3 ПРОЕКТИРОВАНИЕ

3.1 Проектирование структур хранения данных

Данные приложения будут храниться в реляционной БД. В качестве СУБД была выбрана MSSQL. Структура БД составляется на основе концептуальной модели, представленной в на рисунке 1.1.

В реляционной базе данных данные хранятся в таблицах. Каждая строка данных в таблице идентифицируется уникальным “ключом”, который называется первичным ключом (PK).

В таблице 3.1 определены первичные и внешние ключи для отношений.

Таблица 3.1 – Первичные и внешние ключи отношений

№	Название таблицы	Первичный ключ	Внешний ключ
1	Board	id	List.boardId
2	List	id	Card.listId
3	Card	id	-
4	AuditLog	id	-
5	User	id	UserOrgs.usrId, AuditLog.userId
6	Org	id	UserOrg.orgId, AuditLog.orgId
7	ENTITY_TYPE	id	AuditLog.entityId
8	UserOrgs	id	-

В таблицах 3.2 – 3.5 представлены поля каждой из таблиц базы данных.

Таблица 3.2 – Поля таблицы «Board»

Название поля	Тип данных поля	Описание
id	PRIMARY KEY AUTOINCREMENT	Уникальный идентификатор доски
orgId	String	ID организации (берется из Clerk)
title	String	Название доски
imageId	String	ID картинки (берется из Unsplash API)
imageThumbUrl	String	Маленька картинка (берется из Unsplash API)
imageFullUrl	String	Большая картинка (берется из Unsplash API)
imageUserName	String	Имя пользователя, создавшего картинку (берется из Unsplash API)
imageLinkHTML	String	Ссылка на страницу создателя картинки (берется из Unsplash API)
createdAt	DateTime (Default value now())	Дата создания доски
updatedAt	DateTime	Дата обновления доски
lists	Array	Список листов доски

Таблица 3.3 – Поля таблицы «List»

Название поля	Тип данных поля	Описание
id	PRIMARY KEY AUTOINCREMENT	Уникальный идентификатор листа задач
title	String	Название листа
order	Int	Положение листа на странице
boardId	FOREIGN KEY	Внешний ключ для связи с таблицей “Board”
cards	Array	Список задач в листе
createdAt	DateTime (Default value now())	Дата создания листа
updatedAt	DateTime	Дата обновления листа

Таблица 3.4 – Поля таблицы «Card»

Название поля	Тип данных поля	Описание
id	PRIMARY KEY AUTOINCREMENT	Уникальный идентификатор задачи
title	String	Название задачи
order	Int	Положение листа на странице
description	String	Описание задачи
cards	Array	Список задач в листе
createdAt	DateTime (Default value now())	Дата создания задачи
updatedAt	DateTime	Дата обновления задачи

Таблица 3.5 – Поля таблицы «AuditLog»

Название поля	Тип данных поля	Описание
id	PRIMARY KEY AUTOINCREMENT	Уникальный идентификатор лога
orgId	String	Уникальный идентификатор организации (берется из Clerk)
action	Array	Перечисление действий (CREATE, UPDATE, DELETE)
entityId	Array	ID перечисления “Entity”. Возможные значения (BOARD, LIST, CARD)
entityType	Array	Тип
userId	String	Уникальный идентификатор пользователя (берется из Clerk)
userImage	String	Картинка пользователя (берется из Clerk)
userName	String	Имя пользователя (берется из Clerk)
createdAt	DateTime (Default value now())	Дата создания лога
updatedAt	DateTime	Дата обновления лога

Следует отметить, что сущности ACTION и ENTITY_TYPE объявлены как перечисления. Подробнее об перечислениях в фреймворке Prisma, можно

почитать в их документации [7].

3.2 Разработка архитектуры программной системы

Физическое представление приложения не может быть полным, если отсутствует информация о ее топологии и необходимых аппаратных средствах. Помимо сведений о компьютерах, обрабатывающих информацию, необходимо определить, как будет осуществляться связь между ними и какие дополнительные ресурсы (принтеры, модемы, маршрутизаторы и т.д) должны быть задействованы.

Сложные системы могут реализовываться на различных вычислительных платформах и технологиях доступа к базам данных.

Интеграция приложения с интернетом определяет необходимость решения дополнительных вопросов при проектировании, таких как обеспечение безопасности и доступности информации для пользователей.

Доступ и манипулирование данными в рамках двух- или трехуровневой технологии «клиент-сервер» также требуют размещения больших БД в различных сегментах сети, их резервного копирования, архивирования, кэширования для обеспечения необходимой производительности системы в целом.

Диаграмма развертывания предназначена для визуализации элементов и компонентов системы, существующих на этапе ее исполнения, к которым относятся исполняемые файлы, динамические библиотеки, таблицы БД. Диаграмма развертывания для разрабатываемого web-приложения «Командный менеджер задач» представлена на рисунке 3.2.

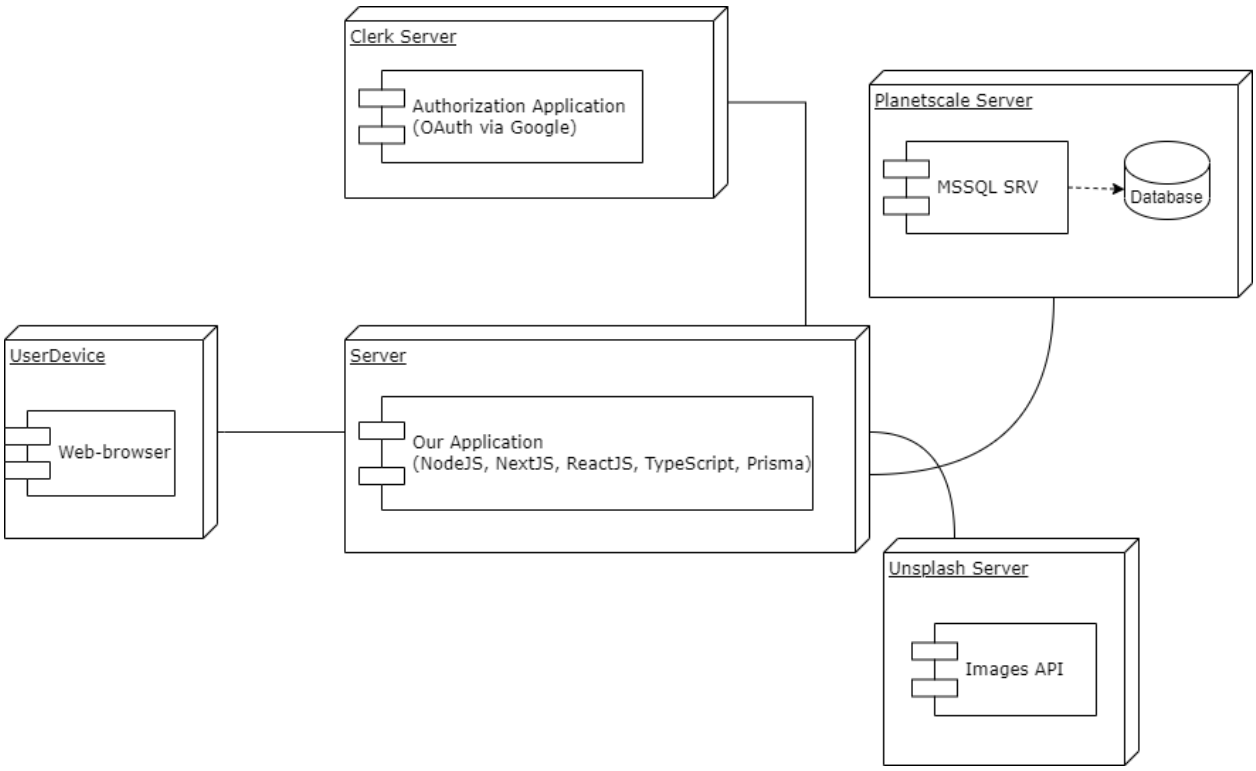

Рисунок 3.1 – Архитектура разработанного web-приложения

Приложение находится на сервере, в свою очередь основная БД, находится на другом сервер, что позволяет обеспечить сохранность данных, при выходе из строя сервера с самим приложением (БД может располагаться где угодно, даже локально на том же сервере). Так же сервер приложения использует сервис «Clerk», который позволяет удобно и быстро разработать систему регистрации и авторизации пользователей, по протоколу OAuth. В моем случае, OAuth протокол настроен на авторизацию и регистрацию пользователей через Google (настройки в сервисе «Clerk»). Пользователи могут получить доступ к приложению через сеть интернет со своих устройств, будь то ПК или смартфон.

3.3 Разработка архитектуры компонентов программной системы

Концепция паттерна MVC (model-view-controller) предполагает разделение приложения на три компонента.

Контроллер (controller) интерпретирует действия пользователя,

оповещая модель о необходимости изменений. Представляет собой класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде преставления.

Представление (view) – это визуальная часть или пользовательский интерфейс приложения. HTML – страница, которую пользователь видит, зайдя на сайт, либо запустив приложение.

Модель (model) представляет данные и реагирует на команды контроллера, изменяя свое состояние. Представляет собой класс, описывающий логику используемых данных.

Общая схема взаимодействия этих компонентов представлена на рисунке 3.2.

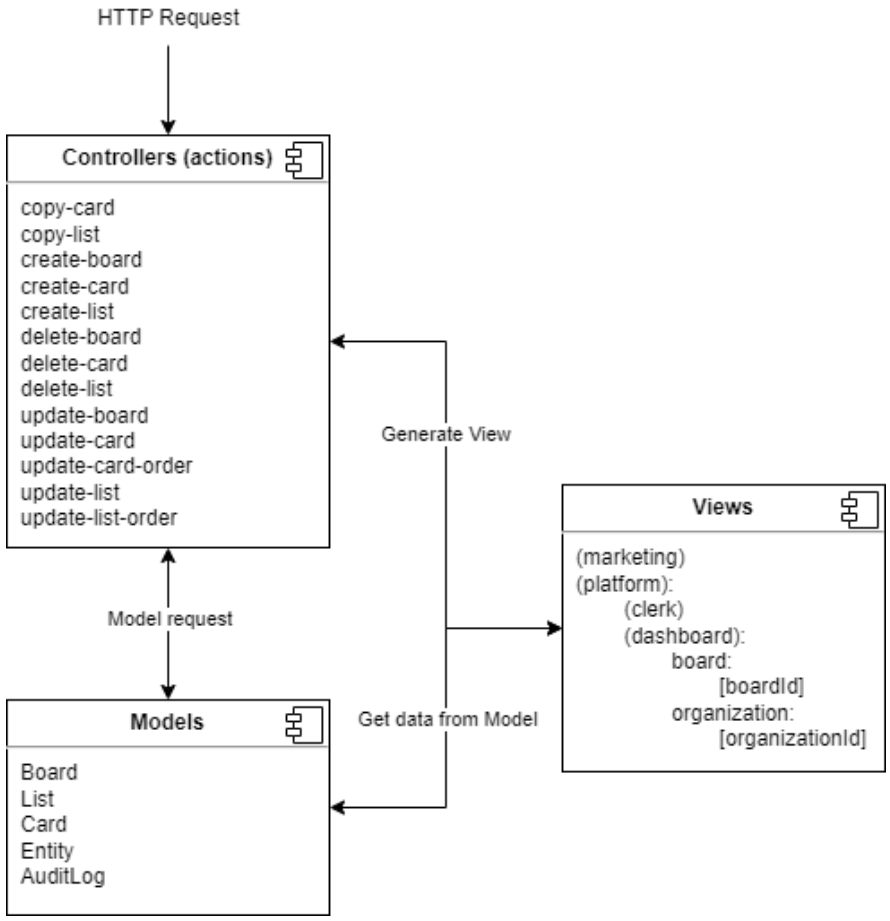

Рисунок 3.2 – Схема взаимодействия MVC компонентов web-приложения

3.4 Разработка интерфейса программной системы

Пользовательский интерфейс приложения сформирован с использованием компонентов ReactJS, компонентов Shadcn UI и Tailwind CSS.

Tailwind CSS – это современный CSS-фреймворк, который предлагает набор готовых классов, позволяющих быстро и легко создавать пользовательский интерфейс. Он отличается от других фреймворков тем, что не предоставляет готовых компонентов, а вместо этого фокусируется на создании модульных и переиспользуемых классов, которые можно комбинировать для создания нужных стилей.

Основные особенности Tailwind CSS:

1. Компонентная архитектура: Tailwind CSS не предоставляет готовых компонентов, но предлагает модульную архитектуру классов, которые можно комбинировать для создания нужного внешнего вида компонентов. Это дает большую гибкость и контроль над стилями.

2. Атомарные классы: Фреймворк использует атомарные классы, что означает, что каждый класс отвечает за конкретное свойство стиля. Например, вместо класса .button-red можно использовать отдельные классы .bg-red-500, .text-white, .p-2 для определения фона, текста и отступов кнопки соответственно.

3. Гибкость и настраиваемость: Tailwind CSS предлагает множество вариантов настройки и переопределения стилей. Вы можете настроить цвета, шрифты, отступы и другие свойства, чтобы адаптировать фреймворк под нужды вашего проекта.

4. Расширяемость: Вы можете легко расширять фреймворк, добавляя собственные классы или настраивая существующие. Это позволяет создавать уникальные стили, не нарушая принципов фреймворка.

5. Производительность: Tailwind CSS использует методологию "Just-in-Time" (JIT), которая позволяет генерировать только используемые классы,

что улучшает производительность и сокращает размер CSS-файла.

Пример интерфейса представлен на рисунках 3.3 – 3.8.

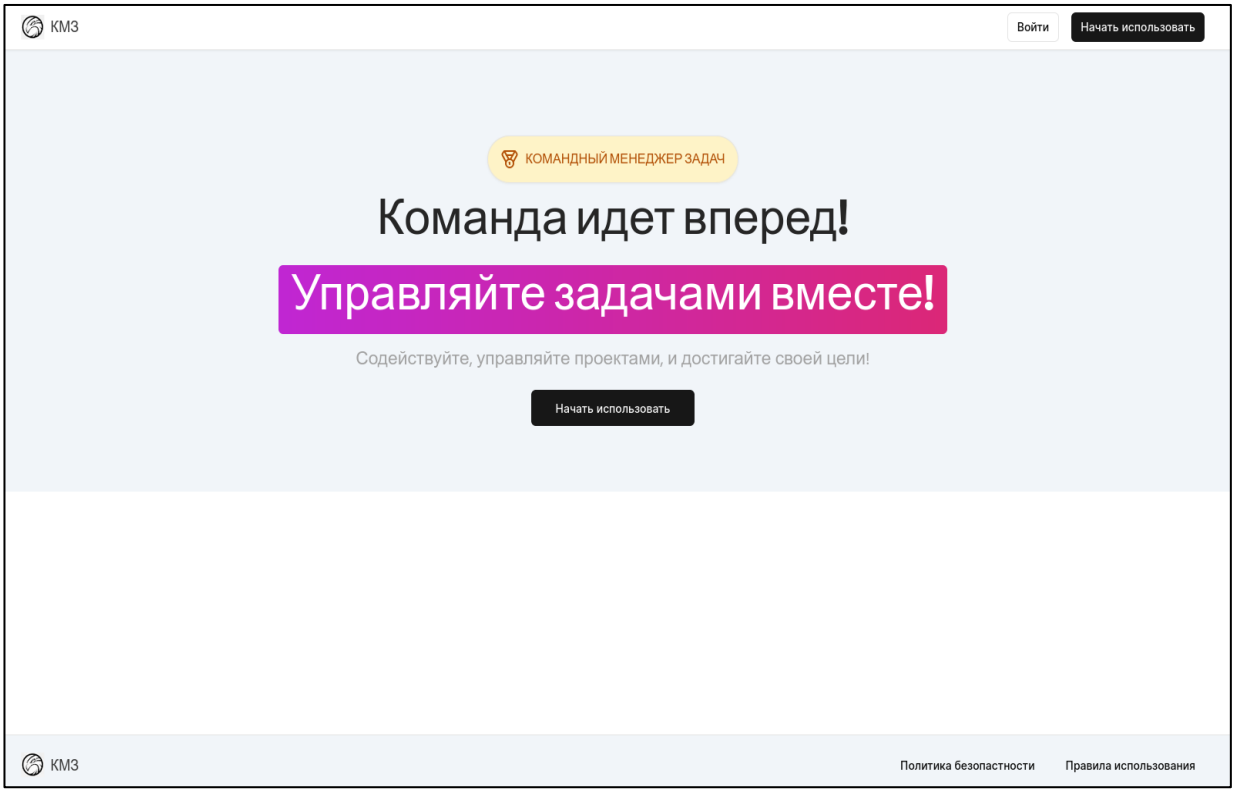

Рисунок 3.3 – Стартовая страница

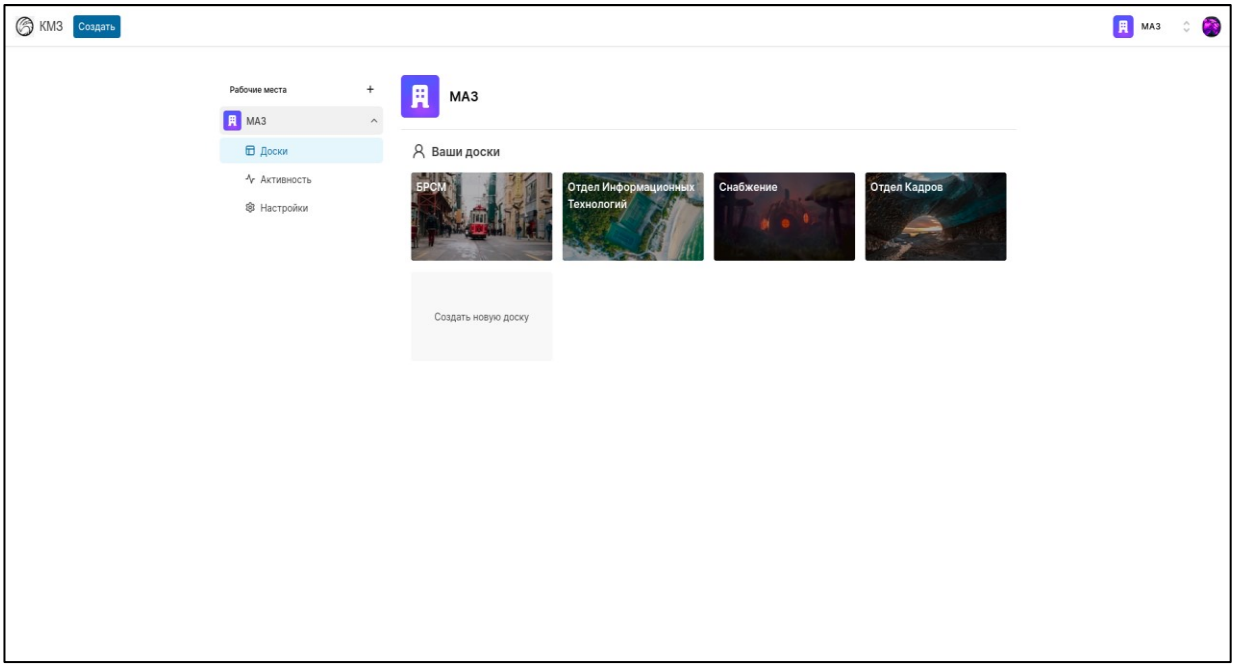

Рисунок 3.4 – Страница авторизованного пользователя

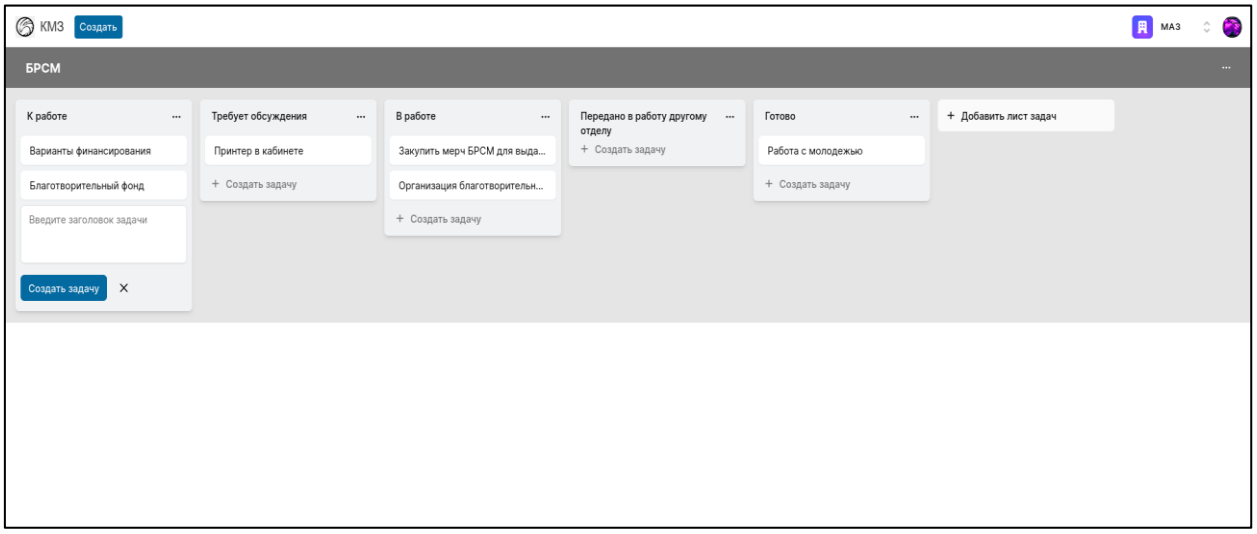

Рисунок 3.5 – Страница определенной доски организации

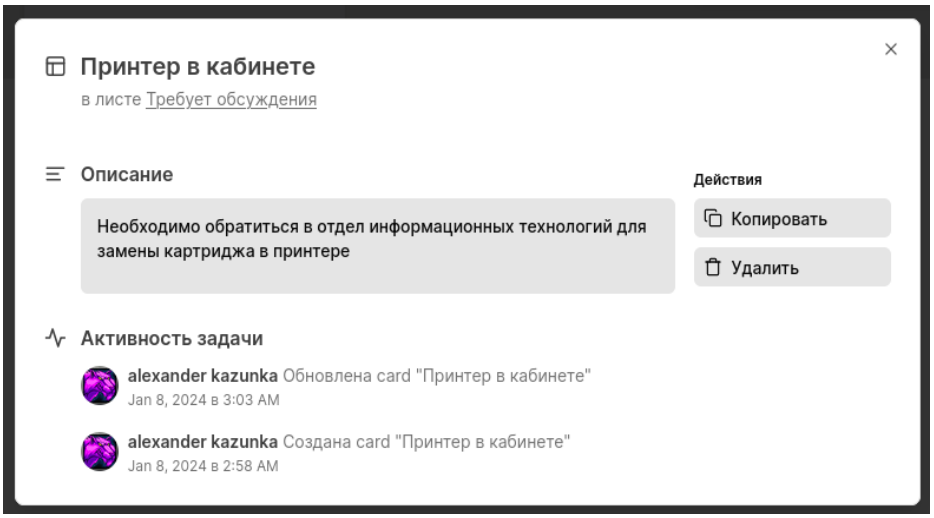

Рисунок 3.6 – Модальное окно задачи

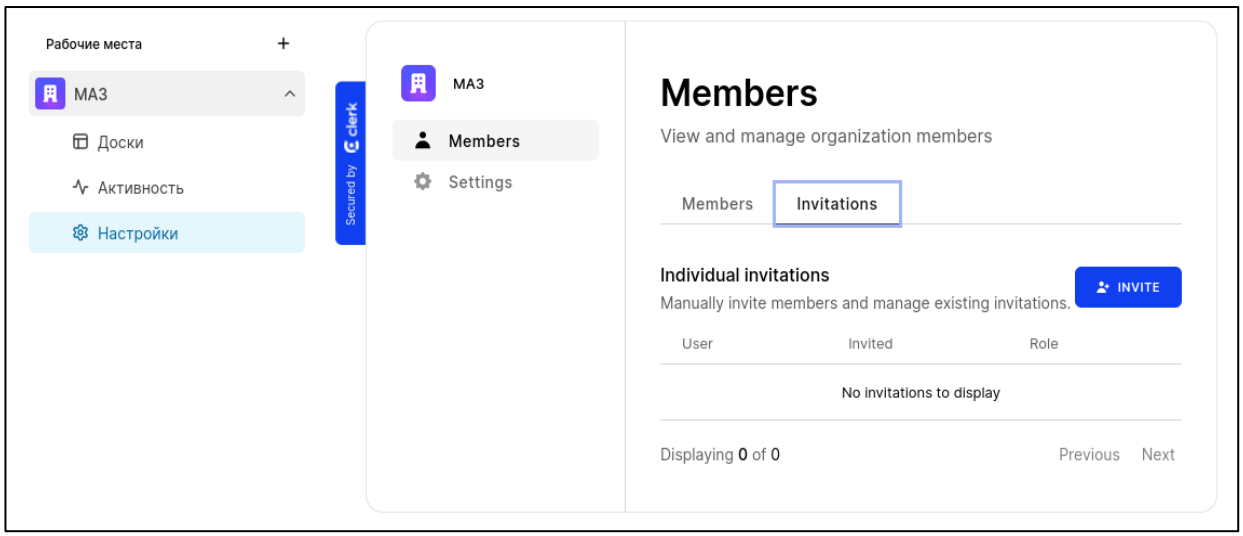

Рисунок 3.7 – Настройки организации

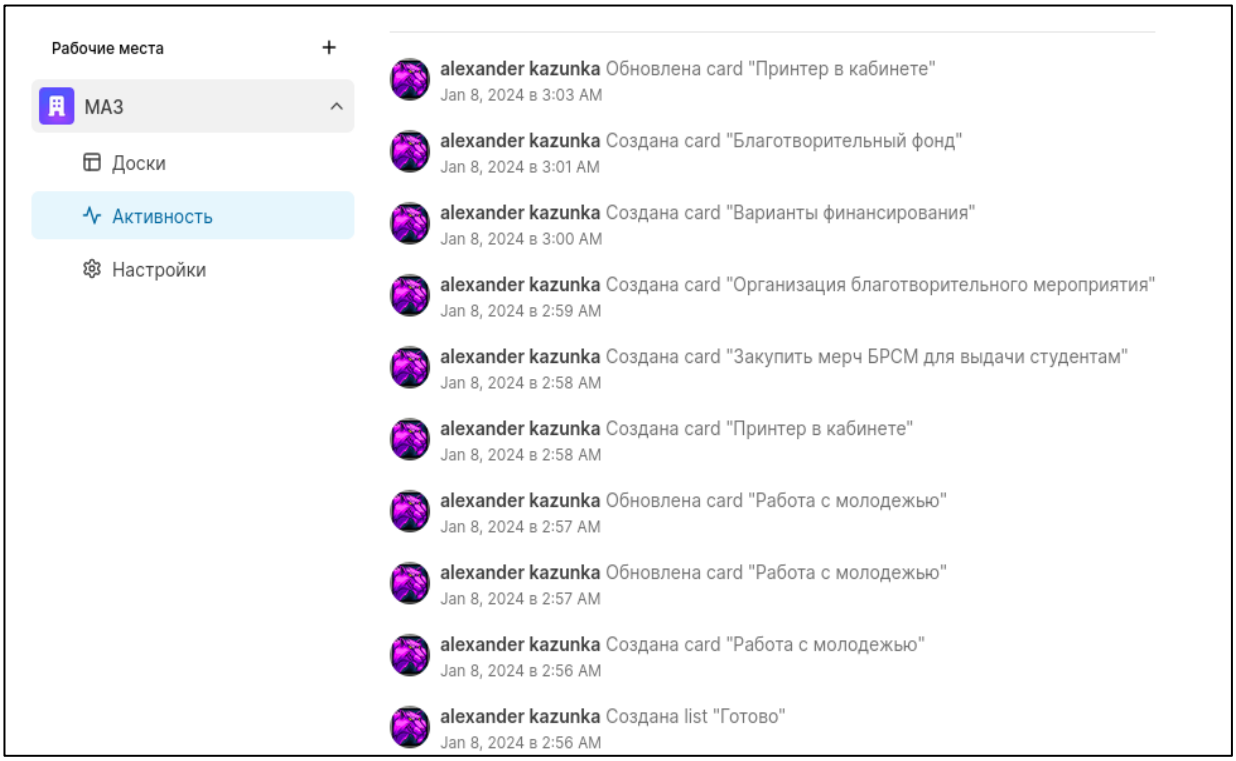

Рисунок 3.8 – Активность по организации