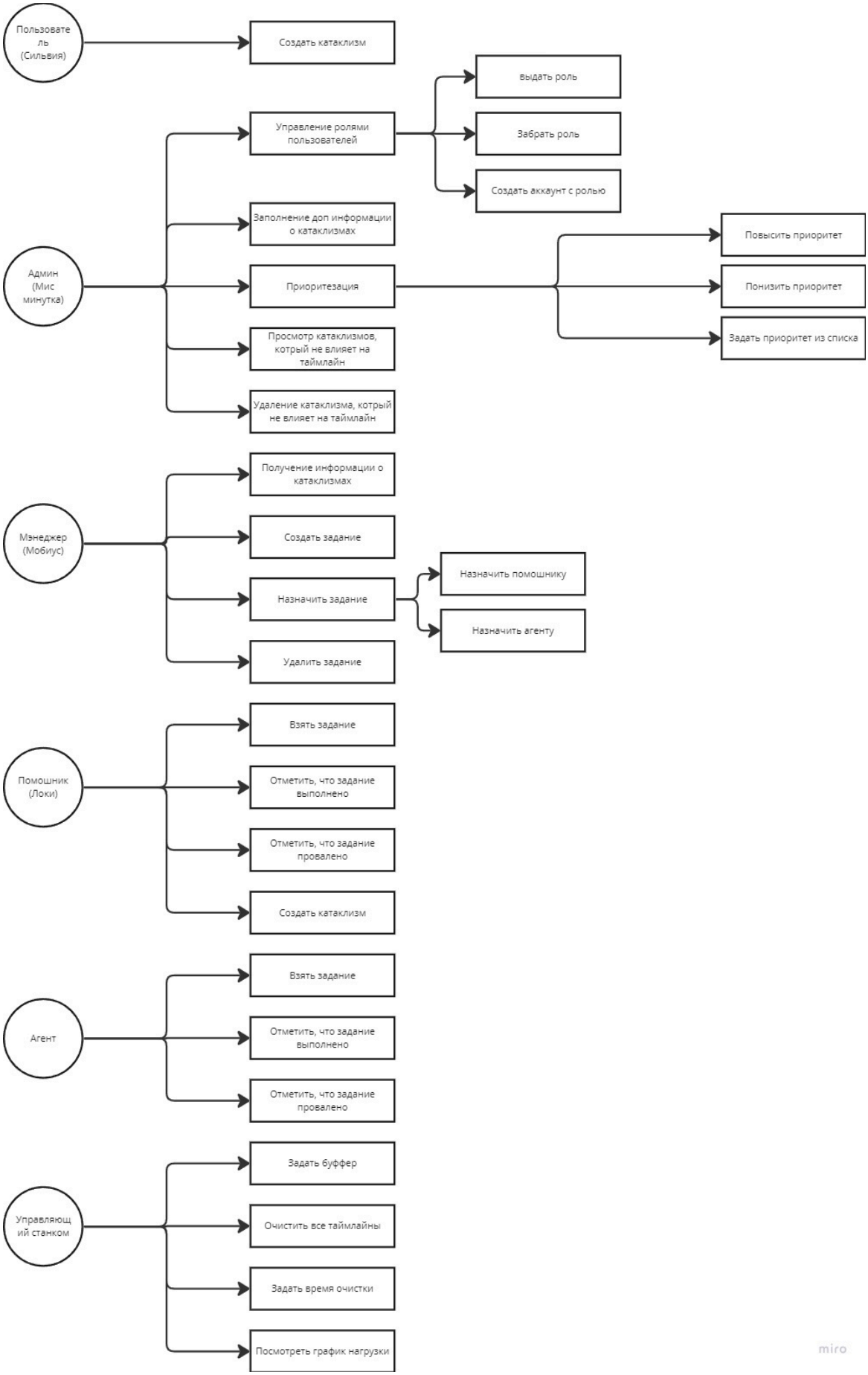
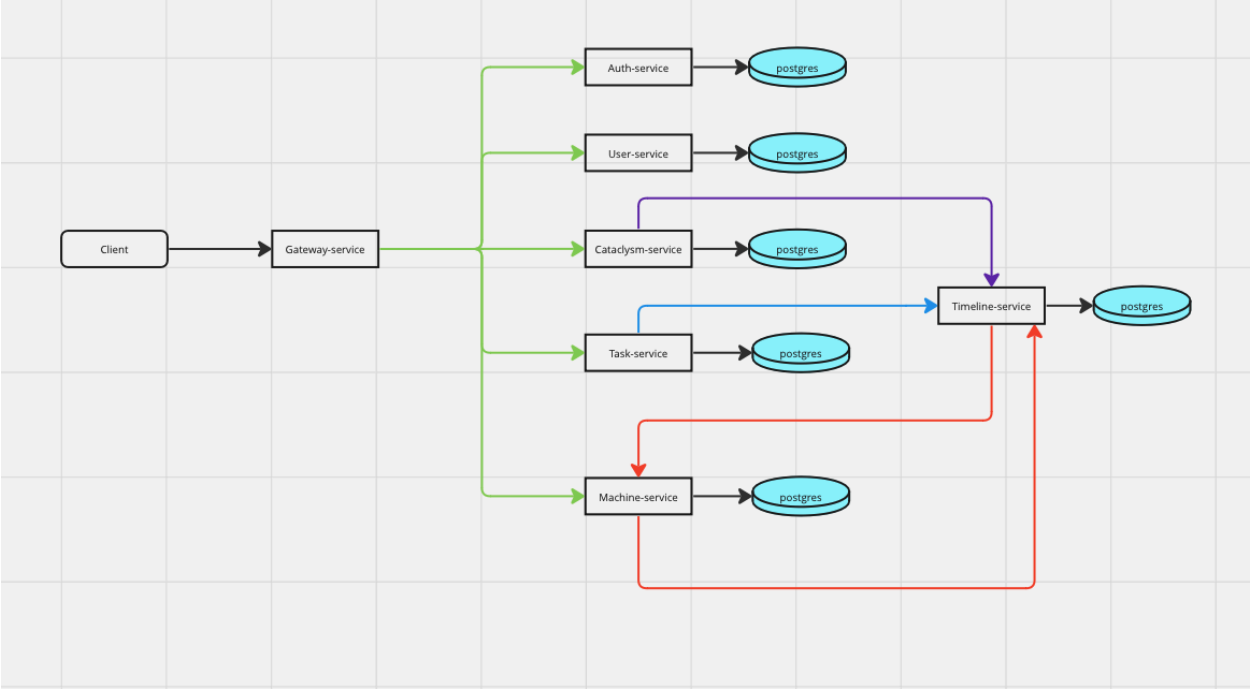
Проект по сериалу: “Локи”

Описание

"Временной Контроллер" - это интерактивная информационная платформа, вдохновленная сериалом "Локи", предназначенная для управления конфликтами мультивселенной и регулирования таймлайнов. Платформа позволяет пользователям взять на себя роль агентов УВИ (Управления Временными Изменениями), работать с различными версиями Локи, включая Сильвию, решать проблемы, вызванные нарушениями в таймлайне, и восстанавливать порядок в мультивселенной. Пользователи могут анализировать разветвления в таймлайнах, отправляться в миссии для устранения аномалий, сотрудничать или соперничать с другими агентами и версиями Локи, а также принимать стратегические решения, влияющие на будущее мультивселенной.


Пользователь (Сильвия)
Создать катаклизм
Управление ролями пользователей
выдать роль
Забрать роль
Создать аккаунт с ролью
Заполнение доп информации о катаклизмах
Приоритезация
Повысить приоритет
Понизить приоритет
Задать приоритет из списка
Просмотр катаклизмов, котрый не влияет на таймлайн
Удаление катаклизма, котрый не влияет на таймлайн
Админ (Мис минутка)
Получение информации о катаклизмах
Создать задание
Назначить задание
Назначить помошнику
Назначить агенту
Удалить задание
Мэнеджер (Мобиус)
Взять задание
Отметить, что задание выполнено
Отметить, что задание провалено
Создать катаклизм
Помошник (Локи)
Взять задание
Отметить, что задание выполнено
Отметить, что задание провалено
Агент
Задать буффер
Очистить все таймлайны
Задать время очистки
Посмотреть график нагрузки
Управляющий станком



Роли

1. Роль: Пользователь (Сильвия)
Функционал:

- Создать катаклизм
    - Ввод информации о катаклизме
    - Присвоение приоритета
    - Отметка катаклизма, который не включен в таймлайн

2. Роль: Админ (Мисс Минутка)
Функционал:

- Управление ролями пользователей
    - Выдать роль
    - Забрать роль
    - Создать аккаунт с ролью
- Заполнение доп информации о катаклизмах
- Присвоение приоритета
    - Повысить приоритет
    - Понизить приоритет
    - Выбрать приоритет из списка
- Просмотр катаклизмов, которые не влияют на таймлайн
- Удаление катаклизма, который не влияет на таймлайн

3. Роль: Менеджер (Мобиус)
Функционал:

- Получение информации о катаклизмах
- Создать задание
- Назначить задание
    - Назначить помощнику
    - Назначить агенту
- Удалить задание

4. Роль: Помощник (Локи)
Функционал:

- Отметить, что задание выполнено
- Отметить, что задание провалено
- Создать катаклизм
- Взять задание

5. Роль: Агент
Функционал:

- Взять задание
- Отметить, что задание выполнено
- Отметить, что задание провалено

6. Роль: Управляющий станком
Функционал:

- Задать буфер
- Очистить все таймлайны
- Задать время очистки
- Посмотреть график нагрузки

Технические требования

1. Платформа

Система должна быть разработана как веб-приложение, обеспечивающее доступ через браузер.

2. Технологии

Фронтенд: React.
Бэкенд: Java Spring.

3. База данных

1) Для хранения основных данных приложения использовать PostgreSQL.
    a) Данные должны быть структурированы и храниться в нормализованном виде для обеспечения эффективного доступа и манипуляций.

4. Аутентификация и авторизация

Использование механизма аутентификации и авторизации Spring Security для обеспечения безопасного доступа к функционалу в зависимости от роли пользователя.

5. Дизайн и интерфейс

1. Интерфейс должен быть разработан с использованием библиотеки React
2. Интерфейс должен быть разработан по принципу “desktop first”

6. Хранение данных

1. PostgreSQL**:** Для хранения основных данных приложения.

7. Безопасность

1. Обеспечение безопасности хранения паролей пользователей с использованием хэширования.
2. Реализация механизмов защиты от SQL-инъекций для обеспечения целостности и безопасности данных.

8. Масштабируемость

1. Проектирование системы с учетом микросервисной архитектуры для гибкого добавления нового функционала.

9. Тестирование

1. Разработка и проведение unit-тестов для каждого сервиса для обеспечения высокого уровня надежности и минимизации ошибок.

Дополнительные требования

1. Документация

1. Документирование API с помощью инструмента Swagger для удобства использования и интеграции.
2. Документирование кода с помощью JavaDoc для облегчения понимания и поддержки кодовой базы.

2. Обучение пользователей

1. Написание подробной обучающей документации для пользователей, объясняющей основные функции и возможности системы.

3. Поддержка

1. Использование адреса электронной почты для заявок о неисправной работе системы

4. Обновления

1. Ведение проекта на основе Git Flow для организации и управления процессом разработки, тестирования и внедрения изменений.
2. Регулярное обновление и поддержка кодовой базы с учетом обратной связи от пользователей и появляющихся требований.