МИНИСТЕРСТВО НАУКИ и высшего образования РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ»

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

филиал «РКТ» МАИ в г. Химки Московской области

**Специальность 09.02.03 «Программирование в компьютерных системах»**

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**ПМ.02«Разработка и администрирование баз данных**»

**Студент**

**Группы МП-31** Молоткин Антон  **/ ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ )**

**Руководитель**

**практики от организации \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)**

**Руководитель**

**практики от филиала Шумаев А.Ю. / ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ )**

**2020г.**

**ПРОГРАММА ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ**

По специальности 09.02.03 «Программирование в компьютерных системах»

|  |  |  |
| --- | --- | --- |
| Дата | Наименование выполняемых работ | Количество дней практики |
| 11.06.20 | Ознакомление с программой практики. Прохождение инструктажа по технике безопасности и охране труда, изучение внутреннего распорядка организации и правил работы. Разработка задания на производственную практику. | 4 |
| 12.06.20 | Описание структуры сети, в которой может функционировать разработанная база данных. | 4 |
| 13.06.20 | Описание сетевого оборудования необходимого для работы с базой данных. | 4 |
| 15.06.20 | Описание программного обеспечения необходимого для работы с базой данных по сети. | 4 |
| 16.06.20-18.06.20 | Разработка клиентских интерфейсов для клиент-серверных приложений. | 16 |
| 20.06.20 | Создание инфокоммуникационной системы.  Выполнение сетевых настроек для взаимодействия с СУБД. | 4 |
| 22.06.20-24.06.20 | Создание концептуальной модели данных.  Создание логической модели данных.  Создание физической модели данных.  Описание механизмов обеспечения целостности базы данных. | 16 |
| 25.06.20-01.07.20 | Описание СУБД представленной базы данных и ее возможностей. Внесение различных данных в базу данных.  Структурирование запросов базы данных.  Определение методов создания хранимых процедур и триггеров. Создание хранимых процедур и триггеров базы данных. | 50 |
| 02.07.20-03.07.20 | Определение способов управления правами пользователей.  Описание распределения прав пользователей и управления ими в базе данных. | 12 |
| 03.07.20-04.07.20 | Определение методов создания и синхронизации реплик базы данных. Описание существующих механизмов репликации в базе данных. Определение методов создания резервных копий базы данных. Описание существующих механизмов резервного копирования в базе данных. | 12 |
| 06.07.20 | Составление отчёта по практике | 6 |
| 07.07.20 | Подготовка к защите отчёта по практике. | 6 |
| 08.07.20 | Итоговая аттестация по производственной практике - зачёт | 6 |

Руководитель практики от филиала «РКТ» МАИ преподаватель

Шумаев А.Ю.

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

**АТТЕСТАЦИОННЫЙ ЛИСТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

студент Молоткин Антон Максимович

*ФИО*

обучающийся (аяся) на 3-ем курсе по специальности СПО 09.02.03 «Программирование в компьютерных системах»

успешно прошел(ла) производственную практику по профессиональному модулю   
**ПМ.02 «Разработка и администрирование баз данных»**

в объеме 144 часа с 11.06.2020г. по 08.07.2020г.

в организации филиала «РКТ» МАИ

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| Виды работ, выполненных обучающимся(ейся) во время практики | Объем работ | Качество выполнения работ в соответствии с технологией и (или) требованиями организации, в которой проходила практика |
| Ознакомление с программой практики. Прохождение инструктажа по технике безопасности и охране труда, изучение внутреннего распорядка организации и правил работы. Разработка задания на производственную практику. | 4 |  |
| Описание структуры сети, в которой может функционировать разработанная база данных. | 4 |  |
| Описание сетевого оборудования необходимого для работы с базой данных. | 4 |  |
| Описание программного обеспечения необходимого для работы с базой данных по сети. | 4 |  |
| Разработка клиентских интерфейсов для клиент-серверных приложений. | 16 |  |
| Создание инфокоммуникационной системы.  Выполнение сетевых настроек для взаимодействия с СУБД. | 4 |  |
| Создание концептуальной модели данных.  Создание логической модели данных.  Создание физической модели данных.  Описание механизмов обеспечения целостности базы данных. | 16 |  |
| Описание СУБД представленной базы данных и ее возможностей. Внесение различных данных в базу данных.  Структурирование запросов базы данных.  Определение методов создания хранимых процедур и триггеров. Создание хранимых процедур и триггеров базы данных. | 50 |  |
| Определение способов управления правами пользователей.  Описание распределения прав пользователей и управления ими в базе данных. | 12 |  |
| Определение методов создания и синхронизации реплик базы данных. Описание существующих механизмов репликации в базе данных. Определение методов создания резервных копий базы данных. Описание существующих механизмов резервного копирования в базе данных. | 12 |  |
| Составление отчёта по практике | 6 |  |
| Подготовка к защите отчёта по практике. | 6 |  |
| Итоговая аттестация по производственной практике - зачёт | 6 |  |

Руководитель практики от филиала «РКТ» МАИ Шумаев А.Ю.

|  |  |  |
| --- | --- | --- |
| Дата\_\_\_\_\_\_\_\_\_\_\_\_\_ | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *подпись* | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *Расшифровка подписи* |

**ДНЕВНИК** **ПРОХОЖДЕНИЯ ПРАКТИКИ**

|  |  |  |
| --- | --- | --- |
| Дата | Наименование выполняемых работ | Подпись  руководителя |
| 11.06.20 | Ознакомление с программой практики. Прохождение инструктажа по технике безопасности и охране труда, изучение внутреннего распорядка организации и правил работы. Разработка задания на производственную практику. |  |
| 12.06.20 | Описание структуры сети, в которой может функционировать разработанная база данных. |  |
| 13.06.20 | Описание сетевого оборудования необходимого для работы с базой данных. |  |
| 15.06.20 | Описание программного обеспечения необходимого для работы с базой данных по сети. |  |
| 16.06.20-18.06.20 | Разработка клиентских интерфейсов для клиент-серверных приложений. |  |
| 20.06.20 | Создание инфокоммуникационной системы.  Выполнение сетевых настроек для взаимодействия с СУБД. |  |
| 22.06.20-24.06.20 | Создание концептуальной модели данных.  Создание логической модели данных.  Создание физической модели данных.  Описание механизмов обеспечения целостности базы данных. |  |
| 25.06.20-01.07.20 | Описание СУБД представленной базы данных и ее возможностей. Внесение различных данных в базу данных.  Структурирование запросов базы данных.  Определение методов создания хранимых процедур и триггеров. Создание хранимых процедур и триггеров базы данных. |  |
| 02.07.20-03.07.20 | Определение способов управления правами пользователей.  Описание распределения прав пользователей и управления ими в базе данных. |  |
| 03.07.20-04.07.20 | Определение методов создания и синхронизации реплик базы данных. Описание существующих механизмов репликации в базе данных. Определение методов создания резервных копий базы данных. Описание существующих механизмов резервного копирования в базе данных. |  |
| 06.07.20 | Составление отчёта по практике |  |
| 07.07.20 | Подготовка к защите отчёта по практике. |  |
| 08.07.20 | Итоговая аттестация по производственной практике - зачёт |  |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc54290504)

[1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОЕКТА 8](#_Toc54290505)

[1.1 Система управления версиями Git 8](#_Toc54290506)

[1.2 Интернет сервис GitHub 9](#_Toc54290507)

[2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА 13](#_Toc54290508)

[2.2 Тестирование программного продукта 22](#_Toc54290509)

[2.3 Баг-репорты программного продукта 25](#_Toc54290510)

[ЗАКЛЮЧЕНИЕ 28](#_Toc54290511)

[СПИСОК ИСПОЛЬЗВАННЫХ ИСТОЧНИКОВ 29](#_Toc54290512)

# ВВЕДЕНИЕ

Информационные технологии не стоят на месте, с каждым годом появляется всё больше систем, появляются новые инструменты. Скорость работы программ увеличивается по мере увеличения мощности серверов, в наши дни, это особенно актуально, так как от скорости принятия решений зависит успех бизнеса. Так же, каждый год повышается пропускная способность каналов связи, что упрощает синхронную работу подразделений, находящихся в разных регионах.

Практика – приобретение практических навыков работы в области создания, функционирования и использования систем управления базами данных

Цель учебной практики — развитие профессионального интереса, формирование мотивационно-целостного отношения к профессиональной деятельности, готовности к выполнению профессиональных задач в соответствии с нормами морали, профессиональной этики и служебного этикета;

Объект – профессиональная деятельность оператора электронно-вычислительных и вычислительных машин.

Цель – формирование знаний, умений и навыков, профессиональных компетенций, профессионально значимых личностных качеств.

# 1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОЕКТА

## 1.1 Система управления версиями Git

Git — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года.

Система спроектирована как набор программ, специально разработанных с учётом их использования в сценариях. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером оболочки к репозиториям Git, а StGit использует Git для управления коллекцией исправлений (патчей).

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов, и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов. Любое файловое дерево в системе можно превратить в репозиторий git, отдав команду создания репозитория из корневого каталога этого дерева (или указав корневой каталог в параметрах программы). Репозиторий может быть импортирован с другого узла, доступного по сети. При импорте нового репозитория автоматически создаётся рабочая копия, соответствующая последнему зафиксированному состоянию импортируемого репозитория (то есть не копируются изменения в рабочей копии исходного узла, для которых на том узле не была выполнена команда commit).

Репозиторий Git бывает локальный и удалённый. Локальный репозиторий — это подкаталог git, создаётся (в пустом виде) командой git init и (в непустом виде с немедленным копированием содержимого родительского удалённого репозитория и простановкой ссылки на родителя) командой git clone [1].

Практически все обычные операции с системой контроля версий, такие, как коммит и слияние, производятся только с локальным репозиторием. Удалённый репозиторий можно только синхронизировать с локальным как «вверх» (push), так и «вниз» (pull).

Наличие полностью всего репозитория проекта локально у каждого разработчика даёт Git ряд преимуществ перед SVN. Так, например, все операции, кроме push и pull, можно осуществлять без наличия интернет-соединения.

Команда push передаёт все новые данные (те, которых ещё нет в удалённом репозитории) из локального репозитория в репозиторий удалённый. Для исполнения этой команды необходимо, чтобы удалённый репозиторий не имел новых коммитов в себя от других клиентов, иначе push завершается ошибкой, и придётся делать pull и слияние.

Команда pull — обратна команде push. В случае, если одна и та же ветвь имеет независимую историю в локальной и в удалённой копии, pull немедленно переходит к слиянию.

Имя ветви по умолчанию: master. Имя удалённого репозитория по умолчанию, создаваемое git clone во время типичной операции «взять имеющийся проект с сервера себе на машину»: origin.

Таким образом, в локальном репозитории всегда есть ветвь master, которая есть последний локальный коммит, и ветвь origin/master, которая есть последнее состояние удалённого репозитория на момент завершения исполнения последней команды pull или push.

Команда fetch (частичный pull) — берёт с удалённого сервера все изменения в origin/master, и переписывает их в локальный репозиторий, продвигая метку origin/master [1].

1.2 Интернет сервис GitHub

GitHub — сервис онлайн-хостинга репозиториев, обладающий всеми функциями распределённого контроля версий и функциональностью управления исходным кодом — всё, что поддерживает Git и даже больше. Обычно он используется вместе с Git и даёт разработчикам возможность сохранять их код онлайн, а затем взаимодействовать с другими разработчиками в разных проектах.

Также GitHub может похвастаться контролем доступа, багтрекингом, управлением задачами и вики для каждого проекта. Цель GitHub — содействовать взаимодействию разработчиков [1].

К проекту, загруженному на GitHub, можно получить доступ с помощью интерфейса командной строки Git и Git-команд. Также есть и другие функции, такие как документация, запросы на принятие изменений (pull requests), история коммитов, интеграция со множеством популярных сервисов, email-уведомления, эмодзи, графики, вложенные списки задач, система @упоминаний, похожая на ту, что в Twitter, и т.д.

Git — это инструмент, позволяющий реализовать распределённую систему контроля версий, а GitHub — это сервис для проектов, использующих Git.

GitHub также является одним из крупнейших онлайн-хранилищ (англоязычной) совместной работы по всему миру.

Репозиторий или хранилище — это каталог, в котором хранятся файлы проекта. Он может быть расположен в хранилище GitHub или в локальном хранилище на компьютера. Можно хранить файлы кодов, изображения, аудио или всё, что связано с проектом, в хранилище.

Система “Звезд”. Одной из отличительных особенностей GitHub является система звёзд. Чтобы выразить интерес к репозиторию, его нужно отметить звездой. Это можно сделать с помощью кнопки «Star». Что позволяет отслеживать интересные проекты и находить похожие. Это также один из самых важных рейтинговых механизмов, поскольку чем больше звезд имеет репозиторий, тем он популярнее. Поэтому в результатах поиска он будет в топовых местах. Крупные проекты могут иметь десятки тысяч звезд.

Также в GitHub есть трендовая страница, на которой представлены репозитории, которые получают наибольшее количество звезд за определенный период времени (за день, неделю, месяц).

Fork - это копия репозитория. Это то же самое, что branch в Git.

Если есть возможность внести вклад в уже существующие проекты, в которые нет прав на внесения изменений путем отправки (push) изменений, можно создать собственное ответвление (“fork”) проекта. Это означает, что GitHub создаст собственную копию проекта, данная копия будет находиться в репозитории и можно легко делать изменения путем отправки (push) изменений. Также другой человек может разветвить репозиторий, внести некоторые изменения, а затем создать запрос на внесение этих изменений.

Pull Request. Человек может создать свое собственное ответвление (“fork”) проекта, внести некоторые изменения и затем сделать Pull Request., чтобы замерджили эти изменения.

Как только сделано Pull Request, рассматривается основными разработчиками проекта. В зависимости от количества и сложности изменений, которое внесено в код, разработчику может потребоваться разное количество времени, чтобы убедиться, что изменения совместимы с проектом.

У проекта может быть четкий график изменений, которые разработчики хотят внедрить. Тогда запросы на внесение изменений в код будут рассмотрены быстро. Но запрос может и не быть рассмотрен вовсе.

Управление проектами

Кроме обратной связи и новых знакомств, GitHub также предоставляет некоторые функции по управлению проектами.

Одна из таких функций - Projects. Нововведение в GitHub. Она основывается на методологии Канбан, которая помогает эффективно организовать работу.

Wiki предназначен для использования в качестве документации для пользователей. Одним из самых впечатляющих видов использования Wiki, является язык программирования Go GitHub Wiki.

Еще один популярный инструмент для управления проектами -вехи. Это часть страницы "issues ". Используя эту функцию можно назначать контрольные точки проекта.

Тэги в Git - это указатель на конкретный коммит, и если он выполняется последовательно, он помогает вернуться к предыдущей версии вашего кода, не ссылаясь на конкретные коммиты.

Сравнение коммитов

GitHub предлагает множество инструментов для работы с вашим кодом.

Одна из самых важных вещей, которые нужно сделать - сравнить одну ветку с другой. Или сравнить последний коммит с используемой версией, чтобы увидеть, какие изменения были внесены.

Ветка- это копия репозитория. Можно использовать ветку, когда есть возможность сделать разработку изолированно.

Работа с веткой не повлияет на центральное хранилище или другие ветки. Если сделана эта работа, то есть возможность объединить собственную ветку с другими ветками и центральным репозиторием, используя запрос на извлечение.

Запрос на извлечение означает, что сообщаете другим, что вы передали изменения, внесённые вами в ветке, в главный репозиторий. Соавторы хранилища могут принять или отклонить запрос на извлечение. После его открытия обсуждается и анализируется работа с соавторами.

Инструкции по созданию запросов на извлечение в GitHub:

* Перейти в хранилище и найти ветку меню
* В меню выбрать ветку, которая содержит нужный коммит;
* Нажать кнопку «Новый запрос» на извлечение рядом с меню ветки;
* Вставить заголовок и описать запрос;
* Нажать кнопку «Создать запрос» на извлечение

Форкинг репозитория означает, что создаётся новый проект на основе существующего репозитория, разветвление репозитория означает, что копируется существующий репозиторий, нужно внести некоторые необходимые изменения, сохранить новую версию в качестве нового репозитория и называть собственным проектом.

Это отличная функция, которая ускоряет разработку проекта. Поскольку это совершенно новый проект, центральное хранилище не будет затронуто. Если «главный» репозиторий обновлён, то также можно применить это обновление к текущему форку.

Инструкции по созданию репозиторий в GitHub:

* Найти репозиторий, который необходимо разветвлять
* Найти кнопку Форк

GitHub не ограничен только для разработчиков. GitHub это отличная платформа, которая меняет работу разработчиков. Тем не менее, каждый, может эффективно управлять своим проектом и работать совместно [1].

# 2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

2.1 Разработка Gulp проекта

Gulp — это таск-менеджер для автоматического выполнения часто используемых задач, например, минификации, тестирования, объединения файлов, написанный на языке программирования JavaScript. Программное обеспечение использует командную строку для запуска задач, определённых в файле Gulpfile. Создан как ответвление от проекта Grunt, чтоб взять из него лучшие практики. Распространяется через менеджер пакетов NPM под MIT лицензией.

Gulp предоставляет по-настоящему широкие возможности. Он облегчит и ускорит frontend разработку. Основные задачи, которые сборщик проектов вам поможет решить:

* Создание веб-сервера для отладки
* Автоматическая перезагрузка страниц при внесении изменений (LiveReload)
* Слежение за изменениями в файлах проекта
* Использование препроцессоров HTML, CSS, JS
* Объединение файлов и их минификация
* Автоматическое создание вендорных префиксов для браузеров (Autoprefixer)
* Автоматизация управления файлами и директориями
* Запуск и контроль внешних команд операционной системы
* Запуск и контроль приложений
* Оптимизация изображений (сжатие, изменение размеров и т.д.)
* Выгрузка проекта на внешний сервер с помощью FTP, SFTP, Git и т.д.
* Подключение и использование дополнительных плагинов
* Автоматизация ручного труда

Первое, что нужно сделать это установить node.js. Данный пакет превращает клиентский язык javaScript в серверный. Далее необходимо скачать инсталлятор с официального сайта. После установки node.js можно проверить правильность инсталляции. Для этого, надо открыть консольное окно и ввести команду node –v. Если все правильно установилось, то в ответ можно увидеть версию пакета. На рисунке 1 показан результат команды.

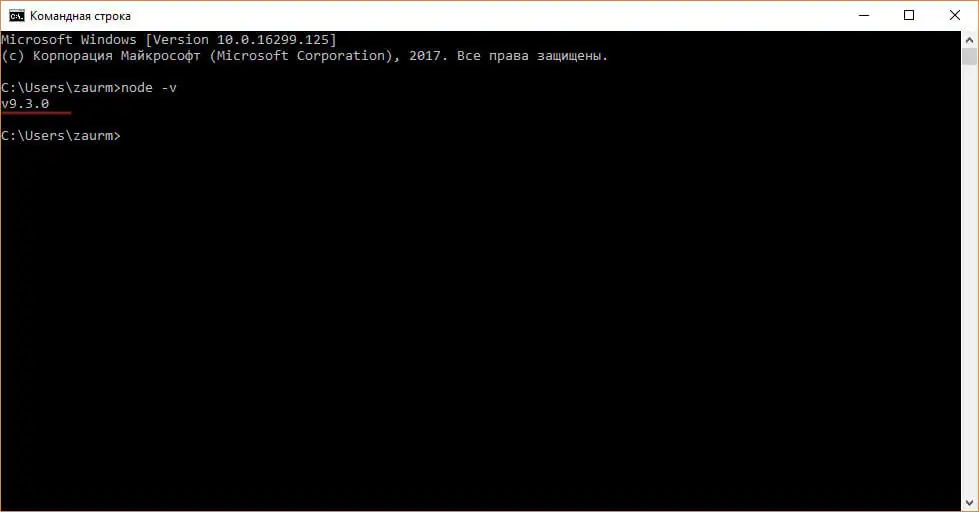


Рисунок 1 – Результат установки пакета node.js

Далее устанавливаем Gulp. Для этого надо написать команду npm i gulp –g. Проверить правильность установки можно так же, как и с node.js, введя в консольное окно команду gulp –v. На рисунке 2 показан результат команды.

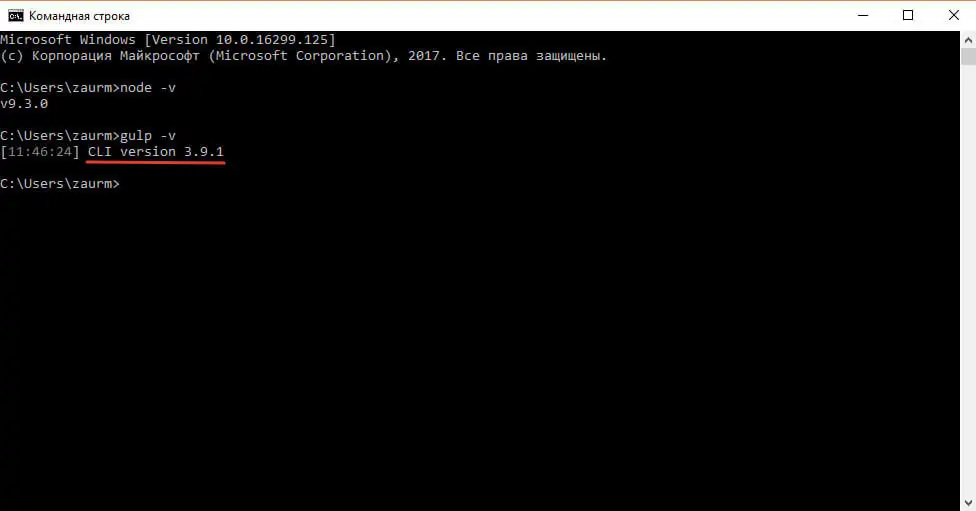


Рисунок 2 – Результат установки Gulp

Для начала работы Gulp надо создать папку проекта. Для примера была создана папка под названием gproject. Перейдем в папку проекта и запустим консоль команд для данного каталога. Наиболее быстрый вариант сделать это зажать клавишу «Shift» и удерживая ее щелкнуть правой кнопкой мыши на пустой области окна каталога. Откроется контекстное меню, в котором выбираем «Открываем окно PowerShell здесь». Данный пункт может называться и по-другому — «Открыть окно команд». Далее, запускаем инициализацию проекта командой npn init. Заполняем необходимые поля проекта по шагам. После ввода названия жмем Enter и переходим с следующему шагу.

* package-name: вводим название проекта маленькими буквами
* version: оставляем по умолчанию — 1.0.0
* description: вводим описание проекта
* entry point: (index.js), test command:, git repository:, keywords: — данные шаги оставляем по умолчанию, жмем Enter и переходим к следующему шагу
* author: впишите имя автора
* license: оставляем по умолчанию
* Is this ok? — вводим «yes» и жмем Enter поле чего в папке проекта появится файл package.json.

На рисунке 3 изображены данные шаги

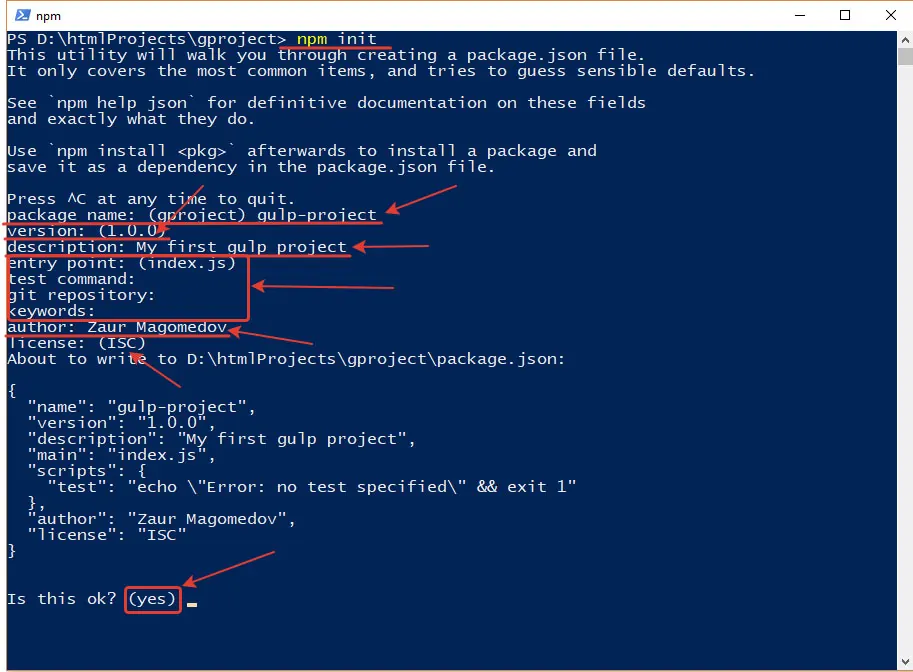


Рисунок 3 – Настройка проекта

Файл package.json содержит в себе информацию о проекте и информацию об установленных пакетах (плагинов). Далее. установим Gulp локально в папку нашего проекта. Для этого пишем следующую команду npm i gulp --save-dev. Флаг —save-dev говорит установщику установить gulp локально в папку проекта.

При установке gulp название пакета и его версия автоматически пропишутся в файле package.json. Вообще такой подход позволяет сохранять файл package.json со всеми установленными пакетами (зависимости), а при разворачивании нового проекта достаточно скопировать данный файл и запустить команду установки в консоли проекта — npm install и все пакеты в проект установятся автоматически.

Далее, создадим структуру проекта. Структуру будет создаваться с учетом логики по общепринятым правилам хорошего тона веб-разработчиков. Чуть ниже приведен пример структуры проекта:

gproject

* dist/
* src/
  + images/
  + js/
  + sass/
  + index.html
* node\_modules/
* gulpfile.js
* package.json

dist — в данную папку будут складываться все готовые файлы после компиляции. Другое ее название —папка продакшена, т.е. итог работы. Папка dist содержит в себе вложенные папки.

src — здесь будут храниться исходники файлов проекта. В дочерних папках может быть очень много файлов, зависит от размера проекта. Каждые типы файлов объединяются в общий файл и копируются в папку продакшена. В корне лежит файл index.html

image — неоптимизированные изображения и иконки проекта.

js — файлы javaScript.

sass — файлы препроцессоров SASS или SCSS.

node\_modules — все модули и зависимости gulp.

gulpfile.js — это главный файл пакета Gulp.

Gulpfile.js — важный файл без которого Gulp проект работать не может. Здесь записываются все задачи для выполнения — это так называемые таски (task), что в переводе с английского означает — задача. В основном весь gulpfile состоит из сплошных тасков. Таски создаются для подключаемых дополнительных плагинов. Без таска плагин работать не будет.

Для начала, надо создать в корне своего проекта файл gulpfile.js и вставить в него следующий код:

var gulp = require("gulp");

gulp.task('myFirstTask', function() {

console.log('Привет, я твой первый таск!');

});

gulp.task("default", ["myFirstTask"]);

Для проверки, вводим команду — gulp и если ошибок нет, программа должна выдать «Привет, я твой первый таск!». Результат показан на рисунке 4

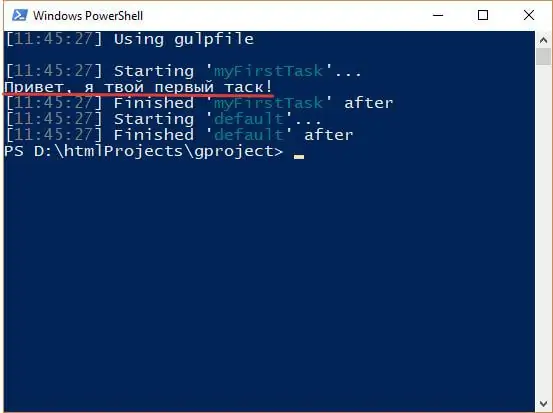


Рисунок 4 – Создание первого таска

Далее, создадим небольшой проект для верстки, установив некоторые плагины, которые упростят работу. Открываем gulpfile.js и cразу под строкой var gulp = require("gulp"); запишем подключение плагинов:

var sass = require("gulp-sass"), // переводит SASS в CSS

cssnano = require("gulp-cssnano"), // Минимизация CSS

autoprefixer = require('gulp-autoprefixer'), // Проставляет вендорные префиксы в CSS для поддержки старых браузеров

imagemin = require('gulp-imagemin'), // Сжатие изображений

concat = require("gulp-concat"), // Объединение файлов - конкатенация

uglify = require("gulp-uglify"), // Минимизация javascript

rename = require("gulp-rename"); // Переименование файлов

Для установки плагинов необходимо написать команду: npm i gulp-sass gulp-cssnano gulp-autoprefixer gulp-imagemin gulp-concat gulp-uglify gulp-rename --save-dev

Теперь необходимо создать таски для плагинов. Внизу показан текущий код в файле gulpfile.js:

// Подключаем Gulp

var gulp = require("gulp");

// Подключаем плагины Gulp

var sass = require("gulp-sass"), // переводит SASS в CSS

cssnano = require("gulp-cssnano"), // Минимизация CSS

autoprefixer = require('gulp-autoprefixer'), // Проставлет вендорные префиксы в CSS для поддержки старых браузеров

imagemin = require('gulp-imagemin'), // Сжатие изображение

concat = require("gulp-concat"), // Объединение файлов - конкатенация

uglify = require("gulp-uglify"), // Минимизация javascript

rename = require("gulp-rename"); // Переименование файлов

// Копирование файлов HTML в папку dist

gulp.task("html", function() {

return gulp.src("src/\*.html")

.pipe(gulp.dest("dist"));

});

// Объединение, компиляция Sass в CSS, простановка венд. префиксов и дальнейшая минимизация кода

gulp.task("sass", function() {

return gulp.src("src/sass/\*.sass")

.pipe(concat('styles.sass'))

.pipe(sass())

.pipe(autoprefixer({

browsers: ['last 2 versions'],

cascade: false

}))

.pipe(cssnano())

.pipe(rename({ suffix: '.min' }))

.pipe(gulp.dest("dist/css"));

});

// Объединение и сжатие JS-файлов

gulp.task("scripts", function() {

return gulp.src("src/js/\*.js") // директория откуда брать исходники

.pipe(concat('scripts.js')) // объеденим все js-файлы в один

.pipe(uglify()) // вызов плагина uglify - сжатие кода

.pipe(rename({ suffix: '.min' })) // вызов плагина rename - переименование файла с приставкой .min

.pipe(gulp.dest("dist/js")); // директория продакшена, т.е. куда сложить готовый файл

});

// Сжимаем картинки

gulp.task('imgs', function() {

return gulp.src("src/images/\*.+(jpg|jpeg|png|gif)")

.pipe(imagemin({

progressive: true,

svgoPlugins: [{ removeViewBox: false }],

interlaced: true

}))

.pipe(gulp.dest("dist/images"))

});

// Задача слежения за измененными файлами

gulp.task("watch", function() {

gulp.watch("src/\*.html", ["html"]);

gulp.watch("src/js/\*.js", ["scripts"]);

gulp.watch("src/sass/\*.sass", ["sass"]);

gulp.watch("src/images/\*.+(jpg|jpeg|png|gif)", ["imgs"]);

});

// Запуск тасков по умолчанию

gulp.task("default", ["html", "sass", "scripts", "imgs", "watch"]);

Разберем состав отдельного таска:

gulp.task('taskname', function () {

return gulp.src('source-files')

.pipe(plugin-name())

.pipe(gulp.dest('folder'))

});

taskname — название таска — название может произвольным;

source-files — директория до файлов исходников, необходимо указать полный путь и расширения файлов, например, путь до всех фалов с расширением .js будет таким — templatename/js/\*.js;

.pipe() — метод, который позволяет работать с установленным плагином

plugin-name — название плагина;

dest(«filder-path») — путь до папки продакшена, т.е. куда выгрузить после отработки плагина

Данный проект готов к работе. Для проверки его работоспособности надо написать в командной строке – gulp. Если нет ошибок, то появится результат, как на рисунке 5.

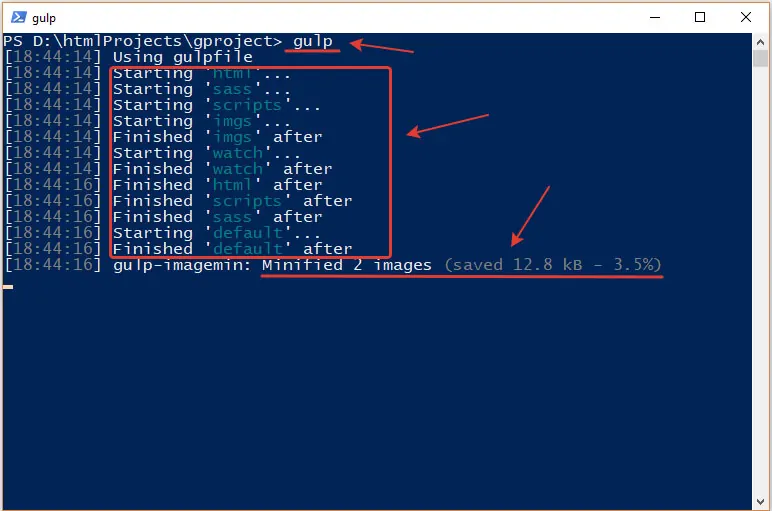


Рисунок 5 – Таски для плагинов

2.2 Тестирование программного продукта

Тестирование программного продукта проводят на этапах создания самого продукта, и на этапах технического сопровождения и дальнейшего развития.

Тестированием – называется ряд мероприятий, связанных с различного рода испытаниями объекта тестирования с целью установления соответствия или несоответствия его характеристик определенным требованиям заказчика и выявления дефектов. Дефектами могут быть ошибки в работе или неприемлемое качество функционирования в определенных условиях эксплуатации.

Результатами тестирования является список выявленных несоответствий и дефектов, обычно, без указания их причин. Опытный специалист по качеству продукции в состоянии провести первичную диагностику некорректного поведения объекта и сообщить диагноз разработчику программного продукта [3].

Виды тестирования на этапе создания программного продукта подразделяются по требуемым от процесса целям: функциональное тестирование проверяет работу выполняемой программы и заложенных в нее функций, нефункциональное тестирование затрагивает производительность на различных режимах эксплуатации, тестирует пользовательский интерфейс, отказоустойчивость и прочее.

Функциональное тестирование проводится на различных уровнях тестирования, перечень которых зависит от сложности приложения: компонентное и интеграционное тестирование.

Компонентное тестирование – тестирование, при котором производится проверка на работоспособность отдельных компонентов программного продукта, значение которого сфокусировано на специфике, назначении и функциональных особенностях.

Интеграционное тестирование – происходит после компонентного тестирования, назначение которого является выявление дефектов взаимодействия различных подсистем на уровне потоков управления у правления данными.

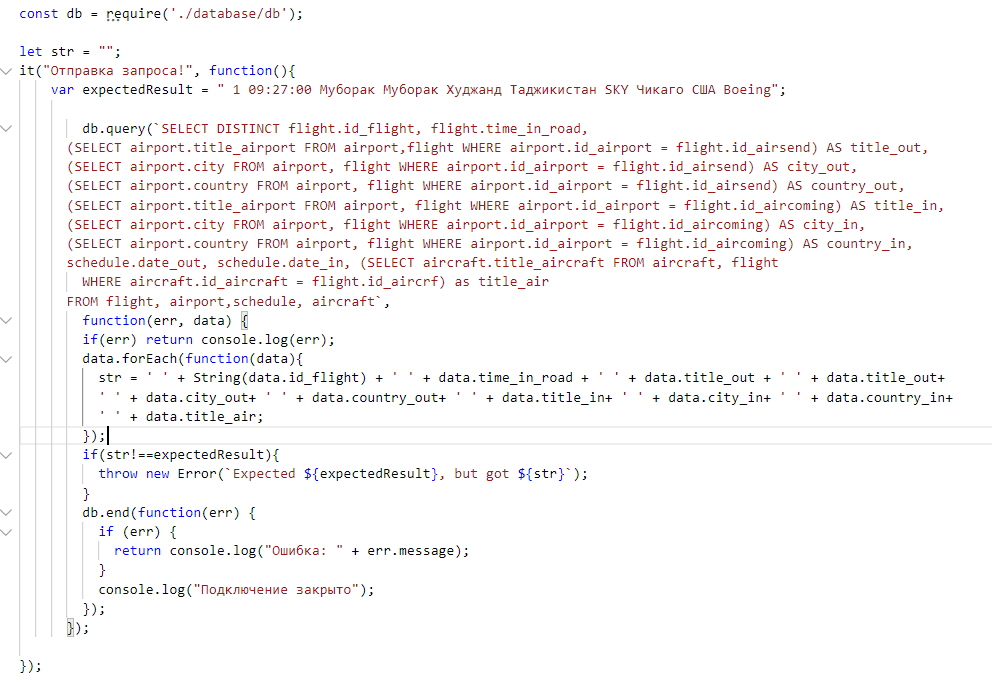
Нефункциональное тестирование рассматривает второстепенные качества функционирования программного продукта. Эти качества менее важны, чем корректное выполнение основных функций.

На примере работы программного продукта сайта «United Airlines w3.aero», помимо компонентного и интеграционного тестирования будут рассмотрены такие методы, как позитивное и негативное тестирования.

Позитивное тестирование представляет собой применение сценариев при тестировании, которые должны отображать нормальную (штатную) работу системы. С помощью такого тестирования возможно определить то, для чего была создана система. Позитивное тестирование направлено на проверку работы системы с типами данных, для которых система разрабатывалась.

Негативным тестированием называется тестирование, при котором так же используются сценарии, отображающие ненормальную (внештатную) работу системы. Негативное тестирование предназначено для проверки работы системы на устойчивость к различным воздействиям, валидации (процессу оценки) неверных данных, обработки исключительных ситуаций [4].

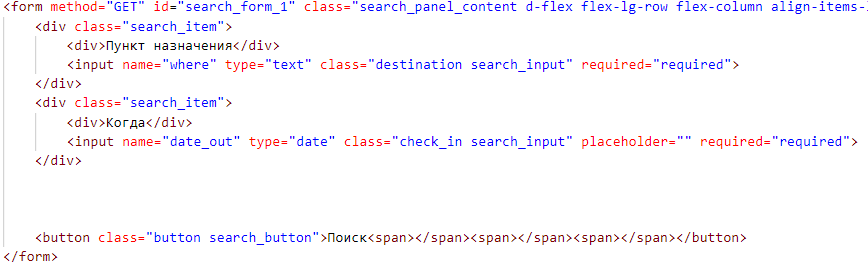
В качестве примера компонентного тестирования был выбран код программы для проверки ввода SQL-запроса в БД. На рисунке 2 отображен фрагмент кода программного продукта проверяющей функции, проверяющей SQL-запрос в базу данных.

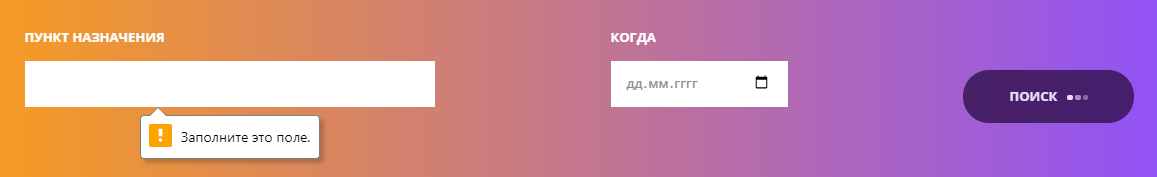
  
Рисунок 2 – Код проверяющей функции

Для примера интеграционного тестирования была выбран код программы, объединяющий модули клиентского и серверной частей, с функцией подключения к базе данных. На рисунке 3 отображен код программы.

  
Рисунок 3 – Код объединения частей клиента и сервера, с подключением к БД

В качестве примера негативного тестирования рассматриваются поля для ввода данных на сайте страницы. Поля для ввода не должны пустыми, поэтому используются атрибуты тега input – required, отображающие требование к заполнению этих полей ввода. На рисунках 4 и 5 отображены протестированный код полей ввода и результат негативного тестирования.

  
Рисунок 4 – Код программы «Негативного» тестирования

  
Рисунок 5 – Результат «Негативного» тестирования

Из нефункционального тестирования программного продукта со стороны клиентской части будет отмечена быстрая загрузка сайта в окне веб-браузера; отличный дизайн изготовленного программного продукта, с интуитивно понятным и простым интерфейсом для пользователя. Из отрицательных моментов тестирования можно отметить отсутствие масштабирование сайта под мобильные или планшетные устройства и невозможность открыть некоторые страницы сайта.

## 2.3 Баг-репорты программного продукта

Баг репорт (bug report) или дефект репорт (defect report) – это документ, описывающий ситуацию или последовательность действий, приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.

Баг репорт – это технический документ, выражаемый проблему техническим языком. В описании бага должна использоваться правильная терминология при использовании названий элементов пользовательского интерфейса (например: editbox, listbox, combobox, textarea, button, menu и так далее), действий пользователя (например: click link, press the button, select menu item и так далее) и полученных результатах (например: window is opened, system crashed и так далее).

Обязательными полями документа баг репорта являются разделы шапки, окружения, описания, и дополнения. В раздел шапки входят поля: короткого описания проблемы (summary), название тестируемого проекта (project), компонент приложения тестируемого продукта (component), номер версии программного продукта на котором была найдена ошибка (version), серьезность бага влиять на работу (severity), приоритет бага на устранение (priority), статус бага (status), автора составившего баг репорт (author), назначение человека из группы разработчиков для решения проблемы (assigned to).

Раздел окружение требует наличия поля общей информации об окружении, на котором был найден баг, этот момент содержит в себе информацию про операционную систему, сервисный набор для тестирования веб-разработок, имя и версию браузера, на котором был найден баг и многое другое.

Раздел описания состоит из следующих полей: шаги воспроизведения ситуации приведших к ошибке (steps to reproduce), фактический результат полученный после прохождения шагов к воспроизведению (result), ожидаемый правильный результат (expected result).

Раздел дополнения содержит в себе прикрепленную информацию, проясняющую причину ошибки, указывающую на способ решения проблемы. В этом разделе может содержаться файл с логами, снимок рабочего стола компьютера (screenshot) или любая другая информация.

Короткое описание проблемы требует уместить смысл всего баг репорта коротко и ясно, используя правильную терминологию описание что и в каком месте в программном продукте не работает. Серьезностью ошибки описывается влияние на функциональную работоспособность программного продукта. Влияние описывается распространённой пятиуровневой системой градации серьезности, от блокирующего работу программного продукта ошибки (уровень S1) и критически необходимой для тестирования (уровень S2), до значительной (уровень S3), незначительной (уровень S4) и тривиальной (уровень S5). Шаги к воспроизведению описывают четкую последовательность действий, с упоминанием всех вводимых данных, и промежуточных результатов [4].

Основными ошибками в составлении баг репорта считаются:

* недостаточность предоставленных данных - необходимость вносить все необходимые данные в баг репорт, иначе проблема может не проявиться при всех вводимых значениях и под любым вошедшим пользователем;
* определение серьезности - часто происходит завышение или занижение серьезности дефекта, что сказывается на очередности при решении проблемы;
* язык описания - использование неправильной терминологии или сложные речевые обороты, вводящие в заблуждение человека, решающего проблему;
* Отсутствие ожидаемого результата – в случае, когда не было указано требуемое поведение системы, то процесс исправление дефекта будет замедлен.

На примере программного продукта «w3.aero United Airlines» были составлены следующий баг репорт:

Короткое описание: Невозможность открытия страницы при нажатии на ее вкладку

Проект: «w3.aero United Airlines»;

Компонент: <div class=”search”></div>;

Версия: commit fe2af2ae503b2b6a64b23df324b28e2ef1aaa68f

Серьезность: S3 Значительная;

Приоритет: P3 Низкий;

Статус: Открытый;

Автор: Молоткин Антон;

Назначен на: Марию Уткину;

Окружение: Операционная система Microsoft Windows 10, веб-браузер Yandex Browser 20.9.1.112;

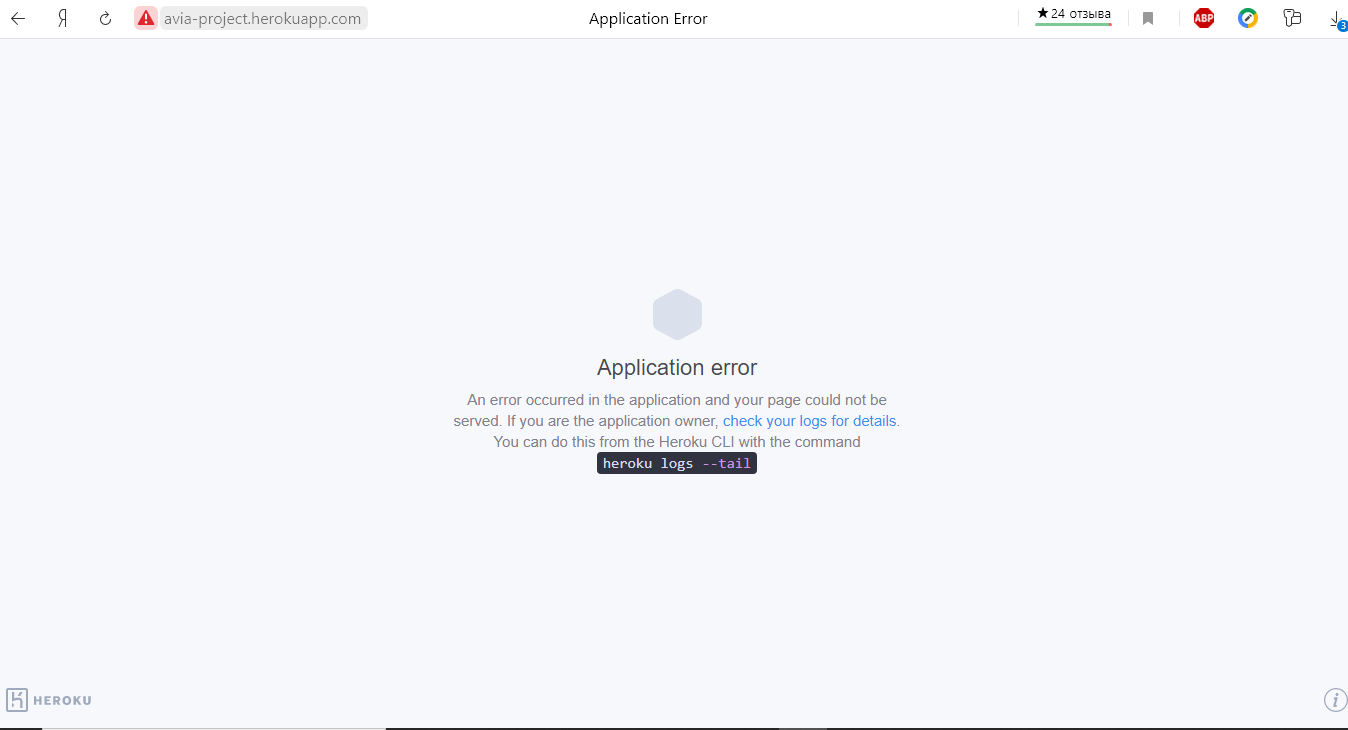
Шаги воспроизведения:

1. Открыть сайт «w3.aero United airlines»
2. Нажать на вкладку «Рейсы»

Фактический результат Показ ошибки приложения (application error);

Ожидаемый результат: Открытие страницы;

Дополнительные материалы:



Скриншот ошибки

# ЗАКЛЮЧЕНИЕ

В ходе учебной практики по ПМ.02 Разработка и администрирование базы данных

* проведена анализ предметной области;
* спроектирована информационная сиситма;
* разработана модель базы данных и приложений;
* создана база данных;
* создан макет приложения;

В процессе проектирования и разработки, выработаны умение и навыки за счет выполнения систематичесиких упражнений. Приобретены практические навыки работы в области создания, функционмрование и использования систем управления базами данных, а также комплексное освоение профессиональных компетеций. Разработана объкты базы данных. Реализована база данных в конкретной СУБД по специальности Программирование в компьютерных системах.

Также, были описаны возможности таск-менеджера Gulp и был создан проект с его использованием.

# СПИСОК ИСПОЛЬЗВАННЫХ ИСТОЧНИКОВ

1. Git и GitHub. Описание и история системы управления версиями [Электронный ресурс]. – Режим доступа: <https://tproger.ru/translations/difference-between-git-and-github/>.

2. Тестирование программного продукта. Описание видов тестирования [Электронный ресурс]. – Режим доступа: <http://codingcraft.ru/testing.php/>.

3. Негативное и позитивное тестирование. Описание видов тестирования [Электронный ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/positive-negative-testing/>.

4. Баг-репорт. Описание последовательности действий при поиске некорректностей в работе [Электронный ресурс]. – Режим доступа: http://www.protesting.ru/testing/bugreport.html.

6. Штрауб Б. Git для профессионального программиста / Штрауб Б, Чакон С. - СПб: Питер, 2019. – 494 с.

7. Таск-менеджер Gulp. Создание проекта. - https://zaurmag.ru/programmy-i-servisy/gulp.html