*Федеральное государственное бюджетное образовательное учреждение высшего образования*

*«Владимирский государственный университет*

*имени Александра Григорьевича и Николая Григорьевича Столетовых»*

*Кафедра информационных систем и программной инженерии*

***ПОЯСНИТЕЛЬНАЯ ЗАПИСКА***

***к курсовому проекту по дисциплине   
"Технологии программирования"***

***на тему***

*Проектирование и разработка программной системы   
«Дистанционное обучение студента»*

*Выполнил: студент гр. ПРИ-116*

*Чернова Д.В.*

*Принял: доц. кафедры ИСПИ*

*Вершинин В.В.*

*Владимир, 2018*

**Аннотация**

Суть курсовой работы заключается в разработке программной системы дистанционного обучения студентов в образовательном учреждении. Работа состоит из пояснительной записки и графических материалов. Графическая часть представляет собой 5 чертежей, выполненных на разных листах формата А4 в масштабе 1:8.

Пояснительная записка содержит:

* 81 страницу;
* 17 рисунков.

The essence of the course work is the development of a software system for distance education. The work consists of an explanatory note and a graphic part. Graphical part includes 5 drawings, made on different sheets of A4 format on a scale of 1: 8.

The explanatory memorandum contains:

• 81 pages;

• 17 figures.

**Содержание**

[1. ВВЕДЕНИЕ 3](#_Toc503176518)

[2. Постановка задачи 4](#_Toc503176519)

[3. Описание предметной области 5](#_Toc503176520)

[3.1. Общие сведения о предмете автоматизации 5](#_Toc503176521)

[3.2. Словарь предметной области 6](#_Toc503176522)

[3.3. Сценарий взаимодействия пользователя с системой 7](#_Toc503176523)

[4. проектирование системы 9](#_Toc503176524)

[4.1 Диаграмма прецедентов 9](#_Toc503176525)

[4.2 Диаграмма классов 11](#_Toc503176526)

[4.3 Диаграмма состояний 13](#_Toc503176527)

[4.4 Диаграмма последовательности 14](#_Toc503176528)

[4.5 Структура базы данных 15](#_Toc503176529)

[5. реализация системы 17](#_Toc503176530)

[5.1 Принципы организации проекта 17](#_Toc503176536)

[5.2 Реализация модели 17](#_Toc503176537)

[5.3 Реализация поддержки различных типов пользователей 17](#_Toc503176538)

[5.4 Реализация поддержки различных справочников 18](#_Toc503176539)

[5.5 Реализация каталога курсов 18](#_Toc503176540)

[5.6 Реализация каталога курсов для преподавателя 18](#_Toc503176541)

[5.7 Пользовательский интерфейс 18](#_Toc503176542)

[6. заключение 25](#_Toc503176543)

[7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc503176544)

[ПРИЛОЖЕНИЕ А Исходный код программной системы 27](#_Toc503176545)

**1. ВВЕДЕНИЕ**

В данном курсовом проекте производится проектирование и реализация программной системы дистанционного обучения студентов в образовательном учреждении.

На данный момент множество образовательных учреждений хранят различные данные, например, оценки, задания, различные справочники курсов и пользователей – в электронном виде.

Программа предоставляет широкие возможности студентам и преподавателям. Пользуясь любым устройством, имеющим подключение к сети интернет, они могут просматривать свои курсы, которые проходят в рамках учебного процесса.

В ходе выполнения курсового проекта, требуется решить следующие задачи:

* анализ предметной области;
* разработка проекта программной системы;
* разработка прототипа программной системы.

Результатом выполнения курсового проекта будет работающее программное обеспечение, позволяющее вести учет дистанционного обучения в образовательном учреждении.

В ходе выполнения курсового проекта выполнена разработка проекта информационной системы с использованием языка UML, реализация диаграмм с использованием среды Microsoft Office Visio, реализация программного обеспечения с использованием технологий разработки .NET Core, Entity Framework, ReactJS. Среда программирования –Visual Studio Code.

1. **Постановка задачи**

Необходимо разработать программное обеспечение, ведущее учет дистанционного обучения студентов образовательного учреждения.

Для решения поставленной задачи необходимо последовательно выполнить этапы анализа, проектирования и реализации системы.

**Анализ предметной области.** Результатом данного этапа работы является описание бизнес-процесса учета знаний студентов и словарь предметной области.

**Проектирование системы.** Для проектирования системы необходимо использовать язык UML. Проектирование должно включать в себя следующие UML-диаграммы:

* диаграмма прецедентов;
* диаграмма классов;
* диаграмма состояний;
* диаграмма последовательностей.

Также этап проектирования включает схему базы данных в виде ER-диаграммы логической моделей данных.

Программная реализация.

Минимальный набор функций:

1. создание и поддержка справочников пользователей;
2. создание и поддержка справочников курсов, тем и задач;
3. просмотр содержимого курсов;
4. внесение ответа на задание.

Система должна быть реализована на языке C# в виде веб-приложения с использованием технологий .NET Core.

1. **Описание предметной области**
   1. **Общие сведения о предмете автоматизации**

Темой данного курсового проекта является программная система учёта дистанционного обучения в образовательном учреждении. Подобная программная система позволяет образовательным учреждениям расширить функциональные возможности как преподавателей, так и учеников. Пользователями системы дистанционного обучения станут студенты и преподаватели (роль «Студент» и «Преподаватель»), а также руководство учебного заведения (роль «Администратор»).

Программа предоставляет широкие возможности студентам и преподавателям. Пользуясь любым устройством, имеющим подключение к сети интернет, они могут просматривать курсы, материалы, темы и задания, получаемое в рамках учебного процесса. Ведение базы курсов и базы данных студентов и их заданий, помимо бумажных аналогов, позволит контролировать успеваемость, а также грамотно планировать расписание и выполнение заданий, что, в свою очередь, отразится на успешном выполнении учебного плана.

Пользуясь программной системой дистанционного обучения, персонал так же получит широкие возможности для планирования и контроля учебного процесса. В системе будут реализованы следующие функции: возможность написать или прикрепить ответ к заданию для студента, управление пользователями и ролями для администратора, возможность управления курсами для преподавателей. Возможность работы с системой, пользуясь любым устройством имеющем доступ к интернету, намного удобней, чем заполнение бумажного аналога в рамках одного занятия. Также это позволит некоторым преподавателям сэкономить время, которое обычно затрачивается на проведение аудиторных занятий.

Администрация образовательного учреждения с помощью программной системы сможет добавлять преподавателей и студентов в базу данных, создавать списки новых и текущих групп и формировать для них удобное расписание. Система предусматривает три роли для администрации образовательного учреждения: администратор, студент и преподаватель. Предполагается, что деканат или аналогичное руководство может самостоятельно найти администратора или выбрать его из числа имеющихся сотрудников, поэтому более подробное разделение на роли не предусмотрено. При этом происходит разделения обязанностей. Функции добавления студентов, формирования групп и расписания будут доступны администратору, преподаватель сможет редактировать курсы, темы и задания, а студент получает возможность просматривать содержимое курса, отвечать на задания и следить за своей успеваемостью.

Регистрация новых пользователей производится персоналом учебного заведения.

* 1. **Словарь предметной области**

Программная система - программная система дистанционного обучения студента в образовательном учреждении.

Гость – любой неавторизированный участник программной системы. Не имеет доступа к каким-либо функциям.

Пользователь – какой-либо зарегистрированный участник программной системы. Каждый пользователь имеет единственно определенную роль.

Справочник – фрагмент базы данных, хранящий информацию о группе определенных объектов. Пользователи с ролями, для которых задан соответствующий доступ, могут выполнять редактирование/добавление этих справочников, а также информации в них.

Группа – совокупность студентов, выделенных в отдельную группу. Каждая группа имеет свой уникальный перечень предметов, тестов и домашних заданий. Некоторые курсы могут пересекаться, но не совпадать полностью.

Каталог курсов – предоставляет функции просмотра справочника курсов и их содержимого.

Электронный журнал – предоставляет функции просмотра справочника групп и студентов.

Студент – роль, при которой пользователь, который имеет функцию просмотра своего каталога курсов, а также принадлежит к какой-либо группе. Регистрацию в системе производит администратор.

Преподаватель – роль, при которой пользователь имеет доступ к каталогу курсов. Регистрацию в системе производит администратор.

Администратор – роль, при которой пользователь может подтверждать внесённые изменения и управлять создание пользователей и предметов. Учетная запись уникальна и заведомо добавлена в базу данных программной системы.

Персонал – пользователь, который имеет одну из следующих ролей: администратор, студент, преподаватель.

* 1. **Сценарий взаимодействия пользователя с системой**

В приложении используется разделение по ролям. В соответствии с должностями, были получены следующие роли:

* Студент
* Преподаватель
* Администратор

Пользователь с ролью *Студент* может:

* Просматривать свои курсы, темы в них и задания согласно своей группе
* Просматривать информацию о предмете или задании
* Отвечать на задания

Пользователь с ролью *Преподаватель* может:

* Просматривать список групп и студентов;
* Просматривать список курсов;
* Оценивать ответ студентов;
* Добавлять курсы, темы и задания по ним.

Пользователь с ролью *Администратор* может также:

* Осуществлять добавление, изменение и удаление пользователей

Гость имеет возможность:

* Авторизоваться;
* Зарегистрироваться;

1. **проектирование системы**
   1. **Диаграмма прецедентов**

Объект информатизации исследуется в границах одного образовательного учреждения. Действующими лицами (в нотации UML – актерами) являются: Студент, Преподаватель, Администратор и Гость. Ниже приведен перечень и описание некоторых прецедентов в проектируемой системе.

***Название:*** «Просмотр каталога курсов»

***Предусловие:*** пользователь имеет доступ к веб-сайту программной системы.

***Действующее лицо:*** студент

***Основной поток:*** Действующее лицо открывает окно приложения, отображающую информацию, заполненную в каталоге. С помощью клавиш навигации пользователь может посмотреть подробное содержание курса. Прецедент при этом использует функции просмотра каталога, просмотра тем и просмотра заданий. При завершении работы действующее лицо закрывает окно приложения.

***Название:*** «Управление каталогом курсов»

***Предусловие:*** персонал имеет доступ к веб-сайту программной системы.

***Действующее лица:*** преподаватель.

***Основной поток:*** действующее лицо открывает форму управления каталогом курсов и выполняет необходимые ему действия с помощью модальных окон, также с помощью клавиш навигации имеет возможность активировать прецедент «редактирование курса». В окне представлена информация о курсе и поля для их редактирования. При завершении работы действующее лицо сохраняет изменения и закрывает окно приложения.

***Альтернативный поток:*** пользователь ввёл данные, но по какой-либо причине решил не завершать сохранение. Пользователь может нажать на кнопку «Закрыть» или нажать ЛКМ в любом месте вне модального окна. Модальное окно закроется, изменения не сохранятся.

***Название:*** «Добавить пользователя»

***Действующее лица:*** администратор.

***Основной поток:*** действующее лицо открывает форму заполнения информации о пользователе с помощью кнопки «Добавить» в форме управления пользователями. Выполняет корректный ввод данных в формы (выполняет ввод следующих полей: email, логин, пароль) выбирает роль из выпадающего списка и нажимает кнопку «Добавить». Система создает нового пользователя с выбранной ролью и сохраняет данные в БД.

**Альтернативный поток 1:** Действующее лицо неправильно заполнило поля, при этом получило сообщение об ошибке. В таком случае, данный пользователь должен перепроверить и исправить введенные данные.

**Альтернативный поток 2:** Действующее лицо не желает добавлять нового пользователя по каким-либо причинам и нажимает кнопку «Закрыть». Система не создает нового пользователя и не сохраняет введенные данные, происходит переадресация на предыдущую страницу.

**Альтернативный поток 3:** Если введенные ключевые поля принадлежат уже зарегистрированному пользователю, система выдает сообщение об ошибке.

Диаграмма прецедентов представлена на рисунке 1.

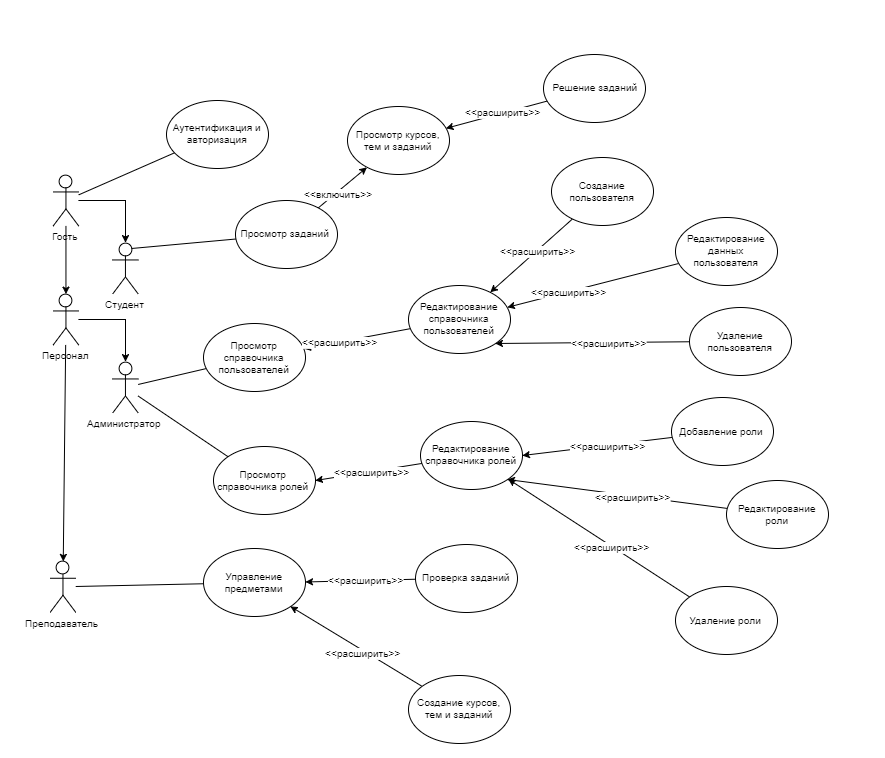


Рисунок 1. Диаграмма прецедентов

* 1. **Диаграмма классов**

Для проектирования структур и моделей данных программной системы была разработана диаграмма классов, представляющая основные сущности системы. Диаграмма классов представлена на рисунке 2[2].

Программной система реализована частично по паттерну проектирования MVC, методы доступа и взаимодействия с базой данных будут реализованы в классах DAO. На данном уровне абстракции эти методы исключены из диаграммы.

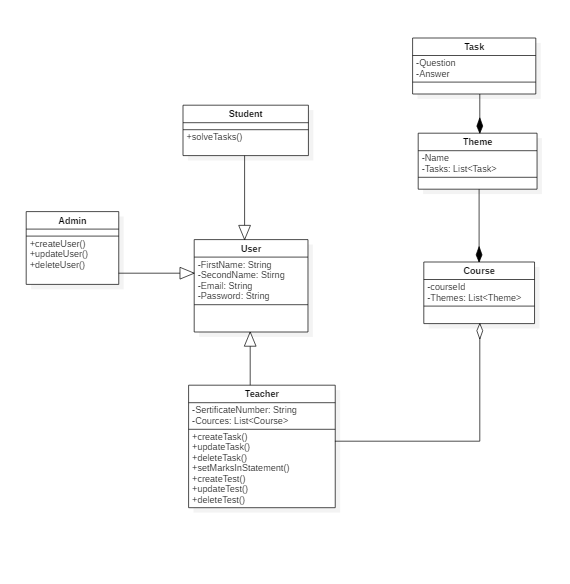


Рисунок 2. Диаграмма классов

В результате анализа были выделены следующие основные сущности:

*Group (Группа) –* сущность, объединяющая студентов и преподавателей. Группа имеет свой определённый идентификатор. Предполагается, что студент обязательно прикреплён к какой-то группе. И совокупность студентов, изучающих одни и те же предметы, называются группой*.* Так как у группы нет никаких атрибутов, и она имеет значение только на уровне базы данных, эта сущность так же не вынесена на диаграмму классов.

*Task (Задание) –* материал, содержащий вопрос или тестовое задание, которые необходимо выполнить студенту. Создаётся преподавателем (Teacher).

*Course (Предмет) –* отображает основную информацию о курсе;

*Student(Студент)* – пользователь программной системы, использующий её для дистанционного обучения;

*Teacher(Преподаватель)* – пользователь программной системы, который выполняет функции управления каталогом курсов;

*Admin (Администратор)* – пользователь программной системы, который выполняет функции создания и управления пользователями.

* 1. **Диаграмма состояний**

Одним из важнейших свойств данной системы является возможность создать и позже проверить задание, выдаваемое студенту, так как взаимодействие между пользователями осуществляется только дистанционно.

Состояние объекта «Задание».

1. Пользователь (преподаватель) составляет задание (текстовый вопрос или перечень вопросов), с кнопок навигации добавляет необходимые задания. Объект «задание» создан в системе. Но пока преподаватель не нажмёт кнопку «Сохранить», задание не заносится в базу данных.

2. Пользователь сохраняет изменения, либо отменяет их, как было описано в соответствующем прецеденте. Объект добавляется в БД. После добавления он доступен всем пользователям, которым доступен курс.

3. После отправки, задание доступно всем пользователям. Студенты могут выполнять их, загружая или заполняя формы с ответами. Преподаватель, в свою очередь, может оценить задание.

Диаграмма состояний представлена на рис.3.

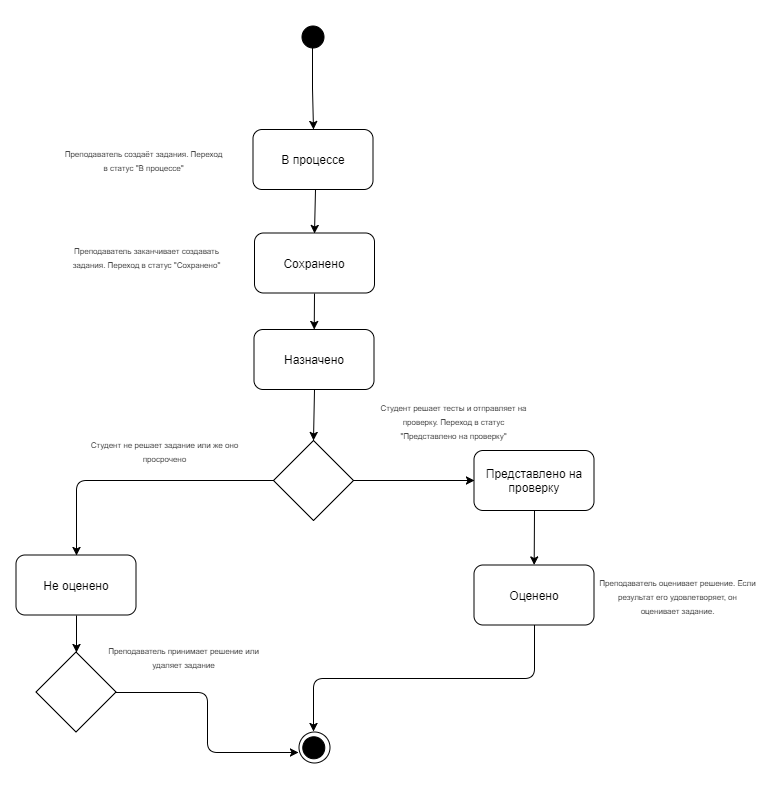


Рисунок 3. Диаграмма состояний объекта «Задание».

* 1. **Диаграмма последовательности**

Диаграмма последовательности разработана для прецедента «Создать пользователя».

Диаграмма последовательностей представлена на рисунке 4.



Рисунок 4. Диаграмма последовательностей для прецедента «Создать пользователя».

* 1. **Структура базы данных**

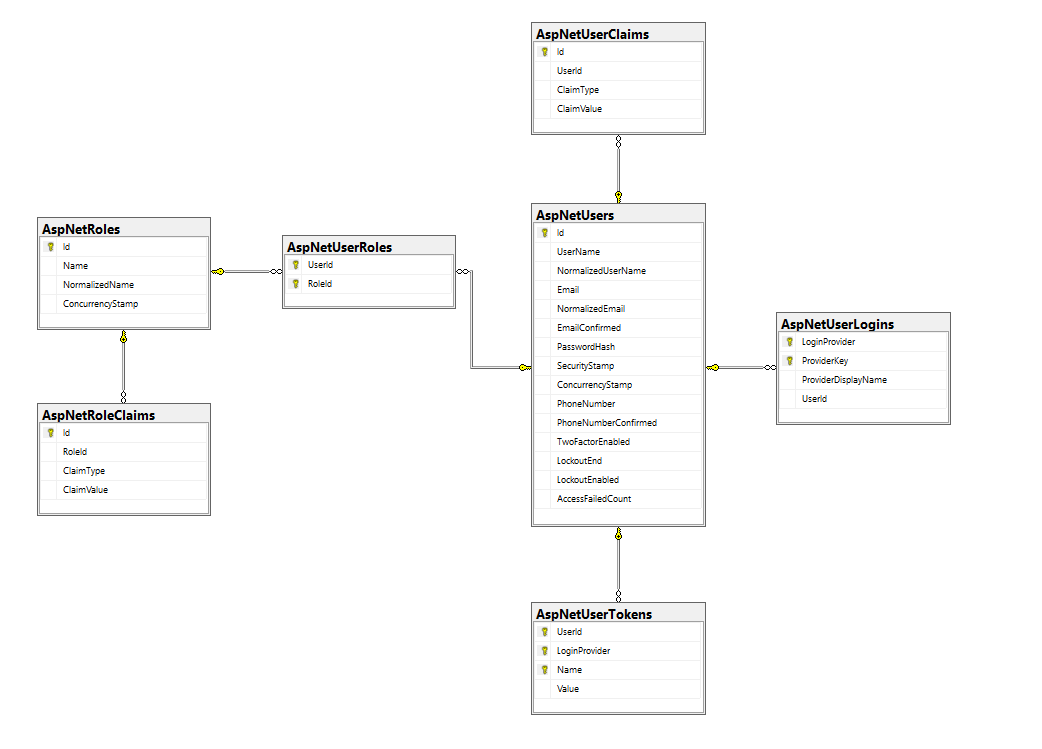
Разработанная модель структуры базы данных является нормализованной к третьей нормальной форме. Схема модели базы данных приведена на рис.5 и рис.6. 

Рисунок 5. Схема модели базы данных

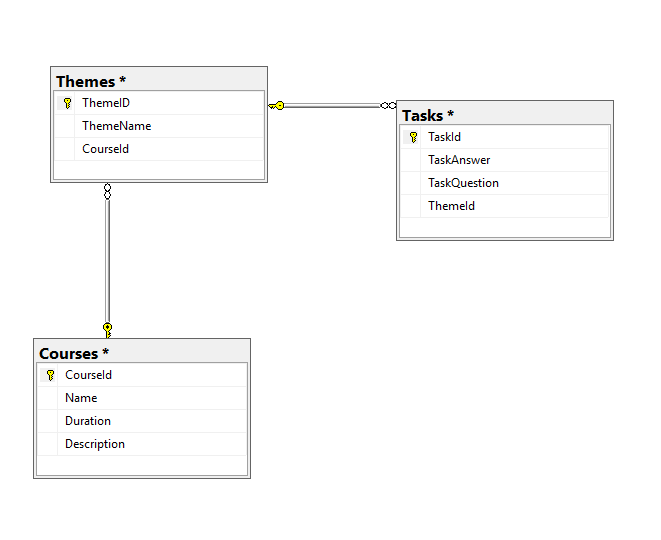
****

Рисунок 6. Схема модели базы данных

1. **реализация системы**



6. 1. **Принципы организации проекта**

Разрабатываемая система построена частично на архитектуре MVC с использованием технологии Entity Framework, языка интегрированных запросов LINQ. Система представляет собой веб-приложение, и, таким образом, разделяется на три части: контроллер, модель, представление.

Представления построены с использованием ReactJS – фреймворк, который использует компонентную модель и преобразовывает код JSX в HTML. Структурно весь JavaScript-код находится в директории ClientApp. Исходный код приложения приведен в приложении А.

* 1. **Реализация модели**

В данной программной системе использовался подход Code First.То есть сначала создавались модели, а потом по ним с помощью возможности .NET строилась база данных. Возможность работать с миграциями позволила изменять модели (добавлять новые, изменять существующие) без потери данных. Также данный метод позволяет не работать напрямую со скриптами создания и изменения таблиц.

* 1. **Реализация поддержки различных типов пользователей**

Для реализации регистрации, аутентификации и авторизации пользователей были использованы стандартные методы Microsoft Identity.

Методы и представления имеют ограниченный доступ. Пользователям, в зависимости от роли, доступны различные функции. Это обеспечивается ограничениями, реализованными с помощью встроенных средств.

Регистрацию пользователей в системе производит администратор программной системы.

* 1. **Реализация поддержки различных справочников**

Администратор имеет возможность работы с базой данных. С помощью различных форм пользователи с ролью администратор могут взаимодействовать с такими справочниками как: справочник пользователей, групп и курсов, тем и заданий.

* 1. **Реализация каталога курсов**

Студент нажимает кнопку меню «Мои курсы» и попадает на страницу, которая отображает данные о курсах, на которые он записался. С помощью кнопки «Показать больше» студент может увидеть более подробную информацию о курсе, а также список тем и заданий в каждой теме.

* 1. **Реализация каталога курсов для преподавателя**

Учитель нажимает кнопку меню «Курсы» и попадает на страницу, которая отображает список курсов. Учителю доступны три кнопки: «Показать больше», «Редактировать» и «Удалить». Использование кнопки «Показать больше» аналогично тому, что видит студент. Нажимая на кнопку «Удалить», учитель видит всплывающее окно. Подтверждая действие, студент обновляет курс. Страница обновляется. Нажимая на кнопку «Редактировать», учитель видит модальное окно с полями, доступными для редактирования. Учитель вводит новые данные, нажимает на кнопку «Сохранить изменения». Модальное окно закрывается, страница обновляется.

* 1. **Пользовательский интерфейс**

Пользовательский интерфейс представлен на Рисунках 7-17. Формы добавления и редактирования имеют схожий друг с другом интерфейс, поэтому отдельные страницы не были добавлены в примеры интерфейса.

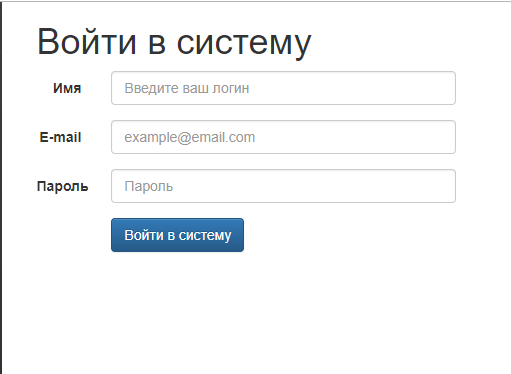


Рисунок 7. Страница входа

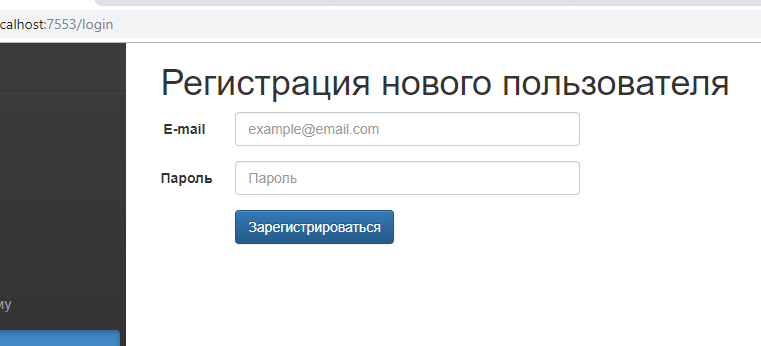


Рисунок 8. Страница регистрации

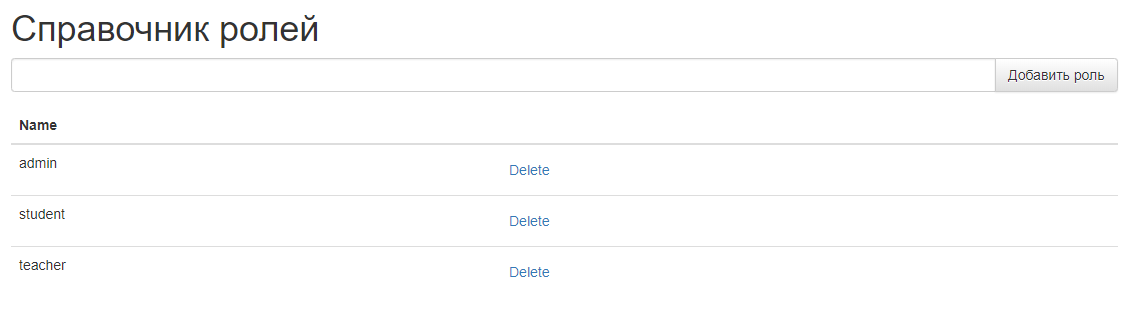


Рисунок 9. Справочник ролей

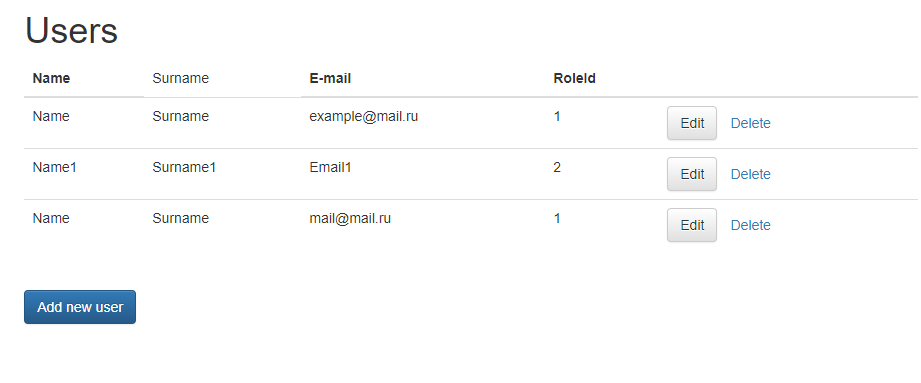


Рисунок 10. Справочник пользователей

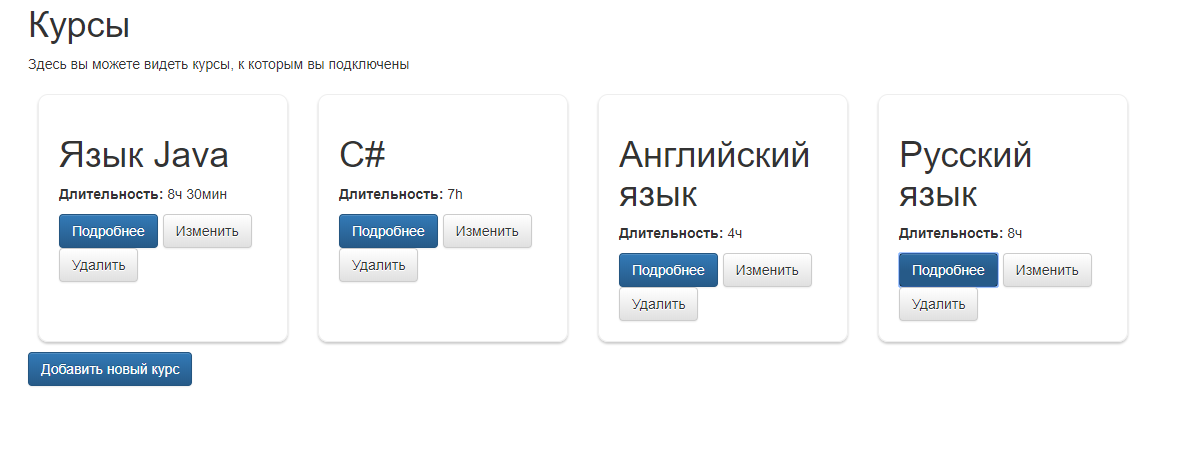


Рисунок 11. Справочник курсов

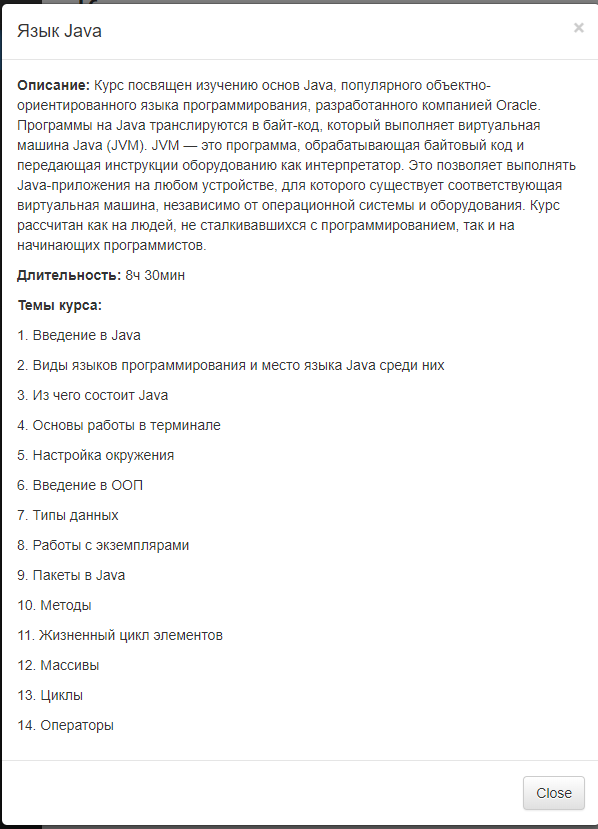


Рисунок 12. Подробное описание курса



Рисунок 13. Редактор тем

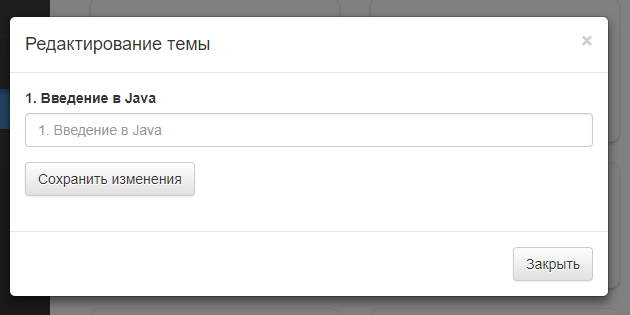


Рисунок 14. Редактирование темы

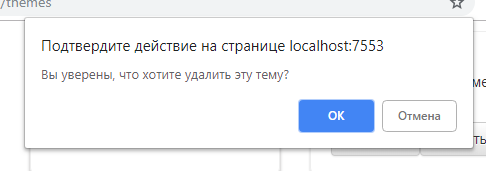


Рисунок 15. Подтверждение удаления

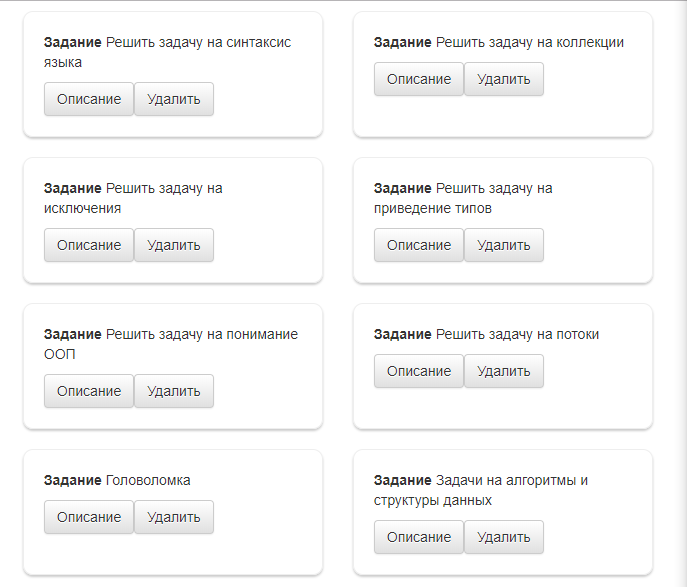


Рисунок 16. Список заданий

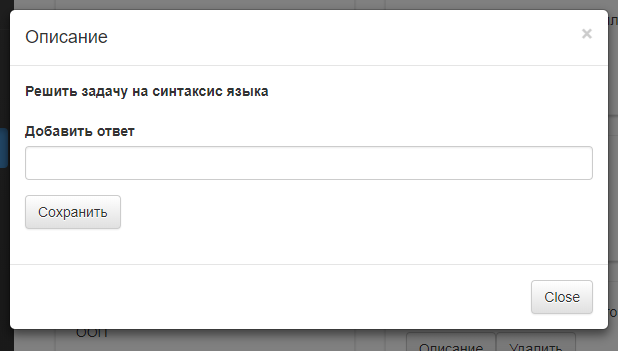


Рисунок 17. Описание задания

* 1. **заключение**

В данном курсовом проекте спроектирована и реализована программная система учета дистанционного обучения в учебном заведении. Для выполнения поставленной задачи было выполнено:

* Анализ предметной области и формирование требований к системе;
* Разработка проекта программной системы
* Реализация программного обеспечения

Разработка проекта программного продукта была выполнена на языке UML. Программное обеспечение было реализовано на языке C# в виде веб-приложения, построенного на архитектуре MVC на базе программной платформы Microsoft .NET.

Разработанная система позволяет:

* Создавать и поддерживать справочники учеников и преподавателей;
* Формировать курсы, темы и заданиях в них;
* Вносить ответы на задания;
* Просматривать справочник курсов

Данная программная система может быть использована образовательными учреждениями с целью электронного представления данных учета знаний учащихся, составления и поддержки справочников курсов, а также контроля справочников студентов, групп и персонала.

* 1. **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Фримен А., Сандерсон С. ASP.NET MVC 4 Framework с примерами на C# для профессионалов. 4-е издание. Вильямс, 2014
2. Гради Б., Рамбо Дж., Якобсон Ив., Язык UML. Руководство пользователя, ДМК Пресс, 2007 г.
3. Гольцман В. MySQL 5.0 Библиотека программиста. СПБ.:Питер, 2010
4. Фримен А., Джозеф C. Раттц-мл - LINQ. Язык интегрированных запросов в C# 2010 для профессионалов Вильямс, 2011

**ПРИЛОЖЕНИЕ А Исходный код программной системы**

**Модели:**

**Course.cs:**

using System.Collections.Generic;

using Education.Models;

namespace Education

{

public class Course

{

public int CourseId { get; set; }

public string Name { get; set; }

public string Duration {get; set;}

public string Description {get; set;}

}

}

**Task.cs:**

namespace Education.Models

{

public class Task

{

public int TaskId {get;set;}

public string TaskAnswer {get;set;}

public string TaskQuestion {get;set;}

public int ThemeId {get;set;}

}

}

**Theme.cs:**

using System.Collections;

using System.Collections.Generic;

namespace Education.Models

{

public class Theme

{

public int ThemeID {get; set;}

public string ThemeName {get;set;}

public int CourseId{get;set;}

}

}

**Credentials.cs:**

using System.ComponentModel.DataAnnotations;

using Microsoft.AspNetCore.Identity;

namespace Education.Models

{

public class Credentials {

[Required]

[Display(Name = "User")]

public IdentityUser User { get; set; }

[Required]

[StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]

[DataType(DataType.Password)]

[Display(Name = "Password")]

public string Password { get; set; }

[EmailAddress]

[Display(Name = "Email")]

public string Email { get; set; }

[Required]

[Display(Name = "RoleId")]

public string RoleId { get; set; }

[Required]

[Display(Name = "OldRoleId")]

public string OldRoleId { get; set; }

}

}

**Interface:  
Crud.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Education.Models;

using Microsoft.AspNetCore.Mvc;

namespace Education.Models

{

public interface Crud<T> : IDisposable

{

List<T> getAll();

T Get(int id);

T Create (T t);

T Update (T t);

int Delete(int id);

}

}

**CoureRepositoryImpl.cs**:

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Web;

//using System.Data.Entity;

using Education.DAL;

using Education.Models;

using Microsoft.EntityFrameworkCore;

//using Microsoft.EntityFrameworkCore;

namespace Education.Models.Courses

{

public class CourseRepositoryImpl : Crud<Course>

{

private EducationContext db;

public CourseRepositoryImpl(EducationContext context)

{

this.db = context;

}

public List<Course> getAll()

{

try

{

//return db.Courses.Include(c => c.Themes).ThenInclude(t => t.Tasks).ToList();

return db.Courses.ToList();

} catch

{

throw new NullReferenceException();

}

}

public Course Get(int id)

{

return db.Courses.Find(id);

}

public Course Create (Course course)

{

try

{

db.Courses.Add(course);

db.SaveChanges();

return course;

}

catch

{

throw;

}

}

public Course Update(Course course)

{

try

{

if (course != null)

db.Entry(course).State=EntityState.Modified;

// db.Entry(course).State =

// Microsoft.EntityFrameworkCore.EntityState.Modified;

db.SaveChanges();

return course;

} catch

{

throw;

}

}

public int Delete(int id)

{

try

{

Course course = db.Courses.Find(id);

if (course != null)

db.Courses.Remove(course);

db.SaveChanges();

return 1;

} catch

{

throw;

}

}

private bool disposed = false;

public virtual void Dispose(bool disposing)

{

if (!this.disposed)

if (disposing)

db.Dispose();

this.disposed = true;

}

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

}

}

**AccountController.cs:**

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Authorization;

using System.Threading.Tasks;

using System.Collections;

using JWT;

using JWT.Serializers;

using JWT.Algorithms;

using Microsoft.Extensions.Options;

using System;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.AspNetCore.Mvc;

using Education.Models;

using System.Collections.Generic;

using System.Linq;

namespace Education.Controllers

{

public class AccountController : Controller

{

private readonly UserManager<IdentityUser> \_userManager;

private readonly SignInManager<IdentityUser> \_signInManager;

private readonly JWTSettings \_options;

public AccountController(UserManager<IdentityUser> userManager,

SignInManager<IdentityUser> signInManager,

IOptions<JWTSettings> optionsAccessor)

{

\_userManager = userManager;

\_signInManager = signInManager;

\_options = optionsAccessor.Value;

}

[HttpPost]

[Route("api/Auth/Register")]

public async Task<IActionResult> Register([FromBody] Credentials Credentials)

{

if (ModelState.IsValid)

{

var user = new IdentityUser { UserName = Credentials.Email, Email = Credentials.Email };

var result = await \_userManager.CreateAsync(user, Credentials.Password);

if (result.Succeeded)

{

await \_signInManager.SignInAsync(user, isPersistent: false);

return new JsonResult(new Dictionary<string, object>

{

{ "access\_token", GetAccessToken(Credentials.Email) },

{ "id\_token", GetIdToken(user) }

});

}

return Errors(result);

}

return Error("Unexpected error");

}

[HttpPost("sign-in")]

[Route("api/Auth/sign-in")]

public async Task<IActionResult> SignIn([FromBody] Credentials Credentials)

{

if (ModelState.IsValid)

{

var result = await \_signInManager.PasswordSignInAsync(Credentials.Email, Credentials.Password, false, false);

if (result.Succeeded)

{

var user = await \_userManager.FindByEmailAsync(Credentials.Email);

return new JsonResult(new Dictionary<string, object>

{

{ "access\_token", GetAccessToken(Credentials.Email) },

{ "id\_token", GetIdToken(user) }

});

}

return new JsonResult("Unable to sign in") { StatusCode = 401 };

}

return Error("Unexpected error");

}

private string GetIdToken(IdentityUser user)

{

var payload = new Dictionary<string, object>

{

{ "id", user.Id },

{ "sub", user.Email },

{ "email", user.Email },

{ "emailConfirmed", user.EmailConfirmed },

};

return GetToken(payload);

}

private string GetAccessToken(string Email)

{

var payload = new Dictionary<string, object>

{

{ "sub", Email },

{ "email", Email }

};

return GetToken(payload);

}

private string GetToken(Dictionary<string, object> payload)

{

var secret = \_options.SecretKey;

payload.Add("iss", \_options.Issuer);

payload.Add("aud", \_options.Audience);

payload.Add("nbf", ConvertToUnixTimestamp(DateTime.Now));

payload.Add("iat", ConvertToUnixTimestamp(DateTime.Now));

payload.Add("exp", ConvertToUnixTimestamp(DateTime.Now.AddDays(7)));

IJwtAlgorithm algorithm = new HMACSHA256Algorithm();

IJsonSerializer serializer = new JsonNetSerializer();

IBase64UrlEncoder urlEncoder = new JwtBase64UrlEncoder();

IJwtEncoder encoder = new JwtEncoder(algorithm, serializer, urlEncoder);

return encoder.Encode(payload, secret);

}

private JsonResult Errors(IdentityResult result)

{

var items = result.Errors

.Select(x => x.Description)

.ToArray();

return new JsonResult(items) { StatusCode = 400 };

}

private JsonResult Error(string message)

{

return new JsonResult(message) { StatusCode = 400 };

}

private static double ConvertToUnixTimestamp(DateTime date)

{

DateTime origin = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);

TimeSpan diff = date.ToUniversalTime() - origin;

return Math.Floor(diff.TotalSeconds);

}

}

}

**CourseController.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Education.Models.Courses;

using Education.DAL;

using Microsoft.AspNetCore.Mvc;

namespace Education.Controllers

{

public class CourseController : Controller

{

private readonly EducationContext \_context;

private CourseRepositoryImpl courseRepositoryImpl;

public CourseController(EducationContext context)

{

\_context = context;

courseRepositoryImpl = new CourseRepositoryImpl(\_context);

}

[HttpGet]

[Route("api/course/getall")]

public ActionResult GetAll()

{

//IEnumerable<Course> list = courseRepositoryImplgetAllCourses();

return Ok(new

{

courses = courseRepositoryImpl.getAll()

});

}

[HttpPost]

[Route("api/course/create")]

public Course create([FromBody]Course course)

{

return courseRepositoryImpl.Create(course);

}

[HttpGet]

[Route("api/course/details/{id}")]

public Course details(int id)

{

return courseRepositoryImpl.Get(id);

}

[HttpPut]

[Route("api/course/edit/{id}")]

public Course Edit([FromBody]Course course)

{

return courseRepositoryImpl.Update(course);

}

[HttpDelete]

[Route("api/course/delete/{id}")]

public int Delete(int id)

{

return courseRepositoryImpl.Delete(id);

}

[HttpGet]

[Route("api/course/getcourselist")]

public IEnumerable<Course> Details()

{

return courseRepositoryImpl.getAll();

}

}

}

**RoleController.cs**:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Education.Models.Users;

using Education.DAL;

using Microsoft.AspNetCore.Mvc;

using Education.Models;

using Education.Models.Themes;

using Education.Models.Roles;

using Microsoft.AspNetCore.Identity;

namespace Education.Controllers

{

public class RoleController : Controller

{

private readonly RoleManager<IdentityRole> \_roleManager;

private readonly EducationContext \_context;

public RoleController(EducationContext context, RoleManager<IdentityRole> roleManager) {

\_roleManager = roleManager;

\_context = context;

}

[HttpGet]

[Route("api/Role/GetRoles")]

public IActionResult GetRoles()

{

return Ok(new

{

roles = \_roleManager.Roles.ToList()

});

}

[HttpPost]

[Route("api/Role/Create")]

public async Task<IActionResult> Create([FromBody] string roleName)

{

if (!string.IsNullOrEmpty(roleName))

{

IdentityResult result = await \_roleManager.CreateAsync(new IdentityRole(roleName));

if (result.Succeeded)

{

return Ok( new {

roles = \_roleManager.Roles.ToList()

});

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return Ok();

}

[HttpDelete]

[Route("api/Role/Delete/{id}")]

public async Task<IActionResult> Delete([FromRoute]string id)

{

IdentityRole role = await \_roleManager.FindByIdAsync(id);

if (role != null)

{

IdentityResult result = await \_roleManager.DeleteAsync(role);

}

return Ok(new {

roles = \_roleManager.Roles.ToList()

});

}

}

}

**TaskController.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Education.Models.Users;

using Education.DAL;

using Microsoft.AspNetCore.Mvc;

using Education.Models;

using Education.Models.Themes;

using Education.Models.Tasks;

namespace Education.Controllers

{

public class TaskController : Controller

{

private readonly EducationContext \_context;

private TaskRepositoryImpl taskRepositoryImpl;

public TaskController(EducationContext context) {

\_context = context;

taskRepositoryImpl = new TaskRepositoryImpl(\_context);

}

[HttpGet]

[Route("api/task/getall")]

public ActionResult GetAll() {

return Ok(new {

tasks = taskRepositoryImpl.getAll()

});

}

[HttpPost]

[Route("api/task/create")]

public Education.Models.Task create([FromBody]Education.Models.Task task) {

return taskRepositoryImpl.Create(task);

}

[HttpPut]

[Route("api/task/edit/{id}")]

public Education.Models.Task Edit([FromBody] Education.Models.Task task) {

return taskRepositoryImpl.Update(task);

}

[HttpDelete]

[Route("api/task/delete/{id}")]

public int Delete(int id) {

return taskRepositoryImpl.Delete(id);

}

}

}

**ThemeController.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Education.Models.Users;

using Education.DAL;

using Microsoft.AspNetCore.Mvc;

using Education.Models;

using Education.Models.Themes;

namespace Education.Controllers

{

public class ThemeController : Controller

{

private readonly EducationContext \_context;

private ThemeRepositoryImpl themeRepositoryImpl;

public ThemeController(EducationContext context) {

\_context = context;

themeRepositoryImpl = new ThemeRepositoryImpl(\_context);

}

[HttpGet]

[Route("api/theme/getall")]

public ActionResult GetAll() {

return Ok(new {

themes = themeRepositoryImpl.getAll()

});

}

[HttpPost]

[Route("api/theme/create")]

public Theme create([FromBody]Theme theme) {

return themeRepositoryImpl.Create(theme);

}

[HttpPut]

[Route("api/theme/edit/{id}")]

public Theme Edit([FromBody]Theme theme) {

return themeRepositoryImpl.Update(theme);

}

[HttpDelete]

[Route("api/theme/delete/{id}")]

public int Delete(int id) {

return themeRepositoryImpl.Delete(id);

}

}

}

**UserController.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Identity;

using Microsoft.EntityFrameworkCore;

using Education.DAL;

using Education.Models;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.Extensions.Logging;

using Microsoft.AspNetCore.Authorization;

using Microsoft.Extensions.Options;

using System;

namespace Education.Controllers

{

public class UserController : Controller

{

private readonly EducationContext \_context;

private readonly UserManager<IdentityUser> \_userManager;

private readonly ILogger<UserController> \_logger;

public UserController(EducationContext context, UserManager<IdentityUser> userManager) {

\_context = context;

\_userManager = userManager;

}

[HttpGet]

[Route("api/User/GetUsers")]

public IActionResult GetUsers()

{

\_logger.LogInformation("Get Users");

return Ok(new

{

users = \_userManager.Users

});

}

[HttpGet]

[Route("api/User/GetUserRoles")]

public IActionResult GetUserRoles()

{

\_logger.LogInformation("Get Users Roles");

return Ok(new

{

userRoles = \_context.Roles.ToList()

});

}

[HttpDelete]

[Route("api/User/Delete/{id}")]

async public Task<IActionResult> Delete([FromRoute] string id)

{

var result = await \_userManager.DeleteAsync(await \_userManager.FindByIdAsync(id));

if (result != null)

return Ok( new {

users = \_userManager.Users

});

else

return Error("Unexpected error");

}

[HttpPut]

[Route("api/User/Update")]

async public Task<IActionResult> Update([FromBody] Credentials credentials) {

if (!string.IsNullOrEmpty(credentials.Password))

{

string hashPass = \_userManager.PasswordHasher.HashPassword(credentials.User, credentials.Password);

credentials.User.PasswordHash = hashPass;

}

\_context.Entry(credentials.User).State = EntityState.Modified;

\_context.SaveChanges();

if (string.IsNullOrEmpty(credentials.OldRoleId))

{

await \_userManager.AddToRoleAsync(credentials.User, \_context.Roles.Find(credentials.RoleId).Name);

return Ok();

}

if (!string.IsNullOrEmpty(credentials.RoleId) && !string.IsNullOrEmpty(credentials.OldRoleId))

{

await \_userManager.RemoveFromRoleAsync(credentials.User, \_context.Roles.Find(credentials.OldRoleId).Name);

await \_userManager.AddToRoleAsync(credentials.User, \_context.Roles.Find(credentials.RoleId).Name);

}

return Ok();

}

private JsonResult Error(string message)

{

return new JsonResult(message) { StatusCode = 400 };

}

}

}

**DAL:**

**EducationContext.cs:**

using Microsoft.EntityFrameworkCore;

using Education.Models;

namespace Education.DAL

{

public class EducationContext : DbContext

{

public EducationContext(DbContextOptions<EducationContext> options) : base(options) { }

public DbSet<Course> Courses { get; set; }

public DbSet<User> Users {get; set;}

public DbSet<Role> Roles {get; set;}

public DbSet<Theme> Themes {get;set;}

public DbSet<Task> Tasks {get;set;}

}

}

**UserDbContext.cs**:

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

namespace Education.Models

{

public class UserDbContext : IdentityDbContext<IdentityUser>

{

public UserDbContext(DbContextOptions<UserDbContext> options) : base(options) {

}

}

}

**CourseRepository:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace DistansEducation.Models

{

public class CourseRepository : IRepository

{

private CourseContext db;

public CourseRepository()

{

this.db = new CourseContext();

}

public List<Course> GetCoursesList()

{

return db.Courses.ToList();

}

public Course GetCourse(int id)

{

return db.Courses.Find(id);

}

public void Create (Course course)

{

db.Courses.Add(course);

}

public void Update(Course course)

{

db.Entry(course).State = System.Data.Entity.EntityState.Modified;

}

public void Delete(int id)

{

Course course = db.Courses.Find(id);

if (course != null)

db.Courses.Remove(course);

}

public void Save()

{

db.SaveChanges();

}

private bool disposed = false;

public virtual void Dispose(bool disposing)

{

if (!this.disposed)

if (disposing)

db.Dispose();

this.disposed = true;

}

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

}

}

**IRepository:**

namespace DistansEducation.Models

{

public interface IRepository : IDisposable

{

List<Course> GetCoursesList();

Course GetCourse(int id);

void Create(Course course);

void Update(Course course);

void Delete(int id);

void Save();

}

}

**HomeController:**

using DistansEducation.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace DistansEducation.Controllers

{

public class HomeController : Controller

{

IRepository repository;

public HomeController (IRepository repository) => this.repository = repository;

public HomeController() => this.repository = new CourseRepository();

CourseContext db = new CourseContext();

public ActionResult Index()

{

var model = repository.GetCoursesList();

return View(model);

}

public ActionResult About()

{

ViewBag.Message = "Your application description page.";

return View("About");

}

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

return View();

}

protected override void Dispose(bool disposing)

{

repository.Dispose();

base.Dispose(disposing);

}

}

}

**HomeControllerTest:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Web.Mvc;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using DistansEducation;

using DistansEducation.Controllers;

using DistansEducation.Models;

using Moq;

namespace DistansEducation.Tests.Controllers

{

[TestClass]

public class HomeControllerTest

{

private HomeController controller;

private ViewResult result;

[TestMethod]

public void Index()

{

controller = new HomeController();

result = controller.Index() as ViewResult;

}

[TestMethod]

public void aboutPage()

{

controller = new HomeController();

result = controller.About() as ViewResult;

Assert.AreEqual("About", result.ViewName);

}

[TestMethod]

public void About()

{

controller = new HomeController();

result = controller.About() as ViewResult;

Assert.AreEqual("Your application description page.", result.ViewBag.Message);

}

[TestMethod]

public void Contact()

{

controller = new HomeController();

result = controller.Contact() as ViewResult;

Assert.IsNotNull(result);

}

[TestMethod]

public void IndexViewModelNotNull()

{

var moq = new Mock<IRepository>();

moq.Setup(a => a.GetCoursesList()).Returns(new List<Course>());

controller = new HomeController(moq.Object);

result = controller.Index() as ViewResult;

Assert.IsNotNull(result.Model);

}

}

}

**Frontend:**

**App.js:**

import React from 'react';

import { Route } from 'react-router';

import Layout from './components/Layout';

//import Home from './components/Home';

//import Counter from './components/Counter';

import FetchData from './components/FetchData';

import User from './components/Users/User';

import Course from './components/Courses/Course';

import Role from './components/Roles/Role';

import Task from './components/Courses/Task';

import SignIn from './components/Login/SignIn';

import Themes from './components/Courses/Themes';

import SignUp from './components/Login/Signup';

export default () => (

<Layout>

<Route exact path='/tasks' component={Task} />

<Route path='/users' component={User}/>

<Route path='/courses' component={Course} />

<Route path='/fetchdata/:startDateIndex?' component={FetchData} />

<Route path='/roles' component={Role}/>

<Route path='/themes' component={Themes}/>

<Route path='/signin' component={SignIn}/>

<Route path='/login' component={SignUp}/>

</Layout>

);

**NavMenu.js:**

import React from 'react';

import { Link } from 'react-router-dom';

import { Glyphicon, Nav, Navbar, NavItem } from 'react-bootstrap';

import { LinkContainer } from 'react-router-bootstrap';

import './NavMenu.css';

export default props => (

<Navbar inverse fixedTop fluid collapseOnSelect>

<Navbar.Header>

<Navbar.Brand>

<Link to={'/'}>Education</Link>

</Navbar.Brand>

<Navbar.Toggle />

</Navbar.Header>

<Navbar.Collapse>

<Nav>

<LinkContainer to={'/courses'}>

<NavItem>

<Glyphicon glyph='education' /> Мои курсы

</NavItem>

</LinkContainer>

<LinkContainer to={'/themes'}>

<NavItem>

<Glyphicon glyph='education' /> Темы

</NavItem>

</LinkContainer>

<LinkContainer to={'/tasks'} exact>

<NavItem>

<Glyphicon glyph='th-list' /> Задания

</NavItem>

</LinkContainer>

<LinkContainer to={'/users'} exact>

<NavItem>

<Glyphicon glyph='th-list' /> Пользователи

</NavItem>

</LinkContainer>

<LinkContainer to={'/signin'} exact>

<NavItem>

<Glyphicon glyph='th-list'></Glyphicon> Войти в систему

</NavItem>

</LinkContainer>

<LinkContainer to={'/login'} exact>

<NavItem>

<Glyphicon glyph='th-list'></Glyphicon> Регистрация

</NavItem>

</LinkContainer>

<LinkContainer to={'/roles'}>

<NavItem>

<Glyphicon glyph='th-list'/> Роли

</NavItem>

</LinkContainer>

</Nav>

</Navbar.Collapse>

</Navbar>

);

**User.js:**

import React, { Component } from "react";

import {

Button,

Badge,

FormGroup,

InputGroup,

FormControl

} from "react-bootstrap";

import ModalUserEdit from "./ModalUserEdit";

export default class User extends Component {

displayName = User.name;

constructor(props, context) {

super(props, context);

this.state = {

users: null,

roles: null,

role: null,

userRoles: null,

showModal: false,

action: "",

actions: ["add", "update"],

searchText: "",

credentials: null

};

}

componentDidMount = () => {

fetch("api/Role/GetRoles")

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ roles: data.roles });

});

this.updateView();

};

updateView = () => {

fetch("api/User/GetUsers")

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ users: data.users });

});

fetch("api/User/GetUserRoles")

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ userRoles: data.userRoles });

});

};

render = () => {

let contents =

this.state.roles && this.state.users && this.state.userRoles ? (

this.renderUsersTable(

this.state.users,

this.state.roles,

this.state.userRoles

)

) : (

<div className="loader">Loading...</div>

);

let badge = this.state.users ? (

<Badge>{this.state.users.length}</Badge>

) : (

<div />

);

return (

<div>

<h1>Users of YW-M Stock system {badge}</h1>

<FormGroup>

<InputGroup>

<FormControl

type="text"

onChange={event =>

this.setState({ searchText: event.target.value })

}

/>

<InputGroup.Button>

<Button onClick={this.handleSearch}>Search</Button>

</InputGroup.Button>

</InputGroup>

</FormGroup>

<Button onClick={this.handleAdd}>Add new user</Button>

{contents}

{this.state.credentials && this.state.roles && (

<ModalUserEdit

show={this.state.showModal}

roles={this.state.roles}

userRoles={this.state.userRoles}

action={this.state.action}

actions={this.state.actions}

close={this.handleClose}

save={this.handleSave}

credentials={this.state.credentials}

user={this.state.user}

/>

)}

</div>

);

};

renderUsersTable = users => {

return (

<table className="table">

<thead>

<tr>

<th>Name</th>

<th>E-mail</th>

<th>Role</th>

<th />

</tr>

</thead>

<tbody>

{users.map(user => (

<tr key={user.id}>

<td>{user.userName}</td>

<td>{user.email}</td>

{this.setRole(user.id)}

<td>

<Button bsStyle="link" onClick={() => this.handleEdit(user.id)}>

Edit

</Button>

<Button

bsStyle="link"

onClick={() => this.handleDelete(user.id)}

>

Delete

</Button>

</td>

</tr>

))}

</tbody>

</table>

);

};

setRole = userId => {

const userRole = this.state.userRoles.find(ur => ur.userId === userId);

const roleName = userRole ? this.state.roles.find(

r => r.id === userRole.roleId

).name : ""

return <td>{roleName}</td>;

};

handleClose = () => {

this.setState({ showModal: false });

};

handleSearch = () => {

fetch("api/User/GetUsers")

.then(response => response.json())

.then(data => {

this.setState({

users: data.users.filter(

rec =>

rec.userName

.toLowerCase()

.includes(this.state.searchText.toLowerCase()) ||

rec.email

.toLowerCase()

.includes(this.state.searchText.toLowerCase())

)

});

});

};

handleAdd = () => {

this.setState({

showModal: true,

action: this.state.actions[0],

credentials: {

user: { userName: "", email: "" },

password: "",

roleId: 0,

oldRoleId: 0

}

});

};

handleDelete = id => {

if (!window.confirm("Are you sure you wish to delete this item?")) return;

fetch("api/User/Delete/" + id, { method: "delete" })

.then(response => response.json())

.then(data => {

this.setState({ users: data.users });

});

};

handleEdit = id => {

const u = this.state.users.find(user => user.id === id);

const userRole = this.state.userRoles.find(ur => ur.userId === id);

const roleId = userRole ? this.state.roles.find(

r => r.id === userRole.roleId

).id : null

this.setState(

{

credentials: {

user: u,

password: "",

roleId: roleId,

oldRoleId: roleId

},

action: this.state.actions[1]

},

() => this.setState({ showModal: true })

);

};

}

**UpdateUserEdit.js:**

import React, { Component } from "react";

import {

Modal,

Button,

FormControl,

Form,

ControlLabel,

FormGroup,

Col,

HelpBlock

} from "react-bootstrap";

export default class ModalUserEdit extends Component {

constructor(props) {

super(props);

this.state = {

showModal: true

};

}

componentDidMount = () => {

const { credentials, roles, userRoles, actions, action } = this.props;

this.setState({

credentials: credentials,

roles: roles,

userRoles: userRoles,

actions: actions,

action: action,

});

console.log(this.state.credentials);

};

handleFormChange = (value, field) => {

this.setState(prevState => ({

credentials: {

...prevState.credentials,

[field]: value

}

}));

console.log(this.state.credentials);

};

handleFormUserChange = (value, field) => {

this.setState(prevState => ({

credentials: {

...prevState.credentials,

user: {

...prevState.credentials.user,

[field]: value

}

}

}));

console.log(this.state.credentials);

}

handleSave = () => {

if (this.state.action === this.state.actions[0]) {

fetch("api/Auth/Register", {

method: "post",

headers: {

Accept: "application/json",

"Content-Type": "application/json"

},

body: JSON.stringify(this.state.credentials)

});

}

if (this.state.action === this.state.actions[1]) {

fetch("api/User/Update", {

method: "put",

headers: {

Accept: "application/json",

"Content-Type": "application/json"

},

body: JSON.stringify(this.state.credentials)

});

}

};

render = () => {

const { show, roles, close, action } = this.props;

let help =

this.state.actions && action === this.state.actions[1]

? "You can change your pass by typing new above"

: "";

return (

<div className="static-modal">

<Modal show={show} onHide={close}>

<Modal.Header>

<Modal.Title>User edition</Modal.Title>

</Modal.Header>

<Modal.Body>

<Form horizontal>

<FormGroup controlId="formName">

<Col componentClass={ControlLabel} sm={2}>

Name

</Col>

<Col sm={9}>

<FormControl

type="text"

placeholder="Enter name"

onChange={event => {

this.handleFormUserChange(event.target.value, "userName");

}}

defaultValue={this.props.credentials.user.userName}

/>

</Col>

</FormGroup>

<FormGroup controlId="formEmail">

<Col componentClass={ControlLabel} sm={2}>

Email

</Col>

<Col sm={9}>

<FormControl

type="email"

placeholder="Enter E-mail"

onChange={event => {

this.handleFormUserChange(event.target.value, "email");

}}

defaultValue={this.props.credentials.user.email}

/>

</Col>

</FormGroup>

<FormGroup controlId="formRole">

<Col componentClass={ControlLabel} sm={2}>

Role

</Col>

<Col sm={9}>

<FormControl

componentClass="select"

placeholder="select"

onChange={event => {

this.handleFormChange(event.target.value, "roleId");

}}

>

{roles.map(role => (

<option

key={role.id}

value={role.id}

selected={role.id === this.props.credentials.roleId}

>

{role.name}

</option>

))}

</FormControl>

</Col>

</FormGroup>

<FormGroup controlId="formEmail">

<Col componentClass={ControlLabel} sm={2}>

Pass

</Col>

<Col sm={9}>

<FormControl

type="password"

placeholder="Enter new password"

onChange={event => {

this.handleFormChange(event.target.value, "password");

}}

defaultValue={this.props.credentials.password}

/>

<HelpBlock>{help}</HelpBlock>

</Col>

</FormGroup>

</Form>

</Modal.Body>

<Modal.Footer>

<Button onClick={close}>Close</Button>

<Button onClick={() => this.handleSave()} bsStyle="primary">

Save changes

</Button>

</Modal.Footer>

</Modal>

</div>

);

};

}

**AddUserModal.js:**

import React, { Component } from 'react';

import {

Button, Table,

Badge, ButtonToolbar, Modal,Form, ControlLabel,

FormGroup,

InputGroup,

FormControl

} from "react-bootstrap";

export default class AddUserModal extends Component {

constructor(props, context) {

super(props, context);

this.handleAdd = this.handleAdd.bind(this);

this.handleFormChange = this.handleFormChange.bind(this);

this.toggleCreationModel = this.toggleCreationModel.bind(this);

this.state = {

users: null,

user: {

Name: "",

Surname: "",

Email: "",

RoleID: ""

},

showCreationModal: false

}

}

toggleCreationModel() {

const {showCreationModal} = this.state;

this.setState({showCreationModal: !showCreationModal})

}

handleFormChange = (value, field) => {

//console.log("onChange", value, field)

this.setState(prevState => ({

user: {

...prevState.user,

[field]: value

}

}));

console.log("state.user",this.state.user);

}

handleAdd = () => {

console.log("user",this.state.user);

fetch("api/user/create", {

method: "post",

headers: {

Accept: "application/json",

"Content-Type": "application/json"

},

body: JSON.stringify(this.state.user)

})

.then(response => response.json())

.then(data => {

console.log("after",data);

this.setState({ users: data.users });

})

}

render = () => {

return (

<div>

<ButtonToolbar>

<Button bsStyle="primary" onClick={this.toggleCreationModel}>

Add new user

</Button>

<Modal

show={this.state.showCreationModal}

onHide={this.toggleCreationModel}

dialogClassName="custom-modal">

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-lg">

User creation

</Modal.Title>

</Modal.Header>

<Modal.Body>

{" "}

<Form >

<FormGroup>

<ControlLabel>Name</ControlLabel>{" "}

<FormControl

type="text"

placeholder="Enter name"

onChange={event => this.handleFormChange(event.target.value, "Name")}

/>{" "}

</FormGroup>{" "}

<FormGroup>

<ControlLabel>Surname</ControlLabel>{" "}

<FormControl

type="text"

placeholder="Enter Surname"

onChange={event => this.handleFormChange(event.target.value, "Surname")}

/>

</FormGroup>{" "}

<FormGroup>

<ControlLabel>Email</ControlLabel>{" "}

<FormControl

type="text"

placeholder="Enter email"

onChange={event => this.handleFormChange(event.target.value, "Email")}

/>{" "}

</FormGroup>{" "}

<FormGroup>

<ControlLabel>RoleId</ControlLabel>

<FormControl

type="text"

placeholder="enter roleID"

onChange={event => this.handleFormChange(event.target.value, "RoleID")}

/>{" "}

</FormGroup>

<Button onClick={this.handleAdd} >

New user

</Button>

</Form>

<br />

</Modal.Body>

<Modal.Footer>

<Button onClick={this.toggleCreationModel}>Close</Button>

</Modal.Footer>

</Modal>

</ButtonToolbar>

</div>

)

}

}

**Role.js:**

import React, { Component } from "react";

import {

Button,

Badge,

FormGroup,

InputGroup,

FormControl

} from "react-bootstrap";

export default class Role extends Component {

displayName = Role.name;

constructor(props, context) {

super(props, context);

this.handleDelete = this.handleDelete.bind(this);

this.handleAdd = this.handleAdd.bind(this);

this.state = {

roles: null,

roleName: ""

};

}

componentDidMount = () => {

fetch("api/Role/GetRoles")

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ roles: data.roles });

});

};

render = () => {

let contents = this.state.roles ? (

this.renderRolesTable(this.state.roles)

) : (

<div >Loading...</div>

);

return <div>{contents}</div>;

};

handleDelete = id => {

if (!window.confirm("Are you sure you wish to delete this item?")) return;

fetch("api/Role/Delete/" + id, { method: "delete" })

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ roles: data.roles });

});

};

handleAdd = () => {

fetch("api/Role/Create", {

method: "post",

headers: {

Accept: "application/json",

"Content-Type": "application/json"

},

body: JSON.stringify(this.state.roleName)})

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ roles: data.roles });

})

};

renderRolesTable = roles => {

return (

<div>

<h1>Справочник ролей</h1>

<FormGroup>

<InputGroup>

<FormControl

type="text"

onChange={event =>

this.setState({ roleName: event.target.value })

}

/>

<InputGroup.Button>

<Button onClick={this.handleAdd}>Добавить роль</Button>

</InputGroup.Button>

</InputGroup>

</FormGroup>

<table className="table">

<thead>

<tr>

<th>Name</th>

<th />

</tr>

</thead>

<tbody>

{roles.map(role => (

<tr key={role.id}>

<td>{role.name}</td>

<td>

<Button

bsStyle="link"

onClick={() => this.handleDelete(role.id)}

>

Delete

</Button>

</td>

</tr>

))}

</tbody>

</table>

</div>

);

};

}

**SignUp.js:**

import React, { Component } from "react";

import {

FormGroup,

ControlLabel,

FormControl,

Form,

Col,

Button,

Tooltip,

OverlayTrigger

} from "react-bootstrap";

export default class SignUp extends Component {

displayName = SignUp.name;

constructor(props) {

super(props);

this.state = { email: "", password: "" };

this.handleEmailChange = this.handleEmailChange.bind(this);

this.handlePasswordChange = this.handlePasswordChange.bind(this);

}

render = () => {

const tooltip = (

<Tooltip id="tooltip">

<strong>Please, use your corporative E-mail!</strong>

</Tooltip>

);

return (

<div>

<h1>Регистрация нового пользователя</h1>

<Form horizontal>

<OverlayTrigger placement="left" overlay={tooltip}>

<FormGroup controlId="formHorizontalEmail">

<Col componentClass={ControlLabel} sm={1}>

E-mail

</Col>

<Col sm={5}>

<FormControl type="email" placeholder="example@email.com" onChange={event => this.handleEmailChange(event)}/>

</Col>

</FormGroup>

</OverlayTrigger>

<FormGroup controlId="formHorizontalPassword">

<Col componentClass={ControlLabel} sm={1}>

Пароль

</Col>

<Col sm={5}>

<FormControl type="password" placeholder="Пароль" onChange={event => this.handlePasswordChange(event)}/>

</Col>

</FormGroup>

<FormGroup>

<Col smOffset={1} sm={10}>

<Button onClick={this.handleLoginClick} bsStyle="primary">

Зарегистрироваться

</Button>

</Col>

</FormGroup>

</Form>

</div>

);

}

handleEmailChange = (evt) => {

console.log(this.state);

this.setState({

email: evt.target.value

});

}

handlePasswordChange = (evt) => {

console.log(this.state);

this.setState({

password: evt.target.value

});

}

handleLoginClick = () => {

console.log(this.state);

fetch("api/Auth/Register", {

method: "post",

headers: {

Accept: "application/json",

"Content-Type": "application/json"

},

body: JSON.stringify(this.state)

})

}

}

**SignIn.js:**

import React, { Component } from "react";

import {

FormGroup,

ControlLabel,

FormControl,

Form,

Col,

Button,

Tooltip,

OverlayTrigger

} from "react-bootstrap";

import '../Courses/styles.css';

export default class SignIn extends Component {

displayName = SignIn.name;

constructor(props) {

super(props);

this.state = { userName: "", email: "", password: "" };

this.handleEmailChange = this.handleEmailChange.bind(this);

this.handlePasswordChange = this.handlePasswordChange.bind(this);

}

render = () => {

const tooltip = (

<Tooltip id="tooltip">

<strong>Please, use your corporative E-mail!</strong>

</Tooltip>

);

return (

<div className="my-block">

<h1>Войти в систему</h1>

<Form horizontal>

<FormGroup controlId="formHorizontalUserName">

<Col componentClass={ControlLabel} sm={1}>

Имя

</Col>

<Col sm={5}>

<FormControl type="text" placeholder="Введите ваш логин" onChange={event => this.setState({

userName: event.target.value

})}/>

</Col>

</FormGroup>

<OverlayTrigger placement="left" overlay={tooltip}>

<FormGroup controlId="formHorizontalEmail">

<Col componentClass={ControlLabel} sm={1}>

E-mail

</Col>

<Col sm={5}>

<FormControl type="email" placeholder="example@email.com" onChange={event => this.handleEmailChange(event)}/>

</Col>

</FormGroup>

</OverlayTrigger>

<FormGroup controlId="formHorizontalPassword">

<Col componentClass={ControlLabel} sm={1}>

Пароль

</Col>

<Col sm={5}>

<FormControl type="password" placeholder="Пароль" onChange={event => this.handlePasswordChange(event)}/>

</Col>

</FormGroup>

<FormGroup>

<Col smOffset={1} sm={10}>

<Button onClick={this.handleLoginClick} bsStyle="primary">

Войти в систему

</Button>

</Col>

</FormGroup>

</Form>

</div>

);

}

handleEmailChange = (evt) => {

console.log(this.state);

this.setState({

email: evt.target.value

});

}

handlePasswordChange = (evt) => {

console.log(this.state);

this.setState({

password: evt.target.value

});

}

handleLoginClick = () => {

console.log(this.state);

fetch("api/Auth/sign-in", {

method: "post",

headers: {

Accept: "application/json",

"Content-Type": "application/json",

},

body: JSON.stringify(this.state)

})

.then(response => response.json())

.then(data => {

console.log(data);

localStorage.setItem('access\_token', data.access\_token);

localStorage.setItem('id\_token', data.id\_token);

});;

}

}

**Themes.js:**

import React, { Component } from 'react';

import {

Button, Modal, ButtonToolbar,

Form, ControlLabel,

FormGroup,

FormControl

} from "react-bootstrap";

import './styles.css';

export default class Themes extends Component {

constructor(props) {

super(props);

this.toggleUpdationModel = this.toggleUpdationModel.bind(this);

this.handleDelete = this.handleDelete.bind(this);

this.handleFormChange = this.handleFormChange.bind(this);

this.toggleCreationModel = this.toggleCreationModel.bind(this);

this.state = {

themes: null,

theme: {

ThemeName: "",

CourseId: ""

},

showCreationModal: false,

showUpdationModal: false

};

}

componentDidMount = () => {

fetch("api/theme/getall", { dataType: 'json' })

.then(response => response.json())

.then(data => {

console.log("themes", data);

this.setState({ themes: data.themes });

})

}

handleAdd = () => {

fetch("api/theme/create", {

method: "post",

headers: {

Accept: "application/json",

"Content-Type": "application/json"

},

body: JSON.stringify(this.state.theme)

})

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ themes: data.themes });

})

};

handleUpdate = (themeID) => {

let themeDTO = this.state.theme;

themeDTO = { ...themeDTO, themeID: themeID };

fetch("api/theme/edit/" + themeID, {

method: "put",

headers: {

Accept: "application/json",

"Content-Type": "application/json"

},

body: JSON.stringify(themeDTO)

})

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ themes: data.themes });

})

}

handleDelete = (themeID) => {

console.log("id", themeID);

if (!window.confirm("Вы уверены, что хотите удалить эту тему?")) return;

fetch("api/theme/delete/" + themeID, { method: "delete" })

.then(responce => responce.json())

.then(data => {

console.log(data);

this.setState({ themes: data.themes })

})

}

toggleUpdationModel() {

const { showUpdationModal } = this.state;

this.setState({ showUpdationModal: !showUpdationModal })

}

toggleCreationModel() {

const { showCreationModal } = this.state;

this.setState({ showCreationModal: !showCreationModal })

}

handleFormChange = (value, field) => {

console.log("onChange", value, field)

this.setState(prevState => ({

theme: {

...prevState.theme,

[field]: value

}

}));

console.log(this.state.theme);

}

renderThemeBlock = (theme) => {

return (

<div className="course" key={theme.themeID}>

{/\* <p>ThemeID: {theme.themeID}</p> \*/}

<p>Тема: {theme.themeName}</p>

{/\* <p>CourseId:{theme.courseId}</p> \*/}

<Button bsStyle="default" onClick={() => this.setState({showUpdationModal: theme.themeID})}>Изменить</Button>

<Modal

show={this.state.showUpdationModal === theme.themeID}

onHide={this.toggleUpdationModel}

dialogClassName="custom-modal">

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-lg">

Редактирование темы

</Modal.Title>

</Modal.Header>

<Modal.Body>

{" "}

<Form>

<FormGroup>

<ControlLabel>{theme.themeName}</ControlLabel>{" "}

<FormControl

type="text"

placeholder={theme.themeName}

onChange={event => this.handleFormChange(event.target.value, "ThemeName")}

/>{" "}

</FormGroup>{" "}

{/\* <FormGroup>

<ControlLabel>{theme.cousreId}</ControlLabel>{" "}

<FormControl

type="text"

placeholder={theme.courseId}

onChange={event => this.handleFormChange(event.target.value, "CourseId")}

/>{" "}

</FormGroup>{" "} \*/}

<Button onClick={() => this.handleUpdate(theme.themeID)}>Сохранить изменения</Button>

</Form>

<br />

</Modal.Body>

<Modal.Footer>

<Button onClick={this.toggleUpdationModel}>Закрыть</Button>

</Modal.Footer>

</Modal>

<Button bsStyle="default" onClick={() => this.handleDelete(theme.themeID)}>Удалить</Button>

</div>

)

}

render = () => {

let themes = this.state.themes ? (this.state.themes.map(theme => this.renderThemeBlock(theme))) : <div>Loading...</div>;

return (

<div>

<div className="course-flex">

{themes}

</div>

<Button bsStyle="primary" onClick={this.toggleCreationModel}>Добавить новую тему</Button>

<Modal

show={this.state.showCreationModal}

onHide={this.toggleCreationModel}

dialogClassName="custom-modal">

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-lg">

Theme creation

</Modal.Title>

</Modal.Header>

<Modal.Body>

{" "}

<Form inline>

<FormGroup>

<ControlLabel>Themename</ControlLabel>{" "}

<FormControl

type="text"

placeholder="Enter name of theme"

onChange={event => this.handleFormChange(event.target.value, "ThemeName")}

/>{" "}

</FormGroup>{" "}

<FormGroup>

<ControlLabel>CourseId</ControlLabel>{" "}

<FormControl

type="text"

placeholder="CourseId"

onChange={event => this.handleFormChange(event.target.value, "CourseId")} />

</FormGroup>{" "}

<Button onClick={this.handleAdd} >

Новая тема

</Button>

</Form>

<br />

</Modal.Body>

<Modal.Footer>

<Button onClick={this.toggleCreationModel}>Закрыть</Button>

</Modal.Footer>

</Modal>

</div>

)

}

}

Task.js:

import React, { Component } from 'react';

import './styles.css';

import {

Button, Modal, ButtonToolbar,

Form, ControlLabel,

FormGroup,

FormControl

} from "react-bootstrap";

import './styles.css';

export default class Task extends Component {

constructor(props, context) {

super(props, context);

this.handleAdd = this.handleAdd.bind(this);

this.toggleCreationModel = this.toggleCreationModel.bind(this);

this.handleFormChange = this.handleFormChange.bind(this);

this.handleDelete = this.handleDelete.bind(this);

this.handleUpdate = this.handleUpdate.bind(this);

this.toggleUpdationModel = this.toggleUpdationModel.bind(this);

this.state = {

tasks: null,

task: {

TaskQuestion: "",

ThemeId: "",

TaskAnswer: ""

},

showCreationModal: false,

showUpdationModal: false

}

}

componentDidMount = () => {

fetch("api/task/getall", { dataType: 'json' })

.then(response => response.json())

.then(data => {

this.setState({ tasks: data.tasks });

});

}

handleAdd = () => {

fetch("api/task/create", {

method: "post",

headers: {

Accept: "application/json",

"Content-Type": "application/json"

},

body: JSON.stringify(this.state.task)

})

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ tasks: data.tasks });

})

};

toggleCreationModel() {

const { showCreationModal } = this.state;

this.setState({ showCreationModal: !showCreationModal })

}

toggleUpdationModel() {

const { showUpdationModal } = this.state;

this.setState({ showUpdationModal: !showUpdationModal })

}

renderTaskBlock = (task) => {

return (

<div className="course-large" key={task.taskId}>

<p><strong>Задание </strong>{task.taskQuestion}</p>

<Button bsStyle="default" onClick={() => this.setState({showUpdationModal: task.taskId})}>Описание</Button>

{/\* <Button bsStyle="primary">Answer</Button> \*/}

<Modal

show={this.state.showUpdationModal === task.taskId}

onHide={this.toggleUpdationModel}

dialogClassName="custom-modal">

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-lg">

Описание

</Modal.Title>

</Modal.Header>

<Modal.Body>

{" "}

<Form>

<FormGroup>

<ControlLabel>{task.taskQuestion}</ControlLabel>{" "}

</FormGroup>{" "}

<FormGroup>

<ControlLabel>Добавить ответ</ControlLabel>{" "}

<FormControl

type="text"

placeholder={task.taskAnswer}

onChange={event => this.handleFormChange(event.target.value, "TaskAnswer")}

/>{" "}

</FormGroup>{" "}

<Button onClick={() => this.handleUpdate(task.taskId)}>Сохранить</Button>

</Form>

<br />

</Modal.Body>

<Modal.Footer>

<Button onClick={this.toggleUpdationModel}>Close</Button>

</Modal.Footer>

</Modal>

<Button bsStyle="default" onClick={() => this.handleDelete(task.taskId)}>Удалить</Button>

</div>

)

}

handleFormChange = (value, field) => {

console.log("onChange", value, field)

this.setState(prevState => ({

task: {

...prevState.task,

[field]: value

}

}));

console.log(this.state.task);

}

handleDelete = (taskId) => {

console.log("id", taskId);

if (!window.confirm("Are you sure you want to delete the task?")) return;

fetch("api/task/delete/" + taskId, { method: "delete" })

.then(responce => responce.json())

.then(data => {

console.log(data);

this.setState({ tasks: data.tasks })

})

}

handleUpdate = (taskId) => {

let taskDTO = this.state.task;

taskDTO = { ...taskDTO, taskId: taskId };

fetch("api/task/edit/" + taskId, {

method: "put",

headers: {

Accept: "application/json",

"Content-Type": "application/json"

},

body: JSON.stringify(taskDTO)

})

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ tasks: data.tasks });

})

}

render = () => {

let tasks = this.state.tasks ? (this.state.tasks.map(task => this.renderTaskBlock(task))) : <div>Loading</div>;

return (

<div>

<div >

<div className="course-flex">

{tasks}

</div>

</div>

<Button bsStyle="primary" onClick={this.toggleCreationModel}>Добавить новое задание</Button>

<Modal

show={this.state.showCreationModal}

onHide={this.toggleCreationModel}

dialogClassName="custom-modal">

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-lg">

Task creation

</Modal.Title>

</Modal.Header>

<Modal.Body>

{" "}

<Form >

<FormGroup>

<ControlLabel>TaskQuestion</ControlLabel>{" "}

<FormControl

type="text"

placeholder="Enter name of task"

onChange={event => this.handleFormChange(event.target.value, "TaskQuestion")}

/>{" "}

</FormGroup>{" "}

<FormGroup>

<ControlLabel>ThemeId</ControlLabel>{" "}

<FormControl

type="text"

placeholder="CourseId"

onChange={event => this.handleFormChange(event.target.value, "ThemeId")} />

</FormGroup>{" "}

<Button onClick={this.handleAdd} >

Добавить

</Button>

</Form>

<br />

</Modal.Body>

<Modal.Footer>

<Button onClick={this.toggleCreationModel}>Закрыть</Button>

</Modal.Footer>

</Modal>

</div>

)

}

}

**Course.js:**

import React, { Component } from 'react';

import {

Button, Modal, ButtonToolbar,

Form, ControlLabel,

FormGroup,

FormControl

} from "react-bootstrap";

import './styles.css';

export default class Courses extends Component {

displayName = Courses.name;

constructor(props, context) {

super(props, context);

this.handleShow = this.handleShow.bind(this);

this.handleHide = this.handleHide.bind(this);

this.handleAdd = this.handleAdd.bind(this);

this.handleDelete = this.handleDelete.bind(this);

this.handleShowCreationModal = this.handleShowCreationModal.bind(this);

this.handleHideCreationModal = this.handleHideCreationModal.bind(this);

this.handleHideUpdateModel = this.handleHideUpdateModel.bind(this);

this.handleShowUpdationModel = this.handleShowUpdationModel.bind(this);

this.toggleUpdateModal = this.toggleUpdateModal.bind(this);

this.state = {

courses: null,

themes: null,

theme: {

ThemeName: "",

CourseId: ""

},

course: {

Name: "",

Duration: "",

Description: ""

},

show: false,

showCreationModal: false,

showUpdationModal: false

};

}

componentDidMount = () => {

fetch("api/course/getall", { dataType: 'json' })

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ courses: data.courses });

});

fetch("api/theme/getall", {dataType: 'json'})

.then(response => response.json())

.then(data => {

console.log("themes", data);

this.setState({themes: data.themes});

})

}

handleShow() {

this.setState({ show: true });

}

handleHide() {

this.setState({ show: false });

}

handleShowCreationModal() {

this.setState({ showCreationModal: true });

}

handleHideCreationModal() {

this.setState({ showCreationModal: false });

}

handleShowUpdationModel() {

this.setState({ showUpdationModal: true });

}

handleHideUpdateModel() {

console.log("before", this.state);

this.setState({ showUpdationModal: false });

console.log("after", this.state);

}

toggleUpdateModal() {

const { showUpdationModal } = this.state;

this.setState({ showUpdationModal: !showUpdationModal })

}

handleAdd = () => {

fetch("api/course/create", {

method: "post",

headers: {

Accept: "application/json",

"Content-Type": "application/json"

},

body: JSON.stringify(this.state.course)

})

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ courses: data.courses });

})

};

handleUpdate = (courseId) => {

let courseDTO = this.state.course;

courseDTO = { ...courseDTO, CourseId: courseId };

fetch("api/course/edit/" + courseId, {

method: "put",

headers: {

Accept: "application/json",

"Content-Type": "application/json"

},

body: JSON.stringify(courseDTO)

})

.then(response => response.json())

.then(data => {

console.log(data);

this.setState({ courses: data.courses });

})

}

handleDelete = (courseId) => {

console.log("id", courseId);

if (!window.confirm("Are you sure you want to delete the course?")) return;

fetch("api/course/delete/" + courseId, { method: "delete" })

.then(responce => responce.json())

.then(data => {

console.log(data);

this.setState({ courses: data.courses })

})

}

handleFormChange = (value, field) => {

console.log("onChange", value, field)

this.setState(prevState => ({

course: {

...prevState.course,

[field]: value

}

}));

console.log(this.state.course);

}

renderCourseBlock = (course) => {

let themes = this.state.themes ? (this.state.themes.map(theme => this.renderThemeBlock(theme))) : <div>Loading themes...</div>

return (

<div className="course" key={course.courseId}>

<h1>{course.name}</h1>

<p><strong>Длительность:</strong> {course.duration}</p>

<div className="description">

{/\* <p><strong>Description:</strong> {course.description}</p> \*/}

</div>

<ButtonToolbar>

<Button bsStyle="primary" onClick={() => this.setState({ show: course.courseId })}>

Подробнее

</Button>

<Modal

show={this.state.show === course.courseId}

onHide={this.handleHide}

dialogClassName="custom-modal">

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-lg">

{course.name}

</Modal.Title>

</Modal.Header>

<Modal.Body>

<p><strong>Описание:</strong> {course.description}</p>

<p><strong>Длительность:</strong> {course.duration}</p>

<p><strong>Темы курса:</strong></p>

{/\* <Themes data={course.themes}></Themes> \*/}

{/\* <p>{course.courseId}</p> \*/}

{/\* <Link to={`tasks`}>something</Link> \*/}

{themes}

{/\* <p><strong>Tasks</strong></p> \*/}

</Modal.Body>

<Modal.Footer>

<Button onClick={this.handleHide}>Close</Button>

</Modal.Footer>

</Modal>

<Button bsStyle="default" onClick={() => this.setState({ showUpdationModal: course.courseId })}>Изменить</Button>

<Modal

show={this.state.showUpdationModal === course.courseId}

onHide={this.handleHideUpdateModel}

dialogClassName="custom-modal">

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-lg">

Course Update

</Modal.Title>

</Modal.Header>

<Modal.Body>

{" "}

<Form>

<FormGroup>

<ControlLabel>{course.name}</ControlLabel>{" "}

<FormControl

type="text"

placeholder={course.name}

onChange={event => this.handleFormChange(event.target.value, "Name")}

/>{" "}

</FormGroup>{" "}

<FormGroup>

<ControlLabel>{course.duration}</ControlLabel>{" "}

<FormControl

type="text"

placeholder={course.duration}

onChange={event => this.handleFormChange(event.target.value, "Duration")}

/>{" "}

</FormGroup>{" "}

<FormGroup>

<ControlLabel>{course.description}</ControlLabel>{" "}

<FormControl

type="text"

placeholder={course.description}

onChange={event => this.handleFormChange(event.target.value, "Description")}

/>{" "}

</FormGroup>{" "}

<Button onClick={() => this.handleUpdate(course.courseId)}>Save Changes</Button>

</Form>

<br />

</Modal.Body>

<Modal.Footer>

<Button

onClick={() => this.handleHideUpdateModel()}>Close</Button>

</Modal.Footer>

</Modal>

<Button

bsStyle="default" onClick={() => this.handleDelete(course.courseId)}>Удалить</Button>

</ButtonToolbar>

</div>

)

}

renderThemeBlock = (theme) => {

console.log("themeName", theme.themeID);

return (

<div key={theme.themeID}>

{/\* <p>CourseId:{theme.courseId}</p> \*/}

{/\* <p>{theme.themeID}</p> \*/}

<p>{theme.themeName}</p>

</div>

)

}

render = () => {

let cards = this.state.courses ? (this.state.courses.map(course => this.renderCourseBlock(course))) : <div>Loading courses...</div>;

return (

<div>

<h1>Курсы</h1>

<p>Здесь вы можете видеть курсы, к которым вы подключены</p>

<div className="course-flex">

{cards}

</div>

{/\* <p>There're {Math.floor(Math.random() \* 10)} courses here</p>

<p>There're {Math.floor(Math.random() \* 40)} materials here</p>

<p>There're {Math.floor(Math.random() \* 100)} tasks here</p> \*/}

{/\* <Button bsStyle="link" onClick={() => this.handleAdd}>Add new course</Button> \*/}

<ButtonToolbar>

<Button bsStyle="primary" onClick={this.handleShowCreationModal}>

Добавить новый курс

</Button>

<Modal

show={this.state.showCreationModal}

onHide={this.handleHideCreationModal}

dialogClassName="custom-modal">

<Modal.Header closeButton>

<Modal.Title id="contained-modal-title-lg">

Course creation

</Modal.Title>

</Modal.Header>

<Modal.Body>

{" "}

<Form inline>

<FormGroup>

<ControlLabel>Name</ControlLabel>{" "}

<FormControl

type="text"

placeholder="Enter name of course"

onChange={event => this.handleFormChange(event.target.value, "Name")}

/>{" "}

</FormGroup>{" "}

<FormGroup>

{/\* <ControlLabel>Description</ControlLabel>{" "} \*/}

<FormControl

type="text"

placeholder="Duration"

onChange={event => this.handleFormChange(event.target.value, "Duration")} />

</FormGroup>{" "}

<FormGroup>

<ControlLabel>Description</ControlLabel>{" "}

<FormControl

type="text"

placeholder="Enter description"

onChange={event => this.handleFormChange(event.target.value, "Description")}

/>{" "}

</FormGroup>{" "}

<Button onClick={this.handleAdd} >

New course

</Button>

</Form>

<br />

</Modal.Body>

<Modal.Footer>

<Button onClick={this.handleHideCreationModal}>Close</Button>

</Modal.Footer>

</Modal>

</ButtonToolbar>

</div>

);

}

}

**Использованные стили:**

**Styles.css:**

.course {

display: inline-block;

border: 1px solid #eee;

box-shadow: 0 2px 2px #ccc;

width:250px;

padding:20px;

margin: 10px 20px 10px 10px;

border-radius: 10px;

}

.course:hover {

box-shadow: 0 4px 4px rgb(175, 173, 173);

}

.course-flex {

display: flex;

flex-wrap: wrap;

}

.course-large {

display: inline-block;

border: 1px solid #eee;

box-shadow: 0 2px 2px #ccc;

width:300px;

padding:20px;

margin: 10px 20px 10px 10px;

border-radius: 10px;

}

.description {

background: yellow;

}

.fab {

margin: 0 2px 2px 2px,

}

.my-block {

margin: 0;

background: yellow;

position: absolute;

top: 50%;

left: 50%;

margin-right: -50%;

transform: translate(-50%, -50%)

}

.course-button {

display: inline-block;

border: 1px solid #eee;

box-shadow: 0 2px 2px #ccc;

width:10px;

padding:10px;

margin: 2px 2px 2px 2px;

border-radius: 1px;

}