УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Рефакторинг баз данных и приложений»

Дизайн документ проекта

Группа P34101

Студент:

Базанов Евгений Сергеевич.

Преподаватель:

Логинов Иван Павлович

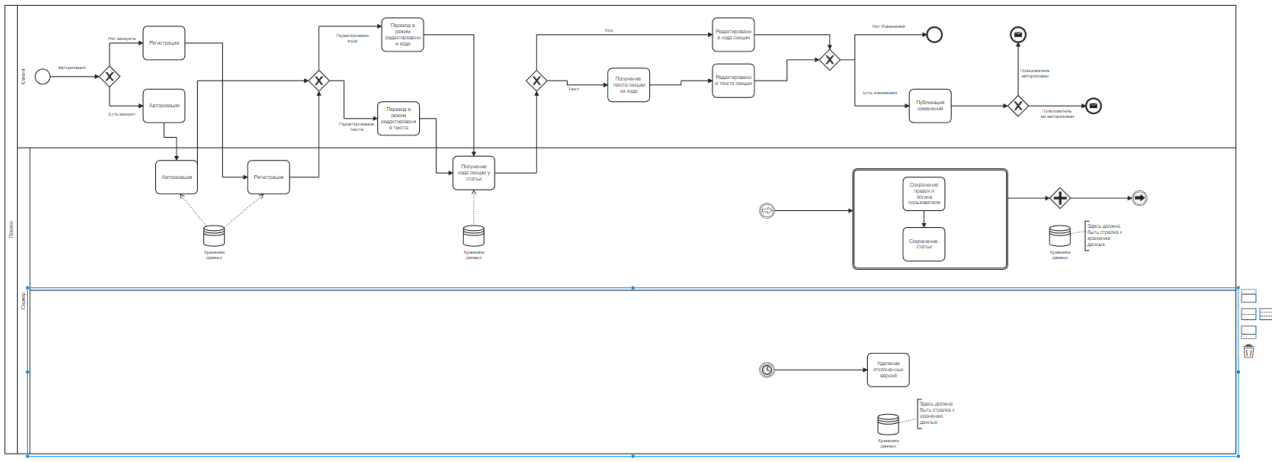
Санкт-Петербург,

2022 год

Описание проекта

Я хотел бы попробовать реализовать бизнес-процесс на основе сайта Википедия. Где будут 2 роли: пользователь, который сможет вносить изменения в статью, просматривать имеющиеся статьи; администратор, который сможет принимать или отклонять правки текста от пользователя, смотреть версии изменения статьи.

Пользовательские сценарии(изменение статьи)



Пользователь

По данной диаграмме пользовательских сценариев можно выделить 3 основных функциональных группы:

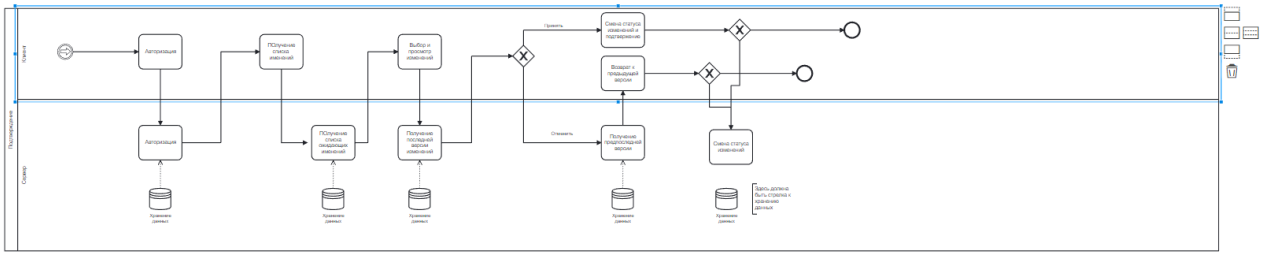
1. Регистрация и авторизация пользователей

2. Редактирование текса/кода секции и отправка изменений администратору на проверку

3. Просмотр статьи или часть статьи

*не относиться к пользовательскому сценарию, но раз в день будут удалятся все нерасмотренные администратором заявки пользователей на изменение текста статьи .

Сценарии для администратора



По данной диаграмме пользовательских сценариев можно выделить 4 основных функциональных группы:

Администратор

1)Авторизация администратора

2)Просмотр версий изменений статей

3)Получение списка изменений

4)Подтверждений или отмена новых пользовательских изменений текста, в случае отката изменений, текс будет последний версии до отправленных правок пользователя , а также сохранение изменений

Так как диаграммы большие, прикрепил в гите файл diagram.bpmn диаграммы ,которые можно посмотреть тут https://demo.bpmn.io/s/start

Этапы разработки(дерево задач

Свою разработку я разделю на 3 этапа.

1 этап создание самого сервиса , crud-репозиториев и всех базовых операций(Получение кода секции ,истории версий, предпоследней версии текста, последней версии текста ; подтверждение изменений, отмена изменений, сохранение новой версии текста)

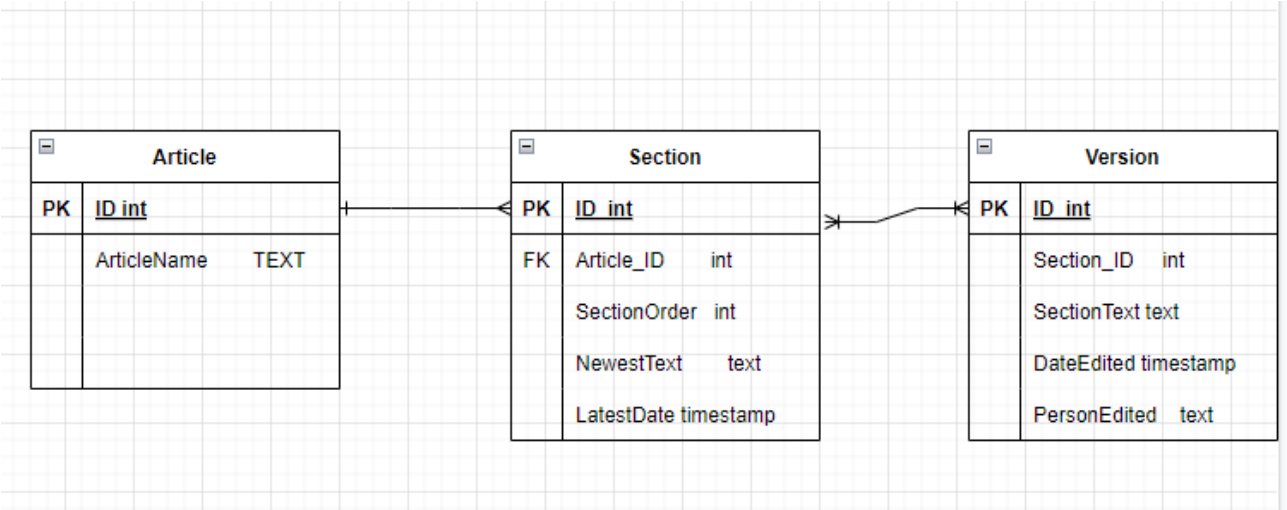
2 этап создание управление транзакциями и разграничение доступа к операциям бизнес-логики в соответствии ролями описанными в самом начале.

3)этап реализовать асинхронное выполнение задач с распределением бизнес-логики между 2 вычислительными узлами(на одном узле клиент-пользователь, который может проделывать свои операции, на втором администратор, который будет сохранять изменения) и выполнением периодических операций с использованием планировщика задач(удаление нерассмотренных в течение суток админом изменений, которые отправил пользователь)

- Первый этап
  - Postgres
    - Создать таблицы базы данных и вспомогательные функции для обработки времени
  - Spring Boot Application
    - Сконфигурировать проект
    - Создать сущности базы данных
    - Создать crud-репозитории и сервисы к ним
    - Подключить swagger для проверка работоспособности программы
- Второй этап
  - Postgres
    - Удалить вспомогательные функции для обработки времени
  - Spring Boot Application
    - Добавить менеджер транзакций Atomikos
    - Создать управление транзакциями с помощью JTA
    - Добавить Spring Security и разграничить выполнение операцией из сервисов в зависимости от ролей.
- Третий этап
  - Spring Boot Application
    - Перенести основной функционал администратора на второй вычислительный узел
    - Добавить асинхронное выполнение задачи
    - Добавить слушателя сообщений jms
  - ActiveMQ
    - Добавить брокер сообщений

Архитектурное решение

ER модель базы данных



1 Спринт

Добавлено:

- *Postgres:*
  - *Реализован sql скрипт по добавлению таблиц в базу данных, вспомогательные функции для работы со временем.*
- *SpringBootApplication*
  - *Добавление сущности, необходимые для реализации проекта*
  - *Создал к сущностям crud-репозитории*
  - *Создал сервисы для сущностей*
  - *Добавил swagger-ui для проверки корректной работы реализованных сервисов.*

*Ссылка на diff(*смотреть diff на github*)*

*Ссылка на запись экрана 1 спринта(*смотреть запись*)*

2 Спринт

Добавлено:

- *SpringBootApplication*
  - *Добавил менеджер управления транзакциями Atomikos .*
  - *Создал управления транзацкциями с помощью JTA.*
  - *Добавил Spring Security*
  - *Добавил разграничений ролей и* выполнение операцией из сервисов в зависимости от ролей

Изменено или удалено:

- *Postgres:*
  - Удалены *вспомогательные функции для работы со временем, они перенесены в SpringBootApplication*
- *SpringBootApplication*
  - *Убрал swagger-ui*

Роли и привилегии:

- Администратор
  - Команды accept/decline. Принять или отклонить отправленную юзером версию текста.
  - Команда all . Позволяет просмотреть все версии текста секции.
  - Команда previous. Позволяет просмотреть предпоследнюю версию текста секции
  - Команда last. Позволяет просмотреть последнюю версию текста секции
  - Команда code . Позволяет просмотреть текущей текста секции
  - Команда create и auth .Регистрация и авторизация.
- Пользователь
  - Команда code . Позволяет просмотреть текущей текста секции
  - Команда create и auth .Регистрация и авторизация.
  - Команда save .Отправить новую версию секции кода.

*Ссылка на diff(*смотреть diff на github*)*

*Ссылка на запись экрана 2 спринта(*смотреть запись*)*