



**МИНОБРНАУКИ РОССИИ**  
**федеральное государственное бюджетное образовательное учреждение**  
**высшего образования**  
**«Московский государственный технологический университет «СТАНКИН»**  
**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

---

**Институт**  
информационных технологий

**Кафедра**  
информационных систем

Проект по дисциплине «Управление данными»  
на тему: «Проектирование БД курьерской службы»

**Студент**  
группа ИДБ-22-06

**Рамазанов М. А.**

**Руководитель**  
старший преподаватель

\_\_\_\_\_

подпись

**Быстрикова В. А.**

\_\_\_\_\_

подпись

Москва 2024 г.

**ОГЛАВЛЕНИЕ**

ВВЕДЕНИЕ .....	3
ГЛАВА 1. ПРОЕКТИРОВАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ .....	4
1.1. АНАЛИЗ ПРЕМЕТНОЙ ОБЛАСТИ .....	4
1.2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ .....	12
1.3. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ .....	16
1.4. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ .....	21
ГЛАВА 2. ОПИСАНИЕ ПРОГРАММНОГО ИНТЕРФЕЙСА К БД.....	28
2.1. ЗАДАЧИ ИНТЕРФЕЙСА И ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ .....	28
2.2. ОПИСАНИЕ РАБОТЫ ИНТЕРФЕЙСА .....	30
ЗАКЛЮЧЕНИЕ .....	57
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ .....	59
ПРИЛОЖЕНИЕ А SQL-ОПЕРАТОРЫ СОЗДАНИЯ ОБЪЕКТОВ БД .....	60
ПРИЛОЖЕНИЕ Б ЗАПОЛНЕНИЕ ТАБЛИЦ ДАННЫМИ .....	65
ПРИЛОЖЕНИЕ В КОД ПРОГРАММЫ .....	67

## **ВВЕДЕНИЕ**

Тема исследования данного проекта направлена на оптимизацию процессов управления курьерской службой с помощью современных технологий. Для эффективного управления необходимо оперировать достаточно большими объемами информации, что без использования информационных технологий вызывает достаточно много проблем.

Кроме того, интеграция информационных технологий в сферу доставки позволяет улучшить качество обслуживания, что влияет на конкурентоспособность компании на рынке.

Основной целью данного проекта является создание и внедрение базы данных, способной эффективно управлять информацией о клиентах и их заказах, курьерах, а также процессами доставки.

Для выполнения данного проекта необходимо будет выполнить следующие задачи:

1. Проектирование базу данных с учетом предметной области.
  - Провести исследование предметной области.
  - Выполнить концептуальное проектирование.
  - Провести логическое проектирование.
  - Осуществить физическое проектирование.
2. Спроектировать интерфейс к созданной БД.

Реализация данного проекта позволит курьерской службе значительно оптимизировать организацию работы, повысить эффективность управления данными и предоставить клиентам более качественное обслуживание.

## ГЛАВА 1. ПРОЕКТИРОВАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ

### 1.1. АНАЛИЗ ПРЕМЕТНОЙ ОБЛАСТИ

Предметная область – конкретная область деятельности, которая описывается и классифицируется на основе определенных критериев. Она содержит описание элементов, событий, отношений и процессов, отражающих различные аспекты этой деятельности. Изучение и анализ предметной области являются первоочередными задачами разработчика программной системы, поскольку это влияет на все аспекты проекта: требования к системе, взаимодействие с пользователями, хранение данных, реализацию функционала и другие аспекты.

Анализ предметной области помогает выделить ключевые сущности, определить начальные требования к функциональности и определить границы проекта.

В данной работе в качестве предметной области рассматривается деятельность курьерских служб.

Изучение данной области является актуальной проблемой в связи с развитием цифровых технологий и спросом на хранение и обработку больших объемов данных. С появлением большого количества информации, включая персональные данные клиентов и сотрудников, данные о заказах, менеджеры курьерских служб столкнулись с необходимостью эффективного управления информацией и использования баз данных для улучшения бизнес-процессов.

Задачи, которые решаются в описываемой предметной области:

- хранение информации о клиентах, заказах, сотрудниках;
- управление заказами;
- сбор обратной связи о работе;
- маршрутизация и оптимизация доставки;
- управление финансовыми транзакциями, выставление счета клиентам;
- отслеживание грузов (определение их местоположения и статуса);

- анализ данных о работе курьеров, эффективности доставки, удовлетворенности клиентов.

Существует множество программных продуктов, решающих задачи в данной области:

- сайты курьерских служб (eda.yandex.ru, dostavista.ru, delivery-club.ru, cdek.ru, cse.ru, fox-express.ru и другие);
- мобильный программный продукт «WB курьеры» от компании Wildberries LLC [1], «Озон Курьер Экспресс» от Internet Solutions LLC, «Додо курьер» от DODO Brands International;
- Системы автоматизации служб доставки EdiCourier от Edisoft [2];

Большинство из вышеперечисленных продуктов предназначено либо для поиска и непосредственного взаимодействия с курьерами (создание портрета курьера, предоставление сведений по заказу, оплата, проверка выполненной работы), либо для получения запросов от клиентов на выполнение работ.

Иные продукты представляют собой систему, которая помогает автоматически планировать маршруты. Данные системы учитывают график курьеров, окна доставки, и далее рассчитывают самый оптимальный маршрут из возможных. Таким образом, выполняется множество задач, которые влияют в будущем на имидж компании.

В качестве первого программного продукта рассмотрим сайт курьерской службы Dostavista (<https://dostavista.ru/>). Dostavista [3] – это сервис срочной курьерской доставки, осуществляющая свою деятельность не только в пределах РФ, но и за ее пределами. Информационная платформа Dostavista предназначена как для клиентов, так и для самих курьеров.

Курьерский портал Dostavista предоставляет следующие основные возможности:

- Управление заказами (рис.1.1). Клиент имеет возможность создать заказ, задать определенные требования к доставке, как по времени, так и по ее

5 способу транспортировки и оплаты. При необходимости можно оформить страхование посылки.

## Расчет и оформление доставки

**Как можно скорее**

Заказ заберёт и доставит ближайший свободный курьер

от 149 ₽

[Подробнее ▾](#)

**Запланировать**

Курьер будет на адресах в удобное вам время

от 149 ₽

**В интервал 4 часа**

Моментально назначим курьера и доставим точно в интервал

от 145 ₽ за адрес

**Как доставить**

Пешком
Легковой автомобиль
Грузовой

**Вес посылки**

до 1 кг
до 5 кг
до 10 кг
до 15 кг

**Адрес доставки**

Покровка, 11

+7 \_\_\_\_-\_\_\_\_-\_\_

✖ Введите адрес, чтобы узнать, когда придет курьер

Дополнительно ▾

**Адрес доставки**

Солянка, 4

+7 \_\_\_\_-\_\_\_\_-\_\_

✖ Введите адрес, чтобы узнать, когда придет курьер

☐ Бесконтактная доставка ?

Дополнительно ▾

+ Добавить адрес

Рис 1.1 Размещение заказа

- Управление доставкой на ходу. Клиенту предоставляется возможность отслеживать состояние заказа в режиме реального времени. Этапы заказа зависят от типа посылки, который заранее обозначается при оформлении.
- История и статистика пользователя (рис 1.2 – 1.3). В данном разделе можно увидеть все заказы, которые были совершены заказчиком. Встроенные фильтры доступны, однако есть возможность создать собственный. При необходимости имеется возможность получить все данные о заказах в виде .xls файла.

## Заказы

Сохранить в \*.xls 

Все заказы

Активные

Завершённые

Черновики

Рис 1.2 История заказов

Месяц	Тип	Число заказов	Общая сумма	Средняя цена заказа	Средняя цена адреса
ИТОГО		0	0	0	0

Рис 1.3 Статистика пользователя

- Возможность интеграции по API с сервисом доставки (рис. 1. 4).

## Документация и поддержка



Документация API

Все основные инструкции по настройке интеграции описаны в документации. Прежде чем переходить к настройке, ознакомьтесь с документацией.



Написать в поддержку

Вопросы, пожелания и заклипания пишите на [api@dostavista.ru](mailto:api@dostavista.ru).

## Интеграция с CRM и CMS

- ☐ retailCRM    ☐ Bitrix  
☐ InSales    ☐ Ecwid  
☐ Мой Склад    ☐ Мегалплан  
☐ amoCRM    ☐ WIX  
☐ Opencart 2.3    ☐ Opencart 3.x  
☐ Shopify    ☐ Битрикс24  
☐ WooCommerce

Выбрать

Рис 1.4 Интеграция по API

В качестве второго программного продукта рассмотрим «Додо Пицца» (<https://play.google.com/store/apps/details?id=ru.dodopizza.app>). Программный продукт «Додо Пицца» [4] предназначен для организации процесса доставки пищевой продукции, согласно предпочтениям клиента.

Среди основных функций данной программы стоит выделить:

- Управление заказом. Заказ создается подобно конструктору: пользователь выбирает различные представленные позиции и их многочисленные модификации, затем оформляет доставку (рис.1.5 – 1.6).

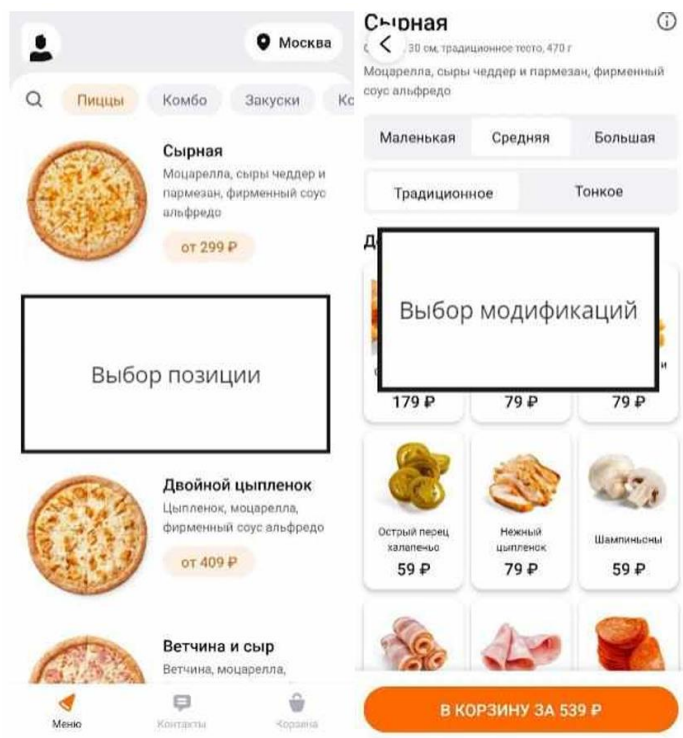


Рис 1.5 Выбор позиции и ее модификаций

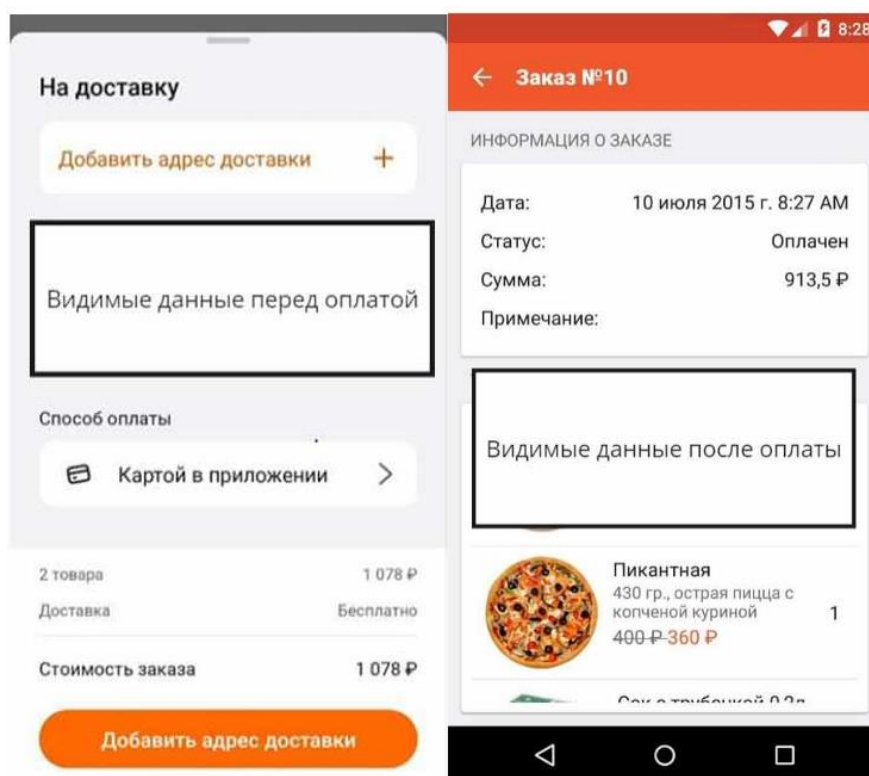


Рис 1.6 Сведения по доставке



- История сделанных заказов. Пользователь имеет возможность получить полные сведения по прошлым заказам, которые были сделаны на этом аккаунте (рис 1.7).



Рис 1.7 История заказов

- Отслеживание состояния в режиме реального времени. Пользователь может не только видеть статус заказа, но и его примерное время прибытия в точку назначения (рис. 1.8). Особенностью данного сервиса является возможность наблюдать за процессом приготовления продукции по специальной камере видеонаблюдения (рис. 1.9).

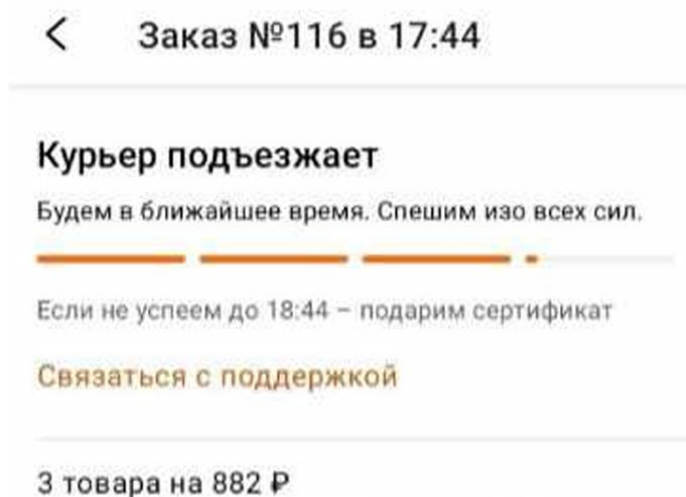


Рис 1.8 Отслеживание заказа

## Сыктывкар, ул. Первомайская, 85

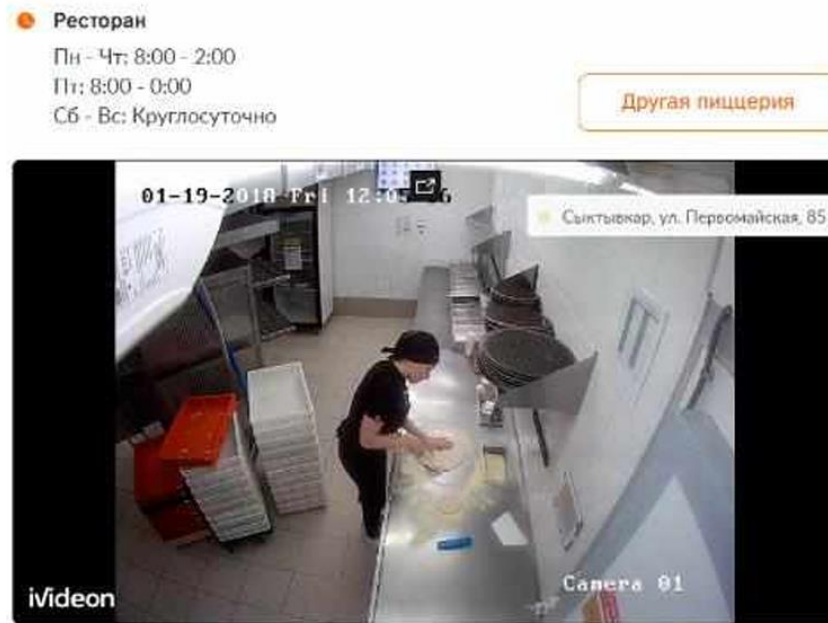


Рис 1.9 Отслеживание процесса приготовления заказа

- Оплата сделанного заказа (рис. 1.10). Клиент может выбрать предпочитаемый способ оплаты из пяти предложенных. При выборе оплаты безналичным расчетом, то автоматически создается запрос на оплату с перенаправлением на сайт банка.

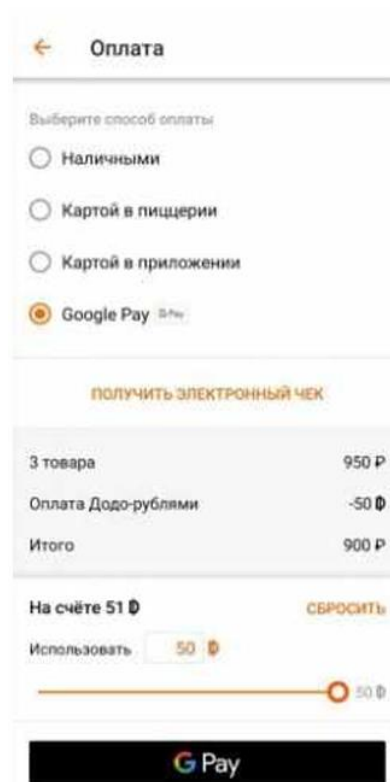


Рис 1.10 Система оплаты заказа

Для всех рассмотренных программных продуктов можно выделить общие функции:

- управление заказом;
- возможность отслеживания заказа;
- сохранение истории о сделанных заказах.

Особенностью портала для курьерской службы Dostavista является наличие возможности создать заказ с пометкой «как можно скорее». Внутренняя система на основе сложившейся ситуации на данный момент рассчитывает самый быстрый вариант доставки и предлагает его пользователю.

Следует отметить, что «Додо Пицца» также имеет свою систему оптимизации доставки, так как сервис гарантирует доставку своей продукции не более чем через за 60 минут с момента приема заказа курьерской службой.

Данные системы похожи, но имеют следующее отличие: Dostavista учитывает приоритетность того или иного заказа на основе решения пользователя при оформлении, а для сервиса «Додо Пицца» каждая заявка имеет одинаковую важность.

После проведения анализа предметной области был выделен перечень функций, которые будут реализованы в данной работе:

- Учет пользователей и регистрация заказов.
- Учет сотрудников курьерской службы.
- Учет доставок в соответствии с запланированным временем и с нарушениями сроков.
- Поиск и фильтрация по дате доставки, ее составу и выполнению в срок.
- Создание истории заказов.
- Отслеживание состояния заказа.

## 1.2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Концептуальное проектирование представляет собой важный этап разработки базы данных, который включает в себя формализованное описание предметной области, не привязанное к конкретной системе управления базами данных. Целью концептуального проектирования является создание обобщенной модели предметной области, определяющей содержание базы данных.

Для успешного выполнения этого этапа необходимо провести анализ задач и хранимых объектов, чтобы определить, какие данные должны быть храниться в базе данных. Ответ на вопрос о содержании базы данных является ключевым для построения эффективной структуры хранения информации и обеспечения нужной функциональности системы.

Для анализа задач часто используется диаграмма вариантов использования, которая помогает разработчику определить, какие задачи должны быть решены с помощью создаваемой базы данных. Диаграмма вариантов использования, также известная как диаграмма прецедентов, отображает взаимосвязь между действующим лицом и способами использования системы. Действующее лицо может быть любым объектом, субъектом или системой, взаимодействующим с моделируемой системой. Иными словами, вариант использования определяет функции, которые система предлагает действующему лицу, описывая набор действий, которые система выполняет при взаимодействии с этим лицом. Диаграммы вариантов использования показывают, что система должна делать, не уточняя как именно будут реализованы эти функции.

Словесное описание проектируемой системы в рамках предметной области приводится ниже.

На сайте потенциальные покупатели могут оставить самостоятельно оформить заказ. В него входят следующие поля: точка получения товара, дата и время получения товара, целевой адрес доставки, вес товара, номер телефона и ФИО получателя, если таковой имеется.

Оплата – финальный этап оформления заказа. После выполнения оплаты, заказ попадает во внутреннюю БД. Автоматизация расчетов с покупателями и ведения бухгалтерского учета на данном этапе не рассматриваются.

Курьер может выбрать и взять в работу заказ, который находится в базе, используя настраиваемую сортировку. Разрешается принимать не более 2 заказов и не более 1 заказа с пометкой «Экспресс» одновременно.

Для обеспечения информированности клиента о текущем состоянии доставки, что особенно важно при срочных заказах, клиент будет иметь возможность отслеживать статус своего заказа. Статусом заказа управляет курьер.

Как только заказ будет помечен как выполненный, пользователь сможет оставить обратную связь. Данное действие можно будет выполнить один раз.

Менеджер курьерской службы отслеживает выполняемость заказов, и на основе этого принимает решения о повышении/понижении ставки, которую получает курьер за выполнение заказа. Кроме того, в обязанности менеджера будет входить просмотр оставленных отзывов и своевременное реагирование на возникшие проблемы.

С проектируемой системой будут взаимодействовать следующие действующие лица:

- клиент,
- курьер,
- менеджер.

Рассмотрим варианты использования каждого из действующих лиц.

*Клиенту* доступны следующие функции:

1. Оформление заказа.
2. Оплата счета.
3. Просмотр статуса текущего заказа.
4. Просмотр истории заказанных услуг.
5. Создание отзывов.

## 6. Редактирование личных данных.

*Курьеру* доступны следующие функции:

1. Поиск доступных заказов с использованием фильтров.
2. Обновление статуса заказа.
3. Просмотр рейтинга и отзывов, оставленных на выполненные заказы.
4. Редактирование личных данных.

*Менеджеру* доступны следующие функции:

1. Повышение/понижение ставки курьеров за заказ(-ы).
2. Анализ выполняемости заказов.
3. Просмотр всех оставленных отзывов.
4. Просмотр доступных заказов с применением фильтров.
5. Редактирование личных данных.

Общая диаграмма вариантов использования представлена на рис. 1.11.

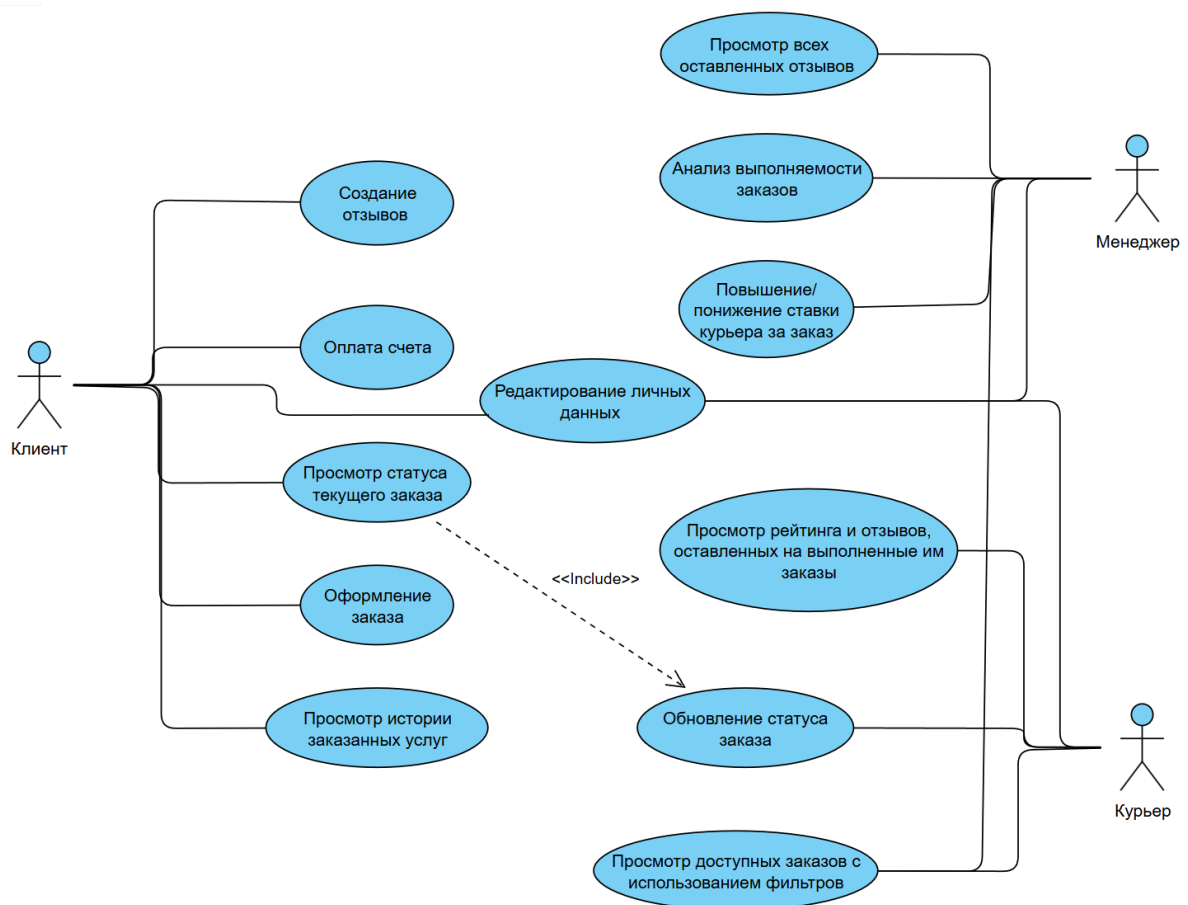


Рис 1.11 Диаграмма вариантов использования

В первом приближении для решения выделенных задач необходимо хранение данных о следующих объектах:

1. Клиент - содержит информацию о всех клиентах системы (фамилия, имя, отчество, дата рождения, номер телефона, адрес проживания).
2. Заказ – содержит полную информацию об оставленном пользователем заказе (адрес получения товара, дата получения заказа, дата выполнения заказа, целевой адрес доставки, номер телефона и ФИО получателя, статус заказа, процент курьера).
3. Курьер – содержит полную информацию о всех курьерах службы (фамилия, имя, отчество, дата рождения, дата регистрации, номер телефона, данные паспорта, количество выполненных заказов).
4. Отзыв – содержит информацию об отзывах пользователей (данные пользователя, номер заказа, текст отзыва, оценка).
5. Тариф – содержит информацию о стоимости доставки в зависимости от выбранного пользователем тарифа.
6. Счёт – содержит данные о проводимых денежных поступления со стороны клиентов (идентификатор оплаты, сумма оплаты, номер заказа).

### 1.3. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Логическое проектирование – важный этап разработки базы данных, который включает в себя создание схемы БД на основе выбранной модели организации данных, не привязанное к конкретной системе управления базами данных. В данной работе будет использоваться реляционная модель.

Целью логического проектирования является определение состава структуры таблиц БД на основе результатов концептуального проектирования, проверка полученной модели с помощью методов нормализации.

Для успешного выполнения этого этапа необходимо не только провести формирование отношений, но и определить ограничения предметной области. Ответы на эти вопросы являются ключевыми для построения полноценной и эффективной структуры хранения информации.

Для выполнения вышеуказанных задач часто используется ER-диаграмма, которая помогает разработчику смоделировать будущую модель отношений. Этот графический инструмент не только отображает основные сущности, но и обозначает взаимосвязи между ними: сущности представлены в виде прямоугольных блоков с указанием принадлежащих им атрибутов, а связи между ними – ромбами, указывающими тип и степень связи.

В процессе проектирования баз данных концептуальный и логический этапы связаны между собой.

Результатом концептуального этапа проектирования является определение объектов, о которых будет храниться информация. Этот этап фокусируется на высокоуровневом представлении данных без уточнения деталей определенной модели базы данных.

На логическом этапе проектирования создаются ER-диаграммы на основе объектов, определенных на концептуальном уровне, и затем эти диаграммы преобразуются в отношения. Таким образом, на логическом этапе каждая сущность представляется в виде таблицы, а ее атрибуты - в виде столбцов данной таблицы.

Построим ER-диаграммы всех сущностей и связей между ними.



Каждый клиент оформляет несколько заказов или ни одного, каждый заказ обязательно оформлен только на одного клиента (рис. 1.12).

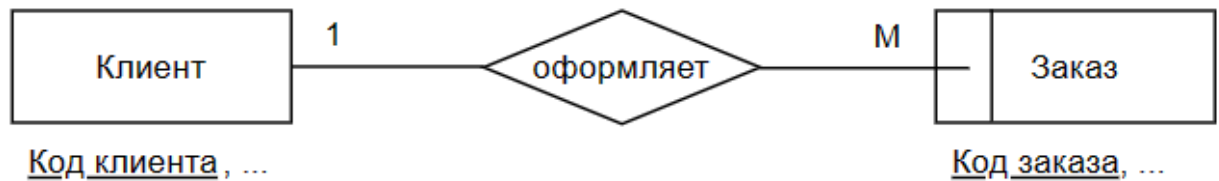


Рис 1.12 ER-диаграмма «клиент-заказ»

Каждый курьер принимает несколько заказов или ни одного, каждый заказ может быть принят только одним курьером или не быть принятым вовсе (рис 1.13).

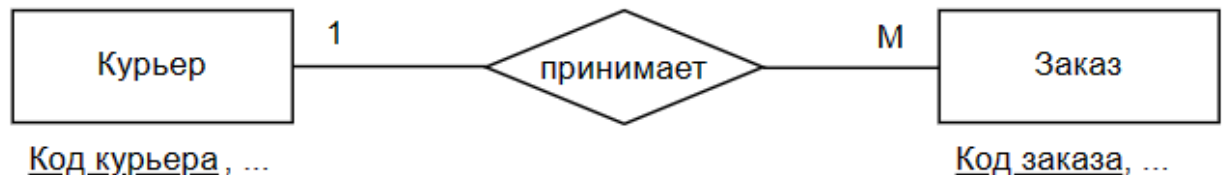


Рис 1.13 ER-диаграмма «курьер-заказ»

Каждый клиент оставляет несколько отзывов или ни одного, каждый отзыв обязательно оставлен только одним клиентом (рис. 1.14).



Рис 1.14 ER-диаграмма «клиент-отзыв»

Каждый тариф может быть применен к нескольким заказам или не применен вовсе, каждый заказ должен быть отнесен только к одному из тарифов (рис. 1.15).

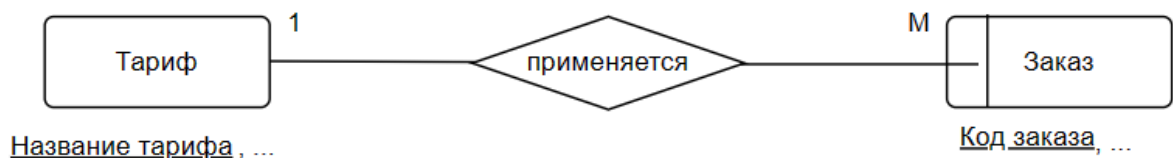


Рис 1.15 ER-диаграмма «тариф-заказ»

Каждый счёт обязательно относится к единственному заказу, каждый заказ обязательно имеет единственные данные об оплате (рис. 1.16).

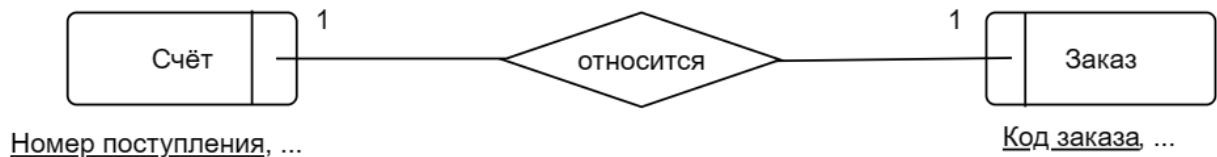


Рис 1.16 ER-диаграмма «счёт-заказ»

Каждый заказ содержит единственный отзыв или не одного, каждый отзыв обязательно относится к только одному заказу (рис. 1.17).



Рис 1.17 ER-диаграмма «заказ-отзыв»

Изучение предметной области представляет собой процесс анализа конкретной области знаний или деятельности с целью выявления и описания основных составляющих, структуры, процессов и взаимосвязей в этой области.

Основная задача анализа предметной области состоит в том, чтобы не только получить полное представление о предмете изучения, а также создать основу для разработки системы или продукта, связанного с данной областью знаний.

Сформируем набор предварительных отношений с указанием предполагаемого первичного ключа для каждого отношения.

Связь ОФОРМЛЯЕТ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Клиент (Код клиента, ...)
2. Заказ (Код заказа, Код клиента, ...)

Связь ПРИНИМАЕТ удовлетворяет условиям правила 5, в соответствии с которым получаем три отношения:

1. Курьер (Код курьера, ...)
2. Заказ (Код заказа, ...)
3. В\_работе (Код заказа, Код курьера, ...)

Связь ОСТАВЛЯЕТ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Клиент (Код клиента, ...)
2. Отзыв (Код отзыва, Код клиента...)

Связь ПРИМЕНЯЕТСЯ удовлетворяет условиям правила 4, в соответствии с которым получаем два отношения:

1. Заказ (Код заказа, Название тарифа, ...)
2. Тариф (Название тарифа, ...)

Связь ОТНОСИТСЯ удовлетворяет условиям правила 1, в соответствии с которым получаем одно отношение:

1. Заказ (Код заказа, Номер поступления, ...)

Связь СОДЕРЖИТ удовлетворяет условиям правила 2, в соответствии с которым получаем два отношения:

1. Заказ (Код заказа, Номер поступления, ...)
2. Отзыв (Код отзыва, Код клиента, Код отзыва, ...)

Таким образом, добавляя к существующим отношениям неключевые атрибуты и соблюдая требования третьей нормальной формы, мы получаем следующие отношения:

- Клиент (КодКлиента, ФИО, НомерТелефона, ДатаРождения, АдресКлиента)
- Заказ (КодЗаказа, КодКлиента, НазваниеТарифа, Вес, АдресПолученияТовара, АдресДоставки, ФИО\_Получателя, НомерТелефонаПолучателя, СтатусЗаказа, НомерПоступления, ПроцентКурьера, ДатаЗаказа, СтоимостьЗаказа, ДатаВыполнения)
- Тариф (НазваниеТарифа, Цена)
- Курьер (КодКурьера, ФИО, ДатаРегистрации, ДатаРождения, КоличествоВыполненныхЗаказов, ДанныеПаспорта, НомерТелефона)
- В\_Работе (КодЗаказа, КодКурьера)
- Отзыв (КодОтзыва, КодКлиента, КодЗаказа, Описание, Оценка)

Ограничения предметной области, или бизнес-правила, представляют собой набор правил и ограничений, которые определяют допустимые параметры для выполнения бизнес-процессов, операций и принятия решений в конкретной предметной области. Эти ограничения могут включать в себя как допустимые значения полей данных, условия для выполнения определенных действий, так и требования к качеству данных, порядку выполнения операций. Бизнес-правила помогают обеспечить согласованность и эффективность в работе организации, устанавливая рамки и правила, которые должны соблюдаться для успешного функционирования бизнес-процессов.

Можно выделить следующие ограничения предметной области:

- клиент не может заказывать доставку товаров с пометкой «Экспресс», вес которых превышает 10 кг;
- клиент должен быть не младше 14 лет, иметь уникальный номер телефона;
- курьер имеет право одновременно взять не более 2 заказов и не более 1 заказа с пометкой «Экспресс» одновременно;
- курьер должен быть не младше 18 лет, иметь уникальный номер телефона;
- доступ к категории «Экспресс» заказов курьер получает после 20 успешно выполненных заказов;
- выполненный заказ нельзя снова взять в работу;
- курьер должен доставлять товар только к указанному клиентом адресу;
- изменить данные доставки нельзя, если заказ приобрел статус «В работе»;
- выполнение доставки возможно только после оформления заказа;
- отзыв может оставить только тот клиент, который оформил заказ, и только после его выполнения, причём сделать это можно не более одного раза;
- процент вознаграждения курьера за заказ составляет от 80 до 90%.

## 1.4. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Физическое проектирование базы данных является одним из ключевых этапов процесса проектирования базы данных, который заключается в реализации уже существующей логической модели данных на конкретной СУБД. Основной целью физического проектирования является определение общей схемы БД (определение связей между таблицами), выбор типов данных и значений по умолчанию для каждого атрибута, определение ключей и иных ограничений.

PostgreSQL [5] и другие системы управления базами данных (СУБД) предоставляют различные возможности для создания и заполнения таблиц. В частности, в PostgreSQL можно использовать инструменты для создания таблиц визуально, определяя атрибуты, их типы данных, ключи и другие ограничения. Также можно использовать SQL-запросы для создания таблиц и заполнения их данными. Данная СУБД обеспечивает удобный пользовательский интерфейс для управления базой данных, позволяя легко создавать, изменять и заполнять таблицы не только в конструкторах, но и с использованием SQL-операторов.

На логическом этапе проектирования базы данных определяются сущности, их атрибуты и связи между сущностями. Это включает создание схемы отношений, включающей таблицы, их атрибуты и ассоциации между ними. На физическом этапе проектирования логическая модель преобразуется в конкретную структуру таблиц, индексов, ключей и других элементов базы данных. Схема отношений, созданная на логическом этапе, служит основой для определения структуры базы данных на физическом уровне. Таким образом, физическое проектирование базы данных связано с логическим этапом через преобразование логической модели в физическую, используя сформированный набор схем отношений в качестве основы.

В качестве СУБД в данной главе будет использоваться PostgreSQL.

На основе выполнения предыдущего этапа был получен список отношений, которые обязательно должны входить в состав описываемой базы данных.

Таким образом, учитывая имеющиеся ограничения предметной области, получаем следующие таблицы (табл. 1.1 – 1.6):

Таблица 1.1

## Требования к структуре таблицы «Клиент»

Столбец	Тип данных	Ноль?	Ключ	Значение по умолчанию	Ограничение
КодКлиента	Целое	Нет	Первичный		
ФИО	Строка	Нет			
Номер Телефона	Строка	Нет	Уникальный		
ДатаРождения	Дата	Нет			14 лет
АдресКлиента	Строка	Нет			

Таблица 1.2

## Требования к структуре таблицы «Тариф»

Столбец	Тип данных	Ноль?	Ключ	Значение по умолчанию	Ограничение
Название Тарифа	Строка	Нет	Первичный		
Цена	Целое	Нет			> 0

Таблица 1.3

## Требования к структуре таблицы «Отзыв»

Столбец	Тип данных	Ноль?	Ключ	Значение по умолчанию	Ограничение	Ссылка
КодОтзыва	Целое	Нет	Первичный			
КодКлиента	Целое	Нет	Внешний			Код Клиента в Клиент
КодЗаказа	Целое	Нет	Внешний			КодЗаказа в Заказ
Описание	Строка	Да				
Оценка	Целое	Нет		1	От 1 до 5	

Требования к структуре таблицы «Заказ»

Столбец	Тип данных	Нуль ?	Ключ	Значение по умолчанию	Ограничение	Ссылка
КодЗаказа	Целое	Нет	Первичный			
КодКлиента	Целое	Нет	Внешний			Код Клиента в Клиент
Название Тарифа	Строка	Нет	Внешний			Название Тарифа в Тариф
Вес	Десятичное	Нет			> 0.0	
Адрес Получения Товара	Строка	Нет				
Адрес Доставки	Строка	Нет				
ФИО Получателя	Строка	Да				
Номер Телефона Получателя	Строка	Да				
Статус Заказа	Строка	Нет		‘Не принят’	‘Не принят’, ‘В работе’, ‘Выполнен’	
Номер Поступления	Целое	Нет	Уникальный			
Процент Курьера	Целое	Нет		80	От 0 до 100	
ДатаЗаказа	Дата	Нет		Текущая дата		
Стоимость Заказа	Десятичное	Нет			> 0.0	
Дата Выполнения	Дата	Да			>= Дата Заказа	

Таблица 1.5

## Отношение «Курьер»

Столбец	Тип данных	Нуль?	Ключ	Значение по умолчанию	Ограничение
КодКурьера	Целое	Нет	Первичный		
ФИО	Строка	Нет			
Дата Регистрации	Дата	Нет		Текущая дата	
ДатаРождения	Дата	Нет			18 лет
Количество Выполненных Заказов	Целое	Нет		0	$\geq 0$
Данные Паспорта	Строка	Нет	Уникальный		
НомерТелефона	Строка	Нет			

Таблица 1.6

## Отношение «ВРаботе»

Столбец	Тип данных	Нуль?	Ключ	Ограничение	Ссылка
КодЗаказа	Целое	Нет	Первичный		КодЗаказа в Заказ
КодКурьера	Целое	Нет			КодКурьера в Курьер

Создание таблиц с использованием языка SQL и заполнение их данными представлены в приложении А и приложении Б соответственно.

Общая схема БД, показывающая связи таблиц и их состав, представлена на рис. 1.18.



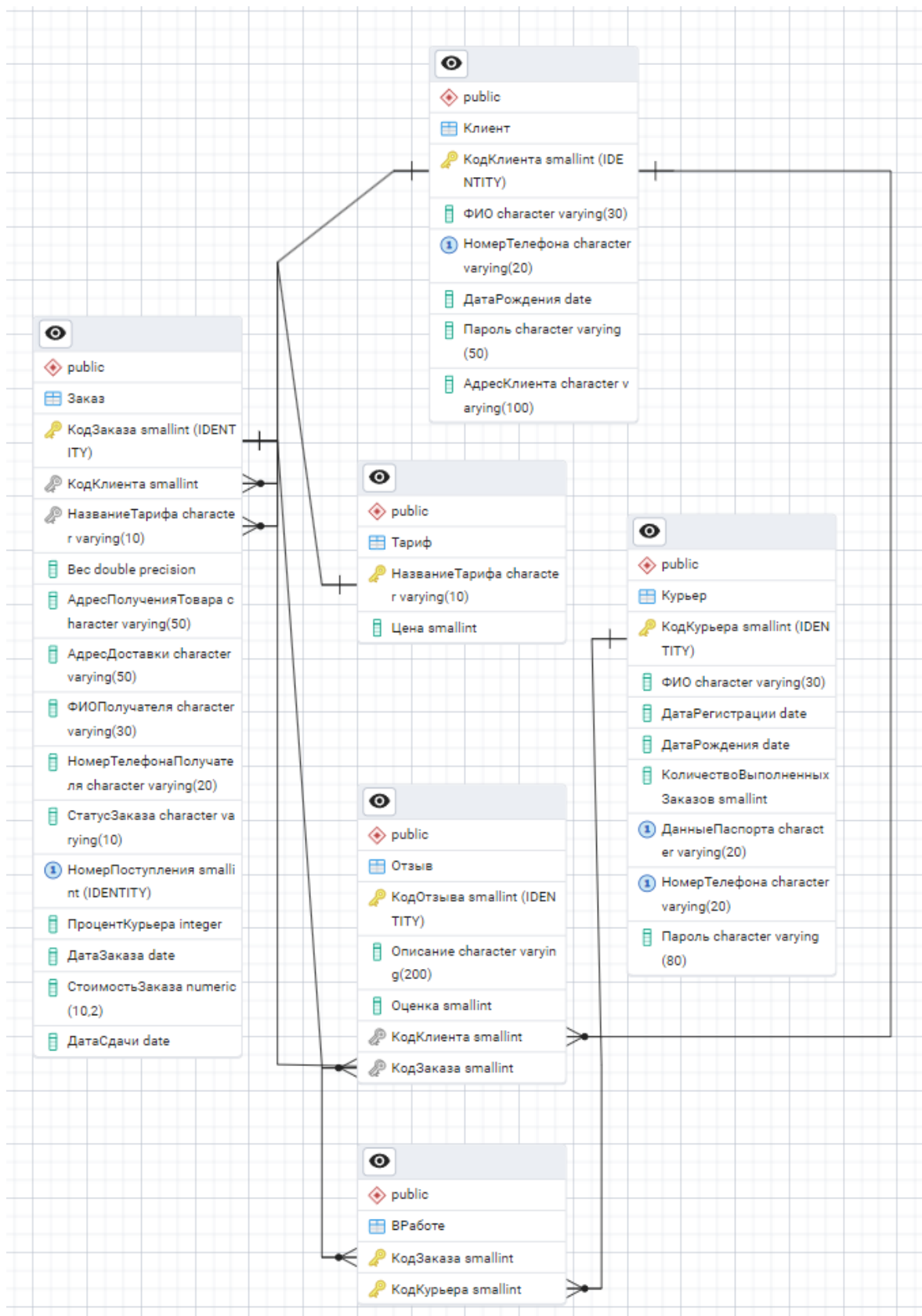


Рис 1.18. Схема БД

В первую очередь были созданы таблицы Клиент (приложение А, п. 1), Заказ (приложение А, п. 2), Тариф (приложение А, п. 3), Курьер (приложение А, п. 4), ВРаботе (приложение А, п. 5), Отзыв (приложение А, п. 6).

Кроме таблиц в базе данных созданы иные объекты: процедуры, представления, триггеры и триггерные функции.

Были реализованы следующие триггеры:

- «trg\_check\_courier\_order» таблицы ВРаботе (приложение А, п. 7) необходим для обеспечения ограничения предметной области: клиент не может заказывать доставку товаров с пометкой «Экспресс», вес которых превышает 10 кг,
- «check\_weigth\_trigger» таблицы Заказ (приложение А, п. 8) необходим для обеспечения ограничения предметной области: клиент не может заказывать доставку товаров с пометкой «Экспресс», вес которых превышает 10 кг,
- «update\_info\_after\_break\_order» таблицы Заказ (приложение А, п. 9) необходим для удаления записи в таблице ВРаботе в случае отказа курьера от принятого им заказа,
- «update\_info\_after\_ending\_order» таблицы Заказ (приложение А, п. 10) необходим для обновления данных о заказе и курьере после выполнения заказа,
- «trg\_check\_unique\_review» таблицы Отзыв (приложение А, п. 11) необходим для обеспечения ограничения предметной области: отзыв на заказ можно оставлять лишь единожды.

Для обеспечения функциональности данных триггеров были созданы следующие триггерные функции:

- «trg\_func\_check\_courier\_orders» (приложение А, п. 12),
- «express\_order\_check» (приложение А, п. 13),
- «update\_vrabote\_after\_break\_order» (приложение А, п. 14),
- «trg\_delete\_vrabote» (приложение А, п. 15),
- «check\_unique\_review» (приложение А, п. 16).

Для принятия заказа в работу используется процедура «courier\_take\_order» (приложение А, п. 17). В качестве входных параметров принимает ID курьера и ID заказа. В результате выполнения процедуры добавляется запись в таблицу ВРаботе, а также статус заказа изменяется на «В работе». Значения параметров определяются на уровне интерфейса.

В этой работе используется представление «ИсторияПользователей» (см. приложение А, пункт 18). Это представление содержит исключительно те данные о заказах, которые необходимы пользователям.

## ГЛАВА 2. ОПИСАНИЕ ПРОГРАММНОГО ИНТЕРФЕЙСА К БД

### 2.1. ЗАДАЧИ ИНТЕРФЕЙСА И ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ

Этот интерфейс создан для обеспечения эффективного взаимодействия пользователей с базой данных. В системе предусмотрено три основные роли пользователей: курьер, клиент и менеджер. Каждая из этих ролей обладает уникальными функциями и возможностями, которые подробно описаны в главе 1, пункте 1.2. Таким образом, интерфейс предоставляет гибкие и специализированные возможности для каждой роли, что позволяет пользователям максимально эффективно использовать базу данных в соответствии с их обязанностями и потребностями.

Для разработки интерфейса к базе данных были использованы следующие современные технологии:

1. *Фреймворк Qt [6] (версия 6.8.0) на языке C++*. Qt 6.8.0 был выбран для создания пользовательского интерфейса благодаря его высокой производительности, кроссплатформенности и обширному набору инструментов для разработки графических приложений. Использование языка C++ в сочетании с Qt позволяет эффективно управлять ресурсами и создавать сложные, отзывчивые интерфейсы. Qt предоставляет богатые библиотеки для работы с графикой, сетевыми соединениями, базами данных и другими необходимыми компонентами, что значительно ускоряет процесс разработки и повышает качество конечного продукта.

2. *Qt Creator [7] (версия 14.0.2) – интегрированная среда разработки (IDE)*. Qt Creator предлагает удобный и интуитивно понятный интерфейс, оснащённый мощными инструментами для написания, отладки и тестирования кода. Эта среда специально оптимизирована для работы с фреймворком Qt, что позволяет разработчикам максимально эффективно использовать все возможности Qt, такие как визуальный дизайнер интерфейсов, автозавершение кода, интеграция с системами контроля версий и средства профилирования производительности. Встроенные шаблоны

проектов и примеры кода помогают ускорить начальные этапы разработки и способствуют соблюдению лучших практик программирования.

Выбор фреймворка Qt и среды разработки Qt Creator был обусловлен их высокой совместимостью и возможностью создавать надёжные, масштабируемые приложения с богатым функционалом. Эти инструменты позволяют разработчикам сосредоточиться на реализации бизнес-логики и улучшении пользовательского опыта, минимизируя время, затрачиваемое на настройку и управление низкоуровневыми аспектами разработки. В результате использование Qt и Qt Creator обеспечивает создание интуитивно понятного, стабильного и производительного интерфейса к базе данных, соответствующего всем современным требованиям и стандартам.

Следует отметить, что в данной работе используются объекты, не входящие в стандартный набор фреймворка Qt. Эти объекты были разработаны самостоятельно на основе уже существующих объектов. Их исходный код представлен в приложении В, п. 9-17.

Для расчета расстояния между адресом доставки и адресом получения применяется следующие инструменты:

1. *API Яндекс.Карт* [8]. Данный сервис предоставляет различные возможности для разработчиков для интеграции картографических сервисов Яндекса в приложения. В частности, для данной работы использовалась технология геокодирования (преобразование адресов в точные географические координаты).

2. *API DistanceMatrix.ai* [9]. Данный API предоставляет технологию расчета расстояния между указанными географическими координатами в разных режимах.

Использование данных инструментов для географического анализа упрощает расчет стоимости доставки, учитывая расстояние между указанными пользователем адресами. Таким образом, интеграция вышеперечисленных сервисов в работу способствует более прозрачному и точному ценообразованию доставки.

## 2.2. ОПИСАНИЕ РАБОТЫ ИНТЕРФЕЙСА

Работа программы «Быстро – просто» начинается с формы авторизации. Используя данную форму, пользователь может как авторизоваться (рис. 2.1, приложение В, п. 2), так и пройти регистрацию, выбрав свою роль, если его аккаунта нет в системе (рис. 2.2 – 2.3, приложение В, п.1).

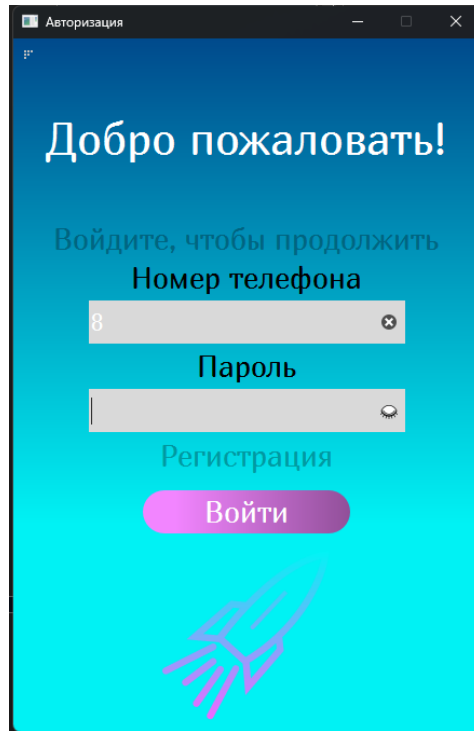


Рис. 2.1 Окно «Авторизация»

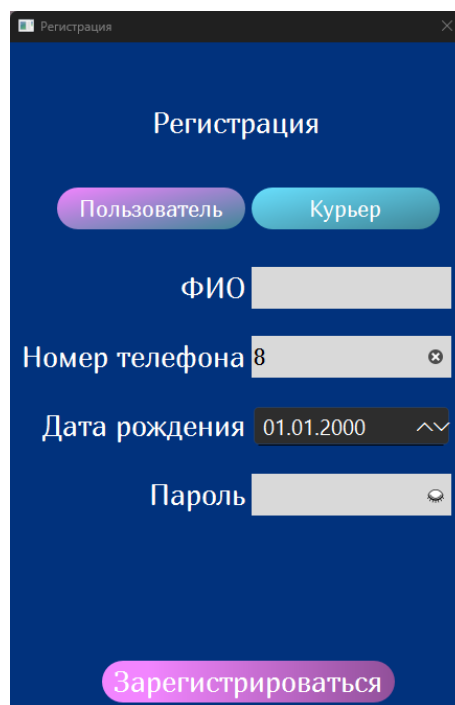
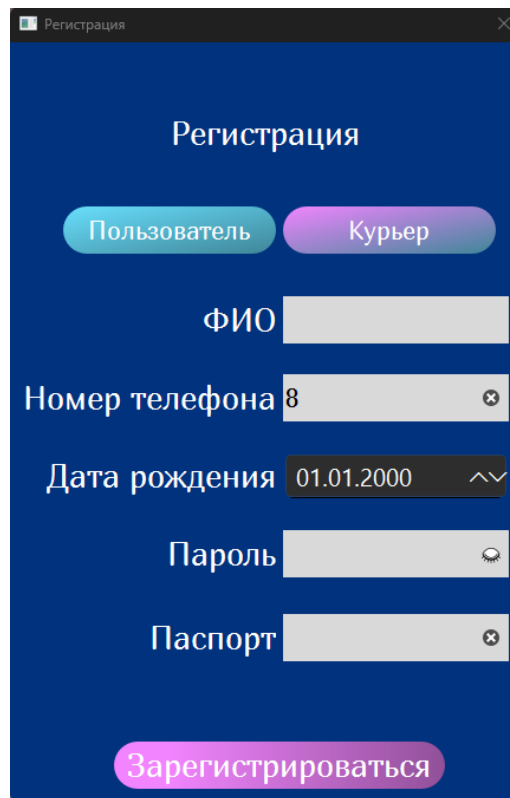


Рис. 2.2 Окно «Регистрация»



The screenshot shows a window titled 'Регистрация' (Registration) with a dark blue background. At the top, there are two buttons: 'Пользователь' (User) in light blue and 'Курьер' (Courier) in purple. Below these are input fields for 'ФИО' (Full Name), 'Номер телефона' (Phone Number) with a dropdown arrow, 'Дата рождения' (Date of Birth) with a date picker showing '01.01.2000', 'Пароль' (Password) with a toggle for visibility, and 'Паспорт' (Passport) with a dropdown arrow. At the bottom is a large purple button labeled 'Зарегистрироваться' (Register).

Рис. 2.3 Окно «Регистрация»

Проверка введенных значений присутствует на обоих окнах. Если данные аккаунта при попытке входа в него неверны, то появляется соответствующее сообщение (рис. 2.4). Если при попытке регистрации введены данные, которые противоречат ограничениям предметной области, то появляется сообщение, которое уведомляет об этом пользователя (рис. 2.5 – 2.7).

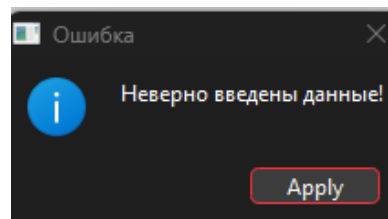


Рис. 2.4 Обработка неверно введенных данных при авторизации

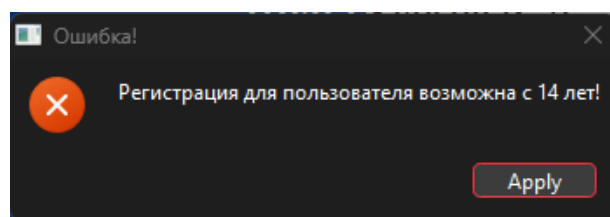


Рис. 2.5 Обработка ошибки возраста клиента

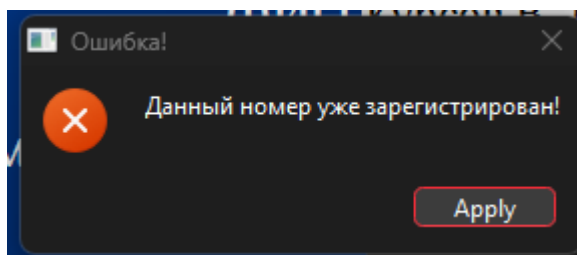


Рис. 2.6 Обработка ошибки при регистрации на неуникальный номер

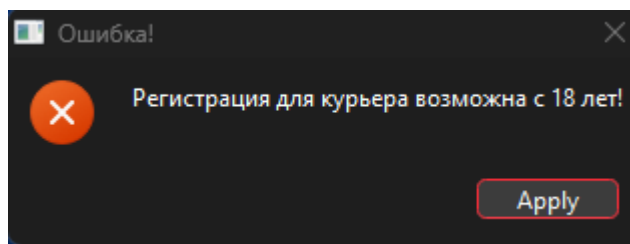


Рис. 2.7 Обработка ошибки возраста курьера

Рассмотрим работу программы со стороны пользователя. Войти в данный режим возможно только при вводе данных, которые принадлежат пользователю.

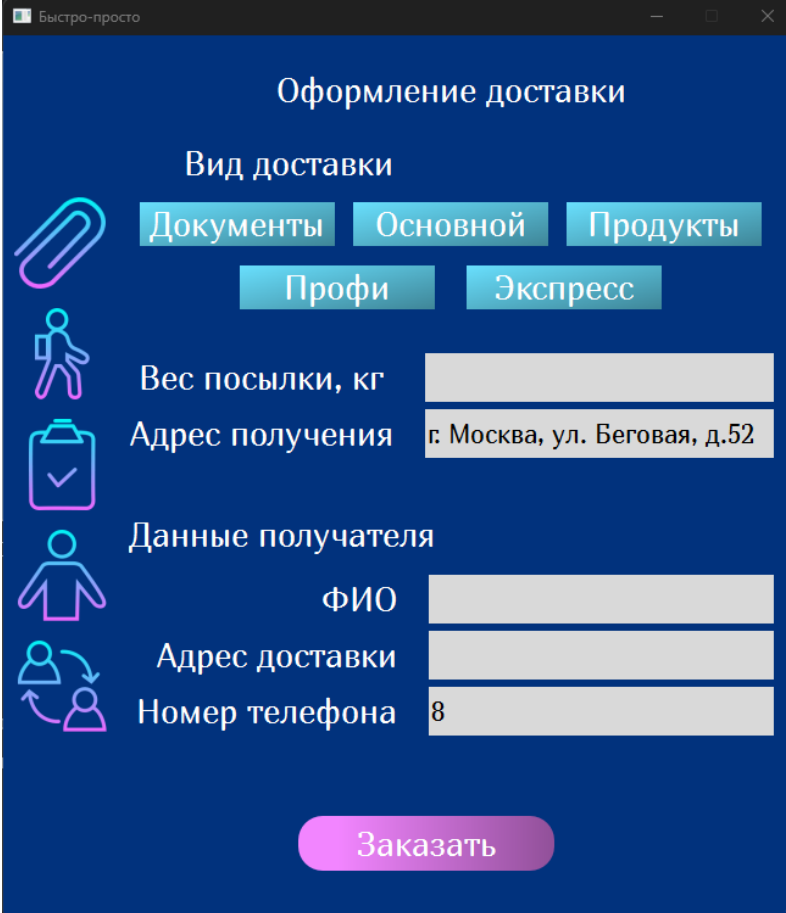
Встречает пользователя окно с оформлением заказа (рис. 2.8, приложение В, п. 3). Пользователь может выбрать один из представленных видов доставки путем нажатия соответствующей кнопки. Далее необходимо заполнить поля: данные о доставляемом товаре, данные о получателе и адрес доставки. Адрес получения устанавливается автоматически: данные о нем можно изменить в личном кабинете.

Если были введены некорректные данные, то при попытке создать заказ возникает ошибка с ее описанием (рис. 2.9-2.11). Ограничение на вес для тарифа «Экспресс» реализуется с помощью триггера «check\_weight\_trigger» (приложение А, п. 8).

Если были введены корректные данные, то при нажатии на кнопку «Заказать» возникает окно оплаты (рис. 2.12, приложение В, п. 4) с описанием суммы к оплате и полями для ввода данных карты. Сумма к оплате формируется на основании стоимости тарифа, а также расстояния между заявленными пользователем точками. В случае, если оплата не была произведена пользователем, то заказ не добавляется в базу данных и возникает



уведомление об этом (рис. 2.13). При успешной оплате также возникает уведомление (рис. 2.14).



The screenshot shows a window titled "Быстро-просто" with a dark blue background. The main heading is "Оформление доставки". Below it, under "Вид доставки", are five buttons: "Документы", "Основной" (highlighted), "Продукты", "Профи", and "Экспресс". To the left of these buttons are icons: a paperclip, a person with a backpack, a clipboard with a checkmark, a person, and two people with arrows. Below the buttons are input fields: "Вес посылки, кг" (empty), "Адрес получения" (filled with "г. Москва, ул. Беговая, д.52"), "Данные получателя" (section header), "ФИО" (empty), "Адрес доставки" (empty), and "Номер телефона" (filled with "8"). At the bottom is a large orange button labeled "Заказать".

Рис. 2.8 Окно «Оформление доставки»

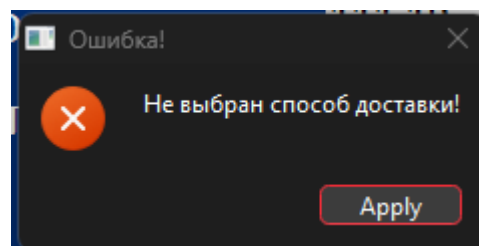


Рис. 2.9 Обработка ошибки при отсутствии выбранного способа доставки

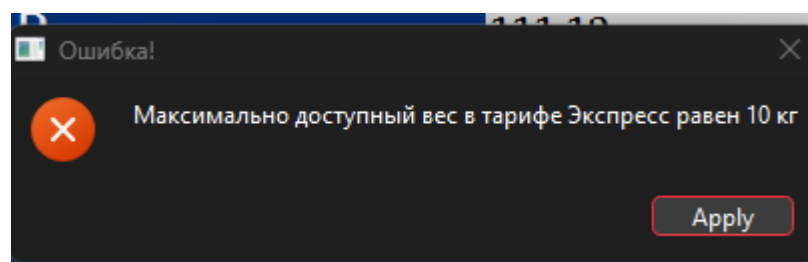


Рис. 2.10 Обработка ошибки некорректного веса

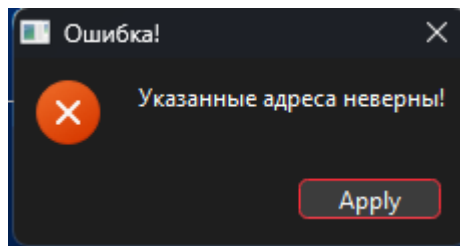


Рис. 2.11 Обработка ошибки отсутствия введенного адреса

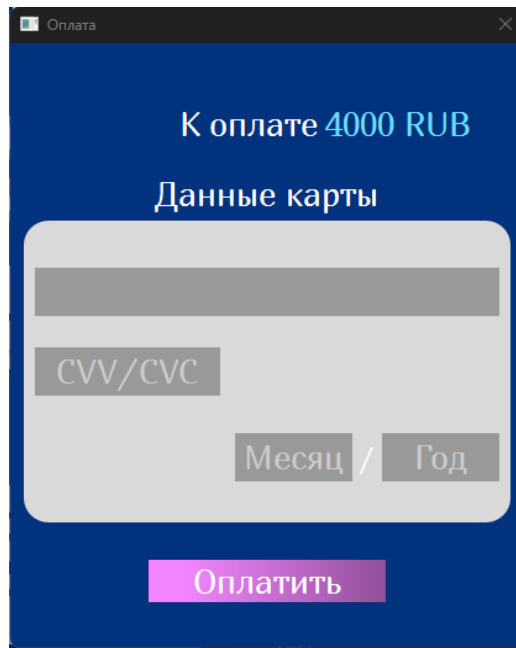


Рис. 2.12 Окно «Оплата»

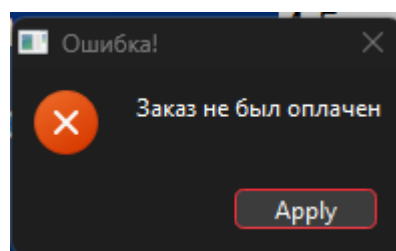


Рис. 2.13 Обработка ошибки при неоплате заказа

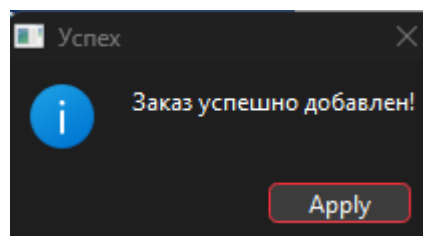


Рис. 2.14 Уведомление о добавлении заявки

При помощи меню (рис. 2.15), находящегося в левой части окна, пользователь может сменить окно. Данное меню видоизменяется в зависимости от роли пользователя.

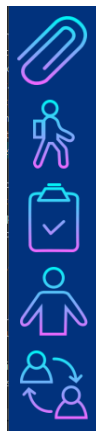


Рис. 2.15 Пример меню пользователей

Рассмотрим окно с текущими заказами пользователя (рис. 2.16, приложение В, п. 3). В данном окне отображаются только те заказы, которые в данный момент либо не приняты, либо находятся в процессе доставки. Для отображения данных используется представление «ИсторияПользователей» (приложение А, п. 18).

Для удобства работы подобные таблицы будут иметь слева «замороженную» колонку с кодом заказа. Если заказ еще не принят в работу, то пользователь имеет возможность изменить его или вовсе удалить путем нажатия соответствующей кнопки в таблице (рис. 2.17).

Если удаление заказа прошло успешно, то появляется уведомление (рис. 2.18). В случае изменения заказа возникает окно редактирования (рис. 2.19, приложение В, п. 5), в котором можно отредактировать некорректные данные. Изменения в заказ вносятся нажатием кнопки «Изменить». Перед изменением данных возникает уведомление для предотвращения случайного изменения (рис. 2.20). Если изменение прошло успешно, то возникает уведомление (рис. 2.21).



Рис. 2.16 Окно «Текущие заказы»

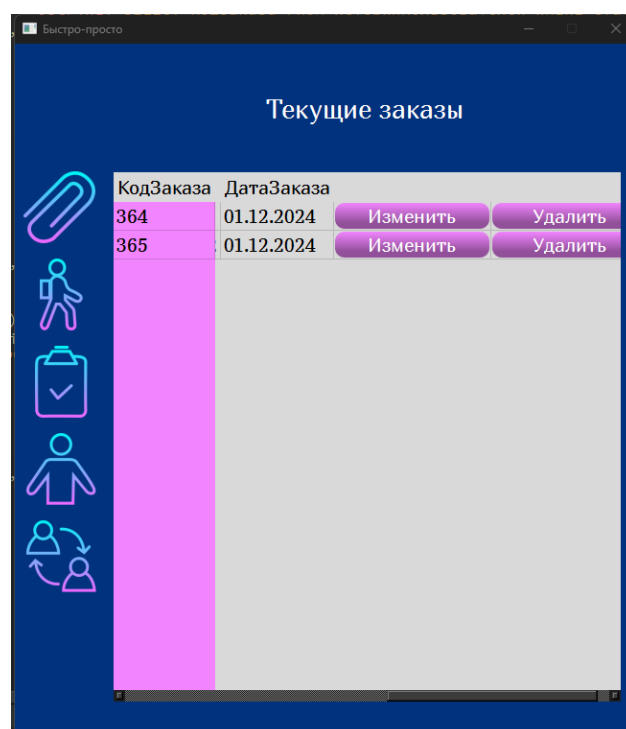


Рис. 2.17 Кнопки для редактирования/удаления заказа

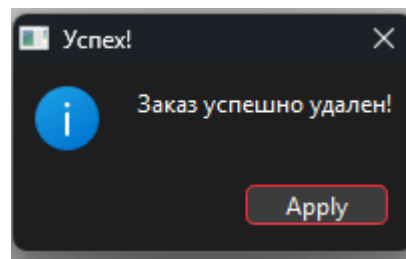


Рис. 2.18 Уведомление о успешном удалении заказа

A screenshot of a 'Редактировать заказ' (Edit order) window. The window has a blue header. Below the header, there are several input fields. The first is 'Вес' (Weight) with the value '0.23'. The second is 'Адрес получения' (Delivery address) with the value 'г. Москва, ул. Беговая, д...'. Below these is a section titled 'Данные получателя' (Recipient data). It contains three more input fields: 'Адрес' (Address) with 'г. Москва, ул. Беговая, д...', 'ФИО' (Full name) with 'Рывыро у.к', and 'Телефон' (Phone) with '8'. At the bottom center, there is a blue button labeled 'Изменить' (Change).

Рис. 2.19 Окно «Редактировать заказ»

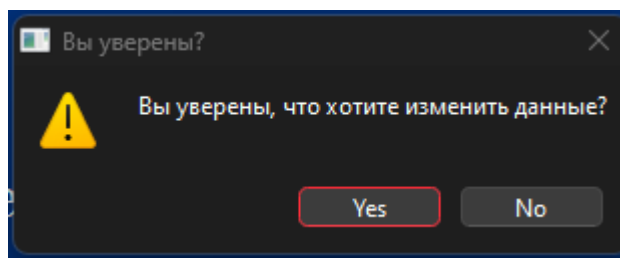


Рис. 2.20 Предупреждающее сообщение о редактировании заказа

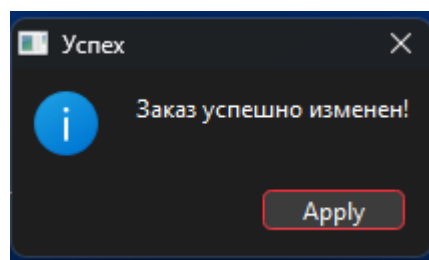
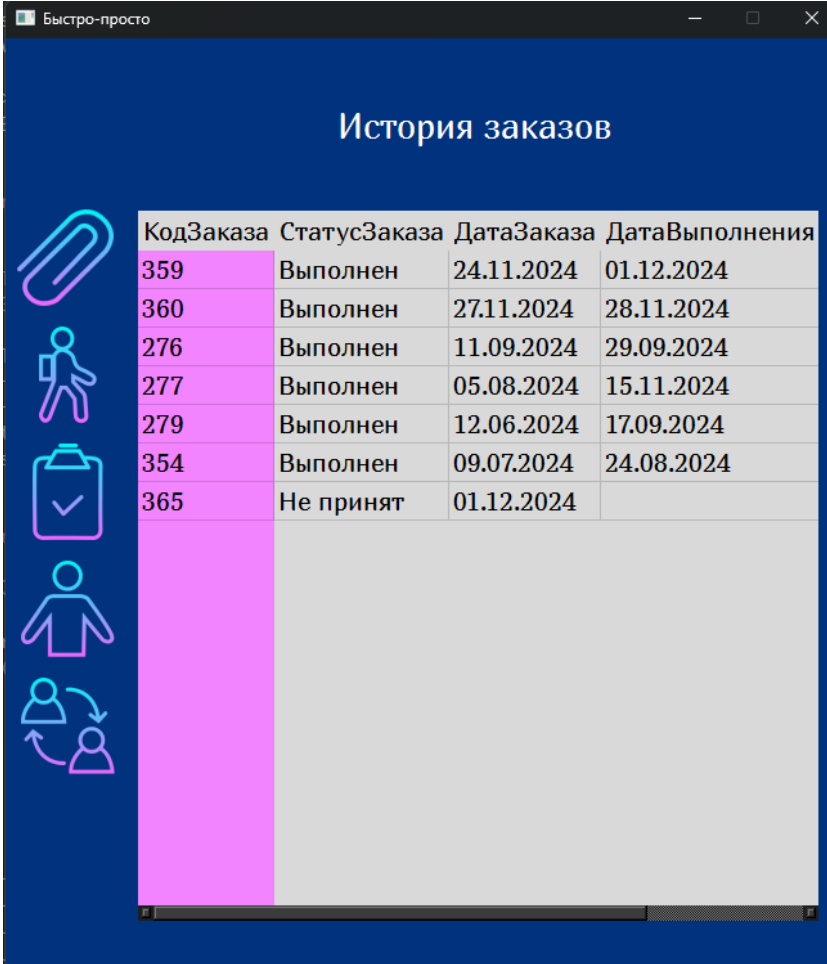


Рис. 2.21 Уведомление об успешном редактировании заказа

Рассмотрим окно с историей заказов пользователя (рис. 2.22, приложение В, п. 3). В данном окне отображаются все заказы пользователя с их статусом и датой создания и выполнения. Для отображения данных используется представление «ИсторияПользователей» (приложение А, п. 18).

Если заказ выполнен, и отзыв на него отсутствует, то пользователь имеет возможность создать отзыв на определенный заказ (рис. 2.23). Если данные условия не выполняются, то кнопка становится недоступной (рис. 2.23). Для дополнительной защиты от повторных отзывов реализован триггер «trg\_check\_unique\_review» (приложение А, п.11), проверяющий отсутствие отзывов на данный заказ.



КодЗаказа	СтатусЗаказа	ДатаЗаказа	ДатаВыполнения
359	Выполнен	24.11.2024	01.12.2024
360	Выполнен	27.11.2024	28.11.2024
276	Выполнен	11.09.2024	29.09.2024
277	Выполнен	05.08.2024	15.11.2024
279	Выполнен	12.06.2024	17.09.2024
354	Выполнен	09.07.2024	24.08.2024
365	Не принят	01.12.2024	

Рис. 2.22 Окно «История заказов»

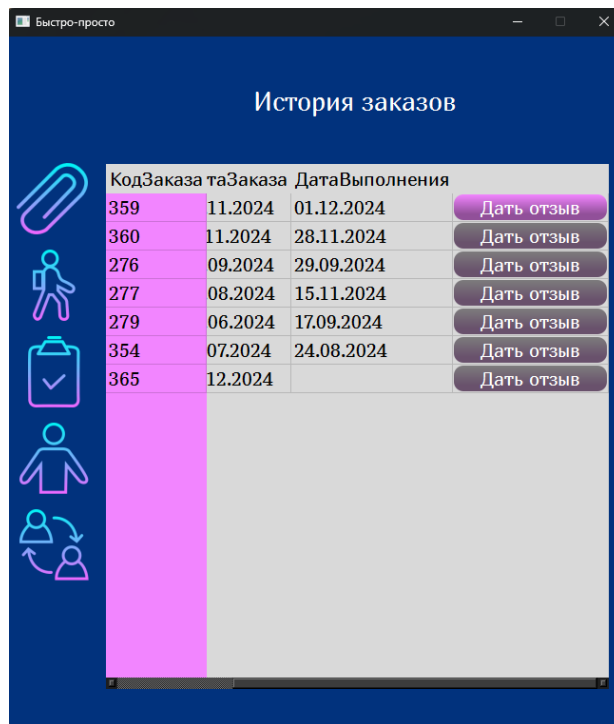


Рис. 2.23 Кнопки для создания отзыва

При нажатии на кнопку для создания отзыва открывается окно (см. рис. 2.24, приложение В, пункт 6), где можно выставить оценку и написать подробный комментарий. Оценка выставляется путем нажатия на звезды, которые расположены на интерфейсе (рис. 2.25). При нажатии на кнопку «Оставить отзыв» отзыв сохраняется, и возникает уведомление о успешно выполненном действии (рис. 2.26).

Выбор оценки для отзыва является обязательным. В случае если оценка не была выбрана пользователем, то возникает ошибка (рис. 2.27).



Рис. 2.24 Окно «Создать отзыв»

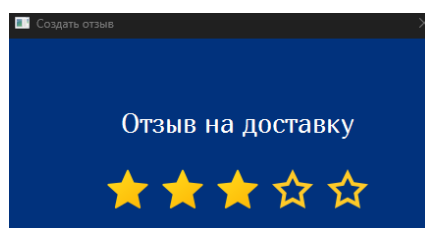


Рис. 2.25 Выбор определенной оценки

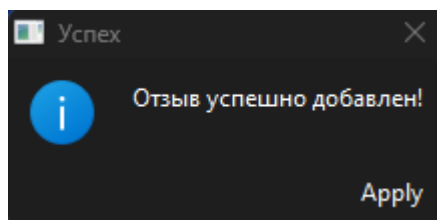


Рис. 2.26 Уведомление об успешном добавлении отзыва

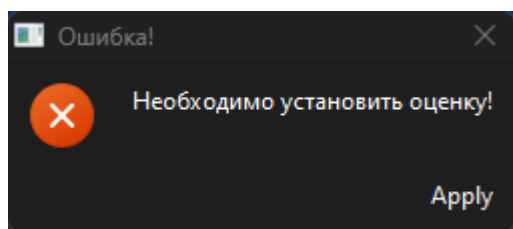


Рис. 2.27 Уведомление об ошибке отсутствия оценки

Рассмотрим работу программы со стороны курьера. Войти в данный режим возможно только при вводе данных, которые принадлежат курьеру.

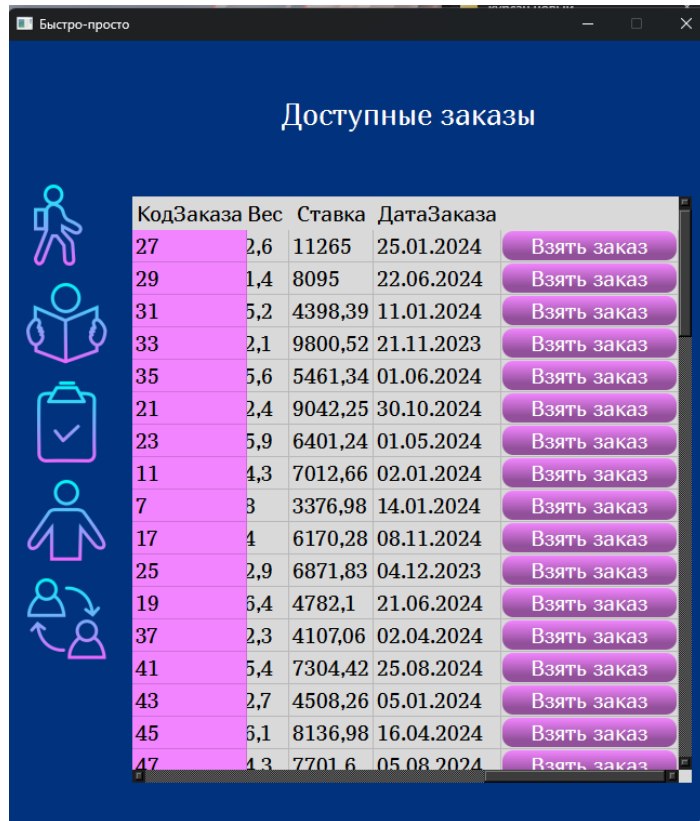
Встречает пользователя окно с доступными заказами (рис. 2.28, приложение В, п. 3). Для удобства использования списка реализован алгоритм сортировки (рис. 2.29-2.30). Чтобы принять заказ, необходимо нажать кнопку «Взять заказ» на соответствующей строке. При успешном выполнении операции возникает уведомление (рис. 2.31). Принятие заказа в работу реализуется через процедуру «courier\_take\_order» (приложение А, п. 17).

Для курьера, имеющего менее 20 выполненных заказов, становятся недоступны кнопки «Взять заказ» (рис. 2.32).

Кроме того, если курьер пытается принять более двух заказов одновременно, появляется предупреждение о невозможности выполнения этого действия (рис. 2.33). Подобное предупреждение появляется, если курьер пытается одновременно взять два заказа с тарифом «Экспресс» (рис. 2.33).



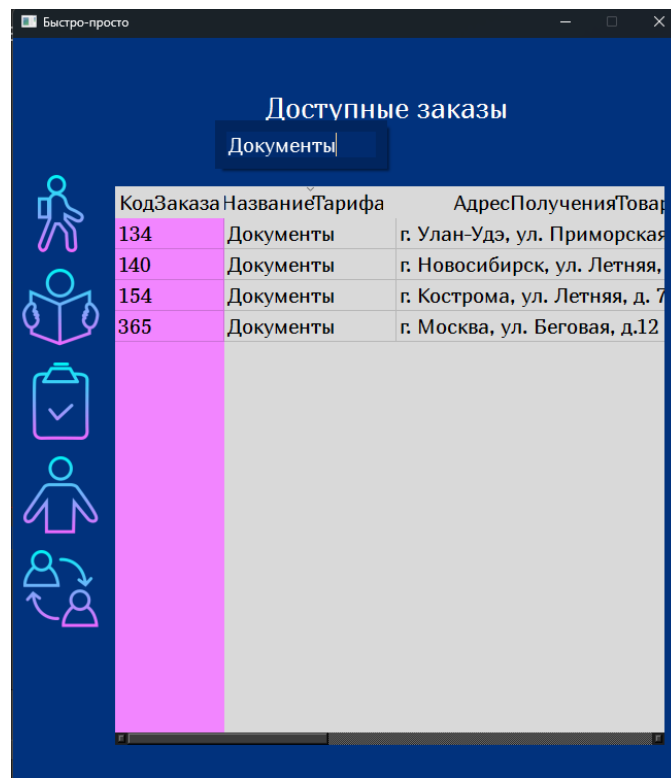
Данные ограничения реализуются использованием триггера «trg\_check\_count\_orders» (приложение А, п. 7).



Доступные заказы

КодЗаказа	Вес	Ставка	ДатаЗаказа	
27	2,6	11265	25.01.2024	Взять заказ
29	1,4	8095	22.06.2024	Взять заказ
31	5,2	4398,39	11.01.2024	Взять заказ
33	2,1	9800,52	21.11.2023	Взять заказ
35	5,6	5461,34	01.06.2024	Взять заказ
21	2,4	9042,25	30.10.2024	Взять заказ
23	5,9	6401,24	01.05.2024	Взять заказ
11	4,3	7012,66	02.01.2024	Взять заказ
7	3	3376,98	14.01.2024	Взять заказ
17	4	6170,28	08.11.2024	Взять заказ
25	2,9	6871,83	04.12.2023	Взять заказ
19	6,4	4782,1	21.06.2024	Взять заказ
37	2,3	4107,06	02.04.2024	Взять заказ
41	5,4	7304,42	25.08.2024	Взять заказ
43	2,7	4508,26	05.01.2024	Взять заказ
45	6,1	8136,98	16.04.2024	Взять заказ
47	4,3	7701,6	05.08.2024	Взять заказ

Рис. 2.28 Окно «Доступные заказы»



Доступные заказы

Документы

КодЗаказа	НазваниеТарифа	АдресПолученияТовара
134	Документы	г. Улан-Удэ, ул. Приморская
140	Документы	г. Новосибирск, ул. Летняя,
154	Документы	г. Кострома, ул. Летняя, д. 7
365	Документы	г. Москва, ул. Беговая, д.12

Рис. 2.29 Сортировка по полю «НазваниеТарифа»

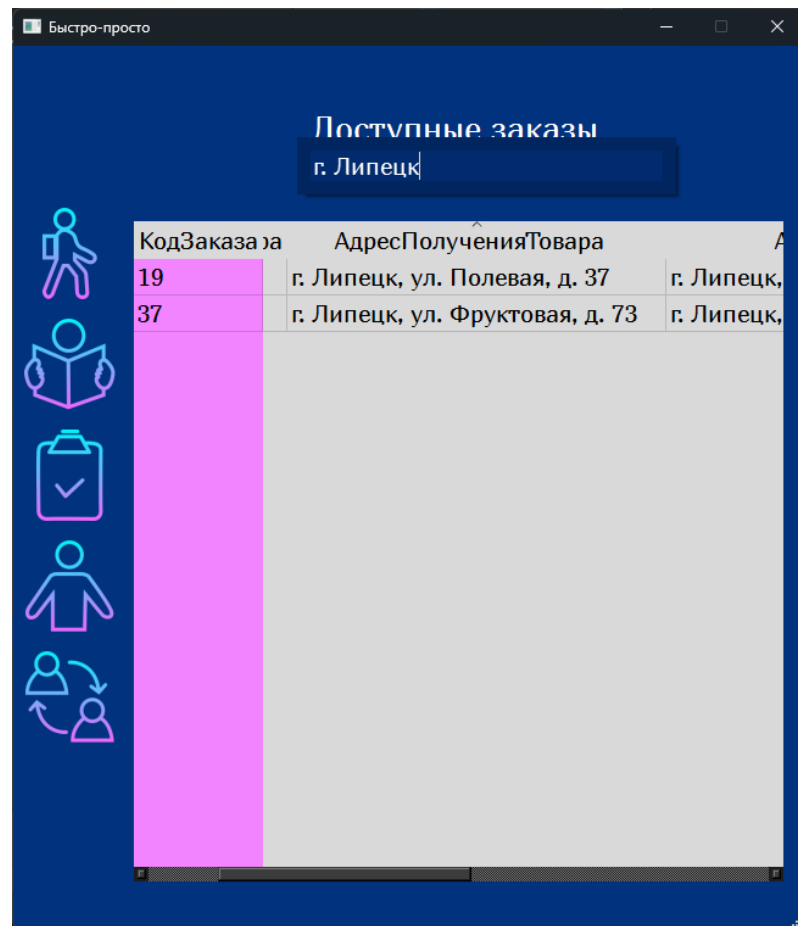


Рис. 2.30 Сортировка по полю «АдресПолученияТовара»

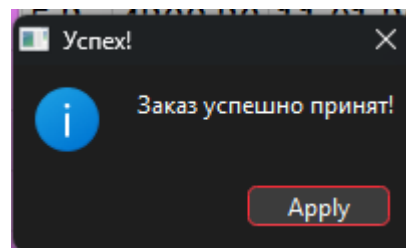



Рис. 2.31 Уведомление о успешности операции

Быстро-просто

### Доступные заказы



КодЗаказа	Вес	Ставка	ДатаЗаказа	
23	5,9	6401,24	01.05.2024	Взять заказ
25	2,9	6871,83	04.12.2023	Взять заказ
27	2,6	11265	25.01.2024	Взять заказ
29	1,4	8095	22.06.2024	Взять заказ
31	5,2	4398,39	11.01.2024	Взять заказ
33	2,1	9800,52	21.11.2023	Взять заказ
35	5,6	5461,34	01.06.2024	Взять заказ
21	2,4	9042,25	30.10.2024	Взять заказ
11	4,3	7012,66	02.01.2024	Взять заказ
7	8	3376,98	14.01.2024	Взять заказ
17	4	6170,28	08.11.2024	Взять заказ
2	3	10599,1	21.06.2024	Взять заказ
19	6,4	4782,1	21.06.2024	Взять заказ
37	2,3	4107,06	02.04.2024	Взять заказ
41	5,4	7304,42	25.08.2024	Взять заказ
43	2,7	4508,26	05.01.2024	Взять заказ
45	6,1	8136,98	16.04.2024	Взять заказ

Рис. 2.32 Заказы, которые невозможно принять в работу

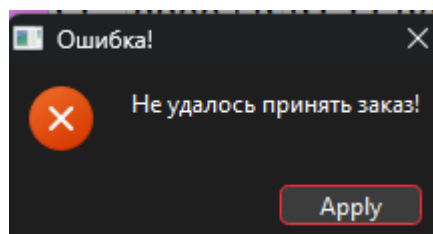


Рис. 2.33 Уведомление о невозможности принять данный заказ в работу

Далее рассмотрим окно с принятыми в работу заказами (рис. 2.34, приложение В, п. 3). Каждый заказ, который находится в данной таблице имеет в кнопки «Завершить заказ», «Отменить заказ» (рис. 2.35).

При нажатии на кнопку «Завершить заказ» возникает соответствующее сообщение (рис. 2.36), и количество выполненных заказов у данного курьера увеличивается на единицу, а дата выполнения заказа устанавливается на текущий день вследствие срабатывания триггера (приложение А, п.10). В случае отмены заказа (кнопка «Отменить заказ») также возникает уведомление (рис. 2.37), и заказ вновь возвращается в список доступных. При

отмене заказа срабатывает триггер «update\_info\_after\_break\_order» (приложение А, п. 9), который удаляет запись о данном заказе в таблице «ВРаботе».



Рис. 2.34 Окно «Принятые заказы»

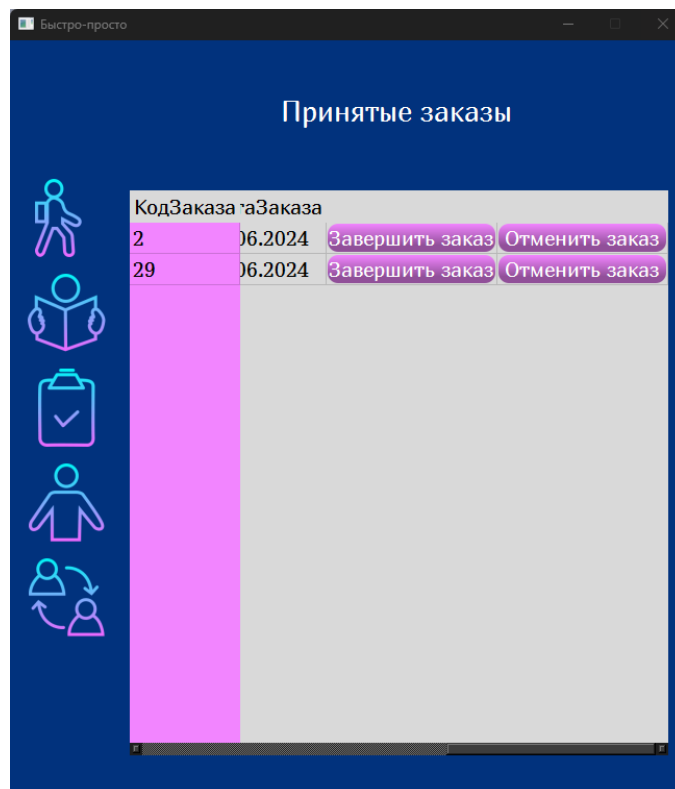


Рис. 2.35 Кнопки взаимодействия с принятым заказом

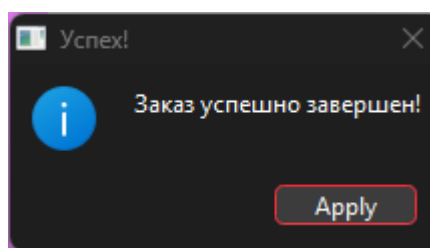


Рис. 2.36 Уведомление о том, что заказ успешно завершен

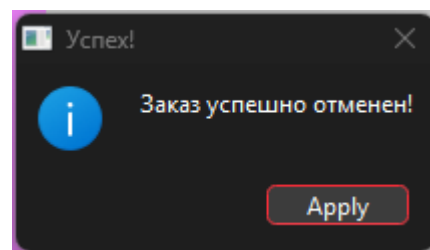


Рис. 2.37 Уведомление о том, что заказ успешно отменен

Также курьер может посмотреть все отзывы, которые были оставлены на его работу, общий рейтинг (среднее арифметическое между всеми оценками пользователей) и количество заказов, которые были выполнены (рис. 2.38, приложение В, п. 3). Элементов взаимодействия в данном окне не предусмотрено.



Рис. 2.38 Окно «Рейтинг»

Рассмотрим работу программы со стороны менеджера. Войти в данный режим возможно только при вводе данных, которые принадлежат пользователю, имеющему определенный номер в системе.

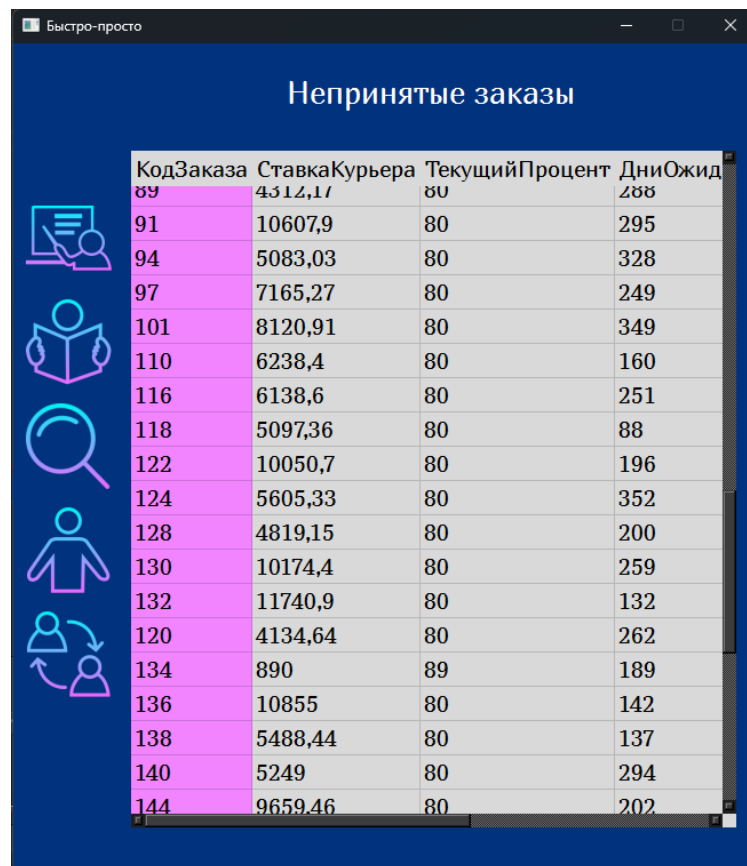
Встречает менеджера окно с таблицей, в которой представлены неприятые в данный момент заказы (рис. 2.39, приложение В, п. 3). Каждый заказ, который находится в данной таблице имеет кнопки «Изменить проц-т», «Смотреть заказ» (рис. 2.40).

Кроме того, в таблице представлены поля, которые необходимы для принятия менеджером решения о повышении ставки на заказ. В случае необходимости к представленному списку заказов можно применить фильтр (рис. 2.41). Если выводимых параметров недостаточно, то имеется кнопка «Смотреть заказ», которая создает новое окно «Информация о заказе» (рис. 2.42, приложение В, п. 7) с иной информацией о выбранном заказе. В данном окне иных функций взаимодействия с заказом не предусмотрено.

Изменить процент для определенного заказа, можно нажатием на кнопку «Изменить проц-т». В данной таблице возникает дополнительное окно «Изменить процент» (рис. 2.43, приложение В, п. 8).

При установке менеджером процента, выходящего за пределы ограничений предметной области, отображается соответствующее уведомление (рис. 2.44).

В случае корректных данных процент, который получает курьер, изменяется и появляется уведомление (рис 2.45).



КодЗаказа	СтавкаКурьера	ТекущийПроцент	ДниОжид
89	4312,17	80	288
91	10607,9	80	295
94	5083,03	80	328
97	7165,27	80	249
101	8120,91	80	349
110	6238,4	80	160
116	6138,6	80	251
118	5097,36	80	88
122	10050,7	80	196
124	5605,33	80	352
128	4819,15	80	200
130	10174,4	80	259
132	11740,9	80	132
120	4134,64	80	262
134	890	89	189
136	10855	80	142
138	5488,44	80	137
140	5249	80	294
144	9659,46	80	202

Рис. 2.39 Окно «Непринятые заказы»

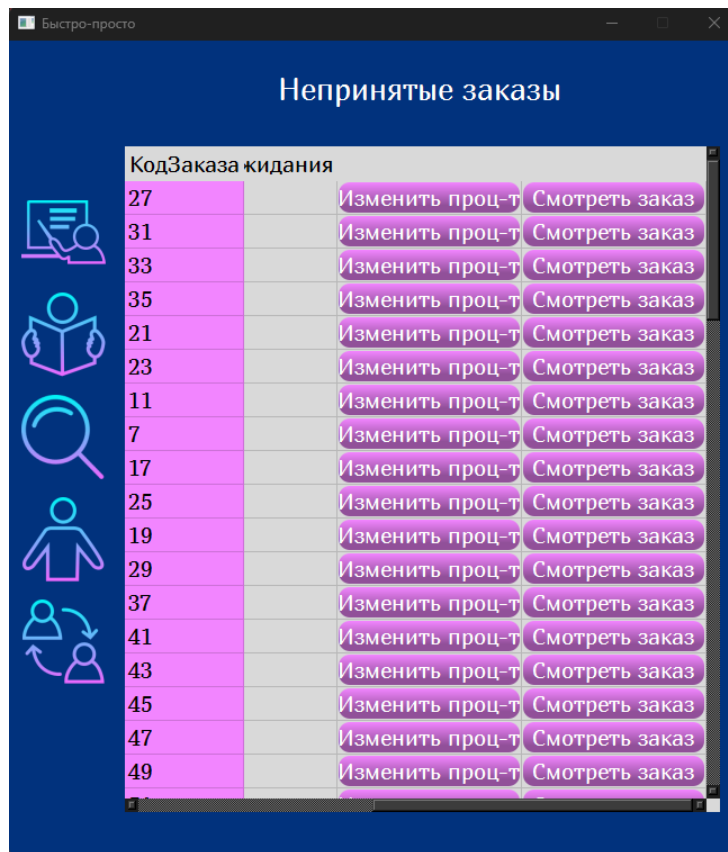


Рис. 2.40 Кнопки для просмотра заказа и редактирования процента курьера

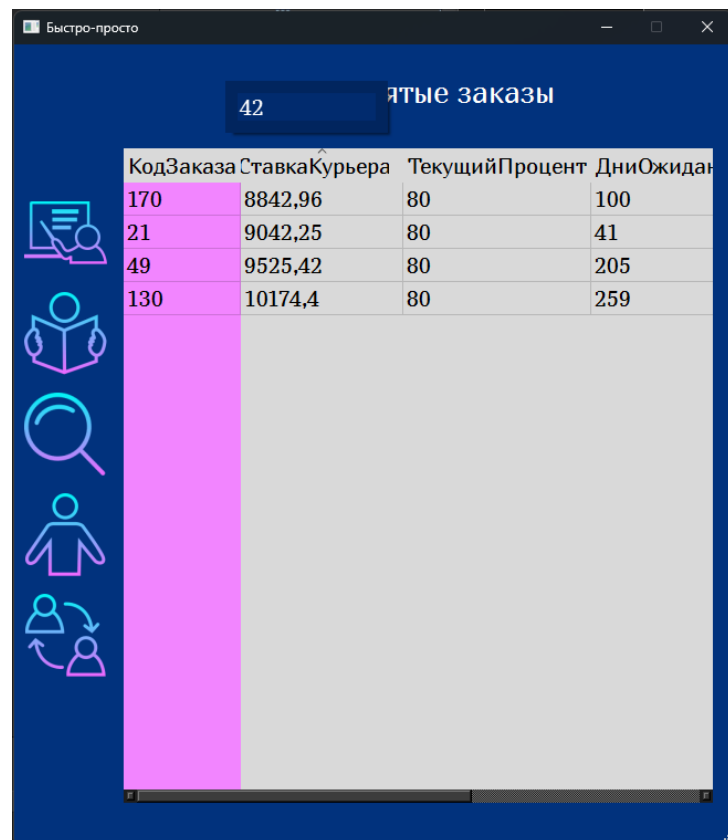


Рис. 2.41 Применение фильтра



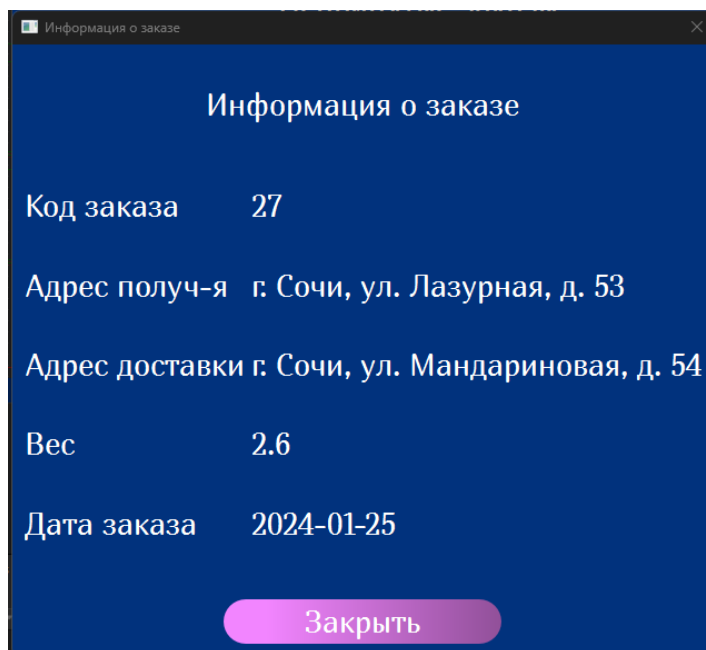


Рис. 2.42 Окно «Информация о заказе»

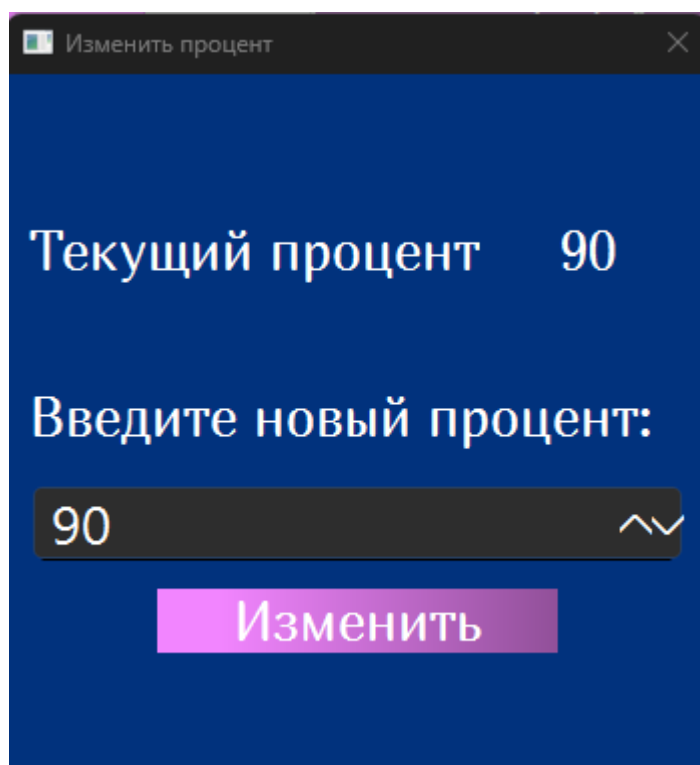


Рис. 2.43 Окно «Изменить процент»

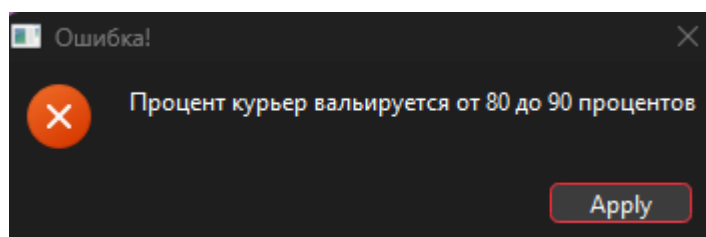


Рис. 2.44 Обработка ошибки введения некорректного значения

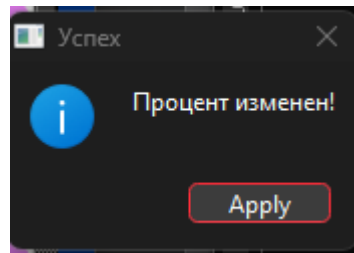


Рис. 2.45 Уведомление об успешном изменении

Рассмотрим окно с аналитикой выполняемости заказов (рис. 2.46, приложение В, п. 3), в которой представлен график с частотой выполнения (зеленая линия) и появления заказов в системе (красная линия) по дням. Данный график сложен в аналитике при изначальном масштабе, но благодаря технологии масштабирования можно увидеть ситуацию за определенный период (рис. 2.47). Иных элементов взаимодействия в данном окне не предусмотрено.

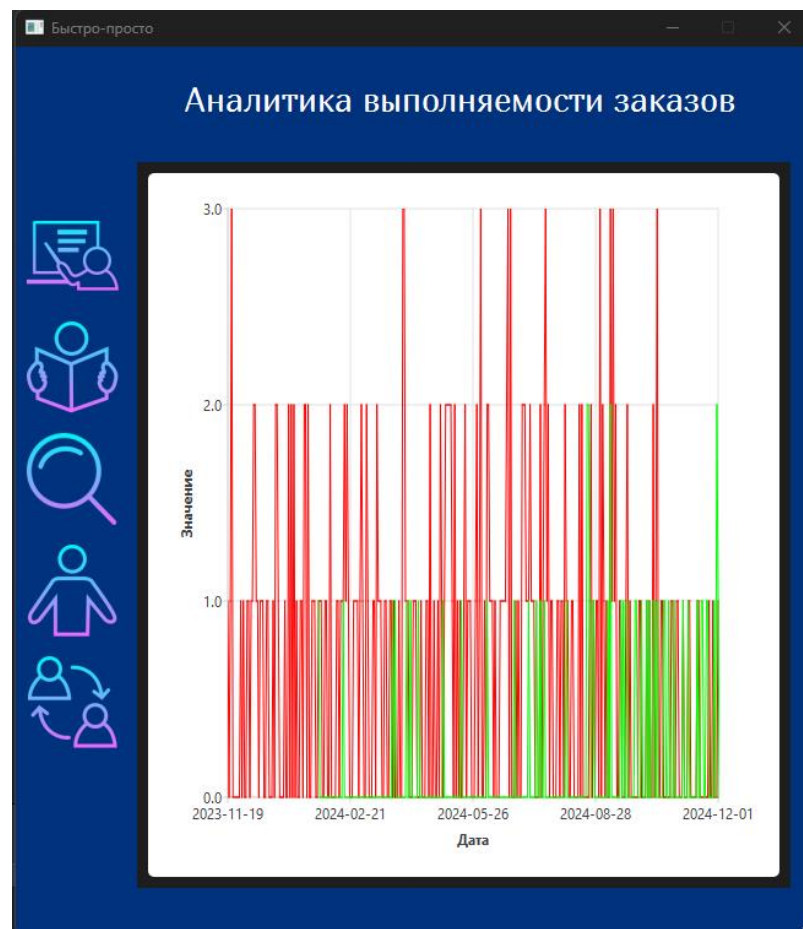


Рис. 2.46 Окно «Аналитика выполняемости заказов»

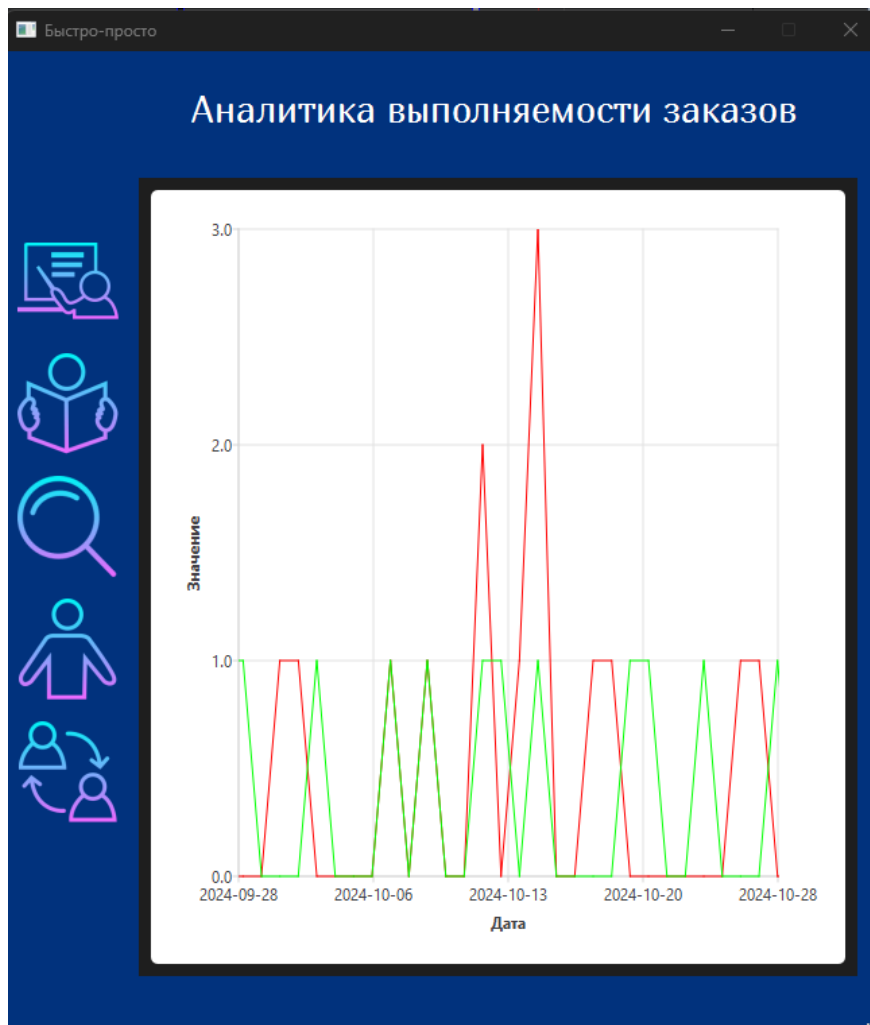
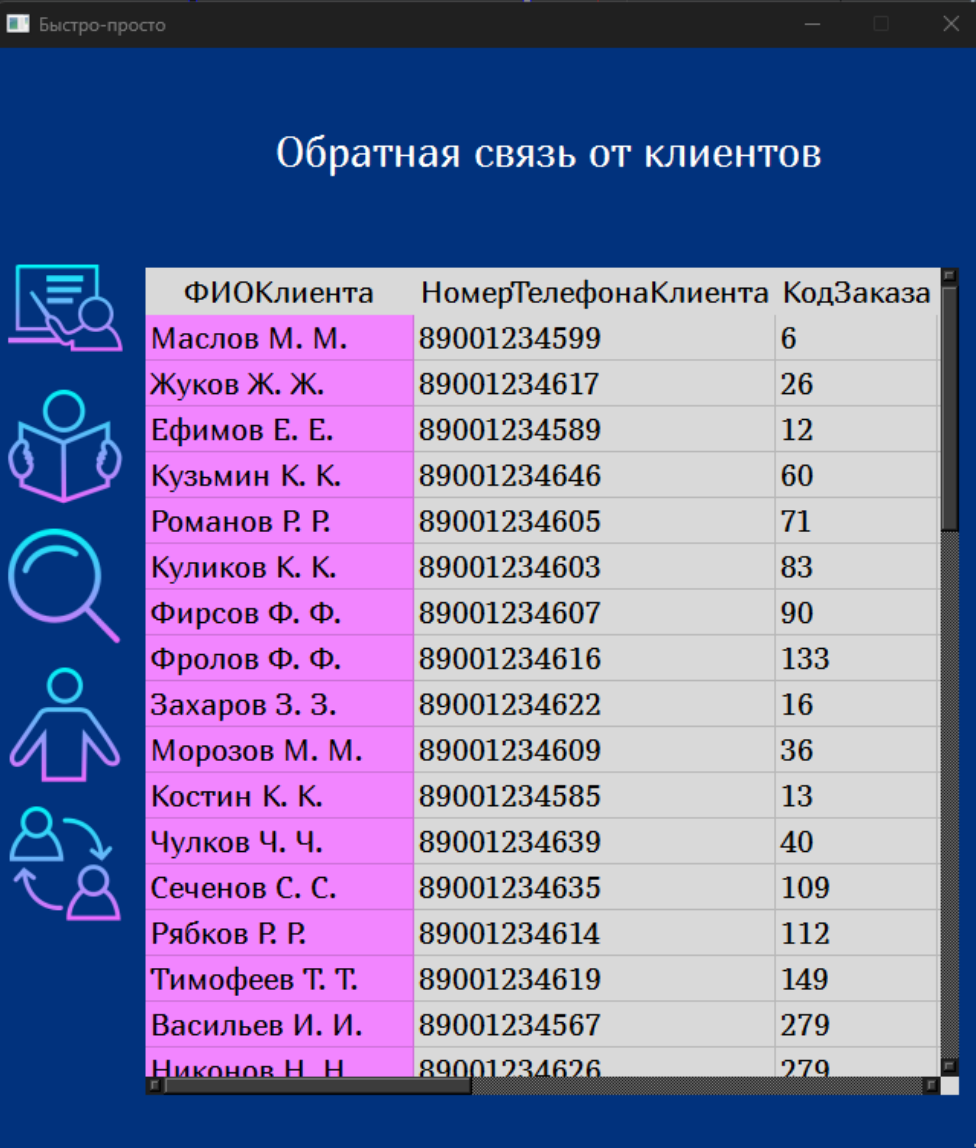


Рис. 2.47 Масштабирование элемента окна

Рассмотрим окно с просмотром обратной связи от клиентов (рис. 2.48, приложение В, п. 3). В таблице, расположенной в данном окне, представлены все отзывы, которые были оставлены пользователями. Если возникли какие-то проблемы во время доставки, то менеджер имеет возможность связаться с неудовлетворенным клиентом.



ФИОКлиента	НомерТелефонаКлиента	КодЗаказа
Маслов М. М.	89001234599	6
Жуков Ж. Ж.	89001234617	26
Ефимов Е. Е.	89001234589	12
Кузьмин К. К.	89001234646	60
Романов Р. Р.	89001234605	71
Куликов К. К.	89001234603	83
Фирсов Ф. Ф.	89001234607	90
Фролов Ф. Ф.	89001234616	133
Захаров З. З.	89001234622	16
Морозов М. М.	89001234609	36
Костин К. К.	89001234585	13
Чулков Ч. Ч.	89001234639	40
Сеченов С. С.	89001234635	109
Рябков Р. Р.	89001234614	112
Тимофеев Т. Т.	89001234619	149
Васильев И. И.	89001234567	279
Никонов Н. Н.	89001234626	279

Рис. 2.48 Окно «Обратная связь от клиентов»

Перейдем к функциям, которые доступны пользователю, независимо от его роли.

Начнем с личного кабинета. В данном окне (рис. 2.49 – 2.51, приложение В, п. 3) можно увидеть роль пользователя, его ФИО и номер телефона. Данные параметры статичны (имеются у каждого пользователя). Если пользователь является курьером, то у него появляется дополнительное изменяемое поле с номер паспорта, если является клиентом, - поле с адресом клиента.

Все вышеописанные параметры можно изменить вводом нового значения соответствующего параметра и нажатием кнопки «Изменить»

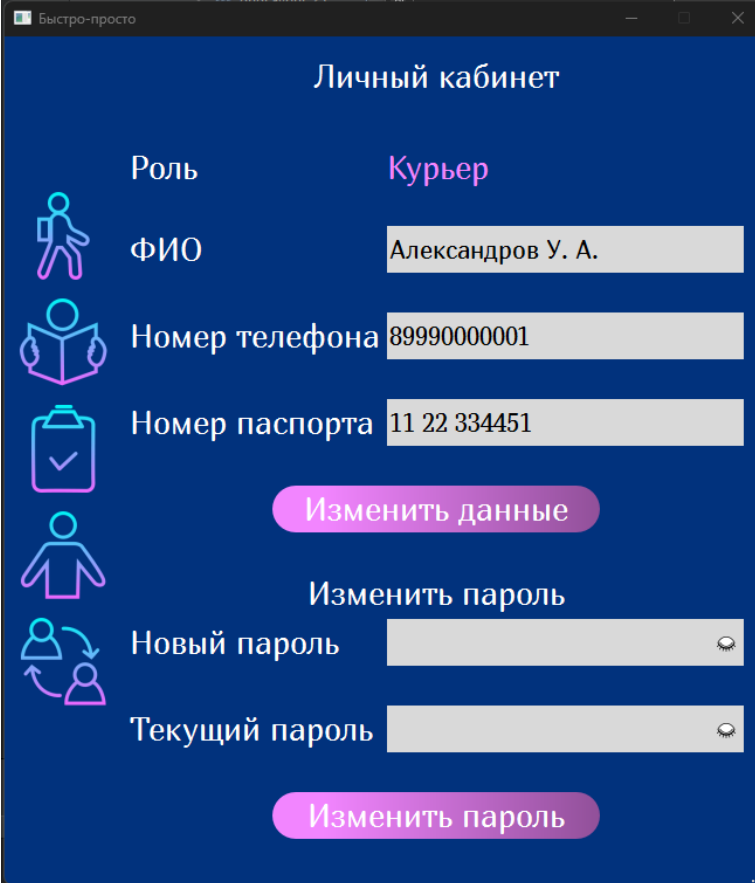
данные». При корректных данных возникнет уведомление об успехе (рис. 2.52).

Если пользователь хочет обновить свой номер телефона на уже существующий в системе, то возникнет ошибка (рис. 2.53).

Кроме того, можно изменить свой пароль для входа в систему. Для этого необходимо ввести свой текущий пароль и новый (который желаете установить).

При условии, что старый пароль указан верно, он будет изменен, и вход в систему будет осуществляется по новому паролю, и появится уведомление о изменении пароля (рис. 2.54), иначе возникнет уведомление об ошибке (рис. 2.55).

Рис. 2.49 Окно «Личный кабинет» для клиента



Быстро-просто

### Личный кабинет

Роль **Курьер**

ФИО Александров У. А.

Номер телефона 89990000001

Номер паспорта 11 22 334451

Изменить данные

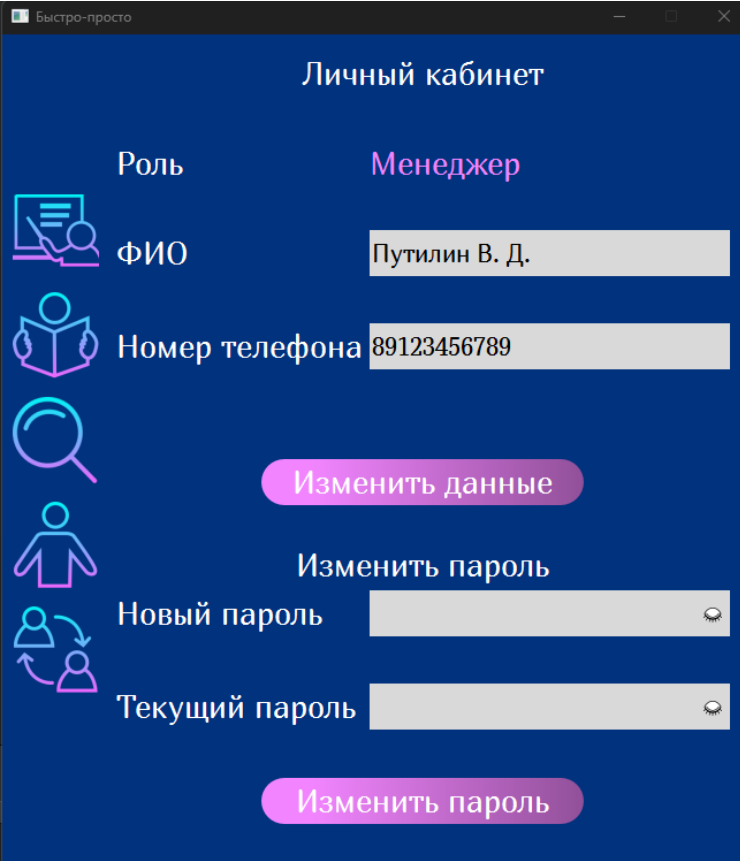
Изменить пароль

Новый пароль

Текущий пароль

Изменить пароль

Рис. 2.50 Окно «Личный кабинет» для курьера



Быстро-просто

### Личный кабинет

Роль **Менеджер**

ФИО Путилин В. Д.

Номер телефона 89123456789

Номер паспорта 11 22 334451

Изменить данные

Изменить пароль

Новый пароль

Текущий пароль

Изменить пароль

Рис. 2.51 Окно «Личный кабинет» для менеджера

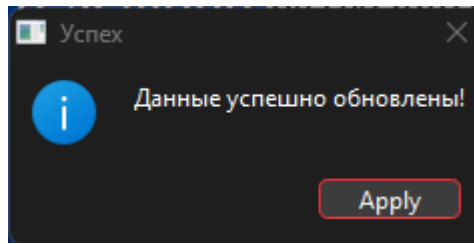


Рис. 2.52 Уведомление об успешном изменении данных

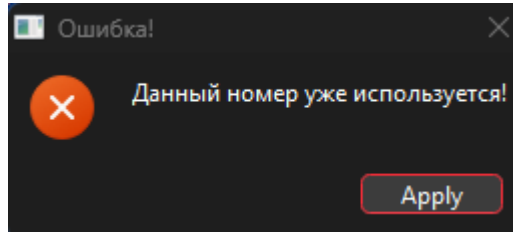


Рис. 2.53 Обработка ошибки ввода существующего номера

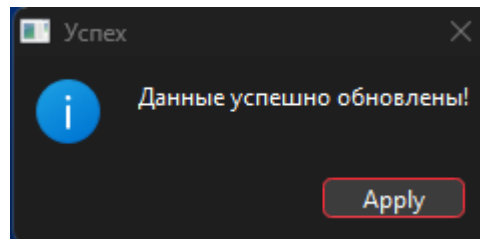


Рис. 2.54 Уведомление об успешном обновлении пароля

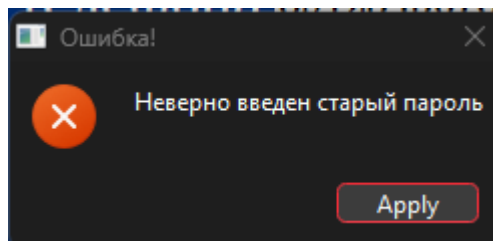


Рис. 2.55 Обработка ошибки некорректно введенного пароля

Следующей функцией, доступной для всех пользователей, является переход в иной аккаунт без закрытия самого приложения. Нажатием на иконку, представленную на рисунке 2.56, пользователь покидает свой текущий аккаунт и переходит на окно с авторизацией (рис. 2. 1).

Стоит отметить, что все данные, которые остаются в интерфейсе после использования системы предыдущим пользователем очищаются, что обеспечивает безопасность пользователей.



Рис. 2.56 Кнопка для перехода в другой аккаунт

Таким образом, разработанный интерфейс предоставляет функциональные возможности для трёх категорий пользователей: клиентов, курьеров и менеджеров. Каждая из этих ролей имеет доступ к специфичным инструментам и функциям, адаптированным под их задачи и обязанности. Клиенты могут оформлять заказы, отслеживать их статус и оставлять отзывы, курьеры — управлять своими доставками и обновлять статус выполнения заказов, а менеджеры обладают возможностью анализа выполняемости заказов.



## ЗАКЛЮЧЕНИЕ

Основной темой данного является проектирование БД курьерской службы.

Актуальность темы создания базы данных для курьерской службы высока и увеличивается с каждым днем, учитывая постоянный рост спроса на услуги доставки. В данный момент многие компании перешли на онлайн-продажи, что требуют надежной и быстрой доставки товаров. Также стоит учитывать, что рынок полнится и более мелкими заказами от частных лиц.

Для создания БД были выполнены четыре необходимых этапа: анализ предметной области, концептуальное, логическое и физическое проектирования.

Анализ предметной области позволил собрать информацию о уже имеющихся решениях и на их основе выделить те функций, которые необходимо реализовать. Для анализа рассматривались следующие ПО: сайт курьерской службы «Dostavista», а также мобильное приложение «Додо Пицца». Каждый из рассматриваемых продуктов имел свои особенности и возможности.

Концептуальное проектирование затрагивает такие аспекты, как описание модели предметной области в общем виде. В процессе проектирования были выделены такие действующие лица, как клиент, курьер и менеджер, а также перечень функций, которые доступны каждому лицу. На основе анализа ДВИ был определен общий список объектов, о которых необходимо хранить информацию.

Логическое проектирование является продолжением концептуального. На основе уже имеющихся сущностей, их взаимосвязей и существующих правил преобразования мы создаем полноценный список отношений, в котором имеются как ключевые, так и неключевые атрибуты. Взаимосвязи между объектами описываются с помощью ER-диаграмм. В результате выполнения данного этапа проектирования было построено пять ER-диаграмм, которые описывают различные взаимодействия между объектами.

Отдельной задачей является определение ограничений предметной области. В результате мы получаем свод правил, используемых в конкретной предметной области.

На этапе физического проектирования были сформулированы требования к структурам таблиц базы данных. С использованием системы управления базами данных PostgreSQL и её графического интерфейса PgAdmin [10] были созданы данные таблицы, настроены их взаимосвязи, осуществлено их наполнение, а также разработаны дополнительные объекты базы данных (процедуры, функции, триггеры и представления). В результате сформировалась полноценная база данных.

На последнем этапе выполнения проекта был разработан интерфейс для взаимодействия трех действующих лиц (клиента, курьера и менеджера) с созданной базой данных. Для создания данной программы использовался фреймворк QT с использованием языка C++.

Функции, которые были определены на концептуальном этапе проектирования БД, нашли отражение в данном интерфейсе для полноценного взаимодействия пользователей с БД. Кроме того, были реализованы следующие функции, которые упрощают работу с приложением:

1. Возможность регистрации в системе.
2. Возможность сменить аккаунт без закрытия самого приложения.

## СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Сайт «WB курьер» [Электронный ресурс] – Режим доступа: <https://wbk.wb.ru/>, свободный. Дата обращения: 11.03.2023 г.
2. Сайт «EdiCourier: Конструктор для организации доставки» [Электронный ресурс] – Режим доступа: <https://ediweb.com/ru-ru/solutions/paas/edicourier>, свободный. Дата обращения: 11.03.2024 г.
3. Сайт «Dostavista» [Электронный ресурс] – Режим доступа: <https://dostavista.ru/>, свободный. Дата обращения: 11.03.2024 г.
4. Сайт «Додо Пицца» [Электронный ресурс] – Режим доступа: <https://dodopizza.ru/>, свободный. Дата обращения: 11.03.2024 г.
5. Сайт «Документация PostgreSQL» [Электронный ресурс] – Режим доступа: <https://www.postgresql.org/docs/>, свободный. Дата обращения: 01.10.2024 г.
6. Сайт «Документация QT» [Электронный ресурс] – Режим доступа: <https://doc.qt.io/>, свободный. Дата обращения: 15.10.2024 г.
7. Сайт «Документация QT Creator» [Электронный ресурс] – Режим доступа: <https://doc.qt.io/qtcreator/index.html>, свободный. Дата обращения: 15.10.2024 г.
8. Сайт «Документация API Яндекс Карты» [Электронный ресурс] – Режим доступа: <https://yandex.ru/maps-api/docs>, свободный. Дата обращения: 20.11.2024 г.
9. Сайт «Документация Distance Matrix API» [Электронный ресурс] – Режим доступа: <https://distancematrix.ai/distance-matrix-api>, свободный. Дата обращения: 22.11.2024 г.
10. Сайт «Документация PgAdmin» [Электронный ресурс] – Режим доступа: <https://www.pgadmin.org/docs/>, свободный. Дата обращения: 01.10.2024 г.

**SQL-операторы создания объектов БД****1. Создание таблицы «Клиент»**

```
CREATE TABLE Клиент (  
КодКлиента SMALLINT PRIMARY KEY,  
ФИО VARCHAR (30) NOT NULL,  
НомерТелефона VARCHAR (20) NOT NULL UNIQUE,  
ДатаРождения DATE NOT NULL CHECK ((CURRENT_DATE -  
ДатаРождения) / 365 >= 14),  
АдресКлиента VARCHAR(50) NOT NULL)
```

**2. Создание таблицы «Заказ»**

```
CREATE TABLE Заказ (  
КодЗаказа SMALLINT PRIMARY KEY,  
КодКлиента SMALLINT NOT NULL REFERENCES Клиент (КодКлиента),  
НазваниеТарифа VARCHAR (10) NOT NULL REFERENCES Тариф  
(НазваниеТарифа),  
Вес FLOAT NOT NULL CHECK (Вес > 0.0),  
АдресПолученияТовара VARCHAR (50) NOT NULL,  
АдресДоставки VARCHAR (50) NOT NULL,  
ФИОПолучателя VARCHAR (30),  
НомерТелефонаПолучателя VARCHAR (20),  
СтатусЗаказа VARCHAR (10) NOT NULL DEFAULT 'Не принят' CHECK  
(СтатусЗаказа = 'Не принят' OR СтатусЗаказа = 'В работе' OR СтатусЗаказа =  
'Выполнен'),  
НомерПоступления SMALLINT NOT NULL UNIQUE,  
ПроцентКурьера INT NOT NULL DEFAULT 80 CHECK (ПроцентКурьера > 0  
AND ПроцентКурьера < 100),  
ДатаЗаказа DATE NOT NULL DEFAULT CURRENT_DATE,  
СтоимостьЗаказа DECIMAL (10, 2) NOT NULL CHECK (СтоимостьЗаказа >  
0.0),  
ДатаВыполнения DATE CHECK(ДатаВыполнения >= ДатаЗаказа))
```

**3. Создание таблицы «Тариф»**

```
CREATE TABLE Тариф (  
НазваниеТарифа VARCHAR (10) PRIMARY KEY,  
Цена SMALLINT NOT NULL CHECK (Цена > 0))
```

**4. Создание таблицы «Курьер»**

```
CREATE TABLE Курьер (  
КодКурьера SMALLINT PRIMARY KEY,
```

ФИО VARCHAR (30) NOT NULL,  
 ДатаРегистрации DATE NOT NULL DEFAULT CURRENT\_DATE,  
 ДатаРождения DATE NOT NULL CHECK ((CURRENT\_DATE -  
 ДатаРождения) / 365 >= 18),  
 КоличествоВыполненныхЗаказов SMALLINT NOT NULL DEFAULT 0  
 CHECK (КоличествоВыполненныхЗаказов >= 0),  
 ДанныеПаспорта VARCHAR (20) NOT NULL UNIQUE,  
 НомерТелефона VARCHAR (20) NOT NULL UNIQUE)

#### 5. Создание таблицы «ВРаботе»

```
CREATE TABLE Клиент (
КодЗаказа SMALLINT NOT NULL REFERENCES Заказ (КодЗаказа),
КодКурьера SMALLINT NOT NULL REFERENCES Курьер (КодКурьера),
PRIMARY KEY (КодЗаказа, КодКурьера))
```

#### 6. Создание таблицы «Отзыв»

```
CREATE TABLE Отзыв (
КодОтзыва SMALLINT PRIMARY KEY,
КодКлиента SMALLINT NOT NULL REFERENCES Клиент (КодКлиента),
КодЗаказа SMALLINT NOT NULL REFERENCES Заказ (КодЗаказа),
Описание VARCHAR (200),
Оценка SMALLINT NOT NULL DEFAULT 1 CHECK (Оценка >= 1 AND
Оценка <= 5))
```

#### 7. Создание триггера «trg\_check\_courier\_order»

```
CREATE TRIGGER trg_check_courier_order
BEFORE INSERT ON ВРаботе
FOR EACH ROW
EXECUTE FUNCTION trg_func_check_couriers_orders();
```

#### 8. Создание триггера «check\_weight\_trigger»

```
CREATE TRIGGER check_weight_trigger
BEFORE INSERT ON Заказ
FOR EACH ROW
EXECUTE FUNCTION express_order_check();
```

#### 9. Создание триггера «update\_info\_after\_break\_order»

```
CREATE TRIGGER update_info_after_break_order
AFTER UPDATE ON Заказ
FOR EACH ROW
WHEN (new.СтатусЗаказа = 'Не принят' AND old.СтатусЗаказа = 'В работе')
EXECUTE FUNCTION update_vrabote_after_break_order();
```

#### 10. Создание триггера «update\_info\_after\_ending\_order»

```
CREATE TRIGGER update_info_after_ending_order
AFTER UPDATE ON Заказ
FOR EACH ROW
WHEN (new.СтатусЗаказа = 'Выполнен' AND old.СтатусЗаказа = 'В работе')
EXECUTE FUNCTION trg_delete_vrabote();
```

#### 11. Создание триггера «trg\_check\_unique\_review»

```
CREATE TRIGGER trg_check_unique_review
BEFORE INSERT ON ОТЗЫВ
FOR EACH ROW
EXECUTE FUNCTION check_unique_review();
```

#### 12. Триггерная функция «trg\_func\_check\_couriers\_orders»

```
CREATE OR REPLACE FUNCTION update_completed_at()
RETURNS TRIGGER AS $$
BEGIN
    IF (SELECT НазваниеТарифа FROM Заказ WHERE КодЗаказа =
NEW.КодЗаказа) = 'Экспресс' THEN
        IF (SELECT COUNT(*) FROM ВРаботе INNER JOIN Заказ ON
ВРаботе.КодЗаказа = Заказ.КодЗаказа WHERE ВРаботе.КодКурьера =
NEW.КодКурьера AND Заказ.НазваниеТарифа = 'Экспресс' AND
Заказ.СтатусЗаказа = 'В работе') = 1 THEN
            RAISE EXCEPTION 'Больше заказов "Экспресс" брать
нельзя!';
        END IF;
    ELSE
        IF (SELECT COUNT(*) FROM ВРаботе INNER JOIN Заказ ON
ВРаботе.КодЗаказа = Заказ.КодЗаказа
        WHERE ВРаботе.КодКурьера = NEW.КодКурьера AND
Заказ.СтатусЗаказа = 'В работе') >= 2 THEN
            RAISE EXCEPTION 'Максимум доступно 2 заказа!';
        END IF;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

#### 13. Триггерная функция «express\_order\_check»

```
CREATE OR REPLACE FUNCTION express_order_check()
RETURNS TRIGGER AS $$
BEGIN
    if NEW.НазваниеТарифа = 'Экспресс' AND NEW.Вес > 10.0 THEN
        RAISE EXCEPTION 'Вес не может превышать 10.0 для
тарифа Экспресс';
```

```

        ROLLBACK;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

#### 14. Триггерная функция «update\_vrabote\_after\_break\_order»

```

CREATE OR REPLACE FUNCTION update_vrabote_after_break_order ()
RETURNS TRIGGER AS $$
BEGIN
    DELETE FROM ВРаботе
    WHERE КодЗаказа = OLD.КодЗаказа;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

#### 15. Триггерная функция «trg\_delete\_vrabote»

```

CREATE OR REPLACE FUNCTION trg_delete_vrabote()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Курьер SET КоличествоВыполненныхЗаказов =
    КоличествоВыполненныхЗаказов +1
    WHERE КодКурьера = (SELECT КодКурьера FROM ВРаботе
    WHERE КодЗаказа = OLD.КодЗаказа);

    UPDATE Заказ SET ДатаСдачи = CURRENT_DATE
    WHERE КодЗаказа = OLD.КодЗаказа;
    RETURN NEW;
END; $$ LANGUAGE plpgsql;

```

#### 16. Триггерная функция «check\_unique\_review»

```

CREATE OR REPLACE FUNCTION trg_delete_vrabote()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Курьер SET КоличествоВыполненныхЗаказов =
    КоличествоВыполненныхЗаказов +1
    WHERE КодКурьера = (SELECT КодКурьера FROM ВРаботе
    WHERE КодЗаказа = OLD.КодЗаказа);

    UPDATE Заказ SET ДатаСдачи = CURRENT_DATE
    WHERE КодЗаказа = OLD.КодЗаказа;
    RETURN NEW; -- Возвращаем новую строку
END; $$ LANGUAGE plpgsql;

```

## 17. Процедура «courier\_take\_order»

```

CREATE OR REPLACE PROCEDURE courier_take_order (Код_Курьер INT,
Код_Заказ INT)
LANGUAGE plpgsql
AS $$
BEGIN
    -- Вставляем данные в таблицу ВРаботе
    INSERT INTO ВРаботе (КодЗаказа, КодКурьера)
    VALUES (Код_Заказ, Код_Курьер);
    -- Обновляем статус заказа на 'В работе'
    UPDATE Заказ
    SET СтатусЗаказа = 'В работе'
    WHERE КодЗаказа = Код_Заказ;
END;
$$;

```

## 18. Представление «ИсторияПользователей»

```

CREATE OR REPLACE VIEW ИсторияПользователей
AS
SELECT Заказ.КодКлиента, Заказ.НазваниеТарифа,
       Заказ.КодЗаказа, Заказ.СтатусЗаказа, Заказ.АдресПолученияТовара,
       Заказ.АдресДоставки, Заказ.ДатаЗаказа, Заказ.ДатаСдачи
FROM Заказ;

```



## ПРИЛОЖЕНИЕ Б

## Заполнение таблиц данными

	КодКлиента [PK] smallint	ФИО character varying (30)	НомерТелефона character varying (20)	ДатаРождения date	Пароль character varying (50)	АдресКлиента character varying (100)
1	1	Васильев И. И.	89001234567	2005-01-15	1234	г. Москва, ул. Беговая, д. 52
2	2	Петров П. П.	89001234568	2003-02-20	23245	г. Москва, ул. 2-ая Фрунзенская, д. 7
3	3	Сидоров С. С.	89001234569	2004-03-25	12321	[null]
4	4	Кузнецов К. К.	89001234570	2006-04-10	qwewq	[null]
5	5	Смирнов С. С.	89001234571	2002-05-12	qwewq	[null]

Рис Б.1 Данные таблицы «Клиент»

	КодКурьера [PK] smallint	ФИО character varying (30)	ДатаРегистрации date	ДатаРождения date	КоличествоВыполненныхЗаказов smallint	ДанныеПаспорта character varying (20)	НомерТелефона character varying (20)
1	1	Александров У. А.	2023-01-01	1990-02-20	22	11 22 334455	89990000001
2	2	Борисова Б. С.	2023-01-02	1991-05-15	0	22 33 445566	89990000002
3	3	Васильев С. В.	2023-01-03	1989-07-10	0	33 44 556677	89990000003
4	4	Григорьев Г. Г.	2023-01-04	1992-11-22	0	44 55 667788	89990000004
5	5	Дмитриев Д. У.	2023-01-05	1990-09-18	0	55 66 778899	89990000005

Рис Б.2 Данные таблицы «Курьер»

	НазваниеТарифа [PK] character varying (10)	Цена smallint
1	Документы	1500
2	Основной	2200
3	Продукты	1500
4	Профи	5500
5	Экспресс	4000

Рис Б.3 Данные таблицы «Тариф»

	КодОтзыва [PK] smallint	Описание character varying (200)	Оценка smallint	КодКлиента smallint	КодЗаказа smallint
1	7	Курьер был вежлив и аккуратен.	5	7	3
2	8	Доставка задержалась на час, но это не критично.	3	50	137
3	19	Опоздание на полтора часа, не очень удобно.	2	19	13
4	23	Опоздание на весь день.	1	23	12
5	24	Курьер был очень вежлив.	5	24	46
6	29	Доставка была отличной, всё соблюдено.	4	29	20

Рис Б.4 Данные таблицы «Отзыв»

	КодЗаказ [PK] smallint	КодКлиента smallint	НазваниеТовара character varying (50)	Вес double	АдресПолученияТовара character varying (50)	АдресДоставки character varying (50)	ФИОПолучателя character varying (30)	НомерТелефона character varying (15)	СтатусЗаказа character varying (10)	НомерПлатежа smallint	Процент integer	ДатаЗаказа date	Стоимость numeric (10, 2)	ДатаСдачи date
1	1	12	Основной	5	г. Москва, ул. Пушкин...	г. Санкт-Петербург, ул. Ле...	Петров Д. В.	82345678901	В работе	1001	80	2024-02-11	5111.07	[null]
2	2	45	Экспресс	3	г. Казань, ул. Чехова, ...	г. Казань, ул. Гоголя, д. 4	Сидоров А. И.	89876543214	Не принят	1002	80	2024-06-21	13248.85	[null]
3	3	7	Продукты	1.5	г. Ростов-на-Дону, ул. ...	г. Ростов-на-Дону, ул. Дос...	Иванов И. И.	81122334453	Выполнен	1003	80	2024-03-16	13861.98	2024-11-...
4	4	88	Экспресс	2	г. Новосибирск, ул. Пр...	г. Новосибирск, ул. Невск...	Смирнов П. В.	82233445566	Выполнен	1004	80	2023-11-22	6235.34	2024-06-...
5	5	21	Основной	4.2	г. Екатеринбург, ул. М...	г. Екатеринбург, ул. Блок...	Кузнецов Н. С.	[null]	Выполнен	1005	80	2024-10-14	6239.53	2024-11-...
6	6	33	Экспресс	6	г. Новороссийск, ул. О...	г. Новороссийск, ул. Пуш...	Федоров В. А.	84455667782	Выполнен	1006	80	2024-08-06	21312.23	2024-10-...
7	7	54	Основной	8	г. Челябинск, ул. Салт...	г. Челябинск, ул. Куприна...	Яковлев И. Н.	85566778893	Не принят	1007	80	2024-01-14	4221.23	[null]
8	8	90	Экспресс	2.5	г. Уфа, ул. Набережна...	г. Уфа, ул. Полтавская, д. ...	Лебедев А. Д.	86677889902	В работе	1008	80	2024-05-20	5000.00	[null]

Рис Б.5 Данные таблицы «Заказ»

	КодЗаказа [PK] smallint	КодКурьера [PK] smallint
1	1	4
2	3	2
3	4	9
4	6	126
5	8	2
6	10	87
7	12	35
8	13	1
9	14	4
10	16	10
11	18	5

Рис Б.6 Данные таблицы «ВРаботе»

**Код программы****1. Программный код окна «Регистрация»**

```

//registrationform.h
#ifndef REGISTRATIONFORM_H
#define REGISTRATIONFORM_H
#include <QDialog>
#include "optionbutton.h"
#include "passwordline.h"
#include "roles.h"
#include <QDate>

namespace Ui {
class registrationForm;
}

class registrationForm : public QDialog
{
    Q_OBJECT

public:
    explicit registrationForm(QWidget *parent = nullptr);
    ~registrationForm();
    optionButton* button_controller;
private slots:
    void on_pushButton_3_clicked();

private:
    Role user_role;
    bool check_date(QDate date);
    void init();
    bool telephone_is_unique(QString telephone);
    PasswordLine* password;
    Ui::registrationForm *ui;
};

#endif // REGISTRATIONFORM_H
//registrationform.cpp

#include "registrationform.h"
#include "ui_registrationform.h"
#include "stylehelper.h"

```

```

#include <QSqlQuery>
#include <QMessageBox>

registrationForm::registrationForm(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::registrationForm)
{
    ui->setupUi(this);
    button_controller = new optionButton();
    password = new PasswordLine(this);
    button_controller->addButton(ui->pushButton);
    button_controller->addButton(ui->pushButton_2);
    connect(ui->pushButton_2, &QPushButton::clicked,
        this, [&]()
        {
            ui->label_6->setVisible(true);
            ui->lineEdit_3->setVisible(true);
            user_role = Role::COURIER;
        });
    connect(ui->pushButton, &QPushButton::clicked,
        this, [&]()
        {
            ui->label_6->setVisible(false);
            ui->lineEdit_3->setVisible(false);
            user_role = Role::USER;
        });
    init();
}

registrationForm::~registrationForm()
{
    delete ui;
    delete button_controller;
}

void registrationForm::init()
{
    //set style option
    this->setStyleSheet(styleHelper::addTextStyle());
    ui->pushButton->setStyleSheet(styleHelper::addPushButtonStyle2());
    ui->pushButton_2->setStyleSheet(styleHelper::addPushButtonStyle2());
    password->setMinimumSize(QSize(0, 40));
    password->setMaximumSize(QSize(1111, 40));
    password->setFont(QFont("Marmelad", 16));
    ui->gridLayout_2->addWidget(password, 11, 2, 1, 2);
}

```

```

ui->pushButton_3->setStyleSheet(styleHelper::addPushButtonStyle());
ui->label->setStyleSheet(styleHelper::addProjectFont("white"));
ui->label_2->setStyleSheet(styleHelper::addProjectFont("white"));
ui->label_3->setStyleSheet(styleHelper::addProjectFont("white"));
ui->label_4->setStyleSheet(styleHelper::addProjectFont("white"));
ui->label_5->setStyleSheet(styleHelper::addProjectFont("white"));
ui->label_6->setStyleSheet(styleHelper::addProjectFont("white"));
ui->dateEdit->setStyleSheet(styleHelper::addTextStyle());
QPalette palette = this->palette();
QBrush brush(QColor(1, 50, 125));
palette.setBrush(QPalette::Window, brush);
setPalette(palette);
ui->label_6->setVisible(false);
ui->lineEdit_3->setVisible(false);

//set functional option
ui->lineEdit_2->setClearButtonEnabled(true);
ui->lineEdit->setClearButtonEnabled(true);
ui->lineEdit_3->setClearButtonEnabled(true);
password->setClearButtonEnabled(true);
ui->lineEdit_2->setInputMask(QString("800000000000"));
ui->lineEdit_3->setInputMask(QString("00 00 000000"));
}

bool registrationForm::telephone_is_unique(QString telephone)
{
    QSqlQuery qry;
    qry.exec("SELECT НомерТелефона FROM Курьер UNION SELECT
НомерТелефона FROM Клиент;");
    while(qry.next())
    {
        if(qry.value(0).toString() == telephone)
            return false;
    }
    return true;
}

void registrationForm::on_pushButton_3_clicked()
{
    if(ui->lineEdit->text() == "" or ui->lineEdit_2->text().length() != 11 or password-
>text() == "")
    {
        QMessageBox::critical(this, "Ошибка! ", "Не все данные формы
заполнены!", QMessageBox::Apply);
        return;
    }
}

```

```

    }
    if(!telephone_is_unique(ui->lineEdit_2->text()))
    {
        QMessageBox::critical(this, "Ошибка! ", "Данный номер уже
зарегистрирован!", QMessageBox::Apply);
        return;
    }
    if (user_role == Role::USER)
    {
        if(!check_date(ui->dateEdit->date()))
        {
            QMessageBox::critical(this, "Ошибка! ", "Регистрация для пользователя
возможна с 14 лет!", QMessageBox::Apply);
            return;
        }
        QSqlQuery qry;
        qry.prepare("INSERT INTO Клиент(ФИО, НомерТелефона,
ДатаРождения, Пароль) VALUES(:fio, :tel, :date, :pass)");
        qry.bindValue(":fio", ui->lineEdit->text());
        qry.bindValue(":tel", ui->lineEdit_2->text());
        qry.bindValue(":date", ui->dateEdit->date().toString("yyyy-MM-dd"));
        qry.bindValue(":pass", password->text());
        if(!qry.exec())
        {
            QMessageBox::critical(this, "Ошибка! ", "Не удалось добавить аккаунт!",
QMessageBox::Apply);
        }
        else
        {
            QMessageBox::information(this, "Успех ", "Аккаунт добавлен!",
QMessageBox::Apply);
            this->close();
            delete this;
        }
        return;
    }
    else if(user_role == Role::COURIER)
    {
        if(!check_date(ui->dateEdit->date()))
        {
            QMessageBox::critical(this, "Ошибка! ", "Регистрация для курьера
возможна с 18 лет!", QMessageBox::Apply);
            return;
        }
        if(ui->lineEdit_3->text() == "" or ui->lineEdit_3->text().length() != 12)

```

```

    {
        QMessageBox::critical(this, "Ошибка! ", "Данные паспорта не
заполнены!", QMessageBox::Apply);
        return;
    }
    QSqlQuery qry;
    qry.prepare("INSERT INTO Курьер(ФИО, ДатаРождения,
ДанныеПаспорта, НомерТелефона, Пароль) VALUES(:fio, :date, :passport, :tel,
:pass)");
    qry.bindValue(":fio", ui->lineEdit->text());
    qry.bindValue(":tel", ui->lineEdit_2->text());
    qry.bindValue(":date", ui->dateEdit->date().toString("yyyy-MM-dd"));
    qry.bindValue(":pass", password->text());
    qry.bindValue(":passport", ui->lineEdit_3->text());
    if(!qry.exec())
    {
        QMessageBox::critical(this, "Ошибка! ", "Не удалось добавить аккаунт!",
QMessageBox::Apply);
    }
    else
    {
        QMessageBox::information(this, "Успех ", "Аккаунт добавлен!",
QMessageBox::Apply);
        //this->close();
        //delete this;
    }
    return;
}
    QMessageBox::critical(this, "Ошибка! ", "Роль не выбрана!",
QMessageBox::Apply);
}

```

```

bool registrationForm::check_date(QDate date)
{
    int daysDiff = date.daysTo(QDate::currentDate());
    double yearsDiff = daysDiff / 365.25;
    if (yearsDiff < 14 and user_role == Role::USER)
        return false;
    else if (yearsDiff < 18 and user_role == Role::COURIER)
        return false;
    return true;
}

```

## 2. Программный код окна «Авторизация»

//authorizationform.h

```

#ifndef AUTHORIZATIONFORM_H
#define AUTHORIZATIONFORM_H

#include <QWidget>
#include "passwordline.h"
#include "mainwindow.h"

namespace Ui {
class AuthorizationForm;
}

class AuthorizationForm : public QWidget
{
    Q_OBJECT
public:
    explicit AuthorizationForm(MainWindow *parent = nullptr);
    ~AuthorizationForm();

protected:
    void paintEvent(QPaintEvent* event) override;
private slots:
    void on_enter_button_clicked();
private:
    Ui::AuthorizationForm *ui;
    void addBackgroundGradient();
    void setWindowStyle();
    PasswordLine* passwordLine;
signals:
    void RoleDefine(Role, int);
};

#endif // AUTHORIZATIONFORM_H

// authorizationform.cpp
#include "authorizationform.h"
#include "ui_authorizationform.h"
#include <QSqlError>
#include <QSqlQuery>
#include "stylehelper.h"
#include "passwordline.h"
#include <QMessageBox>
#include "registrationform.h"

AuthorizationForm::AuthorizationForm(MainWindow *parent)
    : QWidget(nullptr)

```



```

, ui(new Ui::AuthorizationForm)
{
    ui->setupUi(this);
    setWindowStyle();
    connect(ui->pushButton, &QPushButton::clicked, this, [this]()
        {
            registrationForm* form = new registrationForm();
            form->exec();
        });
}

AuthorizationForm::~AuthorizationForm()
{
    delete ui;
    delete passwordLine;
}

void AuthorizationForm::addBackgroundGradient()
{
    QPalette palette = this->palette();
    QLinearGradient gradient(0,0,0, this->height());
    gradient.setColorAt(0, QColor(0,74,140));
    gradient.setColorAt(0.7, QColor(0,242,244));

    QBrush brush(gradient);
    palette.setBrush(QPalette::Window, brush);
    setPalette(palette);
}

void AuthorizationForm::setWindowStyle()
{
    //making design
    addBackgroundGradient();
    ui->hi_label->setStyleSheet(styleHelper::addProjectFont("white"));
    ui->enter_label->setStyleSheet(styleHelper::addProjectFont("rgba(0,0,0, 0.3)"));
    //ui->pushButton->setStyleSheet(styleHelper::addProjectFont("rgba(0,0,0,
0.3)"));
    ui->password->setStyleSheet(styleHelper::addProjectFont("black"));
    ui->telephone->setStyleSheet(styleHelper::addProjectFont("black"));
    this->setStyleSheet(styleHelper::addTextStyle());
    ui->enter_button->setStyleSheet(styleHelper::addPushButtonStyle());

    //forms design
    ui->lineEdit_2->setClearButtonEnabled(true);
    ui->lineEdit_2->setInputMask(QString("800000000000"));

```

```

//configure password form
passwordLine = new PasswordLine(this);
passwordLine->setMinimumSize(QSize(293, 40));
passwordLine->setMaximumSize(QSize(300, 40));
passwordLine->setFont(QFont("Marmelad", 16));
ui->gridLayout_2->addWidget(passwordLine, 8, 1);

}

void AuthorizationForm::paintEvent(QPaintEvent *event)
{
    addBackgroundGradient();
}

void AuthorizationForm::on_enter_button_clicked()
{
    QSqlQuery qry;
    if(qry.exec("SELECT  НомерТелефона,  Пароль,  КодКлиента  FROM
Клиент"))
    {
        while(qry.next())
        {
            if(qry.value(0).toString() == ui->lineEdit_2->text() &&
qry.value(1).toString() == passwordLine->text())
            {
                if(qry.value(2).toInt() == 151)
                {
                    emit RoleDefine(Role::MANAGER, qry.value(2).toInt());
                    return;
                }
                emit RoleDefine(Role::USER, qry.value(2).toInt());
                return;
            }
        }
    }
    if(qry.exec("SELECT НомерТелефона, Пароль, КодКурьера FROM Курьер"))
    {
        while(qry.next())
        {
            if(qry.value(0).toString() == ui->lineEdit_2->text() &&
qry.value(1).toString() == passwordLine->text())
            {
                emit RoleDefine(Role::COURIER, qry.value(2).toInt());
                return;
            }
        }
    }
}

```

```

        }
    }
}
else
{
    qDebug() << "PostgreSQL_authorization_error: ";
}
QMessageBox::information(this, "Ошибка", "Неверно введены данные!",
QMessageBox::Apply);
}

```

### 3. Программный код основного окна

```

//mainwindow.h
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include "databaseconnector.h"
#include <QMainWindow>
#include "freezetablewidget.h"
#include "roles.h"
#include "optionbutton.h"
#include <QMap>
#include "charts_controller.h"
#include "mapdistancecalculator.h"
#include "passwordline.h"

QT_BEGIN_NAMESPACE
namespace Ui {
class MainWindow;
}
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    DatabaseConnector* getDBConnector();
private:
    void authorization();
    void initUserMode(int id);
    void initCourierMode(int id);
    void initManagerMode(int id);
    void initDefaultStyle();

```

```

void initAccountMode();
void initChartController();
void resetChangingUser();

```

```

Ui::MainWindow *ui;
QList <QWidget*> childs;
QMap<QWidget*, FreezeTableWidget*> tables;
DatabaseConnector* db_helper;
optionButton* optionManager;
chartsController* controller;
MapDistanceCalculator* yandex_map_length_controller;
PasswordLine* new_password;
PasswordLine* old_password;
int id_user;
Role role_user;
private slots:
    void checkRole(Role type, int id);
    void on_enterOrder_clicked();
    void on_tabWidget_currentChanged(int index);
};
#endif // MAINWINDOW_H

```

```

//mainwindow.cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QMessageBox>
#include "authorizationform.h"
#include "stylehelper.h"
#include <QTabBar>
#include <QFontDatabase>
#include <QRegularExpressionValidator>
#include <QSqlQuery>
#include <QSqlError>
#include <QSqlQueryModel>
#include "reviewform.h"
#include "correctorderform.h"
#include "payment.h"
#include "change_current_courier_procent.h"
#include <QEventLoop>
#include "checkorderinfo.h"

```

```

MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent)
, ui(new Ui::MainWindow), id_user(-1),

```

```

    optionManager(nullptr),
    db_helper(nullptr)
{
    ui->setupUi(this);
    QFontDatabase::addApplicationFont(":/fonts/Marmelad-Regular.ttf");
    db_helper = new DatabaseConnector();
    controller = new chartsController(this);
    new_password = new PasswordLine(this);
    old_password = new PasswordLine(this);
    db_helper->connect("127.0.0.1", "courier_db", "postgres", "1234", 5432);
    initDefaultStyle();
    authorization();
    initAccountMode();
}

```

```

MainWindow::~MainWindow()
{
    delete ui;
    delete db_helper;
    delete optionManager;
    foreach (QWidget* key, tables) {
        delete tables[key];
    }
    delete controller;
    foreach (QWidget* child, childs) {
        delete child;
    }
    delete yandex_map_length_controller;
}

```

```

DatabaseConnector *MainWindow::getDBConnector()
{
    return db_helper;
}

```

```

void MainWindow::authorization()
{
    foreach (QWidget* child, childs) {
        delete child;
    }
    childs.clear();
    this->hide();
    AuthorizationForm* form = new AuthorizationForm(this);
    childs.append(form);
    form->show();
}

```

```

        connect(form,                               SIGNAL(RoleDefine(Role,int)),           this,
        SLOT(checkRole(Role,int)));
    }

void MainWindow::initUserMode(int id)
{
    ui->weight->setText("");
    ui->your_address->setText("");
    ui->FIO->setText("");
    ui->Address->setText("");
    ui->telephone->setText("");
    //[init styles]
    QPixmap pixmap1(":/resources/clip_icon.png");
    QIcon ButtonIcon1(pixmap1);
    ui->pushButton_4->setIcon(ButtonIcon1);
    ui->pushButton_4->setIconSize(ui->pushButton_4->size());

    QPixmap pixmap2(":/resources/delivery_icon.png");
    QIcon ButtonIcon2(pixmap2);
    ui->pushButton_2->setIcon(ButtonIcon2);
    ui->pushButton_2->setIconSize(ui->pushButton_2->size());

    QPixmap pixmap3(":/resources/done_icon.png");
    QIcon ButtonIcon3(pixmap3);
    ui->pushButton_3->setIcon(ButtonIcon3);
    ui->pushButton_3->setIconSize(ui->pushButton_3->size());

    QPixmap pixmap4(":/resources/re-enter-icon.png");
    QIcon ButtonIcon4(pixmap4);
    ui->pushButton->setIcon(ButtonIcon4);
    ui->pushButton->setIconSize(ui->pushButton->size());

    QPixmap pixmap5(":/resources/personal_account_icon.png");
    QIcon ButtonIcon5(pixmap5);
    ui->pushButton_5->setIcon(ButtonIcon5);
    ui->pushButton_5->setIconSize(ui->pushButton_5->size());

    //order menu design
    ui->tab->setStyleSheet(styleHelper::addProjectFont("white"));
    optionManager = new optionButton();
    QList<QPushButton*> buttons = ui->tab->findChildren<QPushButton*>();
    foreach (QPushButton* button , buttons) {
        if(button != ui->enterOrder)
        {
            optionManager->addButton(button);
        }
    }
}

```

```

        button->setStyleSheet(styleHelper::addPushButtonStyle2());
    }
}
ui->enterOrder->setStyleSheet(styleHelper::addPushButtonStyle());

//line_edit settings
QList<QLineEdit*> line_edit = ui->tab->findChildren<QLineEdit*>();
foreach (QLineEdit* le, line_edit) {
    le->setStyleSheet(styleHelper::addTextStyle());
}
ui->telephone->setInputMask("800000000000");
ui->weight->setValidator(new
QRegularExpressionValidator(QRegularExpression("^\\d{1,3}(\\.\\d{1,2})?$"), ui-
>weight));
//[init styles]

//[init menu buttons functional]
connect(ui->pushButton_2, &QPushButton::clicked, this,
        [this]()
        {
            ui->tabWidget->setCurrentIndex(2);
        });
connect(ui->pushButton_3, &QPushButton::clicked, this,
        [this]()
        {
            ui->tabWidget->setCurrentIndex(1);
        });
connect(ui->pushButton_4, &QPushButton::clicked, this,
        [this]()
        {
            ui->tabWidget->setCurrentIndex(0);
        });
connect(ui->pushButton, &QPushButton::clicked, this,
        [this]()
        {
            authorization();
        });
connect(ui->pushButton_5, &QPushButton::clicked, this,
        [this]()
        {
            ui->tabWidget->setCurrentIndex(9);
        });

```

```

//[init menu buttons functional]

//[init user's tables]
tables[ui->tab_2] = new FreezeTableWidget(this);
tables[ui->tab_3] = new FreezeTableWidget(this);

tables[ui->tab_2]->init(QString("SELECT      КодЗаказа,      СтатусЗаказа,
ДатаЗаказа, ДатаСдачи AS ДатаВыполнения FROM ИсторияПользователей
WHERE "

                                "КодКлиента = %1").arg(id_user),
    {
        QString("SELECT КодЗаказа FROM ИсторияПользователей
WHERE СтатусЗаказа = 'Выполнен' AND КодКлиента = %1 AND КодЗаказа
NOT IN (SELECT КодЗаказа FROM Отзыв WHERE КодКлиента =
%2);").arg(id_user).arg(id_user)
    },
    {
        [id_user = id, table = tables[ui->tab_2], parent =
this](QModelIndex index)
        {
            ReviewForm* form = new ReviewForm(parent, 1, table-
>model()->data(table->model()->index(index.row(), 0)).toInt());
            connect(form, &QDialog::finished, table,
                [table]()
                {
                    table->updateValues();
                });
            form->exec();
        }
    },
    {
        "Дать отзыв"
    });
tables[ui->tab_3]->init(QString("SELECT      КодЗаказа,      СтатусЗаказа,
АдресПолученияТовара,      АдресДоставки,      ДатаЗаказа      FROM
ИсторияПользователей WHERE "

                                "КодКлиента      =      %1      AND      СтатусЗаказа      <>
'Выполнен'").arg(id_user),
    {
        QString("SELECT КодЗаказа FROM ИсторияПользователей
WHERE "

                                "КодКлиента = %1 AND СтатусЗаказа = 'Не принят'
").arg(id_user),

```



```

        QString("SELECT КодЗаказа FROM ИсторияПользователей
WHERE "
        "КодКлиента = %1 AND СтатусЗаказа = 'Не принят'
").arg(id_user)
    },
    {
        [table = tables[ui->tab_3], parent = this](QModelIndex index)
        {
            correctOrderForm* form = new correctOrderForm(parent,
table->model()->data(table->model()->index(index.row(), 0)).toInt());
            connect(form, &QDialog::finished, table,
                    [table]()
                    {
                        table->updateValues();
                    });
            form->exec();
        },
        [table = tables[ui->tab_3], parent = this](QModelIndex index)
        {
            QSqlQuery qry;
            qry.prepare("DELETE FROM Заказ WHERE КодЗаказа =
:order");
            qry.bindValue(":order", table->model()->data(table->model()-
>index(index.row(), 0)).toInt());
            if(!qry.exec()) QMessageBox::critical(parent, "Ошибка!", "Не
удалось удалить заказ! ", QMessageBox::Apply);
            else QMessageBox::information(parent, "Успех!", "Заказ
успешно удален!", QMessageBox::Apply);
            table->updateValues();
        }
    },
    {
        "Изменить",
        "Удалить"
    });

foreach(QWidget* key, tables.keys())
{
    tables[key]->setModel();
}

ui->tab_2->setStyleSheet(styleHelper::addProjectFont("white"));
ui->gridLayout_7->addWidget(tables[ui->tab_2], 2, 0, 2, 4);

ui->tab_3->setStyleSheet(styleHelper::addProjectFont("white"));

```

```

ui->gridLayout_11->addWidget(tables[ui->tab_3], 2, 0, 2, 4);
//![init user's tables]

ui->tabWidget->setCurrentIndex(0);
this->on_tabWidget_currentChanged(ui->tabWidget->currentIndex());
}

void MainWindow::initCourierMode(int id)
{
    //![init menu buttons]
    QPixmap pixmap1(":/resources/delivery_icon.png");
    QIcon ButtonIcon1(pixmap1);
    ui->pushButton_4->setIcon(ButtonIcon1);
    ui->pushButton_4->setIconSize(ui->pushButton_4->size());

    QPixmap pixmap2(":/resources/get_order_icon.png");
    QIcon ButtonIcon2(pixmap2);
    ui->pushButton_2->setIcon(ButtonIcon2);
    ui->pushButton_2->setIconSize(ui->pushButton_2->size());

    QPixmap pixmap3(":/resources/done_icon.png");
    QIcon ButtonIcon3(pixmap3);
    ui->pushButton_3->setIcon(ButtonIcon3);
    ui->pushButton_3->setIconSize(ui->pushButton_3->size());

    QPixmap pixmap4(":/resources/re-enter-icon.png");
    QIcon ButtonIcon4(pixmap4);
    ui->pushButton->setIcon(ButtonIcon4);
    ui->pushButton->setIconSize(ui->pushButton->size());

    QPixmap pixmap5(":/resources/personal_account_icon.png");
    QIcon ButtonIcon5(pixmap5);
    ui->pushButton_5->setIcon(ButtonIcon5);
    ui->pushButton_5->setIconSize(ui->pushButton_5->size());
    //![init menu buttons]
    //![init menu buttons functional]
    connect(ui->pushButton_2, &QPushButton::clicked, this,
        [this]()
        {
            ui->tabWidget->setCurrentIndex(3);
        });
    connect(ui->pushButton_3, &QPushButton::clicked, this,
        [this]()
        {

```

```

        ui->tabWidget->setCurrentIndex(5);
    });
    connect(ui->pushButton_4, &QPushButton::clicked, this,
        [this]()
        {
            ui->tabWidget->setCurrentIndex(4);
        });
    connect(ui->pushButton, &QPushButton::clicked, this,
        [this]()
        {
            authorization();
        });
    connect(ui->pushButton_5, &QPushButton::clicked, this,
        [this]()
        {
            ui->tabWidget->setCurrentIndex(9);
        });
    //![init menu buttons functional]

    //![init functional tables]
    tables[ui->tab_4] = new FreezeTableWidget(this);
    tables[ui->tab_4]->init("SELECT  Заказ.КодЗаказа,  Заказ.НазваниеТарифа,
Заказ.АдресПолученияТовара, "
                        "Заказ.АдресДоставки,                        Заказ.Вес,
Заказ.ПроцентКурьера*СтоимостьЗаказа/100 AS Ставка, Заказ.ДатаЗаказа "
                        "FROM Заказ "
                        "WHERE СтатусЗаказа = 'Не принят'",
    {
        "SELECT КодЗаказа FROM Заказ WHERE "
        "СтатусЗаказа = 'Не принят' AND (НазваниеТарифа <>
'Экспресс' OR( НазваниеТарифа = 'Экспресс' AND "
        "(SELECT Курьер.КоличествоВыполненныхЗаказов FROM
Курьер WHERE КодКурьера = " + QString::number(id) + ") >= 20))"
    },
    {
        [table = tables[ui->tab_4], db = db_helper, parent = this, id_cur
= id](QModelIndex index)
        {
            db->getDB().transaction();
            QSqlQuery qry;
            if(qry.exec("CALL      public.courier_take_order("      +
QString::number(id_cur)  +  ",  "  +  table->model()->data(table->model()-
>index(index.row(), 0)).toString() +");"))
            {
                db->getDB().commit();
            }
        }
    }

```

```

        QMessageBox::information(parent,      "Успех!", "Заказ
успешно принят!", QMessageBox::Apply);
    }
    else
    {
        db->getDB().rollback();
        qDebug() << db->getDB().lastError().text();
        qDebug() << qry.lastQuery();
        QMessageBox::critical(parent, "Ошибка!", "Не удалось
принять заказ! ", QMessageBox::Apply);
    }
    table->updateValues();
}
},
{
    "Взять заказ"
});

```

```

tables[ui->tab_5] = new FreezeTableWidget(this);
tables[ui->tab_5]->init("SELECT Заказ.КодЗаказа, Заказ.НазваниеТарифа, "
                        "Заказ.АдресПолученияТовара,      Заказ.АдресДоставки,
Заказ.Вес, Заказ.ДатаЗаказа "
                        "FROM Заказ INNER JOIN ВРаботе ON Заказ.КодЗаказа =
ВРаботе.КодЗаказа "
                        "WHERE  Заказ.СтатусЗаказа  =  'В  работе'  AND
ВРаботе.КодКурьера = " +QString::number(id),
    {},
    {
        [table = tables[ui->tab_5], parent = this](QModelIndex index)
        {
            QSqlQuery qry;
            qry.prepare("UPDATE Заказ "
                        "SET СтатусЗаказа = 'Выполнен' "
                        "WHERE КодЗаказа = :order;");
            qry.bindValue(":order",      table->model()->data(table-
>model()->index(index.row(), 0));
            if(qry.exec())
                QMessageBox::information(parent,      "Успех!", "Заказ
успешно завершен!", QMessageBox::Apply);
            else
                QMessageBox::critical(parent, "Ошибка!", "Не удалось
завершить заказ! ", QMessageBox::Apply);
            table->updateValues();
        },
        [table = tables[ui->tab_5], parent = this](QModelIndex index)

```

```

    {
        QSqlQuery qry;
        qry.prepare("UPDATE Заказ "
                    "SET СтатусЗаказа = 'Не принят' "
                    "WHERE КодЗаказа = :order;");
        qry.bindValue(":order", table->model()->data(table-
>model()->index(index.row(), 0)));
        if(qry.exec())
            QMessageBox::information(parent, "Успех!", "Заказ
успешно отменен!", QMessageBox::Apply);
        else
            QMessageBox::critical(parent, "Ошибка!", "Не удалось
отменить заказ! ", QMessageBox::Apply);
        table->updateValues();
    }
},
{
    "Завершить заказ",
    "Отменить заказ"
});

```

```

tables[ui->tab_6] = new FreezeTableWidget(this);
tables[ui->tab_6]->init("SELECT          Отзыв.КодЗаказа,Отзыв.Оценка,
Отзыв.Описание "
                    "FROM ВРаботе JOIN Отзыв ON ВРаботе.КодЗаказа =
Отзыв.КодЗаказа "
                    "WHERE ВРаботе.КодКурьера = " + QString::number(id),
    {},
    {},
    {});

```

```

foreach(QWidget* key, tables.keys())
{
    tables[key]->setModel();
}

```

```

ui->tab_4->setStyleSheet(styleHelper::addProjectFont("white"));
ui->gridLayout_10->addWidget(tables[ui->tab_4], 2, 0, 2, 4);

```

```

ui->tab_5->setStyleSheet(styleHelper::addProjectFont("white"));
ui->gridLayout_13->addWidget(tables[ui->tab_5], 2, 0, 2, 4);

```

```

ui->tab_6->setStyleSheet(styleHelper::addProjectFont("white"));
ui->gridLayout_15->addWidget(tables[ui->tab_6], 7, 0, 7, 6);

```

```

//[init functional tables]

ui->tabWidget->setCurrentIndex(3);
this->on_tabWidget_currentChanged(ui->tabWidget->currentIndex());
}

void MainWindow::initManagerMode(int id)
{
    //[init menu button]
    QPixmap pixmap1(":/resources/data_analysis_icon.png");
    QIcon ButtonIcon1(pixmap1);
    ui->pushButton_4->setIcon(ButtonIcon1);
    ui->pushButton_4->setIconSize(ui->pushButton_4->size());

    QPixmap pixmap2(":/resources/get_order_icon.png");
    QIcon ButtonIcon2(pixmap2);
    ui->pushButton_2->setIcon(ButtonIcon2);
    ui->pushButton_2->setIconSize(ui->pushButton_2->size());

    QPixmap pixmap3(":/resources/search_order_icon.png");
    QIcon ButtonIcon3(pixmap3);
    ui->pushButton_3->setIcon(ButtonIcon3);
    ui->pushButton_3->setIconSize(ui->pushButton_3->size());

    QPixmap pixmap4(":/resources/re-enter-icon.png");
    QIcon ButtonIcon4(pixmap4);
    ui->pushButton->setIcon(ButtonIcon4);
    ui->pushButton->setIconSize(ui->pushButton->size());

    QPixmap pixmap5(":/resources/personal_account_icon.png");
    QIcon ButtonIcon5(pixmap5);
    ui->pushButton_5->setIcon(ButtonIcon5);
    ui->pushButton_5->setIconSize(ui->pushButton_5->size());
    //[init menu button]

    //[init menu buttons functional]
    connect(ui->pushButton_4, &QPushButton::clicked, this,
        [this]()
        {
            ui->tabWidget->setCurrentIndex(7);
        });
    connect(ui->pushButton_2, &QPushButton::clicked, this,
        [this]()
        {
            ui->tabWidget->setCurrentIndex(8);
        });
}

```

```

    });
connect(ui->pushButton_3, &QPushButton::clicked, this,
    [this]()
    {
        ui->tabWidget->setCurrentIndex(6);
    });
connect(ui->pushButton, &QPushButton::clicked, this,
    [this]()
    {
        authorization();
    });
connect(ui->pushButton_5, &QPushButton::clicked, this,
    [this]()
    {
        ui->tabWidget->setCurrentIndex(9);
    });
//![init menu buttons functional]

//![init functional elements]
tables[ui->tab_7] = new FreezeTableWidget(this);
tables[ui->tab_7]->init("SELECT Заказ.КодЗаказа, "
    "Заказ.ПроцентКурьера*Заказ.СтоимостьЗаказа/100      AS
СтавкаКурьера, "
    "Заказ.ПроцентКурьера AS ТекущийПроцент, "
    "(CURRENT_DATE - Заказ.ДатаЗаказа) AS ДниОжидания "
    "FROM Заказ WHERE СтатусЗаказа = 'Не принят'",
    {},
    {
        [table = tables[ui->tab_7], parent = this](QModelIndex index)
        {
            change_current_courier_procent* form = new
change_current_courier_procent(parent, table->model()->data(table->model()-
>index(index.row(), 2)).toInt(), table->model()->data(table->model()-
>index(index.row(), 0)).toInt());
            connect(form, &QDialog::finished, table,
                [table]()
                {
                    table->updateValues();
                });
            form->exec();
        },
        [table = tables[ui->tab_7], parent = this](QModelIndex index)
        {
            checkOrderInfo* form = new checkOrderInfo(parent, table-
>model()->data(table->model()->index(index.row(), 0)).toInt());

```

```

        connect(form, &QDialog::finished, table,
                [table]()
                {
                    table->updateValues();
                });
        form->exec();
    }
},
{
    "Изменить проц-т",
    "Смотреть заказ"
});

tables[ui->tab_9] = new FreezeTableWidget(this);
tables[ui->tab_9]->init("SELECT      Клиент.ФИО      AS      ФИОКлиента,
Клиент.НомерТелефона AS НомерТелефонаКлиента, "
    "Заказ.КодЗаказа,  Отзыв.Описание  AS  ОписаниеОтзыва,
Отзыв.Оценка AS ОценкаДоставки "
    "FROM  Отзыв  JOIN  Заказ  ON  Отзыв.КодЗаказа  =
Заказ.КодЗаказа "
    "JOIN Клиент ON Клиент.КодКлиента = Отзыв.КодКлиента",
    {},
    {},
    {});

foreach(QWidget* key, tables.keys())
{
    tables[key]->setModel();
}

ui->tab_7->setStyleSheet(styleHelper::addProjectFont("white"));
ui->gridLayout_17->addWidget(tables[ui->tab_7], 2, 0, 2, 4);

ui->tab_8->setStyleSheet(styleHelper::addProjectFont("white"));

ui->tab_9->setStyleSheet(styleHelper::addProjectFont("white"));
ui->gridLayout_21->addWidget(tables[ui->tab_9], 2, 0, 2, 4);
//![init functional elements]

ui->tabWidget->setCurrentIndex(6);
this->on_tabWidget_currentChanged(ui->tabWidget->currentIndex());
}

//![init default style settings for form]

```



```

void MainWindow::initDefaultStyle()
{
    ui->tabWidget->tabBar()->hide();
    ui->tabWidget->setStyleSheet("border-style: none");
    ui->tabWidget->setDocumentMode(true);
}

void MainWindow::initAccountMode()
{
    old_password->setText("");
    new_password->setText("");
    switch(role_user)
    {
        case Role::COURIER:
            ui->label_28->setVisible(true);
            ui->lineEdit_7->setVisible(true);
            ui->user_role_lk->setText("Курьер");
            ui->label_28->setText("Номер паспорта");
            ui->lineEdit_7->setInputMask(QString("00 00 000000"));
            break;
        case Role::USER:
            ui->label_28->setVisible(true);
            ui->lineEdit_7->setVisible(true);
            ui->label_28->setText("Ваш адрес");
            ui->user_role_lk->setText("Пользователь");
            ui->lineEdit_7->setInputMask(QString());
            break;
        case Role::MANAGER:
            ui->label_28->setVisible(false);
            ui->lineEdit_7->setVisible(false);
            ui->user_role_lk->setText("Менеджер");
            break;
        default:
            break;
    }

    //[init personal account style settings]
    ui->tab_10->setStyleSheet(styleHelper::addProjectFont("white") + '\n' +
styleHelper::addTextStyle());
    ui->user_role_lk->setStyleSheet(styleHelper::addProjectFont("rgba(242, 133,
255, 255)"));
    ui->pushButton_6->setStyleSheet(styleHelper::addPushButtonStyle());
    ui->pushButton_7->setStyleSheet(styleHelper::addPushButtonStyle());

    ui->lineEdit_3->setInputMask(QString("800000000000"));

```

```

new_password->setMinimumSize(QSize(0, 40));
new_password->setMaximumSize(QSize(1111111, 40));
new_password->setFont(QFont("Marmelad", 16));
ui->gridLayout_23->addWidget(new_password, 19, 1);

old_password->setMinimumSize(QSize(0, 40));
old_password->setMaximumSize(QSize(1111111, 40));
old_password->setFont(QFont("Marmelad", 16));
ui->gridLayout_23->addWidget(old_password, 21, 1);

//[init personal account style settings]

//[init pushbuttons action]
connect(ui->pushButton_6, &QPushButton::clicked, this,
        [this]()
        {
            if(ui->lineEdit_3->text().length() != 11)
            {
                QMessageBox::critical(this, "Ошибка! ", "Неверный формат номера
телефона!", QMessageBox::Apply);
                return;
            }
            if(role_user == Role::COURIER)
            {
                if(ui->lineEdit_7->text().length() != 12)
                {
                    QMessageBox::critical(this, "Ошибка! ", "Неверный формат
данных паспорта!", QMessageBox::Apply);
                    return;
                }

                QSqlQuery qry1;
                qry1.prepare("SELECT НомерТелефона FROM Курьер WHERE
КодКурьера <> :curier UNION SELECT НомерТелефона FROM Клиент;");
                qry1.bindValue(":curier", id_user);
                qry1.exec();
                while(qry1.next())
                {
                    if(qry1.value(0).toString() == ui->lineEdit_3->text())
                    {
                        QMessageBox::critical(this, "Ошибка! ", "Данный номер уже
используется!", QMessageBox::Apply);

```

```

        return;
    }
}

```

```

QSqlQuery qry;
qry.prepare("UPDATE Курьер "
            "SET ФИО = :fio, "
            "НомерТелефона = :tel, "
            "ДанныеПаспорта = :pass "
            "WHERE КодКурьера = :curier");
qry.bindValue(":fio", ui->lineEdit->text());
qry.bindValue(":tel", ui->lineEdit_3->text());
qry.bindValue(":pass", ui->lineEdit_7->text());
qry.bindValue(":curier", id_user);

```

```

        if(!qry.exec()) QMessageBox::critical(this, "Ошибка! ", "Не удалось
обновить данные", QMessageBox::Apply);
        else QMessageBox::information(this, "Успех ", "Данные успешно
обновлены!", QMessageBox::Apply);
    }
    else
    {
        QSqlQuery qry1;
        qry1.prepare("SELECT НомерТелефона FROM Курьер UNION
SELECT НомерТелефона FROM Клиент WHERE КодКлиента <> :user;");
        qry1.bindValue(":user", id_user);
        qry1.exec();
        while(qry1.next())
        {
            if(qry1.value(0).toString() == ui->lineEdit_3->text())
            {
                QMessageBox::critical(this, "Ошибка! ", "Данный номер уже
используется!", QMessageBox::Apply);
                return;
            }
        }
    }
}

```

```

QSqlQuery qry;
qry.prepare("UPDATE Клиент "
            "SET ФИО = :fio, "
            "НомерТелефона = :tel, "

```

```

        "АдресКлиента = :address "
        "WHERE КодКлиента = :user");
    qry.bindValue(":fio", ui->lineEdit->text());
    qry.bindValue(":tel", ui->lineEdit_3->text());
    qry.bindValue(":address", ui->lineEdit_7->text());
    qry.bindValue(":user", id_user);

    if(!qry.exec()) QMessageBox::critical(this, "Ошибка! ", "Не удалось
обновить данные", QMessageBox::Apply);
    else QMessageBox::information(this, "Успех ", "Данные успешно
обновлены!", QMessageBox::Apply);
    }
    });
    connect(ui->pushButton_7, &QPushButton::clicked, this,
    [this]()
    {
        if(new_password->text() == "" or old_password->text() == "")
        {
            QMessageBox::critical(this, "Ошибка! ", "Некорректный формат
данных!", QMessageBox::Apply);
            return;
        }
        if(role_user == Role::COURIER)
        {
            QSqlQuery qry;
            qry.prepare("SELECT Пароль FROM Курьер WHERE КодКурьера
= :curier");
            qry.bindValue(":curier", id_user);
            qry.exec();
            if(qry.next())
            {
                QMessageBox::critical(this, "Ошибка! ", "Не удалось обновить
пароль", QMessageBox::Apply);
                return;
            }

            if(qry.value(0).toString() == old_password->text())
            {
                QSqlQuery qry1;
                qry1.prepare("UPDATE Курьер SET Пароль = :password WHERE
КодКурьера = :curier");
                qry1.bindValue(":password", new_password->text());

```

```

        qry1.bindValue(":curier", id_user);
        if(!qry1.exec())
            QMessageBox::critical(this, "Ошибка! ", "Не удалось обновить
пароль", QMessageBox::Apply);
        else
            QMessageBox::information(this, "Успех ", "Данные успешно
обновлены!", QMessageBox::Apply);
    }
    else QMessageBox::critical(this, "Ошибка! ", "Неверно введен
старый пароль", QMessageBox::Apply);
}
else
{
    QSqlQuery qry;
    qry.prepare("SELECT Пароль FROM Клиент WHERE КодКлиента
= :user");
    qry.bindValue(":user", id_user);
    qry.exec();
    if(!qry.next())
    {
        QMessageBox::critical(this, "Ошибка! ", "Не удалось обновить
пароль", QMessageBox::Apply);
        return;
    }

    if(qry.value(0).toString() == old_password->text())
    {
        QSqlQuery qry1;
        qry1.prepare("UPDATE Клиент SET Пароль = :password WHERE
КодКлиента = :user");
        qry1.bindValue(":password", new_password->text());
        qry1.bindValue(":user", id_user);
        if(!qry1.exec())
            QMessageBox::critical(this, "Ошибка! ", "Не удалось обновить
пароль", QMessageBox::Apply);
        else
        {
            QMessageBox::information(this, "Успех ", "Данные успешно
обновлены!", QMessageBox::Apply);
        }
    }
    else QMessageBox::critical(this, "Ошибка! ", "Неверно введен
старый пароль", QMessageBox::Apply);
}

```

```

    });
    //![init pushbuttons action]
}

void MainWindow::initChartController()
{
    controller->addNewLineSeries("SELECT ДатаЗаказа, COUNT(*) FROM Заказ
GROUP BY ДатаЗаказа ORDER BY ДатаЗаказа ASC", QPen(Qt::red));
    controller->addNewLineSeries("SELECT ДатаСдачи, COUNT(*) FROM Заказ
WHERE ДатаСдачи IS NOT NULL GROUP BY ДатаСдачи ORDER BY
ДатаСдачи ASC", QPen(Qt::green));
    controller->init();
    ui->gridLayout_19->addWidget(controller->line1_view(), 2, 0, 2, 4);
    controller->line1_view()->setMinimumSize(500, 300);
}

void MainWindow::resetChangingUser()
{
    //![disconnect events on pushbutton in account form]
    disconnect(ui->pushButton_6, nullptr, nullptr, nullptr);
    disconnect(ui->pushButton_7, nullptr, nullptr, nullptr);
    //![disconnect events on pushbutton in account form]

    //![disconnect menu buttons]
    disconnect(ui->pushButton_2, nullptr, nullptr, nullptr);
    disconnect(ui->pushButton_3, nullptr, nullptr, nullptr);
    disconnect(ui->pushButton_4, nullptr, nullptr, nullptr);
    disconnect(ui->pushButton, nullptr, nullptr, nullptr);
    disconnect(ui->pushButton_5, nullptr, nullptr, nullptr);
    //![disconnect menu buttons]
}

//![init default style settings for form]

//![init main window mode]
void MainWindow::checkRole(Role user, int id)
{
    qDebug() << id;
    childs[0]->close(); //close authorization window
    this->show();
    role_user = user;
    id_user = id;
    resetChangingUser();
    switch(user)
    {

```

```

case Role::USER:
    initUserMode(id);
    break;
case Role::COURIER:
    initCourierMode(id);
    break;
case Role::MANAGER:
    initManagerMode(id);
    initChartController();
    break;
default:
    break;
}

initAccountMode();
}
//![init main window mode]

//![init get_usres_order_pushbutton_action]
void MainWindow::on_enterOrder_clicked()
{
    if(optionManager->active == nullptr)
    {
        QMessageBox::critical(this, "Ошибка! ", "Не выбран способ доставки! ",
        QMessageBox::Apply);
        return;
    }
    if(ui->weight->text() == "" or ui->your_address->text() == "" or ui->Address-
>text() == "")// or ui->FIO->text() == "" or ui->telephone->text().length() != 11)
    {
        QMessageBox::critical(this, "Ошибка! ", "Не все поля заполнены
корректно", QMessageBox::Apply);
        return;
    }
    if(ui->weight->text().toDouble() > 10.0 and optionManager-
>name_of_button(optionManager->active) == "Экспресс")
    {
        QMessageBox::critical(this, "Ошибка! ", "Максимально доступный вес в
тарифе Экспресс равен 10 кг", QMessageBox::Apply);
        return;
    }
    yandex_map_length_controller = new MapDistanceCalculator(this,
"CtSWNkTnMj92mAJJhcX15tWcko77VMgco6K1Y3DQTr5AnUA8XJS0c325EZ
kE0OCv", "fe6b37b2-0fdc-45b2-a93f-d4116fcb4c47");

```

```

QEventLoop l;
connect(yandex_map_length_controller, &MapDistanceCalculator::API_answer,
&l, &QEventLoop::quit);
connect(yandex_map_length_controller,
&MapDistanceCalculator::enter_uncorrect_data, &l, &QEventLoop::quit);
connect(yandex_map_length_controller,
&MapDistanceCalculator::error_distance_calculator, &l, &QEventLoop::quit);
connect(yandex_map_length_controller,
&MapDistanceCalculator::enter_uncorrect_data, this,
    [this]()
    {
        QMessageBox::critical(this, "Ошибка! ", "Указанные адреса неверны!",
QMessageBox::Apply);
        return;
    });
connect(yandex_map_length_controller,
&MapDistanceCalculator::error_distance_calculator, this,
    [this]()
    {
        QMessageBox::critical(this, "Ошибка! ", "Не удалось вычислить
расстояние между адресами! Попробуйте позже.", QMessageBox::Apply);
        return;
    });

//![get data from yandex map API]
yandex_map_length_controller->setCoordinatesOfAddress(ui->Address-
>text());
l.exec();
yandex_map_length_controller->setCoordinatesOfAddress(ui->your_address-
>text());
l.exec();
yandex_map_length_controller->setDistance();
l.exec();
//![get data from yandex map API]

QSqlQuery qry1;
qDebug() << optionManager->active->text();
qry1.prepare("SELECT Цена FROM Тариф WHERE НазваниеТарифа =
:tariff");
qry1.bindValue(":tariff", optionManager->active->text());
qry1.exec(); qry1.next();
Payment* form = new Payment(nullptr,
QString::number(yandex_map_length_controller->getResult()/1000 * 5 +
qry1.value(0).toInt()));
if(form->exec() != QDialog::Accepted)

```



```

{
    QMessageBox::critical(this, "Ошибка! ", "Заказ не был оплачен",
    QMessageBox::Apply);
    delete form;
    return;
}
delete form;
QString qry;
qry.prepare("INSERT INTO Заказ (КодКлиента,НазваниеТарифа, Вес,
АдресПолученияТовара, АдресДоставки, ФИОПолучателя,
НомерТелефонаПолучателя, СтоимостьЗаказа) "
"VALUES (:id_client, :tariff, :weight, :address_get, :address_del, :fio,
:telephone, :price)");

qry.bindValue(":id_client", id_user);
qry.bindValue(":tariff", optionManager->name_of_button(optionManager-
>active));
qry.bindValue(":weight", ui->weight->text().toDouble());
qry.bindValue(":address_get", ui->your_address->text());
qry.bindValue(":address_del", ui->Address->text());
qry.bindValue(":price", yandex_map_length_controller->getResult()/1000 * 5 +
qry1.value(0).toInt());
if(ui->FIO->text() == "")
    qry.bindValue(":fio", QVariant());
else
    qry.bindValue(":fio", ui->FIO->text());
if(ui->telephone->text() == "8")
    qry.bindValue(":telephone", QVariant());
else
    qry.bindValue(":telephone", ui->telephone->text());

ui->weight->setText("");
ui->your_address->setText("");
ui->Address->setText("");
ui->FIO->setText("");
ui->telephone->setText("");

if(!qry.exec())
{
    qDebug() << db_helper->getDB().lastError();
    QMessageBox::critical(this, "Ошибка! ", "Не удалось добавить заказ! "+
db_helper->getDB().lastError().text(), QMessageBox::Apply);
}
else QMessageBox::information(this, "Успех ", "Заказ успешно добавлен!",
QMessageBox::Apply);

```

```

}
//![init get_usres_order_pushbutton_action]

void MainWindow::on_tabWidget_currentChanged(int index)
{
    switch(index)
    {
        case 5:
        {
            QSqlQuery qry;
            qry.exec("SELECT AVG(Отзыв.Оценка) "
                    "FROM ВРаботе JOIN Отзыв ON ВРаботе.КодЗаказа =
Отзыв.КодЗаказа "
                    "WHERE ВРаботе.КодКурьера = " + QString::number(id_user));
            qry.next();
            ui->rating->setText(qry.value(0).toString().left(4));
            qry.exec("SELECT КоличествоВыполненныхЗаказов FROM Курьер
WHERE КодКурьера = " + QString::number(id_user));
            qry.next();
            ui->do_order->setText(qry.value(0).toString());
            tables[ui->tabWidget->widget(index)]->updateValues();
            break;
        }
        case 7:
        {
            controller->updateValues({"SELECT ДатаЗаказа, COUNT(*) FROM
Заказ GROUP BY ДатаЗаказа ORDER BY ДатаЗаказа ASC",
                    "SELECT ДатаСдачи, COUNT(*) FROM Заказ WHERE
ДатаСдачи IS NOT NULL GROUP BY ДатаСдачи ORDER BY ДатаСдачи
ASC"});
            break;
        }
        case 9:
        {
            if(role_user == Role::COURIER)
            {
                QSqlQuery qry;
                qry.prepare("SELECT ФИО, НомерТелефона, ДанныеПаспорта "
                        "FROM Курьер WHERE КодКурьера = :user_id");
                qry.bindValue(":user_id", id_user);
                qry.exec();
                if(qry.next())
                {
                    ui->lineEdit->setText(qry.value(0).toString());
                    ui->lineEdit_3->setText(qry.value(1).toString());
                }
            }
        }
    }
}

```

```

        ui->lineEdit_7->setText(qry.value(2).toString());
    }
    else
    {
        QMessageBox::critical(this, "Ошибка! ", "Не получилось получить
данные о пользователе", QMessageBox::Apply);
    }
}
else
{
    QSqlQuery qry;
    qry.prepare("SELECT ФИО, НомерТелефона, АдресКлиента "
        "FROM Клиент WHERE КодКлиента = :user_id");
    qry.bindValue(":user_id", id_user);
    qry.exec();
    if(qry.next())
    {
        ui->lineEdit->setText(qry.value(0).toString());
        ui->lineEdit_3->setText(qry.value(1).toString());
        ui->lineEdit_7->setText(qry.value(2).toString());
    }
    else
    {
        QMessageBox::critical(this, "Ошибка! ", "Не получилось получить
данные о пользователе", QMessageBox::Apply);
    }
}
break;
}
case 0:
{
    QSqlQuery qry;
    qry.prepare("SELECT    АдресКлиента    FROM    Клиент    WHERE
КодКлиента = :id");
    qry.bindValue(":id", id_user);
    qry.exec(); qry.next();
    ui->your_address->setText(qry.value(0).toString());
    break;
}
default:
{
    tables[ui->tabWidget->widget(index)]->updateValues();
}
}
}
}

```

## 4. Программный код окна «Оплата»

```

//payment.h
#ifndef PAYMENT_H
#define PAYMENT_H

#include <QDialog>

namespace Ui {
class Payment;
}

class Payment : public QDialog
{
    Q_OBJECT

public:
    explicit Payment(QWidget *parent = nullptr, QString price = "0");
    void init();
    ~Payment();

private slots:
    void on_pushButton_clicked();

private:
    Ui::Payment *ui;
};

#endif // PAYMENT_H
//payment.cpp
#include "payment.h"
#include "ui_payment.h"
#include "stylehelper.h"
#include <QRegularExpressionValidator>

Payment::Payment(QWidget *parent, QString price)
    : QDialog(parent)
    , ui(new Ui::Payment)
{
    ui->setupUi(this);
    connect(ui->pushButton, &QPushButton::clicked, this, &QDialog::accept);
    ui->price->setText(price + " RUB");
    init();
}

```

```

void Payment::init()
{
    QPalette palette = this->palette();
    QBrush brush(QColor(1, 50, 125));
    palette.setBrush(QPalette::Window, brush);
    setPalette(palette);

    ui->label_3->setStyleSheet(styleHelper::addProjectFont("white"));
    ui->label->setStyleSheet(styleHelper::addProjectFont("white"));
    ui->price->setStyleSheet(styleHelper::addProjectFont("#69E0FE"));
    ui->pushButton->setStyleSheet(styleHelper::addPushButtonStyle());
    ui->lineEdit->setStyleSheet("QLineEdit{ "
        "background: none;"
        "background-color: #999999;"
        "border: none;"
        "font-family: Marmelad;"
        "}");
    ui->lineEdit_2->setStyleSheet("QLineEdit{ "
        "background: none;"
        "background-color: #999999;"
        "border: none;"
        "font-family: Marmelad;"
        "}");
    ui->lineEdit_3->setStyleSheet("QLineEdit{ "
        "background: none;"
        "background-color: #999999;"
        "border: none;"
        "font-family: Marmelad;"
        "}");
    ui->lineEdit_4->setStyleSheet("QLineEdit{ "
        "background: none;"
        "background-color: #999999;"
        "border: none;"
        "font-family: Marmelad;"
        "}");

    ui->lineEdit->setInputMask("0000 0000 0000 0000");
    ui->lineEdit_2->setInputMask("000");
    ui->lineEdit_3->setValidator(new
QRegularExpressionValidator(QRegularExpression("^0[0-9]|1[0-2]$"),      ui-
>lineEdit_3));
    ui->lineEdit_4->setValidator(new
QRegularExpressionValidator(QRegularExpression("(19\\d{2}|20[0-2][0-4])$"),
ui->lineEdit_4)); // ui->lineEdit->setPlaceholderText("Номер карты");
}

```

```

Payment::~Payment()
{
    delete ui;
}

void Payment::on_pushButton_clicked()
{
    this->accept();
}

```

### 5. Программный код окна «Редактировать заказ»

```

//correctorderform.h
#ifndef CORRECTORDERFORM_H
#define CORRECTORDERFORM_H

#include <QDialog>

namespace Ui {
class correctOrderForm;
}

class correctOrderForm : public QDialog
{
    Q_OBJECT

public:
    explicit correctOrderForm(QWidget *parent = nullptr, int id_order = -1);
    ~correctOrderForm();

private slots:
    void on_pushButton_clicked();

private:
    int id_order;
    void init();
    void setDataIntoLineEdit();
    Ui::correctOrderForm *ui;
};

#endif // CORRECTORDERFORM_H

//correctorderform.cpp
#include "correctorderform.h"
#include "ui_correctorderform.h"

```

```

#include "stylehelper.h"
#include <QSqlQuery>
#include <QRegularExpressionValidator>
#include <QMessageBox>
#include <QSqlError>

correctOrderForm::correctOrderForm(QWidget *parent, int order)
    : QDialog(parent)
    , ui(new Ui::correctOrderForm), id_order(order)
{
    ui->setupUi(this);
    init();
}

correctOrderForm::~correctOrderForm()
{
    delete ui;
}

void correctOrderForm::init()
{
    QPalette palette = this->palette();
    QBrush brush(QColor(1, 50, 125));
    palette.setBrush(QPalette::Window, brush);
    setPalette(palette);

    foreach (QLabel* l, this->findChildren<QLabel*>()) {
        l->setStyleSheet(styleHelper::addProjectFont("white"));
    }
    ui->weight->setStyleSheet(styleHelper::addTextStyle());
    ui->address_get->setStyleSheet(styleHelper::addTextStyle());
    ui->address_del->setStyleSheet(styleHelper::addTextStyle());
    ui->fio->setStyleSheet(styleHelper::addTextStyle());
    ui->telephone->setStyleSheet(styleHelper::addTextStyle());
    ui->telephone->setInputMask("8000000000");
    ui->weight->setValidator(new
QRegularExpressionValidator(QRegularExpression("^\\d+(\\.\\d+)?$"),      ui-
>weight));

    ui->pushButton->setStyleSheet(styleHelper::addPushButtonStyle2());
    setDataIntoLineEdit();
}

void correctOrderForm::setDataIntoLineEdit()
{

```

```

    QSqlQuery qry;
    qry.exec("SELECT    Вес,    АдресПолученияТовара,    АдресДоставки,
ФИОПолучателя, НомерТелефонаПолучателя "
        "FROM Заказ WHERE КодЗаказа = " + QString::number(id_order));
    qry.next();
    ui->weight->setText(qry.value(0).toString());
    ui->address_get->setText(qry.value(1).toString());
    ui->address_del->setText(qry.value(2).toString());
    ui->fio->setText(qry.value(3).toString());
    ui->telephone->setText(qry.value(4).toString());
}

void correctOrderForm::on_pushButton_clicked()
{
    int decision = QMessageBox::warning(this, "Вы уверены?", "Вы уверены, что
хотите изменить данные?", QMessageBox::Yes | QMessageBox::No);
    if (decision == QMessageBox::Yes) {
        QSqlQuery qry;
        qry.prepare("UPDATE Заказ SET "
            "Вес = :weight, "
            "АдресПолученияТовара = :a_get, "
            "АдресДоставки = :a_del, "
            "ФИОПолучателя = :fio, "
            "НомерТелефонаПолучателя = :telephone "
            "WHERE КодЗаказа = :id_order;");

        qry.bindValue(":weight", ui->weight->text().toDouble()); // Привязываем
вес
        qry.bindValue(":a_get", ui->address_get->text()); // Привязываем адрес
получения товара
        qry.bindValue(":a_del", ui->address_del->text()); // Привязываем адрес
доставки
        qry.bindValue(":fio", ui->fio->text()); // Привязываем ФИО
получателя
        qry.bindValue(":telephone", ui->telephone->text()); // Привязываем номер
телефона
        qry.bindValue(":id_order", id_order); // Привязываем id заказа

        if(!qry.exec())
        {
            QMessageBox::critical(this, "Ошибка! ", "Не удалось изменить заказ! ",
            QMessageBox::Apply);
        }
        else

```



```

    {
        QMessageBox::information(this, "Успех ", "Заказ успешно изменен!",
        QMessageBox::Apply);
        this->close();
        delete this;
    }
}
}

```

## 6. Программный код окна «Создать отзыв»

```

//reviewform.h
#ifndef REVIEWFORM_H
#define REVIEWFORM_H

#include <QDialog>
#include <QSqlDatabase>
namespace Ui {
class ReviewForm;
}

class ReviewForm : public QDialog
{
    Q_OBJECT

public:
    explicit ReviewForm(QWidget *parent = nullptr, int id_client = -1, int id_order =
-1);
    ~ReviewForm();
private slots:
    void on_pushButton_clicked();
    void paintGivingRating();
private:
    void init();

    QList<QPushButton*> stars;
    int enteredRating;
    Ui::ReviewForm *ui;
    int client;
    int id_order_review;
};

#endif // REVIEWFORM_H
//reviewform.cpp
#include "reviewform.h"

```

```

#include "ui_reviewform.h"
#include "stylehelper.h"
#include <QSqlQuery>
#include <QMessageBox>
#include <QDebug>
#include <QVariant>
#include <QMetaType>

```

```

ReviewForm::ReviewForm(QWidget *parent, int id_client, int id_order)
    : QDialog(parent),
    ui(new Ui::ReviewForm),    client(id_client),    id_order_review(id_order),
    enteredRating(-1)
{
    ui->setupUi(this);
    qDebug() << client;
    // Находим все кнопки на форме
    for (int i = 0; i < ui->horizontalLayout->count(); ++i) {
        QWidget* widget = ui->horizontalLayout->itemAt(i)->widget();
        if (QPushButton* btn = qobject_cast<QPushButton*>(widget)) {
            stars.append(btn);
        }
    }

    // Подключаем все кнопки к слоту
    for (QPushButton* btn : stars) {
        connect(btn,                      &QPushButton::clicked,          this,
        &ReviewForm::paintGivingRating);
    }
    init();
}

```

```

ReviewForm::~~ReviewForm()
{
    delete ui;
}

```

```

void ReviewForm::on_pushButton_clicked()
{
    if(ui->plainTextEdit->toPlainText().length() > 190)
    {
        QMessageBox::critical(this,    "Ошибка!    ", "Слишком    большой    отзыв
(максимум 190 символов)!", QMessageBox::Apply);
        return;
    }
    if(enteredRating == -1)

```

```

{
    QMessageBox::critical(this, "Ошибка! ", "Необходимо установить оценку!",
    QMessageBox::Apply);
    return;
}
 QSqlQuery qry;
    qry.prepare("INSERT INTO  ОТЗЫВ  (Описание,  Оценка,  КодКлиента,
    КодЗаказа) "
        "VALUES (:info, :mark, :client, :order)");

    // Привязка значений
    if(ui->plainTextEdit->toPlainText() == "")
        qry.bindValue(":info", QVariant());
    else
        qry.bindValue(":info", ui->plainTextEdit->toPlainText());
    qry.bindValue(":mark", enteredRating);
    qry.bindValue(":client", client);
    qry.bindValue(":order", id_order_review);
    if(!qry.exec())
    {
        QMessageBox::critical(this, "Ошибка! ", "Не удалось добавить отзыв!",
        QMessageBox::Apply);
    }
    else
    {
        QMessageBox::information(this, "Успех ", "Отзыв успешно добавлен!",
        QMessageBox::Apply);
        this->close();
        delete this;
    }
}

void ReviewForm::paintGivingRating()
{
    QPushButton* buttonSender = qobject_cast<QPushButton*>(sender());
    if (!buttonSender) {
        qDebug() << "Error: sender is not a QPushButton!";
        return;
    }

    // Обновляем иконки
    for (int i = 0; i < stars.size(); ++i) {
        if (buttonSender == stars[i]) {
            enteredRating = i + 1;

```

```

qDebug() << "Rating entered:" << enteredRating;

// Обновляем иконки
for (int j = 0; j <= i; ++j) {
    QIcon starIcon(":/resources/icons8-star-48-full.png");
    if (starIcon.isNull()) {
        qDebug() << "Icon not loaded!";
    }
    stars[j]->setIcon(starIcon);
}
for (int j = i+1; j < stars.size(); ++j) {
    QIcon starIcon(":/resources/icons8-star-48-.png");
    if (starIcon.isNull()) {
        qDebug() << "Icon not loaded!";
    }
    stars[j]->setIcon(starIcon);
}
}
}

void ReviewForm::init()
{
    QPalette palette = this->palette();
    QBrush brush(QColor(1, 50, 125));
    palette.setBrush(QPalette::Window, brush);
    setPalette(palette);

    ui->label->setStyleSheet(styleHelper::addProjectFont("white"));
    ui->label_2->setStyleSheet(styleHelper::addProjectFont("white"));
    ui->plainTextEdit->setStyleSheet(styleHelper::addTextStyle());
    ui->pushButton->setStyleSheet(styleHelper::addPushButtonStyle());
}

```

## 7. Программный код окна «Информация о заказе»

```

//checkorderinfo.h
#ifndef CHECKORDERINFO_H
#define CHECKORDERINFO_H

#include <QDialog>

namespace Ui {
class checkOrderInfo;
}

```

```

class checkOrderInfo : public QDialog
{
    Q_OBJECT

public:
    explicit checkOrderInfo(QWidget *parent = nullptr, int id_order = -1);
    ~checkOrderInfo();

private slots:
    void on_pushButton_clicked();

private:
    Ui::checkOrderInfo *ui;
    void init();
    int order;
};

#endif // CHECKORDERINFO_H
//checkorderinfo.cpp

#include "checkorderinfo.h"
#include "ui_checkorderinfo.h"
#include "stylehelper.h"
#include <QString>

checkOrderInfo::checkOrderInfo(QWidget *parent, int id_order)
    : QDialog(parent)
    , ui(new Ui::checkOrderInfo), order(id_order)
{
    ui->setupUi(this);
    init();
}

checkOrderInfo::~checkOrderInfo()
{
    delete ui;
}

void checkOrderInfo::on_pushButton_clicked()
{
    this->close();
    delete this;
}

```

```

void checkOrderInfo::init()
{
    QPalette palette = this->palette();
    QBrush brush(QColor(1, 50, 125));
    palette.setBrush(QPalette::Window, brush);
    setPalette(palette);

    this->setStyleSheet(styleHelper::addProjectFont("white"));
    ui->pushButton->setStyleSheet(styleHelper::addPushButtonStyle());

    QSqlQuery qry;
    qry.prepare("SELECT КодЗаказа, АдресПолученияТовара, АдресДоставки,
Бес, ДатаЗаказа FROM Заказ WHERE КодЗаказа = :order");
    qry.bindValue(":order", order);
    qry.exec(); qry.next();
    ui->label_2->setText(qry.value(0).toString());
    ui->label_4->setText(qry.value(1).toString());
    ui->label_6->setText(qry.value(2).toString());
    ui->label_8->setText(qry.value(3).toString());
    ui->label_10->setText(qry.value(4).toString());
}

```

#### 8. Программный код окна «Изменить процент»

```

// change_current_courier_procent.h
#ifndef CHANGE_CURRENT_COURIER_PROCENT_H
#define CHANGE_CURRENT_COURIER_PROCENT_H

#include <QDialog>

namespace Ui {
class change_current_courier_procent;
}

class change_current_courier_procent : public QDialog
{
    Q_OBJECT

public:
    explicit change_current_courier_procent(QWidget *parent = nullptr, int
procent_data = 0, int order_id = 0);
    ~change_current_courier_procent();

private slots:
    void on_pushButton_clicked();

```

```

private:
    int order_id;
    Ui::change_current_courier_procent *ui;
    void init();
};

#endif // CHANGE_CURRENT_COURIER_PROCENT_H

// change_current_courier_procent.cpp
#include "change_current_courier_procent.h"
#include "ui_change_current_courier_procent.h"
#include "stylehelper.h"
#include <QSqlQuery>
#include <QMessageBox>

change_current_courier_procent::change_current_courier_procent(QWidget
*parent, int procent_data, int order)
    : QDialog(parent)
    , ui(new Ui::change_current_courier_procent)
{
    order_id = order;
    ui->setupUi(this);
    ui->now->setText(QString::number(procent_data));
    ui->spinBox->setValue(procent_data);
    init();
}

change_current_courier_procent::~change_current_courier_procent()
{
    delete ui;
}

void change_current_courier_procent::init()
{
    QPalette palette = this->palette();
    QBrush brush(QColor(1, 50, 125));
    palette.setBrush(QPalette::Window, brush);
    setPalette(palette);

    ui->label->setStyleSheet(styleHelper::addProjectFont("white"));
    ui->label_2->setStyleSheet(styleHelper::addProjectFont("white"));
    ui->now->setStyleSheet(styleHelper::addProjectFont("white"));
    ui->spinBox->setStyleSheet(styleHelper::addTextStyle());
    ui->pushButton->setStyleSheet(styleHelper::addPushButtonStyle());
}

```

```

void change_current_courier_procent::on_pushButton_clicked()
{
    if(ui->spinBox->value() < 80 or ui->spinBox->value() > 90)
    {
        QMessageBox::critical(nullptr, "Ошибка! ", "Процент курьер варьируется от
80 до 90 процентов", QMessageBox::Apply);
        return;
    }

    QSqlQuery qry;
    if(!qry.exec("UPDATE Заказ SET ПроцентКурьера = " + QString::number(ui-
>spinBox->value()) + " WHERE КодЗаказа = " + QString::number(order_id)))
    {
        QMessageBox::critical(nullptr, "Ошибка! ", "Не удалось изменить
процент!", QMessageBox::Apply);
    }
    else
    {
        QMessageBox::information(nullptr, "Успех ", "Процент изменен!",
QMessageBox::Apply);
        this->close();
        delete this;
    }
}

```

## 9. Объект «CustomDelegateView»

```

//buttondelegate.h
#ifndef BUTTONDELEGATE_H
#define BUTTONDELEGATE_H

#include <QObject>
#include <QItemDelegate>

class CustomDelegateView : public QItemDelegate
{
    Q_OBJECT
signals:
    QModelIndex signalClicked(QModelIndex index);
public:
    CustomDelegateView(QObject *parent = nullptr, QString text = "");
    void paint(QPainter *painter, const QStyleOptionViewItem &option, const
QModelIndex &index) const override;

```



```

    bool editorEvent(QEvent *event, QAbstractItemModel *model, const
QStyleOptionViewItem &option, const QModelIndex &index) override;
    void addButtonIndexes(const QModelIndex& indexes);
    void clearData();
private:
    QString text;
    QSet<QModelIndex> m_buttonIndexes;
};

```

```
//buttondelegate.cpp
```

```

#include "buttondelegate.h"
#include <QApplication>
#include <QPushButton>
#include <QPainter>
#include <QDebug>

```

```

CustomDelegateView::CustomDelegateView(QObject *parent, QString line) :
    QTableWidgetItem(parent), text(line){ }

```

```

void CustomDelegateView::paint(QPainter *painter, const QStyleOptionViewItem
&option, const QModelIndex &index) const
{
    QTableWidgetItem::paint(painter, option, index);

```

```

    QRect buttonRect = QRect(option.rect.left() + 1, option.rect.top() + 1,
option.rect.width() - 2, option.rect.height() - 2);
    QStyleOptionButton button;
    button.rect = buttonRect;
    button.state = QStyle::State_Enabled;
    button.text = text;

```

```

    QLinearGradient gradient(buttonRect.topLeft(), buttonRect.bottomLeft());
    if(m_buttonIndexes.contains(index))
    {
        gradient.setColorAt(0, QColor(242,133,255));
        gradient.setColorAt(0.8, QColor(145,80,153));
    }
    else
    {
        gradient.setColorAt(0, QColor(125,124,125));
        gradient.setColorAt(0.8, QColor(105,81,108));
    }

```

```
// Установка цвета фона и закругленных углов
```

```

painter->setRenderHint(QPainter::Antialiasing, true);
painter->setBrush(gradient); // Цвет фона
painter->setPen(Qt::NoPen); // Убираем обводку
painter->drawRoundedRect(button.rect, 10, 10); // Рисуем кнопку с
закругленными углами

```

```

// Рисуем текст кнопки
painter->setPen(Qt::white); // Цвет текста
painter->drawText(button.rect, Qt::AlignCenter, button.text);
}

```

```

bool CustomDelegateView::editorEvent(QEvent *event, QAbstractItemModel
*model, const QStyleOptionViewItem &option, const QModelIndex &index)
{
    if(event->type() == QEvent::MouseButtonRelease &&
m_buttonIndexes.contains(index)) {
        emit signalClicked(index);
    }
    else
        QItemDelegate::editorEvent(event, model, option, index);
    return true;
}

```

```

void CustomDelegateView::addButtonIndexes(const QModelIndex& indexes)
{ m_buttonIndexes.insert(indexes);}

```

```

void CustomDelegateView::clearData()
{ m_buttonIndexes.clear();}

```

```

#endif // BUTTONDELEGATE_H

```

## 10.Объект «chartsController»

```

//charts_controller.h
#ifndef CHARTS_CONTROLLER_H
#define CHARTS_CONTROLLER_H

#include <QObject>
#include <QChart>
#include <QLineSeries>
#include <QBarSeries>
#include <QWidget>
#include <QChartView>
#include <tuple>
#include <QBarSet>
#include <QStringList>

```

```

#include <QValueAxis>
#include <QDateTimeAxis>

class chartsController : public QWidget
{
    Q_OBJECT
public:
    chartsController(QWidget* parent = nullptr);
    ~chartsController();
    void init();
    void addNewLineSeries(QString line, QPen pen);
    void updateValues(QList<QString> lines);
    QChartView* line1_view();
    //QChartView* line2_view();
private:
    void getLineData(QString qry, QLineSeries* series);
    int max_value;
    QValueAxis *axisY;
    QDateTimeAxis *axisX;
    std::tuple<QChart*, QList<QLineSeries*>, QChartView*> line_chart;
};

#endif // CHARTS_CONTROLLER_H
//charts_controller.cpp
#include "charts_controller.h"
#include <QSqlQuery>
#include <QDate>

chartsController::chartsController(QWidget* parent) : QWidget(parent),
max_value(0)
{
    std::get<0>(line_chart) = new QChart();
    axisY = new QValueAxis();
    axisX = new QDateTimeAxis();
}

chartsController::~chartsController()
{
    delete std::get<0>(line_chart);
    foreach (QLineSeries* line, std::get<1>(line_chart)) {
        delete line;
    }
    delete std::get<2>(line_chart);
}

```

```

void chartsController::init()
{
    foreach (QLineSeries* series, std::get<1>(line_chart)) {
        std::get<0>(line_chart)->addSeries(series);
    }
    std::get<0>(line_chart)->legend()->hide();
    //![line chart settings]

    axisX->setFormat("yyyy-MM-dd");
    axisX->setTitleText("Дата");

    axisY->setTitleText("Значение");
    axisY->setMin(0);
    axisY->setMax(6);
    std::get<0>(line_chart)->addAxis(axisX, Qt::AlignBottom);
    std::get<0>(line_chart)->addAxis(axisY, Qt::AlignLeft);
    axisY->setTickCount(4);
    foreach (QLineSeries* series, std::get<1>(line_chart)) {
        series->attachAxis(axisX);
        series->attachAxis(axisY);
    }

    std::get<0>(line_chart)->setVisible(true);
    std::get<2>(line_chart) = new QChartView(std::get<0>(line_chart));
    std::get<2>(line_chart)->setRenderHint(QPainter::Antialiasing);
    std::get<2>(line_chart)->setRubberBand(QChartView::HorizontalRubberBand);
    std::get<2>(line_chart)->setVisible(true);
    //![line chart settings]
}

void chartsController::addNewLineSeries(QString line, QPen pen)
{
    QLineSeries* series = new QLineSeries();
    getLineData(line, series);
    series->setPen(pen);
    pen.setWidth(6);
    std::get<1>(line_chart).append(series);
}

QChartView *chartsController::line1_view()
{
    return std::get<2>(line_chart);
}

void chartsController::getLineData(QString qry_1, QLineSeries* series)

```

```

{
    QSqlQuery qry;
    qry.exec(qry_1);
    QDateTime prevDate = QDateTime();
    while(qry.next())
    {
        QDateTime date = QDateTime::fromString(qry.value(0).toString(), "yyyy-
MM-dd");

        if (prevDate.isValid()) {
            while (prevDate.addDays(1) < date) {
                prevDate = prevDate.addDays(1);
                series->append(prevDate.toMsecsSinceEpoch(), 0);
            }
        }

        max_value = std::max(max_value, qry.value(1).toInt());
        series->append(date.toMsecsSinceEpoch(), qry.value(1).toInt());

        prevDate = date;
    }
}

void chartsController::updateValues(QList<QString> lines)
{
    for(int i = 0; i < std::get<1>(line_chart).length(); ++i)
    {
        std::get<0>(line_chart)->removeSeries(std::get<1>(line_chart)[i]);
    }

    for (int i = 0; i < lines.size(); ++i)
    {
        if (i < std::get<1>(line_chart).size())
        {
            std::get<1>(line_chart)[i]->clear();
            getLineData(lines[i], std::get<1>(line_chart)[i]);
            std::get<0>(line_chart)->addSeries(std::get<1>(line_chart)[i]);
        }
    }

    std::get<0>(line_chart)->axes(Qt::Vertical).first()->setMax(max_value);
    foreach (QLineSeries* series, std::get<1>(line_chart)) {
        series->attachAxis(axisX);
        series->attachAxis(axisY);
    }
}

```

```
std::get<0>(line_chart)->update();
}
```

## 11. ОБЪЕКТ «DatabaseConnector»

```
//databaseconnector.h
#ifndef DATABASECONNECTOR_H
#define DATABASECONNECTOR_H

#include <QSqlDatabase>
#include <QSql>
#include <QDebug>

class DatabaseConnector : public QObject
{
public:
    DatabaseConnector();
    ~DatabaseConnector();
    QSqlDatabase connect(const QString& server,
                        const QString& dbName,
                        const QString& userName,
                        const QString& password,
                        const int& port);
    void disconnect();
    QSqlDatabase getDB();

private:
    QSqlDatabase db;
};

#endif // DATABASECONNECTOR_H

//databaseconnector.cpp
#include "databaseconnector.h"

DatabaseConnector::DatabaseConnector()
{
    db = QSqlDatabase::addDatabase("QPSQL");
}

DatabaseConnector::~DatabaseConnector()
{
    qDebug() << "Destroy Database!";
}
```

```

QSqlDatabase DatabaseConnector::connect(const QString &server, const QString
&databaseName, const QString &userName, const QString &password, const int&
port)
{
    db.setConnectOptions("sslmode=disable");
    db.setHostName(server);
    db.setDatabaseName(databaseName);
    db.setUserName(userName);
    db.setPassword(password);
    db.setPort(port);

    if(db.open()) {
        qDebug() << "Connected to database!";
        return db;
    }
    else {
        qDebug() << "Failed to connect to database!";
        return QSqlDatabase();
    }
}

void DatabaseConnector::disconnect()
{
    if (db.isOpen()) {
        db.close();
        qDebug() << "Disconnected from Database!";
    }
}

QSqlDatabase DatabaseConnector::getDB()
{
    return db;
}

```

## 12. ОБЪЕКТ «FreezeTableWidget»

```

//freezetablewidget.h
#ifndef FREEZETABLEWIDGET_H
#define FREEZETABLEWIDGET_H
#include <QTableView>
#include <QSqlQueryModel>
#include <QSqlTableModel>
#include "buttondelegate.h"
#include <QSortFilterProxyModel>
#include <functional>
#include <QList>

```

```

class FreezeTableWidget : public QTableView {
    Q_OBJECT

public:
    FreezeTableWidget(QWidget* parent = nullptr);
    ~FreezeTableWidget();
    void setModel();
    void updateValues();
    void init(QString data_query,
               QList<QString> queries_sort, QList<std::function<void(QModelIndex)>>
               funcs_delegate, QList<QString> texts_on_delegate);
protected:
    void resizeEvent(QResizeEvent *event) override;
    QModelIndex moveCursor(CursorAction cursorAction, Qt::KeyboardModifiers
    modifiers) override;
    void scrollTo (const QModelIndex & index, ScrollHint hint = EnsureVisible)
    override;

private:
    QString data_query;
    QList<QString> queries_sort;
    QTableView *frozenTableView;
    void init_style();
    void updateFrozenTableGeometry();
    QSqlTableModel* sql_model;
    void init_data();

    int ParentX;
    int ParentY;
    QList<CustomDelegateView*> delegates;
    QSortFilterProxyModel *proxyModel;
private slots:
    void updateSectionWidth(int logicalIndex, int oldSize, int newSize);
    void updateSectionHeight(int logicalIndex, int oldSize, int newSize);
};

#endif // FREEZETABLEWIDGET_H

//freezetablewidget.cpp
#include "freezetablewidget.h"
#include <QScrollBar>
#include <QHeaderView>
#include <QSqlQuery>
#include <QMessageBox>

```



```

#include <QDebug>
#include <QPainter>
#include <QFunctionPointer>
#include "buttondelegate.h"
#include <QLineEdit>
#include <QHBoxLayout>

FreezeTableWidget::FreezeTableWidget(QWidget* parent) : data_query("")
{
    ParentX = parent->pos().x();
    ParentY = parent->pos().y();
}

//! [constructor]
void FreezeTableWidget::init(QString data_query,
                             QList<QString> queries_sort,
                             QList<std::function<void(QModelIndex)>> funcs_delegate,
                             QList<QString> texts_on_delegate)
{
    if(funcs_delegate.size() != texts_on_delegate.size())
        return;
    this->data_query = data_query;
    if(queries_sort.empty())
    {
        for(int i = 0; i < funcs_delegate.size(); ++i)
            this->queries_sort.append(data_query);
    }
    else this->queries_sort = std::move(queries_sort);
    frozenTableView = new QTableView(this);
    sql_model = new QSqlTableModel(this);
    foreach(QString line, texts_on_delegate)
        delegates.append(new CustomDelegateView(this, line));
    proxyModel = new QSortFilterProxyModel(sql_model);

    //connect the headers and scrollbars of both tableviews together
    connect(horizontalHeader(),&QHeaderView::sectionResized, this,
            &FreezeTableWidget::updateSectionWidth);
    connect(verticalHeader(),&QHeaderView::sectionResized, this,
            &FreezeTableWidget::updateSectionHeight);
    connect(frozenTableView->verticalScrollBar(),
    &QAbstractSlider::valueChanged,
            verticalScrollBar(), &QAbstractSlider::setValue);
    connect(verticalScrollBar(), &QAbstractSlider::valueChanged,
            frozenTableView->verticalScrollBar(), &QAbstractSlider::setValue);
    for(int i = 0; i < funcs_delegate.size(); ++i)

```

```

        connect(delegates[i], &CustomDelegateView::signalClicked, delegates[i],
        funcs_delegate[i]);
    }
    //! [constructor]

```

```

FreezeTableView::~FreezeTableView()

```

```

{
    delete frozenTableView;
    foreach(CustomDelegateView* delegate, delegates) {
        delete delegate;
    }
    delete sql_model;
    delete proxyModel;
}

```

```

void FreezeTableView::setModel()

```

```

{
    this->setSortingEnabled(true);
    TableView::setModel(sql_model);
    frozenTableView->setModel(model());
    proxyModel->setSourceModel(sql_model);
    TableView::setModel(proxyModel);
    frozenTableView->setModel(model());
    connect(this->horizontalHeader(), &QHeaderView::sectionClicked, this,
    [this](int section)

```

```

    {
        QHeaderView* headerView = this->horizontalHeader();
        QWidget* sortWidget = new QWidget(this, Qt::Popup);

```

```

        // Устанавливаем стиль для sortWidget

```

```

        sortWidget->setStyleSheet("background-color: rgba(1, 50, 125, 128);"
                                "color: white;"
                                "font-family: Marmelad;"
                                "font-size: 20px;");

```

```

        QHBoxLayout* lo = new QHBoxLayout(sortWidget);
        QLineEdit* le = new QLineEdit(sortWidget);

```

```

        connect(le, &QLineEdit::textChanged, this, [this, section](const QString&
        toSort) {
            if (toSort.isEmpty()) {
                proxyModel->setFilterFixedString("");
                proxyModel->sort(section);
            } else {
                proxyModel->setFilterKeyColumn(section);

```

```

        proxyModel->setFilterFixedString(toSort);
        proxyModel->sort(section);
    }
});

lo->addWidget(le);
sortWidget->setLayout(lo);

// Определяем размер и позицию столбца
const int columnWidth = headerView->sectionSize(section); // Ширина
столбца
const int columnPos = headerView->sectionPosition(section); // Позиция
столбца на экране

// Учитываем прокрутку таблицы
const int horizontalOffset = this->horizontalScrollBar()->value();

// Позиционируем sortWidget над столбцом
const int sx = columnPos - horizontalOffset + (ParentX + this->pos().x());
const int sy = ParentY + this->pos().y() - sortWidget->height(); // Отступ
сверху для размещения над столбцом

// Устанавливаем размер и показываем форму
sortWidget->resize(columnWidth, 40);
sortWidget->move(sx, sy);
sortWidget->show();
connect(this->horizontalHeader(), &QHeaderView::sectionClicked, this, [this,
sortWidget, section](int clicked)
{
    proxyModel->setFilterFixedString("");
    proxyModel->sort(section);
});
});
init_style();
}

void FreezeTableWidget::updateValues()
{
    init_data();
}

//! [init part1]
void FreezeTableWidget::init_style()
{

```

```

frozenTableView->setFocusPolicy(Qt::NoFocus);
frozenTableView->verticalHeader()->hide();
frozenTableView->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Fixed);
this->verticalHeader()->hide();
this->setStyleSheet("QTableView { border: none;"
    "background-color: #D9D9D9;"
    "font-family: Marmelad;"
    "font-size: 20px;"
    "selection-background-color: #995067"
    "}"
    "QHeaderView::section"
    "{"
    "background-color: #D9D9D9;"
    "color: black;"
    "font-family: Marmelad;"
    "font-size: 20px;"
    "border: none;"
    "}"
    );
viewport()->stackUnder(frozenTableView);
//! [init part1]

//! [init part2]
frozenTableView->setStyleSheet("QTableView { border: none;"
    "background-color: rgba(242, 133, 255, 255);"
    "font-family: Marmelad;"
    "font-size: 20px;"
    "selection-background-color: rgba(242, 133, 255, 255);"
    "}"
    "QHeaderView::section"
    "{"
    "background-color: #D9D9D9;"
    "color: black;"
    "font-family: Marmelad;"
    "font-size: 20px;"
    "border: none;"
    "}");
for (int col = 1; col < model()->columnCount(); ++col)
    frozenTableView->setColumnHidden(col, true);

frozenTableView->setColumnWidth(0, columnWidth(0) );

frozenTableView->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
frozenTableView->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);

```

```

    frozenTableView->show();
    updateFrozenTableGeometry();
    this->horizontalHeader()-
>setSectionResizeMode(QHeaderView::ResizeToContents);
    frozenTableView->horizontalHeader()-
>setSectionResizeMode(QHeaderView::ResizeToContents);
    setHorizontalScrollMode(ScrollPerPixel);
    setVerticalScrollMode(ScrollPerPixel);
    frozenTableView->setVerticalScrollMode(ScrollPerPixel);
}
//! [init part2]

//! [sections]
void FreezeTableWidget::updateSectionWidth(int logicalIndex, int /* oldSize */, int
newSize)
{
    if (logicalIndex == 0){
        frozenTableView->setColumnWidth(0, newSize);
        updateFrozenTableGeometry();
    }
}

void FreezeTableWidget::updateSectionHeight(int logicalIndex, int /* oldSize */,
int newSize)
{
    frozenTableView->setRowHeight(logicalIndex, newSize);
}

//! [sections]

//! [resize]
void FreezeTableWidget::resizeEvent(QResizeEvent * event)
{
    QTableView::resizeEvent(event);
    updateFrozenTableGeometry();
}
//! [resize]

//! [navigate]
QModelIndex FreezeTableWidget::moveCursor(CursorAction cursorAction,
Qt::KeyboardModifiers modifiers)
{
    QModelIndex current = QTableView::moveCursor(cursorAction, modifiers);

    if (cursorAction == MoveLeft && current.column() > 0

```

```

        && visualRect(current).topLeft().x() < frozenTableView->columnWidth(0) ){
        const    int    newValue    =    horizontalScrollBar()->value()    +
visualRect(current).topLeft().x()
        - frozenTableView->columnWidth(0);
        horizontalScrollBar()->setValue(newValue);
    }
    return current;
}
//! [navigate]

```

```

void FreezeTableWidget::scrollTo (const QModelIndex & index, ScrollHint hint){
    if (index.column() > 0)
        QTableView::scrollTo(index, hint);
}

```

```

//! [geometry]
void FreezeTableWidget::updateFrozenTableGeometry()
{
    frozenTableView->setGeometry(verticalHeader()->width() + frameWidth(),
                                frameWidth(), columnWidth(0),
                                viewport()->height()+horizontalHeader()->height());
}

```

```

void FreezeTableWidget::init_data()
{
    foreach(CustomDelegateView* del, delegates)
        del->clearData();
    //sql_model->clear();
    QSqlQuery qry;
    if(!qry.exec(data_query)) {qDebug() << "data_qry"; }
    sql_model->setQuery(std::move(qry));
    for(int i = 0; i < delegates.size(); ++i)
    {
        sql_model->insertColumns(sql_model->columnCount(), 1);
        sql_model->setHeaderData(sql_model->columnCount()-1, Qt::Horizontal, tr("
"));
    }
    for(int i = 0; i < delegates.size(); ++i)
    {
        QSqlQuery qry_review;
        if(!qry_review.exec(queries_sort[i])) {qDebug() << "review error";}
        QSet<int> existingOrderIds;
        while(qry_review.next()) {
            existingOrderIds.insert(qry_review.value(0).toInt());
        }
    }
}

```

```

    }
    for (int row = 0; row < this->model()->rowCount(); ++row) {
        int orderId = this->model()->data(this->model()->index(row, 0)).toInt();
        if (existingOrderIds.contains(orderId)) {
            delegates[i]->addButtonIndexes(this->model()->index(row, sql_model-
>columnCount()-(delegates.size()-i)));
        }
    }
    this->setItemDelegateForColumn(sql_model->columnCount()-(
delegates.size()-i), delegates[i]);
}
}

```

### 13.Объект «MapDistanceCalculator»

```

#ifndef MAPDISTANCECALCULATOR_H
#define MAPDISTANCECALCULATOR_H
#include <QObject>
#include <QNetworkAccessManager>
#include <QNetworkReply>
#include <QPair>
#include <QVector>

class MapDistanceCalculator : public QObject
{
    Q_OBJECT
public:
    explicit MapDistanceCalculator(QObject* parent = nullptr, QString distance_key
= "", QString geocode_key = "");
    ~MapDistanceCalculator();
    void setCoordinatesOfAddress(QString address);
    void setDistance();
    int getResult();
private slots:
    void handleDistanceResponse(QNetworkReply *reply);
    void handleGeocodeResponse(QNetworkReply *reply);
private:
    int result;
    QVector<QPair<double, double>> current_way;
    QNetworkAccessManager* geocode_manager;
    QNetworkAccessManager* distance_network_manager;
    QString api_geocode_key;
    QString api_distance_key;
signals:
    void API_answer();
    void enter_uncorrect_data();

```

```

    void error_distance_calculator();
};

#endif // MAPDISTANCECALCULATOR_H

//mapdistancecalculator.cpp

#include "mapdistancecalculator.h"
#include <QJsonDocument>
#include <QJsonObject>
#include <QJsonArray>

MapDistanceCalculator::MapDistanceCalculator(QObject *parent,   QString
distance_key,   QString   geocode_key)   :   QObject(parent),
api_distance_key(distance_key), result(-1), api_geocode_key(geocode_key)
{
    geocode_manager = new QNetworkAccessManager();
    distance_network_manager = new QNetworkAccessManager();
    connect(geocode_manager,   &QNetworkAccessManager::finished,   this,
    &MapDistanceCalculator::handleGeocodeResponse);
    connect(distance_network_manager, &QNetworkAccessManager::finished, this,
    &MapDistanceCalculator::handleDistanceResponse);
}

MapDistanceCalculator::~MapDistanceCalculator()
{
    delete geocode_manager;
    delete distance_network_manager;
}

void MapDistanceCalculator::setDistance()
{
    if(current_way.size() != 2)
    {
        qDebug() << "need 2 coordinate for get distance. Use setCoordinatesOfAddress
for set 1 point";
        return;
    }
    QNetworkRequest
request(QString("https://api.distancematrix.ai/maps/api/distancematrix/json?"
                "origins=%1,%2"
                "&destinations=%3,%4"
                "&key=%5")
        .arg(current_way[0].first)
        .arg(current_way[0].second)

```



```

        .arg(current_way[1].first)
        .arg(current_way[1].second)
        .arg(api_distance_key));
distance_network_manager->get(request);
}

int MapDistanceCalculator::getResult()
{
    return this->result;
}

void MapDistanceCalculator::setCoordinatesOfAddress(QString address)
{
    if(current_way.size() == 2)
    {
        qDebug() << "you can add only two points";
        return;
    }
    QNetworkRequest request(QString("https://geocode-
maps.yandex.ru/1.x/?apikey=%1&geocode=%2&results=1&format=json")
        .arg(api_geocode_key)
        .arg(QUrl::toPercentEncoding(address)));
    geocode_manager->get(request);
}

void MapDistanceCalculator::handleDistanceResponse(QNetworkReply *reply)
{
    QByteArray response = reply->readAll();
    QJsonDocument jsonDoc = QJsonDocument::fromJson(response);
    QJsonObject rootObj = jsonDoc.object();

    // Извлекаем массив "rows"
    QJsonArray rowsArray = rootObj["rows"].toArray();

    // Перебираем элементы массива rows (в данном случае один элемент)
    for (const QJsonValue &rowValue : rowsArray) {
        QJsonObject rowObj = rowValue.toObject();

        // Извлекаем массив "elements"
        QJsonArray elementsArray = rowObj["elements"].toArray();

        if(elementsArray.empty()) emit error_distance_calculator();

        for (const QJsonValue &elementValue : elementsArray) {
            QJsonObject elementObj = elementValue.toObject();

```

```

// Извлекаем объект "distance" и его значение "value"
QJsonObject distanceObj = elementObj["distance"].toObject();
int distanceValue = distanceObj["value"].toInt();

qDebug() << "result найден: " << distanceValue;
this->result = distanceValue;
}
}
current_way.clear();
emit API_answer();
}
void MapDistanceCalculator::handleGeocodeResponse(QNetworkReply *reply)
{
    QByteArray response = reply->readAll();
    QJsonDocument jsonDoc = QJsonDocument::fromJson(response);
    if (jsonDoc.isObject()) {
        QJsonObject rootObj = jsonDoc.object();
        QJsonObject responseObj = rootObj["response"].toObject();
        QJsonObject geoObjectCollectionObj = responseObj["GeoObjectCollection"].toObject();
        QJsonArray featureMemberArray = geoObjectCollectionObj["featureMember"].toArray();
        if (!featureMemberArray.isEmpty()) {
            QJsonObject geoObject = featureMemberArray[0].toObject()["GeoObject"].toObject();
            QJsonObject pointObj = geoObject["Point"].toObject();
            QString pos = pointObj["pos"].toString();
            QStringList coordinates = pos.split(" ");
            if (coordinates.size() == 2) {
                double latitude = coordinates[0].toDouble();
                double longitude = coordinates[1].toDouble();
                current_way.append(QPair<double, double>{latitude, longitude});
                qDebug() << "Latitude:" << latitude << "Longitude:" << longitude;
            } else qWarning() << "Ошибка: Неверный формат координат.";
        }
        else
        {
            emit enter_uncorrect_data();
            return;
        }
    }
    emit API_answer();
}

```

## 14.Объект «optionButton»

```

//optionbutton.cpp
#ifndef OPTIONBUTTON_H
#define OPTIONBUTTON_H

#include <QObject>
#include <QPushButton>

class optionButton : public QObject
{
    Q_OBJECT
public:
    optionButton(QObject *parent = nullptr);
    void addButton(QPushButton* object);
    //~optionButton();
    QPushButton* active;
    QString name_of_button(QPushButton* btn);
private:
    QMap<QPushButton*, QString> names;
    void handleButtonClicked(QPushButton *button);
};

#endif // OPTIONBUTTON_H
//optionbutton.cpp
#include "optionbutton.h"

optionButton::optionButton(QObject *parent) : active(nullptr), QObject(parent) {}

void optionButton::addButton(QPushButton *object)
{
    connect(object, &QPushButton::clicked, this, [this, object]() {
        handleButtonClicked(object);
    });
    names[object] = object->text();
}

QString optionButton::name_of_button(QPushButton *btn)
{
    return names[btn];
}

void optionButton::handleButtonClicked(QPushButton *button)
{
    if (active == button) {

```

```

    active = nullptr;
    button->setStyleSheet("QPushButton{ "
        "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1,
y2:1, stop:0 rgba(105, 224, 254, 255), stop:1 rgba(63, 134, 152, 255));"
        "border-radius: 20px;"
        "font-family: Marmelad;"
        "color: white;"
        "}");

    return;
}
if (active) {
    active->setStyleSheet("QPushButton{ "
        "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1,
y2:1, stop:0 rgba(105, 224, 254, 255), stop:1 rgba(63, 134, 152, 255));"
        "border-radius: 20px;"
        "font-family: Marmelad;"
        "color: white;"
        "}");
}

active = button;

active->setStyleSheet(
    "QPushButton{ "
    "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:1, stop:0
rgba(242, 133, 255, 255), stop:1 rgba(63, 134, 152, 255));"
    "border-radius: 20px;"
    "font-family: Marmelad;"
    "color: white;"
    "}");
}

```

## 15. ОБЪЕКТ «PasswordLine»

```

//passwordline.h
#ifndef PASSWORDLINE_H
#define PASSWORDLINE_H

#include <QAction>
#include <QLineEdit>
#include <QToolButton>

class PasswordLine : public QLineEdit
{
    Q_OBJECT

```

```

public:
    PasswordLine(QWidget* parent = nullptr);
private slots:
    void onPressed();
    void onReleased();

protected:
    void enterEvent(QEnterEvent *event);
    void leaveEvent(QEvent *event);
    void focusInEvent(QFocusEvent *event);
    void focusOutEvent(QFocusEvent *event);
private:
    QToolButton *button;
};

#endif // PASSWORDLINE_H
//passwordline.cpp
#include "passwordline.h"

PasswordLine::PasswordLine(QWidget *parent) : QLineEdit(parent) {
    setEchoMode(QLineEdit::Password);
    setClearButtonEnabled(true);
    QAction *action = addAction(QIcon(":/resources/icons8-close-eye-48.png"),
    QLineEdit::TrailingPosition);
    button = qobject_cast<QToolButton *>(action->associatedObjects().last());
    button->setCursor(QCursor(Qt::PointingHandCursor));
    connect(button, &QToolButton::pressed, this, &PasswordLine::onPressed);
    connect(button, &QToolButton::released, this, &PasswordLine::onReleased);
}

void PasswordLine::onPressed() {
    QToolButton *button = qobject_cast<QToolButton *>(sender());
    button->setIcon(QIcon(":/resources/icons8-eye-64.png"));
    setEchoMode(QLineEdit::Normal);
}

void PasswordLine::onReleased() {
    QToolButton *button = qobject_cast<QToolButton *>(sender());
    button->setIcon(QIcon(":/resources/icons8-close-eye-48.png"));
    setEchoMode(QLineEdit::Password);
}

void PasswordLine::enterEvent(QEnterEvent *event)
{
    QLineEdit::enterEvent(event);
}

```

```

}

void PasswordLine::leaveEvent(QEvent *event) {
    QLineEdit::leaveEvent(event);
}

void PasswordLine::focusInEvent(QFocusEvent *event) {
    QLineEdit::focusInEvent(event);
}

void PasswordLine::focusOutEvent(QFocusEvent *event) {
    QLineEdit::focusOutEvent(event);
}

```

#### 16.Объект «Role»

```

//role.h
#ifndef ROLES_H
#define ROLES_H

```

```

enum Role
{
    USER,
    MANAGER,
    COURIER
};

```

```

#endif // ROLES_H

```

#### 17.Объект «styleHelper»

```

#ifndef STYLEHELPER_H
#define STYLEHELPER_H
#include <QString>

class styleHelper
{
public:
    static QString addProjectFont(QString);
    static QString addTextStyle();
    static QString addPushButtonStyle();
    static QString addPushButtonStyle2();
    static QString addPushButtonStyleUse();
    static QString userStyle();
};

```

```
#endif // STYLEHELPER_H
```

```
#include "stylehelper.h"
```

```
QString styleHelper::addProjectFont(QString color_text)
```

```
{
    return "QLabel{ "
        "font-family: Marmelad;"
        "font-size : 20 px;"
        "color: "+color_text +";"
        "}"
}
```

```
QString styleHelper::addTextStyle()
```

```
{
    return "QLineEdit{ "
        "background: none;"
        "background-color: rgb(217, 217, 217);"
        "border: none;"
        "font-family: Marmelad;"
        "color: black;"
        "}"
}
```

```
QString styleHelper::addPushButtonStyle()
```

```
{
    return "QPushButton{ "
        "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:0,
stop:0.157303 rgba(242, 133, 255, 255), stop:1 rgba(145, 80, 153, 255));"
        "border-radius: 20px;"
        "font-family: Marmelad;"
        "color: white;"
        "}"
}
```

```
QString styleHelper::addPushButtonStyle2()
```

```
{
    return "QPushButton{ "
        "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:1, stop:0
rgba(105, 224, 254, 255), stop:1 rgba(63, 134, 152, 255));"
        "border-radius: 20px;"
        "font-family: Marmelad;"
        "color: white;"
        "}"
}
```

```
}
```

```
QString styleHelper::addPushButtonStyleUse()
```

```
{
```

```
    return "QPushButton: hover{ "
        "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:1, stop:0
        rgba(242, 133, 255, 255), stop:1 rgba(63, 134, 152, 255));"
        "}"
```

```
}
```

```
QString styleHelper::userStyle()
```

```
{
```

```
    return "QPushButton{ "
        "qproperty-icon: url(qrc:/resources/rocket.png);"
        "}"
```

```
}
```