Технический проект

Информационная платформа для мониторинга финансовых активов с интегрированным ИИ-ассистентом

Функциональная структура продукта

Функциональная структура описывает основные процессы и архитектурные решения. Использован структурный подход с диаграммами.

Функциональная структура платформы подчеркивает ее модульную архитектуру и динамику взаимодействия элементов, обеспечивая гибкость и эффективность веб-приложения. Основные модули включают:

– контроль доступа и пользовательских профилей, регистрация через email или Google, авторизация с восстановлением пароля, кастомизация аватара и ника;

– настройку персонализированного дашборда виджеты для мониторинга активов и новостей с drag-and-drop;

– визуализацию и анализ графиков поддержка тайм фреймов, индикаторов типа MA, RSI, Bollinger Bands, реал-тайм обновления via WebSocket;

– работу с новостями поиск по ключам или темам, фильтры, добавление комментариев с аватарами и реакциями;

– интеграцию ИИ-ассистента чат для запросов анализа, прогнозов трендов и рекомендаций на основе истории;

– административные инструменты управление аккаунтами, настройка API-ключей, обновление моделей ИИ, модерация контента;

– внешние интеграции и социальные элементы API для данных, сообщество для обсуждений активов.

Такая организация способствует независимости модулей, упрощает обновления и масштабирование, а также повышает пользовательский опыт через seamless интеграцию данных. Диаграммы иллюстрируют потоки и связи между компонентами, подчеркивая аспекты финансового мониторинга.

Контекстная диаграмма представлена на рисунке 1.

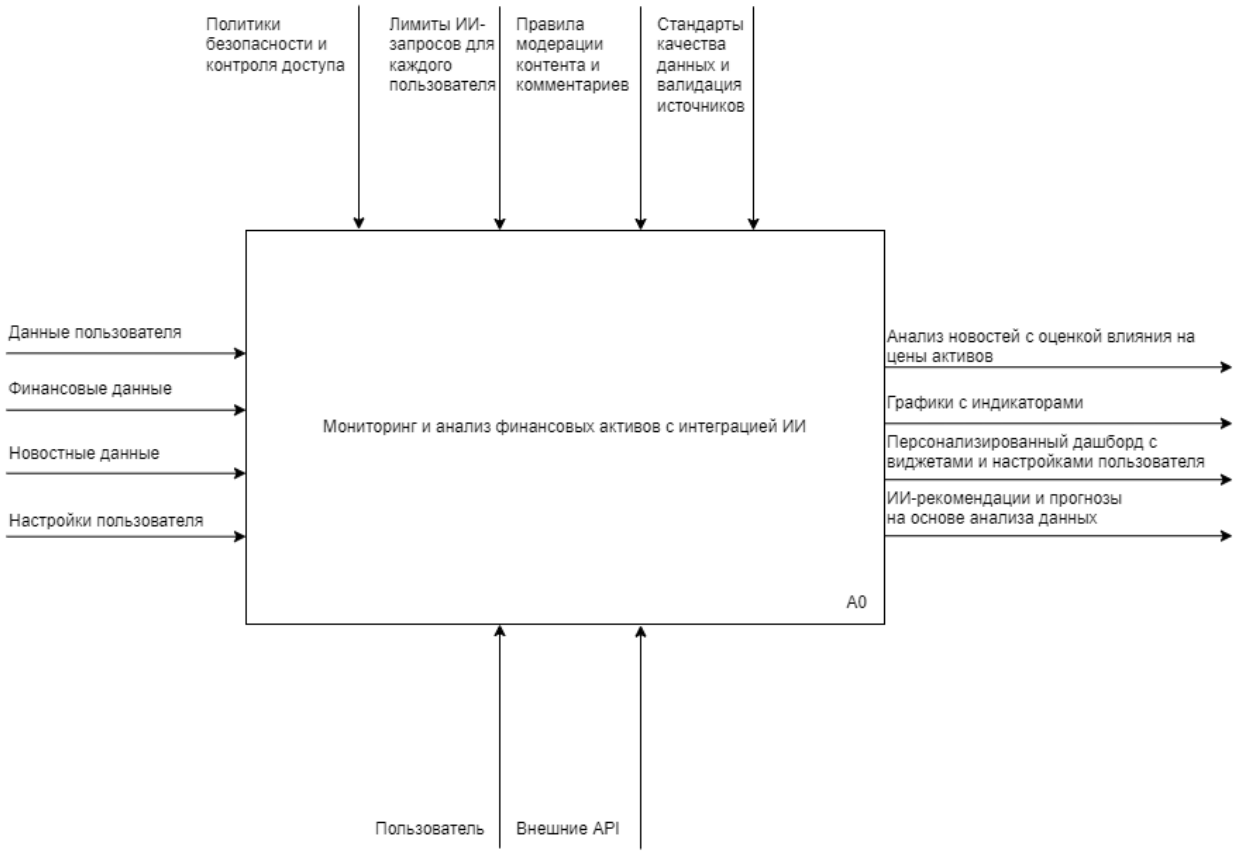

Рисунок 1 – Контекстная диаграмма

Диаграмма демонстрирует границы системы, показывая, какие внешние сущности взаимодействуют с платформой и какие основные потоки данных обеспечивают мониторинг, визуализацию и интеллектуальный анализ финансовых активов без детализации внутренних подпроцессов.

Контекстная диаграмма декомпозиция представлена на рисунке 2.

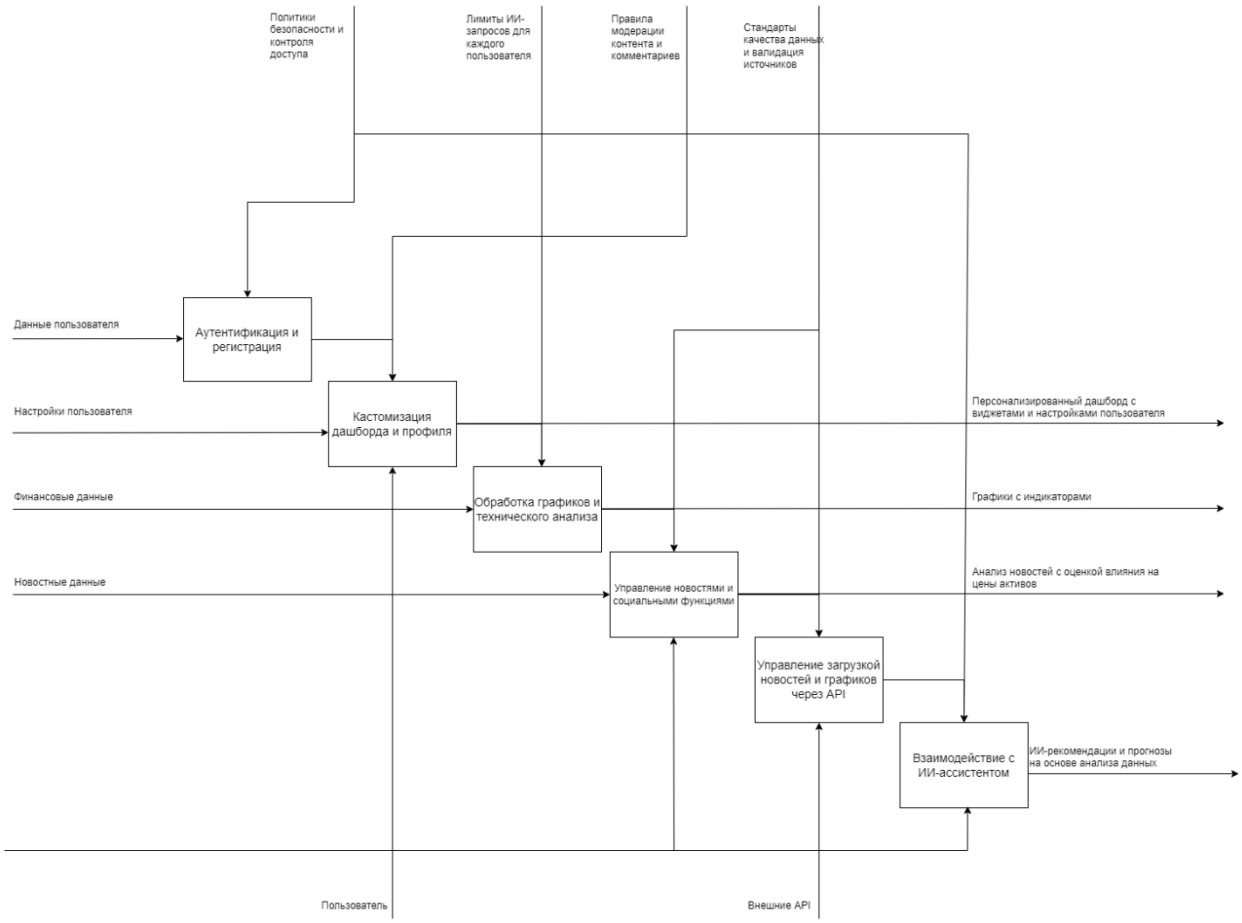

Рисунок 2 – Контекстная диаграмма декомпозиция

Декомпозиция иллюстрирует логическую последовательность и параллельность выполнения операций, подчёркивая центральную роль ИИ-ассистента (А6) как финального этапа интеллектуальной обработки данных, а также показывает, как аутентификация и кастомизация являются обязательными начальными шагами для персонализации всего опыта пользователя.

IDEF3 диаграмма представлена на рисунке 3.

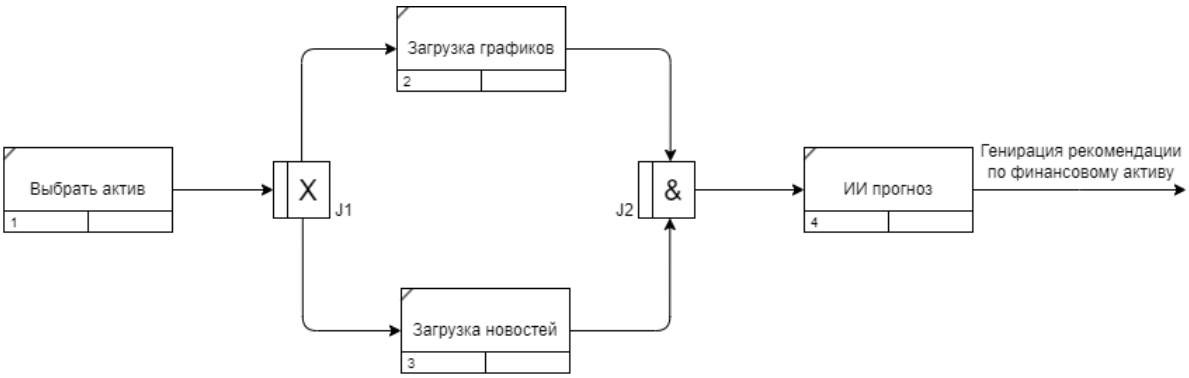

Рисунок 3 – IDEF3 диаграмма

Диаграмма показывает последовательный поток, выбор актива ветвится на параллельные пути либо график актива, либо открыть новость, сливается в ИИ для комплексного анализа, завершая прогнозом.

Спецификация процессов

Диаграмма прецедентов представлена на изображении 4.

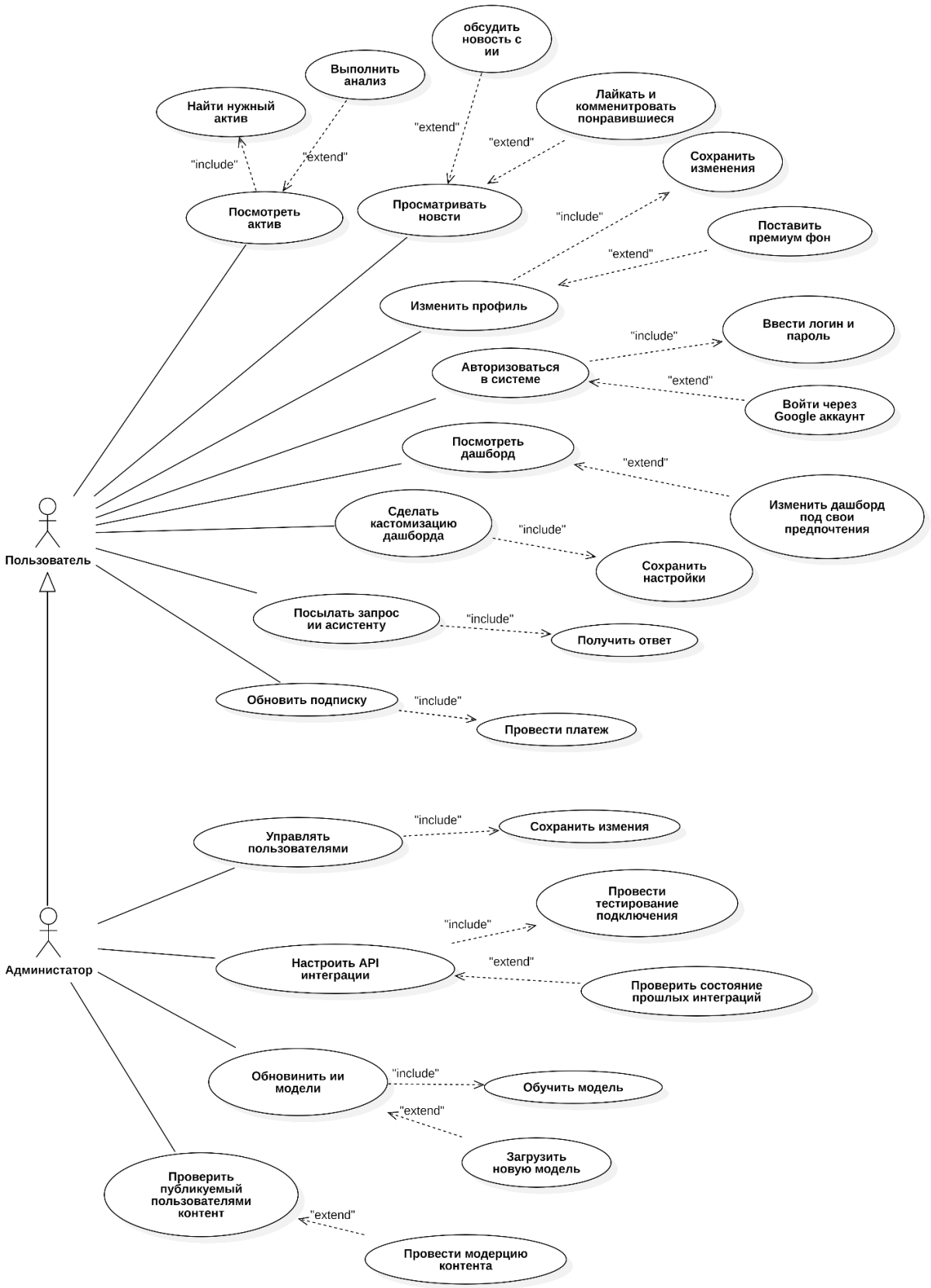

Рисунок 4 – Диаграмма прецендентов

Диаграмма последовательности UML представлена на рисунке 5.

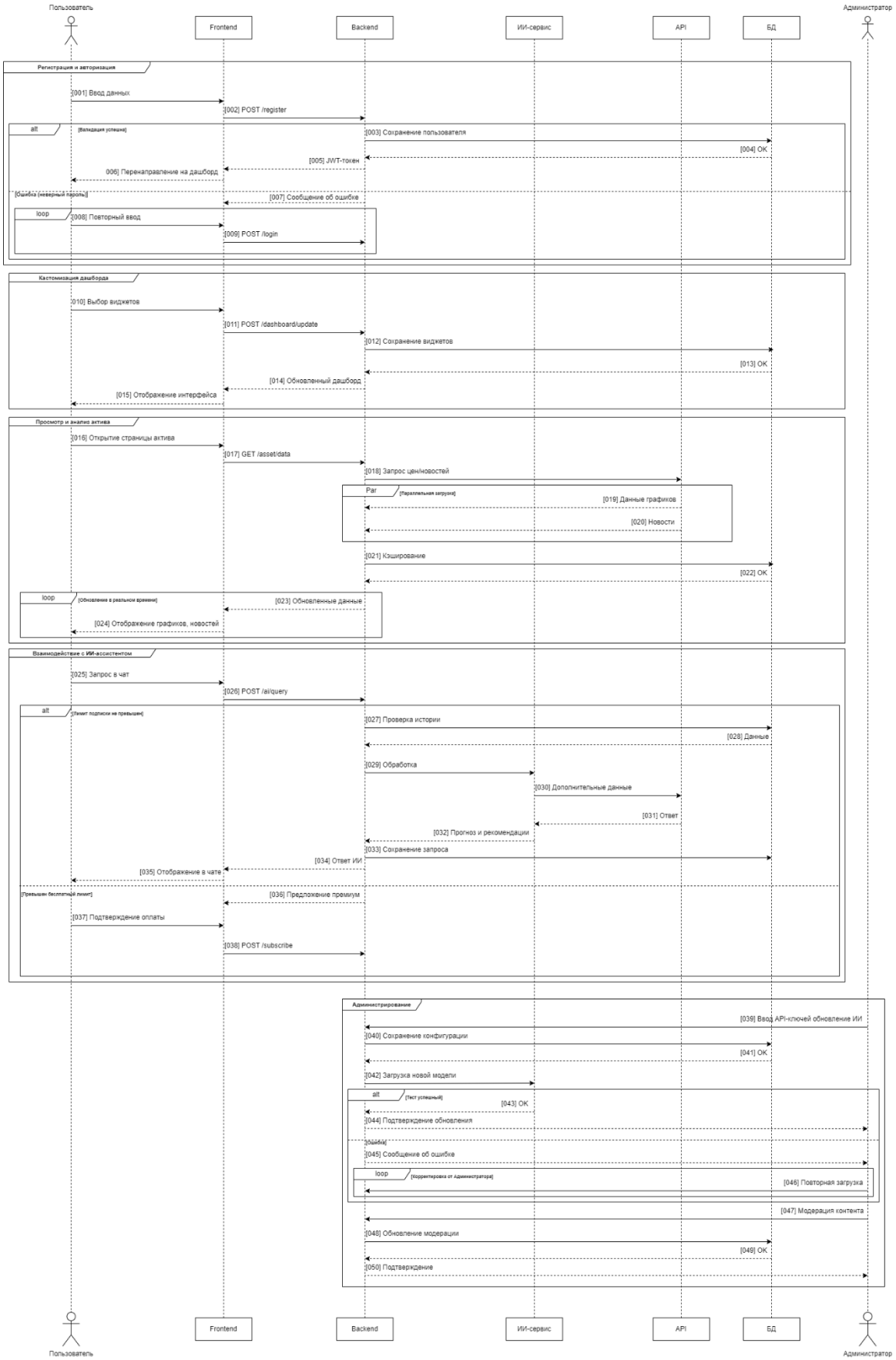

Рисунок 5 – Диаграмма последовательности UML

Структура хранимой информации

ER диаграмма представлена на рисунке 6.

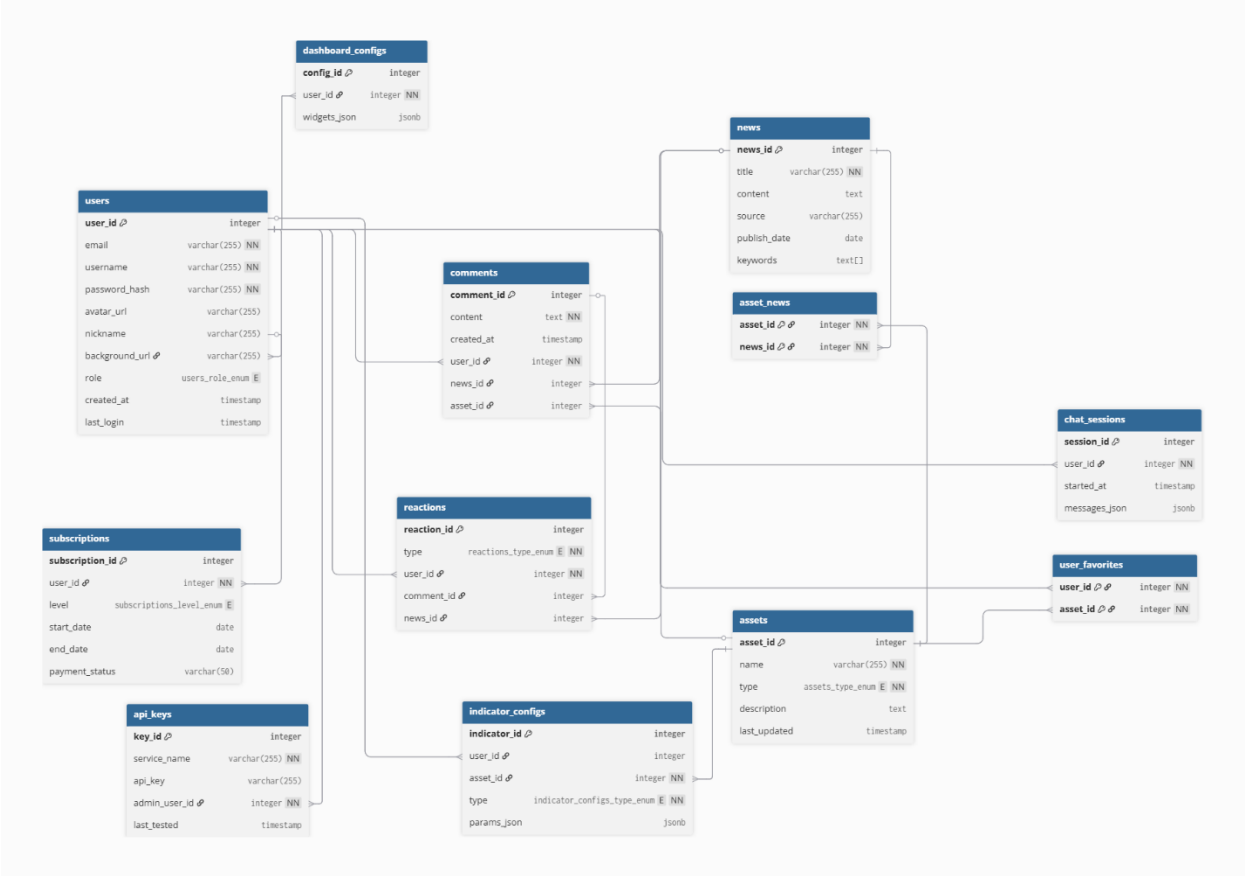

Рисунок 6 – ER-Диаграмма

Диаграмма представляет структуру базы данных платформы для мониторинга финансовых активов с интегрированным ИИ-ассистентом. Все таблицы связаны через внешние ключи, обеспечивая целостность данных и поддержку ключевых бизнес-процессов.

Центральная сущность — users

Таблица users хранит информацию о пользователях системы.

user_id — уникальный идентификатор пользователя, первичный ключ.

email — электронная почта, уникальная и обязательная.

username — имя пользователя, уникальное и обязательное.

password_hash — хэш пароля для безопасной аутентификации.

avatar_url, nickname, background_url — данные для персонализации профиля.

role — роль пользователя: user или admin.

created_at — время создания аккаунта.

last_login — время последнего входа.

От таблицы users исходят связи ко всем основным модулям платформы.

Подписки — subscriptions

Таблица subscriptions управляет монетизацией.

subscription_id — первичный ключ.

user_id — ссылка на пользователя.

level — уровень подписки: free или premium.

start_date, end_date — период действия подписки.

payment_status — статус платежа.

Связь: один пользователь — одна подписка.

Активы — assets

Таблица assets содержит данные о финансовых инструментах.

asset_id — первичный ключ.

name — название актива, например BTC/USD.

type — тип: stock, crypto, currency, index.

description — описание.

last_updated — время последнего обновления.

Активы используются в графиках, дашбордах и анализе.

Новости — news

Таблица news хранит новостные статьи.

news_id — первичный ключ.

title — заголовок.

content — текст новости.

source — источник.

publish_date — дата публикации.

keywords — массив ключевых слов для поиска.

Связь активов и новостей — asset_news

Таблица asset_news реализует отношение многие-ко-многим между активами и новостями.

asset_id и news_id — составной первичный ключ. Позволяет связывать новости с влияющими на них активами.

Комментарии — comments

Таблица comments хранит пользовательские комментарии.

comment_id — первичный ключ.

content — текст комментария.

created_at — время создания.

user_id — автор комментария.

news_id — к какой новости относится (может быть пустым).

asset_id — к какому активу относится (может быть пустым).

Один комментарий может относиться либо к новости, либо к активу, либо быть общим.

Реакции — reactions

Таблица reactions фиксирует реакции пользователей.

reaction_id — первичный ключ.

type — тип: like или dislike.

user_id — кто поставил реакцию.

comment_id — к какому комментарию (опционально).

news_id — к какой новости (опционально).

Реакции могут быть к комментариям или новостям.

Конфигурация дашборда — dashboard_configs

Таблица dashboard_configs сохраняет настройки персонального дашборда.

config_id — первичный ключ.

user_id — владелец дашборда.

widgets_json — JSON с расположением и содержимым виджетов (активы, новости, графики).

Сессии чата с ИИ — chat_sessions

Таблица chat_sessions хранит историю диалогов с ИИ-ассистентом.

session_id — первичный ключ.

user_id — пользователь.

started_at — начало сессии.

messages_json — массив сообщений: запросы пользователя и ответы ИИ.

Используется для персонализации рекомендаций и дообучения модели.

Ключи API — api_keys

Таблица api_keys управляется администраторами.

key_id — первичный ключ.

service_name — название сервиса (например, CoinGecko).

api_key — зашифрованный ключ.

admin_user_id — администратор, добавивший ключ.

last_tested — время последней проверки.

Конфигурация индикаторов — indicator_configs

Таблица indicator_configs хранит настройки технических индикаторов.

indicator_id — первичный ключ.

user_id — пользователь (может быть пустым для глобальных настроек).

asset_id — актив, к которому применяется.

type — тип индикатора: MA, RSI, Bollinger.

params_json — параметры (например, период).

Избранное — user_favorites

Таблица user_favorites хранит избранные активы пользователей.

user_id и asset_id — составной первичный ключ. Позволяет быстро выводить избранное на дашборд.

Связи

Пользователь имеет одну подписку, одну конфигурацию дашборда, множество сессий чата, комментариев, реакций и избранных активов.

Активы связаны с новостями через таблицу asset_news.

Комментарии и реакции могут относиться к новостям или активам.

Администраторы управляют API-ключами и глобальными индикаторами.

ИИ-ассистент использует историю чатов и настройки дашборда для персонализации.

Архитектурно-структурное решение

Платформа построена на многоуровневой клиент-серверной архитектуре с монолитным сервером, разделённым на логические слои для обеспечения масштабируемости, безопасности и удобства разработки. Архитектура следует принципам разделения ответственности клиентская часть отвечает исключительно за пользовательский интерфейс и взаимодействие с пользователем, серверная часть — за всю бизнес-логику, обработку данных, интеграции и искусственный интеллект.

Взаимодействие между компонентами осуществляется через:

- HTTP/HTTPS (RESTful API);
- WebSocket;
- внутренние вызовы функций внутри бэкенда.

Общая структура архитектуры

– Клиентский уровень (Frontend) Одностраничное приложение (SPA) на React 18+ с TypeScript, Material-UI, Redux Toolkit. Отвечает только за отображение и локальную логику (валидация форм, drag-and-drop, кэширование черновиков в localStorage). Все конфиденциальные данные и вычисления — строго на сервере.

– Серверный уровень (Backend) — монолитный многослойный сервер на Python 3.11+ с FastAPI Состоит из четырёх логических слоёв, реализованных в одном приложении:

– Уровень представления (API Layer) REST- и WebSocket-эндпоинты, валидация (Pydantic), аутентификация (JWT + OAuth Google), CORS, rate-limiting.

– Уровень бизнес-логики (Business Logic Layer) Обработка всех запросов пользователя, расчёт индикаторов (NumPy/Pandas/TA-Lib),

монетизация, модерация, интеграция с внешними API (Alpha Vantage, NewsAPI), вызовы модуля ИИ, постановка асинхронных задач в Celery.

– Модуль искусственного интеллекта. Полностью интегрированная часть серверного уровня (пакет app/ai/ или services/ai/). Никаких внешних микросервисов и сетевых вызовов — только обычные внутренние функции Python и Celery-задачи. Использует Hugging Face Transformers / LangChain + fine-tuned GPT-подобную модель + TA-Lib/Scikit-learn для анализа графиков. Персонализация по истории чата из БД, лимиты запросов, кэширование ответов в Redis, обучение/обновление модели через админ-панель.

– Уровень доступа к данным (Data Access Layer) SQLAlchemy (PostgreSQL) + Redis. Обеспечивает транзакции, репликацию и кэширование.

Уровень данных

– PostgreSQL — персистентное хранение (профили, подписки, дашборды, история чата, комментарии).

– Redis — кэш цен в реальном времени, сессии WebSocket, очередь Celery, кэш ответов ИИ.

Детализация клиентской части

– Клиентская часть реализована на React 18+ с TypeScript, Material-UI и Redux Toolkit. Отвечает за:

– отображение дашборда, графиков (Chart.js/Recharts), чата с ИИ, новостей и профилей;

– локальную валидацию и обработку событий;

– отправку запросов на сервер.

– Токены хранятся в HttpOnly cookies, конфиденциальные вычисления отсутствуют.

Детализация серверной части

– Сервер — монолитный многослойный FastAPI-приложение:

– API Layer: REST- и WebSocket-эндпоинты, Pydantic, Uvicorn, JWT + OAuth.

– Business Logic Layer: вся логика приложения + прямые вызовы модуля ИИ.

– Модуль ИИ логически выделен в отдельный пакет, но физически является частью того же процесса.

– Data Access Layer: SQLAlchemy + Redis.

– Развёртывание: Docker + Nginx (reverse-proxy), горизонтальное scaling за load balancer'ом.

Взаимодействие компонентов

– Клиент ↔ Сервер: HTTP/REST + WebSocket.

– Сервер ↔ Внешние API: httpx + ретраи + кэш.

– Бизнес-логика ↔ ИИ: обычные вызовы Python-функций (без HTTP/gRPC).

– ИИ ↔ БД/Redis: через Data Access Layer (история чата, персонализация).

– Безопасность: JWT, rate-limiting, санитизация входов ИИ, дисклеймер в ответах.

Таким образом, модуль искусственного интеллекта полностью встроен в серверную часть и не является отдельным сервисом — это гарантирует минимальную задержку в чате, простоту деплоя и единую систему безопасности. При росте нагрузки его можно будет вынести в микросервис практически без изменений клиентского кода.

Модуль искусственного интеллекта (ИИ-ассистент) реализован как подсервис серверной части на базе предобученных моделей машинного обучения, интегрированных через библиотеки Hugging Face Transformers или LangChain для обработки естественного языка (NLP). ИИ предназначен для анализа запросов пользователей, генерации прогнозов трендов, технического анализа графиков и персонализированных рекомендаций на основе истории взаимодействий.

- Технологии и модели: Основная модель — fine-tuned версия GPT-like. Для анализа графиков используется комбинация с библиотеками TA-Lib (для индикаторов) и Scikit-learn (для предсказаний, например, регрессия на основе исторических данных). Обучение модели происходит оффлайн на датасетах из Kaggle (финансовые данные, новости) с использованием PyTorch; обновление модели администратором через админ-панель (загрузка новых весов).

- Функциональность: ИИ обрабатывает запросы в чате: парсинг текста (NLP для извлечения ключевых слов, например, "прогноз BTC"), интеграция с данными (запрос цен из API, истории чата из БД), генерация ответа (текст + визуализации, если нужно). Персонализация: модель учитывает историю запросов (хранится в chat_sessions таблице БД) для рекомендаций (например, "На основе ваших запросов по крипте, вот анализ ETH"). Лимиты: для бесплатных пользователей — 5 запросов/день (проверяется в бизнес-логике); премиум — безлимит.

- Интеграция: ИИ вызывается через внутренний API-эндпоинт (/ai/internal/query), асинхронно с Celery для тяжелых задач (например, обучение на пользовательских данных). Данные передаются в JSON-формате; ответы кэшируются в Redis для повторных запросов.

- Обучение и улучшение: Модель самообучается на агрегированных данных пользователей (анонимизированных) через периодические патчи; администратор может обновлять модель в админ-панели, загружая новые

датасеты. Метрики качества: точность прогнозов > 70% на тестовых данных, измеряемая через backtesting на исторических ценах.

- Ограничения и риски: ИИ не дает финансовых советов (дисклеймер в ответах), зависит от качества внешних API; риски минимизированы через fallback на базовые ответы при ошибках.

Интерфейсы

Пользовательский интерфейс разработан с учетом нужд трейдеров и администраторов, подчеркивая удобство мониторинга и анализа. Основные принципы, реализованные в системе взаимодействия:

– Приоритет на реальном времени: элементы интерфейса акцентируют обновления цен и новостей, минимизируя задержки для оперативного принятия решений;

– Персонализация контента: дашборд позволяет сравнивать активы и новости в одном окне, облегчая контроль трендов и изменений;

– Интуитивное управление: пользователи напрямую настраивают виджеты, индикаторы и профили без дополнительных переходов;

– Оптимизация workflow: процесс от регистрации до анализа построен последовательно — вход в систему, настройка панели, запрос ассистента, публикация комментариев.

Прототип экранной формы входа в аккаунт представлен на рисунке 5.4.1.

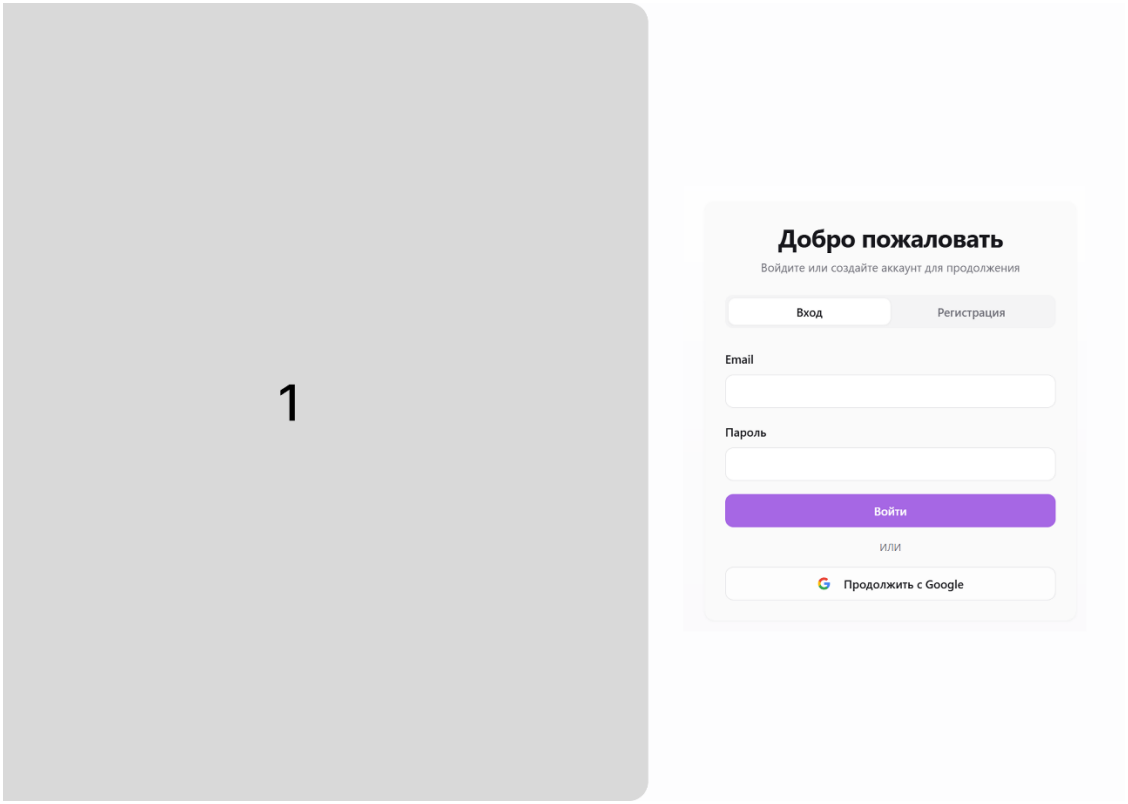

Рисунок 5.4.1 – Прототип экранной формы входа в аккаунт

Прототип экранной формы регистрации аккаунта представлен на рисунке 5.4.2.

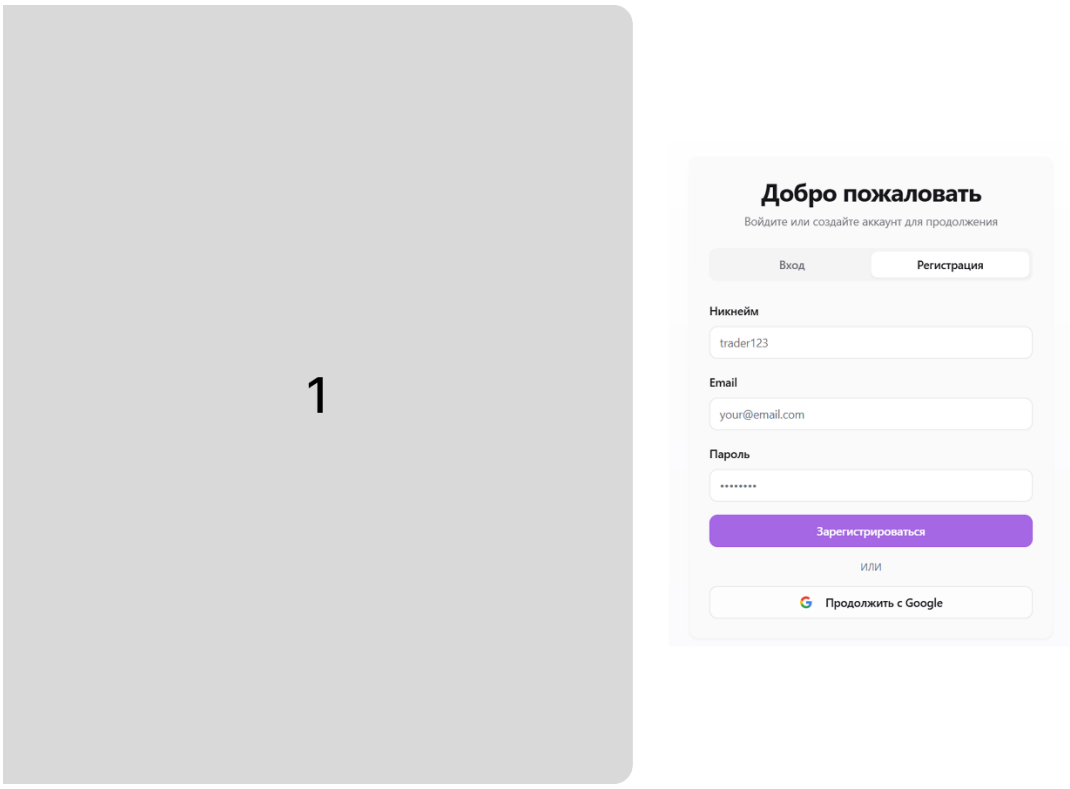

Рисунок 5.4.2 – Прототип экранной формы регистрации аккаунта

На странице регистрации предоставляются возможности:

создавать новый аккаунт через простую форму с полем "Никнейм" для ввода имени пользователя длиной от 3 символов с красной подсветкой ошибки "Слишком коротко" при неверном формате;

вводить пароль в поле «Пароль» типа пароль с валидацией на 8+ символов включая цифру с жёлтой подсветкой слабого пароля;

отправлять форму кнопкой «Зарегистрироваться», после чего приходит письмо подтверждения на почту с последующим перемещением на дашборд;

регистрироваться через Google одной кнопкой с иконкой Google, которая автоматически берёт email, имя и фото из аккаунта Google без ввода данных;

переключиться на форму «Входа» отправить форму на сервер для проверки почты и хэш пароля, после войти уже в существующий аккаунт.

После «Регистрации» или «Входа» пользователь может просмотреть свой профиль. Прототип экранной формы профиля представлен на рисунке 5.4.3.

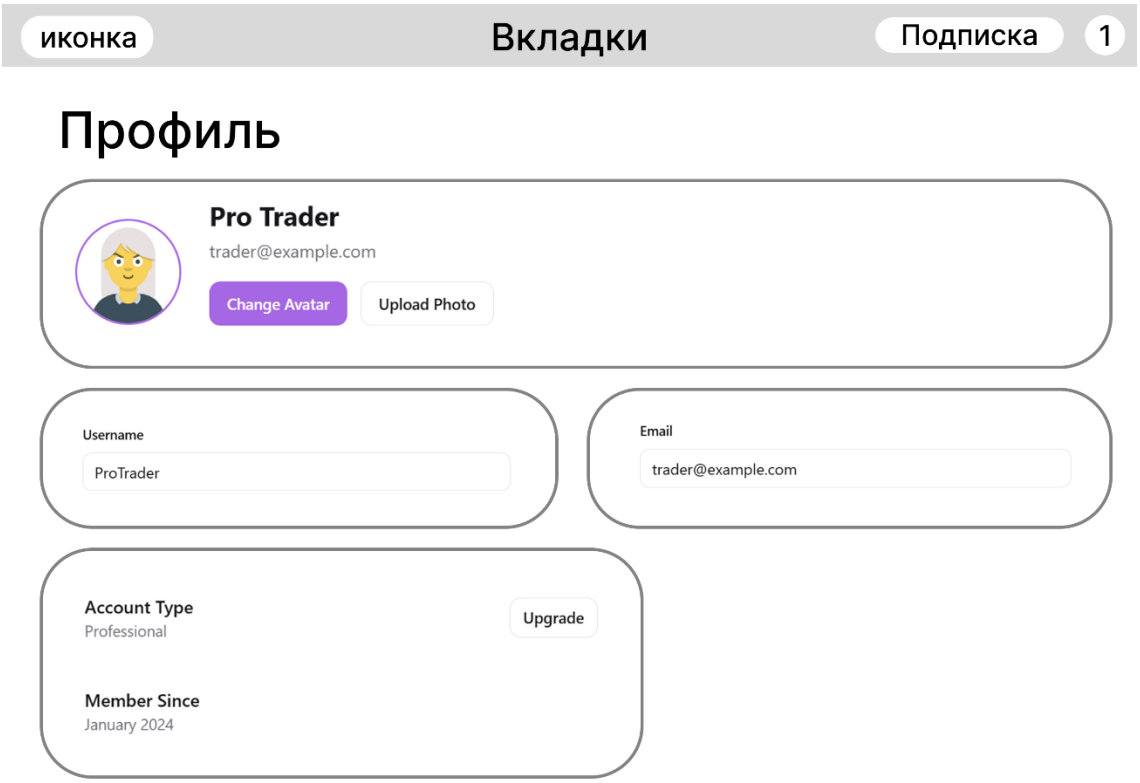

Рисунок 5.4.3 – Прототип экранной формы профиля

На странице профиля предоставляются возможности:

просматривать текущий аватар пользователя в круглой рамке;

редактировать аватар через кнопку «Изменить аватар», которая открывает проводник с автоматическим обрезом фото до квадрата 200x200 пикселей;

изменять никнейм в поле «Никнейм» с текущим значением путём ввода текста длиной 3-20 символов с проверкой уникальности в реальном времени и красной подсветкой ошибки при дубликате;

просматривать email в поле «Почта» только для чтения с возможностью копирования одним кликом;

видеть статус подписки аккаунта;

отображать дату регистрации как неизменяемую информацию;

сохранять все изменения кнопкой «Сохранить изменения».

Как только пользователь настроил свой профиль, он может нажать на иконку сайта и переместиться в главное меню, где у него отобразится кастомный дашборд, который пользователь может менять по своему усмотрению.

Прототип экранной формы кастомного дашборда представлен на рисунке 5.4.4.

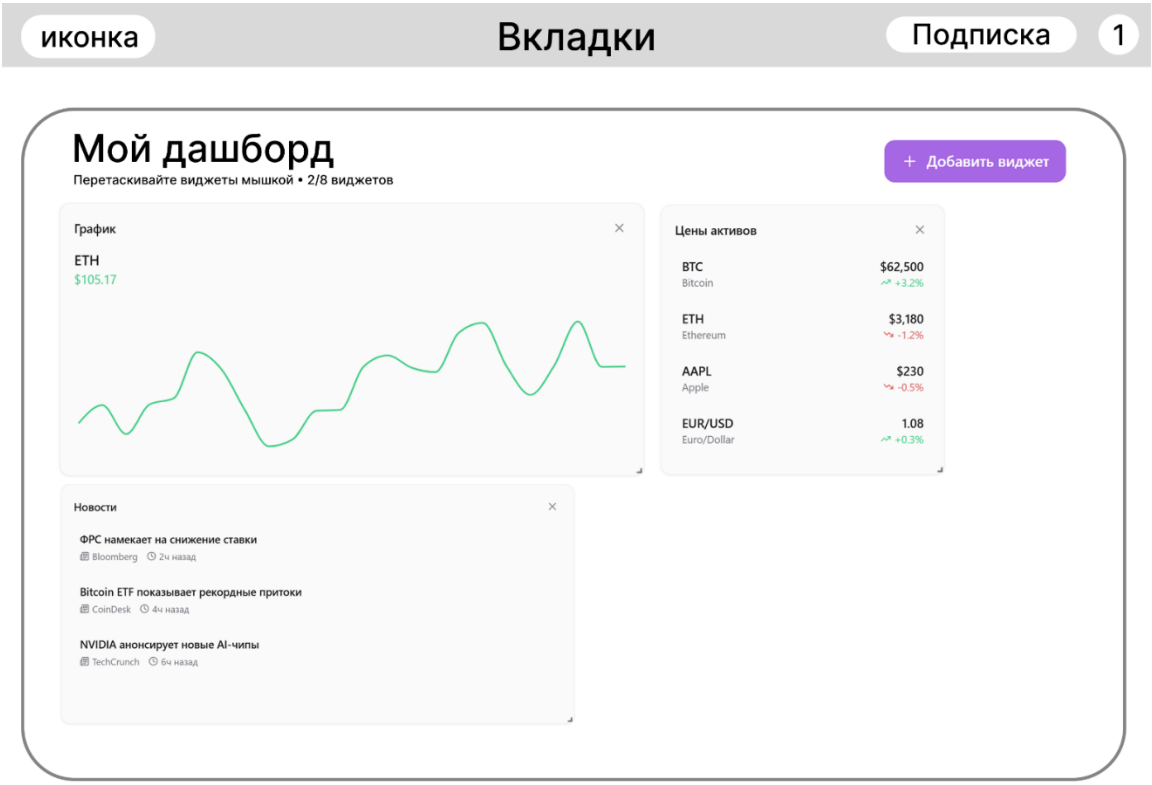

Рисунок 5.4.4 – Прототип экранной формы кастомного дашборда

На странице кастомного дашборда предоставляются возможности:

строить персональный рабочий стол перетаскиванием виджетов мышкой в сетке с автоматическим сохранением в облаке;

добавлять новые виджеты кнопкой «+ Добавить виджет»;

удалять виджет в любой момент времени.

После просмотра дашборда пользователь может пролистнуть страницу вниз тогда он увидит таблицу с обзором рынка, а также слева у него появится панель перемещения между разделами в виде таблетки.

Прототип экранной формы таблицы обзора рынка представлен на рисунке 5.4.5.

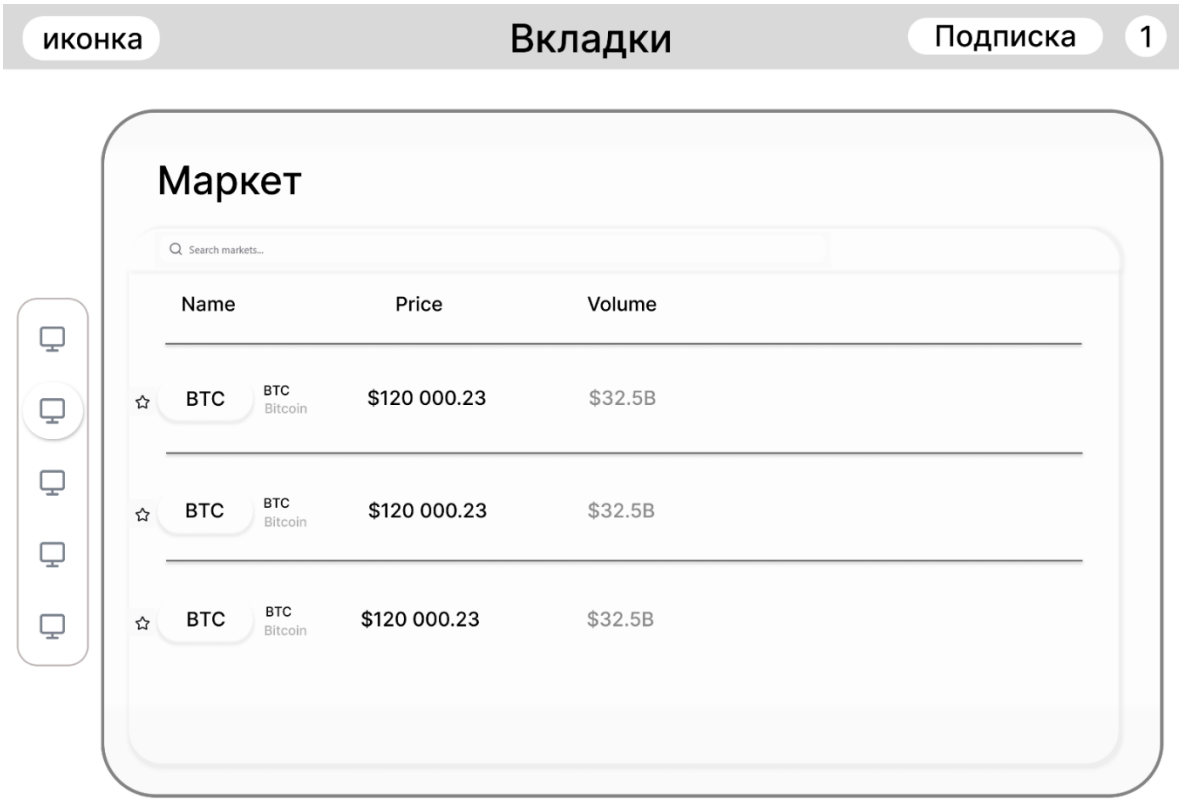

Рисунок 5.4.5 – Прототип экранной формы таблицы обзора рынка

На странице таблицы обзора рынка предоставляются возможности:

быстро сканировать все активы в табличном виде с колонками «Name» для названия с чекбоксом выбора и звёздочкой фаворитов, «Price» с цветным процентом изменения (зелёный вверх, красный вниз), «Volume» в долларах. Это минимальный набор параметров которые могут использоваться, в ходе разработки данные могут поменяться;

искать активы в строке поиска «Поиск» с фильтрацией по названию или символу;

сортировать строки кликом на заголовок колонки с стрелкой направления;

отмечать фавориты звёздочкой, которая добавляет актив в дашборд одним кликом с анимацией заполнения.

Пользователь, пролистав ещё ниже наткнется на страницу с новостями.

Прототип экранной формы страницы новостей представлен на рисунке 5.4.6.

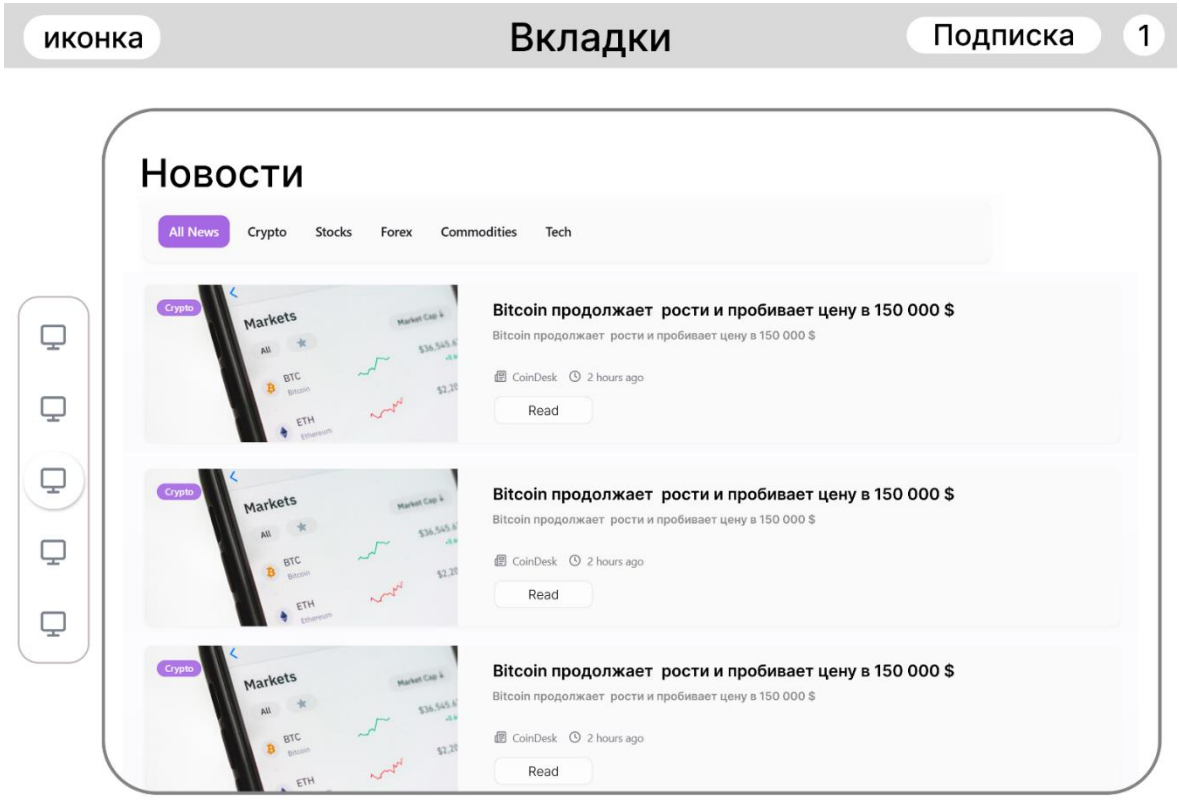

Рисунок 5.4.6 – Прототип экранной формы страницы новостей

На странице новостей предоставляются возможности:

фильтровать ленту новостей;

Нажатие на кнопку "Прочитать полностью" для открытия полной статьи с комментариями комьюнити;

отмечать фавориты звёздочкой в каждой карточке для добавления в дашборд;

искать новости строкой поиска сверху.