|  |  |
| --- | --- |
|  | Zbornik radova Fakulteta tehničkih nauka, Novi Sad |

**UDK (Upisuje biblioteka, ostaviti jedan red)**

**РАЗВОЈ WEB АПЛИКАЦИЈА УПОТРЕБОМ .NET CORE 2.0 ОКВИРА**

**DEVELOPMENT OF WEB APPLICATIONS USING .NET CORE 2.0 FRAMEWORK**

Јован Попов, Милан Видаковић*, Факултет техничких наука, Нови Сад*

**Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО**

**Кратак садржај:** *У овом раду је анализиран .NET Core 2.0 оквир, и његова примена, у комбинацији са Angular оквиром, за развој web апликација. Специфицирани су захтеви и дизајн, имплементирана и тестирана је web апликација која прукупља податке о извођачима са Twitter, Wikipedia и Allevents.in API-ја**.*

**Abstract:***This paper analyses the .NET Core framework, and its use, in combination with the Angular framework, for the development of web applications. It specifies the requirements and the design and describes the implementation and the testing of a web app that collects data about artists from Twitter, Wikipedia and Allevents.in APIs.*

**Кључни речи:** *.NET Core, Angular, Twitter, Wikipedia, Allevents.in.*

1. **УВОД**

Микрософтова .NET платформа [1] постоји још од 2002. године, и њена библиотека ASP.NET [2] je тренутно једна од водећих технологија које се употребљавају за развој web апликација, поред и других функционалности које .NET пружа, као што су мобилне и десктоп апликације. Од 2016. године, паралелно са развојем тренутно стабилне верзије .NET библиотеке, која је тренутно у употреби у комерцијалним пројектима, Микрософт је започео развој најновије верзије њихове библиотеке, .NET Core, а заједно са њиме новог стандарда за .NET платформе, који се зове .NET standard [3]. .NET Core је развијан од нуле, уместо да се надовезују на претходне верзије, и он је за разлику од претходних имплементација, *open source* и такође *cross-platform*.

1. **.NET CORE**

.NET Core [4] је развојна платформа која се одржава од стране Микрософта и .NET заједнице. Основне карактеристике које дефинишу .NET Core су да је *cross-platform* и да подржава Windows, macOS и Linux, oмогућена *command-line* подршка, компатибилност са .NET framework-om [5], Xamarin-om [6] и Mono-om [7], пошто имплементира .NET Standard. Такође, .NET Core подржава *Dependency Injection* механизам унутар своје архитектуре и доноси измене у својој верзији ASP.NET библиотеке, од којих је најбитнија да MVC и WebApi контролери деле исту класу.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**НАПОМЕНА:**

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

.NET Core се састоји од:

* .NET *Runtime*-a, који пружа систем типова, учитавање асемблија, JIT (just in time) компајлер, *garbage collection*, и сличне основне сервисе.
* Основног сета *framework* библиотека које пружају примитивне типове података, класе за колекције, и друге фундаменталне функционалности.
* SDK алата и компајлера језика, које омогућавају основно развојно искуство.
* ‘Dotnet’ *app host*-а, који се користи за покретање .NET Core апликација.

Језици који се могу користити за писање апликација за .NET Core су C#, Visual Basic и F#.

**.NET Standard**

Један од изазова код тренутног развоја .NET-а је тај да се временом дошло до више .NET имплементација. Постоји .NET framework, Xamarin, Mono, Core, и сваки има своје API-је. Приликом развоја Core-a, дошло се до жеље да .NET буде конзистентан на свим платформама, па је развијен .NET стандард, који је спецификација за .NET. Што значи да уколико неко жели да направи .NET платформу која задовољава стандард, онда мора да имплементира сет API-ja прописаних стандардом, који програмери могу да очекују од .NET платформе. Употребом овог стандарда, код може лако да се преноси између различитих .NET платформи.

**.NET Core 2.0**

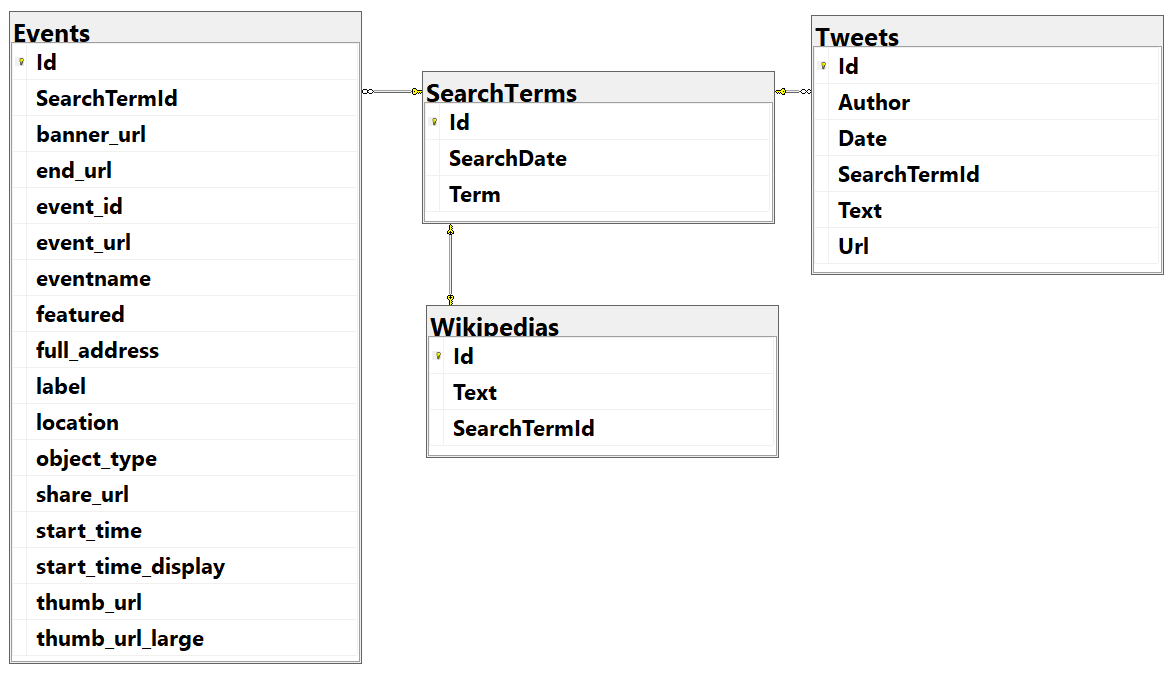
Актуелна верзија .NET Core библиотеке je .NET Core 2.0, издата 14. августа 2017. године. Она између осталог доноси са собом оптимизацију рада, најновију верзију *Entity Framework Core* библиотеке, *Razor pages* као нов начин за израду web апликација и најновију верзију .NET стандарда, NET Standard 2.0.

1. **СПЕЦИФИКАЦИЈА АПЛИКАЦИЈЕ**

Задатак обухвата израду апликације која је заснована на .NET Core 2.0 i Angular 5 технологијама. Апликација омоћуава кориснику да се региструје, пријави, одјави, и претражује извођаче помоћу поља за претрагу. Након уноса извођача, добаве се информације о њима са Allevents.in [8], Twitter [9], и Wikipedia [10] сајтова и прикажу се. Идеја пројекта је да се резултати претраге одређеног извођача чувају у базу, и следећи пут кад се исти извођач претражује унутар одређеног временског периода, уместо да се опет врше позиви свих API-ја, подаци се прочитају из базе и прикажу знатно брже.

Да би се могле користити функционалности сајта, корисник мора бити регистрован у систему, и затим пријављен. Уколико покуша да приступи страници за претрагу а није пријављен, биће преусмерен на страницу за пријаву, на којој се налази линк за регистрацију. Уколико је регистрација успешна, корисник се преусмерава на страницу за претрагу. На њој уноси назив извођача у поље за претрагу и резултати се приказују испод поља. Резултати за сваки засебни API су подељени по страницама уколико их има више, и увек је могућ унос новог извођача што ће освежити табеле са резултатима. Корисник се може у било ком момeнту вратити на страницу за пријаву кликом на “Logout”.

За потребе пројекта коришћена је MS SQL Server база података. Она је потребна за управљање корисничким налозима, и такође за чување података које смо добили за различитих API-ja приликом претраге. Управљање корисничким налозима је реализовано упоребом *Identity Framework* библиотеке, које је део ASP.NET-а, и која са собом доноси све табеле неопходне за кориснике, улоге, и остале функционалности. Модел података који се користи за чување резултата претраге се налази на слици 1.



Слика 1 - Дијаграм модела података

1. **ИМПЛЕМЕНТАЦИЈА**

Пројекат је у основи .NET Core 2.0 web апликација. Core се првенствено користи као *back-end*, дакле за имплементацију REST API сервиса, док се за *front-end* користи Angular клијентска апликација. Пројекат је развијан у *Visual Studio* развојном окружењу.

**.NET Core web апликација**

Пројекат користи шаблон који такође долази са Core 2.0 издањем, који омогућава лако започињање .NET Core 2.0 web пројеката који користи Angular као клијентску апликацију. начин на који овај шаблон функционише, је да је се унутар Core 2.0 web апликације налази фолдер са једноставном Angular клијентском апликацијом, који се састоји од TypeScrypt фајлова који морају да се *transpile*-ују, односно преведу у JavaScript фајлове, што корисник не мора ручно да ради, већ се ово аутоматски уради приликом покретања, односно компајлирања .NET пројекта који се налази у основи. Начин на који је ово урађено је употребом .NET CORE *middleware*-a која се зове *Node Services*, који омогућава комуникацију са Node.js-om из C# кода, који врши превођење TypeScript фајлова.

Овај шаблон такође корiсти WebPack, који се користи да би оптимизовао учитавање JavaScript, CSS и HTML фајлова са сервера на претраживач, и такође омогућава „*hotswap*“ функционалност приликом рада са пројектом, која док је пројекат покренут, слуша у позадини да ли је дошло до неке промене у коду, и у колико јесте, преко *Node Services* *middleware*-a поново искомпајлира клијентски део апликације и аутоматски освежи web апликацију, без потребе да се сам пројекат рестартује.

**Entity Framework**

За рад са базом се користи *Entity Framework Core* 2.0, најновија верзија Микрософтовог *Entity Framework* оквира за рад са базом. Основна класа преко које се врши интеракција је DbContext. Постоје две врсте контекста у .NET оквиру, IdentityDbContext и DbContext. IdentityDbContext користи *Identity Framework* за рад са корисницима, док се DbContext, користи за ентитете у које се чувају подаци у апликацији који нису везани за кориснике.

Оно што је прво битно направити, је класу која наслеђује DbContext, и у њој је неопходно додати као property-је колекције типа DbSet, које представњају ентитете који ће се чувати у бази. Класа која представља модел податка је стандардна POCO (Plain old c# object) класа, и притом, није потребно користити анотације да би се тачно назначило који property је кључ, или страни кључ, пошто је библотека у стању да сама протумачи садржај класе и из ње закључи њене карактеристике, користећи често коришћене шаблоне и конвенције за називе, мада се могу такође и анотације користити уколико постоји потреба. Након што смо дефинисали моделе, потребно је у *Package Manager* конзоли прво укуцати „Add-Migration“, да би се миграција направила, и затим „Update-database“ да би се промене извршиле над базом.

Након што смо ово све подесили, објектима из базе се приступа преко контекста, који нам је доступан преко *Dependency Injection* механизма. Овај објекат затим у себи садржи све DbSet-ове који представљају ентитете из базе, и којима можемо приступити, и затим вршити CRUD операције над њима, и такође приступати објектима са којима постоје везе. Све ове операције су већ имплементиране као део Identity библиотеке, помоћу *Linq* оператора, као што су Select, OrderBy, и тако даље.

**Identity Framework**

За управљање корисничким налозима, регистрацију, пријаву, чување корисника у бази, користи се *Identity Framework*, Микрософтова библиотека за ову сврху, која је део ASP.NET-a, и чија најновија верзија долази са Core 2.0 издањем. Приликом имплементације неопходно је регистровати контекст за базу које је повезан са *Identity* оквиром тако што ћемо направити класу која наслеђује IdentityDbContext, који у себи садржи моделе преко којих чува регистроване кориснике и друге операције везане за њих, од којих се добију табеле у бази након што се уради миграција. Такође је потребно *Identity* сервис додатно конфигурисати у методи за конфигурисање сервиса, и такође направити методе за регистровање и пријаву. Након тога, он омогућава аутентификацију и ауторизацију корисника додавањем [Authorize] анотација на API контролере, које такође као параметре могу да имају, на пример, неопходну улогу корисника.

**Angular 5**

Angular [11] је тренутно једна од најпопуларнијих JavaScript библиотека за прављење *Single Page* апликација. Прва верзија библиотеке се звала AngularJs, и она је и даље веома популарна. Верзија 2.0 је донела много измена, за почетак, употребу TypeScript-а, затим контролери су замењени са компонентама, и уз доста додатних измена, Angular 2 (без „Js“ у свом називу) је постао практично нови оквир. Тренутно актуелна верзија је 5, мада од верзије 2 па на даље, измене нису толико радикалне.

**Twitter API**

Twitter API омогућава различите функционалности везане за Twitter платформу, као што су операције над налозима, претраживање Tweet-ova по различитим параметрима, слање порука и додатне функционалности. Да би се користио, неопходно је регистровати се на њиховом порталу, након чега се добију кључеви помоћу којих ће се ауторизовати позиви API-ја. Приликом имплементацијеје коришћена библиотека TweetSharp [12], која пружа веома ефикасан начин да се позову *endpoint*-и Twitter REST API-ја

**Wikipedia API**

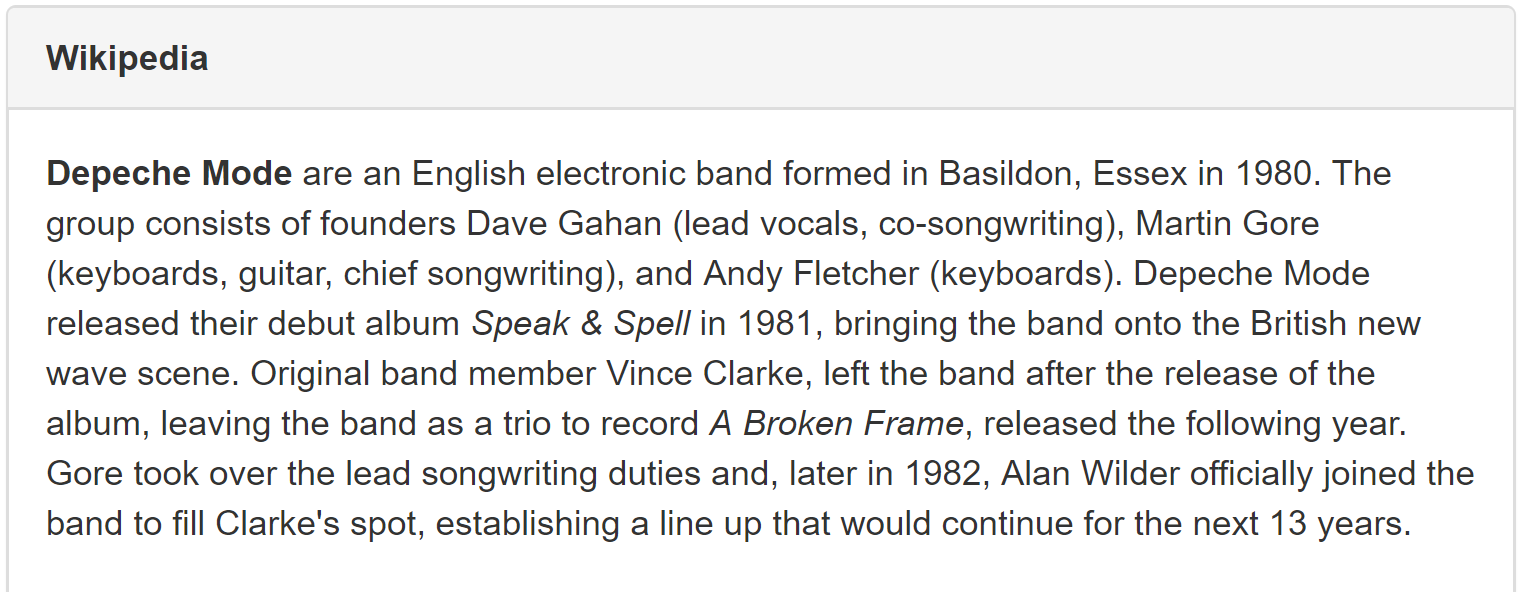
Подаци са Wikipedije се могу преузети употребом MediaWiki [13] API-ја. АPI се користи употребом *query* параметара, као што су „action“ параметар, који може да прима различите вредности, од којих је најбитнија „query“, који назначава да желимо да вршимо претрагу садржаја, затим постоји „format“ параметар, са којим се може назначити формат одговора. Следећи битан параметар је „prop“, што је скраћено од „property“, којим се може може назначити које конкретне информације о одређеној страници нам требају, док „titles“ параметар, служи да проследимо назив странице коју желимо да пронађемо.

**AllEvents.in API**

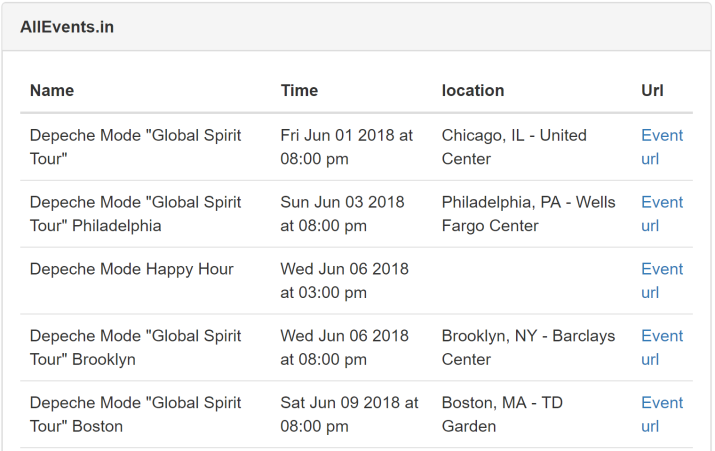
AllEvents.in је сајт који прикупља Facebook догађаје, и чува их у својој бази. Он је посебно користан звог свог API-ja, који је корисна алтернатива за Facebook API, са којим је већ неко време добављање догађаја изузетно тешко. AllEvents.in API је још увек у приватној бета фази, и потребно је регистровати се, и сачекати да се одобри регистрација. Затим се може добити API кључ који ће се користити приликом позива API-ја и такође Url за портал за рад са API-jem, на којем се могу видети примери позива за више програмских језика. API омогућава претрагу догађаја по називу, по географској локацији, по граду и по организатору.

1. **ДЕМОНСТРАЦИЈА**

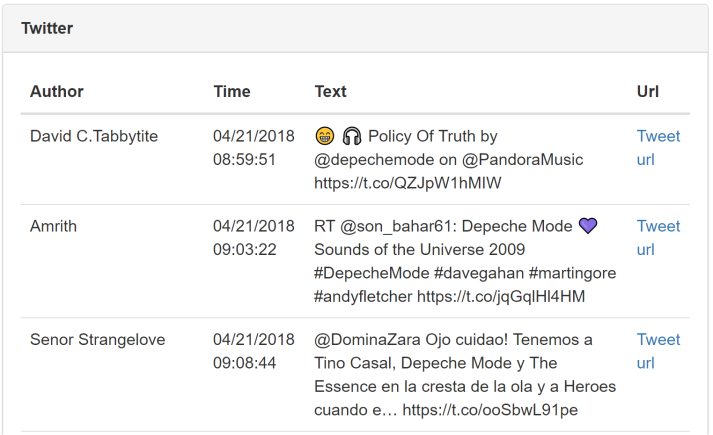
Страница за пријаву је прва коју корисник види, са ње се може прећи на страницу за регистрацију кликом на линк „Register“. Након успешне регистрације, корисник је преусмерен на страницу за претрагу где се у поље за претрагу уноси назив извођача, и након тога ће се појавити резултати претраге испод поља за унос, у засебним картицама за сваки сајт, што се може видети на сликама 2, 3 и 4.



Слика 2 - Резултати са Wikipedia сајта



Слика 3 - Резултати са AllEvents.in сајта



Слика 4 - Резултати са Twitter сајта

Након тога, могуће је отићи на линк пронађене Wikipedia странице кликом на „See more“, на страницу за догађај на AllEvents сајту, и такође отићи на страницу конкретног Tweet-а.

1. **ЗАКЉУЧАК**

Задатак овог рада је био да се истражи .NET Core 2.0 оквир, да се имплементира апликација кроз коју ће се демонстрирати његова употреба, и да се упореди његов рад, имплементација, предности и мане у односу на претходну верзију .NET оквира.

Главна предност Core оквира, у односу на .NET Framework оквир, је то да у основи подржава више платформи, уместо само Windows платформе. Осим тога, боље перформансе, специфичне могућности и генерално израда оквира од нуле, са применом свега што је научено кроз израду претходне верзије оквира, су такође предности.

Мана Core оквира је првенствено то да је још увек је у изради, која већ више година траје, и нису имплементиране све функционалности. Ово се може видети приликом рада са базом. Верзија Entity Framework-а која је била део претходне верзије .NET оквира је била веома користан ORM, пошто је веома квалитетно подржавао различите приступе рада са базом и такође је био веома лак за употребу, док тренутна верзија Entiy Framework Core библиотеке, која је део .NET Core-a има само део тих функционалности. Пре израде апликације употребом .NET Core 2.0 оквира, саветује се да се истражи које функционалности су тренутно подржане и да се из тога процени да ли тренутно има смисла његова употреба за тај конкретан пројекат.

Што се тиче даљег истраживања на ову тему, првенствено је битно пратити развој Core оквира. Нове верзије редовно излазе, и већ је најављена верзија 2.1. Што се проширења апликације тиче, њена основна идеја је прикупљање података са више извора, и могу се лако додати нови извори за прикупљање података поред тренутно коришћених.

1. **ЛИТЕРАТУРА**

[1] Microsoft .NET, https://www.microsoft.com/net/

[2] ASP.NET, https://www.asp.net/

[3] .NET standard, https://docs.microsoft.com/en-us/dotnet/standard/net-standard

[4] .NET Core, https://docs.microsoft.com/en-us/dotnet/core/

[5] .NET framework, https://www.microsoft.com/net/download/dotnet-framework-runtime

[6] Xamarin, https://www.xamarin.com/

[7] Mono, https://www.mono-project.com/

[8] Allevents.in, https://allevents.in/

[9] Twitter, https://twitter.com/

[10] Wikipedia, <https://www.wikipedia.org/>

[11] Angular, <https://angular.io/>

[12] TweetSharp, <https://github.com/shugonta/tweetsharp>

[13] MediaWiki API, https://www.mediawiki.org/wiki/API:Main\_page

**Кратка биографија:**

**Јован Попов** је рођен 1.9.1988, Нови Сад, општина Нови Сад, Србија. 2007. године уписао је Факултет Техничких Наука, одсек *Рачунарство и Аутоматика.* Након завршених основних студија уписао је мастер академске студије на истом факултету, смер *Примењене рачунарске науке и информатика*, *Електронско пословање*. Положио је све испите предвиђене планом и програмом.

**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*.