МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра дискретной математики и информационных технологий

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ И БАЗЫ ДАННЫХ ДЛЯ СИСТЕМЫ ОЦЕНИВАНИЯ И ОБРАТНОЙ СВЯЗИ ВНУТРЕННЕЙ ЛАБОРАТОРИИ УЧЕБНОГО ЦЕНТРА EPAM SYSTEMS SARATOV

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 271 группы
направления 09.04.01 — Информатика и вычислительная техника
факультета КНиИТ
Войтенко Сергея Георгиевича

Научный руководитель
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Заведующий кафедрой
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Саратов 2019

ВВЕДЕНИЕ

Современное общество нельзя представить без технологий. В любой сфере жизни используется вычислительная техника различных мощностей и программное обеспечение для взаимодействия с ней. Написанием данного программного обеспечения занимаются специально обученные люди. Компания может нанимать их в свой штат работников, а может обращаться к компаниям, которые специализируются на написании и поддержании работоспособности программного обеспечения, внесении в него обновлений, направленных на исправление ошибок или добавление нового функционала.

В нашей стране есть вузы, которые занимаются подготовкой данного рода специалистов по различным направлениям. Однако вуз занимается подготовкой базовых навыков программиста, в различных сферах его деятельности, на основе которых студент может выбрать более близкое ему направление развития и дальнейшего трудоустройства. Но именно здесь возникают определенные трудности. Довольно часто выпускник, а возможно еще студент, не имеет опыта для выполнения реальных задач. Тут на помощь приходят учебные курсы, организуемые компаниями, которые позволяют получить дополнительные навыки: работа в команде, выполнение заданий, близких к реальным. При этом ведется контроль успеваемости обучающихся на данных курсах, чтоб предложить дальнейшее трудоустройство тем, кто их успешно прошел.

Передо мной была поставлена задача, разработки и реализации базы данных и доступа к ней для такого рода системы внутренней лаборатории учебного центра EPAM Systems Saratov. Система будет представлять собой веб-приложение, в котором студенты будут получать задания, требуемые для выполнения в процессе прохождения обучения, и видеть оценки за выполненные задания, получать комментарии и советы от руководителей по успеваемости.

Моя задача заключается в проектировании базы данных и последующий ее разработке. Так же разработка серверной части веб приложения организующей доступа к базе данных и передачу извлекаемой информации в клиентскую часть приложения.

Особенностью данной работы является использование комбинации не самых популярных, но наиболее сильных решений Node.JS и MongoDB. Так-

же описывается исследование преимуществ разработки серверной части веб приложения с использование Node.JS.

Магистерская работа состоит из введения, 3 разделов, заключения, списка использованных источников и 2 приложений. Общий объем работы – 64 страницы, из них 51 страница – основное содержание, включая 9 рисунков, 12 страниц приложений, список использованных источников информации – 24 наименования.

1 КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Используемые технологии» посвящен основным технология использованным при разработке и особенностям, таким как:

— MongoDB — документоориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц.Классифицирована как NoSQL, написана на языке C++. MongoDB хранит данные в гибких, похожих на JSON, документах, а это означает, что поля могут варьироваться от документа к документу, а структура данных может изменяться со временем. Это очень удобно вы работаете над новым проектом, бизнес-модель которого еще не до конца ясна и с большой вероятностью проект до финального релиза претерпит множество изменений, в том числе на уровне организации данных.

— JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMA Script. JavaScript обычно используется, как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

— Node.JS — программная платформа, основанная на движке V8, разработанном google для браузера chrome, который транслирует JavaScript в машинный код, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.JS применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.JS и десктопные оконные приложения и даже программировать микроконтроллеры. В основе Node.JS лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

— NPM (node package manager) — это стандартный менеджер пакетов, автоматически устанавливающийся вместе с Node.js. Он используется для скачивания пакетов из облачного сервера npm, либо для загрузки па-

кетов на эти сервера. Крупнейший в мире реестр программного обеспечения. Разработчики с открытым исходным кодом со всех континентов используют npm для обмена и заимствования пакетов, а многие организации также используют npm для управления частной разработкой. Использовать npm можно через веб-сайт, интерфейс командной строки (CLI), реестр.

— Mongoose - это Object Document Mapper - объектно-документный отобразитель. Это означает, что Mongoose позволяет вам определять объекты со строго-типизированной схемой, соответствующей документу MongoDB.
— Nodemailer - это модуль для приложений Node.js, позволяющий легко отправлять электронные письма. Разработка данного модуля началась в 2010 году, когда не было других модулей реализующих данный функционал. Сегодня это одно из самых популярных решений, используемых для отправки писем из приложений, написанных на Node.JS.
— Express.js, или просто Express, фреймворк web-приложений для Node.JS, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб-приложений и API. Де-факто является стандартным каркасом для Node.JS.

Второй раздел «Выбор эффективного инструментария для разработки приложения» посвящен сравнению быстродействия работы серверного приложения написанного на разных популярных технологиях.

Было произведено тестирование систем, чтобы определить лучший инструментарий для построения сервисной части приложения. В связи с тем, что приложение подразумевает активную работу с базой данных и важна скорость, с которой будут обновляться данные в базе и на страницах пользователей, для личного тестирования были выбраны Node.JS, Java Spring и PHP Lumen. Использовался инструментарий wrk и ab HTTP Apache. Тестовое приложение должно было на основании полученного id курса вернуть информацию о нем, список студентов, обучающихся по этому курсу, их оценки и дополнительную информацию о них. По поставленной задаче было написано 3 версии предложения на разном инструментарии, реализующих данный функционал.

Получившиеся результаты показали, что больше всего обработанных запросов в секунду у Node.JS – 453, у Java Spring – 427, а результат PHP

Lumen всего 197.

Третий раздел «Разработка базы данных и back-end части приложения» посвящен разработке серверной части приложения и базы данных для большой системы оценивания и мониторинга успехов студентов внутренней лаборатории EPAM System Saratov, которая будет связываться с клиентской частью системы и обрабатывать запросы.

На первом этапе разработки была спроектирована базы данных MongoDB с помощью модуля Mongoose. Схема получившейся базы показана на рисунке 1.

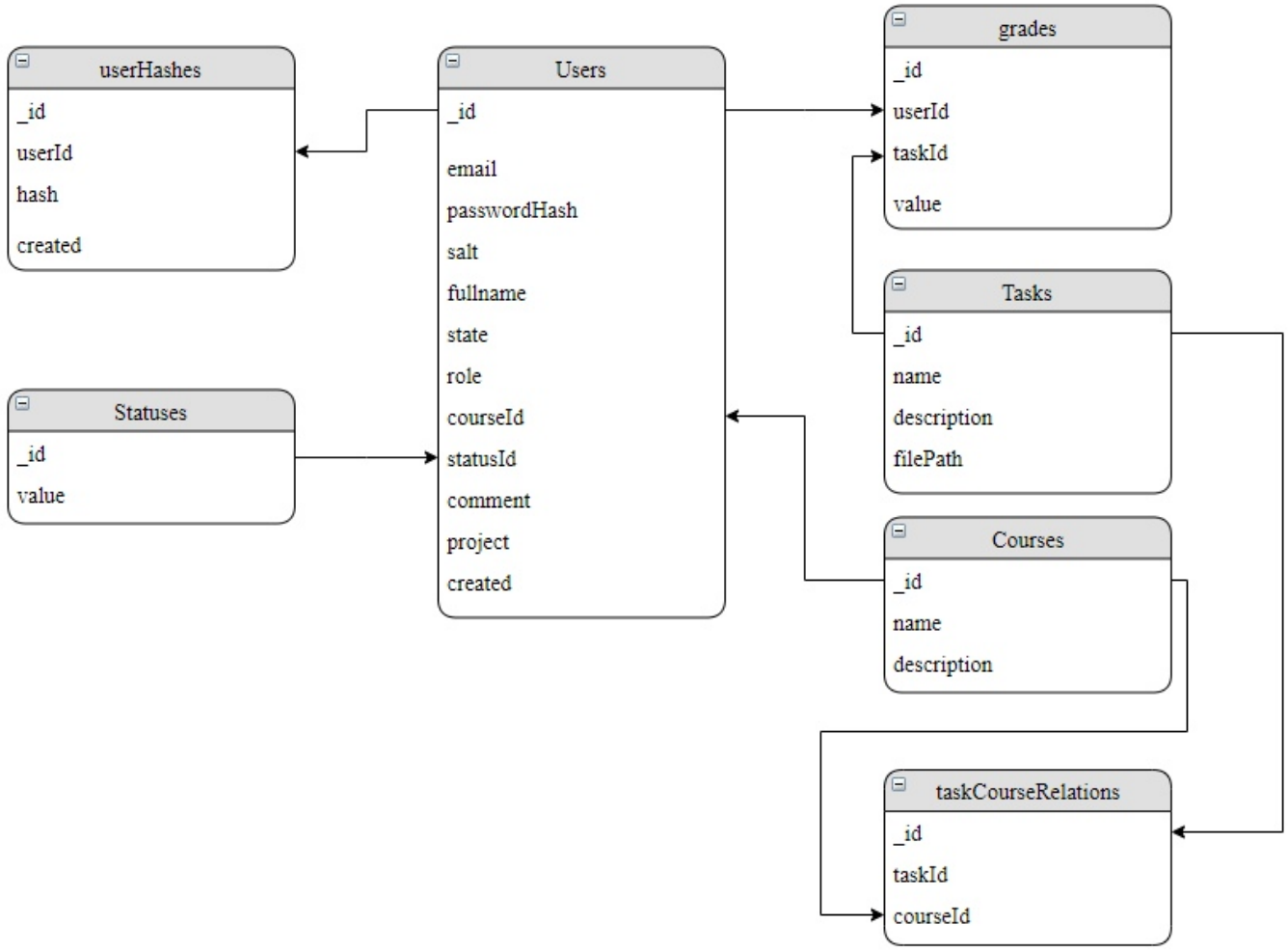

Рисунок 1 – Схема получившейся базы данных.

Следующим этапом после создания схем, и соответствующих коллекций в базе данных, является написание back-end части приложения с API. Application programming interface – программный интерфейс приложения – это описание способов (набор классов, процедур, функций, структур или кон-

стант), которыми одна компьютерная программа может взаимодействовать с другой программой.

В папке controllers находятся все модули, которые производят взаимодействия с базой данных, API. В models хранятся схемы описывающие коллекции, хранящиеся в базе данных. Дополнительные модули хранятся в папке services.

Определение того какой именно модуль должен запускаться в определенный момент производится с помощью router. Он производит поиск по запрошенному URL и типу запроса, передает выполнение в соответствующий модуль. При поступлении некого запроса, router ищет маршрут среди прописанных в приложении, и передает обработку на первый совпавший. Если ни один из маршрутов не совпал – возвращается соответствующая ошибка.

Как и на любом сайте, предоставляющему доступ к различным данным, будь то информации о пользователе, его оценках, или о курсе, заданиях этого курса, необходимо реализовать процесс регистрации пользователей и последующий вход, для разграничения доступа к информации и функционалу приложения в соответствии с ролью пользователя. А именно реализовать процессы аутентификации, авторизации и идентификации. Давнный функционал содержится в модуле auth-controller. В нем реализованы механизмы создания нового пользователя, входа в систему для уже существующего пользователя, выхода с нее, и верификации, проверка на подлинность, зарегистрированного пользователя.

На front-end части приложения пользователь вводит необходимые данные, после проверки они пересылаются на back-end часть приложения для создания пользователя и сохранения его в базе данных. Производится проверка на то, что пользователь с данной электронной почтой еще не зарегистрирован. После создания записи о пользователе в базе производится запуск функции организующей генерацию хеша пользователя, который необходим ему для подтверждения аккаунта и отправка письма на электронную почту.

Пред отправкой сообщения производится вызов двух других функций из модуля hasher.js. CreateUserHash производит генерацию пользовательского хеша на основе ключа и электронной почты пользователя. Далее данный хеш сохраняется в базе данных с id пользователя, которому он принадлежит.

При переходе пользователем по ссылке для верификации, запускает-

ся функция, которая по полученному значению хеша ищет в базе данных пользователя, которому он принадлежит. Если в базе данных нет записи с таким хешем, значит, либо пользователь уже был активирован, либо такого пользователя вовсе не существует. Если запись была найдена, то по содержащемуся в ней id пользователя, происходит его активация, смена состояния state со значение inactive на active. Если пользователь не был активирован он не получит доступа к функционалу веб-приложения.

После того как пользователь прошел активацию, он может войти на сайт. Когда необходимые авторизации данные будут введены корректно, они будут на серверную часть приложения, в которой router передаст их в назначенный обработчик.

Функцией производится поиск пользователя с соответствующим адресом электронной почты, если его нет, тогда возвращается ошибка сообщающая, что такого пользователя нет. Если пользователь был найден, проверяется его состояние – является ли он активированным. В случае успеха, производится проверка правильность введенного пароля, путем хеширование его и сравнения с хранящимся в базе. Если пользователь не активирован или был введен неправильный пароль, возвращаются соответствующие ошибки. В случае успешной авторизации создается сессия с информации пользователя, а именно его id, которая храниться в базе и передается на front-end часть приложения. Сессия храниться в базе до того как она будет удалена, что происходит при нажатии пользователем Logout на сайте, либо до того момент как истечет «время ее жизни» и она будет автоматически удалена системой.

Схема получившегося приложения представлена на рисунке 2.

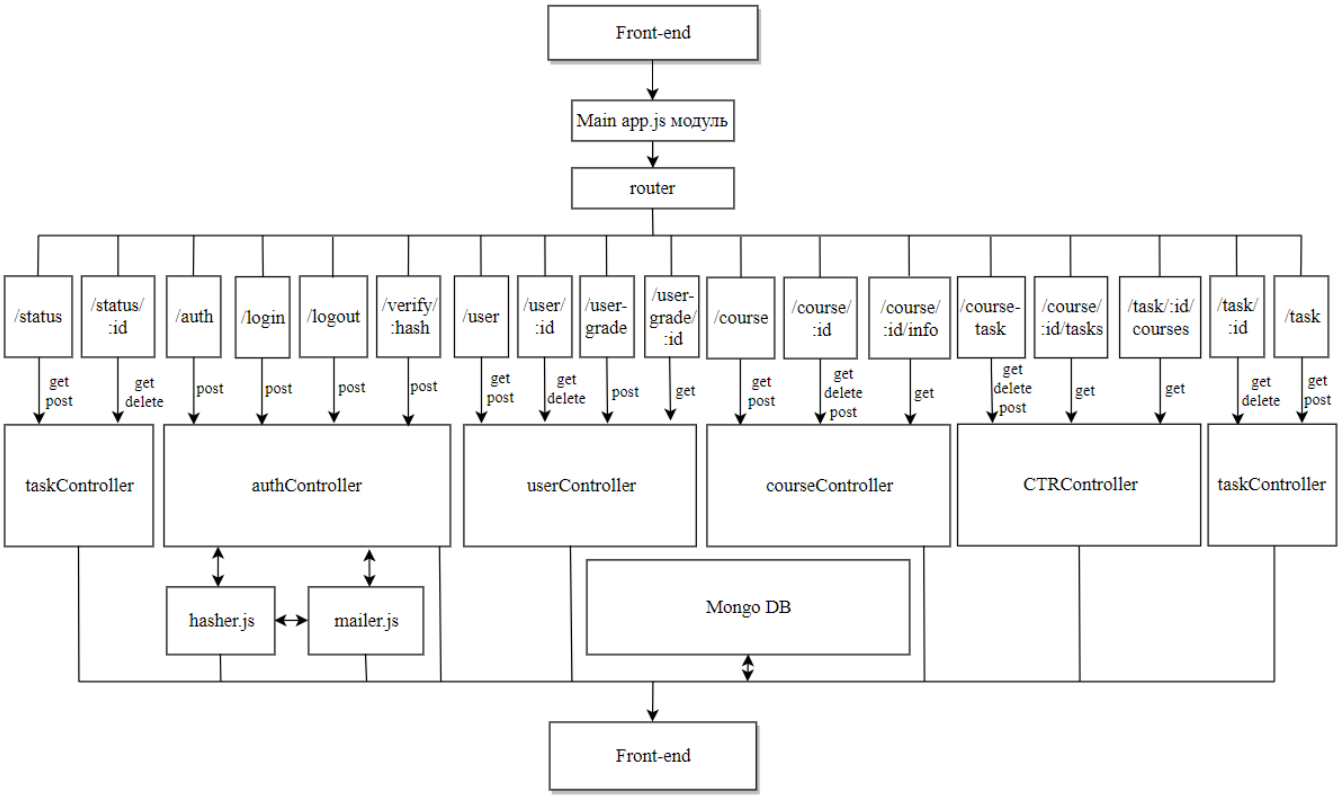

Рисунок 2 – Схема серврной части приложения.

ЗАКЛЮЧЕНИЕ

В данной работе мною были рассмотрены и использованы инструменты для разработки такие как: нереляционная база данных MongoDB, высокоуровневый язык программирования JavaScript, программная платформа Node.JS, менеджер пакетов npm, библиотека для работы с базой данный Mongoose, framework для создания веб-приложения Express, модуль Nodemailer, программа для работы с API Postman, программа с графическим интерфейсом для работы с базой Robo 3T.

Проведено исследование и сравнение различных платформ для разработки серверной части веб приложений, таких как Java с framework Spring и PHP + Lumen. Были написаны 3 тестовых приложения, реализующих обращение к базе данный и выборку из нее, для тестировании с помощью wrk и ab HTTP Apache. Node.JS показал наилучшие результаты, и был выбран для разработки приложения.

В результате проделанной работы, поставленная передо мною задача разработки и реализации базы данных для системы оценивания и back-tracking внутренней лаборатории учебного центра EPAM Systems Saratov, а также back-end приложения для организации обработки запросов и передачи данных на клиентскую часть веб-приложения была выполнена. Было произведено тестирование работоспособности данной системы. Приложение имеет модульную архитектуру и разработано в соответствии с современными паттернами и стандартами объектно-ориентированного программирования, в частности, web-программирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 SQL или NoSQL — вот в чём вопрос [Электронный ресурс]. — URL: https://habr.com/company/ruvds/blog/324936/ (Дата обращения 28.05.2019).

2 Современный учебник Javascript: Введение в JavaScript [Электронный ресурс]. — URL: https://learn.javascript.ru/intro (Дата обращения 28.05.2019).

3 Язык программирования JavaScript: информация для начинающих [Электронный ресурс]. — URL: https://www.internet-technologies.ru/articles/yazyk-programmirovaniya-javascript-informaciya-dlya-nachinayuschih.html (Дата обращения 28.05.2019).

4 Скринкаст по Node.js [Электронный ресурс]. — URL: http://learn.javascript.ru/screencast/nodejs (Дата обращения 28.05.2019).

5 Async/await: 6 причин забыть о промисах [Электронный ресурс]. — URL: https://habr.com/ru/company/ruvds/blog/326074/ (Дата обращения 28.05.2019).

6 Web REST API Benchmark on a Real Life Application [Электронный ресурс]. — URL: https://medium.com/@mihaigeorge.c/web-rest-api-benchmark-on-a-real-life-application-ebb743a5d7a3 (Дата обращения 28.05.2019).

7 *Иванов, А. Д.* Разработка и реализация web-приложения для оценивания студентов и предоставления им материалов для обучения внутренней лаборатории учебного центра EPAM Systems Saratov. / А. Д. Иванов. — 2019.

8 *Zakas, N. C.* Professional JavaScript for Web Developers / N. C. Zakas. — 3rd edition. — Birmingham, UK, UK: Wrox Press Ltd., 2012.

9 *Bass, L.* Software Architecture in Practice / L. Bass, P. Clements, R. Kazman. — 3rd edition. — Addison-Wesley Professional, 2012.

10 *Стефанов, .* JavaScript: шаблоны : [пер. с англ.] / . Стефанов. — 2011.

11 *Zakas, N. C.* Understanding ECMAScript 6: The Definitive Guide for JavaScript Developers / N. C. Zakas. — 1st edition. — San Francisco, CA, USA: No Starch Press, 2016.

12 ECMASript 2015 Language Specification [Электронный ресурс]. — URL: `https://wiki.diphost.ru/Authentication/` (Дата обращения 28.05.2019).