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
«Создание ПО Neo Lab»

по дисциплине «Проектный практикум»

Заказчик: Чернышов Юрий Юрьевич _____

Студенты команды:

Белослудцева М.Н. _____

Бирючев П.А. _____

Гущин К.А. _____

Рыбин И.А. _____

Савенко Д.Р. _____

Санаева Е.А _____

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Проект NEO Lab Manager представляет собой информационную систему, разработанную для управления образовательным процессом в области информационной безопасности. Целью данного проекта является создание комплексного решения, которое будет способствовать повышению эффективности образовательного процесса и формированию компетенций в сфере кибербезопасности среди будущих специалистов.

Основные задачи проекта:

1) Разработка платформы для организации доступа к образовательным материалам и ресурсам лаборатории по информационной безопасности;

2) Создание единой базы знаний в области ИБ, включающей литературу, учебные курсы, техническую документацию и другие релевантные ресурсы;

3) Реализация системы управления образовательным процессом, включая контроль знаний студентов, виртуальных лабораторных работ;

Обоснование актуальности проекта:

1) Рост угроз в сфере ИБ требует постоянного совершенствования подготовки специалистов;

2) Недостаточность современных образовательных решений в области кибербезопасности;

3) Необходимость создания интерактивных и практикоориентированных форматов обучения;

4) Потребность в комплексном подходе к образованию в ИБ, охватывающем как теоретические знания, так и практические навыки.

Область применения: Данный проект будет применяться в университете УрФУ на дисциплинах, связанных с кибербезопасностью и информационной безопасностью от UDV Group.

Ожидаемые результаты проекта:

1) Создание платформы NEO Lab;

2) Возможность создавать свои тесты и настраивать их;

3) Адаптивная верстка как на мобильных телефонах, так и на компьютерах.

Проект имеет важное значение для развития компетенций в сфере информационной безопасности и способствует формированию современного образовательного пространства в области кибербезопасности.

1 ОСНОВНАЯ ЧАСТЬ

1.1 Разбор требований заказчика

Данный проект является достаточно крупным и долгим по выполнению, поэтому мы решили разделить его по модулям:

1) Новости,

2) Витрина проектов,

3) Учебные материалы,

4) Тестирование,

5) Лабораторные работы.

Заказчик предоставил нам свои требования к продукту. Они представлены ниже:

В системе используются роли, за которыми зафиксированы соответствующие права

- «Обучающийся»: имеет возможность читать информационные материалы, выполнять лабораторные работы, проходить тестирование, видеть статистику по своим оценкам, участвовать в соревнованиях;
- «Преподаватель»: может добавлять и изменять информационные материалы в соответствующие курсы, обновлять новости, создавать и изменять лабораторные работы и тесты, проверять работы и выставлять оценки, создавать и изменять задачи для соревнований, управлять соревнованиями;
- «Администратор»: может добавлять и удалять пользователей, подтверждать регистрацию пользователя, имеющего статус «Обучающийся» или «Преподаватель», имеет доступ для мониторинга и управления оборудованием и программным обеспечением, задействованным в работе платформы.

Краткое описание отдельных пунктов меню главного графического интерфейса:

1) «Учебный процесс»: новостные страницы, описание доступных ресурсов, описание содержания доступных лабораторных работ и их принадлежность к курсам, расписание занятий и мероприятий, ссылки на доступные материалы, информация по учебным курсам, организационная информация;

2) «Учебные материалы»: информация об основных и дополнительных источниках информации для прохождения студентами образовательных программ, выполнения практических и оценочных заданий, сведения о дополнительных материалах полезных студентам для расширения знаний по трекам;

3) «Витрина проектов»: страница, на которой пользователи могут выкладывать свои проекты с лендингом, документацией и картинками, также могут просматривать чужие проекты, комментировать их, видеть статистику просмотров и их рейтинг;

4) «Лабораторные работы»:

a. база данных готовых образов и IaaC скриптов для развертывания инфраструктуры для лабораторных работ,

b. система визуализации для отображения инфраструктуры,

c. функциональный блок для взаимодействия с эмуляторами для прохождения лабораторных работ,

d. система мониторинга задействования ресурсов,

e. графический интерфейс для запуска и остановки лабораторной работы,

f. проверка работоспособности инфраструктуры в ходе выполнения работы,

g. Создание лабораторных работ:

- создание необходимых виртуальных машин,
- настройка программного обеспечения,
- описание задачи и рекомендаций по выполнению,
- формирование правил оценивания,

- загрузка в общий репозиторий лабораторных работ.

5) «Тесты»:

a. Открытые тесты (вопрос-выбор ответа из предложенных),

b. Открытые тесты (вопрос-ответ в виде кода программы).

6) «Администрирование»:

a. «Управление инфраструктурой» общий мониторинг всех задействованных ресурсов, просмотр лог файла, управление пользователями, контентом, инфраструктурой.

b. «Управление пользователями»: регистрация новых пользователей, авторизация, редактирование профиля, ролевая модель (студент, преподаватель, администратор), привязка к доступному перечню ресурсов (информационные системы, лабораторные работы, информационные страницы, отчеты)

1.2 План действий

Для данного семестра с заказчиком мы выбрали к разработке модуль «Витрина проектов». Для него мы дописали требования и приступили к разработке.

План действий:

1) Написание технического задания на модуль в соответствии с желаниями заказчика;

2) Создание прототипа дизайна сайта и модуля «Витрина проектов»;

3) Внесение правок в дизайн с заказчиком;

4) Итоговый дизайн сайта;

5) Адаптирование дизайна сайта к мобильным телефонам;

6) Создание бэк-части сайта (Админ-панель, карточки проектов, комментарии);

7) Верстка сайта, добавление анимаций;

8) Соединение бэк-части с фронтом.

2 ЛИЧНЫЕ ОТЧЕТЫ

2.1 Дизайнер

2.1.1 Выполненные задачи

Исходя из целей нашей команды, тимлид сформулировал для меня следующие задачи, которые я выполнила:

1) разработан дизайн карточки проекта в двух форматах;

2) разработан дизайн окна добавления/редактирования проекта;

3) разработан дизайн страницы проекта;

4) разработан дизайн компонентов для фильтрации и сортировки проектов;

5) разработан дизайн страницы витрины проектов;

6) разработан дизайн меню разделов для админ. Панели;

7) разработан дизайн раздела «Пользователи» админ панели;

8) разработан дизайн раздела «Витрина проектов» админ панели.

2.1.2 Технический стек

В данном семестре я использовала следующий технический стек:

1) Figma;

2) Библиотека иконок React Icons.

2.1.3 Пример работы

На странице витрины проектов (рисунок 1) отображаются хедер со всеми разделами сайта, боковое меню с разделами «Мои проекты» и «Все проекты», карточки проектов, кнопка «Добавить проект», выпадающий список с вариантами сортировок, кнопка смены отображения карточек между сеткой (рисунок 1) и списком (рисунок 2), кнопка сортировки по алфавиту.

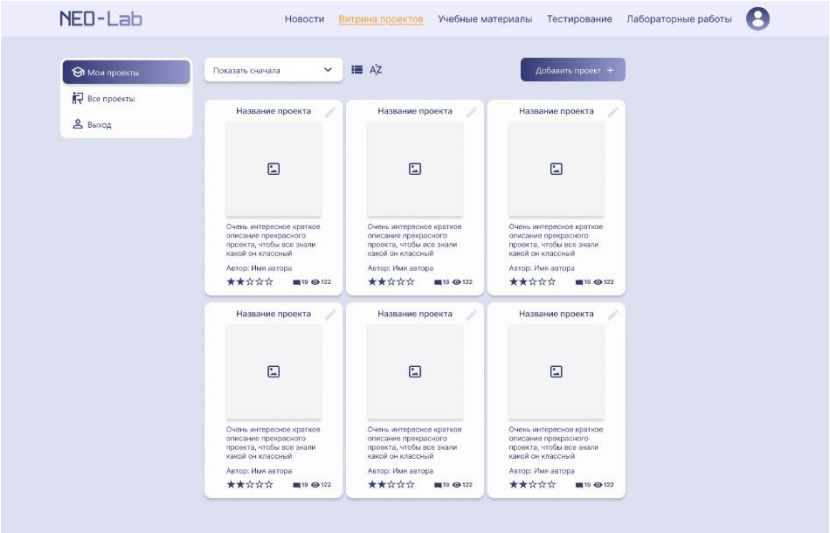

Рисунок 1 - Страница «Мои проекты» (отображение сеткой).

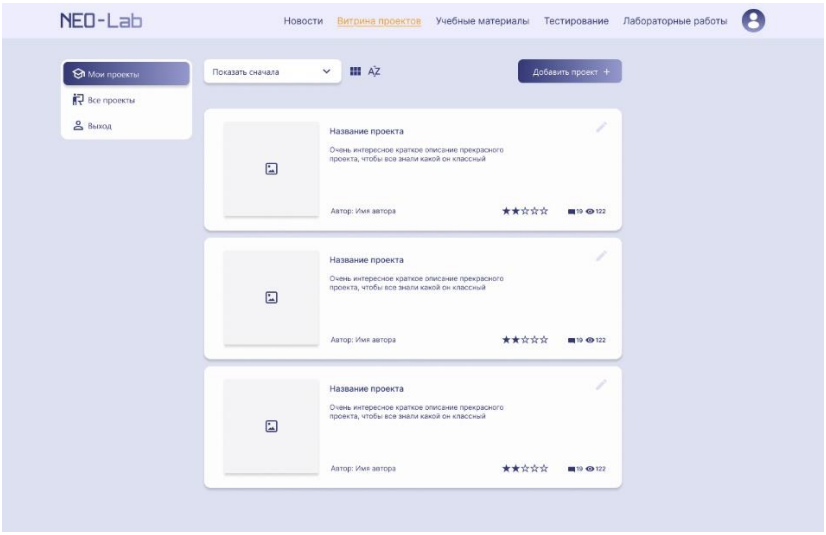

Рисунок 2 - Страница «Мои проекты» (отображение списком).

Карточки проектов состоят из названия проекта, одного изображения, текстового блока с описанием проекта и именем автора, пяти векторных фигур звёзд для отображения рейтинга проекта, иконки и количества комментариев, иконки и количества посещений. Если пользователь находится в разделе «Мои проекты», в правом верхнем углу отображается кнопка для редактирования проекта.

При нажатии на кнопку редактирования проекта пользователю открывается окно редактирования (рисунок 3). Дизайн формы редактирования

основан на блочном стиле для удобства пользователя, между блоками расстояние кратное 4 (как и во всем проекте). На форме отображаются надпись «Редактирование проекта», изображения, пяти текстовых полей разного объема (название проекта, краткое описание, подробное описание, ссылка на лендинг, ссылка на документацию), трех слотов для дополнительных изображений и кнопок «Отмена», «Сохранить» и «Закрыть».

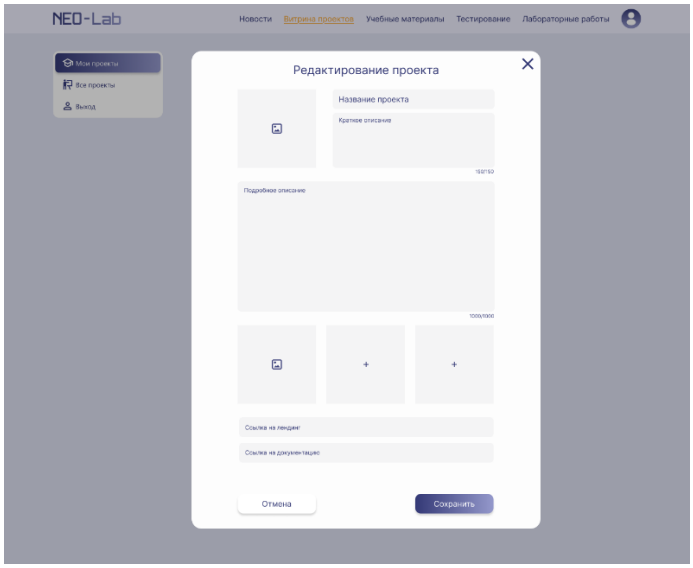

Рисунок 3 - Форма редактирования проекта.

При нажатии на кнопку «Добавить проект» открывается форма добавления проекта (рисунок 4). Она содержит те же элементы, что и форма редактирования, за исключением кнопки «Отменить».

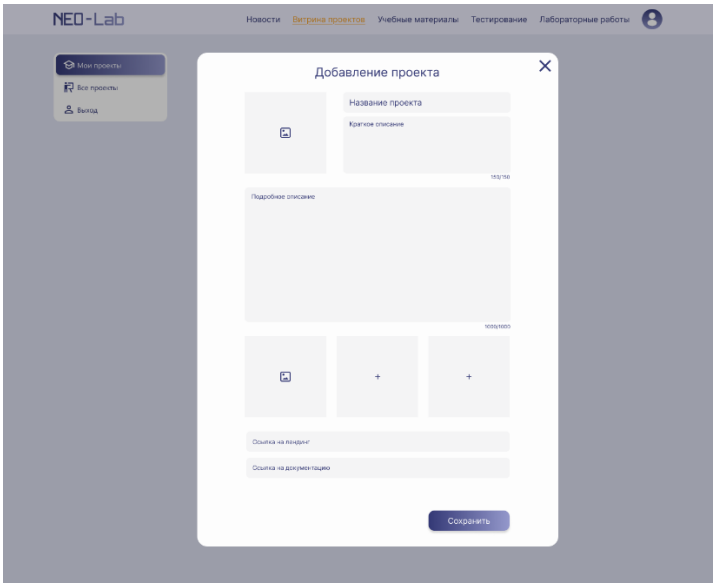

Рисунок 4 - Форма создания проекта.

При нажатии на карточку проекта открывается страница проекта (рисунок 5). Она содержит изображение, название проекта, краткое описание проекта, имя автора, рейтинг, количество просмотров и комментариев, кнопку для перехода на лендинг, большой блок с подробным описанием, блок с тремя дополнительными изображениями, кнопки «Открыть документацию», раздела комментариев, текстового поля для ввода своего комментария и кнопки «Отправить». При этом если страница проекта открыта автором или модератором через админ панель, в правом нижнем углу у комментария отображается кнопка «Удалить».

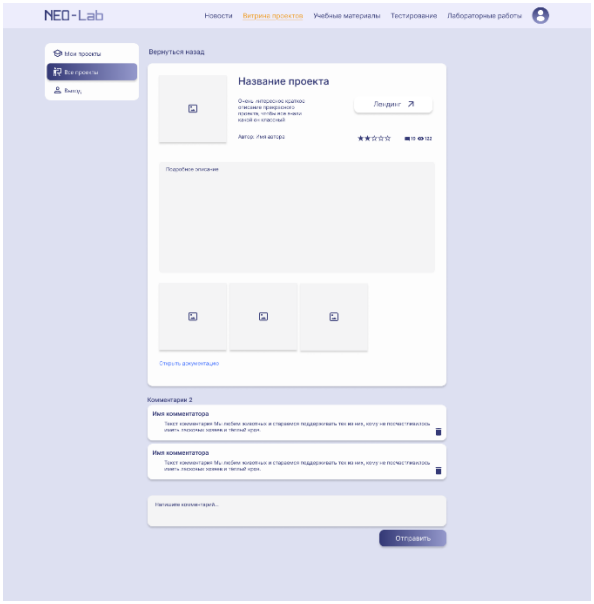

Рисунок 5 - Страница проекта.

В разделе «Пользователи» (рисунок 6) админ панели отображается боковое меню, поисковое поле, таблица со списком пользователей, состоящая из полей «Логин», «Почта», «Тип пользователя», «Активность». В столбце активность отображаются переключатели. Если перевести переключатель в режим неактивности всплывет модальное окно (рисунок 7) с надписью «Вы уверены, что хотите деактивировать пользователя?», кнопками «Да», «Отмена» и закрыть.

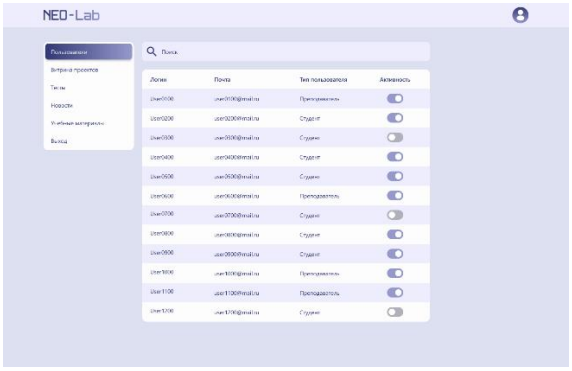

Рисунок 6 - Раздел «Пользователи» в админ панели.

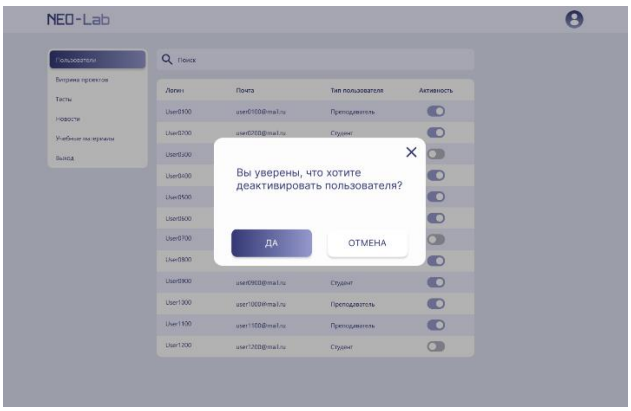

Рисунок 7 - Модальное окно при деактивации пользователя.

В разделе «Витрина проектов» (рисунок 8) отображается боковое меню, поисковое поле и список проектов с возможностью редактирования.

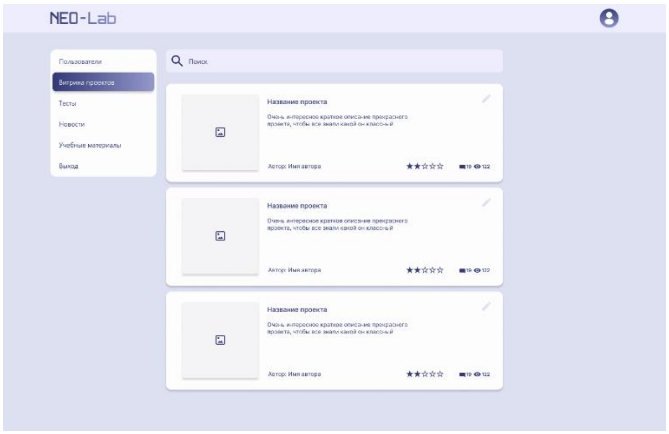

Рисунок 8 - Раздел «Витрина проектов» в админ панели.

2.1.4 Проблемы и их решения

Непростым решением для меня было размещение всех необходимых вариантов сортировок проектов так, чтобы это не утяжеляло страницу. Я просмотрела множество популярных сайтов с подобным размещением карточек, в особенности сайтов зарубежных магазинов. В итоге получилось довольно компактное решение (рисунок 9). Все сортировки разместились в верхней части страницы, основную их часть я спрятала в выпадающий список (рисунок 10), отдельно в виде кнопки вынесла сортировку по алфавиту и рядом расположила кнопку переключения между форматами списка и сетки.

Рисунок 9 - Сортировка проектов.

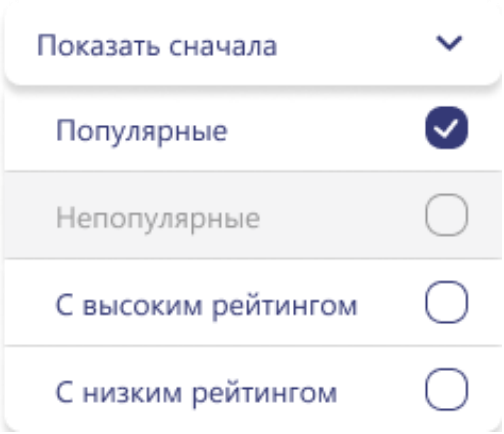

Рисунок 10 - Выпадающий список с вариантами сортировки

2.1.5 Рефлексия

Работая над проектом, я смогла не только применить уже имеющиеся знания в UI-дизайне, но и значительно продвинуться в понимании визуальной иерархии и восприятия интерфейсов пользователями.

Одним из ключевых открытий для меня стало использование теней для создания глубины и визуального отделения элементов. Ранее я редко прибегала к этому инструменту, предпочитая плоские решения. Однако в этом проекте начала активно применять мягкие тени для карточек проектов и интерактивных элементов (кнопок, выпадающих списков), чтобы интерфейс стал более читаемым и «воздушным». Это позволило выделить контент, сделать структуру страницы понятнее и приятнее для восприятия.

2.2 Тимлид-аналитик

2.2.1 Выполненные задачи

За этот семестр я выполнила множество различных задач:

1) проводила собрания с заказчиком;

2) писала техническое задание на различные блоки сайта;

3) вела доску нашего проекта на сайте Miro;

4) провела аналитику по подобным блокам других сайтов;

5) утверждала различные моменты с заказчиком;

6) делала презентации к контрольным точкам;

7) создала таблицу задач;

8) записывала видеозащиты;

9) монтировала видеозащиты;

10) писала отчет;

11) подгоняла отчет под ГОСТ;

12) следила за информацией в группе Проектный Практикум.

Данные задачи я выполняла в срок и достаточно успешно.

2.2.2 Технический стек

Для этого семестра я выбрала такой стек технологий:

1) Miro,

Этот сайт я использовала вместо YouTrack для ведения задач и дедлайнов, а также я использовала его как командное хранилище данных.

2) Word,

Данное приложение я использовала для написания отчета, для написания технического задания и для анализа аналогов нашего проекта.

3) Powerpoint,

Данное приложение я использовала для создания презентаций к контрольным точкам.

4) Capcut,

Данное приложение я использовала для редактирования отснятых роликов для видеозащиты.

5) Google Meet.

Данный сайт я использовала для собраний с заказчиком, чтобы показывать прогресс нашей команды.

2.2.3 Пример работы

За время работы над проектом в этом семестре я сделала таблицу задач по каждому из членов моей команды (рисунок 11).

Ответственный	Задача	Временные затраты
Бирючев Павел Алексеевич (Фронтендер)	Спроектировал и настроил навигацию раздела с проектами	5 ч
	Настроил функциональную UI обертку страницы, сайтбар, расположение вложенных подстраниц	6 ч
	Сверстал вертикальную и горизонтальную карточку проекта	5 ч
	Добавил для карточки проекта экшен слот (слот под кнопку)	1 ч
	Сверстал кнопку создания проекта	0,5 ч
	Сверстал компонент выборки (select)	3 ч
	Сверстал и настроил функционал кнопки смены расположения ленты карточек проекта	3 ч
	Сверстал и собрал страницу "Мои проекты" "Все проекты" из ранее сверстанных компонентов	3 ч
	Сверстал и добавил кнопку редактирования для карточки	2 ч
	Добавил подсказки (тултипы) для элементов страниц (кнопка смены расположения, кнопка редактирования, рейтинг проекта)	1 ч
	Сверстал форму редактирования/создания проекта	7 ч
	Добавил форму редактирования/создания в виде модального окна в приложение. Настроил	1 ч

Рисунок 11 – Отрывок из таблицы «Таблица затрат».

Также, я вела задачи, следила за сроками выполнения и проверяла правильность выполнения (Рисунок 12).

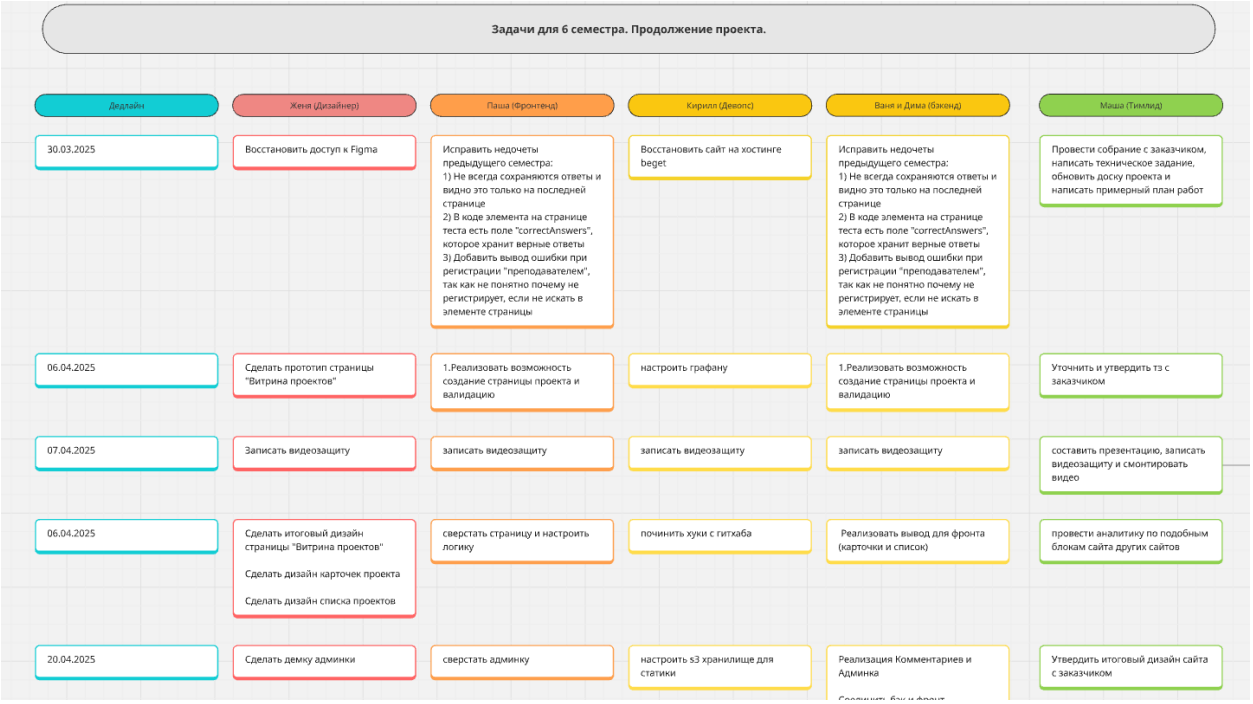

Рисунок 12 – доска задач

2.2.4 Проблемы и их решения

В течение этого семестра мы столкнулись с рядом проблем. В основном это были проблемы, связанные с большой занятостью из-за учебы некоторых членов команды из-за чего мы не все успевали сделать в срок.

Большая занятость фронтенда в учебе повлекла за собой огромные проблемы в виде отсутствия сверстанной админ панели и подключенных энд-поинтов к некоторым аспектом сайта.

Вновь повторилась проблема с техническим заданием. Некоторые члены команды по каким-то причинам проигнорировали некоторые аспекты прописанного технического задания, из-за чего нам пришлось срочно вносить правки, которых можно было избежать, будь они немного внимательнее. Для меня же это звоночек, что нужно тщательнее проверять всё, что я могу увидеть и прописывать тест-кейсы.

2.2.5 Рефлексия

Для меня данный семестр оказался крайне продуктивен. Я проработала некоторые ошибки в работе команды с предыдущего семестра и не дала

повториться им вновь. Благодаря этому наша команда, хоть и с небольшой задержкой, но справилась с поставленными задачами. Я выделила для себя еще несколько аспектов, которые в следующем семестре я исправлю, и на этот раз наша команда придет к завершению проекта раньше, чем планируется Проектным практикумом.

2.3 Фронтенд

2.3.1 Выполненные задачи

За данный семестр я выполнил следующие задачи:

1) Спроектировал и настроил навигацию раздела с проектами;

2) Настроил функциональную UI обертку страницы, сайдбар, расположение вложенных подстраниц;

3) Сверстал вертикальную и горизонтальную карточку проекта;

4) Добавил для карточки проекта экшен слот (слот под кнопку);

5) Сверстал кнопку создания проекта;

6) Сверстал компонент выборки (select);

7) Сверстал и настроил функционал кнопки смены расположения ленты карточек проекта;

8) Сверстал и собрал страницу "Мои проекты", "Все проекты" из ранее сверстанных компонентов;

9) Сверстал и добавил кнопку редактирования для карточки;

10) Добавил подсказки (тултипы) для элементов страниц (кнопка смены расположения, кнопка редактирования, рейтинг проекта);

11) Сверстал форму редактирования/создания проекта;

12) Добавил форму редактирования/создания в виде модального окна в приложение. Настроил функционал открытия и закрытия модального окна через стейт менеджер;

13) Сверстал страницу проекта;

14) Сверстал и добавил функциональность скачивания файлов проекта в виде отдельных кнопок на странице проекта;

15) Сверстал ленту комментариев к проекту;

16) Подключил загрузку ленты проектов с бэкенда;

17) Подключил загрузку отдельной страницы проекта с бэкенда;

18) Подключил создание проекта, загрузку файлов проекта через FormData;

19) Добавил кастомный аватар пользователя, генерирующийся относительно имени пользователя.

2.3.2 Технический стек

В рамках этого семестра я использовал следующий перечень технологий:

1) React;
2) TypeScript;
3) TailwindCSS;
4) Framer Motion;
5) Zustand;
6) NextUI;
7) React Hook Form;
8) Recharts;
9) Zod;
10) Tanstack Query;
11) Axios;
12) Nuqs;
13) Esling;
14) Prettier.

2.3.3 Пример работы

Листинг 1 - Компонент страницы ProjectPreviewPage; - Компонент страницы проекта. Разделен на отдельные подкомпоненты, например на компонент ProjectDetailsCard (карточка проекта), ProjectComments (лента комментариев к проекту) и компоненты-кнопки для взаимодействия со страницей и навигацией по приложению. Также в начале компонента происходит загрузка данных через хук useProjectDetails (Tanstack Query запрос). Loader запроса (момент, когда данные грузятся) и ошибка запроса обрабатывается корневым Suspense компонентом. В результате получаем

минималистичный компонент, в котором подгружаются и сразу отображаются данные.

```
export const ProjectPreviewPage = () => {
  const { projectId = '' } = useParams();
  const { data } = useProjectDetails(projectId);
  return (
    <section className="w-full max-w-[712px] flex flex-col gap-1 items-start mb-20">
      <div className="flex flex-row justify-between w-full">
        <BackButton />
        <ProjectDetailsActions project={data} />
      </div>
      <div className="flex flex-col gap-6">
        <ProjectDetailsCard project={data} />
        <ProjectComments projectId={data.id} />
      </div>
    </section>
  );
};
```

Листинг 2 - Компонент ProjectDetailsActions; - Компонент, являющийся частью страницы проекта. Содержит в себе две кнопки для скачивания документации и перехода на сайт проекта. Также для каждой кнопки добавлено описание label, которое отобразится при наведении на кнопку. Внутри компонента также происходит подгрузка файла документации через

хук useProjectFiles. Данный хук чаще всего уже берет готовые данные из кэша, а не вызывает повторный запрос данных с бэкенда.

```
export const ProjectDetailsActions = ({ project }: { project:
Pick<ProjectDTO, 'landingURL' | 'name' | 'id'> }) => {
  const {
    data: { documentation },
  } = useProjectFiles(project.id);
  const handleDocsOpen = (fileName = 'document') => {
     …
  };
  const handleProjectOpen = () => {
    window.open(project.landingURL, '_blank');
  };
  return (
    <div className="flex flex-row gap-1">
      <ProjectDetailsButton
        onClick={handleProjectOpen}
        label="Демонстрационный проект"
        icon={CgWebsite}
      />
      <ProjectDetailsButton
        onClick={() => handleDocsOpen(project.name)}
        label="Скачать документацию"
        icon={SiGoogledocs}
      />
    </div>
```

```
  );
};
```

Листинг 3 - Метод createModalStore; Метод createModalStore позволяет создать состояние модального окна с уже подготовленными методами открытия/закрытия и передачи данных или опций для модального окна. То есть вся логика взаимодействия с состояниями открытия вынесена в этот метод. Любая логика, связанная напрямую с содержанием различных модальных окон, будет пробрасываться через опции при вызове методов нужного модального окна.

```
interface ModalStore<T> {
  isOpen: boolean;
  open: (options?: T) => void;
  close: () => void;
  setOpen: (isOpen: boolean) => void;
  options: T | null;
  setOptions: (options: T) => void;
}
export const createModalStore = <T>() =>
  create<ModalStore<T>>(set => ({
    isOpen: false,
    open: options => set({ isOpen: true, options: options ??
null }),
    close: () => set({ isOpen: false, options: null }),
    setOpen: isOpen => set({ isOpen }),
    options: null,
    setOptions: options => set({ options }),
```

```
}));
```

2.3.4 Проблемы и их решения

В данном семестре я столкнулся со следующими проблемами:

1) Загрузка файлов на бэкенд;

Для проекта необходимо было добавлять большие файлы фотографий и документации, для этого пришлось изучить различные способы взаимодействия с FormData, а также разобраться как реализовать загрузку через React Hook Form (библиотека для работы с формами). Проблему удалось просто и быстро решить, данные пробрасываются, как и раньше в метод запроса, а уже в самом методе запроса генерируется FormData с нужными полями и отправляется на бэкенд.

2) Повторение логики модальных окон;

Количество модальных окон начинает расти, соответственно растет количество кода и повторяющейся логики. Решение оказалось простым, просто вынести общую логику взаимодействия с модальным окном в кастомную функцию стейт менеджера и просто генерировать шаблон для нужного модального окна.

3) Нехватка подсказок.

Существуют определенные минималистичные кнопки и компоненты, логика которых может быть не сразу понятна пользователю. Например, кнопки скачивания документации на странице проектов. Это полезно, чтобы пользователь не боялся пользоваться сайтом и сразу понимал, что произойдет если нажать на определенную кнопку. Поэтому решил добавлять подсказки (тултипы) – надпись над кнопкой о том, что она делает.

2.3.5 Рефлексия

Я научился обрабатывать файлы, фотографии и отправлять их на бэкенд.

Я научился выделять общую логику стейт менеджеров в определенные функции-шаблоны, тем самым избавился от повторяющегося кода.

Я научился писать поддерживаемый код, могу, например создать пустой компонент страницы по определенному пути и получить уже полноценную страницу с боковой панелью. Это очень удобно, когда каждая страница работает как отдельный виджет внутри одной общей страницы, с одним общим шаблоном. То есть мне не приходится заботиться об окружении страницы, я просто пишу новый код и ничего не копирую/вставляю.

2.4 Бэкенд

2.4.1 Выполненные задачи

За этот семестр мной были выполнены следующие задачи:

1) реализовано создание карточек проектов с возможностью прикрепления файлов;

2) реализовано обновление карточек, включая изменение метаданных и замену файлов;

3) реализовано удаление карточек с очисткой связанных файлов в хранилище;

4) интегрировано хранение файлов в Yandex Object Storage (совместимом с Amazon S3);

5) настроено логирование операций для отслеживания ошибок;

6) протестированы API-эндпоинты с использованием Swagger и Postman;

7) обеспечено взаимодействие с PostgreSQL для хранения метаданных карточек.

2.4.2 Технический стек

В рамках этого семестра я использовал следующий стек технологий:

1) язык программирования C#;

2) фреймворк .NET 8;

3) DDD (Domain-Driven Design);

4) PostgreSQL;

5) Yandex Object Storage (через Amazon S3 SDK);

6) Swagger;

7) Postman;

8) ILogger (встроенный в .NET).

2.4.3 Пример работы

Листинг 4 - Метод CreateAsync, который отвечает за загрузку файла в хранилище.

Логика работы:

1) проверяет, что файл не пустой;

2) нормализует путь к директории (заменяет \ на /).

3) формирует objectKey – уникальный идентификатор файла в формате {директория}/{имя_файла}{расширение}.

4) открывает поток файла и загружает его в хранилище с публичными правами доступа (PublicRead).

```
public async Task<string> CreateAsync(IFormFile? file, string
directoryPath, string fileName)
{
    if (file == null || file.Length == 0)
        return string.Empty;
    var normalizedDirectory = directoryPath.Replace('\\', '/');
    var extension = Path.GetExtension(file.FileName);
    var objectKey =
$"{normalizedDirectory}/{fileName}{extension}".TrimStart('/');
    using (var stream = file.OpenReadStream())
    {
        var putRequest = new PutObjectRequest
        {
            BucketName = _bucketName,
            Key = objectKey,
            InputStream = stream,
            ContentType = file.ContentType,
            CannedACL = S3CannedACL.PublicRead
```

```
        };
        await _s3Client.PutObjectAsync(putRequest);
    }
    return objectKey;
}
```

Листинг 5 - Метод DeleteAsync, который удаляет файл по его objectKey.

Логика работы:

1. формирует запрос на удаление (DeleteObjectRequest);

2. отправляет запрос в хранилище;

3. логирует ошибки, если удаление не удалось (например, файл не найден).

```
public async Task DeleteAsync(string objectKey)
{
    try
    {
        var deleteRequest = new DeleteObjectRequest
        {
            BucketName = _bucketName,
            Key = objectKey
        };
        await _s3Client.DeleteObjectAsync(deleteRequest);
    }
    catch (AmazonS3Exception ex)
    {
```

```
            _logger.LogError($"Ошибка при удалении файла
{objectKey}: {ex.Message}");
        }
    }
```

2.4.4 Проблемы и их решения

Я столкнулся с тремя проблемами.

Первой проблемой стали ошибки при загрузке файлов в Yandex Object Storage. Симптомом этой проблемы стала ошибка 403 при загрузке файлов на сервер. Причиной такого явления стали неправильно настроенные права доступа к бакету. Эту проблему я решил одной строчкой: CannedACL = S3CannedACL.PublicRead в PutObjectRequest.

Второй проблемой стало некорректное удаление файлов. Симптомом этой проблемы стало сохранение файлов в хранилище после удаления карточки проекта. Причиной этого события стало отсутствие обработчика ошибок при удалении карточки проекта. Чтобы решить эту проблему я добавил логирование и проверку существования файла перед удалением.

Третьей проблемой я выделил долгую загрузку больших файлов. Симптомом этой проблемы стало зависание загрузки при загрузке файлов больше 100 мб. Я реализовал потоковую загрузку через stream.CopyToAsync() и проблема была решена.

2.4.5 Рефлексия

За время работы над проектом я серьезно прокачался в нескольких направлениях. Я освоил интеграцию S3-совместимых хранилищ в .NET, улучшил понимание микросервисной архитектуры – лучше начал разделять ответственность между сервисами, научился эффективно отлавливать и логировать ошибки, а также понял, насколько важна потоковая обработка данных.

2.5 Бэкенд

2.5.1 Выполненные задачи

В рамках разработки бэкенд-части сервиса карточек проектов LMS были выполнены следующие задачи:

1) реализован CRUD для комментариев: создание, получение списка, обновление и удаление;

2) реализация Рейтинга;

3) реализация АПИ администратора;

4) декомпозиция задач на бекэнд;

5) настроен маппинг DTO для безопасного обмена данными между слоями приложения;

6) протестированы API-эндпоинты с использованием Swagger и Postman;

7) обеспечено взаимодействие с PostgreSQL.

2.5.2 Технический стек

В рамках этого семестра я использовал следующий стек технологий:

1) язык программирования C#;

2) фреймворк .NET 8;

3) DDD (Domain-Driven Design);

4) PostgreSQL;

5) Yandex Object Storage (через Amazon S3 SDK);

6) Swagger;

7) Postman;

8) ILogger (встроенный в .NET).

2.5.3 Пример работы

Листинг 6 - Метод GetAllRatingsForProjectAsync; - используется для получения списка всех оценок проекта. Этот метод проверяет существование проекта, запрашивает его оценки и преобразует список сущностей.

```
public async Task<List<RatingDto>>
GetAllRatingsForProjectAsync(Guid projectId)
    {
        var project = await
_projectRepository.GetByIdAsync<ProjectCard>(projectId);
        if (project == null)
        {
            throw new NotFoundException($"Project with id
{projectId} not found.");
        }
        var ratings = await
_ratingRepository.GetAllRatingsForProjectAsync(projectId);
        return _mapper.Map<List<RatingDto>>(ratings);
    }
```

Листинг 7 - Метод GetAllRatingsForProjectAsync; - используется, чтобы анализировать оценки проекта.

```
public async Task<List<RatingDto>>
GetAllRatingsForProjectAsync(Guid projectId)
    {
        var project = await
_projectRepository.GetByIdAsync<ProjectCard>(projectId);
        if (project == null)
        {
            throw new NotFoundException($"Project with id
{projectId} not found.");
        }
```

```
        var ratings = await
_ratingRepository.GetAllRatingsForProjectAsync(projectId);
        return _mapper.Map<List<RatingDto>>(ratings);
    }
```

2.5.4 Проблемы и их решения

Проверка существования проекта.

Проблема: При создании или обновлении рейтинга можно было добавить оценку к несуществующему проекту, что приводило к ошибкам в данных.

Решение: Добавили проверку через _projectRepository.GetByIdAsync перед любой операцией с рейтингом. Если проект не найден — выбрасывается NotFoundException.

Дублирование оценок от одного пользователя.

Проблема: Пользователь мог оставить несколько оценок для одного проекта, что искажало средний рейтинг.

Решение: Реализовали метод GetUserRatingForProjectAsync, который проверяет существующую оценку пользователя. Если она есть — обновляем значение, а не создаем новую запись.

Некорректное обновление рейтинга проекта.

Проблема: После добавления, изменения или удаления оценки средний рейтинг проекта не обновлялся автоматически.

Решение: Добавили вызов UpdateProjectRatingAsync после каждой операции с оценками. Этот метод пересчитывает средний рейтинг и сохраняет его в проекте.

Валидация значения рейтинга.

Проблема: В базу данных могли попасть значения вне диапазона 1–5 (например, 0 или 6).

Решение: Добавили явную проверку if (ratingDto.Value is < 1 or > 5) с исключением BadRequestException.

2.5.5 Рефлексия

За этот семестр я научился работе с AutoMapper для упрощения преобразования объектов. Попрактиковался использовать кастомные исключения для точечной обработки ошибок. Разобрался с оптимизацией запросов к БД: проверкой существования проекта перед операциями с комментариями. Закрепил навыки работы с async/await в цепочке вызовов репозиториев.

2.6 Девопс

2.6.1 Выполненные задачи

За этот семестр я выполнил следующие задачи:

1) Развернул новый микросервис с использованием Docker Compose;

2) Настроил PostgreSQL для работы с сервисом;

3) Интегрировал системы мониторинга: Prometheus и Grafana;

4) Реализовал сбор метрик для PostgreSQL с помощью pg_exporter;

5) Обновил CI/CD-пайплайн в Jenkins для автоматизации развертывания;

6) Проект развернул в Yandex Cloud с использованием Terraform.

2.6.2 Технический стек

В течение этого семестра я использовал следующий список технологий:

1) Docker,

2) Docker Compose,

3) Terraform,

4) Yandex Cloud,

5) PostgreSQL,

6) Prometheus,

7) Grafana,

8) pg_exporter,

9) CI/CD: Jenkins.

2.6.3 Пример работы

Листинг 8 - Jenkins-пайплайн; - автоматизирует процесс очистки окружения, обновления кода и деплоя микросервисов в Docker.

Ключевые моменты:

1) Очистка Docker (Clean Docker)

a. Удаляются существующие контейнеры (udv-cyberlab-tests-1, udv-cyberlab-identity-1, udv-cyberlab-projects-1);

b. Выполняется docker system prune для освобождения места.

2) Обновление кода (Checkout)

a. Переход в директорию /UDV-CyberLab;

b. Обновление кода через git pull;

c. Вывод содержимого docker-compose.yml для проверки.

3) Сборка и деплой (Build and Deploy)

a. Установка переменных окружения для работы с Yandex Cloud Storage.

b. Запуск сервисов через docker-compose up -d --build.

Особенности:

a. Используются секретные ключи (Yandex Access/Secret Key), которые хранятся в переменных окружения Jenkins.

b. Для работы с облачным хранилищем Yandex заданы параметры:

i. YANDEX_BUCKET_NAME — имя бакета.

ii. YANDEX_SERVICE_URL — эндпоинт API.

```
pipeline {
    agent any
    environment {
        ***
    }
    stages {
        stage('Clean Docker') {
            steps {
                sh '''
                    docker rm -f udv-cyberlab-tests-1 udv-
cyberlab-identity-1 udv-cyberlab-projects-1
                    docker system prune -af --volumes
                '''
            }
        }
        stage('Checkout') {
            steps {
```

```
                sh '''
                cd /UDV-CyberLab
                    git pull
                    cat docker-compose.yml
                '''
            }
        }
        stage('Build and Deploy') {
            steps {
                sh '''
                export YANDEX_BUCKET_NAME="udv-bucket"
                export
YANDEX_SERVICE_URL="https://storage.yandexcloud.net"
                export YANDEX_ACCESS_KEY="*****"
                export YANDEX_SECRET_KEY="*****"
                cd /UDV-CyberLab
                    docker-compose up -d --build
                '''
            }
        }
    }
```

2.6.4 Проблемы и их решения

Проблема: Ошибка подключения Prometheus к PostgreSQL

После развертывания инфраструктуры выяснилось, что Prometheus не собирает метрики с PostgreSQL. В логах pg_exporter наблюдались ошибки соединения, а в Grafana отсутствовали данные. При проверке конфигурации обнаружилось, что:

- В prometheus.yml был указан неверный target (использовался localhost:9187 вместо pg_exporter:9187);
- В docker-compose.yml не было явного указания сети (network), из-за чего контейнеры не видели друг друга;

• В pg_exporter не были корректно прописаны переменные окружения для подключения к БД.

Решение:

1. Исправлен prometheus.yml;

```
yaml
Copy
Download
scrape_configs:
  - job_name: 'postgres'
    static_configs:
      - targets: ['pg_exporter:9187']
```

2. Добавлена общая сеть в docker-compose.yml;

```
yaml
Copy
Download
networks:
  app_net:
    driver: bridge
```

3. Проверены переменные окружения pg_exporter (логин, пароль, порт БД).

После перезапуска системы метрики начали поступать в Prometheus, а дашборды в Grafana отображали актуальные данные.

Проблема: Jenkins-пайплайн падал на этапе деплоя

После успешного прохождения тестов пайплайн завершался с ошибкой:

```
Copy
Download
Permission denied: cannot execute 'docker-compose up'
```

Анализ:

• пользователь jenkins не состоял в группе docker;

• в скрипте использовались команды без sudo, что запрещено настройками безопасности;

• в docker-compose.yml были жестко зашиты пути к volumes, недоступные для Jenkins.

Решение:

1) Добавление пользователя jenkins в группу docker:

```
bash
Copy
Download
sudo usermod -aG docker jenkins
```

2) Изменение скрипта деплоя в Jenkinsfile:

```
groovy
    Copy
    Download
    stages {
      stage('Deploy') {
        steps {
          sh 'sudo docker-compose -f /path/to/docker-
compose.yml up -d'
        }
      }
    }
```

3) Замена абсолютных путей в volumes на относительные.

Пайплайн стал стабильно деплоить приложение без ручного вмешательства.

2.6.5 Рефлексия

Работа над проектом позволила мне значительно углубить знания и приобрести практический опыт в следующих областях:

1) Работа с облачной инфраструктурой

a. Освоил развертывание и управление ресурсами в Yandex Cloud через Terraform, научился работать с модулями и настраивать безопасность.

b. Понял важность корректного планирования ресурсов (CPU, RAM, дискового пространства) для облачных инстансов.

2) Мониторинг и метрики

a. На практике настроил связку Prometheus + Grafana, включая создание кастомных дашбордов.

b. Узнал о специфике сбора метрик для PostgreSQL через pg_exporter и важности правильного конфигурирования таргетов.

3) CI/CD и автоматизация

a. Улучшил навыки написания Jenkins Pipeline, включая работу с секретами (credentials) и обработку ошибок.

b. Осознал, как важно логировать каждый этап пайплайна для быстрой диагностики проблем.

4) Проблемы и их решение

a. Столкнулся с нюансами сетевого взаимодействия контейнеров в Docker (например, необходимость явного указания сетей).

b. Научился анализировать логи и системные ошибки, что ускорило процесс отладки.

5) Безопасность

a. Понял критичность защиты ключей доступа (Yandex Cloud, БД) и необходимость использования переменных окружения вместо хардкода.

Вывод: Проект стал отличной возможностью применить теоретические знания на практике. Теперь я лучше понимаю, как выстраивать надежную и отказоустойчивую инфраструктуру.

ЗАКЛЮЧЕНИЕ

За этот семестр наша команда сделала большую работу. Нам пришлось полностью с нуля продумывать проект, его архитектуру, его дизайн, заново выявлять требования заказчика. Мы работали не всегда слаженно, не всегда в сроки. Мы столкнулись с огромным количеством проблем, на которые пришлось потратить большое количество времени, но, несмотря на это, мы успели завершить данный модуль – Витрина проектов. В дальнейшем мы планируем продолжить работу над данным проектом, так как он является довольно интересным и многому учит нас.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Figma. Справочный центр [Электронный ресурс]. – URL: https://help.figma.com/hc/ru (дата обращения: 24.04.2025);

2. Dannaway A. 16 простых и эффективных правил дизайна UI [Электронный ресурс] // Habr. – URL: https://habr.com/ru/companies/ruvds/articles/732942/ (дата обращения: 18.05.2025);

3. Грашин А. А. Методология дизайн-проектирования: учебное пособие [Электронный ресурс]. – М.: МГТУ им. Н. Э. Баумана, 2019. – 72 с. – URL: https://design.bmstu.ru/assets/methods/methodolog-dis-proekt-2019.pdf (дата обращения: 27.03.2025);

4. Андрющенко О. Э. Разработка концепции веб-сайта для основной образовательной программы магистратуры «Графический дизайн» СПбГУ: магистерская диссертация [Электронный ресурс]. – СПб.: СПбГУ, 2019. – 85 с. – URL: https://expert-mik.ru/master/razrabotka-konczepczii-veb-sajta-dlya-osnovnoj-obrazovatelnoj-programmy-magistratury-graficheskij-dizajn-spbgu/ (дата обращения: 02.04.2025);

5. Stepik. Введение в дизайн: онлайн-курс [Электронный ресурс]. – URL: https://stepik.org/course/94293/promo (дата обращения: 19.03.2025);

6. Менеджер пакетов NPM [Электронный ресурс]. – URL: https://www.npmjs.com/ (дата обращения: 21.05.2025);

7. Официальная документация Tanstack Query [Электронный ресурс]. – URL: https://tanstack.com/query/v5/docs/framework/react/guides/infinite-queries (дата обращения: 25.04.2025);

8. Официальная документация NextUI (HeroUI) [Электронный ресурс]. – URL: https://www.heroui.com/docs/guide/introduction (дата обращения: 29.03.2025);

9. Официальная документация React [Электронный ресурс]. – URL: https://react.dev/reference/react (дата обращения: 15.05.2025);

10. Официальная документация React Hook Form [Электронный ресурс]. – URL: https://react-hook-form.com/docs (дата обращения: 17.04.2025);

11. Документация MDN Form Data [Электронный ресурс]. – URL: https://developer.mozilla.org/en-US/docs/Web/API/FormData (дата обращения: 11.04.2025);

12. Документация Learn JS с примерами по Form Data [Электронный ресурс]. – URL: https://learn.javascript.ru/formdata (дата обращения: 07.05.2025);

13. Документация Amazon S3 для .NET – AWS SDK for .NET [Электронный ресурс]. – URL: https://docs.aws.amazon.com/sdk-for-net/v3/developer-guide/s3-apis-intro.html (дата обращения: 22.03.2025);

14. Yandex Object Storage – Документация Yandex Cloud [Электронный ресурс]. – URL: https://cloud.yandex.ru/docs/storage/ (дата обращения: 16.05.2025);

15. Domain-Driven Design (DDD) – Книга Эрика Эванса "Предметно-ориентированное проектирование" [Электронный ресурс]. – URL: https://domainlanguage.com/ddd/ (дата обращения: 05.04.2025);

16. Работа с PostgreSQL в .NET – Entity Framework Core Docs [Электронный ресурс]. – URL: https://learn.microsoft.com/en-us/ef/core/providers/npgsql/ (дата обращения: 13.04.2025);

17. Видео по микросервисной архитектуре – Microservices in .NET [Электронный ресурс]. – URL: https://www.youtube.com/watch?v=d5_FjQy6gZs (дата обращения: 23.05.2025).