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** Галямин Данила Сергеевич **/ ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)**

**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

Галямин Данила Сергеевич

*ФИО*

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

успешно прошел производственную практику по профессиональному модулю   
**ПМ.02 «Разработка и администрирование базы 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](#_Toc44705804)

[1 Теоретические основы проекта 9](#_Toc44705805)

[1.1 Система контроля версий Git 9](#_Toc44705806)

[1.2 Языки программирования 9](#_Toc44705807)

[1.3 Среда разработки Node.js 12](#_Toc44705808)

[1.4 Вспомогательные технологии разработки 14](#_Toc44705809)

[2 Разработка программного продукта 14](#_Toc44705810)

[2.1 Разработка серверной части 14](#_Toc44705811)

[2.1.1 Установка Node.js 14](#_Toc44705812)

[2.1.2 Установка модулей 16](#_Toc44705813)

[2.1.3 Создание сервера приложения 17](#_Toc44705814)

[ЗАКЛЮЧЕНИЕ 21](#_Toc44705815)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 22](#_Toc44705816)

[ПРИЛОЖЕНИЯ 23](#_Toc44705817)

# ВВЕДЕНИЕ

Практика – одна из важнейших составляющих профессиональной подготовки студента и имеет целью закрепление и углубление знаний, полученных студентами в процессе теоретического обучения, приобретение необходимых умений, навыков и опыта работы по специальности, а также опыта работы. Студенты конкретизируют и закрепляют теоретические знания, приобретенные в процессе обучения, овладевают навыками практической работы, стараются применить полученные знания на практике.

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

Стержневые идеи современных информационных технологий базируются на концепции баз данных. Согласно этой концепции, основой информационных технологий являются данные, которые должны быть организованы в базы данных в целях адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей.

База данных (БД) — это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов. В ходе практики для разработки была выбрана СУБД MySQL. MySQL – свободная реляционная система управления базами данных с открытым исходным кодом с моделью клиент-сервер.

Данная производственная практика будет проходить на предприятии РКТ «МАИ».

Филиал «Ракетно-космическая техника» Московского авиационного института (национального исследовательского университета) расположен в г.Химки, Московской области. Филиал готовит специалистов со средним профессиональным образованием для ракетно-космического комплекса и других отраслей промышленности. Выпускники филиала успешно работают на предприятиях г.о. Химки, Москвы и Московской области.

Филиал создан по инициативе предприятий ФГУП «НПО им. С.А. Лавочкина, АО «НПО Энергомаш имени академика В.П. Глушко» и ОАО «МКБ «Факел» имени академика П.Д. Грушина».

Сегодня филиал «РКТ» МАИ предлагает:

* Перспективные и востребованные специальности
* Новейшие образовательные стандарты
* Коллектив преподавателей-профессионалов
* Современные методы обучения
* Отсрочка от призыва на военную службу для юношей
* Стипендии студентам бюджетных отделений
* Обучение по целевому набору
* Связь с ведущим предприятиями округа
* Помощь в трудоустройстве выпускникам
* Широкий спектр дополнительных образовательных услуг
* Разнообразная студенческая жизнь
* Получение диплома государственного образца

При наличии диплома филиала поступление в ВУЗ по результатам внутренних испытаний (без сдачи ЕГЭ)

Выпускники филиала имеют приоритетное право приема на работу на ведущие оборонные предприятия городского округа Химки.

# 1 Теоретические основы проекта

## 1.1 Система контроля версий Git

Git — распределённая система управления версиями. Система спроектирована как набор программ, специально разработанных с учётом их использования в сценариях.

GitHub — сервис онлайн-хостинга репозиториев, обладающий всеми функциями распределённого контроля версий и функциональностью управления исходным кодом — всё, что поддерживает Git и даже больше.

## 1.2 Язык программирования JavaScript

Язык программирования JavaScript

JavaScript - мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. JavaScript появился в 1995 году как способ программирования веб-страниц в браузере Netscape Navigator. С тех пор язык был принят во всех остальных ведущих графических браузерах. Это сделало возможным применение современных веб-приложений - приложений, с которыми можно взаимодействовать напрямую, не перезагружая страницу при каждом действии.

JavaScript позволяет реализовать те функции страницы, которые невозможно реализовать стандартными тегами HTML. Сценарии запускаются в результате наступления какого-нибудь события, например, нажатие на кнопку или изменение размера окна. JavaScript имеет доступ к свойствам документа и свойствам браузера.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкани — что придаёт языку дополнительную гибкость

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком Си имеет коренные отличия:

* объекты с возможностью интроспекции;
* функции как объекты первого класса;
* автоматическое приведение типов;
* автоматическая сборка мусора;
* анонимные функции.

В JavaScript доступ к отладчикам становится особенно полезным при разработке крупных нетривиальных программ из-за различий в реализациях разных браузеров (в частности, в отношении объектной модели документа). Полезно иметь доступ к отладчику для каждого из браузеров, в которых будет работать веб-приложение.

По состоянию на ноябрь 2009 года, Internet Explorer, Opera, Firefox, Safari, и Google Chrome имеют отладчики сценариев.

Internet Explorer имеет три отладчика: Microsoft Visual Studio — самый полный из них, за ним следует Microsoft Script Editor и наконец, свободный Microsoft Script Debugger, гораздо более простой, чем два других. Бесплатный Microsoft Visual Web Developer Express предоставляет ограниченную версию с отладочной функцией JavaScript в Microsoft Visual Studio.

JavaScript может быть подключен или внедрен в веб-документ тремя способами:

* Расположение внутри страницы между тегами <script></script> которые рекомендуется помещать внутри контейнера <head>. Количество скриптов в документе неограниченно;
* Расположение внутри тегов HTML;
* Вынесение в отдельный файл и его подключение с помощью <script type="application/javascript" src="http://Путь\_к\_файлу\_со\_скриптом"></script>.

## 1.3 Среда разработки Node.js

Node.js — это среда выполнения JavaScript, которая в основном используется для создания веб-приложений. Другими словами, это реализация JavaScript на стороне сервера, используемая для написания серверной части приложения. (Хотя многие платформы Node.js также могут работать с внешним интерфейсом.)

Node.js отлично подходит для масштабируемых высоконагруженных проектов. С его помощью также можно создавать синхронные кроссплатформенные решения. И чаще всего Node.js используется для создания веб-серверов.

Использование Node.js в Full-stack разработке (разработке полного цикла) позволяет писать и front-end — пользовательскую часть приложения, и back-end — серверную часть приложения на одном и том же языке программирования JavaScript. Это удобно для связывания частей приложения. Также это позволяет front-end разработчикам достаточно освоить и back-end разработку.

Преимущества NodeJS:

* Простой и широко известный JavaScript. Конечно, платформа предполагает собственные инструменты и особенности, например, здесь нет браузерных API, coockie или DOM, зато присутствуют собственные библиотеки и другие интересные решения. Но в основном используются возможности и синтаксис всем привычного JavaScript.
* Богатая стандартная библиотека. Платформа изначально обладала широким набором возможностей, а в новых версиях библиотека пополняется и улучшается.
* Огромное внешних библиотек и готовых модулей. Использование пакетного менеджера NPM позволяет постоянно развивать экосистему Node. Сегодня число опенсорсных пакетов в нем перевалило за цифру 500 тысяч и постоянно растет.
* Удобство для построения быстрых приложений, поскольку Node способен обрабатывать огромное количество одновременных соединений с высокой пропускной способностью. Это привлекает множество программистов, которые хотят создавать сайты с поддержкой миллионов посетителей;
* Эффективность. В веб-приложении больше всего времени требуется, чтобы выполнить все ваши запросы к базе данных. С помощью Node.js вы можете выполнить все их сразу, уменьшая время отклика для медленных запросов;

Ниже приведено несколько примеров того, что можно создать с помощью Node.js.

* Одностраничные приложения. Это веб-приложения, которые работают в браузере и не требуют повторной загрузки страницы при каждом ее использовании для получения новых данных. Некоторые примеры одностраничных приложений включают в себя приложения социальных сетей, электронную почту, приложения карт, текст в Интернете, инструменты для рисования и т. д.
* Приложения в режиме реального времени. Это веб-приложения, которые способствуют пользователям получать информацию сразу после публикации автором, а не требовать периодических проверок источника на наличие обновлений.
* Приложения потоковой передачи данных. Это приложения (или службы), которые отправляют данные или содержимое по мере их поступления (или создания), сохраняя подключение для продолжения загрузки дополнительных данных, содержимого или компонентов по мере необходимости. К некоторым примерам относятся приложения потоковой передачи видео и аудио.
* API REST. Это интерфейсы, которые предоставляют данные для взаимодействия с веб-приложением другого пользователя. Например, служба API календаря может предоставлять даты и время для концертного помещения, которое может использоваться веб-сайтом локальных событий другого пользователя.
* Готовые к просмотру приложения на стороне сервера (SSR). Эти веб-приложения могут работать как на клиенте, так и на сервере, позволяя динамически отображать любое известное содержимое и быстро перехватывать неизвестное по мере его доступности.
* Средства командной строки. Они позволяют автоматизировать повторяющиеся задачи, а затем распространить ваше средство по обширной экосистеме Node.js.
* Программирование оборудования. Хотя Node.js не так популярен, как веб-приложения, в настоящее время Node.js пользуется все большей популярностью для сбора данных с датчиков, маяков, передатчиков, двигателей или всего, что генерирует большие объемы данных. Node.js может обеспечить сбор данных, анализ этих данных, обмен данными между устройством и сервером и принятие мер на основе анализа.

## 1.4 Вспомогательные технологии разработки

MySQL – это система управления реляционными базами данных, с открытым исходным кодом, созданная компанией MySQL AB в 1995 году.

База данных представляет собой структурированную совокупность данных, которые могут представлять из себя списки, перечни документов и другого большого количества информации. В реляционных базах данных содержимое хранится не в одном скопе, а в отдельных таблицах. Таблицы связываются между собой с помощью отношений, что позволяет создавать запросы, состоящие из нескольких таблиц. Программное обеспечение MySQL является системой клиент-сервер, содержащая в себе многопоточный SQL-сервер, обеспечивающий поддержку вычислительных машин баз данных, клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов.

Основные характеристики MySQL:

* Разработан на языке программирования С++;
* Кроссплатформенный;
* Многопоточный за счет использования потоков ядра процессора;
* Быстрая работа с таблицами;
* Быстрое распределение памяти;
* Быстрые соединения между таблицами (с использованием метода one-sweep multi-join);
* Быстрые SQL-функции, реализованные с помощью оптимизированных библиотек классов;
* Доступность использования хеш-таблиц в памяти компьютер, в качестве запасных таблиц [4].

# 2 Разработка программного продукта

## 2.1 Разработка серверной части

### 2.1.1 Установка Node.js

Для загрузки необходимо перейти на официальный сайт https://nodejs.org/en/. На главной странице представлены сразу две возможные опции для загрузки: самая последняя версия NodeJS и LTS-версия. Страница загрузка представлена на рисунке 1.

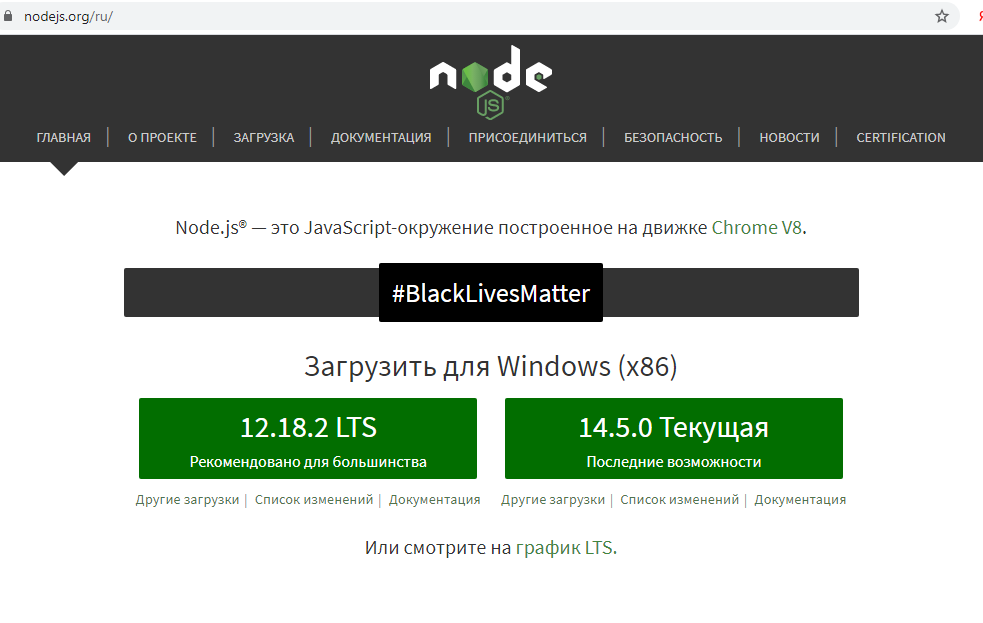


Рисунок 1 – Страница загрузки Node.js

При открытии скачанного файла запускается программа установшик.

После успешной установки можно ввести в командной строке/терминале команду node -v, и в ней отобразится текущая версия node.js, что представлено на рисунке 2.



Рисунок 2 – Проверка текущей версии Node.js

Для разработки под Node JS достаточно простейшего текстового редактора, в частности, Notepad++. Также можно использовать более изощренные редакторы типа Atom, Sublime, Visual Studio Code, либо среды разработки, которые поддерживают работу с Node.JS, например, Visual Studio или WebStorm.

Для запуска файла с кодом необходимо необходимо в консоли/терминале перейти в каталог, содержащий данный файл, с помощью команды cd. Затем необходимо выполнить команду node имя\_файла.js. При успешном запуске в консоли будет представлен результат работы программы.

### 2.1.2 Установка модулей

Для автоматизации установки и обновления пакетов, как правило, применяются систему управления пакетами или менеджеры. Непосредственно в Node.js для этой цели используется пакетный менеджер NPM (Node Package Manager). NPM по умолчанию устанавливается вместе с Node.js, поэтому ничего доустанавливать не требуется. Но можно обновить установленную версию до самой последней. Для этого в командной строке/терминале надо запустить команду npm install npm@latest –g

Чтобы узнать текущую версию npm, в командной строке/терминале надо ввести команду npm –v.

Для более удобного управления конфигурацией и пакетами приложения в npm применяется файл конфигурации package.json. Так в папку проекта добавляется файл package.json, например, с таким содержимым:

{

"name": "modulesapp",

"version": "1.0.0"

}

Здесь определены только две секции: имя проекта - modulesapp и его версия - 1.0.0. Это минимально необходимое определение файла package.json. Данный файл может включать гораздо больше секций.

Для установки модуля, например Express, в проект используется команда npm install express.

После установки express в папке проекта появится подпапка node\_modules, в которой будут хранится все установленные внешние модули. В частности, в подкаталоге node\_modules/express будут располагаться файлы фреймворка Express.

И после выполнения команды, если мы откроем файл package.json, то мы увидим информацию о пакете:

{

"name": "modulesapp",

"version": "1.0.0",

"dependencies": {

"express": "^4.17.1"

}

}

Информация обо всех добавляемых пакетах, которые используются при работе приложения, добавляется в секцию dependencies.

Для загрузки модулей применяется в коде используется функция require(), в которую передается название модуля. К примеру, для использования модуля express необходимо ввести const express = require("express");

### 2.1.3 Шаблон проектирования MVC

Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

* Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
* Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

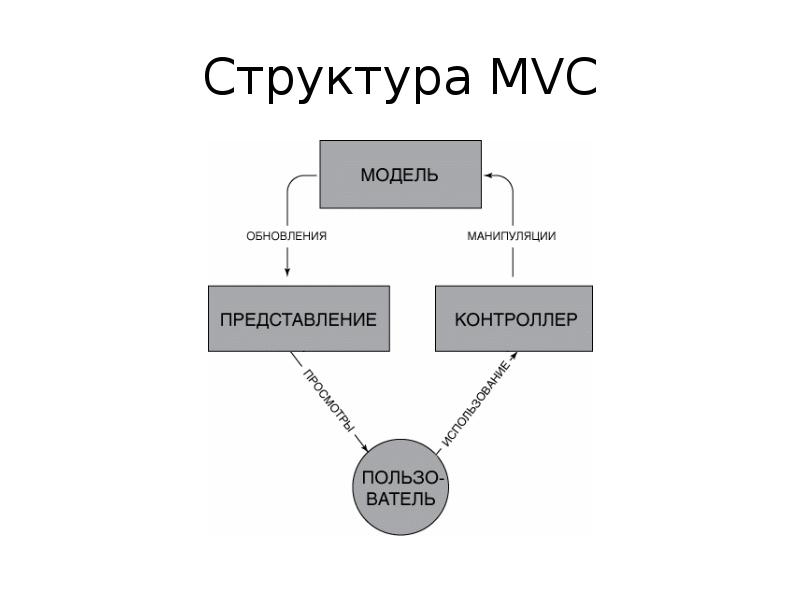


Рисунок 3 – Схема MVC-паттерна

За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

* К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;
* Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) — для этого достаточно использовать другой контроллер;
* Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

Концепция MVC позволяет разделить модель, представление и контроллер на три отдельных компонента:

* Модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность. Модель не зависит от представления (не знает как данные визуализировать) и контроллера (не имеет точек взаимодействия с пользователем) , просто предоставляя доступ к данным и управлению ими. Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом может быть встроено уведомление «наблюдателей». Модель, за счёт независимости от визуального представления, может иметь несколько различных представлений для одной «модели».
* Вид обеспечивает различные способы представления данных, которые получены из модели. Он может быть шаблоном, который заполняется данными. Может быть несколько различных видов, и контроллер выбирает, какой подходит наилучшим образом для текущей ситуации.
* Контроллер обеспечивает «связь» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.

Самое очевидное преимущество, получаемое от использования концепции MVC — это чёткое разделение логики представления (интерфейса пользователя) и логики приложения.

Поддержка различных типов пользователей, которые используют различные типы устройств является общей проблемой наших дней. Предоставляемый интерфейс должен различаться, если запрос приходит с персонального компьютера или с мобильного телефона. Модель возвращает одинаковые данные, единственное различие заключается в том, что контроллер выбирает различные виды для вывода данных.

Помимо изолирования видов от логики приложения, концепция MVC существенно уменьшает сложность больших приложений. Код получается гораздо более структурированным, и, тем самым, облегчается поддержка, тестирование и повторное использование решений.

Стоит также отметить, что реализация паттерна MVC может отличаться в зависимости от задачи. Например, в веб-разработке модель и вид взаимодействуют друг с другом через контроллер, а в приложениях модель может сама уведомлять вид, что нужно что-то изменить.

### 2.1.4 Создание сервера приложения

При разработке сервера был выбран фреймворк Express. Express.js — это самый простой и быстрый фреймворк Node.js, который используется в качестве промежуточного обработчика для управления серверами и маршрутами.

Express.js подходит для разработки простых приложений, которые могут обрабатывать несколько запросов одновременно и опираются на возможности технологии Express.

Express предоставляет ряд готовых абстракций, которые упрощают создание сервера и серверной логики, в частности, обработка отправленных форм, работа с куками, CORS и так далее.

Код созданного сервера изображен на рисунке 4.



Рисунок 4 – Созданный сервер приложения

Все необходимые модули подключаются с помощью метода require(). Для использования Express в начале необходимо создать объект, который будет представлять приложение. Для приложения устанавливается шаблонный процессор handlebars, указывается папка, содержащая статичные файлы, и парсер данных. Далее создается объект шаблонного процессора и указывается папка с частичными представлениями (блоки кода, которые можно использовать много раз в разных частях приложения). Затем создается объект системы маршрутизации, которой передается объект приложени и сервер начинает прослушивание по указанному порту и выводит в консоли сообщение о том, что сервер запущен.

На рисунке 5 изображен код файла подключения к базе данных MySQL.

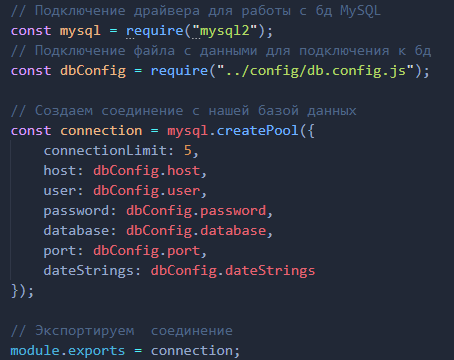


Рисунок 5 – Подключение к базе данных MySQL

Для подключения к базе данных MySQL используется драйвер mysql2. Для создания подключения используется метод createConnection() или createPool(), которые в качестве параметра принимают объект с свойствами, содержащими необходимые данные для успешного соединения. Запросы к базе данных осуществляются с помощью метода query(sqlString, callback) или execute(sqlString, callback), где sqlString - выполняемая SQL-команда, а callback - функция обратного вызова, через параметры которой мы можем получить результаты выполнения SQL-команды или возникшую ошибку.

На рисунке 6 изображен код системы маршрутизации.



Рисунок 6 – Система маршрутизации

Каждому возможному маршруту назначается обработчик в соответствии с методом, по котомору его можно достичь. В качестве таких методов можно использовать ряд функций, в частности:

* use
* get
* post
* put
* delete

Когда приходит запрос Express сопоставляет запрошенный адрес с каждым из маршрутов. Затем выбирается первый совпавший маршрут. При совпадении маршрута вызывается его функция обработчика.

На рисунке 7 представлен пример одного из таких обработчиков.

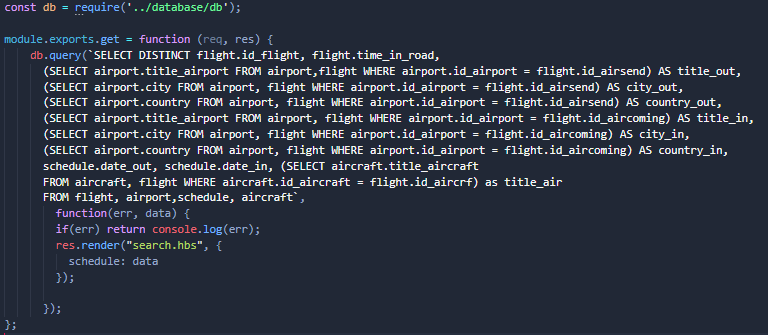


Рисунок 7 – контроллер searchController

Каждый файл представляет собой отдельный контроллер, который содержит набор функций-обработчиков маршрутов.

Контроллеры подключаются как стандартные модули, и затем функции контроллеров используются для обработки маршрутов. Кроме того, для упрощения логической организации маршруты, которые обрабатываются контроллером одним контроллером, могут быть вынесены в отдельный роутер.

# 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;
* разработано веб-приложение.

Также был получен опыт разработки в команде. Практика показываает умение работать в команде и развивать свои коммуникабельные способности.

Была изучена работа системы контроля версий Git и работа с веб-сервисом GitHub, основанном на Git и позволяющем размещать собственные проекты и осуществлять совместную разработку.

Также в ходе практики были изучены и освоены принципы работы со средой выполнения Node.js и фреймворком Express, были изучены основы разработки современных веб-приложений.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1.Git и Github: что это такое и чем отличается [Электронный ресурс]/ https://tproger.ru/translations/difference-between-git-and-github/- статья в интернете.

2. Макфарланд Д. Большая книга CSS. 2-е изд.- СПб.: Питер, 2012. — 560 с.

3. Веб-программировании [Электронный ресурс]/ https://webstudio2u.net/ru/programming/183-what-is-denwer.html- статья в интернете.

4. Приведение MySQL в сеть [Электронный ресурс]/ https://www.phpmyadmin.net/- статья в интернете.