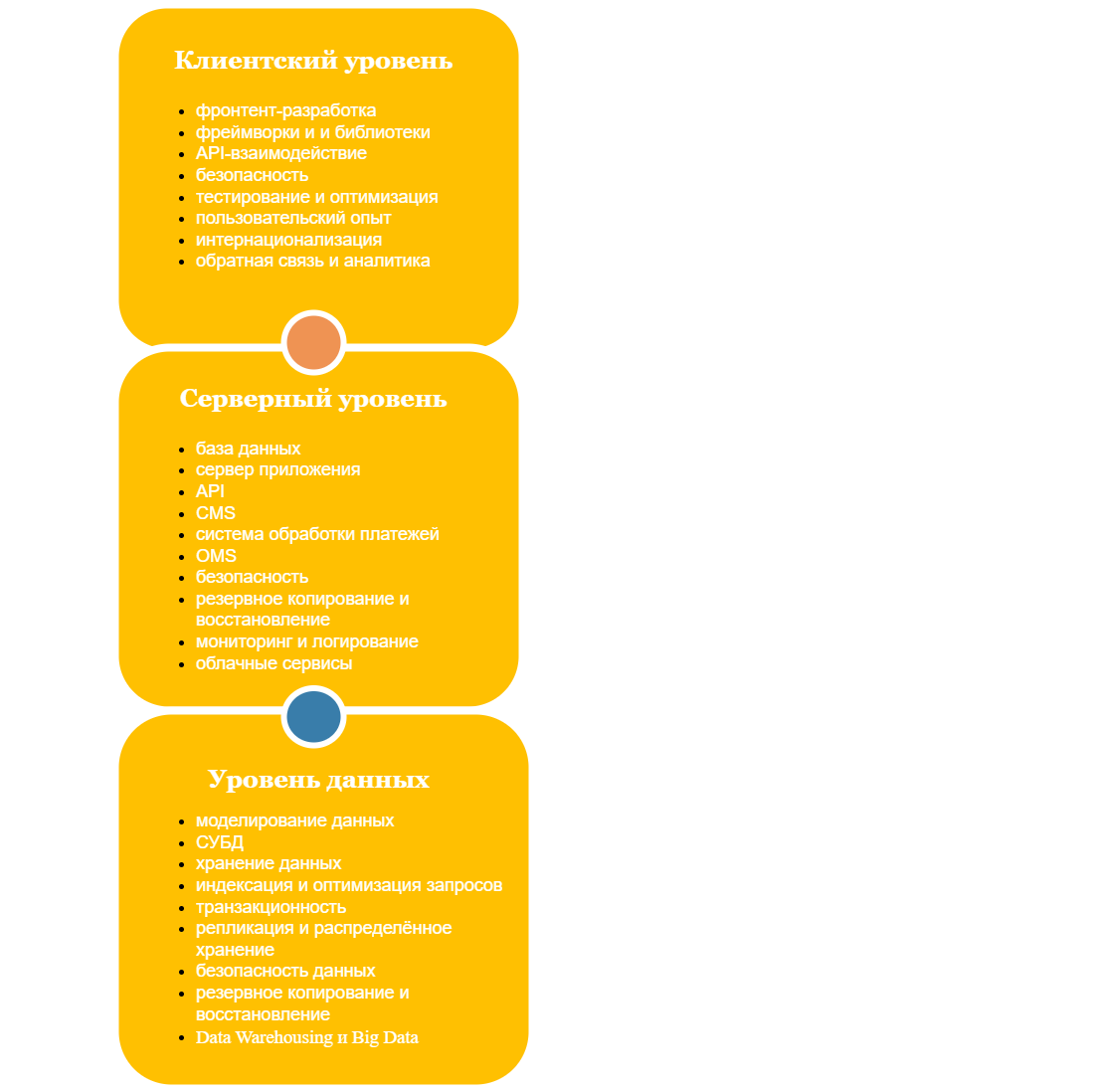
* 1. **Постановки на реализацию системы и её компонентов**

1.3.1. Описание архитектуры решения

1.3.1.1. Описание бизнес-архитектуры

Трёхуровневая клиент-серверная архитектура интернет-магазина — это модель разделения функциональности приложения на три отдельных уровня, каждый из которых выполняет свои задачи. Это обеспечивает гибкость, масштабируемость и удобство в управлении.



Трёхуровневая клиент-серверная архитектура для интернет-магазина включает следующие компоненты:

1. Уровень представления (Клиентский уровень, Presentation Layer) - интерфейс пользователя, через который клиенты взаимодействуют с интернет-магазином; скрипты на стороне клиента, которые обеспечивают динамическое взаимодействие с пользователем и дизайн приложения, корректно отображающийся на различных разрешениях экрана. Он реализуется в виде веб-страниц, Сам уровень отвечает за отображение товаров, корзины покупок, форм оформления/возврата заказа и других пользовательских интерфейсов.

Данный уровень реализуется через веб-фронтенд и приложения, которые предоставляют пользовательский интерфейс для взаимодействия с магазином. Основные аспекты реализации включают:

- фронтенд-разработка: использование технологий веб-разработки, таких как HTML, CSS, Bootstrap, Foundation JavaScript, для создания интерактивного и отзывчивого интерфейса, который корректно отображается на различных размерах экрана;

- фреймворки и библиотеки: применение современных фреймворков (React, Angular, Vue.js) и библиотек для ускорения разработки и повышения качества пользовательского опыта;

- API-взаимодействие: интеграция с серверной частью через API для обмена данными, такими как информация о товарах, заказах и статусах доставки (при помощи REST, GraphQl, JSON, WebSocket, OAuth, CORS, SDK);

- безопасность: реализация мер безопасности для защиты данных пользователей, включая шифрование данных и безопасную аутентификацию (HTTPS, CSRF-токены, WAF);

- тестирование и оптимизация: проведение тестирования пользовательского интерфейса на предмет ошибок и оптимизация производительности для быстрой загрузки страниц (YSlow, Hotjar - для анализа поведения пользователей на сайте, включая тепловые карты, записи сессий и опросы, BrowserStack, Selenium, aXe - автоматического тестирования веб-сайтов на соответствие стандартам доступности, UglifyJS - для минификации JavaScript-кода, TinyPNG - для сжатия изображений без потери качества, Optimizely);

- пользовательский опыт (UX): исследование и проектирование UX для обеспечения логичного и понятного потока действий пользователя, минимизации сложности и улучшения общего впечатления от использования магазина (Abode Trget, Mixpanel);

- интернационализация: поддержка многоязычности и локализации, чтобы обеспечить доступность магазина для пользователей из разных стран (i18next, React Intl, Angular Translate, Navigator.language);

- обратная связь и аналитика: внедрение инструментов для сбора обратной связи от пользователей и аналитики поведения, что позволяет улучшать интерфейс и функциональность магазина (Hotjar, Mixplanet, Intercom, Optimizely).

2. Уровень бизнес-логики (Серверный уровень, Business Logic Layer) - обработка данных, выполнение транзакций, управление пользователями и продуктами; выполняется бизнес-логика, такая как расчёт стоимости заказа, управление инвентарём, регистрация и аутентификация пользователей; шифрование и защита от атак; обработка платежей и выполнение заказов. Уровень служит связующим звеном между клиентским уровнем и уровнем данных.

Серверный уровень в архитектуре решения интернет-магазина обеспечивает основную бэкенд-функциональность и управление данными. Он включает в себя следующие компоненты:

- база данных: центральное хранилище для всех данных магазина, включая информацию о продуктах, пользователях, заказах и транзакциях. Используются различные СУБД, такие как MySQL, PostgreSQL или NoSQL-решения типа MongoDB;

- сервер приложений: выполняет бизнес-логику и обрабатывает запросы от клиентского уровня. Это сервер на базе Node.js, Ruby on Rails, Django;

- API (Application Programming Interface): позволяет клиентскому уровню взаимодействовать с сервером, обмениваясь данными в формате, например, JSON и XML;

- система управления контентом (CMS): упрощает процесс управления контентом сайта, позволяя добавлять и редактировать продукты и другие данные без прямого вмешательства разработчиков (Shopify, Magento, Contentful);

- системы обработки платежей: интегрируются с платежными шлюзами для обработки онлайн-транзакций и управления финансовыми потоками;

- системы управления заказами (OMS): управляют всем процессом заказа, от приема до доставки, включая управление запасами и логистику (Magento, Directus, Zapier);

- безопасность: включает механизмы аутентификации, авторизации, шифрования данных и защиты от вредоносных атак (HTTPS, сетевые брэндмаутеры, API Security, шифрование, MFA, IDS/IPS, WAF);

- резервное копирование и восстановление: обеспечивает сохранность данных и быстрое восстановление системы в случае сбоев (Acronis Cyber Backup, MySQL/PostgreSQL, tar, Ansible/Chef/Puppet);

- мониторинг и логирование: отслеживает работу серверов и фиксирует ошибки для оперативного реагирования на проблемы (Nagios, Zabbix, New Relic, Docker);

- облачные сервисы: используются для гибкости, масштабируемости и оптимизации затрат на инфраструктуру (AWS, Docker Swarm, Amazon RDS).

3. Уровень данных (База данных) - данные интернет-магазина, включая информацию о товарах, заказах, клиентах, настройках магазина и транзакциях; обеспечение создания, чтения, обновления и удаления данных (CRUD), быстрый доступ и их надёжное хранение; гарантия сохранности данных и возможности восстановления после сбое.

Уровень данных в архитектуре решения интернет-магазина — фундаментальный компонент, который управляет хранением, извлечением и обработкой всех данных, связанных с работой магазина. Основные аспекты реализации уровня данных:

- моделирование данных: прежде всего, необходимо разработать структуру базы данных, определяющую сущности, их атрибуты и связи между ними. Это реализовавается с использованием методологий нормализации для реляционных баз данных или документо-ориентированных моделей для NoSQL-решений;

- системы управления базами данных (СУБД): реляционные СУБД, такие как MySQL или PostgreSQL, хорошо подходят для транзакционных систем с жёсткой структурой данных. NoSQL-решения, например MongoDB или Cassandra, предпочтительны для гибких схем данных и больших объёмов неструктурированных данных;

- хранение данных: данные могут храниться на локальных серверах или в облачных хранилищах. Облачные сервисы, такие как Amazon S3 или Google Cloud Storage, предлагают гибкость и масштабируемость;

- индексация и оптимизация запросов: для обеспечения быстрого доступа к данным используются индексы. Оптимизация запросов включает в себя анализ и улучшение производительности запросов к базе данных (MySQL/MariaDB, EXPLAIN/ANALYZE);

- транзакционность: важно обеспечить целостность данных при выполнении операций, которые могут включать несколько шагов. Системы управления транзакциями гарантируют, что все операции в рамках одной транзакции либо полностью выполняются, либо откатываются (Two-phase commit (2PC), XA Transactions);

- репликация и распределённое хранение: для повышения доступности и отказоустойчивости данных используется репликация. Распределённые системы хранения позволяют распределять данные по нескольким узлам (Master-Slave Replication, HDFS, MongoDB, Apache Kafka, Kubernetes);

- безопасность данных: включает в себя шифрование данных в покое и в передаче, а также механизмы контроля доступа и аутентификации пользователей (Hyperledger Fabri, HTTPS, сетевые брэндмаутеры, API Security, шифрование);

- резервное копирование и восстановление: регулярное создание резервных копий данных и стратегии их восстановления в случае сбоев или потери данных (Acronis Cyber Backup, MySQL/PostgreSQL, tar, Ansible/Chef/Puppet);

- Data Warehousing и Big Data: для аналитических целей могут использоваться специализированные решения для хранения больших объёмов данных и их обработки (Apache Spark, Apache Kafka).

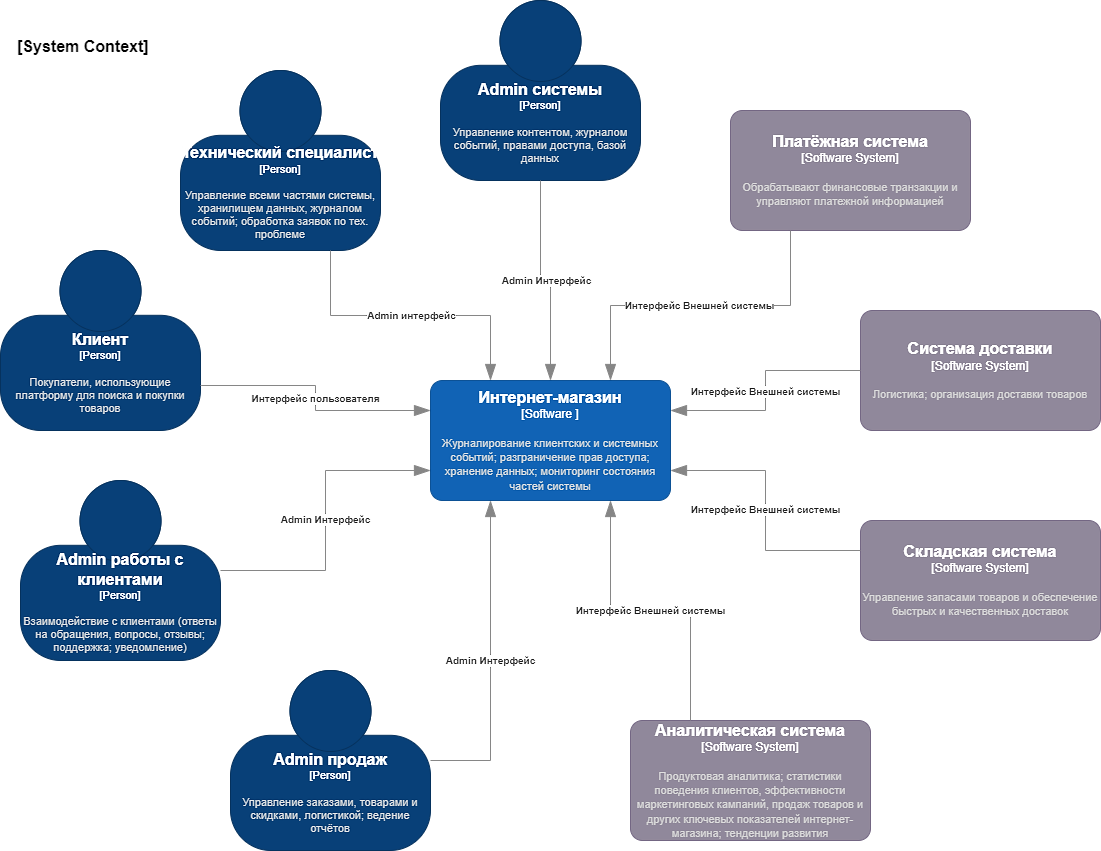
Данная архитектура позволяет разрабатывать и обслуживать каждый уровень независимо, что упрощает обновление системы, расширение функционала и интеграцию с другими сервисами и приложениями. Также, она способствует повышению безопасности и производительности интернет-магазина.

1.3.1.2. Описание программной архитектуры (C4 Diagram - контекстный уровень

и уровень контейнеров)

C4-диаграмма для архитектуры интернет-магазина визуализирует структуру системы, ее компоненты, связи между ними и взаимодействие с внешними системами.

Уровень 1. Контекст - Context Level - общее представление системы, ее интерфейсы и связи с внешним окружением. Интернет-магазин представляется как отдельная система, взаимодействующая с пользователями, поставщиками и внешними сервисами.



Основные участники и их взаимодействие с системой интернет-магазина:

- клиенты: пользователи, которые взаимодействуют с интернет-магазином для просмотра и покупки товаров;

- администратор продаж: отвечает за управление продажами, акциями и логистикой;

- администратор по работе с клиентами: занимается обработкой запросов клиентов, возвратами, обращениями и уведомлением;

- администратор системы: отвечает за поддержку и обслуживание информационной системы интернет-магазина;

- технический специалист: работает над поддержкой и развитием технической инфраструктуры, включая серверы и сетевые решения.

Взаимодействия между интернет-магазином и внешними системами:

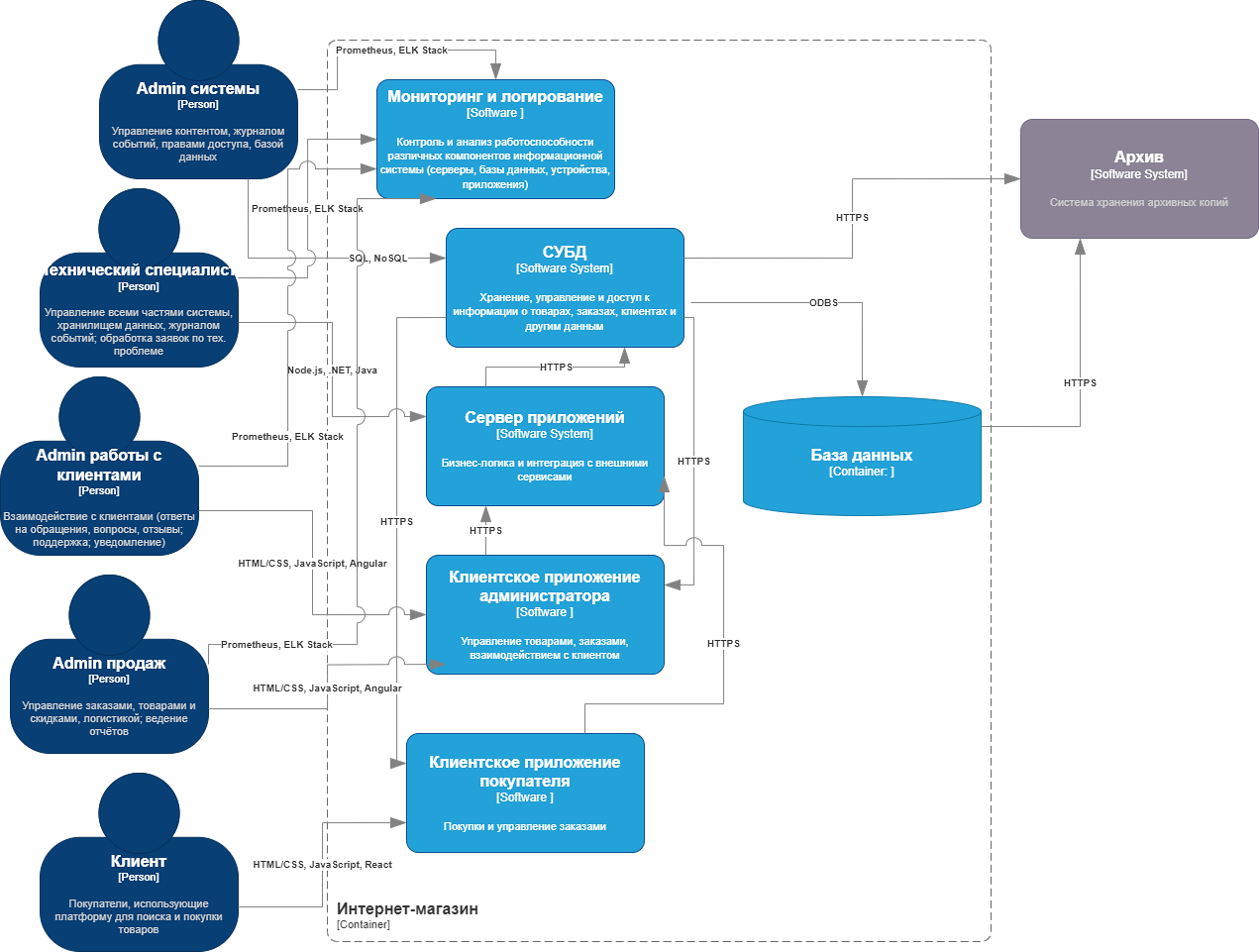
- платежная система: обрабатывает транзакции клиентов и передает данные о платежах в интернет-магазин;

- складская система: управляет запасами товаров и информирует интернет-магазин о наличии и движении товаров;

- система доставки заказов: отвечает за логистику и доставку товаров клиентам;

- аналитическая система: анализирует данные о продажах, поведении клиентов и эффективности бизнес-процессов.

Уровень 2. Контейнеры - Container Level - на этом уровне можно увидеть разбиение системы на контейнеры, внутри которых находятся отдельные компоненты. Например, контейнеры могут представлять пользовательский интерфейс, бизнес-логику, базу данных и внешние API. Каждый контейнер имеет свои функции и связи с другими контейнерами.

1. Веб-приложение (для клиентов):

- описание: пользовательский интерфейс для клиентов, позволяющий просматривать и покупать товары.

- технологии: HTML/CSS, JavaScript, React.

- взаимодействие: Клиенты используют веб-приложение для выполнения покупок и управления своими заказами.

2. Административный интерфейс (для администраторов продаж и по работе с клиентами):

- описание: интерфейс для управления каталогом товаров, заказами и взаимодействия с клиентами;

- технологии: HTML/CSS, JavaScript, Angular;

- взаимодействие: администраторы продаж и по работе с клиентами используют этот интерфейс для управления продажами и обслуживания клиентов.

3. Система управления базами данных (для администратора системы):

- описание: централизованное хранилище данных, содержащее информацию о товарах, заказах и клиентах;

- технологии: SQL, NoSQL;

- взаимодействие: администратор системы использует систему управления для настройки и обслуживания баз данных.

4. Сервер приложений (для технического специалиста):

- описание: сервер, обрабатывающий бизнес-логику и интеграцию с внешними сервисами;

- технологии: Node.js, .NET, Java;

- взаимодействие: технический специалист отвечает за поддержку и развитие сервера приложений.

5. Система мониторинга и логирования (общая для всех ролей):

- описание: служба для отслеживания работы системы и регистрации событий.

- технологии: Prometheus, ELK Stack.

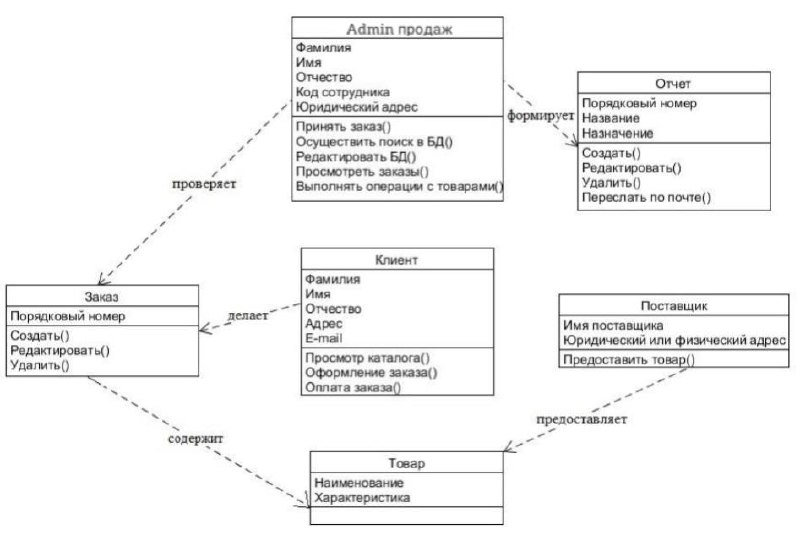
- взаимодействие: все роли используют эту систему для мониторинга состояния интернет-магазина и анализа логов.

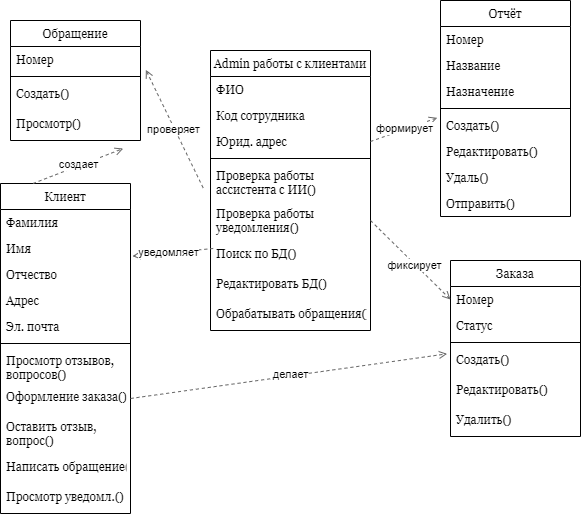
1.3.1.3. Описание архитектуры данных (Концептуальное - Class Diagram,

логическое и физическое - ER-diagram)

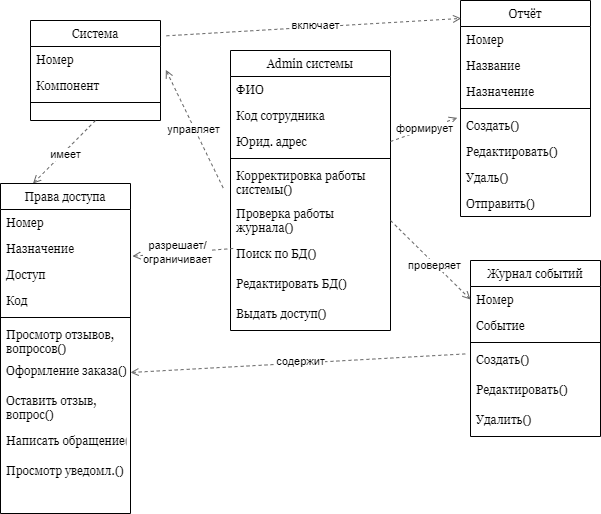
Концептуальная диаграмма (Class Diagram) интернет-магазина разделена на модули для большей детализации взаимосвязей.

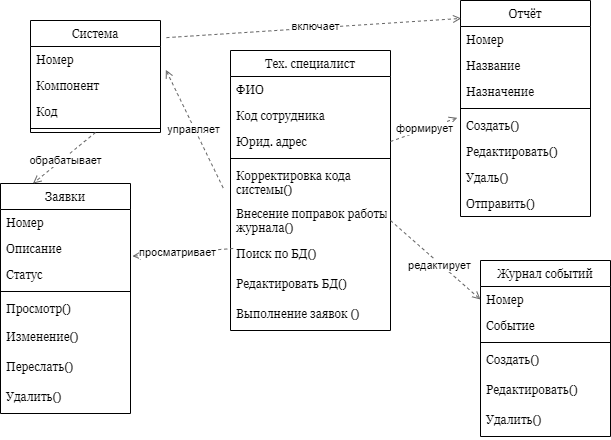
Спецификация классов интернет-магазина:

Модуль заказа: администратор продаж – класс объектов-пользователей, всячески взаимодействует с товарами, добавляя, редактируя и удаляя их; заказ – класс объектов-заказов от клиентов интернет-магазина; клиент – класс объектов-пользователей, зарегистрированные пользователи, которые потенциально могут купить товар; поставщик – класс объектов-поставщиков, предоставляют товар по приемлемым ценам для закупки; товар – класс объектов-ассортимента, товары которые продаются через магазин; отчет – класс объектов-документов, использующихся для создания отчетов по продажам; 

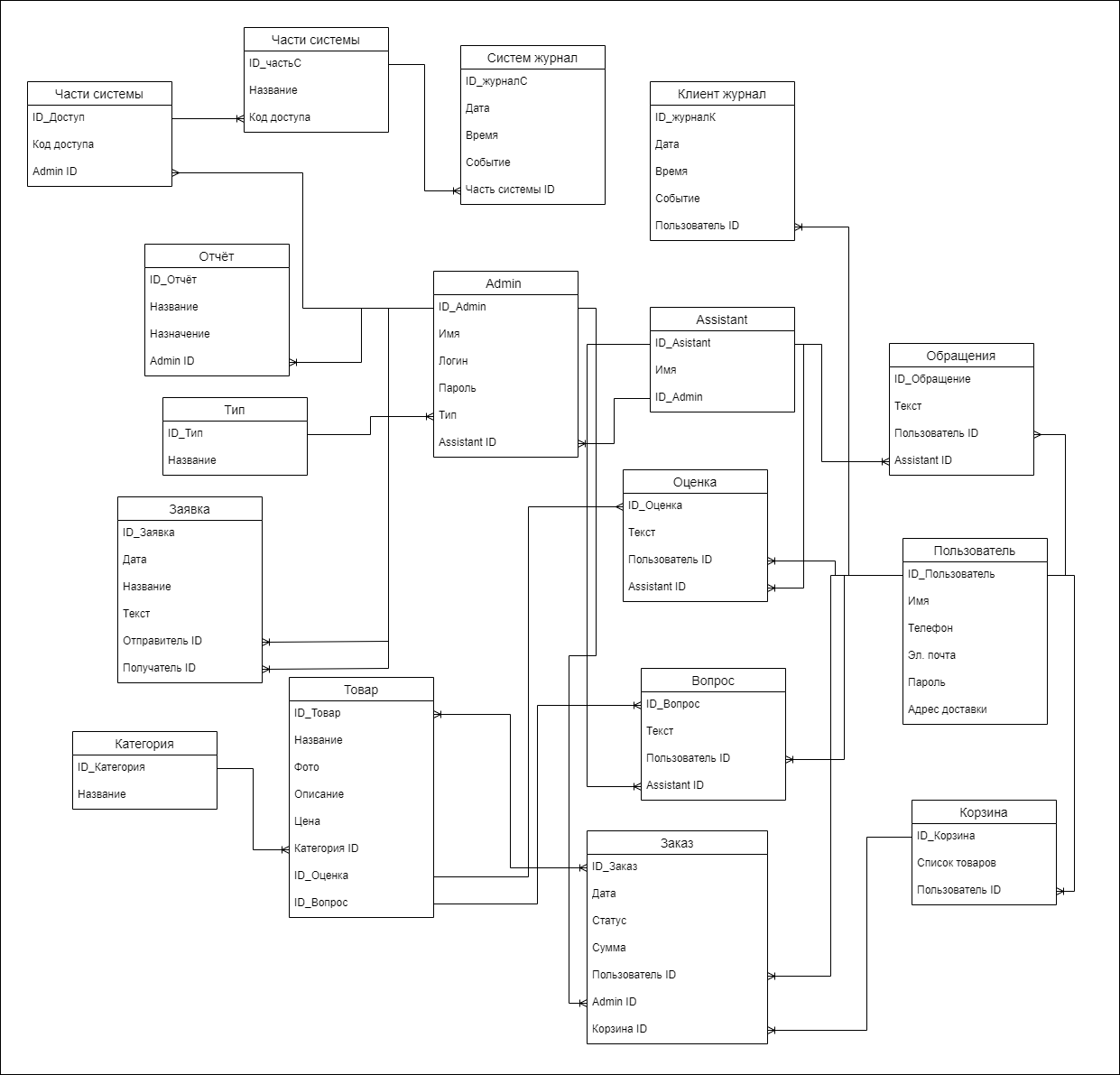
Модуль обратной связи: администратор по работе с клиентами – класс взаимодействует с обращениями, отзывами, вопросами, учётными записями покупателей и их уведомлением; обращения – класс объектов, включающий проблемы, вопросы, отзывы клиентов; клиент – класс объектов-пользователей, зарегистрированные пользователи, которые потенциально могут купить товар; заказ – класс объектов-заказов от клиентов интернет-магазина; отчет – класс объектов-документов, использующихся для создания отчетов по выполнению работы с клиентами; 

Модуль системы: администратор системы – класс взаимодействует системными операциями, транзакциями и контентом; права доступа – класс, который содержит данные для доступа к определённым частям системы; журнал событий – класс записей, характеризующих поведение клиентов и системы в целом; система – класс частей общей системы, которые состоят из компонентов; отчет – класс объектов-документов, использующихся для создания отчетов по выполнению работы системы;



Модуль технической работы: администратор технической работы – класс участвует в работе системы, управляет кодом и её структурой; заявки – класс объектов, которые описывают проблему в функционировании определённых частей системы; журнал событий – класс записей, характеризующих поведение клиентов и системы в целом; отчет – класс объектов-документов, использующихся для создания отчетов по устранению проблем и статистике работы элементов системы; 

Логическая диаграмма (ER) для интернет-магазина отображает сущности, их атрибуты и связи между ними на концептуальном уровне; фокусируется на бизнес-правилах и отношениях, не углубляясь в детали реализации; помогает понять, как данные связаны в системе, и служит основой для создания физической модели.



Физическая диаграмма (ER) для интернет-магазина представляет реальную структуру базы данных, включая таблицы, поля, типы данных, первичные ключи; описывает, как логическая модель будет реализована в конкретной системе управления базами данных (СУБД).

