**Министерство науки и высшего образования Российской Федерации** **Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Московский государственный технический университет** **имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Техническое задание по курсовой работе

по предмету "Сетевые технологии в АСОИУ"

Вариант №19 "Сервис микроблогов"

Исполнители:

Харитонов А.А. гр. ИУ5-64Б

Верин Д. С. гр. ИУ5-64Б

Емельянова Т. И. гр. ИУ5-65Б

Утверждает:

Галкин В.А.

Москва 2023 г.

1. **Введение**

Требуется разработать сервис микроблогов в рамках курсовой работы. Микроблог – это список создаваемых пользователем публичных текстовых сообщений. Пользователь имеет возможность просматривать микроблоги других пользователей и создавать свою реакцию на их сообщения в виде лайков, комментариев и репостов. В таком контексте разрабатываемый сервис микроблогов можно причислить к множеству приложений типа “социальная сеть”.

1. **Исполнители и роли**

В рамках курсовой работы информационный проект создаётся коллективно. При этом функционально роли проекта разделяются на 3 независимых модуля: фронтенд, бэкенд и протокол webSocket.

Так, исполнители данного ТЗ разделяют поставленные роли следующим образом:

* Татьяна Емельянова Игоревна (ИУ5-65Б) – фронтенд модуль приложения
* Верин Дмитрий Сергеевич (ИУ5-64Б) – бэкенд + БД модули приложения
* Харитонов Андрей Алексеевич (ИУ5-64Б) – webSocket модуль приложения

1. **Основания для разработки**

Основанием для разработки является учебный план МГТУ им. Баумана кафедры ИУ5 на 6 семестр.

1. **Требования к программному изделию**

Основные требования излагаются описанием варианта курсовой работы и в дальнейшем корректируются преподавателем при составлении технического задания.

* 1. **Функциональные требования**
     1. **Общие функциональные требования**
        1. Должна быть реализована система авторизации/регистрации для идентификации пользователя
        2. Должна быть реализована личная страница пользователя, с отображением информации о нём и его сообщениях (постах)
        3. Должен быть реализован поиск чужого блога
        4. Должна быть реализована система создания сообщения (поста) на личной странице пользователя
        5. Должна быть реализована система пересылки (репоста) чужого сообщения на личную страницу
        6. Должна быть реализована система комментирования сообщений (постов)
     2. **Функциональные требования к фронтенд модулю**

Логикой приложения предусматривается 2 роли: гость и пользователя. Гость – это неавторизованный пользователь. Он должен иметь возможность только просматривать личные страницы (блоги) авторизованных пользователей. При попытке взаимодействовать с функционалом, доступным авторизованному пользователю, гость обязан быть перенаправлен на форму авторизации.

* + - 1. Форма регистрации

На данной странице происходит регистрация пользователя. Необходимые дынные для регистрации: username, password, email, телефон, дата рождения, ФИО.

* + - 1. Форма входа

На данной странице происходит авторизация пользователя. Необходимые данные для авторизации: email или username, password.

* + - 1. Страница пользователя

На данной странице отображается основная информация о пользователе, а также лента его постов. Если это личная страница пользователя, то ему становится доступен функционал по её редактированию: создание постов, редактирование личной информации.

* + - 1. Страница редактирования информации о пользователе

На данной странице происходит редактирование личной информации пользователя. Переход на неё описан в п.4.1.1.3 текущего списка.

* + - 1. Страница списка комментариев поста

На данной странице отображается пост и список комментариев данного поста. Переход на неё осуществляется с помощью иконки “комментарии” поста.

* + - 1. Страница списка подписчиков

На данной странице список подписчиков личной страницы пользователя (блога). Переход на неё осуществляется с помощью кнопки “подписчики”, расположенной на главной странице пользователя.

* + - 1. Страница подписок пользователя

На данной странице список подписок пользователя. Переход на неё осуществляется с помощью кнопки “подписки”, расположенной на главной странице пользователя. Если пользователь является владельцем данной страницы, то он может удалять подписки.

* + - 1. Страница создания поста

На данной странице происходит создание нового поста пользователя. Переход на неё описан в п.4.1.1.3 текущего списка.

* + - 1. Страница оповещений о новых событиях (Лента)

На данной странице присутствует лента с новыми постами, добавленными авторами каналов, на которые подписан пользователь. С помощью неё можно переходить на страницу пользователя, кому принадлежит новый пост.

* + - 1. Страница поиска блога

На данной странице производится поиск блогов по их названию. Обязан выводиться список всех блогов, чьи названия семантически схожи поставленной пользователем текстовой строке.

* + 1. **Функциональные требования к бэкенд модулю**

В данном приложении бэкенд имеет только HTTP (ajax) эндпоинты. Для авторизации используется библиотека Djoser, которая реализует соответствующие эндпоинты. В соответствии с методологией REST синтезированы ресурсы, описанные ниже. Так как система имеет 1 роль авторизованного пользователя, все методы имеют ограничение авторизации, кроме метода создания (регистрации) пользователя.

* + - 1. Пользователь (GET, POST, PUT, DELETE)
* **users/ GET – список пользователей**

Параметры для фильтрации: id, username, firstname, lastname

Параметр для пэйджирования ответа: page

Ответ должен иметь массив пользователь, отсортированный по числу подписчиков

* **users/ POST – создать пользователя**

Параметры, необходимые для регистрации:username, password, email, birthday, firstname, lastname

* **users/ PUT – изменить пользователя по его id**

Параметр для изменения конкретного пользователя: id

Также параметры, являющиеся изменяемыми моделями (см. ER-диаграмму)

* **users/ DELETE – удалить пользователя по id**

Параметр для удаления конкретного пользователя: id

* + - 1. Пост (GET, POST, PUT, DELETE)
* **posts/ GET – поиск постов**

Параметры для фильтрации: id, user\_id, start\_date, end\_date, content

Параметр для пэйджирования ответа: page

Ответ должен иметь массив постов, отсортированный по дате публикации

* **posts/ POST – создать пост**

Параметры для создания поста: content

Параметр для создания репоста: repostID

* **posts/ PUT – изменить пост по id**

Параметр для изменения конкретного поста: id

Параметр изменения содержания поста: content

* **posts/ DELETE – удалить пост по id**

Параметр для удаления конкретного поста: id

* + - 1. Комментарий (GET, POST, PUT, DELETE)
* **comments/ GET – поиск комментариев**

Параметры для фильтрации: id, user\_id, start\_date, end\_date, content

Параметр для пэйджирования ответа: page

Ответ должен иметь массив комментариев, отсортированный по дате публикации

* **comments/ POST – создать комментарий**

Параметры для создания комментария: content, post\_id

* **comments/ PUT – изменить комментарий по id**

Параметр для изменения конкретного поста: id

Параметр изменения содержания поста: content

* **comments/ DELETE – удалить комментарий по id**

Параметр для удаления конкретного поста: id

* + - 1. Лайк (GET, POST, DELETE)
* **likes/ GET – поиск лайков**

Параметры для фильтрации: user\_id, post\_id

Ответ должен иметь массив лайков

* **likes/ POST – создать лайк**

Параметры для создания лайка: post\_id

* **likes/ DELETE – удалить лайк**

Параметр для удаления конкретного лайка: post\_id

* + - 1. Уведомления (GET, DELETE)
* **notifications/ GET – поиск комментариев**

Параметры для фильтрации: user\_id

Ответ должен иметь массив уведомлений, отсортированный по дате публикации

* **notifications/ DELETE – удалить комментарии по user\_id**

Параметр для удалений уведомлений: user\_id

* + - 1. Подписки (GET, POST, DELETE)
* **subscriptions/ GET – поиск комментариев**

Параметры для фильтрации: user\_id

Ответ должен иметь семантически отсортированный массив подписок

* **subscriptions/ POST – создать комментарий**

Параметры для создания подписки: user\_id

* **subscriptions/ DELETE – удалить комментарий по id**

Параметр для удаления конкретной подписки: user\_id

* + 1. **Функциональные требования к webSocket модулю**
       1. **connect WS протокол**

Параметр id: установление websocket соединения по сохранённому идентификатору сессии, если он сохранён.

Требуется удалить старое соединение в случае переподключения клиента.

* + - 1. **setActivePage WS протокол**

Параметр user\_id\_page: id страницы (ленты) пользователя, активной на фронтенд модуле в данный момент.

* + - 1. **addPostEvent/ gRPC протокол**

Параметр post\_id: id нового поста

Требуется сохранять информацию об опубликованном посте в таблицу уведомлений, если нет открытого соединения или соединение не корректно с пользователем, которому необходимо передать пост.

* + - 1. **deletePostEvent/ gRPC протокол**

Параметр post\_id: id удалённого поста

* + - 1. **changePostEvent/ gRPC протокол**

Параметр post\_id: id изменённого поста

Обязан обновить информацию всего поста на клиенте

* + - 1. **changePostLikesEvent/ gRPC протокол**

Параметр post\_id: id изменённого поста

Обязан обновить только информацию о лайках на клиенте

* 1. **Технические требования**

В соответствие каждой описанной в п.2 текущего документа роли требуется разработать веб-сервис. Каждому веб-сервису соотносится свой набор технологий:

* Фронтенд модуль: основная технология – React (Node.js)
* Бэкенд модуль: основная технология – Django (DRF) + MySql для организации БД.
* WebSocket модуль: основная технология – Flask(socketIO) + Node.js(socketIO).

Для демонстрации проекта необходимо устройство, содержащее ОС Windows или любую другую ОС, пригодную для развёртывания модулей; браузер Chrome, минимум 512Мб свободной оперативной памяти (без учёта памяти, занимаемой другими приложениями и ОС), минимум двухъядерный процессор частотой 1.1 ГГц, сопутствующие компоненты, необходимые для работы устройства.

1. **Требования к программной документации**

Требуется разработать следующий комплект документов: ТЗ, РПЗ, ПМИ, РП, РСА, диаграмма прецедентов, диаграмма последовательности, ER-диаграмма, диаграмма деятельности, диаграмма развертывания.

1. **Стадии и этапы разработки**
   1. 3 неделя семестра – черновой вариант ТЗ.
   2. 8 неделя семестра – ТЗ, исправление с учетом замечаний; макет и дизайн сайта в figma, диаграмма прецедентов, ER-диаграмма, диаграмма развертывания, документация бэкенд эндпоинтов в swagger.
   3. 12 неделя семестра – полный комплект документов согласно п.4 текущего документа.
   4. 14-15 неделя семестра – защита проекта, презентация по диаграммам, демонстрация приложения.
2. **Порядок контроля и приёмки**

С 3 по 8 неделю семестра происходит корректировка и уточнение текущего документа до полного утверждения. Далее происходит разработка приложения согласно утвержденному ТЗ. В конце семестра происходит публичная презентация проекта и его защита.