**2**. **Проектирование системы**

**2.1 Техническое задание проекта**

Цель проекта: веб-платформа, на которой пользователи могут заказывать и продавать товары, выполненные по индивидуальному проекту. Проект охватывает сбор и описание требований, проектирование архитектуры, модели данных, пользовательских сценариев и интерфейсов [37].

Целевая аудитория: люди, стремящиеся к уникальности – молодёжь, творческие личности, энтузиасты кастомной продукции. Продавцы – частные мастера и небольшие бренды.

Веб-платформа кастомных товаров предоставляет:

* + витрину готовых изделий;
  + механизм заказа индивидуальных товаров;
  + прямую коммуникацию с авторами;
  + продвижение товаров через платные услуги;
  + безопасную сделку с заморозкой оплаты;
  + техподдержку через чат;
  + адаптацию под мобильные устройства.

Функциональные блоки включают в себя:

* + регистрация и авторизация;
  + поддержка email с подтверждением OTP;
  + роли: покупатель, автор, администратор;
  + форма включает: имя, фамилию, дату рождения, email, пароль, роль;
  + для авторов дополнительно: фото, категории, чекбокс «принимаю индивидуальные заказы».

Демо-режим:

* + для незарегистрированных пользователей выдаётся временный токен;
  + доступны: просмотр товаров, категорий, авторов;
  + при попытке использовать закрытые функции (чат, заказ) – возврат 403 и показ модального окна регистрации.

Индивидуальные заказы:

* + пользователь заходит в чат с автором;
  + общение свободное, при согласовании условий автор создаёт оффер;
  + оффер содержит: название, описание, срок, цену;
  + после подтверждения оффера пользователем – списание средств и заморозка;
  + после завершения заказа автор помечает как выполненный, пользователь подтверждает, средства переводятся автору.

Платное продвижение товаров:

* + автор может купить продвижение в выдаче;
  + сроки: 3/7/14 дней, разные тарифы;
  + продвинутые товары отображаются выше в выдаче;
  + при равных условиях сортируются по дате покупки продвижения;
  + при полной выдаче допускается ротация и расширенные сортировки.

Чат:

* + двустороннее общение между пользователем и автором;
  + поддержка текста, изображений, вложений;
  + интеграция индивидуальных заказов (офферы создаются в чате).

Техподдержка:

* + внедрение внешнего виджета (например, tawk.to);
  + пользователи могут задать вопрос в любой момент;
  + жалобы на товары, авторов, отзывы обрабатываются вручную оператором поддержки;
  + жалобы попадают в техподдержку через webhook или email.

Для проекта выбраны следующие технологии и проведен сравнительный анализ (таблицы 2.1­–2.5):

* + Frontend: HTML/CSS, JavaScript, React или Vue [14];
  + Backend: Go [15];
  + база данных: PostgreSQL [16];
  + чат: WebSocket [17];
  + OTP: через сторонние Email API;
  + платежи: ЮKassa [18], [19];

Таблица 2.1 – Сравнение с аналогами Frontend инструментов

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | React | Vue | Angular |
| Тип | Библиотека | Фреймворк | Фреймворк |
| Синтаксис | JSX | HTML-шаблоны | TypeScript |
| Реактивность | Virtual DOM | Реактивная модель | Change Detection |
| Размер | ~40 KB | ~20 KB | ~100 KB |
| Экосистема | Огромная (Next.js, Redux) | Большая (Vuex, Nuxt) | Полная (RxJS, Angular CLI) |
| Сложность | Средняя | Низкая | Высокая |
| Популярность | Очень высокая | Высокая | Высокая (корпоративный сектор) |

Таблица 2.2 – Сравнение аналогов Backend инструментов

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | Go (Golang) | Node.js (Express/Nest.js) | Python (Django/FastAPI) |
| Тип языка | Компилируемый | Интерпретируемый | Интерпретируемый |
| Производительность | Очень высокая | Быстро | Средняя |
| Параллелизм | Горутины | Event Loop | GIL ограничивает |
| Фреймворки | Gin, Echo, Fiber | Express, Nest.js, Fastify | Django, FastAPI, Flask |
| Синтаксис | Минимализм, строгий | Гибкий (JS/TS) | Читаемый |
| Размер проекта | Маленький (~10MB) | Зависит от node\_modules | Средний (виртуальное окружение |

Продолжение таблицы 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | Go (Golang) | Node.js (Express/Nest.js) | Python (Django/FastAPI) |
| Базы данных | Любые (драйверы для SQL/NoSQL) | MongoDB, PostgreSQL | ORM (Django), SQLAlchemy |

Таблица 2.3 Сравнение аналогов баз данных

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | PostgreSQL | MySQL | MongoDB |
| Тип СУБД | Реляционная (SQL) | Реляционная (SQL) | Документ (NoSQL) |
| Скорость | Высокая (оптимизирована для сложных запросов) | Быстрая (лучше для простых операций) | Быстрая (для JSON-данных) |
| Масштабируемость | Горизонтальная (через шардинг) + вертикальная | Вертикальная (репликация) | Горизонтальная (шардинг) |
| Транзакции | ACID (полная поддержка) | ACID (но зависит от движка) | Нет (только в отдельных случаях) |
| JSON-поддержка | (лучшая среди SQL) | (с MySQL 5.7+) | Основной формат |

Таблица 2.4 Сравнение аналогов реализации функций чата

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | WebSocket | HTTP Long Polling | Server-Sent Events (SSE) |
| Протокол | ws/wss (двусторонний) | HTTP (клиент – сервер) | HTTP (сервер – клиент) |
| Направление | Полнодуплексный | Полудуплексный | Односторонний (сервер – клиент) |
| Задержка | Минимальная  (1 RTT) | Высокая (повторные запросы) | Низкая |
| Поддержка браузеров | Все современные | Все | Все, кроме IE/Edge (legacy) |

Таблица 2.5 Сравнение аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | ЮKassa | CloudPayment | Tinkoff Pay |
| Основное направление | Универсальный агрегатор | Универсальный агрегатор | Платежный агрегатор банка |

Продолжение таблицы 2.5

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | ЮKassa | CloudPayment | Tinkoff Pay |
| Комиссия (основная, онлай) | 2.5% – 6% | 2.2% – 5% | 1.9% – 5% |
| Комиссия за вывод | Обычно бесплатно (зависит от тарифа) | Обычно бесплатно (зависит от тарифа) | Обычно бесплатно |
| Способы оплаты | Очень широкий: карты РФ/МИР, электронные кошельки (ЮMoney, QIWI), операторы, интернет-банки, СБП, крипто-рубли и др. | Широкий: карты, СБП, электронные кошельки, интернет-банки | Карты (особенно Тинькофф), СБП, интернет-банки (часто фокус на экосистеме Тинькофф) |

Платёжная архитектура:

* + пользователь пополняет баланс через ЮKassa;
  + при оформлении заказа – средства замораживаются;
  + после подтверждения выполнения – перечисляются автору.

Безопасность:

* + хеширование паролей(bcrypt/scrypt);
  + использование HTTPS повсеместно;
  + защита от XSS, CSRF, SQL-инъекций;
  + проверка прав доступа на сервере;
  + ограничения на загрузку файлов.

Панель администратора:

* + управление пользователями, жалобы, товарами;
  + возможность блокировки профиля;
  + просмотр заказов, статистики;
  + возможность модерации и удаления контента.

Мобильная адаптация:

* + интерфейсы полностью адаптированы под смартфоны;
  + минималистичный дизайн: чёрный фон + зелёные акценты;
  + использование bottom sheet и бургер-меню;
  + упрощённые формы взаимодействия.

Ключевой необходимостью является обеспечить быстрый, безопасный и ролевой процесс регистрации, который подтверждает владение email через одноразовый код (OTP), а также собирает профильные данные в зависимости от выбранной роли (покупатель или автор).

Регистрация доступна только неавторизованным пользователям. При наличии активной сессии (access token) кнопка регистрации в UI не отображается [20].

Регистрация состоит из двух частей:

* + подтверждение контакта email через OTP используя сторонний сервис SMT и корпоративной почты;
  + заполнение основной формы с персональными и ролевыми данными.

В процессе регистрации пользователь выбирает роль user – покупатель или seller – автор

Поля регистрационной формы, которые включают в себя типовые поля представлены в таблице 2.6.

Таблица 2.6 – Поля регистрационной формы

| Поле | Обязательно | Видимость | Валидация |
| --- | --- | --- | --- |
| Имя | Да | Всегда | min 2, max 30, кириллица/латиница |
| Фамилия | Да | Всегда | Аналогично |
| Дата рождение | Да | Всегда | ≥ 14 лет |
| Email | Да | Всегда | Уникальный, валидный |
| OTP-код | Да | Всегда | 6 цифр |
| Роль | Да | Всегда | Покупатель / автор |
| Категория товаров | Да | Только  для авторов | Из справочных категорий |
| Чекбокс «Индивидуальные заказ» | Да | Только для авторов | Логическое значение |

Этапы регистрации представляет ряд стандартных действий, а именно:

* + ввод контактных данных;
  + пользователь указывает email;
  + жмёт кнопку «Получить код»;
  + с этого момента UI блокирует остальную часть формы до подтверждения кода.

Отправка и валидация OTP:

* + сервер генерирует 6-значный код и сохраняет его хэш в таблице;
  + код отправляется пользователю по email;
  + пользователь вводит код, сервер сравнивает с хэшем, проверяет срок действия, количество попыток и флаг used;
  + завершение регистрации – после подтверждения кода, пользователь заполняет форму регистрации.

После отправки формы сервер:

* + создает запись пользователя в Users;
  + присваивает роль user или seller;
  + возвращает access token;
  + пользователь автоматически считается авторизованным и перенаправляется в личный кабинет.

Пример тела содержащий необходимые элементы кода запроса предоставлен в приложении А с необходимой структурой и описанием строк для удобства внедрения.

Таблица 2.7 – Поведение UI

|  |  |
| --- | --- |
| Условие | Реакция интерфейса |
| Пользователь авторизован | Кнопка «Регистрация» не отображается |
| Введен email, не подтверждён OTP | Блок остальных полей |
| OTP истёк или неверный | Сообщение об ошибке |

Продолжение таблицы 2.7

|  |  |
| --- | --- |
| Условие | Реакция интерфейса |
| Email уже используется | Ошибка: «Пользователь уже существует» |
| Все поля валидны | Кнопка «Зарегистрироваться» активна |

Безопасность:

* + коды OTP хранятся в хэшированном виде (bcrypt);
  + время жизни кода – 5 минут;
  + попытки ввода ограничены (3 попытки);
  + повторная отправка возможна через 30-60 секунд;
  + уникальный идентификатор – Email.

**2.2 Гостевой доступ, поддержка и этапы платёжного процесса**

Назначение – демо-режим предназначен для демонстрации платформы новым пользователям без обязательной регистрации. Он позволяет ознакомиться с публичным контентом (лендинг, товары, авторы) и побуждает к регистрации при попытке воспользоваться защищённым функционалом [21].

Таблица 2.8 – Архитектура доступа

|  |  |  |
| --- | --- | --- |
| Тип пользователя | Auth-token | Доступ |
| Гость | Временный JWT (role: guest) | Только публичные ресурсы (просмотр товаров, категорий, авторов) |
| Зарегистрированный | Полный JWT (role: user/seller) | Весь разрешённый функционал |

Получение гостевого токена.

При первом заходе на сайт, если пользователь не авторизован:

* + фронтенд инициирует POST /auth/guest-token;
  + бэкенд выдает временный токен JWT с ролью guest.

Пример токена прикреплен в приложении Б.

Таблица 2.9 – Права доступа для guest

|  |  |
| --- | --- |
| Разрешено | Запрещено |
| Просматривать каталог товаров | Оформлять заказы |
| Смотреть карточки авторов и категории | Открывать чаты |
| Искать и фильтровать товары | Оставлять отзывы |
| Перейти на страницу регистрации | Писать сообщения, загружать товары (автор) |

Поведение при попытке доступа к недоступной функции

* гость вызывает API-функцию, доступ к которой запрещён (role: guest).
* бэкенд возвращает 403 Forbidden.
* фронтенд перехватывает 403 и:
  + 1. выводит модальное окно регистрации;
    2. вариант: редирект на /register с сохранением контекста.

Пример: Гость нажимает «Написать продавцу» – API POST /messages возвращает 403, тогда на экране отображается: «Чтобы воспользоваться чатом, необходимо зарегистрироваться».

UI-поведение:

* + все публичные действия доступны без ограничений;
  + защищённые кнопки активны, но при клике инициируют проверку авторизации;
  + поведение одинаковое на SPA и SSR-страницах;
  + повторная попытка действия после регистрации/входа выполняется автоматически при наличии сохранённого контекста (опционально).

Пример реализации логики прикреплен в приложении Б.

Общий принцип платёжной архитектуры:

Платформа использует модель с внутренним балансом пользователя, через которую проходят все финансовые операции. Пользователь предварительно пополняет свой баланс через платёжный шлюз ЮKassa, после чего средства:

* + замораживаются при оформлении заказа;
  + размораживаются и перечисляются автору после подтверждения выполнения.

Такой подход упрощает юридическую модель и контроль над транзакциями, а также обеспечивает гарантию возврата средств до момента приёма заказа.

Таблица 2.10 – Участники схемы

|  |  |
| --- | --- |
| Роль | Функция |
| Покупатель | Пополняет баланс, оформляет заказы |
| Платформа | Хранит замороженные средства, управляет статусами |
| Продавец | Получает выплату после подтверждённого выполнения заказа |
| ЮKassa | Обрабатывает внешние платежи, переводит средства на баланс платформы |

Пополнение баланса используя внедрения сервиса ЮKassa содержит в себе следующие шаги, строки базы данных представлены в таблице 2.11:

* + пользователь инициирует пополнение через UI (например, на экране «Баланс»);
  + запрос отправляется на бэкенд: POST /payments/initiate;
  + бэкенд генерирует платёжную сессию в ЮKassa через их API;
  + ЮKassa открывает форму оплаты (в iframe или редиректе).

После успешной оплаты:

* + платформа получает webhook от ЮKassa (/payments/yookassa/webhook);
  + баланс пользователя увеличивается на сумму платежа и сохраняется в базе данных, также изменяется на стороне сайта, изменяя числовое значение на сумму платежа.

Таблица 2.11 – Платежи

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Комментарий |
| id | UUID | PK |
| user\_id | UUID | FK – Users |
| amount | DECIMAL | Сумма пополнения |
| status | ENUM | pending, success, failed |
| method | VARCHAR | Способ оплаты |
| transaction\_id | VARCHAR | ID транзакции ЮKassa |
| created\_at | TIMESTAMP |  |

Оформление заказа:

* + при оформлении заказа пользователь выбирает товар;
  + бэкенд проверяет, достаточно ли средств на балансе.

Сумма заказа отображается в таблице 2.12:

Таблица 2.12 – Замороженные платежи

|  |  |
| --- | --- |
| Поле | Комментарий |
| id | UUID |
| user\_id | Кто оплатил |
| order\_id | К какому заказу |
| amount | Сумма к заморозке |
| locked\_at | Время блокировки |
| Status | locked, released, transferred, canceled |

Подтверждение выполнения:

* + после того как заказ выполнен, пользователь подтверждает это на клиенте.

Таблица 2.13 – Выплаты

|  |  |
| --- | --- |
| Поле | Комментарий |
| id | UUID |
| seller\_id | Кому перевели |
| order\_id | За что |
| amount | Сумма |

Возврат средств (в случае отмены):

* + если заказ отменён (по инициативе пользователя или по жалобе), замороженная сумма возвращается на баланс;
  + бэкенд меняет статус в order\_locks на released.

Примеры API прикреплены в приложении В.

Необходимо внедрить на платформу внешний чат поддержки через сервис Tawk.to, без собственной backend-реализации. Чат должен работать как для авторизованных, так и для неавторизованных пользователей.

Задачи:

* + установить виджет Tawk.to на все клиентские страницы сайта (кроме панели администратора);
  + если пользователь авторизован – передавать имя и email в чат (через Tawk\_API.visitor);
  + если не авторизован – чат должен запросить эти данные у пользователя вручную;
  + при открытии чата в offline-режиме должно быть возможно оставить сообщение;
  + виджет должен быть загружен асинхронно;
  + скрипт не должен тормозить основную загрузку сайта.

При реализации необходимо использовать готовый скрипт от Tawk.to с заменой ID на конкретный с панели интеграции [22].

Также необходимо подставить реальные значения name и email для авторизованного пользователя.

Пример скрипта прикреплен в приложении Г.

Внедрение происходит в фронтенд-приложении, исключением является панель администратора и экраны 404/500 и системных ошибок.

Требования:

* + виджет не должен влиять на производительность основной страницы;
  + при ошибке загрузки скрипта (например, Tawk.to недоступен) – никаких UI-крашей;
  + пользователь должен иметь возможность открыть чат с любого экрана;
  + проверка авторизации должна происходить до инициализации скрипта.

Задача считается выполненной, если:

* + виджет отображается на клиенте;
  + при авторизации – имя/email пользователя передаются автоматически;
  + при отсутствии авторизации – Tawk.to запрашивает эти данные;
  + нет ошибок при загрузке, нет утечек в консоли.

Необходимо позволить пользователю (авторизованному) пожаловаться на:

* + товар;
  + продавца;

Жалоба не требует разработки полноценной системы тикетов – она передаётся напрямую в техподдержку через Tawk.to или другой внешний сервис (email/slack/webhook). Вся дальнейшая обработка ведётся вручную оператором.

Таблица 2.14 – Доступность подачи жалоб

|  |  |
| --- | --- |
| Объект жалобы | Гдe показываем кнопку |
| Продукт | Страница товара |
| Продавец | Профиль автора |
| Отзыв | Под конкретным комментарием |

Поведение UI при нажатии на кнопку «Пожаловаться»:

* + открывается модальное окно с текстовым полем: «Опишите причину жалобы»;
  + пользователь отправляет жалобу;
  + жалоба уходит на бэкенд по POST /complaints;
  + жалоба записывается в таблицу complaints для истории.

Примеры запросов прикреплены в приложении Г.

Таблица 2.15 – Поведение UI

|  |  |
| --- | --- |
| Сценарий | Реакция |
| Жалоба отправлена успешно | Показываем toast: «Жалоба отправлена. Спасибо!» |
| Ошибка при отправке | «Произошла ошибка. Повторите позже» |
| Повторная жалоба тем же пользователем | Заблокировать кнопку |

Такое поведение интерфейса обеспечивает пользователю прозрачную и предсказуемую обратную связь при взаимодействии с системой. Наличие toast-уведомлений помогает оперативно информировать пользователя о результате действия, не перегружая интерфейс. Блокировка повторной отправки жалобы тем же пользователем предотвращает спам и снижает нагрузку на модерацию, обеспечивая при этом корректную работу механизма обратной связи.

**2.3 Механика платного продвижения и сценарии оформления индивидуального заказа**

Платное продвижение – это временная платная подписка, активируемая для отдельного товара, которая повышает его приоритет в выдаче. Продвижение влияет только на сортировку и не меняет других характеристик товара [38].

* активное продвижение определяется как CURRENT\_TIMESTAMP BETWEEN start\_at AND end\_at;
* один товар может иметь только одно активное продвижение;
* период действия – 3/7/14 дней (через тариф).

Выдача товаров в каталоге строится по приоритету:

* boosted\_at DESC – все товары с активным продвижением идут первыми;
* далее обычные товары: created\_at DESC.

Даже если все товары в категории активировали платное продвижение, порядок определяется по:

* start\_at – более свежее продвижение выше;
* при равных условиях – created\_at товара (то есть кто раньше выложил – ниже).

Процесс покупки продвижения:

* продавец заходит на свою страницу товара;
* нажимает кнопку «Продвинуть товар»;
* выбирает тариф;
* сервер проверяет «Есть ли уже активное продвижение?» и «Достаточно ли средств на балансе?»;
* списывает средства, создает запись в boosts;
* в выдаче товар моментально поднимается.

Сценарий индивидуальных заказов можно представить следующим образом:

* вся логика индивидуальных заказов реализована через чаты между пользователем и автором;
* автор – единственный, кто может создать оффер (предложение по заказу);
* пользователь может инициировать диалог и вести переговоры в свободной форме, без ограниченных шаблонов.

Общий сценарий:

* пользователь заходит в профиль автора:
  1. через карточку товара – ссылка на автора – страница профиля;
  2. в профиле отображается кнопка «Написать автору» или «Заказать

индивидуальный товар»;

* 1. нажимает – открывается чат (если уже есть – переходит в него).
* общение в чате:
  1. пользователь свободно описывает, что хочет;
  2. может прикрепить фото, написать текст, обсудить идею;
  3. автор отвечает, уточняет детали.
* автор создает индивидуальный заказ (оффер):
  1. в чате у автора имеется кнопка: «Создать заказ»;
  2. открывается модальное окно, которое содержит информацию о заказе, а именно его название, описание, цену и срок выполнения.

В чате появляется карточка с заказом которое содержит информацию, введенную автором, после чего покупатель в праве оплатить заказ.

* после оплаты:
  1. списывается сумма с баланса пользователя;
  2. заказ получает статус in\_progress;
  3. деньги замораживаются.
* после выполнения:
  1. автор нажимает «Заказ выполнен»;
  2. пользователь подтверждает – деньги переводятся автору.

UI / UX элементы для пользователя:

* + кнопка «Написать автору» – переход в чат;
  + карточка оффера с кнопкой «Оплатить»;
  + статус заказа: В работе / Выполнен / Отклонён.

Для автора:

* кнопка в чате: «Создать индивидуальный заказ»;
* форма создания заказа (описание, цена, срок);

Таблица 2.16 – Ошибки / ограничения

|  |  |
| --- | --- |
| Сценарий | Поведение |
| Пользователь не авторизован | Кнопка чата недоступна |
| Автор пытается создать 2 заказа подряд | Блок, если предыдущий ещё не завершён |
| Недостаточно средств у пользователя | Кнопка «Оплатить» ведёт на пополнение |

Продолжение таблицы 2.16

|  |  |
| --- | --- |
| Сценарий | Поведение |
| Превышен лимит по цене / сроку | Валидация формы |

Для повышения прозрачности пользовательского взаимодействия при индивидуальных заказах в интерфейсе чата отображается статус заказа в режиме реального времени. Это реализовано с помощью WebSocket-соединения, что позволяет обеим сторонам – заказчику и исполнителю – видеть актуальную информацию без необходимости вручную обновлять страницу. Такой подход минимизирует риски недопонимания и повышает уровень доверия между пользователями [39].

Каждый индивидуальный заказ сохраняется в основной структуре заказов платформы – таблице Orders. При этом:

* В поле type указывается значение custom, что позволяет легко отличать индивидуальные заказы от стандартных.
* В поле custom\_config сохраняется сериализованная конфигурация заказа, полученная из пользовательской формы (например, описания, пожелания, параметры кастомизации и т. д.).

Это обеспечивает:

* единый подход к хранению заказов;
* возможность корректной агрегации данных в статистике;
* полноценный анализ пользовательских предпочтений и бизнес-показателей;
* поддержку функционала повторных обращений (например, дозаказов или редизайна).

**3 Разработка модели системы**

**3.1 Use-case и ER диаграммы**

Use case диаграмма (диаграмма вариантов использования) это один из основных видов диаграмм в унифицированном языке моделирования (UML), который используется для описания взаимодействия пользователей с проектируемой системой.

Use case диаграммы помогают наглядно представить требования к системе и её взаимодействие с пользователями, что упрощает процесс проектирования и разработки [23]. Необходимы диаграммы представлены на рисунках 3.1-3.4.

Система включает четыре типа пользователей:

* неавторизованные пользователи – базовый доступ без регистрации;
* пользователи (покупатели) – зарегистрированные клиенты;
* авторы – создатели товаров с правами продавцов;
* модераторы – контролируют соблюдение правил.

Авторы наследуют все возможности обычных пользователей и дополняют их функциями продавца.

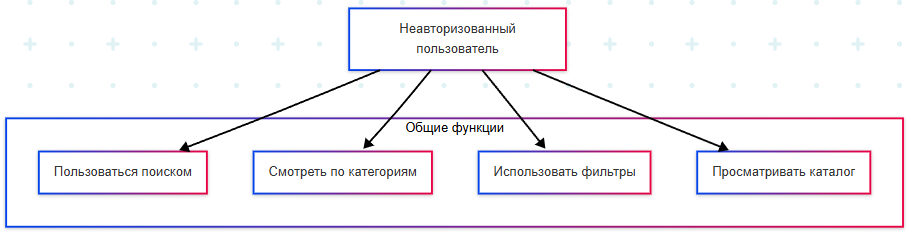


Рисунок 3.1 – Права неаторизованного пользователя

Все типы пользователей могут:

* просматривать каталог товаров;
* использовать фильтры для поиска;
* находить товары по категориям;
* пользоваться поисковой системой.

Эти функции не требуют авторизации.

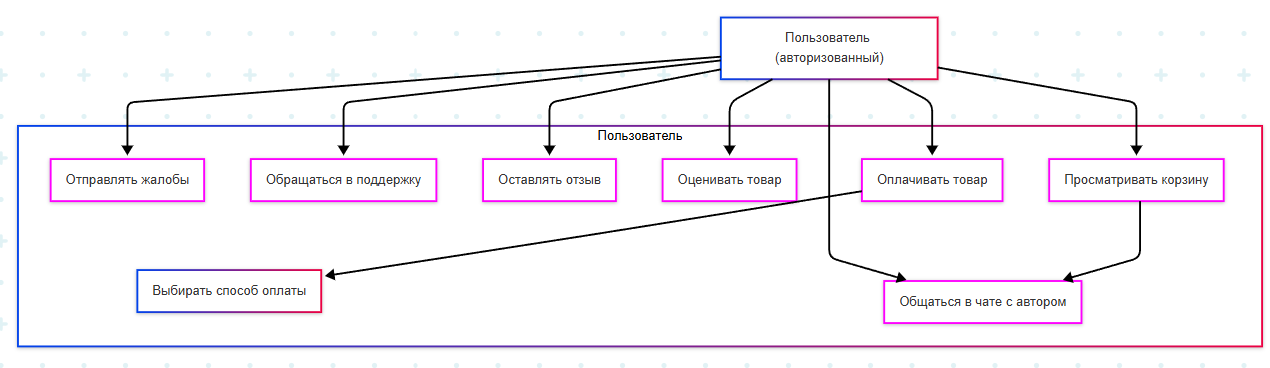


Рисунок 3.2 – Права авторизованного пользователя

Авторизованные пользователи дополнительно могут:

* управлять корзиной покупок;
* оплачивать товары (с обязательным выбором способа оплаты);
* общаться с авторами в чате;
* оценивать товары и оставлять отзывы;
* обращаться в техническую поддержку;
* отправлять жалобы на нарушения.

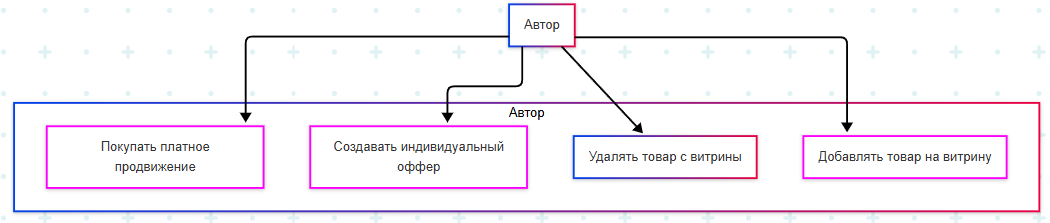


Рисунок 3.3 – Права автора

Создатели товаров обладают всеми правами покупателей плюс:

* публикация товаров на своей витрине;
* удаление товаров со своей витрины;
* создание индивидуальных предложений;
* покупка платного продвижения товаров.

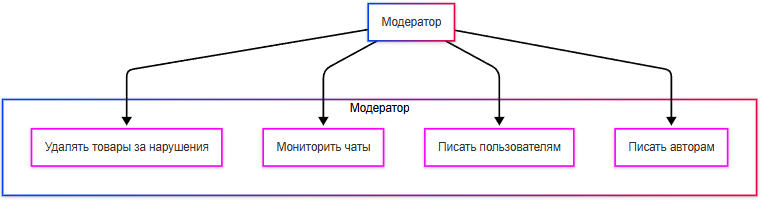


Рисунок 3.4 – Права модератора

Модераторы обеспечивают контроль за системой:

* удаление товаров за нарушения правил;
* мониторинг переписок в чатах;
* отправка сообщений пользователям;
* общение с авторами по административным вопросам.

ER-диаграмма представляет собой визуальное средство моделирования логической структуры базы данных, описывающее сущности предметной области, их свойства (атрибуты) и взаимосвязи между ними. В процессе проектирования информационных систем, особенно на ранних этапах анализа, ER-диаграммы играют важную роль в формализации требований и обеспечивают понятное и структурированное представление данных для всех участников проекта – от аналитиков до разработчиков [40].

В контексте веб-платформы, предназначенной для продажи индивидуальных (кастомных) товаров, ER-диаграмма позволяет детализированно отразить ключевые компоненты системы, такие как пользователи, авторы, товары, заказы, переписка, а также показать, как эти элементы связаны между собой. Например, один пользователь может оформить несколько заказов, каждый заказ может включать один или несколько товаров, а товар принадлежит определённому автору. Диаграмма изображенная на рисунке 3.5 отображает актуальную модель, также составлен словарь данных (таблица 3.1–3.10).

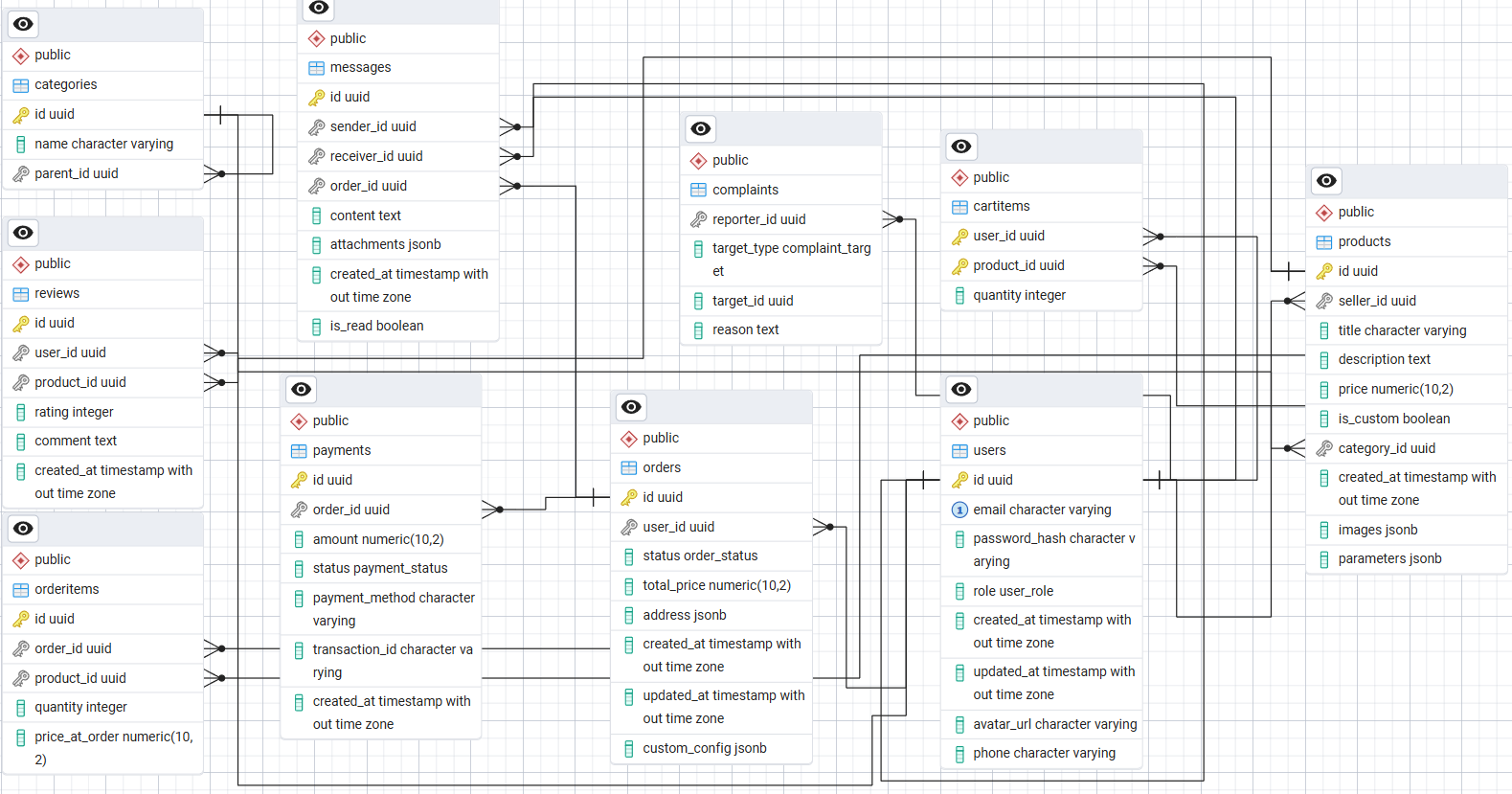


Рисунок 3.5 – ER-диаграмма

Таблица 3.1 –Пользователи платформы

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| Id | UUID (PK) | Уникальный идентификатор |
| Email | VARCHAR | Email (уникальный) |
| password\_hash | VARCHAR | Хеш пароля |
| Role | ENUM | user / seller / admin |
| created\_at | TIMESTAMP | Дата регистрации |
| updated\_at | TIMESTAMP | Дата обновления профиля |
| avatar\_url | VARCHAR | Ссылка на аватар (опционально) |
| phone | VARCHAR | Телефон (опционально) |

Связи:

1:N с products (продавец)

1:N с orders (покупатель)

1:N с messages (как sender и receiver)

1:N с reviews, complaints, payments, cartitems

Таблица 3.2 – Товары на платформе

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| id | UUID (PK) | ID товара |
| seller\_id | UUID (FK) | Автор (ссылка на users.id) |
| title | VARCHAR | Название товара |
| description | TEXT | Описание |
| price | DECIMAL | Цена |
| is\_custom | BOOLEAN | Индивидуальный ли товар |
| category\_id | UUID (FK) | Категория |
| created\_at | TIMESTAMP | Дата добавления |
| images | JSONB | Массив ссылок на изображения |
| parameters | JSONB | Кастомные характеристики |

Связи:

1:N с orderitems, reviews, cartitems, complaints

Таблица 3.3 – Категории товаров

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| id | UUID (PK) | ID категории |
| name | VARCHAR | Название категории |
| parent\_id | UUID (FK) | Родительская категория (если есть) |

Таблица 3.4 – Заказы

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| id | UUID (PK) | ID заказа |
| user\_id | UUID (FK) | Покупатель |
| status | ENUM | pending, paid, in\_progress и т. д. |
| total\_price | DECIMAL | Общая сумма |
| address | JSONB | Данные доставки |
| created\_at | TIMESTAMP | Дата создания |
| updated\_at | TIMESTAMP | Дата обновления |
| custom\_config | JSONB | Настройки для кастомных заказов |

Связи:

1:N с orderitems, payments, messages

Таблица 3.5 – Позиции в заказе

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| id | UUID (PK) | ID позиции |
| order\_id | UUID (FK) | Связь с заказом |
| product\_id | UUID (FK) | Продукт |
| quantity | INT | Кол-во |
| price\_at\_order | DECIMAL | Цена на момент оформления |

Состав заказа

Таблица 3.6 – Корзина пользователя

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| user\_id | UUID (PK) | Пользователь |
| product\_id | UUID (PK) | Товар |
| quantity | INT | Количество |

Промежуточное хранилище перед оформлением заказа

Таблица 3.7 – Чат-сообщения

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| id | UUID (PK) | ID сообщения |
| is\_read | BOOLEAN | Прочитано/не прочитано |
| sender\_id | UUID (FK) | Отправитель |
| receiver\_id | UUID (FK) | Получатель |
| order\_id | UUID (FK) | Заказ (опционально) |
| content | TEXT | Текст сообщения |
| attachments | JSONB | Вложения (ссылки на файлы) |

Таблица 3.8 – Отзывы

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| id | UUID (PK) | ID отзыва |
| user\_id | UUID (FK) | Кто оставил |
| product\_id | UUID (FK) | На какой товар |

Продолжение таблицы 3.8

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| rating | INT | Оценка 1-5 |
| comment | TEXT | Комментарий |
| created\_at | TIMESTAMP | Дата |

Таблица 3.9 – Платежи

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| id | UUID (PK) | ID платежа |
| order\_id | UUID (FK) | Заказ |
| amount | DECIMAL | Сумма |
| status | ENUM | pending / success / failed |
| method | VARCHAR | Метод (card, sbp и т.д.) |
| transaction\_id | VARCHAR | ID транзакции в ЮKassa |
| created\_at | TIMESTAMP | Дата оплаты |

Таблица 3.10 – Жалобы

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| reporter\_id | UUID (FK) | Кто пожаловался |
| target\_type | ENUM | 'product', 'seller', 'review' |
| target\_id | UUID | ID объекта жалобы |
| reason | TEXT | Причина жалобы |

Таким образом, ER-диаграмма помогает не только структурировать данные, но и задать правила взаимодействия между ними, обеспечивая логическую целостность системы. Это критически важно при разработке платформы, ориентированной на индивидуальные заказы, где важны точность, гибкость и возможность адаптации под уникальные запросы покупателей [24].

Благодаря четко сформулированным связям между сущностями можно реализовать механизмы, обеспечивающие непротиворечивость данных, например, ограничение удаления авторов, за которыми закреплены товары, или валидацию корректности состава заказов. Использование типов данных, таких как JSONB, позволяет обеспечить расширяемость модели за счёт поддержки гибких параметров товаров, что особенно важно в условиях кастомизации. Кроме того, JSONB облегчает хранение нестандартизированных атрибутов.

**3.2 Диаграмма последовательности и IDEF-0**

Диаграмма последовательности представляет собой один из ключевых инструментов визуального моделирования, предназначенный для отображения взаимодействия между объектами системы во времени. Данный тип диаграмм позволяет зафиксировать порядок вызовов, сообщений и откликов между участниками процесса при выполнении определённого сценария.

В рамках проектирования веб-платформы по продаже кастомных товаров диаграмма последовательности используется для формализации логики [25].

Диаграмма, представленная на рисунке 3.6 демонстрирует линейную цепочку событий, начиная с открытия пользователем веб-платформы, авторизации и просмотра каталога товаров, до добавления товара в корзину, пополнения баланса через ЮKassa и завершения заказа. Взаимодействия между участниками представлены в виде сообщений (запросов/ответов) с учётом логики бизнес-процесса и временной последовательности [41].

Основные участники (lifelines):

* User – пользователь платформы;
* фронтэнд – клиентская часть (браузер);
* бэкэнд – серверное приложение (Go API);
* DB – база данных PostgreSQL;
* ЮKassa – внешний платёжный шлюз.

Ключевые моменты:

* проверка и получение списка товаров с сервера;
* аутентификация и получение токена;
* инициализация платежа и webhook-уведомление от платёжной системы;
* создание заказа, фиксация суммы, заморозка средств;
* подтверждение успешного оформления заказа пользователю.

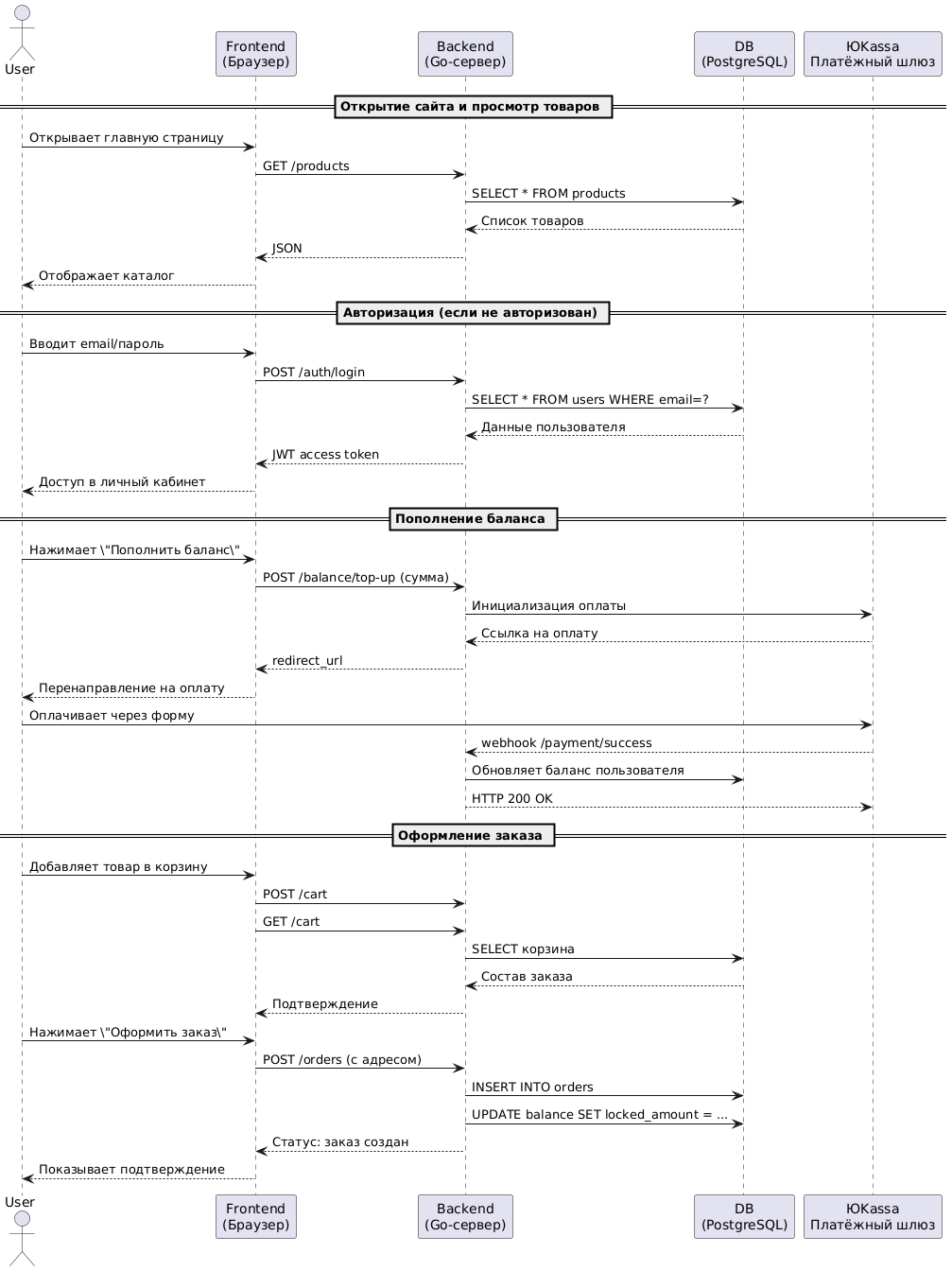


Рисунок 3.6 – Диаграмма последовательности

IDEF-0 отображенная на рисунке 3.7 необходима чтобы отразить верхнеуровневую архитектуру взаимодействия платформы с внешними и внутренними сущностями – входами, управляющими элементами, механизмами и выходами в терминах нотации [26].

Входы:

* запросы пользователей – действия покупателей, включающие просмотр товаров, регистрацию, оформление заказов, переписку и жалобы;
* запросы авторов – действия продавцов: добавление товаров, настройка профиля, общение с клиентами, обработка заказов;
* заказы – заявки, поступающие от пользователей, как на готовые товары, так и на индивидуальные проекты.

Управление (Control):

* бизнес-правила – регламентирующие механизмы платформы: логика размещения, комиссии, бустов, порядок обработки заказов;
* политика модерации – правила допустимого контента, проверка профилей, товаров, жалоб;
* условия оплаты и безопасности – сценарии взаимодействия с платёжной системой, валидация заказов, удержание средств (эскроу), правила возвратов и подтверждения доставки.

Механизмы (Mechanism):

* бэкэнд (Go API) – программный интерфейс серверной части платформы, обрабатывающий бизнес-логику, запросы, авторизацию, чаты и офферы;
* PostgreSQL – основная реляционная база данных, хранящая пользователей, товары, заказы, отзывы, переписки;
* UI – интерфейс пользователя, обеспечивающий взаимодействие всех ролей с системой (адаптивный веб-интерфейс);
* ЮKassa – интеграция с платёжным шлюзом для пополнения баланса, заморозки средств и оплаты заказов;
* интеграция техподдержки – подключённый сторонний сервис (например, Tawk.to) для обработки обращений пользователей и жалоб.

Выходы (Output):

* оформленные заказы – результат успешного взаимодействия между автором и покупателем, завершённые и оплаченные сделки;
* продвижение товаров – активные бусты, подписки на приоритетное размещение, улучшение видимости автора;
* отзывы, жалобы – пользовательская обратная связь, направляемая в систему модерации и технической поддержки.

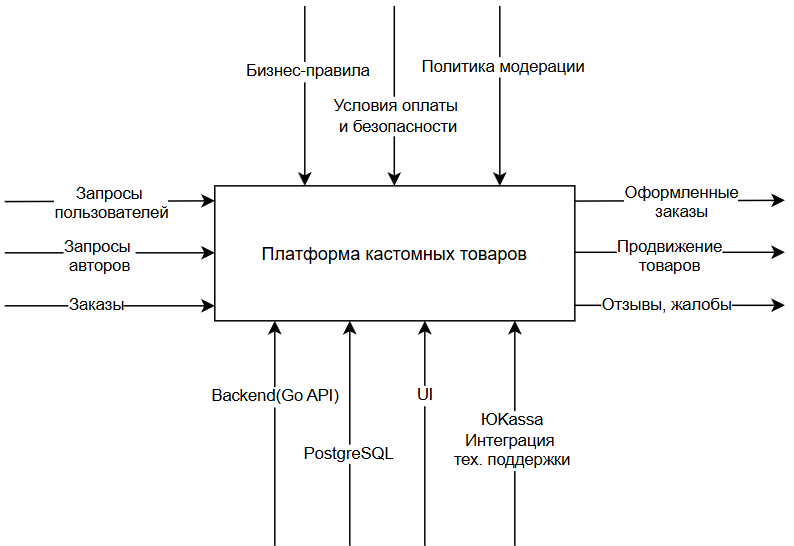


Рисунок 3.7 – IDEF-0

Данная диаграмма A-0 чётко отражает функциональную декомпозицию верхнего уровня платформы, определяя ключевые входы, механизмы и управленческие факторы. Она используется как основа для последующей детализации в уровнях A1, A2 и т.д., где можно будет разложить процессы заказа, модерации, оплаты и коммуникации по подфункциям. Декомпозиция на уровне A-0 позволяет выявить основные логические блоки системы и установить границы ответственности между ними. Она позволяет на раннем этапе выявить взаимосвязи между основными процессами и внешними сущностями, обеспечивая основу для системного подхода к проектированию.

**3.3 Прототипирование интерфейса**

Прототипирование интерфейса представляет собой этап предварительной визуализации пользовательского интерфейса цифрового продукта с целью формализации структуры, логики взаимодействия и ключевых функциональных элементов до начала этапа непосредственной разработки.

Основной задачей прототипирования является обеспечение возможности ранней проверки концепции интерфейса, выявления потенциальных проблем в пользовательском опыте, а также согласования проектных решений с заинтересованными сторонами (заказчиком, разработчиками, аналитиками).

Прототип служит инструментом моделирования интерфейса, позволяющим оценить сценарии взаимодействия и провести предварительное тестирование без значительных временных и финансовых затрат. Это особенно актуально на этапе проектирования, поскольку внесение изменений на основе замечаний и обратной связи осуществляется существенно быстрее и экономичнее, чем в процессе переработки уже реализованного продукта [27].

Дизайн платформы выполнен в тёмной цветовой теме, что создаёт ощущение премиальности и не отвлекает внимание от визуальной составляющей кастомных товаров. Все элементы интерфейса – иконки, кнопки, карточки – оформлены в едином минималистичном стиле, подчёркивая целостность визуального языка. В качестве акцента используется бирюзово-зелёный цвет, выделяющий ключевые действия (CTA) и элементы навигации. В центре внимания находится уникальность представленных изделий и индивидуальность их авторов, что подчёркивается подачей контента и персонализацией. Все компоненты интерфейса интуитивно понятны и адаптированы под специфику кастомного формата: лёгкое взаимодействие, визуальная демонстрация изделий и удобная покупка.

Было разработано 6 предварительных прототипов системы:

* главная страница;
* список товаров;
* карточка товара;
* корзина
* профиль автора;
* чат.

Прототип главной страницы представленный на рисунке 3.8 играет важную роль в формировании первого впечатления о платформе и задаёт общее направление пользовательского взаимодействия. Он позволяет наглядно представить, как будет выглядеть и функционировать основной входной интерфейс, а также демонстрирует ключевые элементы навигации, визуальной идентики и контентной структуры.

Главная страница обеспечивает пользователя базовой информацией о платформе – кто мы, что предлагаем и как это работает. Она направляет внимание на ценностное предложение сервиса, подчёркивая уникальность кастомных товаров и возможности взаимодействия с авторами. Важнейшая функция прототипа – маршрутизация: за счёт продуманной компоновки элементов и акцентов пользователь интуитивно переходит к ключевым действиям – просмотру каталога, регистрации, авторизации, изучению преимуществ платформы или началу общения с автором.

Таким образом, прототип главной страницы не просто отражает внешний облик интерфейса, но и закладывает логику пользовательского пути, обеспечивая основу для дальнейшего проектирования и реализации системы [42].

Прототип позволяет выявить возможные узкие места в навигации и взаимодействии до начала разработки, что делает его важным инструментом на этапе планирования. Он служит связующим звеном между бизнес-целями платформы и ожиданиями пользователей, демонстрируя, каким образом визуальные и функциональные элементы могут быть объединены в цельный и удобный интерфейс.

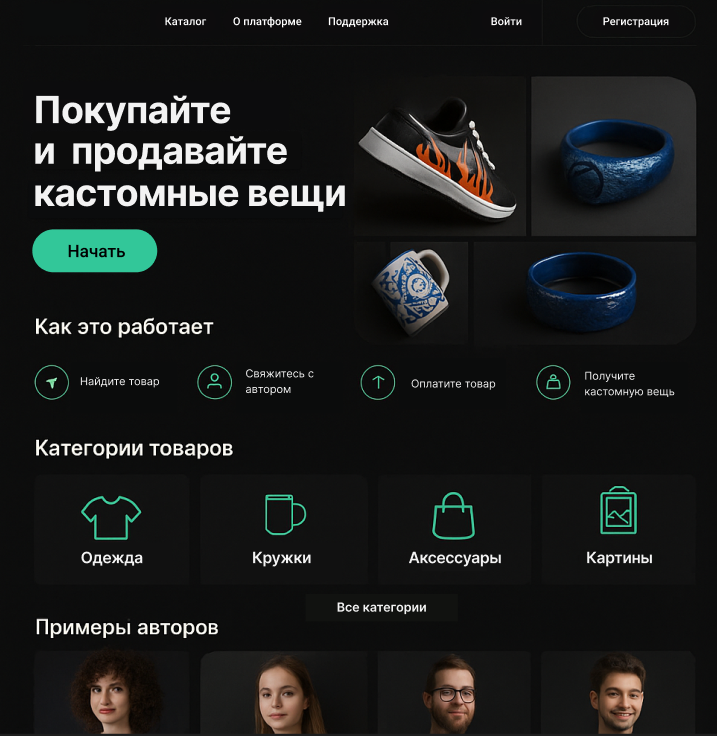


Рисунок 3.8 – Главная страница

Верхняя панель навигации:

* разделы: «Каталог», «О платформе», «Поддержка», а также кнопки «Войти» и «Регистрация»;
* чёткая структура для быстрого доступа к информации и действиям.

Блок с УТП и призывом к действию:

* заголовок: «Покупайте и продавайте кастомные вещи» – подчёркивает основную идею платформы;
* кнопка «Начать» – выделена и ведёт к первому действию (например, регистрации или каталогу);
* сбоку – карусель или статичная подборка примеров кастомных товаров.

Секция «Как это работает»:

* найдите товар;
* свяжитесь с автором;
* оплатите;
* получите кастомную вещь.
* иконки выполнены в едином стиле, упрощая восприятие.

Категории товаров:

* карточки с иконками: «Одежда», «Кружки», «Аксессуары», «Картины»;
* акцентная подсветка при наведении или выборе категории.

Блок «Примеры авторов»:

* аватары и имена – подчёркивается индивидуальность и живой характер площадки;
* кнопка «Все категории» позволяет развернуть больше товаров.

Список товаров представленный на рисунке 3.9 выполняет ключевую функцию в пользовательском интерфейсе платформы, так как именно через него осуществляется первичное знакомство покупателя с ассортиментом представленных изделий. Он служит основным каналом визуальной презентации кастомных товаров и одновременно выполняет навигационную и коммерческую роли.

Наличие списка товаров позволяет пользователю быстро оценить доступные предложения, сравнить их по визуальным характеристикам, цене и описанию, а также при необходимости перейти к детальной карточке конкретного изделия. Это значительно упрощает процесс выбора, делает его интуитивным и минимизирует барьеры к покупке. Для платформы, ориентированной на индивидуальные и уникальные товары, это особенно важно – визуальный ряд и понятное представление становятся основным способом донесения ценности продукта [43].

Кроме того, список товаров реализует возможность быстрой покупки или начала диалога с автором, если требуется уточнение деталей, что важно в кастомной модели взаимодействия. Таким образом, данный компонент не только решает задачу отображения, но и активно вовлечён в процесс взаимодействия между продавцом и покупателем.

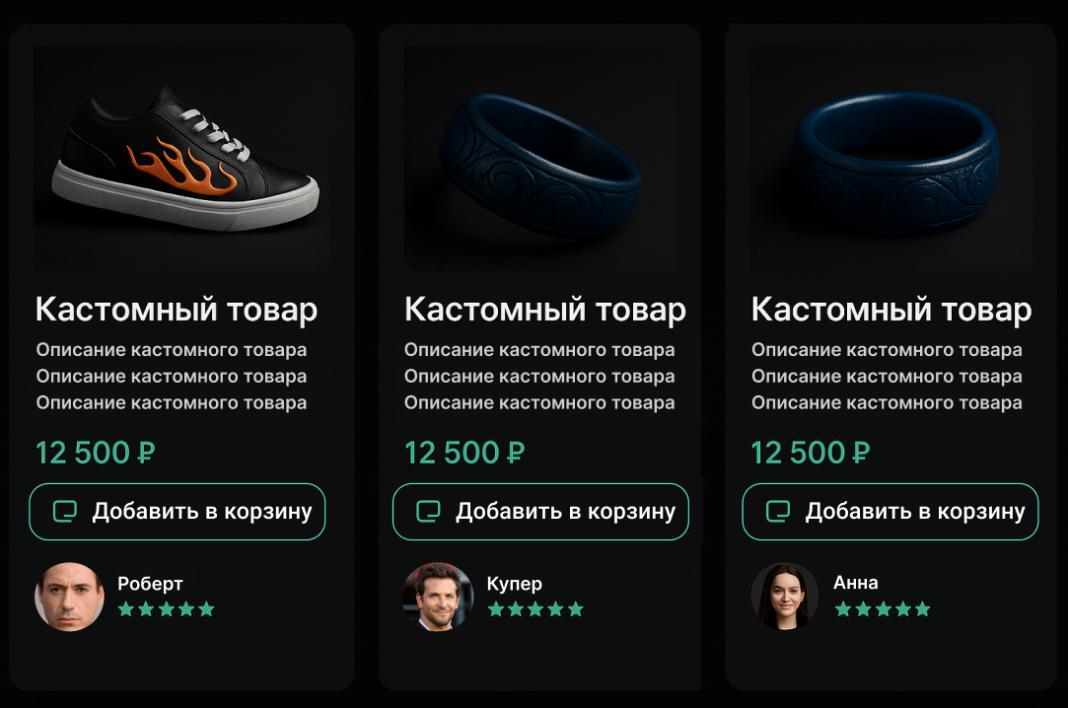


Рисунок 3.9 – Список товаров

Страница представленная на рисунке 3.10 имеет следующий состав компонентов:

* изображение товара;
* название;
* описание;
* цена;
* кнопка «Купить»;
* информация об авторе: имя, аватар, рейтинг, переход в профиль.

Требования к доступу:

* возможность покупки доступна только авторизованным пользователям.

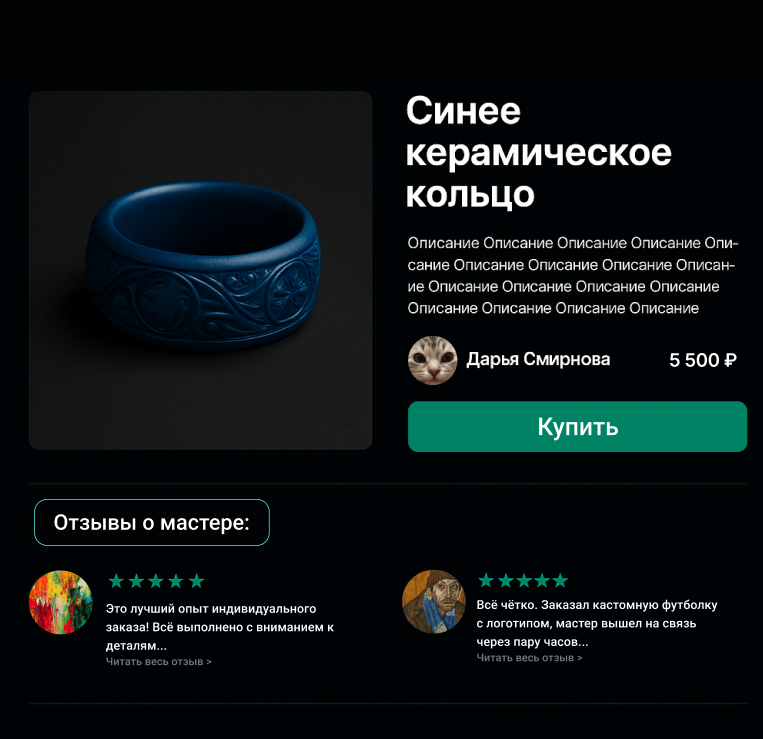


Рисунок 3.10 – Карточка товара

Фото изделия:

* крупное изображение синего керамического кольца на тёмном фоне, подчёркивающее фактуру и дизайн.

Описание товара:

* название: «Синее керамическое кольцо»;
* краткое текстовое описание (в данном случае – плейсхолдер);
* имя автора с аватаром (Дарья Смирнова) и цена (5 500 ₽);
* кнопка «Купить» – выделена, способствует быстрому действию.

Блок отзывов о мастере:

* название блока: «Отзывы о мастере»;

Отзывы включают:

* аватар покупателя или изображение заказа;
* звёздный рейтинг;
* фрагмент текста и ссылку на полный отзыв;

Примеры подчеркивают удовлетворённость покупателей и высокую вовлечённость мастеров.

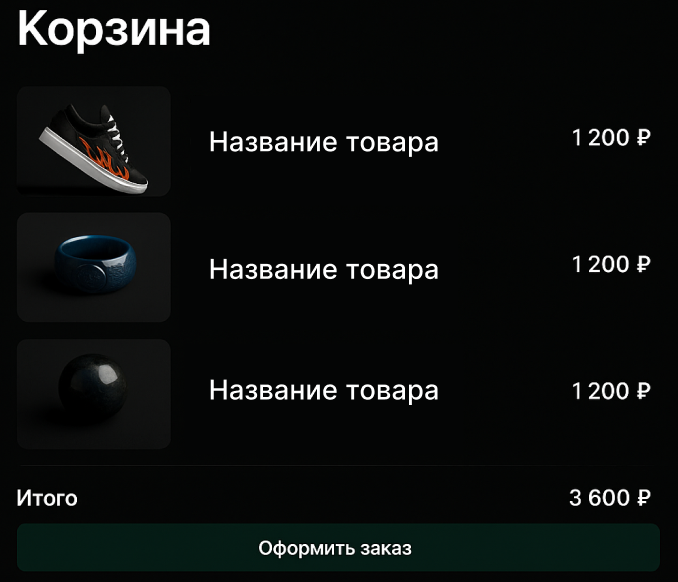


Рисунок 3.11 – Корзина

В корзине, представленной на рисунке 3.11 происходит формирование предварительного состава заказа и переход к оформлению.

Состав компонентов:

* список товаров в корзине: изображение, название, цена;
* итоговая стоимость;
* кнопка «Оформить заказ».

Функциональные требования:

* подсчёт общей стоимости;
* проверка на авторизацию;
* валидация перед переходом к этапу оформления.

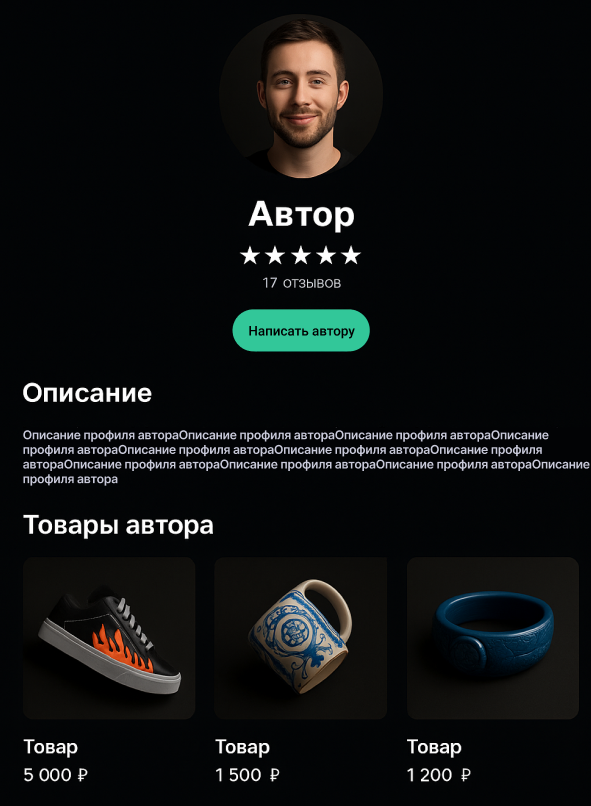


Рисунок 3.12 – Профиль автора

На рисунке 3.12 представлена страница профиля автора на маркетплейсе кастомных товаров. Дизайн выполнен в тёмной теме с акцентами бирюзово-зелёного цвета, подчёркивающими важные действия.

Верхняя карточка профиля:

* аватар автора – крупное фото в круглой рамке;
* имя – крупный заголовок «Автор»;
* рейтинг – 5 звёзд с подписью «17 отзывов»;
* кнопка «Написать автору» – выделена акцентным бирюзово-зелёным цветом, подчёркивает доступность для связи.

Описание автора:

* раздел «Описание» оформлен заголовком и абзацем текста (заполнен плейсхолдером), в котором автор рассказывает о себе, своих навыках и особенностях работы.

Товары автора:

* галерея товаров, представленных в виде карточек:
* каждая карточка включает изображение товара, название (плейсхолдер «Товар») и цену в рублях.
* витрина демонстрирует разнообразие стиля и диапазон цен, подчеркивая индивидуальный подход автора к каждому изделию.

Особенности реализации:

* единый визуальный стиль: тёмный фон, белый и акцентный текст, минимализм и визуальная строгость;
* страница фокусируется на личности автора и его уникальных товарах, что соответствует идеологии кастомного маркетплейса;
* кнопка «Написать автору» интегрирована с системой диалогов, где пользователь может уточнить детали, загрузить изображения и обсудить индивидуальный заказ.

Такое оформление способствует доверию к автору и удобству коммуникации, а визуальный стиль подчеркивает премиальность и креативность платформы.

Кроме того, интерфейс страницы продуман таким образом, чтобы не перегружать пользователя лишними элементами — всё внимание сосредоточено на авторе и его работах. Пространство используется эффективно: карточки товаров выстроены в аккуратную сетку, а визуальные акценты помогают быстро ориентироваться в информации. Благодаря продуманной типографике и контрастной цветовой палитре, контент остаётся легко читаемым даже при длительном просмотре.

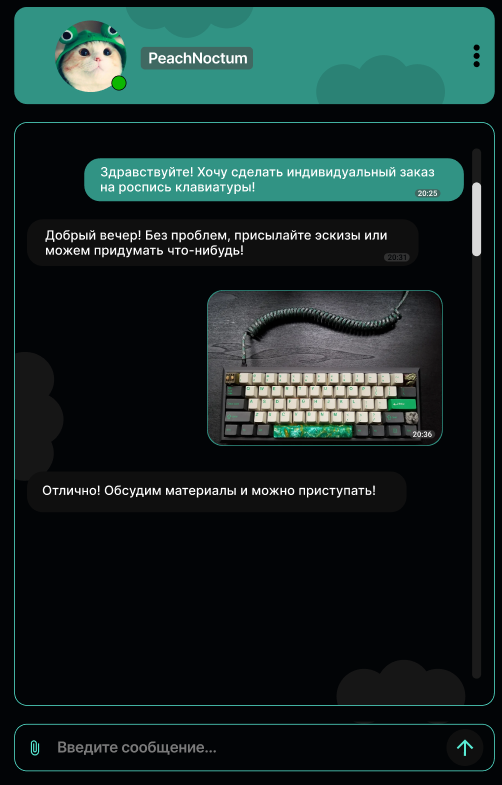


Рисунок 3.13 **–** Чат

Страница, изображённая на рисунке 3.13 реализует встроенную систему общения между покупателем и автором изделия. Это ключевой инструмент для согласования индивидуальных заказов, обмена материалами, визуалами и уточнения деталей изготовления.

Верхняя панель чата содержит:

* аватар автора;
* имя (никнейм) пользователя (например, PeachNoctum);
* индикатор статуса (в сети / не в сети);
* иконку действий (три точки), предполагающую доступ к дополнительным функциям (жалоба, удалить чат, открыть профиль).

Основная область переписки:

* сообщения от покупателя отображаются справа и выделены бирюзово-зелёным цветом;
* сообщения автора – слева, на тёмном фоне, с метками времени;
* в диалоге может присутствовать изображение (например, эскиз товара или фотоматериала), вставленное как вложение, с отображением времени отправки.

Нижняя панель ввода сообщения включает:

* поле для ввода текста (placeholder: «Введите сообщение...»);
* иконку вложения файла (скрепка);
* кнопку отправки (иконка стрелки), стилизованную под визуальный стиль всей платформы.