2024

Описание архитектурного решения для магазина электронных товаров

Дипломная работа в рамках курса «Основы архитектуры программного обеспечения»

Арсений Остапенко

Оглавление

[1. Введение и цели 3](#_Toc173071357)

[1.1. Обзор требований 3](#_Toc173071358)

[1.2. Цели по качеству решения 4](#_Toc173071359)

[1.3. Ключевые участники (стейкхолдеры) 4](#_Toc173071360)

[2. Архитектурные ограничения 5](#_Toc173071361)

[3. Границы и окружение системы 5](#_Toc173071362)

[4. Стратегия решения 6](#_Toc173071363)

[5. Компонентный состав 9](#_Toc173071364)

[5.1. Верхнеуровневая компонентная схема системы 9](#_Toc173071365)

[5.1.1. Уровень 1: диаграмма контекста системы 9](#_Toc173071366)

[5.1.2. Уровень 2: диаграмма контейнеров системы 10](#_Toc173071367)

[5.1.3. Уровень 3: диаграмма компонентов системы 11](#_Toc173071368)

[5.2. Основные интерфейсы RestAPI 12](#_Toc173071369)

[5.3. Детализация компонентов и агрегатов 14](#_Toc173071370)

[6. Динамическое представление 15](#_Toc173071371)

[7. Схема развёртывания 20](#_Toc173071372)

[8. Архитектурные и дизайн-решения (ADR) 21](#_Toc173071373)

[9. Требования к атрибутам качества 21](#_Toc173071374)

[9.1. Дерево атрибутов качества 21](#_Toc173071375)

[9.2. Сценарии проверки атрибутов качества 23](#_Toc173071376)

[10. Риски и технический долг 23](#_Toc173071377)

[11. Словарь 23](#_Toc173071378)

[12. Приложения 24](#_Toc173071379)

[12.1. Приложение 1: Structurizr код для генерации C4 модели архитектуры приложения 24](#_Toc173071380)

[12.2. Приложение 2: Swagger код для генерации интерактивной генерации в рамках спецификации OpenAPI 28](#_Toc173071381)

[12.3. Приложение 3: Dbdiagram код для генерации ER диаграммы базы данных 37](#_Toc173071382)

[12.4. Приложение 4: Plantuml код в для генерации диаграмм последовательностей для пользовательских сценарие 39](#_Toc173071383)

[12.4.1. Код диаграммы последовательности входа пользователя в систему 39](#_Toc173071384)

[12.4.2. Код диаграммы последовательности поиска товара в магазине 40](#_Toc173071385)

[12.4.3. Код диаграммы последовательности оформления заказа пользователем 41](#_Toc173071386)

[12.4.4. Код диаграммы последовательности проведения оплаты пользователем 43](#_Toc173071387)

[12.4.5. Код диаграммы последовательности оформления доставки заказа 45](#_Toc173071388)

# Введение и цели

|  |  |
| --- | --- |
| **ЦЕЛЬ** | **ОПИСАНИЕ** |
| **Увеличение продаж и количества зарегистрированных пользователей** | Не менее 10% всего экономически активного населения РФ (8.3 млн человек) зарегистрированы на сайте и осуществляют покупки хотя бы 2 раза в год (8.3 млн человек \* 2 покупки в год / 365 дней = 45479 покупок в день). |
| **Получение конкурентных преимуществ** | Интернет магазин предоставляет возможность быстро купить наиболее востребованные населением страны товары в среднем ценовом диапазоне.    У пользователя есть возможность быстро и эффективно найти желаемый товар на сайте, оплатить его и выбрать удобный способ доставки. |
| **Исполнение требований безопасности** | Обеспечить безопасность персональных данных, предоставляемых пользователем при регистрации.    Обеспечить безопасность данных платежных средств пользователей.    Обеспечить безопасность данных о заказанных товаров пользователем. |

## Обзор требований

Для того чтобы эффективно описать варианты пользовательских сценариев использовался поход из создания user story map. В ходе данной работы получилась картина представленная на Рисунке 1, где описываются сценарии, которые нужно достичь в ближайшей реализации. На Рисунке 2 добавлены сценарии «на будущее», которые не будут описаны в данной работе, но имеют бизнес ценность для проекта.

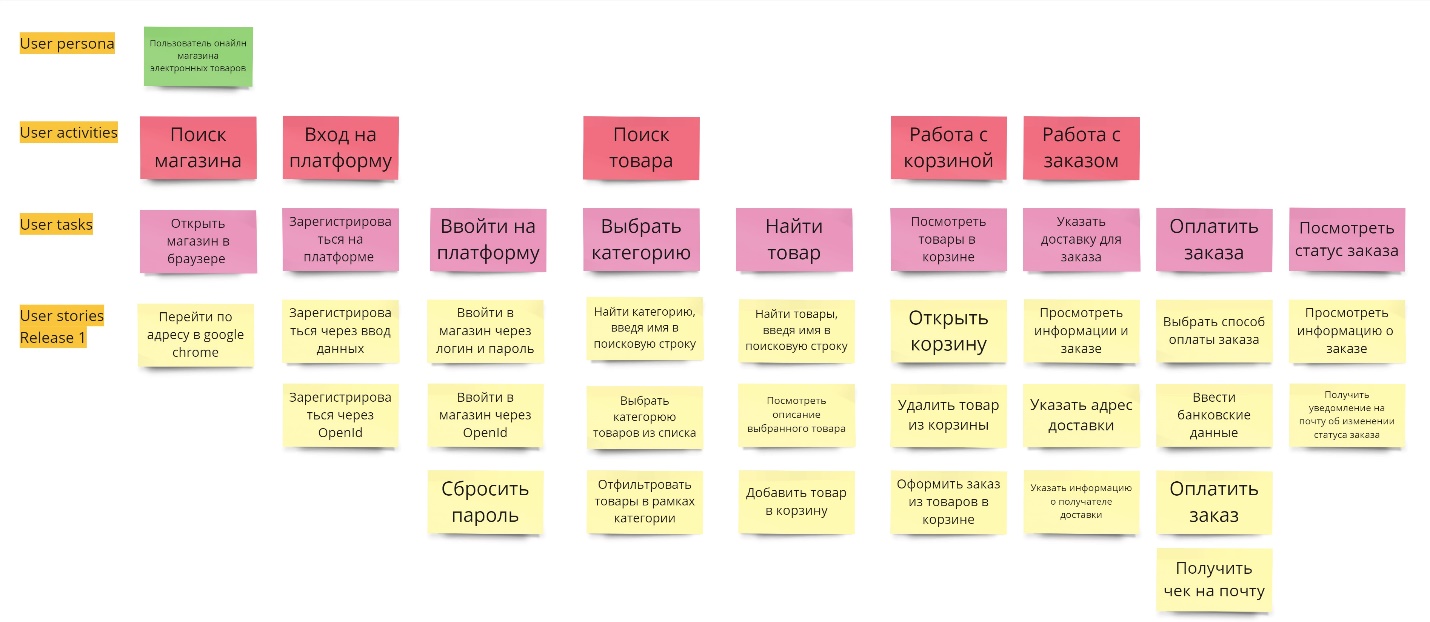


Рисунок User Story Map для онлайн магазина электронных товаров

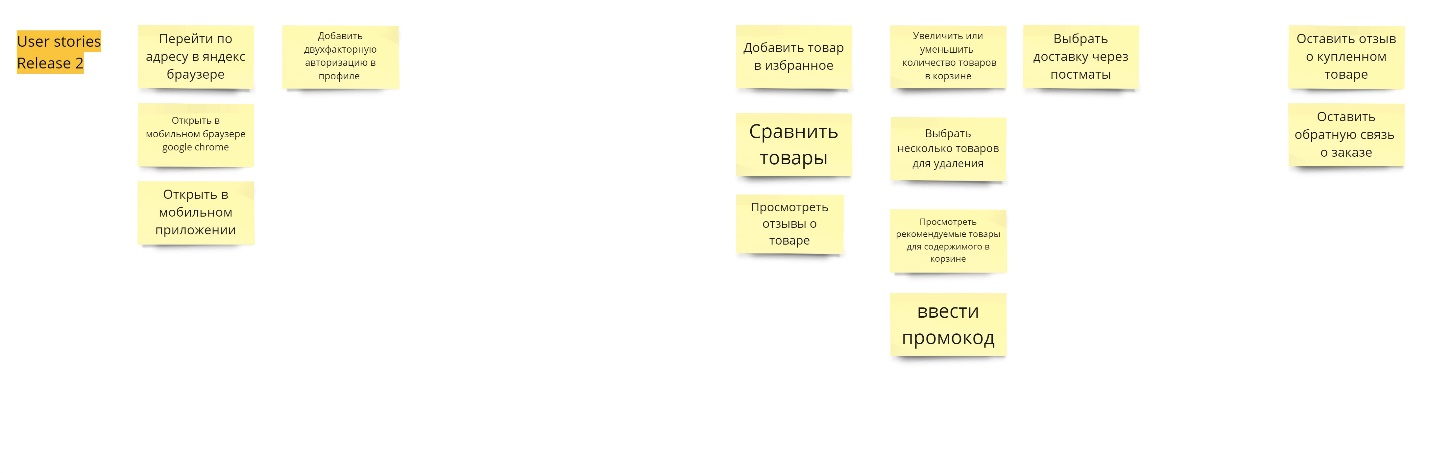


Рисунок User Story Map для сценариев в будущих релизах

Для этапа MVP

|  |  |
| --- | --- |
| **№ СЦЕНАРИЯ** | **КРАТКОЕ ОПИСАНИЕ** |
| **Регистрация на платформе** | Пользователь регистрируется на платформе. Для аутентификации используется стандарт OpenID. |
| **Вход в систему** | Пользователь может войти в систему, используя аккаунты OpenID. |
| **Поиск и фильтрация товаров** | У пользователя есть возможность фильтровать товары как по его характеристикам, таким как тип прибора, модель, производитель, цена, наличие на складе и т.д. |
| **Добавление понравившегося товара в корзину** | Пользователь может добавить товар в корзину и продолжить поиск следующего товара. В конце поиска все выбранные товары будут ожидать в корзине их оплаты. |
| **Доставка товаров** | Пользователь может выбрать способ доставки товара, включающий самовывоз из магазины или доставку курьером |
| **Оплата товаров** | Пользователь может оплатить товары из корзины различными способами, включая онлайн-оплату банковской картой и оплату при получении. |
| **Отслеживание статуса заказа** | Пользователь может отследить статус заказа на всех этапах, от оформления до получения. |
| **Администрирование товаров** | Администратор магазина может редактировать информацию о товаре (его характеристики, стоимость и доступность для заказа). |
| **Интеграция с внешними системами (CRM, WMS)** | Платформа успешно обменивается данными со внутренними системами. Такими как CRM – для повышения продаж, и WMS для мониторинга наличия товара. |
| **Уведомление пользователей** | Пользователь получает уведомления на почту при оплате заказа и смене статуса доставки. |

Пользовательские сценарии для будущих версий продукта

|  |  |
| --- | --- |
| **№ СЦЕНАРИЯ** | **КРАТКОЕ ОПИСАНИЕ** |
| **Добавление двухфакторной аутентификации** | Пользователь может включить 2фа для своей учетки с целью повышения безопасности аккаунта. |
| **Коллекции товаров** | Пользователь может добавлять товары в группы «избранное», «понравившееся», «посмотреть позже», «сравнить»  Пользователь может зайти в коллекции, где увидит ранее отобранные товары |
| **Сравнение товаров** | Пользователь может открыть коллекцию «сравнить» , где будет таблица характеристик из ранее добавленных товаров |
| **Просмотр и создание отзывов на товара** | Пользователь может посмотреть отзыв на любой товар и оставить отзыв на купленный товар |
| **Использование промокодов** | Пользовать может ввести промокод для получения скидок на товар |
| **Участие в программе лояльности** | Пользователь может участвовать в программе лояльности |
| **Расширение типов доставки** | Пользователь может заказать доставку через постаматы. Пользователь может заказать доставку из большего списка операторов |
| **Расширение способов оплаты** | Пользователь может оплатить товар из большего списка платежных систем |
| **Оформление возврата товара через сайт** | Пользовать может оставить заявку на возврат товара на сайте |

## Цели по качеству решения

|  |  |
| --- | --- |
| **НАЗВАНИЕ ЦЕЛИ** | **КРАТКОЕ ОПИСАНИЕ** |
| **Масштабируемость** | Увеличение количества пользователей будет требовать поэтапного увеличения производственных мощностей. Для достижения этого нужно изначально разработать систему способную легко расширяться горизонтально. |
| **Доступность/Актуальность** | На платформе находится не менее 20 тыс. артикулов различных товаров. Информация о товаре может обновляться в любой момент времени (не чаще 1 раза в час) и должна отражаться на сайте с задержкой не более 120 сек. |
| **Безопасность** | Платформа работает с персональными данными пользователя   * 1. ФИО (могут указать настоящие при авторизации)   2. Электронная почта (могут указать для отслеживания статуса)   3. Номера телефонов (для авторизации по коду??)   4. Данные банковских карт (при оплате)   5. Адрес (при доставке курьером)   6. Товары в корзине, история покупок   Доступ к этим данным должен иметь только пользователь – владелец данных и администратор. |
| **Удовлетворенность пользователей** | Система обеспечивает высокую производительность, скорость выполнения операций находится на приемлемом для пользователя уровне.    Система обеспечивает высокую доступность для пользователей. |

## Ключевые участники (стейкхолдеры)

|  |  |
| --- | --- |
| **РОЛЬ/ИМЯ** | **ОЖИДАНИЯ** |
| **Технический директор** | - Высший технический орган, отвечающий за рассмотрение и утверждение архитектуры. - Ожидает разработки развернутой и хорошо структурированной архитектуры, соответствующей долгосрочным техническим целям и возможностям масштабирования. - Требует подробной документации для оценки технической реализуемости системы и потенциальных рисков. - Обеспечивает руководство группой разработчиков на основе стратегического направления архитектуры и обеспечивает согласование с техническим видением организации. |
| **Владелец продукта** | - Основной представитель клиента для команды и ключевым связующим звеном между бизнес-стратегией и технологической стратегией;  - Отвечает за видение конечного продукта и его ценности для пользователя. |
| **Заинтересованные стороны бизнеса** | - Основные выгодоприобретатели, ожидают получение прибыли и обеспечения минимальных затрат при разработке и поддержке конечного продукта/системы. |
| **Команда разработки** | - Отвечают за реализацию ИТ-системы в соответствии с системными и бизнес требованиями;  - Привлекаются в последующем обслуживании системы по результатам анализа ее работы; |
| **Специалист по безопасности** | - Обеспечивает соблюдение действующих стандартов кибербезопасности;  - Проверяет систему на соответствие требованиям законодательства РФ по управлению персональными данными. |
| **Юристы** | - Обеспечивают правовое сопровождение по вопросам функционирования системы. |
| **Техническая поддержка** | - Консультируют пользователей по почте и по телефону  - Решают типовые проблемы пользователей  - Описывают дефекты системы встреченные пользователями и отправляют описания в команду разработки |

# Архитектурные ограничения

|  |  |  |
| --- | --- | --- |
| **КАТЕГОРИЯ** | **ОГРАНИЧЕНИЕ** | **ОПИСАНИЕ** |
| **Техническое** | Технологический стек | Рекомендуется при создании нового сервиса использовать в качестве языка программирования Java и Spring Boot фреймворк, либо Go |
| **Юридические** | Соблюдение законодательства | * + Необходимость соблюдения закона о персональных данных исключает возможность аренды недорогих мощностей за рубежом   + ЗОПП требует реализовать сценарии возврата товара |
| **Организационные** | Организации  команд | Используем гибкие методологии разработки для быстрого выкладывания задач в продакшен с целью помочь бизнесу адаптироваться к изменениям на рынке.  Разделить разработчиков на команды FRONTEND и BACKEND. Часть BACKEND разработчиков должна выполнять функции третьей линии поддержки. На начальном этапе необходим как минимум один человек на должность Database Engineer. Нанять DEVOPS инженеров для поддержки стедов |
| **Безопасности** | Доступность данных | Необходимо разработать ролевую модель для пользователей, с целью защиты функционала администратора. А также проверять принадлежность данных к их владельцам при CRUD операциях. |
| **Экономические** | Финансовые ограничения | На этапе MVP необходимо сделать минимально жизнеспособный продукт при минимуме затрат. Если результаты первой версии приложения будут удовлетворительными, то финансовые ограничения могут быть ослаблены для дальнейшего развития проекта как в плане бизнеса так и в техническом плане. |

# Границы и окружение системы

|  |  |  |  |
| --- | --- | --- | --- |
| **ПАРТНЕР** | **ОПИСАНИЕ ИНТЕРФЕЙСА** | **ВХОД** | **ВЫХОД** |
| **Аутенфикация**    **OpenID Connect Provider** | Вход пользователей в систему посредством аккаунтов OpenID | Данные для формы входа. | Результат аутентификации JWT – токен. |
| **Оплата**  **Payment API** | Передача данных для совершения платежа из компоненты «Оплата». Подключение к внешней онлайн-кассе.  Rest API, протокол HTTP, API. | Данные для платежа: сумма, валюта, счет получателя,  наименование платежа. | Платеж совершен, данные транзакции |
| **Доставка** | Внешний сервис. | Адрес – куда доставляем, получатель, аналитики – получает ли другой человек, бесконтактная доставка. Параметры груза (ВГХ). | Доставка груза |
| **Уведомления**  **Email Service** | Почтовый сервер компании  Протокол: TLS, SMTP. | * + Уведомление   + Почта того, кто отправляет   + - Почта того, кто получает | Платеж совершен, данные транзакции |

# Стратегия решения

|  |  |  |  |
| --- | --- | --- | --- |
| **КАТЕГОРИЯ** | **ОПИСАНИЕ ПРИНЯТОГО РЕШЕНИЯ** | **РАССМОТРЕННЫЕ АЛЬТЕРНАТИВЫ** | **ОБОСНОВАНИЕ РЕШЕНИЯ** |
| **Технологии** | Язык бэкенда: Java | Kotlin, C#, PHP, Go | Решающие факторы:   1. На Java написано большинство бэкендов, как следствие данный язык имеет множество специализированных библиотек для создания веб приложения (<https://habr.com/ru/articles/730954/>) 2. Большой объем специалистов на рынке. 3. Перспективы разработки мобильного приложения на Kotlin. Kotlin и Java - родственные языки, что делает эффективной связку Java backend-a с Kotlin приложением.     Важные факторы, но проигнорированные из-за недостатка информации   1. На какой платформе будет развернут сервер 2. На каком языке написано MVP приложение 3. Предпочтение заказчика 4. Знакомство команды с выбранным языком программирования |
|  | База данных: PostgreSQL | MySQL, MongoDB, ORACLE, MSSQL | Основные критерии выбора:   1. Доступна в России (отсеяли ORACLE, MSSQL, частично MySQL) 2. Объектно-реляционного типа (отсеили MongoDB) 3. Работает на операционной системе linux 4. Выдерживает хранение и обработку большого объема данных (PostgreSQL>MySQL) |
|  | REST API + RabbitMQ | SOAP API | - JSON более легковесен чем XML, что уменьшит нагрузку на сеть и положительно скажется на производительности  - REST де факто современный стандарт проектирования АПИ. Так как приложение новое и нет требований в поддержке легаси АПИ, то выбор REST является более оптимальным для совместимости с современными технологиями и дальнейшего развития тех стека. |
|  | Kubernetes | Docker Compose | Решение принято в сторону открытого программного обеспечения Kubernetes , доступного в России |
| **Декомпозиция** | Использование сторонних сервисов:   * Платежный сервис * Почтовый сервис * OpenID | Включение данных сервисов в состав проекта и их разработка | - Возможность использования готовых решений  - Экономия времени и ресурсов команды  - Независимость внешних и достаточно неизменных систем от модулей решения |
|  | Пакетирование данных по фичам | Пакетирование по слоям, по ограниченным контекстам и агрегатам | - Более масштабируемая структура по сравнению с разбиением по техническим слоям  - Низкая связанность кода  - Более простая в реализации, чем пакетирование по ограниченным контекстам и агрегатам |
| **Архитектурный стиль** | MSA | Монолит, SOA | - Монолит не подходит исходя из бизнес цели по высокой масштабируемости системы  - Независимость процессов проектирования, разработки, тестирования, развертывания и поставки разных сервисов;  - лучшая скорость выхода бизнес-функции на рынок;  - независимая масштабируемость сервисов,  улучшение надежности (отказоустойчивости). |
| **Цели качества** | Масштабируемость |  | Увеличение количества пользователей будет требовать поэтапного увеличения производственных мощностей. Для достижения этого нужно изначально разработать систему способную легко расширяться горизонтально |
|  | Доступность/Актуальность |  | Доступность информации о товарах на платформе важна для обеспечения актуальности данных и удовлетворения ожиданий пользователей. С более чем 20 тысячами артикулов, регулярное обновление информации гарантирует точные данные о наличии, ценах и характеристиках товаров, что повышает доверие и удовлетворенность клиентов. Быстрая обработка изменений с задержкой не более 120 секунд минимизирует риск ошибок, таких как продажа товара, которого нет в наличии, и улучшает общий пользовательский опыт, способствуя повышению продаж и лояльности клиентов. |
|  | Безопасность |  | Безопасность персональных данных важна для защиты конфиденциальности пользователей, предотвращения мошенничества и кражи личных данных, а также для соблюдения законодательных требований. Утечка таких данных может привести к серьезным финансовым и репутационным потерям для пользователей и компаний, а также к юридическим последствиям для организации, не обеспечившей должный уровень защиты. Сохранение доверия клиентов и предотвращение возможных злоупотреблений также являются ключевыми причинами, по которым безопасность данных критически важна. |

# Компонентный состав

## Верхнеуровневая компонентная схема системы

Применяем в качестве целевого решения микросервисную архитектуру. Каждый сервис имеет собственную базу данных. Каждый сервис в системе общается между собой через сетевые интерфейсы. Внутренние и внешние запросы проходят через API GATEWAY который выполняет функции маршрутизации. Для асинхронных взаимодействий используется брокер сообщений.

При графического описания системы использовалась C4 нотация, сделанная при помощи подхода «diagrams as code» в Structurizr DSL. Код представлен в главе «Приложения» (см. Приложение 1: Structurizr код для генерации C4 модели архитектуры приложения).

Для понимания элементов диаграммы представлена легенда (см. Рисунок 3)

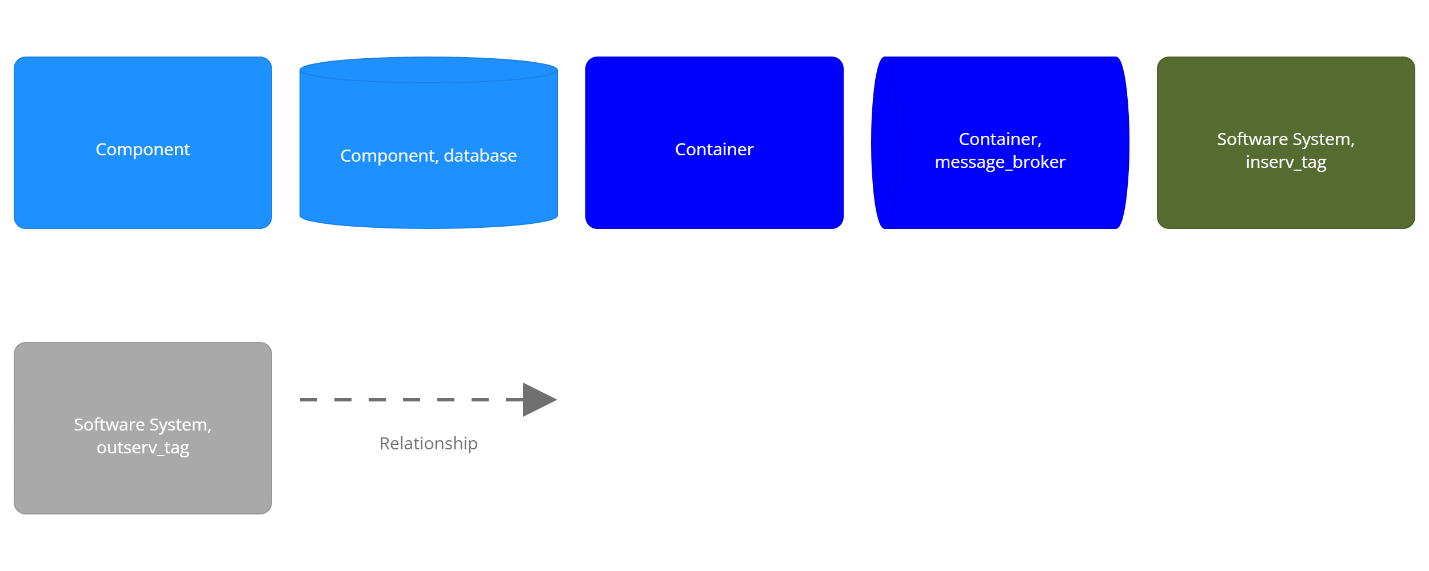


Рисунок : Легенда использоватнных элементов С4 диаграммы

## Уровень 1: диаграмма контекста системы

Диаграмма, описывающая систему и её связи верхнеуровнево представлена на Рисунок 4. Она показывает, как система взаимодействует с внешними сущностями (пользователями, внешними системами). Исходя из данной диаграммы можно увидеть, что

* Основными пользователями системы являются клиенты и администраторы сайта
* Магазин взаимодействует с внешними сервисами, которые можно разделить на две группы
  + Внешние системы. Данные системы нами не контролируются и используются для создания новых бизнес возможностей для пользователей: облегченный вход и регистрация, быстрая оплата, доставка товаров, отправка уведомлений.
  + Внутренние системы. Эти системы являются частью нашей системы, но не относятся напрямую к онлайн магазину. Таких системы всего две: CRM, WMS.

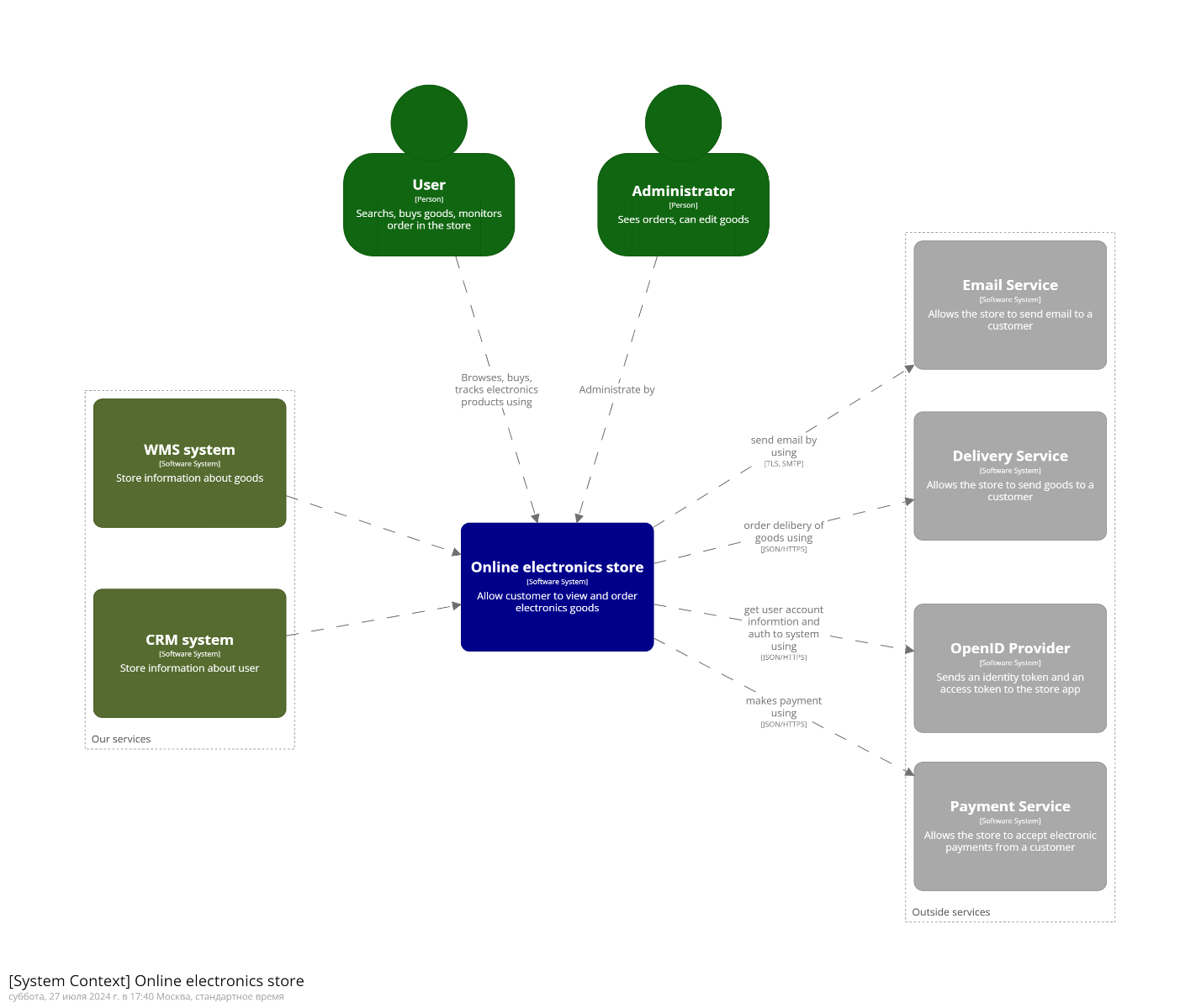


Рисунок Диаграмма контекста системы "Online electronics store"

## Уровень 2: диаграмма контейнеров системы

На следующем диаграмме (см Рисунок 5)мы углубляемся в систему “Online Electronic store”. На данной схеме видно, что

* 1. пользователи взаимодействуют могут взамодействовать с системой через единственную точку “Front Server”, использую https протокол. Других систем, таких как одностраничное приложение или мобильное приложение нет
  2. “Front Server” – интерпретирует действия пользователя в браузере и формирует на “Back server” запросы для получения и изменения данных.
  3. “Back server” взаимодействует с базой данной через протокол. И в приложении используется брокер сообщений

На данной схеме видно, что с внутренними сервисами компании приложение взаимодействует асинхронно через брокер сообщений. А с внешними сервисами используются синхронный формат интеграций.

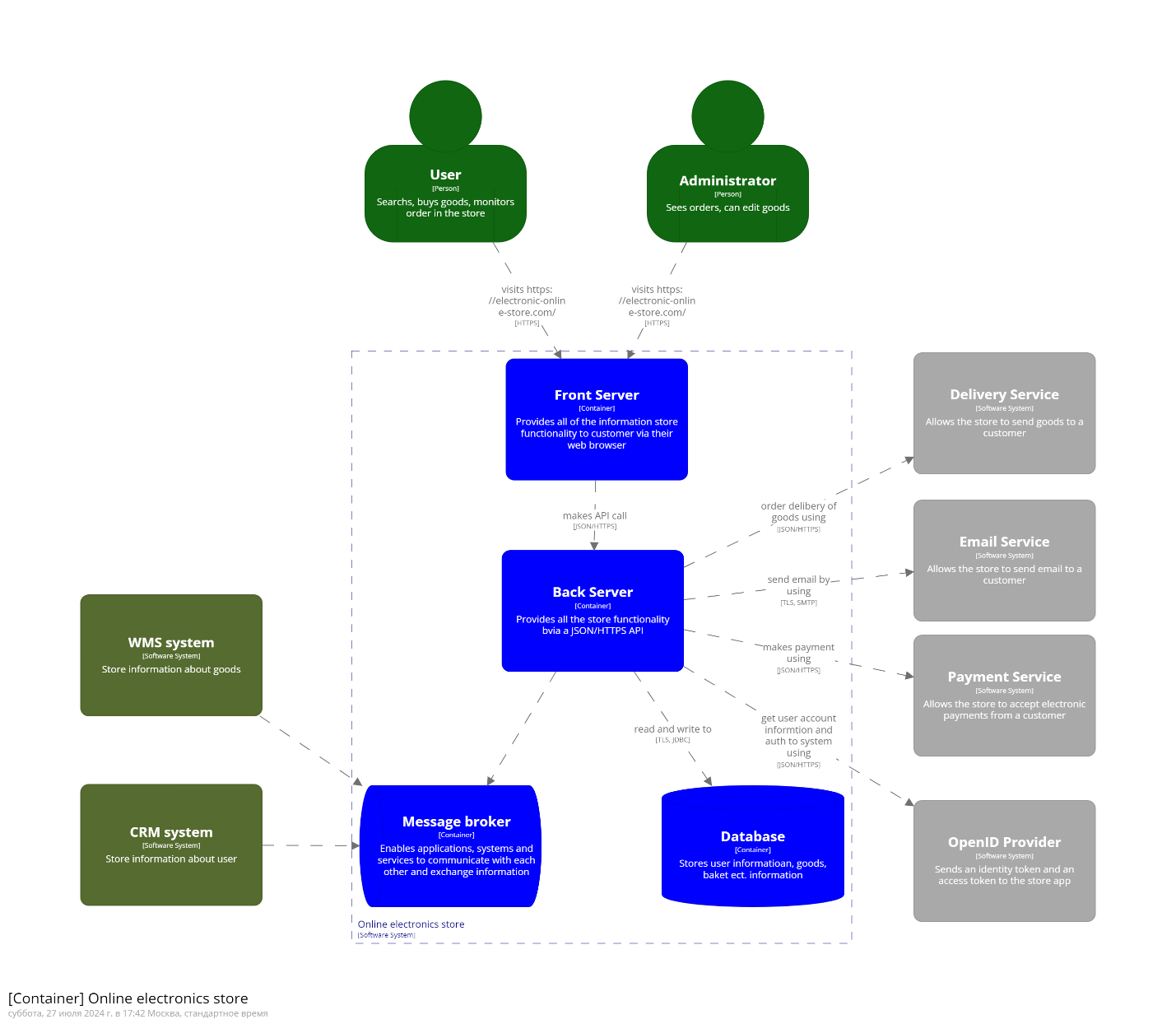


Рисунок : диаграмма контейнеров системы "Online electronic store"

## Уровень 3: диаграмма компонентов системы

Для полноты картины углубление в следующий уровень произведено в двух контейнерах «Back Server» и «Database». Данный уровень детализирует структуру внутри контейнеров системы (см Рисунок 6).

Здесь мы видим, что все запросы от “Front Server” проходят через GatewayAPI, который уже сам маршрутизирует запросы на нужный микросервис. Сами микросервисы имеют свою базу хранения данных и взаимодействуют между собой через GatewayAPI в случае синхронных взаимодействий либо через брокер сообщений, если моментального ответа не требуется (например для создания почтовых уведомлений)

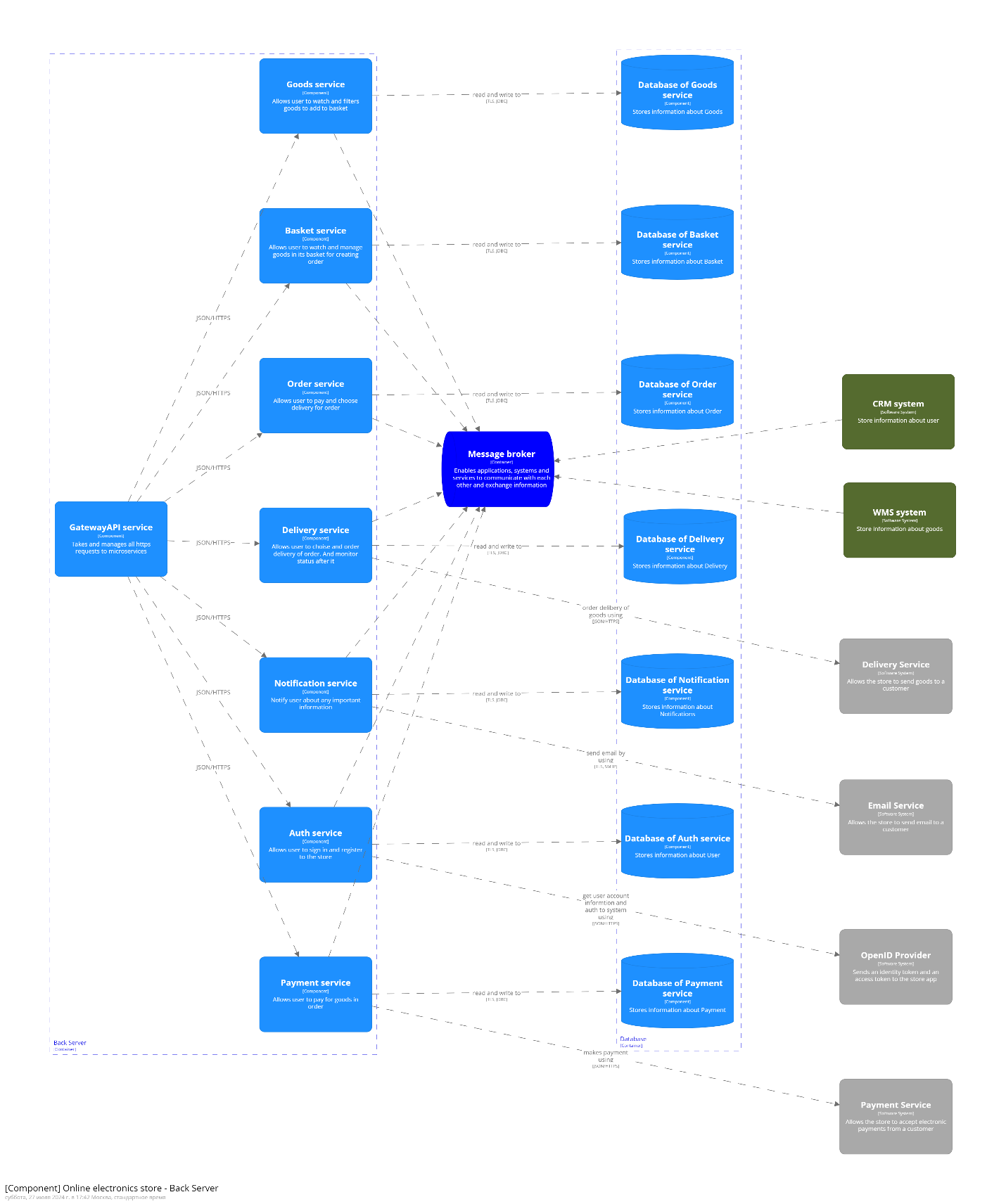


Рисунок диаграмма компонентов систем "Back Server" и "DataBase"

## Основные интерфейсы RestAPI

|  |  |
| --- | --- |
| **КОНТРОЛЛЕР** | **ОПИСАНИЕ** |
| **Аутенфикация и авторизация (/auth)** | Регистрация пользователей. Выполнение аутентификации и авторизации пользователей. Восстановление пароля. |
| **Управление товарами (/products)** | Поиск и просмотр товаров. Добавление товара в корзину. |
| **Управление корзиной (/baskets)** | Просмотр содержимого корзины. Управление товарами в корзине. Создание заказа из содержимого корзины |
| **Управление заказом (/orders)** | Просмотр заказа. Добавление необходимых данных в заказ (платежи и доставка). Оформление заказа. |
| **Управление платежами (/paymetns)** | Создание платежа. Обновление статуса платежа |
| **Управление доставкой(/deliveries))** | Создание доставки |

Аутенфикация и авторизация (/auth):

|  |  |  |
| --- | --- | --- |
| **МЕТОД** | **ИНТЕРФЕЙС** | **ОПИСАНИЕ ИНТЕРФЕЙСА** |
| POST | /auth/sign-in | Вход в систему |
| POST | /auth/sign-out | Выход из системы |
| POST | /auth/registration | Регистриация нового пользователя в системе |
| POST | /auth/password/restore | Скинуть ссылку на смену пароля |

Управление товарами (/products):

|  |  |  |
| --- | --- | --- |
| **МЕТОД** | **ИНТЕРФЕЙС** | **ОПИСАНИЕ ИНТЕРФЕЙСА** |
| POST | /products | Добавление товара (только для администратора) |
| GET | /products/search | Поиск товаров |
| GET | /products/{id} | Просмотр товара по индентификатору |
| PATCH | /products/{id}/add-to-basket | Добавить товар в корзину по индентификатору товара |
| DELETE | /products | Удаление товара (только для администратора) |

Управление корзиной (/basket):

|  |  |  |
| --- | --- | --- |
| **МЕТОД** | **ИНТЕРФЕЙС** | **ОПИСАНИЕ ИНТЕРФЕЙСА** |
| GET | /baskets | Просмотр корзины |
| PATCH | /basket/product/{id}/add | Добавление товара в корзину |
| PATCH | /baskets/product/remove | Удалить товары из корзины |
| PATCH | /baskets/product/{id}/decrease\_amount | Уменьшить на 1 количество товаров в корзине |
| PATCH | /baskets/product/{id}/increase\_amount | Увеличить на 1 количество товаров в корзине |
| POST | /baskets/order | Создать заказ из продуктов в корзине |
| DELETE | /basket/clean | Очищает все товары из корзины |

Управление заказом (/order)

|  |  |  |
| --- | --- | --- |
| **МЕТОД** | **ИНТЕРФЕЙС** | **ОПИСАНИЕ ИНТЕРФЕЙСА** |
| POST | /orders/create | Создание заказа из товаров в корзине |
| GET | /orders/{id} | Получение заказа по индентификатору |
| GET | /orders/search | Просмотр заказов |
| DELETE | /orders/{id} | Удаление заказа |
| PATCH | /orders/{id}/send | Сохранение информации по заказу и оформление |

Управление платежами(/payments):

|  |  |  |
| --- | --- | --- |
| **МЕТОД** | **ИНТЕРФЕЙС** | **ОПИСАНИЕ ИНТЕРФЕЙСА** |
| POST | /payments | Оплата товаров в заказе и как следствие создание платежа |
| PATCH | /payments/{id} | Обновление статусу платежа |

Управление доставкой(/deliveries):

|  |  |  |
| --- | --- | --- |
| **МЕТОД** | **ИНТЕРФЕЙС** | **ОПИСАНИЕ ИНТЕРФЕЙСА** |
| POST | /deliveries | Создание доставки заказа |

На основе описанного скелета АПИ создана спецификация в Swagger-e, которая позволяет интерактивно взаимодействовать с интерфейсом приложения и расширяет описание параметров запроса и вариантов ответа на запрос. Внешний вид первой страницы спецификации можно увидеть на Рисунок 7, а код представлен в Приложение 2: Swagger код для генерации интерактивной генерации в рамках спецификации OpenAPI.

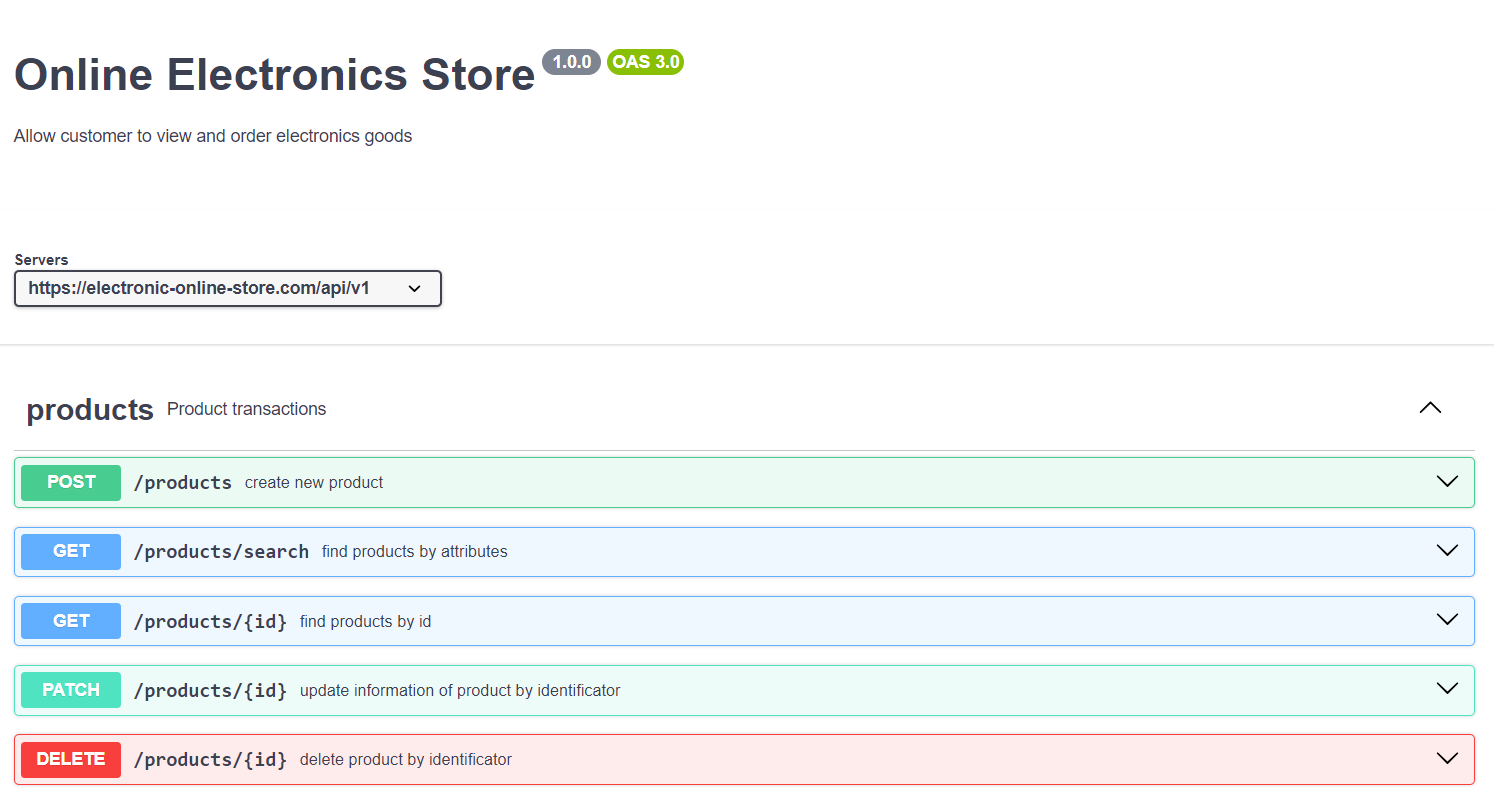


Рисунок : АПИ приложения в Swagger UI

## Детализация компонентов и агрегатов

Структура будущей бд оформлена в виде ER диаграммы на Рисунок 8. Диаграмма составлена при помощи инструмента отрисовывающего диаграмму из кода DBdiagram. Код хранится в главе приложения с названием Приложение 3: Dbdiagram код для генерации ER диаграммы базы данных

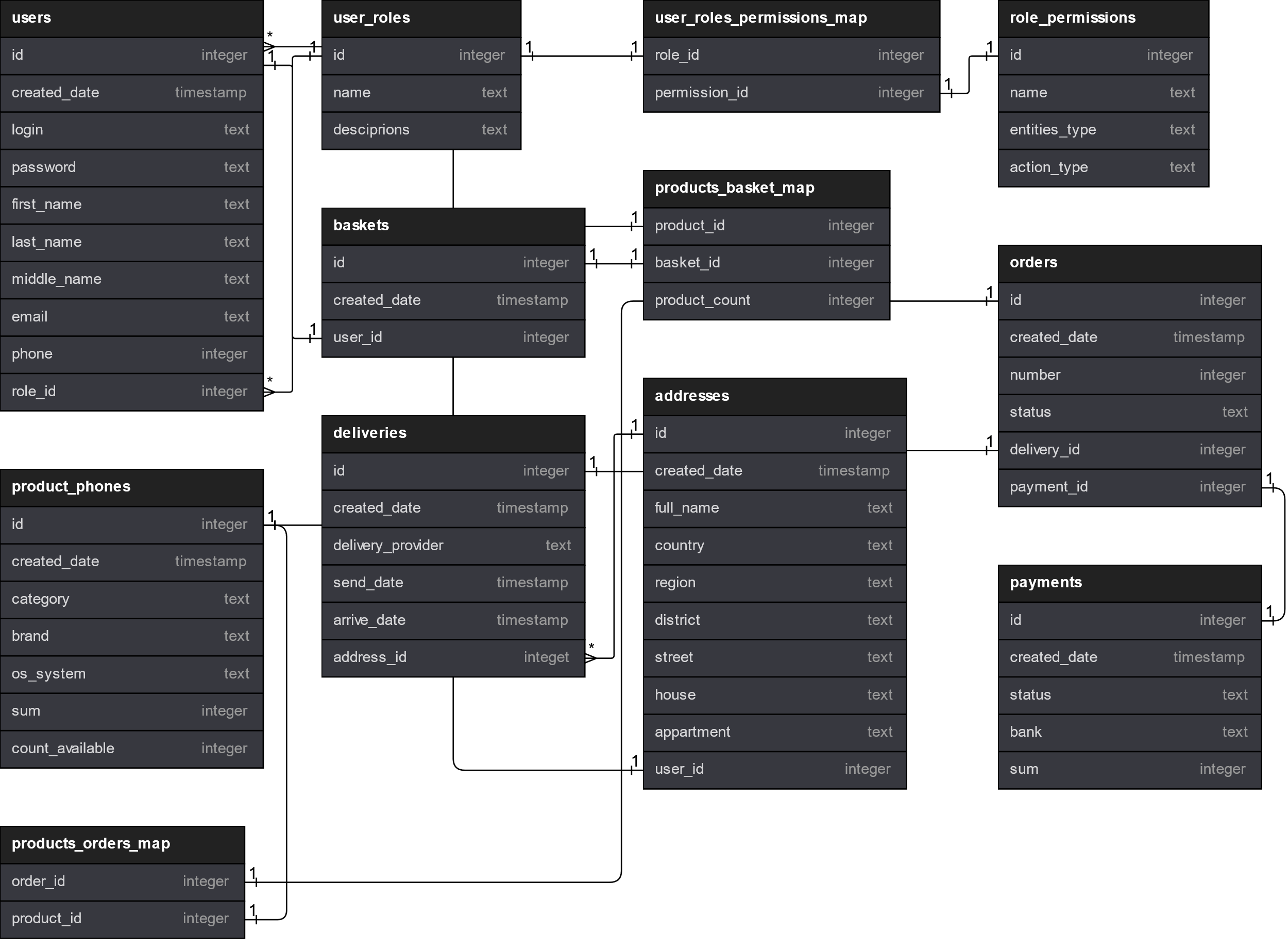


Рисунок : диаграмма отношений таблиц в базе данных

# Динамическое представление

В данном разделе представлены диаграммы последовательности основных процессов на платформе. Так как диаграммы получились довольно, то в данной работе они представлены в векторном фомате, позволяющим без потери качества масштабировать диаграмму. Все диаграммы выполнены при помощи инструмента plantUML, который также работает по принципу «diagrams as code». Код каждой диаграммы представлен в Приложение 4: Plantuml код в для генерации диаграмм последовательностей для пользовательских сценариев.

На Рисунке 7 показан логика входа пользователя в систему через логин и пароль, а также быстрый вход пользователя на примере Яндекс ID. Для изучения интерфейса Яндекса использовалась официальная документация по ссылке https://yandex.ru/dev/id/doc/ru/.

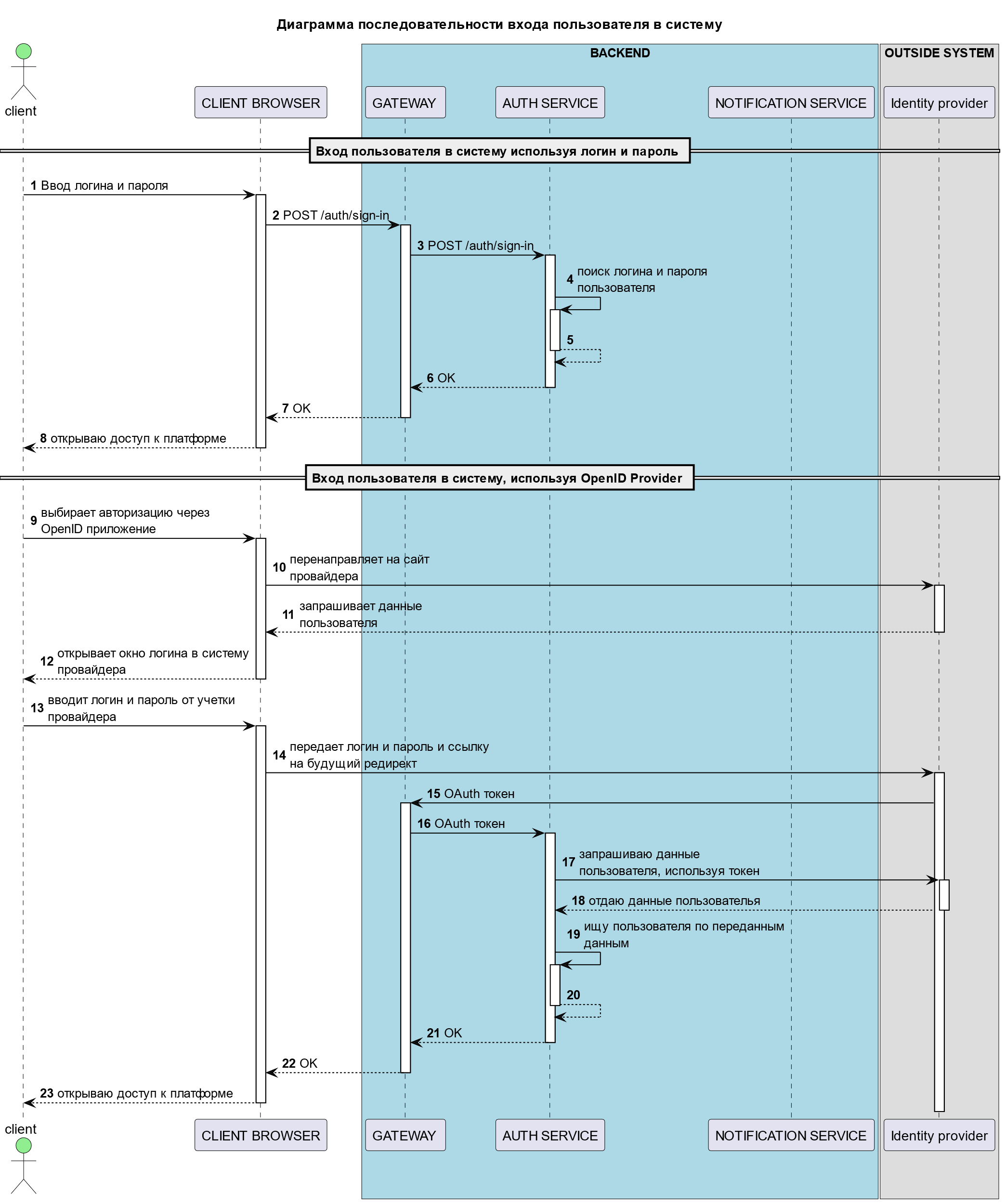


Рисунок : диаграмма последовательности входа пользователя в систему

На Рисунке 8 показана логика поиска и добавления товара в корзину.

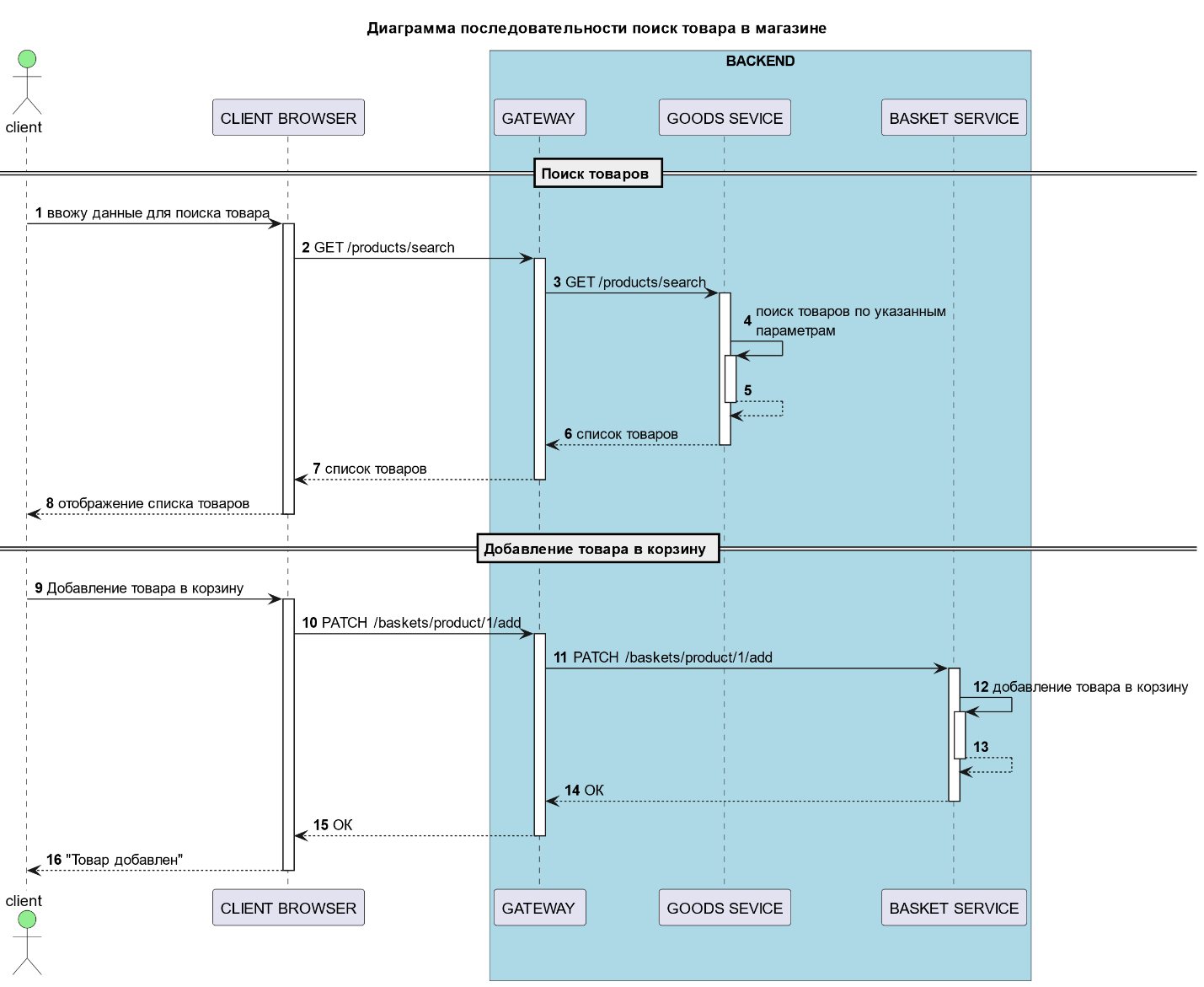


Рисунок : диаграмма последовательности поиска товара в магазине

На Рисунке 9 показан алгоритм оформления заказа пользователем. Чтобы диаграмма излишне не разрослась в размерах подробное описание оплаты товаров и формирования доставки вынесены в отдельные диаграммы.

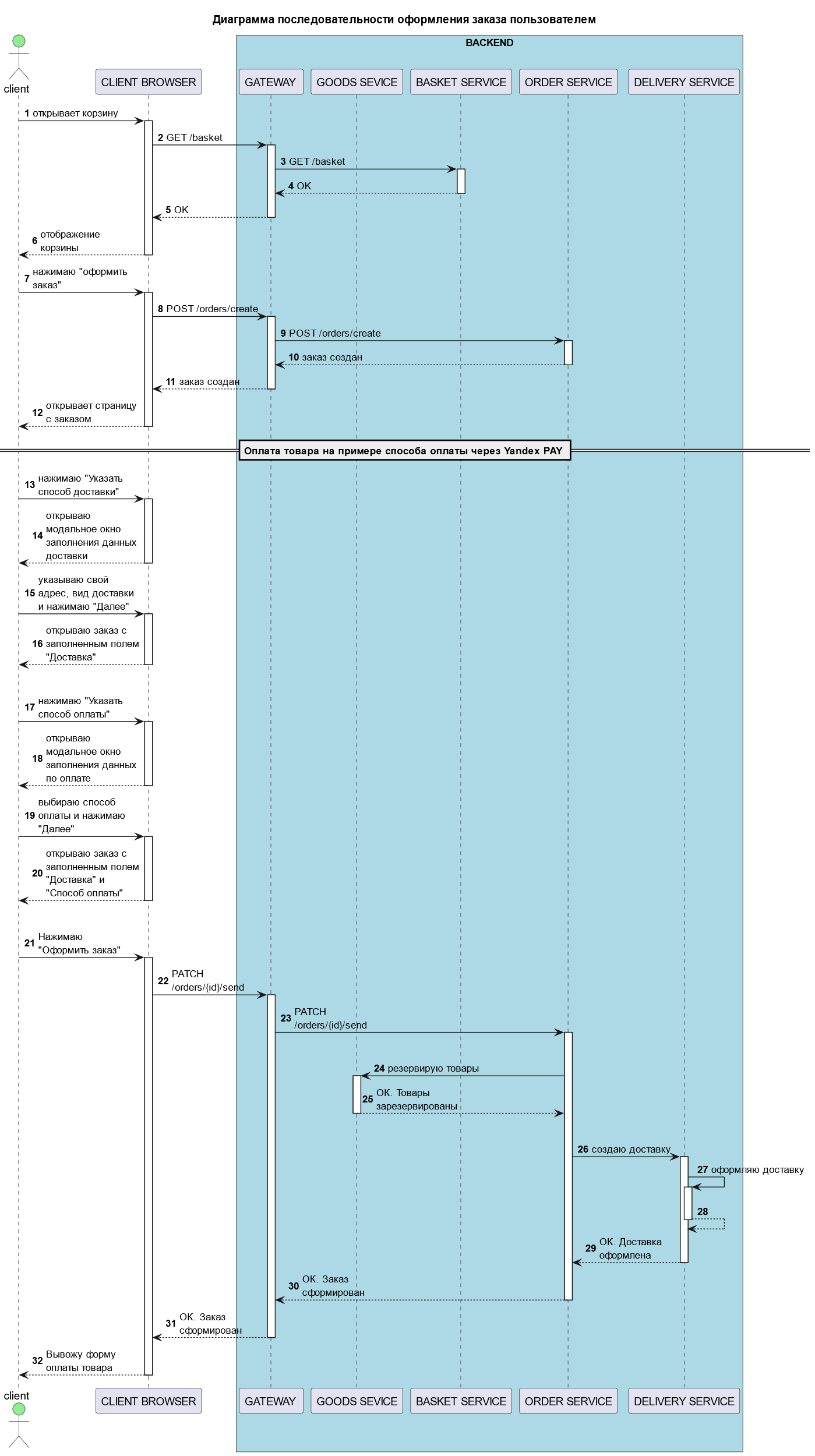


Рисунок : диаграмма последовательности оформления заказа пользователем

На Рисунке 10 описаны процесс проведения оплаты пользователем. В качестве примера сторонней системы оплаты выбран YandexPay. Для изучения АПИ данного сервиса использовалась официальная документация https://pay.yandex.ru/ru/docs/custom/integration-guide.

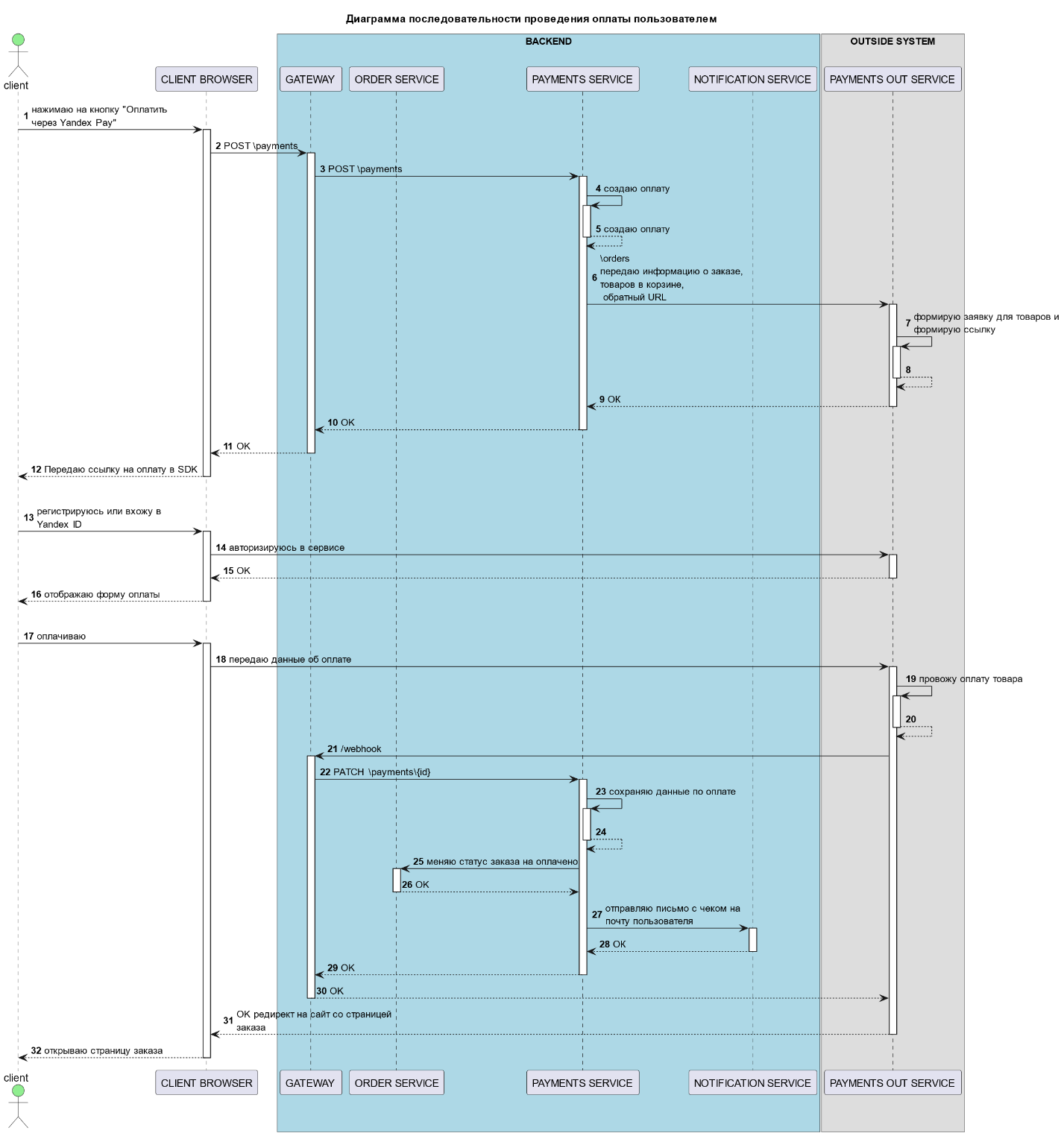


Рисунок : диаграмма последовательности проведения оплаты пользователем

На Рисунке 11 отображена диаграмма оформления доставки заказа на примере сторонней системы доставки СДЭК. Для изучения их АПИ использовалась официальная докумнтация по ссылке https://www.cdek.ru/ru/integration/api.

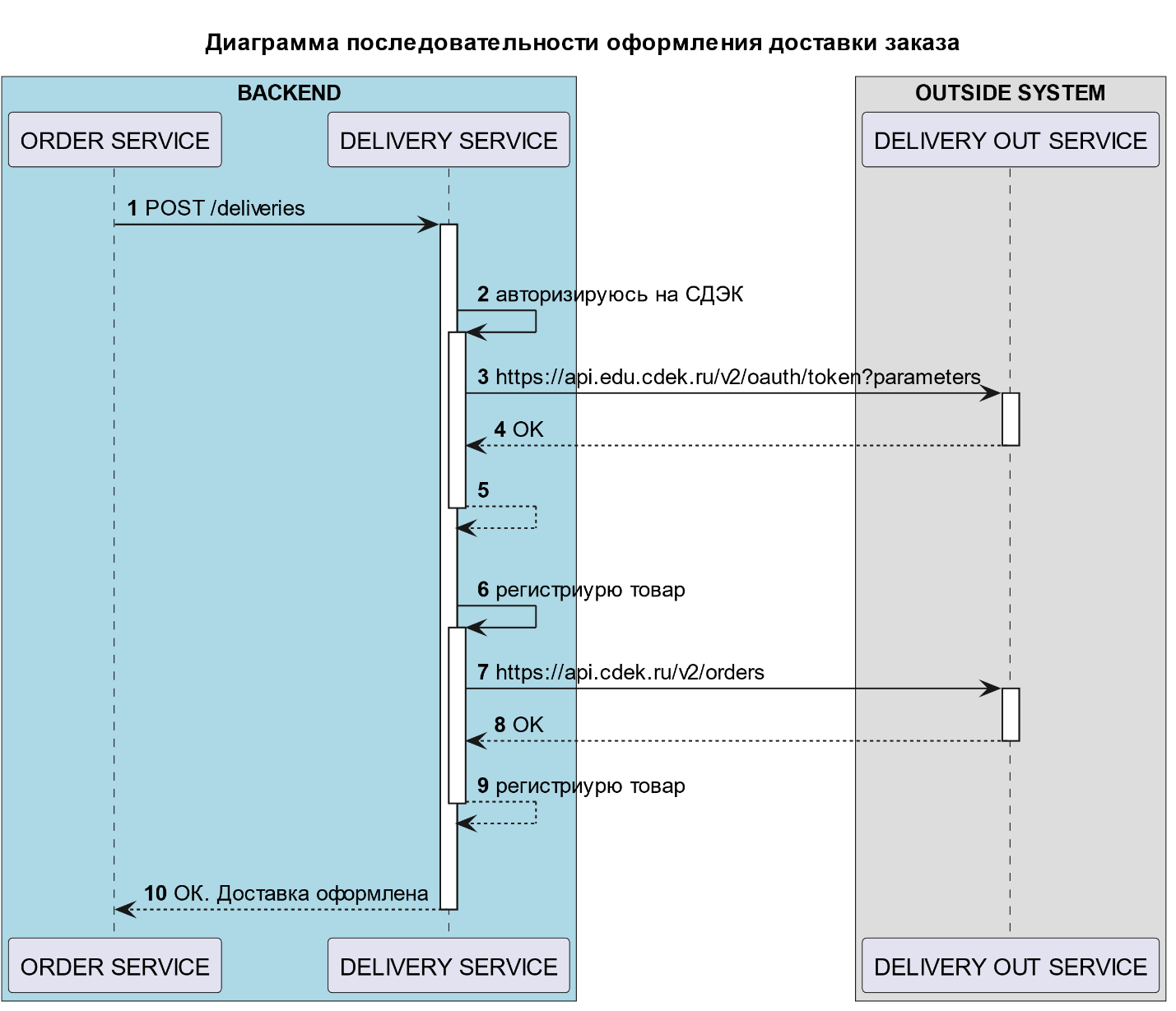


Рисунок : диаграмма последовательности оформления доставки заказа

# Схема развёртывания

|  |  |
| --- | --- |
| **ОСОБЕННОСТИ** | **ОПИСАНИЕ** |
| **Выбор среды исполнения приложения** | Используем контейнерную среду, как наиболее приспособленную к микросервисной архитектуре. Использование "голого железа" для запуска приложения невыгодно со стороны сложности администрирования множества приложений в единой среде. Использование Виртуальных машин отбрасывается, так как микросервисы написаны на общей технологии и требования к ядру среды одинаковы |
| **Выбор репозитория кода** | В качестве репозитория кода выбран GitLab. Проект с открытым исходным кодом, с для хранения проекта в приватном доступе, что делает его более безопасным для хранения чувствительных данных. Возможность создавать личные репозитории, удобный и красивый интерфейс также являются плюсом. |
| **Деплой приложения через GitLab CI** | GitLab CI — это инструмент, встроенный в GitLab. Это один из лучших CI/CD-инструментов, поскольку он предлагает различные функции, такие как просмотр кода, CI/CD, непрерывное развертывание и многое другое в рамках единой информационной панели. |
| **Выбор стратегии развертывания, исключающей время простоя** | Будем использовать стратегию Canary – для обновления пода мы будем разворачивать его реплику из новой версии приложения, пускать на него часть трафика для проверки работоспособности новой версии, после подтверждения работоспособности будем выводить из эксплуатации старый под. |

# Архитектурные и дизайн-решения (ADR)

|  |  |
| --- | --- |
| **ADR** | **АРГУМЕНТАЦИЯ** |
| **Решении о выборе типа архитектуры - «Микросервисная архитектура (MSA)»** | - Независимость процессов проектирования, разработки, тестирования, развертывания и поставки разных сервисов;  - лучшая скорость выхода бизнес-функции на рынок;  - независимая масштабируемость сервисов,  улучшение надежности (отказоустойчивости). |
| **Решение о применении в проекте единой реляционной СУБД с репликацией по типу Master-Master (Active-Passive)** | - для обеспечения должной производительности используем единую СУБД с репликацией в запасную, в случае сбоев используем Passive БД как основную;  - простота развертывания, поддержки; |

# Требования к атрибутам качества

## Дерево атрибутов качества

|  |  |
| --- | --- |
| **АТРИБУТ** | **ОПИСАНИЕ** |
| **Производительность** | Среднее время отклика составляет не менее 3 секунд при одновременной работе на платформе более чем 10000 пользователей без снижения производительности |
| **Готовность** | Время восстановление платформы после аппаратных сбоев – 15 минут, и 5 минут – после сбоя БД. Время работы – 99.9% |
| **Удобство использования** | Дизайн прост и интуитивно понятен. Сценарии пользовательского пути проработаны. Найти товар можно при помощи трех переходов: фильтр, отображение списка товаров, выбранный товар.    Приложение оптимизировано под разные экраны 1920 px, 1200 px, 960 px. В приоритет ставится разрешение для экранов ноутбуков. В будущем планируется распространить верстку на 375 px, для разработки приложения сайта для мобильных устройств. |
| **Гибкость** | - поддерживается непрерывная интеграция и доставка для быстрого и надежного развёртывания изменений;  - платформа имеет конфигурируемый механизм правил, который позволяет вносить изменения без необходимости изменения кода;  - код задокументирован, прописаны комментарии и чёткие инструкции для разработчиков; |
| **Защищенность** | Система перед изменением и отображением данных пользователей проверяет данные на их принадлежность к пользователю. Например, пользователь "А" не может открыть страницу со списком заказов пользователя "В"    Система защищена от SQL инъекций.    Система защищена от ДДОС атак. Невозможно истощить ресурсы системы большим количеством "сложных" запросов.    В системе существует ролевая система прав. Пользователь с ролью "Пользователь" не может использовать функционал доступный роли "Администратор" |
| **Изменчивость** | - платформа состоит из простых компонентов,  которые можно обновлять или заменять, не затрагивая другие части системы;  - поддерживается добавление или удаление аппаратных или программных компонентов без влияния на функциональность платформы;  - платформу легко модифицировать и расширять по мере развития потребностей бизнеса или появления новых требований; |
| **Масштабируемость** | Используется распределенная архитектура, есть возможность увеличивать или уменьшать масштаб в зависимости от спроса. Приложение справляется с внезапным увеличением пользовательского трафика во время пиковых нагрузок. |

## Сценарии проверки атрибутов качества

|  |  |
| --- | --- |
| **РИСК** | **ОПИСАНИЕ** |
| **Производительность** | На платформе проводится короткая по времени скидочная акция из-за чего происходит наплыв большого количества пользователей на систему. Платформа выдерживает требование в производительности |
| **Готовность** | Один из серверов системы отказывает вследствие чрезвычайного происшествия (пожар, потом, конфискация оборудования ит.д.). Система успевает переключиться на дублирующий сервер, не нарушая требование к готовности |
| **Удобство пользователя** | Пользователь на компьютере с разрешением в 1920px решает открыть платформу в двух соседних окнах браузера. Платформа адаптируется к сокращению разрешения в два раза |
| **Защищенность** | Владельцы заказывают проверку платформы на защищенность у внешнего аудита. Платформа соответствует требованиям высокой защищенности персональных данных и недоступности критичного функционала |

# Риски и технический долг

|  |  |
| --- | --- |
| **РИСКИ** | **ОПИСАНИЕ** |
| **Недоступность внешних систем** | Если внешние системы по оплате или доставке откажут, то платформа не сможет оформлять заказы и начнет терять в прибыли |
| **Высокая нагрузка** | При повышении нагрузки выше ожидаемого из-за популярности платформы, кратковременных акций или ддос атак есть риск появления узких мест системы, которые могут негативно повлиять на доступность и производительность системы |
| **Уязвимости в свободном ПО** | Платформа ограничена в выборе платного ПО из-за санкций, из-за чего выбор пал на открытое ПО. Несмотря на плюсы открытого ПО, оно не гарантирует отсутствие уязвимостей. Более того, ввиду открытости исходного кода и продуктов в целом – об уязвимостях становится известно раньше (по сравнению с проприетарным ПО) не только специалистам по кибербезопасности, но и злоумышленникам |
| **Разрастание пакетов с кодом** | Из-за выбранного стиля декомпозиции проекта по фичам есть риск разрастания самих пакетов с кодом в проекте. Это может привести к сложной навигации и как следствию к увеличению времени разработки новых бизнес возможностей |

# Словарь

|  |  |
| --- | --- |
| **ТЕРМИН** | **ОПРЕДЕЛЕНИЕ** |
| **Платформа** | Сайт магазина электронных товаров и программное обеспечение, которое позволяет пользователям совершать с покупки |
| **Пользователь** | Человек вошедший на сайт и использующий его функционал для решения своих задач |
| **Администратор** | Сотрудник, управляющий сайтом. Обновляет информацию и по возможности решает проблемы пользователей |
| **Товар** | Товар, представленный на платформе, который можно приобрести при наличии |
| **Категория товара** | Группа товаров, которые воспринимаются покупателями как взаимосвязанные и взаимозаменяемые |
| **Фильтр** | Настройка поиска по характеристикам товара для быстрого нахождения нужного пользователю товара |
| **Корзина** | Область куда пользователь добавляет товары для будущей покупки и откуда оформляет заказ |
| **Заказ** | Оформленная покупка товара |
| **Платеж** | Описание уплаты заказа пользователем |
| **Доставка** | Описание способа и времени получения заказа с товарами пользователем |

# Приложения

## Приложение 1: Structurizr код для генерации C4 модели архитектуры приложения

|  |
| --- |
| workspace {  # !identifiers hierarchical  model {  u = person "User" "Searchs, buys goods, monitors order in the store"  a = person "Administrator" "Sees orders, can edit goods"  ss\_app = softwareSystem "Online electronics store" "Allow customer to view and order electronics goods"{  cr\_front = container "Front Server" "Provides all of the information store functionality to customer via their web browser "    cr\_back = container "Back Server" "Provides all the store functionality bvia a JSON/HTTPS API" {  ct\_gateway = component "GatewayAPI service" "Takes and manages all https requests to microservices"  ct\_auth = component "Auth service" "Allows user to sign in and register to the store"  ct\_goods = component "Goods service" "Allows user to watch and filters goods to add to basket"  ct\_basket = component "Basket service" "Allows user to watch and manage goods in its basket for creating order"  ct\_orders = component "Order service" "Allows user to pay and choose delivery for order"  ct\_payment = component "Payment service" "Allows user to pay for goods in order"  ct\_delivery = component "Delivery service" "Allows user to choise and order delivery of order. And monitor status after it"  ct\_notification = component "Notification service" "Notify user about any important information"  }    cr\_queue = container "Message broker" "Enables applications, systems and services to communicate with each other and exchange information"{  tags "message\_broker"  }    db = container "Database" "Stores user informatioan, goods, baket ect. information "{  tags "database"  ct\_goods\_db = component "Database of Goods service" "Stores information about Goods"{  tags "database"  }  ct\_auth\_db = component "Database of Auth service" "Stores information about User"{  tags "database"  }  ct\_basket\_db = component "Database of Basket service" "Stores information about Basket"{  tags "database"  }  ct\_orders\_db = component "Database of Order service" "Stores information about Order"{  tags "database"  }  ct\_payment\_db = component "Database of Payment service" "Stores information about Payment"{  tags "database"  }  ct\_delivery\_db = component "Database of Delivery service" "Stores information about Delivery"{  tags "database"  }  ct\_notification\_db = component "Database of Notification service" "Stores information about Notifications"{  tags "database"  }  }  }  group "Our services" {  sscrm = softwareSystem "CRM system" "Store information about user" {  tags "inserv\_tag"  }    sswms = softwareSystem "WMS system" "Store information about goods"{  tags "inserv\_tag"  }  }  group "Outside services" {  ssauth = softwareSystem "OpenID Provider" "Sends an identity token and an access token to the store app " {  tags "outserv\_tag"  }  sspay = softwareSystem "Payment Service" "Allows the store to accept electronic payments from a customer" {  tags "outserv\_tag"  }  ssdel = softwareSystem "Delivery Service" "Allows the store to send goods to a customer" {  tags "outserv\_tag"  }  ssemail = softwareSystem "Email Service" "Allows the store to send email to a customer" {  tags "outserv\_tag"  }  }  u -> ss\_app "Browses, buys, tracks electronics products using"  u -> cr\_front "visits https://electronic-online-store.com/" "HTTPS"  a -> ss\_app "Administrate by "  a -> cr\_front "visits https://electronic-online-store.com/" "HTTPS"  cr\_front -> ct\_gateway "makes API call" "JSON/HTTPS"  cr\_back -> db "read and write to" "TLS, JDBC"    ct\_auth -> ct\_auth\_db "read and write to" "TLS, JDBC"  ct\_goods -> ct\_goods\_db "read and write to" "TLS, JDBC"  ct\_basket -> ct\_basket\_db "read and write to" "TLS, JDBC"  ct\_payment -> ct\_payment\_db "read and write to" "TLS, JDBC"  ct\_orders -> ct\_orders\_db "read and write to" "TLS, JDBC"  ct\_delivery -> ct\_delivery\_db "read and write to" "TLS, JDBC"  ct\_notification -> ct\_notification\_db "read and write to" "TLS, JDBC"  ct\_gateway -> ct\_auth "JSON/HTTPS"  ct\_gateway -> ct\_goods "JSON/HTTPS"  ct\_gateway -> ct\_basket "JSON/HTTPS"  ct\_gateway -> ct\_orders "JSON/HTTPS"  ct\_gateway -> ct\_payment "JSON/HTTPS"  ct\_gateway -> ct\_delivery "JSON/HTTPS"  ct\_gateway -> ct\_notification "JSON/HTTPS"  ct\_auth -> db "TLS, SMTP"  ct\_goods -> db "TLS, SMTP"  ct\_basket -> db "TLS, SMTP"  ct\_payment -> db "TLS, SMTP"  ct\_orders -> db "TLS, SMTP"  ct\_delivery -> db "TLS, SMTP"  ct\_notification -> db "TLS, SMTP"    ct\_auth -> cr\_queue ""  ct\_goods -> cr\_queue ""  ct\_basket -> cr\_queue ""  ct\_payment -> cr\_queue ""  ct\_orders -> cr\_queue ""  ct\_delivery -> cr\_queue ""  ct\_notification -> cr\_queue ""    sscrm -> cr\_queue ""  sswms -> cr\_queue ""    # ct\_goods -> sscrm "exchanges data" "JSON/HTTPS"  # ct\_orders -> sswms "exchanges data" "JSON/HTTPS"  ct\_auth -> ssauth "get user account informtion and auth to system using" "JSON/HTTPS"  ct\_payment -> sspay "makes payment using" "JSON/HTTPS"  ct\_delivery -> ssdel "order delibery of goods using" "JSON/HTTPS"  ct\_notification -> ssemail "send email by using" "TLS, SMTP"  }  views {  systemContext ss\_app "Diagram1" {  include \*  # autolayout tb  }  container ss\_app "Diagram2" {  include \*  # autolayout tb  }  component cr\_back "Diagram3" {  include element.type==component cr\_queue "element.tag==inserv\_tag" "element.tag==outserv\_tag"  # autolayout tb  }    component cr\_back "Diagram4" {  include element.type==component element.type==container element.type==softwareSystem  # autolayout tb  }  styles {  element "Element" {  color white  shape RoundedBox  }  element "Software System" {  #DarkBlue  background #00008B  shape RoundedBox  }  element "Container" {  #Blue  background #0000FF  }  element "Component" {  #DodgerBlue  background #1E90FF  }  element "Person" {  background #116611  shape person  }  element "database" {  shape cylinder  }    element "message\_broker" {  shape pipe  }  element "outserv\_tag" {  background #A9A9A9  }  element "inserv\_tag" {  #DarkOliveGreen  background #556B2F  }  }  }  } |

## Приложение 2: Swagger код для генерации интерактивной генерации в рамках спецификации OpenAPI

|  |
| --- |
| **openapi**: 3.0.0  **info**:    title: Online Electronics Store    description: Allow customer to view and order electronics goods    version: 1.0.0  **servers**:    - url: https://electronic-online-store.com/api/v1  **tags**:    - name: products      description: Product transactions    - name: baskets      description: Basket transactions    - name: orders      description: Order transactions    - name: payments      description: Payment transactions    - name: delivery      description: Delivery transactions    - name: auth      description: Access to Online electronics store  **paths**:  **/products**:  **post**:        tags:          - products        summary: create new product  *responses*:          '200':            description: OK            content:              application/json:  **schema**:                  $ref: *'#/components/schemas/Product'*  **'403'**:            $ref: *'#/components/responses/Forbidden'*  **/products/search**:  **get**:        tags:          - products        summary: find products by attributes  *parameters*:          - in: query            name: category            required: false  **schema**:              type: string          - in: query            name: minimum\_sum            required: false  **schema**:              type: integer          - in: query            name: maximum\_sum            required: false  **schema**:              type: integer          - in: query            name: avaliable            required: false  **schema**:              type: boolean          - in: query            name: page            required: true  **schema**:              type: integer          - in: query            name: size            required: true  **schema**:              type: integer  *responses*:          '200':            description: An array of products            content:              application/json:  **schema**:                  type: array  **items**:                    $ref: *'#/components/schemas/Product'*  **'403'**:            $ref: *'#/components/responses/Unauthorized'*  **'500'**:            $ref: *'#/components/responses/UnexpectedError'*  **/products/{id}**:  **get**:        tags:          - products        summary: find products by id  *parameters*:          - in: path            name: id            required: true  **schema**:              type: integer  *responses*:          '200':            description: The product            content:              application/json:  **schema**:                  $ref: *'#/components/schemas/Product'*  **patch**:        tags:          - products        summary: update information of product by identificator  *responses*:          '200':            description: OK            content:              application/json:  **schema**:                  $ref: *'#/components/schemas/Product'*  **'403'**:            $ref: *'#/components/responses/Forbidden'*  **delete**:        tags:          - products        summary: delete product by identificator  *responses*:          '200':            description: OK            content:              application/json:  **schema**:                  $ref: *'#/components/schemas/Product'*  **'403'**:            $ref: *'#/components/responses/Forbidden'*  **/baskets**:  **get**:        tags:          - baskets        summary: open own basket  *responses*:          '200':            description: Basket with products            content:              application/json:  **schema**:                  $ref: *'#/components/schemas/Basket'*  **/baskets/product/{id}/add**:  **patch**:        tags:          - baskets        summary: add product to basket  *parameters*:          - in: path            name: id            required: true  **schema**:              type: integer  *responses*:          '200':            description: OK  **/baskets/product/{id}/remove**:  **patch**:        tags:          - baskets        summary: remove product from basket  *parameters*:          - in: path            name: id            required: true  **schema**:              type: integer  *responses*:          '200':            description: 'OK'  **/baskets/product/{id}/deacrease-amount**:  **patch**:        tags:          - baskets        summary: descriment count of product in the basket  *parameters*:          - in: path            name: id            required: true            description: 'Product ID'  **schema**:              type: integer  *responses*:          '200':            description: 'OK'  **/baskets/product/{id}/increase-amount**:  **patch**:        tags:          - baskets        summary: increment count of product in the basket  *parameters*:          - in: path            name: id            required: true            description: 'Product ID'  **schema**:              type: integer  *responses*:          '200':            description: 'OK'  **/basket/clean**:  **delete**:        tags:          - baskets        summary: clean basket from all goods  *responses*:          '200':            description: 'OK'  **/orders/{id}**:  **get**:        tags:          - orders        summary: find order by id  *parameters*:          - in: path            name: id            description: Order ID            required: true  **schema**:              type: integer  *responses*:          '200':            description: Order            content:              application/json:  **schema**:                  $ref: *'#/components/schemas/Order'*  **delete**:        tags:          - orders        summary: delete order  *parameters*:          - in: path            name: id            description: Order ID            required: true  **schema**:              type: integer  *responses*:          '200':            description: Order is deleted  **/orders/create**:  **post**:        tags:          - orders        summary: create order from products in basket  *responses*:          '200':            description: 'OK'  **/orders/search**:  **get**:        tags:          - orders        summary: find order by attributes  *parameters*:          - in: query            name: number            description: Order number  **schema**:              type: string          - in: query            name: created\_date\_from  **schema**:              type: string          - in: query            name: created\_date\_to  **schema**:              type: string          - in: query            name: status  **schema**:              $ref: *'#/components/schemas/OrderStatusEnum'*  *responses*:          '200':            description: Order            content:              application/json:  **schema**:                  type: array  **items**:                    $ref: *'#/components/schemas/Order'*  **/orders/{id}/send**:  **patch**:        tags:          - orders        summary: pays order and send  *parameters*:          - in: path            name: id            required: true  **schema**:              type: integer  *responses*:          '200':            description: Order send  **/deliveries**:  **post**:        tags:          - delivery        summary: create delivery for the order  *responses*:          '200':            description: OK  **/payments**:  **post**:        tags:          - payments        summary: create payment for the order  *responses*:          '200':            description: OK  **/payments/{id}**:  **patch**:        tags:          - payments        summary: update status of payment  *parameters*:          - in: path            name: id            required: true  **schema**:              type: integer  *responses*:          '200':            description: OK  **/auth/sign-in**:  **post**:        tags:          - auth        summary: login to system  *responses*:          '200':            description: OK  **/auth/sign-out**:  **post**:        tags:          - auth        summary: logout from system  *responses*:          '200':            description: OK  **/auth/registration**:  **post**:        tags:          - auth        summary: registration in platform  *responses*:          '200':            description: OK  **/auth/password/restore**:  **post**:        tags:          - auth        summary: create link to restore forgotten password  *responses*:          '200':            description: OK  **components**:  **schemas**:  **Product**:        required:          - id        type: object        properties:  **id**:            type: integer            format: int64            example: 1  **created\_date**:            type: string            format: date-time            example: '2024-07-13 00:00:000'  **category**:            type: string            example: 'phones'  **sum**:            type: integer            example: '10000'  **avaliable\_cnt**:            type: integer            example: 7  **Basket**:        type: object        properties:  **id**:            type: integer            format: int64            example: 1  **products**:            type: array  **items**:              $ref: *'#/components/schemas/Product'*  **OrderStatusEnum**:        type: string        format: data        enum:          - unsent          - payed          - sent          - arrived          - error          - stopped  **Order**:        type: object        properties:  **id**:            type: integer            format: int64            example: 1  **created\_date**:            type: string            format: data  **number**:            type: integer            format: int64  **status**:            $ref: *'#/components/schemas/OrderStatusEnum'*  **delivery\_id**:            type: integer            format: int64  **payment\_id**:            type: integer            format: int64  **responses**:      Unauthorized:        description: Unauthorized.      UnexpectedError:        description: Unexpected error.      Forbidden:        description: This action is not within your authorized scope. |

## Приложение 3: Dbdiagram код для генерации ER диаграммы базы данных

|  |
| --- |
| // Use DBML to define your database structure  // Docs: https://dbml.dbdiagram.io/docs  Table users {    id integer    created\_date timestamp    login text    password text    first\_name text    last\_name text    middle\_name text    email text    phone integer    role\_id integer  }  Table user\_roles {    id integer    name text    desciprions text  }  Table user\_roles\_permissions\_map {    role\_id integer    permission\_id integer  }  Table role\_permissions {    id integer    name text    entities\_type text    action\_type text  }  Table product\_phones {    id integer    created\_date timestamp    category text    brand text    os\_system text    sum integer    count\_available integer  }  Table baskets {    id integer    created\_date timestamp    user\_id integer  }  Table products\_basket\_map {    product\_id integer    basket\_id integer    product\_count integer  }  Table orders {    id integer    created\_date timestamp    number integer    status text    delivery\_id integer    payment\_id integer  }  Table products\_orders\_map {    order\_id integer    product\_id integer  }  Table deliveries {    id integer    created\_date timestamp    delivery\_provider text    send\_date timestamp    arrive\_date timestamp    address\_id integet  }  Table addresses {    id integer    created\_date timestamp    full\_name text    country text    region text    district text    street text    house text    appartment text    user\_id integer  }  Table payments {    id integer    created\_date timestamp    status text    bank text    sum integer  }  /\*  users --> user\_roles  user\_roles\_permissions\_map --> user\_roles  user\_roles\_permissions\_map --> role\_permissions  addresses --> users  baskets --> users  products\_basket\_map -> product\_phones  baskets -> product\_phones  orders -> payments  orders -> deliveries  products\_orders\_map -> orders  products\_orders\_map -> product\_phones  \*/  Ref: users.role\_id > user\_roles.id  Ref: user\_roles\_permissions\_map.role\_id - user\_roles.id  Ref: user\_roles\_permissions\_map.permission\_id - role\_permissions.id  Ref: addresses.user\_id < users.id  Ref: baskets.user\_id - users.id  Ref: products\_basket\_map.basket\_id - baskets.id  Ref: products\_basket\_map.product\_id - product\_phones.id  Ref: orders.delivery\_id - deliveries.id  Ref: orders.payment\_id - payments.id  Ref: products\_orders\_map.order\_id - orders.id  Ref: products\_orders\_map.product\_id - product\_phones.id  Ref: deliveries.address\_id > addresses.id |

## Приложение 4: Plantuml код в для генерации диаграмм последовательностей для пользовательских сценариев

## Код диаграммы последовательности входа пользователя в систему

|  |
| --- |
| @startuml '<SETTINGS>' autonumber autoactivate on skinparam maxMessageSize 210 '</SETTINGS>' title "Диаграмма последовательности входа пользователя в систему"  Actor client #LightGreen 'Actor administrator #Red participant "CLIENT BROWSER" as browser ' box "BACKEND" #LightBlue  participant GATEWAY as gateway  participant "AUTH SERVICE" as auth ' participant "GOODS SEVICE" as goods ' participant "BASKET SERVICE" as basket ' participant "ORDER SERVICE" as orders ' participant "DELIVERY SERVICE" as delivery ' participant "PAYMENTS SERVICE" as payments  participant "NOTIFICATION SERVICE" as notification end box ' 'box "INTERNAL SYSTEM" #Yellow 'participant "CRM" as crm 'participant "WMS" as wms 'end box ' box "OUTSIDE SYSTEM" participant "Identity provider" as idpr\_out\_serv 'participant "DELIVERY OUT SERVICE" as delivery\_out\_serv 'participant "PAYMENTS OUT SERVICE" as payments\_out\_serv end box '<\PARTICIPANTS>'  '<LOGIC>' == Вход пользователя в систему используя логин и пароль == client -> browser: Ввод логина и пароля browser -> gateway: POST /auth/sign-in gateway -> auth: POST /auth/sign-in auth -> auth: поиск логина и пароля пользователя auth --> auth: auth --> gateway: OK gateway --> browser: OK browser --> client: открываю доступ к платформе == Вход пользователя в систему, используя OpenID Provider == client -> browser: выбирает авторизацию через OpenID приложение browser -> idpr\_out\_serv: перенаправляет на сайт провайдера idpr\_out\_serv --> browser: запрашивает данные пользователя browser --> client: открывает окно логина в систему провайдера client -> browser: вводит логин и пароль от учетки провайдера browser -> idpr\_out\_serv: передает логин и пароль и ссылку на будущий редирект idpr\_out\_serv -> gateway: OAuth токен gateway -> auth: OAuth токен auth -> idpr\_out\_serv: запрашиваю данные пользователя, используя токен idpr\_out\_serv --> auth: отдаю данные пользователья auth -> auth: ищу пользователя по переданным данным auth --> auth: auth --> gateway: OK gateway --> browser: OK browser --> client: открываю доступ к платформе '<\LOGIC>'  @enduml |

## Код диаграммы последовательности поиска товара в магазине

|  |
| --- |
| @startuml  '<SETTINGS>' autonumber autoactivate on skinparam maxMessageSize 210 '</SETTINGS>'  title "Диаграмма последовательности поиска товара в магазине"  '<PARTICIPANTS>' Actor client #LightGreen participant "CLIENT BROWSER" as browser ' box "BACKEND" #LightBlue  participant GATEWAY as gateway  participant "GOODS SEVICE" as goods  participant "BASKET SERVICE" as basket end box '<\PARTICIPANTS>'  '<LOGIC>' == Поиск товаров == client -> browser: ввожу данные для поиска товара browser -> gateway: GET /products/search gateway -> goods: GET /products/search goods -> goods: поиск товаров по указанным параметрам goods --> goods: goods --> gateway: список товаров gateway --> browser: список товаров browser --> client: отображение списка товаров == Добавление товара в корзину == client -> browser: Добавление товара в корзину browser -> gateway: PATCH /baskets/product/1/add gateway -> basket: PATCH /baskets/product/1/add basket -> basket: добавление товара в корзину basket --> basket basket --> gateway: ОК gateway --> browser: ОК browser --> client: "Товар добавлен" '<\LOGIC>'  @enduml |

## Код диаграммы последовательности оформления заказа пользователем

|  |
| --- |
| @startuml '<SETTINGS>' autonumber autoactivate on skinparam maxMessageSize 125 '</SETTINGS>'  title "Диаграмма последовательности оформления заказа пользователем"  '<PARTICIPANTS>' Actor client #LightGreen 'Actor administrator #Red participant "CLIENT BROWSER" as browser ' box "BACKEND" #LightBlue  participant GATEWAY as gateway ' participant "AUTH SERVICE" as auth  participant "GOODS SEVICE" as goods  participant "BASKET SERVICE" as basket  participant "ORDER SERVICE" as orders  participant "DELIVERY SERVICE" as delivery ' participant "PAYMENTS SERVICE" as payments ' participant "NOTIFICATION SERVICE" as notification end box ' 'box "INTERNAL SYSTEM" #Yellow 'participant "CRM" as crm 'participant "WMS" as wms 'end box ' 'box "OUTSIDE SYSTEM" 'participant "Identity provider" as idpr 'participant "DELIVERY OUT SERVICE" as delivery\_out\_serv 'participant "PAYMENTS OUT SERVICE" as payments\_out\_serv 'end box '<\PARTICIPANTS>'  '<LOGIC>' client -> browser: открывает корзину browser -> gateway: GET /basket gateway -> basket: GET /basket basket --> gateway: OK gateway --> browser: OK browser --> client: отображение корзины client -> browser: нажимаю "оформить заказ" browser -> gateway: POST /orders/create gateway -> orders: POST /orders/create 'orders -> orders: создаю заказ с продуктами orders --> gateway: заказ создан gateway --> browser: заказ создан browser --> client: открывает страницу с заказом   == Оплата товара на примере способа оплаты через Yandex PAY==  client -> browser: нажимаю "Указать способ доставки" browser --> client: открываю модальное окно заполнения данных доставки client -> browser: указываю свой адрес, вид доставки и нажимаю "Далее" browser --> client: открываю заказ с заполненным полем "Доставка" ||| client -> browser: нажимаю "Указать способ оплаты" browser --> client: открываю модальное окно заполнения данных по оплате client -> browser: выбираю способ оплаты и нажимаю "Далее" browser --> client: открываю заказ с заполненным полем "Доставка" и "Способ оплаты" ||| ''https://pay.yandex.ru/ru/docs/payments/basic client -> browser: Нажимаю "Оформить заказ" browser -> gateway: PATCH /orders/{id}/send gateway -> orders: PATCH /orders/{id}/send ||| orders -> goods: резервирую товары 'goods -> goods: резервирую товары goods --> orders: ОК. Товары зарезервированы ||| orders -> delivery: создаю доставку delivery -> delivery: оформляю доставку return delivery --> orders: ОК. Доставка оформлена '||| 'orders -> payments: провожу оплату 'payments -> payments: провожу оплату 'return 'payments --> orders: ОК. Заказ оплачен '||| 'orders -> notification: отправляю сообщение с чеком операции ''notification -> notification: отправляю через сервис письмо на почту клиента 'notification --> orders: ОК. Письмо отправлено. '||| 'orders -> basket: очищаю корзину ''basket -> basket: удаляю товары из корзины 'basket --> orders: ОК. корзина очищена   'orders -> orders: меняю статус заказа orders --> gateway: ОК. Заказ сформирован gateway --> browser: ОК. Заказ сформирован browser --> client: Вывожу форму оплаты товара  '<\LOGIC>'  @enduml |

## Код диаграммы последовательности проведения оплаты пользователем

|  |
| --- |
| @startuml '<SETTINGS>' autonumber autoactivate on skinparam maxMessageSize 210 '</SETTINGS>'  title "Диаграмма последовательности проведения оплаты пользователем"  '<PARTICIPANTS>' Actor client #LightGreen 'Actor administrator #Red participant "CLIENT BROWSER" as browser ' box "BACKEND" #LightBlue  participant GATEWAY as gateway ' participant "AUTH SERVICE" as auth ' participant "GOODS SEVICE" as goods ' participant "BASKET SERVICE" as basket  participant "ORDER SERVICE" as orders ' participant "DELIVERY SERVICE" as delivery  participant "PAYMENTS SERVICE" as payments  participant "NOTIFICATION SERVICE" as notification end box ' 'box "INTERNAL SYSTEM" #Yellow 'participant "CRM" as crm 'participant "WMS" as wms 'end box ' box "OUTSIDE SYSTEM" 'participant "Identity provider" as idpr\_out\_serv 'participant "DELIVERY OUT SERVICE" as delivery\_out\_serv participant "PAYMENTS OUT SERVICE" as payments\_out\_serv end box '<\PARTICIPANTS>'  '<LOGIC>' ''https://pay.yandex.ru/ru/docs/payments/basic client -> browser: нажимаю на кнопку "Оплатить через Yandex Pay" browser -> gateway: POST \payments gateway -> payments: POST \payments payments -> payments: создаю оплату payments --> payments: создаю оплату payments -> payments\_out\_serv: \orders \nпередаю информацию о заказе, \nтоваров в корзине, \n обратный URL payments\_out\_serv -> payments\_out\_serv: формирую заявку для товаров и формирую ссылку payments\_out\_serv --> payments\_out\_serv: payments\_out\_serv --> payments: OК payments --> gateway: OK gateway --> browser: OK browser --> client: Передаю ссылку на оплату в SDK ||| client -> browser: регистрируюсь или вхожу в Yandex ID browser -> payments\_out\_serv: авторизируюсь в сервисе payments\_out\_serv --> browser: OK browser --> client: отображаю форму оплаты ||| client -> browser: оплачиваю browser -> payments\_out\_serv: передаю данные об оплате payments\_out\_serv -> payments\_out\_serv: провожу оплату товара payments\_out\_serv --> payments\_out\_serv payments\_out\_serv -> gateway: /webhook gateway -> payments: PATCH \payments\{id} payments -> payments: сохраняю данные по оплате payments --> payments: payments -> orders: меняю статус заказа на оплачено orders --> payments: OK payments -> notification: отправляю письмо с чеком на почту пользователя notification --> payments: OК payments --> gateway: OK gateway --> payments\_out\_serv: OK payments\_out\_serv --> browser: OK редирект на сайт со страницей заказа browser --> client: открываю страницу заказа  '<\LOGIC>'  @enduml |

## Код диаграммы последовательности оформления доставки заказа

|  |
| --- |
| @startuml '<SETTINGS>' autonumber autoactivate on skinparam maxMessageSize 210 '</SETTINGS>'   title "Диаграмма последовательности оформления доставки заказа"  '<PARTICIPANTS>' box "BACKEND" #LightBlue participant "ORDER SERVICE" as orders participant "DELIVERY SERVICE" as delivery end box box "OUTSIDE SYSTEM" participant "DELIVERY OUT SERVICE" as delivery\_out\_serv end box '<\PARTICIPANTS>'  '<LOGIC>' orders -> delivery: POST /deliveries ||| delivery -> delivery: авторизируюсь на СДЭК delivery -> delivery\_out\_serv: https://api.edu.cdek.ru/v2/oauth/token?parameters delivery\_out\_serv --> delivery: OK delivery --> delivery ||| delivery -> delivery: регистриурю товар delivery -> delivery\_out\_serv: https://api.cdek.ru/v2/orders delivery\_out\_serv --> delivery: OK delivery --> delivery: регистриурю товар ||| delivery --> orders: ОК. Доставка оформлена  '<\LOGIC>'  @enduml |