СОДЕРЖАНИЕ

Роли

Таблица 1 – Распределение ролей в команде проекта.

№	Исполнитель	Роли	Описание
1	Артем	Менеджер проекта, релиз-менеджер	Отвечает за управление проектом, за то, что ожидания заинтересованных сторон будут верно поняты и проведены через проект а так же успешно внедрены в инфраструктуру заказчика
2	Илья	Разработчик, архитектор	Отвечает за систему в целом, за проектирование и осуществление реализации, вырабатывает архитектуру решения, включая сервисы, технологии и стандарты, которые будут использованы в ходе работы над решением.
3	Аскар	Разработчик, архитектор	Отвечает за систему в целом, за проектирование и осуществление реализации, вырабатывает архитектуру решения, включая сервисы, технологии и стандарты, которые будут использованы в ходе работы над решением.
4	Шепелев Всеволод	Бизнес-аналитик, тестер	Отвечает за качество решения с точки зрения заказчика и будущих пользователей, за понимание потребностей пользователей и их надлежащую реализацию в решении, за понимание того как, и успешное получение бизнес-отдачи от внедрения разрабатываемого решения, которое в результате сможет получить заказчик.

Первичный анализ системы

Система представляет собой веб-сайт, на котором реализована система интернет-магазина, а именно – создание учетной записи, просмотр доступных в базе данных товаров и полный процесс оформления заказа.

Сущности – это абстракции, являющиеся основными элементами модели. Между сущностями можно выстраивать взаимодействия за счет отношения между атрибутами класса.

Атрибут – именованная единица определенного типа, содержащаяся в классе. Атрибут используется для представления информации о сущности. Каждый объект класса имеет уникальный набор атрибутов.

Набор сущностей в нашей базе данных:

- users;
- orders;
- order_items;
- products;
- categories.

Набор связей между сущностями:

- users : orders – 1 : M;
- orders : order_items – 1 : M;
- products : order_items – 1 : M;
- categories : products – 1 : M.

Набор атрибутов сущности «users»:

- **id (PK)**
- email
- password
- name
- surname
- phone_number
- address

Набор атрибутов сущности «orders»:

- **id (PK)**
- user_id (FK)
- status
- created_at

Набор атрибутов сущности «order_items»:

- order_id (FK)
- product_id (FK)
- quanity

Набор атрибутов сущности «products»:

- **id (PK)**
- name
- description
- category_id (FK)
- quanity
- status
- price
- photo

Набор атрибутов сущности «categories»:

- **id (PK)**
- name
- description

Сущности видов подтип/супертип:

Супертип – тип сущности, в котором реализуются общие атрибуты для всех сущностей, которые используют супертип.

Подтип – сущность, являющаяся членом супертипа, которая заимствует его атрибуты, но изолирована в нем, выполняет отдельную роль.

В нашей модели отсутствуют сущности типа супертип и, как следствие, подтипы.

Модель БД в нотации IDEF1X

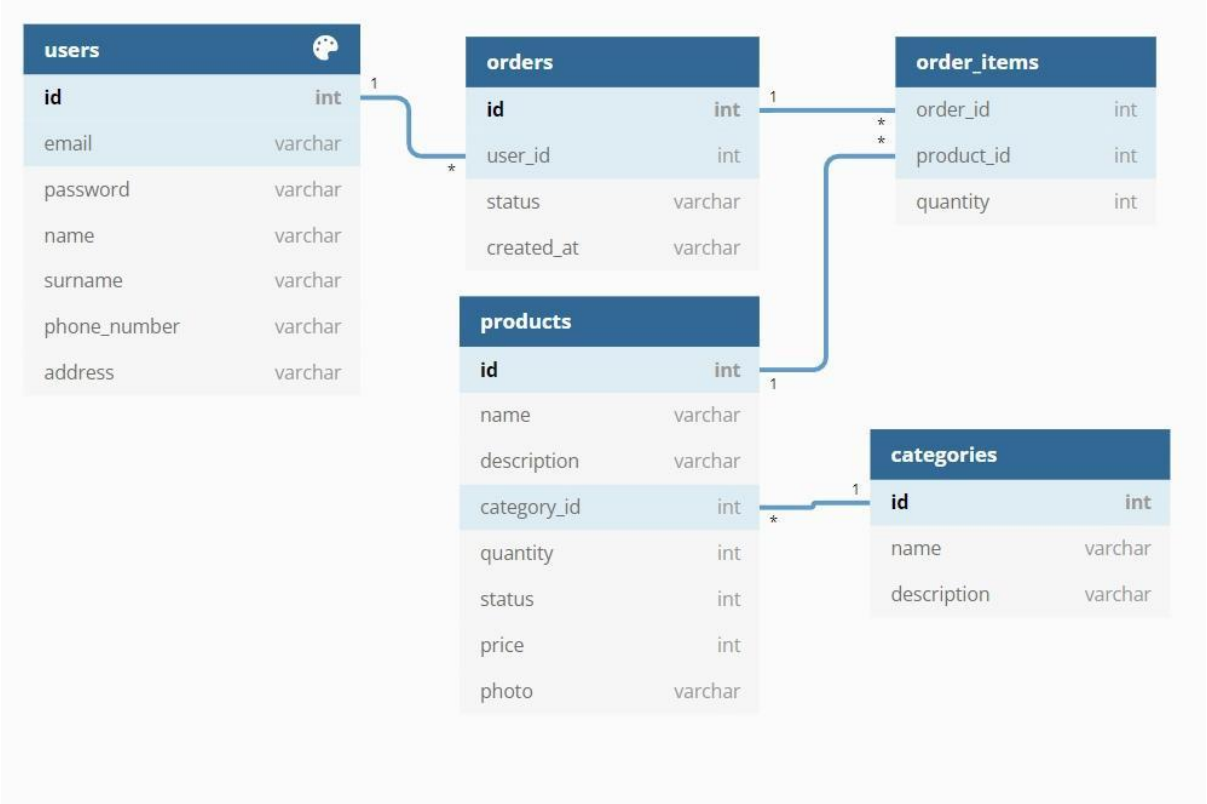

Рисунок 1 – Схема БД в нотации IDEF1X

Методология моделирования IDEF1X – расширение стандарта IDEF1, используется для описания взаимодействия информации в системе. Основа модели – концепция «сущность-связь», которая позволяет определить конкретные взаимодействия и взаимосвязи между сущностями. Модель используется для идентификации сущностей, атрибутов и связей между данными. Может использоваться для автоматизации системы.

В нашей курсовой работе мы используем модель в нотации IDEF1X для определения структуры базы данных и взаимодействия с ней.

Модель системы в нотации IDEF0

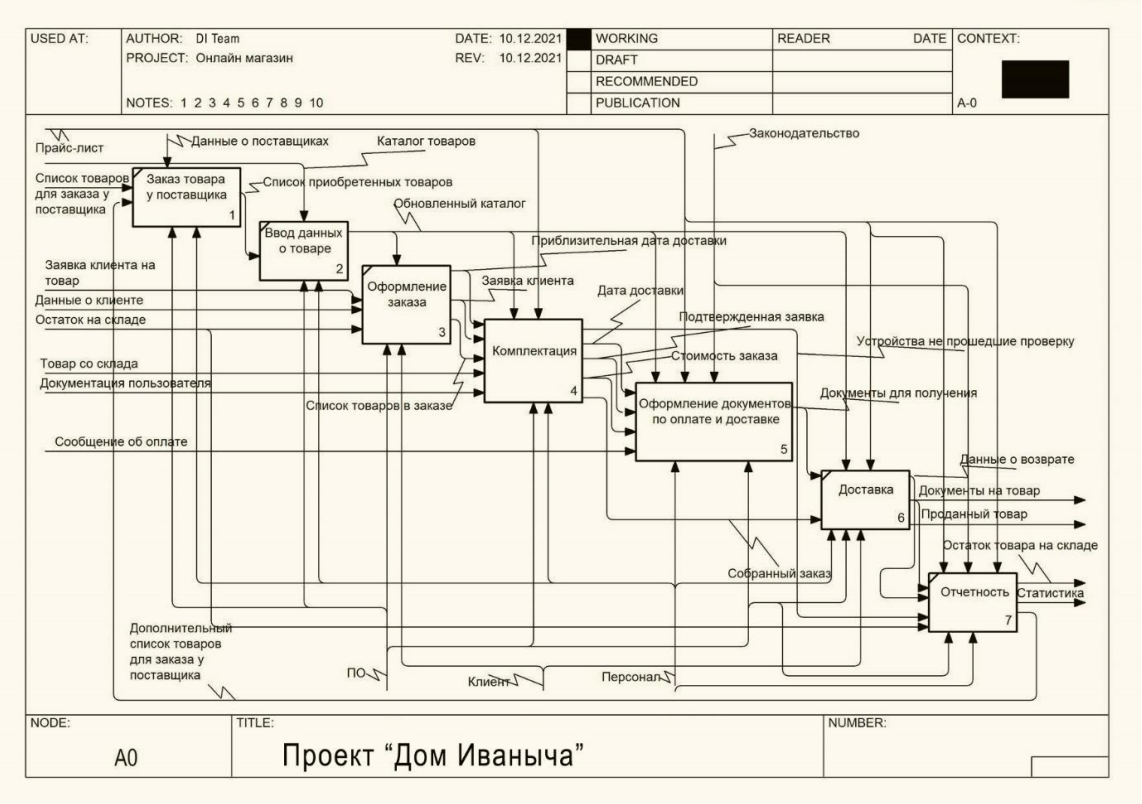

Рисунок 2 – Полная теоретическая система проекта в нотации IDEF0

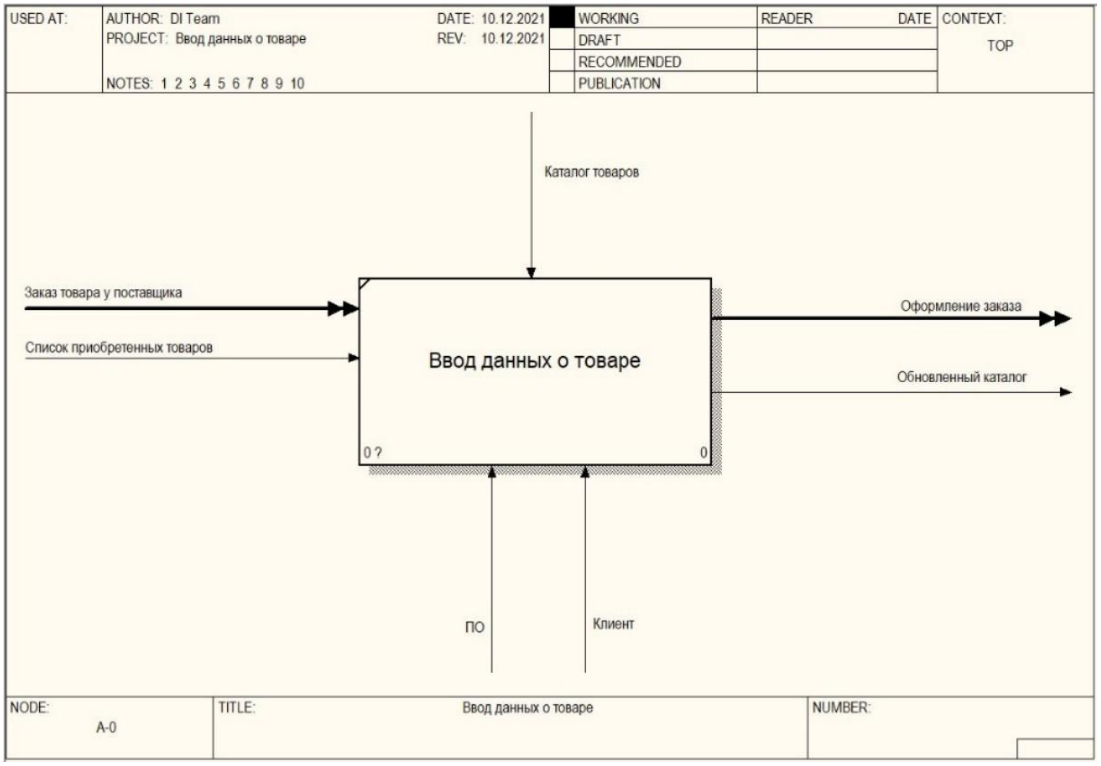

Рисунок 3 – Отдельный сценарий системы на примере ввода данных в нотации IDEF0

Методология моделирования IDEF0 – нотация графического моделирования, которая используется для описания функционирования модели. Она отражает структуру и функции всей системы и отдельных компонентов в частности, потоки и взаимодействия информации и материальных объектов. В данной модели рассматриваются полные логические взаимодействия, а не временная последовательность.

Стандарт IDEF0 показывает систему как набор модулей. Существует два правила реализации модели: функции идут сверху вниз по важности; каждая сторона отвечает за свое событие:

- Левая сторона – стрелка входа;
- Правая сторона – стрелка выхода;
- Верхняя сторона – стрелка управления;
- Нижняя сторона – стрелка механизма.

В нашей курсовой работе мы используем модель в нотации IDEF0 для анализа системы «Интернет-магазин» в общем (теоретически реализуемом) случае и в частных моментах, для понимания того, как реализовать взаимодействия между структурами, объектами и данными.

Диаграмма потоков данных в нотации Гейна – Сарсона

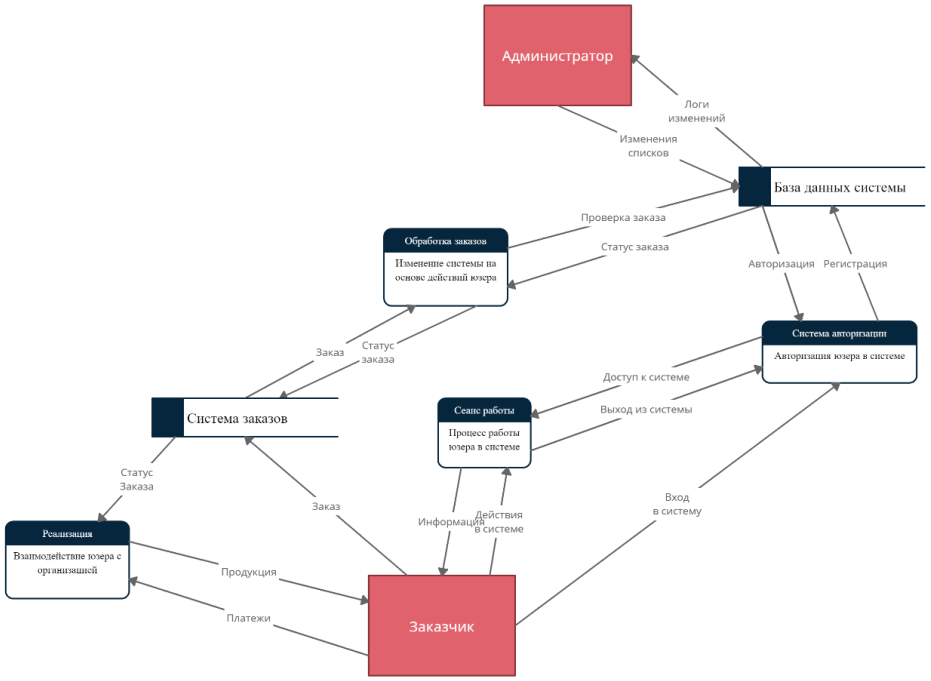

Рисунок 4 – Диаграмма потоков данных в нотации Гейна – Сарсона

Нотация Гейна – Сарсона – DFD (диаграммы потоков данных), которые строятся из элементов:

- функция (действие в системе);
- поток данных (объект, над которым происходит действие);
- хранилище данных (структура хранения данных);
- внешняя сущность (внешний по отношению к системе объект).

Функции, хранилища и сущности связываются дугами – потоками данных. Потоки могут соединяться и разделяться. Есть правила построения диаграммы:

- Функции преобразуют входящие данные в выходящие;
- Хранилища не изменяют данные, только хранят;
- Преобразование потоков во внешних сущностях игнорируются.

В нашей диаграмме есть две внешние сущности: Администратор и Заказчик. Они находятся за пределами системы, в то время как организация (в роли склада, платежной системы, взаимодействия с заказчиком и т.п.) отмечена на модели как функция реализации, хоть и может быть вынесена отдельно. Четыре подсистемы выполняют функции:

авторизации, сеанса работы, реализации и обработки заказов. Есть две БД: полная БД системы и промежуточная система с заказами. Все остальное – потоки данных системы.

Основное взаимодействие модели – обмен данными по поводу товаров и заказов между пользователем и всеми функциями системы.

Диаграмма прецедентов

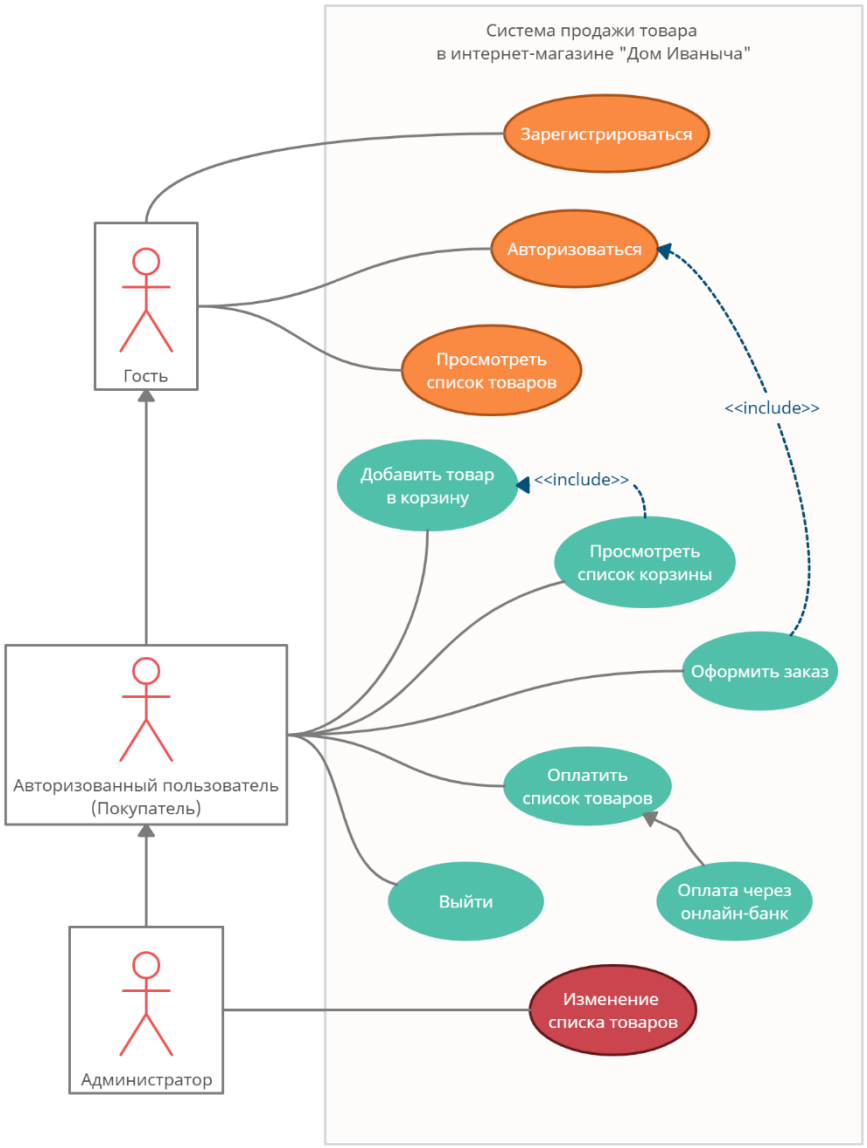

Рисунок 5 – Диаграмма прецедентов интернет-магазина

Диаграмма прецедентов – диаграмма, показывающая актеров, прецеденты и отношения между ними. Каждый актер в диаграмме выполняет прецеденты, некоторые прецеденты включают в себя другие. В терминах UML мы также можем использовать включения (includes) и исходы (extends).

Актер – некая роль в диаграмме, которую пользователи играют по отношению к системе.

В нашей работе используется диаграмма прецедентов для описания функционала и взаимосвязей между функциями и актерами. Сложные шаги в прецеденте можно представить другими прецедентами.

Также в нашей версии модели есть три актера – Гость, Покупатель (юзер) и Администратор (супер-юзер). Взаимодействия между ними происходят, в основном, через прецеденты – Администратор отвечает за управление системой, может вносить правки в БД и любые функции; Гость может только видеть функционал магазина; Покупатель - взаимодействовать с системой на определенных условиях.

Диаграмма классов

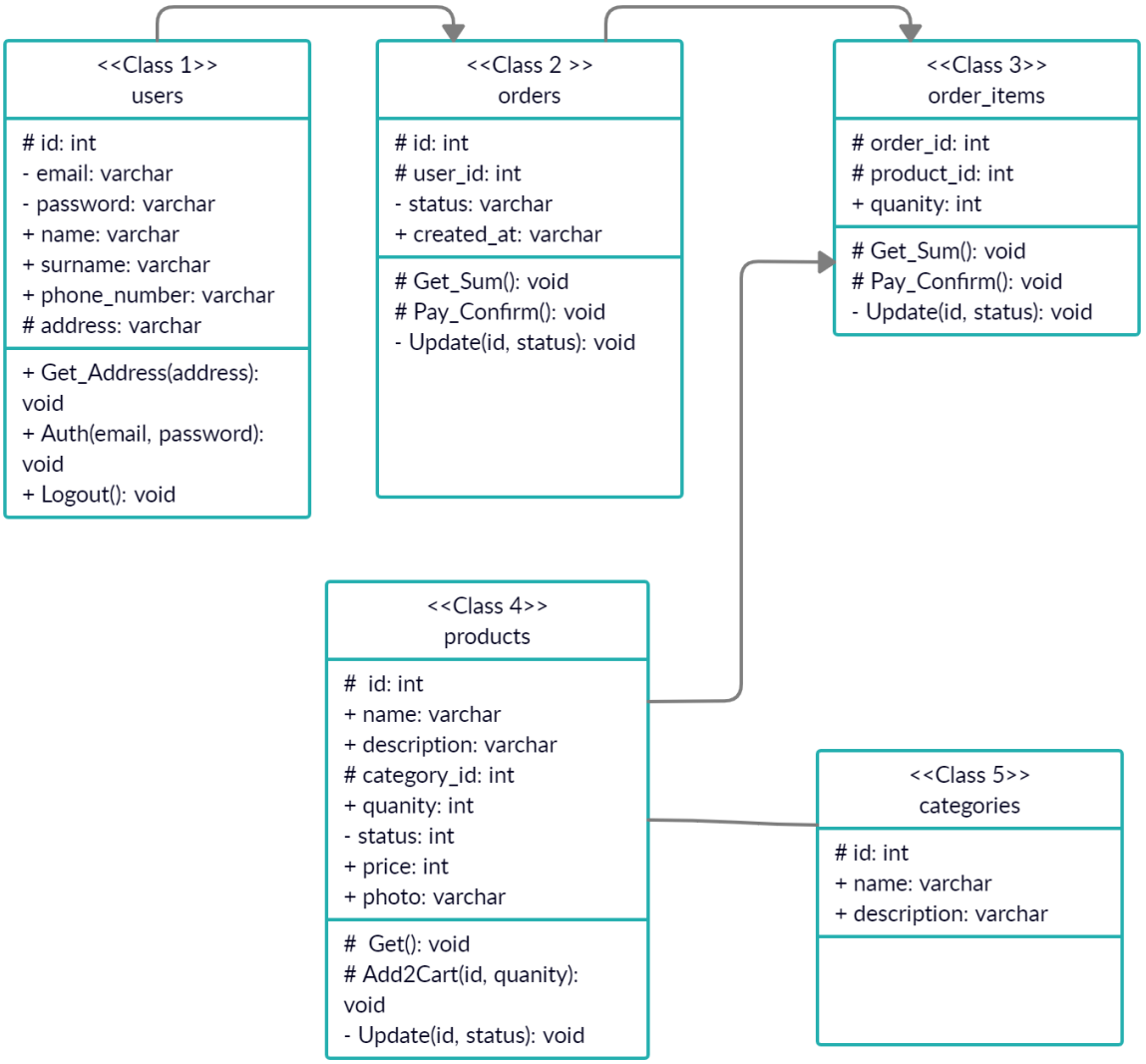

Рисунок 6 – Диаграмма классов интернет-магазина

Диаграмма классов – диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их кооперации, атрибутов, методов, интерфейсов и взаимодействий.

Диаграмма классов является ключевым элементом в объектно-ориентированном моделировании. На диаграмме присутствуют классы в рамках и содержат три компонента:

- Имя класса – пишутся по центру полужирным шрифтом, начинаются с заглавной буквы, если класс абстрактный – просто пишется полужирным шрифтом;
- Атрибуты класса – выровнены по левому краю, со строчной буквы;
- Методы класса - выровнены по левому краю, со строчной буквы.

Диаграмма последовательности

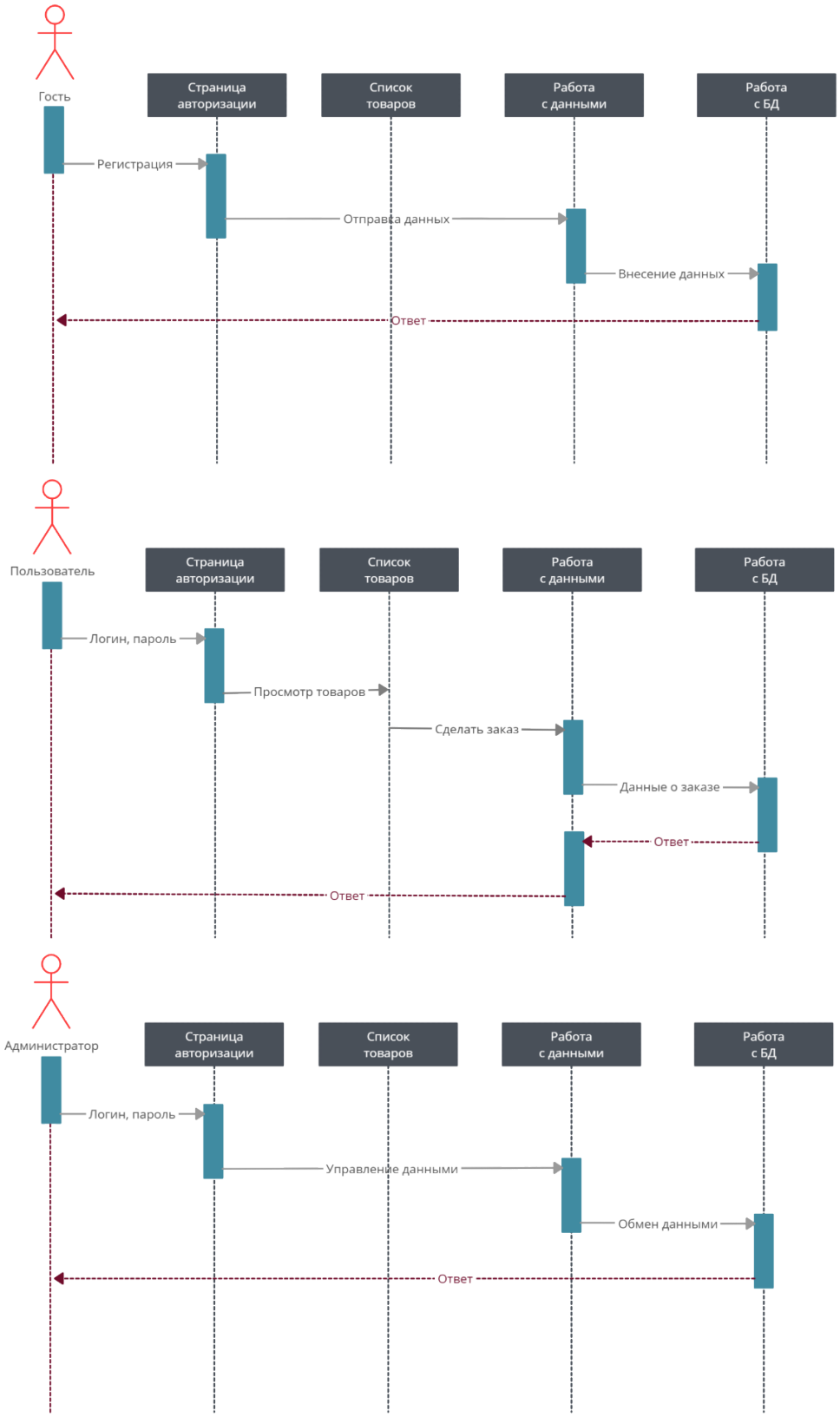

Рисунок 7 – Диаграмма последовательностей интернет-магазина

Диаграмма последовательности используются для уточнения и расширения диаграмм прецедентов, более детального описания логики сценариев использования.

Диаграмма содержит объекты, которые взаимодействуют в рамках заданного сценария. В сценарии они обмениваются данными и ответами.

Объекты обозначаются прямоугольниками.

Сообщения – линиями со стрелками.

Возвращаемые результаты - пунктирными линиями со стрелками.

Прямоугольники на вертикальных линиях – «время жизни».

В нашей курсовой работе мы используем диаграмму последовательности для анализа взаимодействий объектов в системе, чтобы в дальнейшем реализация была структурированной.

Диаграмма кооперацій

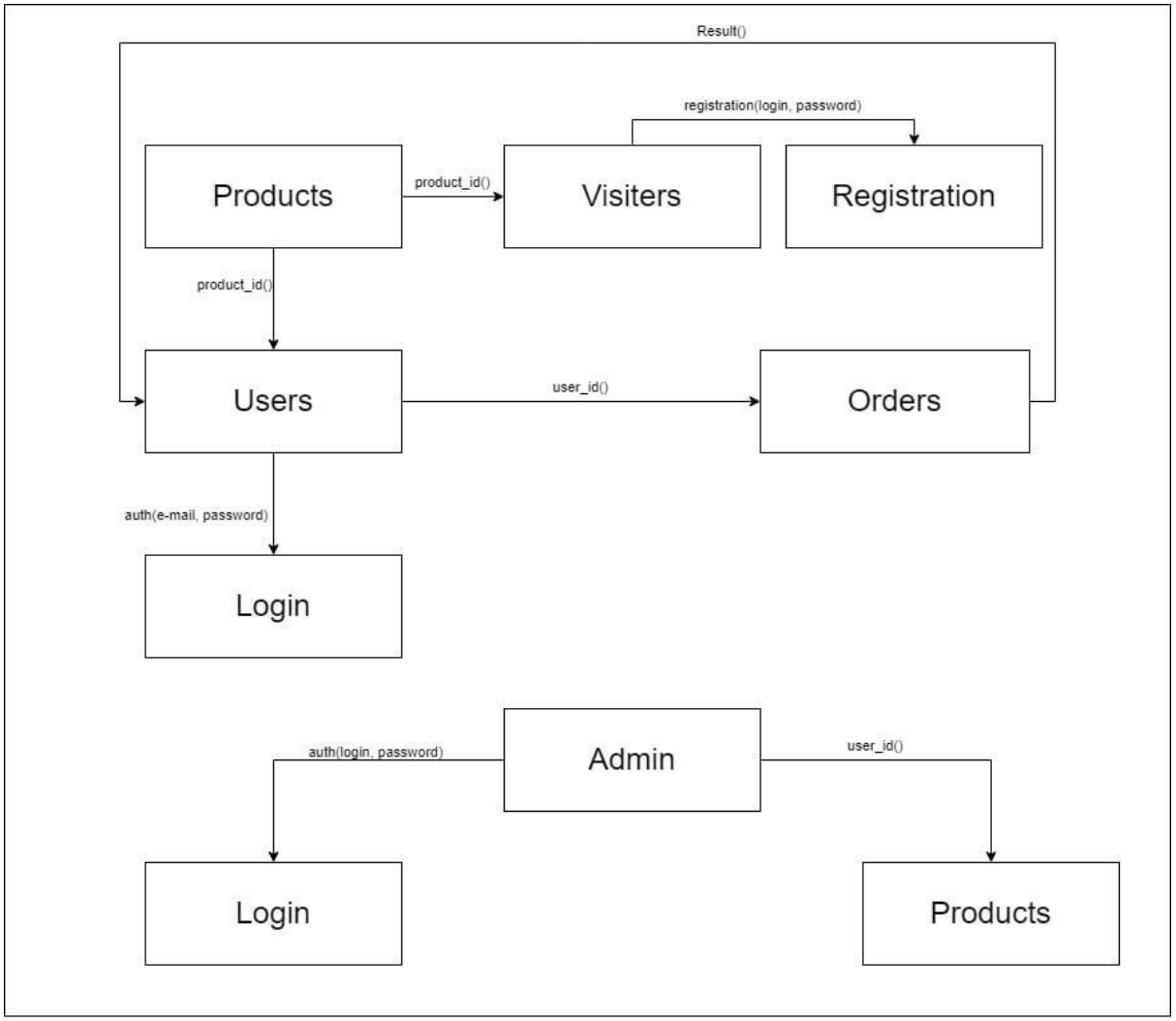

Рисунок 8 – Диаграмма кооперацій интернет-магазина

Диаграмма кооперацій – диаграмма, на которой изображаются взаимодействия между частями структуры или ролями кооперацій. На диаграмме указываются отношения между объектами, в то время как само время в качестве измерения не используется.

На диаграмме, прежде всего, указываются сами объекты и, возможно, их атрибуты. Далее в виде соединительных линий указываются ассоциации между объектами. Дополнительно могут быть указаны динамические связи в качестве потоков сообщений.

В нашей модели диаграмма последовательности используется для обозначения связей и ассоциаций, необходимых для работы системы.

Диаграмма деятельности

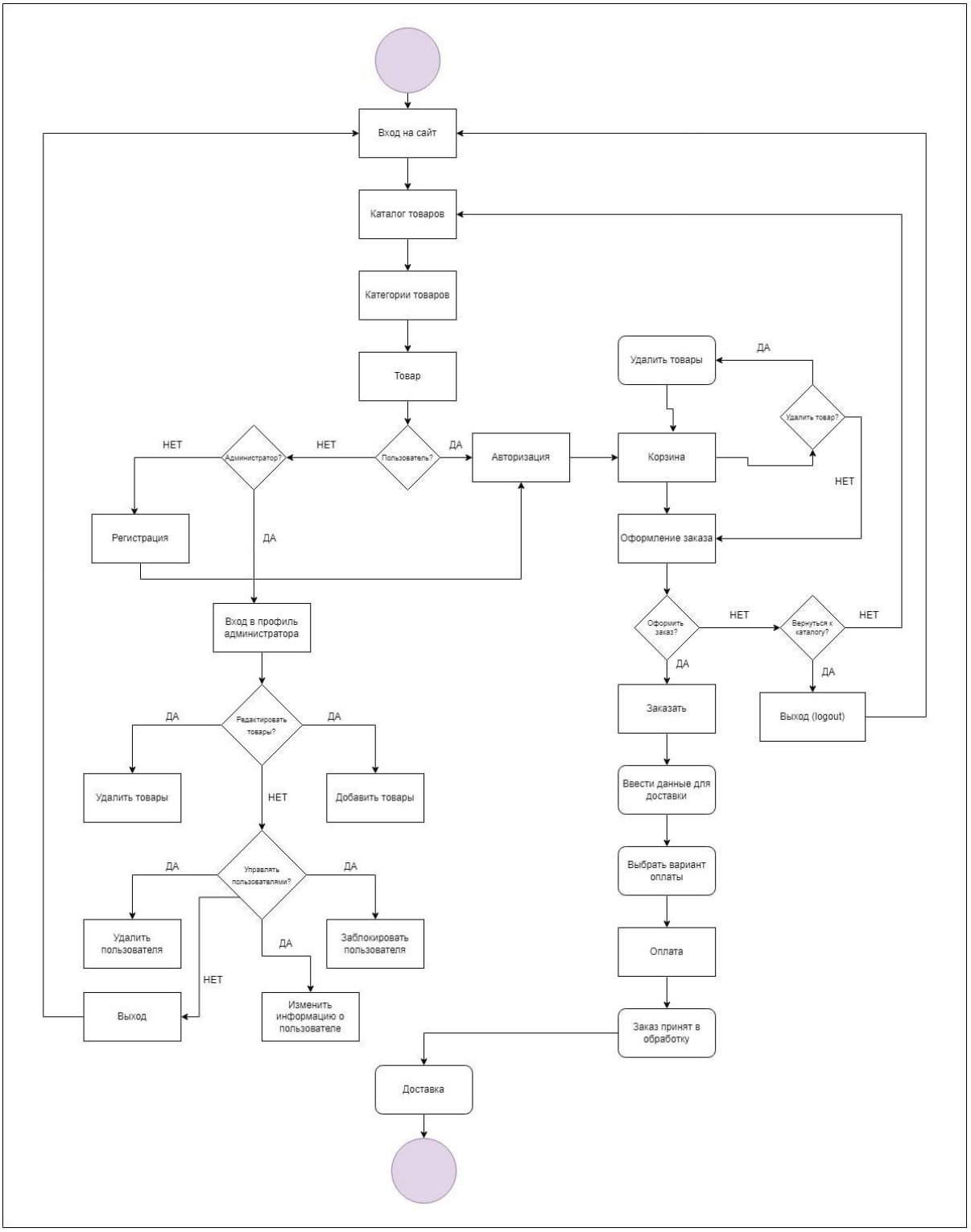

Рисунок 9 – Диаграмма деятельности интернет-магазина

Диаграмма деятельности – технология, описывающая логику процедур, бизнес-процессы и потоки работ. Основная идея диаграммы в том, чтобы поддерживать параллельные процессы системы.

В нашей работе диаграмма деятельности используется по своему прямому назначению – описание логики процедур, процессов и потоков проекта.

Диаграмма развертывания системы

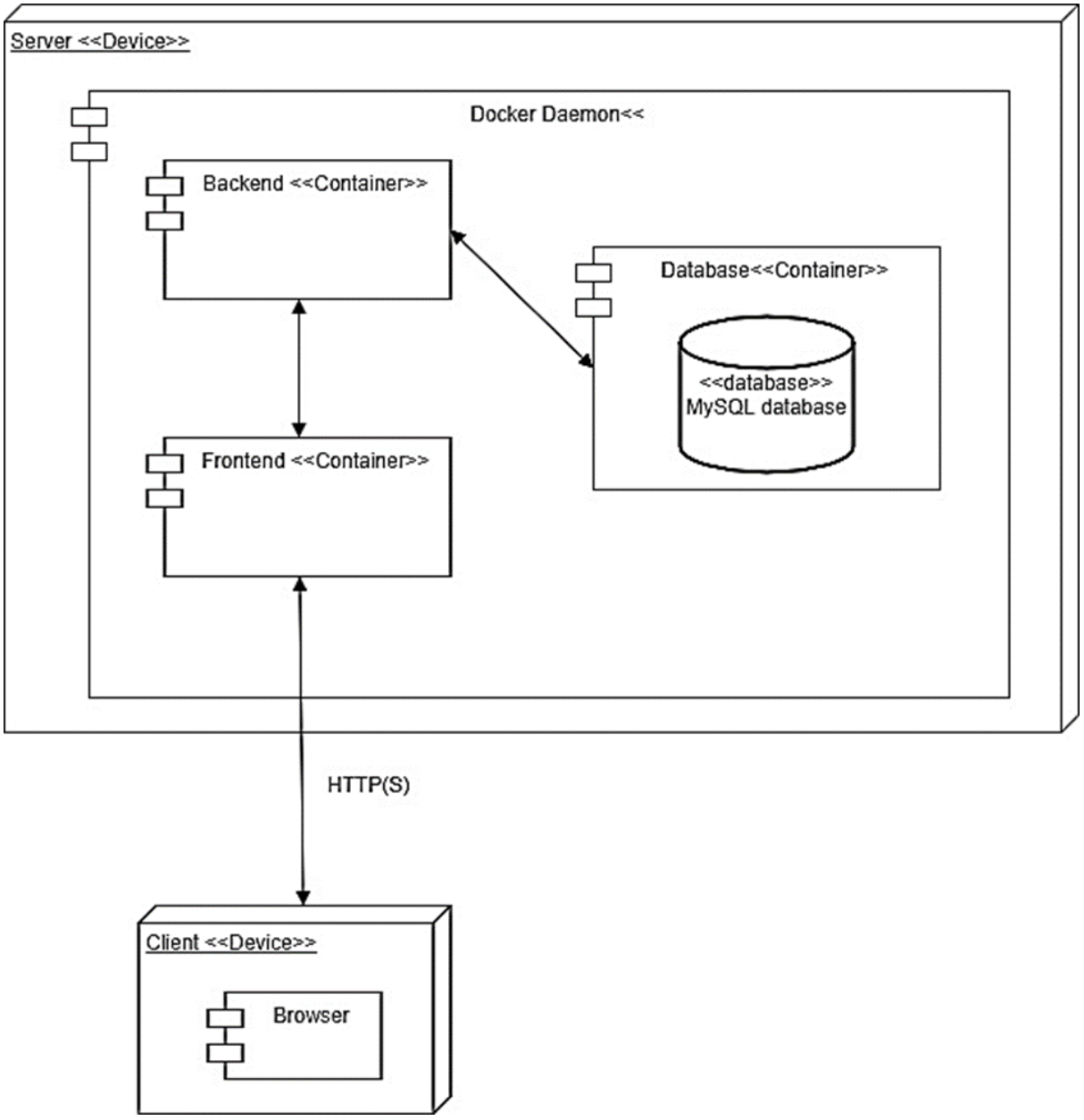

Рисунок 10 – Диаграмма развертывания интернет-магазина

Диаграмма развертывания – тип диаграммы, которая определяет физическое оборудование, отвечающее за запуск и работоспособность системы. Она обозначает способы развертывания ПО на оборудовании, а также отображает программные части системы на исполняющие устройства. В общем случае – диаграмма моделирует распределение ПО по физическим узлам.

В нашей системе диаграмма описывает модель клиент-серверного взаимодействия для итогового веб-приложения.

БД

В нашей курсовой работе в качестве системы управления БД используется СУБД MySQL, т.к. мы посчитали её более приемлемой к применению с нашими инструментами разработки.

У неё есть ряд преимуществ:

- Бесплатна в использовании
- Гибкая система поддержки форматов (чисел, времени, строк и др.)
- Быстрая работа

Для написания бэкэнда был выбран компилируемый многопоточный язык программирования Golang, который легко интегрируется со средой MySQL.

Для написания фронтэнда был выбран JavaScript-фреймворк Vue.js для простоты создания пользовательского интерфейса.

В нашем проекте мы использовали SQL запросы:

- Создание таблиц (CREATE TABLE)
- Простой выбор данных (SELECT)
- Выбор данных с условиями (MAX(), MIN() и др.)
- Изменение таблиц для задания правил/атрибутов/ограничений (ALTER TABLE)
- Создание уникальных индексов (CREATE UNIQUE INDEX)