

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**   
**«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»**

Факультет Физико-технический

Кафедра Компьютерных технологий (КТ)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | КТ |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | Т. В. Ермоленко |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе бакалавра 2 курса

на тему:

РАЗРАБОТКА МИКРО-СЕРВИСА ДЛЯ ПРЕОБРАЗОВАНИЯ HTML РАЗМЕТКИ В ИЗОБРАЖЕНИЕ

|  |  |  |  |
| --- | --- | --- | --- |
| Автор работы |  |  | А. В. Горбань |

подпись

|  |  |  |
| --- | --- | --- |
| Направление | 09.03.01 | Информатика и вычислительная техника |

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель Котенко В.Н.

подпись

Консультанты по разделам:

Техническое задание \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент Т.В. Шарий

подпись

Нормоконтроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. лаборант В.Г. Медведева

подпись

Курсовая работа защищена \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата итоговая оценка комиссия

Подписи членов комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Донецк  
2022

ГОУ ВПО «Донецкий национальный университет»

кафедра Компьютерных технологий

Утверждаю

Зав. кафедрой КТ

\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

**ЗАДАНИЕ**

на курсовую работу студента 2 курса Горбань А. В.

*Тема курсовой работы*: Разработка микро-сервиса для преобразования HTML разметки в изображение

*Краткая постановка задачи*: 1. Изучить и проанализировать предметную область по преобразовании HTML разметки в изображение. 2. Ознакомиться с программными продуктами для создания приложений по преобразованию HTML разметки в изображение. 3. Разработать техническое задание на создание приложения. 4. Разработать проект программного обеспечения системы. 5. Разработать средствами языка TypeScript приложение, которое позволяет преобразовать HTML разметку в изображение. 6. Протестировать программное обеспечение. 7. Оформить отчёт.

*Исходные данные*: 1. Документация по преобразованию HTML разметки в изображение. 2. Документация по языку программирования TypeScript.

*Ожидаемые результаты*: серверное приложение — микро-сервис для преобразования HTML разметки в изображение

*Календарный план работы:*

|  |  |  |
| --- | --- | --- |
| Даты консультаций | Этапы выполнения работы | Отметки о выполнении |
| 30.01.2017 | Постановка задачи и обсуждение литературы | выполнено |
| 07.02.2017 | Предварительное утверждение содержания отчёта | выполнено |
| 17.02.2017 | Утверждение проекта, алгоритмов, методов, технологий | выполнено |
| 03.03.2017 | Ход реализации проекта | выполнено |
| 23.03.2017 | Обсуждения организации тестирования программы | выполнено |
| 10.04.2017 | Демонстрация программного продукта руководителю | выполнено |
| 17.04.2017 | Оформление отчёта | выполнено |
| 24.04.2017 | Предоставление отчёта руководителю | выполнено |

Дата выдачи задания 30.01.2017 года

Студент И.И. Иванов

Руководитель В.Н. Котенко

**АННОТАЦИЯ**

1. Отчет о курсовой работе: 100 с., 48 рис., 14 табл., 2 приложения, 12 источников.
2. Объект исследования – браузер, сервер.
3. Предмет исследования – серверное приложение для управления браузером.
4. Цель работы – разработка серверного приложения для преобразования HTML разметки в изображение, которая позволяла бы генерировать приложение по HTML разметке, присланной в XHR запросе.
5. Метод исследования – анализ возможностей языка TypeScript на платформе Node.js для создания и обработки серверного приложения.
6. В работе были использованы возможности платформы Node.js, языка программирования TypeScript а так же express, express-fileupload, fs-extra, puppeteer, ajv, lodash и uuid пакетов.
7. В результате решения задачи было разработано приложение для преобразования HTML разметки в изображение. Приложение позволяет генерировать изображение по присланным HTML, CSS, изображениям и опциям, посредством XHR запроса.
8. Главный модуль приложения работает с браузером, запущенном на сервере. Главной операцией приложения является основной контроллер.
9. Приложение html2img-service может использоваться back-end и front-end разработчиками, web-мастерами и дизайнерами для генерации изображений по HTML разметке.

TypeScript, NODE.JS, WEB-SERVICE, MICRO-SERVICE, GNU/LINUX, XHR, CHROMIUM, HTML, CSS, JSON-SCHEMA, FILE SYSTEM, PUPPETEER, OBJECT-ORIENTED PROGRAMMING, EXPRESS

**СОДЕРЖАНИЕ**

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#__RefHeading___Toc494639103)

[1.1 Состояние вопроса 6](#__RefHeading___Toc494639104)

[1.2 Актуальность и цель работы 6](#__RefHeading___Toc494639105)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 8](#__RefHeading___Toc494639106)

[2.1 Описание области применения и исходных данных приложения 8](#__RefHeading___Toc494639107)

[2.2 Требования к пользовательским интерфейсам 8](#__RefHeading___Toc494639108)

[2.3 Требования к аппаратным, программным и коммуникационным интерфейсам 9](#__RefHeading___Toc494639109)

[2.4 Требования к пользователям продукта 9](#__RefHeading___Toc494639110)

[2.5 Требования к адаптации на месте 10](#__RefHeading___Toc494639111)

[2.6 Функции продукта 10](#__RefHeading___Toc494639112)

[2.7 Ограничения 11](#__RefHeading___Toc494639113)

[4 РАЗРАБОТКА МИКРО-СЕРВИСА ДЛЯ ПРЕОБРАЗОВАНИЯ HTML РАЗМЕТКИ В ИЗОБРАЖЕНИЕ 13](#__RefHeading___Toc494639115)

[4.1 Входные и выходные данные приложения 13](#__RefHeading___Toc494639116)

[4.2 Проектирование структуры приложения 14](#__RefHeading___Toc494639117)

[4.3 Описание алгоритмов работы скриптов системы или (Описание объектов и их взаимодействия для ООП) 16](#__RefHeading___Toc494639118)

[5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 24](#__RefHeading___Toc494639119)

[5.1 Аппаратные и программные средства создания и эксплуатации интернет-приложения 24](#__RefHeading___Toc494639120)

[5.2 Руководство пользователя 25](#__RefHeading___Toc494639121)

[5.3 Описание контрольных примеров 25](#__RefHeading___Toc494639122)

[ЗАКЛЮЧЕНИЕ 27](#__RefHeading___Toc494639123)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 28](#__RefHeading___Toc494639124)

[ПРИЛОЖЕНИЕ А Экранные формы 29](#__RefHeading___Toc494639125)

[ПРИЛОЖЕНИЕ Б Фрагменты листинга 30](#__RefHeading___Toc494639126)

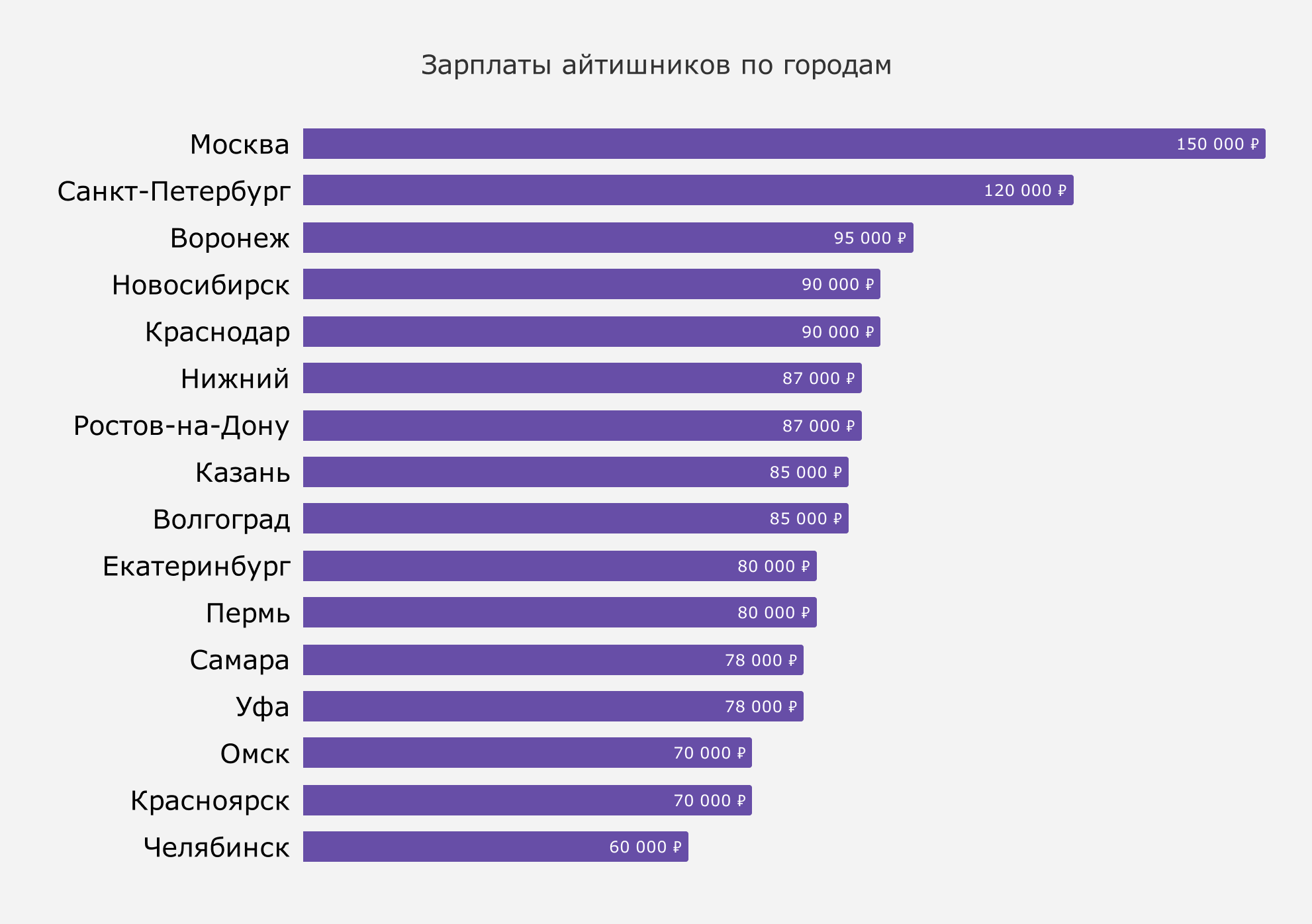
1. ВВЕДЕНИЕ
2. Современные информационные технологии в настоящее время быстрыми темпами внедряются во все сферы социально-культурной деятельности человека. Почему бы не использовать их для облегчения деятельность IT-специалистов?
3. В данной курсовой работе будет реализовано приложение, которое позволит front-end, back-end разработчикам, web-мастерам и дизайнерам позволит обеспечить простую автоматическую генерацию изображений в проектах.
4. 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Состояние вопроса

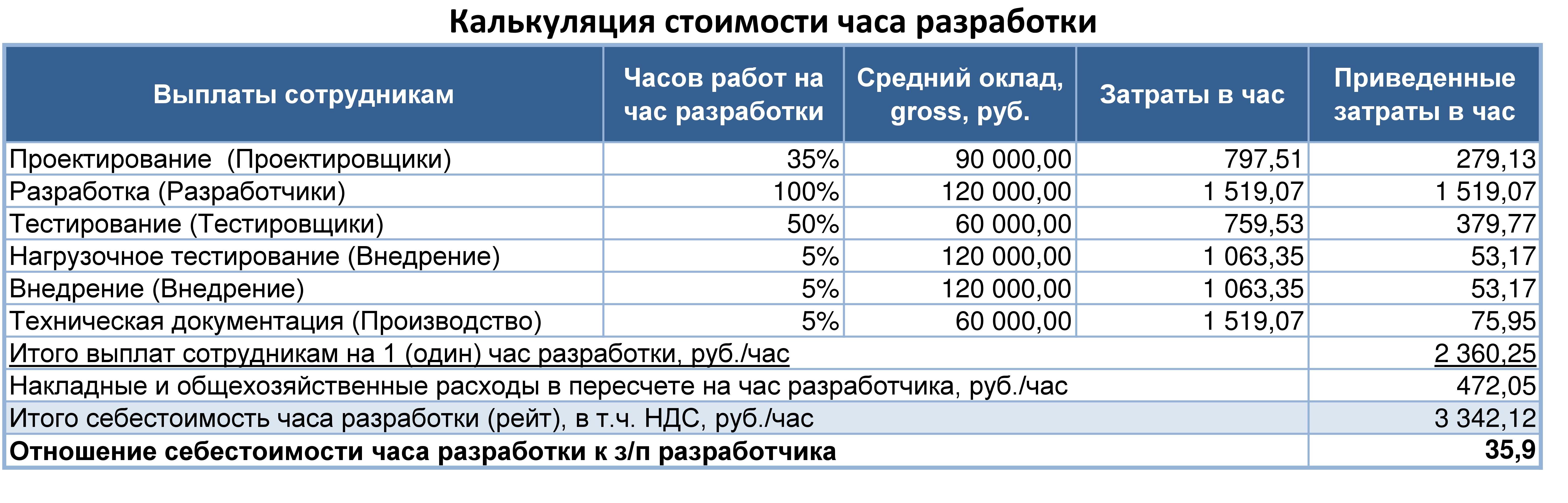
1. В настоящее время IT-индустрия охватывает все большие сферы деятельность человека. Создается все больше различных web-сайтов, приложений и сервисов, которые необходимо поддерживать и развивать. Соответственно, потребность в специалистах данной сферы постоянно растет.

1.2 Актуальность и цель работы

1. На рисунке 1.1 показан уровень средней заработной платы IT-специалистов по городам Российской Федерации на 2020 г.

Рисунок 1.1 — Средняя заработная плата IT-специалистов по городам Российской Федерации на 2020 г.

1. На рисунке 1.2 изображена таблица расчета стоимости 1 часа разработки.

Рисунок 1.2 - Таблица расчета стоимости 1 часа разработки.

1. Как видно из приведенных рисунков потребность рынка в квалифицированных IT-специалистах весьма значительна. Соответственно, и стоимость их работы высока.
2. Для снижения стоимости и сокращения сроков разработки зачастую используются готовые программные решения.
3. В процессе разработки программных продуктов периодически возникает потребность в генерации изображений. Это могут быть чеки, билеты, фирменные водные знаки, и тому подобное.
4. Для решения данной проблемы разработчик может воспользоваться микро-сервисом «html2img-service».
5. Целью данной работы будет разработка микро-сервиса, который позволит быстро и легко генерировать необходимые изображения.
6. 2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

2.1 Описание области применения и исходных данных приложения

1. Данное приложение предназначено для использования в программных продуктах по средствам API.
2. Исходными данными для приложения являются:
3. тип возвращаемого изображения;
4. ширина изображения;
5. высота изображения;
6. качество изображения;
7. флаг, нужно ли убирать задний фон изображения;
8. HTML разметка;
9. CSS стили;
10. изображения.

**2.2 Требования к пользовательским интерфейсам**

1. Так как пользователями разрабатываемого программного обеспечения являются разработчиками, пользовательский интерфейс представлен в виде RESTful API.
2. RESTful API позволяют разрабатывать все виды веб-приложений со всеми возможными операциями CRUD (create, read, update, delete).
3. Руководящие принципы REST предлагают использовать определенные метод HTTP для определенного типа действий с ресурсами на сервере (хотя технически возможно нарушить это руководство, но это крайне не рекомендуется).
4. Методы HTTP используемые в RESTful API:

* GET — для получения информации о ресурсе;
* POST — для добавления информации о ресурсе;
* PUT — для полного изменения или замены информации о ресурсе;
* PATCH — для частичного изменения информации о ресурсе;
* DELETE — для удаления информации о ресурсе.

2.3 Требования к аппаратным, программным и коммуникационным интерфейсам

1. Для установки и работы приложения необходимо иметь вычислительную систему следующей минимальной аппаратной конфигурации:
2. Процессор: 1.0 ГГц;
3. Оперативная память: 512 Мб;
4. Свободное место на жёстком диске: 150Мб;
5. Видеоадаптер: 64Мб;
6. Интернет-соединение.

2.4 Требования к пользователям продукта

1. Системой должны иметь возможность пользоваться следующие категории пользователей: front-end, back-end разработчики, web-мастера и дизайнеры.
2. Front-end и back-end разработчики:

* Понимание принципов функционирования приложения и работа с XHR запросами;
* Понимание принципов автоматизации за счет написания скрипта при помощи языков программирования.

1. Web-мастера и дизайнеры:

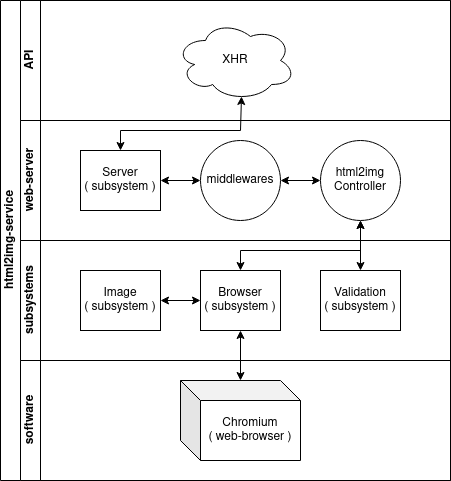
* Понимание принципов функционирования приложения и работа с XHR запросами.

2.5 Требования к адаптации на месте

1. Необходимо предварительно установленное программное обеспечение платформа Node.js и браузер chromium. Так-же в качестве дополнительного программного обеспечения, могут быть использованы демон-менеджер процессов PM2 и web-сервер Nginx или Apache.
2. Необходима целевая директория, в которой будет находиться корень программы.

2.6 Функции продукта

1. Приложение должно обладать функционалом по работе с браузером, обработке изображений, управлении web-сервером и валидации входящих данных.
2. Приложение обладает одной, единой главной функцией, которая доступна всем классам пользователей.
3. На рисунке 2.1 показана схема функциональной структуры приложения html2img-service.

Рисунок 2.1 — Схема функциональной структуры приложения

2.7 Ограничения

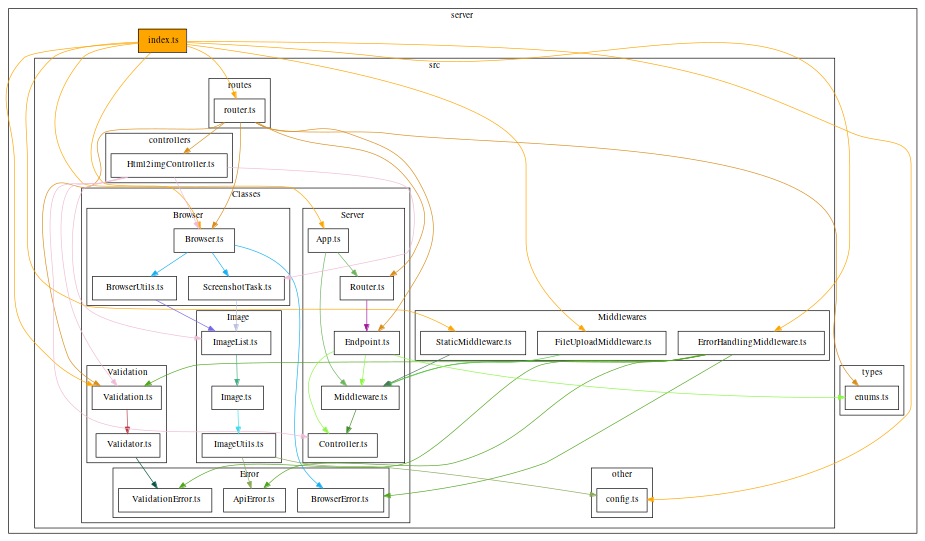
1. приложение будет поддерживать только изображения форматов: png, jpeg и webp;
2. приложение должно использоваться в рамках локальной или глобальной сети;
3. приложение будет некорректно работать с браузерами основанными не на chromium.
4. 3 ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ
5. Описание среды разработки:
6. Visual Studio Code — это редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений.
7. Postman — это платформа API, позволяющая разработчикам проектировать, создавать, тестировать и повторять свои API.
8. Node.js — это программная платформа, основанная на движке V8, превращающая JavaScript из узкоспециализированного языка в язык общего назначения.
9. TypeScript — это язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.
10. Описание производственной среды:
11. Node.js — это программная платформа, основанная на движке V8, превращающая JavaScript из узкоспециализированного языка в язык общего назначения.
12. PM2 (опционально) — это менеджер процессов для JavaScript-среды Node.js.
13. Nginx — это веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах.
14. 4 РАЗРАБОТКА МИКРО-СЕРВИСА ДЛЯ ПРЕОБРАЗОВАНИЯ HTML РАЗМЕТКИ В ИЗОБРАЖЕНИЕ

4.1 Входные и выходные данные приложения

1. Входные данные приложения приходят в URN и GET-параметрах из URI и теле XHR запроса.
2. Входными данными приложения являются 8 переменных:
3. Тип возвращаемого изображения приходит в URN;
4. Ширина возвращаемого изображения (опционально) приходит в GET-параметрах;
5. Высота возвращаемого изображения (опционально) приходит в GET-параметрах;
6. Качество возвращаемого изображения (опционально) приходит в GET-параметрах;
7. Флаг, отвечающий за наличие заднего фона возвращаемого изображения (опционально) приходит в GET-параметрах;
8. HTML разметка возвращаемого изображения приходит в теле запроса;
9. CSS стили возвращаемого изображения (опционально) приходят в теле запроса;
10. Изображения для вставки в HTML разметку и CSS стили возвращаемого изображения (опционально) приходят в теле запроса;
11. Выходными данными приложения является сгенерированное изображение или ошибка.

4.2 Проектирование структуры приложения

1. Структурно приложение состоит из следующих подсистем и  
   TypeScript-скриптов:
2. подсистема управления сервером — классы: App, Controller, Endpoint, Middleware и Router;
3. подсистема управления браузером — классы: Browser, BrowserUtils и ScreenshotTask;
4. подсистема манипуляции изображениями — классы: Image, ImageList и ImageUtils;
5. подсистема валидации — классы: Validation и Validator;
6. классы ошибок: ApiError, BrowserError и ValidationError;
7. функциональные классы: Html2ImgController, FileUploadMiddleware, ErrorHandlingMiddleware, StaticMiddleware и MainRouter;
8. настройки среды выполнения — файл config.ts;
9. декларация типов — файлы: enum.ts, index.d.ts и utils.d.ts.
10. Схема взаимодействия классов и скриптов приложения приведена   
    на рисунке 4.1.

Рисунок 4.1 – Структура микро-сервиса

4.3 Описание алгоритмов работы скриптов системы или (Описание объектов и их взаимодействия для ООП)

1. Класс Browser — модуль для взаимодействия с браузером.
2. Поля классы Browser:

* protected \_browser — экземпляр класса Browser от puppeteer.

1. Методы класса Browser:

* private \_launchBrowser — запускает браузер;
* public closeBrowser — закрывает браузер;
* private \_createPage — создает страницу в браузере;
* private сalcNativeHeight — подсчитывает нативную высоту контента на странице;
* public screenshot — делает скриншот контента страницы.

1. Класс BrowserUtils — модуль с утилитами для класса Browser.
2. Поля класса BrowserUtils:

* static htmlRootId — html id для корневого тега.

1. Методы класса BrowserUtils:

* static generateViewport — генерирует настройки viewport для страницы;
* static insertImgIntoHTML — вставляет изображения в html разметку;
* static generateContent — генерирует html разметку.

1. Класс ScreenshotTask — данные и параметры для скриншота.
2. Поля класса ScreenshotTask:

* public data — объект, в котором хранятся html и css;
* public imgs — хранится экземпляр класса ImageList:
* public opts — объект, в котором хранятся параметры для скриншота, а именно: width, height, returnImgType, quality и omitBackground.

1. Класс Image — модуль, для работы с отдельным изображением.
2. Поля класса Image:

* private \_name — наименование изображения;
* private \_path — путь к изображению;
* private \_file — файл изображения.

1. Геттеры и сеттеры класса Image:

* get uri — отдает сгенерированный uri изображения.

1. Методы класса Image:

* public move — перенести изображение из временной директории в постоянную;
* public removeTmp — удалить изображение из временной директории;
* public remove — удалить изображение из постоянной директории.

1. Класс ImageList — список изображений.
2. Поля класса ImageList:

* public imgs — массив экземпляров класса Image и их названий.

1. Методы класса ImageList:

* private \_prepareImgs — подготавливает изображения для поля imgs;
* public removeImgs — удаляет изображения из постоянной директории.

1. Класс ImageUtils - модуль с утилитами для класса Image.
2. Методы класса ImageUtils:

* private static imgMimetypeToExtension — преобразовывает mime тип изображения в расширение изображения;
* public static generateName — генерирует наименование изображения;
* public static generatePath — генерирует путь к изображению;
* public static generateUri — генерирует uri изображения.

1. Класс ApiError — класс ошибки, расширяющий базовый класс Error, предназначенный для выбрасывания ошибок из контроллеров и миддлвейров.
2. Поля класса ApiError:

* public status — поле для хранения числового HTTP статуса ошибки
* public message — поля для хранения сообщения ошибки (унаследовано от класса Error);
* public name — поля для хранения наименования ошибки (унаследовано от класса Error).

1. Методы класса ApiError:

* public static badRequest — статический метод для создания экземпляра класса ApiError с преднастройками;
* public static unauthorized — статический метод для создания экземпляра класса ApiError с преднастройками;
* public static forbidden — статический метод для создания экземпляра класса ApiError с преднастройками;
* public static notFound— статический метод для создания экземпляра класса ApiError с преднастройками;
* public static internal— статический метод для создания экземпляра класса ApiError с преднастройками;

1. Класс BrowserError — класс ошибки, расширяющий базовый класс Error, предназначенный для выбрасывания ошибок из подсистемы работы с браузером.
2. Поля класса BrowserError:

* public message — поля для хранения сообщения ошибки (унаследовано от класса Error);
* public name — поля для хранения наименования ошибки (унаследовано от класса Error).

1. Класс ValidationError — класс ошибки, расширяющий базовый класс Error, предназначенный для выбрасывания ошибок из подсистемы валидации.
2. Поля класса ValidationError:

* public info — поле для хранения исходный ошибок ajv валидации;
* public message — поля для хранения сообщения ошибки (унаследовано от класса Error);
* public name — поля для хранения наименования ошибки (унаследовано от класса Error).

1. Класс Validation — класс, предназначенный для генерации валидаторов по валидационным схемам.
2. Поля класса Validation:

* private \_ajv — экземпляр класса ajv

1. Методы класса Validation:

* private \_activateAjvPlugins — приватный метод для активации ajv плагинов;
* public generateValidator — метод для генерации валидотора по переданной JSON Schema валидации;
* public getErrorMessage — метод для получения сообщения ошибки из ajv ошибки.

1. Класс Validator — класс предоставляющий возможность валидации данных по предопределенной JSON Schema валидации.
2. Поля класса Validator:

* private \_validator — поле, хранящее ajv валидатор.

1. Методы класса Validator:

* public validate — метод для валидации переданных данных.

1. Класс App — главный класс приложения отвечающий за запуск и работы web-сервера.
2. Поля класса App:

* private \_app — поле хранящее в себе экземпляр класса Express.Application;
* private \_port — поле, хранящее в себе порт, который необходимо прослушивать web-серверу.

1. Методы класса App:

* private useMiddlewares — метод для регистрации миддлвейров в приложении;
* private useRoutes — метод для регистрации роутеров в приложении;
* public start — метод для запуска web-сервера.

1. Абстрактный класс Controller — абстрактный класс, предназначенный для создания контроллеров
2. Геттеры и сеттеры абстрактного класса Controller:

* get controller — геттер для получения функции-контроллера.

1. Методы абстрактного класса Controller:

* protected abstract \_controller — абстрактный метод функции-контроллера.

1. Абстрактный класс Middleware — абстрактный класс, расширяющий абстрактный класс Controller, предназначенный для создания миддлвейра.
2. Класс Endpoint — класс конечной точки запроса.
3. Поля класса Endpoint:

* private \_method — поле, хранящее метод конечной точки;
* private \_route — поле, хранящее путь конечной точки;
* private \_beforeMiddlewares — поле, хранящее middlewares, которые должны быть выполнены до выполнения контроллера;
* private \_ controller — поле, хранящее контроллер конечной точки;
* private \_afterMiddlewares — поле, хранящее middlewares, которые должны быть выполнены после выполнения контроллера.

1. Методы класса Endpoint:

* public use — метод для использования конечной точки в роутере.

1. Абстрактный класс Router — абстрактный класс, предназначенный для создания роутера.
2. Поля абстрактного класса Router:

* private \_router — поле, хранящее экземпляр класса ExpressCore.Express;
* private \_route — поле, хранящее путь;
* protected abstract \_endpoints — абстрактное поле, хранящее массив экземпляров класса Endpoint.

1. Геттеры и сеттеры абстрактного класса Router:

* get route — геттер для получения пути;
* get router — геттер для получения роутера.

1. Методы абстрактного класса Router:

* protected useEndpoints — метод для регистрации конечных точек в роутере.

1. Класс Html2imgController — класс, расширяющий абстрактный класс Controller, основной контроллер приложения.
2. Поля класса Html2IngController:

* private \_browser — поле, хранящее экземпляр класса Browser;
* private \_validation — поле, хранящее экземпляр класса Validation;
* private \_paramsValidator — поле, хранящее экземпляр класса Validator для валидации параметров запроса;
* private \_queryVaildator — поле, хранящее экземпляр класса Validator для валидации GET-параметров запроса;
* private \_bodyValidator — поле, хранящее экземпляр класса Validator для валидации тела запроса.

1. Геттеры и сеттеры класса Html2ImgController:

* get controller — геттер для получения функции-контроллера (унаследован от абстрактного класса Controller).

1. Методы класса Html2ImgController:

* protected \_controller — метод функция-контроллер (реализация абстрактного метода от абстрактного класса Controller).

1. Класс ErrorHandlingMiddleware — класс, расширяющий абстрактный класс Middleware, middleware — обработчик ошибок.
2. Геттеры и сеттеры класса ErrorHandlingMiddleware:

* get controller — геттер для получения функции-контроллера (унаследован от абстрактного класса Middleware).

1. Методы класса ErrorHandlingMiddleware:

* protected \_controller — метод функция-контроллер (реализация абстрактного метода от абстрактного класса Middleware).

1. Класс FileUploadMiddleware — класс, расширяющий абстрактный класс Middleware, middleware — адаптер для библиотеки express-fileupload для парсинга изображений, приходящих в теле запроса.
2. Геттеры и сеттеры класса FileUploadMiddleware:

* get controller — геттер для получения функции-контроллера (унаследован от абстрактного класса Middleware).

1. Методы класса FileUploadMiddleware:

* protected \_controller — метод функция-контроллер (реализация абстрактного метода от абстрактного класса Middleware).

1. Класс StaticMiddleware — класс, расширяющий абстрактный класс Middleware, middleware — адаптер для middleware Express.static для назначения статических файлов и директорий web-сервера.
2. Геттеры и сеттеры класса StaticMiddleware:

* get controller — геттер для получения функции-контроллера (унаследован от абстрактного класса Middleware).

1. Методы класса StaticMiddleware:

* protected \_controller — метод функция-контроллер (реализация абстрактного метода от абстрактного класса Middleware).

1. Класс MainRouter — класс, расширяющий абстрактный класс Router, основной роутер приложения.
2. Поля класса MainRouter:

* protected \_endpoints — абстрактное поле, хранящее массив экземпляров класса Endpoint (реализация абстрактного поля от абстрактного класса Router).

1. Геттеры и сеттеры класса MainRouter:

* get route — геттер для получения пути (унаследовано от абстрактного класса Router);
* get router — геттер для получения роутера (унаследовано от абстрактного класса Router).

1. Методы класса MainRouter:

* protected useEndpoints — метод для регистрации конечных точек в роутере (унаследовано от абстрактного класса Router).

1. 5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

5.1 Аппаратные и программные средства создания и эксплуатации интернет-приложения

1. Аппаратные требования для работы сервиса в среде разработки:

* Процессор с тактовой частотой от 1.3 ГГц;
* Оперативная память от 2 ГБ;
* Видеокарта;
* Монитор;
* Клавиатура.

1. Программные требования для работы сервиса в среде разработки:

* Операционная система — Linux, Windows или MacOS
* Среда выполнение Node.js;
* IDE;
* Web-браузер на основе chromium.

1. Аппаратные требования для работы сервиса в производственной среде:

* Процессор с тактовой частотой от 1.0 ГГц;
* Оперативная память от 1 ГБ;
* Сетевой адаптер.

1. Программные требования для работы сервиса в среде разработки:

* Операционная система — Linux, Windows или MacOS
* Среда выполнение Node.js;
* Менеджер процессов для JavaScript-среды Node.js PM2;
* Web-браузер на основе chromium.

5.2 Руководство пользователя

1. Для работы сервиса необходимо его инсталлировать:
2. Инсталлировать среду выполнения Node.js следуя инструкциям по установке;
3. Склонируйте локально репозиторий сервиса с https://github.com/AntonGorban/html2img-service.git;
4. Установить зависимости проекта, командой npm install;
5. Соберите приложение командой npm run build;
6. Переместите собранное приложение из директории build в подготовленную директорию для приложения;
7. Запустите приложение с помощью PM2, командой pm2 start index.js;
8. Протестируйте приложение.
9. Дополнительно вы можете настроить домен для сервиса используя Nginx или Apache. Так-же вы можете настроить firewall ufw или iptables.

5.3 Описание контрольных примеров

1. Для использования микро-сервиса в вашем проекте вы можете использовать fetch, axios и т.п. для JavaScript / TypeScript, или подобные библиотеки в других языках программирования.
2. Для использования микро-сервиса, вам нужно отправить XHR запрос методом POST на URI по шаблону:

* Если вы не подключали домен: (протокол)://(ip):(port)/(тип возвращаемого изображения);
* Если вы подключали домен: (протокол)://(домен)/(тип возвращаемого изображения).

1. Тестировать микро-сервис мы будем с помощью программного обеспечения Postman.
2. Создайте новый запрос в Postman, выберите метод запроса POST (рисунок А.1). Введите URL-адрес сервиса по вышеописанному шаблону (рисунок А.2). Далее введите параметры запроса во вкладке «Params» (рисунок А.3) и тело запроса на вкладке «Body», выбрав «form-data»  
   (рисунок А.4). В тело запроса вы можете добавлять изображения с придуманным полем «KEY». В дальнейшем ваши изображения будут вставлены в HTML и CSS, на месте слов соответствующим «KEY».
3. После заполнения всех необходимых данных, вы можете нажать на кнопку «Send». Таким образом вы отправите запрос на сервер, он провалидирует ваши данные, и если все хорошо, сгенерирует и отдаст вам изображение (рисунок А.5). А если, ваши данные окажутся невалидными, тогда сервер отправит вам ошибку (рисунок А.6).
4. ЗАКЛЮЧЕНИЕ
5. В результате разработки спроектирован микро-сервис для преобразования HTML разметки в изображение.
6. Разработано приложение, которое предоставляет возможность генерации изображения на основе HTML разметки.
7. Приложение осуществляет генерацию изображения на основе HTML разметки.
8. Для решения задачи был использован язык программирования TypeScript.
9. Приложение html2img-service может использоваться front-end и back-end разработчиками, web-мастерами и дизайнерами для генерации изображений на основе HTML разметки.
10. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ
11. Express.js : официальная документация фреймворка «Express»: [сайт] — URL: https://expressjs.com/en/starter/installing.html;
12. TypeScript : официальная документация по языку программирования «TypeScript»: [сайт] — URL: https://www.typescriptlang.org/docs/;
13. Wanago : блог по написанию приложения на Express использую язык программирования TypeScript : [сайт] — URL: https://wanago.io/courses/typescript-express-tutorial/;
14. Puppeteer : Официальная документация библиотеки «Puppeteer» : [сайт] — URL: https://pptr.dev;
15. Node package manager (npm) : Официальная документация библиотеки «express-fileupload» : [сайт] — URL: https://www.npmjs.com/package/express-fileupload;
16. Node package manager (npm) : Официальная документация библиотеки «fs-extra» : [сайт] — URL: https://www.npmjs.com/package/fs-extra;
17. AJV : официальная документация библиотеки «ajv» : [сайт] — URL: https://ajv.js.org/guide/getting-started.html;
18. Lodash : Официальная документация библиотеки «Lodash» : [сайт] — URL: https://lodash.com/docs/4.17.15.
19. ПРИЛОЖЕНИЕ А  
      
    Пример использования



Рисунок A.1 – Главная страница приложения



Рисунок A.2 – Ввод кода и названия вида вина

1. ПРИЛОЖЕНИЕ Б  
     
   Фрагменты листинга
2. Листинг Б1 – Файл «connect.inc»

<?php

// Переменные для работы с базой данных

$data\_baza\_pc="localhost";

$data\_baza\_user="root";

$data\_baza\_password="admin";

$data\_baza\_name="VINO";

$data\_baza\_table\_vidvina="vidvina";

$data\_baza\_table\_cvetvina="cvetvina";

$data\_baza\_table\_viderjkavina="viderjkavina";

$data\_baza\_table\_thraneniyavina="thraneniyavina";

$data\_baza\_table\_garnir="garnir";

$data\_baza\_table\_productigarnira="productigarnira";

$data\_baza\_table\_nazvanievina="nazvanievina";

// Устанавливаем связь с удаленным компьютером

mysql\_pconnect("$data\_baza\_pc","$data\_baza\_user","$data\_baza\_password") or die("Ошибка при соединении с удаленным хостом!");

// Соединяемся с базой данных на сервере

$db=mysql\_select\_db($data\_baza\_name);

?>

1. Листинг Б2 – Скрипт «vidivin.php»

<?php

include("connect.inc");

// Проверяем на ошибку соединения с базой данных

if (!$db) { print "Нет такой базы данных.";} else {

// \*\*\*\* Работа с таблицей vidvina базы данных \*\*\*\*\*\*\*\*\*

print "<HTML><HEAD>";

print "<link type=\"text/css\" rel=stylesheet href=\"css/style.css\">";

print "<TITLE>Виды вин</TITLE></HEAD>";

print "<div align=center>";

// Проверяем, существует ли таблица c именем vidvina

$q=mysql\_query("desc $data\_baza\_table\_vidvina");

// Если не существует, то создаем таблицу и ее поля

if (mysql\_errno()!=0) {

mysql\_query("create table $data\_baza\_table\_vidvina(

kodvidavina tinyint NOT NULL primary key, vidvina varchar(13))");

print "<div align=center> Таблица Виды вина создана успешно! </div>";}

print "<BR><BR>Существующие виды вин и их коды<BR><BR>";

print "<form name=formvidivin method=post action=\"dobav\_vidvina.php\">";

// Таблица существует

print "<table><tr>";

// Получаем коды видов вин из таблицы

$q=mysql\_query("select kodvidavina from $data\_baza\_table\_vidvina");

// Выводим элемент формы Select для кодов видов вин

print "<td>";print "<select size=\"5\" name=\"f1\_kodvidavina\">";

while ($row=mysql\_fetch\_row($q)) {

print "<option value='$row[0]'>$row[0]</option>";}

print "</select>";print "</td>";

// Получаем виды вин из таблицы

$q=mysql\_query("select vidvina from $data\_baza\_table\_vidvina");

// Выводим элемент формы Select для видов вин

print "<td>";print "<select size=\"5\" name=\"f1\_vidvina\">";

while ($row=mysql\_fetch\_row($q)) {

print "<option value='$row[0]'>$row[0]</option>";}

print "</select>"; print "</td>"; print "</tr></table>";

print "<BR>"; print "<BR>"; print "<BR>"; print "<table><tr><td>";

print "Введите код вида вина для добавления в таблицу ВИДЫ ВИН";

print "</td>"; print "<td>";

print "<input type=\"text\" name=\"f\_kodvidavina\">";

print "</td>"; print "</tr>"; print "<tr><td>";

print "Введите вид вина для добавления в таблицу ВИДЫ ВИН";

print "</td>"; print "<td>";

print "<input type=\"text\" name=\"f\_vidvina\">";

print "</td>"; print "</tr></table>";

print "<input type=\"submit\" value=\"Добавить вид вина\" name=\"B1\">";

print "&nbsp;&nbsp;&nbsp;";

print "<input type=\"reset\" value=\"Отмена\" name=\"B2\">";

print "<BR>"; print "</form>";

// \*\*\*\*\*\*\*\*\*\*\*\*\*\* Для удаления \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

print "<BR>"; print "<form name=formdelvidivin method=post action=\"udalit\_vidvina.php\">";

print "<table><tr><td>";

print "Введите код вида вина для удаления из таблицы ВИДЫ ВИН";

print "</td>";

// Получаем коды видов вин из таблицы

$q=mysql\_query("select kodvidavina from $data\_baza\_table\_vidvina");

// Выводим элемент формы Select для кодов видов вин

print "<td>"; print "<select size=\"1\" name=\"fd\_kodvidavina\">";

print "<option selected value=''></option>";

while ($row=mysql\_fetch\_row($q)) {

print "<option value='$row[0]'>$row[0]</option>";}

print "</select>"; print "</td>"; print "</tr>"; print "</table>";

print "<input type=\"submit\" value=\"Удалить вид вина\" name=\"B3\">";

print "&nbsp;&nbsp;&nbsp;";

print "<input type=\"reset\" value=\"Отмена\" name=\"B4\">";

print "<BR>";print "</form>";

print"<a href='index.htm'> Вернуться на главную </a>";

print "</div></BODY></HTML>";} // \*\*\* конец работы с таблицей БД

?>