Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42:004.738.12
DOI: https://doi.org/10.24867/04BE26Eminovski

ИЗРАДА ВЕБ АПЛИКАЦИЈЕ ЗА УПРАВЉАЊЕ БИБЛИОТЕЧКИМ ИНФОРМАЦИОНИМ СИСТЕМОМ НА ПЛАТФОРМИ ASP.NET

WEB APPLICATION FOR MANAGING LIBRARY INFORMATION SYSTEM ON THE ASP.NET PLATFORM

Јасмина Еминовски, Милан Видаковић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај: *У раду је анализиран ASP.NET оквир и начин на који је имплементирана апликација за управљање библиотечким информационим системом. Фокус рада је на наменски направљеном object-relation маперу и шаблону репозиторијума помоћу којег се мапер користи. Такође, приказан је и објашњен начин креирања табеларног приказа података помоћу наменски направљеног JavaScript додатка за приказ, филтрирање и сортирање података у табели.*

Abstract: *The paper analyzes ASP.NET framework and one implementation of an application for managing library information system. The focus is on the custom-made object-relation mapper and usage of the repository pattern. The paper also explains a way of creating data tables in UI using a custom-made JavaScript plugin to display, filter and sort data.*

Кључне речи: *.NET, ASP.NET, Repository pattern, custom ORM, JavaScript, plugin.*

1. УВОД

У склопу Мајкрософтове .NET платформе [1] која се бави развојем различитих врста апликација (десктоп, мобилне, итд.), важно место заузимају web апликације. Скуп алата и библиотека намењених за развој web апликација се назива ASP.NET [2]. Ова технологија тренутно представља једну од водећих технологија за развој web апликација у свету. Задатак овог рада је приказ могућности које ASP.NET технологија пружа. Апликација за управљање радом библиотеке – LMS (*Library management system*), не представља реални библиотечки информациони систем, већ један његов модел и треба је посматрати као анализу случаја (енгл. *case study*) за конкретни задатак рада.

2. СПЕЦИФИКАЦИЈА АПЛИКАЦИЈЕ

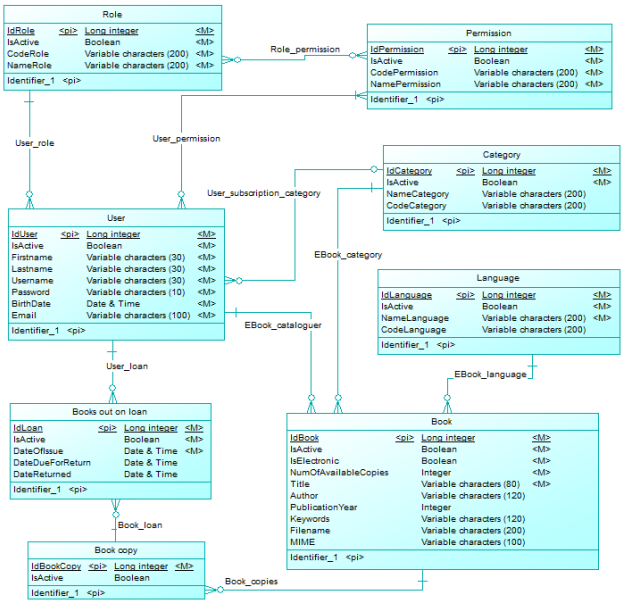
Апликација за управљање радом библиотеке представља информациони систем намењен за лакше управљање пословним процесима у библиотеци али и сервис који могу користити чланови библиотеке. Постоје три типа регирстрованих корисника: администратор, библиотекар и претплатник. Осим њих сервису могу приступи и посетиоци – нерегистровани корисници који имају могућност да виде доступне категорије књига и књиге (традиционалне и електронске).

Сви регистровани корисници поред пријаве и одјаве са система могу да врше преглед и измену личних података, преглед основних ентитета у систему (књиге и категорије), претражују електронске књиге и размењују поруке – чет.

Претплатници додатно могу да виде историју својих изнајмљених књига и да преузму дигиталне књиге које припадају категорији на коју су претплаћени.

Администратор и библиотекар обављају *CRUD* (engl. *create, read, update, delete*) операције над ентитетима. Администратор обавља регистрацију нових корисника било ког типа, док библиотекар може додати само новог претплатника.

Поред тога обавља и задуживање и раздуживање књига у систему. Модел података приказан је на слици број 1 у наставку.



Слика 1. *Дијаграм модела података*

Сваки ентитет има идентификатор и индикатор активности који служи за логичко брисање појаве ентитета. Поред улога у систему користе се и дозволе за обављање одређених активности. Дозволе се могу директно доделити кориснику од стране администратора.

НАПОМЕНА:
Овај рад је проистекао из мастер рада чији ментор је био проф. др Милан Видаковић.

3. ТЕХНОЛОГИЈЕ И АЛАТИ

ASP.NET MVC 5

Постоје три различите технологије које се могу користити у оквиру ASP.NET оквира: Web Forms [3], MVC [4], i ASP.NET Web Pages [5]. Сходно претходном знању и искуству у раду коришћена је MVC технологија у верзији 5.2.3 која се ослања на истоимени архитектонски шаблон. *Model-View-Controller* [6] је један од најважнијих архитектонских шаблона у рачунарству и заснива се на подели архитектуре према одговорности на ове три компоненте. Модел представљају класе које описују податке. Поглед – *view* представља изглед апликације односно *.cshtml* странице. Контролер је задужен за комуникацију између корисника и тока апликалације односно начина функционисања. MVC је у ASP.NET оквиру покривен одређеним конвенцијама које олакшавају процес развоја апликације.

RAZOR СИНТАКСА

Razor [7] синтакса је једноставан начин за уградњу програмске логике у web страницу. Ова синтакса је основ view компоненте у ASP.NET MVC. На *.cshtml* страницама могу постојати две врсте садржаја: клијентски и серверски. Серверски код је задужен за логику и динамичко креирање клијентског садржаја кога чине HTML елементи, CSS стилови или нека JavaScript логика. Оваквом употребом серверског кода омогућено је комплетно креирање web апликација без посебног познавања неке клијентске технологије, а отуда и потиче назив ASP (*active server pages*). Сама синтакса је врло једноставна и нема много правила.

ИМПЛЕМЕНТАЦИОНО ОКРУЖЕЊЕ

Коришћен *је Microsoft Visual Studio Community 2015* [8].

БАЗА ПОДАТАКА

За чување података коришћена је *Microsoft SQL Server 2014* релациона база података и *Microsoft SQL Server Management Studio 2014*. Иако је база релационог карактера, није коришћено ограничење референцијалног интегритета, већ је брига о вези између података обављена на апликативном нивоу.

LUCENE .NET

За потребе рада са електронским књигама коришћена је библиотека Lucene.NET [9] у верзији 4.8.0 која представља *open-source* програмску библиотеку за индексирање, претрагу и анализу текста. Електронске књиге представљају .pdf фајлове приликом чијег отрпремања се врши индексирање мета-података и садржаја ради напредне претраге.

SIGNALR

Да би обављање фунцкија било могуће у реалном времену коришћена је библиотека SignalR [10] која је део ASP.NET оквира. Примену је у конкретној имплементацији нашла за потребе размене порука између корисника – чет. Овим приступом је омогућена двосмерна комуникација између клијента и сервера Сервер може да иницира неку функционалност на клијентској страни ако је остварена потребна конекција и ако је клијент доступан. SignalR се заснива на принципу мрежних сокета (*web socket*) [11].

4. ИМПЛЕМЕНТАЦИЈА

Пројекат чине два дела: први ASP.NET MVC апликација и други DAL (енг. *data access layer) class library* део намењен за рад са базом. MVC апликација има референцу на DAL потпројекат. Део клијентске логике је обављен уз помоћ *razor* синтаксе, али се велики део базира на чистом JavaScript језику и *jQuery [12]* библиотеци.

СТРУКТУРА И КОНФИГУРАЦИЈА

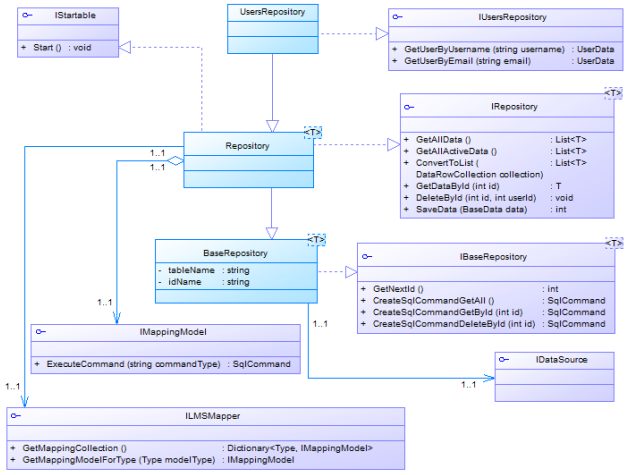
MVC апликација представља скуп компоненти које функционишу по приципима MVC архитектонског шаблона. Приликом креирања овакве апликације аутоматски се генеришу главни фолдери намењени за груписање класа модела, контролера и *view* фајлова. Том приликом настаје и фолдер који садржи статичке ресурсе за скрипт, стил фајлове и слике. Конфигурациона логика неопходна за покретање апликације је у *Global.asax* фајлу. У њему се обавља регистрација DI (енг. *dependency injection*) *Autofac* [13] контејнера неопходног за разрешавање зависности између класа у апликацији. *Global.asax* такође служи за регистрацију свих зона (енг. *area*), рута, филтера и пакета скрипт и стил фајлова који се користе на клијентској стани. Ради чистије архитектуре коришћен је концепт зоне – *area*, која представља MVC шаблон у малом. За сваки ентитет постоји посебна зона у којој се налазе контролери и *view* странице. Одвојен је део који обухвата глобалне сервисе (за ауторизацију и активацију одређених инфраструктурних елемената) од дела који обухвата пословну логику и дела који обухвата класе инфраструктуре (помоћни сервиси за валидацију, филтрирање, сортирање, изградњу објеката итд.). Такође, груписани су у фолдере делови намењени за индексирање и претрагу електронских књига и део за размену порука.

LMS OBJECT-RELATION MAPPING И РАД СА БАЗОМ

У овом поглављу је описан део архитектуре задужен за рад са базом података и обављање основних CRUD (engl. *create*, *read*, *update*, *delete*) операција. Имплементирано решење се заснива на *Repository* шаблону [14] без коришћења *Entity Framework*-а [15] уз *custom object-relational* мапер компоненту.

Сврха шаблона репозиторијума је сакривање логике чувања, добављања и мапирања података из базе у доменске моделе и обрнуто. Класа која представља омотач око базе података је *SqlServerDatabase*. Она користи елементе ADO.NET [16] библиотеке и представља извор података за репозиторијум. Репозиторијума има онолико колико и ентитета односно табела у бази. Разликују се три нивоа апстракције. Најнижи ниво апстракције представља *BaseRepository* са заједничким операцијама за све табеле – читање и брисање. Ове SQL команде са

разликују у називу табеле и називу колоне која представља идентификатор. *Insert* и *update* команде се разликују код сваког доменског модела и зависе од пропертија тог модела. У репозиторијуму су ове операције обухваћене методом *Save* и налазе се на другом нивоу апстракције у овиру *Repository* класе. У оквиру ове класе се преко одговарајућег *MappingModel*-а се обављају неопходне конверзије и генеришу команде за базу. На највишем нивоу апстракције се налазе специфичне методе везане за одређени ентитет и табелу. У примеру испод је у приказан *UsersRepository* који има методе за добављане корисника према корисничком имену и е-маилу поред основних метода које наслеђује.



Слика 2 - *Class* дијаграм за имплементирани *Repository* шаблон

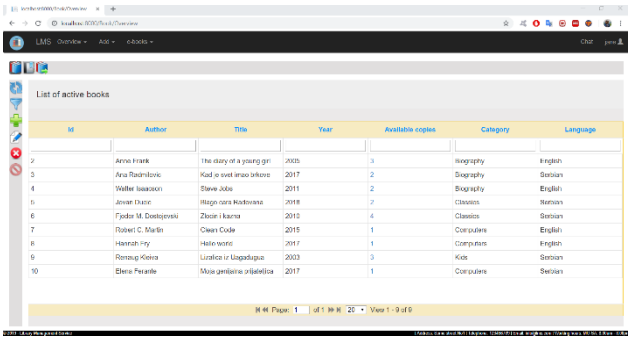
MappingModel је генеричког типа и веже се за тип ентитета о чијим конверзијама и командама за чување ће бити задужен. Сваки доменски модел има пропертије, који се преко атрибута *DBColumn* означавају као одговарајуће колоне у бази. Типови података не морају бити исти у бази као и у доменском моделу у апликацији. За конверзију типова се приликом мапирања података користе методе које се добављају помоћу својих назива и складиште у колекције објекта *MappingModel* класе. Како би логика била генерична коришћени су делегати и стабла израза за њихову припрему. *LMSMapper* класа има улогу да повеже репозиторијум са мапинг моделом и тиме омогући позив метода са листинга у наставку преко мапинг модела у оквиру репозиторијума.

```
public T ExecuteConversion(DataRow dataRow)
{
  return conversionStatement(dataRow);
}

public SqlCommand ExecuteCommand(string commandType)
{
  SqlCommand command;
  string query = "";

  if (commandType.Equals(ORMConstant.Insert))
  {
    command = insertCommandStatement(
      (T)Convert.ChangeType(Model, typeof(T)));
    query = ExecuteInsertQuery();
  }
  else
  {
    command = updateCommandStatement(
      (T)Convert.ChangeType(Model, typeof(T)));
    query = ExecuteUpdateQuery();
  }
  command.CommandText = query;
  return command;
}
```

Листинг 1. *Методе доступне из MappingModel<T> класе*

LMSGRID

LMSGrid представља наменски направљен JavaScript додатак – *plugin* за табеларни приказ података на web страници. Преко ове компоненте се врши приказ највећег броја ентитета у оквиру апликације (књиге, категорије, корисници итд.). Овако креирана табела се може прилагодити у смислу опција које пружа кориснику. Могуће је филтрирати, сортирати, страничити податке. Такође, у зависности од пратећег *sidebar*-а и акција на њему, грид може да промени стање и прошири се *checkbox* елементима који омогућавају селекцију одређеног реда у табели. На слици 3 је пример табеле креиране помоћу *LMSGrid* додатка



Слика 3. *LMSGrid табела са приказом активних књига*

За инстанцирање компоненте главни податак је путања до акције контролера која добавља *view* моделе који ће се приказати у оквиру табеле. Поред путање потребно је проследити информације о колонама: натпис, назив који се подудара са називом пропертија у *view* моделу, тип, опционо стил, ширину, путању ка некој другој акцији итд. Припрема података за филтрирање се обавља на клијентској страни. Исти *endpoint* се активира у случају да се добављају сви подаци или се корисник врши филтрирање. На листингу 2 се може видети метода за добављање података из примера.

```
public JsonResult GetAllActive(FilterSorterModel    filterSorterModel)
{
  var viewModels = BookService.GetAll(true);
  var filterSorter = new
    DataCollectionFilterSorter<BookViewModel>();
  IEnumerable<BookViewModel> enumBooks = viewModels
    .AsEnumerable();
  enumBooks = filterSorter.FilterAndSort
    (enumBooks,filterSorterModel);
  return Json(enumBooks,
    JsonRequestBehavior.AllowGet);
}
```

Листинг 2. *Метода за добављање активних књига*

Кључни елемент је *DataCollectionFilterSorter* класа која обавља филтрирање и сортирање помоћу прослеђених података са клијентске стране уз ослонац на *LINQ* [17] и делегате. За сортирање је довољна само *OrderBy* операција из *LINQ*. За филтрирање је потребно направити делегат који ће на основу прослеђеног назива пропертија, тражене вредности, формата и типа траженог података и *view* модела, вратити *bool* вредност као индикатор припадности тог истог модела филтрираној колекцији.

5. ЗАКЉУЧАК

Задатак овог рада била је демонстрација рада .NET платформи и њеном оквиру намењеном web апликацијама – ASP.NET. Проблем библиотечког система треба посматрати као case-study за побољшање вештина у раду са базом, структурама података, JavaScript објектима… Такође имплементација апликације, која је основ овог рада, је служила за боље упознавање са архитектонским и дизајн шаблонима као што су *MVC*, *Repository*, *Builder*, *Factory* и други.

У раду нису приказани сви делови решења већ они који би читаоцу били можда најзанимљивији. Интересантно би такође било описати начин размене порука између корисника апликације помоћу SignalR технологије, као и индексирање и претрагу електронских књига помоћу Lucene.NET библиотеке. Ови делови имплементације се у великој мери своде на коришћење готовог кода из библиотека па је у раду остављен простор за опис компоненти које су јединствене и специфичне за ово конкретно решење.

Показало се да ASP.NET оквир оставља пуно простора за сопствене идеје и решења, али истовремено пружа стабилно окружење за брз и ефикасан развој web апликација. Конфигурација је врло једноставна и лако је укључити додатне библиотеке.

Овај систем би се могао унапредити у више различитих праваца. Филтрирање помоћу *LMSGrid* компоненте би могло бити проширено додатним параметром по каквом услову се одређени појам филтрира: услов једнакости, услов садржавања, услов веће/мање за бројеве. Тренутно је могуће филтирати само по једнакости за било који тип. Из перспективе корисничког интерфејса унос појмова за филтрирање би могао да се унапреди тако да се једноставна инпут поља замене компонентама попут *datepicker*-а или *selectbox*-а. У раду са базом би у случају усложњавања шеме и повећања броја рекорда у бази, требало би размотрити о увођењу референцијалног интегритета ради избегавања потенцијалних проблема на апликативном нивоу. *Repository* механизам би у том случају могао и да се прошири неком врстом кеш колекција које би допринеле побољшању перформанси смањењем броја упита ка бази.

6. ЛИТЕРАТУРА

[1] Microsoft .NET, https://dotnet.microsoft.com/
[2] ASP.NET, https://docs.microsoft.com/en-gb/aspnet/overview
[3] ASP.NET Web Forms, https://docs.microsoft.com/en-gb/aspnet/web-forms/
[4] ASP.NET MVC, https://docs.microsoft.com/en-gb/aspnet/mvc/mvc4
[5] ASP.NET Web Pages, https://docs.microsoft.com/en-gb/aspnet/web-pages/
[6] MVC pattern, https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
[7] Razor view, https://weblogs.asp.net/scottgu/introducing-razor
[8] Visual Studio IDE, https://visualstudio.microsoft.com/vs/
[9] Lucene.NET, https://lucenenet.apache.org/
[10] Signal R, https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr
[11] Web sockets, https://medium.com/@dominik.t/what-are-web-sockets-what-about-rest-apis-b9c15fd72aac
[12] jQuery, https://jquery.com/
[13] Autofac, https://autofaccn.readthedocs.io/en/latest/index.html
[14] Repository pattern, https://deviq.com/repository-pattern/
[15] Entity Framework, http://www.entityframeworktutorial.net/what-is-entityframework.aspx
[16] ADO.NET, https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/
[17] LINQ, https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/

Кратка биографија:

Јасмина Еминовски рођена је 2.8.1994 у Војнићу, општина Војнић, Р.Хрватска. 2013. године је уписала Факултет техничких наука, одсек *Рачунарство и аутоматика.* Након завршених основних академских студија, уписала је мастер академске студије на истом факултету, смер *Примењене рачунарске науке и информатика*, *Електронско пословање*.

Милан Видаковић је рођен у Новом Саду 1971. године. На Факултету техничких наука у Новом Саду завршио је докторске студије 2003. године. На истом факултету је 2014. године изабран за редовног професора из области *Примењене рачунарске науке и информатика*.