|  |  |
| --- | --- |
|  | **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»**

Дисциплина «Разработка интернет-приложений»

Техническое задание

Тема: «Эпоха географических открытий»

Студент: Терентьева С.Д.

Группа ИУ5-53Б

Преподаватель: Канев А.И.

2023г.

**СОДЕРЖАНИЕ**

[1. Цель 3](#_Toc150775711)

[2. Назначение 3](#_Toc150775712)

[3. Задачи 3](#_Toc150775713)

[4. Методы веб-сервиса 4](#_Toc150775714)

[5. Функциональные требования 13](#_Toc150775715)

[6. Требования к программному обеспечению: 21](#_Toc150775716)

[7. Требования к аппаратному обеспечению: 22](#_Toc150775717)

## Цель

Реализовать систему для автоматизации документирования исторических открытий, которая будет состоять из веб-приложения, веб-сервиса, удаленного сервиса и мобильного приложения.

## Назначение

Основное назначения разработанной системы заключается в автоматизации и оптимизации процесса документирования исторических открытий. Система позволит быстро находить информацию об исторических открытиях и первооткрывателях, которые их совершили. Также она позволит сотрудникам – быстро и легко создавать заявки на добавления новых открытий, а администраторам – принимать или отклонять их, тем самым повысив эффективность документирования исторических открытий.

## Задачи

* 1. Разработать базовый дизайн системы
  2. Создать базу данных PostgreSQL на Django
  3. Реализовать веб-сервис на Django REST Framework
  4. Создать интерфейс гостя на React
  5. Развернуть веб-приложение на GitHub Pages для предварительной демонстрации работы
  6. Написать документацию методов API
  7. Добавить аутентификацию и авторизацию, основанную на JWT и Redis
  8. Создать интерфейс пользователя на React, с использованием Redux Toolkit
  9. Создать нативное приложение на React Native
  10. Создать асинхронный веб-сервис на Go
  11. Создать интерфейс администратора на React
  12. Задокументировать ТЗ, РПЗ и диаграммы

## Методы веб-сервиса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Метод** | **Описание** | **URL** | **Входные данные** | **Выходные данные** |

|  |
| --- |
| 4.1 Методы первооткрывателей |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4.1.1 | GET | Возвращает список первооткрывателей, удовлетворяющих переданным критериям  Будут возвращены первооткрыватели, отфильтрованные по статусу «действует», а также id чернового открытия, пользователя если оно есть  Доступно всем пользователям | api/pioneers/search | {  “query”: char(100),  “access\_token”: char(100)  } | {  “draft\_discovery\_id”: int,  “pioneers”: [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  },  …  ]  } |
| 4.1.2 | GET | Возвращает первооткрывателя  Доступно всем пользователям | api/pioneers/<pioneer\_id>/ | {  “pioneer\_id”: int  } | {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  } |
| 4.1.3 | PUT | Обновляет информацию о первооткрывателе  Доступно только администратору | api/pioneers/<pioneer\_id>/update/ | {  “pioneer\_id”: int,  “access\_token”: int  } | {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  } |
| 4.1.4 | DELETE | Меняет статус первооткрывателя на  «удален»  Доступно только администратору  Возвращает список из всех первооткрывателей, отфильтрованных по статусу «действует», | api/pioneers/<pioneer\_id> /delete/ | {  “pioneer\_id”: int,  “access\_token”: int  } | [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  },  …  ] |
| 4.1.5 | POST | Создает нового первооткрывателя  Доступно только администратору | api/pioneers/create/ | {  “access\_token”: int  } | [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  },  …  ] |
| 4.1.6 | POST | Добавляет первооткрывателя в открытие  Возвращает список всех первооткрывателей в открытии  Доступно только авторизованным пользователям | api/pioneers/<pioneer\_id> /add\_to\_discovery/ | {  “pioneer\_id”: int,  “access\_token”: int  } | [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  },  …  ] |
| 4.1.7 | GET | Возвращает картинку первооткрывателя в бинарном виде  Доступно всем пользователям | api/pioneers/<pioneer\_id> /image / | {  “pioneer\_id”: int  } | {  “image”: bytes  } |
| 4.1.8 | PUT | Обновляет картинку первооткрывателя  Возвращает обновленную картинку первооткрывателя  Доступно только авторизированным пользователям | api/pioneers/<pioneer\_id> /update\_image / | {  “pioneer\_id”: int,  “image”: bytes  } | {  “image”: bytes  } |

|  |
| --- |
| 4.2 Методы открытий |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4.2.1 | GET | Возвращает список открытий, удовлетворяющих переданным критериям  Критерии:  Статус, пользователь, диапазон даты формирования  Доступно только авторизированным пользователям | api/discoveries/search/ | {  “access\_token”: char(100),  “status”: int,  “user”: int,  “date\_start”: long,  “date\_end”: long  } | [  {  "id": int,  "date": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": " datetime  "owner": {  “id”: int,  “email”: bool,  “is\_moderator”: bool  },  “moderator”: {  “id”: int,  “email”: bool,  “is\_moderator”: bool  },  "pioneers": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  },  …  ]  },  …  ] |
| 4.2.2 | GET | Возвращает открытие  Доступно только авторизированным пользователям | api/discoveries/<discovery\_id>/ | {  “discovery\_id”: int,  “access\_token”: char(100)  } | {  "id": int,  "date": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": " datetime  "owner": {  “id”: int,  “email”: bool,  “is\_moderator”: bool  },  “moderator”: {  “id”: int,  “email”: bool,  “is\_moderator”: bool  },  "pioneers": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  },  …  ]  } |
| 4.2.3 | PUT | Обновляет год открытия  Доступно с ключом доступа | api/discoveries/<discovery\_id>/ update\_audience/ | {  “discovery\_id”: char(100),  “year”: int,  “secret\_key”: char(100)  } | Статус 200 / 403 |
| 4.2.4 | PUT | Обновляет информацию об открытии  Возвращает обновленное открытие  Доступно только авторизированным пользователям | api/discoveries/<discovery\_id>/update/ | {  “discovery\_id”: int,  “access\_token”: char(100)  } | {  "id": int,  "date": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": " datetime  "owner": {  “id”: int,  “email”: bool,  “is\_moderator”: bool  },  “moderator”: {  “id”: int,  “email”: bool,  “is\_moderator”: bool  },  "pioneers": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  },  …  ]  } |
| 4.2.5 | PUT | Обновляет статус открытия с «черновик»  на «в работе»  В случае успеха меняет дату формирования открытия на текущую  Можно обновить только при наличии статуса «в работе»*,* иначе возвращает ошибку 405  Возвращает обновленное открытие  Доступно только авторизированным пользователям | api/discoveries/<discovery\_id>/update\_status\_user/ | {  “discovery\_id”: int,  “access\_token”: char(100)  } | {  "id": int,  "date": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": " datetime  "owner": {  “id”: int,  “email”: bool,  “is\_moderator”: bool  },  “moderator”: {  “id”: int,  “email”: bool,  “is\_moderator”: bool  },  "pioneers": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  },  …  ]  } |
| 4.2.6 | PUT | Меняет статус открытия с «в работе» на «завершён» или «отклонён»  В случае успеха меняет дату завершения открытия на текущую  Можно обновить только при наличии статуса «в работе»*,* иначе возвращает ошибку 405  Возвращает обновленное открытие  Доступно только авторизированным пользователям | api/discoveries/<discovery\_id>/update\_status\_admin/ | {  “discovery\_id”: int,  “access\_token”: char(100)  } | {  "id": int,  "date": int,  "status": int,  "date\_created": datetime,  “date\_formation": datetime,  "date\_complete": " datetime  "owner": {  “id”: int,  “email”: bool,  “is\_moderator”: bool  },  “moderator”: {  “id”: int,  “email”: bool,  “is\_moderator”: bool  },  "pioneers": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  },  …  ]  } |
| 4.2.7 | DELETE | Меняет статус открытия с «черновик» на «удалён»  Можно выполнить только при наличии у открытия статуса «черновик»*,* иначе возвращает ошибку 405  В случае успеха возвращает статус 200  Доступно только авторизированным пользователям | api/discoveries/<discovery\_id>/delete/ | {  “discovery\_id”: int,  “access\_token”: char(100)  } | Статус 200 / 404 / 405 |
| 4.2.8 | DELETE | Удаляет первооткрывателя из открытия  Возвращает список оставшихся первооткрывателей в открытии  Доступно только авторизированным пользователям | api/discoveries/<discovery\_id>/delete\_pioneer/<pioneer\_id>/ | {  “discovery\_id”: int,  “pioneer\_id”: int,  “access\_token”: char(100)  } | {  "pioneers": [  {  “id”: int,  “name”: char(100),  “description”: text,  “status”: int,  “image”: char(100),  “date\_birthday”: int,  “date\_death”: int,  },  …  ]  } |

|  |
| --- |
| 4.4 Методы авторизации и аутентификации |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4.4.1 | POST | Регистрирует нового пользователя в базе данных  Если в системе уже существует пользователем с такой почтой, то возвращает ошибку 409  В случае успеха возвращает статус 201, id и access\_token созданного пользователя  Доступен всем пользователям | api/register | {  “email”: char(100),  “password”: char(100)  } | {  “user\_id”: int,  “access\_token”: char(100)  } |
| 4.4.2 | POST | Выполняет аутентификацию пользователя по логину и паролю  В случае отправки невалидных данных возвращает ошибку 401  В случае успеха возвращает статус 200, а также данные о пользователе | api/login | {  “email”: char(100),  “password”: char(100)  } | {  “user\_id”: int,  “email”: char(100),  “is\_moderator”: bool,  “access\_token”: char(100)  } |
| 4.4.3 | POST | Выполняет аутентификацию пользователя по токену  В случае отправки невалидного токена возвращает ошибку 401  В случае успеха возвращает статус 200, а также данные о пользователе | api/check | {  “access\_token”: char(100)  } | {  “user\_id”: int,  “email”: char(100),  “is\_moderator”: bool,  “access\_token”: char(100)  } |
| 4.4.4 | POST | Добавляет его токен в черный список  Доступно только авторизированным пользователям | api/logout | {  “access\_token”: char(100)  } | Статус 200 / 401 |

|  |
| --- |
| 4.5 Методы удаленного сервиса |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4.5.1 | POST | Определяет год открытия  Действие запускается через 4 секунды. Вероятность успеха 70% | api/calculate\_ year | {  “discovery\_id”: int  } | Статус 200 / 403 |

## Функциональные требования

* 1. Главное меню
     1. Первооткрыватели – переход на страницу со списком / таблицей первооткрывателей
        1. Доступно **всем** пользователям
        2. Для **обычных** пользователей выполняется переход на страницу со списком первооткрывателей (5.4)
        3. Для **администраторов** выполняется переход на страницу с таблицей первооткрывателей (5.5)
     2. Открытия – переход на страницу с историей открытий (5.8)
        1. Доступно только **авторизированным** пользователям.
     3. Вход – переход на страницу с формой авторизации (5.2)
        1. Доступно только **неавторизированным** пользователям.
     4. Выход
        1. Доступно только **авторизированным** пользователям.
        2. При нажатии отправляется запрос на API (4.4.4), который добавляет токен пользователя в черный список. Также токен удаляется из cookies, обнуляется состояние пользователя в Redux и происходит переадресация на страницу со списком первооткрывателей (5.4)
  2. Страница с формой авторизации
     1. Доступна только **неавторизованным** пользователям
     2. Содержит
        1. Тестовое поле для ввода почты
        2. Текстовое поле для ввода пароля
        3. Кнопку **Войти**
        4. Кнопку **Уже зарегистрированы?**
     3. Действия
        1. При нажатии на кнопку **Войти** отправляется запрос на API (4.4.2), в котором передаются введенные данные формы. Если введеные данные верны, то в ответе возвращается access токен, который сохраняется в cookies, а также происходит переадресация на страницу со списком первооткрывателей (5.4)
        2. При нажатиина кнопку **Уже зарегистрированы?** происходит перенаправление пользователя на страницу регистрации (5.3)
  3. Страница с формой регистрации
     1. Доступна только **неавторизованным** пользователям
     2. Содержит
        1. Тестовое поле для ввода почты
        2. Текстовое поле для ввода пароля
        3. Кнопку **Зарегистрироваться**
        4. Кнопку **Уже есть аккаунт?**
     3. Действия
        1. При нажатии на кнопку **Зарегистрироваться** отправляется запрос на API (4.4.1), в котором передаются введенные данные формы. Если введеные данные верны, то в ответе возвращается access токен, который сохраняется в куки, а также происходит переадресация на страницу со списком первооткрывателей (5.4)
        2. При нажатиина кнопку **Уже есть аккаунт?** перенаправляет пользователя на страницу авторизации (5.2)
  4. Страница со списком первооткрывателей
     1. Доступна **всем** пользователям
     2. Содержит
        1. Кнопку **«Новое открытие»**
           1. Доступна только **авторизированным** пользователям (**не администраторам**)
        2. Текстовое поле для поиска первооткрывателей по имени
        3. Первооткрывателей в виде карточек. Каждая карточка состоит из:
           1. Фото первооткрывателя
           2. Имени первооткрывателя
           3. Кнопки **«Открыть»**
           4. Кнопка **«Добавить»**

Доступна только **авторизированным** пользователям (**не администраторам**)

* + 1. Действия
       1. При нажатии на кнопку **«Открыть»** (5.4.2.2.3) происходит перенаправление пользователя на страницу с подробным описанием первооткрывателя (5.6)
       2. При нажатии на кнопку **«Добавить»** (5.4.2.2.4) отправляется запрос на API (4.1.6), а также меняется состояние кнопки **«Новое открытие»** (5.4.2.1)
       3. При нажатии на кнопку **«Новое открытие**» (5.4.2.1) происходит перенаправление пользователяна страницу с черновым открытием (5.7), если оно существует у пользователя
       4. При каждом наборе символов в поле поиска (5.4.2.2) отправляется запрос на API (4.1.1), который возвращает список первооткрывателей, отфильтрованный по имени
  1. Страница с таблицей первооткрывателей
     1. Доступна только **администраторам**
     2. Содержит
        1. Текстовое поле для поиска первооткрывателей по имени
        2. Таблицу с первооткрывателями, содержащую столбцы:
           1. № первооткрывателя
           2. Имя первооткрывателя
           3. Год рождения первооткрывателя
           4. Год смерти первооткрывателя
     3. Действия
        1. При нажатии на любую строку таблицы (5.2.2.2) происходит перенаправление пользователя на страницу редактирования первооткрывателя (5.9)
        2. При наборе символов в поле поиска (5.5.2.1) отправляется запрос на API (4.1.1), который фильтрует таблицу первооткрывателей по имени
  2. Страница с подробным описанием первооткрывателя
     1. Доступна **всем** пользователям
     2. Содержит:
        1. Имя первооткрывателя
        2. Фото первооткрывателя
        3. Биографию
        4. Год рождения
        5. Год смерти
  3. Страница чернового открытия
     1. Доступна только **авторизованным** пользователям (**не администраторам**)
     2. Содержит
        1. Карточки первооткрывателей в открытии. Каждая карточка содержит:
           1. Имя первооткрыватели
           2. Кнопку **«Удалить»**
        2. Блок с календарем для определения время открытия
        3. Текстового поля с названием дисциплины
        4. Текстового поля с ФИО преподавателя
        5. Кнопку **«Удалить»**
        6. Кнопку **«Отправить»**
     3. Действия
        1. При нажатии на кнопку **«Удалить»** (5.7.2.1.2) отправляется запрос на API (4.2.8), который удаляет первооткрывателя из открытия
        2. При нажатии на кнопку **«Удалить»** (5.7.2.5) отправляется запрос на API (4.2.7), который удаляет черновое открытие и перенаправляет пользователя на страницу со списком первооткрывателей (5.4)
        3. При нажатии на кнопку **«Отправить»** отправляетсязапрос на API (4.2.5), который формирует новое открытие и после чего перенаправляет пользователя на страницу с историей его открытий (5.8)
  4. Страница с историей открытий
     1. Доступна только **авторизованным** пользователям
     2. Содержит
        1. Таблицу с историей открытий, содержащую столбцы
           1. № Открытия
           2. Статус
           3. Имена первооткрывателей, совершивших открытие
           4. Дату формирования открытия
           5. Год открытия
           6. Кнопка подтвердить

Доступен только **администраторам**

* + - * 1. Кнопка отклонить

Доступен только **администраторам**

* + - 1. Фильтр по пользователю
         1. Доступен только **администраторам**
         2. Выполняется на стороне клиента
      2. Фильтр по дате формирования открытия в виде диапазона
      3. Фильтр по статусу открытия
         1. **Пользователю** доступны статусы

Черновик

В работе

Завершён

Отклонён

Удалён

* + - * 1. **Администратору** доступны статусы

В работе

Завершён

Отклонён

* + 1. Действия
       1. При изменении состояния **фильтров** (5.8.2.3 и 5.8.2.4) отправляется запрос на API (4.2.1), который обновляет таблицу с открытиями
       2. При нажатии на любую строчку таблицы происходит перенаправление пользователя на страницу открытия (5.8.2)
       3. При нажатии на кнопку **Подтвердить** (5.8.2.1.6) отправляется запрос на API (4.2.6), который меняет статус открытия с «в работе»на «завершён»
       4. При нажатии на кнопку **Отклонить** (5.8.2.1.7) отправляется запрос на API (4.2.6), который меняет статус открытия с «в работе» на «завершён»
  1. Страница редактирования / добавления первооткрывателя
     1. Доступна только **администраторам**
     2. Содержит
        1. Файловое поле, для загрузки / изменения фото первооткрывателя
        2. Текстовое поле для ввода / изменения имени первооткрывателя
        3. Текстовое поле для ввода / биографии первооткрывателя
        4. Текстовое поле для ввода даты рождения первооткрывателя
        5. Текстовое поле для ввода даты смерти первооткрывателя
        6. Кнопку **«Удалить»**
        7. Кнопку **«Сохранить»**
        8. Кнопку **«Добавить»**
     3. Действия
        1. При нажатии на кнопку **«Удалить»** (5.9.2.6)выполняется запрос на API (4.1.4), а также происходит переадресация на страницу со списком первооткрывателей (5.4)
        2. При нажатии на кнопку **«Сохранить»** (5.9.2.7)выполняется запрос на API (4.1.4), а также происходит переадресация на страницу со списком первооткрывателей (5.4)
        3. При нажатии на кнопку **«Добавить»** (5.9.2.8)выполняются два запроса на API: сначала на добавление нового первооткрывателя (4.1.5), а затем на обновление первооткрывателя (4.1.3), также происходит переадресация на страницу со списком первооткрывателей (5.4)

## Требования к программному обеспечению:

* 1. Серверная часть
     1. Операционная система (любая из)
        1. Linux **(**6.4.12 и выше)
        2. Windows (22H2 и выше)
     2. Веб-сервер Nginx (1.25.3 и выше)
     3. Minio (RELEASE.2023-12-14T18-51-57Z и выше)
     4. Redis (7.2 и выше)
     5. Python (3.10 и выше) с фреймворком Django (4.1 и выше)
     6. СУБД PostgreSQL (16.1 и выше)
     7. Доступ к интернету
  2. Клиентская часть
     1. Операционная система (любая из)
        1. Windows (22H2 и выше)
        2. MacOS (11.7.10 и выше)
        3. Linux **(**6.4.12 и выше)
     2. Веб-браузер (любой из)
        1. Safari (16.5.2 и выше)
        2. Chrome (119.0.6045 и выше)
        3. Opera (105.0.4970.16 и выше)
     3. Доступ к интернету

## Требования к аппаратному обеспечению:

* 1. Серверная часть
     1. Процессор минимум 2-ядерный с частотой от 2 ГГц
     2. Оперативная память от 4 Гб
     3. Место на жестком диске от 2 Гб
  2. Клиентская часть
     1. Процессор с частотой от 1ГГц
     2. Оперативная память от 512 Мб