Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Разработка веб-платформы для молодежного культурно-образовательного центра «5 этаж»»

по дисциплине «Проектный практикум»

Заказчик: Закирова Ирина Линовна _____

Куратор: Закирова Ирина Линовна
Магистр педагогики, Исполнительный директор
Свердловской областной общественной организации
«Уральский клуб нового образования» _____

Студенты команды: W4C&D

Гагельганц Антон Владимирович _____

Гатаулина Алина Константиновна _____

Гошуренко Юрий Юрьевич _____

Попов Андрей Вячеславович _____

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном цифровом мире, где молодежь привыкла получать информацию и услуги онлайн, организация культурно-образовательных мероприятий по-прежнему остается сложным и неудобным процессом. Наш проект направлен на решение этой проблемы путем создания специализированной веб-платформы, которая станет связующим звеном между культурным центром и его аудиторией.

Цель нашего проекта, разработать веб-платформу для молодежного культурно-образовательного центра, которая позволит организациям создавать и управлять экскурсиями, а пользователям — записываться на них.

Платформа должна включать:

– личные кабинеты организаций с возможностью: создания и управления экскурсиями, формирования расписаний для экскурсий, просмотра списка записавшихся пользователей, редактирования информации об экскурсиях;

– личные кабинеты пользователей с возможностью: просмотра доступных экскурсий и их расписаний, записи на экскурсии, просмотра истории записей и предстоящих событий;

– функционал платформы содержит: удобный интерфейс для создания и редактирования экскурсий и возможность гибкой настройки расписания экскурсий.

Актуальность проекта обусловлена растущим спросом на цифровизацию культурной сферы и необходимостью адаптировать ее под потребности современной молодежи. Внедрение платформы не только упростит процесс организации экскурсий, но и повысит их доступность, что в будущем будет способствовать развитию культурно-образовательной среды и увеличению вовлеченности молодого поколения. Проект представляет собой важный шаг на пути создания комфортной цифровой экосистемы для взаимодействия культурных центров с их аудиторией. Он сочетает в себе

практическую пользу для организаторов мероприятий и удобство для пользователей.

Для достижения поставленной цели необходимо решить следующие ключевые задачи:

1) разработать систему личных кабинетов для организаторов с полным функционалом управления экскурсиями:

– реализовать модуль создания и редактирования экскурсий;

– разработать инструменты формирования гибкого расписания;

– создать систему управления списками участников;

– реализовать возможность редактирования информации о мероприятиях.

2) спроектировать и внедрить пользовательские личные кабинеты:

– разработать каталог доступных экскурсий с системой фильтрации;

– реализовать удобный интерфейс записи на события;

– разработать систему уведомлений и оповещений;

– создать раздел истории посещений и предстоящих событий.

3) обеспечить интеграцию всех компонентов системы:

– наладить взаимодействие между кабинетами организаторов и пользователей;

– реализовать синхронизацию данных в реальном времени;

– обеспечить стабильную работу всех модулей платформы.

Решение этих задач позволит создать современную цифровую платформу, отвечающую потребностям как организаторов, так и пользователей платформы.

Разрабатываемая веб-платформа предназначена для цифровизации процесса организации и посещения культурно-образовательных экскурсий.

Платформа позволит создать единое цифровое пространство для взаимодействия культурного учреждения с его аудиторией, отвечая современным требованиям к удобству, доступности и эффективности.

По завершении проекта планируется достижение следующих ключевых результатов:

1) функциональная веб-платформа, обеспечивающая:

– полноценную автоматизацию процессов организации экскурсий;

– удобный механизм взаимодействия между организаторами и посетителями.

2) перспективы развития:

– создание мобильного приложения на базе платформы;

– интеграция с популярными календарными сервисами;

– развитие системы рекомендаций на основе интересов пользователей;

3) социально-культурный эффект:

– повышение доступности культурных мероприятий для молодежи;

– увеличение вовлеченности в культурно-образовательные программы;

1 Проделанная работа участниками команды

1.1 Backend-разработчик

В ходе работы был организован четкий процесс разработки backend-архитектуры, учитывающий потребности в масштабируемости и безопасности. В качестве технологического стека использовались Flask и SQLAlchemy, что обеспечивало целостность и согласованность компонентов.

Отдельное внимание было уделено построению системы управления данными. Определены логичные схемы взаимодействия сущностей и тщательно проработаны API-эндпоинты. Внедрённая JWT-аутентификация с ролевой моделью гарантирует защищённый доступ к ресурсам, разграничивая права пользователей, резидентов и администраторов.

Среди важных технических решений: оптимизированная структура базы данных PostgreSQL с продуманными табличными связями, RESTful API, поддерживающий все CRUD-операции и автоматическая генерация API-документации через Swagger.

На каждом этапе разработки проводилось тестирование и согласование с командой, что позволило создать эффективное backend-решение, отвечающее всем техническим требованиям. Такой подход обеспечивает оптимальное сочетание производительности, безопасности и функциональности, а также своевременную реализацию проекта.

1.2 Frontend-разработчик

Разработка frontend-части проекта потребовала тщательного проектирования пользовательских сценариев и создания адаптивного интерфейса для различных ролей - от участников до администраторов. Выбор Vue 3 с Composition API позволил организовать эффективное взаимодействие с данными, сохраняя код понятным и удобным для поддержки. Pinia

обеспечила централизованное управление состоянием, включая хранение токенов аутентификации, данных пользователей и списков экскурсий, что упростило синхронизацию данных между компонентами.

Особое внимание уделили безопасности: интеграция JWT в заголовки запросов и ролевая модель маршрутизации гарантировали доступ только авторизованным пользователям с соответствующими правами. Динамическая загрузка ключевых страниц оптимизировала производительность, а использование Naive UI ускорило разработку интерфейсов.

На всех этапах работы велась тесная координация с backend-разработчиком, что позволило создать согласованную систему обработки данных и предусмотреть возможные ошибки API. Реализованный механизм мгновенного обновления состояния в Pinia, обеспечивает оперативное отражение изменений в форме создания экскурсий. Такой подход гарантирует надежную интеграцию frontend с серверной частью, оптимальную производительность и соответствие техническим требованиям проекта.

1.3 Дизайнер

Разработана структурированная система проектирования интерфейсов, ориентированная на потребности молодёжной аудитории. Регулярные обсуждения с заказчиком позволяли своевременно вносить правки и корректировать концепцию. Вся работа велась в графическом редакторе Figma, что обеспечивало единый стиль всех элементов.

Основное внимание уделялось созданию интуитивно понятного интерфейса. Были разработаны логичные схемы навигации и продуманные пользовательские сценарии (приложение A, B, C, D, E, F, G).

Подобранная цветовая гамма с акцентом на оранжевые тона, отражает энергичность и креативность проекта. Основные элементы дизайна включают

типографику с четкими, хорошо читаемыми шрифтами с элементами графики для заголовков, а также сбалансированную композицию с выраженным зонированием и визуальными акцентами на ключевых разделах (события, новости, регистрация), что обеспечивает комфортное взаимодействие пользователей с интерфейсом.

Все этапы работы согласовывались с заказчиком, что позволило создать дизайн, полностью соответствующий поставленным задачам. Такой подход гарантирует достижение оптимального баланса между эстетикой и функциональностью.

1.4 Тимлид

Для эффективного управления разработкой был выстроен четкий процесс координации работы команды. Еженедельные рабочие созвоны позволяют синхронизировать текущие задачи и оперативно решать возникающие вопросы. Вся работа организована через доску задач в YouGile, где каждая задача имеет сроки и приоритеты выполнения (рисунок 1). Для наглядного планирования сроков и контроля прогресса была построена диаграмма Ганта (рисунок 2).

Взаимодействие с заказчиком осуществляется через регулярные еженедельные встречи, где обсуждается текущий прогресс и уточняются требования. Для оперативной коммуникации создан отдельный Telegram-чат, позволяющий быстро согласовывать срочные вопросы. Все поступающие от заказчика требования и правки фиксируются в YouGile, где они распределяются между членами команды.

Тимлид ежедневно контролирует выполнение задач, оперативно реагирует на возникающие проблемы и при необходимости корректирует сроки выполнения. Такой подход обеспечивает прозрачность рабочего процесса, позволяет своевременно вносить изменения в соответствии с

новыми требованиями и гарантирует достижение поставленных целей в установленные сроки.

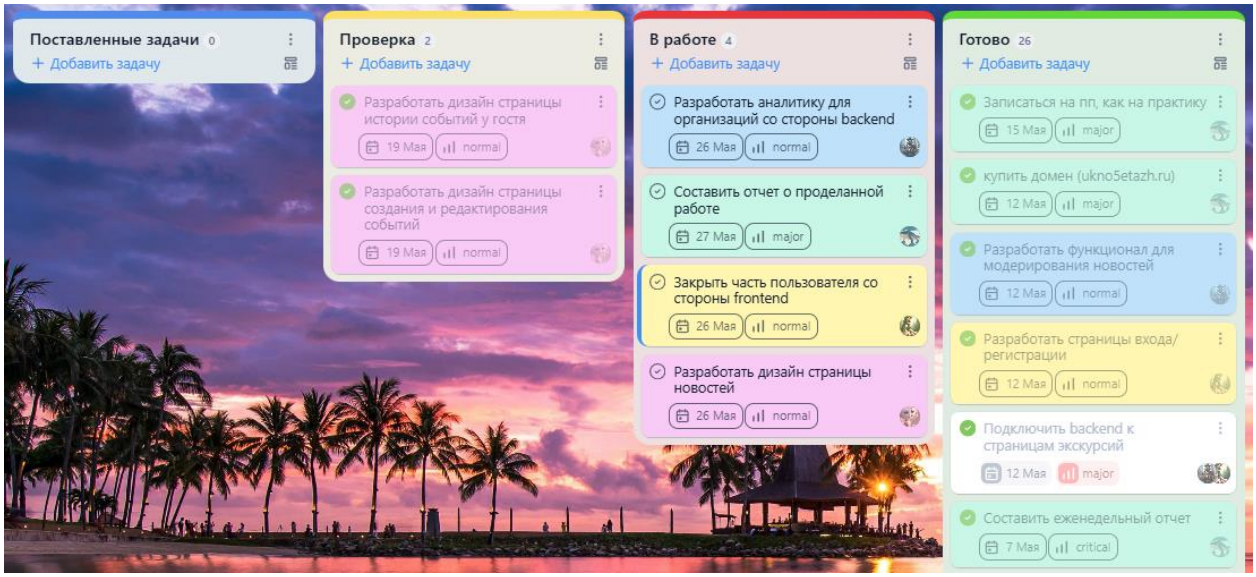


Рисунок 1 – Доска YouGile

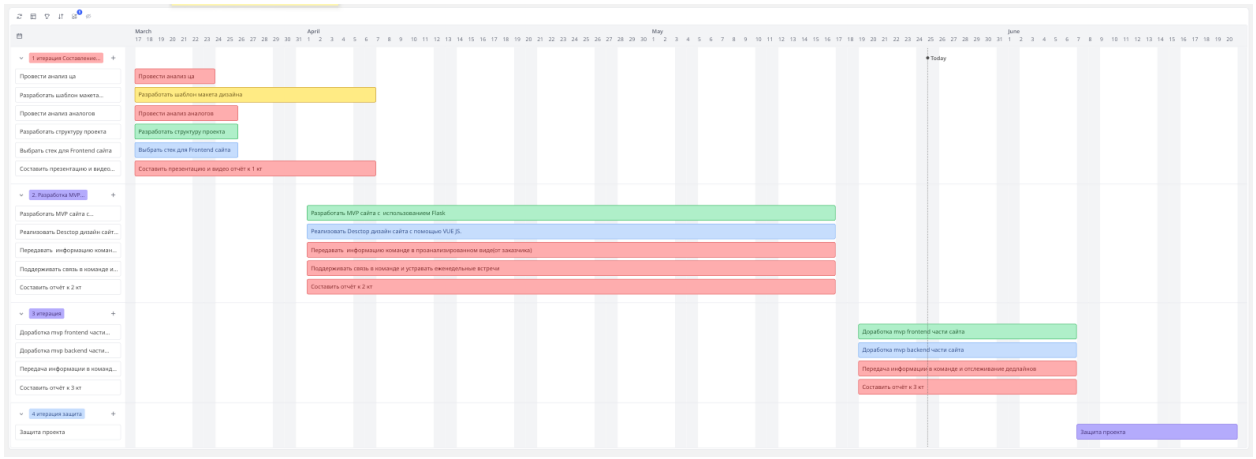


Рисунок 2 – Диаграмма Ганта

2 Разбор требований и формирование backlog

2.1 Анализ требований заказчика

Заказчик (молодежный культурно-образовательный центр) нуждается в веб-платформе, которая позволяет организациям:

1) создавать и редактировать экскурсии (название, описание, даты, лимит участников):

– управлять расписанием (гибкое добавление сеансов, повторяющиеся события);

– вести учет записавшихся.

2) предоставляет пользователям:

– удобный каталог экскурсий с фильтрами;

– простую запись через личный кабинет;

– возможность отмены/переноса записи.

3) критичный функционал:

– личные кабинеты для организаторов и пользователей;

– система бронирования с мгновенным подтверждением;

– список доступных экскурсий.

На данный момент уже разработана базовая версия платформы с личными кабинетами, системой бронирования и календарём экскурсий. Дальнейшее развитие системы будет осуществляться с учётом обратной связи первых пользователей и организаторов мероприятий.

3 Анализ и сопоставление аналогов разрабатываемого продукта

3.1 Анализ аналогов

Для понимания положения нашего проекта на рынке и выявления возможностей для его улучшения, важно провести анализ существующих аналогичных платформ. Этот анализ поможет определить сильные и слабые стороны конкурентов, а также выявить уникальные возможности для нашего продукта.

3.2 Креативный кластер «На Заводе»

Сильные стороны:

– возможность записи на экскурсии;

– удобство интерфейса платформы.

Слабые стороны:

– создание расписания для экскурсий;

– личные кабинеты пользователя;

– фильтр мероприятий;

– личные кабинеты резидентов.

3.3 Креативный кластер «Хлебозавод №6»

Сильные стороны:

– возможность записи на экскурсии;

– удобство интерфейса платформы.

Слабые стороны:

– создание расписания для экскурсий;

– личные кабинеты пользователя;

– фильтр мероприятий;

– личные кабинеты резидентов.

3.4 Сайт Рельсы

Сильные стороны:

– возможность записи на экскурсии;

– фильтр мероприятий;

– удобство интерфейса платформы.

Слабые стороны:

– создание расписания для экскурсий;

– личные кабинеты пользователя;

– личные кабинеты резидентов.

3.5 Сравнение и выводы

Анализ аналогов показывает, что на рынке существуют различные платформы для проведения экскурсий, каждая из которых имеет свои сильные и слабые стороны. Однако, многие из них не в полной мере удовлетворяют специфические потребности пользователей.

Наш сайт имеет возможность занять уникальную нишу, предложив специализированные функции для пользователей и организаторов.

4 Обзор архитектуры программного продукта

4.1 Общая архитектура

UKNO Backend реализован как RESTful веб-сервис на базе микрофреймворка Flask, обеспечивающий управление пользователями, экскурсиями, бронированиями и новостями. Взаимодействие с клиентскими приложениями происходит через HTTP REST API.

Для аутентификации и авторизации используется JSON Web Tokens (JWT), что позволяет строить масштабируемую и stateless систему безопасности. Доступ к ресурсам API контролируется с учётом ролей пользователей, что обеспечивает разграничение прав доступа.

UKNO Frontend реализован, как одностраничное приложение (SPA) на Vue 3 с использованием Composition API, обеспечивающее интерактивный пользовательский интерфейс для работы с экскурсиями, бронированиями и профилями пользователей.

Взаимодействие с backend происходит через REST API с применением JWT-аутентификации, что позволяет обеспечить безопасный и масштабируемый обмен данными.

4.2 Технологический стек

4.2.1 Backend часть

а) Flask — легковесный Python-фреймворк для создания веб-приложений;

б) Flask-Restx — расширение для Flask, упрощающее организацию REST API и автоматическую генерацию документации Swagger (OpenAPI);

в) JWT (JSON Web Tokens) — стандарт для stateless аутентификации и авторизации с токенами, содержащими информацию о правах пользователя;

г) SQLAlchemy — ORM для удобного и безопасного взаимодействия с реляционной базой данных.

д) Swagger (через Flask-Restx) — автоматическая генерация интерактивной документации для API, упрощающей разработку и тестирование клиентских приложений.

База данных реализована на PostgreSQL — надежной реляционной системе управления базами данных с поддержкой транзакций и сложных связей, обеспечивающей целостность данных и высокую производительность.

4.2.2 Frontend часть

а) VUE 3 — прогрессивный JavaScript-фреймворк для создания реактивных пользовательских интерфейсов с использованием Composition API;

б) Pinia — современное хранилище состояний (state management) для Vue 3, обеспечивающее централизованное управление данными и реактивность;

в) Vue Router — библиотека для навигации между страницами SPA с поддержкой динамических маршрутов и защиты доступа;

г) Naive UI — компонентная библиотека для быстрого построения интерфейсов с готовыми элементами (кнопки, формы, модальные окна).

д) Axios — HTTP-клиент для взаимодействия с бэкенд-API, поддерживающий перехватчики (interceptors) для автоматической обработки ошибок и JWT-аутентификации;

е) Swagger (через Flask-Restx) — автоматическая генерация интерактивной документации для API, упрощающей разработку и тестирование клиентских приложений.

4.3 Описание основных компонентов и связей между ними

4.3.1 Backend часть

Система построена на модульной архитектуре, где каждый компонент отвечает за конкретную функциональность, обеспечивая стабильную работу платформы.

Ключевые элементы:

а) REST API — предоставляет полный набор CRUD-операций для работы с основными сущностями: пользователями, ролями, экскурсиями, сессиями, бронированиями и новостями;

б) модель пользователей реализована с разграничением ролей: обычные пользователи, резиденты и администраторы, что обеспечивает дифференцированный доступ к функционалу платформы;

в) безопасная аутентификация и авторизация на основе JWT с проверкой ролей, обеспечивающая защищённый доступ к API;

г) реляционная база данных со связями и ограничениями целостности, оптимизированная для бизнес-логики продукта.

Такой подход обеспечивает масштабируемость, безопасность и удобство дальнейшего развития системы.

В структуре базы данных реализованы таблицы с чёткими связями для хранения информации (рисунок 3):

– пользователи и их роли;

– категории экскурсий, форматы и возрастные группы;

– основные данные об экскурсиях, сессиях и связанных с ними фотографиях;

– бронирования с возможностью отслеживания статусов;

– теги для классификации экскурсий;

– новостные статьи и их авторы.

Связи между таблицами гарантируют целостность данных и поддерживают реализацию бизнес-правил.

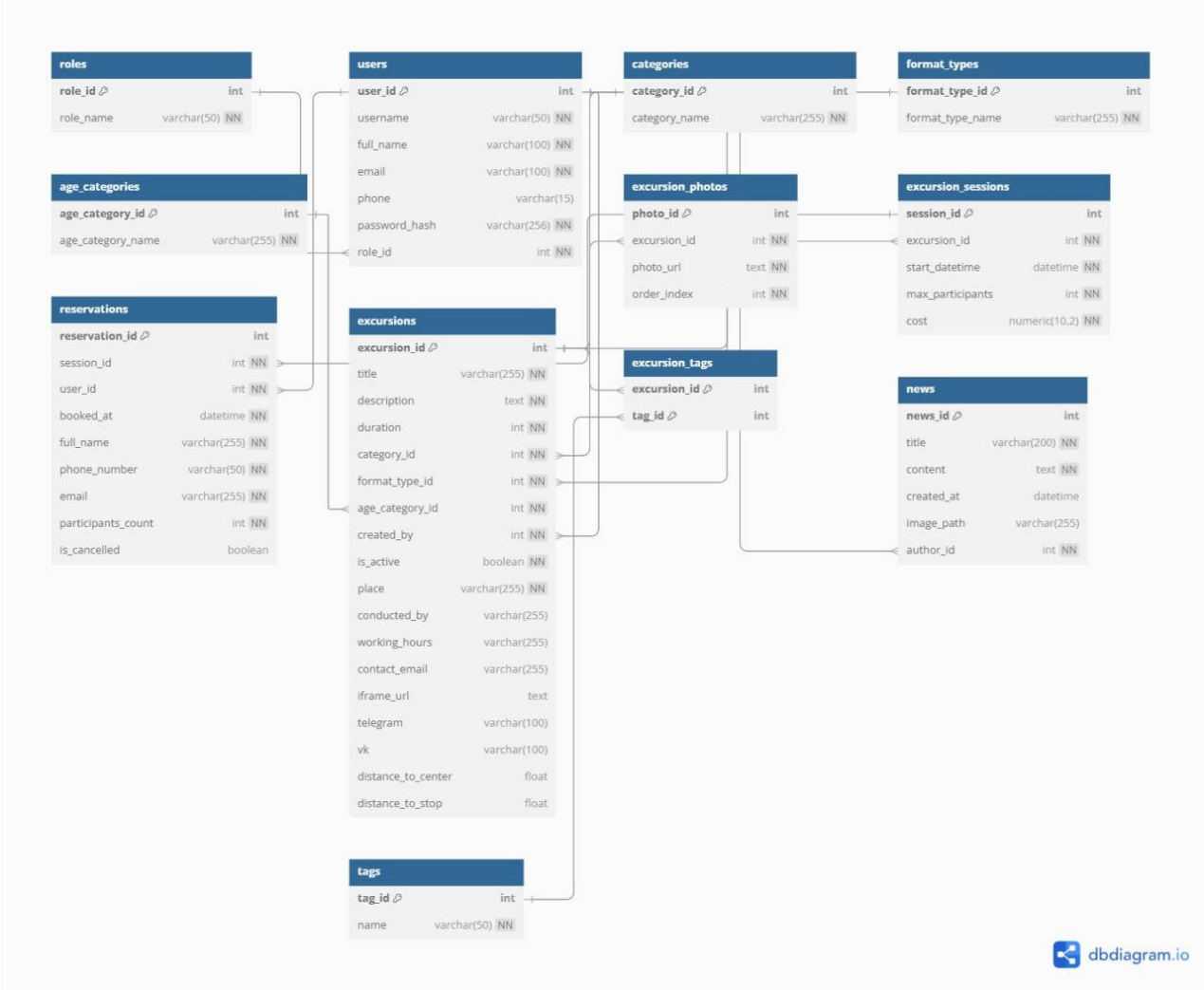


Рисунок 3 – База данных

4.3.1 Примеры backend-реализаций

1) Пример реализации аутентификации с использованием JWT:

Ниже представлен пример функции, которая выполняет аутентификацию пользователя с проверкой логина, пароля и роли, а также генерирует JWT токен.

```
def authenticate_user(username, password, required_role=None):
    user = get_user_by_username(username)
    if not user or not user.check_password(password):
        return None

    if required_role and user.role.role_name != required_role:
        return None

    return create_access_token(
        identity=user.username,
        additional_claims={"role": user.role.role_name}
    )
```

Описание используемых функций и параметров:

– get_user_by_username(username) — получение пользователя из базы данных по имени;

– check_password(password) — метод пользователя для проверки корректности пароля;

– required_role — необязательный параметр для ограничения доступа по роли пользователя;

– create_access_token — функция из библиотеки Flask-JWT-Extended, создающая JWT с идентификатором пользователя и ролью в дополнительных клеймах.

При каждом запросе к защищённым ресурсам роль и идентификатор пользователя извлекаются из токена, что исключает необходимость дополнительных обращений к базе данных и позволяет строить stateless систему.

2) Разграничение доступа по ролям — пример декоратора:

Для ограничения доступа к определённым API-эндпоинтам используется декоратор, проверяющий роль пользователя в JWT.

```
from functools import wraps
from flask_jwt_extended import verify_jwt_in_request, get_jwt
from http import HTTPStatus

def resident_required(fn):
    @wraps(fn)
    def wrapper(*args, **kwargs):
        verify_jwt_in_request()
        claims = get_jwt()
        if claims.get("role") != "resident":
            return {"message": "Доступ запрещён"},
HTTPStatus.FORBIDDEN
        return fn(*args, **kwargs)
    return wrapper
```

Принцип работы заключается в том, что сначала декоратор проверяет наличие и валидность JWT в запросе. Затем извлекает роль пользователя из дополнительных клеймов токена. После чего разрешает выполнение функции только если роль совпадает с «resident». В противном случае возвращает ошибку 403 Forbidden.

3) Пример добавления сессий экскурсии в базу данных:

Функция add_sessions показывает, как на уровне бизнес-логики создаются и сохраняются объекты сессий экскурсии с использованием SQLAlchemy ORM.

```
from datetime import datetime
from app import db
from models import ExcursionSession

def add_sessions(excursion, sessions):
    """
    Добавляет сессии к экскурсии.
```

```
    :param excursion: объект экскурсии
    :param sessions: список словарей с данными сессий (start_datetime, max_participants, cost)
    """
    for s in sessions:
        start_dt = datetime.fromisoformat(s["start_datetime"])
        session = ExcursionSession(
            excursion_id=excursion.excursion_id,
            start_datetime=start_dt,
            max_participants=s["max_participants"],
            cost=s["cost"]
        )
        db.session.add(session)
    db.session.commit()
```

Формат даты и времени — ISO 8601, что упрощает интеграцию с frontend. Все сессии добавляются в транзакцию и сохраняются одной операцией.

4.3.2 Frontend часть

В основе платформы лежит модульная архитектура с четким разделением функционала между компонентами, что обеспечивает стабильность системы.

Инструменты разработки:

а) Vite — современный сборщик проекта с горячей перезагрузкой (HMR) для ускоренной разработки;

б) ESLint и Prettier — инструменты линтинга и форматирования для поддержания единого стиля кода;

в) Vue DevTools — расширение для отладки состояния и производительности Vue-приложения.

Ключевые особенности:

а) модульная структура — приложение разделено на независимые модули (публичная лента событий, личный кабинет, панель резидента, админ-панель), что упрощает поддержку и развитие кода;

б) роутинг на основе ролей — доступ к разделам контролируется через Vue Router с проверкой прав пользователя (гость, пользователь, резидент, администратор);

в) централизованное состояние — управление данными (например, токеном авторизации, списком экскурсий) осуществляется через Pinia, что обеспечивает реактивность и предсказуемость состояния приложения;

г) Stateless-аутентификация — JWT-токен хранится в Pinia и LocalStorage, что позволяет сохранять сессию пользователя после перезагрузки страницы без необходимости постоянных запросов к серверу.

В структуре хранилища для управления глобальным состоянием используется Pinia Store, который обеспечивает:

– централизованное хранение данных (токен аутентификации, информация о пользователе, командах);

– инкапсуляцию логики работы с API;

– реактивное обновление интерфейса при изменении состояния;

– сохранение данных в LocalStorage для поддержки сессии после перезагрузки страницы.

4.3.3 Примеры frontend-реализаций

Защита маршрутов. Глобальный навигационный хук beforeEach проверяет:

– наличие токена для доступа к защищённым страницам;

– роль пользователя для административных разделов;

– редиректы для авторизованных пользователей (например, с /login на главную).

```
router.beforeEach(async (to, from, next) => {
  const store = useDataStore();
  const isAuthenticated = !!store.token;

  // Проверка авторизации
  if (to.meta.requiresAuth && !isAuthenticated) {
    next('/login'); // Редирект на логин
    return;
  }
  // Блокировка доступа к логину/регистрации для авторизованных
  if ((to.name === 'Login' || to.name === 'Register') && isAuthen-
ticated) {
    next('/');
    return;
  }
  // Проверка ролей
  if (to.meta.requiredRole && store.userInfo?.role !== to.meta.re-
quiredRole) {
next('/');
return;
  }
  next(); // Разрешить переход
});
```

Родительский компонент.

```
<template>
  <div class="page-wrapper">
    <div class="feed-wrapper">
        <h3>Подбери <span class="orange">событие</span> на свой
вкус</h3>
```

```
      <div class="feed-actions">
        <n-input
          v-model:value="searchQuery"
          placeholder="Найти событие"
          round
          clearable
        >
          <template #prefix>
            <n-icon :component="SearchOutline" />
          </template>

<div class="feed">
        <div class="event-count">
            <p  v-if="excursions.length">{{  excursions.length  }}
предложения</p>
        </div>
        <div class="events">
          <EventCard
              v-for="(excursion, i) in excursions['excursions']"
//генерация карточек событий
            :key="i"
            :excursion="excursion" // передаём в дочерний элемент
данные
          />
        </div>
      </div>
<script setup>
import  EventCard  from  './components/event-card.vue';  //импорт
дочернего элемента
import  {  useDataStore  }  from  '@/stores/counter';  //  импорт
хранилища
const store = useDataStore();
const searchQuery = ref('');
const  excursions  =  computed(()  =>  store.getExcursions);  //
отслеживание данных в store
```

```
onBeforeMount(async () => {
// Начало запросы на получение данных до загрузки
  try {
    await store.FetchExcursions();
  } catch (error) {
    console.error('Ошибка при загрузке экскурсий:', error);
  }
});
</script>
<style scoped>
.feed-wrapper {
  display: flex;
  flex-direction: column;
  max-width: 1800px;
  position: relative;
}
```

Дочерний элемент.

```
<template>
  <div class="card-wrapper" v-if="excursion"> // проверка на то,
что данные уже получены и можно отображать элемент
    <div class="preview-img">
      <img
        :src="getMainImage"
        :alt="excursion.title"
        @error="handleImageError"
        class="event-image"
      >
    </div>
    <div class="content">
      <div class="title">
          <h5>{{ excursion.title }}</h5> // вставляем полученные
данные в элемент
      </div>
```

```
      <div class="descript">
        <p>{{ excursion.description }}</p>
        <p v-if="nearestSession">
           {{ formattedDate }} | {{ formattedTime }} | {{ excur-
sion.category.category_name }}
        </p>
        <p v-else>
          Нет запланированных сеансов
        </p>
      </div>

  <div v-else class="loading"> // если данные ещё загружаются
    Загрузка данных...
  </div>
</template>
<script setup>
import { computed } from 'vue';
import { useRouter } from 'vue-router';
const props = defineProps({ // полученные от родителя данные
  excursion: {
    type: Object,
    required: true
  }
});
const formattedDate = computed(() => { // форматирование данных
  if (!nearestSession.value) return '';
  const date = new Date(nearestSession.value.start_datetime);
  return date.toLocaleDateString('ru-RU', {
    day: 'numeric',
    month: 'long',
    year: 'numeric'
  });
});
```

```
const handleImageError = (e) => { // обработка ошибки загрузки фото из полученных данных
  e.target.src = '/develop/2025-05-01 00.11.54 1.png';
};
</script>
<style scoped>
.card-wrapper{
  display: flex;
  flex-direction: column;
  justify-content: center;
  gap: 20px;
  padding: 20px;
  border: 1px solid black;
  border-radius: 15px;
  max-width: 400px;
}
```

4.4 Обоснование выбора архитектурного решения

Backend часть:

а) Flask — выбран за лёгкость и гибкость, идеально подходящих для REST API;

б) Stateless с JWT — обеспечивает безопасную аутентификацию без хранения сессий, упрощая масштабируемость;

в) REST API на Flask-Restx — стандартизирует эндпоинты и автоматизирует документацию через Swagger;

г) PostgreSQL — надёжная реляционная СУБД с поддержкой сложных связей между данными.

д) оптимизированная база данных — структура таблиц гарантирует целостность данных и быстрый доступ.

Frontend часть:

а) гибкость — Vue 3 и Composition API позволяют легко масштабировать логику компонентов;

б) производительность — Vite и динамические импорты ускоряют загрузку приложения;

в) безопасность — JWT и защищённые маршруты ограничивают доступ неавторизованных пользователей;

г) удобство разработки — Pinia и Naive UI сокращают время на реализацию сложных функций.

5 Описание методологии разработки

5.1 Ключевые особенности методологии

В рамках разработки проекта использовалась итеративная методология. Этот подход позволил оперативно адаптировать функционал под изменяющиеся требования заказчика и последовательно совершенствовать продукт на основе поступающей обратной связи.

Итеративный процесс — проект разбивался на этапы (итерации), каждая из которых включала анализ требований, реализацию функционала, тестирование и демонстрацию промежуточных результатов.

Постоянная доработка под заказчика — заказчик активно принимал участие на каждом этапе, предоставляя уточнения по функционалу и внешнему виду интерфейса, что позволило сделать продукт максимально соответствующим его ожиданиям.

5.2 Информация о процессе разработки

5.2.1 Проектирование системы

Разработка архитектуры на основе паттерна MVC, что позволило четко разделить логику данных, интерфейса и взаимодействий между ними.

5.2.2 Создание API

Реализованы эндпоинты для работы с пользователями, экскурсиями и записями на них. Flask API разрабатывался с учетом простоты интеграции с frontend (Vue.js).

5.2.3 Разработка интерфейса

Созданы компоненты Vue.js для выполнения пользовательских сценариев.

5.2.4 Интеграция и тестирование

Проводилось тестирование для проверки корректности взаимодействия между интерфейсом и серверной частью.

5.3 Отчет о результатах тестирования на промежуточных этапах

Тестирование проводилось вручную пользователями. Роль тестировщиков выполняли заказчик и конечные пользователи. Тесты включали проверку ключевых пользовательских сценариев:

– регистрация и вход в систему;

– редактирование экскурсии;

– просмотр и фильтрация результатов;

– сортировка результатов;

– добавление новостей;

– редактирование опубликованных новостей;

– управление резидентами;

– проверка работы уведомлений на почту.

5.4 Разбор выявленных ошибок

5.4.1 Ошибки в валидации данных

Описание: при отправке формы создания экскурсии в формате multipart/form-data с полем data (JSON) и фотофайлами, сервер не валидировал структуру JSON, что приводило к неконтролируемым исключениям.

Решение: добавлена предварительная проверка формата JSON с помощью json.loads(data) и генерация ответа с ошибкой 400 при неверной структуре

5.4.2 Неверная обработка ролей в JWT

Описание: некоторые защищённые эндпоинты неправильно проверяли роль пользователя — использовалось сравнение с claims['role'], что вызывало KeyError, если поле отсутствовало.

Решение: изменена проверка на безопасный метод claims.get("role"), добавлены тесты на авторизацию без роли.

5.4.3 Конфликт имён файлов при загрузке фотоэкскурсий

Описание: При загрузке нескольких файлов с одинаковыми именами (например, image.jpg) происходила перезапись ранее загруженных изображений, что приводило к потере данных.

Решение: Имя каждого файла теперь шифруется с использованием UUIDперед сохранением, чтобы обеспечить уникальность и предотвратить конфликты.

5.4.4 Неудаление фотографий при удалении экскурсии

Описание: при удалении экскурсии, связанные с ней фотографии, оставались в базе данных и на файловой системе, что приводило к "мусорным" данным и засорению хранилища.

Решение: в SQLAlchemy-модели связи между экскурсиями и фотографиями было добавлено каскадное удаление, и при удалении экскурсии добавлена логика удаления файлов с диска.

5.4.5 «Мерцание» данных при загрузке

Описание: при переходе между страницами сначала показывался пустой список, потом резко появлялись данные.

Решение: внедрили скелетоны загрузки.

5.4.6 Потерянный state при перезагрузке

Описание: при обновлении страницы данные в store (например, токен авторизации) сбрасывались.

Решение: добавили persist-плагин, который автоматически сохраняет нужные поля в localStorage и восстанавливает их при загрузке страницы.

6 Информация о планировании деятельности в ходе разработки

6.1 Планирование этапов разработки

6.1.1 Начальный этап (Анализ и проектирование)

Для планирования деятельности команды использовался **Scrum-подход**, который позволил структурировать процесс разработки и обеспечить четкое распределение задач. Работа была организована в виде спринтов, каждый из которых длился 1–2 недели. Начальный этап:

– формулировка требований совместно с заказчиком;

– разработка архитектурного решения на основе MVC;

– составление общего плана разработки, включая ключевые функциональные модули и сроки выполнения.

6.1.2 Разработка (Итеративный процесс)

Этапы разработки:

– создание API на Flask: реализация CRUD-операций, логики работы с базой данных и авторизации;

– разработка компонентов интерфейса на Vue.js;

– интеграция frontend и backend.

6.1.3 Тестирование и доработка

Этапы тестирования и доработки:

– проведение тестирования;

– исправление выявленных ошибок;

– доработка функционала на основе обратной связи от заказчика.

6.1.4 Финальный этап (Релиз)

Финальный этап:

– оптимизация и рефакторинг кода;

– развёртывание системы и подготовка документации.

6.2 Распределение задач между участниками команды разработчиков

6.2.1 Руководитель проекта

В задачи руководителя проекта входило:

– организация процесса разработки;

– установление сроков и контроль их соблюдения;

– взаимодействие с заказчиком: сбор и уточнение требований, обсуждение промежуточных результатов.

6.2.2 Backend-разработчик

В задачи backend-разработчика входило:

– реализация серверной части с использованием Flask;

– разработка API-эндпоинтов для работы с пользователями, экскурсиями и личными кабинетами.

– настройка базы данных и написание запросов;

– устранение проблем с сессиями и валидацией данных.

6.2.3 Frontend-разработчик

В задачи backend-разработчика входило:

– разработка пользовательского интерфейса на Vue.js;

– создание динамических компонентов и маршрутизации страниц;

– интеграция с API, отладка взаимодействия с сервером.

6.3 Примеры распределения задач

Создание экскурсий и запись на них: разработка API и бизнес-логики — backend-разработчик, проверка корректности отображения и взаимодействия — frontend-разработчик.

Регистрация и авторизация: реализация — backend-разработчик, проверка интерфейса — frontend-разработчик.

ЗАКЛЮЧЕНИЕ

В ходе разработки платформы был проделан значительный объем работы, направленный на создание современного веб-сервиса, решающего ключевые проблемы организации культурно-образовательных мероприятий. Сервис предоставляет культурным центрам современный инструментарий для управления экскурсиями, расписанием и участниками, а также дает возможность постоянно расширять перечень предлагаемых мероприятий. Для пользователей реализован удобный механизм поиска и записи на экскурсии через интуитивный интерфейс с системой фильтрации и уведомлений, а также мгновенным подтверждением. Основная цель проекта заключалась в создании удобного цифрового решения, объединяющего организаторов культурных мероприятий и их аудиторию, и на данный момент сервис готовится к тестовому запуску.

Общие выводы

Разработка нашего продукта продемонстрировала важность тщательного анализа требований пользователей и заказчика. Был проведен сбор и анализ данных, что позволило более точно определить функциональные и нефункциональные требования к платформе. В результате этого процесса были выделены и реализованы ключевые функции, которые обеспечат преимущество нашему проекту.

Положительные стороны разработанного решения

Удобный и современный интерфейс: наш сервис предлагает интуитивно понятный интерфейс, который позволяет пользователям легко находить интересующие экскурсии, просматривать подробную информацию и удобно записываться на события.

Надежная система бронирования: платформа автоматически проверяет доступность мест и предотвращает двойные бронирования, гарантируя корректность каждого заказа и отсутствие временных конфликтов.

Функционал для организаторов: административная панель предоставляет полный контроль над экскурсиями — от создания и редактирования мероприятий до управления участниками. Реализована возможность добавления новых администраторов и резидентов, масштабирования системы для работы с большим количеством экскурсий. Кодовая база оптимизирована для быстрой работы и простого расширения функционала.

Было произведено ручное тестирование сайта. Тесты включали проверку ключевых пользовательских сценариев:

– регистрация и вход в систему;

– редактирование экскурсии;

– просмотр и фильтрация результатов;

– сортировка результатов;

– добавление новостей;

– редактирование опубликованных новостей;

– управление резидентами;

– проверка работы уведомлений на почту.

По результатам пользовательского тестирования были выявлены несколько проблем, решение которых должно было балансировать между удобством для пользователя и требованиям заказчика. Постоянная обратная связь от конечных пользователей помогла находить баланс между этими аспектами.

После внесения всех доработок интерфейс стал более интуитивным, а функционал — стабильным. Система успешно прошла проверки на корректность ключевых пользовательских сценариев и готова к эксплуатации

Области для улучшения

К области для улучшения хочется отнести такие пункты как:

– добавление аналитики посещений и продажи билетов;

– внедрение платежной системы;

– реализация предварительного просмотра добавляемых событий;

– реализация плавных анимаций для кнопок, переходов и других интерактивных элементов.

На данный момент наш сайт готовится к запуску и тестированию на реальных пользователях. Ожидается, что тестирование позволит выявить оставшиеся проблемы и улучшить сервис, сделав его ещё более полезным и удобным для пользователей и заказчика. В дальнейшем проект продолжит дорабатываться в рамках пятого семестра, путем внедрения новых функций и улучшений на основе полученной обратной связи, чтобы максимально удовлетворить потребности заказчика и пользователей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Коберн А. Современные методы описания функциональных требований к системам / А. Коберн. – Москва: Издательство «Лори», 2012. – 264 с. – ISBN 978-5-85582-326-4.

2. Welcome to Flask-RESTX’s documentation! / [Электронный ресурс] // Flask-RESTX. — URL: https://flask-restx.readthedocs.io/en/latest/index.html (дата обращения: 10.03.2025).

3. Flask-SQLAlchemy 3.1.1 / [Электронный ресурс] // Flask-SQLAlchemy · PyPI. — URL: https://pypi.org/project/Flask-SQLAlchemy/ (дата обращения: 15.03.2025).

4. Flask-JWT-Extended’s Documentation / [Электронный ресурс] // Flask-JWT-Extended’s Documentation. — URL: https://flask-jwt-extended.readthedocs.io/en/stable/ (дата обращения: 15.03.2025).

5. Welcome to Flask — Flask Documentation / [Электронный ресурс] // Welcome to Flask — Flask Documentation. — URL: https://flask.palletsprojects.com/en/stable/ (дата обращения: 10.03.2025).

6. Vue. js - The Progressive JavaScript Framework / [Электронный ресурс] // Vue. js - The Progressive JavaScript Framework. — URL: https://vuejs.org/guide/introduction.html (дата обращения: 10.03.2025).

7. PostgreSQL Downloads / [Электронный ресурс] // Downloads - PostgreSQL. — URL: https://flask.palletsprojects.com/en/stable/ (дата обращения: 23.04.2025).

8. PostgrePro Что такое PostgreSQL? / PostgrePro. – 2020. — URL: https://postgrespro.ru/docs/postgresql/12/intro-whatis
(дата обращения: 20.03.2025).

9. JavaScript WebSocket / [Электронный ресурс] // JavaScript. – 2025. — URL: https://learn.javascript.ru/websocket (дата обращения: 02.03.2025).

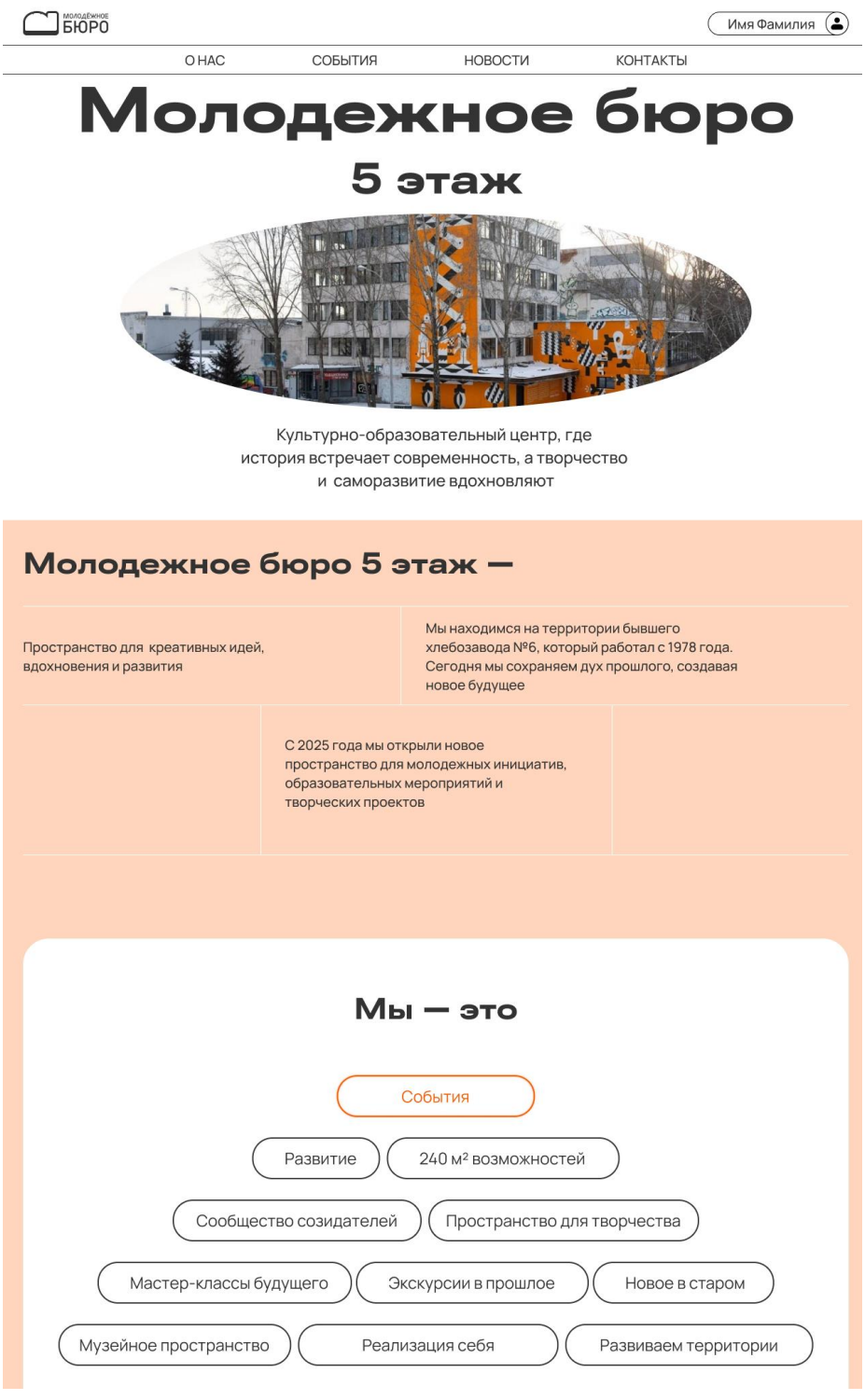
10. Proglib Об Agile / [Электронный ресурс] // Denver. – 2022. — URL: https://proglib.io/p/klyuchevye-razlichiya-mezhdu-agile-scrum-i-kanban-2022-02-

24 (дата обращения 7.03.2025).

ПРИЛОЖЕНИЕ А

(обязательное)

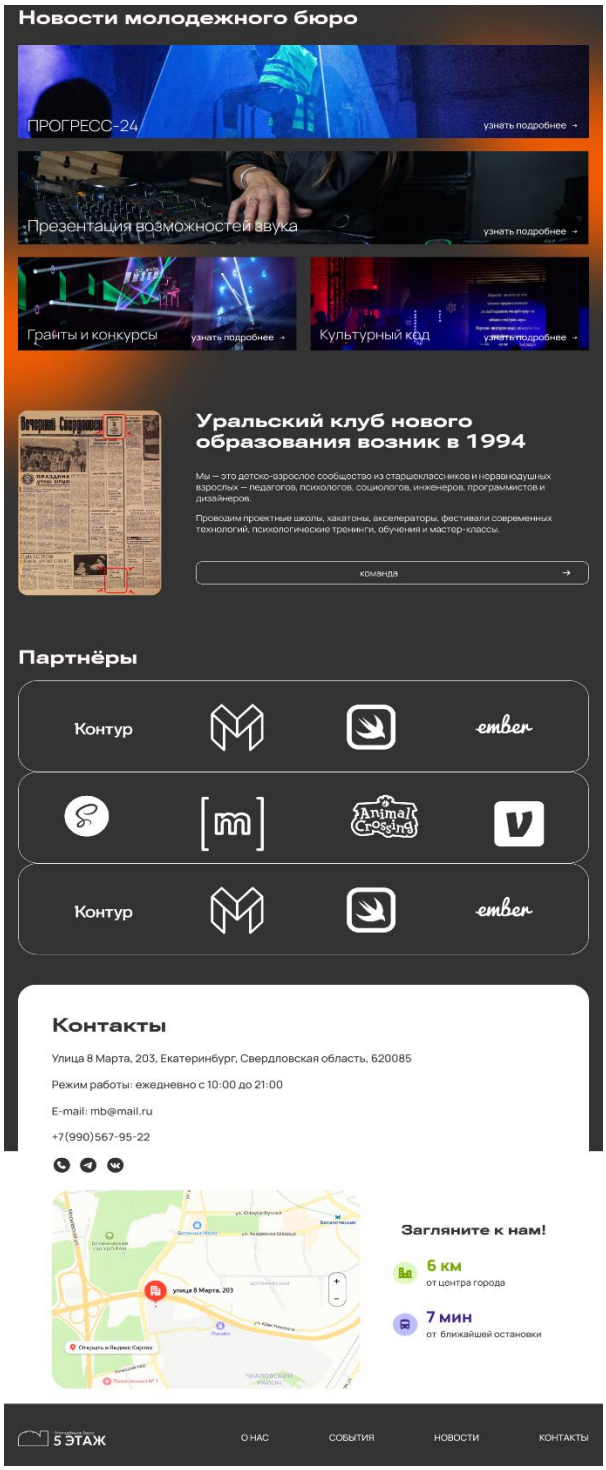
Макеты Figma



ПРИЛОЖЕНИЕ В

(обязательное)

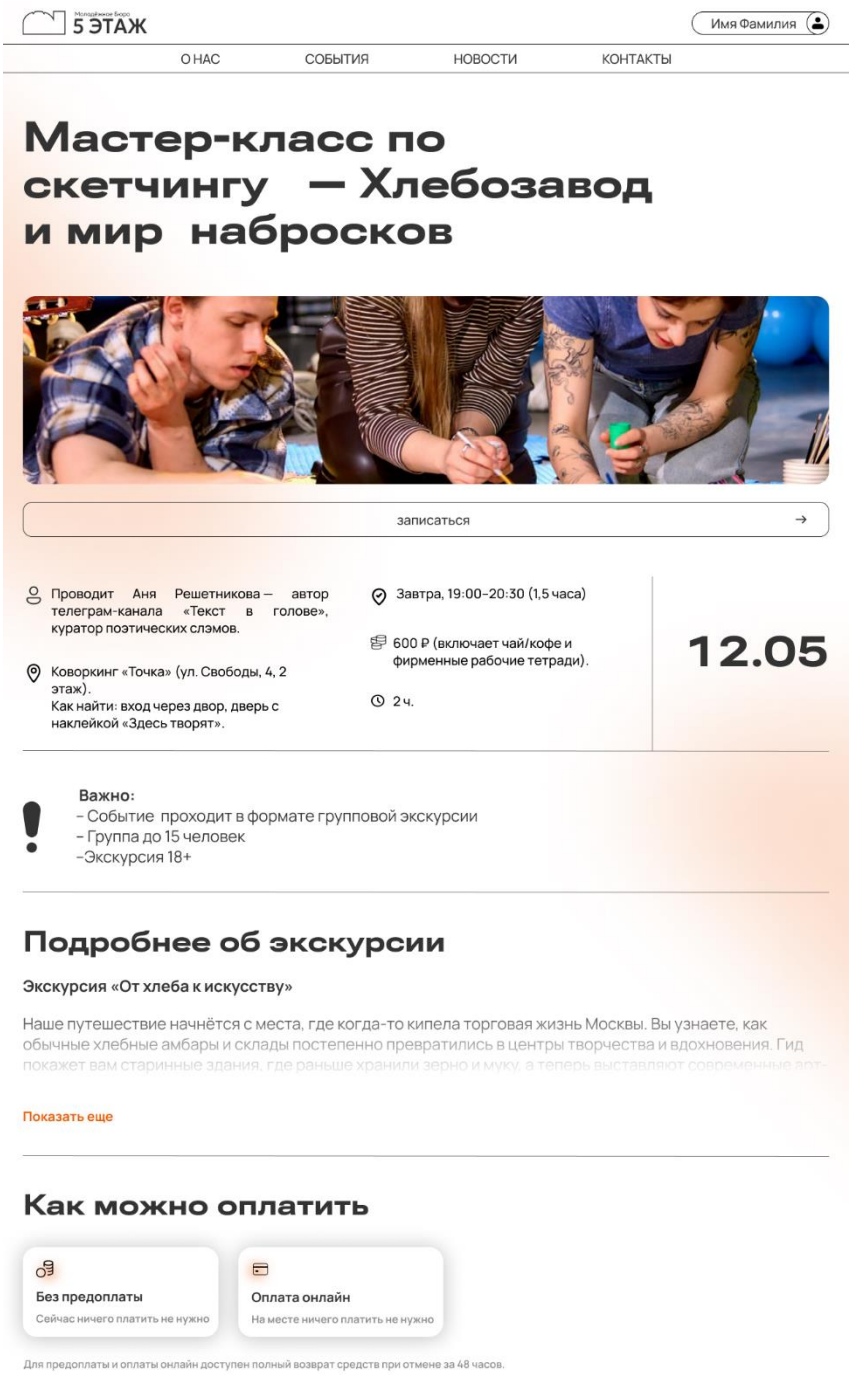
Макеты Figma



ПРИЛОЖЕНИЕ С

(обязательное)

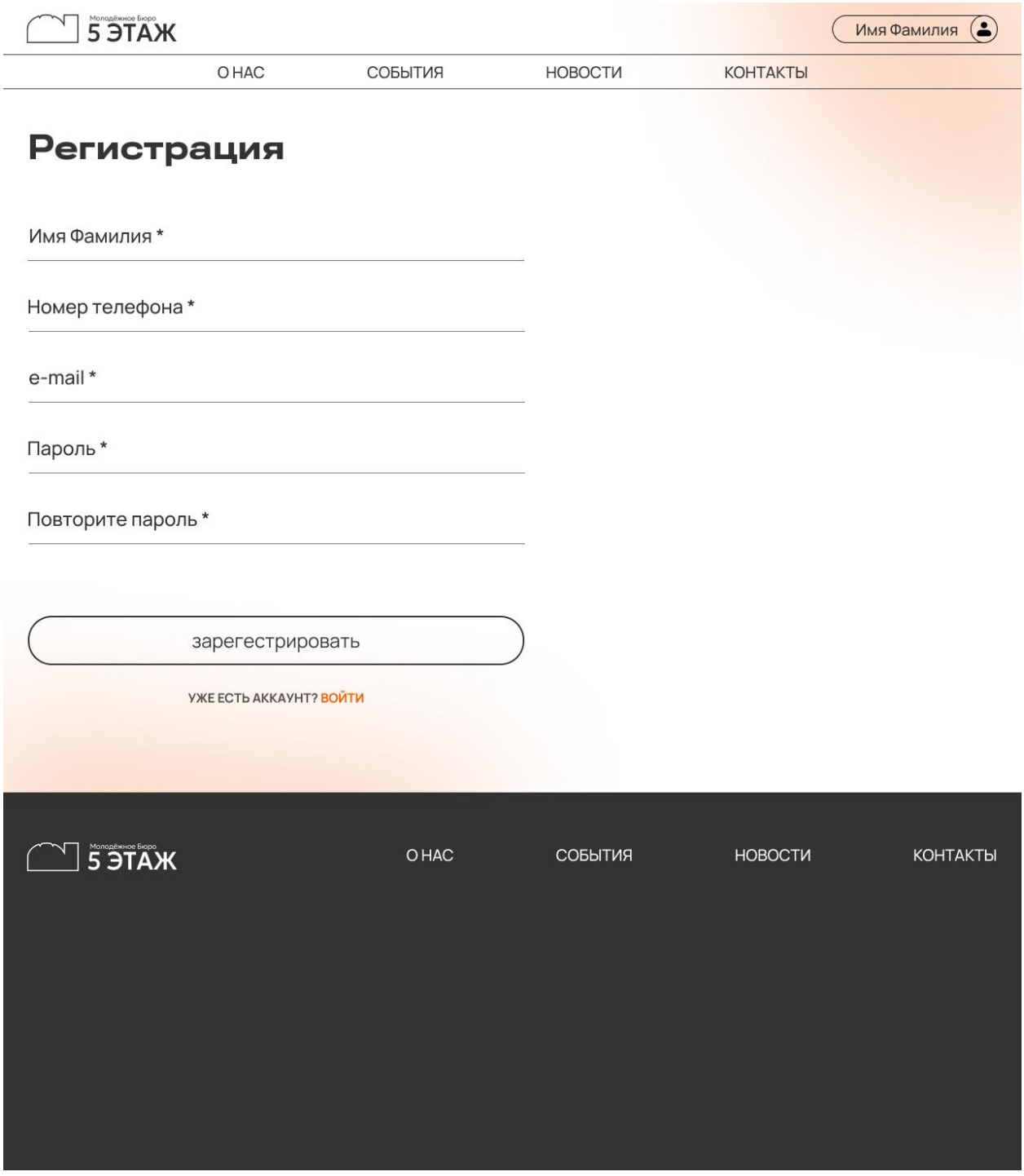
Макеты Figma



ПРИЛОЖЕНИЕ D

(обязательное)

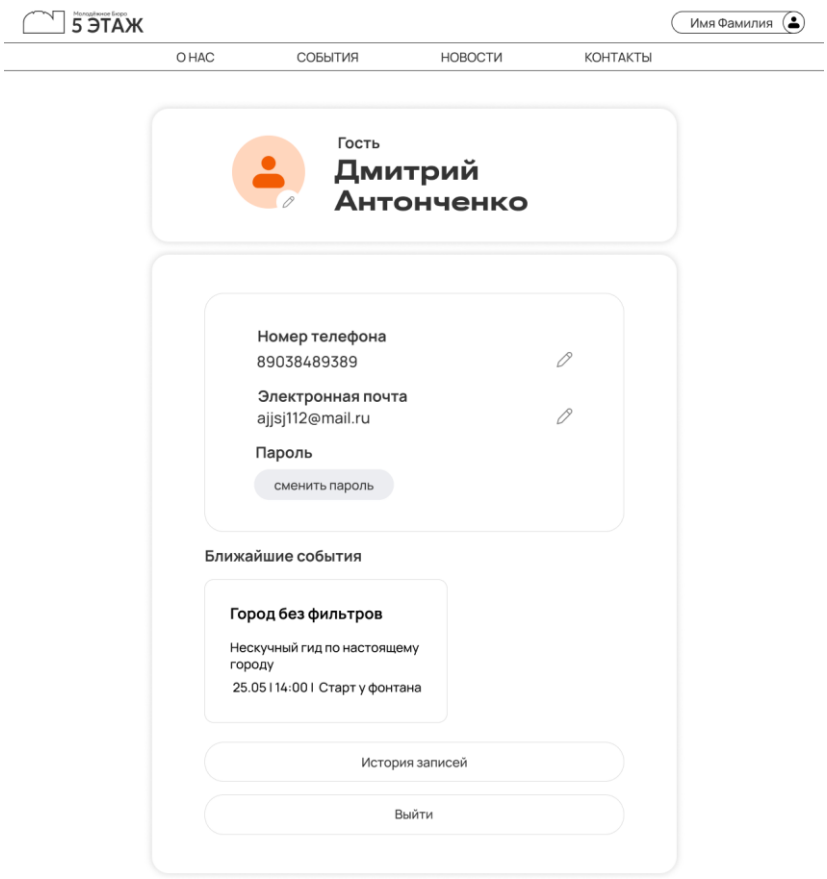
Макеты Figma



ПРИЛОЖЕНИЕ Е

(обязательное)

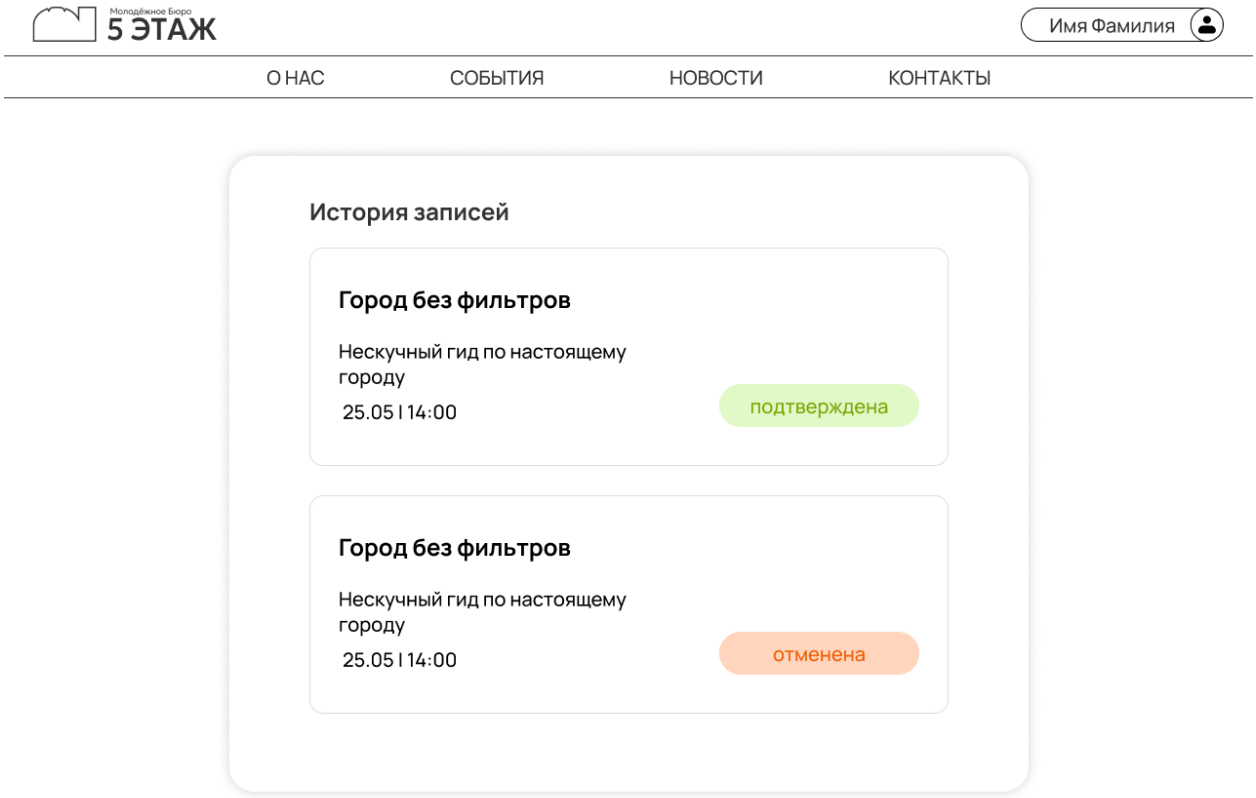
Макеты Figma

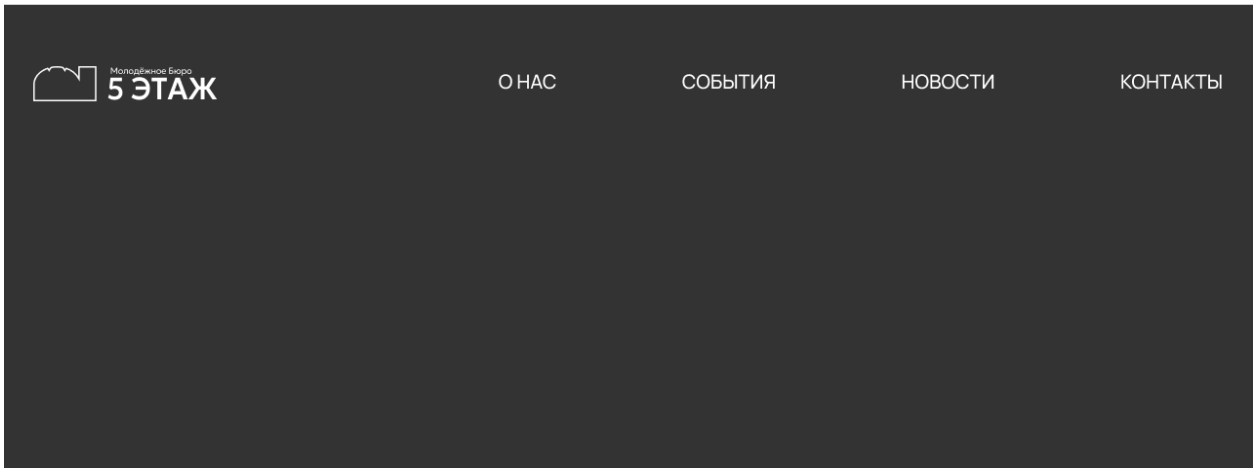


ПРИЛОЖЕНИЕ F

(обязательное)

Макеты Figma





ПРИЛОЖЕНИЕ G

(обязательное)

Макеты Figma