Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ

Кафедра «Математической кибернетики и информационных технологий»

UI/UX

Лабораторная работа

Выполнил:

студент группы БВТ2002

Татаринов К.В.

Москва 2022

Оглавление

[1. Цель работы 3](#_Toc115642683)

[2. Постановка задачи 3](#_Toc115642684)

[3. Разработка макета 5](#_Toc115642685)

[4. Разработка сайта 7](#_Toc115642686)

[4.1. Структура сайта 7](#_Toc115642687)

[4.2. Разработка программного кода 7](#_Toc115642688)

[4.2.1. Файловая система приложения 9](#_Toc115642689)

[4.2.2. Компоненты приложения 10](#_Toc115642690)

[4.2.3. Мультиязычность 15](#_Toc115642691)

[4.2.4. Проекты (портфолио) 18](#_Toc115642692)

[4.2.5. Валидация типов 20](#_Toc115642693)

[4.2.6. Код страницы (Лендинг пейдж) 21](#_Toc115642694)

[4.2.7. Система отслеживания версий. Github Pages 22](#_Toc115642695)

[5. Результат индивидуальной работы в ходе учебной практики 23](#_Toc115642696)

[6. Заключение 26](#_Toc115642697)

# 1. Цель работы

Целью лабораторной работы является обучение ReactJS. В ходе работы будет изучено компоненты React’a, хуки, пропы и состояния.

# 2. Постановка задачи

Изученные во время учебы навыки необходимо применить при   
выполнении собственного задания – Сайта-портфолио. Разрабатываемый сайт является сборником выполненных работ в сфере Web-Разработки.

Портфолио специалиста, который продвигает услуги в интернете – одна из ключевых вещей в презентации навыков и опыта проектов. Целевую аудиторию сайта можно разделить на 5 групп: –Веб-студии, которые могут воспользоваться услугами представленного специалиста или предложить работу; –Небольшие компании, которым необходим сайт; –Индивидуалы, которым необходимо создание собственного сайта-визитки/портфолио; –Другие специалисты данной области, для поиска вдохновения и идей. Главная задача сайта – в удобном формате предоставить краткую информацию о веб-разработчике, выполненные проекты, контакты, заинтересовать потенциального клиента, показать уровень профессионализма.

Исходя из поставленных задач была построена примерная концепция сайта:

1. Сайт должен быть мультиязычным (RU/EN) для расширения целевой аудитории. Были выбраны языки: русский и английский. английский, так как этот язык является интернациональным. А русский из-за территориального расположения специалиста
2. Сайт должен простым способом донести информацию о разработчике и его навыках
3. Главное, для чего создается сайт-портфолио – показать примеры работ
4. Простым способом донести информацию, о том, как можно связаться с специалистом
5. Минималистичный дизайн, для простого донесения до аудитории нужной информации путем использования небольшого количества элементов и цветом, акцентирование на главном, привлечения внимания клиентов. В целом минимализм очень привлекателен для многих людей и очень чистый, благодаря чему внимание аудитории не рассеивается

# 3. Разработка макета

Первоначальный этап в разработке веб приложения это создание макета.

В программе Figma был разработан макет сайта на двух языках, pdf-резюме на двух языках, Favicon, Модальных окон (см. рисунок 1).

Разработанный макет был выполнен в стиле минимализм.

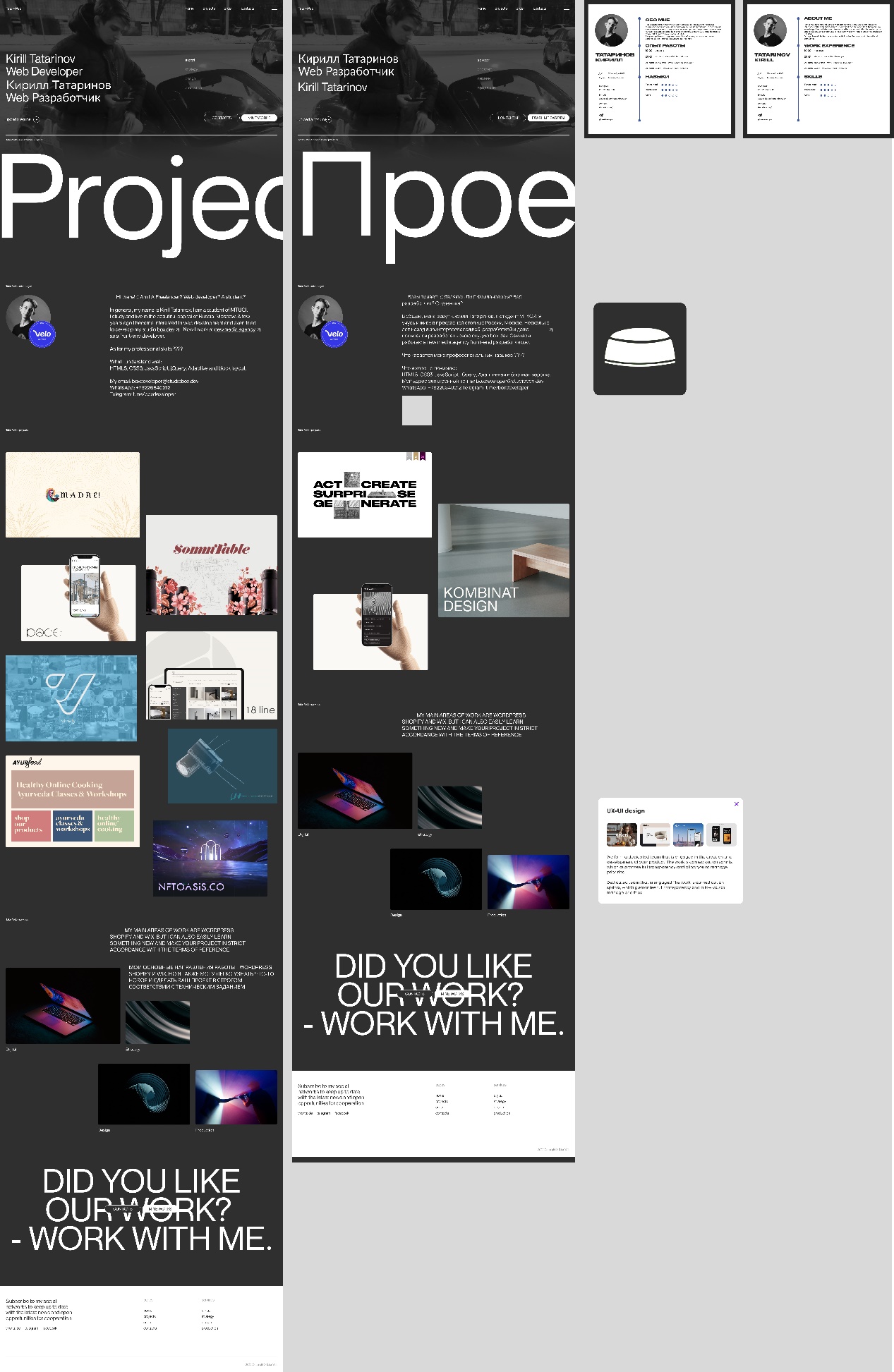


Рисунок 1 – Разработанный макет сайта в Figma

# 4. Разработка сайта

# 4.1. Структура сайта

Сайт будет представлен на одной странице, которая содержит:

1. “Hero” блок, в котором есть якорные кнопки на важные секции сайта
2. Блок “Обо мне” рассказывает краткую информацию о специалисте
3. Секция “Проекты”, по сути является портфолио разработчика, сборник выполненных проектов
4. Секция “Услуги” – краткая информация о предоставляемых специалистом услуг в сфере Web-разработки
5. “Call to action” блок, призывающий пользователя к действию: Целевое действие “Связаться с разработчиком”

# 4.2. Разработка программного кода

Как было указано ранее в данной работе будет использован ReactJS. В соответствие с официальной документацией React создаем новое приложение React (см. рисунок 2)

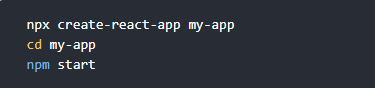


Рисунок 2 – Создание ReactJS приложения

Для всей работы над ReactJS приложением было выбрано использовать программу JetBrains WebStorm (см. рисунок 3)

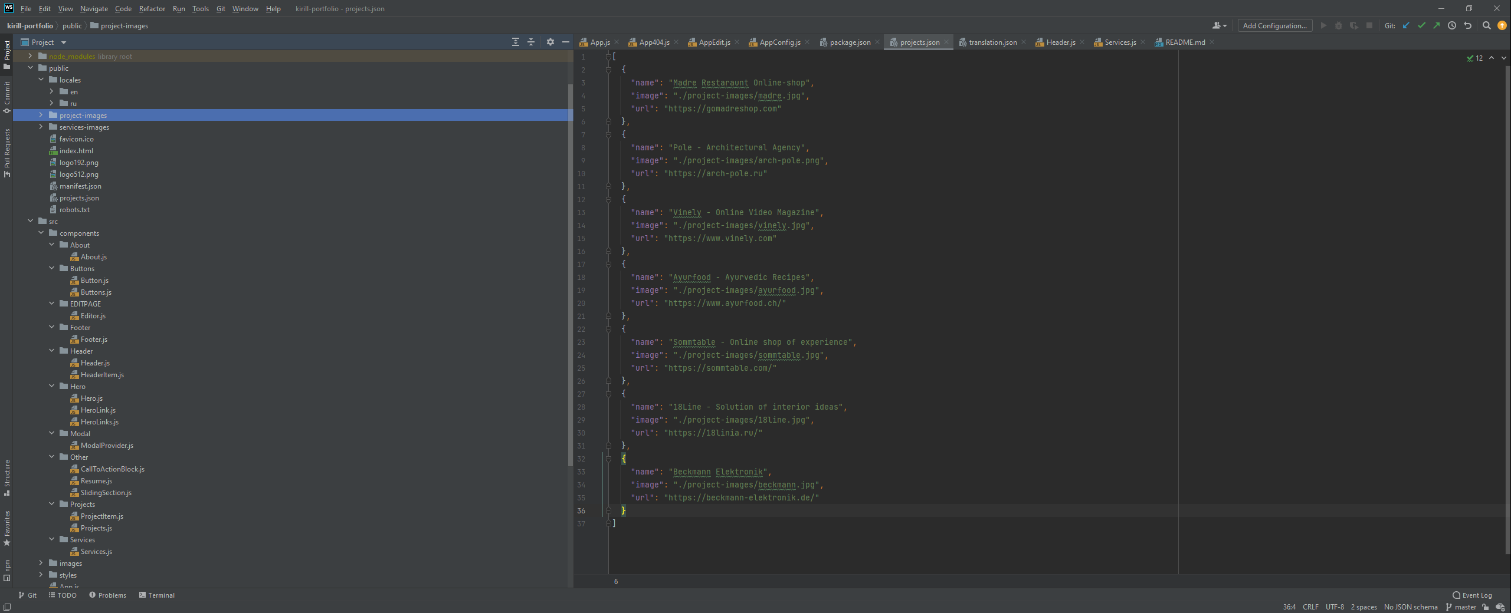


Рисунок 3 – программа для работы над приложением

# 4.2.1. Файловая система приложения

Дальнейшим шагом следует создание файловой системы приложения. Описание директорий и файловой системы, используемой в данном приложении, представлено на рисунке 4

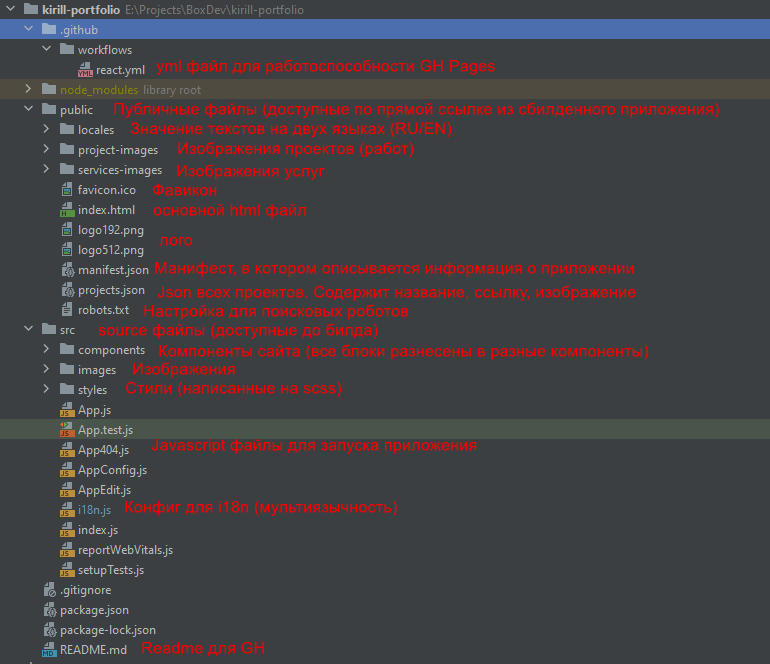


Рисунок 4 – Файловая система приложения

# 4.2.2. Компоненты приложения

В соответствие с файловом системой (см. рисунок 3) были созданы компоненты в директории /src/components/ (см. рисунок 5). Пример одного из компонентов, блок “Hero” представлен на рисунке 6.

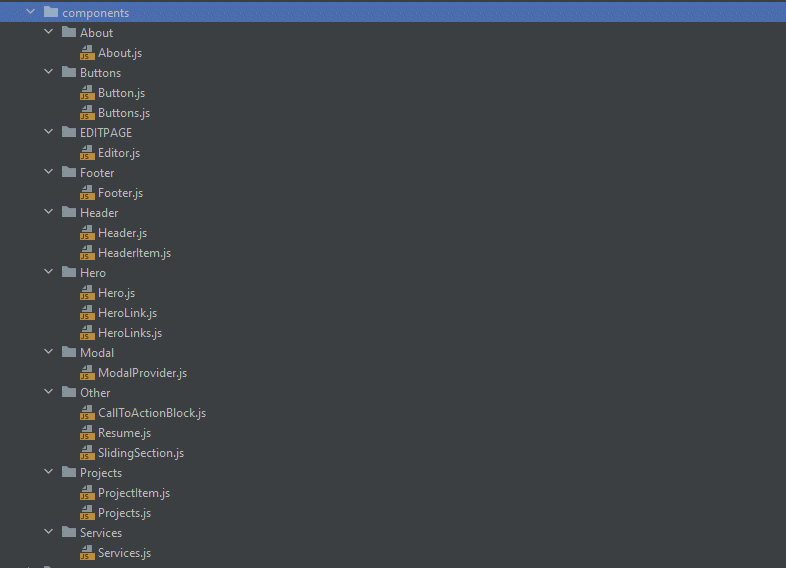


Рисунок 5 – Компоненты приложения

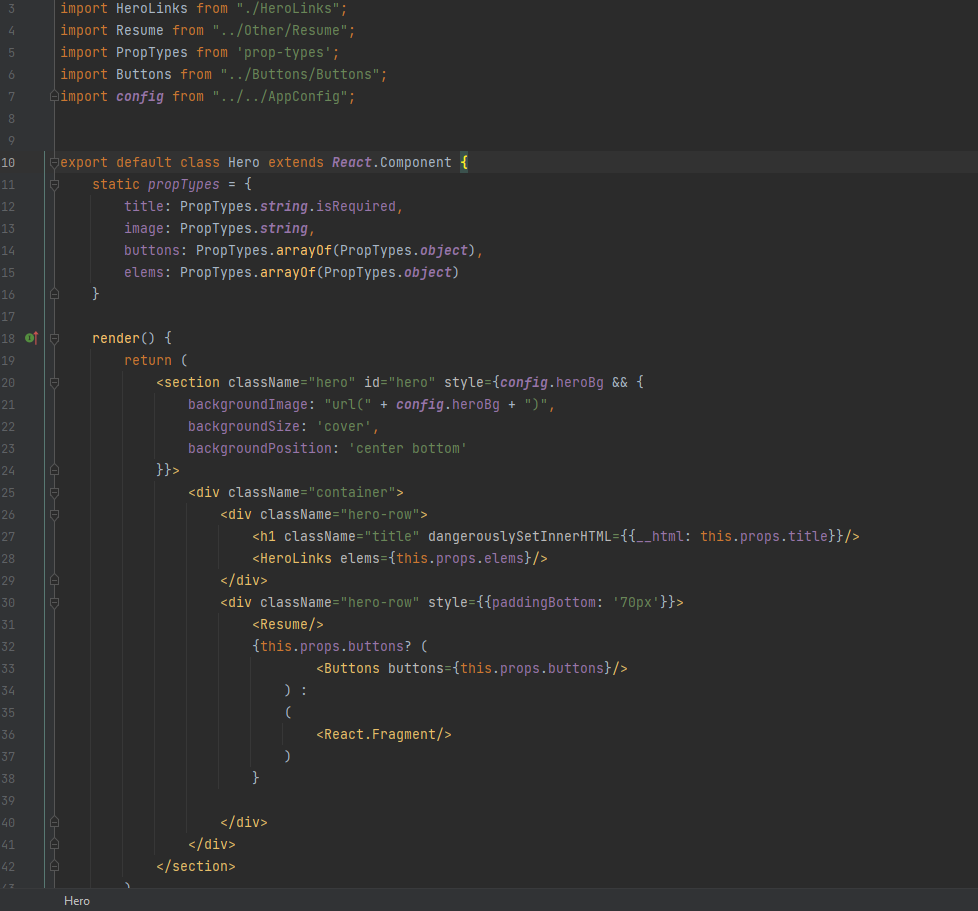


Рисунок 6 – Компонент: блок “Hero”

Для использования компонента на странице пользуемся структурой похожей на DOM-Теги и передаем JSX-атрибуты (см. рисунок 7)



Рисунок 7 – Использование пользовательского React Компонента “Hero”

Любой html элемент выглядит очень плохо без стилей поэтому по аналогии с компонентами (см. рисунок 5) создаем структуру стилей (см. рисунок 8).

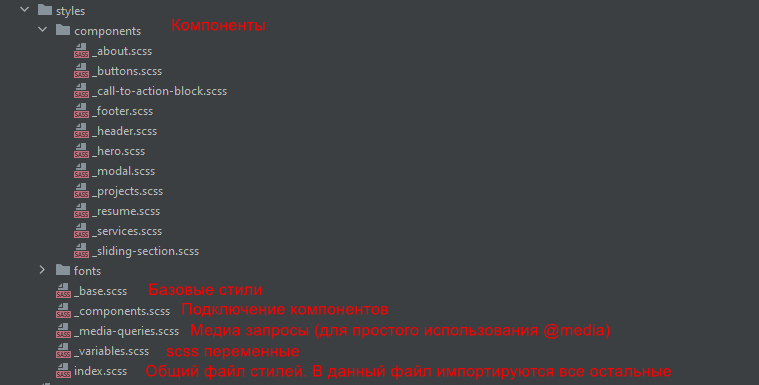


Рисунок 8 – Стили проекта

Для удобства в данной работе был использован препроцессор SCSS.

В директории /src/styles/components/ создаем стили для определенного компонента (см. рисунок 9) и в дальнейшем подключаем его в \_components.scss (см.рисунок 10), а его в свою очередь в index.scss (см.рисунок 11).

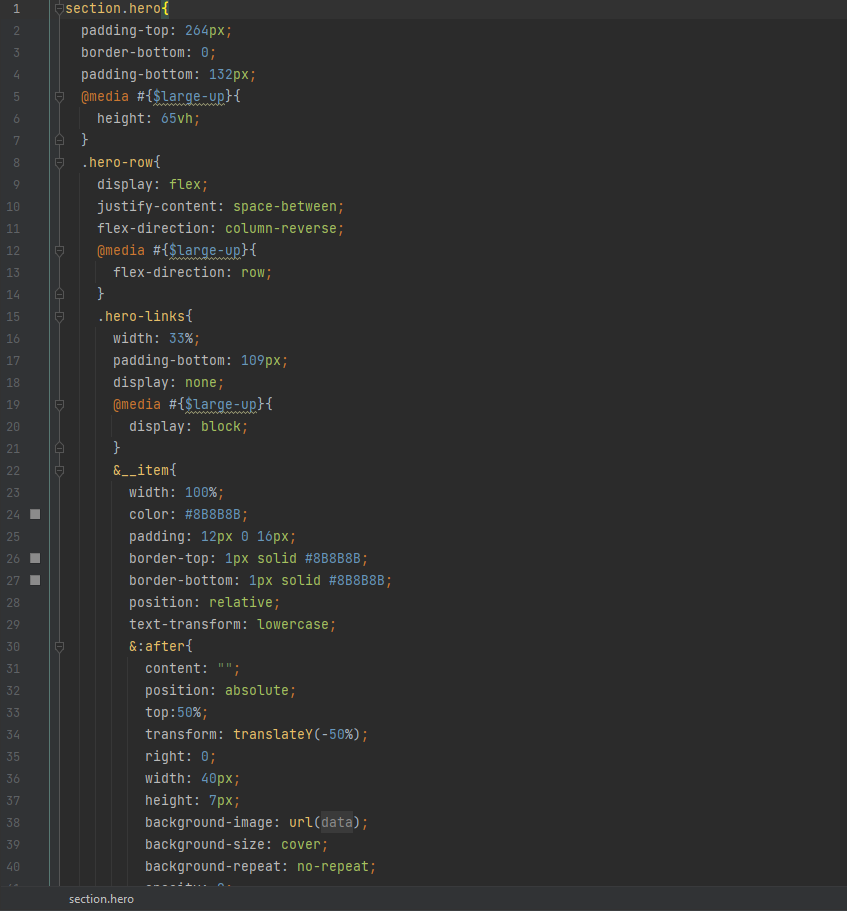


Рисунок 9 – Стили блока “Hero”

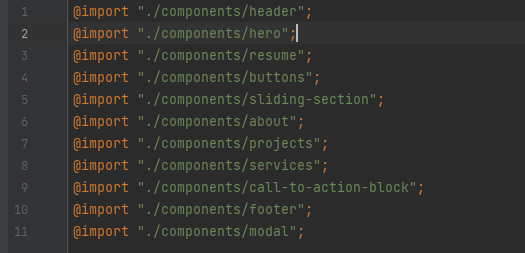


Рисунок 10 – Файл \_components.scss



Рисунок 11 – Файл index.scss

Действия описанные в главе 4.2.1 были выполнены для всех остальных компонентов приложения

# 4.2.3. Мультиязычность

На рисунке 7 можно заметить, что в JSX-атрибутах используется конструкция *t("Hero.title")*, это мультиязычность выполненная с помощью модуля i18n.js. В файл app.js мы импортируем хук *useTranslation* и объявляем его в коде (см. рисунок 12). Предварительно был модуль был настроен (см. рисунок 13) и созданы json файлы с переводами на Русском (см. рисунок 14) и Английском (см. рисунок 15). Стандартным языком был выбран Английский.

После данных предварительных настроек (рисунки 12 – 15) мы можем обращаться к элементам Json объекта по их названию. Например чтобы получить заголовок блока “Hero”, на текущем языке, в коде будет использована конструкция *t("Hero.title")* (см. рисунок 7)

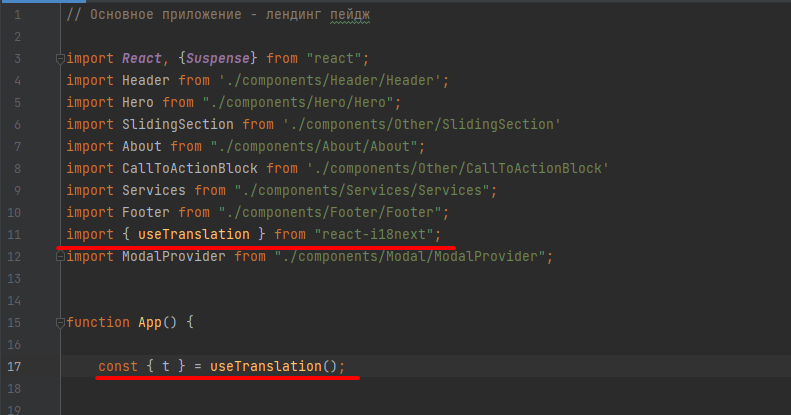


Рисунок 12 – Использование модуля i18n



Рисунок 13 – Конфиг для i18n

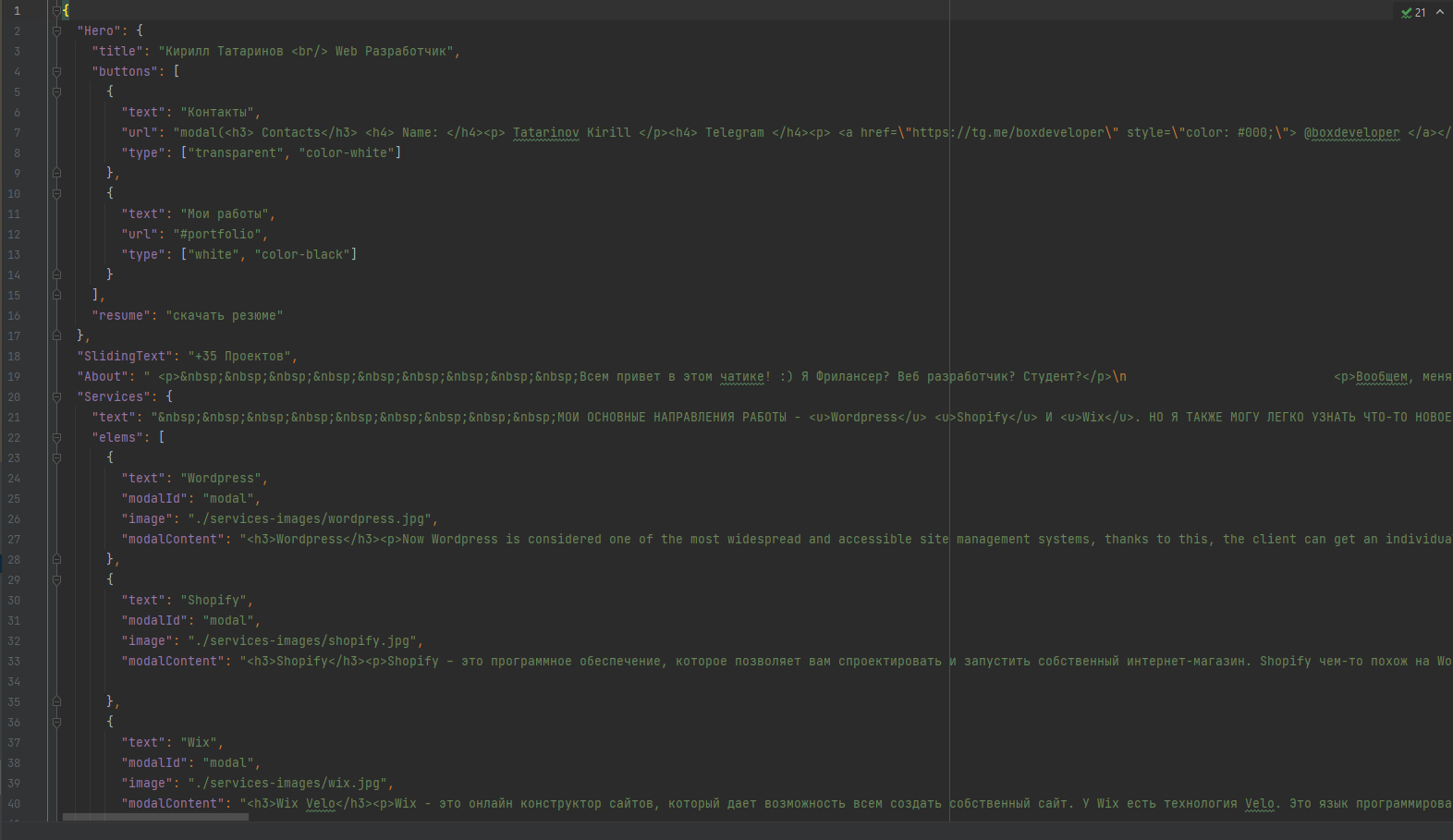


Рисунок 14 – Русский перевод сайта

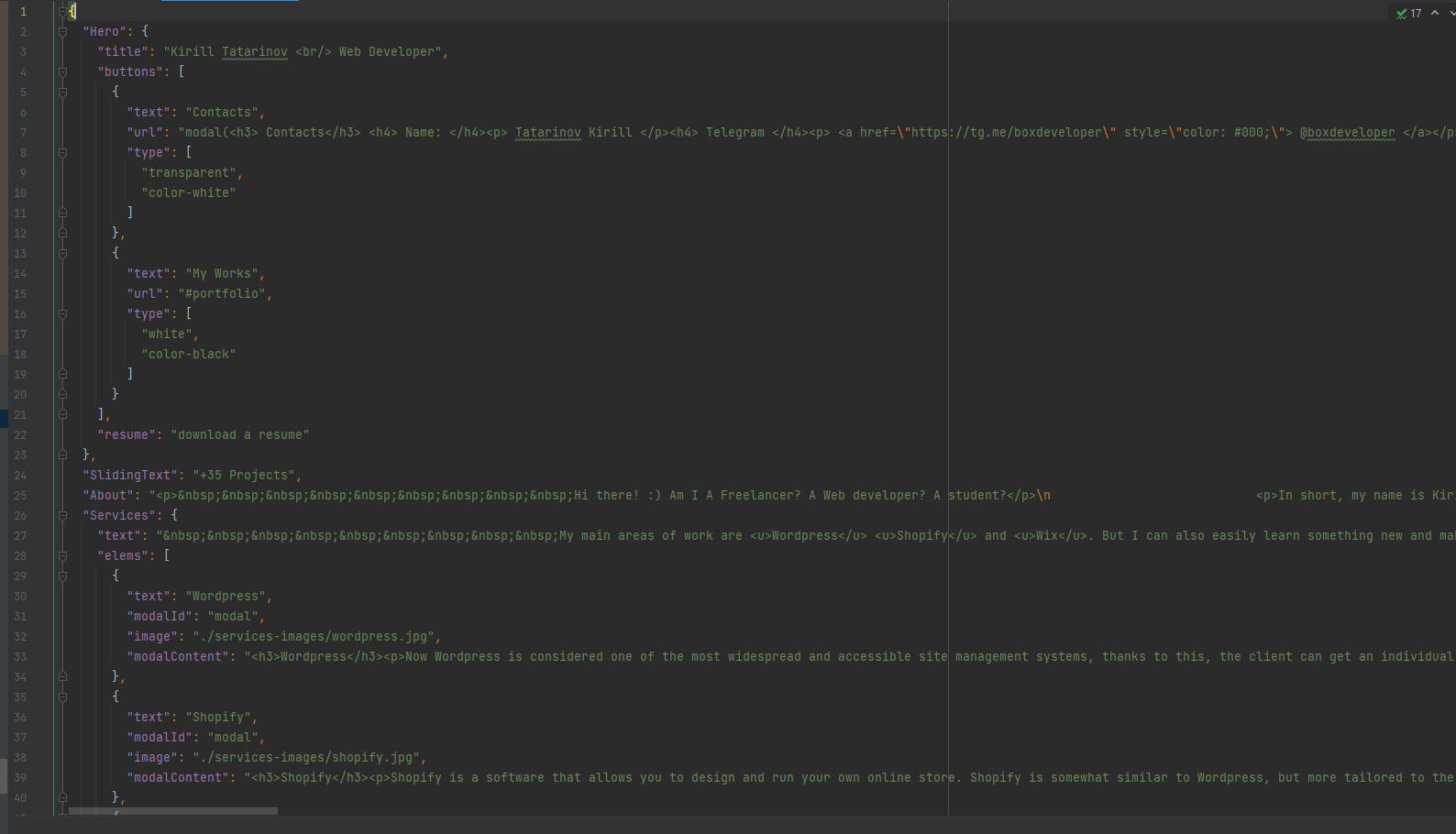


Рисунок 15 – Английский перевод сайта

Для изменения текущего языка в блоке header используется функцию changeLanguage (см. рисунок 16)

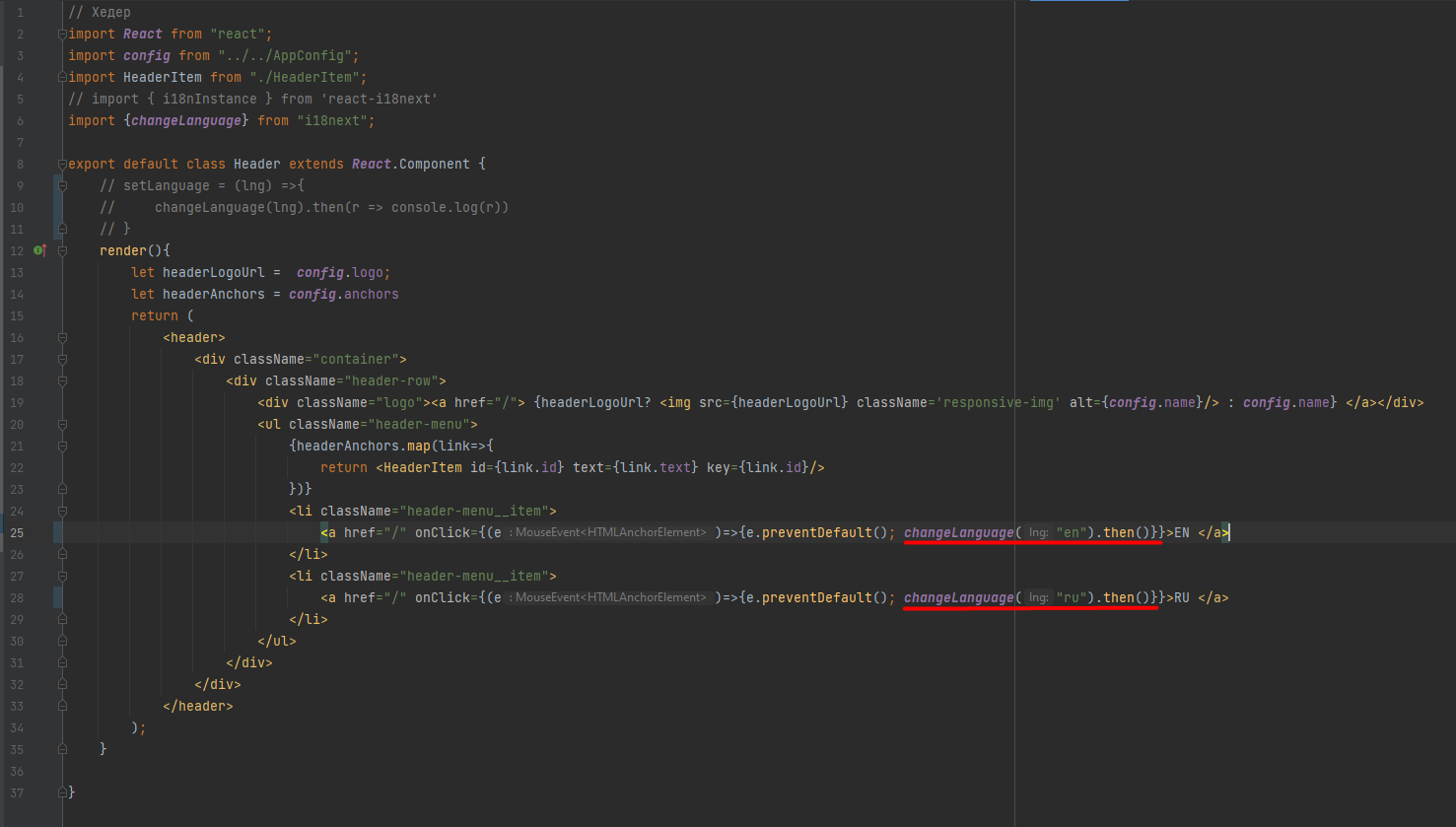


Рисунок 16 – Изменения языка в хедере

# 4.2.4. Проекты (портфолио)

Само портфолио отображается в компоненте Projects.js (см. рисунок 17), для каждого элемента портфолио имеется собственный компонент ProjectsItem.js (см. рисунок 18).

В файле Projects.js (см. рисунок 17) мы через цикл проходим по всем элементам проектов, которые хранятся в Json (см. рисунок 19). Проекты было выбрано хранить в Json, для легкости редактирования списка проектов

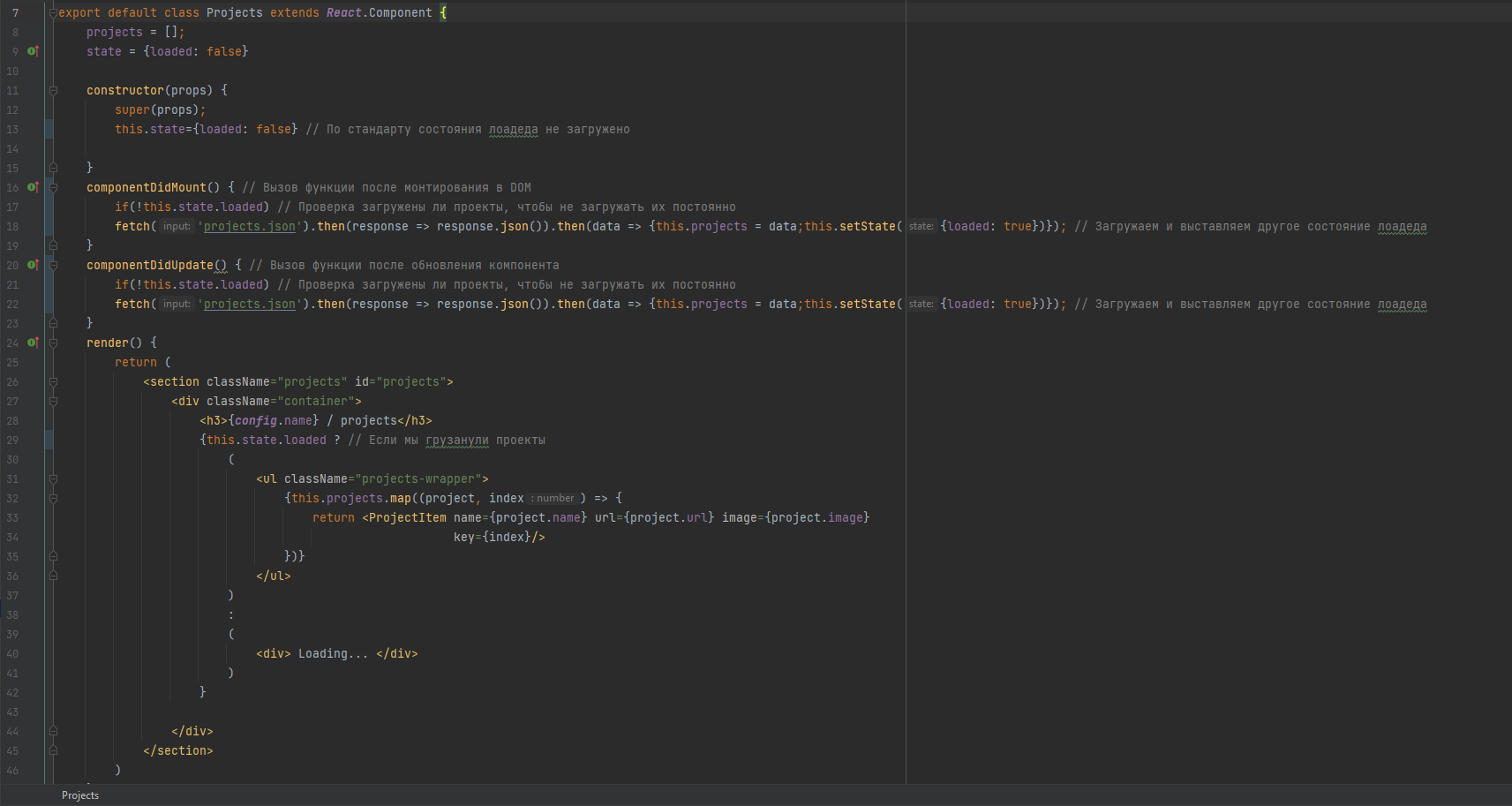


Рисунок 17 – Блок проектов. Вывод проектов через цикл

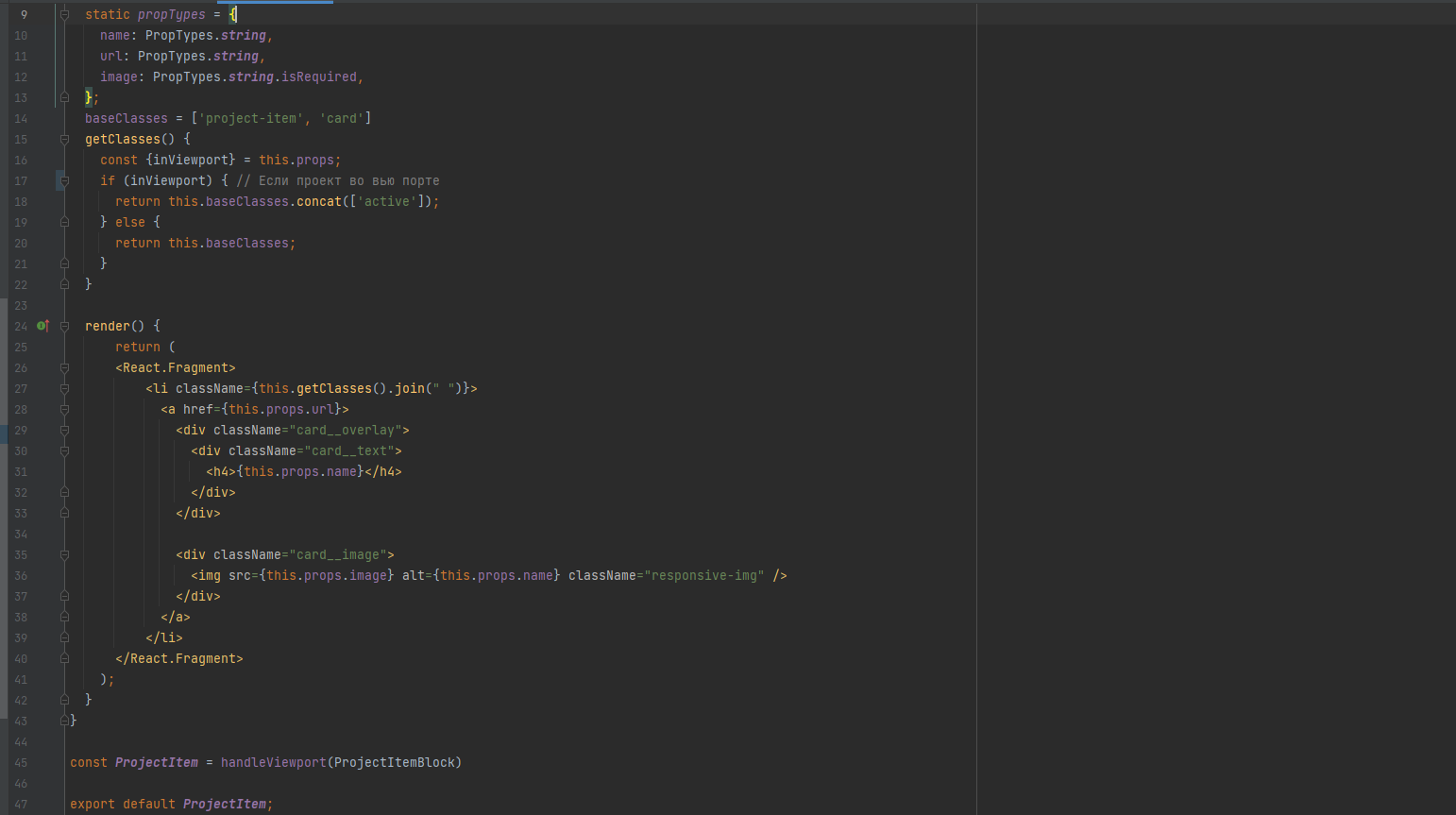


Рисунок 18 – Карточка проекта. Выводим информацию переданную из json

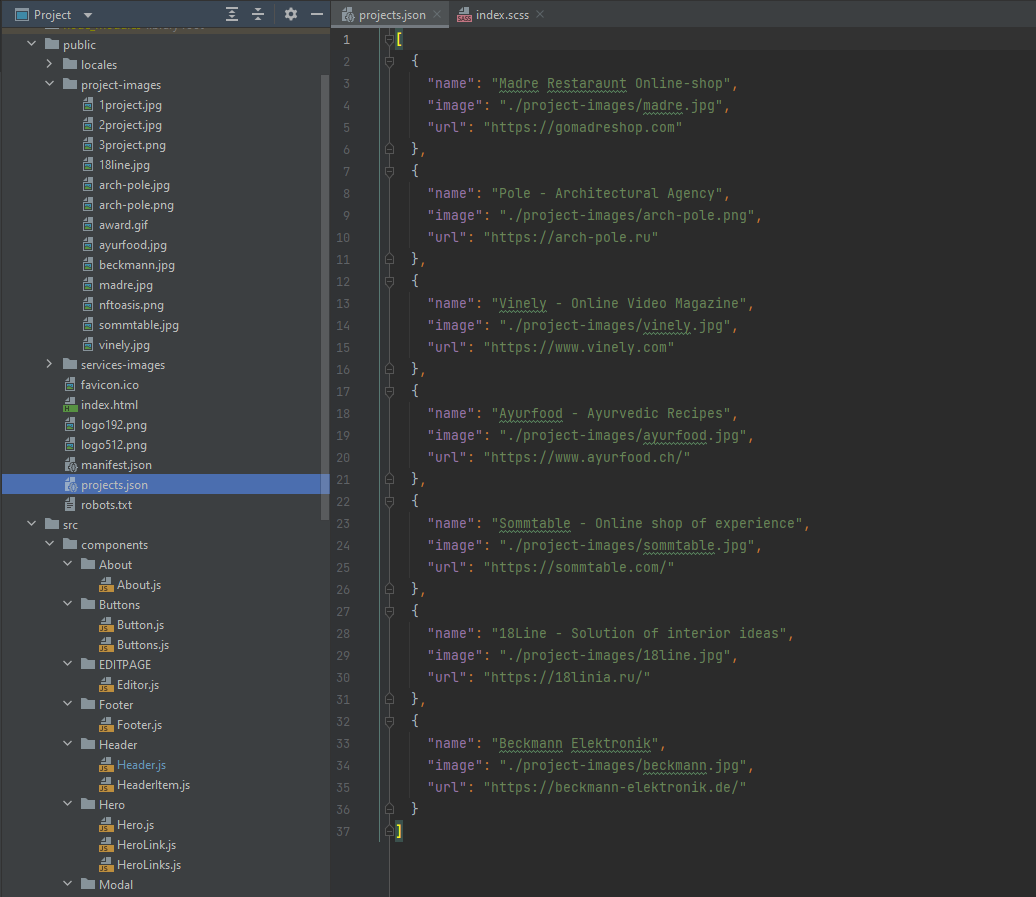


Рисунок 19 – Портфолио в json

# 4.2.5. Валидация типов

В компонентах, которые принимают JSX-атрибуты может возникнуть ситуация в которой в атрибуте передается не те данные, которые ожидались.

Например, в блоке портфолио (см. главу 4.2.4), так как блок заполняется через json, может возникнуть ситуация, что вместо ссылки на изображение будет передан объект изображения, код такого не ожидает и может крашнуться. Или например, в блоке “Hero” (см. главу 4.2.2) ожидается, что ссылки будут переданы как массив и никакой другой тип не будет правильно воспринят кодом, поэтому в каждом таком элементе был установлен валидатор propTypes (см. рисунок 20)

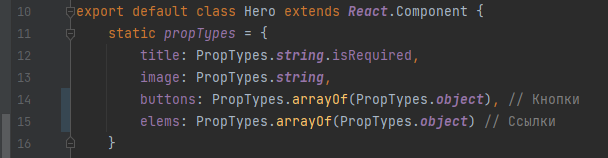


Рисунок 20 – Валидатор типов

# 4.2.6. Код страницы (Лендинг пейдж)

После настройки всех компонентов и модулей можем собрать нашу страницу с помощью React и пользовательских компонентов (см. рисунок 21)

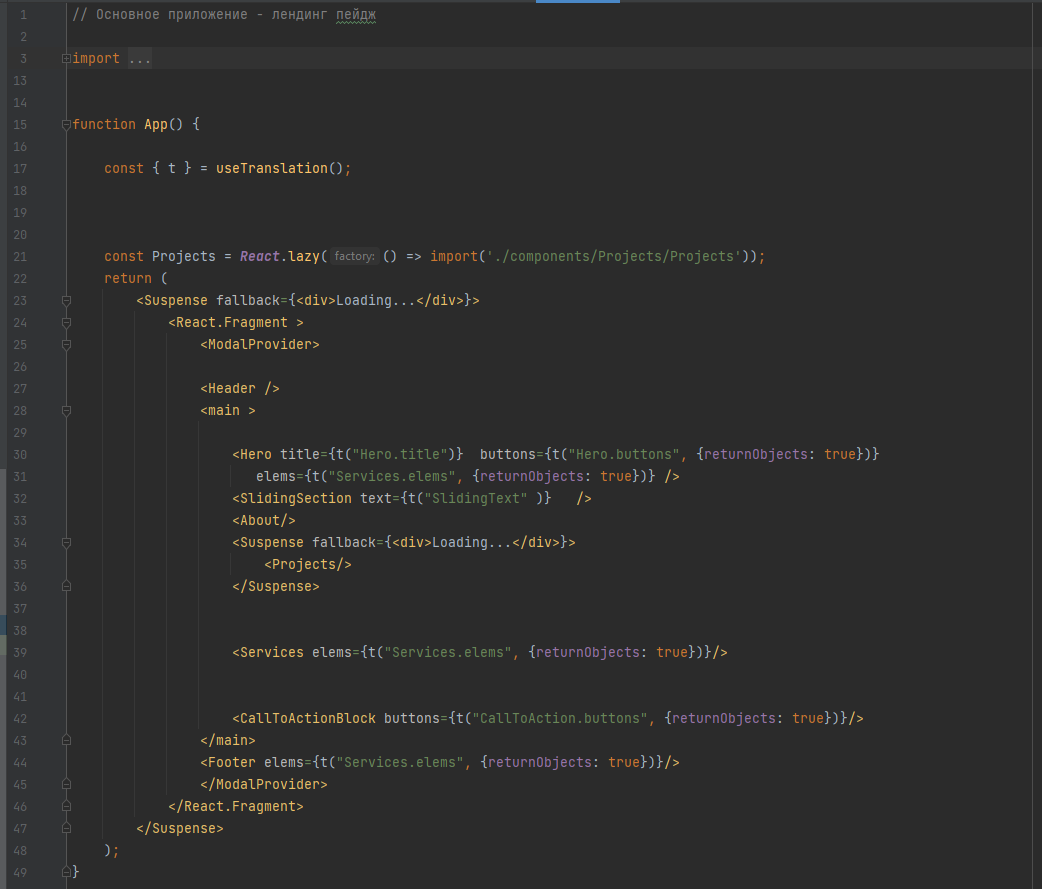


Рисунок 21 – Код страницы

# 4.2.7. Система отслеживания версий. Github Pages

Весь данный проект выполнялся с использованием системы отслеживания версий Github (см. рисунок 22)

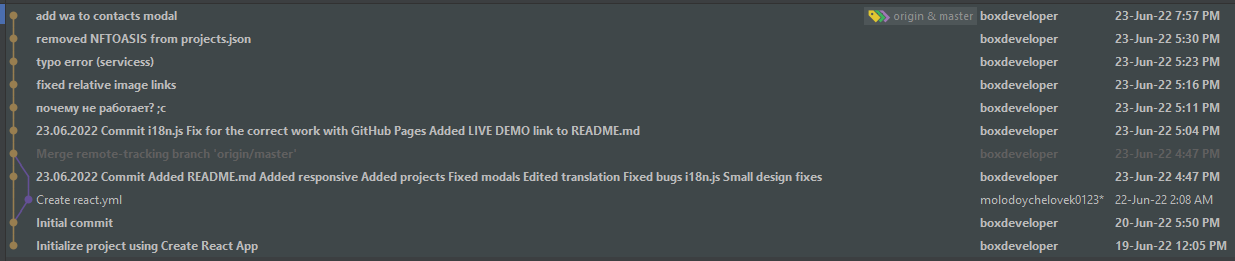


Рисунок 22 – Коммиты

Репозиторий находится в открытом доступе по ссылке: <https://github.com/molodoychelovek0123/portfolio-rjs>. Также через систему Github Pages была создана на собранный проект, посмотреть готовый проект можно по ссылке: <https://molodoychelovek0123.github.io/portfolio-rjs/>

# 5. Результат индивидуальной работы в ходе учебной практики

Результатом работы в ходе учебной практике является программный продукт: Сайт-портфолио. На рисунках 23 – 28 представлены скриншоты выполненного сайта, посмотреть готовый проект можно по ссылке: <https://molodoychelovek0123.github.io/portfolio-rjs/>

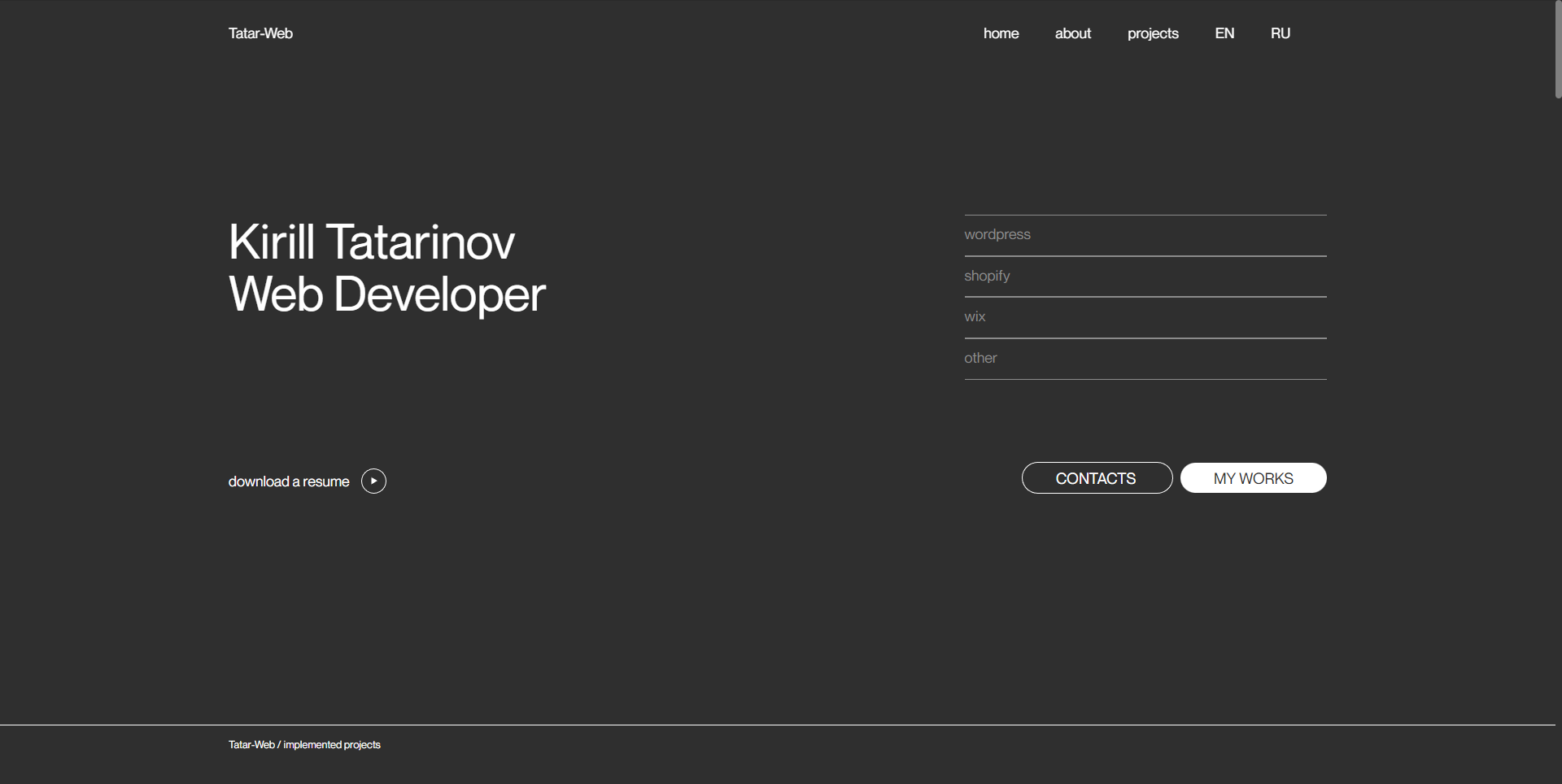


Рисунок 23 – “Hero” блок

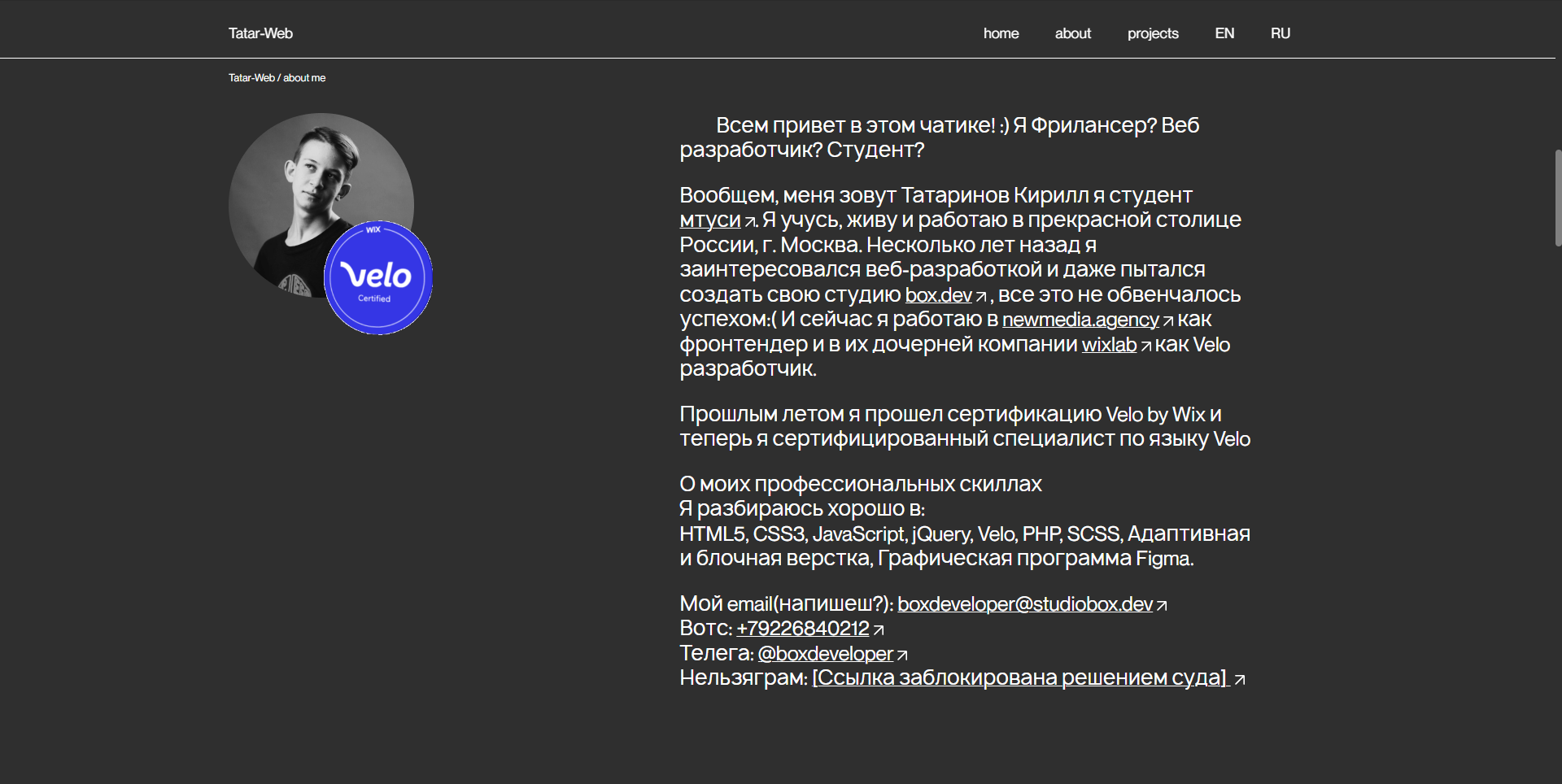


Рисунок 24 – Блок “Обо мне”

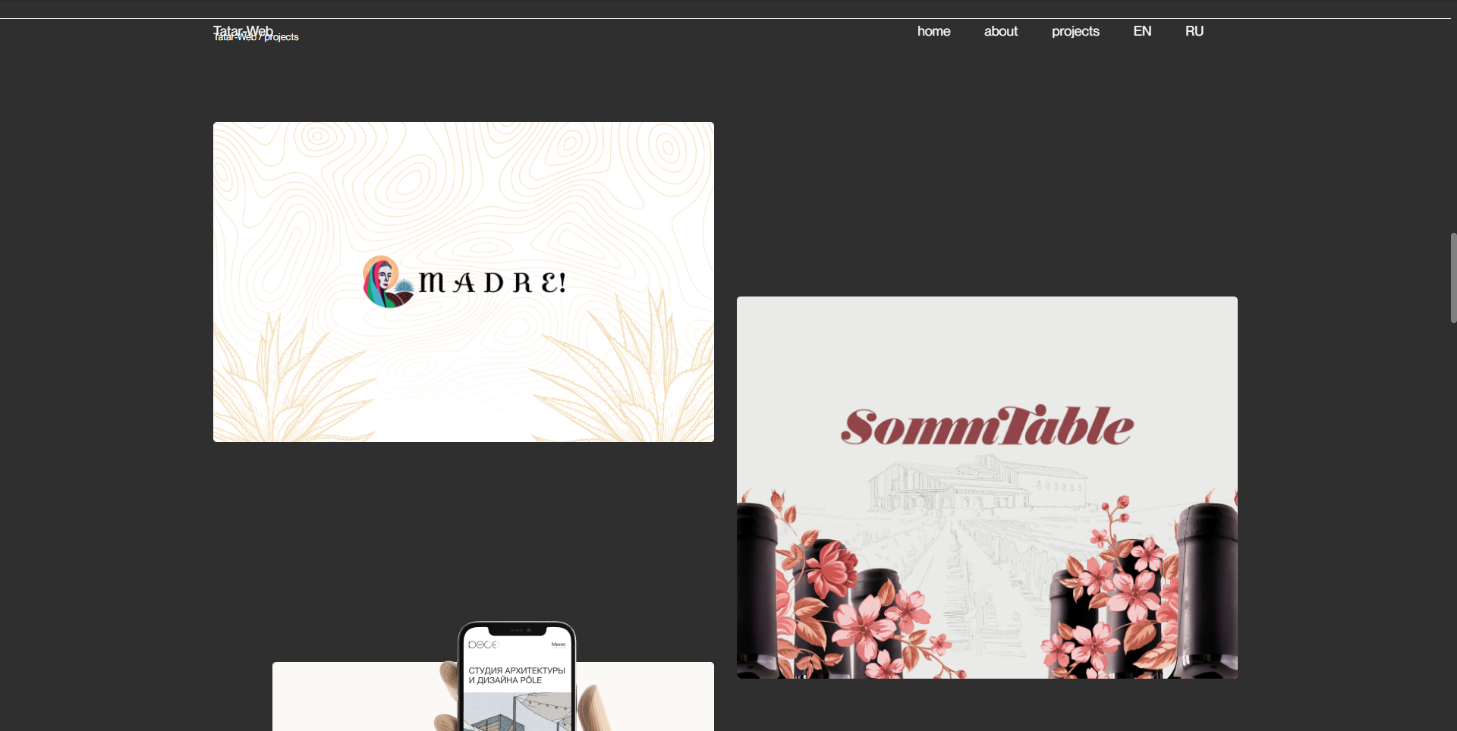


Рисунок 25 – Секция портфолио. Список выполненных проектов

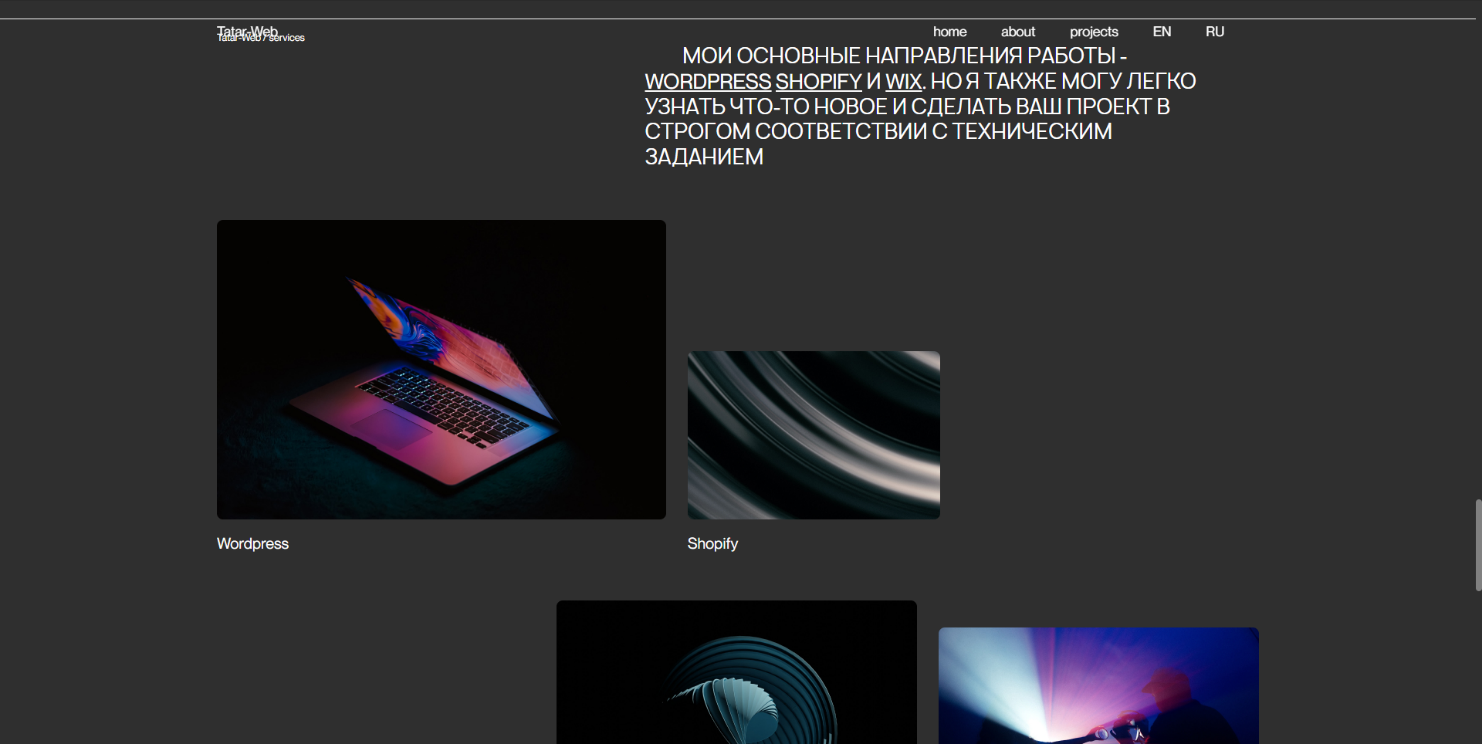


Рисунок 26 – Секция “Услуги”

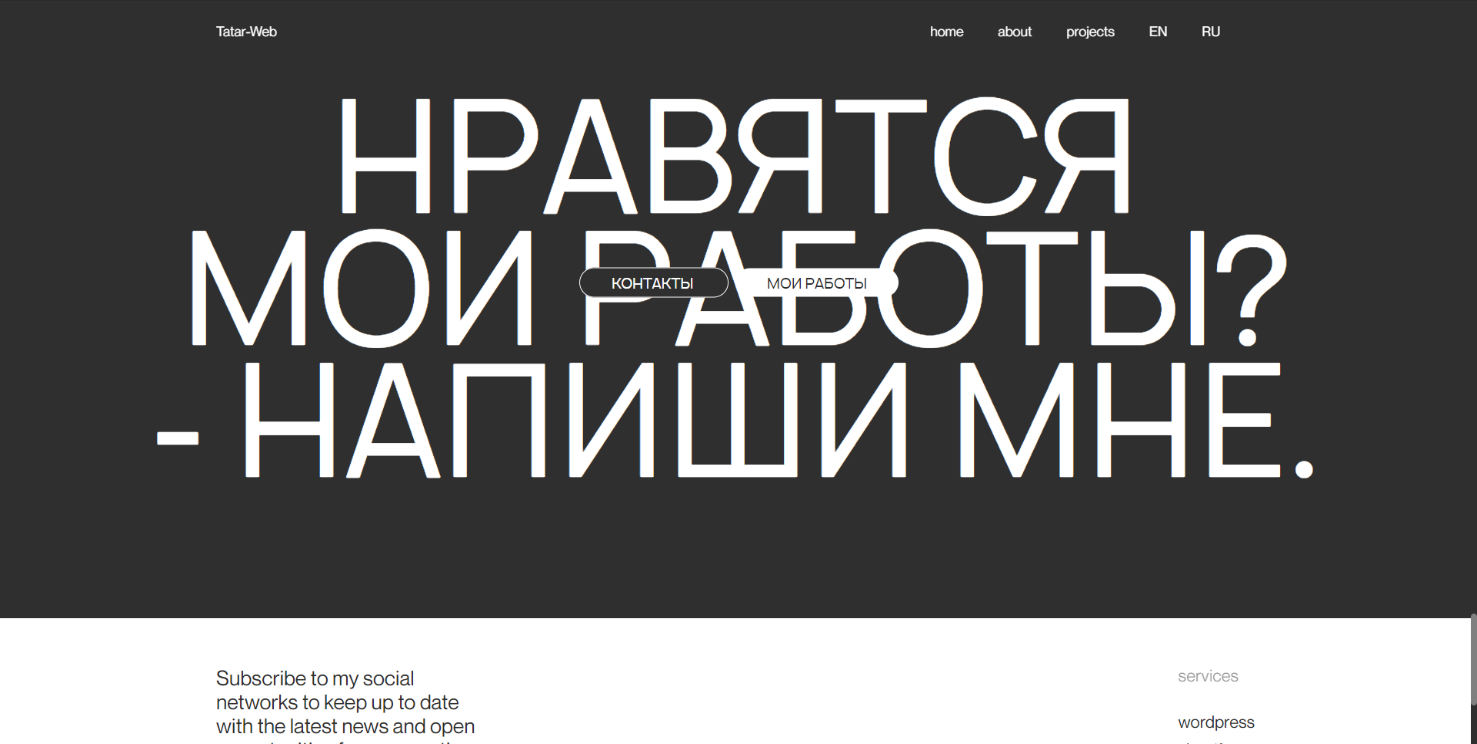


Рисунок 27 – “Call To Action” блок и Футер

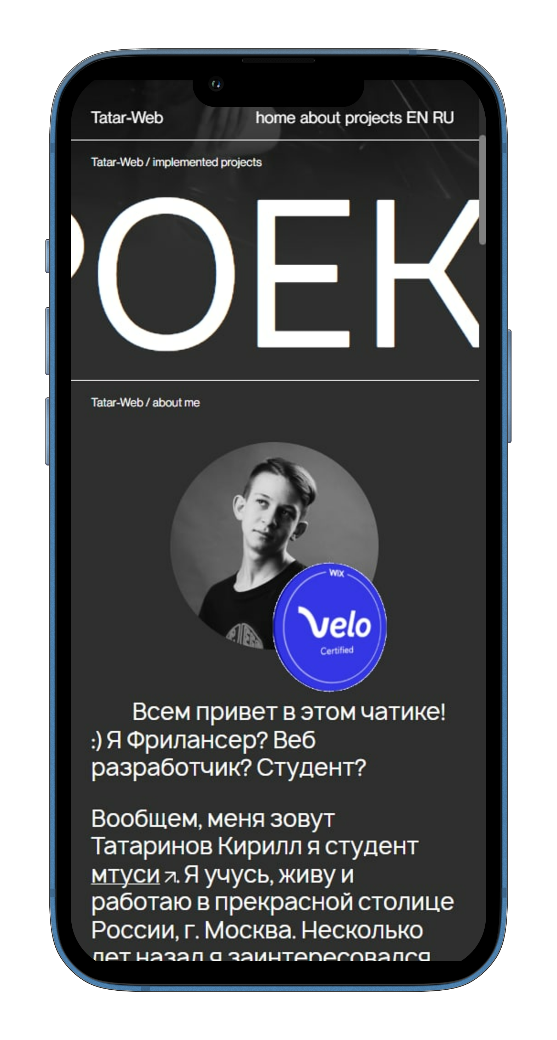


Рисунок 28 – Адаптивная версия сайта

# 6. Заключение

В итоге моей деятельности были выполнены поставленные задачи и разработан Сайт-портфолио.

Репозиторий проекта: <https://github.com/molodoychelovek0123/portfolio-rjs>

Посмотреть готовый проект можно по ссылке: <https://molodoychelovek0123.github.io/portfolio-rjs/>