2. Экспериментальный раздел

2.1 Описание программы

Программа поделена на 2 части: API и сайт - каждая часть имееь модульную структуру. При запуске двух частей сразу проект выполняется проект сайте. Backend часть приложения (API) написана на языке Python с использованием веб-фреймворка FastAPI в среде программирования PyCharm 2025.1.2 с использованием системы управления базой данных MySQL; Frontend часть приложения (сайт) написана на HTML, CSS, JavaScript с использованием фреймворка Vue.js в среде Visual Studio Code.

Схема взаимодействия модулей программы представлена на рисунке 2.1.1. Описание модулей и методов API представлено в таблице 2.1.1. Описание модулей и методов сайта представлено в таблице 2.1.2.

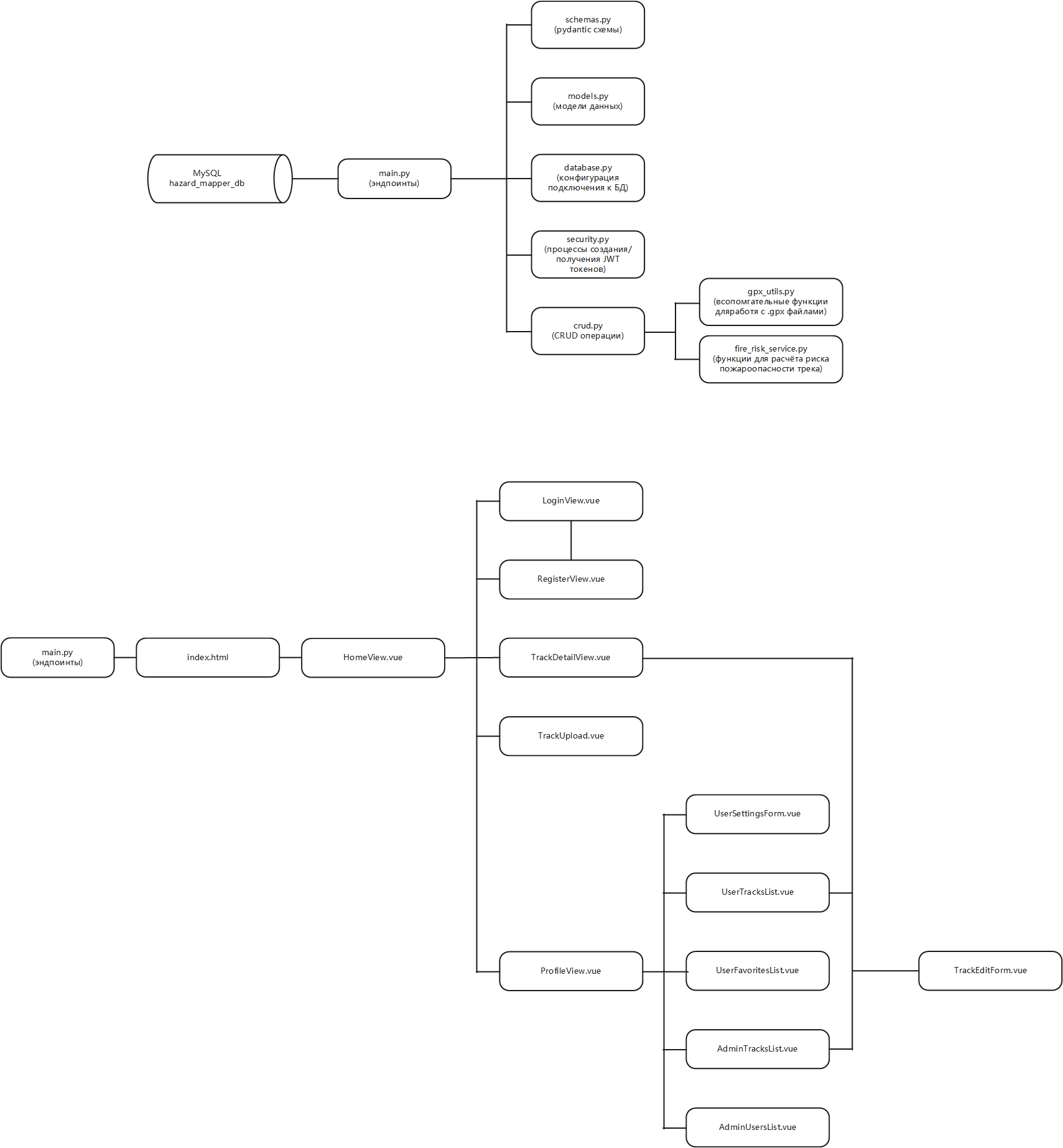


Рисунок 2.1.1 – Схема взаимодействия модулей

Таблица 2.1.1. - Описание модулей API

|  |  |
| --- | --- |
| Методы | Назначение |
| 1 | 2 |
| main.py | |
| @app.post("/register", response\_model=schemas.User)  def register\_user(user: schemas.UserCreate, db: Session = Depends(get\_db)) | Регистрация нового пользователя |
| @app.post("/login")  async def login(response: Response, user: schemas.UserLogin, db: Session = Depends(get\_db)) | Авторизация пользователя с установкой JWT-токена |
| @app.post("/logout")  async def logout(response: Response): | Выход из аккаунта (удаление сессионной куки) |
| @app.get("/users/me", response\_model=schemas.User)  async def get\_current\_user(session\_id: str = Cookie(default=None), db: Session = Depends(get\_db)) | Получение данных текущего пользователя |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| @app.put("/users/me", response\_model=schemas.User)  async def update\_current\_user(user\_data: schemas.UserUpdate, db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user)): | Обновление профиля пользователя |
| @app.put("/admin/users/{user\_id}/active", response\_model=schemas.User)  async def update\_user\_active(user\_id: int, active\_data: schemas.UserActiveUpdate, db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user)): | Активация/деактивация пользователя |
| @app.get("/users/{user\_id}", response\_model=schemas.User)  async def get\_user\_data(user\_id: int, db: Session = Depends(get\_db)): | Получение данных о пользователе по ID |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| @app.get("/users/me/tracks", response\_model=schemas.TrackPaginate)  async def get\_current\_user\_tracks(db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user), skip: int = Query(0, ge=0, description="Количество пропускаемых записей"), limit: int = Query(10, le=100, description="Максимальное количество записей"),) | Получение треков текущего пользователя |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| @app.get("/admin/users", response\_model=schemas.UserPaginate)  async def get\_all\_users(skip: int = Query(0, ge=0, description="Количество пропускаемых записей"), limit: int = Query(10, le=100, description="Максимальное количество записей"), db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user)) | Получение всех пользователей (админ) |
| @app.get("/tracks/load", response\_model=schemas.TrackPaginate) async def get\_all\_tracks(db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user, use\_cache=False), skip: int = Query(0, ge=0, description="Количество пропускаемых записей"), limit: int = Query(10, le=100, description="Максимальное количество записей"),): | Получение треков с пагинацией |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| @app.post("/tracks/upload")  async def upload\_track(title: str = Form(...), description: str | None = Form(None), file: UploadFile = File(...), db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user)): | Загрузка трека из GPX-файла |
| @app.get("/tracks/{track\_id}", response\_model=schemas.TrackDetail) async def get\_track\_details(track\_id: int, db: Session = Depends(get\_db)): | Детальная информация о треке |
| @app.put("/tracks/{track\_id}", response\_model=schemas.Track) async def update\_track(track\_id: int, track\_data: schemas.TrackUpdate, db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user)): | Обновление данных трека |
| @app.delete("/tracks/{track\_id}") async def delete\_track(track\_id: int, db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user)): | Удаление трека |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| @app.post("/tracks/{track\_id}/favorite", response\_model=schemas.Track) async def favorite\_track(track\_id: int, db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user)): | Добавление трека в избранное |
| @app.delete("/tracks/{track\_id}/favorite", response\_model=schemas.Track) async def unfavorite\_track(track\_id: int, db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user)): | Удаление трека из избранного |
| @app.post("/tracks/{track\_id}/comments", response\_model=schemas.Comment) async def create\_comment(track\_id: int, comment\_data: schemas.CommentCreate, db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user)): | Создание комментария к треку |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| @app.get("/favorites", response\_model=schemas.TrackPaginate)  async def get\_favorite\_tracks(db: Session = Depends(get\_db), current\_user: models.User = Depends(get\_current\_user), skip: int = Query(0, ge=0, description="Количество пропускаемых записей"), limit: int = Query(10, le=100, description="Максимальное количество записей"),): | Получение избранных треков |
| @app.get("/tracks/{track\_id}/fire\_risk", response\_class=HTMLResponse)  async def get\_fire\_risk\_map(request: Request, track\_id: int, db: Session = Depends(get\_db)): | Получение карты пожароопасности для трека |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| schemas.py | |
| class UserBase(BaseModel) | Базовые данные пользователя (email, username) |
| class UserCreate(UserBase) | Данные для регистрации (с паролем) |
| class UserLogin(BaseModel) | Данные для входа (email, пароль) |
| class User(UserBase) | Полные данные пользователя (включая id, статусы) |
| class UserUpdate(BaseModel) | Обновляемые данные профиля |
| class UserPaginate(BaseModel) | Пагинированный список пользователей |
| class UserActiveUpdate(BaseModel) | Статус активности пользователя |
| class CommentBase(BaseModel) | Базовые данные комментария |
| class CommentCreate(BaseModel) | Данные для создания комментария |
| class Comment(CommentBase) | Полные данные комментария (с метаданными) |
| class CommentWithAuthor(Comment) | Комментарий с информацией об авторе |
| class TrackBase(BaseModel) | Базовые данные трека |
| class TrackPointBase(BaseModel) | Координаты точки трека |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| class Track(TrackBase) | Основные данные трека (с флагом избранного) |
| class TrackCreate(TrackBase) | Данные для создания трека |
| class TrackDetail(Track) | Детали трека (с владельцем и точками) |
| class TrackStats(BaseModel) | Статистика трека (дистанция, высоты) |
| class TrackUpload(BaseModel) | Данные для загрузки трека (с файлом) |
| class TrackUpdate(BaseModel) | Обновляемые данные трека |
| class TrackPaginate(BaseModel) | Пагинированный список треков |
| class TrackPoint(TrackPointBase) | Точка трека с идентификаторами |
| class FavoriteBase(BaseModel) | Базовые данные избранного |
| class FavoriteCreate(FavoriteBase) | Данные для создания избранного |
| class Favorite(FavoriteBase) | Полные данные избранного |
| database.py | |
| def get\_db() | Генератор сессий БД для внедрения зависимостей |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| security.py | |
| def create\_session\_token(data: dict, expires\_delta: timedelta = None) | Создаёт JWT-токен с указанными данными и сроком действия |
| def verify\_session\_token(session\_id: Optional[str] = Cookie(None)) | Проверяет JWT-токен из куки и возвращает ID пользователя |
| crud.py | |
| def get\_user(db: Session, user\_id: int) | Получение пользователя по ID |
| def get\_all\_users(db: Session, skip: int = 0, limit: int = 100, user\_id: int = None) | Получение всех пользователей с пагинацией |
| def get\_user\_by\_email(db: Session, email: str) | Поиск пользователя по email |
| def get\_user\_by\_username(db: Session, username: str) | Поиск пользователя по никнейму |
| def create\_user(db: Session, user: schemas.UserCreate) | Создание нового пользователя |
| def update\_user(db: Session, user\_data: schemas.UserUpdate, current\_user: models.User) | Обновление данных пользователя |
| def update\_user\_active(db: Session, user\_id: int, is\_active: bool) | Активация/деактивация пользователя |
| def authenticate\_user(db: Session, email: str, password: str) | Проверка авторизационных данных |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| def get\_track(db: Session, track\_id: int) | Получение трека по ID |
| def update\_track(db: Session, track\_id: int, track\_data: schemas.TrackUpdate) | Обновление данных трека |
| def get\_tracks(db: Session, skip: int = 0, limit: int = 100) | Получение треков с пагинацией |
| def get\_tracks\_by\_user(db: Session, user\_id: int, skip: int = 0, limit: int = 100) | Получение треков пользователя |
| def create\_track\_with\_points(db: Session, track\_data: schemas.TrackCreate, points: list, image: bytes, user\_id: int) | Создание трека с точками маршрута |
| def get\_track\_with\_details(db: Session, track\_id: int) | Получение детальной информации о треке |
| def delete\_track(db: Session, track\_id: int) | Удаление трека |
| def add\_to\_favorites(db: Session, user\_id: int, track\_id: int) | Добавление трека в избранное |
| def remove\_from\_favorites(db: Session, user\_id: int, track\_id: int) | Удаление трека из избранного |
| def get\_favorite\_tracks(db: Session, user\_id: int, skip: int = 0, limit: int = 100) | Получение избранных треков пользователя |
| def is\_favorite(db: Session, user\_id: int, track\_id: int) | Проверка наличия трека в избранном |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| def create\_comment(db: Session, comment\_data: schemas.CommentCreate, user\_id: int, track\_id: int) | Создание комментария к треку |
| gpx\_utils.py | |
| def calculate\_difficulty(distance\_km, elevation\_gain) | Рассчитывает сложность трека (1-5) по дистанции и набору высоты |
| def parse\_gpx(gpx\_content: str) -> Tuple[List[dict], dict]: | Парсит GPX-файл, возвращает точки трека и статистику |
| def generate\_track\_image(points: List[dict]) -> bytes: | Генерирует изображение трека с подложкой карты |
| def get\_track\_region(points: List[dict]) -> str | None: | Определяет регион трека через обратное геокодирование |
| fire\_risk\_service.py | |
| def load\_fire\_data(): | Загружает исторические данные о пожарах из CSV-файлов |
| def calculate\_fire\_risk(track\_points): | Рассчитывает уровень пожароопасности для точек трека |
| def generate\_risk\_map(track\_id: int, db): | Генерирует интерактивную карту с визуализацией риска пожаров |

Таблица 2.1.2. - Описание модулей сайта

|  |  |  |
| --- | --- | --- |
| Методы | Назначение | |
| 1 | 2 | |
| api.js | | |
| login(email, password) | | Авторизация пользователя |
| register(username, email, password) | | Регистрация нового пользователя |
| logout() | | Выход из аккаунта |
| getCurrentUser() | | Получение данных текущего пользователя |
| getUserData(user\_id) | | Получение данных пользователя по ID |
| getAllUsers(skip, limit) | | Получение списка пользователей (админ) |
| updateCurrentUser(userData) | | Обновление профиля текущего пользователя |
| updateUserActive(userId, isActive) | | Активация/деактивация пользователя |
| getUserTracks(skip, limit) | | Получение треков текущего пользователя |
| getAllTracks(skip, limit) | | Получение всех треков с пагинацией |
| uploadTrack(formData) | | Загрузка нового трека (GPX-файл) |
| updateTrack(trackId, updateData) | | Обновление данных трека |

Продолжение таблицы 2.1.2

|  |  |
| --- | --- |
| 1 | 2 |
| deleteTrack(trackId) | Удаление трека |
| getTrackDetails(trackId) | Получение детальной информации о треке |
| favoriteTrack(trackId) | Добавление трека в избранное |
| unfavoriteTrack(trackId) | Удаление трека из избранного |
| getFavoriteTracks(skip, limit) | Получение избранных треков |
| createComment(trackId, commentData) | Создание комментария к треку |
| getFireRiskMap(trackId) | Получение карты пожароопасности |
| auth.js | |
| login(email, password) | Авторизация пользователя |
| register(username, email, password) | Регистрация нового пользователя |
| logout() | Выход из системы |
| fetchUser() | Загрузка данных текущего пользователя |
| favorites.js | |
| toggleFavorite(track) | Переключает статус избранного для трека |
| loadFavorites() | Загружает список избранных треков с сервера |

Продолжение таблицы 2.1.2

|  |  |
| --- | --- |
| 1 | 2 |
| isFavorite(trackId) | Проверяет наличие трека в избранном |
| updateTracksFavoriteStatus(tracks) | Обновляет статус избранного для массива треков |
| AuthHeader.vue | |
| handleLogoutButtonClick() | Обработчик выхода из системы с редиректом |
| LoginForm.vue | |
| handleSubmit() | Обработчик отправки формы, вызывает метод login хранилища |
| RegisterForm.vue | |
| computed passwordMismatch | Вычисляемое свойство для проверки совпадения паролей |
| handleSubmit() | Обработчик регистрации с валидацией паролей |
| TrackCard.vue | |
| computed difficultyText | Вычисляемое свойство для текста сложности |
| fetchUser() | Загрузка данных автора трека |
| formatDate() | Форматирование даты создания трека |

Продолжение таблицы 2.1.2

|  |  |
| --- | --- |
| 1 | 2 |
| toggleFavorite() | Обработчик переключения статуса избранного |
| difficultyText | Вычисляемое свойство для текста сложности |
| onMunted | Хук жизненного цикла для загрузки данных |
| TrackUpload.vue | |
| handleSubmit() | Обработчик отправки формы и загрузки данных |
| initMap() | Инициализация и отрисовка карты Leaflet |
| nextTick(initMap) | Ожидание обновления DOM перед инициализацией карты |
| onBeforeUnmount | Хук для очистки карты при размонтировании |
| MapView.vue | |
| onMounted | Хук инициализации карты при монтировании компонента |
| watch | Реактивное обновление карты при изменении точек |

Продолжение таблицы 2.1.2

|  |  |
| --- | --- |
| 1 | 2 |
| initMap() | Основная функция инициализации карты |
| updateMap() | Функция обновления маршрута при изменении данных |
| TrackEditForm.vue | |
| handleSubmit() | Сохранение изменений трека |
| cancel() | Отмена редактирования |
| UserFavoritesList.vue | |
| handleTrackUpdate() | Обработчик обновления трека |
| refresh() | Инициация обновления данных |
| UserSettingsForm.vue | |
| handleSubmit() | Отправляет данные профиля на сервер, проверяет пароли, обрабатывает ответ |
| UserTracksList.vue | |
| loadMyTracks() | Загружает список пользовательских треков |
| refresh() | Обновляет список треков с начальной пагинацией |

Продолжение таблицы 2.1.2

|  |  |
| --- | --- |
| 1 | 2 |
| confirmDelete(trackId) | Удаляет трек с подтверждением через alert |
| startEditing(trackId) | Активирует режим редактирования выбранного трека |
| cancelEdit() | Прекращает редактирование без сохранения изменений |
| handleTrackUpdate(updatedTrack) | Обновляет данные трека в списке после сохранения |
| nextPage() | Переход к следующей странице пагинации |
| prevPage() | Переход к предыдущей странице пагинации |
| formatDate(dateString) | Преобразует дату в локальный формат дд.мм.гггг |
| HomeView.vue | |
| fetchTracks() | Загружает список треков с сервера |
| handleTrackUpdate(updatedTrack) | Обновляет данные трека в интерфейсе |
| nextPage() | Переход к следующей странице треков |

Продолжение таблицы 2.1.2

|  |  |
| --- | --- |
| 1 | 2 |
| prevPage() | Переход к предыдущей странице треков |
| computed currentPage | Возвращает текущий номер страницы |
| computed totalPages | Рассчитывает общее количество страниц |
| ProfileView.vue | |
| formatDate(dateString) | Преобразует дату в локальный формат дд.мм.гггг |
| loadFavoriteTracks() | Загружает список избранных треков пользователя |
| handleUserUpdate(updatedUser) | Обновляет данные пользователя в профиле |
| deleteTrack(trackId) | Удаляет трек и обновляет списки треков |
| handleTrackUpdate(updatedTrack) | Обновляет данные трека в интерфейсе |
| TrackDetailView.vue | |
| fetchTrackData() | Загружает данные текущего трека с сервера |

Продолжение таблицы 2.1.2

|  |  |
| --- | --- |
| 1 | 2 |
| openFireRiskMap() | Открывает карту пожароопасности в новом окне |
| formatDate(dateString) | Преобразует дату в локализованный формат дд.мм.гггг, чч:мм |
| addComment() | Добавляет новый комментарий к треку |

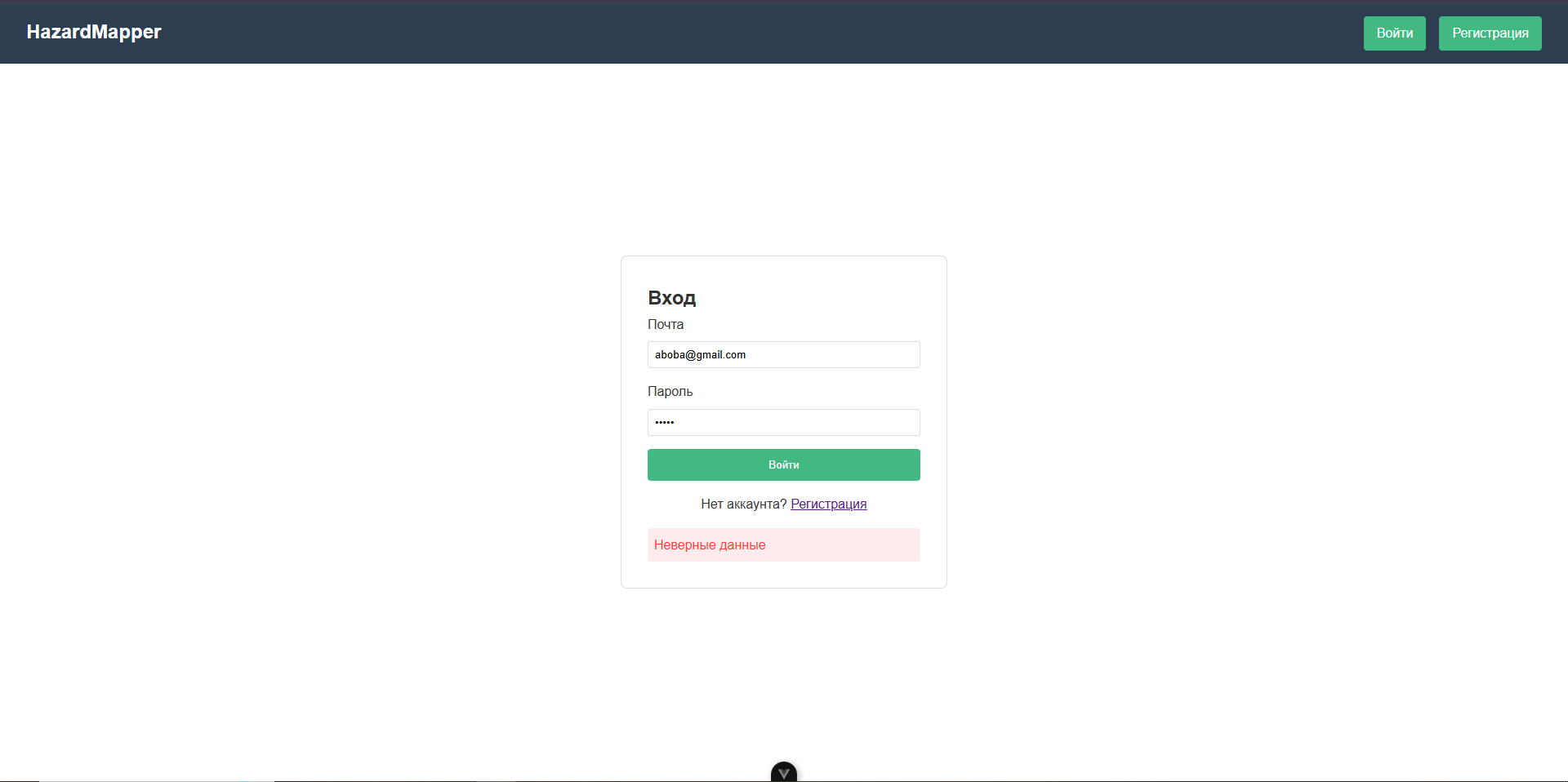
Код программы представлен в приложении Г.

2.2 Протокол тестирования программного продукта

В ходе тестирования программного продукта на корректных и некорректных данных не было обнаружено ошибок, которые влияли бы на работу самого программного продукта и всей системы.

Данный программный продукт удовлетворяет всем предъявленным требованиям, имеет комфортный интерфейс и интуитивно понятный функционал, исключает появления системных ошибок.

В таблицах 2.2.1 – 2.2.11 представлены протоколы тестирования.

Таблица 2.2.1 – Тестирование входа с некорректными данными

|  |  |  |
| --- | --- | --- |
| Поле | Описание | |
| 1 | 2 | |
| Дата теста | 20.06.2025 | |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий | |
| Заголовок/название теста | | Тестирование входа с некорректными данными |
| Резюме испытания | | Отображение сообщения ошибки при попытке входа с неверным паролем |
| Этапы теста | | 1 запуск API;  2 запуск сайта;  3 нажатие на кнопку «Войти»;  4 введите почту;  5 введите пароль;  6 нажатие на кнопку «Войти». |
| Тестовые данные | | 1 почта «aboba@gmail.com»;  2 пароль «wrong»; |
| Ожидаемый результат | | Отображение сообщения «Неверные данные» |
| Фактический результат | | Сообщение отобразилось, входа не последовало |

Рисунок 2.2.1 – Результат тестирования входа с некорректными данными

Таблица 2.2.2 – Тестирование входа без данных

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 20.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование входа без данных |
| Резюме испытания | Отображение сообщения об ошибки при попытке входа без данных |
| Этапы теста | 1 запуск API;  2 запуск сайта;  3 нажатие на кнопку «Войти»;  4 нажатие на кнопку «Войти». |
| Тестовые данные | 1 почта «»;  2 пароль «»; |
| Ожидаемый результат | Отображение сообщения «Заполните это поле» около первого пустого поля |
| Фактический результат | Сообщение отобразилось |

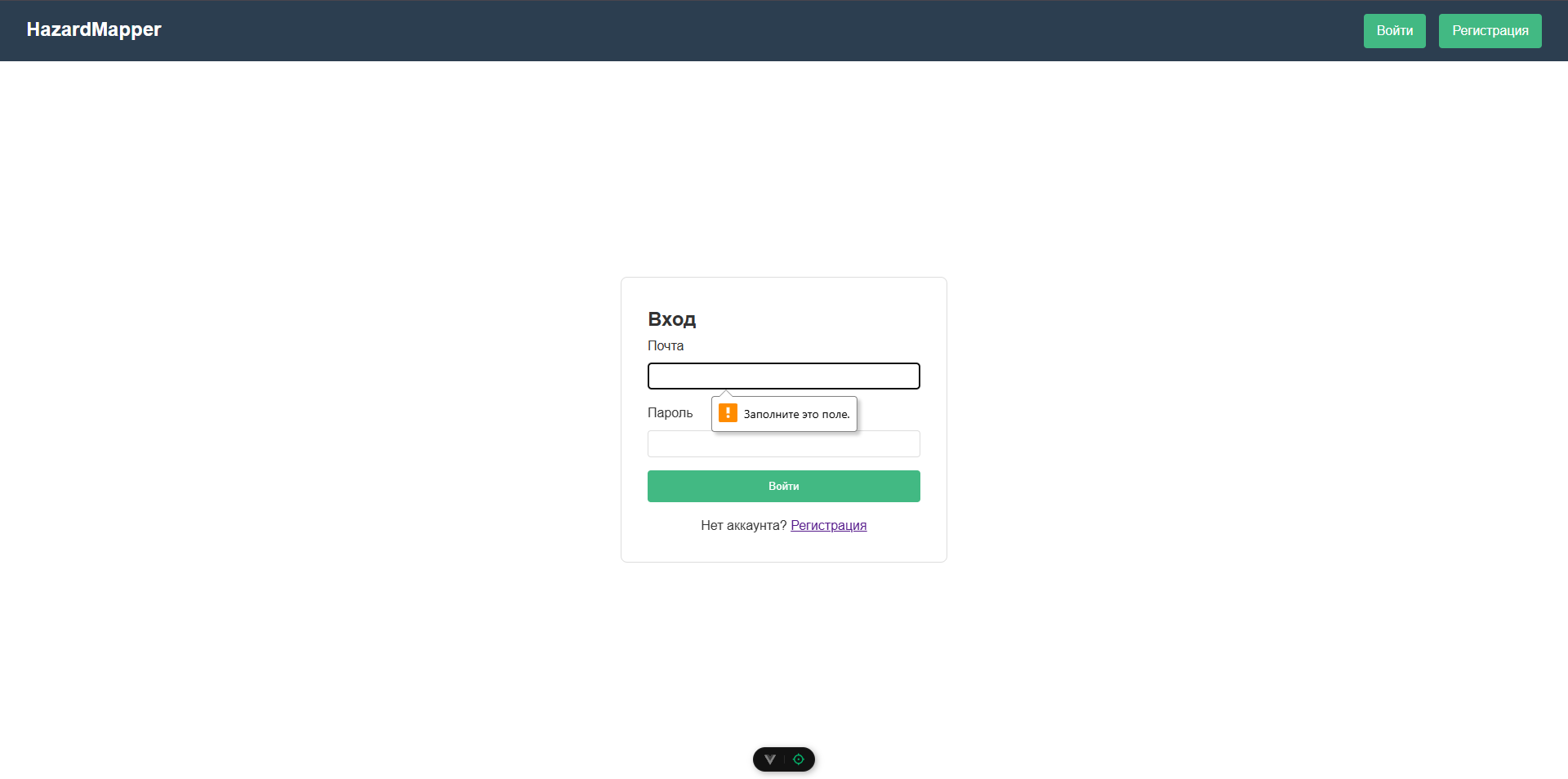


Рисунок 2.2.2 – Тестирование входа пользователя без данных

Таблица 2.2.3 – Тестирование входа пользователя корректными данными

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 20.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование входа с корректными данными |
| Резюме испытания | Переадресация на главную страницу сайта с отображающимся никнеймом в шапке сайта |
| Этапы теста | 1 запуск API;  2 запуск сайта;  3 нажатие на кнопку «Войти»;  4 введите почту;  5 введите пароль;  6 нажатие на кнопку «Войти». |
| Тестовые данные | 1 Электронная почта «aboba@gmail.com»;  2 пароль «aboba». |
| Ожидаемый результат | Переадресация на главную страницу сайта |
| Фактический результат | Вход в аккаунт произошел, передаресация произошла, никнейм отображается в шапке сайта |

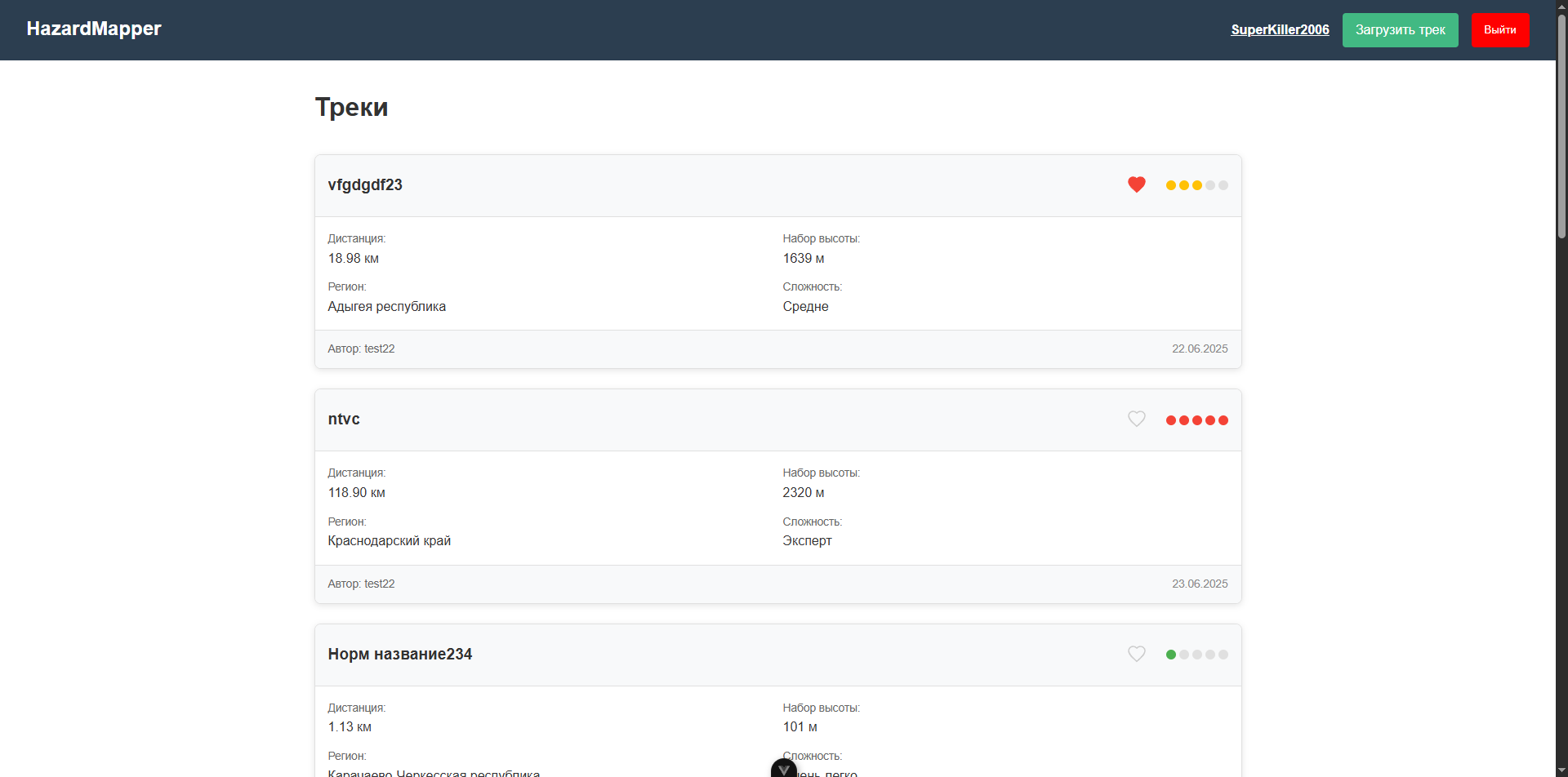


Рисунок 2.2.3 – Результат авторизации с корректными данными

Таблица 2.2.4 – Тестирование загрузка трека с неккоректными данными

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 20.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование загрузки трека с пустым названием |
| Резюме испытания | Отображение ошибки с сообщением «Название не может быть пустым» |
| Этапы теста | 1 запуск API;  2 запуск сайта;  3 нажатие на кнопку «Войти»;  4 введите почту;  5 введите пароль;  6 нажатие на кнопку «Войти»;  7 нажмите на кнопку «Загрузить трек»;  8 загрузите .gpx файл;  9 нажмите на кнопку «Загрузить трек». |
| Тестовые данные | 1 Электронная почта «aboba@gmail.com»;  2 пароль «aboba».  3. .gpx файл «B0P0L3QA.gpx» |
| Ожидаемый результат | Отображение ошибки с сообщением «Ошибка загрузки: Название не может быть пустым» |
| Фактический результат | Ошибка отобразилась, трек не загрузился |

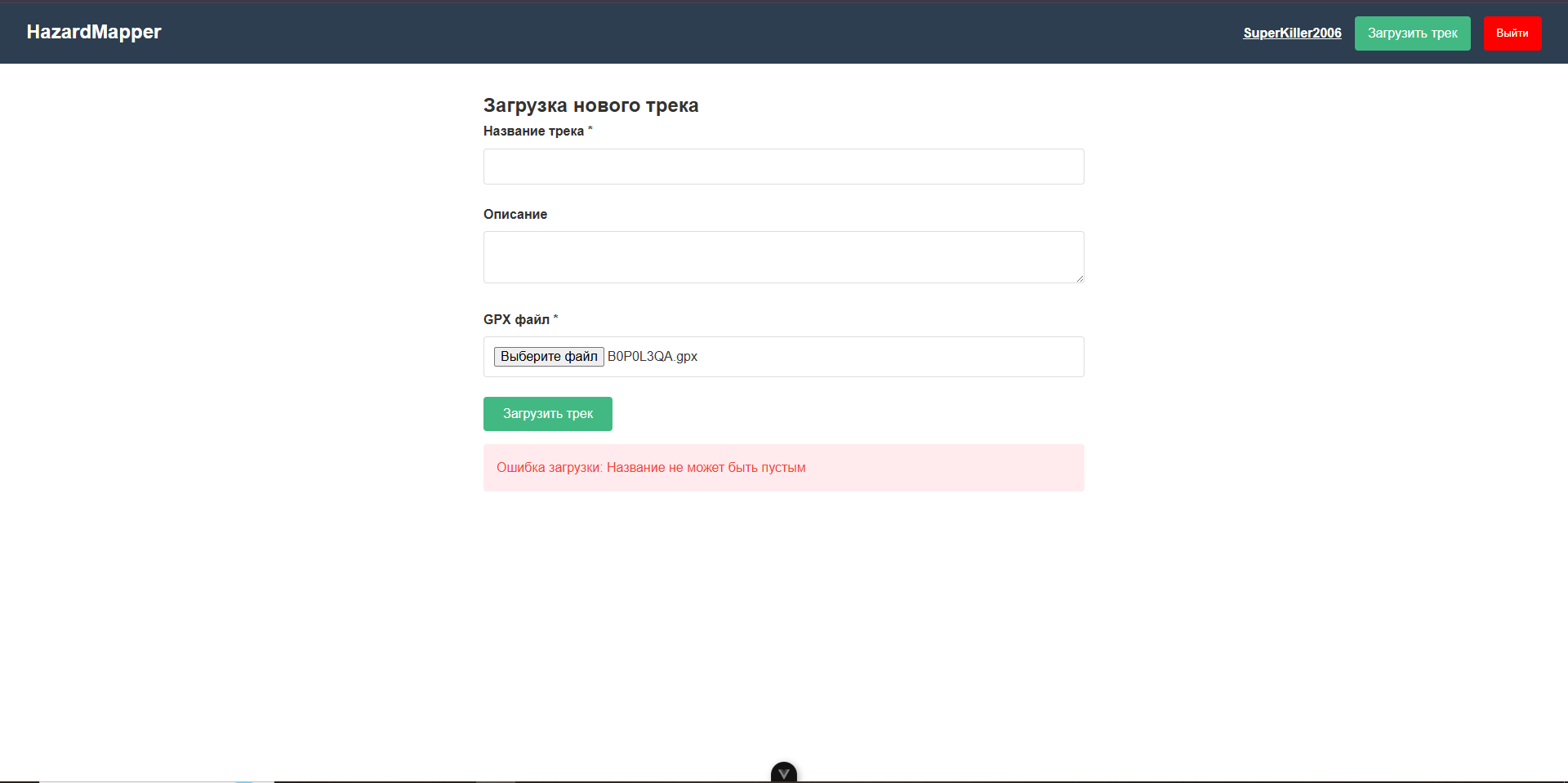
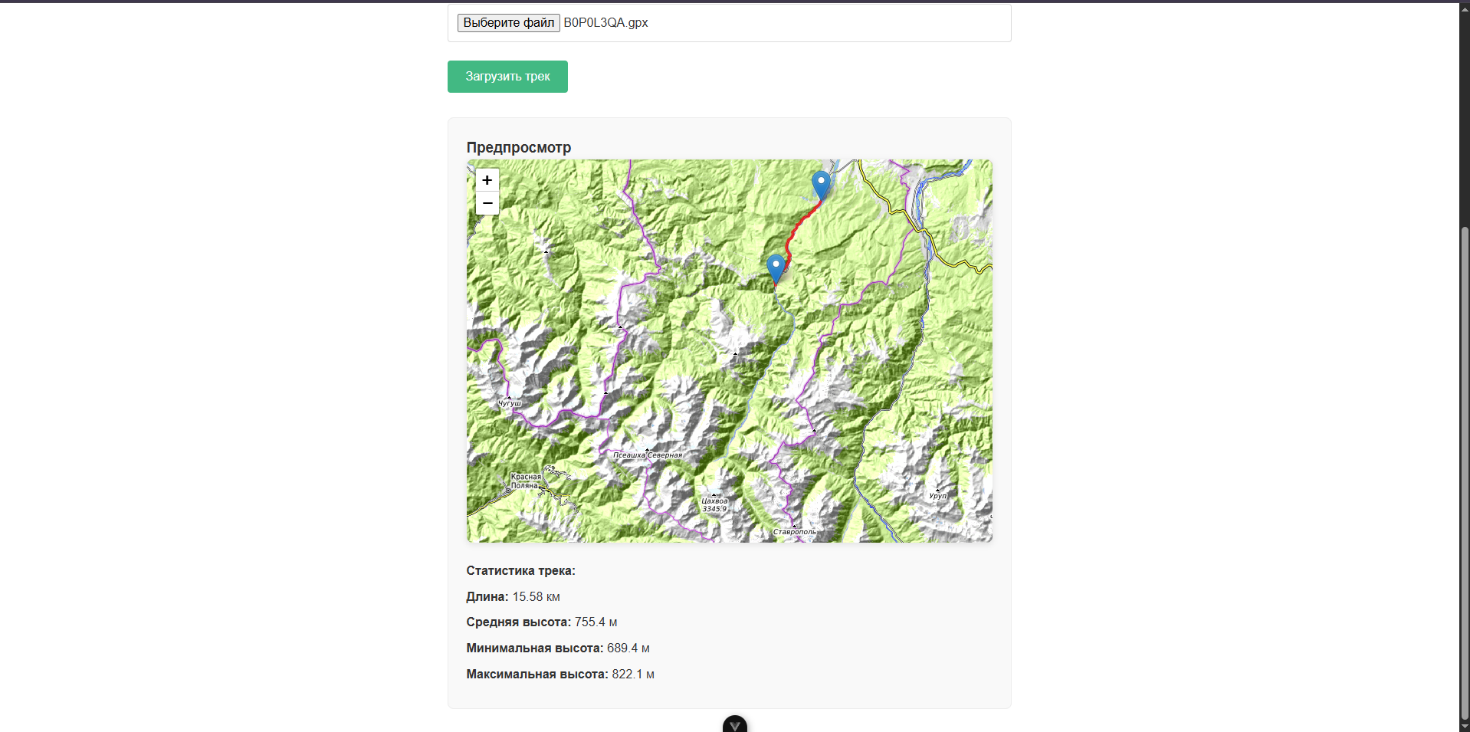


Рисунок 2.2.4 – Тестирование загрузки трека с пустым названием

Таблица 2.2.5 – Тестирование добавления результатов с корректными данными

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 20.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование добавления трека с корректными данными |
| Резюме испытания | Отображение предосмотра трека |
| Этапы теста | 1 запуск API;  2 запуск сайта;  3 нажатие на кнопку «Войти»;  4 введите почту;  5 введите пароль;  6 нажатие на кнопку «Войти»;  7 нажмите на кнопку «Загрузить трек»;  8 введите название трека;  9 загрузите .gpx файл;  10 нажмите на кнопку «Загрузить трек». |

Продолжение таблицы 2.2.5

|  |  |
| --- | --- |
| 1 | 2 |
| Тестовые данные | 1 Электронная почта «aboba@gmail.com»;  2 пароль «aboba».  3 название трека «Тестовый трек»  4 .gpx файл «B0P0L3QA.gpx» |
| Ожидаемый результат | Отображение предосмотра трека |
| Фактический результат | Предосмотр отобразился, трек загружен |

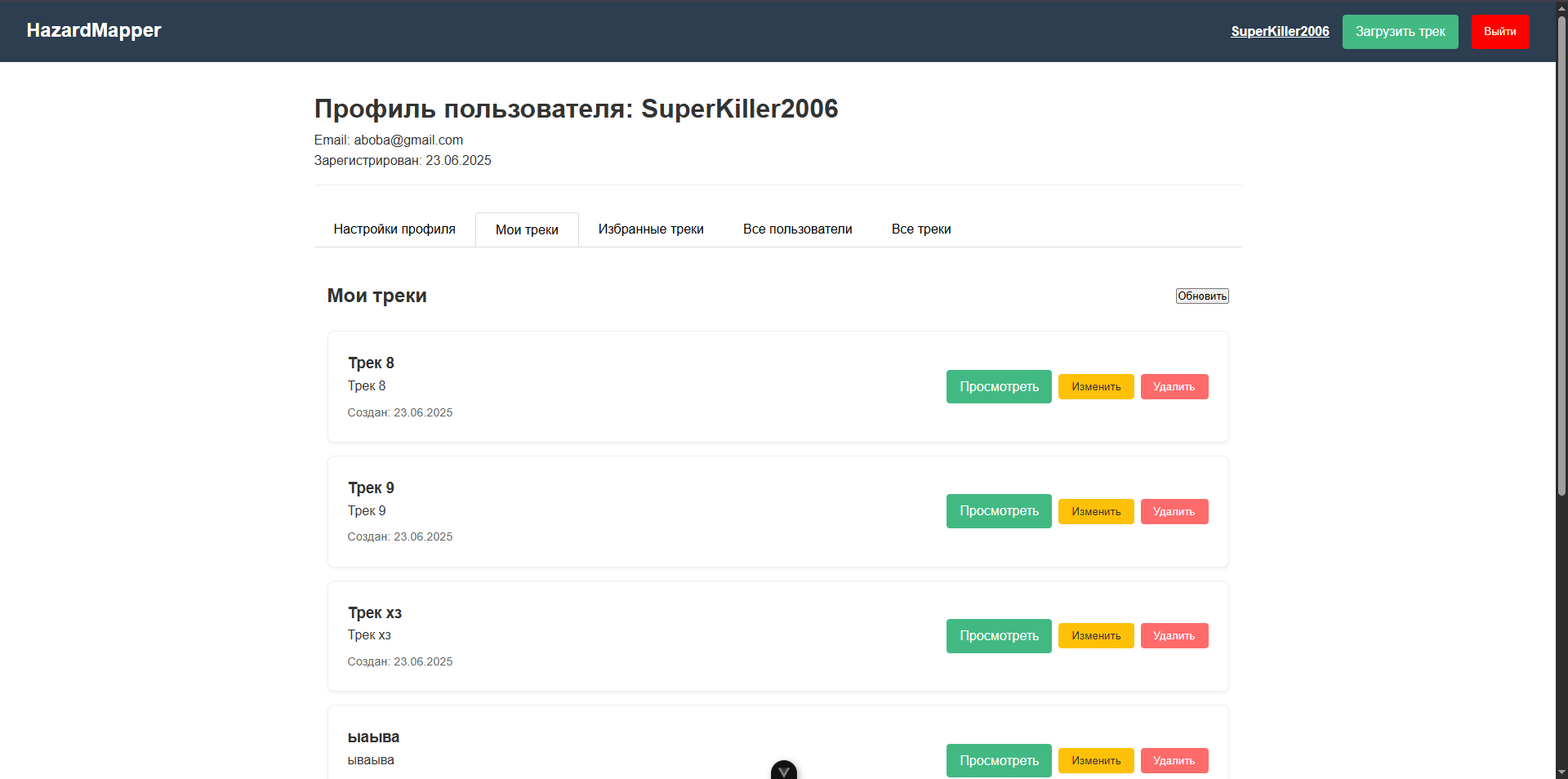
Рисунок 2.2.5 – Тестирование добавления трека с корректными данными

Таблица 2.2.6 – Тестирование удаление трека

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 20.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование удаление трека |
| Резюме испытания | Корректное отображение списка треков без удалённого трека |

Продолжение таблицы 2.2.6

|  |  |
| --- | --- |
| 1 | 2 |
| Этапы теста | 1 запуск API;  2 запуск сайта;  3 нажатие на кнопку «Войти»;  4 введите почту;  5 введите пароль;  6 нажатие на кнопку «Войти»;  7 нажать на свой никнейм;  8 перейти во вкладку «Мои треки»;  9 нажать на кнопку «Удалить» первого трека;  10 подтвердить действие; |
| Тестовые данные | 1 Электронная почта «aboba@gmail.com»;  2 пароль «aboba». |
| Ожидаемый результат | Отображение нового списка треков без удалённого трека |
| Фактический результат | Новый список без удалённого трека был отображён, запись удалена |

Рисунок 2.2.6.1 – Тестирование удаление трека. Список до удаления

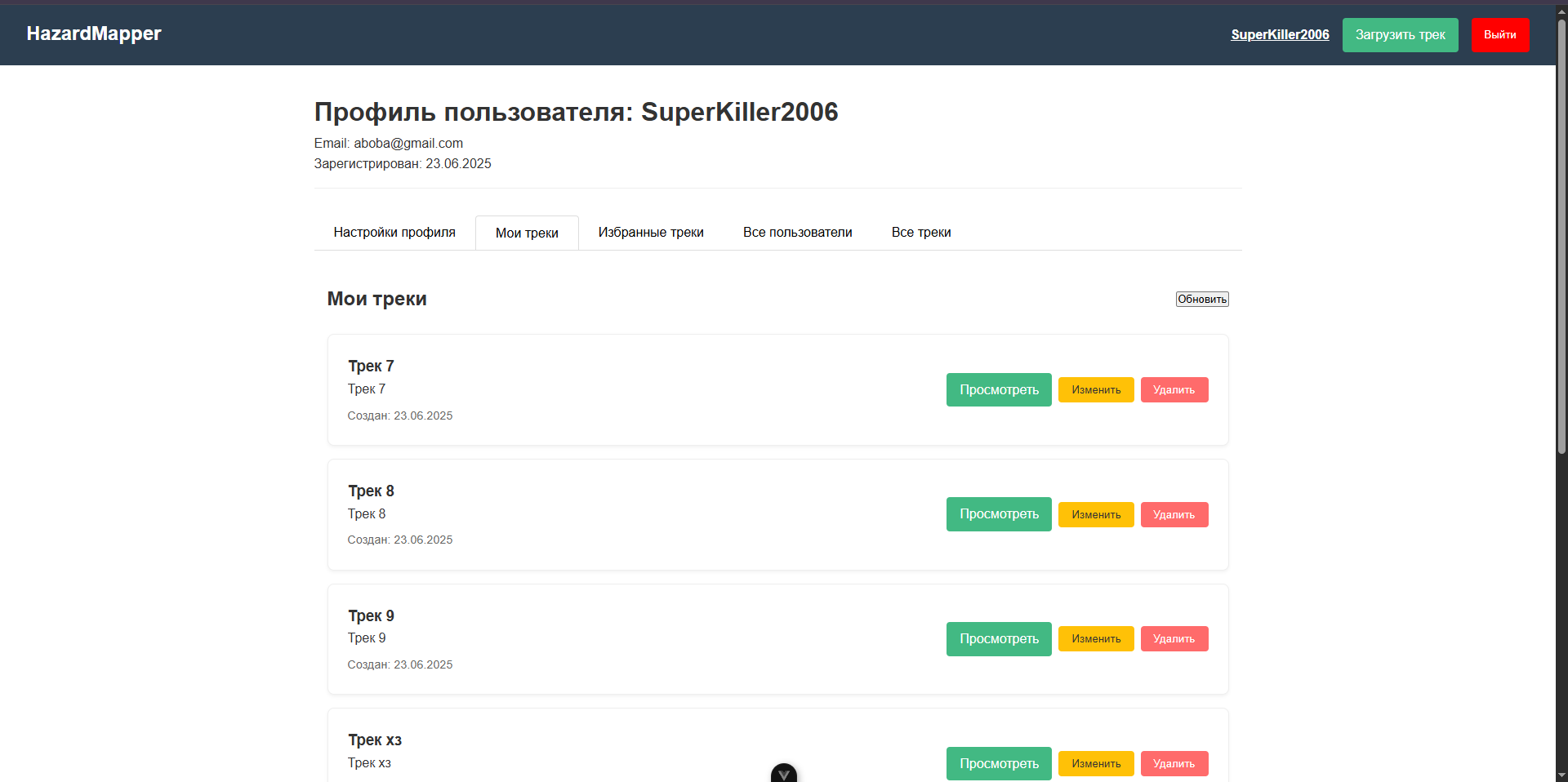
Рисунок 2.2.6.2 – Тестирование удаление трека. Список после удаления

Таблица 2.2.7 – Тестирование изменение данных трека на пустые данные

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 20.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование изменение данных трека на пустые данные |
| Резюме испытания | Отображение сообщения с ошбикой «Ошибка обновления: Название не может быть пустым» |
| Этапы теста | 1 запуск API;  2 запуск сайта;  3 нажатие на кнопку «Войти»;  4 введите почту;  5 введите пароль;  6 нажатие на кнопку «Войти»;  7 нажать на свой никнейм;  8 перейти во вкладку «Мои треки»;  9 нажать кнопку «Изменить» первого трека;  10 ввести новое название;  11 нажать кнопку «Сохранить». |

Продолжение таблицы 2.2.7

|  |  |
| --- | --- |
| Тестовые данные | 1 Электронная почта «aboba@gmail.com»;  2 пароль «aboba»;  3 новое название « »; |
| Ожидаемый результат | Отображение сообщения «Ошибка обновления: Название не может быть пустым» |
| Фактический результат | Сообщение отобразилось, запись не удалена |

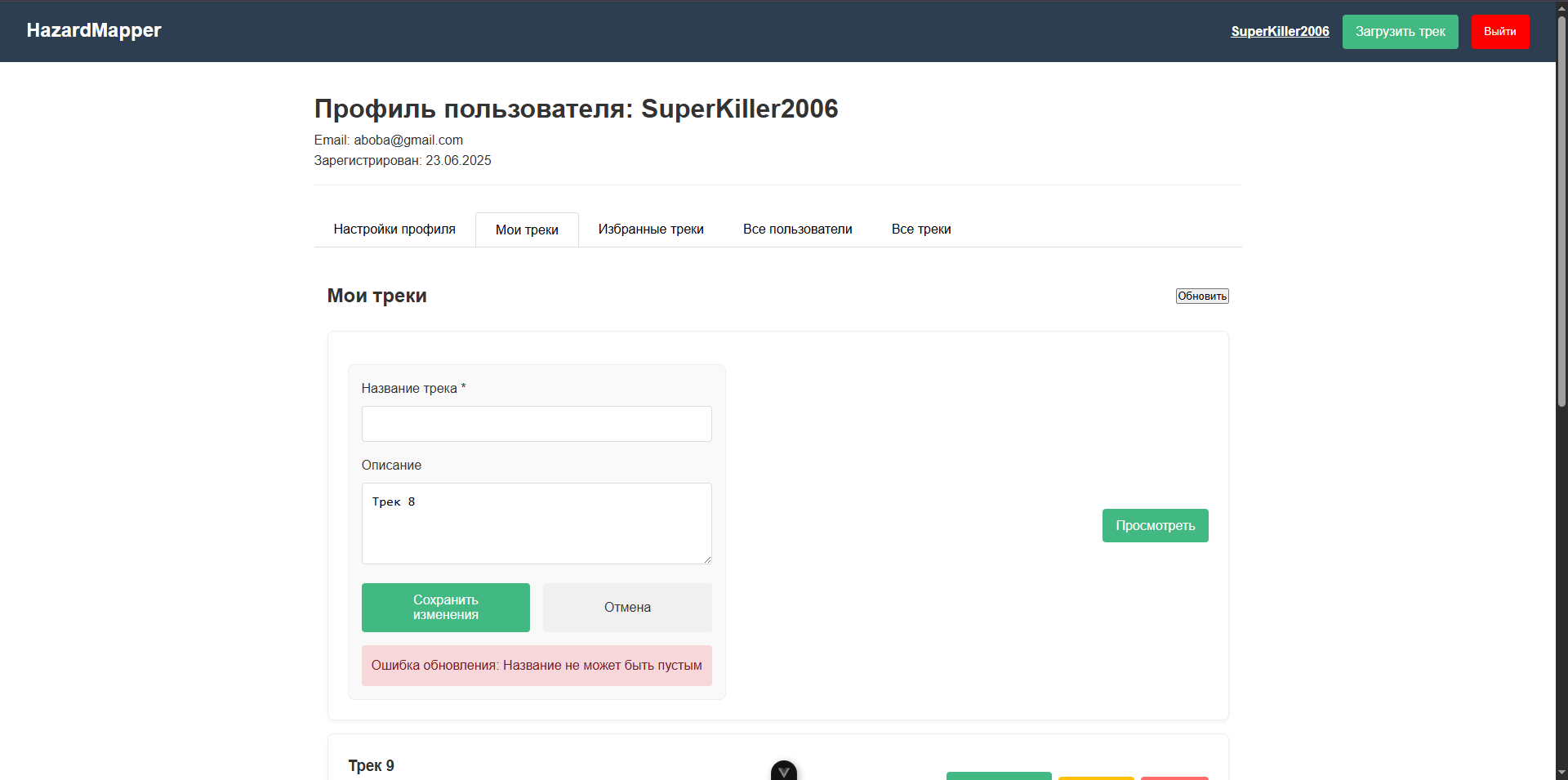


Рисунок 2.2.7 – Тестирование изменение данных трека на пустые данные

Таблица 2.2.8 – Тестирование добавление трека в избранное

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 20.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | естирование добавление трека в избранное |
| Резюме испытания | Отображение трека как избранного на главной странице |

Продолжение таблицы 2.2.8

|  |  |
| --- | --- |
| Этапы теста | 1 запуск API;  2 запуск сайта;  3 нажатие на кнопку «Войти»;  4 введите почту;  5 введите пароль;  6 нажатие на кнопку «Войти»;  7 нажмите на кнопку сердечка первого трека; |
| Тестовые данные | 1 Электронная почта «aboba@gmail.com»;  2 пароль «aboba». |
| Ожидаемый результат | Отображение трека, помеченного как избранное (карсное сердечко) |
| Фактический результат | Трек отображён как избранный, запись добавлена в избранное |

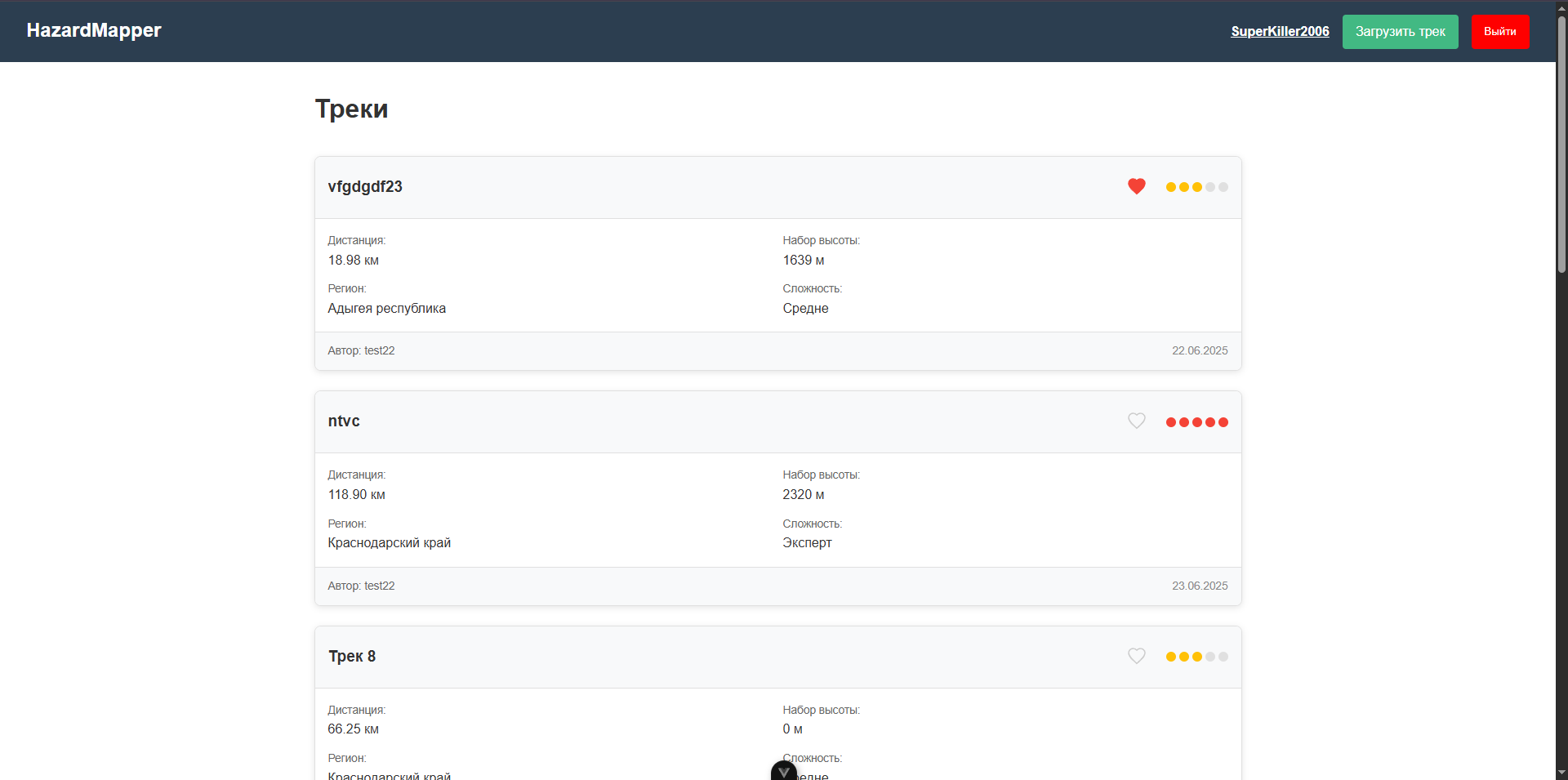
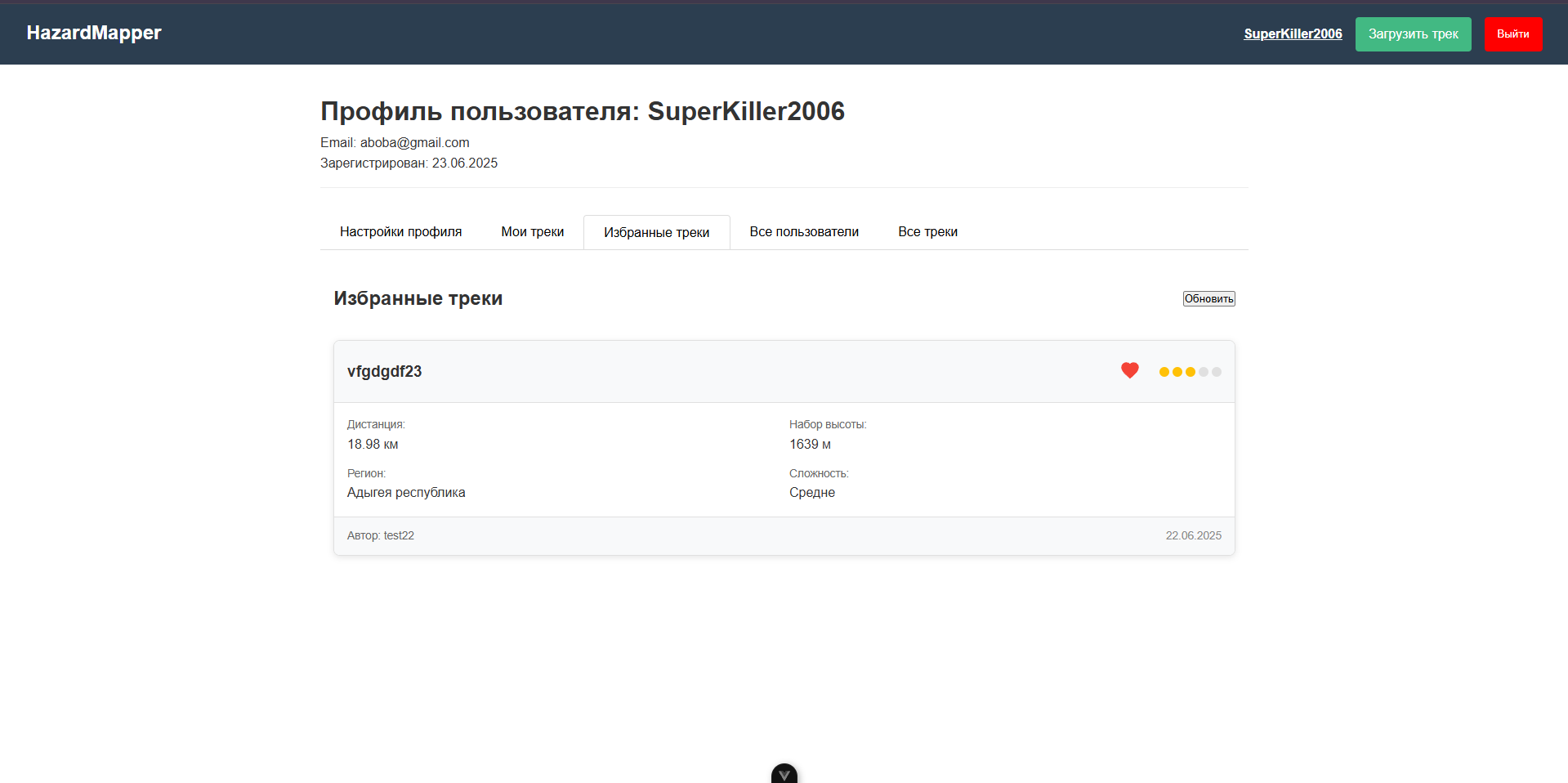


Рисунок 2.2.8 – Тестирование добавление трека в избранное

Таблица 2.2.9 – Тестирование удаления трека из избранного в личном профиле

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 20.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование удаления трека из избранного в личном профиле |

Продолжение таблицы 2.2.9

|  |  |
| --- | --- |
| Резюме испытания | Отображение нового списка избранного без удалённого трека |
| Этапы теста | 1 запуск API;  2 запуск сайта;  3 нажатие на кнопку «Войти»;  4 введите почту;  5 введите пароль;  6 нажатие на кнопку «Войти»;  7 нажать на свой никнейм;  8 перейти во вкладку «Избранные треки»;  9 нажмите на кнопку сердечка первого трека; |
| Тестовые данные | 1 Электронная почта «aboba@gmail.com»;  2 пароль «aboba». |
| Ожидаемый результат | Отображение нового списка избранного без удалённого трека |
| Фактический результат | Новый список избранного отобразился, запись был удалена из избранного |

Рисунок 2.2.9.1 – Тестирование удаления трека из избранного в личном профиле. Список до удаления.

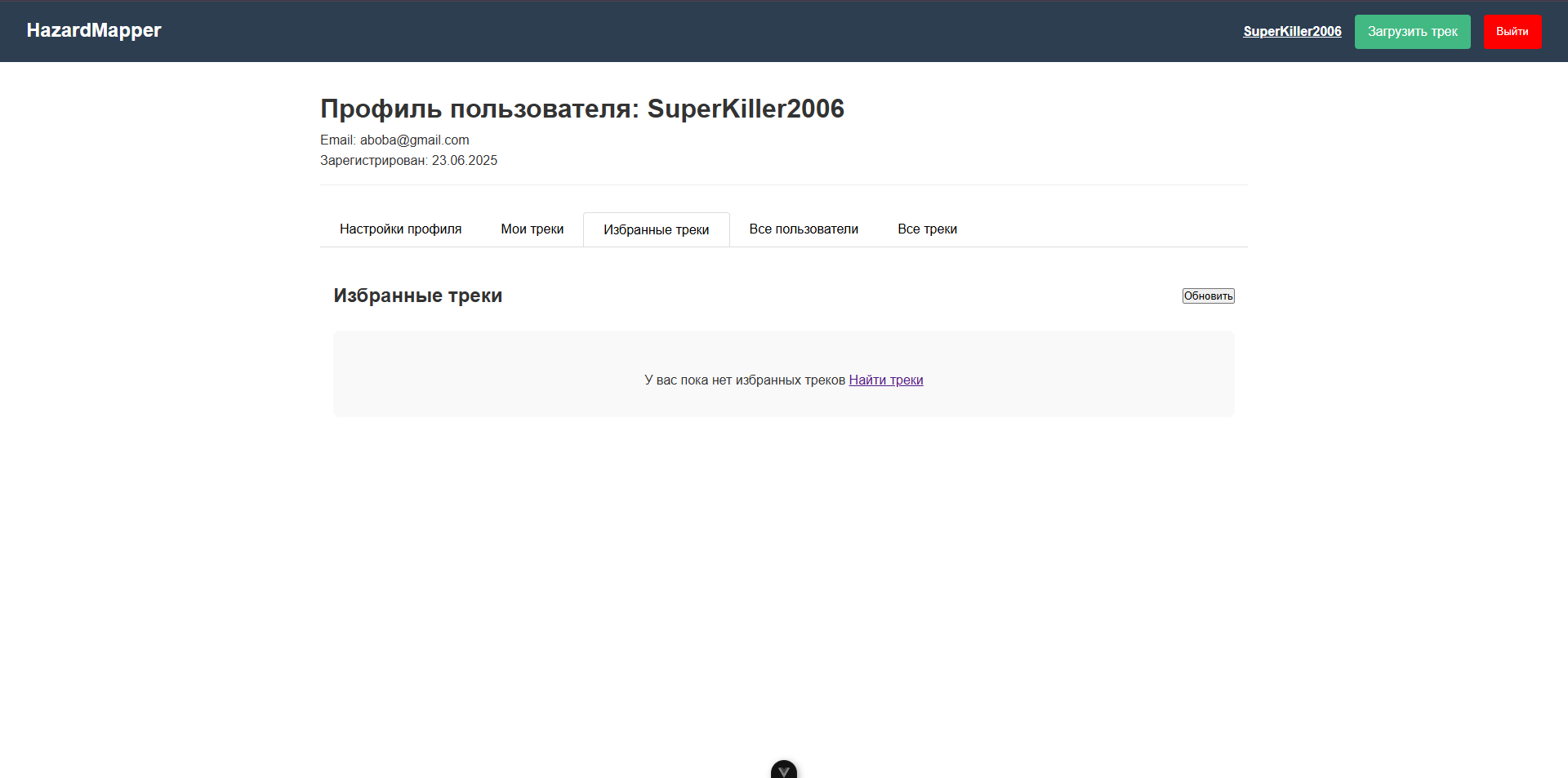
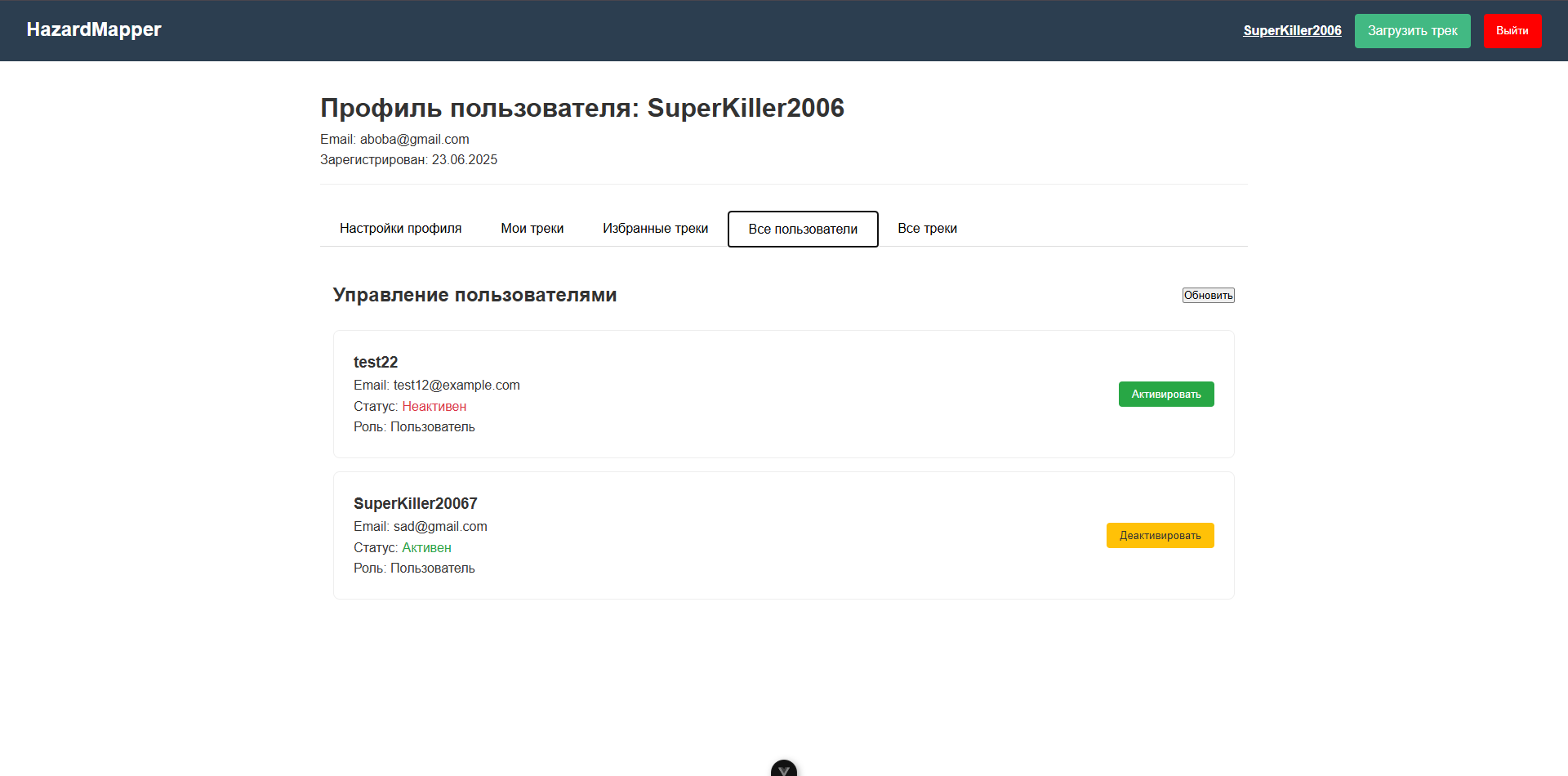


Рисунок 2.2.9.2 – Тестирование удаления трека из избранного в личном профиле. Список после удаления.

Таблица 2.2.10 – Тестирование деактивация пользователя для Админа

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 20.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование деактивация пользователя |
| Резюме испытания | Отображение пользователе в списке как неактивного |
| Этапы теста | 1 запуск API;  2 запуск сайта;  3 нажатие на кнопку «Войти»;  4 введите почту;  5 введите пароль;  6 нажатие на кнопку «Войти»;  7 нажать на свой никнейм;  8 перейти во вкладку «Все пользователи»;  9 нажмите на кнопку «Деактивировать» второго пользователя; |
| Тестовые данные | 1 Электронная почта «aboba@gmail.com»;  2 пароль «aboba». |
| Ожидаемый результат | Пользователь отображается со статусом «Неактивен» |
| Фактический результат | Пользователь отобразился как «Неактвен» |



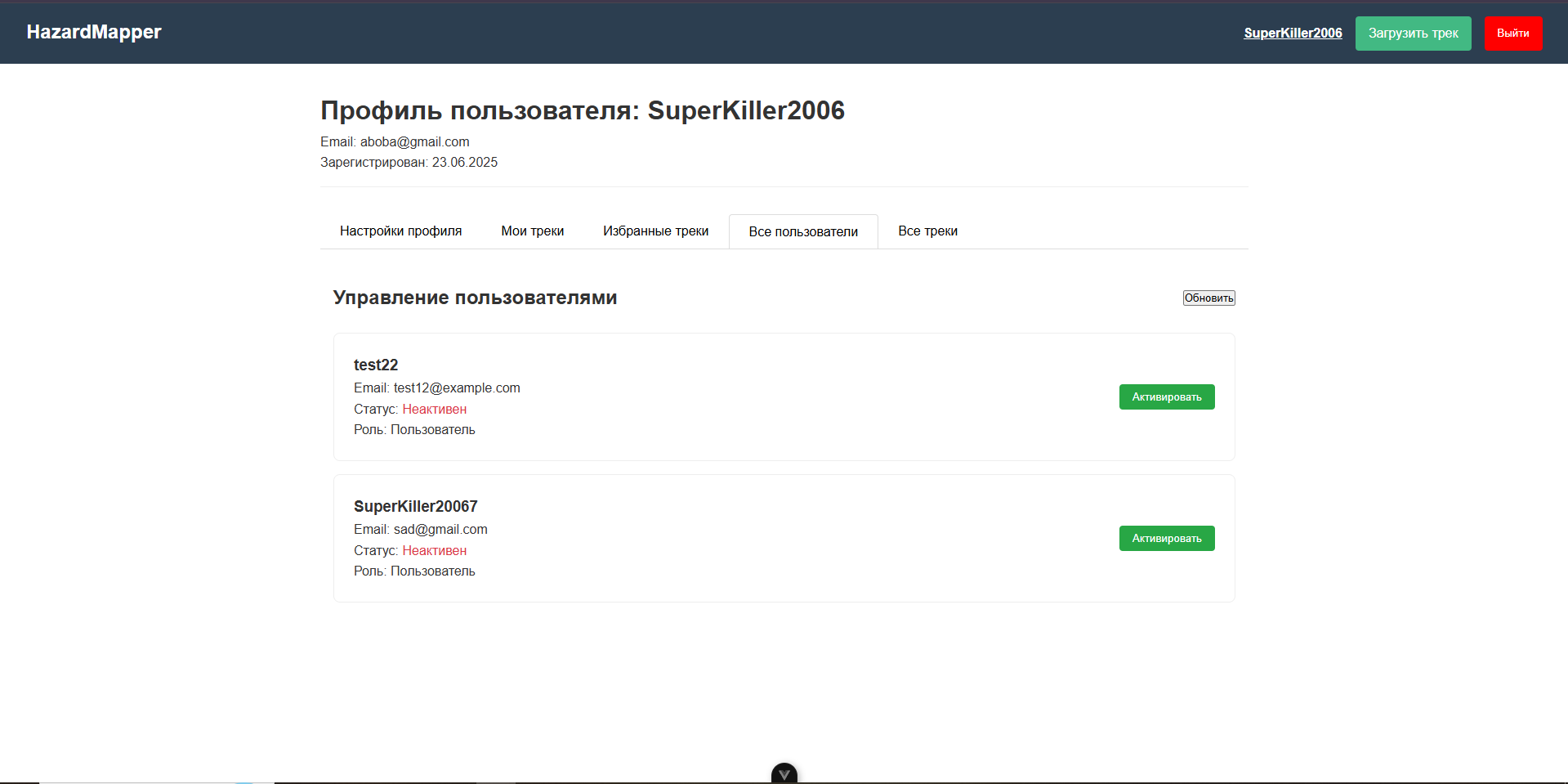
Рисунок 2.2.10.1 – Тестирование деактивация пользователя для Админа. Пользователь до деактивации

Рисунок 2.2.10.2 – Тестирование деактивация пользователя для Админа. Пользователь после деактивации

Таблица 2.2.11 – Тестирование регистрации с корректными данными

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 20.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование регистрации с корректными данными |
| Резюме испытания | Регистрация нового пользователя в системе |
| Этапы теста | 1 запуск API;  2 запуск сайта;  3 нажатие на кнопку «Регистрация»;  4 введите никнейм;  5 введите почту;  6 введите пароль;  7 введите пароль снова;  8 нажатие на кнопку «Зарегистрироваться»; |
| Тестовые данные | 1 Никнейм «test»  2 Электронная почта «test@gmail.com»;  3 пароль «test12345;  4 пароль снова «test12345». |
| Ожидаемый результат | Переадресация на главную страницу, отображение никнейма в шапке сайта, регистрация нового пользователя |
| Фактический результат | Переадресация на главную страницу, отображение никнейма в шапке сайта, регистрация нового пользователя |

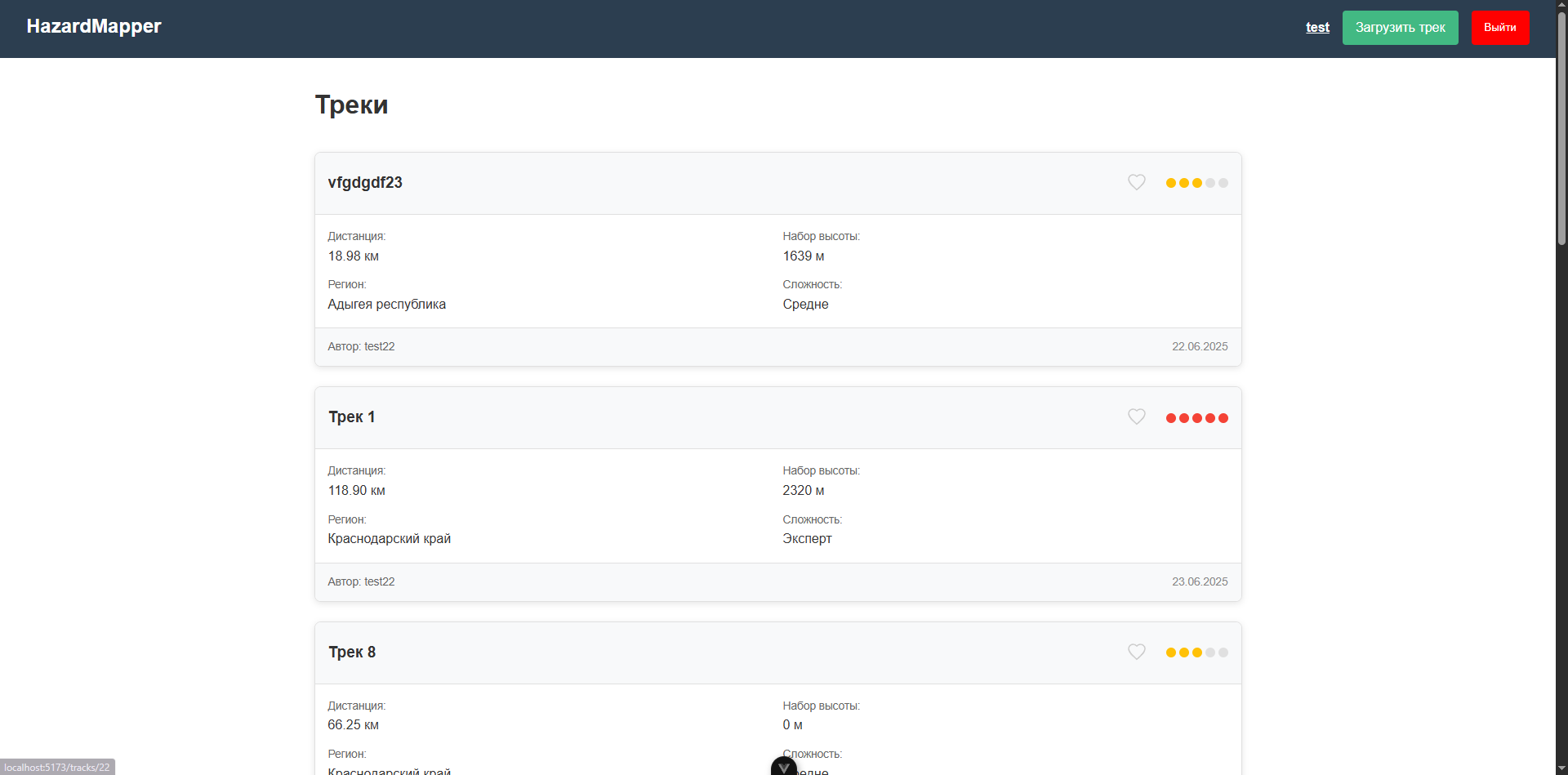


Рисунок 2.2.11 – Тестирование регистрации с корректными данными

2.3 Руководство пользователя

Назначение системы

Программное обеспечение «Система картографирования природных опасностей на треках» предназначено для мониторинга природных пожаров, надежного хранении, обработке и наглядном отображении пространственных данных.

Основной целью данной информационной системы является обеспечение точного хранения, обработки и анализа данных о треках, авторизованных пользователей, и самого важного – рисков пожаров различных треков. Условия применения системы

Программное обеспечение разрабатывается для персональной вычислительной техники со следующими характеристиками:

* Microsoft Windows 10;
* процессор 2 ГГц;
* 2 ГБ ОЗУ;
* 10 ГБ свободного пространства на диске;
* разрешение экрана монитора не менее 1920 × 1080;

Система предназначена для использования администраторами сайта, рядовыми пользователями интернета, учебных организаций и другими лицами, имеющими навыки работы с графическими интерфейсами современных операционных систем. Подготовка системы к работе

Для правильного функционирования сайта необоходимо запустить API: откройте командную строку и при помощи команды «cd» перейдите в папку с проектом, затем пропищите команду «uvicorn app.main:app» (рисунок 2.3.1).

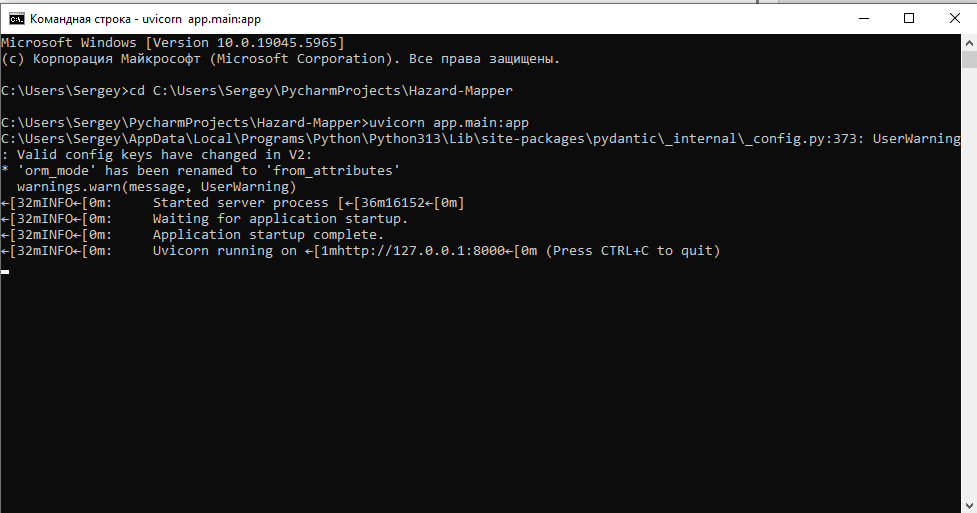


Рисунок 2.3.1 – Запуск API

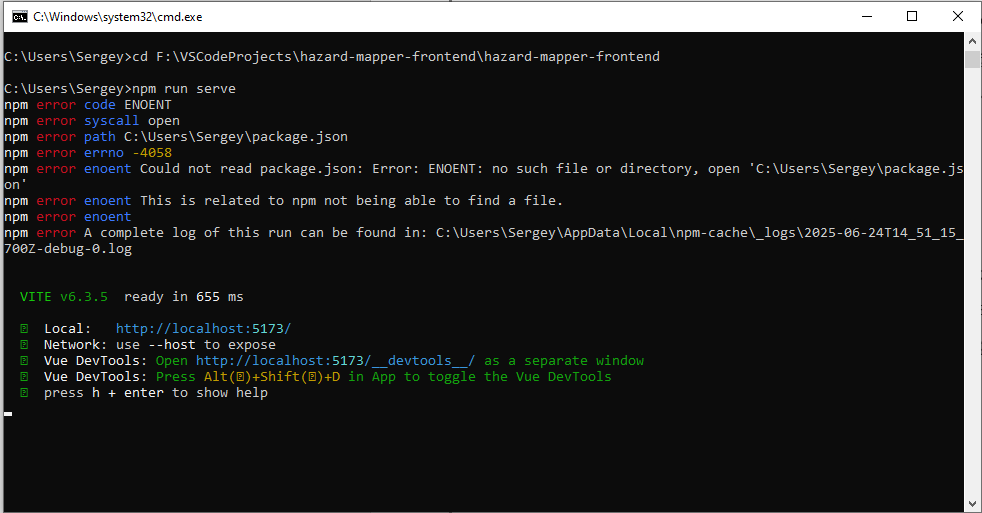
Когда API запущен, необходимо запустить сайт на том же localhost: откройте командную строку и при помощи команды «cd» перейдите в папку с проектом сайта и напишите комнду «npm run dev», затем откройте сайт по указанной в командной строке ссылке. (рисунки 2.3.2.1, 2.3.2.2)

Рисунок 2.3.2.1 – Запуск сайта

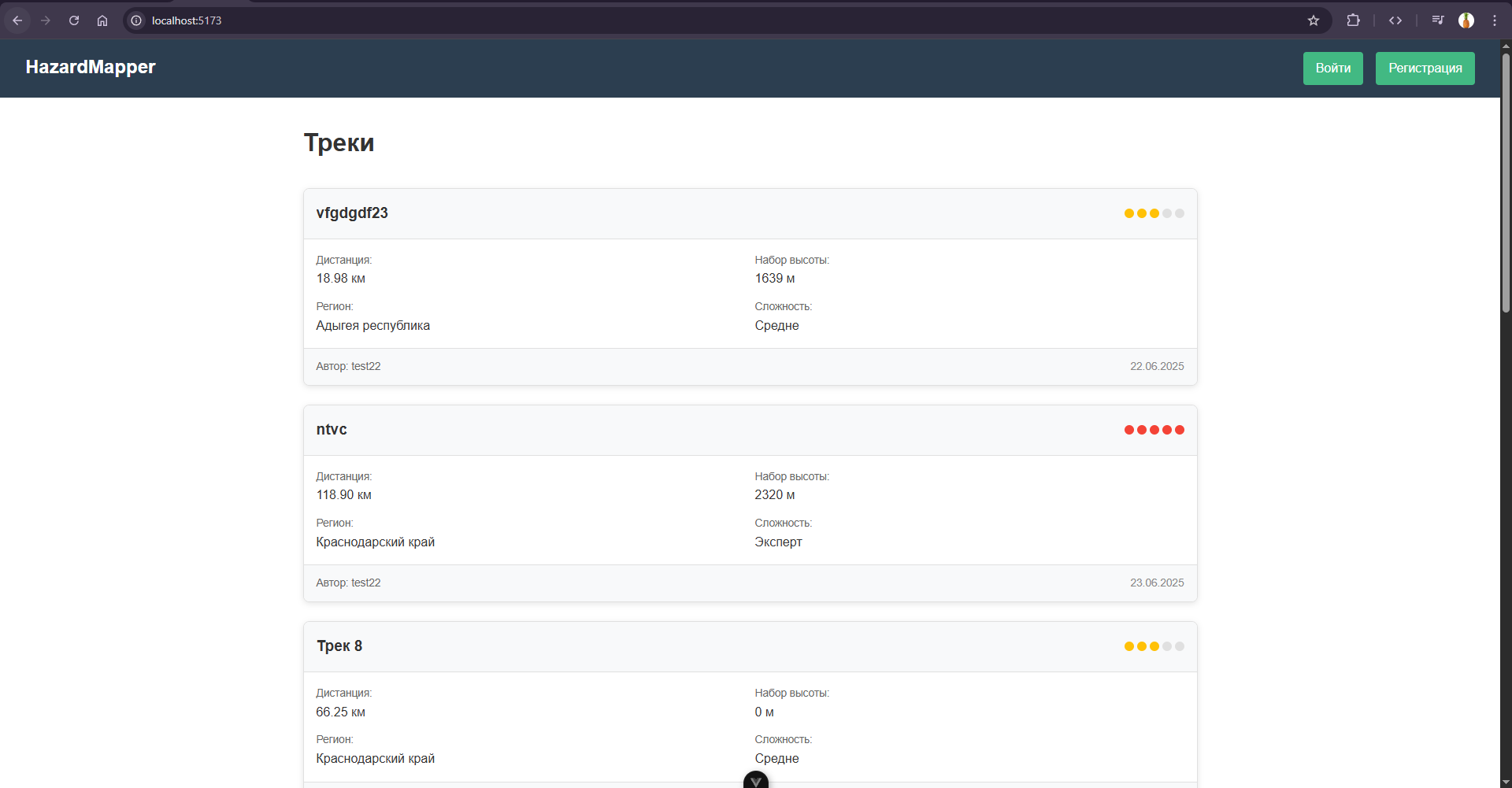


Рисунок 2.3.2.2 – Результат перехода по ссылке

Описание операций

После запуска приложения, необходимо зарегистрироваться на сайте, чтобы было доступно больше функций. Необходимо нажать на кнопку «Регистрация» на шапке сатйа, произойдёт переадресация на форму регистрации. (рисунок 2.3.3)

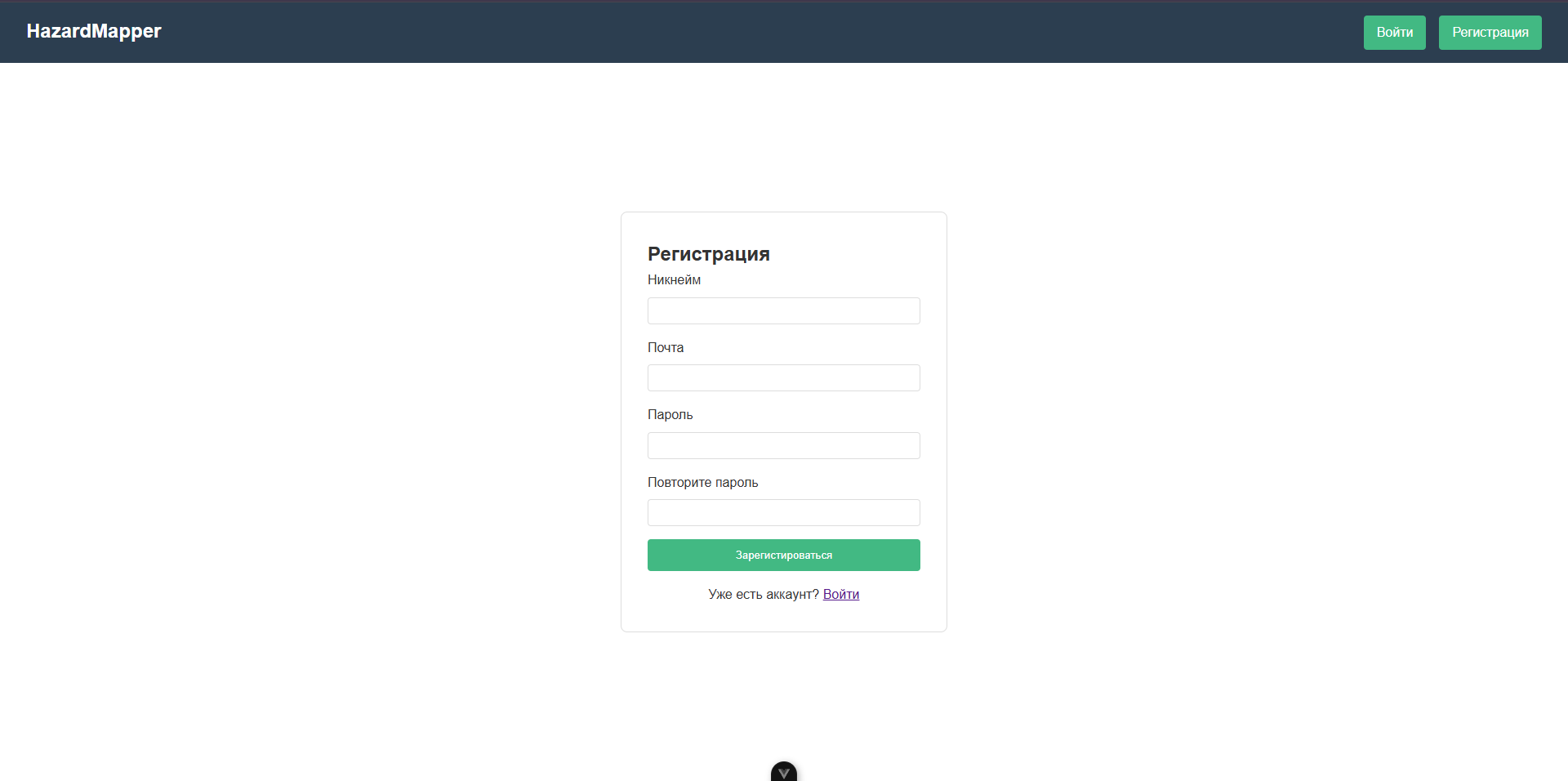


Рисунок 2.3.3 – Форма регистрации

Чтобы пройти регистрацию необходимо ввести уникальный никнейм и почту, а также ввести пароль и точно такой же ещё раз. После успешной регистрации, должна произойти передаресация на главную страницу с треками. В шапке теперь должен отображаться никнейм и кнопки выхода с загрузкой трека. (рисунок 2.3.4)

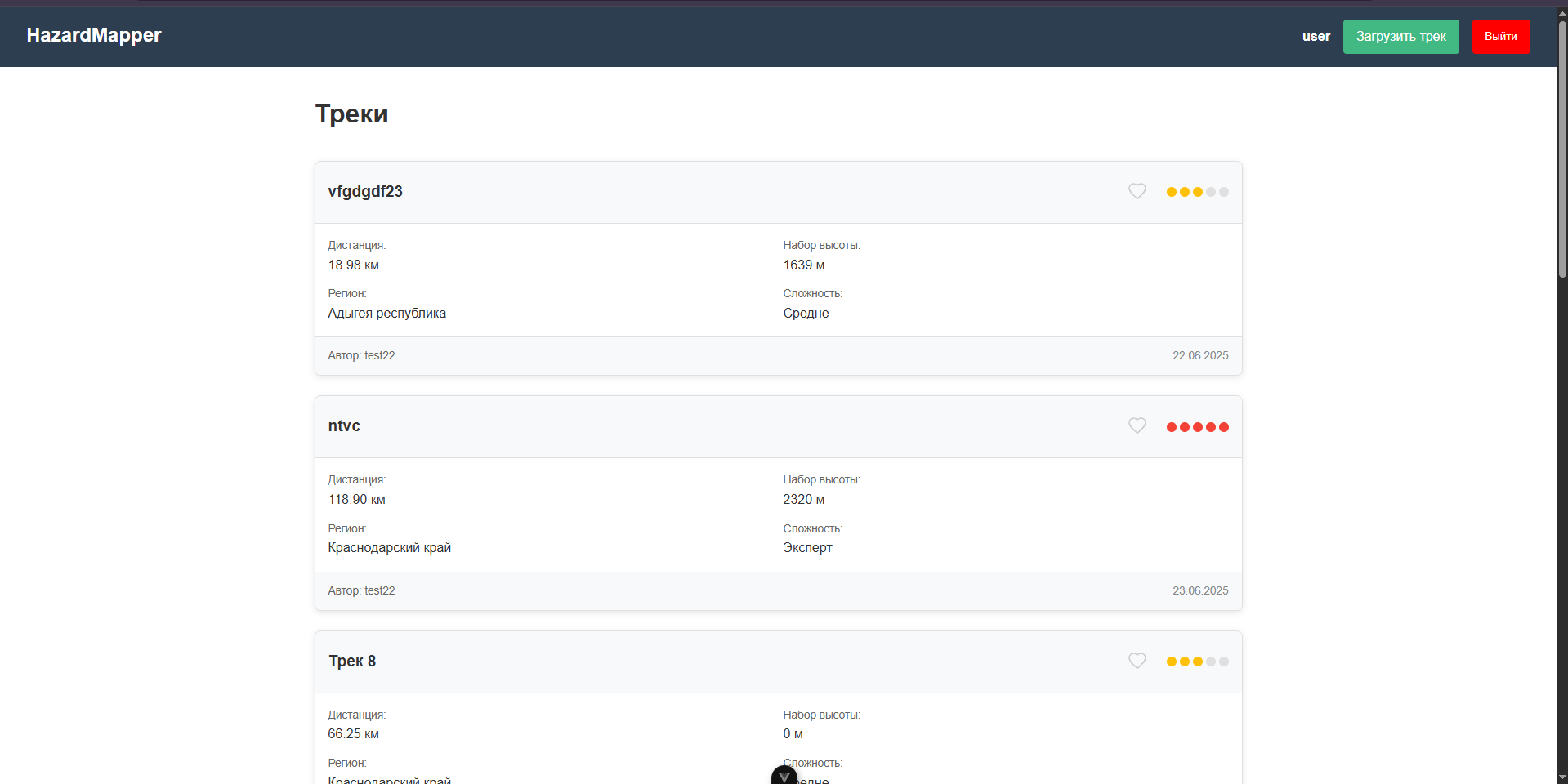


Рисунок 2.3.4 – Интерфейс сайта после успешной регистрации

На главной странице отображается список треков, который содержит следующие данные о нём: название, автор, дата загрузки, длина трека, регион, набор высоты, сложность (также есть индикатор в шапке трека). Трек можно добавить/удалить в/из избранное(-го), нажав на сердечко в шапке. (рисунок 2.3.5)

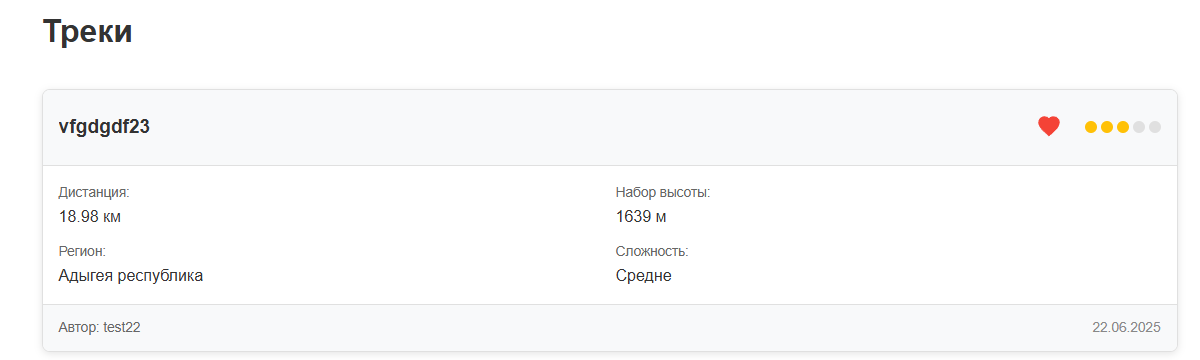


Рисунок 2.3.5 – Добавление трека в избранное

Также, если нажать на трек, произойдёт переадресация на страницу с детальной информацией о треке, где можно посмортеть на интерактивную топографическую карту с треком и оставлять комментарии под треком. (рисунок 2.3.6)

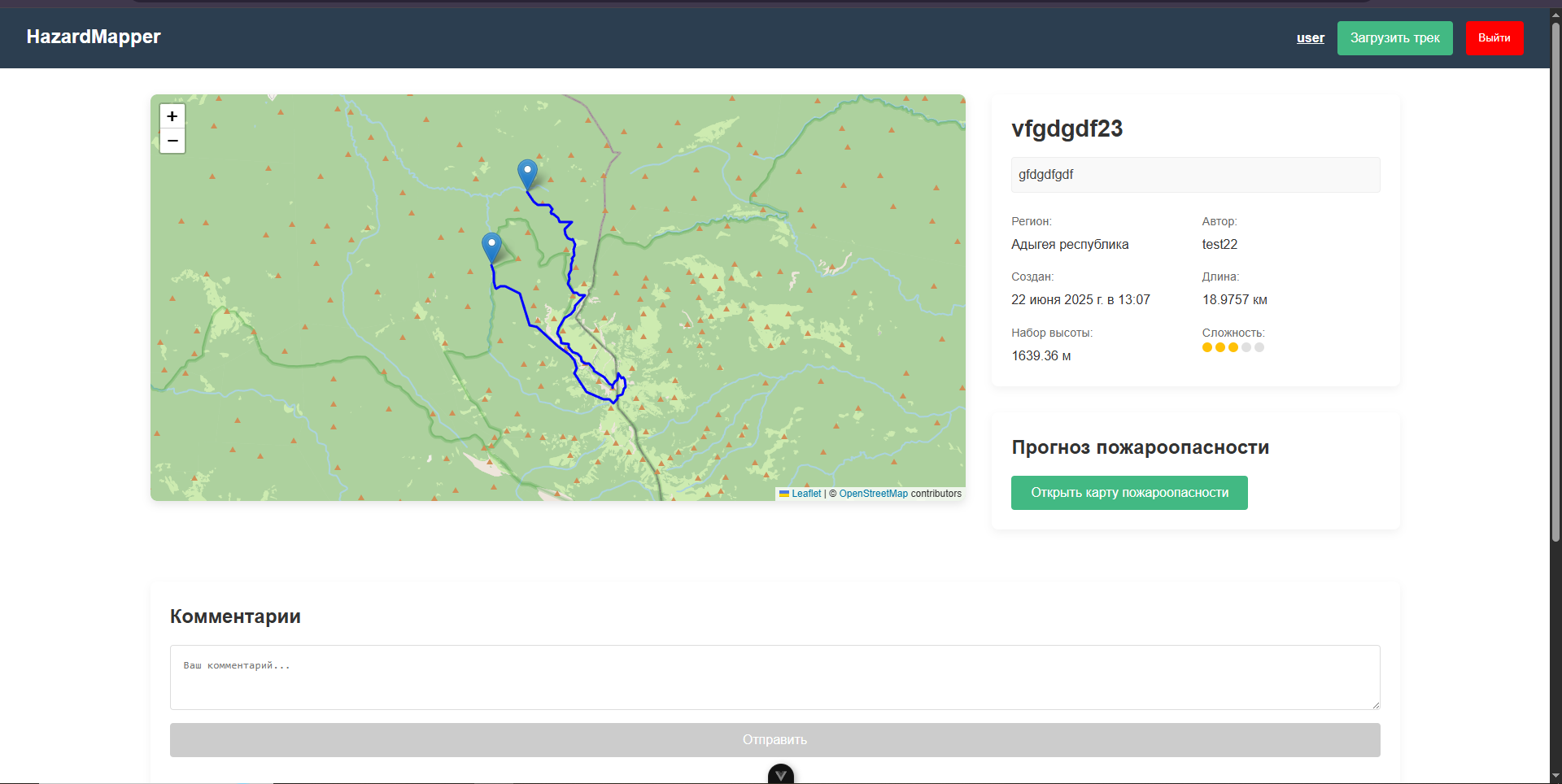


Рисунок 2.3.6 – Детальная информация о треке

Также на данной странице есть кнопка «Открыть карту пожароопасности», нажав которую через некоторое время откроется окно с картой того же трека, но с индикаторами риска пожаров в определённых местах трека. (рисунок 2.3.7)

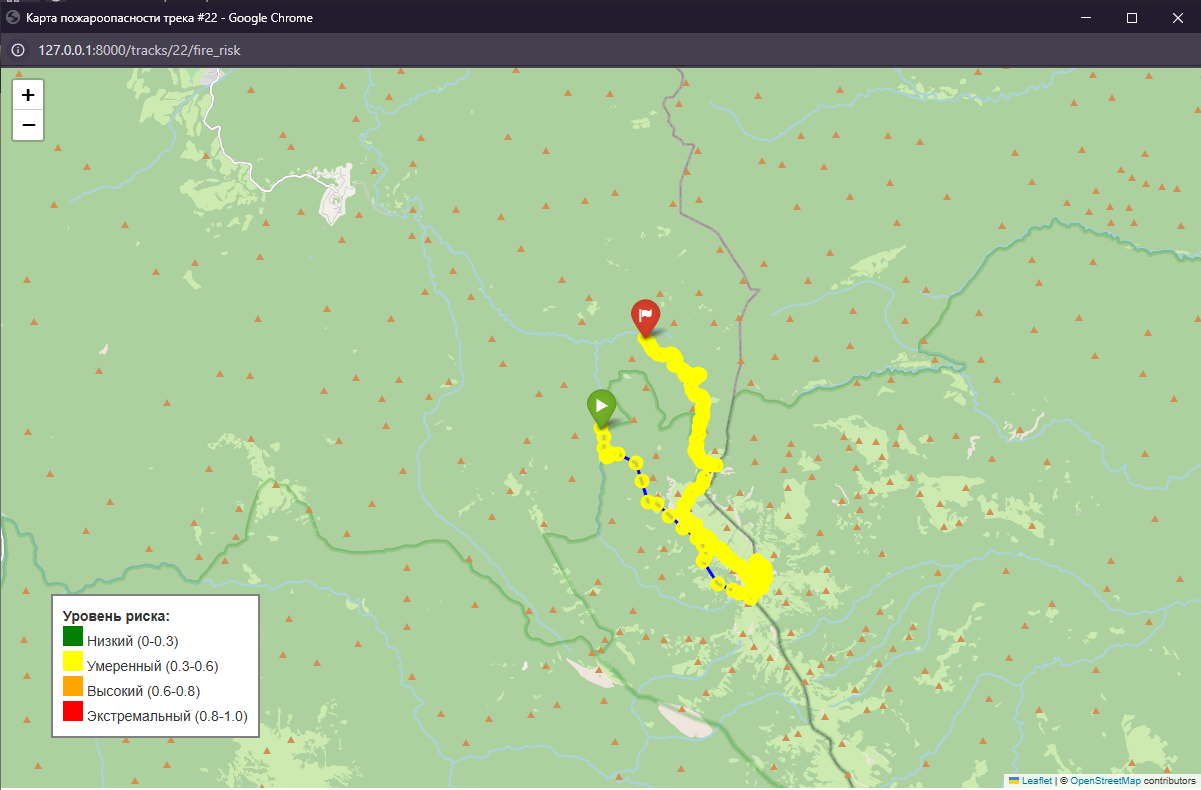


Рисунок 2.3.7 – Карта с рисками пожаров участков трека

Также на странице с детальной информации трека можно оставлять комментарии, секция которых находится внизу страницы. (рисунок 2.3.8)

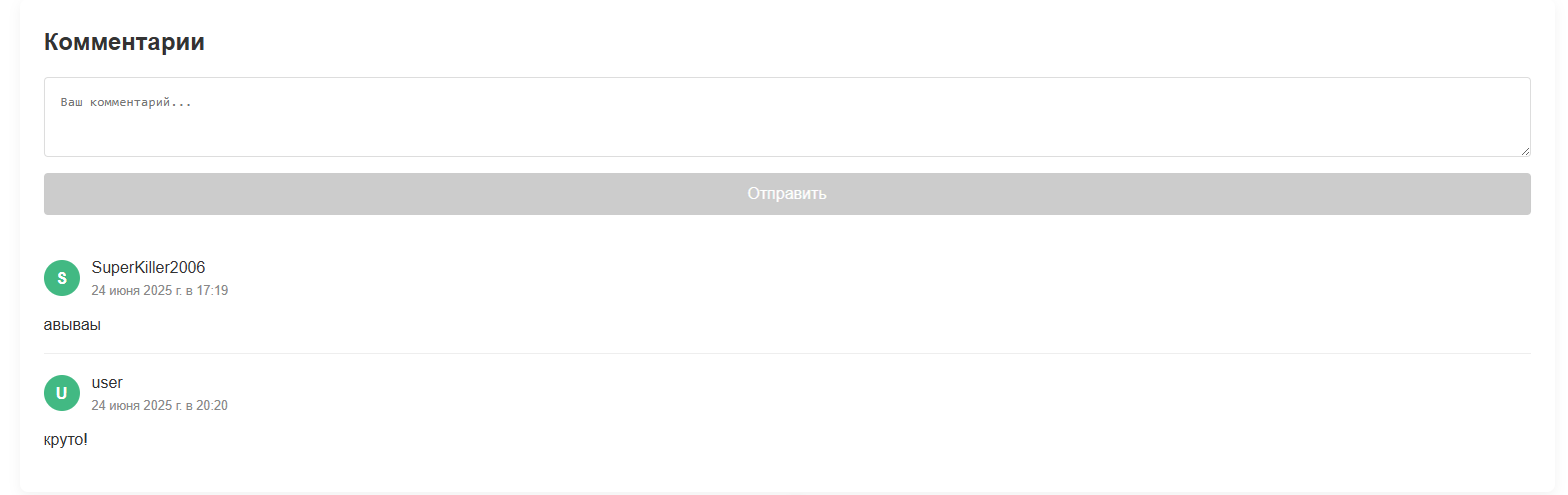


Рисунок 2.3.8 – Секция комментариев под треком

Есть также возможность загружать свои треки. Нажав на кнопку «Загрузить трек» в шапке сайта, произойдёт переадресация на форму загрузки трека. (рисунок 2.3.9)

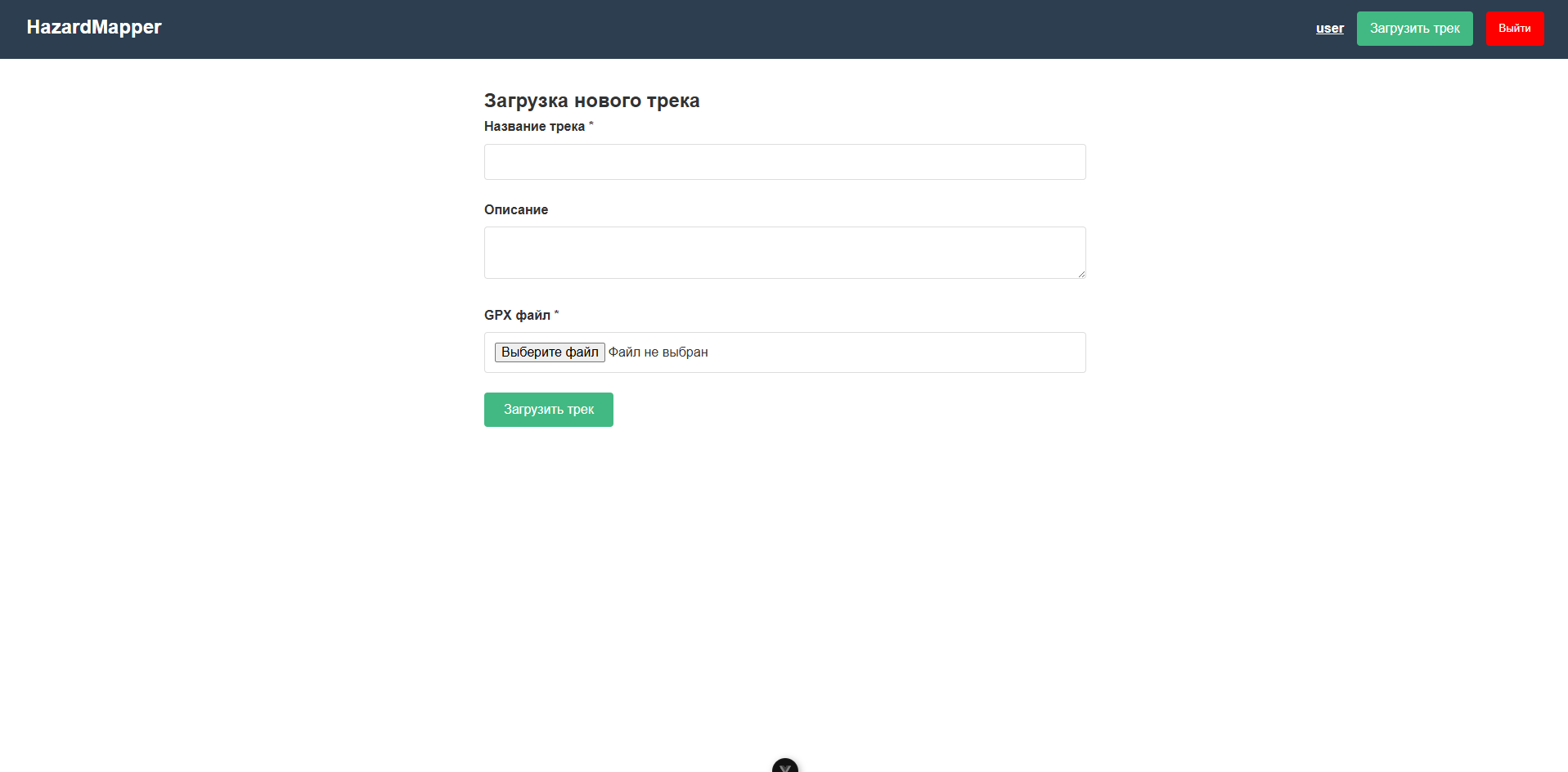


Рисунок 2.3.9 – Форма загрузки трека

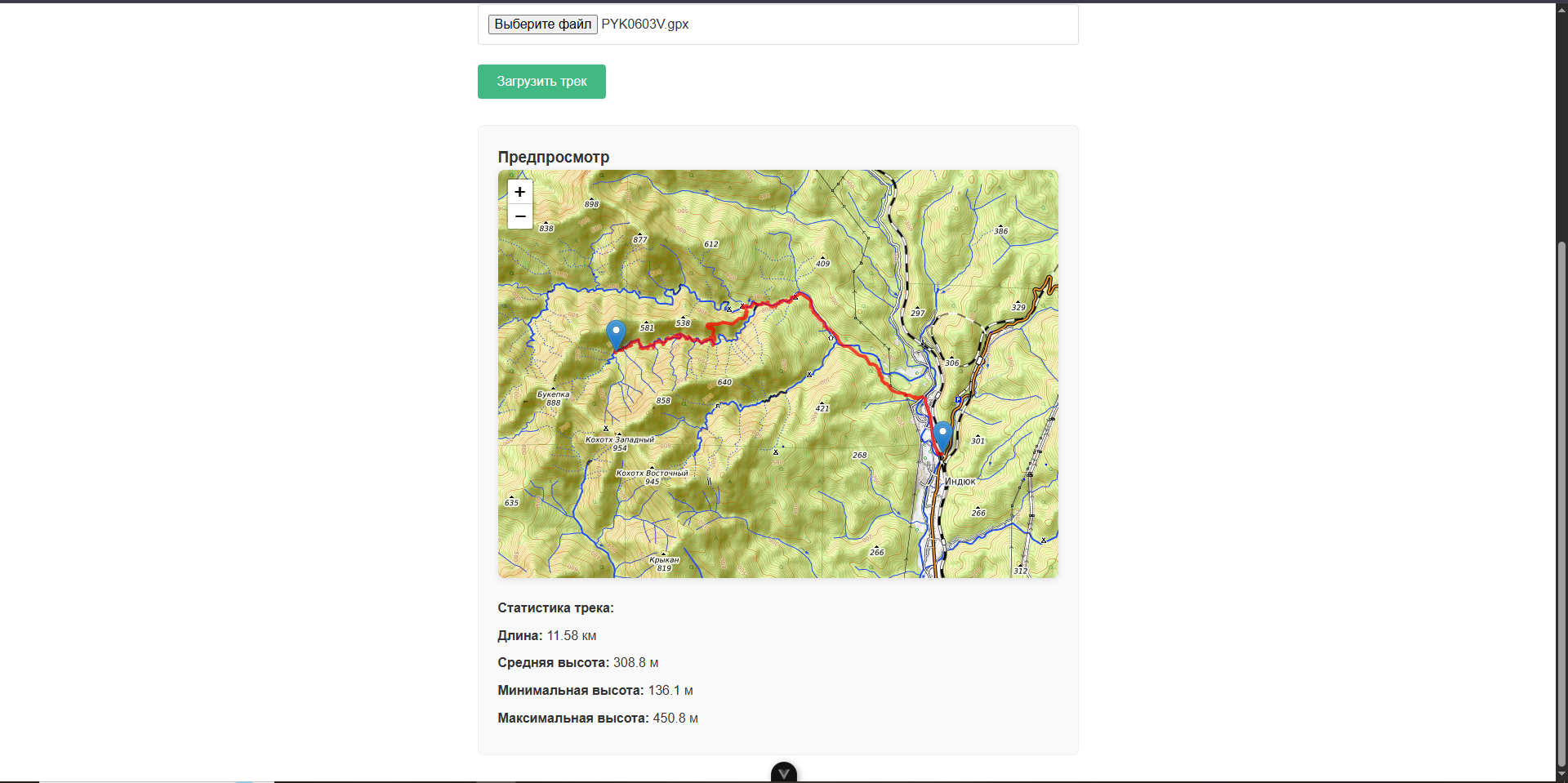
Чтобы загрузить свой трек необходимо заполнить обязательные поля: название трека, GPX файл и нажать кнопку «Загрузить трек». После нажатия, спустя время, появится предосмотр трека с некоторой информацией. (рисунок 2.3.10.) 

Рисунок 2.3.10 – Предосмотр загруженного трека

Чтобы посмотреть свои загруженные треки, необходимо перейти в личный профиль, нажав на свой никнейм в шапке сайта и перейти во вкалдку «Мои треки». (рисунок 2.3.11)

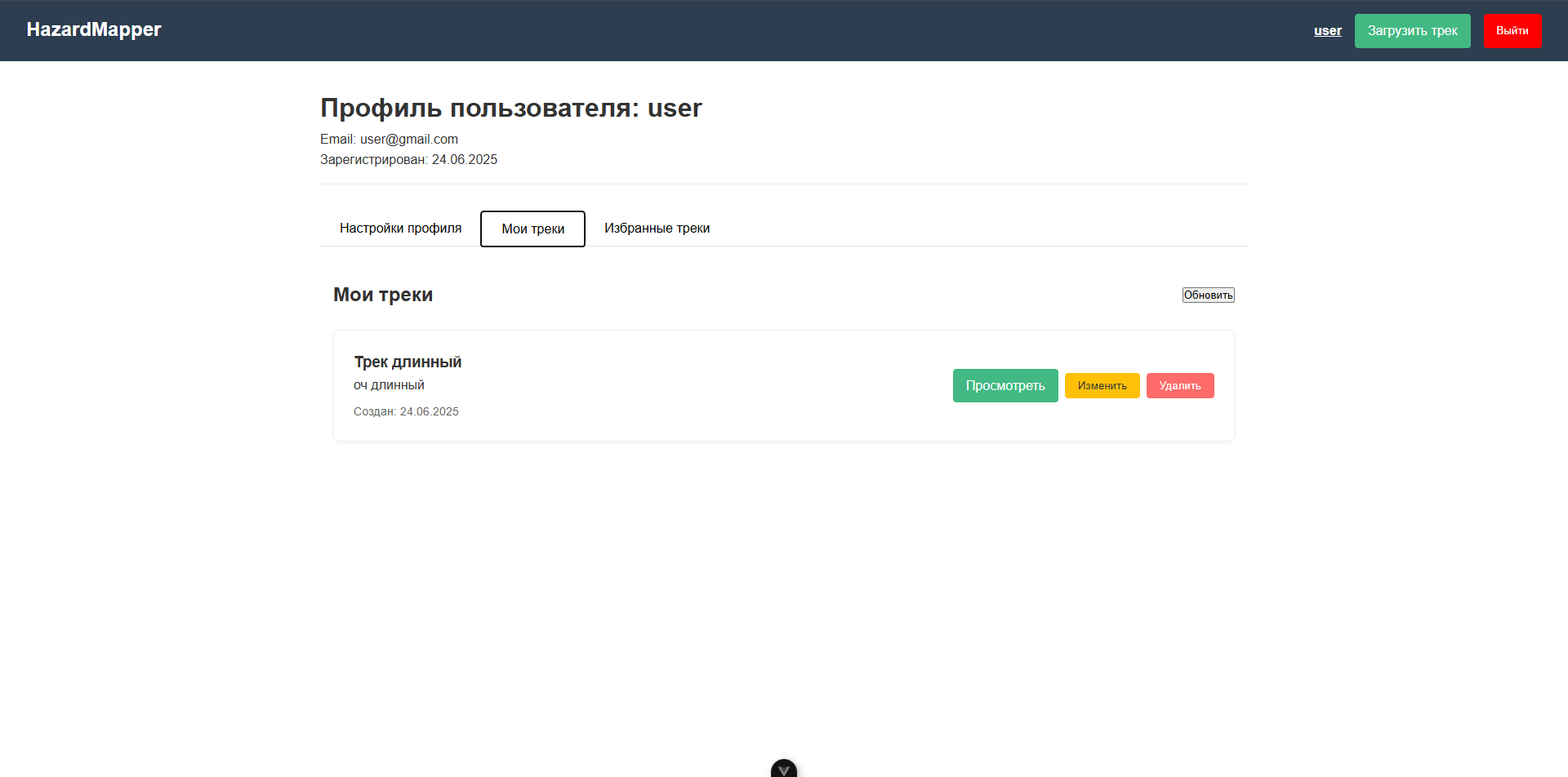


Рисунок 2.3.11 – Вкладка «Мои треки» в личном профиле

Загруженные треки можно изменять и удалять, нажав на кнопки «Изменить» и «Удалить» соответственно. При нажатии кнопки «Изменить» заместо информации о треке появится форма редактирования трека. (рисунок 2.3.12)

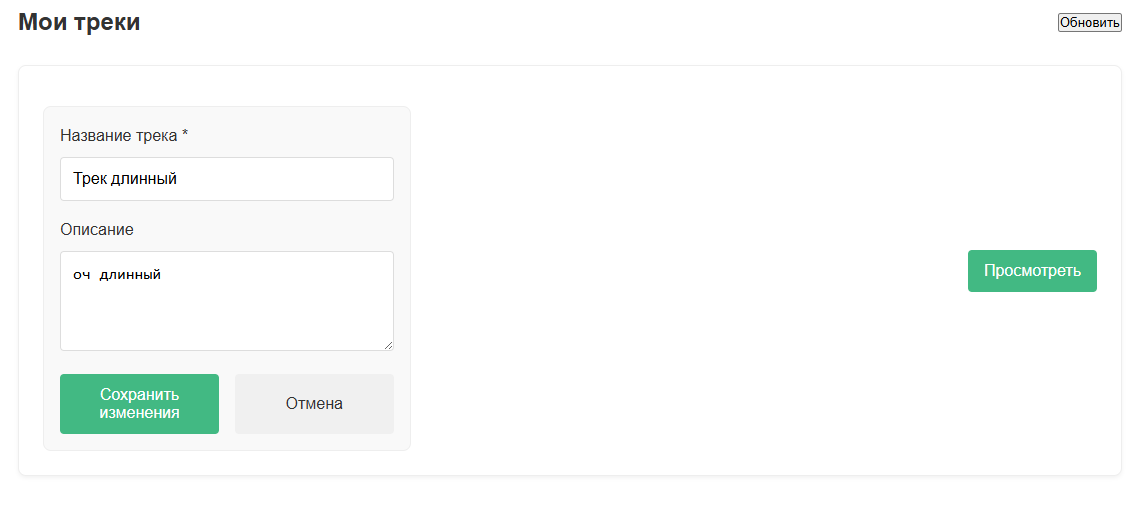


Рисунок 2.3.12 – Форма редактирования трека

Изменять можно описание и название, при этом название не должно быть пустым. (рисунок 2.3.13)

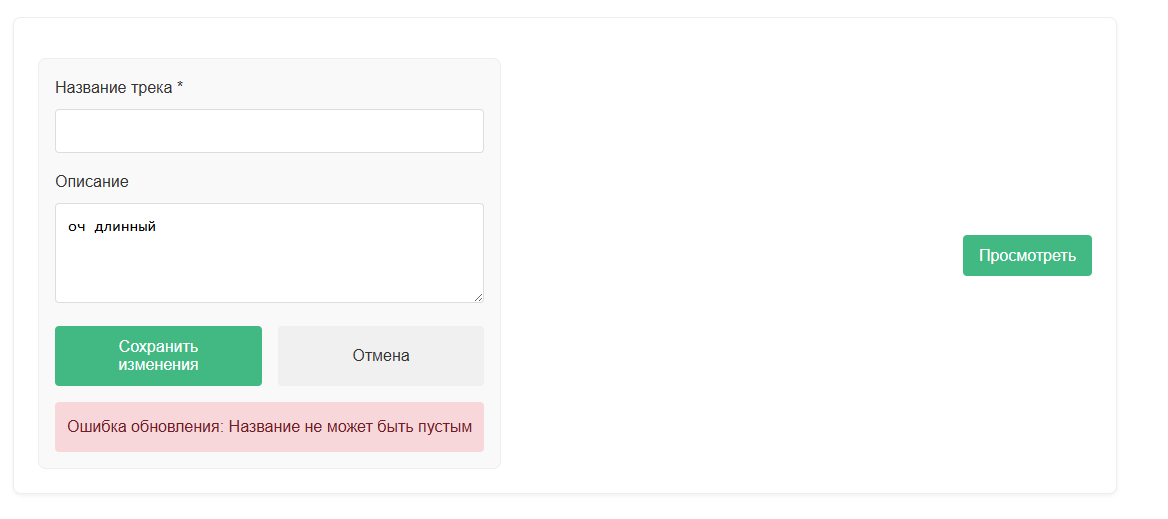


Рисунок 2.3.13 – Ошибка во время редактирования трека

В личном профиле также можно изменять свои личные данные во вкладке «Настройки профиля» по кнопке «Сохранить изменения». Ограничения изменения данных такие же, как и при регистрации. (рисунок 2.3.14)

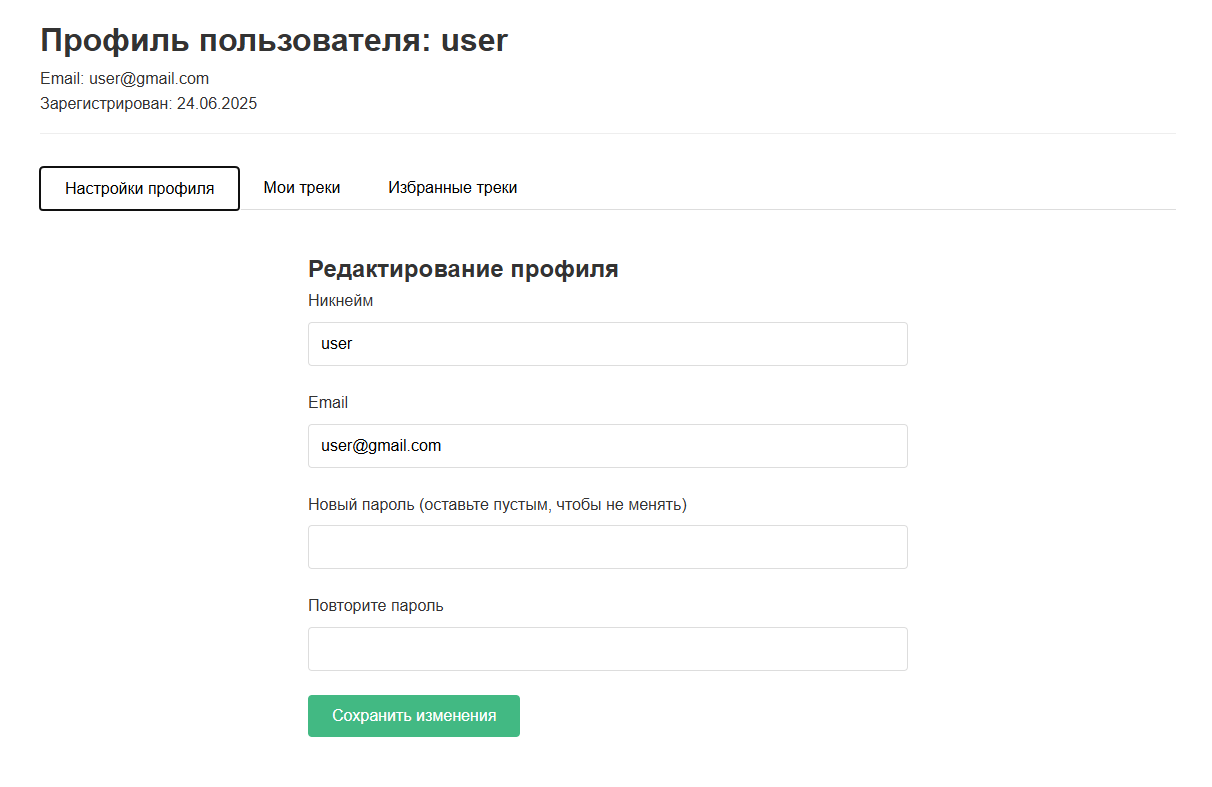


Рисунок 2.3.14 – Вкладка настройки профиля

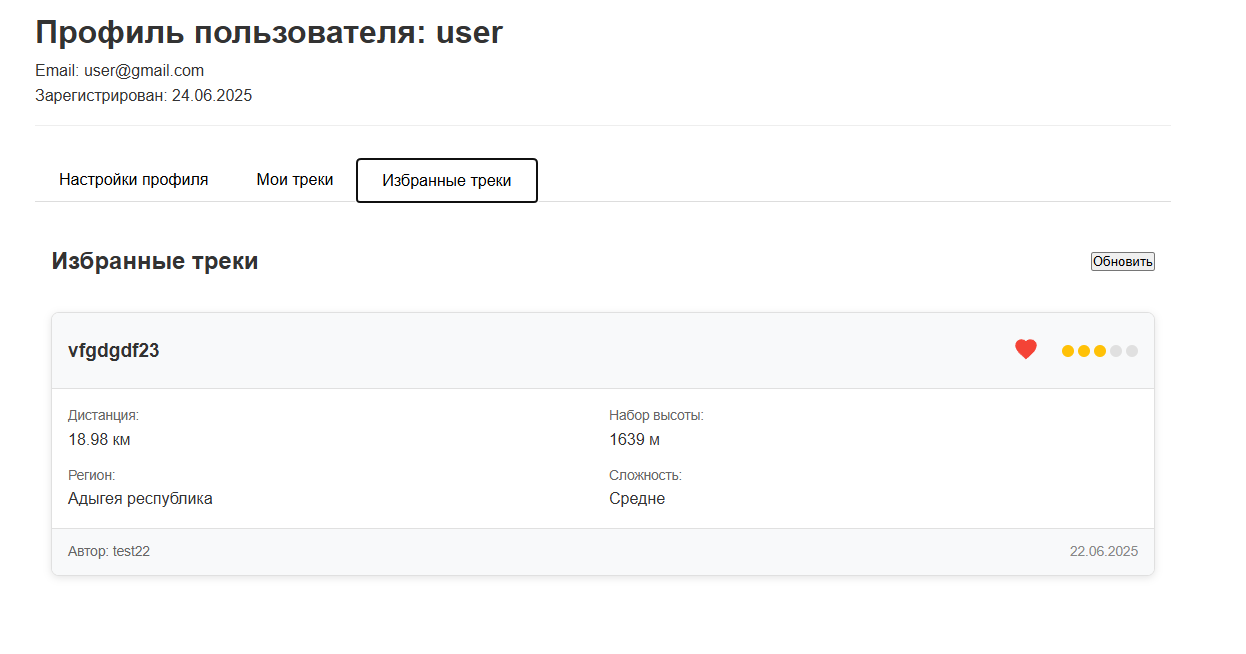
В личном профиле есть 3 последняя влкадка «Избранные треки», где будут отображаться треки, которые были добавлены в избранное. (рисунок 2.3.15)

Рисунок 2.3.15 – Вкладка избранных треков

Если пользователь помечен как админ, то у него добавляется 2 дополнительные вкладки в личном профиле: «Все пользователи» и «Все треки». (рисунок 2.3.16)

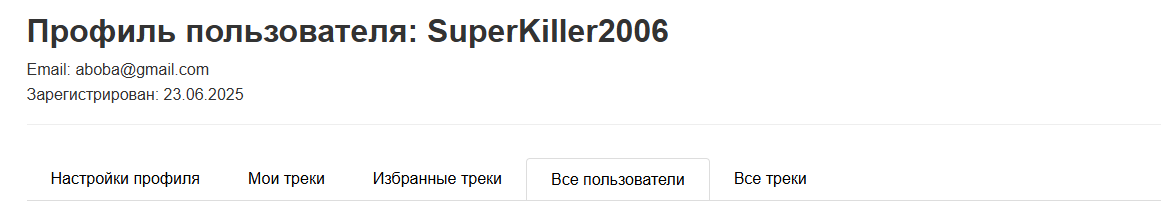


Рисунок 2.3.16 – Вкладки, доступные администратору

Во вкладке «Все пользователи» будуь отображаться все пользоватеи сайта и некоторая информация о них. Администратор может управлять статусом пользователей, изменяя его на «Активен» и «Неактивен». Неактивные пользователи больше не смогут авторизоваться на сайте. (рисунок 2.3.17)

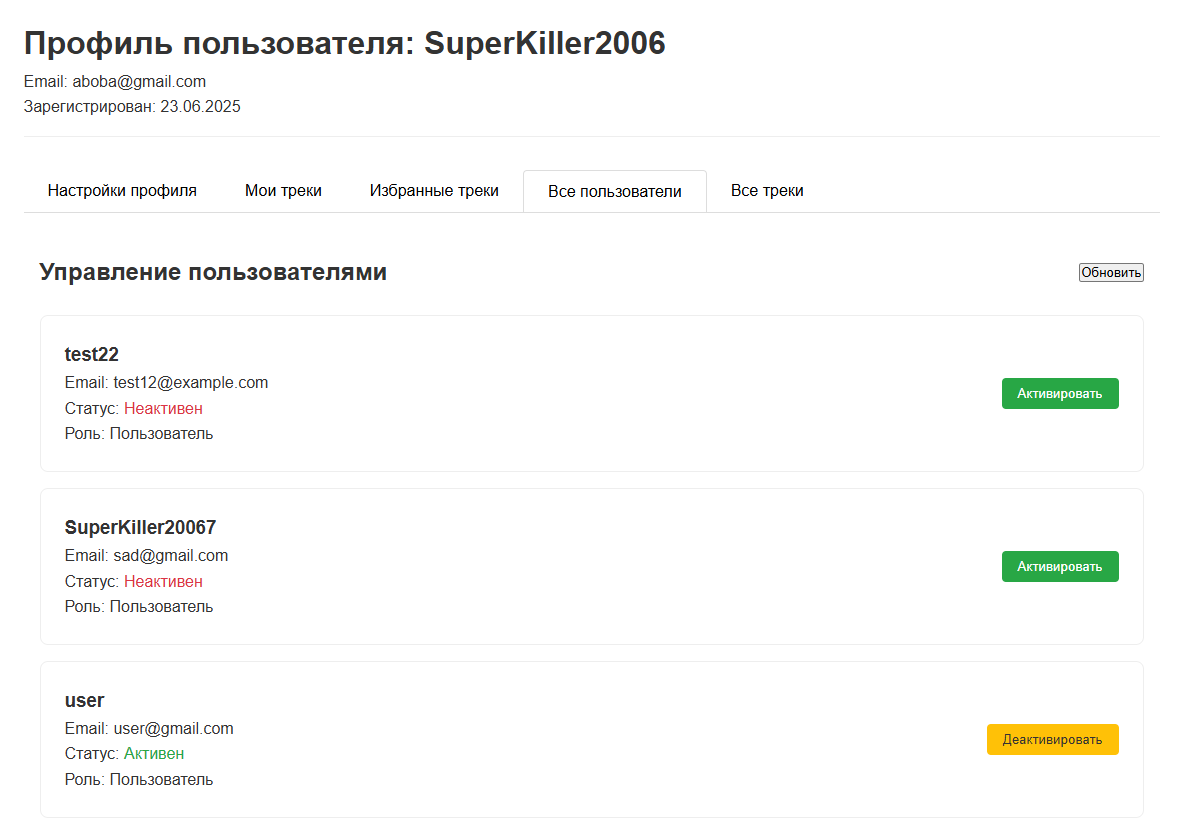


Рисунок 2.3.17 – Вкладка со всеми пользователями

Во вкладке «Все треки» будуь отображаться все треки, загруженные на. Влкадка похожа на «Мои треки» только данная вкладка предоставляет управление вссеми треками, а не только. (рисунок 2.3.18)

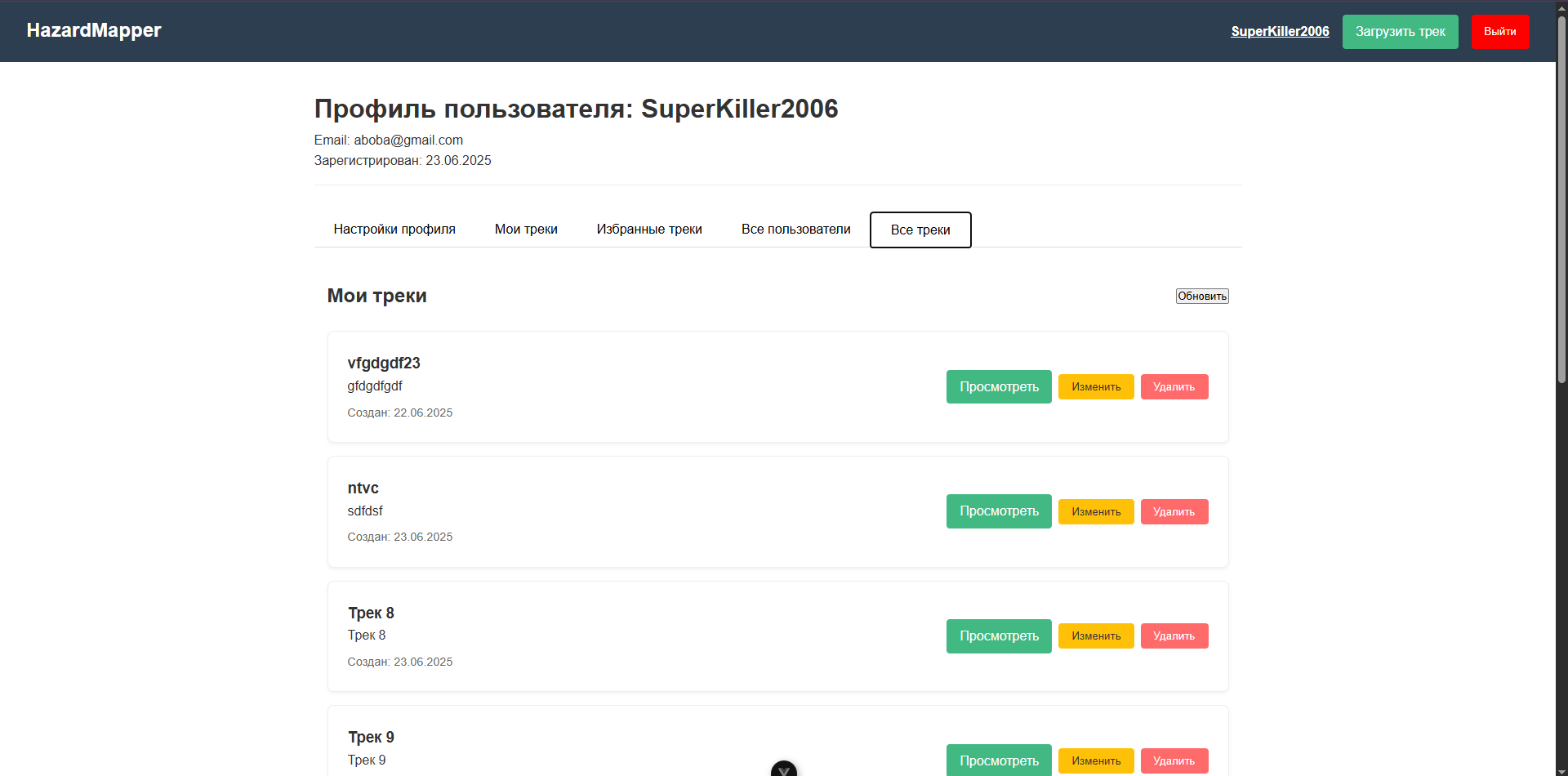


Рисунок 2.3.18 – Вкладка со всеми треками

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта были разработаны структура и алгоритм работы API и сайта, предназначенного для картографирования природных опасностей на треках.

В ходе разработки были изучены особенности реализации эндпоинтов API и компонентов Vue для построения клиентских приложений с удобным и визуально понятным пользовательским интерфейсом.

Результатом работы стало создание полнофункционального API и сайта, позволяющего пользователям проходить регистрацию и авторизацию, загружать свои треки, добавлять треки в избранное, оставлять комменатрии и просматривать карту рисков пожаров треков. Администраторам, сайт позволяет управлять всеми треками и пользователями.

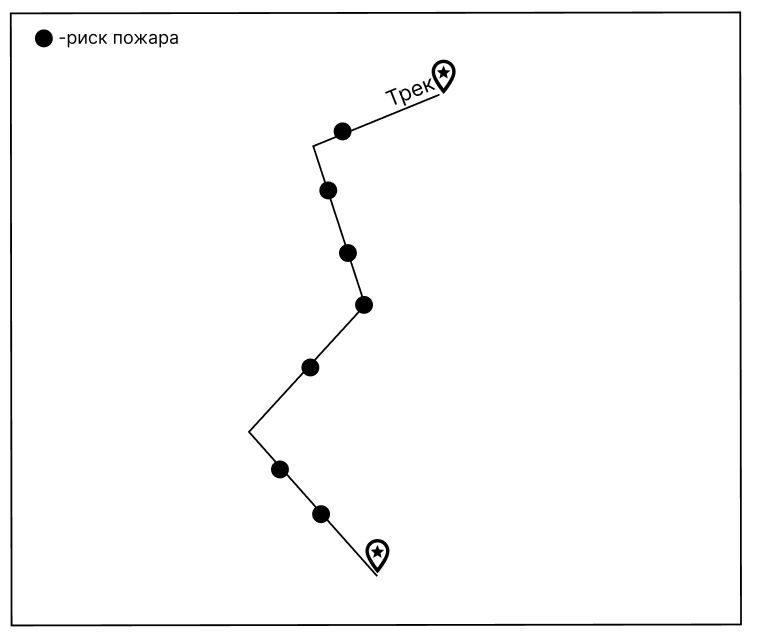
Разработка API велась на языке python в среде PyCharm 2025.1.2 с использованием фреймворка FastAPI и системы управления базами данных MySQL. Разработка сайта велась на языках HTML, CSS и JavaScript в середе Visual Studio Code с использованием фреймворка Vue.js.

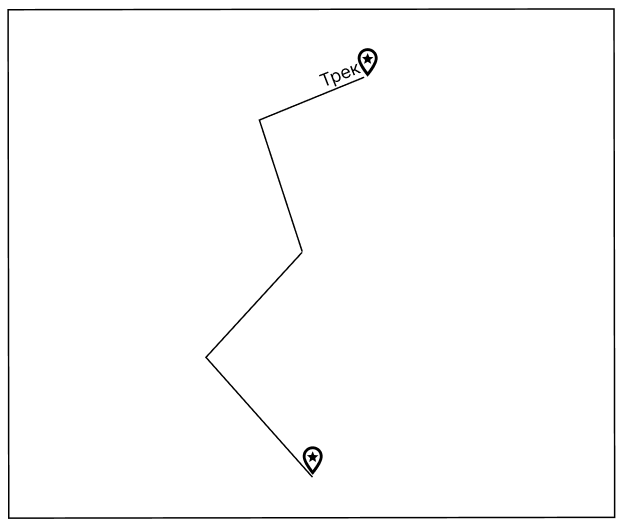
Были проведены опытная эксплуатация и отладочное тестирование разработанного API и сайта в частности. В ходе тестирования были обнаружены и устранены отдельные ошибки, в том числе добавлена валидация входных данных и улучшено форматирование выходной информации. Также было составлено руководство пользователя.

На основе данных контрольного примера приложение корректно сформировало данные о треке, карту трека и карту рисков пожаров трека, совпадающие с ожидаемыми значениями, что подтверждает правильность реализованных алгоритмов и интерфейсов.

Шаблоны выходных документов

|  |  |
| --- | --- |
| {Название трека} | |
| Дистанция:  {Длина трека} км  Регион:  {Регион трека} | Набор высоты:  {Набор высоты} м  Сложность:  {Сложность трека} |
| Автор: {Автор трека} {Дата загрузки} | |





Входные данные контрольного примера

Таблица Б.1 – Треки

|  |  |  |  |
| --- | --- | --- | --- |
| ID | ID Пользователя | Название | Описание |
| 43 | 2 | Трек небольшой | Небольшой лёгкий трек |
| 44 | 2 | Трек длиный | - |
| 29 | 1 | Трек 1 | Не придумал описание |

Таблица Б.3 – Избранное

|  |  |
| --- | --- |
| ID Пользователя | ID Трека |
| 5 | 22 |

Таблица Б.4 – Комментарии

|  |  |  |  |
| --- | --- | --- | --- |
| ID | ID Пользователя | ID Трека | Текст |
| 10 | 2 | 45 | авываы |
| 11 | 5 | 22 | круто! |
| 12 | 5 | 45 | круто! |

Таблица Б.5 – Пользователи

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Никнейм | Почта | Пароль |
| 1 | test22 | test12@example.com | test |
| 2 | SuperKiller2006 | aboba@gmail.com | aboba |
| 4 | SuperKiller20067 | sad@gmail.com | aboba |

Выходные данные контрольного примера

Таблица В.1 – Треки

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | ID Пользователя | Название | Регион | Описание | Длина (м) | Набор высоты (м) | Сложность | Дата загрузки |
| 43 | 2 | Трек небольшой | Краснодарский край | Небольшой лёгкий трек | 5487.06 | 0 | 3 | 2025-06-23 19:10:34 |
| 44 | 2 | Трек длиный | Краснодарский край | - | 5487.06 | 0 | 3 | 2025-06-23 19:54:08 |
| 29 | 1 | Трек 1 | Краснодарский край | Не придумал описание | 118899 | 1639.36 | 5 | 2025-06-23 12:47:35 |

Таблица В.2 – Точки треков

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID Записи | ID Трека | ID Точки | Широта | Долгота | Высота | Время замера |
| 4748 | 29 | 644 | 44.7061277 | 37.850866 | 37.098376928124225 | 2025-06-07 07:57:56 |
| 4749 | 29 | 645 | 44.7061082 | 37.8516631 | 43.24734543687994 | 2025-06-07 07:58:04 |
| 4750 | 29 | 646 | 44.7060883 | 37.8516958 | 43.59534729919456 | 2025-06-07 07:58:04 |
| 13634 | 44 | 0 | 44.197620485713244 | 40.615950844292236 | - | - |
| 13635 | 44 | 1 | 44.19794365571769 | 40.616133212176074 | - | - |
| 13636 | 44 | 2 | 44.1983897391551 | 40.61658383911611 | - | - |
| 13303 | 43 | 0 | 44.197620485713244 | 40.615950844292236 | - | - |
| 13304 | 43 | 1 | 44.19794365571769 | 40.616133212176074 | - | - |
| 13305 | 43 | 2 | 44.1983897391551 | 40.61658383911611 | - | - |

Таблица В.3 – Избранное

|  |  |  |
| --- | --- | --- |
| ID Пользователя | ID Трека | Дата добавления |
| 5 | 22 | 2025-06-24 20:13:18 |

Таблица В.4 - Комментарии

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | ID Пользователя | ID Трека | Текст | Дата написания |
| 10 | 2 | 45 | авываы | 2025-06-24 17:19:37 |
| 11 | 5 | 22 | круто! | 2025-06-24 20:19:34 |
| 12 | 5 | 45 | круто! | 2025-06-24 20:20:11 |

Таблица Б.5 – Пользователи

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Никнейм | Почта | Пароль | Дата регистрации | Является админом | Активен |
| 1 | test22 | test12@example.com | $2b$12$jY25lpXGIvTrXBRMnNH11eixOPNaJMKIwtm2TVb9zjTWyyFQC5yCq | 2025-06-19 01:16:06 | 0 | 0 |
| 2 | SuperKiller2006 | aboba@gmail.com | $2b$12$fgYkniSXgmDkKOFG3zMYl.RcuAPL548thzPeNOIvY5RA3s1dH6yi. | 2025-06-23 15:21:47 | 1 | 1 |
| 4 | SuperKiller20067 | sad@gmail.com | $2b$12$j.ZGpk1ZcvS8rOn9juLsie1L2lqJ4HZqcCx4xuVO/ZCt5QTJV8BiO | 2025-06-24 12:51:10 | 0 | 0 |

Код API

Файл main.py

from fastapi import FastAPI, Depends, HTTPException, status, Response, Query, UploadFile, Request

from fastapi.middleware.cors import CORSMiddleware

from fastapi.params import Cookie, File, Form

from sqlalchemy.orm import Session

from starlette.responses import JSONResponse

from app.database import get\_db, engine

from app import models, schemas, crud, security

from app.utils import gpx\_utils

from datetime import timedelta

import base64

import os

import uvicorn

from fastapi.responses import HTMLResponse

from fastapi.templating import Jinja2Templates

from app.utils.fire\_risk\_service import generate\_risk\_map

# Инициализация API

app = FastAPI()

# Настройки

SESSION\_EXPIRE\_MINUTES = 60 \* 24 \* 7 # 7 дней

# CORS (для того, чтобы можно было отправлять запросы с браузера)

app.add\_middleware(

CORSMiddleware,

allow\_origins=["http://localhost:5173"], # Разрешает запросы только с указанных origins

allow\_credentials=True, # Разрешает передачу cookies

allow\_methods=["\*"], # Разраешает все HTTP-методы (GET, POST и т.д.)

allow\_headers=["\*"], # Разраешает все заголовки

)

templates = Jinja2Templates(directory="app/templates")

@app.post("/register", response\_model=schemas.User)

async def register\_user(user: schemas.UserCreate, db: Session = Depends(get\_db)):

"""

Эндпоинт для регистрации нового юзера

:param user: данные нового пользователя

:param db: сессия с БД

:return: данные зарегистрированного пользователя

"""

# Проверяем уникальность почты

db\_user = crud.get\_user\_by\_email(db, user.email)

if db\_user:

raise HTTPException(status\_code=400, detail="Почта уже существует")

db\_user = crud.get\_user\_by\_username(db, user.username)

if db\_user:

raise HTTPException(status\_code=400, detail="Никнейм уже занят")

# Регистрируем пользователя

return crud.create\_user(db, user)

@app.post("/login")

async def login(

response: Response,

user: schemas.UserLogin,

db: Session = Depends(get\_db)

):

"""

Эндпоинт для авторизации пользователя с установкой JWT токена в куки

:param response: объект, создаваемый FastAPI для установки дополнительных параметров к ответу

:param user: авторизовывающийся пользователь

:param db: сессия с БД

:return: response с дополнительным сообщением

"""

# Проверка авторизации

db\_user = crud.authenticate\_user(db, user.email, user.password)

if not db\_user:

raise HTTPException(status\_code=401, detail="Неверные данные")

if not db\_user.is\_active:

raise HTTPException(status\_code=401, detail="Пользователь деактивирован!")

# Если авторизация удалась, то генерируем JWT токен

session\_token = security.create\_session\_token(

data={"sub": str(db\_user.id)},

expires\_delta=timedelta(minutes=SESSION\_EXPIRE\_MINUTES)

)

# Устанавливаем его в куки

response.set\_cookie(

key="session\_id",

value=session\_token,

max\_age=SESSION\_EXPIRE\_MINUTES \* 60,

httponly=True,

secure=False, # True in production

samesite="lax",

)

# Возвращаем response

return {"message": "Успешный вход"}

@app.post("/logout")

async def logout(response: Response):

"""

Эндпоинт для выхода с аккаунта пользователя

:param response: response: объект, создаваемый FastAPI для установки дополнительных параметров к ответу

:return: response с дополнительным сообщением

"""

# Отчищаем куки

response.delete\_cookie("session\_id")

# Возвращаем response

return {"message": "Успешный выход"}

@app.get("/users/me", response\_model=schemas.User)

async def get\_current\_user(

session\_id: str = Cookie(default=None),

db: Session = Depends(get\_db)

):

"""

Эндпоинт для получения текущего авторизованного пользователя

:param session\_id: название куки

:param db: сессия с БД

:return: пользователь без пароля

"""

# Если куки не установлены, то вызваем ошибку

if not session\_id:

return None

# Иначе находим и возвращаем пользователя

user\_id = security.verify\_session\_token(session\_id)

user = crud.get\_user(db, int(user\_id))

if not user:

raise HTTPException(status\_code=404, detail="Пользователь не найден")

return user

@app.put("/users/me", response\_model=schemas.User)

async def update\_current\_user(

user\_data: schemas.UserUpdate,

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user)

):

"""

Эндпоинт для обновления профиля пользователя

:param user\_data: новые данные о пользователе

:param db: сессия БД

:param current\_user: текущий пользователь

:return: обновлённый пользователь

"""

# Проверка уникальности email

if user\_data.email and user\_data.email != current\_user.email:

existing\_user = crud.get\_user\_by\_email(db, user\_data.email)

if existing\_user:

raise HTTPException(status\_code=400, detail="Пользователь с такой почтой уже существует")

# Проверка уникальности username

if user\_data.username and user\_data.username != current\_user.username:

existing\_user = crud.get\_user\_by\_username(db, user\_data.username)

if existing\_user:

raise HTTPException(status\_code=400, detail="Пользователь с таким ником ежуе существует")

# Обновляем поля

current\_user = crud.update\_user(db, user\_data, current\_user)

return current\_user

@app.put("/admin/users/{user\_id}/active", response\_model=schemas.User)

async def update\_user\_active(

user\_id: int,

active\_data: schemas.UserActiveUpdate,

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user)

):

"""

Эндпоинт для обновления статуса пользователя

:param user\_id: id обновляемого пользователя

:param active\_data: нова информация о пользователе

:param db: сессия БД

:param current\_user: текущий пользователь

:return: пользователь с новыми данными

"""

# Проверяем является ли текущий пользователь админом или владельцем аккаунта

if not current\_user.is\_admin and current\_user.id != user\_id:

raise HTTPException(status\_code=403, detail="Нет доступа")

# Обновляем данные пользователя, если такой существует

user = crud.update\_user\_active(db, user\_id, active\_data.is\_active)

if not user:

raise HTTPException(status\_code=404, detail="Пользователь не найден")

return user

@app.get("/users/{user\_id}", response\_model=schemas.User)

async def get\_user\_data(user\_id: int, db: Session = Depends(get\_db)):

"""

Эндпоинт для получения данных о пользователе

:param user\_id: id пользователя

:param db: сессия БД

:return: пользователь

"""

# Получаем данные о пользователе

user = crud.get\_user(db, user\_id)

if not user:

raise HTTPException(404, "Пользователь не найден")

# Возвращаем, если такой пользователь существует

return user

@app.get("/users/me/tracks", response\_model=schemas.TrackPaginate)

async def get\_current\_user\_tracks(

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user),

skip: int = Query(0, ge=0, description="Количество пропускаемых записей"),

limit: int = Query(10, le=100, description="Максимальное количество записей"),

):

"""

Эндпоинт для получения треков текущего пользователя с настройками

:param db: сессия БД

:param current\_user: текущий пользователь

:param skip: количество пропускаемых записей

:param limit: максимальное количество записей

:return:

"""

# Получаем треки текущего пользователя с настройками и их общее количество в БД

tracks = crud.get\_tracks\_by\_user(db, current\_user.id, skip, limit)

total\_tracks = db.query(models.Track).filter(models.Track.user\_id == current\_user.id).count()

return {

"tracks": tracks,

"total": total\_tracks,

"skip": skip,

"limit": limit

}

@app.get("/admin/users", response\_model=schemas.UserPaginate)

async def get\_all\_users(

skip: int = Query(0, ge=0, description="Количество пропускаемых записей"),

limit: int = Query(10, le=100, description="Максимальное количество записей"),

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user)

):

"""

Эндпоинт для получения пользователей с настройками

:param skip: количество пропускаемых записей

:param limit: максимальное количество записей

:param db: сессия БД

:param current\_user: текущий пользователь

:return: список пользователей

"""

# Если текущий пользователь не админ, то не пропускаем дальше

if not current\_user.is\_admin:

raise HTTPException(status\_code=403, detail="Нет доступа")

# Получаем пользователей и их общее количество в БД

users = crud.get\_all\_users(db, skip, limit, current\_user.id)

total = db.query(models.User).count()

return {

"users": users,

"total": total,

"skip": skip,

"limit": limit

}

@app.get("/tracks/load", response\_model=schemas.TrackPaginate)

async def get\_all\_tracks(

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user, use\_cache=False),

skip: int = Query(0, ge=0, description="Количество пропускаемых записей"),

limit: int = Query(10, le=100, description="Максимальное количество записей"),

):

"""

Эндпоинт для получения треков с настройками

:param db: сессия БД

:param current\_user: текущий пользователь

:param skip: количество пропускаемых записей

:param limit: максимальное количество записей

:return: список треков

"""

# Получаем треки с заданными параметрами

tracks = crud.get\_tracks(db, skip=skip, limit=limit)

# Считаем общее количество треков в БД

total\_tracks = db.query(models.Track).count()

# Добавляем флаг избранного для авторизованных пользователей

if current\_user:

for track in tracks:

track.is\_favorite = crud.is\_favorite(db, current\_user.id, track.id)

else:

# Для неавторизованных устанавливаем false

for track in tracks:

track.is\_favorite = False

return {

"tracks": tracks,

"total": total\_tracks,

"skip": skip,

"limit": limit

}

@app.post("/tracks/upload")

async def upload\_track(

title: str = Form(...),

description: str | None = Form(None),

file: UploadFile = File(...),

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user)

):

"""

Эндпоинт для загрузки нового трека

:param title: название трека

:param description: описание трека

:param file: .gpx файл трека

:param db: сессия БД

:param current\_user: текущий пользователь

:return: информация о треке

"""

# Проверка формата файла

if not file.filename.endswith('.gpx'):

raise HTTPException(400, "Неверный формат файла (Только .gpx)")

if not title:

raise HTTPException(400, "Название не может быть пустым")

# Чтение и парсинг GPX

contents = await file.read()

points, stats = gpx\_utils.parse\_gpx(contents.decode('utf-8'))

if len(points) == 0:

raise HTTPException(400, "Трек должен иметь хотя бы одну координату")

# Генерация изображения

image = bytes(1)#gpx\_utils.generate\_track\_image(points)

region = gpx\_utils.get\_track\_region(points)

# Создание объекта трека

track\_data\_for\_create = schemas.TrackCreate(

title=title,

description=description,

region=region,

total\_distance=stats["total\_distance"],

elevation\_gain=stats["elevation\_gain"],

difficulty=stats["difficulty"],

)

# Создание трека в БД

track = crud.create\_track\_with\_points(

db,

track\_data\_for\_create,

points,

image,

current\_user.id

)

return {

"track": track,

"points": points,

"stats": stats

}

@app.get("/tracks/{track\_id}", response\_model=schemas.TrackDetail)

async def get\_track\_details(track\_id: int, db: Session = Depends(get\_db)):

"""

Эндпоинт для получения детальной информации о треке

:param track\_id: id трека

:param db: сессия БД

:return:

"""

# Получаем детальную информацию

track = crud.get\_track\_with\_details(db, track\_id)

if not track:

raise HTTPException(404, "Трек не найден")

return track

@app.put("/tracks/{track\_id}", response\_model=schemas.Track)

async def update\_track(

track\_id: int,

track\_data: schemas.TrackUpdate,

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user)

):

"""

Эндпоинт для изменения данных о треке

:param track\_id: id трека

:param track\_data: новая информация о треке

:param db: сессия БД

:param current\_user: текущий пользователь

:return:

"""

# Проверяем, что трек существует и принадлежит пользователю или пользователь админ

track = crud.get\_track(db, track\_id)

if not track or (track.user\_id != current\_user.id and not current\_user.is\_admin):

raise HTTPException(status\_code=404, detail="Трек не найден или у вас нет прав")

if not track\_data.title:

raise HTTPException(400, "Название не может быть пустым")

# Обновляем данные трека

updated\_track = crud.update\_track(db, track\_id, track\_data)

if not updated\_track:

raise HTTPException(status\_code=404, detail="Трек не найден")

return updated\_track

@app.delete("/tracks/{track\_id}")

async def delete\_track(

track\_id: int,

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user)

):

"""

Эндпоинт для удаления трека

:param track\_id: id трека

:param db: сессия БД

:param current\_user: текущий пользователь

:return: сообщение о статусе

"""

# Получаем трек из БД

track = crud.get\_track(db, track\_id)

# Проверяем, что трек принадлежит пользователю или пользователь админ

if not track or (track.user\_id != current\_user.id and not current\_user.is\_admin):

raise HTTPException(status\_code=404, detail="Отказано в доступе")

# Удаляем трек и все связанные данные

crud.delete\_track(db, track\_id)

return {"message": "Трек успешно удалён"}

@app.post("/tracks/{track\_id}/favorite", response\_model=schemas.Track)

async def favorite\_track(

track\_id: int,

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user)

):

"""

Эндпоинт для добавления трека в избранное

:param track\_id: id трека

:param db: сессия БД

:param current\_user: текущий пользователь

:return: добавленный трек

"""

# Получаем трек из БД

track = crud.get\_track(db, track\_id)

if not track:

raise HTTPException(status\_code=404, detail="Трек не найден")

# Добавляем в избранное текущего пользователя, если такой трек есть

crud.add\_to\_favorites(db, current\_user.id, track\_id)

# Обновляем флаг избранного

track.is\_favorite = True

return track

@app.delete("/tracks/{track\_id}/favorite", response\_model=schemas.Track)

async def unfavorite\_track(

track\_id: int,

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user)

):

"""

Эндпоинт для удаления трека из избранного

:param track\_id: id трека

:param db: сессия БД

:param current\_user: текущий пользователь

:return: удалённый трек

"""

# Получаем трек из БД

track = crud.get\_track(db, track\_id)

if not track:

raise HTTPException(status\_code=404, detail="Трек не найден")

# Удаляем из избранного текущего пользователя, если такой трек есть

crud.remove\_from\_favorites(db, current\_user.id, track\_id)

# Обновляем флаг избранного

track.is\_favorite = False

return track

@app.post("/tracks/{track\_id}/comments", response\_model=schemas.Comment)

async def create\_comment(

track\_id: int,

comment\_data: schemas.CommentCreate,

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user)

):

"""

Эндпоинт для создания комментария к треку

:param track\_id: id трека

:param comment\_data: содержимое комментария

:param db: сессия БД

:param current\_user: текущий пользователь

:return: полная информация ок комментарии

"""

# Создаём комментарий

db\_comment = crud.create\_comment(db, comment\_data, current\_user.id, track\_id)

return db\_comment

@app.get("/favorites", response\_model=schemas.TrackPaginate)

async def get\_favorite\_tracks(

db: Session = Depends(get\_db),

current\_user: models.User = Depends(get\_current\_user),

skip: int = Query(0, ge=0, description="Количество пропускаемых записей"),

limit: int = Query(10, le=100, description="Максимальное количество записей"),

):

"""

Эндпоинт для получения избранных треков пользователя

:param db: сессия БД

:param current\_user: текущий пользователь

:param skip: количество пропускаемых записей

:param limit: максимальное количество записей

:return: избранные треки и настройки пагинации

"""

tracks = crud.get\_favorite\_tracks(db, current\_user.id, skip, limit)

total\_tracks = db.query(models.Favorite).filter(

models.Favorite.user\_id == current\_user.id

).count()

for track in tracks:

track.is\_favorite = True

return {

"tracks": tracks,

"total": total\_tracks,

"skip": skip,

"limit": limit

}

@app.get("/tracks/{track\_id}/fire\_risk", response\_class=HTMLResponse)

async def get\_fire\_risk\_map(

request: Request,

track\_id: int,

db: Session = Depends(get\_db)

):

"""

Эндпоинт для получения карты с оценкой пожароопасности для трека

:param request: запрос

:param track\_id: id трека для которого делается оценка

:param db: сессия БД

:return: HTML код карты

"""

try:

# Генерируем HTML карты

map\_html = generate\_risk\_map(track\_id, db)

# Возвращаем полноценную HTML страницу

return templates.TemplateResponse(

"fire\_risk.html",

{

"request": request,

"map\_html": map\_html,

"track\_id": track\_id

}

)

except Exception as e:

raise HTTPException(status\_code=500, detail=str(e))

if \_\_name\_\_ == "\_\_main\_\_":

uvicorn.run(app, host="0.0.0.0", port=8000)

Файл crud.py

from typing import List, Any

from sqlalchemy.orm import Session, joinedload

from app import models, schemas

from app.utils import gpx\_utils

from passlib.context import CryptContext

pwd\_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

# User CRUD-ы

def get\_user(db: Session, user\_id: int):

return db.query(models.User).filter(models.User.id == user\_id).first()

def get\_all\_users(db: Session, skip: int = 0, limit: int = 100, user\_id: int = None):

if user\_id:

return db.query(models.User).filter(models.User.id != user\_id).all()

return db.query(models.User).offset(skip).limit(limit).all()

def get\_user\_by\_email(db: Session, email: str):

return db.query(models.User).filter(models.User.email == email).first()

def get\_user\_by\_username(db: Session, username: str):

return db.query(models.User).filter(models.User.username == username).first()

def create\_user(db: Session, user: schemas.UserCreate):

hashed\_password = pwd\_context.hash(user.password)

db\_user = models.User(

email=user.email,

username=user.username,

password\_hash=hashed\_password

)

db.add(db\_user)

db.commit()

db.refresh(db\_user)

return db\_user

def update\_user(db: Session, user\_data: schemas.UserUpdate, current\_user: models.User):

if user\_data.username:

current\_user.username = user\_data.username

if user\_data.email:

current\_user.email = user\_data.email

if user\_data.password:

hashed\_password = pwd\_context.hash(user\_data.password)

current\_user.password\_hash = hashed\_password

db.commit()

db.refresh(current\_user)

return current\_user

def update\_user\_active(db: Session, user\_id: int, is\_active: bool):

user = db.query(models.User).filter(models.User.id == user\_id).first()

if user:

user.is\_active = is\_active

db.commit()

db.refresh(user)

return user

def authenticate\_user(db: Session, email: str, password: str):

user = get\_user\_by\_email(db, email)

if not user:

return False

if not pwd\_context.verify(password, user.password\_hash):

return False

return user

# Tracks CRUD-ы

def get\_track(db: Session, track\_id: int):

return db.query(models.Track).filter(models.Track.id == track\_id).first()

def update\_track(db: Session, track\_id: int, track\_data: schemas.TrackUpdate):

db\_track = db.query(models.Track).filter(models.Track.id == track\_id).first()

if not db\_track:

return None

if track\_data.title:

db\_track.title = track\_data.title

if track\_data.description is not None:

db\_track.description = track\_data.description

db.commit()

db.refresh(db\_track)

return db\_track

def get\_tracks(db: Session, skip: int = 0, limit: int = 100):

return db.query(models.Track).offset(skip).limit(limit).all()

def get\_tracks\_by\_user(db: Session, user\_id: int, skip: int = 0, limit: int = 100):

return db.query(models.Track).filter(models.Track.user\_id == user\_id).offset(skip).limit(limit).all()

def create\_track\_with\_points(

db: Session,

track\_data: schemas.TrackCreate,

points: list,

image: bytes,

user\_id: int

):

db\_track = models.Track(

title=track\_data.title,

region=track\_data.region,

description=track\_data.description,

total\_distance=track\_data.total\_distance,

elevation\_gain=track\_data.elevation\_gain,

difficulty=track\_data.difficulty,

user\_id=user\_id

)

db.add(db\_track)

db.commit()

db.refresh(db\_track)

for i, point in enumerate(points):

db\_point = models.TrackPoint(

track\_id=db\_track.id,

point\_index=i,

latitude=point['latitude'],

longitude=point['longitude'],

elevation=point.get('elevation'),

point\_time=point.get('time')

)

db.add(db\_point)

if image:

db\_image = models.TrackImage(

track\_id=db\_track.id,

image=image

)

db.add(db\_image)

db.commit()

return db\_track.id

def get\_track\_with\_details(db: Session, track\_id: int):

return db.query(models.Track).\

options(

joinedload(models.Track.points),

joinedload(models.Track.images),

joinedload(models.Track.owner),

joinedload(models.Track.comments).joinedload(models.Comment.author)

).filter(models.Track.id == track\_id).first()

def delete\_track(db: Session, track\_id: int):

track = get\_track(db, track\_id)

db.delete(track)

db.commit()

return {"message":"Успех"}

# Избранное CRUD-ы

def add\_to\_favorites(db: Session, user\_id: int, track\_id: int):

# Проверяем, не добавлен ли уже трек

existing\_fav = db.query(models.Favorite).filter(

models.Favorite.user\_id == user\_id,

models.Favorite.track\_id == track\_id

).first()

if existing\_fav:

return existing\_fav

fav = models.Favorite(user\_id=user\_id, track\_id=track\_id)

db.add(fav)

db.commit()

db.refresh(fav)

return fav

def remove\_from\_favorites(db: Session, user\_id: int, track\_id: int):

fav = db.query(models.Favorite).filter(

models.Favorite.user\_id == user\_id,

models.Favorite.track\_id == track\_id

).first()

if fav:

db.delete(fav)

db.commit()

return True

return False

def get\_favorite\_tracks(db: Session, user\_id: int, skip: int = 0, limit: int = 100):

return db.query(models.Track).join(models.Favorite).filter(

models.Favorite.user\_id == user\_id

).offset(skip).limit(limit).all()

def is\_favorite(db: Session, user\_id: int, track\_id: int):

return db.query(models.Favorite).filter(

models.Favorite.user\_id == user\_id,

models.Favorite.track\_id == track\_id

).first() is not None

# Комментарии CRUD-ы

def create\_comment(db: Session, comment\_data: schemas.CommentCreate, user\_id: int, track\_id: int):

db\_comment = models.Comment(

content=comment\_data.content,

user\_id=user\_id,

track\_id=track\_id

)

db.add(db\_comment)

db.commit()

db.refresh(db\_comment)

return db\_comment

Файл database.py

from sqlalchemy import create\_engine

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import sessionmaker

DATABASE\_URL = "mysql+mysqlconnector://root:1234@localhost/hazard\_mapper\_db"

# Создаём движок

engine = create\_engine(DATABASE\_URL, pool\_pre\_ping=True)

SessionLocal = sessionmaker(

autocommit=False, # Ручное управление транзакциями

autoflush=False, # Отключение авто синхронизацию с изменениями с БД

bind=engine # Привязываем движок

)

# Базовый класс для всех ORM-моделей (таблиц БД)

Base = declarative\_base()

# Генератор сессий БД, чтобы у каждого API-запроса была своя

def get\_db():

db = SessionLocal()

try:

yield db

finally:

db.close()

Файл models.py

from sqlalchemy import Column, Integer, String, Text, Boolean, TIMESTAMP, ForeignKey, Date, Float

from sqlalchemy.orm import relationship

from app.database import Base

class User(Base):

\_\_tablename\_\_ = "users"

id = Column(Integer, primary\_key=True, index=True, autoincrement=True)

username = Column(String(50), unique=True, nullable=False)

email = Column(String(100), unique=True, nullable=False)

password\_hash = Column(String(255), nullable=False)

created\_at = Column(TIMESTAMP, server\_default="CURRENT\_TIMESTAMP")

is\_admin = Column(Boolean, default=False)

is\_active = Column(Boolean, default=True)

tracks = relationship("Track", back\_populates="owner")

comments = relationship("Comment", back\_populates="author")

class Track(Base):

\_\_tablename\_\_ = "tracks"

id = Column(Integer, primary\_key=True, index=True, autoincrement=True)

user\_id = Column(Integer, ForeignKey("users.id"), nullable=False)

title = Column(String(255), nullable=False)

region = Column(String(255))

description = Column(Text)

total\_distance = Column(Float, default=0.0) # общая дистанция в метрах

elevation\_gain = Column(Float, default=0.0) # суммарный набор высоты

difficulty = Column(Integer, default=0) # общая сложность 1-5

created\_at = Column(TIMESTAMP, server\_default="CURRENT\_TIMESTAMP")

owner = relationship("User", back\_populates="tracks")

points = relationship("TrackPoint", back\_populates="track", cascade="all, delete-orphan", passive\_deletes=True)

comments = relationship("Comment", back\_populates="track", cascade="all, delete-orphan", passive\_deletes=True)

images = relationship("TrackImage", back\_populates="track", cascade="all, delete-orphan", passive\_deletes=True)

class TrackPoint(Base):

\_\_tablename\_\_ = "track\_points"

id = Column(Integer, primary\_key=True, index=True, autoincrement=True)

track\_id = Column(Integer, ForeignKey("tracks.id"), nullable=False)

point\_index = Column(Integer, nullable=False)

latitude = Column(Float, nullable=False)

longitude = Column(Float, nullable=False)

elevation = Column(Float)

point\_time = Column(TIMESTAMP)

track = relationship("Track", back\_populates="points")

class TrackImage(Base):

\_\_tablename\_\_ = "track\_images"

id = Column(Integer, primary\_key=True, index=True, autoincrement=True)

track\_id = Column(Integer, ForeignKey("tracks.id"), nullable=False)

image = Column(Text) # Храним base64 или путь к файлу

created\_at = Column(TIMESTAMP, server\_default="CURRENT\_TIMESTAMP")

track = relationship("Track", back\_populates="images")

class Comment(Base):

\_\_tablename\_\_ = "comments"

id = Column(Integer, primary\_key=True, index=True, autoincrement=True)

user\_id = Column(Integer, ForeignKey("users.id"), nullable=False)

track\_id = Column(Integer, ForeignKey("tracks.id"), nullable=False)

content = Column(Text, nullable=False)

created\_at = Column(TIMESTAMP, server\_default="CURRENT\_TIMESTAMP")

author = relationship("User", back\_populates="comments")

track = relationship("Track", back\_populates="comments")

class Favorite(Base):

\_\_tablename\_\_ = "favorites"

user\_id = Column(Integer, ForeignKey("users.id"), primary\_key=True)

track\_id = Column(Integer, ForeignKey("tracks.id", ondelete="CASCADE"), primary\_key=True)

created\_at = Column(TIMESTAMP, server\_default="CURRENT\_TIMESTAMP")

Файл schemas.py

from typing import List

from fastapi import UploadFile

from fastapi.params import File, Form

from pydantic import BaseModel, EmailStr

from datetime import datetime, date

from app.models import TrackPoint, TrackImage

# Схемы пользователя

class UserBase(BaseModel):

email: EmailStr

username: str

class UserCreate(UserBase):

password: str

class UserLogin(BaseModel):

email: EmailStr

password: str

class User(UserBase):

id: int

is\_active: bool

created\_at: datetime

is\_admin: bool

class Config:

orm\_mode = True

class UserUpdate(BaseModel):

username: str | None = None

email: EmailStr | None = None

password: str | None = None

class UserPaginate(BaseModel):

users: List[User]

total: int

skip: int

limit: int

class UserActiveUpdate(BaseModel):

is\_active: bool

# Схемы комментариев

class CommentBase(BaseModel):

content: str

class CommentCreate(BaseModel):

content: str

class Comment(CommentBase):

id: int

user\_id: int

track\_id: int

created\_at: datetime

class Config:

orm\_mode = True

class CommentWithAuthor(Comment):

author: User

class Config:

orm\_mode = True

# Схемы треков

class TrackBase(BaseModel):

title: str

region: str | None = None

description: str | None = None

total\_distance: float

elevation\_gain: float

difficulty: int

class TrackPointBase(BaseModel):

point\_index: int

latitude: float

longitude: float

elevation: float | None = None

point\_time: datetime | None = None

class Track(TrackBase):

id: int

user\_id: int

created\_at: datetime

is\_favorite: bool | None = None

class Config:

orm\_mode = True

class TrackCreate(TrackBase):

pass

class TrackDetail(Track):

owner: User

points: List[TrackPointBase] = []

comments: List[CommentWithAuthor] = []

class Config:

orm\_mode = True

class TrackStats(BaseModel):

total\_distance: float

avg\_elevation: float

min\_elevation: float

max\_elevation: float

class TrackUpload(BaseModel):

title: str = Form(...)

description: str = Form(None)

file: UploadFile = File(...)

class TrackUpdate(BaseModel):

title: str | None = None

description: str | None = None

class TrackPaginate(BaseModel):

tracks: List[Track]

total: int

skip: int

limit: int

class TrackPoint(TrackPointBase):

id: int

track\_id: int

class Config:

orm\_mode = True

# Схемы избранного

class FavoriteBase(BaseModel):

track\_id: int

class FavoriteCreate(FavoriteBase):

pass

class Favorite(FavoriteBase):

user\_id: int

created\_at: datetime

class Config:

orm\_mode = True

Файл security.py

from jose import jwt, JWTError

from datetime import datetime, timedelta

from fastapi import HTTPException, status, Depends, Cookie

from typing import Optional

# Настройки сессии

SESSION\_SECRET = "secretik" # Устанавливаем секрет JWT токена

SESSION\_ALGORITHM = "HS256" # Алгоритм шифрования

def create\_session\_token(data: dict, expires\_delta: timedelta = None):

"""

Создаёт JWT токен с указанными данными и временем действия

:param data: данные для записи в токен

:param expires\_delta: время действия токена (по умолчанию 15 минут)

:return: JWT токен

"""

to\_encode = data.copy()

if expires\_delta:

expire = datetime.now() + expires\_delta

else:

expire = datetime.now() + timedelta(minutes=15)

to\_encode.update({"exp": expire})

return jwt.encode(to\_encode, SESSION\_SECRET, algorithm=SESSION\_ALGORITHM)

def verify\_session\_token(session\_id: Optional[str] = Cookie(None)):

"""

Проверяет токен

:param session\_id: Куки с именем session\_id

:return: id пользователя либо ошибка

"""

# Если куки не установлены рейзим ошибку

if session\_id is None:

raise HTTPException(

status\_code=status.HTTP\_401\_UNAUTHORIZED,

detail="Не авторизован",

)

try:

# Декодируем JWT токен

payload = jwt.decode(session\_id, SESSION\_SECRET, algorithms=[SESSION\_ALGORITHM])

# Достаём от туда id пользователя

user\_id: str = payload.get("sub")

# Если не удаётся, то рейзим ошибку либо возвращаем user id

if user\_id is None:

raise HTTPException(status\_code=400, detail="Некорректный токен")

return user\_id

except JWTError:

raise HTTPException(status\_code=401, detail="Недействительный токен")

Файл gpx\_utils.py

from typing import Tuple, List

import gpxpy

import gpxpy.gpx

import numpy as np

import pandas as pd

import geopandas as gpd

import contextily as ctx

from shapely.geometry import LineString

import matplotlib.pyplot as plt

import io

import geocoder

from matplotlib.backends.backend\_agg import FigureCanvasAgg as FigureCanvas

def calculate\_difficulty(distance\_km, elevation\_gain):

"""

Рассчитывает сложность трека по шкале 1-5

:param distance\_km: общая длина трека

:param elevation\_gain: общий набор высоты на всём треке

:return: число от 1 до 5

"""

# Коэффициенты для расчета

distance\_factor = distance\_km / 20 # 20 км = сложность 1

elevation\_factor = elevation\_gain / 1000 # 1000м набора = сложность 1

# Формула сложности (можно настроить)

raw\_score = distance\_factor + elevation\_factor \* 1.5

# Ограничиваем от 1 до 5

difficulty = max(1, min(5, round(raw\_score)))

return difficulty

def parse\_gpx(gpx\_content: str) -> Tuple[List[dict], dict]:

"""

Парсит .gpx контент

:param gpx\_content: .gpx контент, получаемый из файла

:return: список точек (словарей) и характеристики трека

"""

# Парсим трек

gpx = gpxpy.parse(gpx\_content)

# Определяем списки с данными о треке

points = []

elevations = []

times = []

total\_distance = 0.0

# Проходимся по каждому треку

for track in gpx.tracks:

# По каждому сегменту

for segment in track.segments:

# Запоминаем прошлую точку

prev\_point = None

# По каждой точке

for point in segment.points:

# Записываем в список точек параметры текущей

points.append({

'latitude': point.latitude,

'longitude': point.longitude,

'elevation': point.elevation,

'time': point.time

})

# Если высота точки определена, то записываем высоту в список

if point.elevation is not None:

elevations.append(point.elevation)

# То же самое с временем

if point.time:

times.append(point.time)

# Если есть предыдущая точка, то измеряем расстояние до неё

if prev\_point:

total\_distance += point.distance\_2d(prev\_point)

prev\_point = point

# Рассчитываем набор высоты

elevation\_gain = 0

prev\_elevation = None

elevations = []

for point in points:

if point['elevation'] is not None:

elevations.append(point['elevation'])

if prev\_elevation is not None and point['elevation'] > prev\_elevation:

elevation\_gain += point['elevation'] - prev\_elevation

prev\_elevation = point['elevation']

# Рассчитываем сложность

distance\_km = total\_distance / 1000

difficulty = calculate\_difficulty(distance\_km, elevation\_gain)

# Вычисляем характеристики трека

stats = {

'total\_distance': total\_distance,

'elevation\_gain': elevation\_gain,

'difficulty': difficulty,

'avg\_elevation': np.mean(elevations) if elevations else 0,

'min\_elevation': min(elevations) if elevations else 0,

'max\_elevation': max(elevations) if elevations else 0,

'start\_time': min(times) if times else None,

'end\_time': max(times) if times else None

}

# Возвращаем список точек и характеристики трека

return points, stats

def generate\_track\_image(points: List[dict]) -> bytes:

# Параметры изображения

TARGET\_WIDTH = 2000 # Ширина в пикселях

TARGET\_DPI = 300 # Качество вывода

# Ваши координаты трека

coordinates = [(point['longitude'], point['latitude']) for point in points]

# Создаем GeoDataFrame

geometry = LineString(coordinates)

gdf = gpd.GeoDataFrame(geometry=[geometry], crs="EPSG:4326")

gdf\_web = gdf.to\_crs(epsg=3857)

# Рассчитываем границы и соотношение сторон

minx, miny, maxx, maxy = gdf\_web.total\_bounds

dx, dy = maxx - minx, maxy - miny

aspect\_ratio = dx / dy

# Вычисляем размеры в пикселях

width\_in = TARGET\_WIDTH / TARGET\_DPI

height\_in = width\_in / aspect\_ratio

# Создаем фигуру с точным контролем размеров

fig = plt.figure(figsize=(width\_in, height\_in), dpi=TARGET\_DPI)

ax = fig.add\_subplot(111)

ax.set\_axis\_off()

# Устанавливаем границы с отступом 5%

padding = 0.05

ax.set\_xlim(minx - dx \* padding, maxx + dx \* padding)

ax.set\_ylim(miny - dy \* padding, maxy + dy \* padding)

# Добавляем подложку высокого качества

ctx.add\_basemap(

ax,

source=ctx.providers.OpenTopoMap,

crs=gdf\_web.crs.to\_string(),

zoom='auto',

reset\_extent=False,

interpolation='lanczos'

)

# Рисуем трек с правильными параметрами стиля

line = gdf\_web.geometry.iloc[0]

# Для LineString

if line.geom\_type == 'LineString':

x, y = line.xy

ax.plot(

x, y,

color='red',

linewidth=4,

solid\_capstyle='round', # Скругленные концы

solid\_joinstyle='round', # Скругленные углы

antialiased=True

)

# Для MultiLineString

elif line.geom\_type == 'MultiLineString':

for segment in line:

x, y = segment.xy

ax.plot(

x, y,

color='red',

linewidth=4,

solid\_capstyle='round',

solid\_joinstyle='round',

antialiased=True

)

# Генерируем изображение без сжатия

canvas = FigureCanvas(fig)

buf = io.BytesIO()

canvas.print\_figure(

buf,

format='png',

dpi=TARGET\_DPI,

bbox\_inches='tight',

pad\_inches=0,

facecolor='white',

)

plt.close(fig)

# Получаем бинарные данные

image\_data = buf.getvalue()

buf.close()

return image\_data

def get\_track\_region(points: List[dict]) -> str | None:

"""

Определение региона с использованием ArcGIS

:param points: точки трека

:return: Название региона или None

"""

# Если точек не будет, то вернём None

if not points:

return None

# Достаём первую точку трека

first\_point = points[0]

lat, lng = first\_point['latitude'], first\_point['longitude']

# Обратное геокодирование

g = geocoder.arcgis(

(lat, lng),

method='reverse',

lang='ru',

timeout=10

)

# Если что-то пошло не так, тоже вернём None

if not g.ok:

return None

# Извлекаем регион из сырых данных

if g.raw and 'address' in g.raw:

address = g.raw['address']

# Пробуем разные ключи для региона

return (

address.get('Region')

or address.get('Subregion')

or address.get('State')

)

# Если не сработает прошлый вариант

return (

getattr(g, 'region', None)

or getattr(g, 'state', None)

)

Файл file\_risk\_service.py

import os

import glob

import pandas as pd

import numpy as np

from scipy.spatial import KDTree

import folium

import logging

from datetime import datetime

from app.database import get\_db

from app import crud

# Настройка логгирования

logging.basicConfig(level=logging.INFO)

logger = logging.getLogger(\_\_name\_\_)

# Конфигурация

FIRE\_DATA\_DIR = "app/fire\_data"

SEARCH\_RADIUS = 0.05 # Радиус поиска пожаров в градусах (~5.5 км)

def load\_fire\_data():

logger.info("Загрузка данных о пожарах...")

fire\_dfs = []

years = list(range(datetime.now().year - 1, 2014, -1))

for year in years:

for file in glob.glob(os.path.join(FIRE\_DATA\_DIR, f"\*{year}\*.csv")):

try:

df = pd.read\_csv(file, low\_memory=False)

df = df.rename(columns={'bright\_ti4': 'brightness', 'acq\_date': 'date'})

# Векторизованное преобразование типов

for col in ['latitude', 'longitude', 'brightness', 'frp']:

if col in df:

df[col] = pd.to\_numeric(df[col], errors='coerce')

# Фильтрация и преобразование даты

df = df.dropna(subset=['latitude', 'longitude', 'brightness', 'frp'])

df['date'] = pd.to\_datetime(df['date'], errors='coerce')

df = df.dropna(subset=['date'])

if not df.empty:

df['year'] = year

fire\_dfs.append(df)

logger.info(f"Загружен {file}: {len(df)} записей")

except Exception as e:

logger.error(f"Ошибка чтения {file}: {str(e)}")

if not fire\_dfs:

raise ValueError("Не удалось загрузить данные о пожарах")

return pd.concat(fire\_dfs, ignore\_index=True)

def calculate\_fire\_risk(track\_points):

"""Расчет пожароопасности для точек трека"""

fire\_data = load\_fire\_data()

fire\_coords = fire\_data[['latitude', 'longitude']].values

fire\_tree = KDTree(fire\_coords)

risks = np.zeros(len(track\_points))

for i, point in enumerate(track\_points):

point\_coord = [point.latitude, point.longitude]

indices = fire\_tree.query\_ball\_point(point\_coord, r=SEARCH\_RADIUS)

if not indices:

continue

nearby = fire\_data.iloc[indices]

distances = np.linalg.norm(nearby[['latitude', 'longitude']].values - point\_coord, axis=1)

# Расчет факторов риска

count\_factor = min(len(nearby) / 20, 1.0)

intensity\_factor = nearby['frp'].mean() / 50

brightness\_factor = (nearby['brightness'].mean() - 300) / 200

proximity\_factor = (SEARCH\_RADIUS - distances.min()) / SEARCH\_RADIUS

# Комбинированный риск

risk\_score = (

0.4 \* count\_factor +

0.3 \* intensity\_factor +

0.2 \* brightness\_factor +

0.1 \* proximity\_factor

)

risks[i] = np.clip(risk\_score, 0.0, 1.0)

return risks

def generate\_risk\_map(track\_id: int, db):

"""Генерация карты с рисками для трека"""

track = crud.get\_track\_with\_details(db, track\_id)

if not track:

raise ValueError("Трек не найден")

points = [{"latitude": p.latitude, "longitude": p.longitude} for p in track.points]

risks = calculate\_fire\_risk(track.points)

# Создание карты

map\_center = [points[0]['latitude'], points[0]['longitude']]

m = folium.Map(location=map\_center, zoom\_start=12)

# Цветовая схема

COLOR\_SCALE = [

(0.0, 'green'),

(0.3, 'yellow'),

(0.6, 'orange'),

(0.8, 'red')

]

# Добавление трека

folium.PolyLine(

[[p['latitude'], p['longitude']] for p in points],

color='blue',

weight=3

).add\_to(m)

# Добавление маркеров риска

for point, risk in zip(points, risks):

color = next(c for t, c in reversed(COLOR\_SCALE) if risk >= t)

folium.CircleMarker(

location=[point['latitude'], point['longitude']],

radius=6,

color=color,

fill=True,

fill\_opacity=0.8,

popup=f"Риск: {risk:.2f}"

).add\_to(m)

# Маркеры начала/конца

folium.Marker(

[points[0]['latitude'], points[0]['longitude']],

icon=folium.Icon(icon='play', color='green'),

popup='Начало'

).add\_to(m)

folium.Marker(

[points[-1]['latitude'], points[-1]['longitude']],

icon=folium.Icon(icon='flag', color='red'),

popup='Конец'

).add\_to(m)

# Легенда

legend\_html = '''

<div style="position:fixed; bottom:50px; left:50px; background:white; padding:10px; border:2px solid grey; z-index:1000;">

<b>Уровень риска:</b><br>

<i style="background:green; width:20px; height:20px; display:inline-block;"></i> Низкий (0-0.3)<br>

<i style="background:yellow; width:20px; height:20px; display:inline-block;"></i> Умеренный (0.3-0.6)<br>

<i style="background:orange; width:20px; height:20px; display:inline-block;"></i> Высокий (0.6-0.8)<br>

<i style="background:red; width:20px; height:20px; display:inline-block;"></i> Экстремальный (0.8-1.0)

</div>

'''

m.get\_root().html.add\_child(folium.Element(legend\_html))

# Возвращаем только HTML карты

return m.\_repr\_html\_()

Код сайта

Файл index.html

<!DOCTYPE html>

<html lang="">

<head>

<meta charset="UTF-8">

<link rel="icon" href="src/assets/98678.svg">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Hazard Mapper</title>

</head>

<body>

<div id="app"></div>

<script type="module" src="/src/main.js"></script>

</body>

</html>

Файл main.js

import { createApp } from 'vue'

import { createPinia } from 'pinia'

import App from './App.vue'

import router from './router'

import VueSweetalert2 from 'vue-sweetalert2';

import 'sweetalert2/dist/sweetalert2.min.css';

const app = createApp(App)

app.use(createPinia())

app.use(router)

app.use(VueSweetalert2);

app.mount('#app')

Файл App.vue

<template>

<router-view />

</template>

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

line-height: 1.6;

color: #333;

}

</style>

Файл auth.js

import { defineStore } from 'pinia'

import { ref } from 'vue'

import api from '@/services/api'

export const useAuthStore = defineStore('auth', () => {

const user = ref(null)

const isAuthenticated = ref(false)

const isLoading = ref(false)

const error = ref(null)

const login = async (email, password) => {

try {

isLoading.value = true

error.value = null

await api.login(email, password)

await fetchUser()

} catch (err) {

error.value = err.response.data.detail

throw err

} finally {

isLoading.value = false

}

}

const register = async (username, email, password) => {

try {

isLoading.value = true

error.value = null

await api.register(username, email, password)

await login(email, password)

} catch (err) {

error.value = err.response.data.detail

throw err

} finally {

isLoading.value = false

}

}

const logout = async () => {

try {

isLoading.value = true

await api.logout()

user.value = null

isAuthenticated.value = false

} catch (err) {

error.value = 'Ошибка выхода из системы'

console.error('Logout error:', err)

} finally {

isLoading.value = false

}

}

const fetchUser = async () => {

try {

const response = await api.getCurrentUser()

user.value = response.data

isAuthenticated.value = true

} catch (err) {

user.value = null

isAuthenticated.value = false

}

}

// Initialize on app start

fetchUser()

return {

user,

isAuthenticated,

isLoading,

error,

login,

register,

logout,

fetchUser

}

})

Файл favorites.js

import { defineStore } from 'pinia'

import { ref } from 'vue'

import api from '@/services/api'

export const useFavoritesStore = defineStore('favorites', () => {

// Состояние

const favorites = ref([]) // Массив ID избранных треков

const loading = ref(false)

const error = ref(null)

// Действия

const toggleFavorite = async (track) => {

try {

loading.value = true

error.value = null

const wasFavorite = track.is\_favorite

if (track.is\_favorite) {

await api.unfavoriteTrack(track.id)

// Удаляем из локального хранилища

const index = favorites.value.indexOf(track.id)

if (index !== -1) {

favorites.value.splice(index, 1)

}

} else {

await api.favoriteTrack(track.id)

// Добавляем в локальное хранилище

if (!favorites.value.includes(track.id)) {

favorites.value.push(track.id)

}

}

// Возвращаем обновлённый объект трека

return {

...track,

is\_favorite: !wasFavorite

}

} catch (err) {

error.value = 'Ошибка при обновлении избранного'

console.error('Error toggling favorite:', err)

return false

} finally {

loading.value = false

}

}

const loadFavorites = async () => {

try {

loading.value = true

error.value = null

const response = await api.getFavoriteTracks()

favorites.value = response.data.tracks.map(track => track.id)

} catch (err) {

error.value = 'Ошибка загрузки избранных треков'

console.error('Error loading favorites:', err)

} finally {

loading.value = false

}

}

// Геттеры

const isFavorite = (trackId) => {

return favorites.value.includes(trackId)

}

const updateTracksFavoriteStatus = (tracks) => {

if (!tracks) return []

return tracks.map(track => {

return {

...track,

is\_favorite: favorites.value.includes(track.id)

}

})

}

// Возвращаем всё, что должно быть доступно снаружи

return {

favorites,

loading,

error,

toggleFavorite,

loadFavorites,

isFavorite,

updateTracksFavoriteStatus

}

})

Файл api.js

import axios from 'axios'

const API\_URL = 'http://localhost:8000'

const api = axios.create({

baseURL: API\_URL,

withCredentials: true,

})

export default {

login(email, password) {

return api.post('/login', { email, password })

},

register(username, email, password) {

return api.post('/register', { username, email, password })

},

logout() {

return api.post('/logout')

},

getCurrentUser() {

return api.get('/users/me')

},

getUserData(user\_id) {

return api.get(`/users/${user\_id}`)

},

getAllUsers(skip = 0, limit = 10) {

return api.get('/admin/users', { params: { skip, limit } })

},

updateCurrentUser(userData) {

return api.put('/users/me', userData)

},

updateUserActive(userId, isActive) {

return api.put(`/admin/users/${userId}/active`, { is\_active: isActive })

},

getUserTracks(skip = 0, limit = 10) {

return api.get('/users/me/tracks', { params: { skip, limit } })

},

getAllTracks(skip = 0, limit = 10) {

return api.get('/tracks/load', {

params: { skip, limit }

})

},

uploadTrack(formData) {

return api.post('/tracks/upload', formData, {

headers: {

'Content-Type': 'multipart/form-data'

}

})

},

updateTrack(trackId, updateData) {

return api.put(`/tracks/${trackId}`, updateData)

},

deleteTrack(trackId) {

return api.delete(`/tracks/${trackId}`)

},

getTrackDetails(trackId) {

return api.get(`/tracks/${trackId}`)

},

favoriteTrack(trackId) {

return api.post(`/tracks/${trackId}/favorite`)

},

unfavoriteTrack(trackId) {

return api.delete(`/tracks/${trackId}/favorite`)

},

getFavoriteTracks(skip = 0, limit = 10) {

return api.get('/favorites', { skip, limit })

},

createComment(trackId, commentData) {

return api.post(`/tracks/${trackId}/comments`, commentData)

},

getFireRiskMap(trackId) {

return api.get(`/tracks/${trackId}/fire\_risk`)

}

}

Файл index.js

import { createRouter, createWebHistory } from 'vue-router'

import HomeView from '@/views/HomeView.vue'

import LoginView from '@/views/LoginView.vue'

import RegisterView from '@/views/RegisterView.vue'

import TrakUploadView from '@/views/TrakUploadView.vue'

import ProfileView from '@/views/ProfileView.vue'

import TrackDetailView from '@/views/TrackDetailView.vue'

const routes = [

{ path: '/', name: 'Home', component: HomeView },

{ path: '/login', name: 'Login', component: LoginView },

{ path: '/register', name: 'Register', component: RegisterView },

{ path: '/track\_load', name: 'TrackLoad', component: TrakUploadView },

{ path: '/profile', name: 'Profile', component: ProfileView},

{ path: '/tracks/:id', name: 'TrackDetail', component: TrackDetailView, props: true },

]

const router = createRouter({

history: createWebHistory(),

routes

})

export default router

Файл AuthHeader.vue

<template>

<header class="header">

<div class="logo">

<router-link to="/">

HazardMapper

</router-link>

</div>

<div class="auth-section">

<template v-if="authStore.isAuthenticated">

<router-link to="/profile" class="user-link">

{{ authStore.user.username }}

</router-link>

<router-link to="/track\_load" class="btn">Загрузить трек</router-link>

<button @click="handleLogoutButtonClick" class="logout-btn">

Выйти

</button>

</template>

<template v-else>

<router-link to="/login" class="btn">Войти</router-link>

<router-link to="/register" class="btn">Регистрация</router-link>

</template>

</div>

</header>

</template>

<script setup>

import { useAuthStore } from '@/stores/auth'

import { useRouter } from 'vue-router'

const authStore = useAuthStore()

const router = useRouter()

const handleLogoutButtonClick = async () => {

try {

await authStore.logout()

router.push('/')

} catch (error) {

console.error('Ошибка выхода:', error)

}

}

</script>

<style scoped>

.header {

display: flex;

justify-content: space-between;

padding: 1rem 2rem;

background-color: #2c3e50;

color: white;

}

.logo a {

color: white;

font-size: 1.5rem;

font-weight: bold;

text-decoration: none;

}

.auth-section {

display: flex;

align-items: center;

gap: 1rem;

}

.btn {

padding: 0.5rem 1rem;

background-color: #42b983;

color: white;

border-radius: 4px;

text-decoration: none;

}

.logout-btn {

padding: 0.5rem 1rem;

background: red;

border: none;

color: white;

border-radius: 4px;

cursor: pointer;

height: 100%;

}

.user-link {

color: white;

font-weight: bold;

}

</style>

Файл LoginForm.vue

<template>

<div class="login-form">

<h2>Вход</h2>

<form @submit.prevent="handleSubmit">

<div class="form-group">

<label>Почта</label>

<input type="email" v-model="email" required>

</div>

<div class="form-group">

<label>Пароль</label>

<input type="password" v-model="password" required>

</div>

<button type="submit" :disabled="authStore.isLoading">

{{ authStore.isLoading ? 'Происходит вход...' : 'Войти' }}

</button>

<div class="form-footer">

<p>Нет аккаунта? <router-link to="/register">Регистрация</router-link></p>

</div>

<div v-if="authStore.error" class="error">

{{ authStore.error }}

</div>

</form>

</div>

</template>

<script setup>

import { ref } from 'vue'

import { useAuthStore } from '@/stores/auth'

import { useRouter } from 'vue-router'

const authStore = useAuthStore()

const router = useRouter()

const email = ref('')

const password = ref('')

const handleSubmit = async () => {

try {

await authStore.login(email.value, password.value)

router.push('/')

} catch (error) {

console.error('Ошибка входа:', error)

}

}

</script>

<style scoped>

.login-form {

max-width: 400px;

margin: 0 auto;

padding: 2rem;

border: 1px solid #ddd;

border-radius: 8px;

width: 100%;

}

.form-group {

margin-bottom: 1rem;

}

label {

display: block;

margin-bottom: 0.5rem;

}

input {

width: 100%;

padding: 0.5rem;

border: 1px solid #ddd;

border-radius: 4px;

}

button {

width: 100%;

padding: 0.75rem;

background-color: #42b983;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.form-footer {

margin-top: 1rem;

text-align: center;

}

.error {

margin-top: 1rem;

padding: 0.5rem;

background-color: #ffebee;

color: #f44336;

border-radius: 4px;

}

</style>

Файл MapView.vue

<template>

<div ref="mapContainer" class="map-container"></div>

</template>

<script setup>

import { ref, onMounted, watch } from 'vue'

import L from 'leaflet'

import 'leaflet/dist/leaflet.css'

// Fix для корректного отображения маркеров

import markerIcon from 'leaflet/dist/images/marker-icon.png'

import markerShadow from 'leaflet/dist/images/marker-shadow.png'

const props = defineProps({

points: {

type: Array,

default: () => []

},

zoom: {

type: Number,

default: 13

}

})

const mapContainer = ref(null)

let map = null

let polyline = null

// Создаём кастомную иконку

const defaultIcon = L.icon({

iconUrl: markerIcon,

shadowUrl: markerShadow,

iconSize: [25, 41],

iconAnchor: [12, 41],

popupAnchor: [1, -34],

shadowSize: [41, 41]

})

onMounted(() => {

if (props.points.length > 0) {

initMap()

}

})

watch(() => props.points, (newPoints) => {

if (newPoints.length > 0) {

if (!map) {

initMap()

} else {

updateMap()

}

}

})

const initMap = () => {

// Центрируем карту на первой точке

const firstPoint = props.points[0]

map = L.map(mapContainer.value).setView(

[firstPoint.latitude, firstPoint.longitude],

props.zoom

)

// Добавляем слой OpenStreetMap

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

}).addTo(map)

// Создаём маршрут

const latLngs = props.points.map(point =>

[point.latitude, point.longitude]

)

polyline = L.polyline(latLngs, {color: 'blue'}).addTo(map)

// Добавляем маркеры

L.marker([firstPoint.latitude, firstPoint.longitude], {

icon: defaultIcon

}).addTo(map).bindPopup('Начало маршрута')

if (props.points.length > 1) {

const lastPoint = props.points[props.points.length - 1]

L.marker([lastPoint.latitude, lastPoint.longitude], {

icon: defaultIcon

}).addTo(map).bindPopup('Конец маршрута')

}

// Подгоняем границы карты под маршрут

map.fitBounds(polyline.getBounds())

}

const updateMap = () => {

if (polyline) {

map.removeLayer(polyline)

}

const latLngs = props.points.map(point =>

[point.latitude, point.longitude]

)

polyline = L.polyline(latLngs, {color: 'blue'}).addTo(map)

map.fitBounds(polyline.getBounds())

}

</script>

<style scoped>

.map-container {

height: 100%;

width: 100%;

min-height: 400px;

border-radius: 8px;

overflow: hidden;

z-index: 0;

}

</style>

Файл RegisterForm.vue

<template>

<div class="register-form">

<h2>Регистрация</h2>

<form @submit.prevent="handleSubmit">

<div class="form-group">

<label>Никнейм</label>

<input type="text" v-model="username" required>

</div>

<div class="form-group">

<label>Почта</label>

<input type="email" v-model="email" required minlength="8">

</div>

<div class="form-group">

<label>Пароль</label>

<input type="password" v-model="password" required minlength="8">

</div>

<div class="form-group">

<label>Повторите пароль</label>

<input type="password" v-model="confirmPassword" required>

</div>

<button type="submit" :disabled="authStore.isLoading">

{{ authStore.isLoading ? 'Происходит регистрация...' : 'Зарегистироваться' }}

</button>

<div class="form-footer">

<p>Уже есть аккаунт? <router-link to="/login">Войти</router-link></p>

</div>

<div v-if="passwordMismatch" class="error">

Пароли не совпадают

</div>

<div v-if="authStore.error" class="error">

{{ authStore.error }}

</div>

</form>

</div>

</template>

<script setup>

import { ref, computed } from 'vue'

import { useAuthStore } from '@/stores/auth'

import { useRouter } from 'vue-router'

const authStore = useAuthStore()

const router = useRouter()

const username = ref('')

const email = ref('')

const password = ref('')

const confirmPassword = ref('')

const passwordMismatch = computed(() => {

return password.value && password.value !== confirmPassword.value

})

const handleSubmit = async () => {

if (passwordMismatch.value) return

try {

await authStore.register(username.value, email.value, password.value)

router.push('/')

} catch (error) {

console.error('Registration failed:', error)

}

}

</script>

<style scoped>

.register-form {

max-width: 400px;

margin: 0 auto;

padding: 2rem;

border: 1px solid #ddd;

border-radius: 8px;

width: 100%;

}

.register-form {

max-width: 400px;

margin: 0 auto;

padding: 2rem;

border: 1px solid #ddd;

border-radius: 8px;

}

.form-group {

margin-bottom: 1rem;

}

label {

display: block;

margin-bottom: 0.5rem;

}

input {

width: 100%;

padding: 0.5rem;

border: 1px solid #ddd;

border-radius: 4px;

}

button {

width: 100%;

padding: 0.75rem;

background-color: #42b983;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.form-footer {

margin-top: 1rem;

text-align: center;

}

.error {

margin-top: 1rem;

padding: 0.5rem;

background-color: #ffebee;

color: #f44336;

border-radius: 4px;

}

</style>

Файл TrackCard.vue

<template>

<div class="track-card" :class="`difficulty-${track.difficulty}`">

<div class="card-header">

<h3>{{ track.title }}</h3>

<div class="other-header-info">

<button

v-if="authStore.isAuthenticated"

@click.stop="toggleFavorite"

class="favorite-btn"

:class="{ active: track.is\_favorite }"

:disabled="favoritesStore.loading"

>

<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="none" stroke="currentColor">

<path d="M12 21.35l-1.45-1.32C5.4 15.36 2 12.28 2 8.5 2 5.42 4.42 3 7.5 3c1.74 0 3.41.81 4.5 2.09C13.09 3.81 14.76 3 16.5 3 19.58 3 22 5.42 22 8.5c0 3.78-3.4 6.86-8.55 11.54L12 21.35z"/>

</svg>

</button>

<div class="difficulty-indicator">

<span v-for="i in 5" :key="i" :class="{ active: i <= track.difficulty }"></span>

</div>

</div>

</div>

<router-link :to="{ name: 'TrackDetail', params: { id: track.id } }" class="track-link">

<div class="card-content">

<div class="track-meta">

<div class="stat-item">

<span class="stat-label">Дистанция:</span>

<span class="stat-value">{{ (track.total\_distance / 1000).toFixed(2) }} км</span>

</div>

<div class="stat-item">

<span class="stat-label">Набор высоты:</span>

<span class="stat-value">{{ track.elevation\_gain.toFixed(0) }} м</span>

</div>

<div class="stat-item">

<span class="stat-label">Регион:</span>

<span class="stat-value">{{ track.region || 'Не указан' }}</span>

</div>

<div class="stat-item">

<span class="stat-label">Сложность:</span>

<span class="stat-value">{{ difficultyText }}</span>

</div>

</div>

</div>

<div class="card-footer">

<span class="author">Автор: {{ user.username }}</span>

<span class="date">{{ formatDate(track.created\_at) }}</span>

</div>

</router-link>

</div>

</template>

<script setup>

import { computed, ref, onMounted } from 'vue'

import api from '@/services/api'

import { useAuthStore } from '@/stores/auth'

import { useFavoritesStore } from '@/stores/favorites'

const authStore = useAuthStore()

const favoritesStore = useFavoritesStore()

const user = ref({})

const emit = defineEmits(['update-track'])

const props = defineProps({

track: {

type: Object,

required: true

}

})

const localTrack = ref({...props.track})

onMounted(async () => {

await fetchUser()

})

const fetchUser = async () =>{

try {

const response = await api.getUserData(props.track.user\_id)

console.log(response.data)

user.value = response.data

} catch (err) {

user.value = 'Неизвестен'

throw err

}

}

const difficultyTexts = {

1: 'Очень легко',

2: 'Легко',

3: 'Средне',

4: 'Сложно',

5: 'Эксперт'

}

const difficultyText = computed(() => {

return difficultyTexts[props.track.difficulty] || 'Не указана'

})

const formatDate = (dateString) => {

return new Date(dateString).toLocaleDateString()

}

const toggleFavorite = async () => {

const updatedTrack = await favoritesStore.toggleFavorite(props.track)

localTrack.value = updatedTrack

emit('update-track', updatedTrack)

}

</script>

<style scoped>

.track-card {

border: 1px solid #e0e0e0;

border-radius: 8px;

overflow: hidden;

transition: all 0.3s ease;

background: white;

box-shadow: 0 2px 8px rgba(0,0,0,0.1);

}

.track-card:hover {

transform: translateY(-5px);

box-shadow: 0 6px 16px rgba(0,0,0,0.15);

}

.track-link {

text-decoration: none;

}

.card-header {

display: flex;

justify-content: space-between;

align-items: center;

padding: 1rem;

background-color: #f8f9fa;

border-bottom: 1px solid #e0e0e0;

}

.other-header-info {

display: flex;

align-items: center;

gap: 1rem;

}

.favorite-btn {

background: none;

border: none;

cursor: pointer;

padding: 0.5rem;

color: #ccc;

transition: all 0.3s ease;

}

.favorite-btn:hover {

color: #f44336;

}

.favorite-btn.active {

color: #f44336;

fill: #f44336;

}

.favorite-btn svg {

width: 24px;

height: 24px;

stroke-width: 1.5;

}

.favorite-btn.active svg {

fill: currentColor;

}

.card-header h3 {

margin: 0;

font-size: 1.2rem;

color: #333;

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

max-width: 250px;

}

.difficulty-indicator {

display: flex;

gap: 4px;

}

.difficulty-indicator span {

display: inline-block;

width: 12px;

height: 12px;

border-radius: 50%;

background-color: #e0e0e0;

}

.difficulty-indicator span.active {

background-color: #42b983;

}

.track-card.difficulty-1 .difficulty-indicator span.active { background-color: #4caf50; }

.track-card.difficulty-2 .difficulty-indicator span.active { background-color: #8bc34a; }

.track-card.difficulty-3 .difficulty-indicator span.active { background-color: #ffc107; }

.track-card.difficulty-4 .difficulty-indicator span.active { background-color: #ff9800; }

.track-card.difficulty-5 .difficulty-indicator span.active { background-color: #f44336; }

.card-content {

padding: 1rem;

display: flex;

gap: 1rem;

}

.track-meta {

flex: 2;

display: grid;

grid-template-columns: 1fr 1fr;

gap: 0.75rem;

}

.stat-item {

display: flex;

flex-direction: column;

}

.stat-label {

font-size: 0.85rem;

color: #666;

}

.stat-value {

font-weight: 500;

color: #333;

}

.card-footer {

display: flex;

justify-content: space-between;

padding: 0.75rem 1rem;

background-color: #f8f9fa;

border-top: 1px solid #e0e0e0;

font-size: 0.85rem;

}

.author {

color: #666;

}

.date {

color: #888;

}

</style>

Файл TrackUpload.vue

<template>

<div class="track-upload">

<h2>Загрузка нового трека</h2>

<form @submit.prevent="handleSubmit">

<div class="form-group">

<label>Название трека \*</label>

<input v-model="title" type="text" maxlength="80" required>

</div>

<div class="form-group">

<label>Описание</label>

<textarea v-model="description"></textarea>

</div>

<div class="form-group">

<label>GPX файл \*</label>

<input

type="file"

ref="fileInput"

accept=".gpx"

required

@change="handleFileChange"

>

</div>

<button type="submit" :disabled="isLoading">

{{ isLoading ? 'Загрузка...' : 'Загрузить трек' }}

</button>

<div v-if="error" class="error">

{{ error }}

</div>

</form>

<div v-if="previewData" class="preview">

<h3>Предпросмотр</h3>

<!-- Контейнер для карты -->

<div ref="mapContainer" class="preview-map"></div>

<div class="track-stats">

<h4>Статистика трека:</h4>

<p><strong>Длина:</strong> {{ (previewData.stats.total\_distance / 1000).toFixed(2) }} км</p>

<p><strong>Средняя высота:</strong> {{ previewData.stats.avg\_elevation.toFixed(1) }} м</p>

<p><strong>Минимальная высота:</strong> {{ previewData.stats.min\_elevation.toFixed(1) }} м</p>

<p><strong>Максимальная высота:</strong> {{ previewData.stats.max\_elevation.toFixed(1) }} м</p>

</div>

</div>

</div>

</template>

<script setup>

import { ref, onBeforeUnmount, nextTick } from 'vue'

import api from '@/services/api'

// Импортируем Leaflet напрямую

import L from 'leaflet'

import 'leaflet/dist/leaflet.css'

const title = ref('')

const description = ref('')

const fileInput = ref(null)

const previewData = ref(null)

const isLoading = ref(false)

const error = ref(null)

// Рефы для карты

const mapContainer = ref(null)

let map = null

let polyline = null

let startMarker = null

let endMarker = null

// Обработчик файла

const handleFileChange = () => {

if (!fileInput.value.files.length) return

}

// Основная функция загрузки

const handleSubmit = async () => {

try {

isLoading.value = true

error.value = null

const formData = new FormData()

formData.append('title', title.value.trim())

formData.append('description', description.value.trim())

formData.append('file', fileInput.value.files[0])

const response\_upload = await api.uploadTrack(formData)

previewData.value = {

stats: response\_upload.data.stats,

points: response\_upload.data.points

}

// Инициализируем карту после получения данных

nextTick(initMap)

} catch (err) {

error.value = 'Ошибка загрузки: ' + (err.response?.data?.detail || err.message)

} finally {

isLoading.value = false

}

}

// Инициализация карты

const initMap = () => {

// Удаляем старую карту если есть

if (map) {

map.remove()

map = null

}

// Проверяем что есть данные и контейнер

if (!previewData.value || !mapContainer.value) return

const points = previewData.value.points

const coords = points.map(p => [p.latitude, p.longitude])

// Создаем карту

map = L.map(mapContainer.value, {

zoomControl: true,

attributionControl: false

})

// Добавляем слой с картой

L.tileLayer('https://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {

maxZoom: 17,

attribution: '&copy; <a href="https://opentopomap.org">OpenTopoMap</a> contributors'

}).addTo(map)

// Создаем полилинию

polyline = L.polyline(coords, {

color: '#ff0000',

weight: 4,

opacity: 0.7,

lineJoin: 'round',

lineCap: 'round'

}).addTo(map)

// Добавляем маркеры старта и финиша

if (points.length > 0) {

startMarker = L.marker([points[0].latitude, points[0].longitude], {

title: 'Старт'

}).addTo(map)

endMarker = L.marker([points[points.length - 1].latitude, points[points.length - 1].longitude], {

title: 'Финиш'

}).addTo(map)

}

// Центрируем карту на треке

map.fitBounds(polyline.getBounds(), {

padding: [20, 20]

})

}

// Очищаем карту при удалении компонента

onBeforeUnmount(() => {

if (map) map.remove()

})

</script>

<style scoped>

.track-upload {

max-width: 800px;

margin: 0 auto;

padding: 2rem;

}

.form-group {

margin-bottom: 1.5rem;

}

label {

display: block;

margin-bottom: 0.5rem;

font-weight: bold;

}

input, textarea {

width: 100%;

padding: 0.75rem;

border: 1px solid #ddd;

border-radius: 4px;

font-size: 1rem;

}

button {

padding: 0.75rem 1.5rem;

background-color: #42b983;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 1rem;

}

button:disabled {

background-color: #ccc;

}

.preview {

margin-top: 2rem;

padding: 1.5rem;

border: 1px solid #eee;

border-radius: 8px;

background-color: #f9f9f9;

}

.preview-map {

height: 500px;

width: 100%;

margin-bottom: 1.5rem;

border-radius: 8px;

overflow: hidden;

box-shadow: 0 2px 8px rgba(0,0,0,0.1);

}

/\* Исправляем стили иконок Leaflet \*/

:deep(.leaflet-marker-icon) {

background-image: none !important;

}

:deep(.leaflet-div-icon) {

background: none !important;

border: none !important;

}

.preview-image {

display: flex;

align-items: center;

justify-content: center;

}

.preview-image img {

max-width: 100%;

height: 1000px;

border-radius: 4px;

box-shadow: 0 2px 8px rgba(0,0,0,0.1);

}

.track-stats {

margin-top: 1.5rem;

}

.track-stats p {

margin: 0.5rem 0;

}

.error {

margin-top: 1rem;

padding: 1rem;

background-color: #ffebee;

color: #f44336;

border-radius: 4px;

}

</style>

Файл TrackEditForm.vue

<template>

<div class="track-edit-form">

<form @submit.prevent="handleSubmit">

<div class="form-group">

<label>Название трека \*</label>

<input

v-model="formData.title"

type="text"

maxlength="80"

required

>

</div>

<div class="form-group">

<label>Описание</label>

<textarea v-model="formData.description"></textarea>

</div>

<div class="form-actions">

<button type="submit" :disabled="loading">

{{ loading ? 'Сохранение...' : 'Сохранить изменения' }}

</button>

<button type="button" @click="cancel" :disabled="loading">

Отмена

</button>

</div>

<div v-if="error" class="error">

{{ error }}

</div>

</form>

</div>

</template>

<script setup>

import { ref } from 'vue'

import api from '@/services/api'

const props = defineProps({

track: {

type: Object,

required: true

}

})

const emit = defineEmits(['update', 'cancel'])

const formData = ref({

title: props.track.title,

description: props.track.description || '',

})

const loading = ref(false)

const error = ref(null)

const handleSubmit = async () => {

try {

loading.value = true

error.value = null

formData.value.title = formData.value.title.trim()

const response = await api.updateTrack(props.track.id, formData.value)

emit('update', response.data)

} catch (err) {

error.value = 'Ошибка обновления: ' + (err.response?.data?.detail || err.message)

} finally {

loading.value = false

}

}

const cancel = () => {

emit('cancel')

}

</script>

<style scoped>

.track-edit-form {

padding: 1rem;

border: 1px solid #eee;

border-radius: 8px;

margin-top: 1rem;

background-color: #f9f9f9;

}

.form-group {

margin-bottom: 1rem;

}

label {

display: block;

margin-bottom: 0.5rem;

font-weight: 500;

}

input, textarea {

width: 100%;

padding: 0.75rem;

border: 1px solid #ddd;

border-radius: 4px;

font-size: 1rem;

}

textarea {

min-height: 100px;

}

.form-actions {

display: flex;

gap: 1rem;

margin-top: 1rem;

}

button {

padding: 0.75rem 1.5rem;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 1rem;

flex: 1;

}

button[type="submit"] {

background-color: #42b983;

color: white;

}

button[type="button"] {

background-color: #f0f0f0;

color: #333;

}

.error {

margin-top: 1rem;

padding: 0.75rem;

background-color: #f8d7da;

color: #721c24;

border-radius: 4px;

}

</style>

Файл UserFavoritesList.vue

<template>

<div class="user-favorites-list">

<div class="header">

<h2>Избранные треки</h2>

<button @click="refresh" :disabled="loading">

Обновить

</button>

</div>

<div v-if="loading" class="loading">

Загрузка избранных треков...

</div>

<div v-else-if="error" class="error">

{{ error }}

<button @click="refresh">Повторить попытку</button>

</div>

<div v-else>

<div v-if="tracks.length === 0" class="empty-state">

У вас пока нет избранных треков

<router-link to="/" class="btn">

Найти треки

</router-link>

</div>

<div v-else class="tracks-container">

<TrackCard

v-for="track in tracks"

:key="track.id"

:track="track"

@update-track="handleTrackUpdate"

/>

</div>

</div>

</div>

</template>

<script setup>

import TrackCard from '@/components/TrackCard.vue'

import { defineProps, defineEmits, ref } from 'vue'

const props = defineProps({

tracks: Array,

loading: Boolean,

error: String

})

const emit = defineEmits(['refresh', 'track-updated'])

const localTracks = ref([...props.tracks])

const handleTrackUpdate = (updatedTrack) => {

if (!updatedTrack.is\_favorite) {

const index = localTracks.value.findIndex(t => t.id === updatedTrack.id)

if (index !== -1) {

localTracks.value.splice(index, 1)

}

}

emit('track-updated', updatedTrack)

}

const refresh = () => {

emit('refresh')

}

</script>

<style scoped>

.header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1.5rem;

}

.tracks-container {

display: flex;

flex-direction: column;

gap: 1.5rem;

margin-top: 2rem;

}

.empty-state {

text-align: center;

padding: 2rem;

background-color: #f9f9f9;

border-radius: 8px;

}

.empty-state .btn {

margin-top: 1rem;

display: inline-block;

}

.loading, .error {

text-align: center;

padding: 2rem;

}

.error button {

margin-top: 0.5rem;

padding: 0.5rem 1rem;

background-color: #f44336;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

</style>

Файл UserSettingsForm.vue

<template>

<div class="user-settings-form">

<h2>Редактирование профиля</h2>

<form @submit.prevent="handleSubmit">

<div class="form-group">

<label>Никнейм</label>

<input type="text" v-model="formData.username" required>

</div>

<div class="form-group">

<label>Email</label>

<input type="email" v-model="formData.email" required>

</div>

<div class="form-group">

<label>Новый пароль (оставьте пустым, чтобы не менять)</label>

<input type="password" v-model="formData.password">

</div>

<div class="form-group">

<label>Повторите пароль</label>

<input type="password" v-model="formData.confirmPassword">

</div>

<button type="submit" :disabled="loading">

{{ loading ? 'Сохранение...' : 'Сохранить изменения' }}

</button>

<div v-if="successMessage" class="success">

{{ successMessage }}

</div>

<div v-if="error" class="error">

{{ error }}

</div>

</form>

</div>

</template>

<script setup>

import { ref, reactive } from 'vue'

import api from '@/services/api'

const props = defineProps({

user: {

type: Object,

required: true

}

})

const emit = defineEmits(['update'])

const formData = reactive({

username: props.user.username,

email: props.user.email,

password: '',

confirmPassword: ''

})

const loading = ref(false)

const error = ref(null)

const successMessage = ref(null)

const handleSubmit = async () => {

if (formData.password && formData.password !== formData.confirmPassword) {

error.value = 'Пароли не совпадают'

return

}

try {

loading.value = true

error.value = null

successMessage.value = null

const updateData = {

username: formData.username,

email: formData.email

}

if (formData.password) {

updateData.password = formData.password

}

const response = await api.updateCurrentUser(updateData)

// Очищаем поля паролей

formData.password = ''

formData.confirmPassword = ''

successMessage.value = 'Профиль успешно обновлен'

emit('update', response.data)

} catch (err) {

error.value = 'Ошибка обновления: ' + (err.response?.data?.detail || err.message)

} finally {

loading.value = false

}

}

</script>

<style scoped>

.user-settings-form {

max-width: 600px;

margin: 0 auto;

}

.form-group {

margin-bottom: 1.5rem;

}

label {

display: block;

margin-bottom: 0.5rem;

font-weight: 500;

}

input {

width: 100%;

padding: 0.75rem;

border: 1px solid #ddd;

border-radius: 4px;

font-size: 1rem;

}

button {

padding: 0.75rem 1.5rem;

background-color: #42b983;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 1rem;

}

.success {

margin-top: 1rem;

padding: 0.75rem;

background-color: #d4edda;

color: #155724;

border-radius: 4px;

}

.error {

margin-top: 1rem;

padding: 0.75rem;

background-color: #f8d7da;

color: #721c24;

border-radius: 4px;

}

</style>

Файл UserTracksList.vue

<template>

<div class="user-tracks-list">

<div class="header">

<h2>Мои треки</h2>

<button @click="loadMyTracks" :disabled="loading">

Обновить

</button>

</div>

<div v-if="loading" class="loading">

Загрузка треков...

</div>

<div v-else-if="error" class="error">

{{ error }}

<button @click="loadMyTracks">Повторить попытку</button>

</div>

<div v-else>

<div v-if="tracks.length === 0" class="empty-state">

Вы еще не создали ни одного трека

<router-link to="/track\_load" class="btn">

Создать первый трек

</router-link>

</div>

<div v-else class="tracks-container">

<div v-for="track in tracks" :key="track.id" class="track-item">

<div v-if="editingTrackId !== track.id" class="track-info">

<div class="track-info">

<h3>{{ track.title }}</h3>

<p>{{ track.description || 'Без описания' }}</p>

<div class="track-meta">

<span>Создан: {{ formatDate(track.created\_at) }}</span>

</div>

</div>

</div>

<TrackEditForm

v-else

:track="track"

@update="handleTrackUpdate"

@cancel="cancelEdit"

/>

<div class="track-actions">

<router-link :to="`/tracks/${track.id}`" class="btn">

Просмотреть

</router-link>

<button

v-if="editingTrackId !== track.id"

@click="startEditing(track.id)"

class="btn edit"

>

Изменить

</button>

<button

v-if="editingTrackId !== track.id"

@click="confirmDelete(track.id)"

class="btn delete"

>

Удалить

</button>

</div>

</div>

<div class="pagination-container" v-if="pagination.total > pagination.limit">

<button

@click="prevPage"

:disabled="pagination.skip === 0">

Назад

</button>

<span>Страница {{ currentPage }} из {{ totalPages }}</span>

<button

@click="nextPage"

:disabled="pagination.skip + pagination.limit >= pagination.total">

Вперед

</button>

</div>

</div>

</div>

</div>

</template>

<script setup>

import { defineEmits, onMounted, ref, computed } from 'vue'

import TrackEditForm from '@/components/profile/TrackEditForm.vue'

import api from '@/services/api'

// Состояние для моих треков

const tracks = ref([])

const loading = ref(false)

const error = ref(null)

const pagination = ref({

skip: 0,

limit: 10,

total: 0

})

onMounted (async () => {

loadMyTracks()

})

// Загрузка моих треков

const loadMyTracks = async () => {

try {

tracks.value = true

error.value = null

const response = await api.getUserTracks(pagination.value.skip, pagination.value.limit)

tracks.value = response.data.tracks

pagination.value.total = response.data.total

} catch (error) {

error.value = 'Ошибка загрузки треков: ' + error.message

} finally {

loading.value = false

}

}

// Добавим метод для обновления списка

const refresh = async () => {

pagination.value.skip = 0

await loadMyTracks()

}

// Сделаем метод доступным извне

defineExpose({

refresh

})

const emit = defineEmits(['delete', 'update-track'])

const formatDate = (dateString) => {

return new Date(dateString).toLocaleDateString()

}

// При удалении трека обновим список сразу

const confirmDelete = async (trackId) => {

if (confirm('Вы уверены, что хотите удалить этот трек? Это действие нельзя отменить.')) {

try {

loading.value = true

await emit('delete', trackId)

// Удаляем трек из локального списка

const index = tracks.value.findIndex(t => t.id === trackId)

if (index !== -1) {

tracks.value.splice(index, 1)

}

// Обновляем пагинацию

pagination.value.total -= 1

} catch (error) {

console.error('Ошибка удаления трека:', error)

error.value = 'Ошибка удаления трека'

} finally {

loading.value = false

}

}

}

const editingTrackId = ref(null)

const startEditing = (trackId) => {

editingTrackId.value = trackId

}

const cancelEdit = () => {

editingTrackId.value = null

}

// Улучшим обработку обновления трека

const handleTrackUpdate = (updatedTrack) => {

const index = tracks.value.findIndex(t => t.id === updatedTrack.id)

if (index !== -1) {

// Обновляем трек в списке

tracks.value[index] = updatedTrack

}

editingTrackId.value = null

emit('track-updated', updatedTrack)

}

const currentPage = computed(() => Math.floor(pagination.value.skip / pagination.value.limit) + 1)

const totalPages = computed(() => Math.ceil(pagination.value.total / pagination.value.limit))

// Методы пагинации

const nextPage = () => {

pagination.value.skip += pagination.value.limit

loadMyTracks()

}

const prevPage = () => {

pagination.value.skip = Math.max(0, pagination.value.skip - pagination.value.limit)

loadMyTracks()

}

</script>

<style scoped>

.header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1.5rem;

}

.track-item {

display: flex;

justify-content: space-between;

padding: 1.5rem;

border: 1px solid #eee;

border-radius: 8px;

margin-bottom: 1rem;

background: white;

box-shadow: 0 2px 4px rgba(0,0,0,0.05);

}

.track-info {

flex: 1;

}

.track-info h3 {

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

max-width: 250px;

}

.track-info p {

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

max-width: 300px;

}

.track-meta {

display: flex;

gap: 1rem;

margin-top: 0.5rem;

font-size: 0.9rem;

color: #666;

}

.track-actions {

display: flex;

gap: 0.5rem;

align-items: center;

}

.btn {

padding: 0.5rem 1rem;

border-radius: 4px;

text-decoration: none;

cursor: pointer;

border: none;

}

.btn {

background-color: #42b983;

color: white;

}

.btn.edit {

background-color: #ffc107;

color: #333;

}

.btn.delete {

background-color: #ff6b6b;

}

.empty-state {

text-align: center;

padding: 2rem;

background-color: #f9f9f9;

border-radius: 8px;

}

.empty-state .btn {

margin-top: 1rem;

display: inline-block;

}

.loading, .error {

text-align: center;

padding: 2rem;

}

.error button {

margin-top: 0.5rem;

padding: 0.5rem 1rem;

background-color: #f44336;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.btn.edit {

background-color: #ffc107;

color: #333;

}

.track-actions {

display: flex;

gap: 0.5rem;

}

.pagination-container {

display: flex;

justify-content: center;

align-items: center;

gap: 1rem;

margin-top: 2rem;

padding: 1rem;

}

.pagination-container button {

padding: 0.5rem 1rem;

background-color: #42b983;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.pagination-container button:disabled {

background-color: #cccccc;

cursor: not-allowed;

}

</style>

Файл AdminTracksList.vue

<template>

<div class="user-tracks-list">

<div class="header">

<h2>Мои треки</h2>

<button @click="loadAllTracks" :disabled="loading">

Обновить

</button>

</div>

<div v-if="loading" class="loading">

Загрузка треков...

</div>

<div v-else-if="error" class="error">

{{ error }}

<button @click="loadAllTracks">Повторить попытку</button>

</div>

<div v-else>

<div v-if="tracks.length === 0" class="empty-state">

Вы еще не создали ни одного трека

<router-link to="/upload" class="btn">

Создать первый трек

</router-link>

</div>

<div v-else class="tracks-container">

<div v-for="track in tracks" :key="track.id" class="track-item">

<div v-if="editingTrackId !== track.id" class="track-info">

<h3>{{ track.title }}</h3>

<p>{{ track.description || 'Без описания' }}</p>

<div class="track-meta">

<span>Создан: {{ formatDate(track.created\_at) }}</span>

</div>

</div>

<TrackEditForm

v-else

:track="track"

@update="handleTrackUpdate"

@cancel="cancelEdit"

/>

<div class="track-actions">

<router-link :to="`/tracks/${track.id}`" class="btn">

Просмотреть

</router-link>

<button

v-if="editingTrackId !== track.id"

@click="startEditing(track.id)"

class="btn edit"

>

Изменить

</button>

<button

v-if="editingTrackId !== track.id"

@click="confirmDelete(track.id)"

class="btn delete"

>

Удалить

</button>

</div>

</div>

<div class="pagination-container" v-if="pagination.total > pagination.limit">

<button

@click="prevPage"

:disabled="pagination.skip === 0">

Назад

</button>

<span>Страница {{ currentPage }} из {{ totalPages }}</span>

<button

@click="nextPage"

:disabled="pagination.skip + pagination.limit >= pagination.total">

Вперед

</button>

</div>

</div>

</div>

</div>

</template>

<script setup>

import { defineEmits, onMounted, ref, computed } from 'vue'

import TrackEditForm from '@/components/profile/TrackEditForm.vue'

import api from '@/services/api'

// Состояние для моих треков

const tracks = ref([])

const loading = ref(false)

const error = ref(null)

const pagination = ref({

skip: 0,

limit: 10,

total: 0

})

onMounted (async () => {

loadAllTracks()

})

// Загрузка моих треков

const loadAllTracks = async () => {

try {

tracks.value = true

error.value = null

const response = await api.getAllTracks(pagination.value.skip, pagination.value.limit)

tracks.value = response.data.tracks

pagination.value.total = response.data.total

} catch (error) {

error.value = 'Ошибка загрузки треков: ' + error.message

} finally {

loading.value = false

}

}

// Добавим метод для обновления списка

const refresh = async () => {

pagination.value.skip = 0

await loadAllTracks()

}

// Сделаем метод доступным извне

defineExpose({

refresh

})

const emit = defineEmits(['delete', 'update-track'])

const formatDate = (dateString) => {

return new Date(dateString).toLocaleDateString()

}

// При удалении трека обновим список сразу

const confirmDelete = async (trackId) => {

if (confirm('Вы уверены, что хотите удалить этот трек? Это действие нельзя отменить.')) {

try {

loading.value = true

await emit('delete', trackId)

// Удаляем трек из локального списка

const index = tracks.value.findIndex(t => t.id === trackId)

if (index !== -1) {

tracks.value.splice(index, 1)

}

// Обновляем пагинацию

pagination.value.total -= 1

} catch (error) {

console.error('Ошибка удаления трека:', error)

error.value = 'Ошибка удаления трека'

} finally {

loading.value = false

}

}

}

const editingTrackId = ref(null)

const startEditing = (trackId) => {

editingTrackId.value = trackId

}

const cancelEdit = () => {

editingTrackId.value = null

}

// Улучшим обработку обновления трека

const handleTrackUpdate = (updatedTrack) => {

const index = tracks.value.findIndex(t => t.id === updatedTrack.id)

if (index !== -1) {

// Обновляем трек в списке

tracks.value[index] = updatedTrack

}

editingTrackId.value = null

emit('track-updated', updatedTrack)

}

const currentPage = computed(() => Math.floor(pagination.value.skip / pagination.value.limit) + 1)

const totalPages = computed(() => Math.ceil(pagination.value.total / pagination.value.limit))

// Методы пагинации

const nextPage = () => {

pagination.value.skip += pagination.value.limit

loadAllTracks()

}

const prevPage = () => {

pagination.value.skip = Math.max(0, pagination.value.skip - pagination.value.limit)

loadAllTracks()

}

</script>

<style scoped>

.header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1.5rem;

}

.track-item {

display: flex;

justify-content: space-between;

padding: 1.5rem;

border: 1px solid #eee;

border-radius: 8px;

margin-bottom: 1rem;

background: white;

box-shadow: 0 2px 4px rgba(0,0,0,0.05);

}

.track-info {

flex: 1;

}

.track-info h3 {

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

max-width: 250px;

}

.track-info p {

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

max-width: 300px;

}

.track-meta {

display: flex;

gap: 1rem;

margin-top: 0.5rem;

font-size: 0.9rem;

color: #666;

}

.track-actions {

display: flex;

gap: 0.5rem;

align-items: center;

}

.btn {

padding: 0.5rem 1rem;

border-radius: 4px;

text-decoration: none;

cursor: pointer;

border: none;

}

.btn {

background-color: #42b983;

color: white;

}

.btn.edit {

background-color: #ffc107;

color: #333;

}

.btn.delete {

background-color: #ff6b6b;

}

.empty-state {

text-align: center;

padding: 2rem;

background-color: #f9f9f9;

border-radius: 8px;

}

.empty-state .btn {

margin-top: 1rem;

display: inline-block;

}

.loading, .error {

text-align: center;

padding: 2rem;

}

.error button {

margin-top: 0.5rem;

padding: 0.5rem 1rem;

background-color: #f44336;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.btn.edit {

background-color: #ffc107;

color: #333;

}

.track-actions {

display: flex;

gap: 0.5rem;

}

.pagination-container {

display: flex;

justify-content: center;

align-items: center;

gap: 1rem;

margin-top: 2rem;

padding: 1rem;

}

.pagination-container button {

padding: 0.5rem 1rem;

background-color: #42b983;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.pagination-container button:disabled {

background-color: #cccccc;

cursor: not-allowed;

}

</style>

Файл AdminUsersList.vue

<template>

<div class="admin-users-list">

<div class="header">

<h2>Управление пользователями</h2>

<button @click="refresh" :disabled="loading">

Обновить

</button>

</div>

<div v-if="loading" class="loading">

Загрузка пользователей...

</div>

<div v-else-if="error" class="error">

{{ error }}

<button @click="refresh">Повторить попытку</button>

</div>

<div v-else>

<div class="users-container">

<div v-for="user in users" :key="user.id" class="user-item">

<div class="user-info">

<h3>{{ user.username }}</h3>

<p>Email: {{ user.email }}</p>

<p>Статус:

<span :class="user.is\_active ? 'active' : 'inactive'">

{{ user.is\_active ? 'Активен' : 'Неактивен' }}

</span>

</p>

<p>Роль: {{ user.is\_admin ? 'Администратор' : 'Пользователь' }}</p>

</div>

<div class="user-actions">

<button

v-if="user.is\_active"

@click="deactivateUser(user.id)"

class="btn deactivate"

>

Деактивировать

</button>

<button

v-else

@click="activateUser(user.id)"

class="btn activate"

>

Активировать

</button>

</div>

</div>

</div>

<div class="pagination-container" v-if="pagination.total > pagination.limit">

<button

@click="prevPage"

:disabled="pagination.skip === 0"

>

Назад

</button>

<span>Страница {{ currentPage }} из {{ totalPages }}</span>

<button

@click="nextPage"

:disabled="pagination.skip + pagination.limit >= pagination.total"

>

Вперед

</button>

</div>

</div>

</div>

</template>

<script setup>

import { ref, onMounted, computed, defineExpose, defineEmits } from 'vue'

import api from '@/services/api'

const users = ref([])

const loading = ref(false)

const error = ref(null)

const pagination = ref({

skip: 0,

limit: 10,

total: 0

})

onMounted(() => {

fetchUsers()

})

const fetchUsers = async () => {

try {

loading.value = true

error.value = null

const response = await api.getAllUsers(pagination.value.skip, pagination.value.limit)

users.value = response.data.users

pagination.value.total = response.data.total

} catch (err) {

error.value = 'Ошибка загрузки пользователей'

} finally {

loading.value = false

}

}

const refresh = () => {

pagination.value.skip = 0

fetchUsers()

}

// Сделаем метод доступным извне

defineExpose({

refresh

})

const emit = defineEmits(['deactivate', 'activate'])

const deactivateUser = async (userId) => {

try {

await api.updateUserActive(userId, false)

const user = users.value.find(u => u.id === userId)

if (user) {

user.is\_active = false

}

} catch (err) {

error.value = 'Ошибка деактивации пользователя'

}

}

const activateUser = async (userId) => {

try {

await api.updateUserActive(userId, true)

const user = users.value.find(u => u.id === userId)

if (user) {

user.is\_active = true

}

} catch (err) {

error.value = 'Ошибка активации пользователя'

}

}

const currentPage = computed(() => Math.floor(pagination.value.skip / pagination.value.limit) + 1)

const totalPages = computed(() => Math.ceil(pagination.value.total / pagination.value.limit))

const nextPage = () => {

pagination.value.skip += pagination.value.limit

fetchUsers()

}

const prevPage = () => {

pagination.value.skip = Math.max(0, pagination.value.skip - pagination.value.limit)

fetchUsers()

}

</script>

<style scoped>

.header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 1.5rem;

}

.user-item {

display: flex;

justify-content: space-between;

padding: 1.5rem;

border: 1px solid #eee;

border-radius: 8px;

margin-bottom: 1rem;

background: white;

}

.user-info h3 {

margin-top: 0;

}

.user-actions {

display: flex;

align-items: center;

}

.btn {

padding: 0.5rem 1rem;

border-radius: 4px;

cursor: pointer;

border: none;

}

.btn.deactivate {

background-color: #ffc107;

color: #333;

}

.btn.activate {

background-color: #28a745;

color: white;

}

.active {

color: #28a745;

font-weight: 500;

}

.inactive {

color: #dc3545;

font-weight: 500;

}

.pagination-container {

display: flex;

justify-content: center;

align-items: center;

gap: 1rem;

margin-top: 2rem;

padding: 1rem;

}

</style>

Файл HomeView.vue

<template>

<div class="home-view">

<AuthHeader />

<main>

<h1>Треки</h1>

<div v-if="loading" class="loading">

Загрузка треков...

</div>

<div v-else-if="error" class="error">

{{ error }}

<button @click="fetchTracks">Повторить попытку</button>

</div>

<div v-else>

<div v-if="tracks.length == 0" class="empty-state">

<p>Пока нет ни одного трека</p>

</div>

<div v-else class="tracks-container">

<TrackCard

v-for="track in tracks"

:key="track.id"

:track="track"

@update-track="handleTrackUpdate"

/>

<div class="pagination-container" v-if="pagination.total > pagination.limit">

<button

@click="prevPage"

:disabled="pagination.skip === 0">

Назад

</button>

<span>Страница {{ currentPage }} из {{ totalPages }}</span>

<button

@click="nextPage"

:disabled="pagination.skip + pagination.limit >= pagination.total">

Вперед

</button>

</div>

</div>

</div>

</main>

</div>

</template>

<script setup>

import { ref, onMounted, computed } from 'vue'

import api from '@/services/api'

import TrackCard from '@/components/TrackCard.vue'

import AuthHeader from '@/components/AuthHeader.vue'

import { useFavoritesStore } from '@/stores/favorites'

import { useAuthStore } from '@/stores/auth'

const authStore = useAuthStore()

const favoritesStore = useFavoritesStore()

const tracks = ref([])

const loading = ref(true)

const error = ref(null)

const pagination = ref({

skip: 0,

limit: 10,

total: 0

})

onMounted(async () => {

document.title = "Главная"

await fetchTracks()

if (authStore.isAuthenticated) {

await favoritesStore.loadFavorites()

}

})

const fetchTracks = async () => {

try {

loading.value = true

error.value = null

const response = await api.getAllTracks(

pagination.value.skip,

pagination.value.limit

)

// Сохраняем треки как есть - сервер должен возвращать is\_favorite

tracks.value = response.data.tracks

pagination.value.total = response.data.total

} catch (err) {

error.value = 'Ошибка загрузки треков'

throw err

} finally {

loading.value = false

}

}

const handleTrackUpdate = (updatedTrack) => {

const index = tracks.value.findIndex(t => t.id === updatedTrack.id)

if (index !== -1) {

tracks.value[index] = updatedTrack

}

}

const currentPage = computed(() => Math.floor(pagination.value.skip / pagination.value.limit) + 1)

const totalPages = computed(() => Math.ceil(pagination.value.total / pagination.value.limit))

// Методы пагинации

const nextPage = () => {

pagination.value.skip += pagination.value.limit

fetchTracks()

}

const prevPage = () => {

pagination.value.skip = Math.max(0, pagination.value.skip - pagination.value.limit)

fetchTracks()

}

</script>

<style scoped>

.home-view {

min-height: 100vh;

display: flex;

flex-direction: column;

}

main {

padding: 2rem;

max-width: 1200px;

margin: 0 auto;

width: 100%;

}

.tracks-container {

display: flex;

flex-direction: column;

gap: 1.5rem;

margin-top: 2rem;

}

.loading {

text-align: center;

padding: 2rem;

}

.pagination-container {

display: flex;

justify-content: center;

align-items: center;

gap: 1rem;

margin-top: 2rem;

padding: 1rem;

}

.pagination-container button {

padding: 0.5rem 1rem;

background-color: #42b983;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

.pagination-container button:disabled {

background-color: #cccccc;

cursor: not-allowed;

}

.empty-state {

text-align: center;

padding: 2rem;

font-size: 1.2rem;

color: #666;

}

.error {

background-color: #ffebee;

color: #f44336;

padding: 1rem;

border-radius: 4px;

text-align: center;

margin-top: 1rem;

}

.error button {

margin-top: 0.5rem;

padding: 0.5rem 1rem;

background-color: #f44336;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

}

</style>

Файл LoginView.vue

<template>

<div class="login-view">

<AuthHeader />

<main>

<LoginForm />

</main>

</div>

</template>

<script setup>

import AuthHeader from '@/components/AuthHeader.vue'

import LoginForm from '@/components/LoginForm.vue'

</script>

<style scoped>

.login-view {

min-height: 100vh;

display: flex;

flex-direction: column;

}

main {

display: flex;

justify-content: center;

align-items: center;

flex-grow: 1;

padding: 2rem;

}

</style>

Файл ProfileView.vue

<template>

<div class="profile-view">

<AuthHeader />

<main>

<div class="profile-header">

<h1>Профиль пользователя: {{ user.username }}</h1>

<p>Email: {{ user.email }}</p>

<p>Зарегистрирован: {{ formatDate(user.created\_at) }}</p>

</div>

<div class="tabs">

<button

v-for="tab in tabs"

:key="tab.id"

@click="activeTab = tab.id"

:class="{ active: activeTab === tab.id }"

>

{{ tab.label }}

</button>

</div>

<div class="tab-content">

<div v-if="activeTab === 'settings'">

<UserSettingsForm :user="user" @update="handleUserUpdate" />

</div>

<div v-if="activeTab === 'my-tracks'">

<UserTracksList

@delete="deleteTrack"

@track-updated="handleTrackUpdate"

/>

</div>

<div v-if="activeTab === 'favorites'">

<UserFavoritesList

:tracks="favoriteTracks"

:loading="loadingFavorites"

:error="errorFavorites"

@refresh="loadFavoriteTracks"

@track-updated="handleTrackUpdate"

/>

</div>

<div v-if="activeTab === 'all-users'">

<AdminUsersList />

</div>

<div v-if="activeTab === 'all-tracks'">

<AdminTracksList

@delete="deleteTrack"

@track-updated="handleTrackUpdate"/>

</div>

</div>

</main>

</div>

</template>

<script setup>

import { ref, onMounted, watch, computed } from 'vue'

import { useAuthStore } from '@/stores/auth'

import AuthHeader from '@/components/AuthHeader.vue'

import UserSettingsForm from '@/components/profile/UserSettingsForm.vue'

import UserTracksList from '@/components/profile/UserTracksList.vue'

import UserFavoritesList from '@/components/profile/UserFavoritesList.vue'

import AdminUsersList from '@/components/admin/AdminUsersList.vue'

import AdminTracksList from '@/components/admin/AdminTracksList.vue'

import api from '@/services/api'

const authStore = useAuthStore()

const user = ref({ ...authStore.user })

const userTracksListRef = ref(null)

const tabs = computed(() => {

const baseTabs = [

{ id: 'settings', label: 'Настройки профиля' },

{ id: 'my-tracks', label: 'Мои треки' },

{ id: 'favorites', label: 'Избранные треки' }

]

// Добавляем админские вкладки, если пользователь админ

if (authStore.user?.is\_admin) {

return [

...baseTabs,

{ id: 'all-users', label: 'Все пользователи' },

{ id: 'all-tracks', label: 'Все треки' }

]

}

return baseTabs

})

const activeTab = ref('settings')

// Состояние для избранных треков

const favoriteTracks = ref([])

const loadingFavorites = ref(false)

const errorFavorites = ref(null)

onMounted(() => {

// Загружаем данные при первом открытии соответствующих вкладок

if (activeTab.value === 'favorites') loadFavoriteTracks()

})

// Загружаем данные при переключении вкладок

watch(activeTab, (newTab) => {

if (newTab === 'favorites' && favoriteTracks.value.length === 0) {

loadFavoriteTracks()

}

})

// Форматирование даты

const formatDate = (dateString) => {

return new Date(dateString).toLocaleDateString()

}

// Загрузка избранных треков

const loadFavoriteTracks = async () => {

try {

loadingFavorites.value = true

errorFavorites.value = null

const response = await api.getFavoriteTracks()

favoriteTracks.value = response.data.tracks

} catch (error) {

errorFavorites.value = 'Ошибка загрузки избранных треков: ' + error.message

} finally {

loadingFavorites.value = false

}

}

// Обработка обновления пользователя

const handleUserUpdate = (updatedUser) => {

user.value = updatedUser

authStore.user = updatedUser

}

// Обновленный метод удаления трека

const deleteTrack = async (trackId) => {

try {

await api.deleteTrack(trackId)

// Обновляем список треков

if (userTracksListRef.value) {

await userTracksListRef.value.refresh()

}

// Обновляем избранное

loadFavoriteTracks()

} catch (error) {

console.error('Ошибка удаления трека:', error)

// Можно добавить обработку ошибки

}

}

// Обновленный обработчик изменения трека

const handleTrackUpdate = (updatedTrack) => {

// Обновляем избранное

const index = favoriteTracks.value.findIndex(t => t.id === updatedTrack.id)

if (index !== -1) {

if (!updatedTrack.is\_favorite) {

favoriteTracks.value.splice(index, 1)

} else {

favoriteTracks.value[index] = updatedTrack

}

}

// Обновляем список треков

if (userTracksListRef.value) {

userTracksListRef.value.refresh()

}

}

</script>

<style scoped>

.profile-view {

min-height: 100vh;

display: flex;

flex-direction: column;

}

main {

padding: 2rem;

max-width: 1200px;

margin: 0 auto;

width: 100%;

}

.profile-header {

margin-bottom: 2rem;

padding-bottom: 1rem;

border-bottom: 1px solid #eee;

}

.tabs {

display: flex;

border-bottom: 1px solid #ddd;

margin-bottom: 1.5rem;

}

.tabs button {

padding: 0.75rem 1.5rem;

background: none;

border: none;

cursor: pointer;

font-size: 1rem;

position: relative;

bottom: -1px;

}

.tabs button.active {

border: 1px solid #ddd;

border-bottom: 1px solid white;

border-top-left-radius: 4px;

border-top-right-radius: 4px;

border-bottom: none;

}

.tab-content {

padding: 1rem;

}

</style>

Файл RegisterView.vue

<template>

<div class="register-view">

<AuthHeader />

<main>

<RegisterForm />

</main>

</div>

</template>

<script setup>

import AuthHeader from '@/components/AuthHeader.vue'

import RegisterForm from '@/components/RegisterForm.vue'

</script>

<style scoped>

.register-view {

min-height: 100vh;

display: flex;

flex-direction: column;

}

main {

display: flex;

justify-content: center;

align-items: center;

flex-grow: 1;

padding: 2rem;

}

</style>

Файл TrackDetailView.vue

<template>

<div class="track-detail-view">

<AuthHeader />

<main v-if="loading" class="loading-container">

<div class="loader"></div>

<p>Загрузка данных о треке...</p>

</main>

<main v-else-if="error" class="error-container">

<p>{{ error }}</p>

<button @click="fetchTrackData">Повторить попытку</button>

</main>

<main v-else class="detail-container">

<!-- Верхняя часть: карта и информация о треке -->

<div class="top-section">

<div class="map-section">

<MapView

:points="track.points"

:key="`map-${track.id}`"

/>

</div>

<div class="info-section">

<div class="track-info">

<h1 class="track-title">{{ track.title }}</h1>

<div class="scrollable-description">

<p>{{ track.description }}</p>

</div>

<div class="track-meta">

<div class="meta-item">

<span class="meta-label">Регион:</span>

<span class="meta-value">{{ track.region || 'Не указан' }}</span>

</div>

<div class="meta-item">

<span class="meta-label">Автор:</span>

<span class="meta-value">{{ track.owner.username }}</span>

</div>

<div class="meta-item">

<span class="meta-label">Создан:</span>

<span class="meta-value">{{ formatDate(track.created\_at) }}</span>

</div>

<div class="meta-item">

<span class="meta-label">Длина:</span>

<span class="meta-value">{{ (track.total\_distance / 1000) }} км</span>

</div>

<div class="meta-item">

<span class="meta-label">Набор высоты:</span>

<span class="meta-value">{{ track.elevation\_gain }} м</span>

</div>

<div class="meta-item">

<span class="meta-label">Сложность:</span>

<div class="difficulty-card" :class="`difficulty-${track.difficulty}`">

<div class="difficulty-indicator">

<span v-for="i in 5" :key="i" :class="{ active: i <= track.difficulty }"></span>

</div>

</div>

</div>

</div>

</div>

<div class="forecast-section">

<h2>Прогноз пожароопасности</h2>

<div class="forecast-form">

<button

class="btn"

@click="openFireRiskMap"

:disabled="loadingRiskMap"

>

{{ loadingRiskMap ? 'Подготовка карты...' : 'Открыть карту пожароопасности' }}

</button>

</div>

<div v-if="riskMapError" class="error">

{{ riskMapError }}

</div>

</div>

</div>

</div>

<!-- Нижняя часть: комментарии на всю ширину -->

<div class="comments-section">

<h2>Комментарии</h2>

<div v-if="authStore.isAuthenticated" class="comment-form">

<textarea v-model="newComment" placeholder="Ваш комментарий..." rows="3" maxlength="300"></textarea>

<button class="btn" @click="addComment" :disabled="!newComment.trim()">

Отправить

</button>

</div>

<div v-else class="auth-notice">

<p>Чтобы оставить комментарий, <router-link to="/login">войдите</router-link> в систему.</p>

</div>

<div v-if="track.comments.length === 0" class="no-comments">

Пока нет комментариев. Будьте первым!

</div>

<div v-else class="comments-list">

<div v-for="comment in track.comments" :key="comment.id" class="comment-item">

<div class="comment-header">

<div class="author-avatar">

{{ comment.author.username.charAt(0).toUpperCase() }}

</div>

<div class="author-info">

<span class="author-name">{{ comment.author.username }}</span>

<span class="comment-date">{{ formatDate(comment.created\_at) }}</span>

</div>

</div>

<div class="comment-content">

<p>

{{ comment.content }}

</p>

</div>

</div>

</div>

</div>

</main>

<!-- Модальное окно для изображения -->

<div v-if="selectedImage" class="image-modal" @click.self="selectedImage = null">

<div class="modal-content">

<img :src="'data:image/png;base64,' + selectedImage" alt="Full size track image">

<button class="close-btn" @click="selectedImage = null">×</button>

</div>

</div>

</div>

</template>

<script setup>

import { ref, onMounted } from 'vue'

import { useRoute } from 'vue-router'

import { useAuthStore } from '@/stores/auth'

import api from '@/services/api'

import AuthHeader from '@/components/AuthHeader.vue'

import MapView from '@/components/MapView.vue'

const route = useRoute()

const authStore = useAuthStore()

const track = ref(null)

const loading = ref(true)

const error = ref(null)

const newComment = ref('')

const selectedImage = ref(null)

const loadingRiskMap = ref(false)

const riskMapError = ref(null)

onMounted(() => {

fetchTrackData()

})

const fetchTrackData = async () => {

try {

loading.value = true

error.value = null

const response = await api.getTrackDetails(route.params.id)

track.value = response.data

} catch (err) {

error.value = 'Ошибка загрузки данных о треке'

console.error(err)

} finally {

loading.value = false

}

}

const openFireRiskMap = async () => {

try {

loadingRiskMap.value = true

riskMapError.value = null

const trackId = parseInt(route.params.id)

if (!trackId || isNaN(trackId)) {

riskMapError.value = 'Неверный идентификатор трека'

return

}

await api.getFireRiskMap(trackId)

// Получаем URL для отдельной страницы с картой рисков

const url = `http://127.0.0.1:8000/tracks/${trackId}/fire\_risk`

// Открываем в новом окне

const newWindow = window.open(url, '\_blank', 'width=1200,height=800')

// Фокус на новое окно

if (newWindow) {

newWindow.focus()

} else {

throw new Error('Не удалось открыть новое окно. Возможно, ваш браузер блокирует всплывающие окна.')

}

} catch (err) {

riskMapError.value = 'Ошибка открытия карты: ' +

(err.response?.data?.detail ||

err.response?.data?.message ||

err.message ||

'Неизвестная ошибка')

} finally {

loadingRiskMap.value = false

}

}

const formatDate = (dateString) => {

return new Date(dateString).toLocaleDateString('ru-RU', {

year: 'numeric',

month: 'long',

day: 'numeric',

hour: '2-digit',

minute: '2-digit'

})

}

const addComment = async () => {

if (!newComment.value.trim()) return

try {

const response = await api.createComment(

track.value.id,

{ content: newComment.value }

)

track.value.comments.push({

...response.data,

author: {

id: authStore.user.id,

username: authStore.user.username

}

})

newComment.value = ''

} catch (error) {

console.error('Ошибка добавления комментария:', error)

}

}

</script>

<style scoped>

.track-detail-view {

min-height: 100vh;

display: flex;

flex-direction: column;

}

.detail-container {

display: flex;

flex-direction: column;

gap: 2rem;

padding: 2rem;

max-width: 1600px;

margin: 0 auto;

width: 100%;

}

.top-section {

display: grid;

grid-template-columns: 2fr 1fr;

gap: 2rem;

margin-bottom: 2rem;

}

.map-section {

height: 100%;

background: #f8f9fa;

border-radius: 8px;

overflow: hidden;

box-shadow: 0 4px 12px rgba(0,0,0,0.1);

}

.info-section {

display: flex;

flex-direction: column;

gap: 2rem;

}

.comments-section {

background: white;

padding: 1.5rem;

border-radius: 8px;

box-shadow: 0 4px 12px rgba(0,0,0,0.05);

width: 100%;

}

/\* Остальные стили остаются без изменений \*/

.track-info {

background: white;

padding: 1.5rem;

border-radius: 8px;

box-shadow: 0 4px 12px rgba(0,0,0,0.05);

}

.track-title {

font-size: 1.8rem;

margin: 0 0 1rem 0;

word-break: break-word;

overflow-wrap: break-word;

line-height: 1.3;

}

.scrollable-description {

max-height: 100px;

overflow-y: auto;

margin: 1rem 0;

padding: 0.5rem;

background: #f9f9f9;

border-radius: 4px;

border: 1px solid #eee;

}

.scrollable-description p {

line-height: 1.6;

color: #444;

word-break: break-word;

overflow-wrap: break-word;

white-space: pre-line;

margin: 0;

}

.track-meta {

display: grid;

grid-template-columns: repeat(2, 1fr);

gap: 1rem;

margin-top: 1.5rem;

}

.meta-item {

display: flex;

flex-direction: column;

}

.meta-label {

font-size: 0.9rem;

color: #666;

}

.meta-value {

font-weight: 500;

margin-top: 0.25rem;

}

.difficulty-indicator {

display: flex;

gap: 4px;

}

.difficulty-indicator span {

display: inline-block;

width: 12px;

height: 12px;

border-radius: 50%;

background-color: #e0e0e0;

}

.difficulty-indicator span.active {

background-color: #42b983;

}

.difficulty-card.difficulty-1 .difficulty-indicator span.active { background-color: #4caf50; }

.difficulty-card.difficulty-2 .difficulty-indicator span.active { background-color: #8bc34a; }

.difficulty-card.difficulty-3 .difficulty-indicator span.active { background-color: #ffc107; }

.difficulty-card.difficulty-4 .difficulty-indicator span.active { background-color: #ff9800; }

.difficulty-card.difficulty-5 .difficulty-indicator span.active { background-color: #f44336; }

.track-images {

margin-top: 2rem;

}

.image-grid {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 1rem;

margin-top: 1rem;

}

.image-item {

cursor: pointer;

border-radius: 4px;

overflow: hidden;

transition: transform 0.3s ease;

}

.image-item:hover {

transform: scale(1.05);

}

.image-item img {

width: 100%;

height: 100px;

object-fit: cover;

}

.forecast-section {

background: white;

padding: 1.5rem;

border-radius: 8px;

box-shadow: 0 4px 12px rgba(0,0,0,0.05);

}

.forecast-form {

display: flex;

gap: 1rem;

margin-top: 1rem;

align-items: flex-end;

}

.form-group {

flex: 1;

}

.form-group label {

display: block;

margin-bottom: 0.5rem;

font-size: 0.9rem;

color: #555;

}

.form-group input {

width: 100%;

padding: 0.75rem;

border: 1px solid #ddd;

border-radius: 4px;

}

.forecast-result {

margin-top: 1.5rem;

padding: 1rem;

background: #fff8e1;

border-radius: 4px;

border-left: 4px solid #ffc107;

}

.danger-level {

font-weight: bold;

color: #f44336;

}

.comment-form {

margin-top: 1rem;

display: flex;

flex-direction: column;

gap: 1rem;

}

.comment-form textarea {

width: 100%;

padding: 1rem;

border: 1px solid #ddd;

border-radius: 4px;

resize: vertical;

min-height: 80px;

}

.auth-notice {

margin-top: 1rem;

padding: 1rem;

background: #e3f2fd;

border-radius: 4px;

text-align: center;

}

.comments-list {

margin-top: 1.5rem;

}

.comment-item {

padding: 1rem 0;

border-bottom: 1px solid #eee;

}

.comment-item:last-child {

border-bottom: none;

}

.comment-header {

display: flex;

align-items: center;

gap: 0.75rem;

margin-bottom: 0.75rem;

}

.author-avatar {

width: 36px;

height: 36px;

border-radius: 50%;

background: #42b983;

color: white;

display: flex;

align-items: center;

justify-content: center;

font-weight: bold;

}

.author-info {

display: flex;

flex-direction: column;

}

.author-name {

font-weight: 500;

}

.comment-date {

font-size: 0.8rem;

color: #888;

}

.comment-content {

line-height: 1.5;

color: #333;

}

.comment-content p {

word-break: break-word;

word-wrap: break-word;

overflow-wrap: break-word;

}

.no-comments {

margin-top: 1rem;

padding: 1rem;

background: #f5f5f5;

border-radius: 4px;

text-align: center;

color: #666;

}

.image-modal {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0,0,0,0.8);

display: flex;

align-items: center;

justify-content: center;

z-index: 1000;

}

.modal-content {

position: relative;

max-width: 90%;

max-height: 90%;

}

.modal-content img {

max-width: 100%;

max-height: 80vh;

border-radius: 8px;

}

.close-btn {

position: absolute;

top: -40px;

right: 0;

background: none;

border: none;

color: white;

font-size: 2rem;

cursor: pointer;

}

.loading-container, .error-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 70vh;

gap: 1rem;

}

.loader {

border: 5px solid #f3f3f3;

border-top: 5px solid #42b983;

border-radius: 50%;

width: 50px;

height: 50px;

animation: spin 1s linear infinite;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

.btn {

padding: 0.75rem 1.5rem;

background-color: #42b983;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 1rem;

transition: background-color 0.3s;

}

.btn:hover {

background-color: #359b6d;

}

.btn:disabled {

background-color: #cccccc;

cursor: not-allowed;

}

/\* Стили для контейнера карты \*/

.map-section {

position: relative;

height: 500px;

border-radius: 8px;

overflow: hidden;

box-shadow: 0 4px 12px rgba(0,0,0,0.1);

}

/\* Добавим в конец стилей \*/

.risk-map-container {

height: 100%;

width: 100%;

position: relative;

z-index: 10;

}

/\* Фиксируем легенду в нужном месте \*/

.risk-map-container .folium-map .leaflet-control {

position: absolute;

bottom: 50px;

left: 50px;

background: white;

padding: 10px;

border: 2px solid grey;

z-index: 1000;

}

/\* Исправляем размеры карты \*/

.risk-map-container .folium-map {

height: 100% !important;

width: 100% !important;

}

/\* Убираем лишние отступы \*/

.risk-map-container .folium-map .leaflet-container {

height: 100% !important;

width: 100% !important;

min-height: 500px;

}

/\* Кнопка переключения \*/

.map-controls {

position: absolute;

top: 10px;

left: 50px;

z-index: 1000;

}

.switch-map-btn {

background-color: rgba(255, 255, 255, 0.8);

color: #333;

border: 1px solid #ddd;

padding: 8px 12px;

border-radius: 4px;

cursor: pointer;

font-size: 14px;

display: flex;

align-items: center;

gap: 5px;

}

.switch-map-btn:hover {

background-color: white;

}

.arrow-icon {

font-weight: bold;

}

/\* Дополнительные стили для карты рисков \*/

:deep(.leaflet-container) {

height: 100%;

background-color: #f8f9fa;

}

.forecast-form {

margin-top: 1rem;

}

</style>

Файл TrackUploadView.vue

<template>

<div class="track-load-view">

<AuthHeader />

<main>

<TrackLoad />

</main>

</div>

</template>

<script setup>

import AuthHeader from '@/components/AuthHeader.vue'

import TrackLoad from '@/components/TrackUpload.vue'

</script>

<style scoped>

</style>

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Потопахин В. Искусство алгоритмизации. [Текст] / В. Потопахин. – Москва: ДМК Пресс, 2018. – 320 с.
2. Семакин И. Основы алгоритмизации и программирования. Учебник [Текст] / И. Семакин, А. Шестаков. – 2-е изд., Москва: Академия, 2018. – 304 с.
3. Скиена С. Алгоритмы. Руководство по разработке [Текст] / С. Скиена 2-е изд. – СПб. БХВ-Петербург, 2016. – 720 с.
4. Ruseller [Электронный ресурс] / Введение в хранимые процедуры MySQL 5. – Режим доступа https://ruseller.com/lessons.php?id=1189, свободный
5. Sql-tutorial [Электронный ресурс] / SQL Задачи и решения. – Режим доступа http://www.sql-tutorial.ru/ru/content.html, свободный
6. W3schools [Электронный ресурс] / SQL Tutorial. – Режим доступа https://www.w3schools.com/sql/default.asp, свободный
7. Webmasterwiki [Электронный ресурс] / Базы данных: Основные понятия. – Режим доступа https://www.webmasterwiki.ru/MySQL, свободный
8. Бизли Д. Python. Краткий справочник: полное руководство по языку [Текст] / Д. Бизли – 2-е изд., СПб.: Питер, 2021. – 352 с.
9. Рамальо А. Изучаем FastAPI. Создание API на Python [Текст] / А. Рамальо – Москва: ДМК Пресс, 2022. – 278 с.
10. Любанович М. Изучаем Python. Том 1, Том 2 [Текст] / М. Любанович – 3-е изд., СПб.: Питер, 2020. – 448 с. и 496 с.
11. Флэнаган Д. JavaScript. Подробное руководство [Текст] / Д. Флэнаган – 7-е изд., Киев: Диалектика, 2020. – 816 с. (включает информацию о взаимодействии с HTML/CSS)
12. Николенко С. Современный HTML & CSS. Полный курс [Текст] / С. Николенко – СПб.: Питер, 2021. – 480 с.
13. Vue.js Official Guide [Электронный ресурс] / Документация Vue.js – Режим доступа: https://vuejs.org/guide/introduction.html , свободный
14. FastAPI Official Docs [Электронный ресурс] / Документация FastAPI – Режим доступа: https://fastapi.tiangolo.com/ , свободный/