

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий  
Кафедра Информационных систем и технологий  
Специальность 1-40 01 01 «Программное обеспечение информационных технологий»  
Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

WEB-приложение «IT курсы»

Выполнил студент Почиковская Юлия Сергеевна  
(Ф.И.О.)  
Руководитель проекта асс. Дубовик М.В.  
(учен. степень, звание, должность, подпись, Ф.И.О.)  
Заведующий кафедрой к.т.н., доц. Смелов В.В.  
(учен. степень, звание, должность, подпись, Ф.И.О.)  
Консультанты асс. Дубовик М.В.  
(учен. степень, звание, должность, подпись, Ф.И.О.)  
Нормоконтролер асс. Дубовик М.В.  
(учен. степень, звание, должность, подпись, Ф.И.О.)  
Курсовой проект защищен с оценкой \_\_\_\_\_

Минск 2022

## Содержание

Введение .....	5
1 Постановка задачи .....	6
1.1 Обзор прототипов .....	6
1.2 Постановка задачи .....	8
1.3 Описание используемых технологий .....	8
2 Проектирование приложения .....	10
2.1 Проектирование модели базы данных.....	10
2.2 Проектирование структуры web-приложения .....	10
3 Разработка приложения.....	12
3.1 Реализация модели базы данных.....	12
3.2 Проектирование структуры сервера .....	12
3.3 Проектирование структуры клиента.....	14
4 Тестирование .....	15
4.1 Тестирование формы логина .....	15
4.2 Тестирование работы WebSockets .....	16
4.3 Тестирование работы поиска курсов .....	16
4.4 Тестирование добавления новых данных.....	16
5 Руководство пользователя .....	17
Заключение .....	20
Список литературных источников .....	21
Приложение А .....	22
Приложение Б.....	23
Приложение В .....	24
Приложение Г.....	26
Приложение Д .....	27
Приложение Е.....	28

## Введение

В последнее время все более популярным становится онлайн обучение. Многие IT-компании, предоставляют свободное обучение своим сотрудникам, имеют свою внутреннюю закрытую систему, в которой представлена информация о тренингах, которые может предложить компания. Также существуют независимые сервисы, находящиеся в свободном доступе, предназначенные для предоставления доступа к тренингам всех желающим на платной или бесплатной основе.

А в связи с событиями последних лет в мире, онлайн обучение становится актуальным не только для обычных пользователей, но и для университетов, и школ.

Преимущества, которые предоставляет такой вид обучения способствует развитию новых веб-приложений.

Преимущества

- система работает 24 часа в сутки, 365 дней в году, без перерыва на обед, без выходных и праздничных дней;

- доступ к тренингам может получить любой пользователь, находящийся в любой точке планеты на любом континенте;

- срок и стоимость создания системы несоизмеримо ниже, чем ведение документации и учета всей информации вручную.

Целью моего курсового проекта является создание web-приложения для системы управления курсами, которое предоставляет возможность просмотра информации о курсах, и о тренерах, которые эти курсы ведут.

Основными задачами курсовой работы являются:

- провести анализ прототипов и литературных источников;
- разработать модель базы данных;
- разработать web-приложение с функциональностью, соответствующей теме курсового проекта;

- провести тестирование разработанного web-приложения;

- написать руководство пользователя.

Приложение должно быть разработано с помощью языка JavaScript и программной платформы Node.js, реализуя архитектуру MVC.

Для проектирования базы данных используется система управления базами данных MySQL.

# 1 Постановка задачи

## 1.1 Обзор прототипов

Рассмотрим примеры web-приложений, на которых можно получить информацию о тренингах.

Skillbox [1] – онлайн-университет, в котором обучают 90+ программам для получения востребованных профессий во всем земном шаре. И всё это в режиме online. Страница сайта представлена на рисунке 1.1.

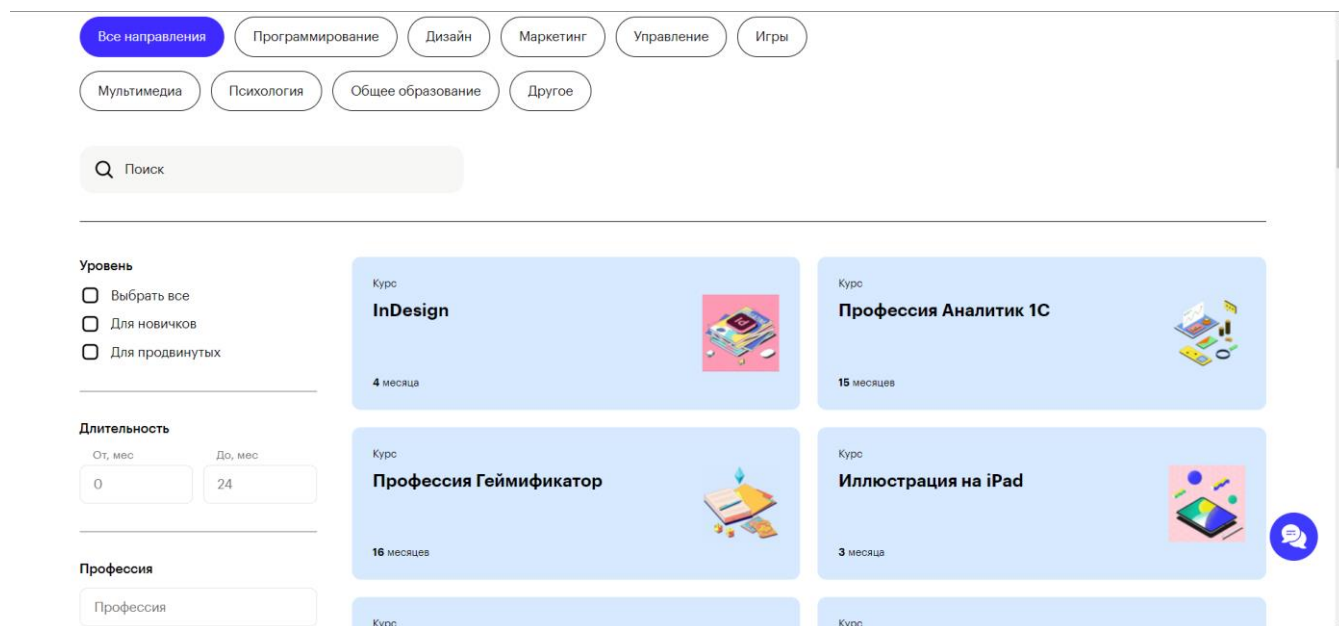


Рисунок 1.1 – Онлайн-университет Skillbox

Здесь обучают профессиям, которые будут всегда востребованы в мире IT. Если осваивать программы больше года, можно не только получить диплом, но и наработать достойное портфолио, составить CV и устроиться на работу ещё ДО завершения курсов.

В Skillbox разноформатное обучение. Самым основным являются видеолекции, после которых обязательным порядком дают практические домашние задания. Проверяют д/з и консультируют по ним педагоги экспертного уровня в мессенджерах. Но помимо этого есть онлайн-семинары и «живые» встречи офлайн-формата.

TeachMeSkills [2] - это школа программирования, где учат востребованным сегодня знаниям. Все программы составлены Senior- и Lead-разработчиками ведущих IT компаний специально для новичков в IT. Главная страница сайта представлена на рисунке 1.2.

Сайт обладает динамическими стилями и информативным слайдером на главной странице.

## Ближайшие курсы

**Front End разработчик**

Старт: 30 мая

[Подробнее →](#)

**Python разработчик**

Старт: 22 июня

[Подробнее →](#)

**Java разработчик**

Старт: 31 мая

[Подробнее →](#)

[Посмотреть все курсы](#)

Рисунок 1.2 – Главная страница TeachMeSkills

GeekBrains [3] В этой онлайн-школе можно найти практически всё, что касается прямо или косвенно IT-сферы. Тут обучают не только программированию или дизайну, но и интернет-маркетингу или системному администрированию. Какой курс GeekBrains не открой везде есть расписанная программа со всеми подробностями и отзывы от пользователей, проходивших тот или иной курс. Для студентов предусмотрена доступная программа стажировок и получение сертификатов о прохождении курсов. Главная страница сайта представлена на рисунке 1.3.

IT

Маркетинг

Дизайн

При покупке любой программы получите бонусом 5 курсов

В ПОДАРОК!

Основы языка Python
 Figma. Начальный уровень
 English для IT специалистов
 Инструменты продвижения в SMM
 Продакт-менеджмент: от идеи до выхода на рынок

Выбрать программу

IT

Обучение до уровня Middle в программировании, тестировании, информационной безопасности, DevOps и аналитике.

Факультет

Помощь с трудоустройством

Факультет Python-разработки

Получите одну из самых востребованных IT-профессий. Вы освоите

Факультет

Помощь с трудоустройством

Разработчик

Получите востребованную профессию и возможно

Отправьте нам сообщение

живо

Рисунок 1.3 – Главная страница GeekBrains

7

Преимущество этого образовательного портала в доступе к множеству бесплатного контента, но главное преимущество – это возможность обучения у топовых айтишников. Осилить современные профессии в мире диджитал можно на факультетах веб-, iOS-, Go-, Java- и Python-разработки, искусственного интеллекта, DevOps и других.

## **1.2 Постановка задачи**

Анализ существующих сайтов и приложений позволил определиться с постановкой задачи.

Итогом разработки должно стать веб-приложение для просмотра информации о курсах, предлагаемых сервисом, просмотра информации о тренерах ведущих курс и изменения (администрирования) данной информации.

Разумеется, необходимо разработать несколько интерфейсов: для пользователя и для администратора.

Интерфейс пользователя упрощен. Он должен давать возможность посмотреть каталог курсов, тренеров, а также детальную информацию о каждом из них.

Интерфейс администратора должен предусматривать всевозможные операции с содержимым сайта, описаниями.

Также можно обозначить функциональные требования курсовой работы:

- авторизация администратора;
- возможность просмотра данных о курсах и тренерах для всех пользователей;
- возможность добавления, изменения и удаления данных о курсах и тренерах администратором.

Данные требования будут реализованы в web-приложении и описаны в главе 3 данной пояснительной записки.

## **1.3 Описание используемых технологий**

При проектировании программного средства в качестве программной платформы для сервера был выбран Node.js [4]. Node.js – программная платформа для разработки серверных web-приложений на основе движка JavaScript Chrome V8. Является средой исполнения приложений на JavaScript. Также ориентирована на события, поддерживает асинхронность и является однопоточной. Также был использован пакет Express.js, который является популярным веб-фреймворком, написанным на JavaScript и работающий внутри среды исполнения node.js. Express не мешает общей производительности приложения т.к. представляет собой тонкий слой основных функций веб-приложений. Также в нём удобно настраивать маршруты приложения, поскольку он использует уже давно известные методы http.

Приложение построено таким образом, чтобы соответствовать архитектуре MVC [5]. Оно разделено на три части Model, View, Controller. Такая архитектура используется для того, чтобы упростить большой по объёму код. MVC применима

к разным видам приложений: серверным веб-приложениям, десктопным приложениям.

Для проектирования базы данных используется система управления базами данных MySQL. Эта серверная система способна эффективно функционировать во взаимодействии с интернет-сайтами и веб-приложениями.

Для удобного взаимодействия приложения с базой данных используется ORM-библиотека Sequelize[6] – применяется для приложений на Node.js, осуществляет сопоставление таблиц в базе данных и отношений между ними с классами. При использовании Sequelize мы можем не писать SQL-запросы, а работать с данными как с обычными объектами. Причем Sequelize может работать с рядом СУБД – MySQL, Postgres, MariaDB, SQLite, MS SQL Server.

Протокол WebSocket тоже будет реализован в приложении. WebSocket [7] является абстракцией над протоколом TCP и используется для обмена данными между клиентом и сервером в режиме реального времени.

## 2 Проектирование приложения

Разработка архитектуры проекта – важная задача в процессе работы над приложением, потому что в зависимости от неё определяется уровень связности между компонентами приложения, и насколько легко можно будет это приложение расширить.

### 2.1 Проектирование модели базы данных

В данной курсовой работе для проектирования базы данных используется система управления базами данных MySQL. MySQL представляет систему управления реляционными базами данных (СУБД). На сегодняшний день это одна из самых популярных систем управления базами данных.

В приложении А представлены таблицы разработанной базы данных ITCOURSES и связи между ними.

Таблица *users* хранит данные об пользователях.

- *id* – генерируется автоматически, однозначно идентифицирует объект;
- *username* – имя пользователя;
- *password* – пароль пользователя, храниться в базе данных в зашифрованном виде;

- *role* – роль пользователя.

Таблица *courses* содержит информацию о проводимых курсах.

- *id* – генерируется автоматически, однозначно идентифицирует объект;
- *language* – язык, который изучается на курсе;
- *courseName* – название курса;
- *courseDescription* – описание курса;
- *teacher* – преподаватель, который ведет курс;
- *image* – обложка курса.

Таблица *enrollments* содержит информацию о пользователях и выбранных ими курсах.

- *id* – генерируется автоматически, однозначно идентифицирует объект;
- *student* – имя студента;
- *course* – название курса;
- *lector* – имя преподавателя.

Таблица *teacher* предназначена для хранения информации о преподавателях.

- *id* – генерируется автоматически, однозначно идентифицирует объект;
- *tname* – имя преподавателя;
- *photo* – фото преподавателя;
- *subject* – язык с которым работает преподаватель.

### 2.2 Проектирование структуры web-приложения

В проекте имеется следующие директории. Структура проекта представлена на рисунке 2.1



Директория *config* содержит в себе файл *models.js*, которые описывает модели данные, а также логику управления этими данными, файл *db.js* содержит в себе строку подключения к базе данных.

Директория *controllers* предназначена для хранения контролеров.

Директория *routers* содержит файлы, которые отвечают за обработку логики запроса пользователя.

Директория *safety* содержит файл, который используется для хеширования паролей пользователей, для безопасного хранения их в базе данных.

Директория *static* предназначен для хранения статических файлов, представленных в виде картинок, css и js-файлов.

Директория *view* хранит в себе представления.

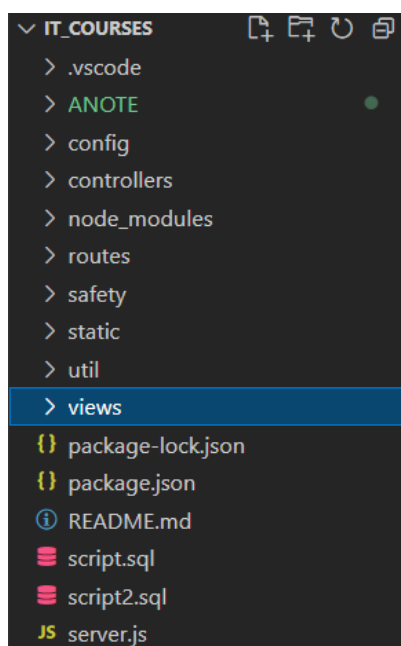


Рисунок 2.1 – Структура программного средства

## 3 Разработка приложения

### 3.1 Реализация модели базы данных

Для удобной работы с данными из базы данных ITCOURSES используется модуль *mysql2*. Чтобы было удобно работать, необходимо разработать модель данных. Листинг создания всех составляющих модели данных представлен в приложении Б.

В процессе создания таблиц нужно также их связать. В листинге 3.1 продемонстрирована связь между таблицей Users и таблицей Enrollments.

```
Users.hasMany(Enroll, {as:'enroll_users', foreignKey:'student', sourceKey:'id'})
```

Листинг 3.1 – Связь между таблицами

База данных к приложению подключается так, как показано в листинге 3.2.

```
global.sequelize = new Sequelize('ITCOURSES', 'root', 'Zalesse2015!',  
{host:'localhost', dialect:'mysql'})
```

Листинг 3.2 – Подключение базы данных

### 3.2 Проектирование структуры сервера

Маршрутизация в приложении организована с помощью роутеров. Для этого используется Express. Ниже в листинге 3.3 представлен код добавления роутеров с учётом их расположения в директориях проекта. Также мы настраиваем приложение на использование роутеров по определённому пути.

```
let authRouter = require('./routes/auth.route')  
let coursesRouter = require('./routes/courses.route')  
let teacherRouter = require('./routes/teacher.route')  
let enrollmentsRouter = require('./routes/enrollments.route')  
app.use(authRouter)  
app.use(coursesRouter)  
app.use(teacherRouter)  
app.use(enrollmentsRouter)
```

Листинг 3.4 – Добавление роутеров для определения маршрутизации запросов

В следующем листинге 3.5, можно увидеть реализацию метода GET роутера *courses* – этот метод должен возвращать полученные данные из базы данных *courses*.

```
getCourses: (req, res, next) => {  
  db.models.Courses.findAll()  
  .then(expense => res.send(JSON.stringify(expense)))  
  .catch((err) => console.log('Error: ' + err.message));}
```

Листинг 3.5 – Обработка GET запроса по адресу '/courses'

Обработка `get` запроса, делает запрос к базе данных метод `findAll()`, после чего отправляет полученные данные на клиент. Клиент обрабатывает полученные данные, формирует поля для вывода данных, и перенаправляет к функции, которая переадресует на страницу `courses.html`, на которой выведет полученные данные.

Листинг всего файла `courses.js` приведён в приложении В. Остальные роутеры построены по аналогии.

Веб-сокеты также являются одним из пунктов реализации приложения. `WebSocket` - это протокол передачи данных, основанный на протоколе TCP обеспечивающий обмен сообщениями между клиентом и сервером в режиме реального времени. Для реализации этой технологии был подключен модуль `socket.io`.

`Socket.IO` — JavaScript-библиотека для веб-приложений и обмена данными в реальном времени. Состоит из двух частей: клиентской, которая запускается в браузере и серверной для `node.js`. Оба компонента имеют похожее API.

В листинге 3.6 показано подключение модуля `socket.io` для работы с веб сокетами.

```
const io = require('socket.io')(server);

app.use(function(req, res, next) {

    req.io = io;
    next();
});
```

Листинг 3.6 – Подключение веб-сокетов

В данном приложении веб-сокеты помогают реализовать чат с администратором в реальном времени. Реализация на стороне клиента представлена в приложении Г. Реализация на стороне сервера представлена в приложении Д.

В приложении предусмотрена авторизация и регистрация пользователей. Также в приложении присутствуют роли администратора и пользователя. Аутентификация – процедура проверки подлинности идентификации пользователя. В приложении происходит проверка путем сравнения введенного пароля с паролем, который сохранен в базе данных и соответствует логину.

Эта функция принимает логин и пароль, который вводит пользователь, дальше идет поиск пользователя с введенным логином в базе данных и проверка пароля с помощью библиотеки `BCrypt`. В случае успеха происходит формирование `JSON Web Token (JWT)`, который в дальнейшем будет использоваться для доступа к приложению, и отправка его пользователю. `JWT` состоит из трех частей: заголовок `header`, полезные данные `payload` и подпись `signature`.

Для формирования `JWT` используется библиотека `jsonwebtoken`.

### 3.3 Проектирование структуры клиента

На стороне клиента используются представления. В приложении есть отдельные директория `views` с представлениями для каждого роутера. Структура директории представлена на рисунке 3.1. Главная страница `main.html` открывается после успешной аутентификации, с этой страницы мы можем переходить другие страницы: `courses.html` – страницы курсов, `teacher.html` – страница преподавателей, `mycourses.html` – страница курсов на которые записался пользователь, `chat.html` – чат с администратором.

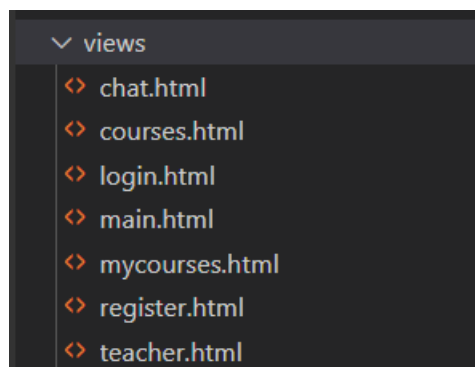


Рисунок 3.1 – Содержимое директория `views`

В директории `static` также хранится дополнительный функционал для клиентской части. В нём расположены клиентские стили, файлы клиентской части кода на JavaScript, картинки используемые на страницах приложения

Описанный директорий показан на рисунке 3.2.

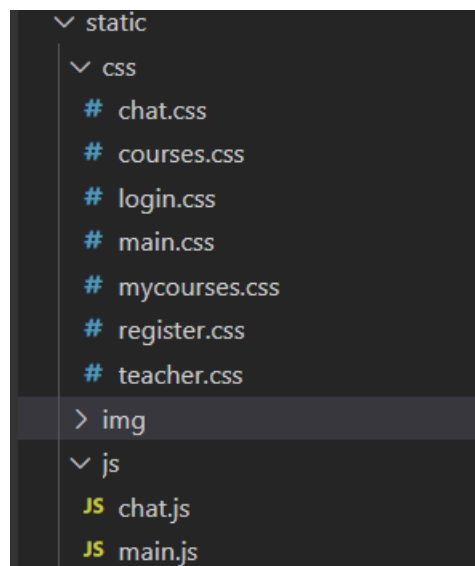


Рисунок 3.2 – Содержимое директория `static`

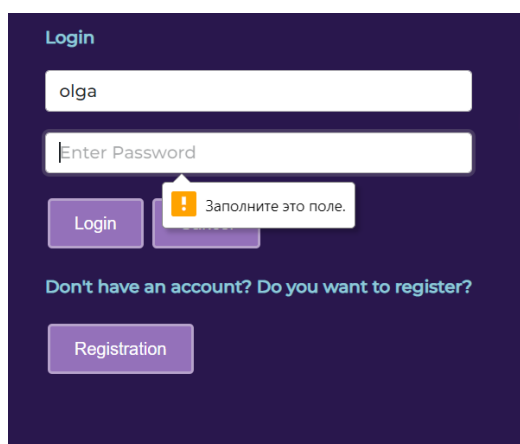
В приложении Е представлен пример страницы `Login.html` приложения

## 4 Тестирование

Для обеспечения корректности работы программы, обрабатываются различные ошибки, возникающие в процессе работы. Данное программное средство использует подключение к базе данных, следовательно, неправильно введенные данные или же их отсутствие может повлечь за собой неработоспособность приложения.

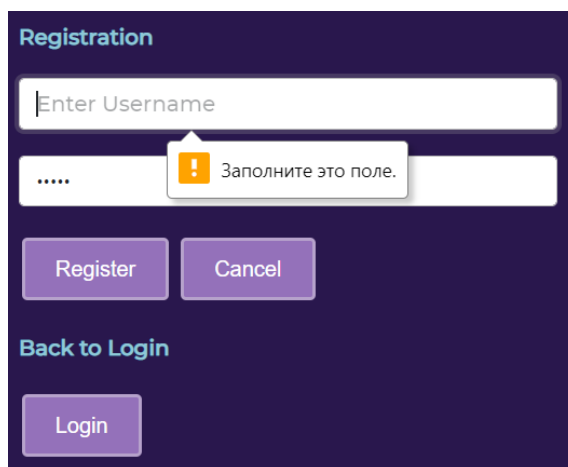
### 4.1 Тестирование формы логина

При заполнении формы логина и регистрации в приложении пользователь может ввести данные не во все обязательные поля. Пример обработки ошибок на страницу Login представлен на рисунке 4.1. Пример обработки ошибок на страницу Registration представлен на рисунке 4.2.



The screenshot shows a 'Login' form with a dark blue background. It contains two input fields: the first has the text 'olga' and the second is labeled 'Enter Password'. Below the password field, a red error message box with a white exclamation mark icon displays the text 'Заполните это поле.' (Fill in this field.). There are two buttons: 'Login' and 'Registration'. Below the buttons, the text 'Don't have an account? Do you want to register?' is visible.

Рисунок 4.1 – Обработка ввода пустых данных Login



The screenshot shows a 'Registration' form with a dark blue background. It contains two input fields: the first is labeled 'Enter Username' and the second is masked with dots. A red error message box with a white exclamation mark icon is positioned over the second field, displaying the text 'Заполните это поле.' (Fill in this field.). There are two buttons: 'Register' and 'Cancel'. Below the buttons, the text 'Back to Login' is visible, followed by a 'Login' button.

Рисунок 4.2 – Обработка ввода пустых данных Registration

## 4.2 Тестирование работы WebSockets

Используя WebSockets, предполагается, что некоторые действия будут происходить в реальном времени. С помощью данной технологии в приложении реализован чат с администратором. Попробуем продемонстрировать работу WebSockets на рисунке 4.3.

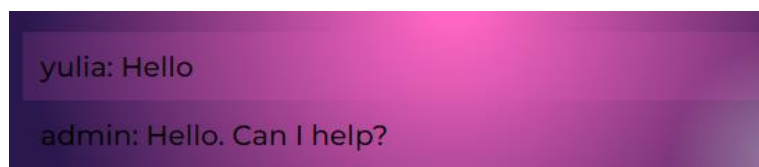


Рисунок 4.3 – Тестирование WebSockets

## 4.3 Тестирование работы поиска курсов

На странице курсов имеется поисковая строка по названию языка программирования. Тестирование данной функции показана на рисунке 4.4.

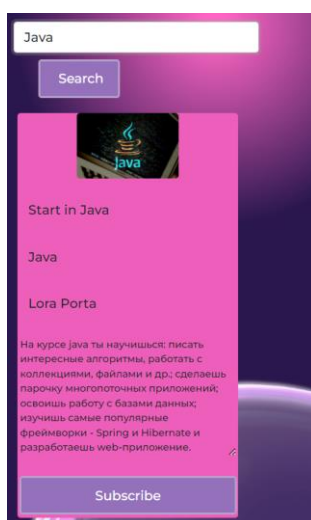


Рисунок 4.4 – Тестирование поиска по названиям курсов

## 4.4 Тестирование добавления новых данных

Администратор может добавлять новые данные в таблицы курсов и преподавателей. В приложении сделана обработка ошибок если поля при заполнении остались пустыми. Пример обработки показан на рисунке 4.5.

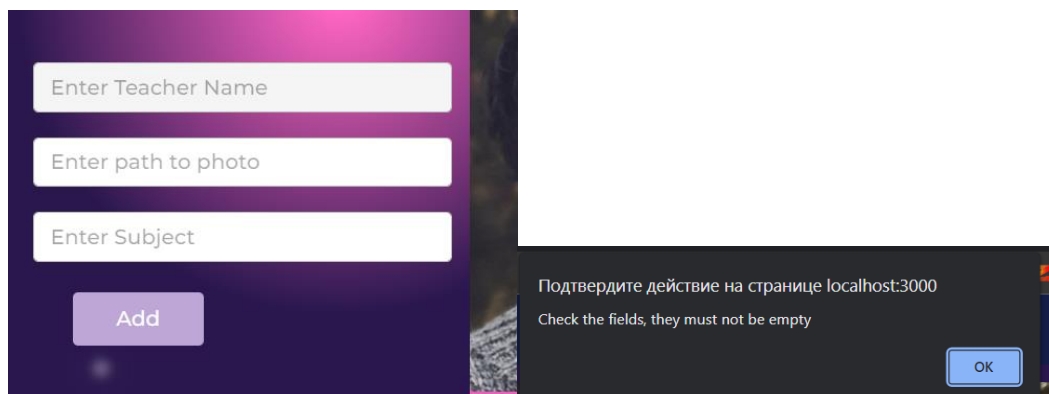


Рисунок 4.5 – Обработка ошибки при пустых полях

## 5 Руководство пользователя

Начальной страницей приложения является страница Login. Только после регистарции/авторизации можно перейти на дальнейшие страницы.

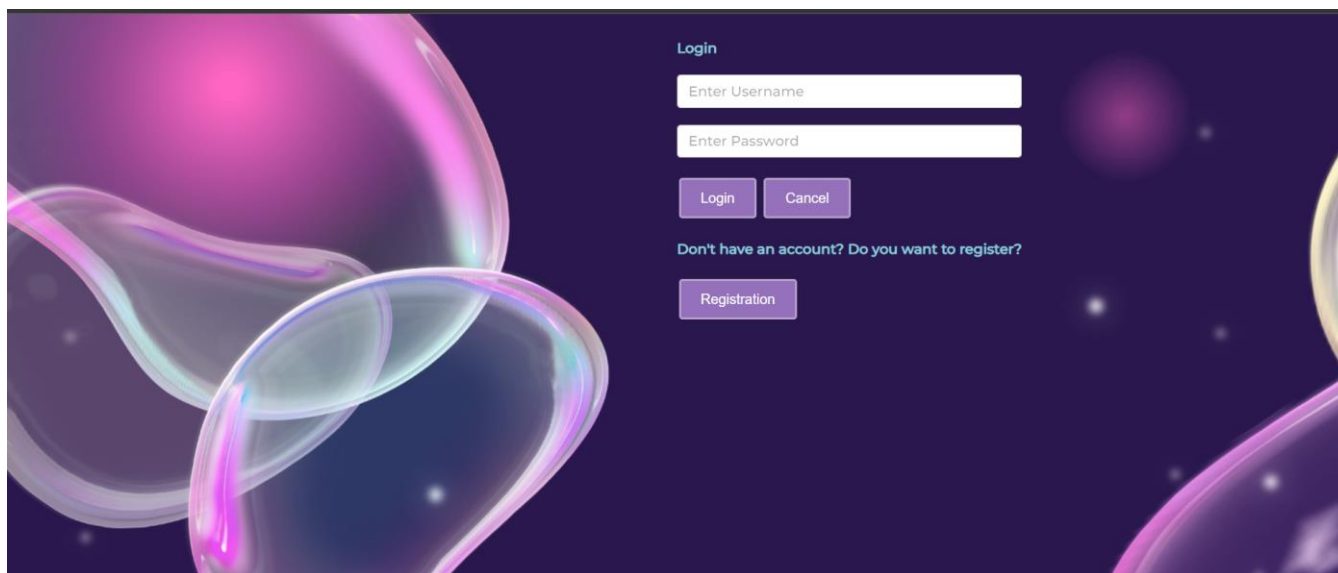


Рисунок 5.1 – Начальная страница web-приложения

Страница main открываемая после успешной авторизации показана на рисунке 5.2.

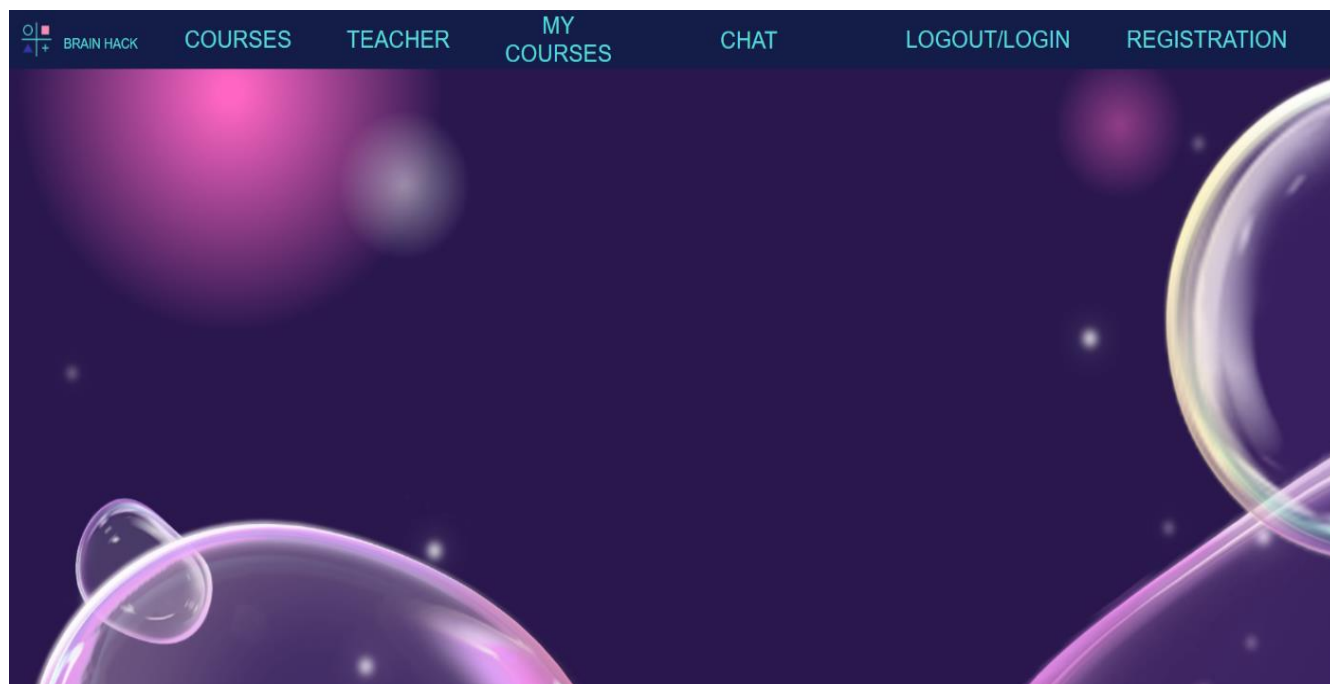


Рисунок 5.2 – Главная страница

Далее пользователь может перейти по вкладкам меню «Courses» или «Teacher», «MyCourses» для предоставления соответствующей информации.

Если пользователь перешел на страницу с курсами, то ему предоставляется интерфейс, представленный на рисунке 5.3. На странице имеется поле поиска.

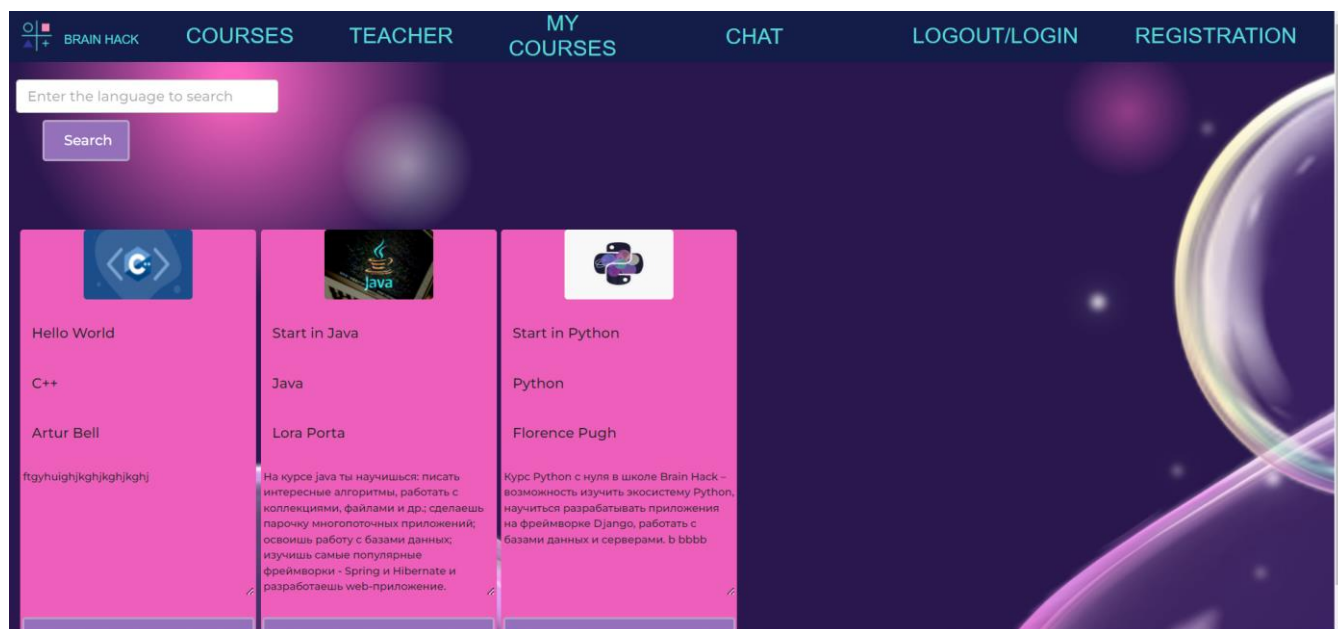


Рисунок 5.3 – Страница со курсами

При переходе на страницу «MyCourses» открывается чат с администратором представленный на рисунке 5.4.



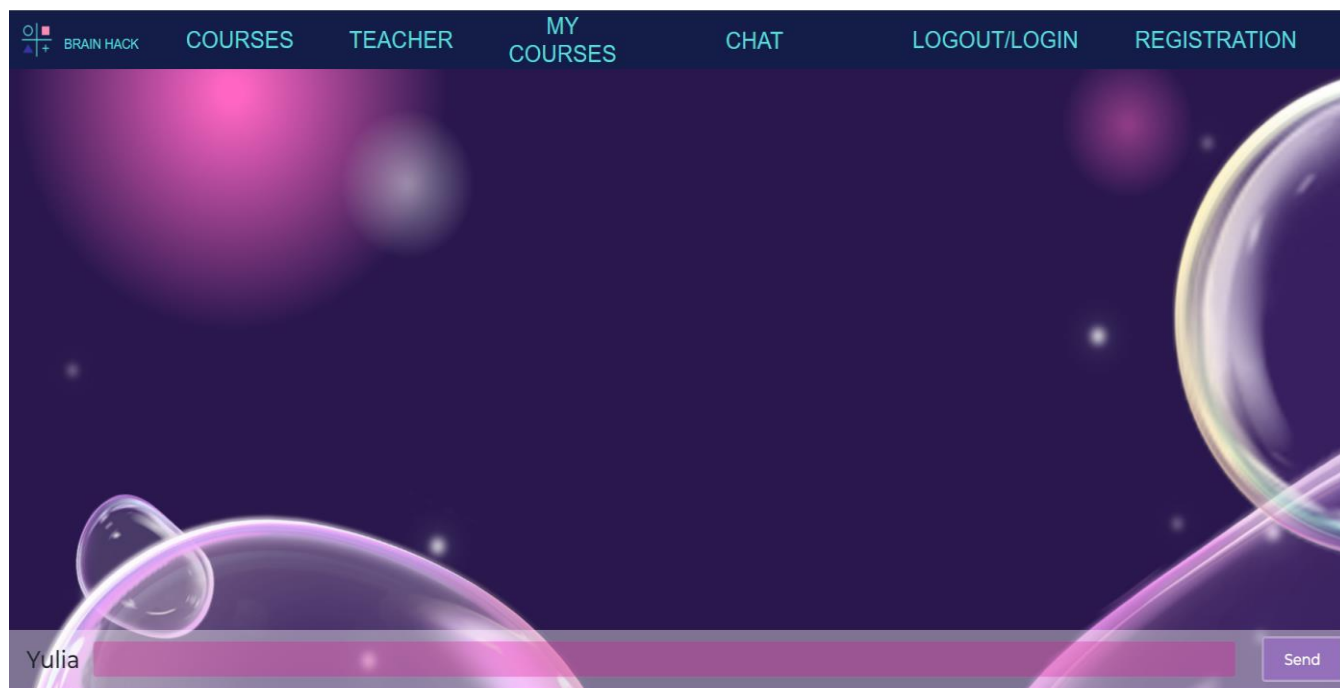


Рисунок 5.4 – Чат с администратором

Пользователь может подписаться на курс, после чего может увидеть его на вкладке «MyCourses» страница представлена на рисунке 5.5.

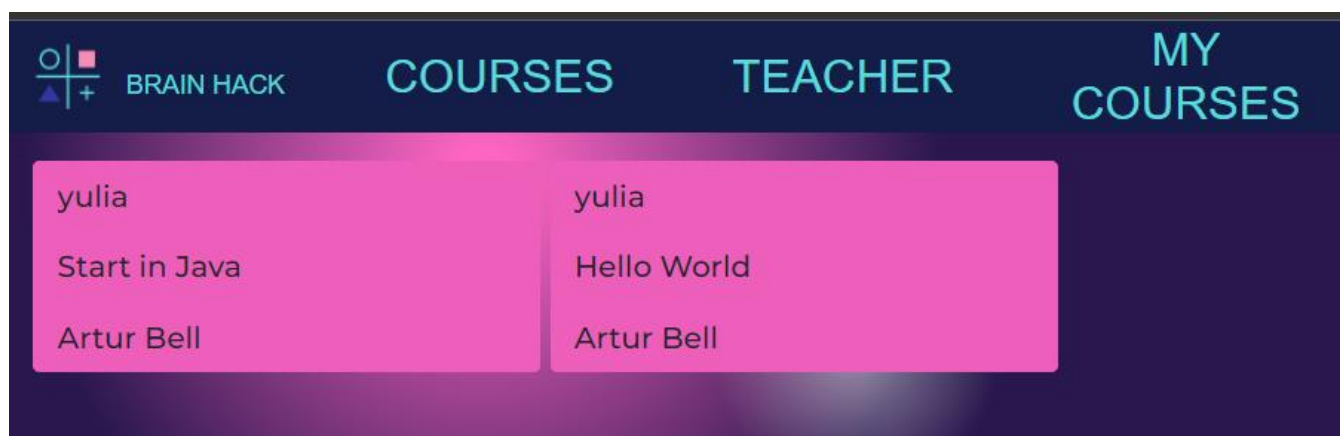


Рисунок 5.5 – Кусы пользователя

## Заключение

В результате проделанной работы было создано веб-приложение ITCOURSES на основе MVC, с помощью которого зарегистрированные пользователи могут узнать о курсах, которые проводятся, а также получить информацию о преподавателях, которые ведут эти курсы.

Перед началом разработки программного средства был произведен аналитический обзор прототипов приложений подобной тематики и определение функциональных возможностей разрабатываемого приложения.

Данный проект представляет собой пользовательский интерфейс, предназначенный для просмотра и изменения данных из базы данных. В процессе выполнения курсовой работы была спроектирована база данных для хранения информации о тренерах, курсах и пользователях. База данных была разработана с помощью реляционной системы управления базами данных MySQL.

Приложение написано с помощью платформы Node.js. Также было написано руководство пользователя для созданного приложения.

В приложении была реализована технология веб-сокетов для обмена сообщениями между клиентом и сервером

При разработке выполнены следующие пункты:

- просмотр информации о курсах, преподавателях и просмотр курсов на которые записался пользователь для зарегистрированных пользователей;
- просмотр информации о курсах и преподавателях для администратора;
- добавление курсов, преподавателей;
- изменение курсов, преподавателей;
- удаление курсов, преподавателей;
- запись на курс;
- поиск курсов по изучаемому языку.

В соответствии с полученным результатом, можно сказать, что разработанная приложение функционирует верно, требования технического задания реализованы в полном объеме, поэтому цель курсового проекта можно считать достигнутой.

## Список литературных источников

1 Онлайн-ресурс для обучения программированию «Skillbox» [Электронный ресурс] – Режим доступа: <https://sales.skillbox.by/> – Дата доступа: 10.05.2022.

2 Онлайн-ресурс школы программирования «TeachMeSkills» [Электронный ресурс] – Режим доступа: <https://teachmeskills.by/> – Дата доступа: 10.05.2022.

3 Онлайн-ресурс для изучения программирования «GeekBrains» [Электронный ресурс] – Режим доступа: <https://geekbrains.by/> – Дата доступа: 10.05.2022.

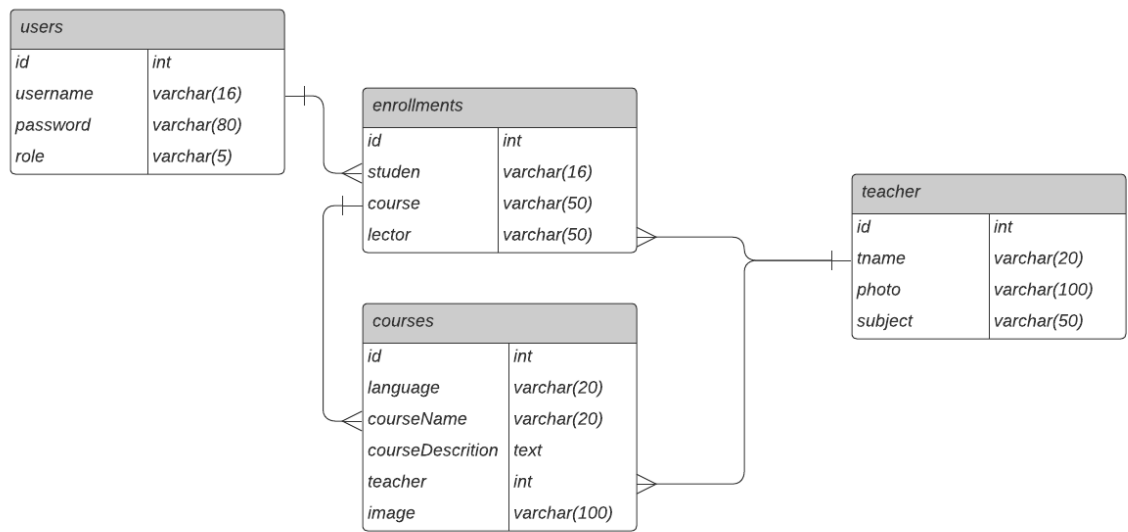
4 Node.js [Электронный ресурс] – Режим доступа: <https://nodejsdev.ru/> – Дата доступа: 16.05.2022.

5 Building and structuring a Node.js MVC application [Электронный ресурс] — Режим доступа: <https://blog.logrocket.com/building-structuring-node-js-mvc-application/> – Дата доступа: 24.05.2022.

6 Sequelize [Электронный ресурс] – Режим доступа: <https://sequelize.org/> – Дата доступа: 24.05.2022.

7 Socket.IO [Электронный ресурс] — Режим доступа: <https://socket.io/> – Дата доступа: 26.05.2022.

## Приложение А



## Приложение Б

```
Users.init (
  {
    id: {type: Sequelize.INTEGER, unique: true, autoIncrementIdentity: true},
    username:{type: Sequelize.STRING, primaryKey:true, allowNull: false, unique:
true},
    password: {type: Sequelize.STRING, allowNull: false},
    role: {type: Sequelize.STRING, validate: {isIn:[['user', 'admin']]]}}
  },
  {sequelize, modelName:'Users', tableName:'users', timestamps: false}
)

Enroll.init(
  {
    id: {type: Sequelize.INTEGER, primaryKey:true, autoIncrementIdentity: true},
    student: {type: Sequelize.INTEGER, allowNull: false},
    course: {type: Sequelize.INTEGER, allowNull: false},
    lector: {type: Sequelize.INTEGER, allowNull: false}
  },
  {sequelize, modelName:'Enroll', tableName:'enrollments', timestamps: false}
)

Courses.init(
  {
    id: {type: Sequelize.INTEGER, unique: true, autoIncrementIdentity: true},
    language: {type: Sequelize.STRING, allowNull: false},
    courseName: {type: Sequelize.STRING, primaryKey:true, allowNull: false},
    courseDescription: {type: Sequelize.STRING, allowNull: false},
    teacher: {type: Sequelize.STRING, allowNull: false},
    image:{type: Sequelize.STRING}
  },
  {sequelize, modelName:'Courses', tableName:'courses', timestamps: false}
)

Teacher.init(
  {
    id: {type: Sequelize.INTEGER, autoIncrementIdentity: true, unique: true},
    tname: {type: Sequelize.STRING, primaryKey:true, allowNull: false},
    photo: {type: Sequelize.STRING},
    subject: {type: Sequelize.STRING, allowNull: false}
  },
  {sequelize, modelName:'Teacher', tableName:'teacher', timestamps: false}
)
```

## Приложение В

```
const { defineRulesFor } = require("../util/abilitites")
const db = require('../config/db')
const path = require('path')
const fs = require('fs')

const CoursesController = {
  getCoursesPage: (req, res, next) => {
    let view = fs.readFileSync('./views/courses.html', "utf8");
    res.send(view);
    // res.sendFile(path.join(__dirname, '../static/html/courses.html'))
  },

  getChatPage: (req, res, next) =>{
    res.sendFile(path.join(__dirname, '../views/chat.html'))
  },

  getCourses: (req, res, next) => {
    db.models.Courses.findAll()
      .then(expense => res.send(JSON.stringify(expense)))
      .catch((err) => console.log('Error: ' + err.message));
  },

  deleteCourses: (req, res, next) => {
    db.models.Enroll.destroy({where:
req.body.courseName}}).then(()=>{res.send();})
      .catch((err) => console.log('Error: ' + err.message))
      .then((res) => {
        console.log(res);
      });
    db.models.Courses.destroy({where:
req.body.courseName}}).then(()=>{res.send();})
      .catch((err) => console.log('Error: ' + err.message))
      .then((res) => {
        console.log(res);
      });
  },

  editCourses: (req, res, next) => {
    db.models.Courses.update({
      language: req.body.language,
      courseDescription: req.body.courseDescription,
      teacher: req.body.teacher,
      image: req.body.image,
      courseName: req.body.courseName
    }, {
      where:{ courseName: req.body.courseName
    }
    }).then((res) => {console.log(res);
    });
  },

  addCourses: (req, res, next) => {
    db.models.Courses.create({
      language: req.body.language,
      courseDescription: req.body.courseDescription,
      teacher: req.body.teacher,
      image: req.body.image,
      courseName: req.body.courseName
    })
    .catch((err) => console.log('Error: '+err.message));
  }
}
```

```

    },
    Search: async (req, res, next) => {
        const resp= await db.models.Courses.findOne({where: {language:
req.body.language}})
        console.log(resp);
        res.send(JSON.stringify(resp));
    }
}

module.exports = CoursesController

```

## Приложение Г

```
const socket = io();
const messages = document.querySelector('.messages')
const form = document.querySelector('.form')
const input = document.querySelector('.input')
const nameBlock = document.querySelector('.name')

const user = prompt('Your name')
nameBlock.innerHTML = `${user}`

form.addEventListener('submit', (e) => {
  e.preventDefault()

  if(input.value){
    socket.emit('chat message', {
      message: input.value,
      name: user
    })
    input.value = ''
  }
})

socket.on('chat message', (data) => {
  const item = document.createElement('li')
  item.innerHTML = `${data.name}</span>: ${data.message}`
  // item.innerHTML = `${data.message}`
  messages.appendChild(item)
})
```



## Приложение Д

```
app.get('/', (req, res) => {
  res.sendFile(__dirname+'/views/main.html')
})

io.on('connection', (socket) => {
  socket.on('chat message', (data) => {
    io.emit('chat message', {
      message: data.message,
      name: data.name
    })
  })
})
```

## Приложение Е

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@500&display=swap"
rel="stylesheet">
  <link rel="stylesheet" href="style/css/login.css">
  <title>Login</title>
</head>
<body>
  <div>
    <h4>Login</h4>
    <form action="/login" method="POST">
      <input class="text-field__input" type="text" placeholder="Enter Username"
name="username" required/>
      <br>
      <input class="text-field__input" type="password" placeholder="Enter
Password" name="password" required/>
      <br>
      <button class="button button1" type="submit">Login</button>
      <button class="button button1" type="reset"
class="cancelbtn">Cancel</button>
    </form>

    <h4> Don't have an account? Do you want to register?</h4>
    <form action = "/register">
      <button class="button button1" type="submit" action = "/register" method
="get">Registration</button>
    </form>
  </div>
</body>
</html>
```