ресурс] / [https://api.stripe.com](https://api.stripe.com/docs/api) – Режим доступа. – <https://stripe.com/docs/api> – Дата доступа: 02.05.2023.
- 10 PayPal API [Электронный ресурс] / [https://developer.paypal.com](https://developer.paypal.com/api/rest/) – Режим доступа. – <https://developer.paypal.com/api/rest/> – Дата доступа: 02.05.2023.
- 11 Documentation Maps JavaScript API [Электронный ресурс] / [https://developers.google.com](https://developers.google.com/maps/documentation/). – Режим доступа. – <https://developers.google.com/maps/documentation/> – Дата доступа: 10.05.2023.
- 12 WebSocket [Электронный ресурс] / [https://learn.javascript.ru](https://learn.javascript.ru/websocket). – Режим доступа. – <https://learn.javascript.ru/websocket> – Дата доступа: 14.05.2023.
- 13 Обзор прогрессивных веб-приложений [Электронный ресурс] / [https://learn.microsoft.com](https://learn.microsoft.com/ru-ru/microsoft-edge/progressive-web-apps-chromium/). – Режим доступа. – <https://learn.microsoft.com/ru-ru/microsoft-edge/progressive-web-apps-chromium/> – Дата доступа: 16.05.2023.
- 14 Начните работу с Bootstrap [Электронный ресурс] / [https://bootstrap-4.ru](https://bootstrap-4.ru/docs/5.3/) – Режим доступа. – <https://bootstrap-4.ru/docs/5.3/> – Дата доступа: 18.05.2023.
- 15 ORM [Электронный ресурс] / [https://ru.wikipedia.org](https://ru.wikipedia.org/wiki/ORM) – Режим доступа. – <https://ru.wikipedia.org/wiki/ORM> – Дата доступа: 20.05.2023.
- 16 Защита от межсайтовых запросов [Электронный ресурс] / [https://djangodoc.ru](https://djangodoc.ru/3.2/ref/csrf/) – Режим доступа. – <https://djangodoc.ru/3.2/ref/csrf/> Дата доступа: 21.05.2023.
- 17 XSS [Электронный ресурс] / [https://blog.skillfactory.ru](https://blog.skillfactory.ru/glossary/xss/) – Режим доступа. – <https://blog.skillfactory.ru/glossary/xss/> Дата доступа: 22.05.2023.

					<i>БГТУ 00.00.ПЗ</i>			
		ФИО	Подпись	Дата				
Разраб.		<i>Руденя П.А.</i>			<i>Список использованных источников</i>		Лист	Листов
Провер.		<i>Берников В.О.</i>					1	1
						<i>74217088, 2023</i>		
Н. контр.		<i>Нистюк О.А.</i>						
УТВ.		<i>Смелов В.В.</i>						

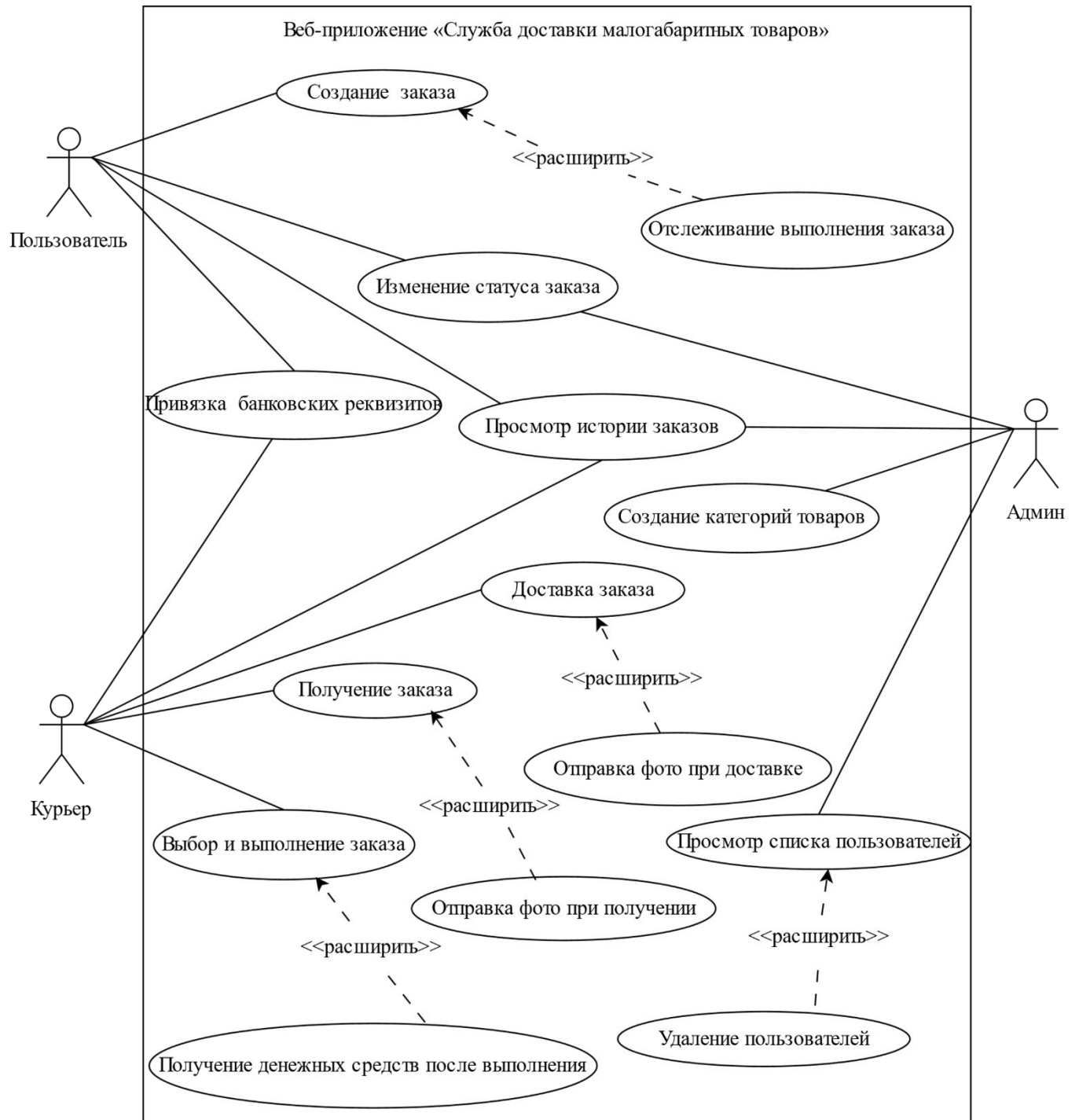
## ПРИЛОЖЕНИЕ А

### Диаграмма архитектуры приложения



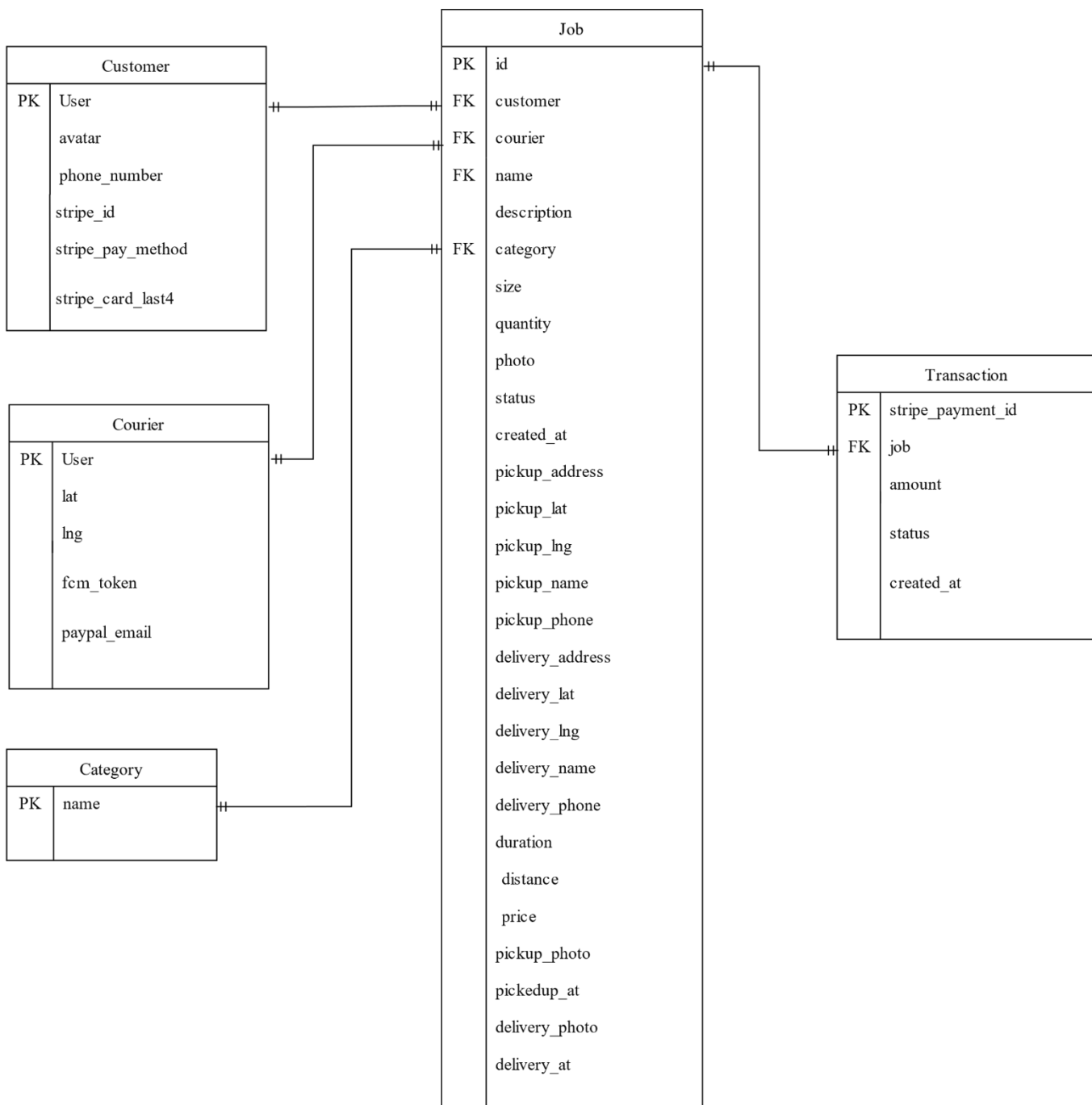
## ПРИЛОЖЕНИЕ Б

### Диаграмма вариантов использования приложения



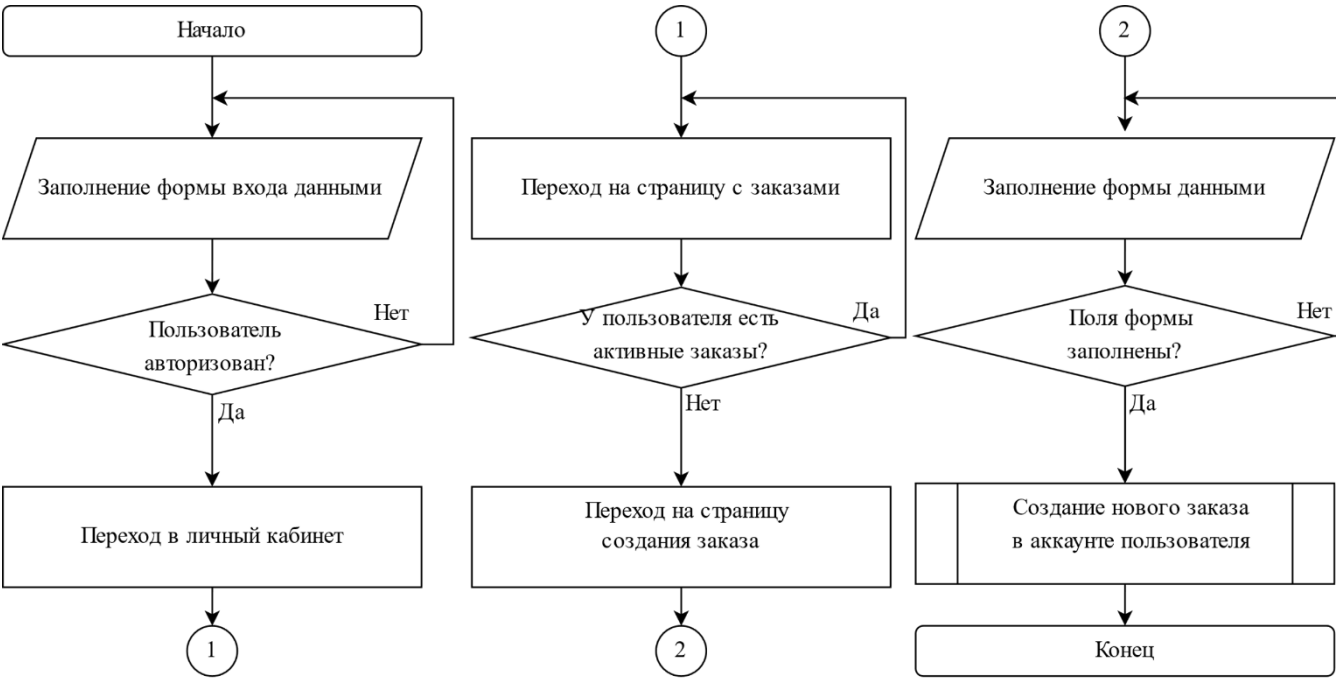
## ПРИЛОЖЕНИЕ В

### Логическая схема базы данных



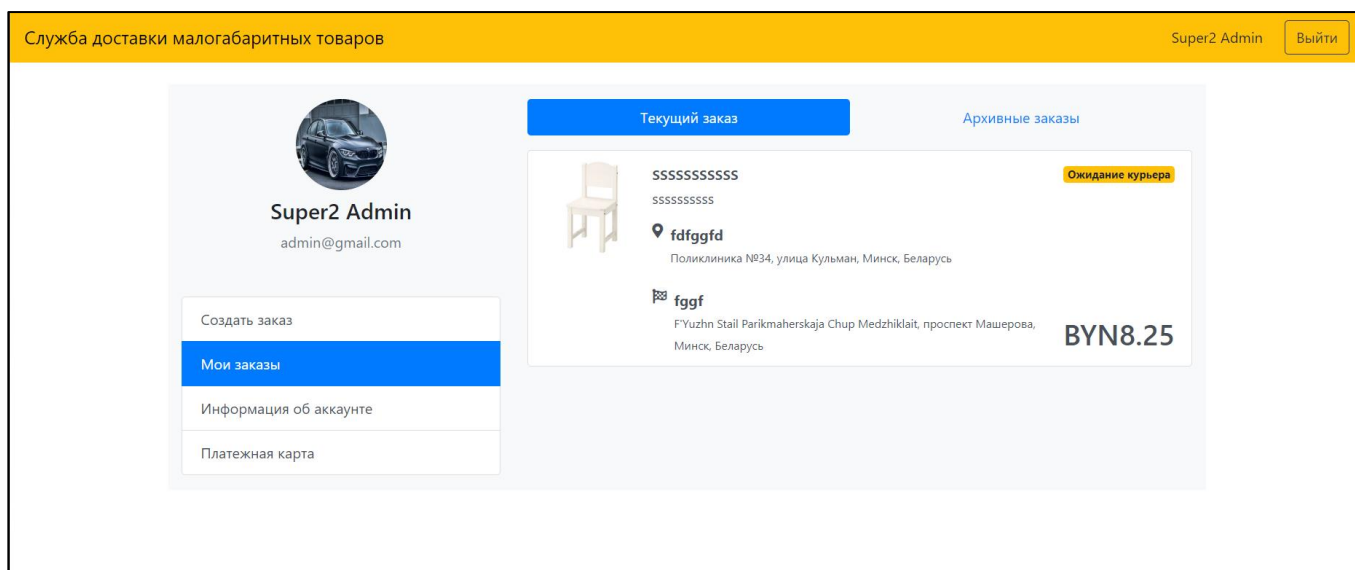


ПРИЛОЖЕНИЕ Г  
Блок-схема создания заказа



## ПРИЛОЖЕНИЕ Д

### Скриншот работы приложения



## ПРИЛОЖЕНИЕ Е

## Результаты расчётов технико-экономического обоснования проекта

Наименование показателя	Значение
Время разработки, мес.	1,36
Количество программистов, чел.	1,00
Зарплата с отчислениями, руб.	1642,09
Расходы на материалы, оплату машинного времени, прочие, руб.	629,023
Дополнительная заработная плата, руб.	490,1
Сумма расходов на материалы, руб.	46,713
Общие отчисления в фонд социальной защиты населения, руб.	1085,18
Сумма расходов на оплату машинного времени, руб.	91,4
Сумма прочих затрат, руб.	490,91
Накладные расходы, руб.	818,18
Себестоимость разработки программного средства, руб.	5668,753
Расходы на сопровождение и адаптацию, руб.	566,88
Полная себестоимость, руб.	6235,633
Цена аналога, руб.	13 000,00
Уровень рентабельности программного средства, %	73,7

## ПРИЛОЖЕНИЕ Ж

### Листинг исходного кода

#### Urls.py

```

from django.contrib import admin
from django.urls import path, include
from django.contrib.auth import views as auth_views
from django.conf import settings
from django.conf.urls.static import static
from django.views.generic import TemplateView
from core import views, consumers
from core.customer import views as customer_views
from core.courier import views as courier_views, apis as courier_apis

customer_urlpatterns = [
    path('', customer_views.home, name="home"),
    path('profile/', customer_views.profile_page, name="profile"),
    path('payment_method/', customer_views.payment_method_page,
name="payment_method"),
    path('create_job/', customer_views.create_job_page,
name="create_job"),

    path('jobs/current/', customer_views.current_jobs_page,
name="current_jobs"),
    path('jobs/archived/', customer_views.archived_jobs_page,
name="archived_jobs"),
    path('jobs/<job_id>', customer_views.job_page, name="job"),
]
courier_urlpatterns = [
    path('', courier_views.home, name="home"),
    path('jobs/available/', courier_views.available_jobs_page,
name="available_jobs"),
    path('jobs/available/<id>', courier_views.available_job_page,
name="available_job"),
    path('jobs/current/', courier_views.current_job_page,
name="current_job"),
    path('jobs/current/<id>/take_photo/',
courier_views.current_job_take_photo_page,
name="current_job_take_photo"),
    path('jobs/complete/', courier_views.job_complete_page,
name="job_complete"),
    path('jobs/archived/', courier_views.archived_jobs_page,
name="archived_jobs"),
    path('profile/', courier_views.profile_page, name="profile"),
    path('payout_method/', courier_views.payout_method_page,
name="payout_method"),
    path('api/jobs/available/', courier_apis.available_jobs_api,
name="available_jobs_api"),

```

```

    path('api/jobs/current/<id>/update/',
courier_apis.current_job_update_api,
name="current_job_update_api"),
    path('api/fcm-token/update/',
courier_apis.fcm_token_update_api, name="fcm_token_update_api"),
]
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('social_django.urls', namespace='social')),
    path('', views.home),
    path('sign-in/',
auth_views.LoginView.as_view(template_name="sign_in.html")),
    path('sign-out/',
auth_views.LogoutView.as_view(next_page="/")),
    path('sign-up/', views.sign_up),

    path('customer/', include((customer_urlpatters, 'customer'))),
    path('courier/', include((courier_urlpatters, 'courier'))),
    path('firebase-messaging-sw.js',
        (TemplateView.as_view(template_name="firebase-messaging-
sw.js", content_type="application/javascript", ))),
]
websocket_urlpatterns = [
    path('ws/jobs/<job_id>/', consumers.JobConsumer.as_asgi())
]
if settings.DEBUG:
    urlpatterns
static(settings.MEDIA_URL,document root=settings.MEDIA_ROOT)

```

+=

## Models.py

```

from django.db import models
from django.contrib.auth.models import User
import uuid
from django.utils import timezone
class Customer(models.Model):
    user = models.OneToOneField(User,on_delete=models.CASCADE)
    avatar = models.ImageField(upload_to='customer/ava-
tars/',blank=True,null=True)
    phone_number = models.CharField(max_length=50,blank =True,ver-
bose_name='Номер телефона',help_text="Номер телефона")
    stripe_customer_id = models.CharField(max_length =
255,blank=True)
    stripe_payment_method_id = models.CharField(max_length =
255,blank=True)
    stripe_card_last4 = models.CharField(max_length =
255,blank=True)
    def __str__(self):
        return self.user.get_full_name()
class Courier(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)

```

```

lat = models.FloatField(default=0)
lng = models.FloatField(default=0)
paypal_email = models.EmailField(max_length=255, blank=True)
fcm_token = models.TextField(blank=True)

def __str__(self):
    return self.user.get_full_name()
class Category(models.Model):
    slug = models.CharField(max_length=225, unique=True)
    name = models.CharField(max_length=225)
    def __str__(self):
        return self.name
class Job(models.Model):
    SMALL_SIZE = "Размер упаковки не превышает 15x15x15см, вес не более 5 кг."
    MEDIUM_SIZE = "Размер упаковки не превышает 40x40x26см, вес не более 15 кг."
    LARGE_SIZE = "Размер упаковки не превышает 60x60x36см, вес не более 25 кг."
    SIZES = (
        (SMALL_SIZE, 'Размер упаковки не превышает 15x15x15см, вес не более 5 кг.'),
        (MEDIUM_SIZE, 'Размер упаковки не превышает 40x40x26см, вес не более 15 кг.'),
        (LARGE_SIZE, 'Размер упаковки не превышает 60x60x36см, вес не более 25 кг.'),
    )
    CREATING_STATUS = 'creating'
    PROCESSING_STATUS = 'processing'
    PICKING_STATUS = 'picking'
    DELIVERING_STATUS = 'delivering'
    COMPLETED_STATUS = 'completed'
    CANCELED_STATUS = 'canceled'
    STATUSES = (
        (CREATING_STATUS, 'Создана'),
        (PROCESSING_STATUS, 'Ожидание курьера'),
        (PICKING_STATUS, 'Курьер выполняет'),
        (DELIVERING_STATUS, 'Доставляется'),
        (COMPLETED_STATUS, 'Выполнена'),
        (CANCELED_STATUS, 'Удаленна'),
    )
    # Step 1
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE, verbose_name='Пользователь', help_text="Пользователь")
    courier = models.ForeignKey(Courier, on_delete=models.CASCADE, null=True, blank=True, verbose_name='Курьер', help_text="Курьер")
    name = models.CharField(max_length=255, verbose_name='Название', help_text="Название")

```

```

description      =      models.CharField(max_length=255,verbose_name
='Описание',help_text="Описание")
category = models.ForeignKey(Category, on_delete=models.SET_NULL,
null=True,                                blank=True,verbose_name
='Категория',help_text="Категория")
size      =      models.CharField(max_length=70,choices=SIZES,      de-
fault=SMALL_SIZE,verbose_name ='Размер',help_text="Размер")
quantity      =      models.IntegerField(default=1,verbose_name
='Количество',help_text="Количество")
photo      =      models.ImageField(upload_to='job/photos/',verbose_name
='Фото товара',help_text="Фото товара")
status      =      models.CharField(max_length=20, choices=STATUSES, de-
fault=CREATING_STATUS,verbose_name ='Статус',help_text="Статус")
created_at      =      models.DateTimeField(default=timezone.now,ver-
bose_name ='Дата создания',help_text="Дата создания")
# Step 2
pickup_address      =      models.CharField(max_length=255,verbose_name
='Адрес получения посылки',help_text="Адрес получения посылки",
blank=True)
pickup_lat      =      models.FloatField(default=0,verbose_name
='Широта',help_text="Широта")
pickup_lng      =      models.FloatField(default=0,verbose_name
='Долгота',help_text="Долгота")
pickup_name      =      models.CharField(max_length=255,      verbose_name
='Дополнительные      данные',help_text="Дополнительные      данные",
blank=True)
pickup_phone      =      models.CharField(max_length=50,      verbose_name
='Номер для связи',help_text="Номер для связи", blank=True)
# Step 3
delivery_address      =      models.CharField(max_length=255,
blank=True,verbose_name ='Адрес доставки посылки',help_text="Адрес
доставки посылки")
delivery_lat      =      models.FloatField(default=0,verbose_name
='Широта',help_text="Широта")
delivery_lng      =      models.FloatField(default=0,verbose_name
='Долгота',help_text="Долгота")
delivery_name      =      models.CharField(max_length=255, blank=True,ver-
bose_name      ='Дополнительные      данные',help_text="Дополнительные
данные")
delivery_phone      =      models.CharField(max_length=50, blank=True,ver-
bose_name ='Номер для связи',help_text="Номер для связи")

# Step 4
duration      =      models.IntegerField(default=0,verbose_name
='Продолжительность',help_text="Продолжительность")
distance      =      models.FloatField(default=0,verbose_name
='Расстояние',help_text="Расстояние")
price      =      models.FloatField(default=0,verbose_name
='Стоимость',help_text="Стоимость")
# Extra info

```

```

    pickup_photo = models.ImageField(upload_to='job/pickup_photos/',
null=True,      blank=True,verbose_name      ='Фото      при      получении
заказа',help_text="Фото при получении заказа")
    pickedup_at = models.DateTimeField(null=True,      blank=True,ver-
bose_name ='Заказ взят в ',help_text="Заказ взят в")
    delivery_photo = models.ImageField(upload_to='job/delivery_pho-
tos/',      null=True,      blank=True,verbose_name      ='Фото      при      доставке
заказа',help_text="Фото при доставке заказа")
    delivered_at = models.DateTimeField(null=True,      blank=True,ver-
bose_name ='Заказ доставлен в ',help_text="Заказ доставлен в")
    def __str__(self):
        return self.name
class Transaction(models.Model):
    IN_STATUS = "in"
    OUT_STATUS = "out"
    STATUSES = (
        (IN_STATUS, 'In'),
        (OUT_STATUS, 'Out'),
    )
    stripe_payment_intent_id      =      models.CharField(max_length=255,
unique=True)
    job = models.ForeignKey(Job, on_delete=models.CASCADE)
    amount = models.FloatField(default=0)
    status = models.CharField(max_length=20, choices=STATUSES, de-
fault=IN_STATUS)
    created_at = models.DateTimeField(default=timezone.now)
    def __str__(self):
        return self.stripe payment intent id

```

## Create\_job.html

```

{% extends 'customer/base.html' %}
{% load bootstrap4 %}
    {% block head %}
        <script
src="https://maps.googleapis.com/maps/api/js?key={{GOOGLE_MAP_API_K
EY}}&callback=initMap&libraries=places&v=weekly"
        defer></script>
    <style>
#pills-tab a {
    color: black;
}
#pills-tab a:hover {
    color: orange;
    text-decoration: none;
}
#pills-tab a.active {
    color: orange;
}
#pickup-map,
#delivery-map {

```



```

    height: 100%;
}
</style>
{% endblock %}
{% block content %}
<div class="container mt-4">
    <div class="row">
        <div class="col-lg-4">
            <div class="card">
                <div class="card-header">
                    КРАТКОЕ ОПИСАНИЕ ЗАКАЗА
                </div>
                <div class="card-body">
                    {% if not job %}
                        <p>Здесь появится краткая информация о вашем
заказе</p>
                    {% else %}
                        {% if step > 1 %}
                            <h4>{{ job.name }}</h4>
                            <span>{{job.quantity}} Предмет</span><br>
                            <span>{{job.get_size_display}} Заказ</span>
                        {% endif %}
                        {% if step > 2 %}
                            <hr>
                            <p
                                class="text-secondary"><small><b>Место
получения заказа</b></small></p>
                            <h4>{{ job.pickup_name }}</h4>
                            <span>{{job.pickup_address}}</span><br>
                        {% endif %}
                        {% if step > 3 %}
                            <hr>
                            <p
                                class="text-secondary"><small><b>Место
доставки товара</b></small></p>
                            <h4>{{ job.delivery_name }}</h4>
                            <span>{{job.delivery_address}}</span><br>
                        {% endif %}
                    {% endif %}
                </div>
            </div>
        </div>
        <!-- Right SIDE -->
        <div class="col-lg-8">
            <!-- Step tabs -->
            <div class="card mb-5">
                <div class="card-body">
                    <ul class="nav nav-pills nav-justified align-
items-center mb-3" id="pills-tab" role="tablist">
                        <li class="nav-item">
                            <a class="{% if step == 1 %}active{%
endif %}" id="pills-info-tab" data-toggle="pill" href="#pills-info"

```

```

role="tab"                aria-controls="pills-info"                aria-
selected="true">Информация о заказе</a>
                                </li>
                                <i class="fas fa-chevron-right"></i>
                                <li class="nav-item">
                                    <a class="{% if step == 2 %}active{%
endif %}" id="pills-pickup-tab" data-toggle="pill" href="#pills-
pickup" role="tab" aria-controls="pills-pickup" aria-
selected="false">Получение</a>
                                </li>
                                <i class="fas fa-chevron-right"></i>
                                <li class="nav-item">
                                    <a class="{% if step == 3 %}active{%
endif %}" id="pills-delivery-tab" data-toggle="pill" href="#pills-
delivery" role="tab" aria-controls="pills-delivery" aria-
selected="false">Доставка</a>
                                </li>
                                <i class="fas fa-chevron-right"></i>
                                <li class="nav-item">
                                    <a class="{% if step == 4 %}active{%
endif %}" id="pills-payment-tab" data-toggle="pill" href="#pills-
payment" role="tab" aria-controls="pills-payment" aria-
selected="false">Оплата</a>
                                </li>
                                <i class="fas fa-chevron-right"></i>
                            </ul>
                        </div>
                    </div>
                <!-- Step forms -->
                <b>Создать заказ</b>
                <div class="tab-content" id="pills-tabContent">
                    <!-- Step 1 -->
                    <div class="tab-pane fade {%if step == 1 %}show
active{% endif %}" id="pills-info" role="tabpanel" aria-
labelledby="pills-info-tab">
                        <h1>Информация о заказе</h1>
                        <form method="POST" enctype="multipart/form-
data">
                            <b class="text-secondary">Информация о
товаре</b><br>
                            <div class="card bg-white mt-2 mb-5">
                                <div class="card-body">
                                    {% csrf_token %}
                                    {% bootstrap_form step1_form %}
                                </div>
                            </div>
                            <input type="hidden" name="step" value="1">
                            <button type="submit" class="btn btn-
warning">Сохранить и продолжить</button>
                        </form>
                    </div>
                </div>

```

```

        <!-- Step 2 -->
        <div class="tab-pane fade {%if step == 2 %}show
active{% endif %}" id="pills-pickup" role="tabpanel" aria-
labelledby="pills-pickup-tab">
            <h1>Получение</h1>
            <form method="POST" enctype="multipart/form-
data">
                <b class="text-secondary">Информация о месте
получения заказа</b><br>
                <div class="card bg-white mt-2 mb-5">
                    <div class="card-body">
                        <div class="row">
                            <div class="col-lg-8">
                                {% csrf_token %}
                                {% bootstrap_form step2_form
exclude='pickup_lat,pickup_lng' %}
                                <input hidden id="pickup_lat"
name="pickup_lat" value="{{ job.pickup_lat }}">
                                <input hidden
id="pickup_lng" name="pickup_lng" value="{{ job.pickup_lng }}">
                            </div>
                            <div class="col-lg-4">
                                <div id="pickup-map"></div>
                                <div id="pickup-infowindow-
content">
                                    <span id="pickup-place-
name" class="title"></span><br />
                                    <span id="pickup-place-
address"></span>
                                </div>
                            </div>
                        </div>
                        <input type="hidden" name="step" value="2">
                        <button type="button" class="btn btn-outline-
warning" onclick="$('#pills-info-tab').tab('show')">Назад</button>
                        <button type="submit" class="btn btn-
warning">Сохранить и продолжить</button>
                    </form>
                </div>
            <!-- Step 3 -->
            <div class="tab-pane fade {%if step == 3 %}show
active{% endif %}" id="pills-delivery" role="tabpanel" aria-
labelledby="pills-delivery-tab">
                <h1>Delivery</h1>
                <form method="POST" enctype="multipart/form-
data">
                    <b class="text-secondary">Информация о месте
доставки товара</b><br>
                    <div class="card bg-white mt-2 mb-5">

```

```

        <div class="card-body">
            <div class="row">
                <div class="col-lg-8">
                    {% csrf_token %}
                    {% bootstrap_form step3_form
exclude='delivery_lat,delivery_lng' %}
                    <input                                hidden
id="delivery_lat"  name="delivery_lat"  value="{{ job.delivery_lat
}}">
                    <input                                hidden
id="delivery_lng"  name="delivery_lng"  value="{{ job.delivery_lng
}}">
                </div>
                <div class="col-lg-4">
                    <div id="delivery-map"></div>
                    <div                                id="delivery-
infowindow-content">
                        <span id="delivery-place-
name" class="title"></span><br />
                        <span id="delivery-place-
address"></span>
                    </div>
                </div>
            </div>
        </div>
        <input type="hidden" name="step" value="3">
        <button type="button" class="btn btn-outline-
warning" onclick="$('#pills-info-tab').tab('show')">Назад</button>
        <button type="submit" class="btn btn-
warning">Сохранить и продолжить</button>
    </form>
</div>
    <!-- Step 4 -->
    <div class="tab-pane fade {%if step == 4 %}show
active{% endif %}" id="pills-payment" role="tabpanel" aria-
labelledby="pills-payment-tab">
        <h1>Оплата</h1>
<form method="POST">
    <b class="text-secondary">Способ оплаты</b>
    <div class="card bg-white mt-2 mb-5">
        <div class="card-body">
            {% csrf_token %}
            <div class="form-group">
                <label>Ваша кредитная/дебетовая карта</label>
                <input class="form-control" value="**** *
{{ request.user.customer.stripe_card_last4 }}"
disabled>
            </div>
            <div class="form-group">
                <label>Стоимость</label>

```

```

        <input class="form-control" value="BYN" {{
job.price }}" disabled>
    </div>
</div>
<input type="hidden" name="step" value="4">
<button type="button" class="btn btn-outline-warning"
onclick="$('#pills-delivery-
tab').tab('show');">Назад</button>
<button type="submit" class="btn btn-warning">Создать
заказ</button>
</form>
</div>

</div>
</div>
</div>
<script>
var pickupLat = parseFloat('{{ job.pickup_lat }}');
var pickupLng = parseFloat('{{ job.pickup_lng }}');
var deliveryLat = parseFloat('{{ job.delivery_lat }}');
var deliveryLng = parseFloat('{{ job.delivery_lng }}');
function initMapByType(type, initLat, initLng) {
    const map = new google.maps.Map(document.getElementById(type +
"-map"), {
        center: { lat: initLat || 53.9, lng: initLng || 27.5656 },
        zoom: 13,
    });
    if (initLat && initLng) {
        new google.maps.Marker({
            position: new google.maps.LatLng(initLat, initLng),
            map: map,
        })
    }
    const input = document.getElementById("id_" + type +
"_address");
    const autocomplete = new google.maps.places.Autocomplete(input);
    that the autocomplete requests use the current map bounds for the
    // bounds option in the request.
    autocomplete.bindTo("bounds", map);
    // Set the data fields to return when the user selects a place.
    autocomplete.setFields(["address_components", "geometry",
"icon", "name"]);
    const infowindow = new google.maps.InfoWindow();
    const infowindowContent = document.getElementById(type + "-
infowindow-content");
    infowindow.setContent(infowindowContent);
    const marker = new google.maps.Marker({
        map,
        anchorPoint: new google.maps.Point(0, -29),
    });

```

```

autocomplete.addListener("place_changed", () => {
  infowindow.close();
  marker.setVisible(false);
  const place = autocomplete.getPlace();
  if (!place.geometry) {
    window.alert("No details available for input: '" +
place.name + "'");
    return;
  }
  // If the place has a geometry, then present it on a map.
  if (place.geometry.viewport) {
    map.fitBounds(place.geometry.viewport);
  } else {
    map.setCenter(place.geometry.location);
    map.setZoom(17); // Why 17? Because it looks good.
  }
  marker.setPosition(place.geometry.location);
  marker.setVisible(true);
  $("#" + type + "_lat").val(place.geometry.location.lat());
  $("#" + type + "_lng").val(place.geometry.location.lng());
  let address = "";
  if (place.address_components) {
    address = [
      (place.address_components[0] &&
        place.address_components[0].short_name) ||
      "",
      (place.address_components[1] &&
        place.address_components[1].short_name) ||
      "",
      (place.address_components[2] &&
        place.address_components[2].short_name) ||
      "",
    ].join(" ");
  }
  infowindowContent.children[type + "-place-icon"].src =
place.icon;
  infowindowContent.children[type + "-place-name"].textContent
= place.name;
  infowindowContent.children[type + "-place-
address"].textContent = address;
  infowindow.open(map, marker);
});}

function initMap() {
  initMapByType("pickup", pickupLat, pickupLng);
  initMapByType("delivery", deliveryLat, deliveryLng);}
</script>
{% endblock %}

```