**Система управления задачами**

**1. Определение целей и требований**

**Цели разработки продукта:**

* Создать систему для управления задачами, подходящую для индивидуального использования и команд.
* Обеспечить легкий и быстрый доступ к функционалу создания, редактирования и отслеживания задач.
* Предоставить возможность получения уведомлений о важных изменениях в задачах.

**Основные функциональные требования:**

* Регистрация и аутентификация пользователей.
* Возможность создания, редактирования и удаления задач.
* Отслеживание статусов задач (например, "новая", "в работе", "завершена").
* Уведомления о создании или изменении задач.

**Основные бизнес-кейсы:**

* **Регистрация пользователя:** Пользователь регистрируется в системе, создавая учетную запись с логином и паролем. После успешной регистрации пользователь может получить доступ к функционалу управления задачами.
* **Аутентификация пользователя:** Аутентифицированный пользователь создает новую задачу, указывая её детали и назначая статус. Пользователь также может выбрать команду (или группу пользователей), которая будет работать над задачей. Это позволяет централизовать задачи и упростить управление проектом.
* **Создание задачи:** Аутентифицированный пользователь создает новую задачу, указывая ее детали и назначая статус. Только зарегистрированные и вошедшие в систему пользователи могут создавать задачи.
* **Изменение задачи:** Аутентифицированный пользователь редактирует свои задачи и обновляет их статусы. Изменения могут вносить только те пользователи, которые имеют соответствующие права доступа.
* **Получение уведомлений:** Аутентифицированный пользователь получает уведомления о важных изменениях в задачах, таких как создание новых задач или обновление статусов. Уведомления отправляются только пользователям, которые имеют доступ к соответствующим задачам.

**Требования к производительности, масштабируемости и надежности:**

* **Производительность:** Система должна обеспечивать быструю обработку запросов на создание, редактирование и удаление задач.
* **Масштабируемость:** Система должна поддерживать увеличение количества пользователей и задач без значительного снижения производительности.
* **Надежность:** Микросервисы должны быть устойчивыми к сбоям, и данные должны быть защищены от потерь.

**2. Архитектурное проектирование**

**Микросервисная архитектура:**

* **User Service (Сервис пользователей):** Обрабатывает регистрацию, аутентификацию и управление профилем пользователя.
* **Task Service (Сервис задач):** Управляет CRUD операциями для задач и их статусами.
* **Notification Service (Сервис уведомлений):** Отправляет уведомления о создании или изменении задач.

**Брокер сообщений:**

* **Выбор брокера сообщений:** Apache Kafka.
  + **Роль:** Отправка и получение сообщений между микросервисами. Это позволит системе эффективно обрабатывать уведомления и события изменений задач в реальном времени.

**3. Выбор технологий**

**Технологический стек для микросервисов:**

1. **User Service:**
   * **Язык программирования:** Java
   * **Фреймворк:** Spring Boot
   * **База данных:** MySQL
2. **Task Service:**
   * **Язык программирования:** Java
   * **Фреймворк:** Spring Boot
   * **База данных:** MySQL
3. **Notification Service:**
   * **Язык программирования:** Java
   * **Фреймворк:** Spring Boot
   * **База данных:** Не требуется (может использовать встраиваемую память или простой файл для хранения логов уведомлений)

* **Язык программирования (Java):**  
  Java выбран из-за своей стабильности, производительности и широкого использования в корпоративной разработке. Он обеспечивает надежную многопоточность и имеет богатую экосистему библиотек и инструментов, что упрощает разработку и поддержку микросервисов.
* **Фреймворк (Spring Boot):**  
  Spring Boot упрощает создание и развертывание приложений, предоставляя автоматическую конфигурацию и шаблоны. Это позволяет быстро разрабатывать и интегрировать микросервисы с минимальными усилиями по настройке, что особенно важно для масштабируемых и гибких систем.
* **База данных (MySQL с Hibernate):**

Для взаимодействия с базой данных будет использоваться Hibernate, который является одним из самых популярных ORM-фреймворков в экосистеме Java. Hibernate предоставляет удобные механизмы для работы с базами данных, обеспечивая автоматическое отображение объектов Java в реляционные таблицы. В комбинации с MySQL, это решение позволяет эффективно управлять данными, упрощая разработку и обеспечивая высокую производительность приложений.

1. Клиентское приложение:

* Язык программирования: JavaScript
* Фреймворк: React

React выбран для клиентской части из-за его высокой производительности и гибкости. Он предоставляет мощные инструменты для создания интерактивных пользовательских интерфейсов с компонентной архитектурой, что упрощает разработку и поддержку пользовательского интерфейса.