Для разработки серверной части проекта была выбрана среда Node.js c фреймворком Express.js. Nodee.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.) 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.

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 1.



Рисунок 1 – Созданный сервер приложения

Все модули подключения с помощью метода require(). Для использования Express в начале надо 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 подключения к базе данных MySQL и объект системы маршрутизации, которой передается объекты приложения и подключения к базе данных. Затем сервер начинается прослушивание по указанному порту и выводит в консоли сообщение о том, что сервер запущен.

На рисунке 2 изображен код файла подключения к базе данных MySQL.

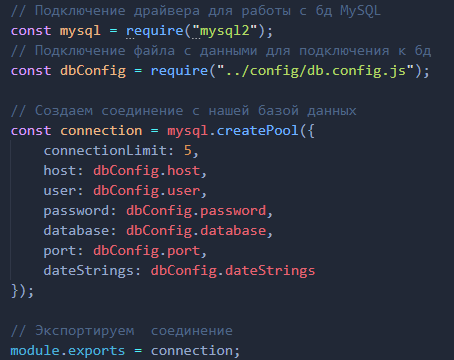


Рисунок 2 – Подключение к базе данных MySQL

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 3 изображен код системы маршрутизации.

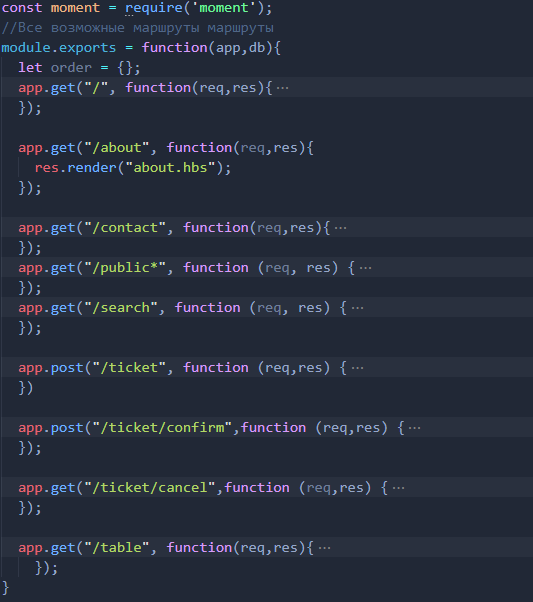


Рисунок 3 – Система маршрутизации

При переходе по одному из таких маршрутов происходит выполнение каких-либо действий или запросов и у клиента отображается соответствующая страница с помощью метода render(), в параметры которого передается строковый аргумент с названием файла страницы, а во второй параметр могут передаваться объекты, которые необходимо использовать на страниц, или происходит переадресация на другой маршрут. Для обработки данных по определенному маршруту можно использовать ряд функций, в частности:

* use
* get
* post
* put
* delete

В качестве первого параметра эти функции могут принимать шаблон адреса, запрос по которому будет обрабатываться. Второй параметр функций представляет функцию, которая будет обрабатывать запрос по совпавшему с шаблоном адресу. Когда приходит запрос Express сопоставляет запрошенный адрес с каждым из маршрутов. Затем выбирается первый совпавший маршрут. При совпадении маршрута вызывается его функция обработчика.