ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет Высшая школа бизнеса
Образовательная программа “Бизнес-информатика”

Проект “Управление требованиями и проектирование ИС”
«Приложение для доставки еды»

Проект выполнили:
Горяев Даян ББИ225, 25%
Патаев Арслан ББИ221, 25%
Икромов Навруз ББИ223, 25%
Чункуров Тимур ББИ223, 25%

Москва,2024

Содержание проекта

Содержание проекта..... 1
1 Высокоуровневые требования для проекта приложения по доставке еды..... 2
1.1 Описание проекта..... 2
1.2 Список вопросов..... 3
2 Техническое задание (ТЗ)..... 6
2.1 Общие сведения..... 6
2.2 Цель проекта..... 6
2.3 Область применения..... 6
2.4 Функциональные требования..... 6
2.5 Нефункциональные требования..... 7
2.6 Приоритизация требований (MoSCoW метод)..... 7
2.7 Ограничения..... 8
2.8 Критерии качества..... 8
3 Выбор модели и методологии..... 9
3.1 Выбор модели..... 9
3.2 Выбор методологии..... 9
4 Диаграмма прецедентов..... 11
4.1 Основные акторы..... 11
4.2 Прецеденты (основные действия)..... 12
5 Выявление первичных классов..... 14
5.1 Restaurant..... 14
5.2 Dish..... 14
5.3 Menu..... 15
5.4 Order..... 15
5.5 Customer..... 15
5.6 Address..... 16
5.7 Location..... 16
5.8 Courier..... 16

5.9 SpecialOffer..... 17
5.10 PaymentMethod..... 17
5.11 DeliverySystem..... 18
5.12 Feedback..... 18
6 Детальная диаграмма классов.....20
6.1 Общее описание..... 20
6.2 Список классов и их атрибутов/методов..... 20
6.3 Диаграмма классов..... 24
7 Динамика системы (последовательность, взаимодействия, состояния.....28
7.2 Диаграммы состояний для классов..... 31
7.2.1 Диаграмма состояний для класса Order..... 31
7.2.2 Диаграмма состояний для класса Customer..... 32
7.2.3 Диаграмма состояний для класса Courier..... 34
7.2.4 Диаграмма состояний для класса System.....35
8 Расширенная матрица трассировки требований:..... 36
8.1 Бизнес требования..... 36
8.2 Функциональные требования..... 39
8.3 Нефункциональные требования..... 41
8.4 Выводы:.....42
9 План реализации тестирования и сопровождения.....44
9.1 План реализации..... 44
9.2 План тестирования для приложения по доставке еды. 46
9.3 План эксплуатации.....49
9.4 Документ по сопровождению программного продукта... 52
10 Вывод.....55

1 Высокоуровневые требования для проекта приложения по доставке еды

1.1 Описание проекта

Приложение должно предоставлять возможность пользователям выбирать и заказывать блюда из различных ресторанов с последующей доставкой до места назначения. Пользователь должен иметь доступ к каталогу ресторанов с возможностью фильтрации по кухне, цене, времени доставки, рейтингу и другим параметрам. Пользователь должен иметь возможность формировать заказ, добавлять или удалять блюда из корзины, а также оплачивать заказ через приложение, используя банковские карты или другие популярные платежные системы. После оформления заказа пользователь должен иметь возможность отслеживать статус доставки в реальном времени, включая этапы: принятие заказа рестораном, приготовление, передачу курьеру и доставку до двери.

Приложение должно предоставлять возможность пользователям оставлять отзывы и оценки как ресторанам, так и курьерам. Пользователь должен иметь доступ к истории своих заказов, включая информацию о заказанных блюдах, ресторанах и затраченных средствах.В случае проблем или вопросов должна быть доступна онлайн-поддержка через чат или телефон. Приложение должно поддерживать систему скидок и акций, предоставляемых ресторанами или платформой, с возможностью их активации при оформлении заказа. Приложение должно поддерживать несколько языков, обеспечивая удобство использования для пользователей из разных регионов.

Система должна интегрироваться с курьерскими службами для автоматического назначения курьеров и оптимизации маршрутов доставки.Приложение должно отправлять уведомления пользователю о статусе его заказа (принят, готовится, в пути, доставлен). Приложение должно быть интегрировано с картографическими сервисами для отслеживания маршрута доставки.

Приложение должно соответствовать стандартам безопасности при работе с данными пользователей и платежной информацией.

1.2 Список вопросов

1. **Каковы основные бизнес-цели проекта?**

(Поможет определить, какие задачи проект должен решить для бизнеса.)

2. **Какие основные функции должна выполнять информационная система?**

(Позволит понять, какие ключевые возможности и процессы должна поддерживать

система.)

3. **Какие существуют ограничения по времени и бюджету?**

(Важны для понимания границ проекта.)

4. **Каковы целевые пользователи системы? Кто будет работать с ней?**

(Поможет определить типы пользователей и их потребности.)

5. **Какие основные процессы работы с клиентами должна поддерживать система**

(например, оформление заказа, доставка, оплата)?

(Позволит идентифицировать ключевые пользовательские сценарии.)

6. **Какие требования по интеграции с внешними системами (например,**

платежные системы, системы управления доставкой)?

(Выясняются необходимые интеграции с внешними сервисами.)

7. **Каковы основные показатели производительности системы?**

(Определение нефункциональных требований, таких как скорость работы

системы.)

8. **Какую информацию система должна собирать и обрабатывать для**

аналитики?

(Важный аспект для бизнеса, связанный с анализом данных.)

9. **Какие способы оплаты должны поддерживаться в системе?**

(Понимание функционала, связанного с финансовыми операциями.)

10. **Есть ли требования по безопасности данных? Если да, то какие?**

(Выясняются требования к защите данных, особенно в случае работы с платежной

информацией.)

11. **Какие системы или процессы планируется автоматизировать с помощью**

данного решения?

(Определение задач, которые можно автоматизировать.)

12. **Каковы требования к масштабируемости системы (например, расширение**

функционала, увеличение числа пользователей)?

(Поможет учесть будущие потребности бизнеса.)

13. **Какие требования к мобильности и доступности системы? Нужно ли**

разрабатывать мобильное приложение или адаптировать систему под

мобильные устройства?

(Выясняется необходимость мобильной версии.)

14. **Какова текущая ИТ-инфраструктура компании? С какими существующими**

системами необходимо будет интегрироваться?

(Понимание существующей технической среды и требований к интеграции.)

15. **Какие показатели качества системы (например, надежность, отказоустойчивость, доступность) важны для вас?**

(Позволяет задать основные атрибуты качества системы.)

2 Техническое задание (ТЗ)

2.1 Общие сведения

- **Название проекта:** Приложение для доставки еды
- **Заказчик:** (Название компании)
- **Дата:** 12.09.2024

2.2 Цель проекта

Приложение должно предоставить пользователям возможность выбора и заказа блюд из различных ресторанов с доставкой до места назначения. Система должна быть интуитивно понятной и интегрированной с внешними системами, обеспечивающими обработку заказов и доставку.

2.3 Область применения

Приложение используется клиентами, ресторанами и курьерскими службами для управления заказами, доставкой и платежами.

2.4 Функциональные требования

1. **Регистрация и авторизация пользователей**
    - **Обязательное:** Поддержка регистрации и авторизации через email и социальные сети.
    - **Опциональное:** Регистрация через Google или Apple ID — опционально для упрощения процесса регистрации, если у пользователя уже есть привязка.
2. **Каталог ресторанов и блюд**
    - **Обязательное:** Доступ к каталогу ресторанов с фильтрацией по типу кухни, рейтингу, цене и времени доставки.
    - **Опциональное:** Фильтрация по вегетарианским или диетическим блюдам — опционально, так как это специфические требования.
3. **Корзина и оформление заказа**
    - **Обязательное:** Возможность добавлять/удалять блюда в корзину и оформлять заказ.
    - **Обязательное:** Выбор способа оплаты (банковская карта, электронные кошельки).
4. **Отслеживание доставки**

    - **Обязательное:** Отслеживание статуса заказа в реальном времени.
    - **Обязательное:** Уведомления на всех этапах заказа (принят, готовится, передан курьеру, доставлен).
5. **Отзывы и оценки**
    - **Желательные:** Возможность оставлять отзывы и оценки ресторанам и курьерам.
6. **История заказов**
    - **Обязательное:** Доступ к истории заказов.
7. **Интеграция с картографическими сервисами**
    - **Желательные:** Интеграция с картографическими сервисами для отслеживания маршрута курьера.
8. **Скидки и акции**
    - **Желательные:** Поддержка скидок и акций — опционально, так как это не обязательно для базовой работы приложения, но улучшает пользовательский опыт.

2.5 Нефункциональные требования

1. **Производительность**
    - **Обязательное:** Система должна поддерживать до 1000 одновременных пользователей без заметной потери производительности.
2. **Безопасность**
    - **Обязательное:** Приложение должно соответствовать стандартам безопасности при работе с платежной информацией.
3. **Масштабируемость**
    - **Обязательное:** Система должна быть масштабируемой для поддержки новых функций и увеличения числа пользователей.
4. **Доступность**
    - **Обязательное:** Приложение должно поддерживать как веб-версию, так и мобильные платформы (iOS и Android).

2.6 Приоритизация требований (MoSCoW метод)

- **Must have (Обязательные требования):**
    - Регистрация и авторизация
    - Каталог ресторанов

    - Корзина и оформление заказа
    - Отслеживание доставки
    - Уведомления
    - Безопасность данных
    - История заказов
- **Should have (Желательные требования):**
    - Интеграция с картографическими сервисами
    - Отзывы и оценки
    - Скидки и акции
- **Could have (Опциональные требования):**
    - Фильтрация по диетическим блюдам
    - Расширенные способы регистрации
- **Won’t have (Не будет в этой версии):**
    - Глубокая аналитика пользователей (отложено на следующие версии)

2.7 Ограничения

- **Время разработки:** 6 месяцев
- **Бюджет:** $50000

2.8 Критерии качества

- Надежность: минимальные простои.
- Производительность: отклик системы в течение 3 секунд при любых нагрузках.
- Безопасность: соответствие стандартам PCI DSS для работы с платежами.

3 Выбор модели и методологии

3.1 Выбор модели

Наиболее подходящей моделью для проекта по созданию приложения для доставки еды является инкрементная модель.

Обоснование выбора инкрементной модели:

1. Разделение на этапы: Проект приложения по доставке еды можно разделить на несколько этапов, где каждый инкремент вносит определенную функциональность, например, первый инкремент включает базовые функции (каталог ресторанов, оформление заказа), второй – систему оплаты и доставки, третий – отзывы и рейтинги, и так далее.
2. Плавный ввод функциональности: Инкрементная модель позволяет постепенно добавлять новые функции, что позволяет выпускать работающие версии приложения на каждом этапе и получать обратную связь от пользователей.
3. Гибкость: Каждый инкремент может корректироваться и совершенствоваться на основе отзывов пользователей и анализа рынка.
4. Минимизация рисков: С каждой итерацией команда может выявлять проблемы и риски, избегая их накопления на финальных этапах разработки.
5. Экономия ресурсов: Начало работы над базовой функциональностью приложения позволяет быстрее выйти на рынок и получать доход, который можно направить на дальнейшее развитие проекта.

Таким образом, инкрементная модель хорошо подходит для данного проекта, так как обеспечивает ранний выпуск продукта и гибкость в развитии функционала.

3.2 Выбор методологии

Для проекта приложения по доставке еды оптимальной методологией разработки будет Scrum. Это объясняется следующими причинами:

1. Гибкость: Scrum позволяет разделить проект на короткие временные отрезки — спринты, в течение которых команда фокусируется на

создании конкретного набора функций. Это соответствует потребности в выпуске новой функциональности в сжатые сроки.
2. Постоянная обратная связь: После каждого спринта продукт может быть продемонстрирован заказчику и пользователям, что дает возможность вовремя выявить проблемы и внести изменения.
3. Роли и ответственность: Scrum вводит четкие роли (Product Owner, Scrum Master, команда), что помогает структурировать процесс работы над проектом. Product Owner в данном случае может собирать требования и обратную связь от пользователей и заказчиков, корректируя приоритеты.
4. Прозрачность: С помощью ежедневных митингов и демонстраций по завершению спринтов все участники проекта будут в курсе текущего статуса и проблем.
5. Приоритеты и планирование: Методология Scrum позволяет постоянно пересматривать приоритеты и подстраивать план проекта под актуальные требования. Мб

Использование Scrum в данном проекте обеспечит прозрачное и гибкое управление процессом разработки, поможет команде быстрее реагировать на изменения и выпустить конкурентоспособное приложение на рынок.

4 Диаграмма прецедентов

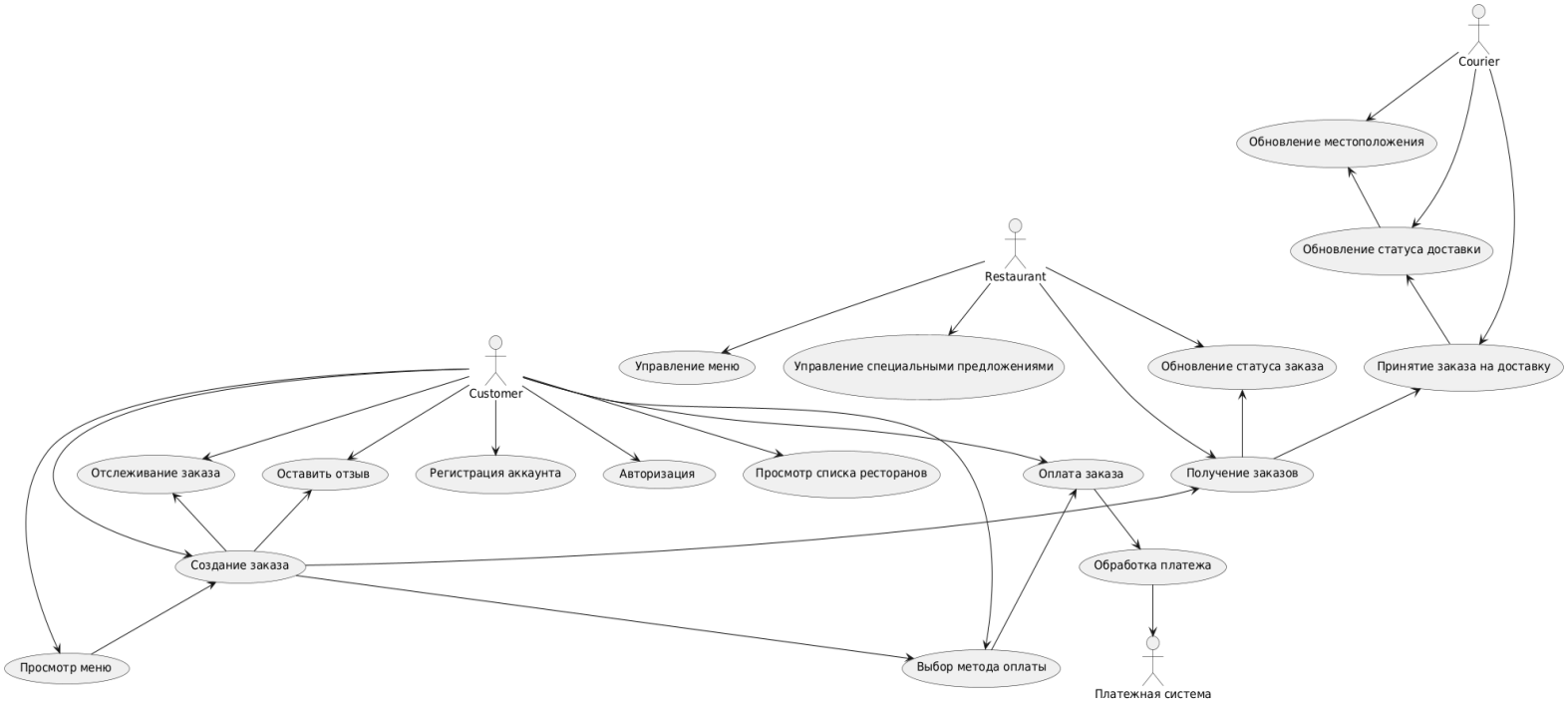


Диаграмма прецедентов на изображении отражает ключевые взаимодействия в системе для приложения по доставке еды между пользователями, рестораном, курьерами и платежной системой. В центре находится пользователь (Customer), который выполняет различные действия в системе. Прецеденты и их взаимодействия показаны стрелками, исходящими от акторов (пользователей системы) к функциональным возможностям системы.

4.1 Основные акторы

1. Customer (Покупатель) — основной пользователь системы, взаимодействующий с функциями заказа.

2. Restaurant (Ресторан) — участник, который управляет меню и специальными предложениями, а также обрабатывает заказы.

3. Courier (Курьер) — отвечает за принятие заказов на доставку и обновление информации о доставке.

4. Платежная система — участвует в обработке платежей, взаимодействует с системой оплаты.

4.2 Прецеденты (основные действия)

1. Покупатель (Customer):

- Регистрация аккаунта
- Авторизация
- Просмотр меню и списка ресторанов
- Создание заказа
- Выбор метода оплаты и оплата заказа
- Отслеживание заказа
- Оставить отзыв

2. Ресторан (Restaurant):

- Управление меню
- Управление специальными предложениями
- Обновление статуса заказа
- Получение заказов

3. Курьер (Courier):

- Принятие заказа на доставку
- Обновление статуса доставки
- Обновление местоположения

4. Платежная система:

- Обработка платежа

Основные взаимодействия:

- Пользователь взаимодействует с рестораном и курьером, создавая заказы, оплачивая их и отслеживая процесс доставки.

- Ресторан обновляет меню, принимает заказы и обновляет их статус.

- Курьер отвечает за доставку, отслеживает местоположение и обновляет статус заказа.

- Платежная система участвует в обработке оплаты, взаимодействуя с покупателем и рестораном.

5 Выявление первичных классов

5.1 Restaurant

- **Атрибуты:**
  - id: Int — идентификатор ресторана.
  - name: String — название ресторана.
  - menu: Menu — меню ресторана.
  - workingHours: String — часы работы.
  - specialOffers: List<SpecialOffer> — список специальных предложений (если применимо).
- **Методы:**
  - getMenu(): Menu — возвращает меню ресторана.
  - setMenu(menu: Menu): void — устанавливает меню ресторана.
  - getSpecialOffers(): List<SpecialOffer> — возвращает список специальных предложений.
  - addSpecialOffer(offer: SpecialOffer): void — добавляет специальное предложение.
  - removeSpecialOffer(offer: SpecialOffer): void — удаляет специальное предложение.
- **Тип связи:** Агрегация с Menu и SpecialOffer.

5.2 Dish

- **Атрибуты:**
  - id: Int — идентификатор блюда.
  - name: String — название блюда.
  - description: String — описание блюда.
  - price: Double — цена блюда.
  - ingredients: List<String> — список ингредиентов блюда.
- **Методы:**
  - getIngredients(): List<String> — возвращает список ингредиентов блюда.

- **Тип связи:** Агрегируется в Menu и Order.

5.3 Menu

- **Атрибуты:**
  - items: List<Dish> — список блюд в меню.
- **Методы:**
  - addItem(item: Dish): void — добавляет новое блюдо в меню.
  - removeItem(item: Dish): void — удаляет блюдо из меню.
  - getItems(): List<Dish> — возвращает список всех блюд.
- **Тип связи:** Агрегация с Restaurant.

5.4 Order

- **Атрибуты:**
  - id: Int — идентификатор заказа.
  - customer: Customer — клиент, сделавший заказ.
  - restaurant: Restaurant — ресторан, из которого заказаны блюда.
  - dishes: List<Dish> — список блюд в заказе.
  - status: OrderStatus — статус заказа (в процессе, доставлен и т.д.).
  - paymentMethod: PaymentMethod — метод оплаты.
  - estimatedDeliveryTime: DateTime — расчетное время доставки.
- **Методы:**
  - calculateEstimatedDeliveryTime(): DateTime — вычисляет расчетное время доставки.
- **Тип связи:** Агрегация с Dish, ассоциации с Customer, Restaurant, PaymentMethod.

5.5 Customer

- **Атрибуты:**
  - id: Int — идентификатор клиента.
  - name: String — имя клиента.

- phone: String — номер телефона.
- addresses: List<Address> — список адресов доставки.
- paymentMethods: List<PaymentMethod> — список методов оплаты.

· **Методы:**

- createOrder(restaurant: Restaurant, dishes: List<Dish>, address: Address, paymentMethod: PaymentMethod): Order — создает новый заказ.

· **Тип связи:** Ассоциации с Order и PaymentMethod, агрегирует Address.

5.6 Address

· **Атрибуты:**

- street: String — улица.
- houseNumber: String — номер дома.
- city: String — город.
- location: Location — географические координаты.

· **Тип связи:** Агрегация с Location.

5.7 Location

· **Атрибуты:**

- latitude: Double — широта.
- longitude: Double — долгота.

· **Тип связи:** Агрегация в Address, ассоциация с Courier.

5.8 Courier

· **Атрибуты:**

- id: Int — идентификатор курьера.
- name: String — имя курьера.
- location: Location — текущее местоположение.
- status: CourierStatus — статус (свободен, занят и т.д.).

- currentOrder: Order — текущий заказ (если есть).

- **Методы:**

  - acceptOrder(order: Order): void — принимает заказ на доставку.
  - updateLocation(newLocation: Location): void — обновляет текущее местоположение.
  - completeDelivery(): void — завершает доставку текущего заказа.

- **Тип связи:** Ассоциация с Order.

5.9 SpecialOffer

- **Атрибуты:**

  - id: Int — идентификатор специального предложения.
  - description: String — описание предложения.
  - validUntil: DateTime — срок действия.
  - discountPercentage: Double — процент скидки.
  - restaurant: Restaurant — ресторан, предлагающий акцию.

- **Методы:**

  - applyOffer(order: Order): Order — применяет акцию к заказу и возвращает обновленный заказ.
  - isValid(): Boolean — проверяет, действительно ли предложение.

- **Тип связи:** Ассоциация с Restaurant.

5.10 PaymentMethod

- **Атрибуты:**

  - id: Int — идентификатор метода оплаты.
  - type: PaymentType — тип (карта, наличные и т.д.).
  - details: PaymentDetails — детали оплаты (например, номер карты).
  - customer: Customer — владелец метода оплаты.

- **Методы:**

- validate(): Boolean — проверяет, действителен ли метод.
- processPayment(amount: Double): PaymentResult — обрабатывает платеж и возвращает результат.

· **Тип связи:** Ассоциация с Customer и Order.

5.11 DeliverySystem

· **Атрибуты:**

- restaurants: List<Restaurant> — список ресторанов.
- couriers: List<Courier> — список курьеров.
- customers: List<Customer> — список клиентов.
- orders: List<Order> — список заказов.

· **Методы:**

- addRestaurant(restaurant: Restaurant): void — добавляет новый ресторан.
- addCourier(courier: Courier): void — добавляет нового курьера.
- addCustomer(customer: Customer): void — добавляет нового клиента.
- findRestaurant(name: String): Restaurant — находит ресторан по имени.
- findCourier(id: Int): Courier — находит курьера по идентификатору.
- findCustomer(id: Int): Customer — находит клиента по идентификатору.
- createOrder(customer: Customer, restaurant: Restaurant, dishes: List<Dish>, address: Address, paymentMethod: PaymentMethod): Order — создает новый заказ.

· **Тип связи:** Ассоциации с Restaurant, Courier, Customer, Order.

5.12 Feedback

· **Атрибуты:**

- id: Int — идентификатор отзыва.
- order: Order — заказ, к которому относится отзыв.
- customer: Customer — автор отзыва.

- rating: Int — оценка.
- comment: String — комментарий.
- date: DateTime — дата отзыва.

- **Тип связи:** Ассоциация с Order и Customer.

6 Детальная диаграмма классов

6.1 Общее описание

Диаграмма представляет собой структурированную модель системы доставки еды, включающую взаимодействие между ресторанами, клиентами, курьерами и заказами. На диаграмме отображено **12 классов**, каждый из которых выполняет специфическую роль в рамках системы.

6.2 Список классов и их атрибутов/методов

На диаграмме представлены следующие классы:

1. Restaurant:

- Атрибуты:
  - id: Int — идентификатор ресторана.
  - name: String — название ресторана.
  - menu: Menu — меню ресторана.
  - workingHours: String — часы работы.
  - specialOffers: List<SpecialOffer> — список специальных предложений (если применимо).
- Методы:
  - getMenu(): Menu — возвращает меню ресторана.
  - setMenu(menu: Menu): void — устанавливает меню ресторана.
  - getSpecialOffers(): List<SpecialOffer> — возвращает список специальных предложений.
  - addSpecialOffer(offer: SpecialOffer): void — добавляет специальное предложение.
  - removeSpecialOffer(offer: SpecialOffer): void — удаляет специальное предложение.
- Тип связи: Агрегация с Menu и SpecialOffer.

2. Dish:

- Атрибуты:
  - id: Int — идентификатор блюда.
  - name: String — название блюда.
  - description: String — описание блюда.
  - price: Double — цена блюда.

    - ingredients: List<String> — список ингредиентов блюда.
- Методы:
    - getIngredients(): List<String> — возвращает список ингредиентов блюда.
- Тип связи: Агрегируется в Menu и Order.

3. Menu:

- Атрибуты:
    - items: List<Dish> — список блюд в меню.
- Методы:
    - addItem(item: Dish): void — добавляет новое блюдо в меню.
    - removeItem(item: Dish): void — удаляет блюдо из меню.
    - getItems(): List<Dish> — возвращает список всех блюд.
- Тип связи: Агрегация с Restaurant.

4. Order:

- Атрибуты:
    - id: Int — идентификатор заказа.
    - customer: Customer — клиент, сделавший заказ.
    - restaurant: Restaurant — ресторан, из которого заказаны блюда.
    - dishes: List<Dish> — список блюд в заказе.
    - status: OrderStatus — статус заказа (в процессе, доставлен и т.д.).
    - paymentMethod: PaymentMethod — метод оплаты.
    - estimatedDeliveryTime: DateTime — расчетное время доставки.
- Методы:
    - calculateEstimatedDeliveryTime(): DateTime — вычисляет расчетное время доставки.
- Тип связи: Агрегация с Dish, ассоциации с Customer, Restaurant, PaymentMethod.

5. Customer:

- Атрибуты:
    - id: Int — идентификатор клиента.
    - name: String — имя клиента.
    - phone: String — номер телефона.

    - addresses: List<Address> — список адресов доставки.
    - paymentMethods: List<PaymentMethod> — список методов оплаты.
- Методы:
    - createOrder(restaurant: Restaurant, dishes: List<Dish>, address: Address, paymentMethod: PaymentMethod): Order — создает новый заказ.
- Тип связи: Ассоциации с Order и PaymentMethod, агрегирует Address.

6. Address:

- Атрибуты:
    - street: String — улица.
    - houseNumber: String — номер дома.
    - city: String — город.
    - location: Location — географические координаты.
- Тип связи: Агрегация с Location.

7. Location:

- Атрибуты:
    - latitude: Double — широта.
    - longitude: Double — долгота.
- Тип связи: Агрегация в Address, ассоциация с Courier.

8. Courier:

- Атрибуты:
    - id: Int — идентификатор курьера.
    - name: String — имя курьера.
    - location: Location — текущее местоположение.
    - status: CourierStatus — статус (свободен, занят и т.д.).
    - currentOrder: Order — текущий заказ (если есть).
- Методы:
    - acceptOrder(order: Order): void — принимает заказ на доставку.
    - updateLocation(newLocation: Location): void — обновляет текущее местоположение.

    - completeDelivery(): void — завершает доставку текущего заказа.
- Тип связи: Ассоциация с Order.

9. SpecialOffer:

- Атрибуты:
    - id: Int — идентификатор специального предложения.
    - description: String — описание предложения.
    - validUntil: DateTime — срок действия.
    - discountPercentage: Double — процент скидки.
    - restaurant: Restaurant — ресторан, предлагающий акцию.
- Методы:
    - applyOffer(order: Order): Order — применяет акцию к заказу и возвращает обновленный заказ.
    - isValid(): Boolean — проверяет, действительно ли предложение.
- Тип связи: Ассоциация с Restaurant.

10. PaymentMethod:

- Атрибуты:
    - id: Int — идентификатор метода оплаты.
    - type: PaymentType — тип (карта, наличные и т.д.).
    - details: PaymentDetails — детали оплаты (например, номер карты).
    - customer: Customer — владелец метода оплаты.
- Методы:
    - validate(): Boolean — проверяет, действителен ли метод.
    - processPayment(amount: Double): PaymentResult — обрабатывает платеж и возвращает результат.
- Тип связи: Ассоциация с Customer и Order.

11. DeliverySystem:

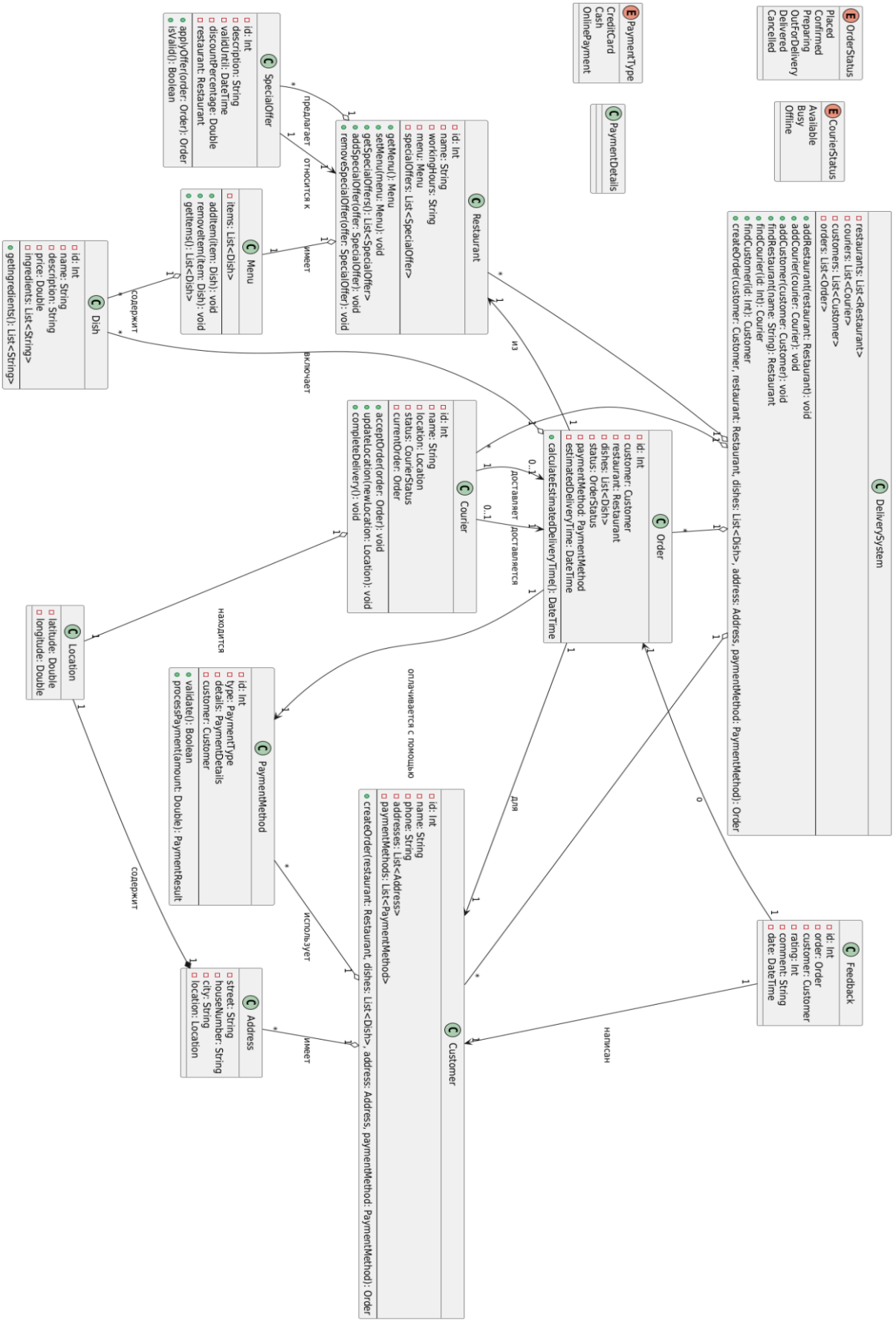
- Атрибуты:
    - restaurants: List<Restaurant> — список ресторанов.
    - couriers: List<Courier> — список курьеров.
    - customers: List<Customer> — список клиентов.

    - orders: List<Order> — список заказов.
- Методы:
    - addRestaurant(restaurant: Restaurant): void — добавляет новый ресторан.
    - addCourier(courier: Courier): void — добавляет нового курьера.
    - addCustomer(customer: Customer): void — добавляет нового клиента.
    - findRestaurant(name: String): Restaurant — находит ресторан по имени.
    - findCourier(id: Int): Courier — находит курьера по идентификатору.
    - findCustomer(id: Int): Customer — находит клиента по идентификатору.
    - createOrder(customer: Customer, restaurant: Restaurant, dishes: List<Dish>, address: Address, paymentMethod: PaymentMethod): Order — создает новый заказ.
- Тип связи: Ассоциации с Restaurant, Courier, Customer, Order.

12. Feedback:

- Атрибуты:
    - id: Int — идентификатор отзыва.
    - order: Order — заказ, к которому относится отзыв.
    - customer: Customer — автор отзыва.
    - rating: Int — оценка.
    - comment: String — комментарий.
    - date: DateTime — дата отзыва.
- Тип связи: Ассоциация с Order и Customer.

6.3 Диаграмма классов



1. Выбор архитектуры

Система построена на основе **клиент-серверной архитектуры**. Это решение обеспечивает четкое разделение логики и упрощает масштабирование системы. Вся основная бизнес-логика и управление данными сосредоточены на сервере, в то время как клиентская часть служит для взаимодействия пользователей с системой.

Основные компоненты:

- **Клиентская часть**: реализует взаимодействие с конечными пользователями, включая создание заказов, отслеживание статуса, и управление меню ресторанов.
- **Серверная часть**: управляет обработкой заказов, координацией курьеров и интеграцией с платежными системами.
- **База данных**: централизованное хранилище данных, содержащее информацию о ресторанах, клиентах, заказах и платежах.

2. Компоненты и элементы архитектуры

2.1. Тип клиента

В системе используются два типа клиентских приложений:

- **Тонкий клиент**: Минимальная логика на клиентской стороне. Подходит для пользователей, которые просто хотят просматривать меню, оформлять заказы и отслеживать их выполнение.
- **Толстый клиент**: Используется администраторами ресторанов и операторами системы. Имеет расширенные функции для управления меню, анализа заказов и создания специальных предложений.

2.2. Серверная часть

Серверная часть системы выполняет ключевую роль:

- **Обработка заказов**: Принимает заказы от клиентов, связывает их с ресторанами и назначает курьеров.
- **Работа с платежами**: Интеграция с платежными системами для обеспечения безопасности транзакций.
- **Управление данными**: Хранение всех данных в единой базе данных для обеспечения целостности и связности.

Сервер поддерживает взаимодействие с клиентами с помощью **REST API**, что позволяет легко интегрировать различные типы клиентов (веб, мобильные приложения) и добавлять новые функции.

3. Использование баз данных

Система использует **централизованную реляционную базу данных** для хранения и управления данными:

- **Таблицы клиентов, заказов, ресторанов и платежей** обеспечивают целостность данных и легко связываются между собой.
- База данных реализует основные реляционные связи, например:
    - Customer ↔ Order (1 ко многим).
    - Restaurant ↔ SpecialOffer (1 ко многим).
    - Order ↔ Dish (многие ко многим).

Реляционная структура позволяет легко выполнять сложные запросы и агрегировать данные, что упрощает создание отчетов и аналитических панелей.

4. Коммуникация и взаимодействие компонентов

Основная логика взаимодействия между компонентами построена на использовании **ассоциаций**, **агрегаций** и **композиций**:

1. **Restaurant ↔ Menu — агрегация**:
    - Ресторан содержит меню, но меню может существовать независимо.
2. **Menu ↔ Dish — агрегация**:
    - Меню состоит из множества блюд, каждое из которых может быть частью нескольких меню.
3. **Order ↔ Dish — композиция**:
    - Заказ включает блюда, и при удалении заказа все связанные блюда теряются.

Коммуникация между клиентами и сервером осуществляется через **HTTP-запросы** (например, POST для создания заказа, GET для получения списка меню). Это обеспечивает гибкость и расширяемость системы.

5. Обоснование выбора серверного решения

Выбор **выделенного сервера** обусловлен следующими факторами:

- **Безопасность**: Выделенный сервер позволяет лучше контролировать доступ к данным и защищать их.
- **Производительность**: Сервер обрабатывает все запросы и операции, что снижает нагрузку на клиентскую часть.
- **Гибкость**: Легкость масштабирования при увеличении числа клиентов или добавлении новых функций.

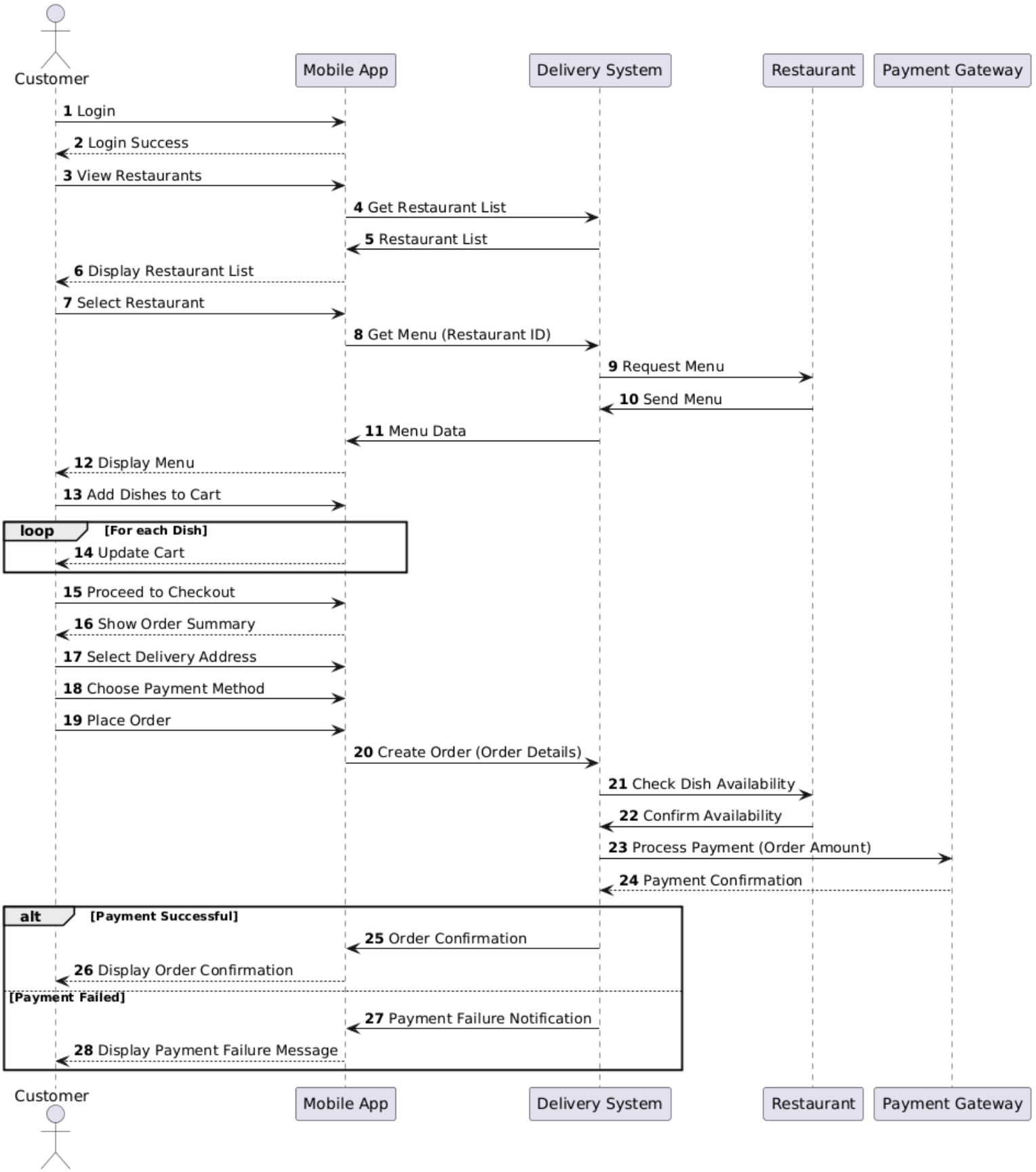
Серверная часть поддерживает реляционные операции с базой данных и предоставляет API для взаимодействия с клиентами.

6. Типы клиентов и их использование

- **Клиент для ресторанов**:
    - Позволяет управлять меню и отслеживать заказы.
    - Включает расширенные функции для работы с отчетностью и аналитикой.
- **Клиент для курьеров**:
    - Отслеживает заказы и предоставляет маршрут доставки.
    - Позволяет обновлять статус доставки в реальном времени.
- **Клиент для конечных пользователей**:
    - Основной интерфейс для создания и отслеживания заказов.
    - Включает функции выбора ресторанов, применения специальных предложений и оценки заказов.

7 Динамика системы (последовательность, взаимодействия, состояния

7.1 Диаграмма состояний



На представленной диаграмме изображен последовательный процесс выполнения заказа в мобильном приложении для доставки еды. Диаграмма описывает взаимодействие пользователя (Customer) с приложением, системой доставки (Delivery System), рестораном (Restaurant) и платежной системой (Payment Gateway). Это диаграмма последовательностей, которая отображает процесс заказа еды шаг за шагом.

Основные участники (акторы):
1. Customer (Пользователь) — человек, который делает заказ через мобильное приложение.
2. Mobile App (Мобильное приложение) — интерфейс, через который пользователь взаимодействует с системой.
3. Delivery System (Система доставки) — система, отвечающая за получение и передачу данных между приложением и рестораном.
4. Restaurant (Ресторан) — принимает заказы и отправляет данные меню.
5. Payment Gateway (Платежная система) — отвечает за обработку платежей.

Основные шаги:

1. Login (1–2): Пользователь авторизуется в приложении, и происходит успешный вход (Login Success).

2. Просмотр списка ресторанов (3–6): Пользователь видит список ресторанов. Приложение запрашивает список ресторанов у системы доставки (Get Restaurant List), а система возвращает его (Restaurant List).

3. Выбор ресторана и отображение меню (7–12): Пользователь выбирает ресторан. Приложение запрашивает меню ресторана (Get Menu), и данные меню возвращаются пользователю (Menu Data).

4. Добавление блюд в корзину (13–14): Пользователь добавляет блюда в корзину. Процесс обновления корзины повторяется для каждого выбранного блюда.

5. Оформление заказа (15–19): Пользователь переходит к оформлению заказа (Proceed to Checkout), выбирает адрес доставки и метод оплаты.

6. Создание заказа (20–24): Приложение создает заказ и передает данные о заказе в систему доставки (Create Order). Система доставки проверяет доступность блюд в ресторане (Check Dish Availability) и подтверждает их доступность (Confirm Availability). Затем платежная система обрабатывает платеж (Process Payment) и отправляет подтверждение оплаты (Payment Confirmation).

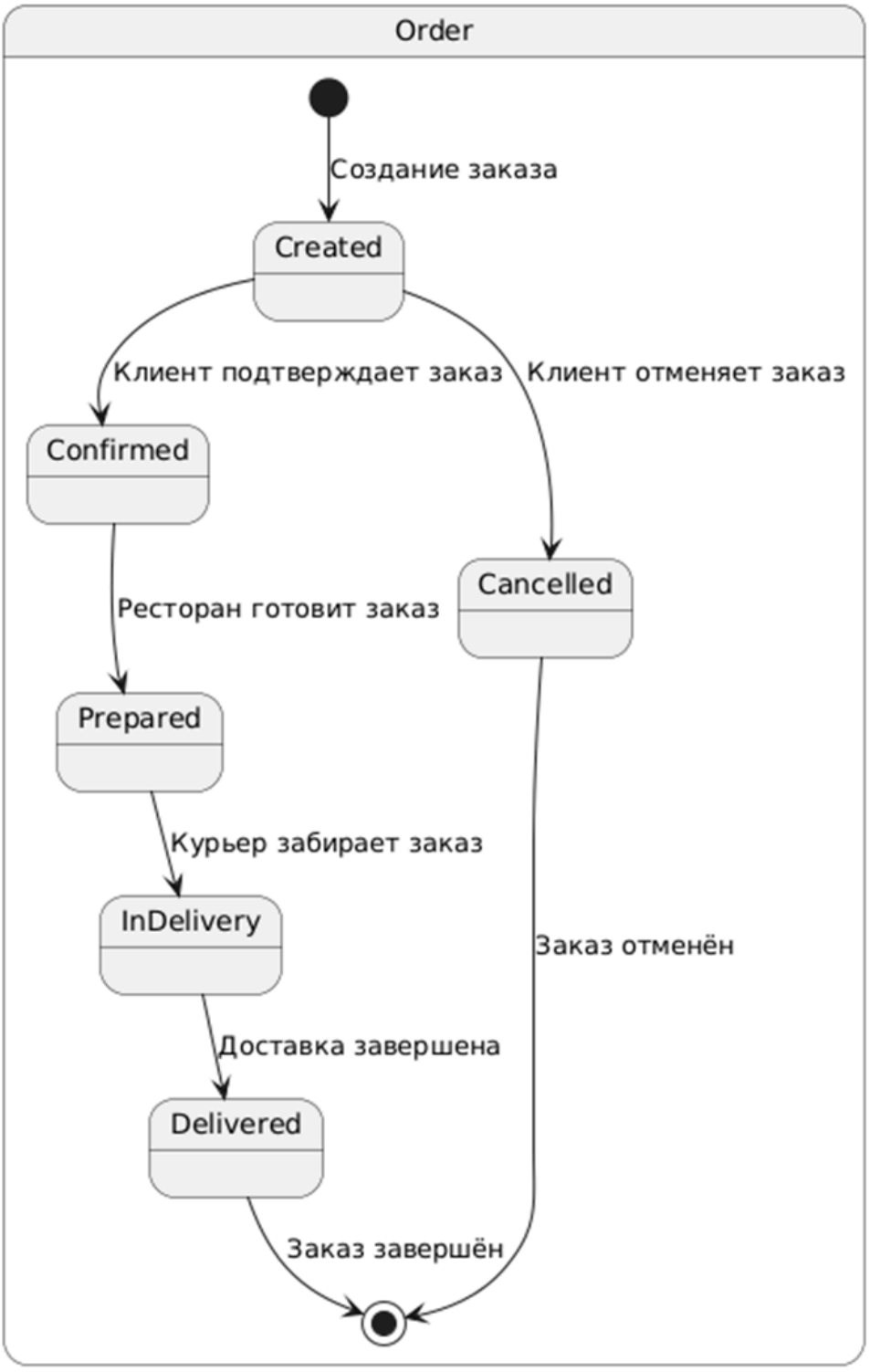
7. Подтверждение заказа (25–28):
- Если оплата прошла успешно, заказ подтверждается (Order Confirmation), и пользователю отображается подтверждение заказа (Display Order Confirmation).
- Если оплата не удалась, пользователь получает уведомление о сбое (Payment Failure Notification) и сообщение о неудачной оплате (Display Payment Failure Message).

Альтернативные сценарии:
- Успешная оплата (alt): Сценарий успешного завершения заказа.
- Неудачная оплата (Payment Failed): Альтернативный путь, если оплата не прошла.

Общая логика:
Диаграмма иллюстрирует полный цикл заказа — от авторизации до завершения процесса оплаты, показывая взаимодействие различных систем.

7.2 Диаграммы состояний для классов

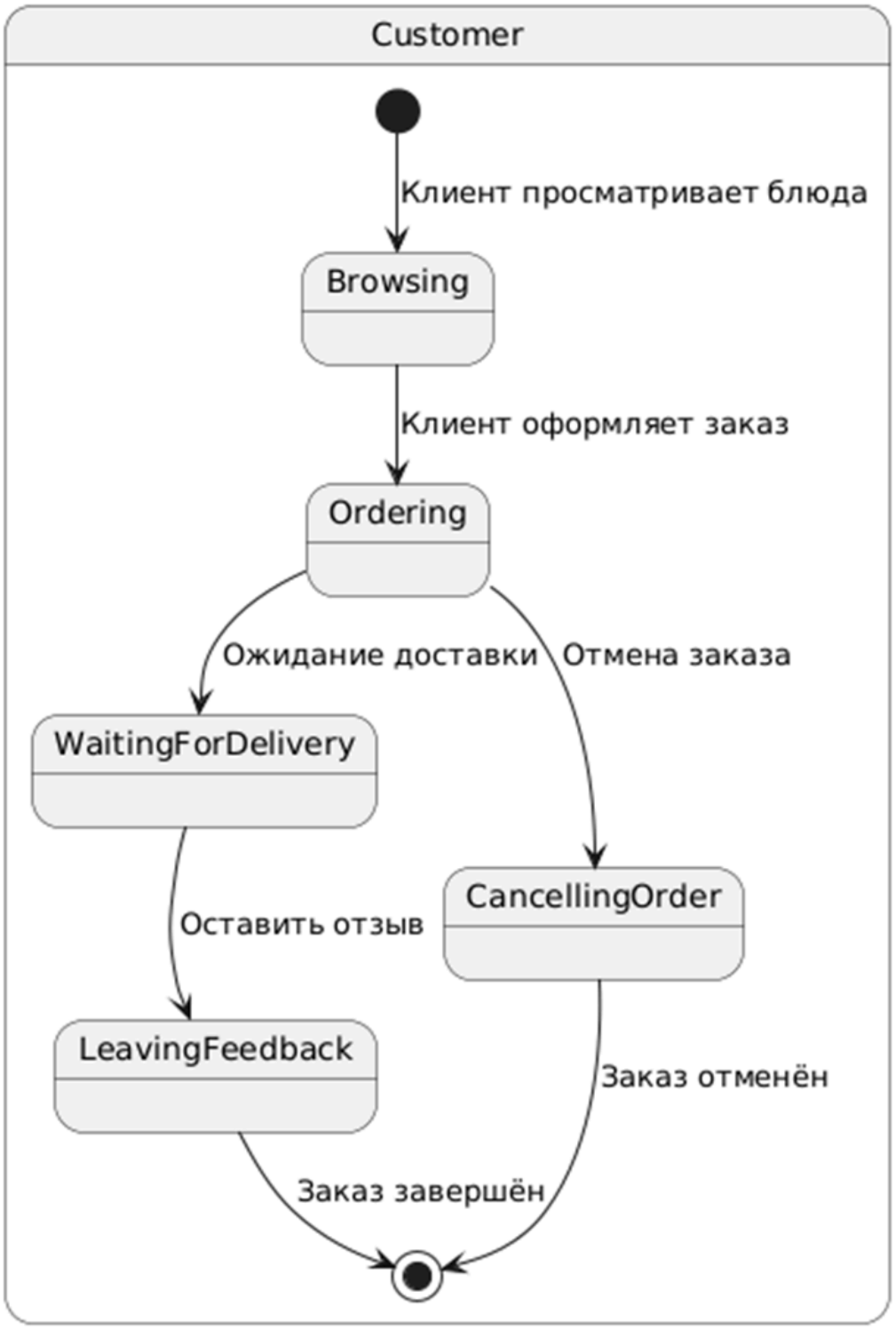
7.2.1 Диаграмма состояний для класса Order



Описание состояний для Order:

- **Created**: Заказ создан.
- **Confirmed**: Заказ подтвержден клиентом.
- **Prepared**: Заказ готовится в ресторане.
- **InDelivery**: Заказ доставляется курьером.
- **Delivered**: Заказ доставлен клиенту.
- **Cancelled**: Заказ отменен.

7.2.2 Диаграмма состояний для класса Customer

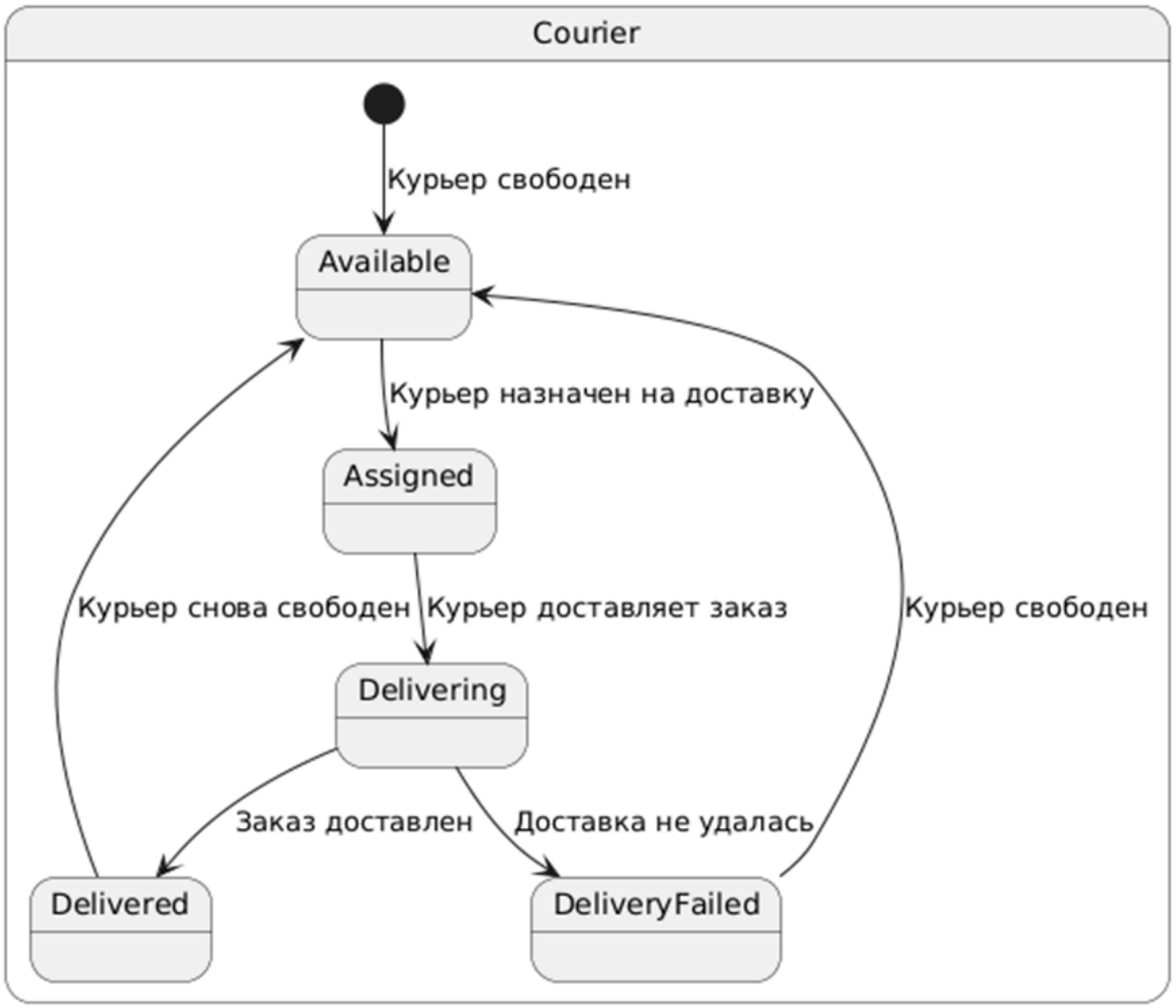


Описание состояний для Customer:

- **Browsing**: Клиент просматривает меню ресторана.
- **Ordering**: Клиент создает заказ.
- **WaitingForDelivery**: Клиент ожидает доставки.
- **LeavingFeedback**: Клиент оставляет отзыв о доставке.

- **CancellingOrder**: Клиент отменяет заказ.

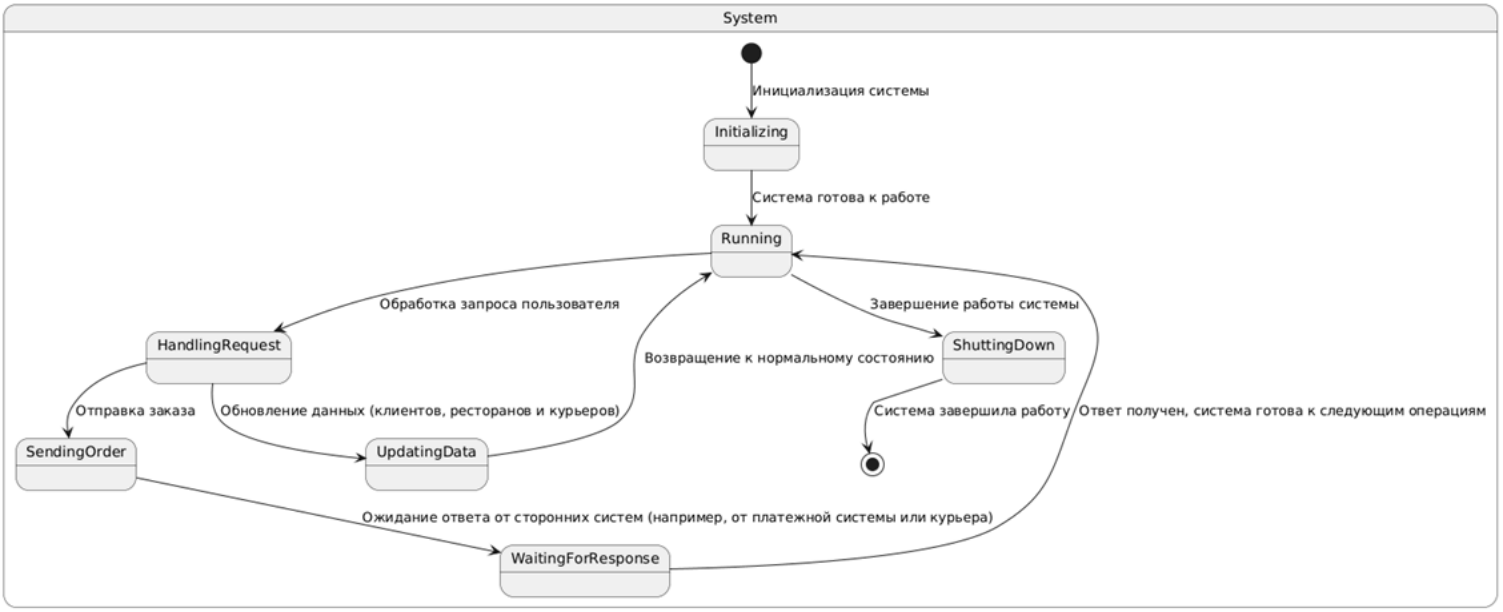
7.2.3 Диаграмма состояний для класса Courier



Описание состояний для Courier:

- **Available**: Курьер доступен для доставки.
- **Assigned**: Курьер назначен для выполнения заказа.
- **Delivering**: Курьер доставляет заказ.
- **Delivered**: Заказ доставлен.
- **DeliveryFailed**: Доставка не удалась.

7.2.4 Диаграмма состояний для класса System



Описание состояний для System:

- **Initializing**: Система запускается, инициализируются все компоненты и данные.
- **Running**: Система готова к работе, пользователи могут отправлять запросы.
- **HandlingRequest**: Система обрабатывает запрос от пользователя (например, создание заказа или поиск ресторана).
- **UpdatingData**: Система обновляет данные, например, информацию о клиентах, ресторанах или курьерах.
- **SendingOrder**: Система отправляет заказ в ресторан или курьеру.
- **WaitingForResponse**: Система ожидает подтверждение от внешних систем (например, о завершении платежа или доставке).
- **ShuttingDown**: Система завершает свою работу, все процессы останавливаются.

8 Расширенная матрица трассировки требований:

8.1 Бизнес требования

ID	Бизнес требование	Как протестировать, что бизнес требование удовлетворено	Юзкейс	Комментарий
1	Приложение должно предоставлять возможность пользователям выбирать и заказывать блюда из ресторанов с последующей доставкой	Провести тестирование, при котором пользователь может выбрать блюда из каталога ресторанов, добавить их в корзину и оформить заказ	Пользователь выбирает ресторан, блюда, добавляет их в корзину и оформляет заказ	Нужно убедиться, что функционал выбора ресторана и оформления заказа работает корректно для всех ресторанов
2	Фильтрация по кухне, цене, времени доставки и рейтингу	Провести тесты с фильтрацией по различным параметрам и убедиться, что фильтры работают корректно	Пользователь применяет фильтры для поиска ресторанов	Тестирование должно включать все возможные комбинации фильтров для разных ресторанова
3	Формирование заказа, добавление/удаление блюд из корзины, оплата	Протестировать добавление и удаление блюд в корзину, оформление заказа и успешную оплату через разные методы	Пользователь оформляет заказ и производит оплату	Убедиться, что оплата корректно обрабатывается всеми поддерживаемыми способами оплаты
4	Отслеживание доставки в реальном времени	Проверить, что статус заказа обновляется корректно на каждом этапе:	Пользователь отслеживает заказ в реальном времени	Нужно протестировать интеграцию с системами доставки и

		принят, готовится, передан курьеру, доставлен		картографическими сервисами
5	Возможность оставлять отзывы и оценки ресторанам и курьерам	Проверить возможность оставления отзыва после завершения доставки	Пользователь завершает заказ и оставляет отзыв о ресторане или курьере	Оценка должна появляться сразу после оставления отзыва
6	Доступ к истории заказов	Протестировать доступ пользователя к полной истории всех заказов	Пользователь просматривает историю заказов	История должна корректно отображать все заказы, включая время, суммы и статусы заказов
7	Онлайн-поддержка через чат или телефон	Проверить доступность и работоспособность функции онлайн-поддержки через чат и телефон	Пользователь обращается в поддержку через чат или телефон	Проверка ответов поддержки на все возможные вопросы пользователя
8	Система скидок и акций	Протестировать активацию скидок и акций при оформлении заказа	Пользователь активирует скидку или акцию при оформлении заказа	Нужно убедиться, что скидки корректно применяются при оформлении заказов
9	Приложение должно поддерживать несколько языков	Провести тестирование языковых версий интерфейса приложения	Пользователь выбирает нужный язык в настройках	Приложение должно корректно переводиться и поддерживать отображение всех элементов на разных языках
10	Интеграция с	Проверить, что	Пользователь	Интеграция с

	курьерскими службами для назначения курьеров и оптимизации маршрутов	курьеры назначаются автоматически при заказе и что маршруты корректно строятся	оформляет заказ, который автоматически назначается курьеру	системой курьеров должна быть протестирована для разных сценариев доставки
11	Уведомления о статусе заказа	Проверить отправку уведомлений на всех этапах заказа (принят, готовится, в пути, доставлен)	Пользователь получает уведомления о каждом этапе обработки заказа	Нужно убедиться, что уведомления приходят вовремя и содержат корректную информацию
12	Соответствие стандартам безопасности при работе с данными пользователей и платежной информацией	Провести пентесты, проверить безопасность при работе с платежной информацией и личными данными	Защищенная авторизация, защита платежных данных, проверка работы шифрования	Приложение должно соответствовать стандартам PCI DSS и обеспечивать безопасность данных пользователей

8.2 Функциональные требования

ID	Функциональное требование	Как протестировать, что функциональное требование	Юзкейс	Комментарий

		удовлетворено		
1	Регистрация и авторизация через email и социальные сети	Провести регистрацию через email и социальные сети, затем протестировать авторизацию	Пользователь регистрируется и входит в систему через email или социальную сеть	Нужно протестировать корректную обработку данных при регистрации, включая двухфакторную аутентификацию
2	Доступ к каталогу ресторанов с фильтрацией по типу кухни, рейтингу, цене и времени доставки	Применить все виды фильтрации и убедиться, что результаты поиска корректны	Пользователь фильтрует рестораны по типу кухни, рейтингу, цене и времени доставки	Необходимо учесть корректность работы фильтров при больших объемах данных, возможные комбинации фильтров
3	Добавление/удаление блюд в корзину и оформление заказа	Проверить возможность добавления и удаления блюд, затем оформить заказ	Пользователь формирует заказ, добавляя блюда в корзину, и успешно его завершает	Важно убедиться, что изменения в корзине корректно отображаются, а заказы корректно отправляются на обработку
4	Выбор способа оплаты (банковская карта, электронные кошельки)	Протестировать все способы оплаты и убедиться, что транзакции обрабатываются правильно	Пользователь выбирает способ оплаты и завершает транзакцию	Все методы оплаты должны быть проверены на корректность обработки данных
5	Отслеживание	Проверить, что	Пользователь	Интеграция с

	статуса заказа в реальном времени	пользователь может отслеживать заказ в приложении с точностью на всех этапах	отслеживает заказ через приложение	системой доставки и правильное отображение этапов заказа
6	Уведомления на всех этапах заказа	Проверить, что уведомления отправляются при каждом изменении статуса заказа	Пользователь получает уведомления о каждом этапе обработки заказа	Уведомления должны приходить своевременно и отображать правильную информацию
7	Возможность оставлять отзывы и оценки ресторанам и курьерам	Протестировать возможность оставить отзыв после завершения заказа	Пользователь оставляет отзыв после завершения заказа	Отзывы и рейтинги должны отображаться мгновенно и влиять на средний рейтинг ресторана/курьера
8	Доступ к истории заказов	Убедиться, что пользователь может просмотреть все свои прошлые заказы	Пользователь просматривает историю заказов	История должна корректно сохраняться, а данные быть доступными на любом этапе использования приложения
9	Интеграция с картографическими сервисами для отслеживания маршрута курьера	Проверить, что пользователь может видеть текущую позицию курьера на карте	Пользователь отслеживает курьера в режиме реального времени через приложение	Интеграция с картографическим сервисом должна отображать текущее местоположение курьера без задержек

10	Поддержка скидок и акций	Протестировать возможность активировать скидки при оформлении заказа	Пользователь активирует промокод или скидку при оформлении заказа	Скидки должны корректно применяться и отображаться в итоговой сумме заказа

8.3 Нефункциональные требования

ID	Нефункциональное требование	Как протестировать, что нефункциональное требование удовлетворено	Юзкейс	Комментарий
1	Поддержка 1000 одновременных пользователей без потери производительности	Провести нагрузочное тестирование с 1000 одновременными пользователями	Одновременно 1000 пользователей совершают заказы	Система должна сохранять стабильную работу при высоких нагрузках, необходимо проверить время отклика и стабильность
2	Соответствие стандартам безопасности при работе с платежной информацией	Провести тесты на соответствие PCI DSS и GDPR, проверить шифрование данных	Проверка работы с платежными данными	Все платежные данные должны передаваться и обрабатываться безопасно, используя шифрование
3	Масштабируемость для поддержки новых функций и увеличения числа пользователей	Провести тесты масштабирования, симулировать увеличение числа пользователей	Увеличение количества пользователей до 10 000	Система должна поддерживать динамическое масштабирован ие,

				автоматическое добавление ресурсов
4	Поддержка веб-версии и мобильных платформ (iOS и Android)	Провести тестирование на разных платформах (веб, мобильные устройства iOS и Android)	Пользователи используют приложение на разных устройствах и платформах	Нужно убедиться, что интерфейс и функционал приложения корректно отображаются и работают на всех поддерживаемых устройствах

8.4 Выводы:

- **Полнота требований**: Основные функциональные и нефункциональные требования охватывают ключевые аспекты работы системы. Однако следует более детально проработать нефункциональные требования в части масштабируемости и обработки ошибок.ф
- **Достаточность требований**: Требования достаточно детализированы, но стоит уделить внимание оптимизации производительности при больших нагрузках, особенно на стороне сервера и базы данных.
- **Ограничения**: В документе обозначены важные технические ограничения, однако нужно уделить больше внимания вопросам интеграции с внешними системами и требованиям безопасности, таким как защита данных пользователей и их обработка согласно нормам GDPR.

9 План реализации тестирования и сопровождения

9.1 План реализации

Этап 1 (1-4 недели):

Подготовительный этап и базовая функциональность:

- Сбор и обсуждение базовых требований с заказчиком.
- Определение целевой аудитории и первичных пользовательских сценариев.
- Реализация базовых функций:
    - **Регистрация и авторизация через email.**
    - **Каталог ресторанов с базовой фильтрацией по кухне.**
    - **Оформление простого заказа без интеграции с внешними системами.**
- База данных:
    - Создание базовой структуры для хранения пользователей, ресторанов и заказов.
- Тестирование:
    - Модульное тестирование регистрации, авторизации, работы с каталогом и простого оформления заказа.
    - Тестирование безопасности авторизации.

Этап 2 (5-8 недели):

Расширение функциональности:

- Добавление дополнительных методов оплаты (банковские карты, электронные кошельки).
- Добавление продвинутой фильтрации ресторанов (по цене, времени доставки).
- Реализация интеграции с курьерской системой:
    - Отслеживание доставки в реальном времени с уведомлениями на каждом этапе.
- Улучшение UX/UI:
    - Доработка интерфейса для удобного выбора ресторанов и оплаты.
- Тестирование:

    - Интеграционное тестирование взаимодействия между модулями: регистрация, оформление заказа, оплата.
    - Тестирование работы уведомлений и отслеживания доставки.

Этап 3 (9-12 недели):

Добавление дополнительного функционала:

- Интеграция с картографическими сервисами (Google Maps или Yandex Maps) для отслеживания курьеров.
- Добавление двухфакторной аутентификации для пользователей.
- Расширение истории заказов для пользователей с отображением предыдущих заказов и отзывов.
- Добавление возможности оставлять отзывы и оценки ресторанам и курьерам.
- Масштабируемость:
    - Подготовка системы к увеличению количества пользователей.
- Тестирование:
    - Модульное и интеграционное тестирование новых функций.
    - Нагрузочное тестирование с моделированием до 500 пользователей.

Этап 4 (13-16 недели):

Оптимизация и завершение основного функционала:

- Внедрение системы скидок и акций для пользователей.
- Оптимизация производительности системы при высокой нагрузке (до 1000 пользователей).
- Полное завершение системы уведомлений.
- Полное тестирование безопасности данных пользователей (соответствие стандартам PCI DSS).
- Тестирование:
    - Интеграционное тестирование взаимодействия всех модулей (заказы, доставка, оплата, отзывы).
    - Нагрузочное тестирование до 1000 одновременных пользователей.

Этап 5 (17-20 недели):

Подготовка к запуску и маркетинг:

- Окончательное системное тестирование на реальных сценариях использования.
- Исправление всех багов и уязвимостей.
- Маркетинговая кампания:
    - Подготовка промо-материалов (видео, статьи, рекламные баннеры).
    - Привлечение пользователей через партнёрство с ресторанами и социальные сети.
- Тестирование:
    - Окончательное тестирование всех системных компонентов на совместимость и производительность.

9.2 План тестирования для приложения по доставке еды

1. Введение

- **Цель тестирования**: Проверка всех функциональных и нефункциональных возможностей системы доставки еды, чтобы убедиться в их соответствии требованиям и стабильной работе приложения.
- **Область тестирования**:
    - Регистрация и авторизация пользователей.
    - Выбор ресторанов и блюд, формирование заказа.
    - Оформление покупки и оплата.
    - Отслеживание доставки.
    - Работа системы лояльности и скидок.
    - Поддержка пользователей.

2. Типы тестирования

1. **Функциональное тестирование**:
    - Проверка регистрации и авторизации новых пользователей.
    - Тестирование процесса выбора блюд, добавления в корзину и оформления заказа.
    - Проверка корректности обработки оплаты и получения подтверждения заказа.

- Тестирование отслеживания заказа в реальном времени.
- Проверка системы скидок и акций.

2. **Интеграционное тестирование**:
   - Тестирование взаимодействия между модулями приложения (каталог ресторанов, система оплаты, модуль отслеживания доставки).
   - Проверка интеграции с картографическими сервисами (Google Maps, Yandex Maps) для отслеживания курьеров.
   - Интеграция с внешними платёжными системами (Stripe, PayPal).
3. **Тестирование производительности**:
   - Оценка производительности при высокой нагрузке (большое количество одновременно оформляющих заказы пользователей).
   - Проверка времени отклика системы при использовании более 1000 пользователей одновременно.
4. **Тестирование безопасности**:
   - Проверка защиты личных данных пользователей.
   - Тестирование безопасности платежных данных и выполнения транзакций (соответствие стандартам PCI DSS).
   - Тестирование двухфакторной аутентификации.

3. Сценарии тестирования

1. **Сценарий: Регистрация нового пользователя**:
   - Проверка процесса регистрации нового пользователя, включая отправку подтверждения регистрации на email.
2. **Сценарий: Оформление заказа**:
   - Тестирование выбора ресторана и блюд, добавления их в корзину, выбора метода оплаты и завершения заказа.
3. **Сценарий: Оплата**:
   - Проверка корректности оформления заказа и успешного выполнения платежа через различные методы (карты, электронные кошельки).
4. **Сценарий: Отслеживание доставки**:
   - Тестирование работы системы отслеживания заказа с уведомлениями на каждом этапе (принят, готовится, передан курьеру, в пути, доставлен).

5. **Сценарий: Система лояльности и скидки**:
    - Тестирование корректности работы скидок и акций для постоянных клиентов, активации промокодов.
6. **Сценарий: Поддержка пользователей**:
    - Проверка работы службы поддержки, включая онлайн-чат и возможность отправки запросов через приложение.

4. Критерии приемки

- Все функциональные возможности приложения работают корректно и без задержек.
- Приложение способно поддерживать работу с минимальной задержкой для 1000 пользователей одновременно.
- Все платежные операции обрабатываются корректно и безопасно.
- Уведомления о статусе заказа доставляются пользователям в реальном времени.

5. Риски

- **Проблемы с безопасностью**: Необходимость тщательного тестирования защиты данных пользователей и платёжной информации, чтобы избежать утечек.
- **Проблемы с производительностью**: Высокая нагрузка может привести к замедлению работы системы.
- **Проблемы с интеграцией**: Возможны ошибки при взаимодействии между компонентами системы (особенно интеграции с внешними сервисами).

6. План выполнения тестирования

- **Подготовка тестового окружения**: Создание тестовой базы данных, настройка платёжных сервисов и тестовых аккаунтов для работы с реальными сценариями.
- **Функциональное тестирование**: Проверка основных функций системы: регистрация, оформление заказа, оплата, отслеживание доставки.
- **Интеграционное тестирование**: Тестирование взаимодействия модулей приложения, интеграция с внешними системами.

- **Тестирование производительности**: Проверка работы приложения при пиковых нагрузках (до 1000 пользователей одновременно).
- **Тестирование безопасности**: Проверка шифрования данных, защиты личных и платёжных данных.
- **Пользовательское тестирование**: Привлечение реальных пользователей для оценки удобства интерфейса и отзывчивости приложения.

7. Отчетность

- **Еженедельные отчёты**: Составление отчётов о результатах тестирования, выявленных ошибках и ходе их исправления.
- **Финальный отчет**: Подготовка заключительного отчета с рекомендациями по улучшению системы и исправлению багов.

8. Заключение

Следуя данному плану тестирования, приложение для доставки еды станет устойчивым к нагрузкам, интуитивно понятным для пользователей и защищённым с точки зрения данных и платежей.

9.3 План эксплуатации

Основные задачи эксплуатации:

1. **Мониторинг системы:**
    - **Реальное время мониторинга:** Для отслеживания состояния приложения и обеспечения его бесперебойной работы используется специализированное программное обеспечение мониторинга (например, Zabbix, Grafana, Prometheus). Это позволяет получать данные в реальном времени о работе всех ключевых компонентов системы, таких как серверы, базы данных и сетевые подключения.
    - **Метрики, отслеживаемые системой мониторинга:**
        - **Время отклика системы:** Важный показатель, отражающий, как быстро приложение реагирует на пользовательские запросы (например, при оформлении заказа или переходе между страницами).

        - **Количество активных пользователей:** Мониторинг числа пользователей, одновременно находящихся в системе, чтобы избежать перегрузок и своевременно увеличивать ресурсы.
        - **Загрузка процессора и памяти серверов:** Это позволяет понимать, когда серверы находятся под высокой нагрузкой, и оперативно реагировать для предотвращения сбоев.
        - **Количество успешных и неуспешных заказов:** Анализ статистики успешных транзакций помогает выявлять системные ошибки или проблемы с процессами оплаты.
        - **Производительность базы данных:** Следит за скоростью запросов к базе данных, предотвращая задержки и ошибки при обработке данных заказов и пользователей.
        - **Ошибки и сбои:** Система мониторинга отслеживает ошибки приложений и инфраструктуры. В случае критических проблем отправляются автоматические уведомления технической команде для быстрого устранения.
    - **Реакция на инциденты:**
        - В случае выявления проблемы, система оповещает через e-mail, SMS или Push-уведомления администраторам, что позволяет быстро реагировать на неполадки и минимизировать время простоя.
2. **Обновления и масштабирование:**
    - **Инкрементные обновления:**
        - В процессе эксплуатации приложения будут регулярно внедряться **инкрементные обновления**. Это обновления, которые вводят новые функции поэтапно, не нарушая основной функционал и минимизируя перерывы в работе.
        - **Планирование обновлений:** Каждое обновление проходит этапы планирования, тестирования на специально созданной тестовой среде и последующей интеграции в основную систему.

        - **Тестирование на ограниченной группе пользователей:** Перед внедрением для всех пользователей обновления сначала применяются к ограниченной группе, чтобы выявить потенциальные баги и проблемы. Это снижает риски внезапных ошибок после выпуска обновлений на массовую аудиторию.
    - **Масштабирование системы:**
        - **Автоматическое масштабирование серверов:** Система использует инструменты для автоскейлинга (например, AWS Auto Scaling, Google Cloud), что позволяет динамически увеличивать мощность серверов при росте нагрузки (например, во время пиковых часов, когда заказы поступают чаще всего).
        - **Прогнозирование нагрузки:** На основе данных о числе активных пользователей и исторических данных по нагрузке система автоматически увеличивает количество серверов или их мощность.
3. **Техническая поддержка:**
    - **Круглосуточная поддержка:** Система технической поддержки работает 24/7, обеспечивая постоянную связь с пользователями через онлайн-чат, телефон и e-mail. Для пользователей разработана удобная система тикетов, с помощью которой можно отслеживать статус своей заявки.
    - **Интеграция с CRM:** Для улучшения качества поддержки и оптимизации процессов, система поддержки интегрируется с CRM-системой, которая позволяет оперативно отслеживать историю взаимодействий с пользователем, его заказы и обращения. Это помогает быстрее решать повторяющиеся проблемы и предоставлять персонализированную помощь.
    - **Оперативное устранение проблем:** В случае возникновения проблем с оформлением заказов, оплатой или доставкой, техподдержка в тесном взаимодействии с разработчиками оперативно устраняет сбои, минимизируя воздействие на пользователей.
4. **Резервное копирование данных:**
    - **Регулярные резервные копии:** Для защиты данных приложения реализуется система ежедневного резервного

копирования всех критических данных, включая заказы, платежные транзакции и пользовательскую информацию. Копии хранятся как в локальных, так и в облачных хранилищах для обеспечения надежности.

- **План восстановления системы:** В случае сбоя или потери данных реализованы автоматизированные процедуры восстановления, которые минимизируют время простоя системы. Тестирование планов восстановления проводится регулярно, чтобы обеспечить надежность в случае экстренных ситуаций.

9.4 Документ по сопровождению программного продукта

Основные аспекты сопровождения:

1. **Поддержка и обновления:**
   - **Регулярные обновления функциональности:**
     - В рамках сопровождения программного продукта разработаны процедуры регулярных обновлений, которые включают как добавление новых функций, так и оптимизацию уже существующих. Например, могут быть добавлены новые способы оплаты, улучшенные алгоритмы фильтрации меню ресторанов или новые языковые локализации для расширения географии пользователей.
     - **Минимизация простоев:** Каждый патч и обновление тщательно тестируется, чтобы минимизировать возможные ошибки и простои. Для критических обновлений предусмотрен процесс "горячего" внедрения, который не требует остановки работы сервера.
   - **Быстрое устранение ошибок:**
     - В рамках сопровождения продукта предусмотрена политика оперативного устранения ошибок (bug fixes). Если пользователь или система мониторинга обнаружат баг, на него немедленно создается тикет, и разработчики

приступают к его исправлению. Цель — максимально быстрое внедрение исправлений без ущерба для пользовательского опыта.

2. **Контроль безопасности:**
    - **Постоянное обновление систем безопасности:**
        - Система безопасности приложения будет регулярно обновляться для соответствия современным стандартам, таким как GDPR (общий регламент по защите данных) и PCI DSS (стандарты безопасности для обработки платежей).
        - Важно учитывать, что каждое обновление системы безопасности предваряется анализом текущих рисков и уязвимостей, после чего внедряются необходимые корректировки.
    - **Регулярные пентесты (penetration tests):**
        - Для выявления возможных уязвимостей проводится регулярное тестирование безопасности (пентесты). Эти тесты имитируют атаки на систему, чтобы проверить ее устойчивость к взломам, утечкам данных или несанкционированному доступу.
    - **Мониторинг попыток несанкционированного доступа:**
        - Система безопасности включает механизмы отслеживания подозрительной активности. Например, многократные неудачные попытки входа или аномальные действия пользователя могут блокироваться системой и вызывать автоматическое оповещение службы безопасности.
    - **Двухфакторная аутентификация:**
        - Для повышения уровня безопасности при авторизации пользователей используется двухфакторная аутентификация (2FA). Это минимизирует риски взлома аккаунтов пользователей.
3. **Расширение функционала:**
    - **Добавление новых возможностей:**
        - Одной из основных задач сопровождения продукта является регулярное расширение его функционала. Это может включать:

        - Введение программ лояльности, где пользователи могут накапливать бонусы за заказы.
        - Поддержка нескольких языков и региональных валют, что позволит выходить на новые рынки.
        - Персонализированные предложения пользователям на основе их истории заказов.
    - **Интеграция с новыми службами доставки:**
        - Для увеличения охвата и улучшения логистики поддерживается возможность интеграции с новыми курьерскими службами или партнерами по доставке. Это поможет сократить время доставки и расширить сеть ресторанов.
4. **Отчеты и аналитика:**
    - **Автоматизация отчетов:**
        - Система автоматически генерирует отчеты для анализа эффективности приложения. Эти отчеты включают информацию о количестве активных пользователей, количестве успешных и неуспешных заказов, среднем времени доставки, числе отзывов и их оценке.
        - Также доступны отчеты для ресторанов, где они могут отслеживать свою производительность, получать данные по продажам и отзывам.
    - **Использование аналитики:**
        - Полученные данные и отчеты используются для оптимизации процессов внутри приложения. Например, на основе анализа времени доставки курьеры могут получать оптимизированные маршруты для улучшения логистики, а маркетинг может настраивать персонализированные предложения для пользователей на основе их предпочтений и истории заказов.

10 Вывод

В рамках реализации проекта приложения для доставки еды наша команда успешно выполнила все запланированные задачи, придерживаясь инкрементной модели разработки. Продукт был создан поэтапно, начиная с базовых функций, таких как регистрация и каталог ресторанов, и заканчивая более сложными элементами, включая интеграцию с курьерскими и платежными системами, а также картографическими сервисами.

Каждый этап разработки сопровождался тщательным тестированием, что позволило оперативно выявлять и устранять возможные ошибки. Мы внедрили проверенные механизмы для масштабируемости, производительности и безопасности, что гарантирует надежную работу приложения при высокой нагрузке.

Особое внимание уделялось безопасности данных пользователей, соответствию стандартам PCI DSS, а также реализации двухфакторной аутентификации для дополнительной защиты. Финальная версия приложения прошла тестирование на реальных сценариях использования, показала отличные результаты и готова к запуску.

Результат проделанной работы – интуитивно понятное, безопасное и гибкое решение, готовое к выходу на рынок и дальнейшему развитию.