Geekbrains

**Разработка портфолио в виде web-страницы (Landing page) – для эффективного поиска работы**

Программа: Разработчик

Специализация: Fullstack

Шестаков Роман Витальевич

г. Тюмень

2025 год

**Содержание**

Введение ( 1 стр.)*.*

Глава 1. Общие сведения ( 2 стр.)*.*

1.1 Цели. Концепции

1.2 Проблемы, типовых портфолио

1.3 Обоснование и выводы

Глава 2. Техническое задание ( 2 стр.)

2.1. Назначение сайта

2.2. Требования к организации языковых версий сайта

2.3. Требования к верстке сайта

2.4. Способы навигации по сайту

2.5. Описание требований к элементам страницы

2.6. Загрузка новых проектов

Глава 3. Работа с макетом сайта ( 8 стр.)

3.1. Описание редактора

3.2. Подготовка макета

3.3. Выгрузка контента в рабочую директорию

Глава 4. Разработка приложения (38 стр.)

4.1. Выбор среды разработки

4.2. Создание сборщика проекта

4.3. Создание системы контроля версий Git

4.4. Создание файловой структуры

4.5. Описание сборки gulpfile.js

4.6. Верстка

4.7. Подготовка сайта для загрузки на хостинг

4.8. Загрузка проекта на хостинг

4.7. Тестирование

Заключение ( 1 стр.)

Список используемой литературы ( 1 стр.)

Приложения ( 1 стр.)

**ВВЕДЕНИЕ**

**Тема проекта:** Разработка web-портфолио в виде Landing page – для эффективного поиска работы

**Цель:** Создать web-портфолио в котором можно эффективно рассказать о себе. С возможностью демонстрации выполненных работ (проектов). Быстрая загрузка проекта в портфолио (пополнение коллекции работ и навыков). Разработчик всегда на связи.

**Какую проблему решает:** Качественная презентация навыков разработчика – в любой момент времени и через любое устройство (*gadget*).

**Задачи:**

1. Составить техническое задание с описанием разделов и их содержанием.
2. Найти шаблон (дизайн) и доработать его - используя, онлаин-сервис Figma
3. Определиться с языками программирования – необходимыми для создания приложения
4. Изучить сборщики проектов – выбрать наиболее подходящий (подготовить его для создания проекта).
5. Поиск контента (фото, изображения, текст – наполнение разделов)
6. Подбор шрифтов – согласно макета
7. Верстка приложения
8. Мониторинг хостинг сервисов (цена, доступность интерфейса) – подбор оптимального
9. Backend – разработка и загрузка приложения на хостинг

**Инструменты:** MicrosoftWord, VSCode, Gulp, Figma, GitHub, Scss, Css, Html, Js, Php, Json, GoogleFonts, Icomoon, Node.js, Npm, W3, Beget, Shadow-generate.

# Глава 1. Общие сведения

* 1. **Цели. Концепции**

Целями проекта является быстрое и качественное донесение информации о компетентности и опыте разработчика в удобном формате для потенциального клиента или работодателя.

К концепциям относится создание web-портфолио в виде Landing page, с описанием ключевых навыков, возможностью демонстрации выполненных работ (проектов) и быстрой связью с разработчиком.

О терминах:

**Портфолио** — это комплект наглядных образцов и кейсов, которые кандидат собрал за время работы.

Главная задача портфолио — показать реальные навыки, наработанный опыт и квалификацию кандидата.

Оно может понадобиться в разных ситуациях, например:

* при трудоустройстве на работу
* для участия в конкурсах и грантах
* для привлечения клиентов и инвесторов

Существует несколько основных видов портфолио, каждый из которых подходит для определённой сферы деятельности:

* Творческое портфолио — используется художниками, дизайнерами, фотографами, писателями. Оно включает в себя примеры работ, иллюстрации, отрывки из текстов и так далее.
* Профессиональное портфолио — чаще всего готовится для выпускников вузов и специалистов различных отраслей. Оно содержит дипломы, сертификаты, отзывы работодателей и примеры проектов.
* Ученическое портфолио — предназначено для школьников и студентов. Оно может включать в себя результаты тестов, рефераты, грамоты и дипломы

**Landing page** «посадочная страница» — это одностраничный сайт с краткой информацией о товаре, услуге или мероприятии. Его задача — превращать посетителей в клиентов.

Задача лендинга — на одной странице коротко рассказать о продукте и мотивировать пользователя совершить конкретное целевое действие:

* купить товар;
* подписаться на рассылку;
* оставить заявку на оказание услуги;
* зарегистрироваться на онлайн-мероприятие;
* забронировать билеты на концерт или выступление;
* получить предложение о сотрудничестве.

Пользователи переходят на лендинг по ссылкам с рекламных баннеров, постов из соцсетей, электронных писем или СМС.

**Web** (Всемирная паутина) — распределённая неоднородная компьютерная система коллективного пользования гипермедийными документами, действующая на базе сети Интернет.

* 1. **Проблемы, типовых портфолио**

Основной объем начинающих разработчиков сталкиваются с проблемой не эффективной коммуникацией с потенциальным клиентом или работодателем, портфолио - выполнено формате (doc, pdf), готовые работы (проекты), предоставлены в виде ссылок на репозиторий или web-сайты.

**Проблема** – для эффективной оценки, клиенту или работодателю нужно просмотреть, большой оббьем информации из различных источников (трата времени, клиента = низкая оценка!).

О терминах:

**Формат doc** — это формат файла, используемый Microsoft Word. Он содержит текст, изображения, графики, таблицы.

**Формат pdf** (Portable Document Format) это универсальный формат файлов, разработанный компанией Adobe Systems в 1993 году. Основная цель создания PDF — обеспечить надёжное и неизменное отображение документов на различных устройствах и платформах

**web-сайт** это одна или несколько логически связанных между собой **web** -страниц, объединённых под одним адресом (доменным именем) и воспринимаемых пользователем как единое целое.

**Репозиторий** (от англ. repository — хранилище) это место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети.

* 1. **Обоснование и выводы**
* Web-портфолио позволит быстрее и эффективнее привлечь клиента или работодателя.
* Каждый новый проект будет загружен в портфолио, что будет увеличивать потенциал разработчика.
* рост потенциала и экономия времени = прибыль!

# Глава 2. Техническое задание

**Техническое задание** (ТЗ) — это документ с подробным описанием требований заказчика к проекту. В нём указывают характеристики продукта, особенности задачи, дополнительные условия, сроки выполнения

**2.1. Назначение сайта**

* официальный одностраничный сайт-портфолио.

К основным функциям сайта относятся:

* быстрая навигация по разделам;
* просмотр ленты проектов;
* переход по ссылке в проект и обратно на сайт;
* рассказать о разработчике;
* обратная связь с разработчиком;
* быстрая смена языка Ru - En;

Сайт должен быть оптимизирован для корректного отображения в браузерах Google Chrome, Opera, Mozilla Firefox, Яндекс.Браузер, MicrosoftEdge, AppleSafari.

Разработка сайта должна осуществляться при помощи следующих технологий: HTML, CSS, Java-Script, PHP.

**2.2.** **Требования к организации языковых версий сайта**

* Язык сайта: русский, английский.
* Оформление и содержание страниц иноязычного сайта аналогично русскоязычному варианту.
* На страницах иноязычного сайта размещается контент, предоставленный Заказчиком.
* В случае не предоставления необходимой информации для каких-либо блоков страницы сайта оформляются без учета данных блоков.
* Смена языка должна осуществляться без перезагрузки страницы.

**2.3.** **Требования к верстке сайта**

Верстка выполняется под браузеры Microsoft Edge, Mozilla Firefox, Google Chrome, Apple Safari и Opera последних официально выпущенных версий на момент подписания Приложения на этапе верстки, для платформ ПК и MAC. Для устаревших браузеров должно выводиться сообщение, что корректная работа возможна на браузерах, указанных выше.

Верстка выполняется только под 100% масштаб браузера и 100% масштаб операционной системы. Адаптивная верстка должна быть разработана для двух точек излома:

* Смартфоны, вертикальное расположение — 360 px (и до 767 px);
* Планшеты, вертикальное расположение — 768 px (и до 1279 px);
* Персональные компьютеры — на разрешения 1280 пикселей и более используется текущая (ПК-версия) версия макета.
* При открытии файлов верстки на устройствах, ширина экрана которых меньше 1280 пикселей, применяется соответствующая адаптивная версия дизайна. При больших разрешениях применяется основной вариант дизайна сайта. На мобильных устройствах, c целью оптимизации производительности, допускается упрощение верстки, в том числе — изменение размеров изображения, отключение или упрощение анимационных эффектов, видео, скриптов.

Для браузеров, в которых отсутствует поддержка стандартов CSS3, допускается незначительное отличие от макетов, вызванное неполной поддержкой CSS3 с сохранением функциональности.

Под корректной версткой понимается соответствие страниц сайта согласованным макетам в масштабе 100% с учетом особенностей отображения HTML-разметки в операционной системе пользователя, в том числе:

* Отображение и сглаживание шрифтов;
* Масштабирование изображений браузером;
* Пользовательские настройки браузера (шрифты, размеры шрифтов по умолчанию и т.п.).

Верстка должна реализовывать анимацию или иные эффекты, описание которых приведено в настоящем техническом задании.

**2.4.** **Способы навигации по сайту**

Сайт должен обладать понятной и удобной системой навигации.

Вверху страницы отображаются облегченная навигационная панель, которая обеспечивает переход к основным пунктам меню сайта (Главная, Новости и т.д.) и обратно.

**2.5.** **Описание требований к элементам страницы** (шапка, контентная часть, футер, мини-карточки, кнопки и т.д.).

**Шапка страницы (header)** – верхняя часть страницы, визуально отделенная от основного контента, содержащая навигаторы и графические элементы оформления.

Шапка страницы содержит:

* перечисление разделов – (Галавная, Обо мне, Знания, Мои работы, Контакты)
* при нажатии происходит плавный переход в раздел по якорной ссылке
* шапка фиксируется при прокрутке
* при размере экрана 425px разделы перемещаются в меню - бургер
* активный раздел выделен

Следующий раздел (под шапкой) страницы содержит:

* элемент переключателя языковой версии RU/EN;
* фото разработчика
* краткая информация

**Контентная часть (main) *–*** середина страницы

* слайдер – перечисление инструментов и навыков
* карточки с описанием и ссылками на готовые проекты (показываем 3 последних)
* в карточке краткое описание проекта + возможность быстрого просмотра основных разделов.
* кнопка – показывает более поздние проекты

**Подвал страницы (footer)** – нижняя часть страницы, визуально отделенная от основного контента.

Подвал страницы содержит:

* телефон;
* иконки социальных сетей (при нажатии – переход в соцсети)
* кнопка – Напишите на почту (форма отправки письма)

**2.6. Загрузка новых проектов –** папка с новым проектом добавляется на хостинг в директорию pages. Данные из pages выгружаются на сайт в виде карточек проекта.

**Хостинг** – (англ. hosting) — услуга по предоставлению вычислительных мощностей для размещения информации на сервере, постоянно находящемся в сети (обычно Интернет). Хостингом также называется услуга по размещению оборудования клиента на территории провайдера с обеспечением подключения его к каналам связи с высокой пропускной способностью (колокация, от англ. collocation).Обычно хостинг входит в пакет по обслуживанию сайта и подразумевает как минимум услугу размещения файлов сайта на сервере, на котором запущено ПО, необходимое для обработки запросов к этим файлам (веб-сервер). Как правило, в обслуживание уже входит предоставление места для почтовой корреспонденции, баз данных, DNS, файлового хранилища на специально выделенном файл-сервере и т. п., а также поддержка функционирования соответствующих сервисов.

**Директории** (файловая система) — механизм организации файлов в файловой системе.

# Глава 3. Работа с макетом

**Макет** — модель объекта в уменьшенном масштабе или в натуральную величину, лишённая функциональности представляемого объекта

**3.1.** **Описание редактора**

Для подготовки макета web-порфолио – используем графический редактор Figma.

**Figma** — это онлайн-графический редактор для создания дизайна интерфейсов, прототипирования и работы с векторными изображениями.

Программа работает прямо в браузере и не требует установки на компьютер, что делает её доступной с любого устройства. Figma также предлагает desktop версию для пользователей, предпочитающих работать с локальным приложением.

Некоторые возможности сервиса:

* рисование и редактирование векторных форм (линии, прямоугольники, эллипсы, полигоны и т. д.);
* инструменты для работы с текстом, включая шрифты, стили и параметры форматирования;
* создание прототипов с интерактивными переходами между экранами и элементами;
* функционал для работы со слоями, группами и фреймами, что упрощает организацию контента;
* встроенная библиотека компонентов, позволяющая быстро добавлять и настраивать стандартные элементы интерфейса;
* инструменты для совместной работы в реальном времени, включая комментирование, редактирование и обсуждение проектов внутри команды;
* интеграция с другими приложениями, такими как Slack, Jira, и другими, для удобства управления проектами.

**Плюсом Figma -** является ее большая популярность, поэтому существует много сообществ. Большой выбор специалистов и готовых работ.

**3.2. Подготовка макета**

Макет выбран из готовых работ, согласно - Технического задания. Макет находится в открытом доступе и предоставляется бесплатно.



Рисунок 1. Макет web-приложения

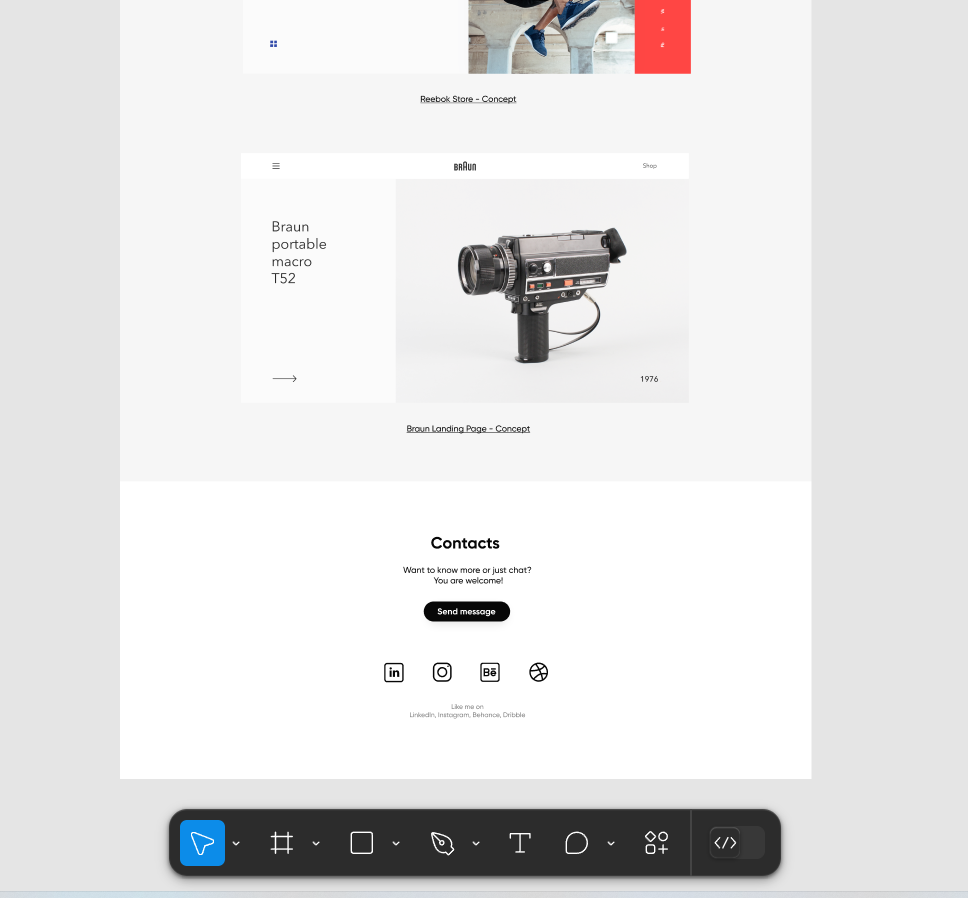


Рисунок 2. Макет web-приложения

Создаем интерактивный макет изображения для карточки проекта.

Изображение создается путем загрузки в figma - скриншота страницы проекта адаптированных под различные устройства (телефон, планшет, компьютер)

**Скриншот** - это снимок (фотография) содержимого экрана на компьютере, телефоне или планшете. Он фиксирует визуальную информацию, создавая изображение текста, графики или любых элементов, присутствующих на дисплее.

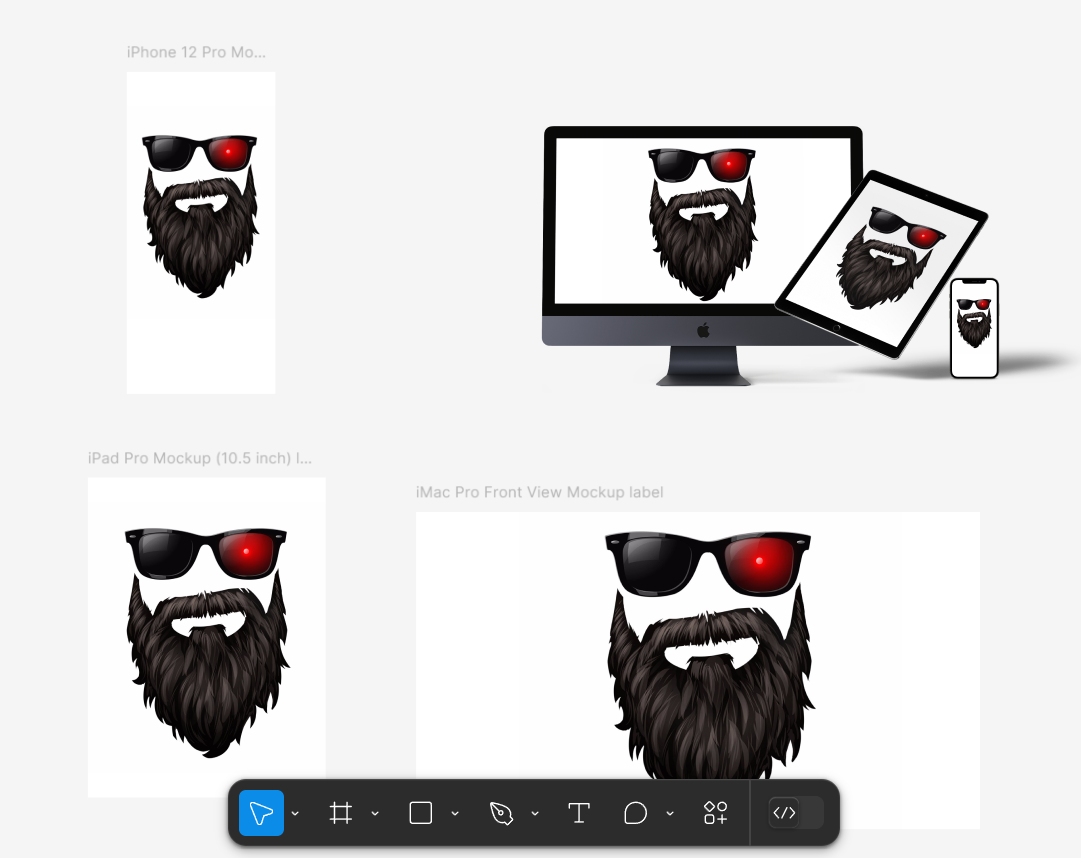


Рисунок 3. Макет preview

**3.3. Выгрузка контента в рабочую директорию**

**Контент** - содержание, которое распространяется через сайты, литературу, музыку и видео. Это любая информация, которую пользователь может посмотреть, почитать или послушать на компьютере или телефоне. Контентом могут быть статьи, рисунки, фильмы, видеоролики, музыка, подкасты и многое другое.

**Директория** - каталог или папка, специальное место на компьютерном носителе информации, в котором хранятся имена файлов и сведения об этих файлах (размер файлов, время их последнего обновления, свойства файлов и т.п.).

Создаем основную директорию проекта – portfolio.

В portfolio создаем директорию source (src) – для храниния исходных файлов.

В src создаем:

* директорию image (img) – в которой сохраняем все картинки и иконки из макета Figma.

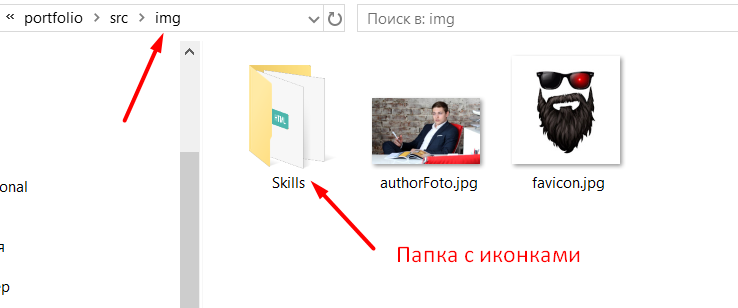


Рисунок 4. Папка с изображениями

* директорию fonts – сохраняем шрифты, согласно макета Figma. Шрифты скачиваем из сервиса fonts-online.ru

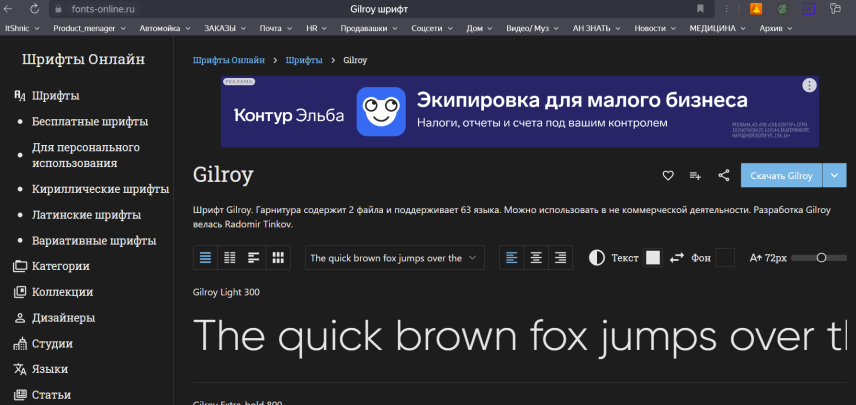


Рисунок 5. Сайт для скачивания шрифтов

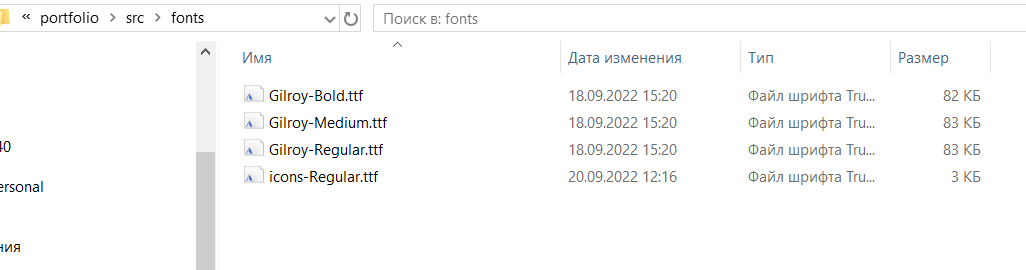


Рисунок 6. Скачанные шрифты в папке

# Глава 4. Разработка приложения

**4.1. Выбор среды разработки**

**IDE** (Integrated Development Environment) — это набор программных инструментов, которые используются для создания ПО.

Среда разработки состоит из четырех компонентов:

* текстовый редактор, который позволяет писать код внутри среды;
* отладчик, который ищет ошибки в программе и ядрах операционной системы;
* транслятор, который может включать в себя компилятор и интерпретатор. Компилятор переводит написанный код в набор машинных команд, а интерпретатор исполняет код;
* средства автоматизации для высокой скорости разработки и сборки проекта.

Для чего нужны IDE?

IDE предназначены для упрощения разработки программы. Как правило, среды разработки имеют следующий функционал:

* писать, просматривать, запускать и править код внутри одного окна;
* тестировать написанный функционал, если среда позволяет эмулировать устройства;
* устанавливать новые компоненты;
* создавать и редактировать проекты, а также управлять ими;
* работать с системой контроля версий ПО;
* разрабатывать графический интерфейс, если интегрированная среда включает в себя визуальный редактор.

Какую IDE выбрать?

При выборе интегрированной среды разработки стоит опираться на следующие критерии:

* Совместимость с вашей платформой. Как правило, IDE совместимы с большинством популярных операционных систем. Однако некоторые среды разработки могут работать только на определенных ОС.
* Простота работы. Этот параметр зависит от вашего удобства: рекомендуем протестировать несколько решений и выбрать наиболее оптимальное.
* Поддерживаемые языки. При выборе среды разработки проверьте, поддерживает ли она язык вашего проекта.
* Стоимость. Существуют бесплатные, условно-бесплатные и платные варианты IDE.

Существует множество IDE:

* Eclipse это свободно распространяемая среда разработки, которая обладает гибким функционалом: к примеру, она включает в себя инструменты отладки и поддержки Git/CVS
* IntelliJ IDEA это среда разработки с бесплатной версией и тестовым периодом для платной лицензии на 30 дней.
* PyCharm это среда разработки, которая распространяется с платной и бесплатной лицензией. Существует бесплатная версия этой IDE, но она предназначена только для Python.
* Visual Studio Code (VS Code) это текстовый редактор кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.

Для разработки приложения, планирую использовать несколько языков программирования:

* Html язык гипертекстовой разметки (Hyper Text Markup Language), позволяет готовить веб-страницу с помощью простых редакторов.
* CSS (англ. Cascading Style Sheets — каскадные таблицы стилей), формальный язык описания внешнего вида документа, написанного с использованием языка разметки.
* PHP (англ. PHP: Hyper text Preprocessor — «PHP: препроцессор гипертекста») скриптовый язык программирования общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг- провайдеров и является одним из лидеров среди языков программирования, применяющихся для создания динамических веб-сайтов.
* JavaScript это высокоуровневый, интерпретируемый язык программирования, используемый в основном для создания интерактивных веб-страниц.

Он позволяет создавать богатые интерактивные пользовательские интерфейсы, обрабатывать события, управлять мультимедийными ресурсами, выполнять валидацию форм и работать с данными в реальном времени.

Некоторые области применения JavaScript:

* Веб-сайты и веб-приложения. Практически на каждом современном сайте используют код, написанный на JS.
* Расширения для браузера. Небольшие простые скрипты, которые добавляют дополнительный функционал — блокируют рекламу, позволяют сохранять аудио, отправляют уведомления о новых письмах или меняют цветовую схему сайта.
* Мобильные приложения. Например, интерфейс для работы с облачным хранилищем, его можно написать на JavaScript и собрать в приложение с помощью специальных инструментов.
* Серверная часть сайтов и программ. Язык программирования JavaScript можно использовать для написания любых сервисов: чатов, компьютерных программ и даже нейросетей. Для этого к нему нужно подключить движок Node.js.
* Игры. На JS можно писать несложные браузерные игры.

Visual Studio Code (VS Code) - наиболее подходящий редактор для написания web-портфолио, так-как:

* обладает более дружелюбным интерфейсом
* позволяет работать с различными языками программирования
* имеет бесплатную версию

Для ускорения процесса разработки необходимо установить plug-in для VS Code:

# Auto Close Tag автоматическое закрытие тега

# eCSStractor для извлечения имен классов из HTML и создания таблицы стилей CSS

# Path Autocomplete автоматическое заполнение пути к файлу

# Prettier обеспечивает единообразие стиля, анализируя ваш код и перепечатывая его по своим собственным правилам, которые учитывают максимальную длину строки, при необходимости перенося код в другое место.

* Russian - Code Spell Checker проверка орфографии
* VsСode Google Translate быстрый перевод

**Plug-in** (англ. plug-in, от plug in «подключать») — независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей. Плагины обычно выполняются в виде разделяемых библиотек.

**4.2. Создание сборщика проекта**

**Сборщики проекта** (бандлеры) — это инструменты, которые помогают объединять и управлять модулями и зависимостями в проекте. Они особенно полезны в больших проектах, где есть множество файлов JavaScript, стилей и других ресурсов.

Некоторые задачи сборщиков:

* Объединение файлов. Сборщики позволяют объединять несколько файлов в один или несколько бандлов. Это уменьшает количество запросов к серверу и повышает производительность загрузки страницы.
* Разрешение зависимостей. Сборщики позволяют импортировать и использовать модули и зависимости в проекте. Они автоматически определяют зависимости модулей и разрешают их, чтобы модули могли взаимодействовать друг с другом.
* Трансформация файлов. Сборщики позволяют применять различные трансформации к файлам, например, компиляцию препроцессоров CSS и JavaScript (например, Sass, Babel) или оптимизацию и минификацию файлов для улучшения производительности.
* Управление ресурсами. Сборщики также позволяют управлять и подключать другие ресурсы, такие как изображения, шрифты, видео и другие файлы, обеспечивая их оптимизацию и интеграцию в проект.

Некоторые популярные сборщики проектов: Gulp, Webpack, Parcel, Rollup и Browserify

Для сборки проекта применяем **Gulp** это таск-менеджер для автоматического выполнения часто используемых задач при разработке приложений с использованием платформы Node.js. Он позволяет, например, формировать файлы стилей, объединять и унифицировать файлы, оптимизировать изображения, транспилировать код.

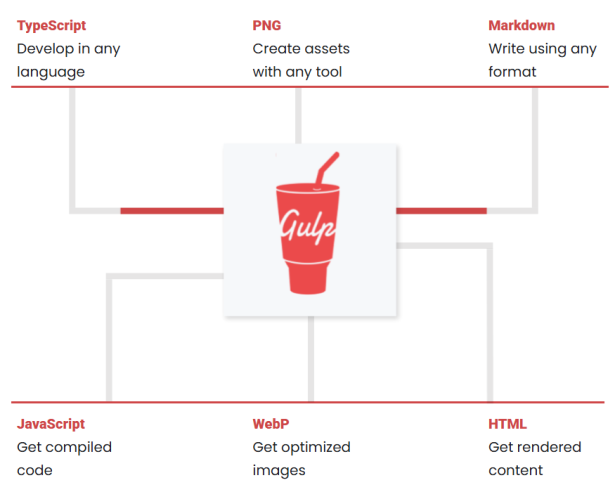


Рисунок 7. Gulp

Рисунок 6. Скачанные шрифты в папке

**Node.js** это бесплатная кросс-платформенная среда выполнения JavaScript с открытым исходным кодом. Она создана на движке V8 от Google, который используется для преобразования JavaScript в компьютерный код.

**NPM** (Node Package Manager) — менеджер пакетов для программной платформы Node.js. С его помощью можно устанавливать библиотеки, управлять зависимостями и редактировать метаданные проекта.

Создаем сборщик проекта:

* Скачиваем Node.js и устанавливаем на ПК

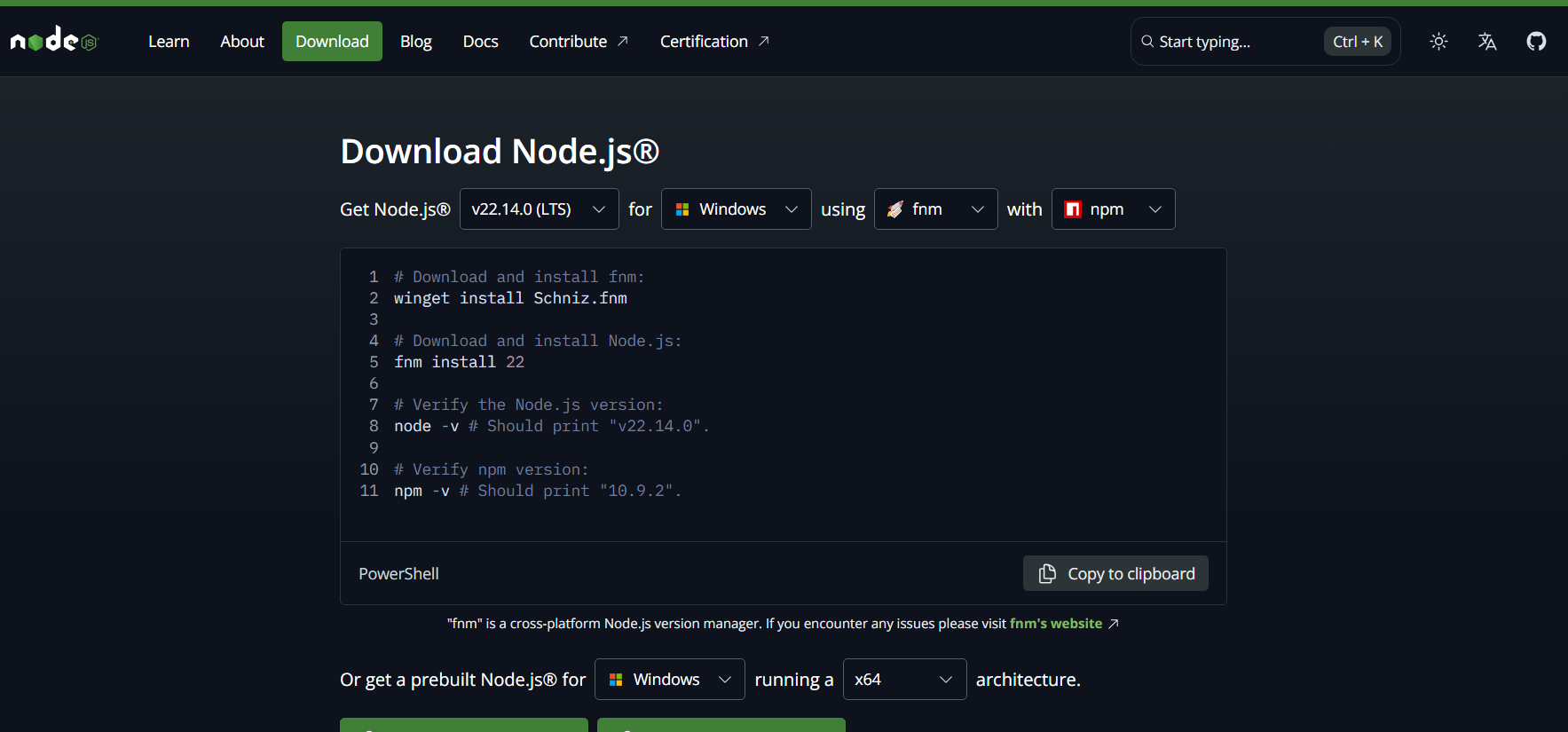


Рисунок 8. Установка node

* В VSCode открываем директорию portfolio

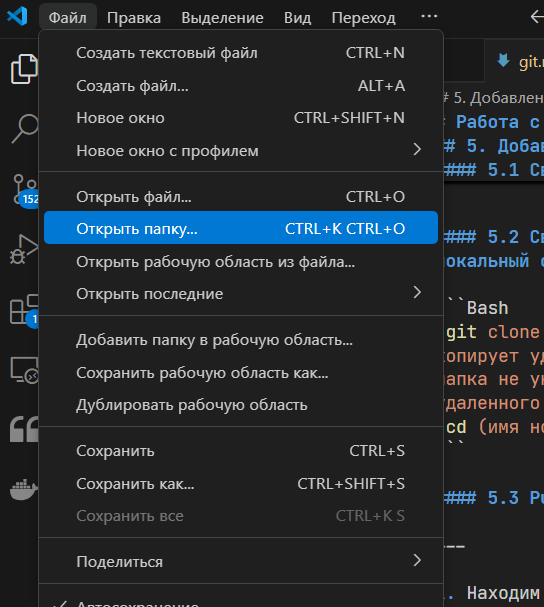
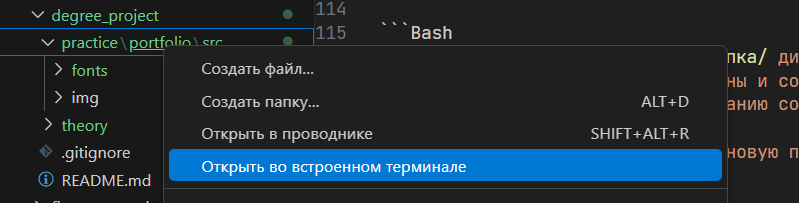


Рисунок 9. Открываем проект в VSCode

* Создаем новый терминал и переходим в директорию portfolio





* В терминале вводим команды node и npm -v. Они отобразят версии установленных Node.js и npm. Если номера версий видны, установка прошла успешно – инициируем новый проект “npm init”, будет создан файл package.json, который будет содержать метаданные о проекте и его зависимостями.

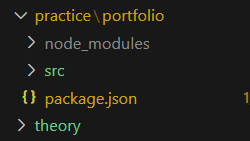


Рисунок 11. Создание package.json

## Установка пакета gulp “npm install --save-dev gulp” - проверяем установленный пакет “gulp --version”

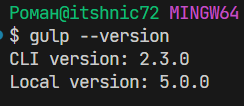


Рисунок 12. Установка Gulp

## В директории portfolio создаем gulpfile.js в котором напишем код для управления сборкой проекта

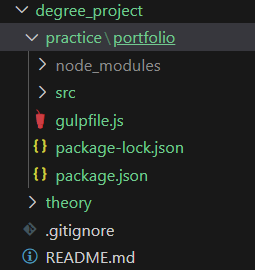


Рисунок 13. Создание gulpfile.js

**4.3. Создание системы контроля версий Git**

**Git** — это распределённая система контроля версий. Она помогает разработчикам управлять состоянием исходного кода на протяжении всей разработки. Система записывает изменения в файлы и позже позволяет откатиться к более ранней версии проекта.

Главная отличительная черта Git — подход к обработке данных. Каждый раз при сохранении данных проекта (комитет) система фиксирует состояние файла (делает снимок) и создаёт ссылку на этот снимок. Последующие изменения отражаются через ссылки на более ранние версии файла, нет необходимости снова сохранять файл целиком.

Благодаря распределённой структуре репозитории Git хранятся на всех клиентских компьютерах, что защищает от потерь данных и позволяет полноценно управлять версиями проекта оффлайн.

**Репозиторий** это хранилище данных, которое можно представить как каталог информации с файлами, локальный или расположенный на каком-либо ресурсе.

В качестве репозитория выбираем github.

Создаем удаленный репозиторий:

* регистрируемся по адресу <https://github.com/>.
* создаем новый удаленный репозиторий

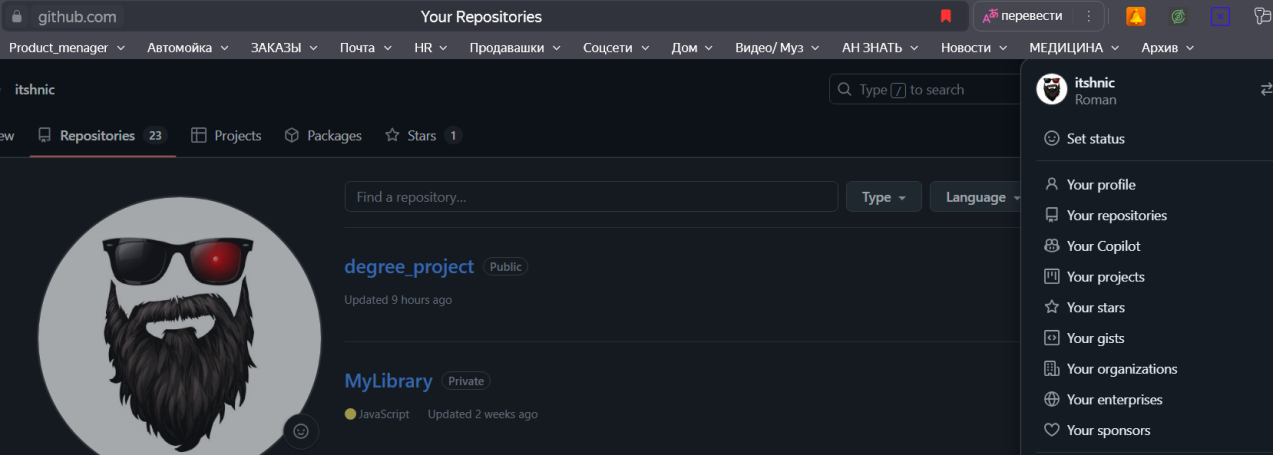


Рисунок 14. Регистрация в github

Создаем локальный репозиторий в директории проекта:

* в терминале проекта пишем команду “git init”
* командой “git remote add origin <https://github.com/itshnic/degree_project.git>” подключаем локальный к удаленному репозиторию

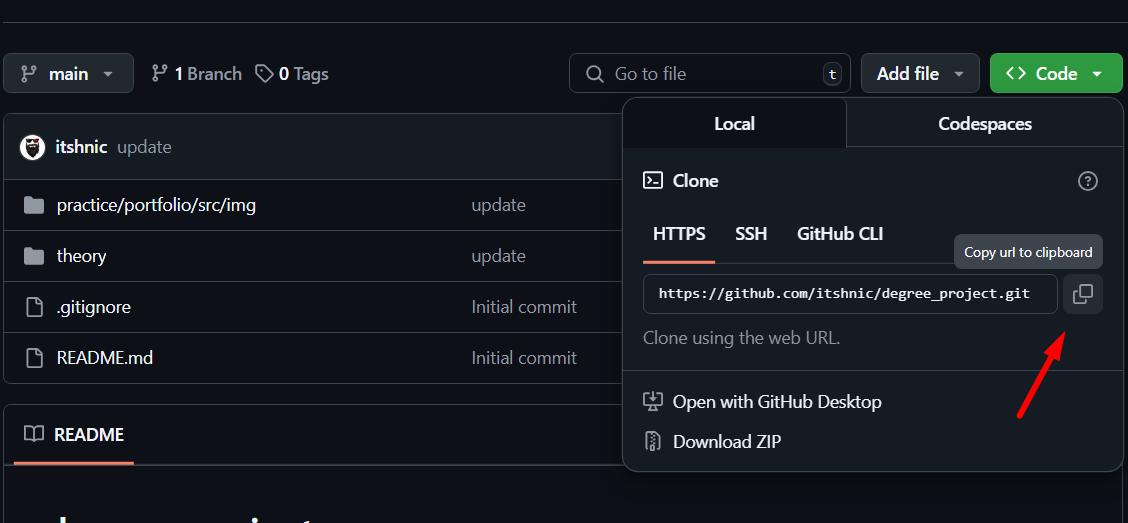


Рисунок 15. Подключение к локальному git

* командой “git branch –M main” меняем главную ветку, локального репозитория на main
* командой “git add .” добавляем информацию о файлах проекта в локальный репозиторий
* командой “git commit –m ‘create commit’” создание снимка проекта
* командой “git push” отправка информации в удаленный репозиторий
* командой “git pull” загрузка информации из локального репозитория
* в директории портфолио создаем фаил .gitignore – текстовый файл, который используется в системе контроля версий Git для определения игнорируемых файлов и каталогов

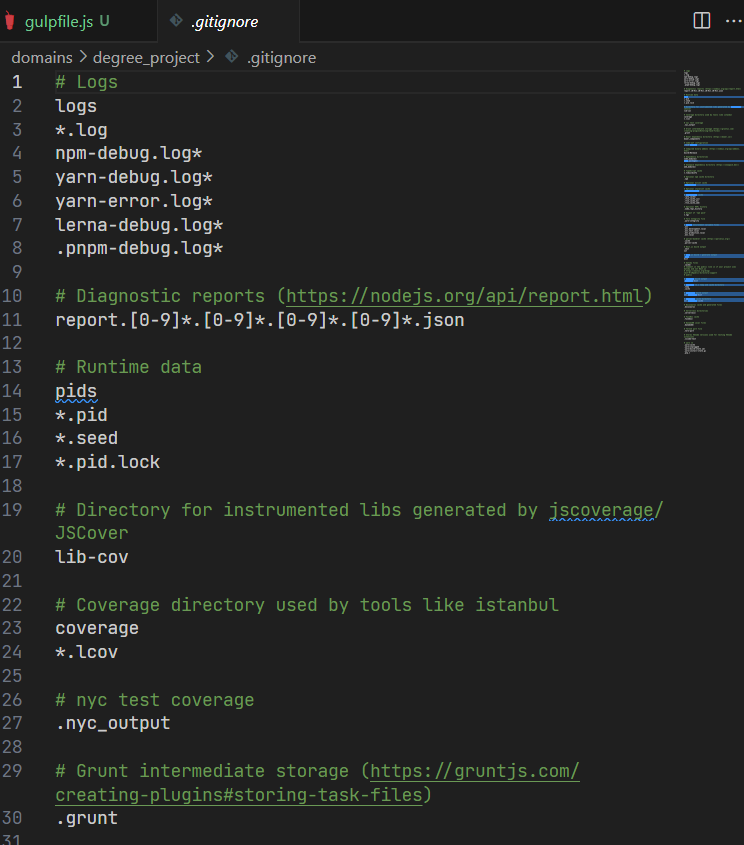


Рисунок 16. Создание файла gitignore

**4.4. Создание файловой структуры**

В “src” создаем директории:

* папку html для файлов (компонентов) с расширением html
* папку js для файлов с расширением js
* папку json для файлов с текстом
* папку pages для добавления новых проектов
* папку scss создания стилей сайта при помощи препроцессоров

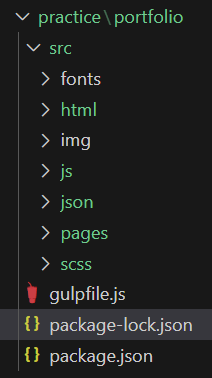


Рисунок 17. Файловая структура папки src

**4.5. Описание сборки в gulpfile.js**

Загружаем модули в node командой “npm install <имя\_пакета> ” и при помощи функции require() импортируем их в переменную

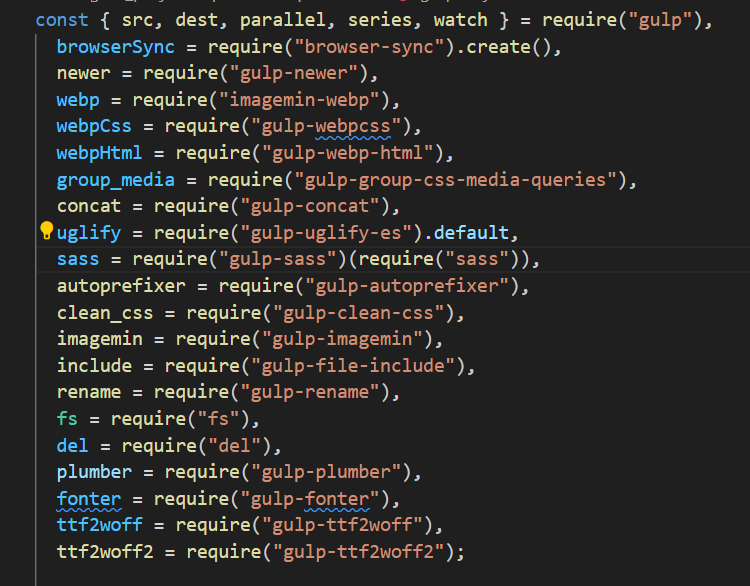


Рисунок 18. Подключение модулей

Пишем пути к файлам и сохраняем их как свойства объекта, используем модуль path



Рисунок 19. Путь к файлам и папкам

Используя модуль browserSync – запускаем браузер при запуске проекта

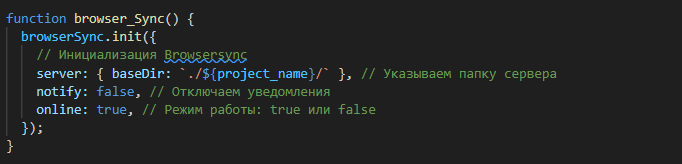


Рисунок 20. Подключение сервера на localhost

Используя модуль include() подключаем файлы из директории html в build (создается автоматически используя модуль dest() )

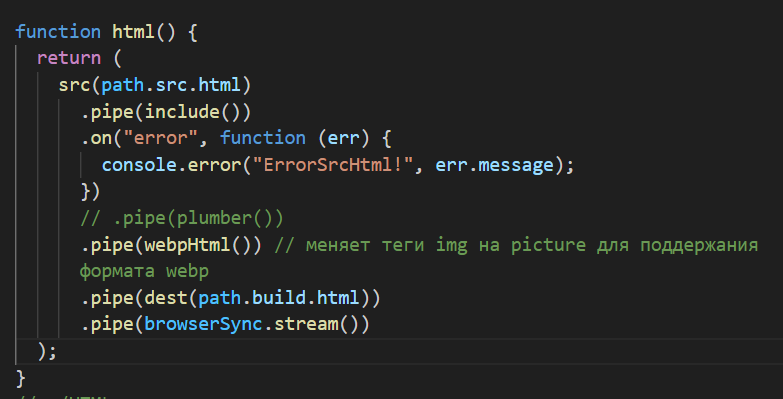


Рисунок 21. Сборка HTML

Финкции для переноса файлов php из директории src в build

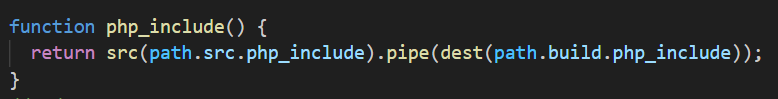


Рисунок 22. Сборка PHP

Финкции для смены формата файла scss на css и добавляем поддержку изображений в формате webp



Рисунок 23. Сборка сss



Рисунок 24. Сборка сss (подключение библиотек)

Финкции для конкатинации и минификации всех файлов css

**Конкатенация** в программировании это процесс последовательного объединения двух или более строк в единое целое.

**Минификация** процесс уменьшения размера исходного кода путём удаления ненужных символов без изменения его функциональности.

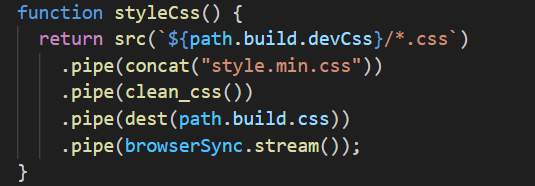


Рисунок 25. Сборка сss (в один фаил)

Функции для конкатинации и минификации всех файлов js используя модули concat(), uglify()

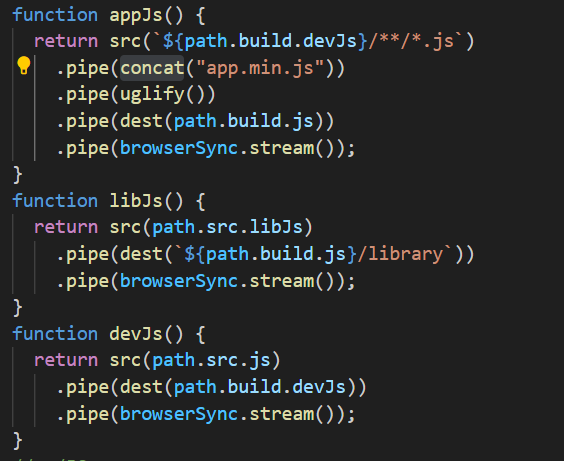


Рисунок 26. Сборка js

Функция - очистка директории с build используя модуль del()

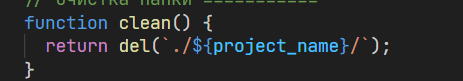


Рисунок 27. Очистка папок перед сборкой

Функция перевода шрифта в форматы ttf ,ttf2 и woff2



Рисунок 28. Работа с шрифтами

Функция создания миксинов в файле стилей (для ускорения процесса верстки)

**Миксин** в CSS — это блок готовых стилей, помогают группировать нужные стили и повторно использовать их в нескольких местах кода или в разных CSS-файлах

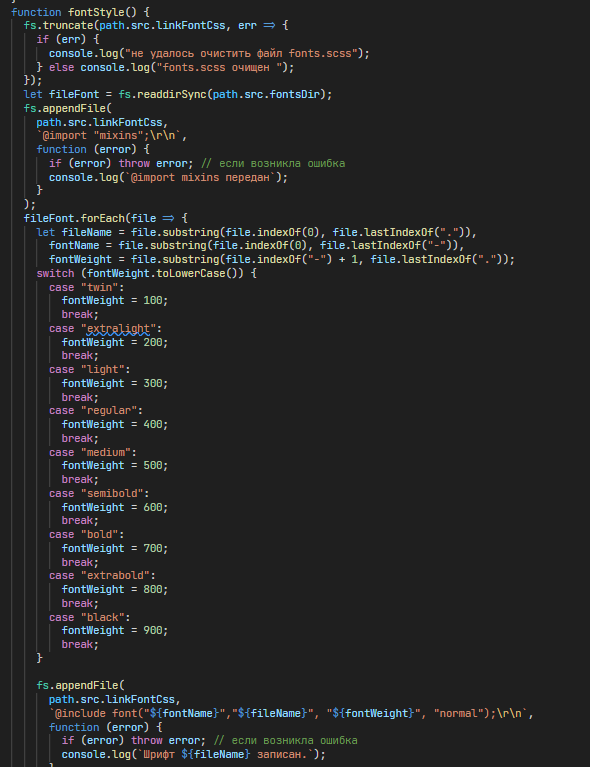


Рисунок 29. Автозаполнение Миксинов

Функция переноса JSON файлов в build

**JSON** (JavaScript Object Notation) — текстовый формат для хранения и обмена структурированными данными. Он основан на синтаксисе объектов в JavaScript, но не зависит от него. Данные представляются в виде пар «ключ-значение», в которых ключи — всегда строки, а значения могут быть представлены различными типами: числовыми, строковыми, логическими.

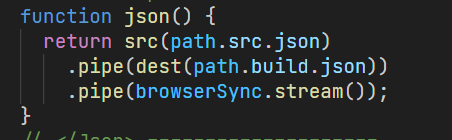


Рисунок 30. Сборка JSON

Функция сжатия изображения без потери качества. Переводим изображения в формат webp используя модуль imagemin(). Функция favicon() - меняет расширение файла и добавляет его в директорию build

**WebP** — это современный формат изображений, разработанный компанией Google в 2010 году. Он предназначен для того, чтобы обеспечивать высокое качество при меньшем размере файла по сравнению с традиционными форматами, такими как JPEG и PNG.

WebP используется в веб-разработке для оптимизации изображений на сайтах и в социальных сетях, а также в мобильных приложениях, где важна скорость загрузки.



Рисунок 31. Сжатие изображений

Функция-наблюдатель за изменениями файлов в директории src

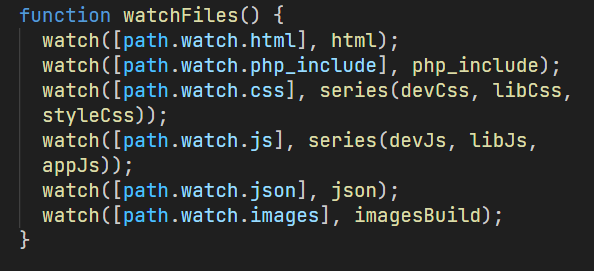


Рисунок 32. Наблюдатель за изменениями в файлах и папках

Прописываем сценарий для сборки проекта в момент разработки и экспортируем его. Запуск при помощи команды – “gulp”



Рисунок 33. Export сценариев

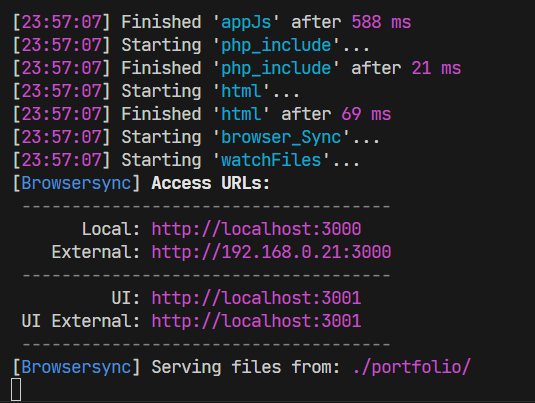


Рисунок 34. Запуск Gulp

**4.6. Верстка**

В директории html создаем index.html, в который подключаем модули для более удобного создания и управления проектом:

* \_head.html метаданные
* \_header.html шапка сайта
* \_js.html подключение стилей
* \_footer.html подвал сайта

Модули подключаем к основному файлу командой @@include('\_header.html',{})

index.html

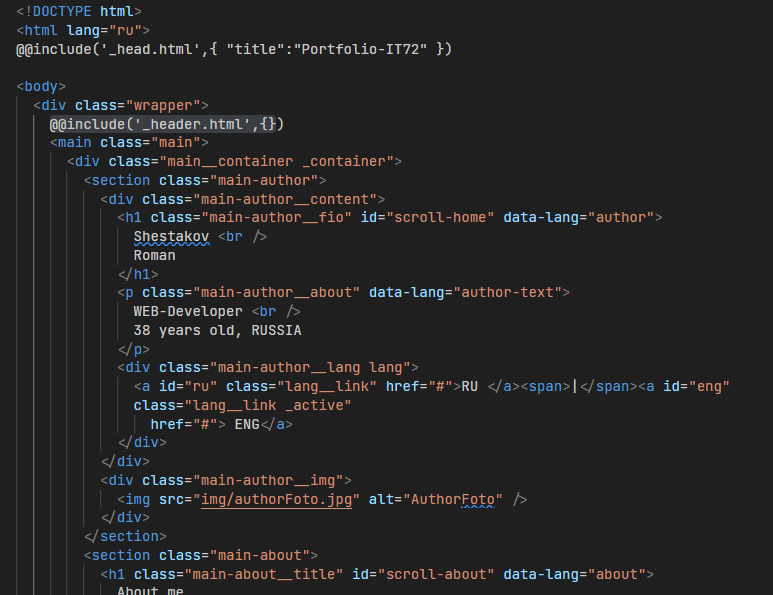


Рисунок 35. Основной модуль HTML

\_head.html

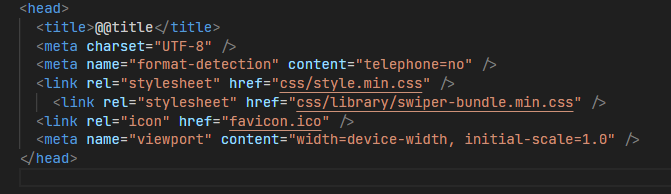


Рисунок 36. Дочерний модуль head

\_header.html



Рисунок 37. Дочерний модуль header

\_js.html

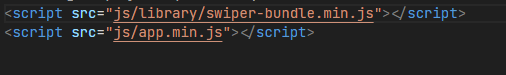


Рисунок 38. Дочерний модуль js

\_footer.html

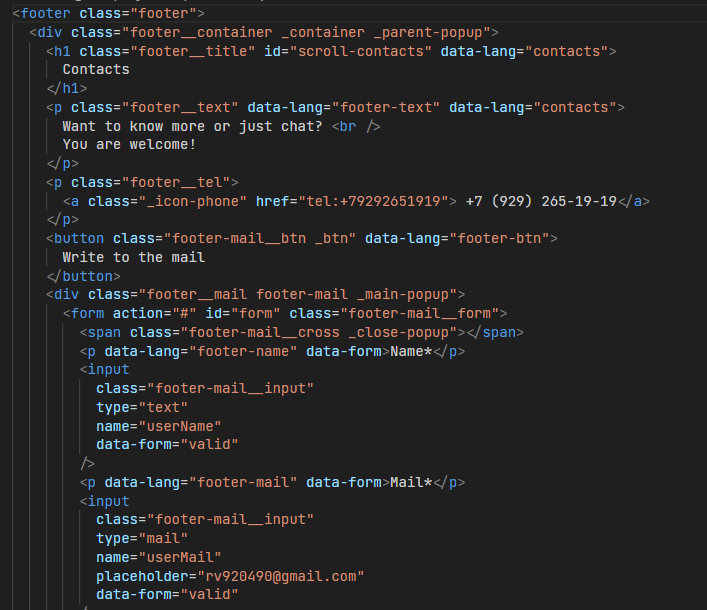


Рисунок 39. Дочерний модуль footer

В директории js создаем фаил script.js в котором пишем функции:

* управление меню бургер (появление списка и смена иконки)

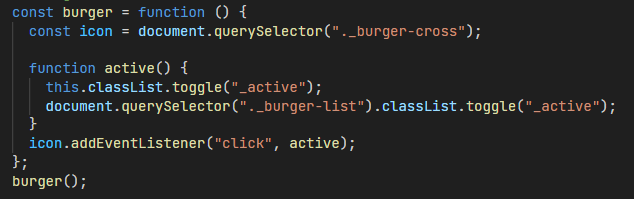


Рисунок 40. Меню бургер

* отправка формы на сервер



Рисунок 41. Отправка формы

* всплывающий блок после клика по кнопке



Рисунок 42. PopUp

* смена языка Ru /En

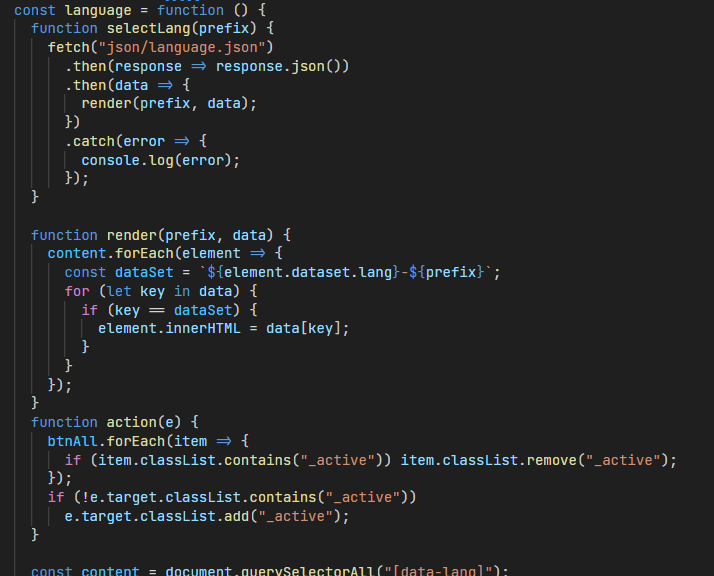


Рисунок 43.Переключение языка

* якорные ссылки (перемещение по разделам сайта при клике по пункту меню)

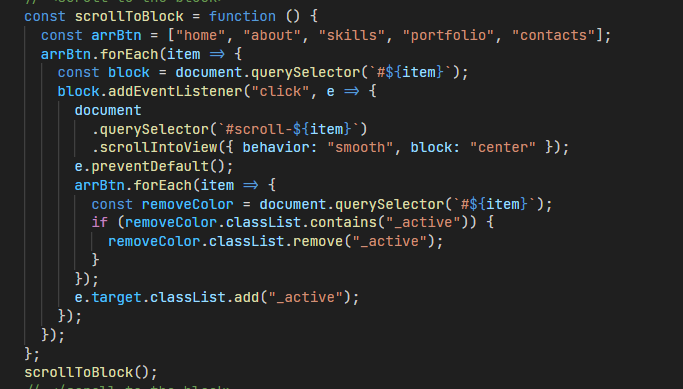


Рисунок 44. Навигация

* подключение слайдера и добавление в него контента



Рисунок 45. Slider

* при клике (показать еще) показывает 3 скрытых карточки



Рисунок 46. Функция – показать еще контент

В директории json создаем фаилы language.json и skills.json (содержит тексты для динамической загрузки)

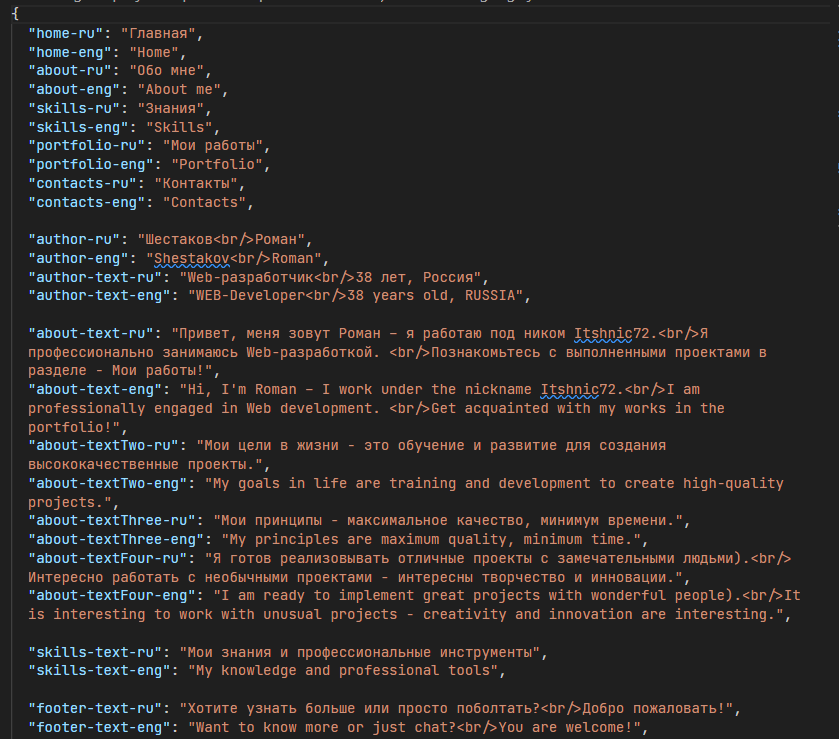


Рисунок 47. JSON файлы

**SCSS** (Sassy Cascading Style Sheets) — это синтаксис препроцессора Sass, который расширяет стандартные возможности CSS с помощью новых синтаксических конструкций, таких как миксины, циклы, переменные и другие.

В директории scss создаем фаилы:

* null.scss пишем обнуляющие стили для всего проекта (убираем лишние отступы, заливки и др)

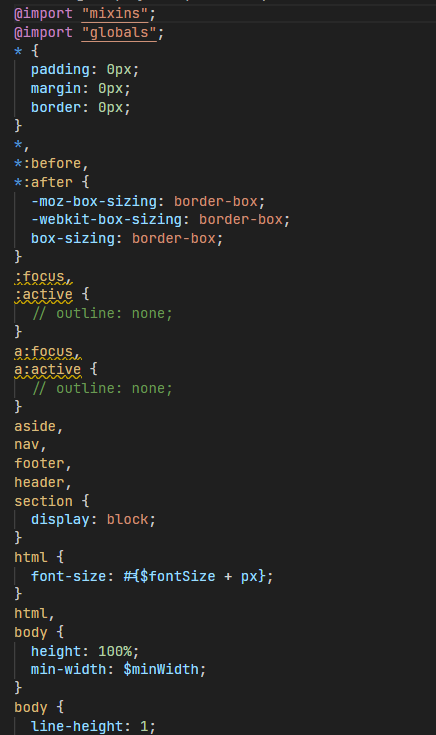


Рисунок 48. Обнуляющие стили

* fonts.scss – заполняется динамически (читает данные из директории fonts) после запуска команды “gulp”. Работает функция fontStyle() из gulfile.js

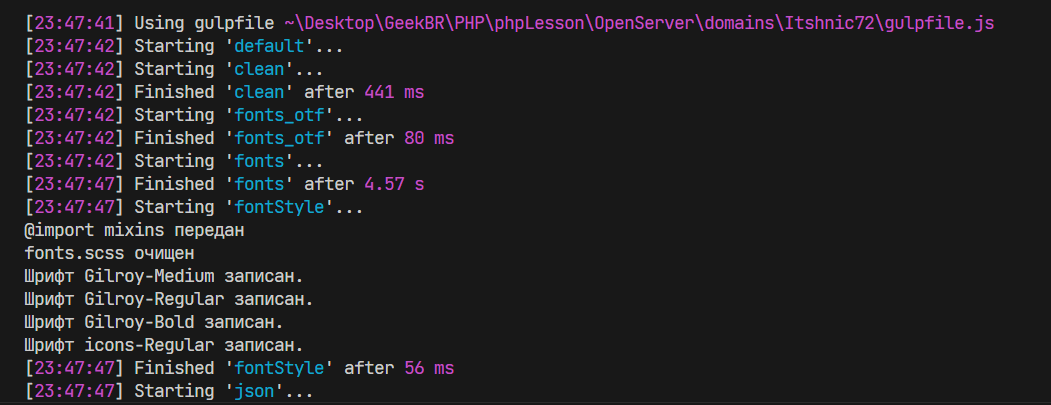


Рисунок 49. Сообщение в терминале



Рисунок 50. Заполненные миксины (подключение шрифтов)

* globals.scss импортируем фаил mixins и прописываем глобальные переменные (данные берем из макета)



Рисунок 51. Глобальные переменные css

* header.scss импортируем фаилы globals и mixins. Создаем стили для шапки сайта

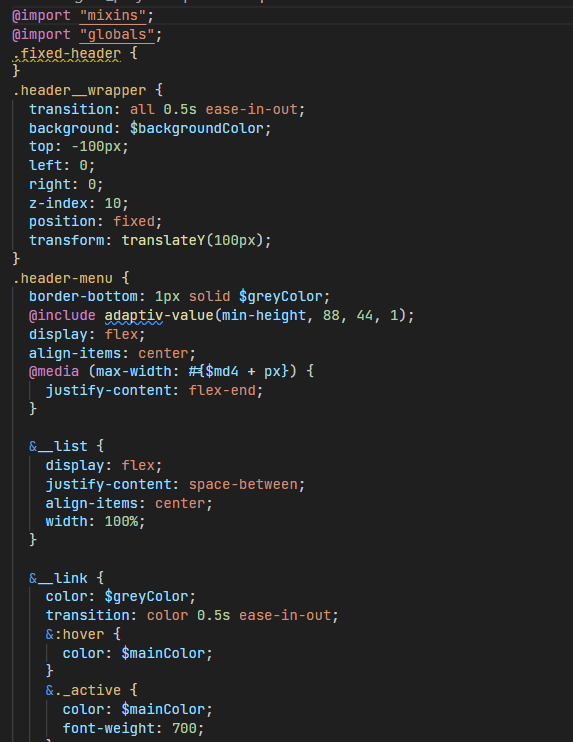


Рисунок 52. Стили для header.html

* main.scss импортируем фаилы globals и mixins. Создаем стили блока main (средняя часть сайта)

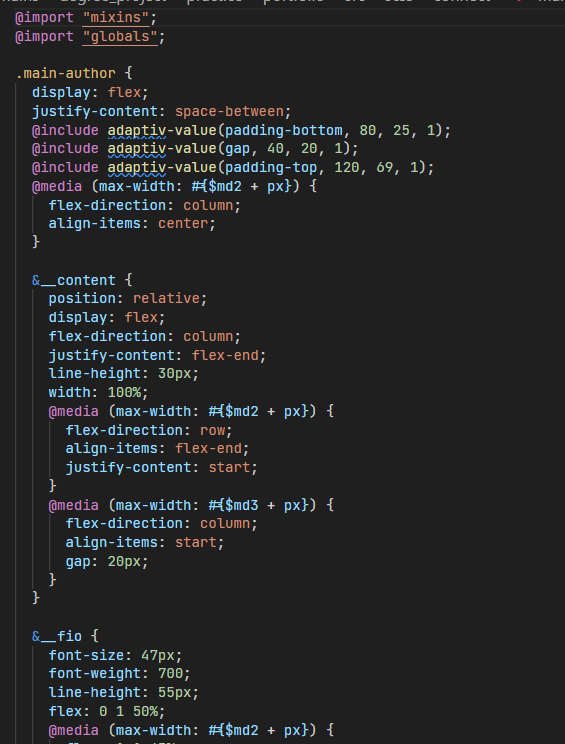


Рисунок 53. Стили для main.html

* icons.scss подключаем иконки предварительно конвертируя в шрифт через приложение https://icomoon.io/app/#/select/font

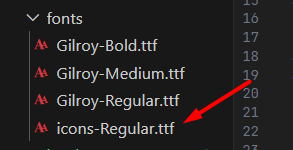


Рисунок 54. Шрифт - иконки

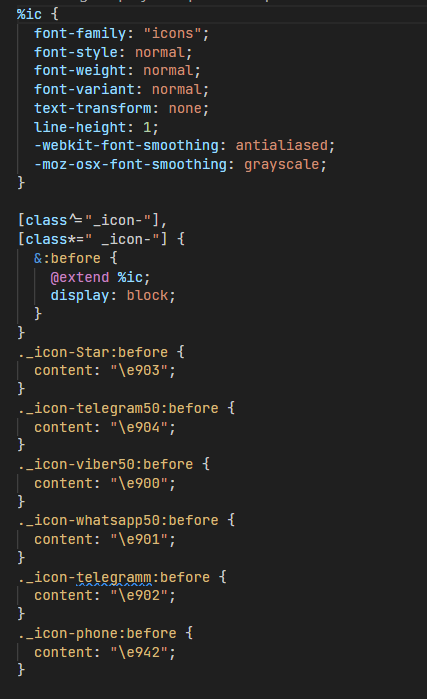


Рисунок 55. Подключение иконок (ч/з миксин)

* mixins.scss создаем миксины – для быстрого написания кода которы позволит сайту быть отзывчивым и адаптивным под любое разрешение. Так-же пишем миксин для подключения шрифтов

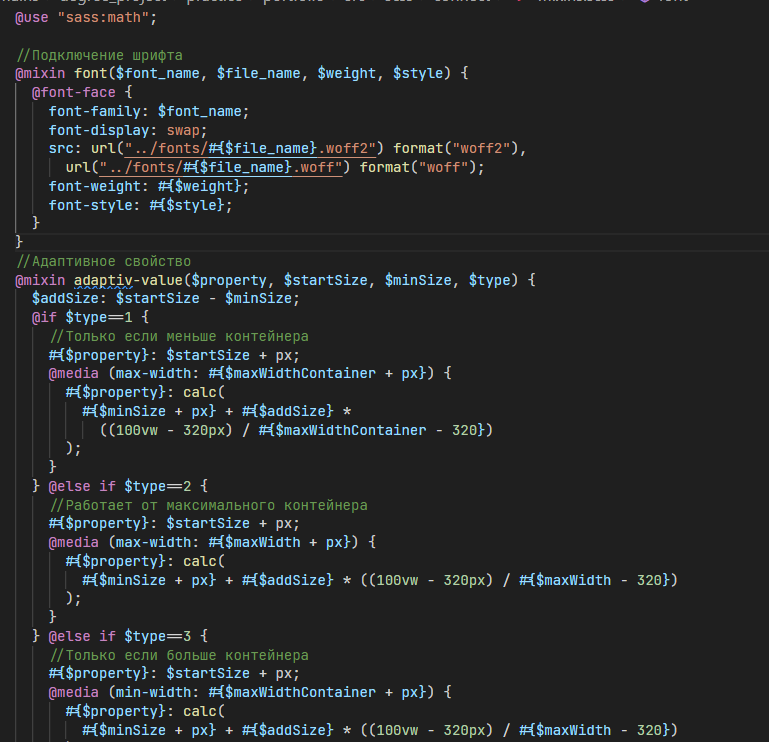


Рисунок 56. Шаблоны миксинов

* footer.scss импортируем фаилы globals и mixins. Создаем стили для подвала сайта

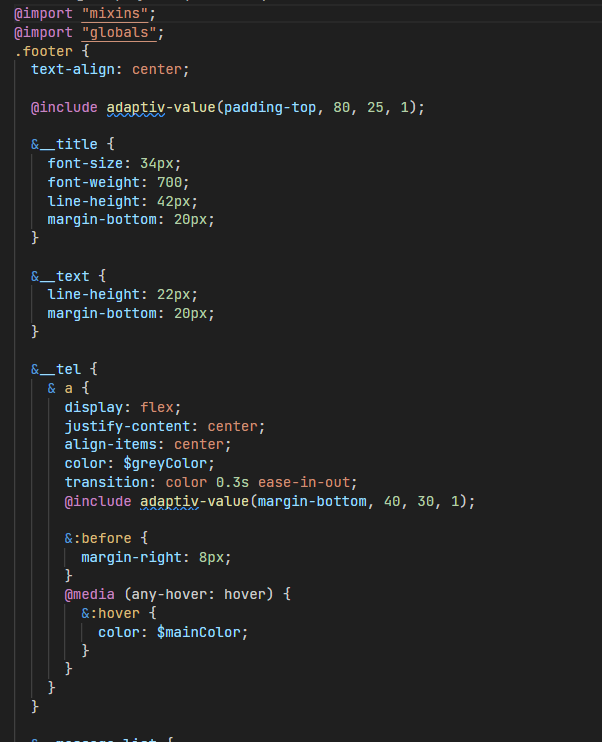


Рисунок 57. Стили для footer.html

В результате получилась страница - портфолио адаптивная под различные разрешения (устройства)

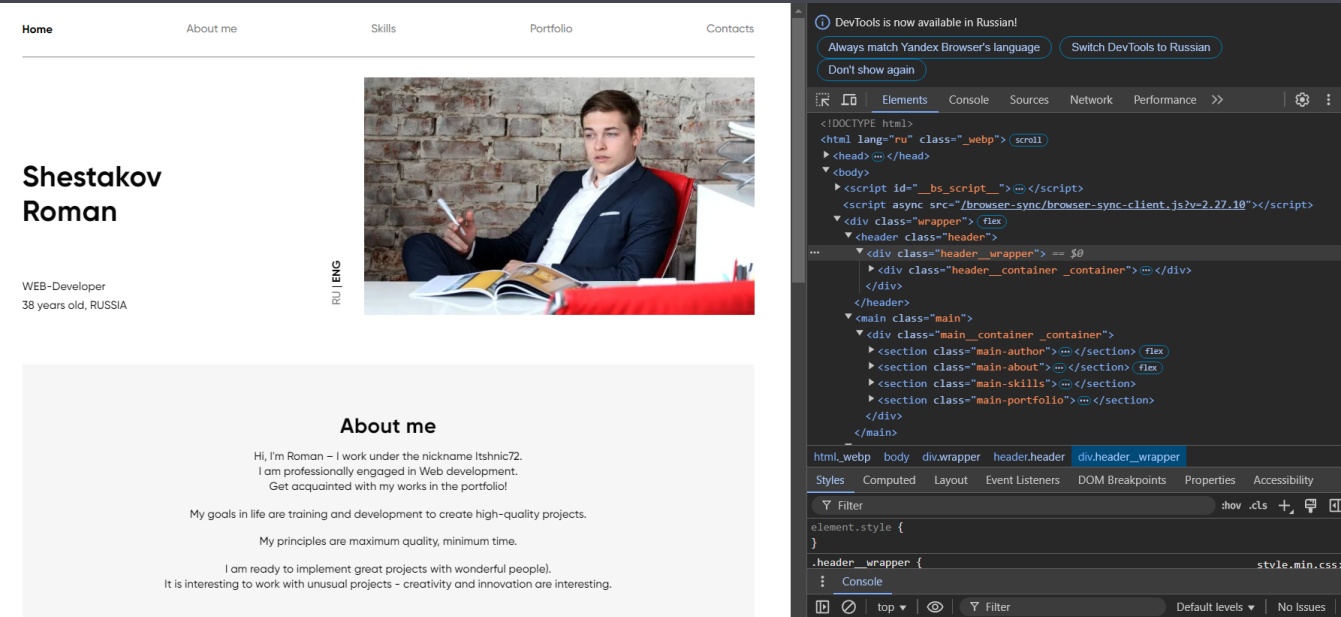


Рисунок 58. Портфолио (разрешение desktop)

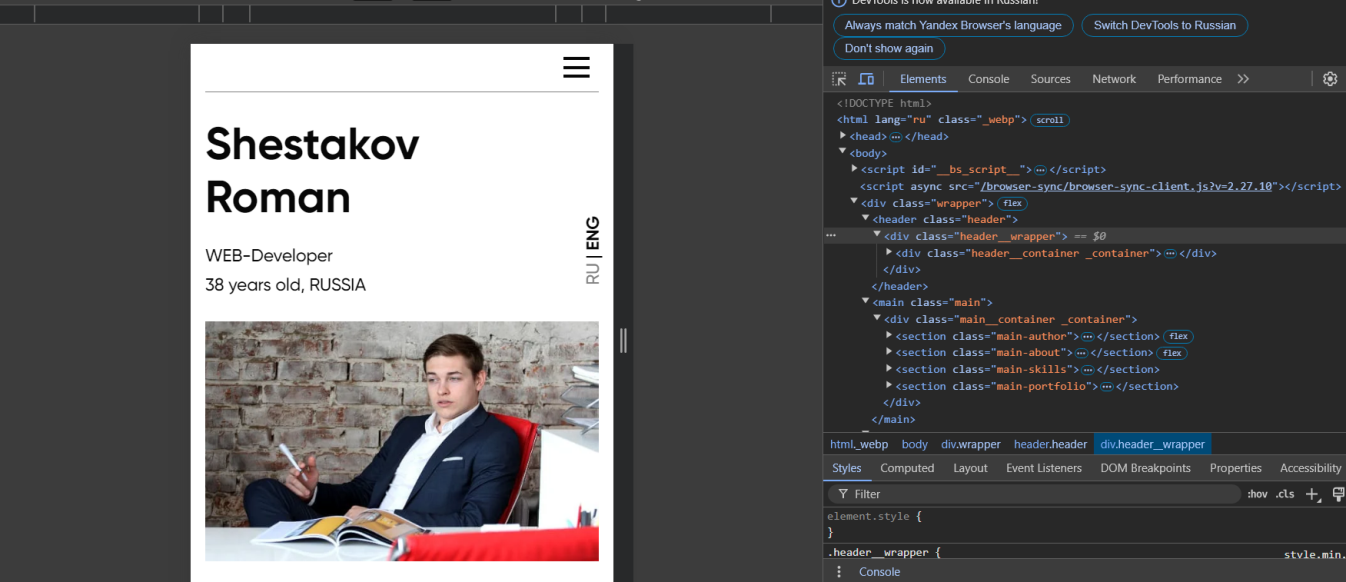


Рисунок 59. Портфолио (разрешение мобильный телефон)

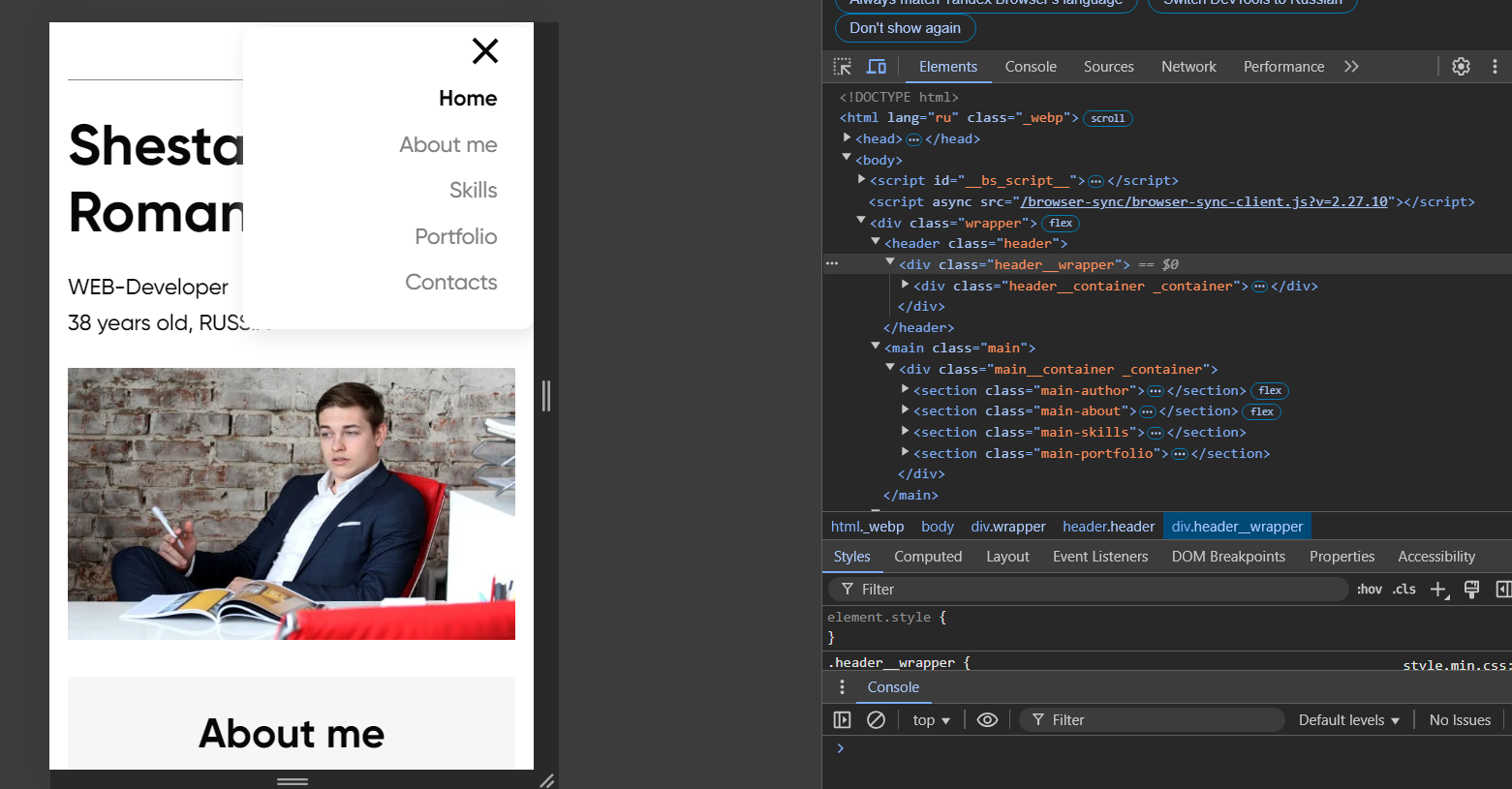


Рисунок 60. Портфолио (меню-бургер открыто)

Для динамической загрузки карточек проекта добавим в директорию pages несколько проектов (верстка сайтов по макетам). Для правильного отображения в макете Figma подготовим preview с помощью скриншота на разных разрешениях экрана.



Рисунок 61. Preview для карточек проектов

В файле language.json добавляем описание проекта на разных языках.

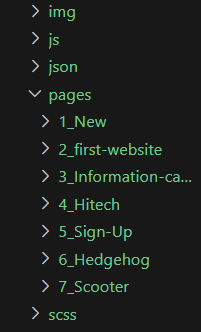


Рисунок 62. Папки проектов в портфолио

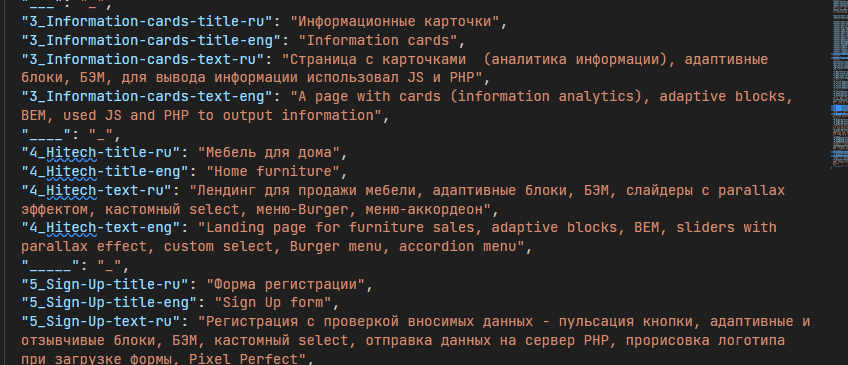


Рисунок 63. Описание проектов в JSON

Рисунок 64. Описание проектов в JSON

**4.7. Подготовка сайта для загрузки на хостинг**

Переименовываем фаил index.html в index.php (директория html)

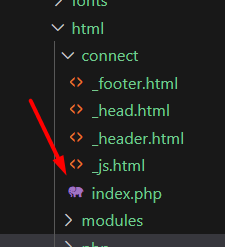
****

Рисунок 64. Html в php

Рисунок 63. Описание проектов в JSON

Для просмотра страницы используем Open Server — это портативная программная среда для веб-разработчиков. Она включает в себя набор серверного программного обеспечения (Apache, Nginx, Sendmail, Ghostscript, PHPMyAdmin, MariaDB, Memcached, MongoDB, MySQL, PostgreSQL, Redis, различные PHP модули и дополнительные расширения и пр.).



Рисунок 65. Запуск Open Server

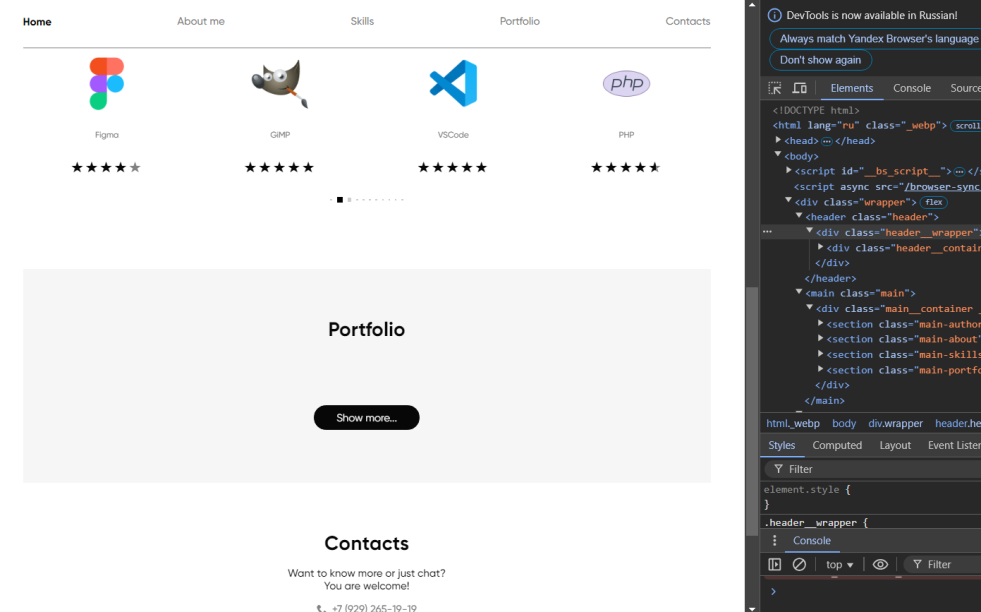


Рисунок 66. Раздел карточки проекта

Создаем директорию html/php в ней фаил function.php. Пишем функцию которая читает папку pages и создает элемент html страницы.

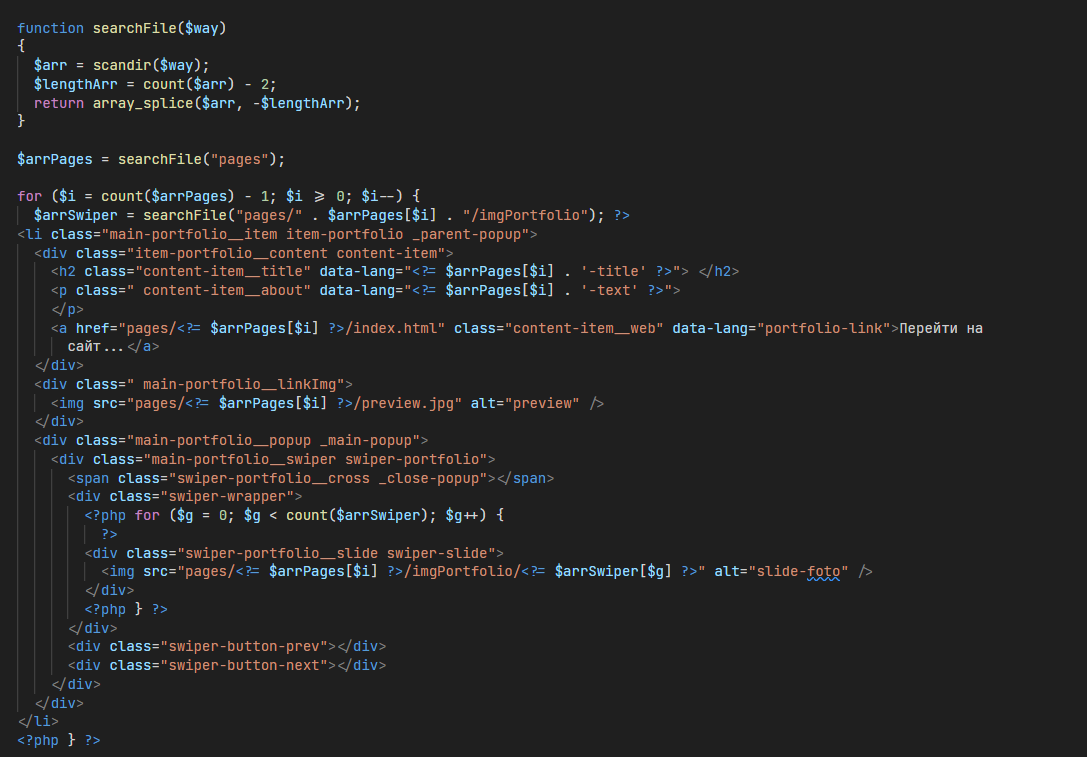


Рисунок 67. Скрипт PHP

Вставляем данные function.php в код index.php

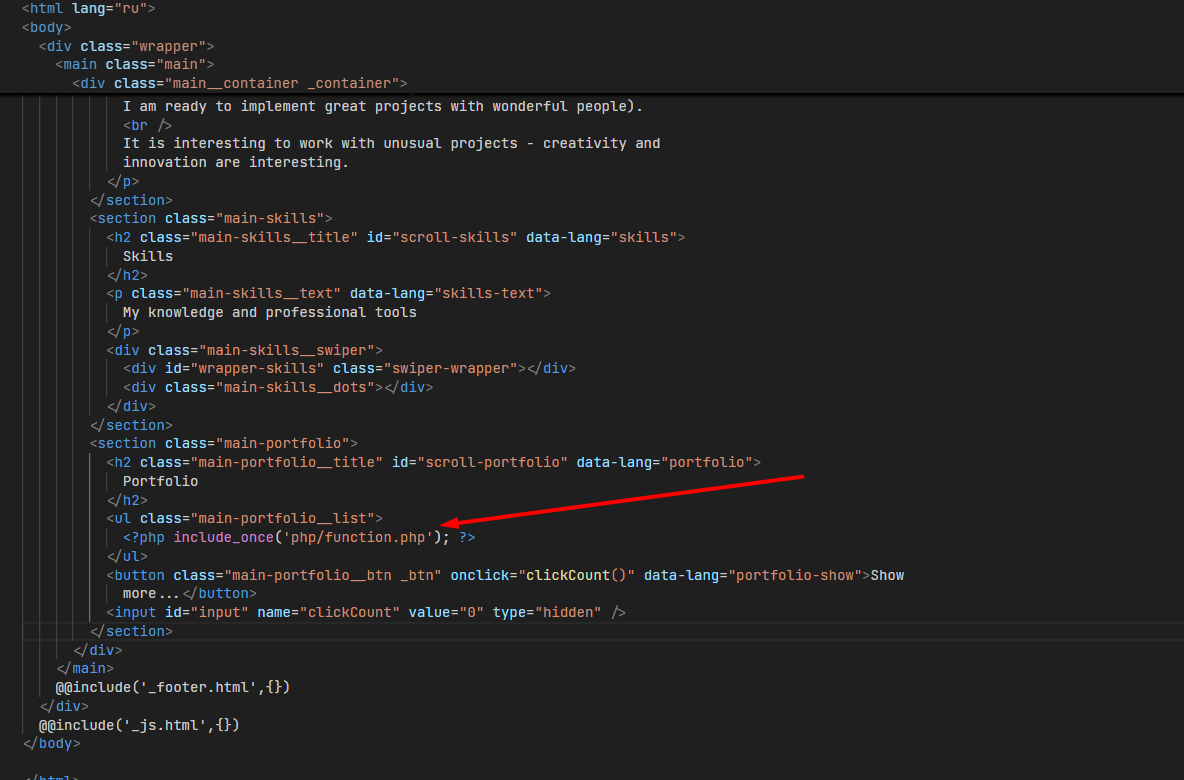


Рисунок 68. Вставка в index.php

Результат:

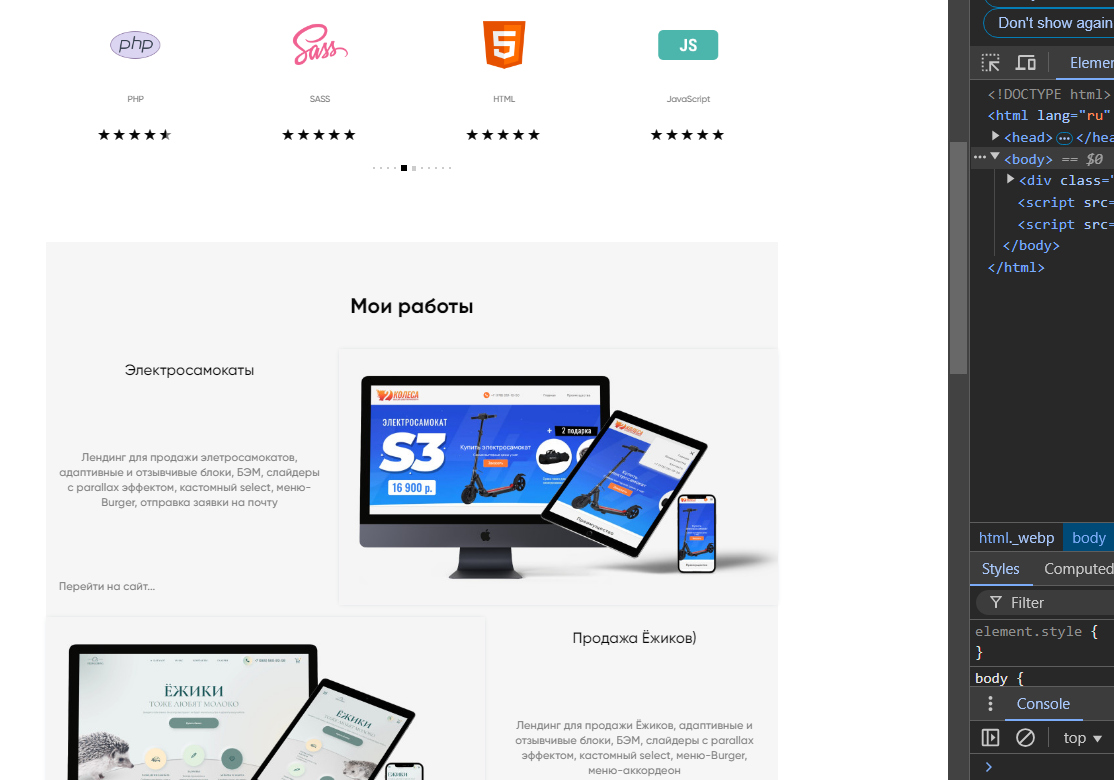


Рисунок 69. Раздел – карточки проектов

Добавим **PHPMailer** — библиотека PHP с открытым исходным кодом для отправки электронной почты. Она позволяет разработчикам создавать приложения для создания и отправки сообщений на электронную почту.

Скачиваем исходные файлы PHPMailer в папку проекта.

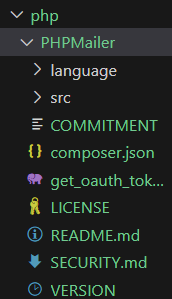
****

Рисунок 70. Установка PHPMailer

В файле server.php настраиваем PHPMailer

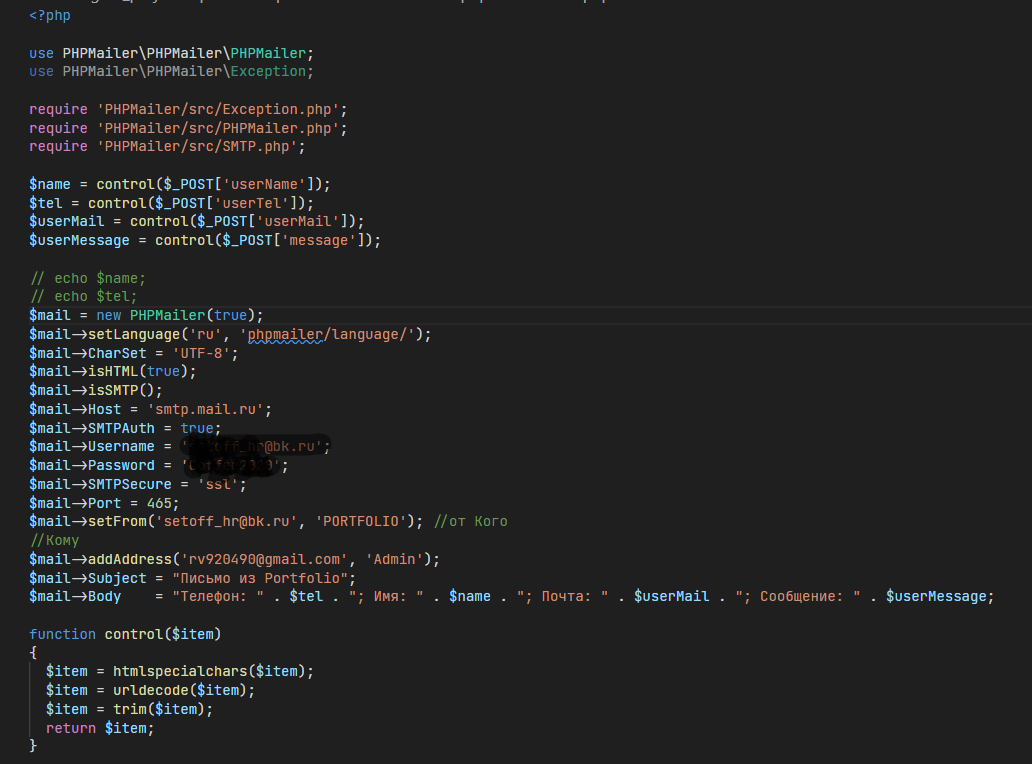


Рисунок 71. Подключение PHPMailer

Тестируем:

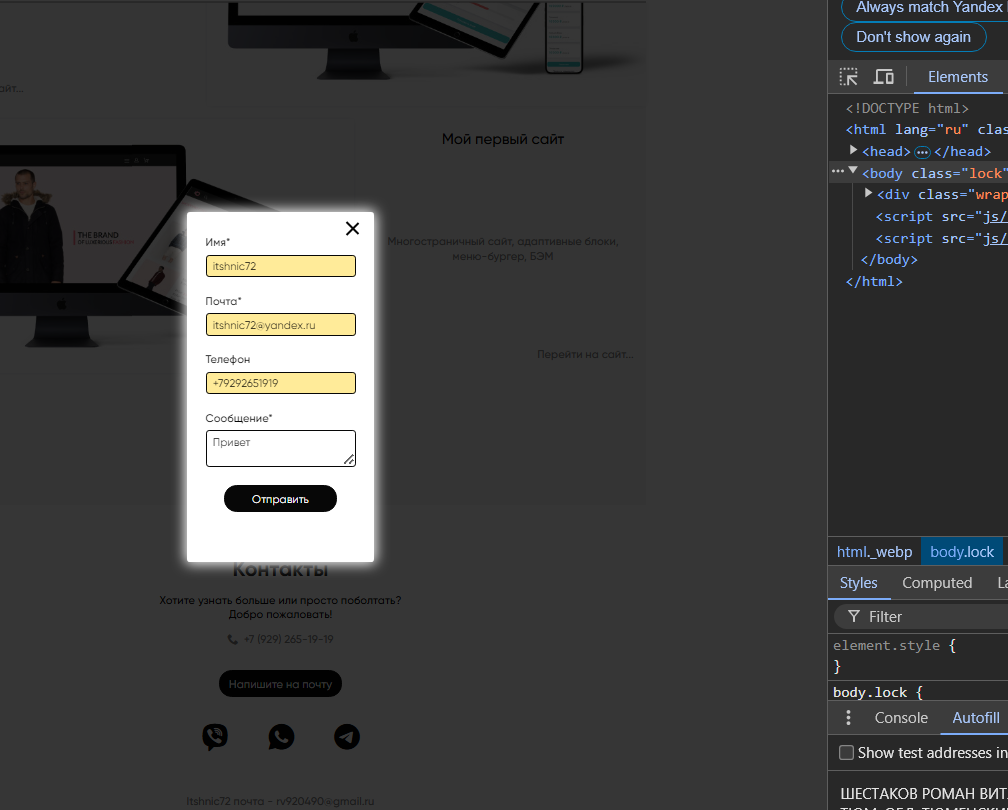


Рисунок 72. Тест формы

По окончанию разработки сборщик Gulp – создал директорию и подготовил файлы (скомпилировал и минифицировал) для загрузки на хостинг.

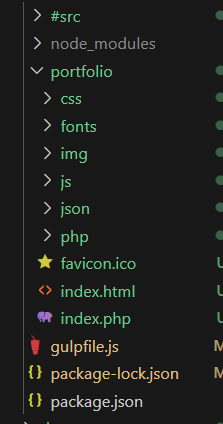


Рисунок 73. Директория для продакшн (Production)

Делаем commit, папки проекта в локальный репозиторий и загружаем данные в удаленный репозиторий используя набор команд:

* git add .
* git commit –m ‘create project’
* git push

**4.8. Загрузка проекта на хостинг**

**Хостинг** — это услуга по размещению данных на сервере с постоянным доступом к интернету. Другими словами, это место, на котором хранятся файлы сайта.

В сети интернет существует множество хостингов, отличаются интерфейсом, ценами, уровнем тех поддержки.

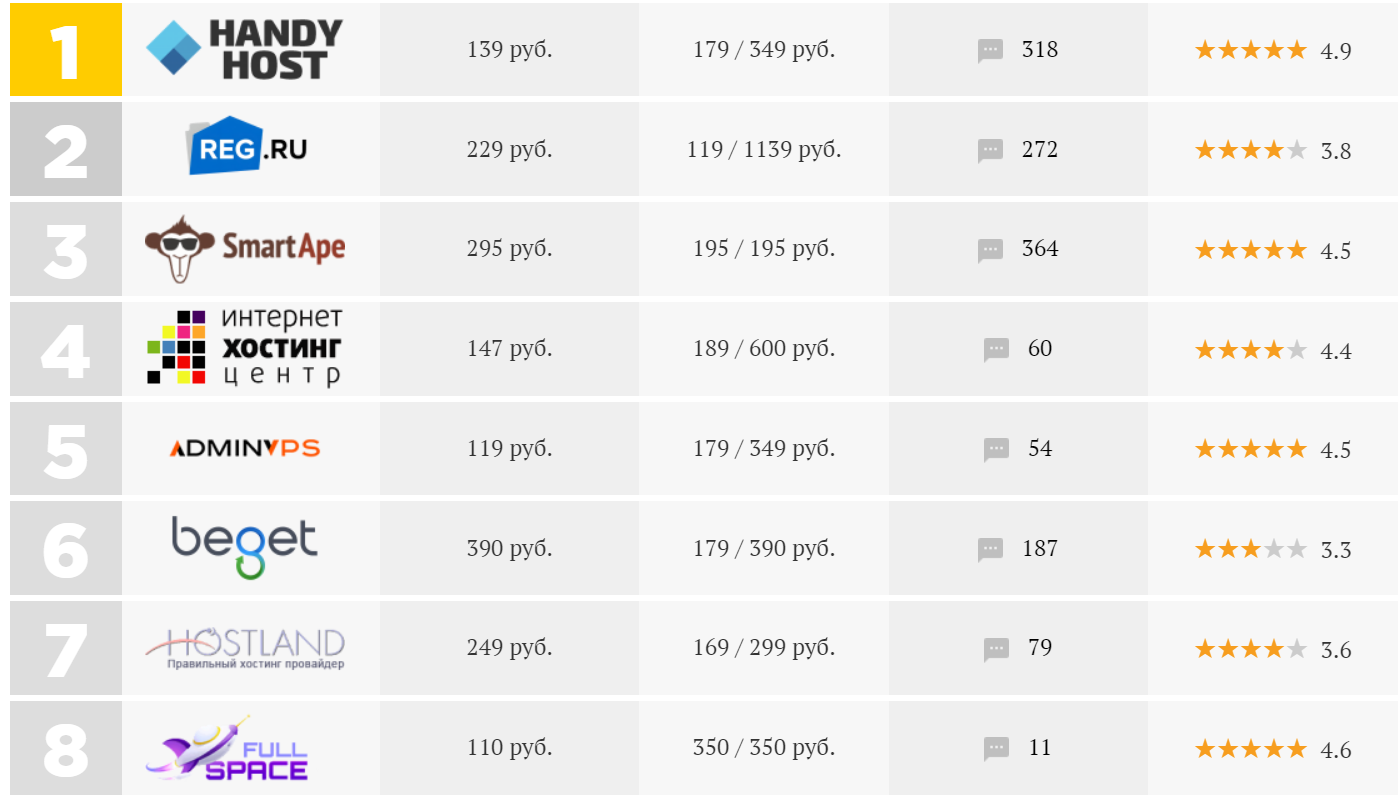


Рисунок 74. Хостинги

Для меня более интересен и доступен хостиг Beget.

Проходим регистрацию, покупаем доменное имя, так-же его можно выбрать из бесплатных.

**Доменное имя** — это уникальное имя сайта, адрес, по которому страницы ресурса доступны в интернете. Оно состоит из букв, цифр и символов.

Структура доменного имени включает несколько уровней:

* Нулевой уровень (корневой домен). Это точка в конце имени, в адресной строке она не указывается.
* Первый уровень (доменная зона). Доменные зоны могут быть международными (.COM, .NET, .ORG), национальными (.RU, .UK, .DE) и тематическими (.EDU, .MOVIE, .CAFE).
* Второй уровень (основное имя сайта). Это часть имени слева от точки, которая отделяет доменную зону.
* Третий, четвертый, пятый уровень и т.д. (поддомены). Каждый уровень отделяется точкой от предыдущего. Например, так может обозначаться раздел сайта.

Домен не связан напрямую ни с сервером сайта, где находятся все данные ресурса, ни с его IP-адресом — уникальным идентификатором для обмена запросами. Но интернет-служба DNS «связывает» между собой доменное имя и IP сайта, поэтому к сайту можно обращаться по домену.

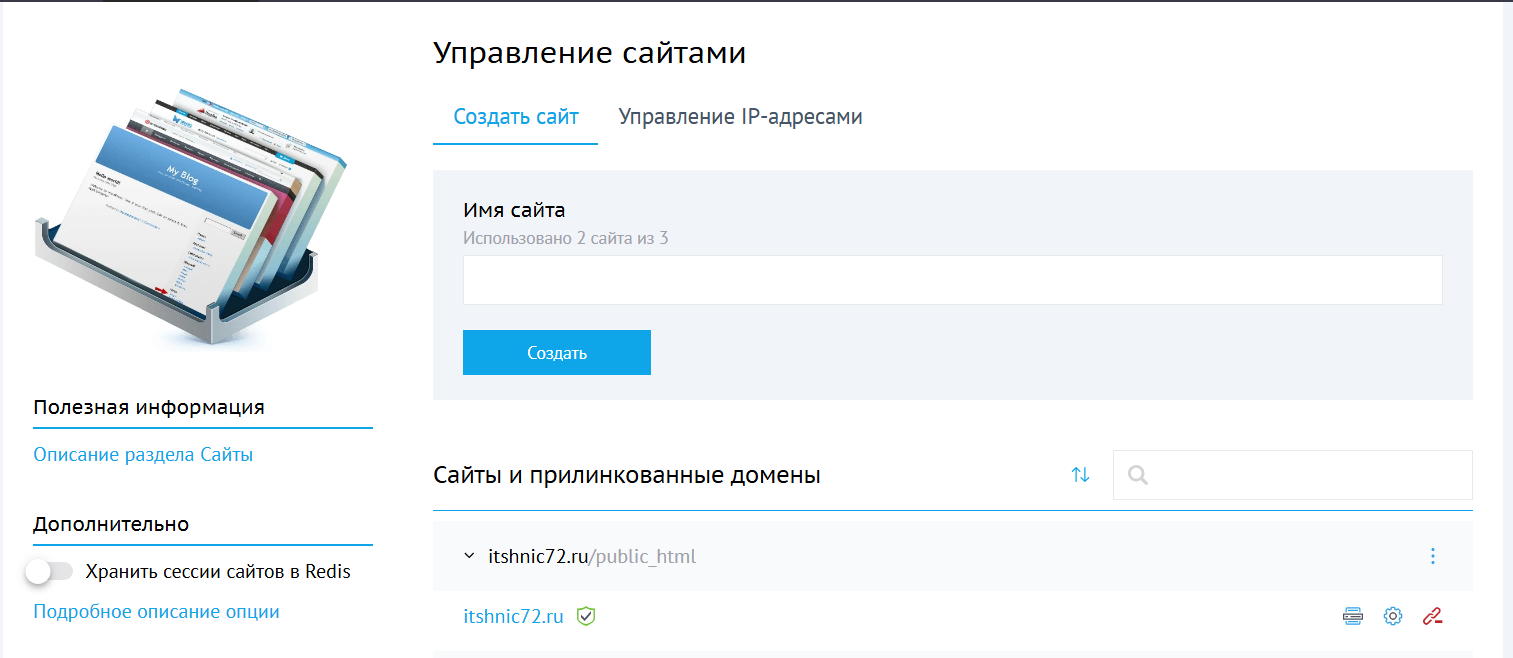


Рисунок 75. Подключение домена

При помощи файлового менеджера загружаем проект.

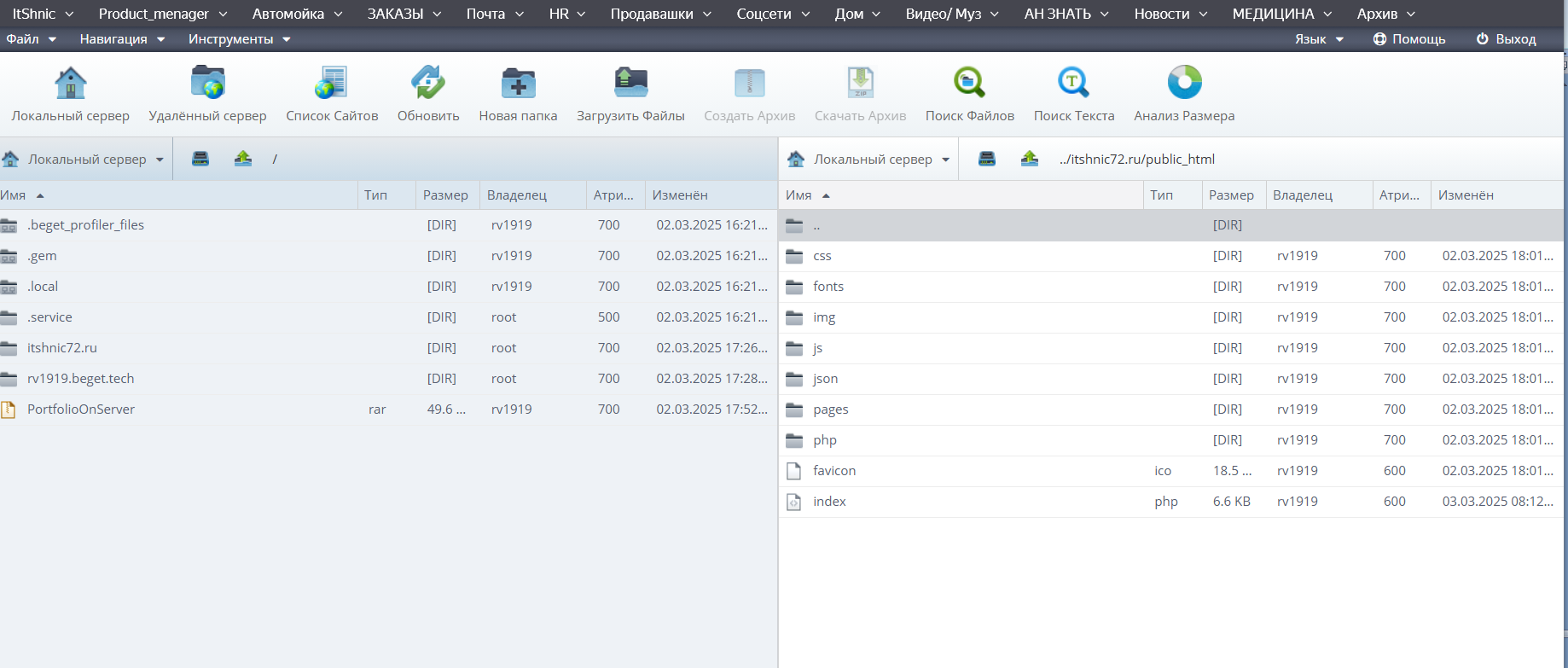


Рисунок 76. Файловый менеджер

Проект теперь доступен в глобальной сети интернет по адресу - <https://itshnic72.ru/>

**4.9 Тестирование**

Проверка корректности разметки при помощи W3C

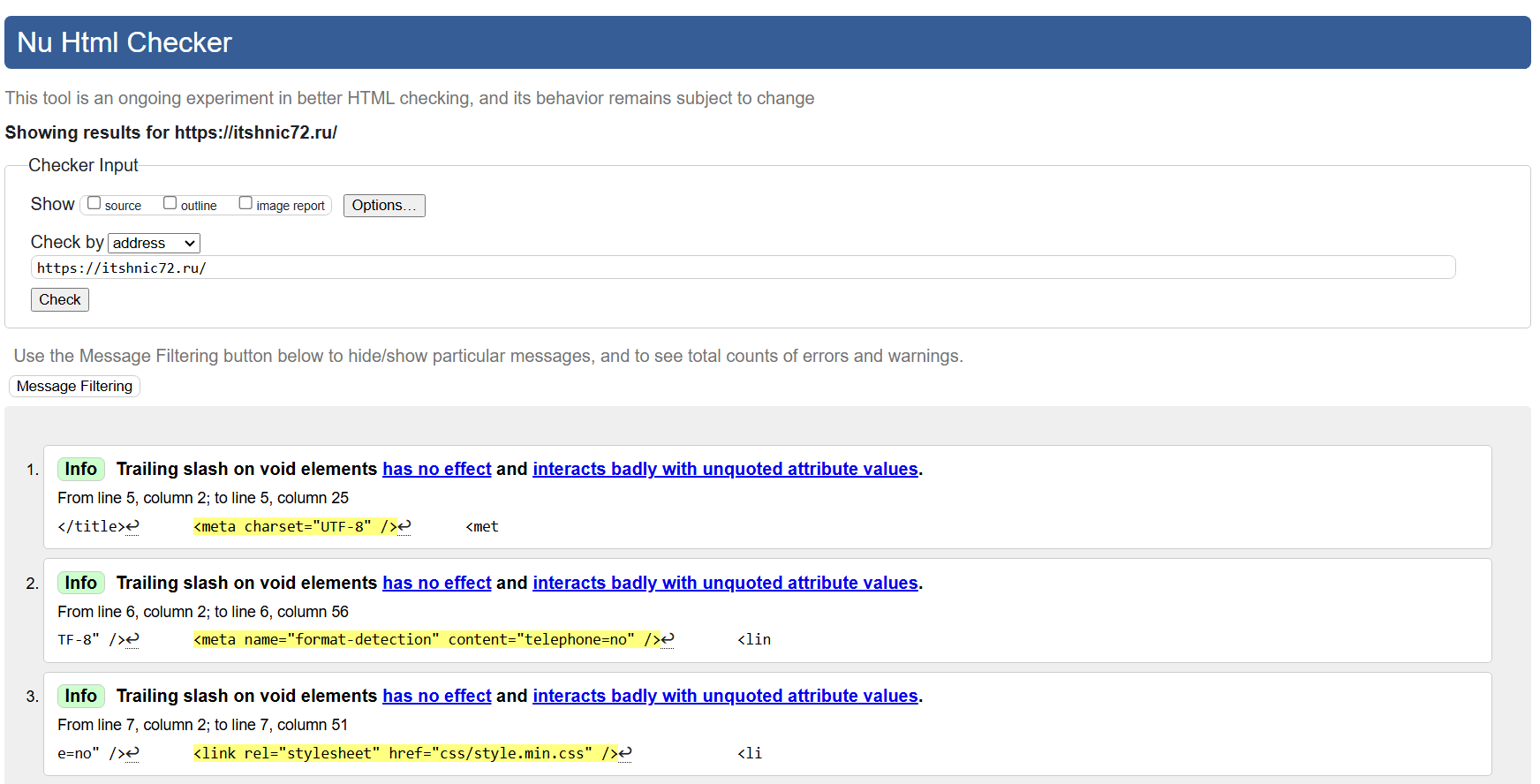
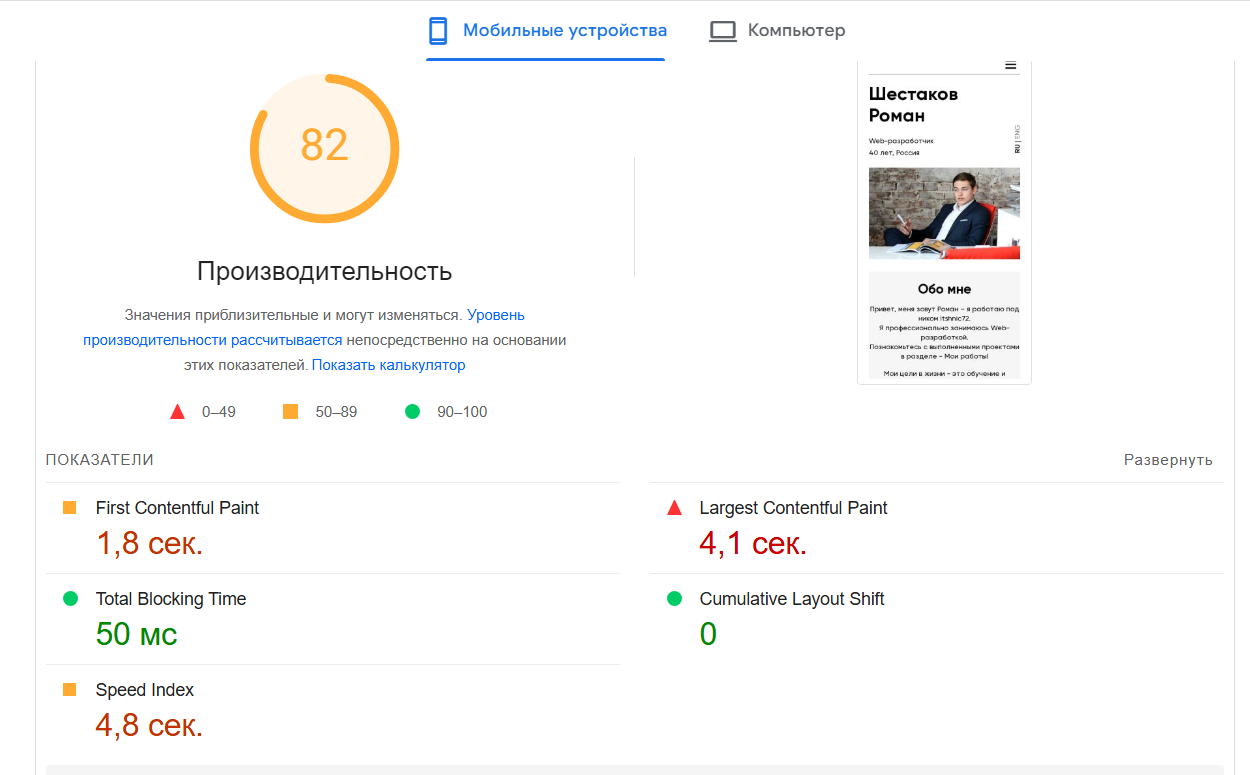


Рисунок 77. Валидность

Проверка производительности и скорости загрузки через сервис [PageSpeed Insights](https://pagespeed.web.dev/)



[Рисунок 78. Производительность](https://pagespeed.web.dev/)

Выводы: Производительность сайта – хорошая. Требуется поработать с валидацией.

# Заключение

В заключение необходимо включить следующее:

1. На данном этапе разработки цели достигнуты – получилось приложение которое способно донести до потенциального клиента или работодателя, информацию о исполнителе или соискателе <https://itshnic72.ru/>.
2. Для разработки приложения использовал сборщик Gulp (помимо основного обучения в 2023 году - решил изучить Gulp и создать свой шаблон для удобной работы). На данный момент считаю Gulp устаревшим инструментом. Изучаю React – более функциональный фреймворк.
3. Приложение доступно из любой точки мира и в любое время. Адаптировано под любой гаджет. Можно начинать использовать web-portfolio как инструмент коммуникации с клиентом.
4. Сейчас доступен первый релиз приложения, но его нужно усовершенствовать – подключить к СУБД (для сохранения входящих запросов, клиентов и др.).

Возможно создать раздел для администратора – управление контентом и загрузка готовых работ, без входа на хостинг.

# Список используемой литературы

1. Принципы качества веб-сайтов по культуре / ред. М.Т.Н. Тем
2. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. / Никсон Робин
3. Head First. Паттерны проектирования / Фримен Эрик, Робсон Элизабет
4. Git для профессионального программиста / Чакон Скотт, Штрауб Бен
5. Wikipedia (<https://ru.wikipedia.org/wiki/Заглавная_страница>)
6. JAVASCRIPT.RU (<https://learn.javascript.ru/>)
7. Sass (https://sass-scss.ru/)

# Приложения

1. Node.js
2. Npm (<https://www.npmjs.com/>)
3. Gulp (<https://gulpjs.com/>)
4. GitHub ([https://github.com/itshnic/degree\_project/](https://github.com/))
5. Web-portfolio (https://itshnic72.ru/)