Course Manager

Proiectul a fost realizat de:

1. Vulpescu Radu-Cristian (1st SCRUM Master)

2. Corfu Alexandru (2nd SCRUM Master)

3. Doboș Iulia

4. Dorneanu Adrian

5. Moroșanu George-Cosmin (aka Juan)

6. Vicol Robert-Ilie

# 1. Abstract

Course Manager este o aplicație web ce are ca scop administrarea mai ușoară a unui curs. Proiectul curent a fost personalizat pentru cursul de Introducere în .NET.

# 2. Descriere

În primul rând, în interiorul aplicației există trei roluri, fiecare cu permisiunile lui. Acestea sunt: owner, assistant și student. Există un singur cont de owner și nu poate fi creat altul. 2.1. Sistemul de login

A fost creat cu ajutorul ASP.NET Identity. Oricine iși poate crea un cont de student, iar acesta trebuie verificat pe e-mail. Conturile de asistent pot fi verificate doar pe adresa de e-mail dedicată aplicației.

Aplicația este împărțită în șase funcționalități: News, Lectures, Katas, Presences, Questions și Uploads.

2.2. News

Conține ultimele noutăți în legatură cu ce se întâmplă sau se va înâmpla în legătură cu desfășurarea cursului. Doar owner-ul și asistenții pot adăuga știri. Știrile conțin un titlu și o descriere, plus posibilitatea de a adăuga atașamente. De asemenea, creatorului știrii îi este pus la dispoziție un tool de personalizare a textului.

## 2.3. Lectures și Katas

Au rolul de a păstra ultimele cursuri predate, respectiv kata înregistrate. Asemeni știrilor acestea au un titlu și o descriere, plus posibilitatea de a adăuga atașamente. Doar owner-ul și asistenții pot adăuga în această secțiune.

2.3. Presences

Conține situația actuală a laboratoarelor. Deoarece acesta este un opțional și laboratoarele se fac cu mai multe grupe deodată, am folosit conceptul de facțiune pentru un grup de grupe. Pentru a crea o facțiune trebuie introduse grupele care vor face parte din ea. Odată creată, în aceasta vor apărea toți studenții care fac parte din acele grupe. Un administrator poate da click pe „Start laboratory”, oferind astfel posibilitatea studenților de a da click pe „Mark as present”. Administratorii au de asemenea posibilitatea de a adăuga note la fiecare student atât la laborator, cât și la kata.

## 2.4. Questions

În această secțiunea sunt întrebările puse de către studențti și răspunsurile pentru acestea. O întrebare sau răspuns poate fi editat sau șters doar de căter autor sau unul dintre administratori.

2.5. Uploads

Fiecare student poate sa încarce tema de laborator sau kata-ul aferent acelei săptămâni.

# 3. Ce ne-am propus să implementăm

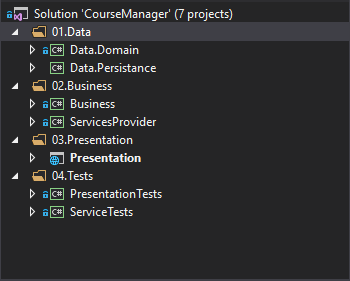
* **News**: s-a dorit implementarea știrilor într-un infinit scroll.
* **Lectures**: s-a dorit implementarea posibilității de a adăuga cursuri cu posibilitatea de personlizare și adaugare de atașamente
* **Katas**: la fel ca lectures cu excepția faptului că s-a mai dorit posibilitatea de a vizualiza kata-ul direct pe site
* **Presence**: această funcțonalitate trebuia să aibă posibilitatea de a vizualiza situația curentă a laboratoarelor: prezență, note (laborator/kata) și posibilitatea de a face prezența online
* **Questions**: s-a propus implementarea unei secțiuni împărțite pe topicuri în care fiecare student să adreseze întrebări în funcție de topic
* **Uploads**: s-a propus implementarea unui sistem de upload în care fiecare student putea alege tipul de temă, profesorul căruia să i-o asigneze și adauga un fișier doar în format .rar sau .zip care să nu depășeasca o anumită marime, iar profesorul să poată vedea toate temele căruia i-au ost trimise, expeditorul și data expedierii.

# 4. Ce am reușit să implementăm

* **News**: s-a implementat cu excepția faptului că nu am avem scroll infinit, ci paginare
* **Lectures**: s-a implementat tot ce s-a dorit
* **Katas**: nu s-a implementat posibilitatea de vizualizare online a kata-urilor
* **Presence**: s-a implementat tot ce s-a dorit
* **Questions**: pe partea de back-end s-a implementat tot ce s-a dorit, ce ar fi făcut în schimb această funcționalitate mult mai atrăgătoare ar fi fost un front-end mai elaborat
* **Uploads**: nu s-a implementat posibilitatea de a asigna tema uploadată cuiva

# 5. Pattern-uri folosite

## 5.1. Onion architecture pattern



Am implementat acest pattern împărțind tot codul pe straturi:

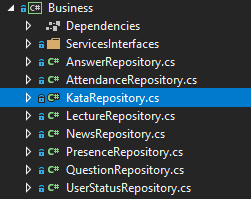
-Data ce conține entitățile, conexiunea la baza de date și interfețele pentru repository

-În business se afla repository-ul împreună cu interfețele pentru servicii și serviciile

-În presentation se află MVC-ul aplicației

-Iar în tests se află evident testele aplicației

## 5.2. Repository pattern

Cine nu și-ar dori să maximizeze cantitatea de cod care poate fi testate?

Ei bine, cu ajutorul acestui pattern, putem face acest fapt, plus faptul că ne oferă o separare a business logicului față de baza de date.

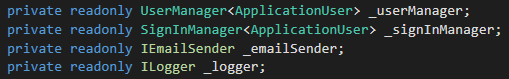
De asemenea acest pattern ne ajută să cream cod mult mai ușor de întreținut și mai ușor de citit.

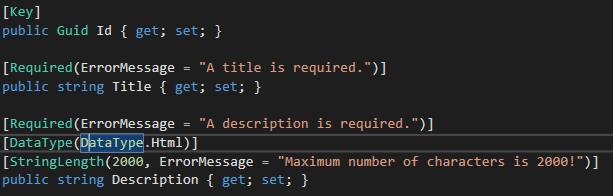
# 6. Best practices

## 6.1. Strcuturarea proiectului pe foldere

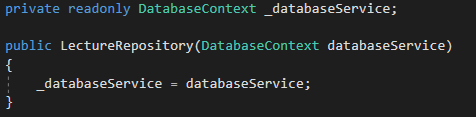
Întotdeauna un cod organizat va fi un cod mult mai ușor de gestionat și de menținut, de aceea organizarea pe foldere a aplicației a fost o prioritate.

## 6.2. Naming conventions



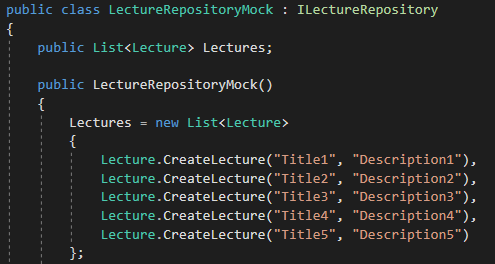


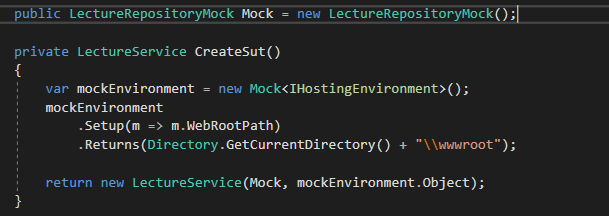
## 6.3. Principiile SOLID



Utilizarea interfețelor

Dependency Injection – ne-a ajutat la crearea de mock-uri



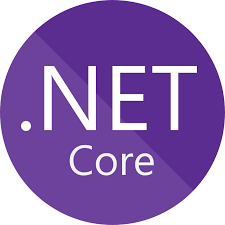
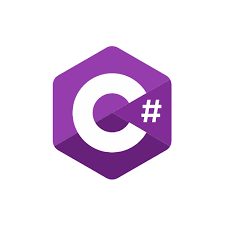


## 6.4. Aplicarea sugestiilor ReSharper

## 6.5. Identarea codului

## 6.6. Utilizarea comentariilor

# 7. Tehnology stack





# 8. API-uri folosite

CK Editor pentru personalizarea texttului introdus la Lectures, Katas si News.