Na kratko

Eventiqon je spletna aplikacija za iskanje, prijavo, beleženje ter organizacijo dogodkov in aktivnosti. Poleg osnovnih funkcionalnosti kot sta vnos in upravljanje dogodkov/aktivnosti, nam aplikacija omogoča tudi iskanje, izvoz, naročanje in rangiranje aktualnih dogodkov in pa samo komunikacijo med udeleženci.

Eventiqon

Faza 2 : zaledni del spletne strani ASP.NET MVC 5 + MS SQL SUPB.

Aleksander Tomić

Vpisna številka: 63130259

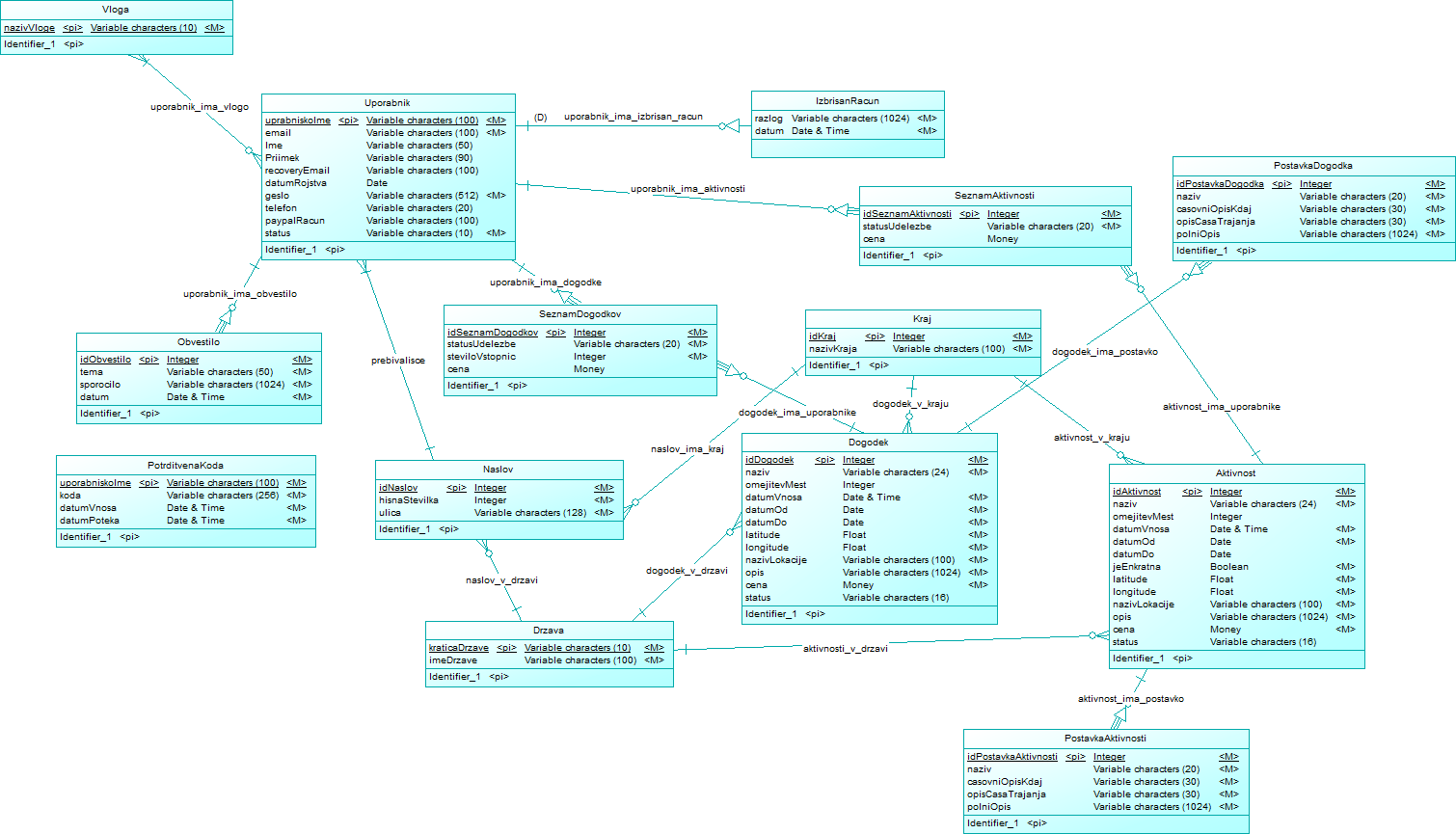
# Načrtovanje in implementacija zalednega dela spletne strani

Izbira primernega ogrodja

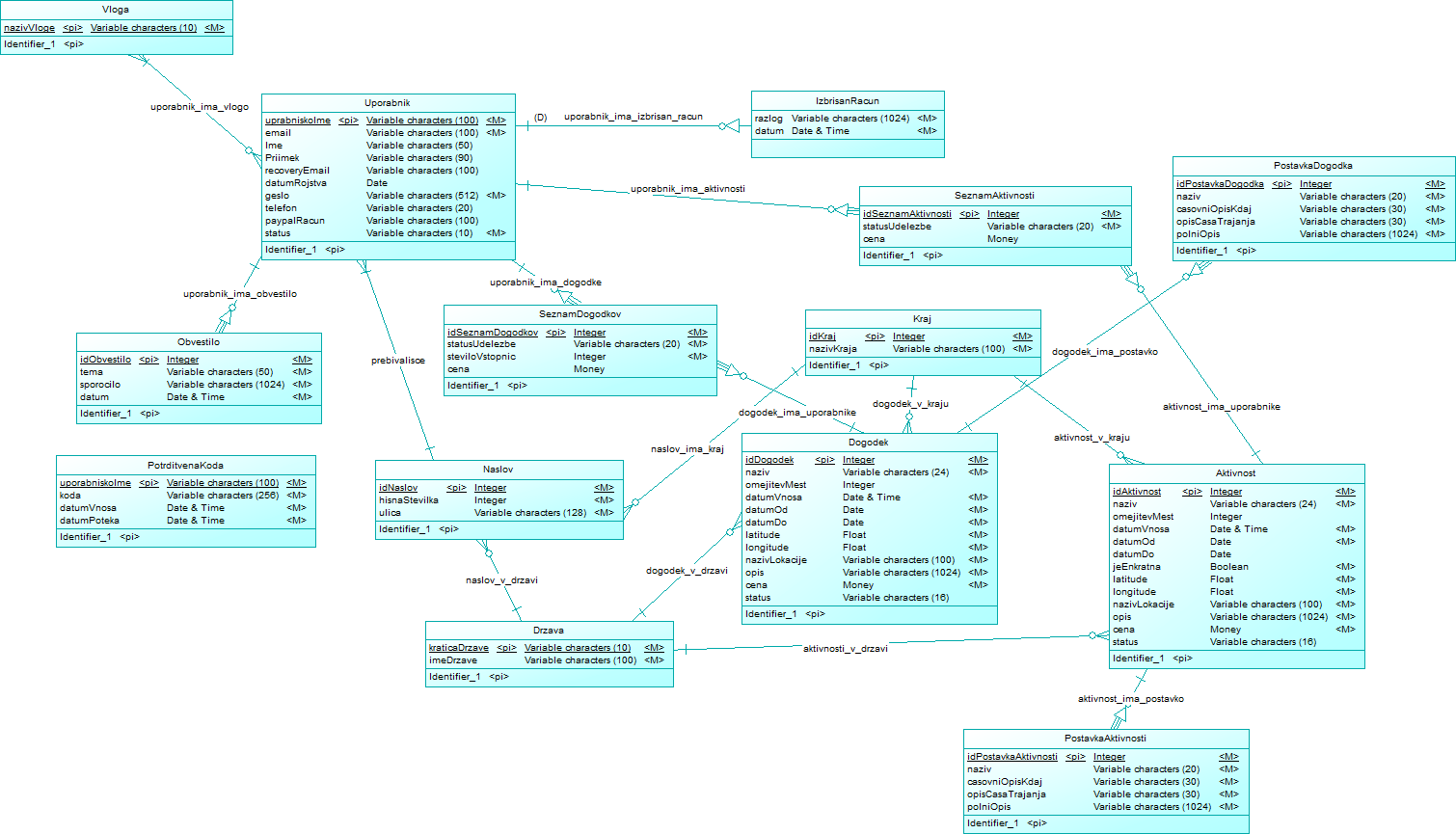
Za ogrodje spletne strani sem si izbral **.NET ASP.NET MVC 5**, saj ponuja dobro strukturo in datotečno organizacijo, poleg tega pa je deležen odlične podpore razvojnega okolja Visual Studio (2015 Community) in skupnosti razvijalcev.

## Načrtovanje podatkovnega modela

Za ciljni SUPB sem si izbral MS SQL Server 2012, konceputalni model pa sem naredil v programskem orodju PowerDesigner 12.5. Konceptualni in logični model lahko najdete v mapi »Konceptualni model«, poleg tega pa se tukaj nahajajo tudi skripte za kreacijo fizičnega modela (DDL stavki). Prožilcev in transakcij na logičnem nivoju nisem definiral, saj za transakcije uporabljam razrede in objekte, ki mi jih ponuja ASP.NET ogrodje. Na spodnji sliki lahko vidite konceptualen model problemske domene »organizator dogodkov«, oziroma bolje rečeno, portala Eventiqon.

Na prvem izseku konceputalnega modela, so bistvenega pomena entitete, ki so v tesnem stiku z uporabnikom – torej vloge uporabnika (ki jih ima lahko več), povezava m:n, kar se kasneje, na logičnem nivoju preslika v vmesno tabelo. Uporabnik ima tudi obvestila celotnega sistema (ne gre za privatna spročila med uporabniki), v primeru izbrisa računa se podatki uporabnika fizično ne izbrišejo, spremeni se le status uporabnika, v tabelo izbrisanRacun pa se doda nova vrstica ki vsebuje razlog in datum izbrisa. Spodnja slika na naslednji strani, predstavlja preostali del konceputalnega modela, kjer so dejanski dogodki in aktivnosti, na katere se uporabnik lahko naroči/prijavi

Tukaj bi bilo smiselno dodati prožilec, ki v primeru dejanske prijave poskrbi, da se v Seznam[Aktivnosti|Dogodkov] prepiše trenutna vrednost cene dogodka/aktivnosti, saj se slednja lahko spreminja. Informacije o udeležbi se hranijo v vmesnih tabelah SeznamAktivnosti (slika spodaj) in SeznamDogodkov (slika zgoraj). Ker ima lahko vsak dogodek več poddogodkov/aktivnosti, sta tu entiteti PostavkaDogodka in PostavkaAktivnosti. Vse skupaj bi lahko posplošili in izvedli generalizacijo, toda zaradi večje izrazljivosti modela, se za to nisem odločil (trenutna oblika je zelo podobna logičnemu modelu).



## namestitev spletne aplikacije

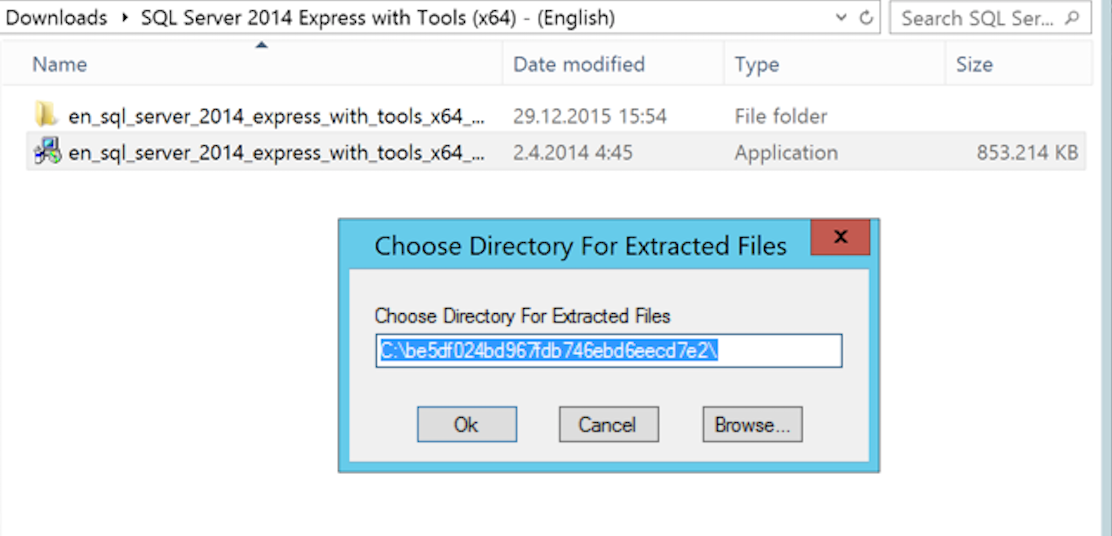
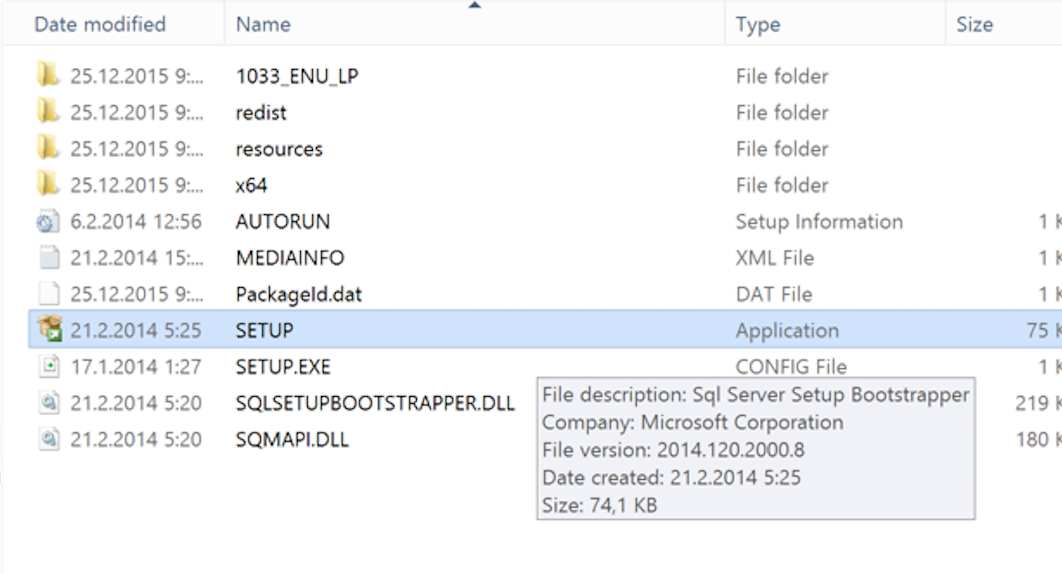
Zahtevano programsko okolje  
Za razvoj in testiranje aplikacije je potrebno imeti ustrezen operacijski sistem, sam sem aplikacijo poganjal na OS Windows Server 2012, uporabljal pa sem razvojno okolje:   
**Visual Studio Community 2015,** ki ga lahko prenesete iz naslednje strani: <https://www.visualstudio.com/products/visual-studio-community-vs>

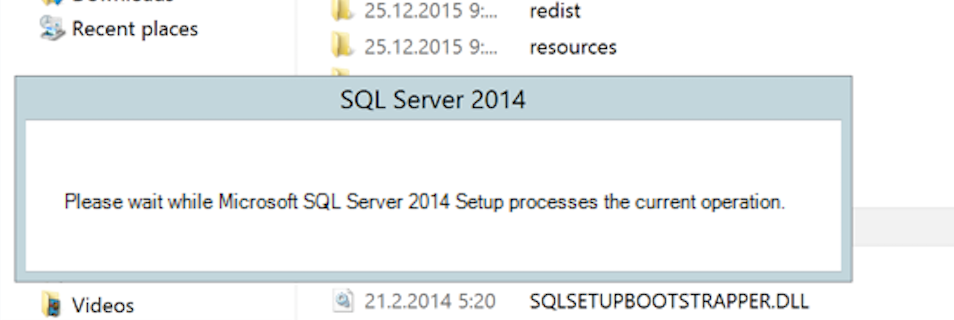
\*Poleg tega pa je potrebno namestiti tudi MSSQL Server 2012, pomagamo si lahko z namestitvenim čarovnikom; SQL Server 2014 Express with Tools (x64) ki ga lahko prenesemo z DreamSparka.

Prenovljena različica vsebuje lokalno podatkovno bazo, shranjeno v .mdf datoteki, tako da ni potrebe po namešanju Windows Server 2012. Dovolj je le, da projekt (v mapi PrijavaNaDogodkeSP\EventiqonWebApp) odpremo v Visual Studio in ga zaženemo. Gesla z namenom niso kriptirana, več o tem je napisano spodaj (v poročilu).

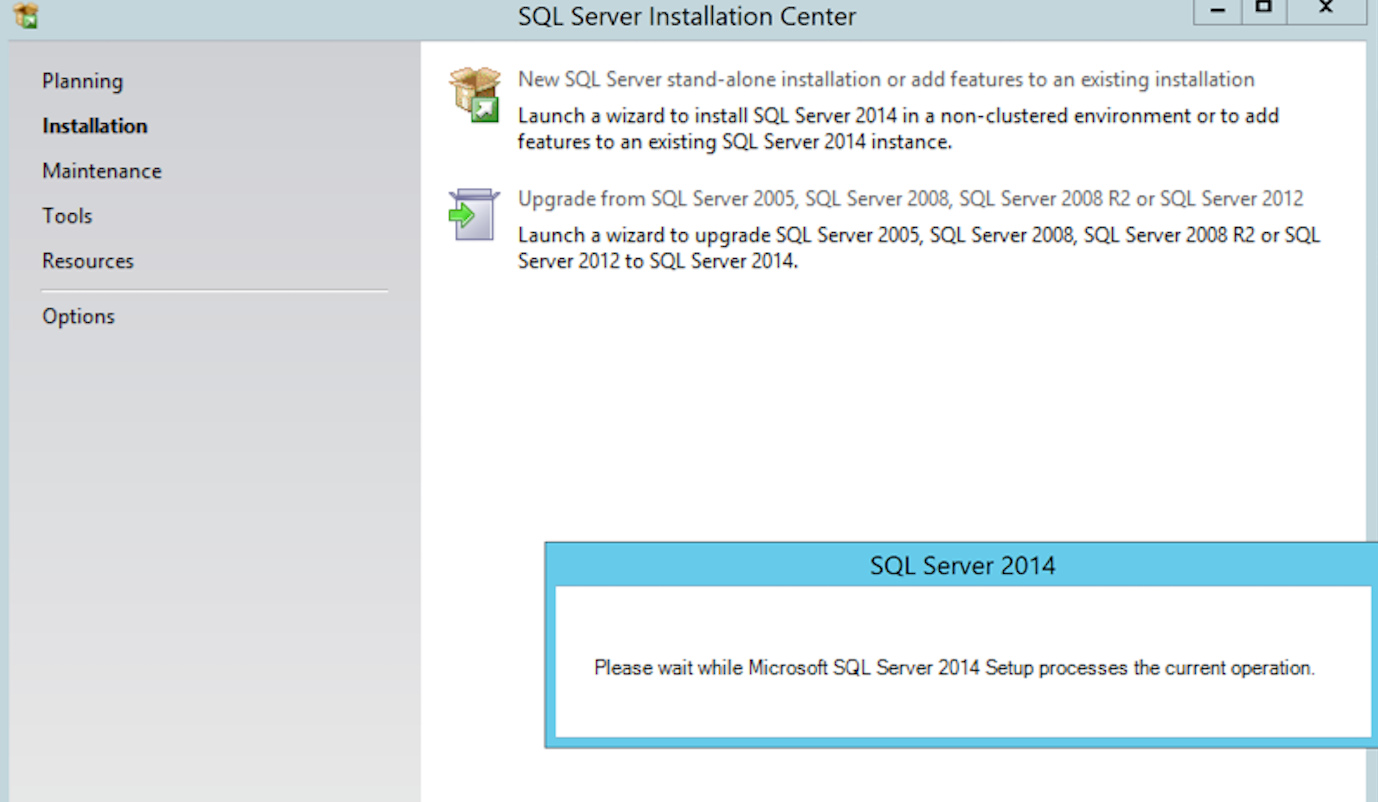
Postavitev strežnika in postavitev baze

Namestimo si operacijski sistem, ki ima spletni strežnik IIS (npr. Windows Server 2012).

Za podatkovno bazo je potrebno namestiti SUPB in ustvariti instanco baze:  
 Zaženemo SETUP.exe

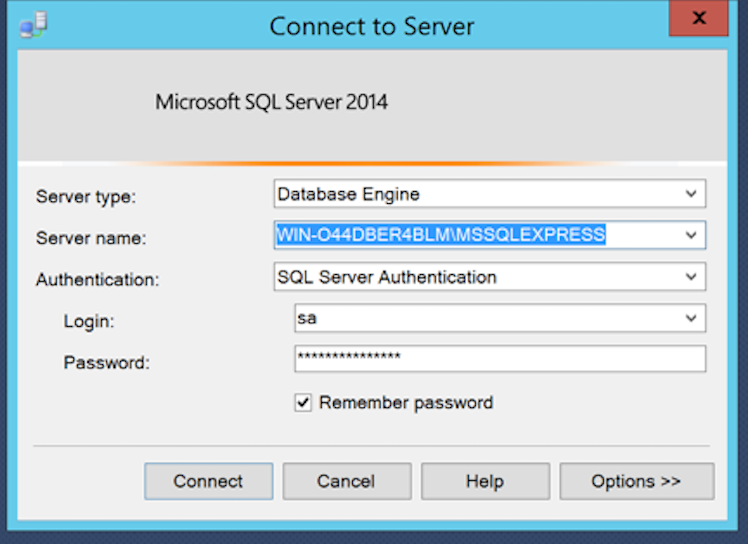


Kliknemo na   
“New SQL Server stand-alone installation or add features to an existing installation”

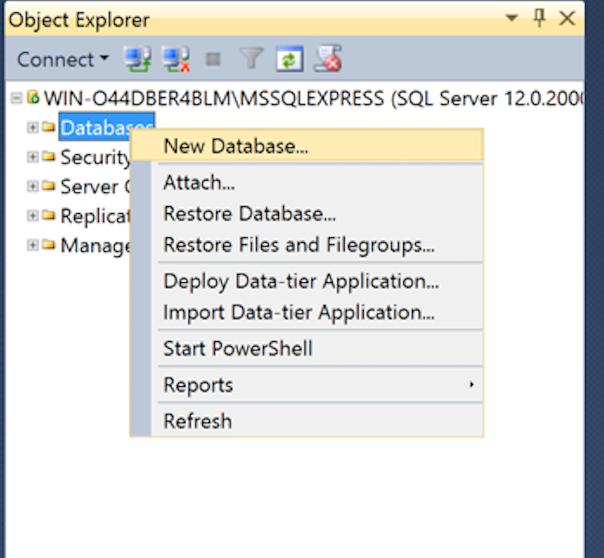


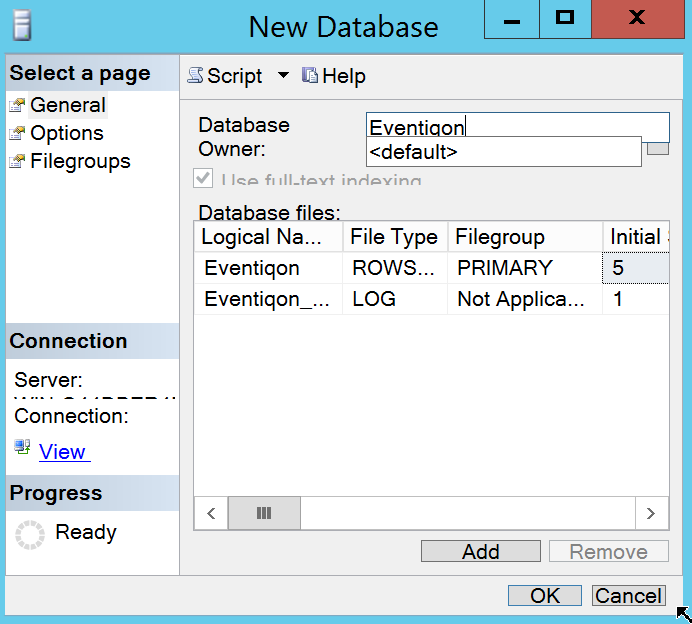
Sledimo navodilom za namestitev nove instance, določimo ji ime, način dostopa (dodajanje uporabnika, način avtentikacije, geslo).

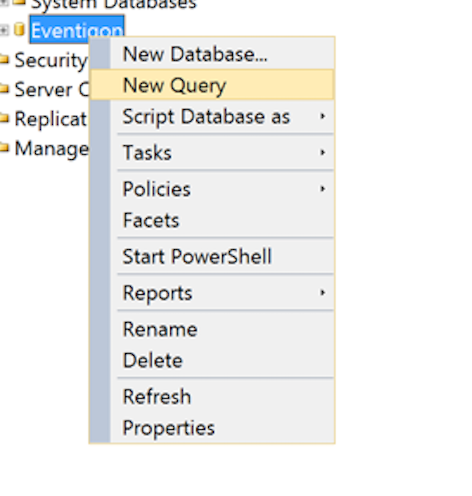
Namestimo tudi “SQL Server 2014 Management Studio (x64)”, se povežemo na ustvarjeno instanco podatkovne baze z ustreznim uporabniškim imenom in geslom.

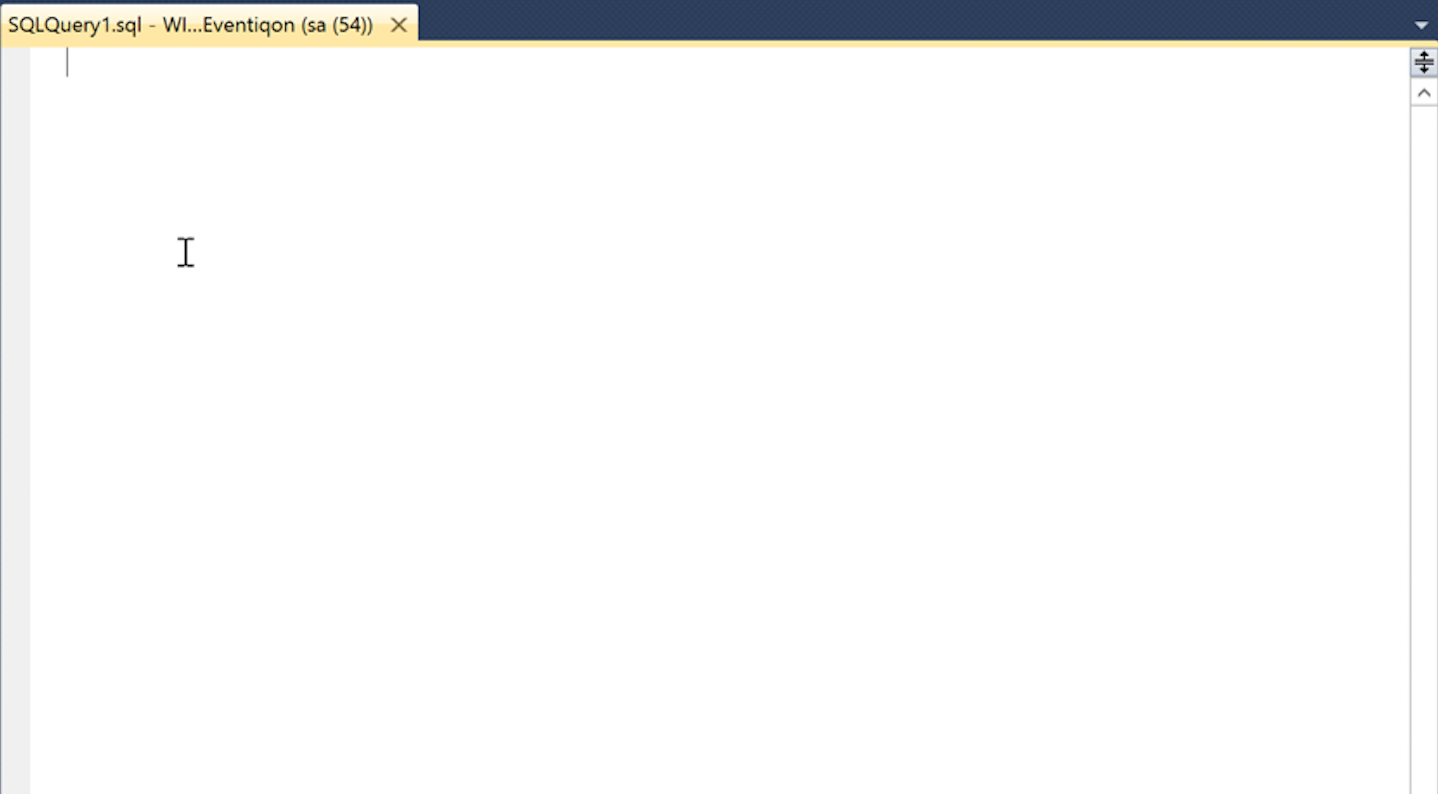


Ustvarimo novo podatkovno shemo z imenom “Eventiqon”.





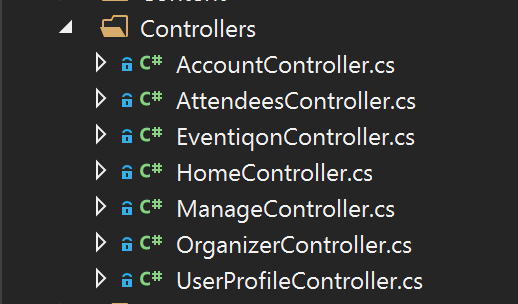


Odpremo novo poizvedbo

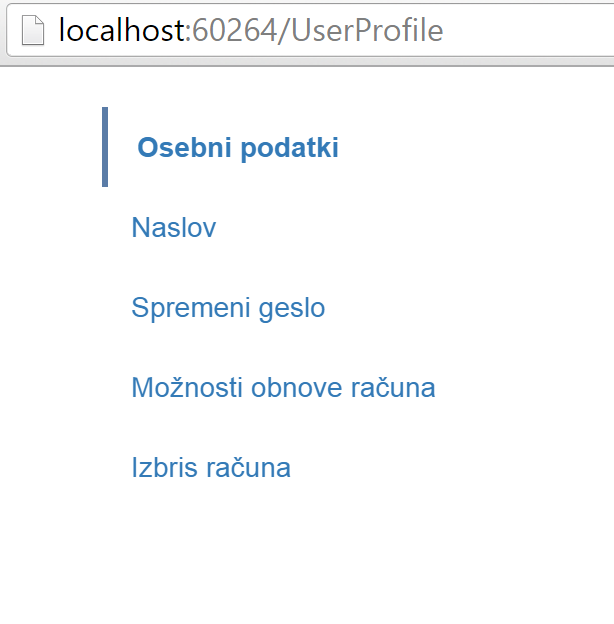
V polje potegnemo in spustimo datoteko “EventiqonSP\_MSSQL\_v1.4\_skripta.sql”, ki se nahaja v mapi “Konceptualni model” in poizvedbo izvedemo. Opcijsko lahko naložimo tudi testne podatke, z zagonom skripte “MSSQLv1.4-testdata.sql”.

Tako, sedaj ko imamo podatkovno bazo pripravljeno, lahko zaženemo strežnik kar preko IDE – odpremo Visual Studio, odpremo projekt – nahaja se v mapi “EventiqonWebApp” in ga zaženemo s pritiskom na tipko CTRL + F5. Odprl se nam bo privzeti spletni brskalni, v njem pa se bo naložila spletna stran.

Krmilni del aplikacije

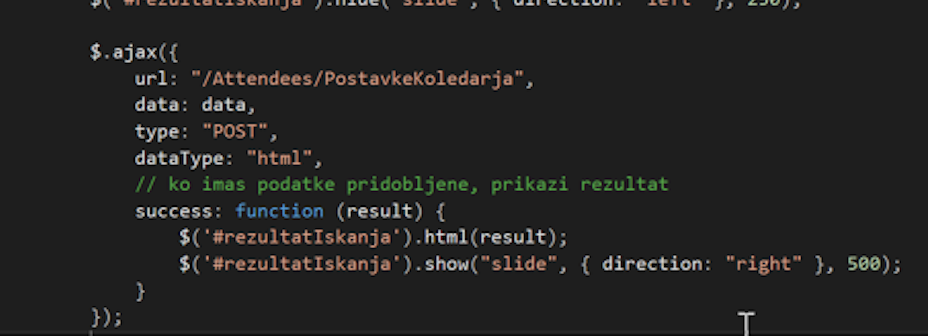


Aplikacijo sestavlja več, funkcijsko povezanih področji, sama implementacija HTTP metod pa se nahaja v krmilnih (angl. controller) razredih projekta. Pri uporabniški nadzorni plošči (UserProfileController) sem za vsako formo (osebni podatki, naslov, spremeni geslo, možnosti obnove računa, izbris računa) pripravil ustrezno [HttpPost] akcijo, ki kot parameter prejme model vhodnih podatkov posamezne forme (za preslikavo iz JSON objekta v objekt modela poskrbi ASP.NET MVC5 ogrodje), vrača pa odgovor v obliki objekta [JsonResult].



V metodah sem poskrbel za validacijo vhodnih podatkov ter v primeru uspešne validacije spremembe tudi zapisal v podatkovno bazo.

Poleg asinhrone komunikacije, kjer se prenaša odgovora tipa json, sem pri udeležbah (Moje udeležbe, AttendeesController.cs) namesto tega, prenašal kar generiran html dokument. Namesto da bi v metodi kontrolerja vračal celotno spletno stran (View) ali le podatke v goli Json obliki, sem vrnil le del strani (return PartialView(model)). Na strani odjemalca pa sem moral rezultat le zamenjati z obstoječim delom strani, pri čemer sem uporabil Javascript knjižnico, jQuery.



### Varnost

Spletna stran je zaščitena pred XSS napadi, saj za prenašanje (mapiranje) parametrov na strani strežnika uporabljam privzeti binder (del [ASP.NET](http://asp.net/) MVC 5 projekta). Da ne more priti do sql injection skrbi Microsoftov Entity Framework, saj vso manipulacijo s podatkovno bazo izvajam neposredno z uporabo Lamda izrazov nad kontekstom enitet (množica entitetnih razredov, uporabljen ORM). Edino kar bi bilo vredno omeniti je, da za vnosne forme ne uporabljam proti-ponarejevalnih žetonov (kar se lahko trivialno reši pri sestavljanju pogleda z Razor sintakso  - nad Action metodami kontrolerjev pa dodamo le ustrezno anotacijo. Prav tako bi lahko enostavno zakriptirali uporabniška gesla z uporabo Hash funkcije ob samem vnosu uporabnika v bazo. Preden bi stran dali v produkcijsko okolje, bi bilo smislno nastaviti tudi ustrezno varnost povezave (da uporabljamo protokol https,  SSL/TLS ).

### Poročilo o tipu predpomnjenja

Predpomnenje se po privzetih nastavitvah pri uporabi anotacije [OutputCache] izvaja na treh nivojih/lokacijah: na spletnem strežniku, proxy strežnikih in na strani odjemalca.   
<http://www.asp.net/mvc/overview/older-versions-1/controllers-and-routing/improving-performance-with-output-caching-cs>

Možno je narediti tudi svoj profil predpomnenja (Cache profile), potrebno mu je nastaviti le ustrezne parametre. Spodaj je primer takega profila, ki za 1 uro cachira željeno vsebino.  
  
<caching>

<outputCacheSettings>

<outputCacheProfiles>

<add name="CacheFor1Hour" duration="3600" varyByParam="none"/>

</outputCacheProfiles>

</outputCacheSettings>  
</caching>

Za predpomnenje statičnih datotek (slike, Javascript datoteke, slogovne kaskadne pole) pa je potrebno ustrezno skonfigurirati spletni strežnik (v mojem primeru IIS verzije 10). To je najenostavneje storiti neposredno s spreminjanjem/manipulacijo Web.config datoteke. Znotraj značke za konfiguriranje spletnega strežnika: “system.webServer” je potrebno dodati naslednje:

<staticContent>

<clientCache cacheControlMode="UseMaxAge"

cacheControlMaxAge="31.00:00:00" />  
</staticContent>

Poleg tega pa predpomnenje izvaja tudi sam Entity Framework (nekaj na to tematiko je opisano pod naslovom “Varnost”).

### Poročilo o testiranju

Regresijskih testov zaradi pomanjkanja časa nisem delal (v primeru odprave hroščev programske kode, se pogosto napiše test, ki  pretestira naslovljeno funkcionalnost - te teste se v kasnejših fazah razvoja prav tako pridoma uporablja). Naredil pa sem performančne teste s pomočjo programa JMeter, pri čemer sem simuliral sočasne dostope večih odjemalcev. Rezultate testiranj si lahko ogledate v mapi “PrijavaNaDogodkeSP\Jmeter testi”, prav tako pa se lahko tudi sami poigrate z napisanimi testi (.jmx datoteki).