Geekbrains

Разработка веб-приложения с использованием менеджера задач GULP

Программа:
Разработчик
Кошель В.Э.

г. Москва
2024 г.

Содержание

Введение……...…3

Глава 1. Анализ существующих решений. Выбор методов, инструментов и технологий.

1.1 Выбор инструмента для выполнения задач……..……..4

1.2 Выбор фреймворков и библиотек для решения отдельных задач……..……..8

1.3 Выбор сервиса для разработки интерфейсов……..……9

Глава 2. Подготовка рабочей среды IDE PhpStorm для разработки

2.1 Установка и настройка Gulp……..…….10

2.2 Установка менеджера отправки e-mail сообщений PHPMailer…….....…..21

2.3 Подключение иконочных шрифтов Font Awesome…….…..……26

2.4 Подключение адаптивного слайдера Slick Slider…….……27

2.5 Ознакомление с макетом веб-приложения в Figma…….…..……29

Глава 3. Разработка веб-приложения

3.1 Верстка веб-приложения, компонентный подход…….....……31

3.2 Стилизация веб-приложения, компонентный подход……...…38

3.3 Оживление веб-приложения на JavaScript……...……..44

3.4 Проверка работоспособности. Сборка проекта…….48

Заключение……...……..49

Список используемой литературы……...……51

Введение

Актуальность. Часто разработчик сталкивается с проблемой недостатка времени при выполнении проекта и наличия большого количества однотипных задач, связанных с написанием многостраничных веб-проектов не являющихся SPA или WPA приложениями и без использования современных JS-фреймворков.

Целью выпускной квалификационной работы является разработка веб-приложения с использованием современного менеджера задач GULP, а также организация рабочего пространства для выполнения однотипных повседневных задач.

Задачи работы:

- Анализ существующих решений, методов, технологий и инструментов, выбор решения;
- Организация рабочего пространства на основе выбранного решения;
- Создание веб-приложения.

Структура работы. Работа состоит из трех частей. Первые две части теоретические, третья часть – практическая, заключение, список литературы и приложений.

Глава 1. Анализ существующих решений. Выбор методов, инструментов и технологий.

1.1 Выбор инструмента для выполнения задач.

В последнее время build-системам стали уделять всё больше внимания, и, под влиянием этого, их в одночасье стало настолько много, что, честно говоря, уже всех и не упомнить. Самыми востребованными и конкурентноспособными на сегодняшний день являются системы: Grunt, Gulp, Webpack, Brunch и Parcel.

При работе с веб-проектом разработчику необходимо выполнять ряд повторяющихся операций:

- проверять HTML-разметку и CSS-правила;
- преобразовывать синтаксис CSS-препроцессоров в «чистый» CSS;
- проверять JS-код на соответствие требуемым стандартам;
- при необходимости объединять несколько файлов (CSS или JS) в один;
- минифицировать большие файлы;
- проверять в браузере результат, желательно по принципу «live-reload» («живая перезагрузка»);
- формировать итоговую сборку проекта.

Выполнение всех этих задач можно организовать с применением специальных модулей, которые распространяются средствами пакетных менеджеров или же таск-менеджеров. Я остановился на выборе между двумя первыми таск-менеджерами Grunt и Gulp.

Таск-менеджер - инструмент для автоматизации задач. В конфигурации раннеров можно записать имена этих задач; функцию, которая их выполняет; плагины для ускорения стандартных действий, но сами задачи могут быть

произвольными.

Например:

- Задачи для деплоя (zip проекта, загрузка проекта на удаленный сервер и т.п.)
- Задачи по сборке проекта (минификация, оптимизация, проверка кода на валидность и т.п.)
- Задачи для миграции данных и т.д.

Итак, перед нами два такс-раннера: Gulp и Grunt. Оба используют node.js и npm, а задачи им ставят, используя javascript.

Grunt - таск-раннер, который разработан для Node.js в 2012 году. Сейчас у него около 11 тысяч плагинов — почти вдвое больше, чем у gulp. Он до сих пор применяется в Adobe Systems, jQuery, Twitter, Mozilla, Bootstrap, Cloudant, Opera, WordPress, Walmart и Microsoft. Во многих статьях написано, что «писать для Grunt намного проще, чем потом это читать», и они во многом правы. Поэтому порог вхождения в эту билд систему чрезвычайно низок: достаточно прочитать одну страницу, чтобы иметь представление, как написать свой конфиг. Написание своих плагинов так же не занимает много времени и сил: всю систему вполне возможно изучить всего за несколько часов. Что касается порога вхождения в уже готовые проекты, то тут дела обстоят немного иначе, Gruntfile не так удобно читать, как писать. Представьте себе ситуацию, когда у вас к проекту подключены 20 плагинов и у каждого из них есть зависимости. Например, concat имеет смысл выполнять только после завершения работы плагина coffee и compass, а imagemin должен выполняться только после выполнения задачи copy. Это всё довольно проблематично отследить, даже если вы не используете шаблонизатор для описания ваших задач. По производительности Grunt здесь прилично сдает свои позиции. Во-первых, использовать node.js и не использовать многопоточность — это просто грустно.

Во-вторых, без дополнительных плагинов, Grunt не умеет работать с кешированием измененных файлов. Т.е. если мы изменяем 1 из 10 000 файлов Coffee, то все десять тысяч будут пересобраны. Конечно, многопоточность можно привить с помощью grunt-concurrent, а работу с кешированием — с помощью grunt-newer, но это больше похоже на monkey-patch Grunt'a, чем на полноценную реализацию. Но и это ещё не всё. Grunt имеет весьма кривой механизм для работы с потоками данных. Представьте, что нам сначала требуется собрать coffee, потом соединить все файлы, минифицировать их и прогнать тесты. Все эти действия связаны с одним и тем же набором файлов, однако Grunt этого не понимает и использует для каждой задачи свой поток ввода-вывода, т.е. каждый раз заново считывает файлы и прогоняет для них ту или иную задачу, после чего записывает результат обратно на диск и так далее, пока все задачи не будут выполнены. Согласитесь, крайне затратные операции, особенно с учётом того, что мы можем использовать для этих же целей один поток ввода-вывода информации и все операции над одними и теме же файлами выполнять в оперативной памяти. Это бы позволило достичь намного лучшего результата по производительности.

Gulp - потоковая система сборки (a streaming build system). Создан как ответвление grunt и насчитывает около 4000 плагинов. Начало работы с Gulp такое же быстрое, как и с Grunt. Однако, оно лишено изъянов, которые мы обсуждали в начале работы с Grunt для больших проектов, т.е. Gulp так же легко читать, как и писать, что делает его максимально располагающим к быстрому старту. Если же вы уже использовали какую-либо систему сборки, то время вхождения в Gulp уменьшится в разы. По производительности по сравнению с Grunt, данная система сборки заметно выигрывает. В отличии от своего предшественника, она поддерживает из коробки возможность форка для выполнения несвязанных стеков задач в разных подпроцессах. Это дает весьма ощутимый прирост скорости. Так же стоит отметить, что Gulp использует всего один поток ввода/вывода при выполнении ряда задач с файлами. Это достигается

несколько иным архитектурным решением, нежели использует Grunt. Если при создании Gruntfile вы будете указывать одни и те же маски для файлов в разных задачах, то в Gulpfile ситуация диаметрально противоположная: мы пляшем от файлов, а не от задач. Последовательное выполнение задач делает Gulp быстрым и мощным, но изредка возникает необходимость все же выполнить задачи синхронно, как в Grunt.

Вывод: Grunt и Gulp на данный момент основные системы сборки, которые имеют активный community и множество активных плагинов. Остальные же практически не имеют плагинов, а это основной фактор при выборе между системами сборки. Grunt действительно уступает Gulp по скорости выполнения задач, но Gulp имеет значительно меньше плагинов. Портировать плагины с Grunt на Gulp не очень просто из-за разной архитектуры систем сборки. Мой выбор пал на Gulp.

В качестве среды для программирования был использован PhpStorm. Одним из главных преимуществ использования PhpStorm является его удобный и интуитивно понятный интерфейс. Кроме того, он имеет встроенные возможности отладки, что облегчает разработчику быстрый поиск и исправление ошибок. Ее одним преимуществом является широкий набор функций, которые включают завершение кода, подсветку синтаксиса, интеграцию с системами контроля версий и поддержку множества языков.

1.2 Выбор фреймворков и библиотек для решения отдельных задач.

В процессе разработки веб-приложения необходимо было решить следующие задачи:

- Отправка e-mail сообщений с разрабатываемого сайта;
- Возможность использования иконочного шрифта;
- Использование плагина для адаптивного слайдера.

Для реализации отправки e-mail сообщений мой выбор пал на PHPMailer. Это очень удобная и популярная библиотека. PHPMailer имеет в своем ассортименте, пожалуй, всё, что можно пожелать от работы с почтой: отправка разными способами, через разные серверы в т.ч. через smtp, возможность шифровать и подписывать ваши письма, чтобы не попадали в спам и многое другое.

В качестве инструмента для использования иконочного шрифта в верстке была выбрана библиотека Font Awesome. Эта библиотека предлагает к использованию масштабируемые векторные конки, которые с легкостью можно персонализировать силами CSS.

Для быстрого создания на сайте адаптивного слайдера любой сложности мой выбор пал на плагин Slick slider. Его функционал позволяет реализовать зацикливание, автопроигрывание, эффекты перехода и многое другое. Отдельно выделю возможность пролистывания слайдера пальцем на устройствах с сенсорным экраном.

1.3 Выбор сервиса для разработки интерфейсов.

В качестве инструмента для разработки интерфейсов мною была выбрана одна из самых популярных сред разработки для веб-разработчиков – IDE PhpStorm. Это мощная интегрированная среда разработки (IDE), которая предлагает широкий спектр возможностей и функций, помогающих разработчикам создавать высококачественные веб-сайты и приложения. Но, как и любой инструмент, PhpStorm имеет как преимущества, так и недостатки.

IDE PhpStorm – это удобная среда программирования, которая выделяется своей функциональностью. Умеет работать с фреймворками и специализированными плагинами для ведущих фреймворков PHP.

PhpStorm интегрирован с самыми популярными системами контроля версий, включая Git, Subversion, Mercurial, Perforce, CVS, и TFS. Все однообразные задачи (например, добавление и удаление файлов) выполняются автоматически.

Одним из главных преимуществ использования PhpStorm является его удобный интерфейс. Он разработан так, чтобы быть интуитивно понятным и простым в использовании, поэтому даже новички могут быстро научиться пользоваться программой без особых усилий. Кроме того, он имеет встроенные возможности отладки, что облегчает разработчикам быстрый поиск и исправление ошибок.

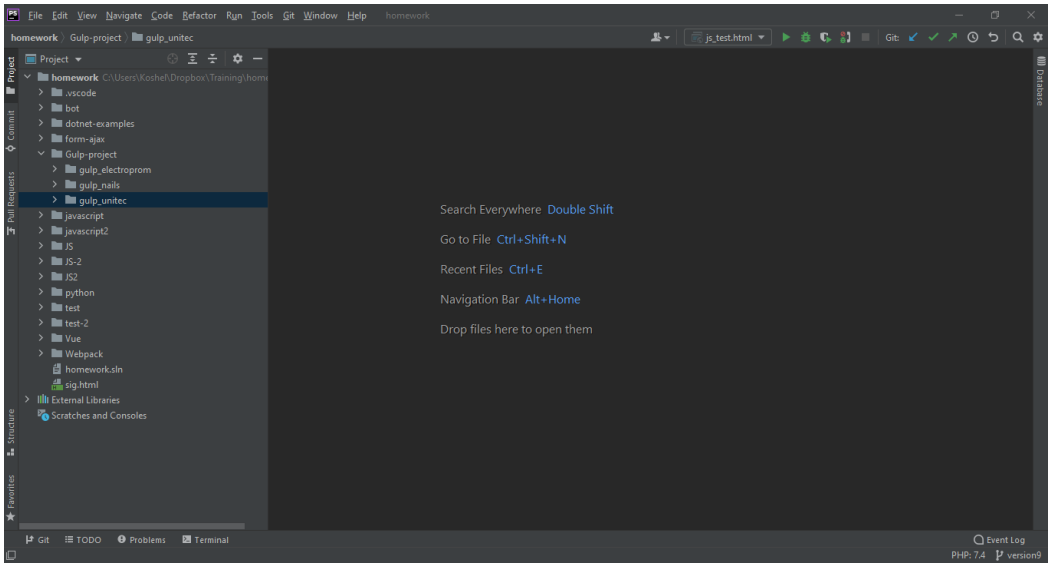
Еще одним преимуществом является широкий набор функций, которые включают завершение кода, подсветку синтаксиса, интеграцию с системами контроля версий, такими как GIT или SVN, поддержку множества языков, включая HTML5, CSS и др.

Глава 2. Подготовка рабочей среды IDE PhpStorm для разработки.

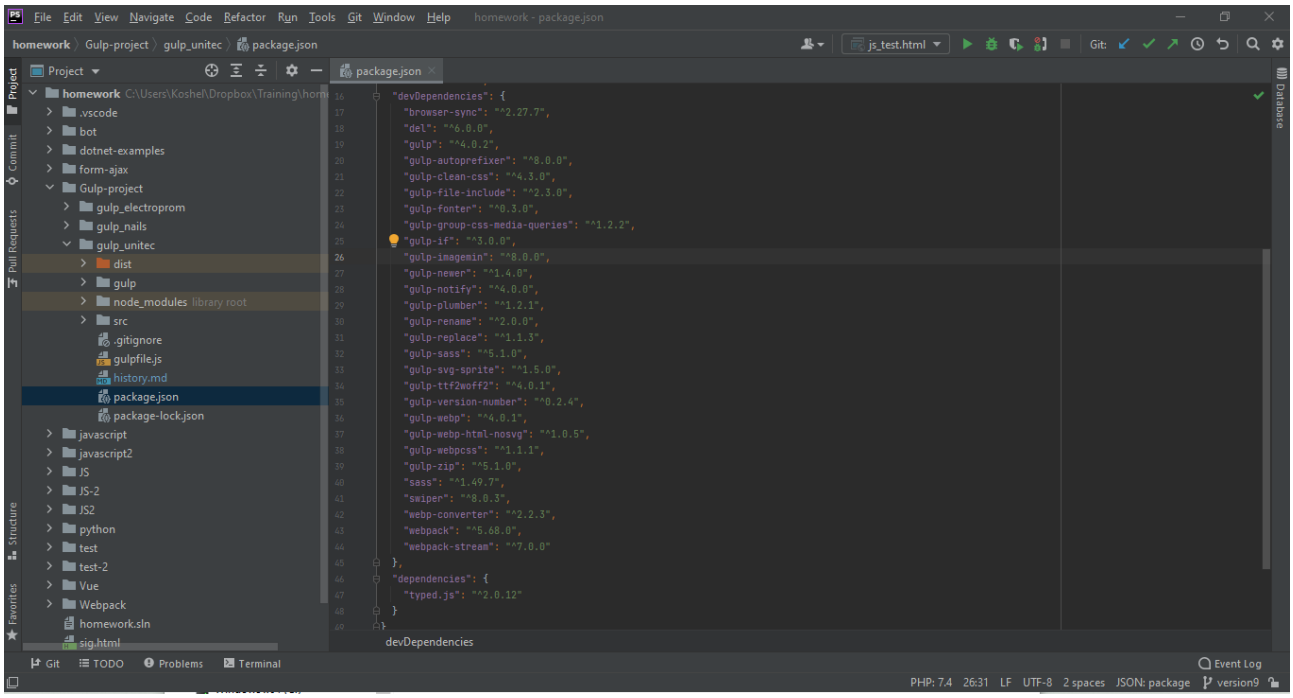
2.1 Установка и настройка Gulp.

Gulp – это менеджер задач и сборщик проектов. В данной главе мы напишем готовую сборку, в которой Gulp будет самостоятельно конвертировать и подключать шрифты, а также записывать подключение в файл стилей. Обновляеть браузер. Собираеть в один несколько HTML\PUG файлов. Работаеть с CSS препроцессорами, например SASS(SCSS) и оптимизировать их. Gulp будет не только сжимает изображения но и конвертировать их в современный формат WEBP, а также самостоятельно подключать результат к HTML и CSS файлам. Собираеть в один несколько JavaScript файлов с помощью WEBPACK, оптимизировать и сжимаь их. Есть возможность работать в синтаксисе ES6. Будет уметь создавать SVG спрайты, создавать ZIP архив с результатом и отправлять готовую верстку на FTP сервер и многое другое!

1. Устанавливаем в нашу операционную систему актуальную версию Node.js (https://nodejs.org/en);
2. Устанавливаем систему контроля версий Git (https://git-scm.com);
3. Создаем папку нашего проекта gulp_unitec и открываем ее в нашем IDE PhpStorm



4. Создаем файл package.json командой из терминала npm init и устанавливаем все необходимые нам для проекта зависимости при помощи менеджера пакетов npm:



5. Глобально устанавливаем пакет Gulp командой npm i gulp-cli-g;
6. Создаем главный файл сценариев gulpfile.js следующего содержания

```
// Основной модуль
import gulp from "gulp";
// Импорт путей
import { path } from "./gulp/config/path.js";
// Импорт общих плагинов
import { plugins } from "./gulp/config/plagins.js";

// Передаём значения в глобальную переменную
global.app = {
  isBuild: process.argv.includes('--build'),
  isDev: !process.argv.includes('--build'),
  path: path,
  gulp: gulp,
  plugins: plugins
}

// Импорт задач
import { copy } from "./gulp/tasks/copy.js";
import { reset } from "./gulp/tasks/reset.js";
import { html } from "./gulp/tasks/html.js";
import { server } from "./gulp/tasks/server.js";
import { scss } from "./gulp/tasks/scss.js";
import { js } from "./gulp/tasks/js.js";
import { images } from "./gulp/tasks/images.js";
```

```
import { otfToTtf, ttfToWoff, fontsStyle } from "./gulp/tasks/fonts.js";
import { svgSpr } from "./gulp/tasks/svg.js";
import { zip } from "./gulp/tasks/zip.js";

// Наблюдатель за изменениями в файлах
function watcher() {
  gulp.watch(path.watch.files, copy);
  gulp.watch(path.watch.html, html);
  gulp.watch(path.watch.scss, scss);
  gulp.watch(path.watch.js, js);
  gulp.watch(path.watch.images, images);
}

export { svgSpr }

// Последовательная обработка шрифтов
const fonts = gulp.series(otfToTtf, ttfToWoff, fontsStyle);

// Основные задачи
const mainTasks = gulp.series(fonts, gulp.parallel(copy, html, scss, js, images));

// Построение сценариев выполнения задач
const dev = gulp.series(reset, mainTasks, gulp.parallel(watcher, server));
const build = gulp.series(reset, mainTasks);
const deployZIP = gulp.series(reset, mainTasks, zip);

// Экспорт сценариев
export { dev }
export { build }
export { deployZIP }

// Выполнение сценария по умолчанию
gulp.task('default', dev);
```

7. Создаем папку src с исходниками нашего проекта;
8. Создаем папку gulp для работы с вспомогательными файлами Gulp;
9. В папке gulp создадим папку config с файлами path.js для настройки путей

```
import * as nodePath from 'path';
const rootFolder = nodePath.basename(nodePath.resolve());
const buildFolder = './dist';
const srcFolder = './src';

export const path = {
  build: {
    js: `${buildFolder}/js/`,
    css: `${buildFolder}/css/`,
    html: `${buildFolder}/`,
    images: `${buildFolder}/img/`,
    fonts: `${buildFolder}/fonts/`,
    files: `${buildFolder}/files/`,
  },
  src: {
    js: `${srcFolder}/js/app.js`,
    images: `${srcFolder}/img/**/*.{jpg,jpeg,png,gif,webp}`,
```

```
    svg: `${srcFolder}/img/**/*.svg`,
    scss: `${srcFolder}/scss/style.scss`,
    html: `${srcFolder}/*.{html,xml}`,
    files: `${srcFolder}/files/**/*.*`,
    svgicons: `${srcFolder}/svgicons/*.svg`,
  },
  watch: {
    js: `${srcFolder}/js/**/*.js`,
    scss: `${srcFolder}/scss/**/*.scss`,
    html: `${srcFolder}/**/*.{html,xml}`,
    images: `${srcFolder}/img/**/*.{jpg,jpeg,png,svg,gif,ico,webp}`,
    files: `${srcFolder}/files/**/*.*`,
  },
  clean: buildFolder,
  buildFolder: buildFolder,
  srcFolder: srcFolder,
  rootFolder: rootFolder,
  ftp: ``
}
```

и файл plagins.js для размещения плагинов при работе с различными сценариями:

```
import replace from "gulp-replace"; // Поис и замена
import plumber from "gulp-plumber"; // Обработка ошибок
import notify from "gulp-notify"; // Сообщения, подсказки
import browsersync from "browser-sync"; // Локальный сервер
import newer from "gulp-newer"; // Проверка обновления images
import ifPlugin from "gulp-if"; // Условное ветвление

// Экспортируем объект
export const plugins = {
  replace: replace,
  plumber: plumber,
  notify: notify,
  browsersync: browsersync,
  newer: newer,
  if: ifPlugin,
}
```

10. В папке gulp создадим папку tasks для постановки отдельных задач.

11. Создаем файл copy.js для копирования файлов:

```
export const copy = () => {
  return app.gulp.src(app.path.src.files)
    .pipe(app.gulp.dest(app.path.build.files))
}
```

12.Создаем файл fonts.js для работы с шрифтами:

```
import fs, { appendFile } from 'fs';
import fonter from 'gulp-fonter';
import ttf2woff2 from 'gulp-ttf2woff2';

export const otfToTtf = () => {
```

```
  // ищем файлы шрифтов .atf
  return app.gulp.src(`${app.path.srcFolder}/fonts/*.otf`, {})
    .pipe(app.plugins.plumber(
      app.plugins.notify.onError({
        title: "FONTS",
        message: "Error: <%= error.message %>"
      })
    ))
    // Конвертируем в .ttf
    .pipe(fonter({
      formats: ['ttf']
    }))
    // выгружаем в исходную папку
    .pipe(app.gulp.dest(`${app.path.srcFolder}/fonts/`))
}

export const ttfToWoff = () => {
  // ищем файлы шрифтов .ttf
  return app.gulp.src(`${app.path.srcFolder}/fonts/*.ttf`, {})
    .pipe(app.plugins.plumber(
      app.plugins.notify.onError({
        title: "FONTS",
        message: "Error: <%= error.message %>"
      })
    ))
    // Конвертируем в .woff
    .pipe(fonter({
      formats: ['woff']
    }))
    // выгружаем в папку с результатом
    .pipe(app.gulp.dest(`${app.path.build.fonts}`))
    // Ищем файлы шрифтов .ttf
    .pipe(app.gulp.src(`${app.path.srcFolder}/fonts/*.ttf`))
    // конвертируем в .wooff2
    .pipe(ttf2woff2())
    // Выгружаем в папку с результатом
    .pipe(app.gulp.dest(`${app.path.build.fonts}`));
}

export const fontsStyle = () => {
  // Файл стилей подключения шрифтов
  let fontsFile = `${app.path.srcFolder}/scss/fonts.scss`;
  // Проверяем существуют ли файлы шрифтов
  fs.readdir(app.path.build.fonts, function (err, fontsFiles) {
    if (fontsFiles) {
      // Проверяем существует ли файл стилей для подключения шрифтов
      if (!fs.existsSync(fontsFile)) {
        // Если файла нет, создаем его
        fs.writeFile(fontsFile, '', cb);
        let newFileOnly;
        for (var i = 0; i < fontsFiles.length; i++) {
          // Записываем подключение шрифтов в файл стилей
          let fontFileName = fontsFiles[i].split('.')[0];
          if (newFileOnly !== fontFileName) {
            let fontName = fontFileName.split('-')[0] ? fontFileName.split('-')[0] :
fontFileName;
            let fontWeight = fontFileName.split('-')[1] ? fontFileName.split('-')[1] :
fontFileName;
            if (fontWeight.toLowerCase() === 'thin') {
              fontWeight = 100;
```

```
            } else if (fontWeight.toLowerCase() === 'extralight') {
              fontWeight = 200;
            } else if (fontWeight.toLowerCase() === 'light') {
              fontWeight = 300;
            } else if (fontWeight.toLowerCase() === 'medium') {
              fontWeight = 500;
            } else if (fontWeight.toLowerCase() === 'semibold') {
              fontWeight = 600;
            } else if (fontWeight.toLowerCase() === 'bold') {
              fontWeight = 700;
            } else if (fontWeight.toLowerCase() === 'extrabold' ||
fontWeight.toLowerCase() === 'heavy') {
              fontWeight = 800;
            } else if (fontWeight.toLowerCase() === 'black') {
              fontWeight = 900;
            } else {
              fontWeight = 400;
            }
            fs.appendFile(fontsFile, `@font-face {\n\tfont-family: ${fontName};\n\tfont-
display: swap;\n\tsrc: url("../fonts/${fontFileName}.woff2") format("woff2"),
url("../fonts/${fontFileName}.woff") format("woff");\n\tfont-weight:
${fontWeight};\n\tfont-style: normal;\n}\r\n`, cb);
            newFileOnly = fontFileName;
          }
        }
      } else {
        // Если файл есть, выводим сообщение
        console.log("Файл scss/fonts.scss уже существует. Для обновления файла нужно его
удалить!");
      }
    }
  });

  return app.gulp.src(`${app.path.srcFolder}`);
  function cb() { }
}
```

13. Создаем файл html.js для работы с файлами html:

```
import fileinclude from "gulp-file-include";
import webpHtmlNosvg from "gulp-webp-html-nosvg";
import versionNumber from "gulp-version-number";

export const html = () => {
  return app.gulp.src(app.path.src.html)
    .pipe(app.plugins.plumber(
      app.plugins.notify.onError({
        title: "HTML",
        message: "Error: <%= error.message %>"
      }))
    )
    .pipe(fileinclude())
    .pipe(app.plugins.replace(/@img\//g, 'img/'))
    .pipe(
      app.plugins.if(
        app.isBuild,
        webpHtmlNosvg()
      )
    )
```

```
    .pipe(
      app.plugins.if(
        app.isBuild,
        versionNumber({
          'value': '%DT%',
          'append': {
            'key': '_v',
            'cover': 0,
            'to': [
              'css',
              'js',
            ]
          },
          'output': {
            'file': 'gulp/version.json'
          }
        })
      )
    )
    .pipe(app.gulp.dest(app.path.build.html))
    .pipe(app.plugins.browsersync.stream());
}
```

14. Создаем файл images.js для работы с файлами изображений:

```
import webp from "gulp-webp";
import imagemin from "gulp-imagemin";

export const images = () => {
  return app.gulp.src(app.path.src.images)
    .pipe(app.plugins.plumber(
      app.plugins.notify.onError({
        title: "IMAGES",
        message: "Error: <%= error.message %>"
      }))
    )
    .pipe(app.plugins.newer(app.path.build.images))
    .pipe(
      app.plugins.if(
        app.isBuild,
        webp()
      )
    )
    .pipe(
      app.plugins.if(
        app.isBuild,
        app.gulp.dest(app.path.build.images)
      )
    )
    .pipe(
      app.plugins.if(
        app.isBuild,
        app.gulp.src(app.path.src.images)
      )
    )
    .pipe(
      app.plugins.if(
        app.isBuild,
        app.plugins.newer(app.path.build.images)
```

```
      )
    )
    .pipe(
      app.plugins.if(
        app.isBuild,
        imagemin({
          progressive: true,
          svgoPlugins: [{ removeViewBox: false }],
          interlaced: true,
          optimizationLevel: 3 // 0 to 7
        })
      )
    )
    .pipe(app.gulp.dest(app.path.build.images))
    .pipe(app.gulp.src(app.path.src.svg))
    .pipe(app.gulp.dest(app.path.build.images))
    .pipe(app.plugins.browsersync.stream());
}
```

15. Создаем файл js.js для работы с файлами javascript через webpack:

```
import webpack from 'webpack-stream';

export const js = () => {
  return app.gulp.src(app.path.src.js, { sourcemaps: app.isDev })
    .pipe(app.plugins.plumber(
      app.plugins.notify.onError({
        title: "JS",
        message: "Error: <%= error.message %>"
      }))
    )
    .pipe(webpack({
      // mode: 'production', // режим сборки или/или !!!
      mode: app.isBuild ? 'production' : 'development',
      output: {
        filename: 'app.min.js'
      }
    }))
    .pipe(app.gulp.dest(app.path.build.js))
    .pipe(app.plugins.browsersync.stream());
}
```

16. Создаем файл reset.js для очистки:

```
import del from "del";
export const reset = () => {
  return del(app.path.clean);
}
```

17. Создаем файл scss.js для работы с препроцессорами:

```
import dartSass from 'sass';
import gulpSass from 'gulp-sass';
import rename from 'gulp-rename';

import cleanCss from 'gulp-clean-css'; // Сжатие CSS файлов
import webpcss from 'gulp-webpcss'; // Вывод WEBP изображений
```

```
import autoprefixer from 'gulp-autoprefixer'; // Добавление вендорных префиксов
import groupCssMediaQueries from 'gulp-group-css-media-queries'; // Группировка медиа
запросов

const sass = gulpSass(dartSass);

export const scss = () => {
  return app.gulp.src(app.path.src.scss, { sourcemaps: app.isDev })
    .pipe(app.plugins.plumber(
      app.plugins.notify.onError({
        title: "SCSS",
        message: "Error: <%= error.message %>"
      })))
    .pipe(app.plugins.replace(/@img\//g, '../img/'))
    .pipe(sass({
      outputStyle: 'expanded'
    }))
    // .pipe(
    //   app.plugins.if(
    //     app.isBuild,
    //     groupCssMediaQueries()
    //   )
    // )
    .pipe(groupCssMediaQueries())
    // .pipe(
    //   app.plugins.if(
    //     app.isBuild,
    //     webpcss(
    //       {
    //         webpClass: ".webp",
    //         noWebpClass: ".no-webp"
    //       }
    //     )
    //   )
    // )
    .pipe(
      webpcss(
        {
          webpClass: ".webp",
          noWebpClass: ".no-webp"
        }
      )
    )
    // .pipe(
    //   app.plugins.if(
    //     app.isBuild,
    //     autoprefixer({
    //       grid: true,
    //       overrideBrowserslist: ["last 3 versions"],
    //       cascade: true
    //     })
    //   )
    // )
    .pipe(
      autoprefixer({
        grid: true,
        overrideBrowserslist: ["last 3 versions"],
        cascade: true
      })
    )
```

```
    // .pipe(app.gulp.dest(app.path.build.css)) // закоментировать, если не сжатый не
нужен!!!
    // .pipe(
    //   app.plugins.if(
    //     app.isBuild,
    //     cleanCss()
    // )
    .pipe(
      cleanCss()
    ) // сжатие css

    .pipe(rename({
      extname: ".min.css"
    }))
    .pipe(app.gulp.dest(app.path.build.css))
    .pipe(app.plugins.browsersync.stream());
}
```

18. Создаем файл server.js для организации виртуального сервера:

```
export const server = (done) => {
  app.plugins.browsersync.init({
    server: {
      baseDir: `${app.path.build.html}`
    },
    notify: false,
    port: 3000,
  });
}
```

19. Создаем файл svg.js для работы с svg изображениями:

```
import svgSprite from "gulp-svg-sprite"

export const svgSpr = () => {
  return app.gulp.src(`${app.path.src.svgicons}`, {})
    .pipe(app.plugins.plumber(
      app.plugins.notify.onError({
        title: "SVG",
        message: "Error: <% error.message %>"
      })
    ))
    .pipe(svgSprite({
      mode: {
        stack: {
          sprite: `../icons/icons.svg`,
          // Создавать страницу с перечнем иконок
          example: true
        }
      },
    }))
    .pipe(app.gulp.dest(`${app.path.build.images}`));
}
```

20. Создаем файл zip.js для упаковки проекта в архив:

```
import del from "del";
import zipPlugin from "gulp-zip";

export const zip = () => {
  del(`./${app.path.rootFolder}.zip`);
  return app.gulp.src(`${app.path.buildFolder}/**/*.*`, {})
    .pipe(app.plugins.plumber(
      app.plugins.notify.onError({
        title: "ZIP",
        message: "Error: <%= error.message %>"
      })
    ))
    .pipe(zipPlugin(`${app.path.rootFolder}.zip`))
    .pipe(app.gulp.dest('./'));
}
```

Настройка Gulp среды завершена. Запуск сборки производится в нескольких режимах в зависимости от стадии разработки проекта. Добавим в json-файл проекта следующие функции командной строки: режим разработчика, режим сборки проекта и режим архивирования проекта в zip-архив.

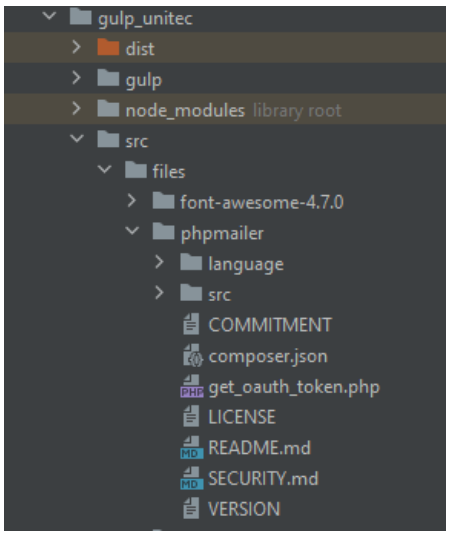
```
"dev": "gulp",
"build": "gulp build --build",
"zip": "gulp deployZIP --build",
```

2.2 Установка менеджера отправки e-mail сообщений PHPMailer.

PHPMailer – полнофункциональный класс создания и передачи электронной почты для PHP. Функциональные возможности PHPMailer:

- Используется многими проектами с открытым исходным кодом: WordPress, Drupal, 1CRM, SugarCRM, Yii, Joomla! и многими другими;
- Интегрированная поддержка SMTP — отправка без локального почтового сервера;
- Отправка электронных писем с несколькими адресами «Кому», «Копия», «Скрытая копия» и «Обратный адрес»;
- Многокомпонентные/альтернативные письма для почтовых клиентов, которые не читают HTML-письма;
- Позволяет добавлять вложения, в том числе встроенные;
- Поддержка содержимого UTF-8, а также 8-битных, base64, двоичных и цитируемых кодировок;
- Аутентификация SMTP с механизмами LOGIN, PLAIN, CRAM-MD5 и XOAUTH2 через транспорты SMTPS и SMTP+STARTTLS;
- Автоматически проверяет адреса электронной почты;
- Защищает от атак с внедрением заголовков;
- Сообщения об ошибках на более чем 50 языках;
- Поддержка подписей DKIM и S/MIME;
- Совместимо с PHP 5.5 и более поздними версиями, включая PHP 8.2.

В качестве альтернативного метода работы с PHPMailer скачал zip-архив с исходниками на ресурсе разработчика библиотеки github https://github.com/PHPMailer/PHPMailer и организовал отдельную папку в проекте:



В качестве сценария для отправки письма на электронную почту написал небольшой код на php на основе предложенных примеров разработчиком библиотеки:

```
<?php
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

require 'phpmailer/src/Exception.php';
require 'phpmailer/src/PHPMailer.php';

$mail = new PHPMailer(true);
$mail->CharSet = 'UTF-8';
$mail->setLanguage('ru', 'phpmailer/language/');
$mail->IsHTML(true);
// от кого письмо
$mail->setFrom('info@unitecsystem.ru', 'Письмо с сайта Юнитек');
// кому отправить
$mail->addAddress('website@electroprom.com');
// тема письма
$mail->Subject = 'Письмо с сайта Юнитек';
// Тело письма
$body = '<h1>Письмо с сайта</h1>';
if (trim(!empty($_POST['firstname']))) {
    $body .= '<p><strong>Имя:</strong> ' . $_POST['firstname'] . '</p>';
}
if (trim(!empty($_POST['telephone']))) {
    $body .= '<p><strong>Телефон:</strong> ' . $_POST['telephone'] . '</p>';
}
if (trim(!empty($_POST['email']))) {
    $body .= '<p><strong>E-mail:</strong> ' . $_POST['email'] . '</p>';
}
if (trim(!empty($_POST['message']))) {
    $body .= '<p><strong>Сообщение:</strong> ' . $_POST['message'] . '</p>';
}
$mail->Body = $body;
//Отправляем
if (!$mail->send()) {
    $message = 'Ошибка';
} else {
    $message = 'Благодарим Вас, данные успешно отправлены!';
}
$response = ['message' => $message]
header('Content-type: application/json');
echo json_encode($response);
?>
```

Код обработки данных заполненной формы пользователем с проверками выглядит следующим образом:

```
const form = document.querySelector('#question-form');
const modalThanksBg = document.querySelector('.modal__thanks-bg');
const modalThanks = document.querySelector('#modal-thanks');
const buttonCloseThanks = document.querySelector('.thanks__button');
```

```
const modalErrorBg = document.querySelector('.modal__error-bg');
const modalError = document.querySelector('#modal-error');
const buttonCloseError = document.querySelector('.error__button');

form.addEventListener('submit', formSend);
async function formSend(event) {
    event.preventDefault();
    let error = formValidate(form);
    const body = document.querySelector('body');
    let formData = new FormData(form);
    if (error === 0) {
        body.classList.add('send-form');
        let response = await fetch('./files/sendmail.php', {
            method: 'POST',
            body: formData
        });
        if (response.ok) {
            let result = await response.json();
            modalThanksBg.classList.add('active');
            modalThanks.classList.add('active');
            document.body.classList.add('modal');
            buttonCloseThanks.addEventListener('click', (event) => {
                document.body.classList.remove('modal');
                modalThanksBg.classList.remove('active');
                modalThanks.classList.remove('active');
            })
            document.addEventListener('click', (event) => {
                if(event.target === modalThanksBg) {
                    document.body.classList.remove('modal');
                    modalThanksBg.classList.remove('active');
                    modalThanks.classList.remove('active');
                }
            })
            form.reset();
            body.classList.remove('send-form');

        } else {
            modalErrorBg.classList.add('active');
            modalError.classList.add('active');
            document.body.classList.add('modal');
            buttonCloseError.addEventListener('click', () => {
                document.body.classList.remove('modal');
                modalErrorBg.classList.remove('active');
                modalError.classList.remove('active');
            })
            document.addEventListener('click', (event) => {
                if(event.target === modalErrorBg) {
                    document.body.classList.remove('modal');
                    modalErrorBg.classList.remove('active');
                    modalError.classList.remove('active');
                }
            })
            body.classList.remove('send-form');
        }
    }
}

function formValidate(form) {
    let error = 0;
    let formReq = form.querySelectorAll('._req');
```

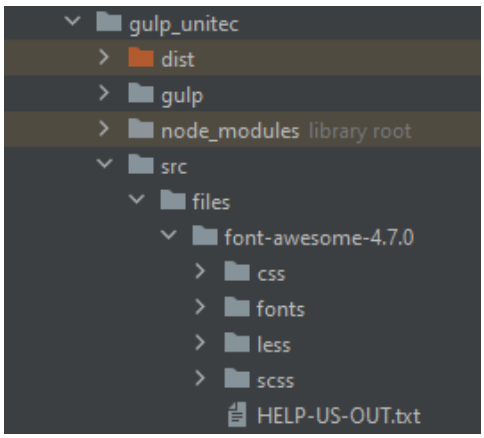
```
    for (let index = 0; index < formReq.length; index++) {
        const input = formReq[index];
        formRemoveError(input);
        if (input.classList.contains('_email')) {
            if (emailTest(input)) {
                formAddError(input);
                error++;
            }
        }
        else if (input.classList.contains('_telephone')) {
            if (telephoneTest(input)) {
                formAddError(input);
                error++;
            }
        }
        else if (input.getAttribute("type") === "radio" && input.checked === false) {
            formAddError(input);
            error++;
        }
        else if (input.getAttribute("type") === "checkbox" && input.checked === false) {
            formAddError(input);
            error++;
        }
        else if (input.getAttribute("type") === "text" && input.value.trim() === '') {
            formAddError(input);
            error++;
        }
        else {
            if (input.classList.contains('_text') && input.value.trim() === '') {
                formAddError(input);
                error++;
            }
        }
    }
    return error;
}

function formAddError(input) {
    input.classList.add('_error');
}

function formRemoveError(input) {
    input.classList.remove('_error');
}

//функция проверки e-mail
function emailTest(input) {
    return !/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,8})+$/.test(input.value);
}
// Функция проверки телефонного номера
function telephoneTest(input) {
    return !/^[+]?[(]?[0-9]{1,4}[)]?[-\s.]?[0-9]{1,4}[-\s.]?[0-
9]{1,4}$/.test(input.value);
}
```

Представленный выше код позволяет обработать будущую форму ввода данных пользователем на сайте, проверить валидность введенных данных

(пробелы, функция проверки e-mail адреса, функция проверки номера телефона и т.п.) и отправить данные методом «Post».

2.3 Подключение иконочных шрифтов Font Awesome.

В качестве иконочного шрифта и css-инструментария была выбрана библиотека Font Awesome, предлагающая масштабируемые векторные иконки, которые можно с легкостью персонализировать и сделать своими стилями css. Подключение иконочного шрифта выбрал с использованием установленных в отдельную папку файлов исходников css:



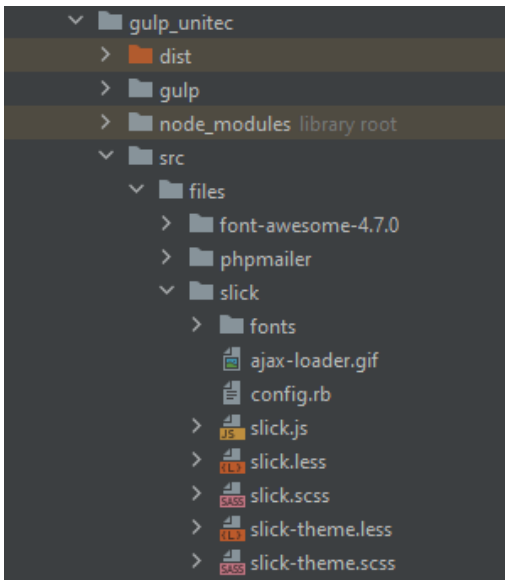
Подключение выполнил ссылкой:

```
<!-- Font Awesome -->
<link rel="stylesheet" href="./files/font-awesome-4.7.0/css/font-awesome.min.css" />
```

2.4 Подключение адаптивного слайдера Slick Slider.

Одной из задач при разработке практически любого веб-приложения является реализация динамического блока страницы, который показывает определенное количество элементов по очередности с заданными характеристиками. Как правило, речь идет о блоках с фотографиями, которые меняются через определенный промежуток времени.

Подключение адаптивного слайдера Slick выбрал с использованием установленных в отдельную папку файлов исходников:



Учитывая, что слайдер Slick требует установленной библиотеки JQuery подключение выполнил ссылками:

```
<!-- JQuery -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<!-- Slick -->
<link rel="stylesheet" type="text/css" href="./files/slick/slick.css" />
<link rel="stylesheet" type="text/css" href="./files/slick/slick-theme.css" />
```

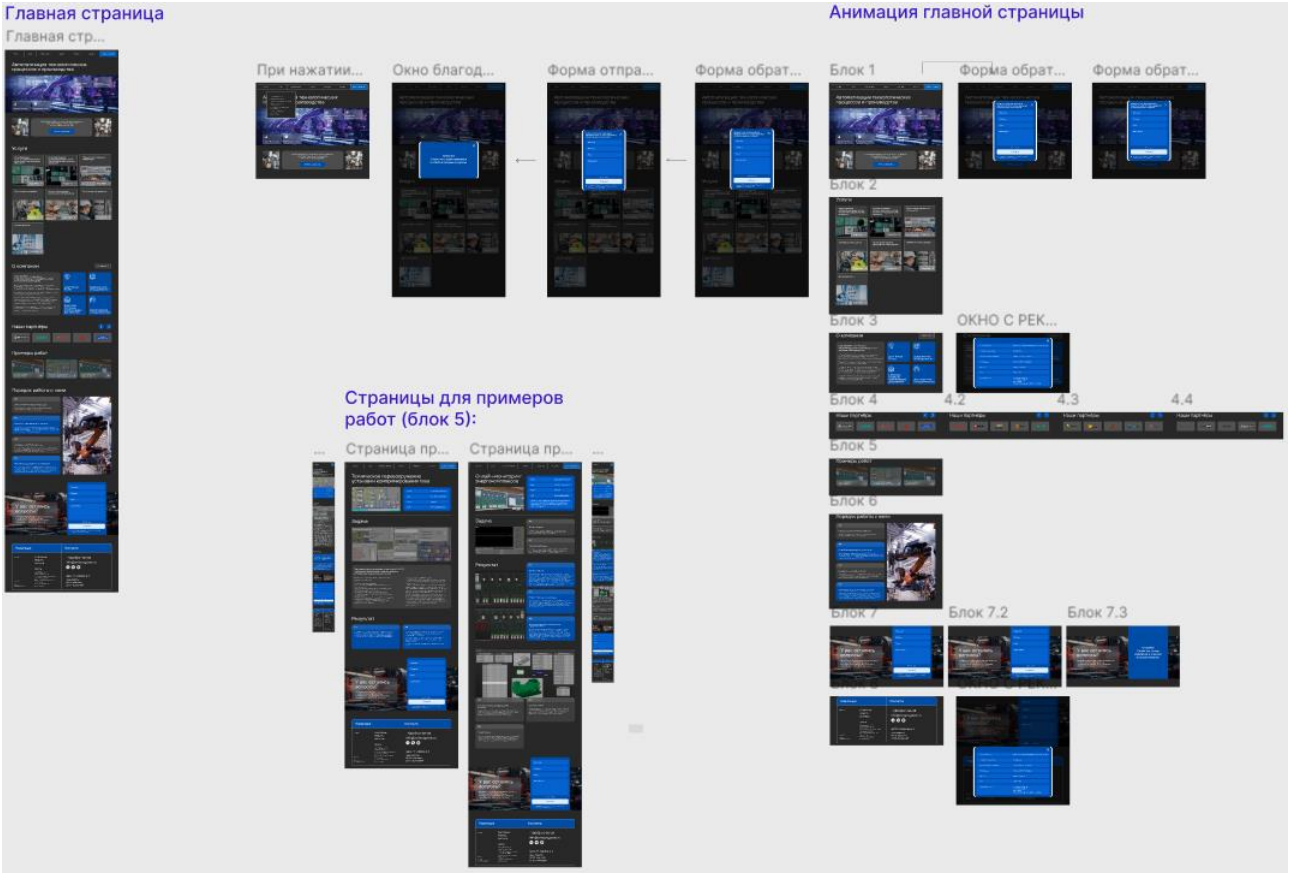
Для реализации дополнительного функционала слайдера подключил js-файл конфигурации с необходимыми мне настройками (автопрокрутка, зацикленная прокрутка, показывать определенное количество слайдов при разных разрешениях экрана, реализация кастомных указателей прокрутки):

```
$(function () {
  $('.clients__slider').slick({
    dots: false,
    infinite: true,
    slidesToShow: 4,
    slidesToScroll: 1,
    autoplay: true,
    autoplaySpeed: 1500,
    arrows: true,
    appendArrows: $('.clients__box-buttons'),
    prevArrow: '<button class="clients-btn clip-buttons" aria-label="Next"
type="button">\n' +
        '            <svg width="31" height="54" viewBox="0 0 31 54" fill="none"\n' +
        '
xmlns="http://www.w3.org/2000/svg">\n' +
        '            <path d="M29 51.5L3 29L29 2" stroke="white" stroke-width="4" stroke-
linecap="round"/>\n' +
        '          </svg>\n' +
        '          </button>',
    nextArrow: '<button class="clients-btn clip-buttons" aria-label="Next"
type="button">\n' +
        '            <svg width="31" height="54" viewBox="0 0 31 54" fill="none"
xmlns="http://www.w3.org/2000/svg">\n' +
        '              <path d="M2 2L28 24.5L2 51.5" stroke="white" stroke-width="4"
stroke-linecap="round"/>\n' +
        '            </svg>\n' +
        '          </button>',
    responsive: [
      {
        breakpoint: 1440,
        settings: {
          slidesToShow: 4,
          slidesToScroll: 1,
          infinite: true,
          dots: false
        }
      },
      {
        breakpoint: 1024,
        settings: {
          slidesToShow: 3,
          slidesToScroll: 1,
          infinite: true,
          dots: false
        }
      },
      {
        breakpoint: 768,
        settings: {
          slidesToShow: 3,
          slidesToScroll: 1,
          infinite: true,
          dots: false
        }
      },
    ]
  });
});
```

2.5 Ознакомление с макетом веб-приложения в Figma.

В качестве сервиса для веб-разработки приложения будем использовать макет в Figma (рисунок 1,2).

Рисунок 1



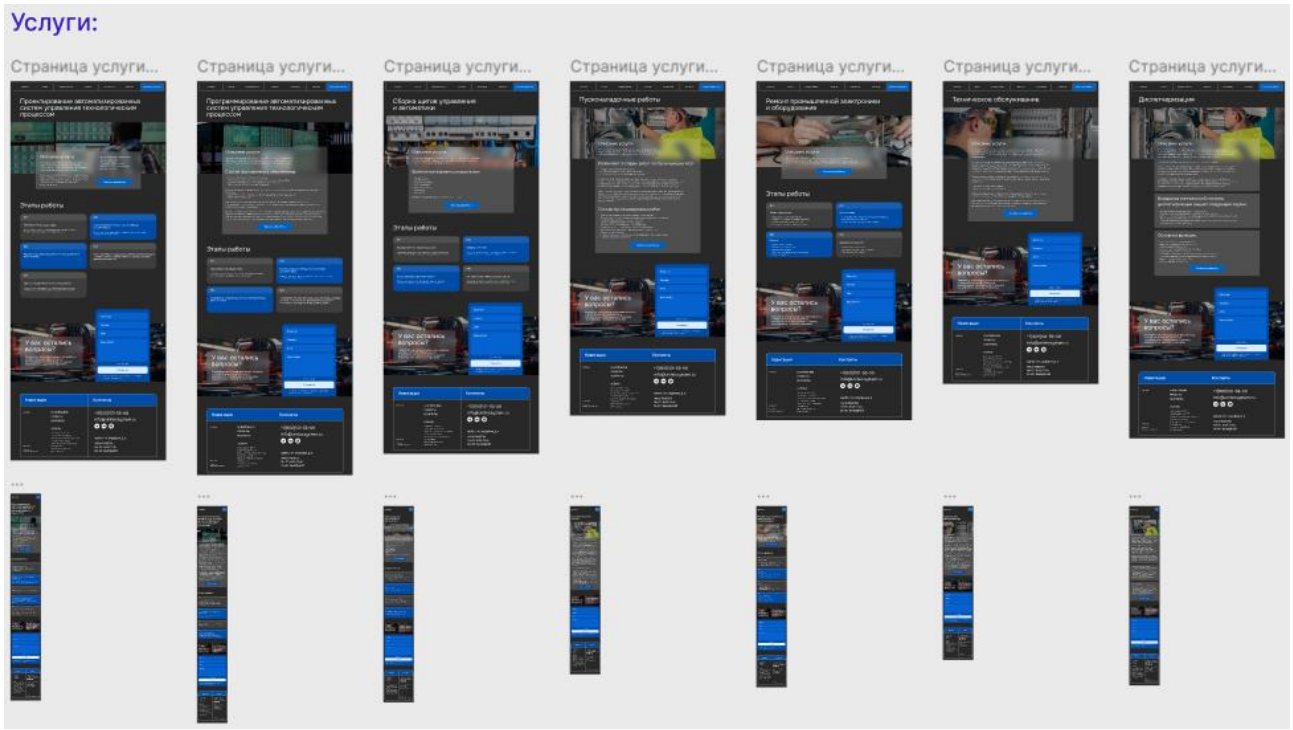
Figma — это онлайн-редактор графики для дизайнеров интерфейсов и веб-разработчиков. Это удобная, бесплатная альтернатива Photoshop. Большое преимущество платформы — возможность работать прямо в браузере. При этом есть и десктопная версия.

Основными задачами, которые предстоит решить разработчику используя макет, являются:

- Определение параметров контента, блоков и элементов;
- Определение параметров текстовых элементов (название, размер, шрифт, цвет, высота строки и т.п.);
- Определение параметров изображений, используемых в макете, их скачивание в требуемом формате;
- Определение размеров и расстояний и границ между элементами и блоками;

- Анализ переиспользуемых блоков и элементов в макете;
- Определение параметров визуализации интерфейсов и анимаций.

Рисунок 2



Глава 3. Разработка веб-приложения

3.1 Верстка веб-приложения, компонентный подход.

Перед началом верстки главной страницы index.html определяем блоки, которые у нас будут переиспользоваться и используя инструменты Gulp выделяем отдельные подключаемые компоненты. Такими блоками будут:

1. head.html – блок для метаданных и подключений. Будет у каждой страницы:

```
<head>
 <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"
/>
  <meta http-equiv="x-ua-compatible" content="ie=edge" />
 <meta name="og:url" content="https://unitecsystem.ru/">
 <meta name="description" content="Автоматизация технологических процессов и производства
и ремонт промышленной электроники и оборудования в ООО &laquo;ЮНИТЭК&raquo;">
  <title>ООО ЮНИТЭК - Автоматизация технологических процессов и производства</title>
  <!-- Icon -->
  <link href="./files/favicon.svg" rel="icon" type="image/x-icon" />
  <!-- JQuery -->
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
  <!-- Slick -->
  <link rel="stylesheet" type="text/css" href="./files/slick/slick.css" />
  <link rel="stylesheet" type="text/css" href="./files/slick/slick-theme.css" />
  <!-- Font Awesome -->
  <link rel="stylesheet" href="./files/font-awesome-4.7.0/css/font-awesome.min.css" />
  <!-- My style -->
  <link href="./css/style.min.css" rel="stylesheet" type="text/css" />
  <!-- My module JS -->
  <script type="module" src="./js/app.min.js"></script>
</head>
```

2. header.html – блок шапки веб-приложения. Этот блок навигации будет присутствовать на каждой странице веб-приложения:

```
<header class="header bg-header" id="header">
  <nav class="navigation fluid-center" id="headerH">
    <div class="navigation__mobile">
      <a class="navbar__item-link drop__close" href="index.html">
       <img class="img__logo" src="./img/logo.svg" alt="logo">
      </a>
      <div class="burger__menu" id="burger-toggle">
        <span class="nav-toggle"></span>
      </div>
    </div>
    <ul class="navigation-bar" id="navigation-bar">
      <li class="navbar__item link-remove">
```

```
        <a class="navbar__item-link drop__close" href="index.html">
         <img class="img__logo" src="./img/logo.svg" alt="logo">
        </a>
      </li>
     <li class="navbar__item link-add">
      <a class="navbar__item-link drop__close" href="index.html">Главная</a>
     </li>
      <li class="navbar__item dropdown-link relative">
        <a class="navbar__item-link" href="#">Услуги</a>
        <div class="services__dropdown" id="dropdown-menu">
          <ul class="dropdown-menu">
            <li><a href="services1.html" class="dropdown-item drop__close">Проектирование
АСУ ТП</a></li>
            <li><a href="services2.html" class="dropdown-item
drop__close">Программирование АСУ ТП</a></li>
            <li><a href="services3.html" class="dropdown-item drop__close">Сборка щитов
управления и автоматики</a></li>
            <li><a href="services4.html" class="dropdown-item
drop__close">Пусконаладочные работы</a></li>
            <li><a href="services5.html" class="dropdown-item drop__close">Ремонт
промышленной электроники и оборудования</a></li>
            <li><a href="services6.html" class="dropdown-item drop__close">Техническое
обслуживание</a></li>
            <li><a href="services7.html" class="dropdown-item
drop__close">Диспетчеризация</a></li>
          </ul>
        </div>
      </li>
      <li class="navbar__item link-remove">
        <a class="navbar__item-link drop__close" href="index.html#working">Порядок
Работы</a>
      </li>
      <li class="navbar__item">
        <a class="navbar__item-link drop__close" href="index.html#example">Проекты</a>
      </li>
      <li class="navbar__item">
        <a class="navbar__item-link drop__close" href="index.html#about">О Компании</a>
      </li>
      <li class="navbar__item">
        <a class="navbar__item-link drop__close" href="#footer">Контакты</a>
      </li>
      <li class="navbar__item link-add">
        <a class="navbar__item-link drop__close" href="tel:+74956166706">+7(800)101-56-
08</a>
      </li>
      <li class="navbar__item link-add">
        <a class="navbar__item-link low-case drop__close"
href="mailto:info@unitecsystem.ru?subject=Mail_from_site_Unitecsystem">info@unitecsystem.
ru</a>
      </li>
      <li class="navbar__box-icons link-add">
        <a href="tg://resolve?domain=GardenEDem9"><i class="drop__close fa fa-telegram"
aria-hidden="true"></i></a>
<!--        <a href="#"><i class="drop__close fa fa-vk fa-2x" aria-
hidden="true"></i></a>-->
        <a href="whatsapp://send?phone=+79878436442"><i class="drop__close fa fa-
whatsapp" aria-hidden="true"></i></a>
      </li>
      <li class="navbar__btn link-add">
        <a class="btn navbar-button clip-rectangle feedback-btn drop__close" href="#"
```

```
>Заказать разработку</a>
      </li>
      <li class="navbar__item link-remove">
        <a class="navbar__item-link feedback-btn" href="#">Заказать Разработку</a>
      </li>
    </ul>
  </nav>
</header>
```

3. modal.html – блоки модальных окон, которые изначально будут скрыты из общей разметки средствами css и их управление будет организовано при помощи JavaScript:

```
<div class="modal-bg modal__feedback-bg">
  <div class="modal__feedback" id="modal-feedback">
    <form class="feedback__content question__form form relative"
action="../src/files/sendmail.php" method="post" id="feedback-form">
      <p class="modal__feedback-box">
        Напишите нам для получения обратной связи, а также задавайте интересующие вас
вопросы
      </p>
      <div class="feedback__button button-close">
        <svg width="16" height="16" viewBox="0 0 16 16" fill="none"
xmlns="http://www.w3.org/2000/svg">
          <path d="M1 1L15 15M1 15L15 1" stroke="white" stroke-width="2"/>
        </svg>
      </div>
      <input type="text" id="formExample10" name="firstname" class="form-control __req"
placeholder="Ваше имя *" />
      <label class="form-label" for="formExample10"></label>
      <input type="tel" id="formExample11" name="telephone" class="form-control
_telephone __req"
             placeholder="Телефон *" />
      <label class="form-label" for="formExample11"></label>
      <input id="formExample12" name="email" class="form-control _email __req"
placeholder="E-mail *" />
      <label class="form-label" for="formExample12"></label>
      <textarea id="formExample13" name="message"  class="form-control _text __req"
rows="5" maxlength="600" placeholder="Ваш вопрос *"></textarea>
      <label class="form-label" for="formExample13"></label>
      <div class="question__form-check">
        <input class="form-checkbox __req" type="radio" id="formExample14" />
        <label class="form-check" for="formExample14">Отправляя свои данные, вы
соглашаетесь с политикой обработки персональных данных.
        </label>
      </div>
      <div class="question__form-button">
        <button type="reset" class="btn form-button clip-rectangle">удалить</button>
        <button type="submit" class="btn form-button clip-rectangle">отправить</button>
      </div>
    </form>
  </div>
</div>
```

4. buttonToTop.html – кнопка для плавной прокрутки вверх

веб-приложения:

```
<div class="button-to-top" id="button-to-top">
  <button class="to-top-btn clip-buttons slick-arrow" aria-label="Up" type="button"
style="">
    <svg width="31" height="54" viewBox="0 0 31 54" fill="none"
xmlns="http://www.w3.org/2000/svg">
      <path d="M29 51.5L3 29L29 2" stroke="white" stroke-width="4" stroke-
linecap="round"></path>
    </svg>
  </button>
</div>
```

5. footer.html – блок подвала веб-приложения, одинаковый для каждой страницы.

```
<footer class="footer content-center" id="footer">
  <div class="footer__box">
    <div class="footer__top">
      <h3 class="footer__top-left working-bg-blue2 clip-left">Навигация</h3>
    </div>
    <div class="footer__top">
      <h3 class="footer__top-right working-bg-blue2 clip-right">Контакты</h3>
    </div>
    <div class="footer__down working-bg-gray2">
        <div class="footer__down-left">
          <div class="item-left-top">
            <a href="index.html#header" class="footer__logo">
             <svg xmlns="http://www.w3.org/2000/svg" version="1.0" width="363.000000pt"
height="58.000000pt" viewBox="0 0 363.000000 58.000000" preserveAspectRatio="xMidYMid
meet">
              <g transform="translate(0.000000,58.000000) scale(0.100000,-0.100000)"
fill="#ffffff" stroke="none"></g>
             </svg>
            </a>
          </div>
          <div class="item-left-down">
            <a href="#" class="button-requisites">Реквизиты</a>
            <a href="#">Политика конфиденциальности</a>
          </div>
        </div>
        <div class="footer__down-right">
          <div class="item-right-top">
            <a href="index.html#about">О КОМПАНИИ</a>
            <a href="index.html#example">ПРОЕКТЫ</a>
            <a href="#footer">КОНТАКТЫ</a>
          </div>
          <nav class="item-right-down">
            <p href="#">УСЛУГИ</p>
            <ul class="footer__ul">
              <li class="footer__link"><a href="services1.html">Проектирование АСУ
ТП</a></li>
              <li class="footer__link"><a href="services2.html">Программирование АСУ
ТП</a></li>
              <li class="footer__link"><a href="services3.html">Сборка щитов управления и
автоматики</a></li>
              <li class="footer__link"><a href="services4.html">Пусконаладочные
```

```
работы</a></li>
              <li class="footer__link"><a href="services5.html">Ремонт промышленной
электроники и оборудования</a></li>
              <li class="footer__link"><a href="services6.html">Техническое
обслуживание</a></li>
              <li class="footer__link"><a href="services7.html">Диспетчеризация</a></li>
            </ul>
          </nav>
        </div>
    </div>
    <div class="footer__down-2 working-bg-gray2">
      <a href="tel:+74956166706">+7(800)101-56-08</a>
      <a
href="mailto:info@unitecsystem.ru?subject=Mail_from_site_Unitecsystem">info@unitecsystem.
ru</a>
      <div class="footer__box-icons">
        <a href="tg://resolve?domain=GardenEDem9">
         <i class="drop__close fa fa-telegram" aria-hidden="true"></i>
        </a>
        <a href="whatsapp://send?phone=+79878436442">
         <i class="drop__close fa fa-whatsapp" aria-hidden="true"></i>
        </a>
      </div>
      <div class="footer__box-address">
        <p>Адрес: г. Бузулук,</p>
        <p>ул. Садовая, д. 5</p>
        <p>Часы работы:</p>
        <p>Пн-Пт: 8:00-17:00</p>
        <p>Сб-Вс: Выходной</p>
      </div>
      <div class="footer__box-politics">
        <a class="button-requisites" href="#">Реквизиты</a>
        <a href="#">Политика конфиденциальности</a>
      </div>
    </div>
  </div>
</footer>
```

Верстка главной страницы в этом случае будет выглядеть так:

```
<!DOCTYPE html>
<html lang="ru">
@@include('html/head.html',{})
<!-- Start your project here-->
<body class="relative">
  <!-- header-->
  @@include('html/header.html',{})
  <!-- header -->
  <!-- modal-requisites-->
  @@include('html/modalRequisites.html',{})
  <!-- modal-requisites -->
  <!--  modal-thanks-->
  @@include('html/modalThanks.html',{})
  <!--  modal-thanks-->
  <!--  modal-error-->
  @@include('html/modalError.html',{})
```

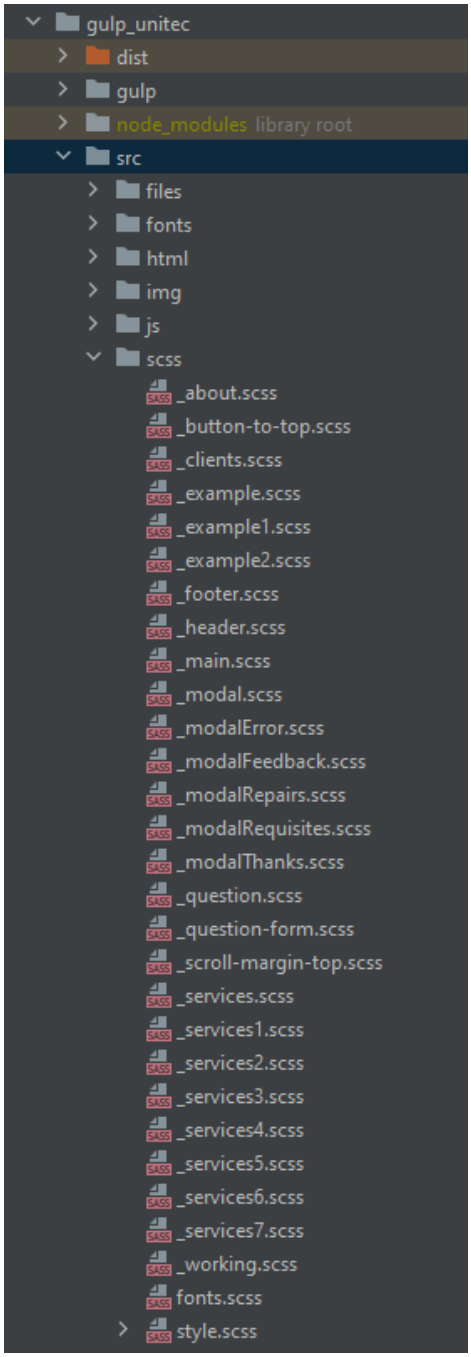
```
  <!--  modal-error-->
  <!--  modal-feedback-->
  @@include('html/modalFeedback.html',{})
  <!--  modal-feedback-->
<main class="main" id="main">
  <div class="title-box content-center">
    <h1 class="main__title section-title">Автоматизация технологических процессов и
производства</h1>
  </div>
  <div class="main__img">
      <div class="content-center main__img-position">
        <div class="main__img-left">
          <div class="main__item">
            <div class="main__item-wrapper">
              <img src="./img/main/main-svg3.svg" alt="main-svg" class="main__item-img">
              <p class="main__item-text">Гарантия качества</p>
            </div>
          </div>
          <div class="main__item">
            <div class="main__item-wrapper">
              <img src="./img/main/main-svg4.svg" alt="main-svg" class="main__item-img">
              <p class="main__item-text">Клиентский сервис</p>
            </div>
          </div>
        </div>
        <div class="main__img-right">
          <div class="main__item">
            <div class="main__item-wrapper">
              <img src="./img/main/main-svg1.svg" alt="main-svg" class="main__item-img">
              <p class="main__item-text">Инновационные решения</p>
            </div>
          </div>
          <div class="main__item">
            <div class="main__item-wrapper">
              <img src="./img/main/main-svg2.svg" alt="main-svg" class="main__item-img">
              <p class="main__item-text">Увеличение производительности</p>
            </div>
          </div>
        </div>
      </div>
    </div>
  <div class="order-box content-center">
    <img src="./img/main/order-img1.webp" alt="order-img" class="order-box__img">
    <div class="order-box__block clip-rectangle">
      <p class="order-box__block-title">Мы выполняем работы по проектированию и внедрению
автоматизированных систем управления технологическими процессами</p>
      <a href="#" class="btn order-button clip-rectangle feedback-btn">Заказать
разработку</a>
    </div>
    <img src="./img/main/order-img2.webp" alt="order-img" class="order-box__img">
  </div>
  <div class="order-box-mobile content-center">
    <div class="order-box__block">
      <p class="order-box__block-title">Мы выполняем работы по проектированию и внедрению
автоматизированных систем управления технологическими процессами</p>
      <a href="#" class="btn order-button clip-rectangle feedback-btn">Заказать
разработку</a>
    </div>
    <img src="./img/main/order-img2.webp" alt="order-img" class="order-box__img">
```

```
  </div>
</main>
  <!-- button-to-up -->
  @@include('html/buttonToTop.html',{})
  <!-- button-to-up -->
  <!-- footer -->
  @@include('html/footer.html',{})
  <!-- footer -->
</body>
</html>
```

Верстка остальных страниц по макету выполняется аналогичным образом с использованием методологии БЭМ и принципа компонентного подхода.

3.2 Стилизация веб-приложения, компонентный подход.

Для упрощения стилизации веб-приложения применим аналогичный верстке компонентный подход. Опорной точкой входа будет файл препроцессора style.scss. Сбрасываем стили, определяем в качестве часто используемых стилей константы и подключаем дополнительные компоненты для каждого блока. Медиа запросы будут также располагаться в своих файлах, привязанных к конкретному блоку, разделу.



Пример точки входа стилей:

```
*,
*::after,
*::before {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}

:root {
  --index: calc(1vw + 1vh);
  --color-nav-link: rgba(255, 255, 255, 1);
  --color-nav-link_hover: rgba(0, 99, 195, 1);
  --bg-content: rgba(40, 40, 40, 1);
  --bg-white: rgba(255, 255, 255, 1);
  --bg-button: rgba(0, 77, 168, 1);
  --bg-block: rgba(72, 72, 72, 1);
  --bg-block1: rgba(81, 81, 81, 1);
  --bg-block2: rgba(62, 62, 62, 1);
  --bg-block3: rgba(25, 95, 177, 1);
  --bg-block4: rgba(0, 77, 168, 1);
  --bg-blur:rgba(255, 255, 255, .2);
  --border-radius: 8px;
  --transition: all .3s ease;
  --transition-btn: all 1s ease;
}

@font-face {
  font-family: Onest;
  src: url('../fonts/Onest-VariableFont_wght.woff2');
}

html {
  scroll-behavior: smooth;
}

a {
  text-decoration: none;
}

body {
  font-family: Onest;
  background-color: var(--bg-content);
  color: var(--color-nav-link);
}

.bg-header {
  background-color: var(--bg-content);
}

.bg-block1 {
  background-color: var(--bg-block1);
}
.bg-block2 {
  background-color: var(--bg-block2);
}
.bg-block3 {
  background-color: var(--bg-block3);
```

```
}
.bg-block4 {
  background-color: var(--bg-block4);
}

.bg-filter {
  filter: brightness(50%);
}

.b-radius {
  border-radius: var(--border-radius);
}

.box-margin {
  margin-top: 60px;
  margin-bottom: 140px;
}

.box-padding {
  padding-top: 60px;
  padding-bottom: 70px;
}

.box-padding-top {
  padding-top: 60px;
}

.btn {
  display: flex;
  align-items: center;
  border: none;
}

.button-blur {
  background: var(--bg-blur);
  backdrop-filter: blur(.5rem);
}

.button-all {
  border: 1px solid rgba(255, 255, 255, .4);
  padding: 25px 35px;
  font-size: 30px;
  font-weight: 500;
  line-height: 32px;
  text-align: left;
  color: var(--color-nav-link);
  border-radius: var(--border-radius);
}

.clip-square {
  clip-path: polygon(5% 0, 95% 0, 100% 5%, 100% 95%, 95% 100%, 5% 100%, 0 95%, 0 5%);
}

.clip-rectangle {
  clip-path: polygon(3% 0,97% 0,100% 10%,100% 90%,97% 100%,3% 100%,0 90%,0 10%);
}

.clip-buttons {
  clip-path: polygon(20% 0,80% 0,100% 20%,100% 80%,80% 100%,20% 100%,0 80%,0 20%);
}
```

```
.clip-item-1 {
  clip-path: polygon(3% 0, 97% 0, 100% 30%, 100% 0%, 100% 100%, 0% 100%, 0 100%, 0 30%);
}

.clip-item-2 {
  clip-path: polygon(0% 0%, 100% 0, 100% 100%, 100% 90%, 97% 100%, 3% 100%, 0 90%, 0 0%);
}

.clip-left {
  clip-path: polygon(5% 0, 100% 0, 100% 100%, 100% 0, 100% 100%, 0 100%, 0 100%, 0 30%);
}

.clip-right {
  clip-path: polygon(0% 0, 95% 0, 100% 30%, 100% 0, 100% 100%, 0 100%, 0 100%, 0 100%);
}

.content-center {
  padding-left: calc(50% - 840px);
  padding-right: calc(50% - 840px);
  margin-left: 1%;
  margin-right: 1%;
}

.d-none {
  display: none;
}

.fluid-center {
  padding-left: calc(50% - 960px);
  padding-right: calc(50% - 960px);
}

.p-top {
  padding-top: 80px;
}

.relative {
  position: relative;
}

.section-title {
  font-size: 80px;
  font-weight: 400;
  line-height: 80px;
  text-align: left;
  color: var(--color-nav-link);
}

.working-bg-blue1 {
  background-color: rgba(25, 95, 177, 1);
}

.working-bg-blue2 {
  background-color: rgba(0, 77, 168, 1);
}

.working-bg-gray1 {
  background-color: rgba(81, 81, 81, 1);
}
```

```
.working-bg-gray2 {
  background-color: rgba(62, 62, 62, 1);
}

@media (max-width: 1440px) {

  .section-title {
    font-size: 60px;
    line-height: 60px;
  }
  .button-all {
    padding: 15px 25px;
    font-size: 20px;
    line-height: 22px;
  }
}

@media (max-width: 1200px) {

  .section-title {
    font-size: 50px;
    line-height: 50px;
  }
}

@media (max-width: 1024px) {

  .section-title {
    max-width: 800px;
    font-size: 40px;
    line-height: 40px;
  }
}

@media (max-width: 768px) {

  .box-margin {
    margin-top: 20px;
    margin-bottom: 60px;
  }
}

@media (max-width: 525px) {

  .section-title {
    max-width: 300px;
    font-size: 29px;
    line-height: 27px;
  }
  .content-center {
    margin-left: 2%;
    margin-right: 2%;
  }
  .box-margin {
    margin-top: 20px;
    margin-bottom: 40px;
  }
}
@import "header";
```

```
@import "main";
@import "services";
@import "about";
@import "clients";
@import "example";
@import "working";
@import "question";
@import "question-form";
@import "footer";
@import "modal";
@import "modalRequisites";
@import "modalRepairs";
@import "modalThanks";
@import "modalError";
@import "modalFeedback";
@import "scroll-margin-top";
@import "example1";
@import "example2";
@import "button-to-top";
@import "services1";
@import "services2";
@import "services3";
@import "services4";
@import "services5";
@import "services6";
@import "services7";
```

3.3 Оживление веб-приложения на JavaScript.

Модульный подход JavaScript и здесь нас не обошел стороной. Сразу же указываем в компоненте head.html, что мы собираемся использовать модульный подход JavaScript:

```
<!-- My module JS -->
<script type="module" src="./js/app.min.js"></script>
```

Создаем точку входа в скрипты, при этом файлом точки входа будет app.js, а в подключениях указываем app.min.js, так как после сборки проекта webpack, который мы настроили в Gulp, соединит все наши модули в один файл.

```
import * as flsFunctions from "./modules/functions.js";
import "../files/slick/slick.js";
import "./modules/burger-toggle.js";
import "./modules/button-to-top.js";
import "./modules/clients-slick.js";
import "./modules/form-post.js";
import "./modules/modal-feedback.js";
import "./modules/modal-requisites.js";
import "./modules/modal-repairs.js";
import "./modules/touch-device.js";
import "./modules/clients-box-render.js";
import "./modules/version.js";
```

В качестве примера разберем несколько наших модулей, которые будут оживлять наше веб-приложение:

1. Кнопка плавной прокрутки вверх веб-приложения button-to-top.js:

```
if(document.querySelector('#button-to-top')) {

    const buttonToTop = document.querySelector('#button-to-top');

    window.addEventListener('scroll', () => {
        if (window.pageYOffset > 500) {
            buttonToTop.classList.add('active');
        } else {
            buttonToTop.classList.remove('active');
        }
    })
    buttonToTop.addEventListener('click', () => {
        window.scrollTo(0, 0);
        // прописать в css html {scroll-behavior: smooth;}
    })
}
```

Данный скрипт работает везде, где истинным является условие наличия селектора в коде html кнопки с идентификатором #button-to-top. Определяем эту кнопку в константу и реализуем два обработчика событий. Первый обработчик события «scroll» на объект window: если прокрутка окна приложения будет составлять более 500 нашей кнопке будет добавлен класс «active», который покажет ее на экране, в противном случае кнопка скрывается. Плавное появление и скрытие кнопки определяется стилистикой позиционирования и трансформации при помощи css. Второй обработчик события «click» навешиваем непосредственно на саму кнопку, при клике на которую встроенный метод window.scrollTo(0, 0) двигает контент до установленных координат. Плавный scroll при этом обеспечивается стилистикой html {scrool-behavior: smooth;}.

2. Адаптивный слайдер блока партнеров clients-box-render.js:

```
if (document.querySelector('#clients')) {
  const divSlider = document.querySelector('#clients-box-render');
  const bd = [
    {
      src: './img/clients/client-slider-1.webp'
    },
    {
      src: './img/clients/client-slider-2.webp'
    },
    {
      src: './img/clients/client-slider-3.webp'
    },
    {
      src: './img/clients/client-slider-4.webp'
    },
    {
      src: './img/clients/client-slider-5.webp'
    },
    {
      src: './img/clients/client-slider-6.webp'
    },
    {
      src: './img/clients/client-slider-7.webp'
    },
    {
      src: './img/clients/client-slider-8.webp'
    },
  ];

  function renderSlider(el) {
    return `<div class="clients__item">
              <img src="${el.src}" alt="slider" class="clients__slider-img">
```

```
            </div>
            `
  }

  bd.forEach((el) => {
    if (el.src) {
      divSlider.insertAdjacentHTML('afterbegin', renderSlider(el));
    }
  });
}
```

Данный скрипт будет работать на страницах, где выполняется условие наличия блока партнеров по селектору идентификатора #clients. Определяем константу divSlider места, куда будем добавлять верстку при помощи функции renderSlider(el), в свою очередь эта функция будет брать данные из простенькой базы данных bd, некого массива объектов, значением по ключу src которых будет путь до картинки. Методом forEach проходимся по базе данных и к каждому элементу применяем функцию рендеринга блока в верстку html:

```
<section class="clients content-center bg-header" id="clients">

    <div class="clients__box">
      <h2 class="section-title">Наши партёры</h2>
      <div class="clients__box-buttons">
      <!--rendering from clients-slick.js-->
      </div>
    </div>
    <div class="clients__slider box-margin" id="clients-box-render">
      <!--rendering from clients-box-render.js-->
    </div>

</section>
```

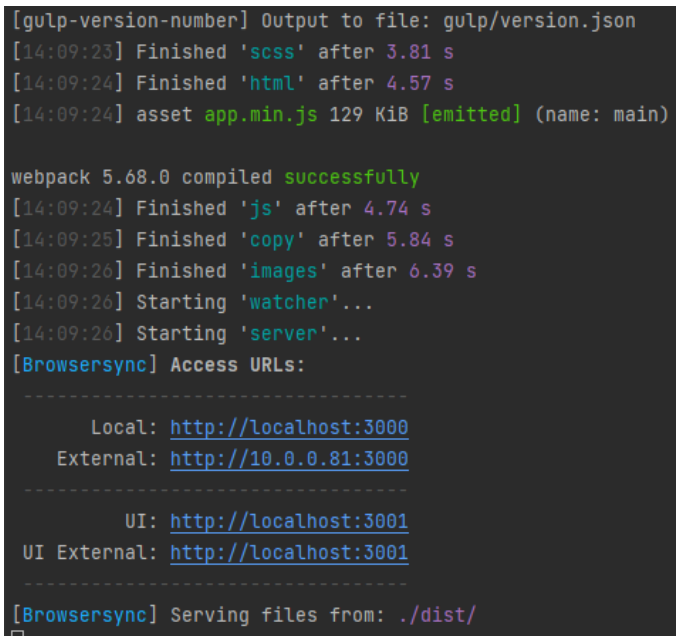
Для оживления слайдера воспользуемся ранее установленной библиотекой Slick slider и опишем поведение слайдера, добавим кнопки прокрутки с собственными стилями и брейкпоинты поведения при различных размерах экрана в файлике clients-slick.js:

```
if (document.querySelector('#clients')) {
  $(function () {
    $('.clients__slider').slick({
      dots: false,
      infinite: true,
      slidesToShow: 4,
      slidesToScroll: 1,
      autoplay: true,
      autoplaySpeed: 1500,
      arrows: true,
      appendArrows: $('.clients__box-buttons'),
```

```
      prevArrow: '<button class="clients-btn clip-buttons" aria-label="Next"
type="button">\n' +
          '            <svg width="31" height="54" viewBox="0 0 31 54" fill="none"\n' +
          '
xmlns="http://www.w3.org/2000/svg">\n' +
          '              <path d="M29 51.5L3 29L29 2" stroke="white" stroke-width="4"
stroke-linecap="round"/>\n' +
          '          </svg>\n' +
          '          </button>',
      nextArrow: '<button class="clients-btn clip-buttons" aria-label="Next"
type="button">\n' +
          '            <svg width="31" height="54" viewBox="0 0 31 54" fill="none"
xmlns="http://www.w3.org/2000/svg">\n' +
          '              <path d="M2 2L28 24.5L2 51.5" stroke="white" stroke-width="4"
stroke-linecap="round"/>\n' +
          '            </svg>\n' +
          '          </button>',
      responsive: [
        {
          breakpoint: 1440,
          settings: {
            slidesToShow: 4,
            slidesToScroll: 1,
            infinite: true,
            dots: false
          }
        },
        {
          breakpoint: 1024,
          settings: {
            slidesToShow: 3,
            slidesToScroll: 1,
            infinite: true,
            dots: false
          }
        },
        {
          breakpoint: 768,
          settings: {
            slidesToShow: 3,
            slidesToScroll: 1,
            infinite: true,
            dots: false
          }
        },
      ]
    });
  });
}
```

3.4 Проверка работоспособности. Сборка проекта.

Запуск сборки в различных режимах осуществляем из консоли при помощи созданных ранее служебных команд. Запускаем режим разработчика командой gulp. Безошибочная компиляция проекта завершается сохранением проекта в папке dist и автоматическим запуском сервера. В этом режиме производится проверка и отладка кода. Задача отслеживающая изменение кода в проекте будет следить за файлами и перезагружать страницу для отображения изменений после сохранения кода.



Режим build и deployZIP позволяют провести полную упаковку проекта с минификацией css и js кода, преобразованием изображений в современный формат webp и упаковку, в последнем случае, готового проекта в zip-архив.

Заключение

Практическое применение IDE PHP Storm в совокупности с настроенной под себя средой Gulp позволил нам автоматизировать такие рутинные повседневные задачи веб-разработки, как:

1. Создание веб-сервера и автоматическая перезагрузка страницы в браузере при сохранении кода, отслеживание изменений в файлах проекта;
2. Использование препроцессоров JavaScript, CSS, HTML;
3. Минификация CSS, JS кода, а также оптимизация и конкатенация отдельных файлов проекта в один;
4. Автоматическое создание вендорных префиксов для CSS;
5. Управление файлами и папками в рамках проекта – создание, удаление, переименование;
6. Реализация компонентного подхода;
7. Запуск и контроль выполнения внешних команд операционной системы;
8. Работа с изображениями и видео любых форматов, сжатие и преобразование в новейшие форматы webp и webv;
9. Обработка ошибок;
10. Деплой проекта.

Кроме того, можно с уверенностью сказать, что Gulp и множество утилит, написанных для него, подходят для решения практически любой задачи при разработке проекта любой сложности - от небольшого сайта до крупного проекта. Настройка менеджера Gulp под конкретные задачи позволяет значительно сократить время веб-разработки.

На сегодняшний день сборщик задач Gulp все еще актуален. Версия последнего релиза Gulp 5.0.0 на момент написания дипломной работы была

опубликована пять месяцев назад. Средний количественный показатель потребности сборщика по результатам обращений пользователей на скачивание составляет более миллиона скачиваний.

Разработанное мною веб-приложение с использованием менеджера задач Gulp было успешно реализовано и размещено на хостинге заказчика: **https://unitecsystem.ru/** за достаточно короткое по продолжительности время - две недели от и до. Цели, поставленные перед собой, были достигнуты в полном объеме. А использование сторонних библиотек, менеджера задач и среды разработки способствовали достижению цели в рекордно короткие сроки, что безусловно доказывают не просто желание, а необходимость использования подобных помощников в решении повседневных задач по работе веб-разработчика.

Если вы работаете над веб-проектом, который не основан на современных JS-фреймворках подобных React, Vue, Angular и других, а также ваш проект не является SPA – Gulp станет вашим другом и помощником!

Список используемой литературы

1. Алексей Куреев. «Сравнение популярных систем сборки для frontend-разработчиков», 2024г. Ресурс: https://habr.com/ru/articles/215131
2. Борис Демченко. "Gulp и его использование", 2022г. Ресурс: https://doka.guide/tools/gulp
3. Библиотека PHPMailer. Ресурс: https://github.com/PHPMailer/PHPMailer
4. Библиотека иконочного шрифта и css-инструментария. Ресурс: https://fontawesome.ru/
5. Библиотека адаптивного слайдера Slick Slider. Ресурс: https://kenwheeler.github.io/slick/
6. ВебДизайн Мастер «Gulp – Актуальное и исчерпывающее руководство» Ресурс: https://webdesign-master.ru/blog/tools/gulp-4-lesson.html
7. Фрилансер по жизни. «GULP 2022 установка настройка плагины», 2022г. Ресурс: https://www.youtube.com/watch?v=jU88mLuLWlk