**Вищий навчальний заклад**

**«Університет «КРОК»**

**Проєкт(третій рівень)**

**на тему:**

**«Проектування і реалізація WEB-порталу»**

|  |  |
| --- | --- |
| Студенти 3 курсу групи КН-19  Недолуженко Вадим Олегович  (прізвище, ім’я, по батькові)  Махно Андрій Андрійович  (прізвище, ім’я, по батькові)  Пахолка Михайло Петрович  (прізвище, ім’я, по батькові) | Керівник проектної роботи  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (посада керівника)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (прізвище, ім’я, по батькові)    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Резолюція «До захисту»)  \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_  (дата) (підпис викладача) |

**Київ – 2022 року**

Зміст

[ВСТУП 3](#_Toc104899153)

[КОМАНДА 4](#_Toc104899154)

[РОЗДІЛ 1. МЕТОДИ ВИРІШЕННЯ ЗАДАЧ ПРОЕКТНОЇ РОБОТИ 5](#_Toc104899155)

[1.1 ASP.NET MVC 5](#_Toc104899156)

[1.1.1 Code first 7](#_Toc104899157)

[1.1.2 Database First 8](#_Toc104899158)

[1.1.3 Model First 8](#_Toc104899159)

[1.1 Метод використаний в проекті 8](#_Toc104899160)

[1.2 Стадії створення проекту 9](#_Toc104899161)

[РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ПРОЕКТОВАНОЇ СИСТЕМИ 10](#_Toc104899162)

[2.1 Use case – diagrama проекта 10](#_Toc104899163)

[2.2 Структура і схеми бази даних 11](#_Toc104899164)

[2.2.1 Схема баз даних 11](#_Toc104899165)

[2.2.2 Таблиці баз даних 12](#_Toc104899166)

[2.1.3 Опис структур таблиць баз даних 15](#_Toc104899167)

[2.3 Програмне забезпечення 18](#_Toc104899168)

[2.3.1 Опис структури файлів проекта 18](#_Toc104899169)

[2.4 Результати тестування програми 21](#_Toc104899170)

[2.4.1 Оцінка необхідної пам'яті і машинного часу для вирішення задач 22](#_Toc104899171)

[2.5 Витрати машинного часу на виконання роботи сайту 23](#_Toc104899172)

[2.6 Інструкція користувача 23](#_Toc104899173)

[ВИСНОВОК 25](#_Toc104899174)

[СПИСОК ВИКОРИСТАННЯ ДЖЕРЕЛ 26](#_Toc104899175)

[Додаток А 28](#_Toc104899176)

# ВСТУП

Проєкт - це перша самостійна наукова праця майбутнього програмного інженера та його команди. При її виконанні студенти поглиблють знання з фундаментальних та професійно-орієнтованих дисциплін, застосовують методику експериментальних досліджень, оволодівають навичками співставлення результатів своїх досліджень із літературними даними, аналізу, узагальнення і літературного оформлення одержаних результатів з теми дослідження, набувають вміння вести науковий пошук, що розвиває в команди творчий підхід до роботи.

В цій роботі можна побачити роботу тьох студентів. Наша робота була направлена на те, щоб зробити локальну модель сайту, з декількома сторінками, а саме: «Розклад», «Електроний Журнал», «Навчальний план», «Список».

У цьому проекті, ми використали сучасну мову програмування C# на платформі Microsoft Visual Studios 2019 з використанням Веб-технології ASP.NET MVC.

Актуальність теми обумовлена необхідністю дізнатися навчальний план для студентів та вчителів коледжу. Складання «Навчального плану» є великим за обсягом даних і його автоматизація є важливим пунктом в освітніх закладах масового навчання. Після автоматизації «Навчального плану» всі користувачі можуть побачити план своєї та інших груп.

Метою проектної роботи було підняти навички в областях програмування та баз даних, а саме ASP.NET MVC, MySQL, Java. Кінцевим результатом роботи була робоча локальна модель сайту. В якому була реалізована база даних з великою кількістю таблиць. Під кінець ми би хотіли зробити більш кращий дизайн, багаторівневу авторизацію та конкретизування таблиць.

В самому проекті ми разом зробили базу даних, і кожний зробив свою таблицю. Під час створення проекту моїм завданням було створення «Навчальний план». А саме, щоб з основної бази даних з таблиці WorkPlan поступали данні до таблиці яку бачить користувач коли заходить на сайт через меню.

Самим головним предметом дослідження цього проекту був не дизайн сайту, не підключення бази даних, а саме її створення те як саме потрібно створювати зв’язки.

Під час роботи з цим проектом ми отримали багато нових знань та навичок таких як створення баз даних, зв’язків в базі даних, сайтів, та в роботі зі складними моделями.

# КОМАНДА

Недолуженко Вадим(Scrum-мастер, програміст)

Створив вкладку Навчальний план Електроний журнал, заповнював відповідну таблицю в базі данних.

Махно Андрій (програміст)

Створив базу данних, створив вкладку Розклад, , заповнював відповідну таблицю в базі данних.

Пахолка Михайло (програміст)

Створив вкладку Список, заповнював відповідну таблицю в базі данних.

# РОЗДІЛ 1. МЕТОДИ ВИРІШЕННЯ ЗАДАЧ ПРОЕКТНОЇ РОБОТИ

## ASP.NET MVC

Платформа ASP.NET MVC являє собою Framework для створення сайтів і веб-додатків за допомогою реалізації патерна MVС [18].

Концепція патерна (шаблону) MVC (model - view - controller) передбачає поділ додатка на три компоненти:

**Контролер** (controller) представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних. Він отримує вводяться користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді подання.

**Подання** (view) - відповідає за відображення інформації (візуалізацію). Одні і ті ж дані можуть представлятися різними способами і в різних форматах. Наприклад, колекцію об'єктів за допомогою різних уявлень можна уявити на рівні користувача інтерфейсу як в табличному вигляді, так і списком; на рівні API можна експортувати дані як в JSON, так в XML або XSLX.

**Модель** (model)- це дані і правила, які використовуються для роботи з даними, які представляють концепцію управління додатком. У будь-якому додатку вся структура моделюється як дані, які обробляються певним чином. Що таке користувач для додатка - повідомлення або книга? Тільки дані, які повинні бути оброблені відповідно до правил (дата не може вказувати в майбутнє, e-mail повинен бути в певному форматі, ім'я не може бути довшим Х символів, і так далі).Модель дає контролеру уявлення даних, які запросив користувач (повідомлення, сторінку книги, фотоальбом, тощо). Модель даних буде однаковою, незалежно від того, як ми хочемо представляти їх користувачеві. Тому ми вибираємо будь-який доступний вид для відображення даних. Модель містить найбільш важливу частину логіки нашого застосування, логіки, яка вирішує завдання, з якою ми маємо справу (форум, магазин, банк, тощо). Контролер містить в основному організаційну логіку для самого додатка.



Рис. 1.1 - Загальну схему взаємодії компонентів MVC

У цій схемі модель є незалежним компонентом - будь-які зміни контролера або подання не зачіпають модель. Контролер і уявлення (view) є відносно незалежними компонентами, і нерідко їх можна змінювати незалежно один від одного.

Завдяки цьому реалізується концепція поділ відповідальності, в зв’язку з чим легше побудувати роботу над окремими компонентами. Крім того, внаслідок чого застосунок буде мати кращу основу для тестувань. І якщо нам, припустимо, важлива візуальна частина або **фронтенд (**все, що браузер може читати, виводити на екран і / або запускати. Тобто це HTML, CSS і JavaScript), то ми можемо тестувати уявлення (view) незалежно від контролера. Або ми можемо зосередитися на бекенда (все, що працює на сервері, тобто «не в браузері» або «на комп'ютері, підключеному до мережі, який відповідає на повідомлення від інших комп'ютерів) і тестувати контролер.

Конкретні реалізації та визначення даного патерну можуть відрізнятися, але в силу своєї гнучкості і простоти він став дуже популярним останнім часом, особливо в сфері веб-розробки.

**Альтернативні назви патерна MVC:**

• model-view-controller

• модель-уявлення-поведінку

• модель-уявлення-контролер

• модель-вид-контролер

Більшість фреймворків для розробки веб-додатків побудовані на парадигмі MVC, тому досить просто зрозуміти принцип роботи будь-якого нового фреймворка, якщо ви стикалися з патерном MVC раніше.

**Функціональні можливості**

Оскільки MVC не має суворої реалізації, то реалізований він може бути по-різному. Немає загальноприйнятого визначення, де повинна розташовуватися бізнес-логіка. Вона може знаходитися як в контролері, так і в моделі. В останньому випадку, модель буде містити всі бізнес-об'єкти з усіма даними і функціями.

Також не вказано, де повинна знаходитися перевірка введених користувачем даних. Проста валідація може зустрічатися навіть у поданні, але частіше вони зустрічаються в контролері чи моделі.

Інтернаціоналізація і форматування даних також не має чітких вказівок по розташуванню.

### Code first

Підхід, називаємий Code First (спочатку код) передбачає мінімальну участь в проектуванні сутностей бази даних програмістом. Він просто пише код, усе інше робить Entity та Visual Studio 2019 [20].

Він підходить у випадках якщо головне в проекті - бізнес логіка, а база даних - це спосіб зберігання даних. Або у випадках якщо проект вже написаний, але в якості джерел даних використані списки, масиви, колекції. Code First дозволяє з мінімальними зусиллями змінити проект з використанням баз даних в якості джерел даних замість стандартних колекцій .NET.

### Database First

Database First був першим підходом, який з'явився в Entity Framework. Даний підхід багато в чому схожий на Model First і підходить для тих випадків, коли розробник вже має готову базу даних [19].

Щоб Entity Framework міг отримати доступ до бази даних, в системі повинен бути встановлений відповідний провайдер. Так, Visual Studio 2019 вже підтримує відповідну інфраструктуру для СУБД MS SQL Server. Для інших СУБД, наприклад, MySQL, Oracle та інших треба встановлювати відповідні провайдери. Список провайдерів для найбільш поширених СУБД можна знайти на сторінці ADO.NET Data Providers.

### **Model First**

Model First представляє ще один підхід до роботи з Entity Framework. Суть даного підходу полягає в тому, що спочатку робиться модель, а потім по ній створюється база даних [21].

## Метод використаний в проекті

За основу сайту ми взяли Шаблон веб-застосування ASP.NET MVC.

В цьому проекті ми перепробували всі три методи які були перераховані вище. Але найбільше нам сподобався метод Code first. Через те, що нам було набагато легше написати код для вже готової бази, ніж постійно прописувати генерацію кожного елементу бази окремо та створювати зв`язки для таблиць.

Використання цього методу передбачає створення бази даних окремо від проекту у програмі Manager SQL. Також цей метод передбачає зв’язування таблиць спочатку в програмі використовуючи ключі, а потім у проекті через Id.

## Стадії створення проекту

Перед створенням проекту ми зробили план всіх деталей проекту:

* Створення вибірки даних та самої бази:

1. Студенти
2. Групи
3. Кафедри
4. Викладачі
5. Журнал пропусків
6. Навчальний план
7. Розклад
8. Електронний журнал
9. Курси
10. Предмети

* Заповнення бази даних
* Створення ключів та зв’язків між таблицями
* Підключення бази до проекту
* Створення та заповнення контролерів
* Створення та заповнення уявлення (view)
* Підключення скриптів(JavaScript, bootstrap,…)
* Створення дизайну проекта

# РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ПРОЕКТОВАНОЇ СИСТЕМИ

Основним інформаційним забезпеченням є Інтернет. А саме статті Ріхтера, Джей Сон, а також форуми для програмістів(cyberforum).

## 2.1 Use case – diagrama проекта

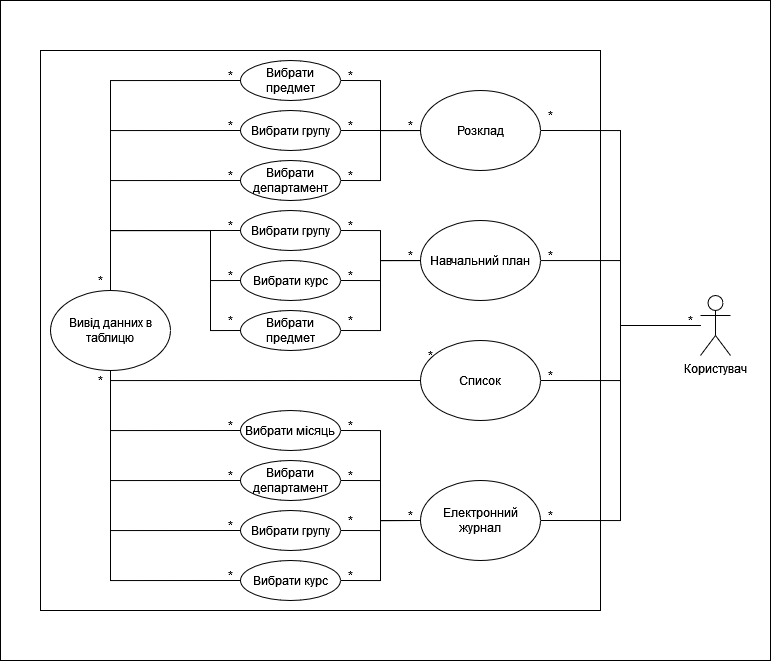


Рис. 2.1 Use case diagrama

Як можна побачити на діаграмі користувач може вибрати одну з 4 вкладок і змінювати будь які данні на його вибір, щоб потім побачити бажаний результат.

## 2.2 Структура і схеми бази даних

### 2.2.1 Схема баз даних

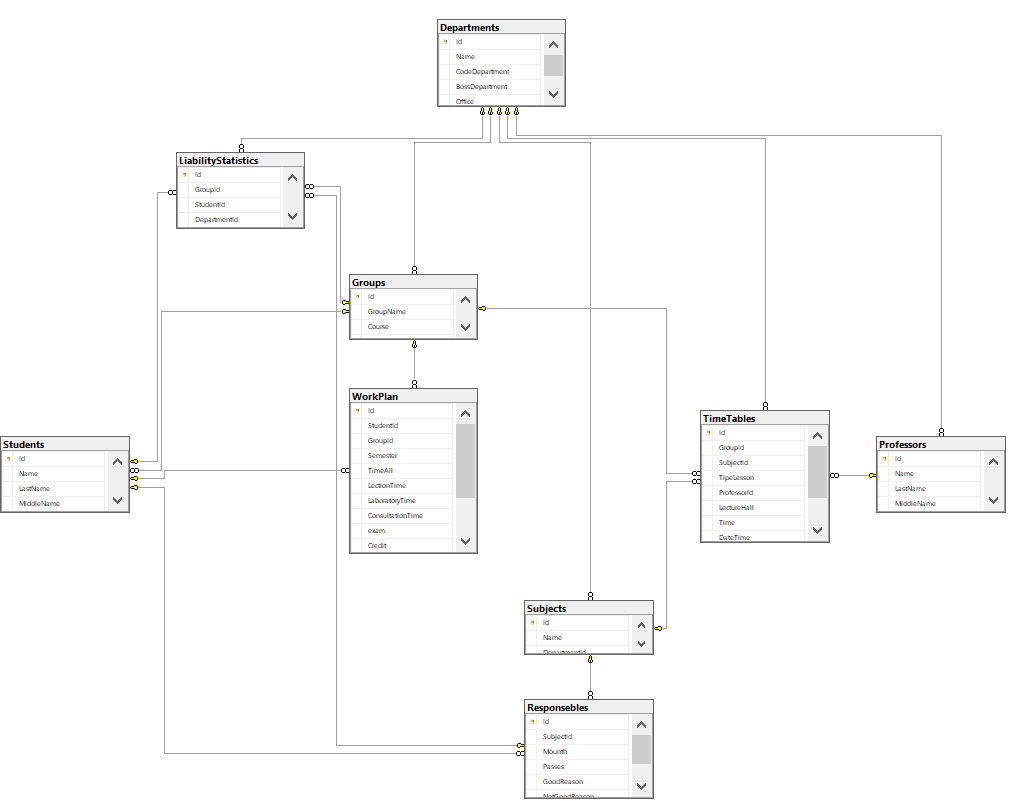


Рис.2.2 Діаграма бази даних

### 2.2.2 Таблиці баз даних

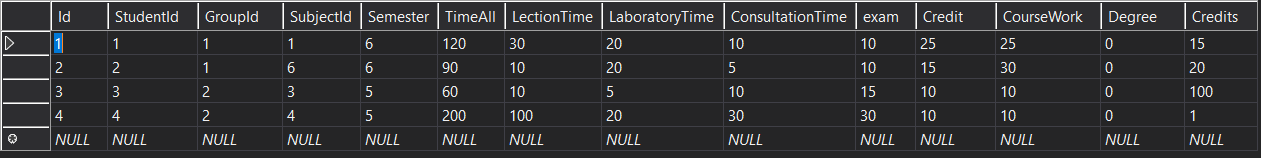


Рис. 2.3 – Таблиця WorkPlan

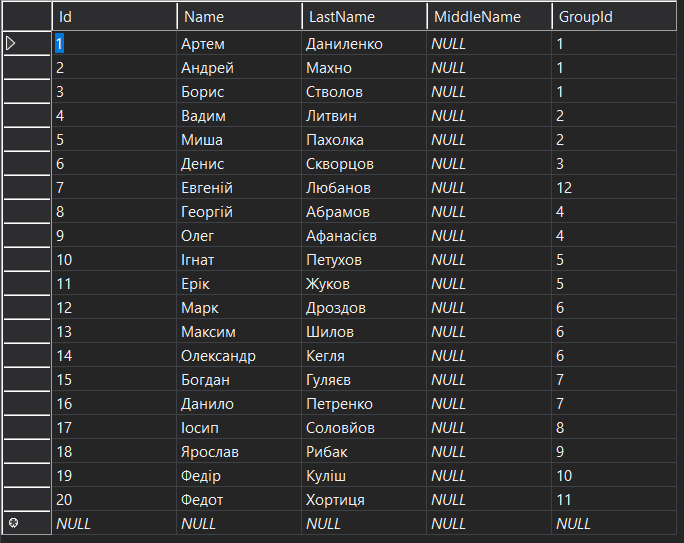


Рис. 2.4 – Таблиця Student

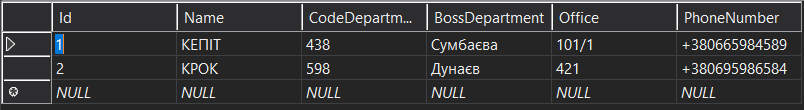


Рис. 2.5 – Таблиця Department

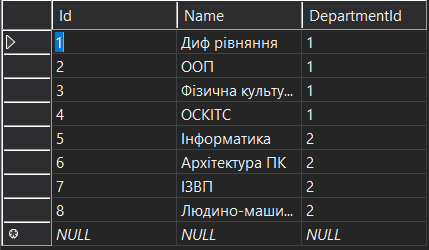


Рис. 2.6 – Таблиця Subject

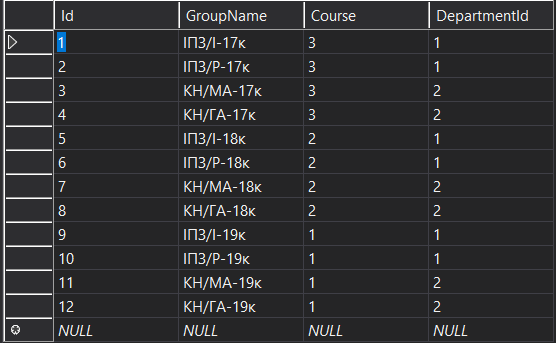


Рис. 2.7 – Таблиця Groups

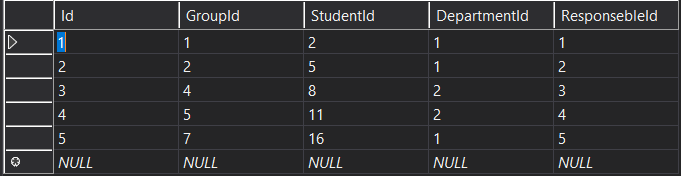


Рис. 2.8 – Таблиця LiabilityStatistics

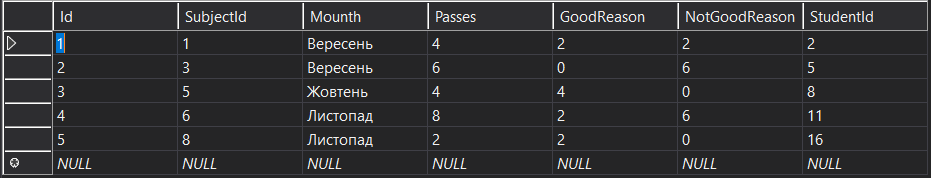


Рис. 2.9 – Таблиця Responsebles

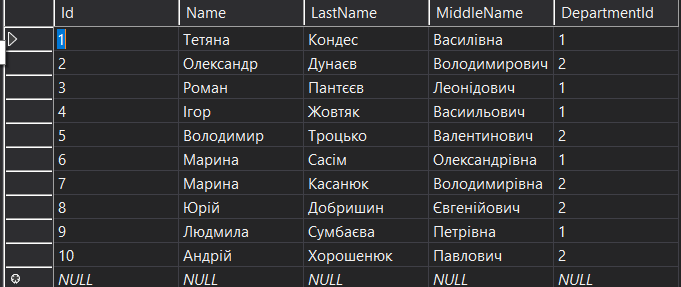


Рис. 2.10 – Таблиця Professors

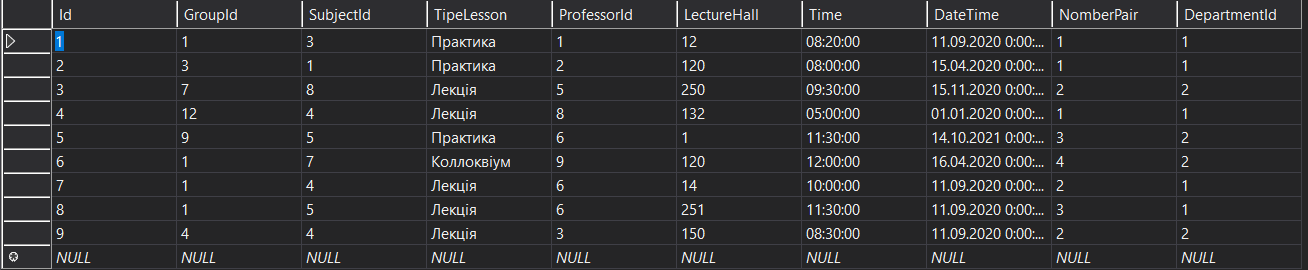


Рис. 2.11 – Таблиця TimeTable

### 2.1.3 Опис структур таблиць баз даних

Таблиця 2.1

Структура таблиці WorkPlan

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Призначення |
| Id | Int | Ідентифікатор |
| StudentId | Int | Ідентифікатор студента |
| GroupId | Int | Ідентифікатор групи |
| SubjectId | Int | Ідентифікатор дисципліни |
| Semester | nchar(10) | Семестр |
| TimeAll | Int | Години |
| LectionTime | Int | Лекція |
| LaboratoryTime | Int | Лабораторна |
| ConsultationTime | Int | Консультація |
| Exam | Real | Екзамен |
| Credit | Real | Диф.залік |
| CourseWork | Int | Курсова робота |
| Degree | Int | Дипломна робота |
| Credits | Int | Кредит |

Таблиця 2.2

Структура таблиці Subject

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Призначення |
| Id | Int | Ідентифікатор |
| Name | nvarchar(50) | Назва дисципліни |
| DepartmentId | Int | Ідентифікатор кафедри |

Таблиця 2.3

Структура таблиці Student

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Призначення |
| Id | Int | Ідентифікатор |
| Name | nvarchar(50) | Ім’я |
| LastName | nvarchar(50) | Прізвище |
| MiddleName | nvarchar(50) | По-батькові |
| GroupId | Int | Ідентифікатор групи |

Таблиця 2.4

Структура таблиці Departments

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Призначення |
| Id | int | Ідентифікатор |
| Name | nvarchar(50) | Назва |
| CodeDepartment | nvarchar(50) | Код кафедри |
| BossDepartment | nvarchar(50) | Голова кафедри |
| Office | nvarchar(50) | Офіс |
| PhoneNumber | nvarchar(50) | Номер телефону |

Таблиця 2.5

Структура таблиці Groups

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Призначення |
| Id | int | Ідентифікатор |
| GroupName | nvarchar(50) | Назва групи |
| Course | nvarchar(50) | Курс |
| DepartmentId | int | Ідентифікатор кафедки |

Таблиця 2.6

Структура таблиці LiabilityStatistics

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Призначення |
| Id | int | Ідентифікатор |
| GroupId | int | Ідентифікатор групи |
| StudentId | int | Ідентифікатор студента |
| DepartmentId | int | Ідентифікатор кафедри |
| ResponsebleId | int | Ідентифікатор відвідуваності |

Таблиця 2.7

Структура таблиці Responsebles

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Призначення |
| Id | int | Ідентифікатор |
| SubjectId | int | Ідентифікатор дисципліни |
| Mounth | nvarchar(50) | Місяць |
| Passes | int | Пропуск |
| GoodReason | int | По поважній причині |
| NotGoodReason | int | Без поважної причини |
| StudentId | int | Ідентифікатор студента |

Таблиця 2.8

Структура таблиці Professors

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Призначення |
| Id | int | Ідентифікатор |
| Name | nvarchar(50) | Ім’я |
| LastName | nvarchar(50) | Прізвище |
| MiddleName | nvarchar(50) | По-батькові |
| DepartmentId | int | Ідентифікатор кафедри |

Таблиця 2.9

Структура таблиці TimeTable

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Призначення |
| Id | int | Ідентифікатор |
| GroupId | int | Ідентифікатор групи |
| SubjectId | int | Ідентифікатор дисципліни |
| TipeLesson | nvarchar(50) | Вид заннятя |
| ProfessorId | int | Ідентифікатор викладача |
| LectureHall | int | Аудиторія |
| Time | time(7) | Час заннятя |
| DateTime | datetime | День зяннятя |
| NomberPair | tinyint | Порядок пари |
| DepartmentId | int | Ідентифікатор кафедри |

## 2.3 Програмне забезпечення

### 2.3.1 Опис структури файлів проекта

В папці Content зберігаються файли які впливають на графічне відображення сайту.

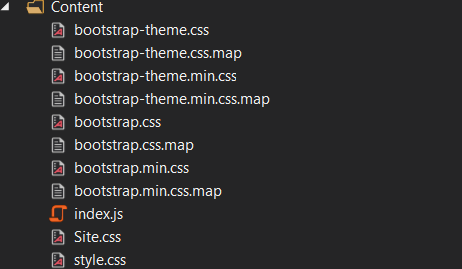


Рис. 2.12 - Папка Content

В папці Controller зберігаються файли які відповідають за вивід та сортування даних.

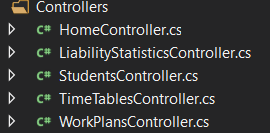


Рис. 2.13 - Папка Controller

В папці DAL зберігається файл який відповідає за навігацію даних.



Рис. 2.14 - Папка DAL

В папці fonts зберігаються файли які відповідають за текстові шрифти.

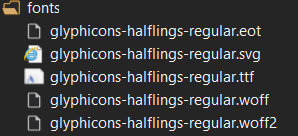


Рис. 2.15 - Папка fonts

В папці Models зберігається моделі даних.

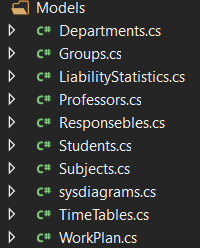


Рис. 2.16 - Папка Models

В папці Scripts зберігаються файли які відповідають за скриптовані анімації.

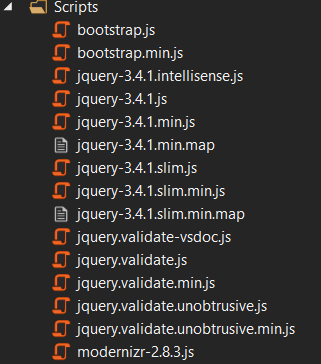


Рис. 2.17 - Папка Scripts

В папці Views зберігаються файли які відповідають за відображення таблиць.

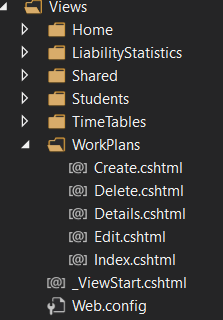


Рис. 2.18 - Папка Views

## 2.4 Результати тестування програми

Результатом виконання програми (рис. 2.19) є вивід таблиці з навчальним планом. Також до цієї таблиці реалізоване сортування за групою, курсом та предметом.

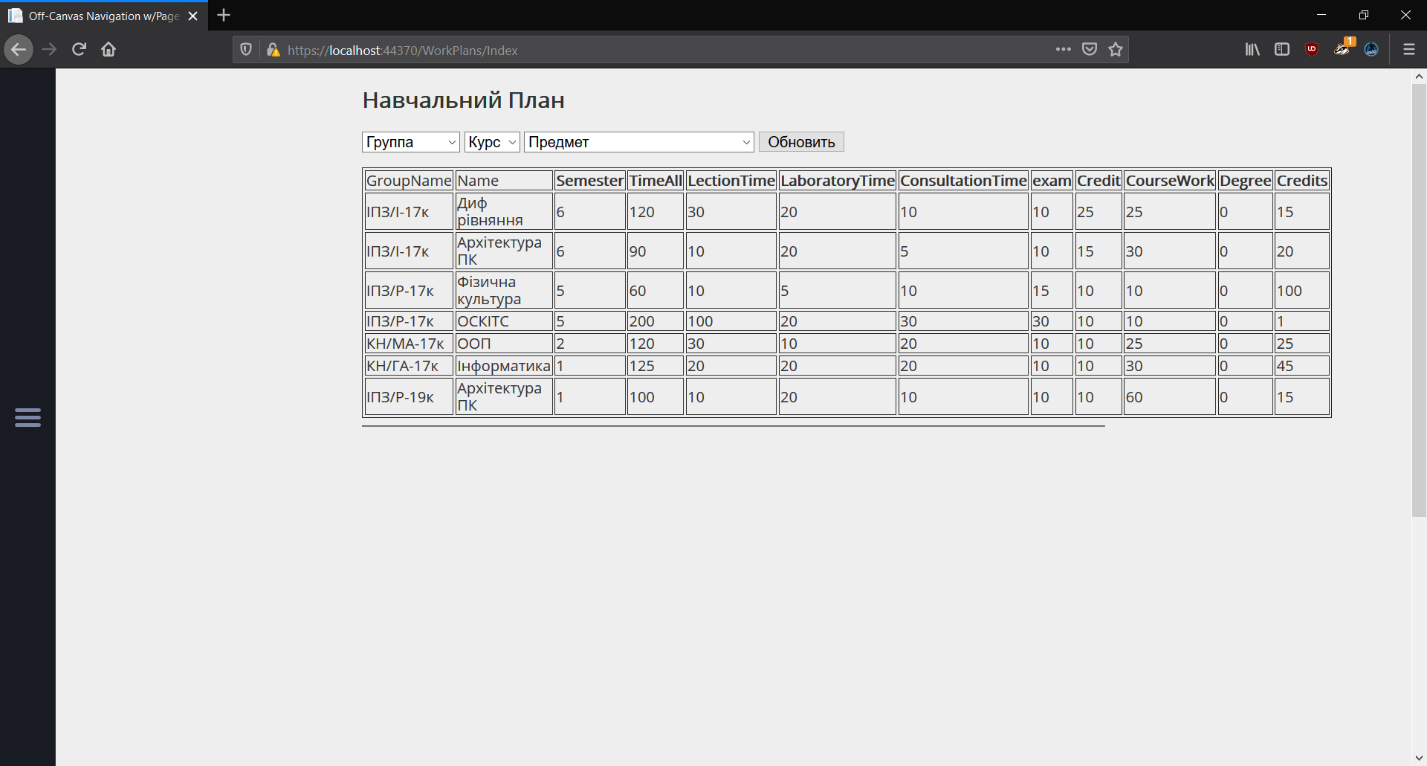


Рис. 2.19 - Приклад роботи сторінки сайту «Навчальний план»

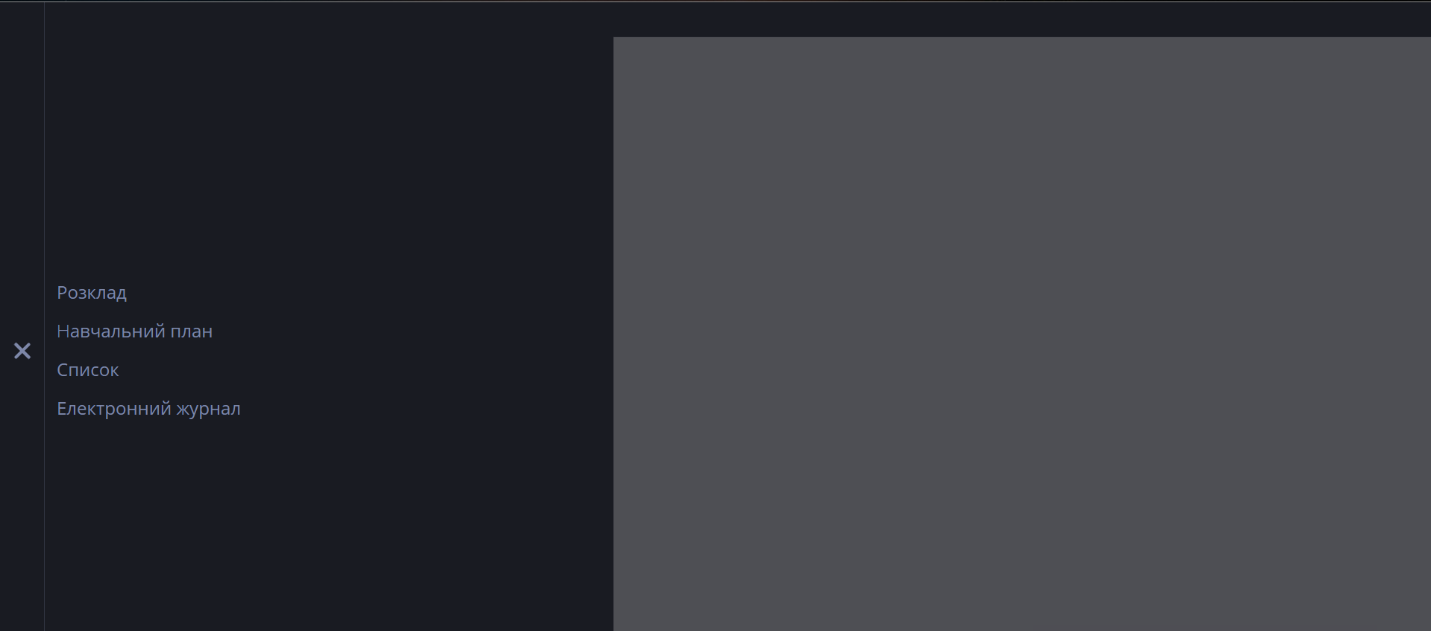


Рис. 2.20 - Приклад роботи меню сайта

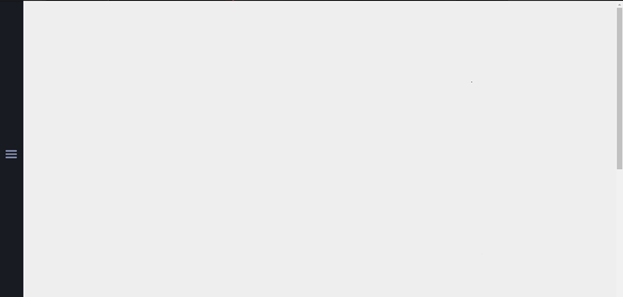


Рис.2.21 – Головна сторінка сайта

### 2.4.1 Оцінка необхідної пам'яті і машинного часу для вирішення задач

За результатами спостереження було виявлено, що сайт використовує незначну кількість пам’яті.

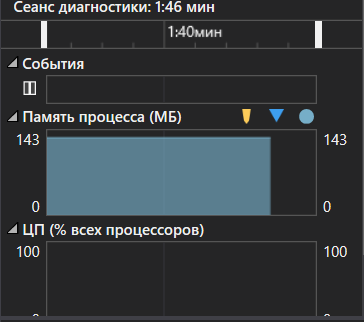


Рис. 2.22 - Дані з Visual Studios 2019

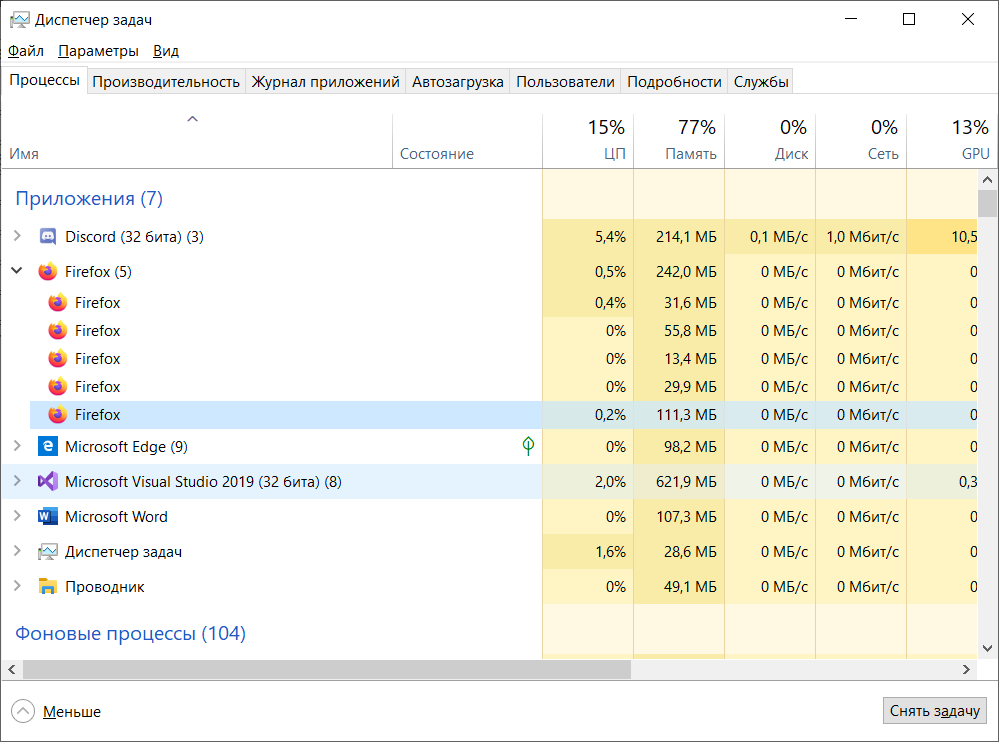


Рис. 2.23 - Дані з диспетчера задач

## 2.5 Витрати машинного часу на виконання роботи сайту

Оскільки цей сайт є локальним час за який сайт виконує роботу безпосередньо залежить від машини на якій він запущений. Після спостереження за сайтом на декількох машинах не було виявлено проблем з надмірних затрат ресурсів машин.

## 2.6 Інструкція користувача

Системні вимоги для роботи проекта:

1. Процесор: Intel Core i3 2.2 GHz;
2. Мінімальний об’єм оперативної пам'яті: 2 Гб;
3. Мінімальний об’єм відео пам’яті: 128 МБ
4. Пристрій введення (мишка, клавіатура);
5. Пристрій виведення (монітор);
6. Операційна система WINDOWSXP\7\VISTA\8\8.1 .
7. Необхідне програмне забезпечення Microsoft Visual Studio.

Робота програми розпочинається з відкриття файлу Krok Site.exe.

Відкривається головна сторінка сайту (Рис.2.21).

Для початку користувач має можливість відкрити меню сайту (рис.2.20), яке знаходиться в лівому краю сторінки.

Після відкриття меню користувач може вибрати будь який пункт меню(рис.2.20), туди входить «Розклад», «Навчальний план», «Список» та «Електронний журнал».

В розділі «Навчальний план» користувач побачить три поля для сортування таблиці та саму таблицю з даними(рис.2.19).

Після використання хелперов (DropDownList) користувачу виведуться дані які він хоче побачити(рис.2.19).

Для виходу із сайту потрібно закрити сайт.

# ВИСНОВОК

Під час виконання проектної роботи ми ознайомилися з ASP.NET, закріпили навички з мови програмування С#. Навчилися застосовувати MySQL та створювати бази даних. А точніше створення таблиці, з’єднання таблиць, заповнення таблиць та підключати бази до проекту.

Під час написання проекту були створена модель, контекст даних, контролер та представлення (views). Кожен із них відповідає за свою часину в сайті.

Під кінець роботи з проектом ми закріпили багато різних знань та навичок які допоможуть нам в майбутньому, а саме в написанні та розробці власних застосувань в середовищі Visual Studio.

# СПИСОК ВИКОРИСТАННЯ ДЖЕРЕЛ

1. Використання сесії [Електронний ресурс] – Режим доступу: <https://stackoverflow.com/questions/14138872/how-to-use-sessions-in-an-asp-net-mvc-4application>
2. Що таке MVC [Електронний ресурс] – Режим доступу: <https://docs.microsoft.com/enus/aspnet/mvc/overview/getting-started/introduction/getting-started>
3. Теорія MVC [Електронний ресурс] – Режим доступу: <https://ru.wikipedia.org/wiki/Model-ViewController>
4. Міграція баз даних [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/mvc5/7.2.php>
5. Базова валідація [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/mvc5/5.12.php>
6. Початок MVC MSDN docs [Електронний ресурс] – Режим доступу:<https://docs.microsoft.com/enus/aspnet/mvc/overview/getting-started/getting-started-with-ef-usingmvc/migrationsand-deployment-with-the-entity-framework-in-an-asp-net-mvc-application>
7. Авторизація та автентифікація [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/mvc5/12.1.php>
8. Viewbag [Електронний ресурс] – Режим доступу: <https://www.tutorialsteacher.com/mvc/viewbaginasp.net-mvc>
9. TFS [Електронний ресурс] – Режим доступу: <https://visualstudio.microsoft.com/ru/tfs/>
10. Code first approach [Електронний ресурс] – Режим доступу: <https://docs.microsoft.com/enus/aspnet/mvc/overview/getting-started/getting-started-with-ef-usingmvc/creating-anentity-framework-data-model-for-an-asp-net-mvc-application>
11. Database first Approach [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/entityframework/2.4.php>
12. Початок ООП [Електронний ресурс] – Режим доступу: <https://tproger.ru/translations/diving-inoopp1/>
13. Як використовувати сеанси в додатку [Електронний ресурс] – Режим доступу: <https://stackoverflow.com/questions/14138872/how-to-use-sessions-in-an-asp-net-mvc-4application>
14. Основи валідації [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/mvc5/7.2.php>
15. Міграція баз даних [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/mvc5/5.12.php>
16. Сортування [Електронний ресурс] – Режим доступу: <https://habr.com/ru/company/microsoft/blog/133845/>
17. Работа с данными в Entity Framework [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/aspnet5/12.1.php>
18. Введение в ASP.NET MVC 5 [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/mvc5/1.1.php>
19. Database First [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/entityframework/2.4.php>
20. Code First [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/entityframework/2.2.php>
21. Model First [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/entityframework/2.5.php>

# Додаток А

#### WorkPlanControler

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using KrokSite.Models;

namespace KrokSite.Controllers

{

public class WorkPlansController : Controller

{

private Model1 db = new Model1();

// GET: WorkPlans

public ActionResult Index(string GroupName, string CourseNumber, string SubjectName)

{

var GroupLst = new List<string>();

var GroupQry = from l in db.WorkPlan

join r in db.Groups on l.GroupId equals r.Id

select r.GroupName;

var CourseLst = new List<string>();

var CourseQry = from l in db.WorkPlan

join r in db.Groups on l.GroupId equals r.Id

select r.Course;

var SubLst = new List<string>();

var SubQry = from l in db.Responsebles

join r in db.Subjects on l.SubjectId equals r.Id

select r.Name;

GroupLst.AddRange(GroupQry.Distinct());

ViewBag.GroupName = new SelectList(GroupLst);

CourseLst.AddRange(CourseQry.Distinct());

ViewBag.CourseNumber = new SelectList(CourseLst);

SubLst.AddRange(SubQry.Distinct());

ViewBag.SubjectName = new SelectList(SubLst);

var table = from m in db.WorkPlan select m;

if (!string.IsNullOrEmpty(GroupName))

{

table = table.Where(y => y.Groups.GroupName == GroupName);

}

if (!string.IsNullOrEmpty(CourseNumber))

{

table = table.Where(y => y.Groups.Course == CourseNumber);

}

if (!string.IsNullOrEmpty(SubjectName))

{

table = table.Where(y => y.Subjects.Name == SubjectName);

}

return View(table.ToList());

}

// GET: WorkPlans/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

WorkPlan = db.WorkPlan.Find(id);

if (workPlan == null)

{

return HttpNotFound();

}

return View(workPlan);

}

// GET: WorkPlans/Create

public ActionResult Create()

{

ViewBag.GroupId = new SelectList(db.Groups, "Id", "GroupName");

ViewBag.StudentId = new SelectList(db.Students, "Id", "Name");

return View();

}

// POST: WorkPlans/Create

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Id,StudentId,GroupId,Semester,TimeAll,LectionTime,LaboratoryTime,ConsultationTime,exam,Credit,CourseWork,Degree,Credits")] WorkPlan workPlan)

{

if (ModelState.IsValid)

{

db.WorkPlan.Add(workPlan);

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.GroupId = new SelectList(db.Groups, "Id", "GroupName", workPlan.GroupId);

ViewBag.StudentId = new SelectList(db.Students, "Id", "Name", workPlan.StudentId);

return View(workPlan);

}

// GET: WorkPlans/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

WorkPlan = db.WorkPlan.Find(id);

if (workPlan == null)

{

return HttpNotFound();

}

ViewBag.GroupId = new SelectList(db.Groups, "Id", "GroupName", workPlan.GroupId);

ViewBag.StudentId = new SelectList(db.Students, "Id", "Name", workPlan.StudentId);

return View(workPlan);

}

// POST: WorkPlans/Edit/5

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "Id,StudentId,GroupId,Semester,TimeAll,LectionTime,LaboratoryTime,ConsultationTime,exam,Credit,CourseWork,Degree,Credits")] WorkPlan workPlan)

{

if (ModelState.IsValid)

{

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

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.GroupId = new SelectList(db.Groups, "Id", "GroupName", workPlan.GroupId);

ViewBag.StudentId = new SelectList(db.Students, "Id", "Name", workPlan.StudentId);

return View(workPlan);

}

// GET: WorkPlans/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

WorkPlan = db.WorkPlan.Find(id);

if (workPlan == null)

{

return HttpNotFound();

}

return View(workPlan);

}

// POST: WorkPlans/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

WorkPlan = db.WorkPlan.Find(id);

db.WorkPlan.Remove(workPlan);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

#### WorkPlan(model)

namespace KrokSite.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Data.Entity.Spatial;

[Table("WorkPlan")]

public partial class WorkPlan

{

[DatabaseGenerated(DatabaseGeneratedOption.None)]

public int Id { get; set; }

public int? StudentId { get; set; }

public int? GroupId { get; set; }

[StringLength(10)]

public string Semester { get; set; }

public int? TimeAll { get; set; }

public int? LectionTime { get; set; }

public int? LaboratoryTime { get; set; }

public int? ConsultationTime { get; set; }

public float? exam { get; set; }

public float? Credit { get; set; }

public int? CourseWork { get; set; }

public int? Degree { get; set; }

public int? Credits { get; set; }

public virtual Groups { get; set; }

public virtual Students { get; set; }

}

}

##### Students(model)

namespace KrokSite.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Data.Entity.Spatial;

public partial class Students

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Students()

{

LiabilityStatistics = new HashSet<LiabilityStatistics>();

Responsebles = new HashSet<Responsebles>();

WorkPlan = new HashSet<WorkPlan>();

}

[DatabaseGenerated(DatabaseGeneratedOption.None)]

public int Id { get; set; }

[StringLength(50)]

public string Name { get; set; }

[StringLength(50)]

public string LastName { get; set; }

[StringLength(50)]

public string MiddleName { get; set; }

public int? GroupId { get; set; }

public virtual Groups { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<LiabilityStatistics> LiabilityStatistics { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Responsebles> Responsebles { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<WorkPlan> WorkPlan { get; set; }

}

}

##### Groups(model)

namespace KrokSite.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Data.Entity.Spatial;

public partial class Groups

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Groups()

{

TimeTables = new HashSet<TimeTables>();

LiabilityStatistics = new HashSet<LiabilityStatistics>();

Students = new HashSet<Students>();

WorkPlan = new HashSet<WorkPlan>();

}

[DatabaseGenerated(DatabaseGeneratedOption.None)]

public int Id { get; set; }

[StringLength(50)]

public string GroupName { get; set; }

[StringLength(50)]

public string Course { get; set; }

public int? DepartmentId { get; set; }

public virtual Departments { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<TimeTables> TimeTables { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<LiabilityStatistics> LiabilityStatistics { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Students> Students { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<WorkPlan> WorkPlan { get; set; }

}

}

##### Departments(model)

namespace KrokSite.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Data.Entity.Spatial;

public partial class Departments

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Departments()

{

TimeTables = new HashSet<TimeTables>();

Groups = new HashSet<Groups>();

LiabilityStatistics = new HashSet<LiabilityStatistics>();

Professors = new HashSet<Professors>();

Subjects = new HashSet<Subjects>();

Subjects1 = new HashSet<Subjects>();

}

[DatabaseGenerated(DatabaseGeneratedOption.None)]

public int Id { get; set; }

[StringLength(50)]

public string Name { get; set; }

[StringLength(50)]

public string CodeDepartment { get; set; }

[StringLength(50)]

public string BossDepartment { get; set; }

[StringLength(50)]

public string Office { get; set; }

[StringLength(50)]

public string PhoneNumber { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<TimeTables> TimeTables { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Groups> Groups { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<LiabilityStatistics> LiabilityStatistics { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Professors> Professors { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Subjects> Subjects { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Subjects> Subjects1 { get; set; }

}

}

##### Subjects(model)

namespace KrokSite.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Data.Entity.Spatial;

public partial class Subjects

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Subjects()

{

Responsebles = new HashSet<Responsebles>();

TimeTables = new HashSet<TimeTables>();

WorkPlan = new HashSet<WorkPlan>();

}

[DatabaseGenerated(DatabaseGeneratedOption.None)]

public int Id { get; set; }

[StringLength(50)]

public string Name { get; set; }

public int? DepartmentId { get; set; }

public virtual Departments { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Responsebles> Responsebles { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<TimeTables> TimeTables { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<WorkPlan> WorkPlan { get; set; }

}

}

#### WorkPlan(view)

##### Index.cshtml

@model IEnumerable<KrokSite.Models.WorkPlan>

@{

ViewBag.Title = "Index";

}

<h2>Навчальний План</h2>

@using (Html.BeginForm())

{

<p>

@Html.DropDownList("GroupName", "Группа")

@Html.DropDownList("CourseNumber", "Курс")

@Html.DropDownList("SubjectName", "Предмет")

<input type="submit" value="Обновить">

</p>

}

<table class="table" border="1">

<tr>

<td>

@Html.DisplayNameFor(model => model.Groups.GroupName)

</td>

<td>

@Html.DisplayNameFor(model => model.Subjects.Name)

</td>

<th>

@Html.DisplayNameFor(model => model.Semester)

</th>

<th>

@Html.DisplayNameFor(model => model.TimeAll)

</th>

<th>

@Html.DisplayNameFor(model => model.LectionTime)

</th>

<th>

@Html.DisplayNameFor(model => model.LaboratoryTime)

</th>

<th>

@Html.DisplayNameFor(model => model.ConsultationTime)

</th>

<th>

@Html.DisplayNameFor(model => model.exam)

</th>

<th>

@Html.DisplayNameFor(model => model.Credit)

</th>

<th>

@Html.DisplayNameFor(model => model.CourseWork)

</th>

<th>

@Html.DisplayNameFor(model => model.Degree)

</th>

<th>

@Html.DisplayNameFor(model => model.Credits)

</th>

<th></th>

</tr>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Groups.GroupName)

</td>

<td>

@Html.DisplayFor(modelItem => item. Subjects.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Semester)

</td>

<td>

@Html.DisplayFor(modelItem => item.TimeAll)

</td>

<td>

@Html.DisplayFor(modelItem => item.LectionTime)

</td>

<td>

@Html.DisplayFor(modelItem => item.LaboratoryTime)

</td>

<td>

@Html.DisplayFor(modelItem => item.ConsultationTime)

</td>

<td>

@Html.DisplayFor(modelItem => item.exam)

</td>

<td>

@Html.DisplayFor(modelItem => item.Credit)

</td>

<td>

@Html.DisplayFor(modelItem => item.CourseWork)

</td>

<td>

@Html.DisplayFor(modelItem => item.Degree)

</td>

<td>

@Html.DisplayFor(modelItem => item.Credits)

</td>

</tr>

}

</table>

##### Create.cshtml

@model KrokSite.Models.WorkPlan

@{

ViewBag.Title = "Create";

}

<h2>Create</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>WorkPlan</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.Id, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Id, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Id, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.StudentId, "StudentId", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("StudentId", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.StudentId, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.GroupId, "GroupId", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("GroupId", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.GroupId, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Semester, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Semester, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Semester, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.TimeAll, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.TimeAll, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.TimeAll, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LectionTime, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.LectionTime, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.LectionTime, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LaboratoryTime, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.LaboratoryTime, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.LaboratoryTime, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.ConsultationTime, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.ConsultationTime, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.ConsultationTime, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.exam, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.exam, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.exam, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Credit, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Credit, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Credit, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.CourseWork, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.CourseWork, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.CourseWork, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Degree, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Degree, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Degree, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Credits, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Credits, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Credits, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Create" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

##### Delete.cshtml

@model KrokSite.Models.WorkPlan

@{

ViewBag.Title = "Delete";

}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>WorkPlan</h4>

<hr />

<dl class="dl-horizontal">

<dt>

@Html.DisplayNameFor(model => model.Groups.GroupName)

</dt>

<dd>

@Html.DisplayFor(model => model.Groups.GroupName)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Students.Name)

</dt>

<dd>

@Html.DisplayFor(model => model.Students.Name)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Semester)

</dt>

<dd>

@Html.DisplayFor(model => model.Semester)

</dd>

<dt>

@Html.DisplayNameFor(model => model.TimeAll)

</dt>

<dd>

@Html.DisplayFor(model => model.TimeAll)

</dd>

<dt>

@Html.DisplayNameFor(model => model.LectionTime)

</dt>

<dd>

@Html.DisplayFor(model => model.LectionTime)

</dd>

<dt>

@Html.DisplayNameFor(model => model.LaboratoryTime)

</dt>

<dd>

@Html.DisplayFor(model => model.LaboratoryTime)

</dd>

<dt>

@Html.DisplayNameFor(model => model.ConsultationTime)

</dt>

<dd>

@Html.DisplayFor(model => model.ConsultationTime)

</dd>

<dt>

@Html.DisplayNameFor(model => model.exam)

</dt>

<dd>

@Html.DisplayFor(model => model.exam)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Credit)

</dt>

<dd>

@Html.DisplayFor(model => model.Credit)

</dd>

<dt>

@Html.DisplayNameFor(model => model.CourseWork)

</dt>

<dd>

@Html.DisplayFor(model => model.CourseWork)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Degree)

</dt>

<dd>

@Html.DisplayFor(model => model.Degree)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Credits)

</dt>

<dd>

@Html.DisplayFor(model => model.Credits)

</dd>

</dl>

@using (Html.BeginForm()) {

@Html.AntiForgeryToken()

<div class="form-actions no-color">

<input type="submit" value="Delete" class="btn btn-default" /> |

@Html.ActionLink("Back to List", "Index")

</div>

}

</div>

##### Details.cshtml

@model KrokSite.Models.WorkPlan

@{

ViewBag.Title = "Details";

}

<h2>Details</h2>

<div>

<h4>WorkPlan</h4>

<hr />

<dl class="dl-horizontal">

<dt>

@Html.DisplayNameFor(model => model.Groups.GroupName)

</dt>

<dd>

@Html.DisplayFor(model => model.Groups.GroupName)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Students.Name)

</dt>

<dd>

@Html.DisplayFor(model => model.Students.Name)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Semester)

</dt>

<dd>

@Html.DisplayFor(model => model.Semester)

</dd>

<dt>

@Html.DisplayNameFor(model => model.TimeAll)

</dt>

<dd>

@Html.DisplayFor(model => model.TimeAll)

</dd>

<dt>

@Html.DisplayNameFor(model => model.LectionTime)

</dt>

<dd>

@Html.DisplayFor(model => model.LectionTime)

</dd>

<dt>

@Html.DisplayNameFor(model => model.LaboratoryTime)

</dt>

<dd>

@Html.DisplayFor(model => model.LaboratoryTime)

</dd>

<dt>

@Html.DisplayNameFor(model => model.ConsultationTime)

</dt>

<dd>

@Html.DisplayFor(model => model.ConsultationTime)

</dd>

<dt>

@Html.DisplayNameFor(model => model.exam)

</dt>

<dd>

@Html.DisplayFor(model => model.exam)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Credit)

</dt>

<dd>

@Html.DisplayFor(model => model.Credit)

</dd>

<dt>

@Html.DisplayNameFor(model => model.CourseWork)

</dt>

<dd>

@Html.DisplayFor(model => model.CourseWork)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Degree)

</dt>

<dd>

@Html.DisplayFor(model => model.Degree)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Credits)

</dt>

<dd>

@Html.DisplayFor(model => model.Credits)

</dd>

</dl>

</div>

<p>

@Html.ActionLink("Edit", "Edit", new { id = Model.Id }) |

@Html.ActionLink("Back to List", "Index")

</p>

##### Edit.cshtml

@model KrokSite.Models.WorkPlan

@{

ViewBag.Title = "Edit";

}

<h2>Edit</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>WorkPlan</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

@Html.HiddenFor(model => model.Id)

<div class="form-group">

@Html.LabelFor(model => model.StudentId, "StudentId", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("StudentId", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.StudentId, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.GroupId, "GroupId", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("GroupId", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.GroupId, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Semester, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Semester, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Semester, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.TimeAll, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.TimeAll, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.TimeAll, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LectionTime, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.LectionTime, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.LectionTime, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LaboratoryTime, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.LaboratoryTime, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.LaboratoryTime, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.ConsultationTime, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.ConsultationTime, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.ConsultationTime, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.exam, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.exam, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.exam, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Credit, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Credit, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Credit, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.CourseWork, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.CourseWork, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.CourseWork, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Degree, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Degree, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Degree, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Credits, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Credits, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Credits, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Save" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}