**ELEKTROTEHNIČKA I PROMETNA ŠKOLA OSIJEK**

Ivan Lazić

Osijek, svibanj 2020.

ZAVRŠNI RAD

**ASP.NET WEB API i MVC web sjedište**

Učenik: Ivan Lazić

Razred: 4ET2

Obrazovni sektor: Elektrotehnika i računalstvo

Zanimanje: Tehničar za računalstvo

Mentor: Ivan Marušić

ZAVRŠNI RAD

**ASP.NET WEB API i MVC web sjedište**

**ELEKTROTEHNIČKA I PROMETNA ŠKOLA OSIJEK**

Opis zadatka:

Učenik treba kroz završni rad prikazati postupak izrade web sjedišta pomoću

navedenih tehnologija. Pri izradi treba dokumentirati korištene alate. Koristiti obrasce

obrađene iz predmeta Napredno i objektno programiranje: enkapsulacija,

nasljeđivanje, kontrola pristupa, obrasci. Rad napisati prema uputama mentora uz

pridržavanje uputa za pisanje završnog rada objavljenih na mrežnim stranicama škole

(format i zadani elementi - opis zadatka, uvod, sadržaj, razrada teme, zaključak,

literatura, prilozi i dr).

**SADRŽAJ**

1. UVOD *1*
2. BAZA PODATAKA *6 – 7*
   1. Projektiranje *6*
   2. Microsoft SQL Server Menagment Studio *7*
3. ASP.NET WEB API *8 – 16*
   1. Početno postavljanje *8 – 9*
   2. Povezivanje s bazom *9 – 11*
   3. Migracije *11*
   4. Slojevi (eng. Layers) *12 – 15*
      1. DAL sloj *12 – 13*
      2. Service sloj *14*
      3. Web sloj *15*
   5. Autentikacija *15 – 16*
   6. Validacija *16*
4. ASP.NET MVC *17 – 25*
   1. Početničko postavljanje *17*
   2. Model *18*
   3. View *18 – 19*
   4. View predlošci *19 – 22*
      1. Pojedinačan prikaz *19*
      2. Prikaz liste *20*
      3. Kreiranje *20 – 21*
      4. Ažuriranje *21*
      5. Brisanje *22*
   5. Kontroler *22 – 23*
   6. HTTP klijent *23 – 24*
   7. Kolačići *24 - 25*
5. ZAKLJUČAK *26*

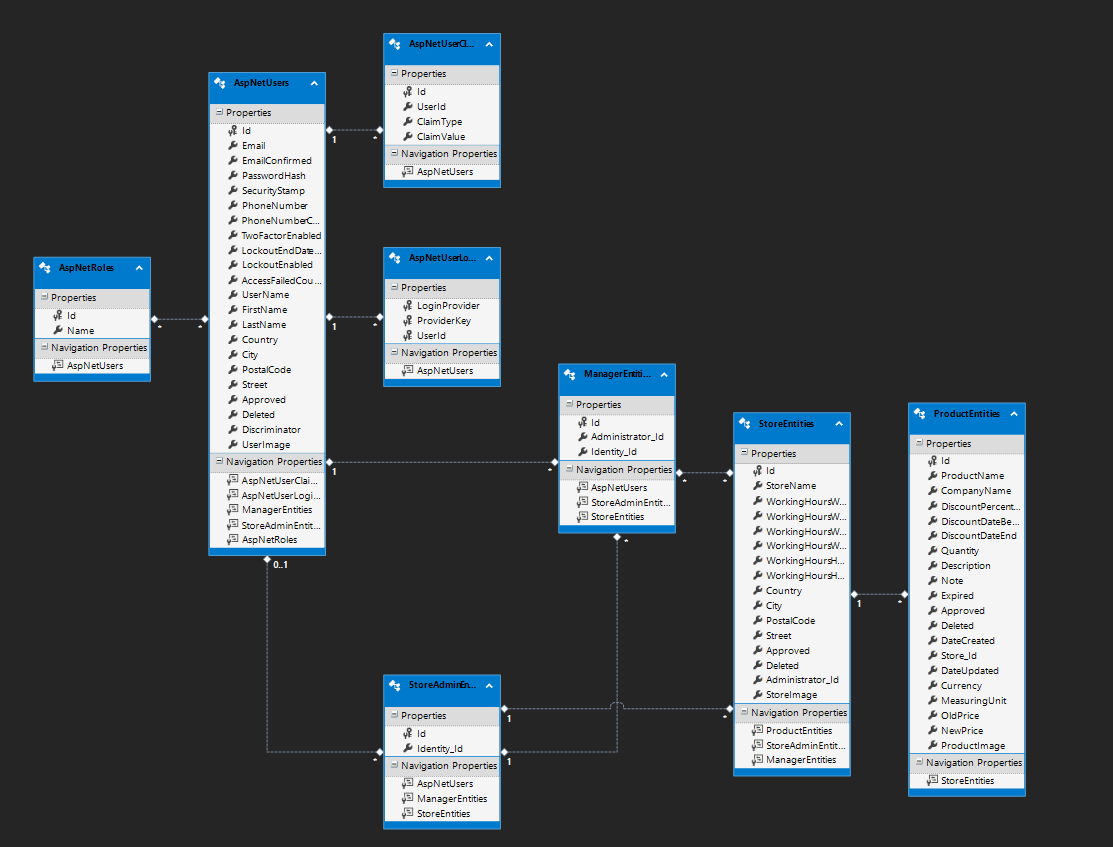
LITERATURA 27

1. **UVOD**

ASP.NET WEB API i MVC web sjedište je web sjedište ili web aplikacija izrađena u razvojnom okruženju Visual Studio 2017 i paketu za razvoj softvera .NET Framework 4.6.1 tvrtke Microsoft. .NET Framework uključuje razne objektno orijentirane programske jezike među kojima je i C# koji se koristi za ovaj projekt. ASP.NET je proširenje .NET Framework-a koje je specijalizirano za izradu web aplikacija. Visual Studio je integrirano razvojno sučelje (eng. Integrated development environment, IDE) koji omogućuje uređivanje i prevođenje programskog koda te otklanjanje grešaka. Visual Studio također podržava i instaliranje vanjskih (NuGet) paketa. NuGet paketi omogućuju unošenje nove funkcionalnosti u programski kod. Neki od NuGet paketa koji se koriste u ovom projektu su paketi za validaciju, komunikaciju, rad s podacima i sl. Projekt se sastoji od tri glavna dijela: baza podataka, serverski dio (eng. Server-side, backend) i sučelje (eng. frontend) koji će biti detaljnije opisani glavnom dijelu zajedno sa izrescima zaslona koda i korisničkog sučelja. Ideja za ovaj projekt je aplikacija koja bi omogućila trgovinama objavu sniženih proizvoda. Kad se administrator trgovine registrira, može dodati fizičke trgovine sa svojom adresom, radnim vremenom i ostalim informacijama. Administrator također može dodati upravitelja koji može biti zadužen za jednu ili više trgovina. Administrator bi mogao dodijeliti trgovinu upravitelju i obratno (dodijeliti upravitelja trgovini). Trgovinom mogu upravljati samo njeni upravitelji i njen administrator. Pod upravljanje se podrazumijeva dodavanje, ažuriranje i brisanje proizvoda. Proizvodi se od strane korisnika mogu sortirati po njihovim datumima sniženja, cijeni i imenu. Ako sniženje istekne, proizvod se premješta u spremnik za neaktivne proizvode. Upravitelj ili administrator mogu osvježiti taj proizvod ponovnim dodavanjem datuma sniženja. Ovakva aplikacija mogla bi pomoći ljudima u potrazi za najboljom ponudom za neki proizvod. Proizvodi velikog broja trgovina nalazili bi se na jednom mjestu za razliku od fizičkih kataloga.

1. **BAZA PODATAKA**
   1. **Projektiranje**

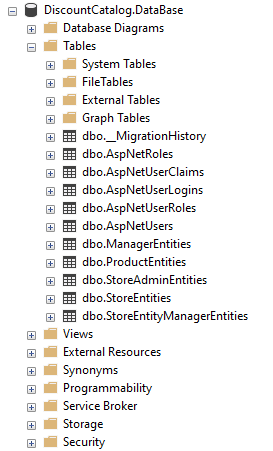
Prvi je korak pri izradi aplikacije projektiranje baze podataka. Pri projektiranju potrebno je odrediti entitete, veze i atribute. Entiteti podrazumijevaju stvari, bića ili pojave o kojima želimo spremati podatke, veze su odnosi među entitetima, a atributi su svojstva entiteta. Paket koji se koristi za izradu i komunikaciju s bazom u ovom projektu je Entity Framework. Prilikom izrade baze podataka u paketu Entity Framework potrebno je odabrati pristup izrade. Pristup koji koristi ova aplikacija je „prvo kod“ (eng. Code first) pristup. Što znači da se modeli (entiteti) kreiraju u samom kodu i pomoću migracija se automatski generiraju relacije (tablice).



Slika 1: ER model baze podataka[[1]](#footnote-1)

* 1. **Microsoft SQL Server Menagment Studio**

SQL Server Management Studio (SSMS) integrirano je okruženje za upravljanje bilo kojom SQL infrastrukturom. Služi za upravljanje i organiziranje baza podatka. U ovom projektu MS SQL Server Menagment Studio služi za pregled i mijenjanje podataka po potrebi.



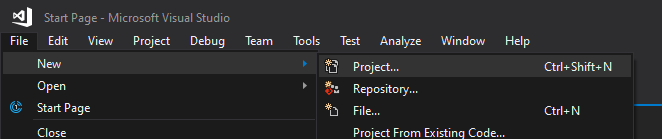
*Slika 2:* Baza aplikacije

1. **ASP.NET WEB API**

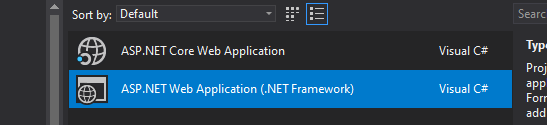
Web API je sučelje za programiranje aplikacija (eng. Application programming interface, API) koji služi za izgradnju usluga temeljenih na HTTP-u kojima se može pristupiti na različitim platformama kao što su web, Windows i mobilne platforme.

* 1. **Početno postavljanje**

Prvo je potrebno kreirati novi ASP.NET WEB API projekt u Visual Studiju. Otvori se prozor u kojem je potrebno odabrati tip projekta. Potrebno je odabrati ASP.NET Web Application (.NET Framework).

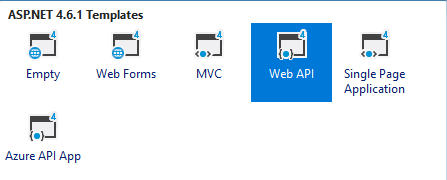


*Slika 3:* Kreiranje novog projekta



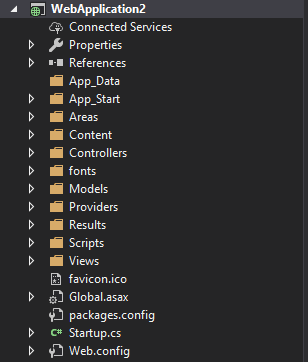
*Slika 4:* Odabir tipa projekta

Nakon toga otvara se prozor s početnim predlošcima (eng. Template). Potrebno je odabrati Web API[[2]](#footnote-2) i promijeniti autentifikaciju u *Individual User Accounts* što će stvoriti potrebne predloške u projekt i omogućiti autentifikaciju korisnika s individualnim korisničkim računima.



*Slika 5:* Odabir predloška

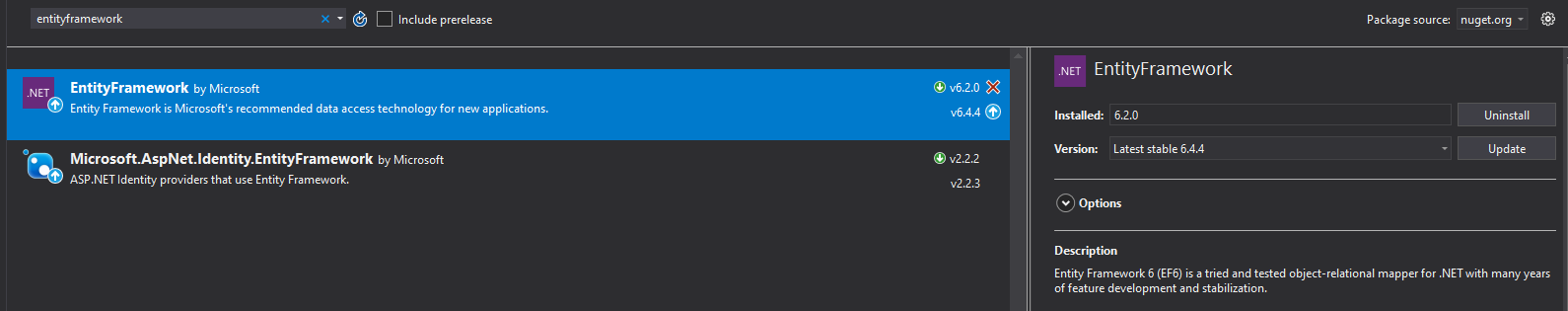
Nakon odabira potrebno je pričekati instaliranje svih paketa i kreiranje svih predložaka. Kada taj proces završi, dobije se Web API aplikacija s MVC predloškom i mogućnošću dodavanja korisničkih računa.



*Slika 6:* Web API projekt

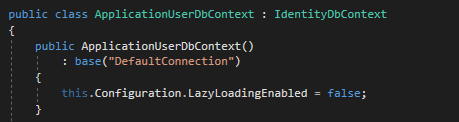
* 1. **Povezivanje s bazom**

Prvi je korak pri povezivanju s bazom dodavanje paketa Entity Framework u projekt pomoću NuGet upravitelja za pakete. Entity Framework omogućuje komunikaciju s bazom pomoću LINQ naredbi u samom kodu. LINQ omogućuje pisanje upita na bazu direktno u C# jeziku.

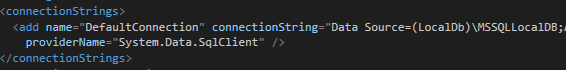


*Slika 7:* Dodavanje paketa Entity Framework

Nakon dodavanja paketa potrebno je stvoriti kontekst (eng. Context) na bazu pomoću *DbContext* klase u samom Entity Framework-u ili proširenje te klase *IdentityDbContext* za koje je potrebno u *Web.config* dodati *Connection string* koji omogućuje spajanje na bazu.

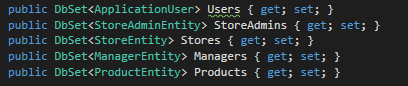


*Slika 8:* Kontekst na bazu

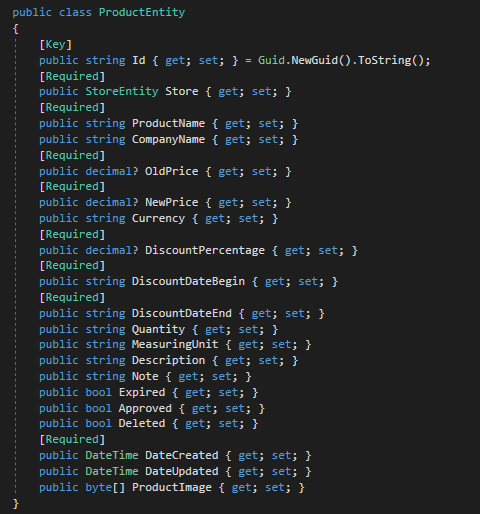


*Slika 9:* Connection string

Nakon toga potrebno je kreirati entitete i ubaciti ih u *DbSet* u *ApplicationUserDbContext* kao svojstva koja predstavljaju tablice u bazi podataka*.* Na slici 11 nalazi se primjer jednog entiteta proizvoda sa svojim svojstvima koji će postati stupci u tablici.



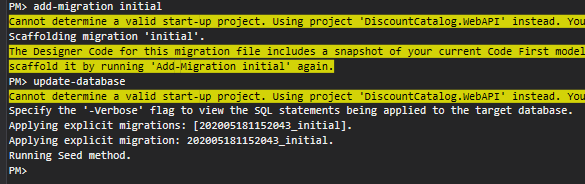
*Slika 10:* DbSet svojstva



*Slika 11:* entitet proizvod

* 1. **Migracije**

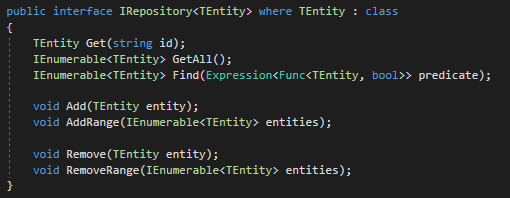
Prilikom korištenja „prvo kod“ pristupa potrebno je pri svakom ažuriranju entiteta pokrenuti migraciju kako bi se ažurirala i baza. Kako bi se uključile migracije potrebno je u upravitelj paketa (eng. Package manager) upisati naredbu *enable-migrations.* Za kreiranje migracije upiše se naredba *add-miration imeMigracije* i *update-database* na kraju kako bi se ažurirala baza.



*Slika 12:* Kreiranje migracije

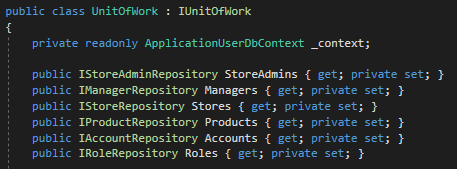
* 1. **Slojevi (eng. Layers)**
     1. **DAL sloj**

DAL ili *Data Access Layer* je sloj u kojem se pomoću Entity Framework-a direktno pristupa podacima. U DAL sloju se nalaze *Repository* i *UnitOfWork* obrazac (eng. Pattern). Repository u sebi sadrži metode za pristup podacima kao što su „dohvati“, „dohvati sve“, „kreiraj“, „ažuriraj“, „obriši“… Na slici 13 nalazi se primjer jednog općeg sučelja *IRepository.* Svrha sučelja *IRepository* je osiguravanje osnovnih metoda ostalim repozitorijima bez ponavljanja koda. Klasa *Repository* implementira sučelje *IRepository* i njegove metode. Svaki sljedeći repozitorij nasljeđuje klasu *Repository* koja sadrži osnovne metode.

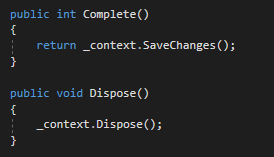


*Slika 13:* Opći repozitorij *IRepository*

UnitOfWork obrazac služi ako se šalje velik upit koji se sastoji od puno malih upita na bazu i jedan ili više njih dojavi grešku, taj upit se neće izvršiti. Entity Framework ima jednu vrstu *UnitOfWork* obrasca ugrađenu, no često nije pouzdan za velike upite. UnitOfWork sastoji se od svih repozitorija, konteksta na bazu i metode *Complete* koja poziva *SaveChanges* metodu u kontekstu koja sprema sve promjene u bazu i metode *Dispose* koja čisti sve nepotrebne instance konteksta i *UnitOfWork*-a.

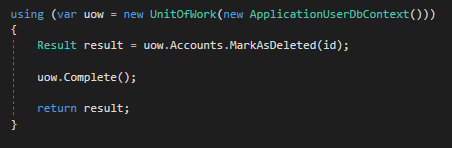


*Slika 14:* UnitOfWork kontekst i svi repozitoriji



*Slika 15:* UnitOfWork Complete i Dispose metode

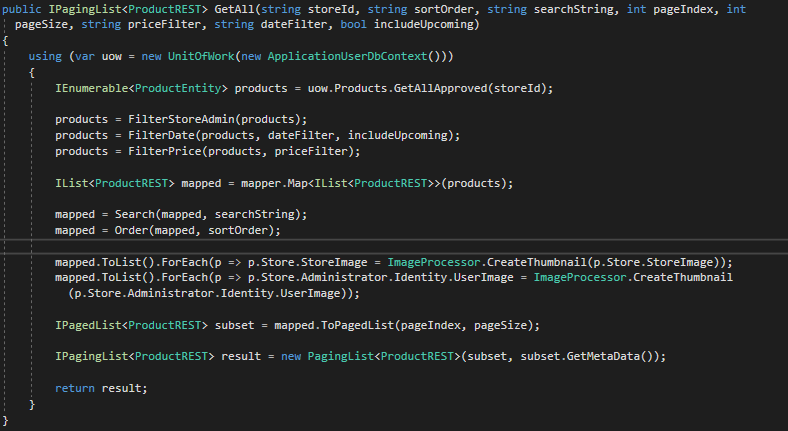
*UnitOfWork* pozivamo pomoću *using* bloka. Kada se u *using* bloku kreira *UnitOfWork* objekt potrebno je istovremeno kreirati i novi kontekst. Primjer korištenja *using* bloka s *UnitOfWork*-om nalazi se na slici 16.



*Slika 16:* Primjer pozivanja UnitOfWork-a

* + 1. **Service sloj**

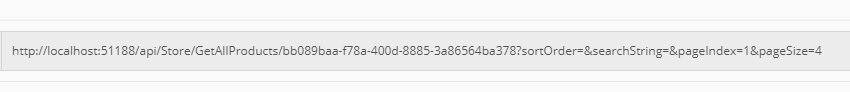
Service sloj je dodatni sloj u ASP.NET Web API aplikaciji koji upravlja komunikacijom između DAL i Web sloja. Service sloj naviše se bavi validacijom i dodatnim procesiranjem podataka. U ovom projektu service sloj dobiva podatke iz DAL sloja, ako se radi o listi podataka, sortira je, poreda je po nekom određenom redu (abacedno, po cijeni i sl.) i ako je poslan upit za pretragom, pretražuje podatke iz liste. Service sloj također služi za filtriranje nepotrebnih podataka i procesiranje podataka kao što su korisničke slike. Primjer jedne metode *GetAll* u klasi *ProductService* nalazi se na slici 17*.* U toj metodi vidljivo je dohvaćanje podataka iz DAL sloja, mapiranje pomoću paketa *AutoMapper*, filtriranje, sortiranje, pretraživanje, procesiranje slika i pretvaranje u listu koja omogućuje podjelu velikog broja podataka na stranice (eng. Paiging).



*Slika 17:* primjer GetAll metode

* + 1. **Web sloj**

Web sloj je krajnji sloj API. Sastoji se od *ApiController*-a koji kontroliraju krajnje dočke (eng. Endpoints) koje poziva korisničko sučelje kako bi primio ili poslao podatke. Krajnje točke pozivaju se preko adrese najčešćeg formata: *http://adresaServera/api/Kontroler/Metoda?Parametri*

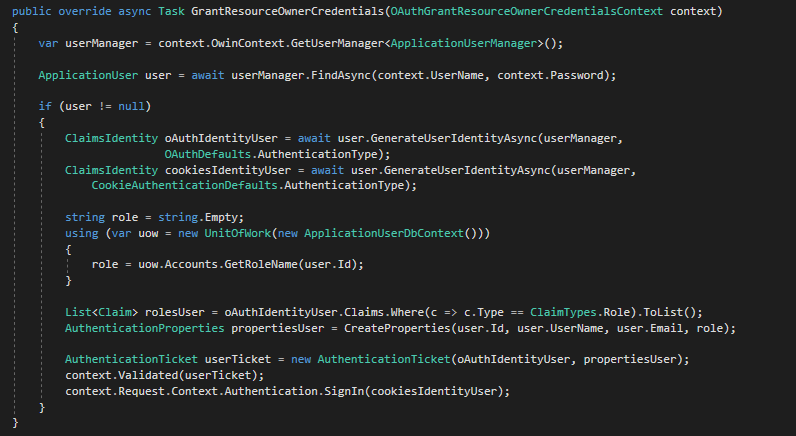


*Slika 18:* Primjer jednog zahtjeva na API

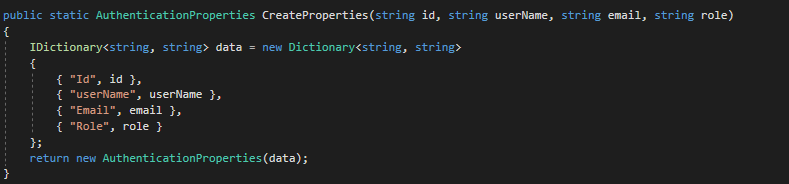
* 1. **Autentikacija**

Postoji nekoliko načina autentikacije. Ovaj projekt koristi OAuth 2, JWT i bearer token autentikaciju. OAuth 2 je programski okvir za autorizaciju koji omogućuje aplikacijama da dobiju ograničeni pristup korisničkim računima na HTTP usluzi. JWT (JSON Web Token) je otvoreni standard sigurnog prijenosa informacija putem JSON objekta, a bearer token je znakonvni niz (eng. String) koji je generiran od strane servera i poslan klijentu kako bi ga mogao koristiti prilikom svakog zahtjeva i dobiti pristup podacima za koje je potrebna autentikacija.

Prilikom zahtjeva za prijavu (eng. Login) u aplikaciju pokreće se *GrantResourceOwnerCredentials* metoda koja pronađe korisnika u bazi, provjeri njegove informacije (korisničko ime i lozinka) i pošalje korisničko ime, email, id i ulogu (eng. Role) koje korisničko sučelje sprema u kolačiće (eng. Cookies).



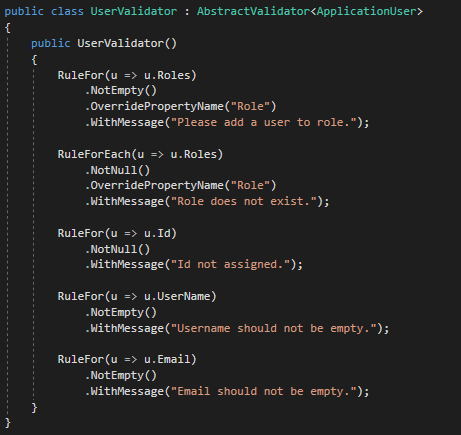
*Slika 19:* Provjera korisničkih podataka



*Slika 20*: kreiranje korisničkih podataka za slanje

* 1. **Validacija**

Validacija u ovom projektu rješena je pomoću paketa *FluentValidation. FluentValidation* funkcionira na način da klasa validator nasljedi klasu *AbstractValidator* kojoj se mora odrediti tip. Tip će biti entitet koji se validira. Sva pravila po kojima se validira entitet dodaju se konstruktor klase validator pomoću metode *RuleFor.* Neka pravila koja se mogu koristiti su metode *NotNull, NotEmpty, GreaterThan, LessThan…*



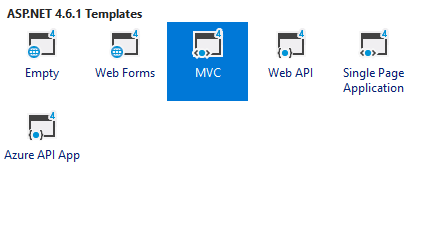
*Slika 21:* Validator korisnika

1. **ASP.NET MVC**

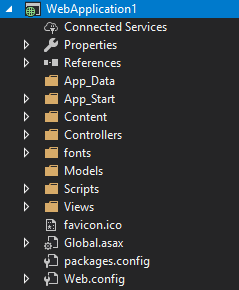
ASP.NET MVC (Model, View, Controller) je obrazac koji se koristi za odvajanje korisničkog sučelja, podataka i logike aplikacije. Korištenjem MVC uzorka za web stranice, zahtjevi se preusmjeravaju na kontroler koji je odgovoran za rad s modelom i dohvaćanje podataka. Kada kontroler dobije podatke poziva Razor view i prikazuje podatke.

* 1. **Početno postavljanje**

Kao i kod ASP.NET WEB API projekta potrebno je kreirati novi projekt i odabrati ASP.NET Web Application (.NET Framework) i odabrati MVC predložak bez autentikacije. Stvorit će se novi MVC projekt prikazan na slici 23. Projekt se sastoji od mapa App\_Start u kojoj se nalazi sve što se mora pokrenuti na samom pokretanju aplikacije, mapa Content u kojoj se nalaze .css datoteke za bootstrap temu i pojedinačne stranice. U mapi Models kreiramo svoje modele, u mapi Scripts nalaze se JavaScript datoteke za bootstrap, jquery i sl. I u mapi Views se nalaze Razor View datoteke.



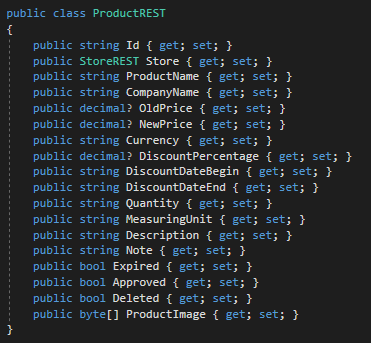
*Slika 22:* odabir MVC predloška



*Slika 23:* MVC projekt

* 1. **Model**

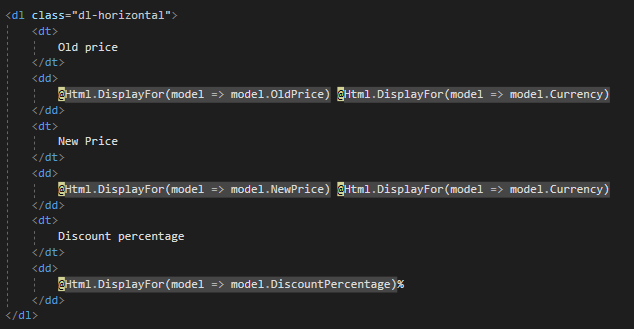
Model predstavlja podatke i poslovnu logiku aplikacije i omogućuje View-u prikaz podataka. Primjer jednog modela vidljiv je na slici 24.



*Slika 24:* Model proizvod

* 1. **View**

View je .cshtml datoteka koja omogućuje prikaz podataka i interakciju s korisnikom. Omogućuje C# kod u html stranici s kojim se mogu koristiti *Html* „pomagači“ (eng. Helpers). Html pomagači omogućuju dinamičan prikaz podataka i kreiranje formi. Jedan primjer dinamičkog prikaza je *Html.DisplayFor* metoda koja može prikazati jedno svojstvo modela u html stranici. *Html.EditorFor* omogućuje prikladno kreiranje ulaznih (eng. input) elemenata u stranici, npr. ako je svojstvo u modelu tekstualno, ulaz će isto biti tekstualan. Primjer prikaza cijena jednog proizvoda nalazi se na slici 25.



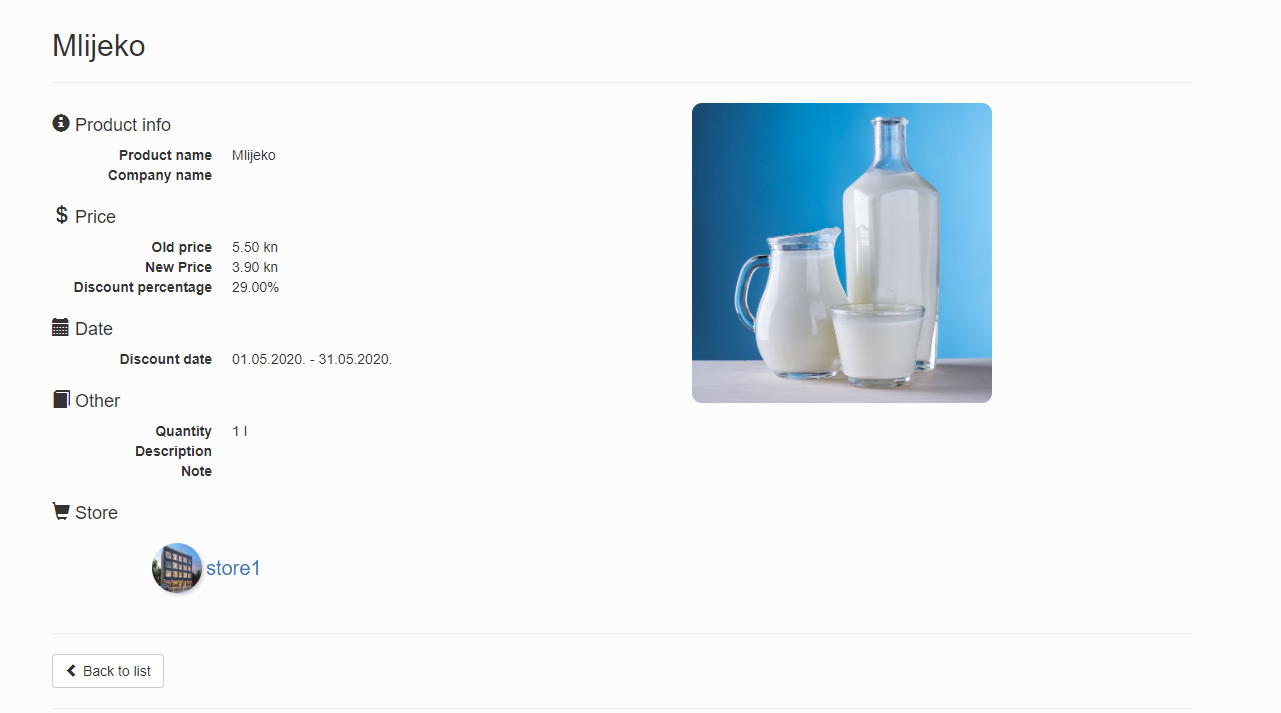
*Slika 25:* Prikaz cijena jednog proizvoda

* 1. **View predlošci**

View se kreira desnim klikom na kontroler i klikom na gumb *Add View.* Otvara se prozor u kojem je moguć odabir predloška i vrste View-a. MVC nudi predloške za pojedinačan prikaz, prikaz liste, kreiranje, ažuriranje, brisanje i prazan predložak.

* + 1. **Pojedinačan prikaz**

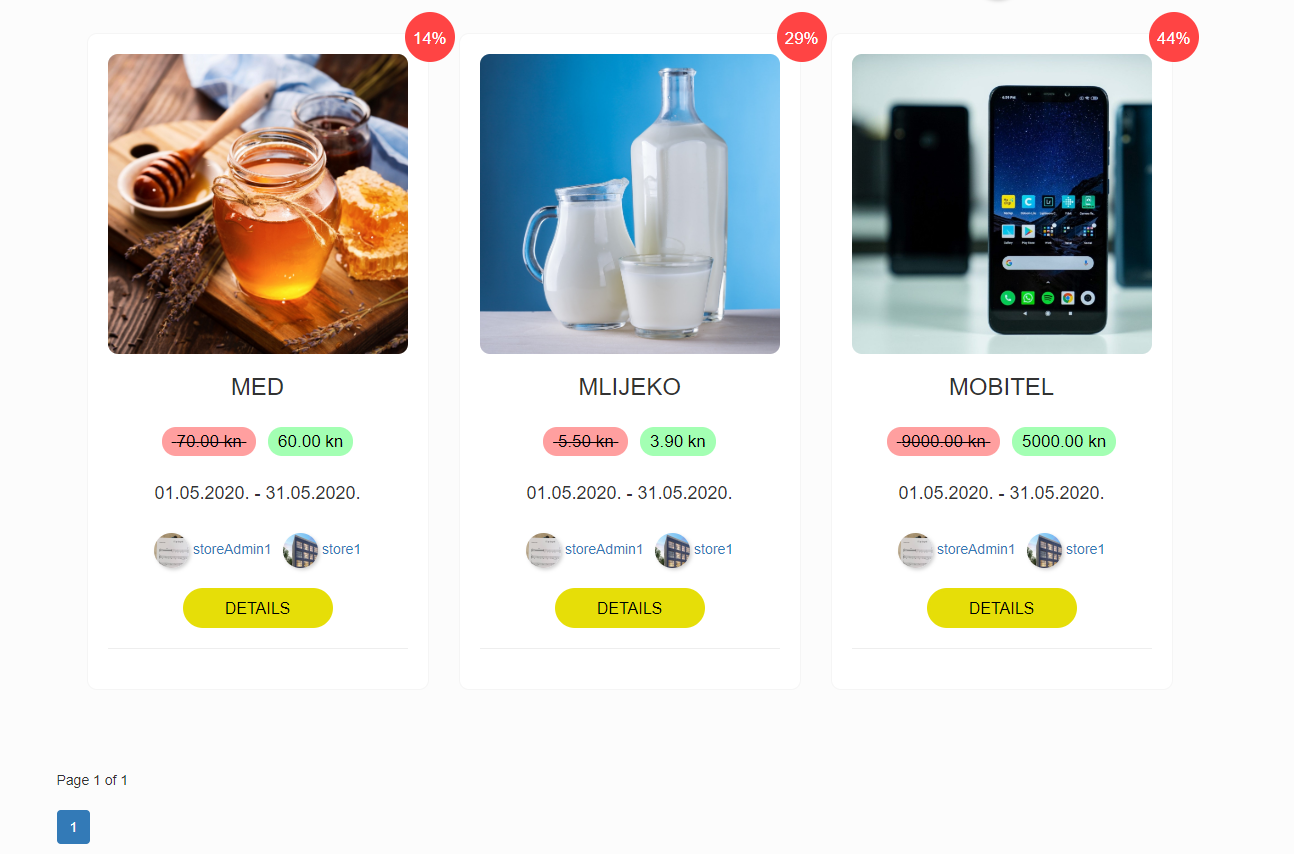
Predložak za pojedinačan prikaz stvara Razor View s prikazom svih svojstava modela. Uz zadane elemente dodani su i gumbi za navigaciju.



*Slika 26:* Pojedinačan prikaz proizvoda

* + 1. **Prikaz liste**

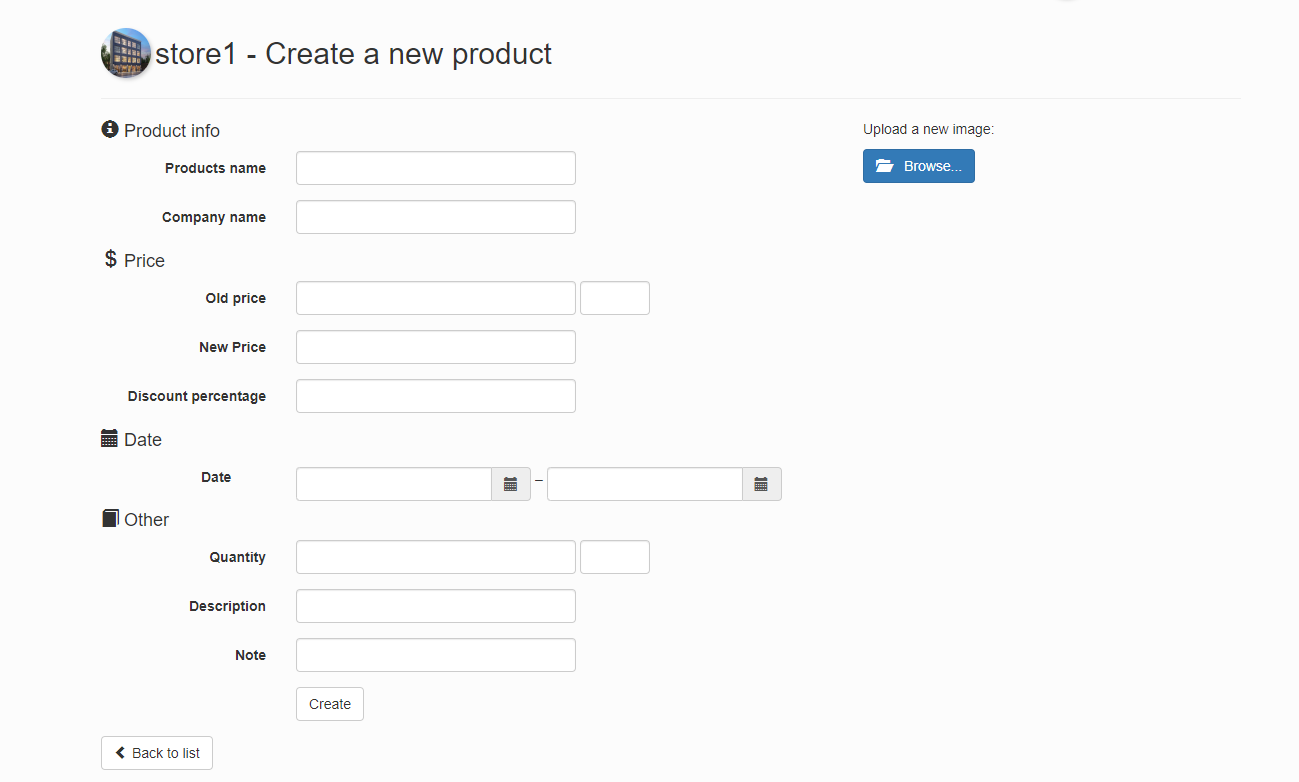
Predložak za prikaz liste stvara tablicu sa svojstvima modela. U prikazu liste proizvoda na slici 27 koriste se kartice umjesto tablice.



*Slika 27:* Prikaz liste proizvoda

* + 1. **Kreiranje**

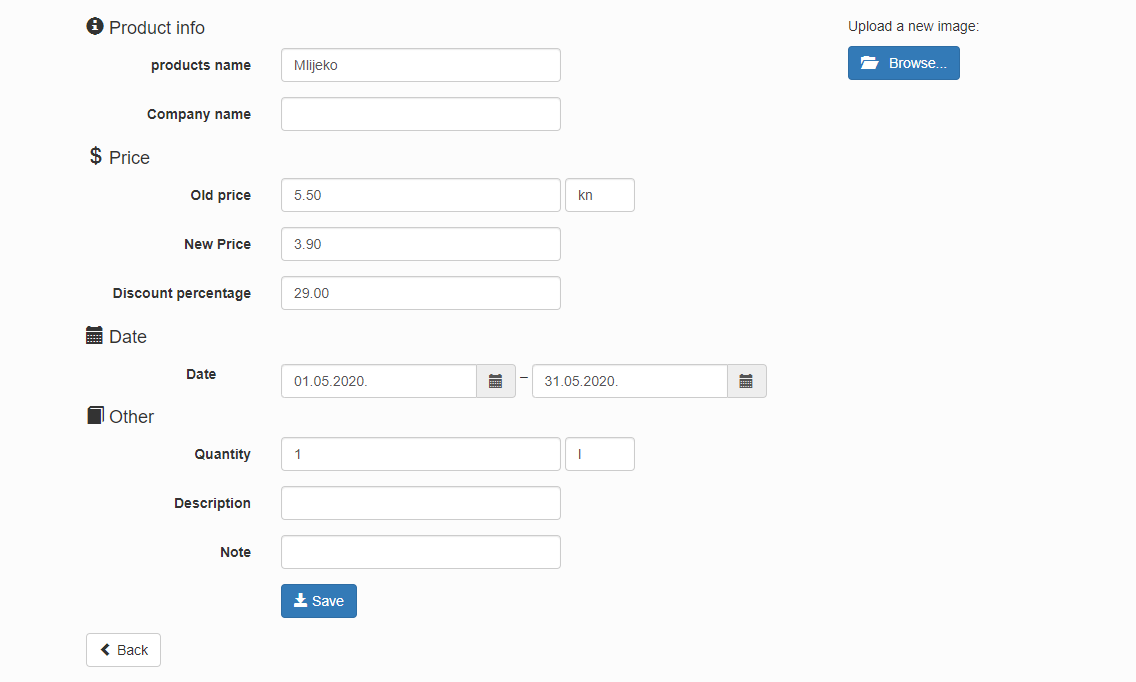
Predložak za kreiranje stvara ulazne elemente unutar forme. Nakon pritiska gumba za slanje podataka, podatke preuzima kontroler.



*Slika 28:* Forma za kreiranje proizvoda

* + 1. **Ažuriranje**

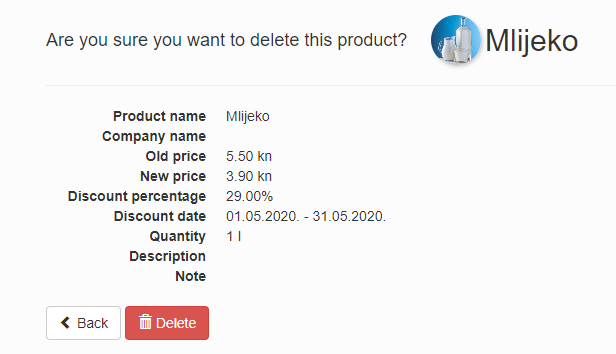
Predložak za ažuriranje stvara istu formu kao i predložak za kreiranje i popuni ulazne elemente s modelom koji se ažurira.



*Slika 29:* Forma za ažuriranje proizvoda

* + 1. **Brisanje**

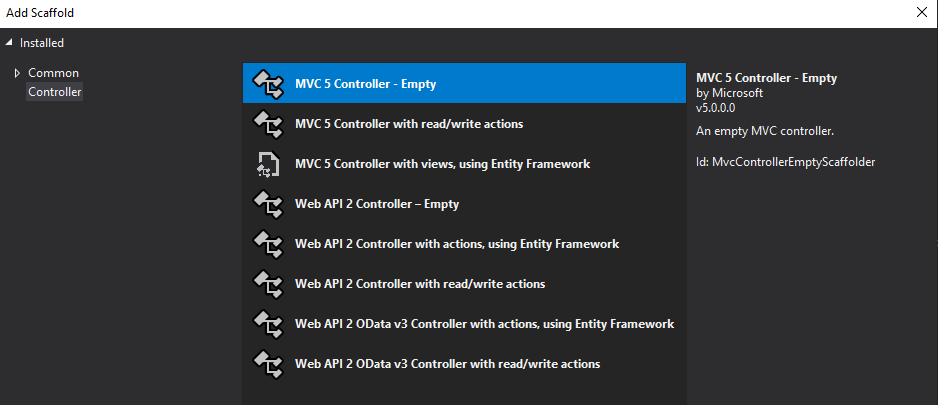
Nakon pritiska gumba *Delete* otvara se *View* koji rezimira sve podatke modela koji se briše i traži potvrdu o brisanju. Brisanje nekog entiteta ne briše taj entitet iz baze, nego ga obilježi kao obrisanog mijenjanjem svojstva *Deleted* u *true.* Entitet je moguće vratiti po potrebi u View-u za prikaz obrisanih entiteta do kojeg se dolazi pritiskom na gumb „otpad“.



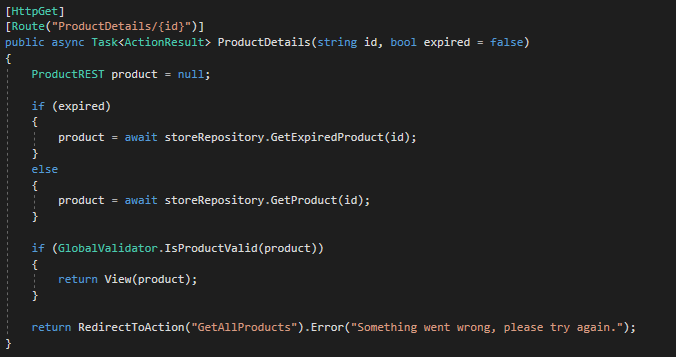
*Slika 30:* Potvrda prije brisanja

* 1. **Kontroler**

Kontroler je klasa koja nasljeđuje *Mvc.Controller* klasu koja u sebi sadrži *Action* metode. *Action* metode mogu pozvati *View* ili preusmjeriti zahtjev na neki drugi kontroler. Svaki kontroler mora imati *Controller* na kraju imena i mora se nalaziti u mapi *Controller*. Kontroler se kreira desnim klikom na *Controller* mapu i klikom na gumb *Add Controller*. Otvorit će se prozor s osnovnim predlošcima, ali uglavnom se odabire *MVC 5 Controller – Empty*. U ovom projektu kontroler preuzima podatke iz *Repository* klase koja nasljeđuje *MVCRepository* klasu koja sadrži HTTP klijent. Na slici 32 nalazi se primjer jedne akcije (eng. Action) *ProductDetails* koja prima informacije iz repozitorija, provjerava ispravnost proizvoda pomoću *GlobalValidator* klase i vraća *View* ili preusmjerava na drugu akciju ovisno o ispravnosti modela proizvoda.



*Slika 31:* Odabir predloška kontrolera



*Slika 32:* Primjer akcije u kontroleru

* 1. **HTTP klijent**

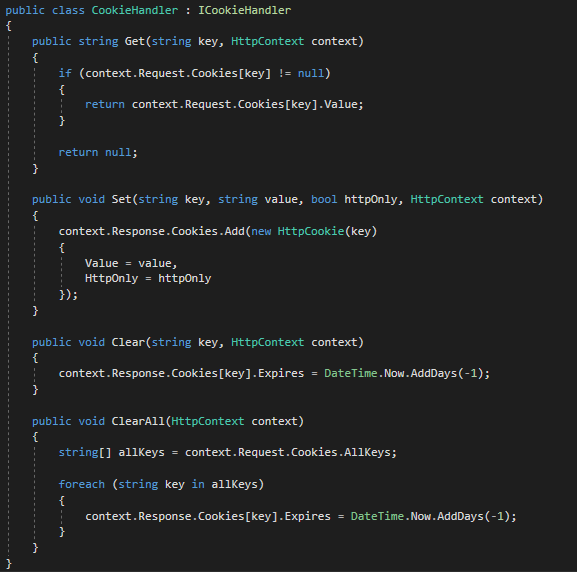
HTTP klijent je klasa koja omogućuje slanje podataka na server i primanje podataka sa servera pomoću URL-a.



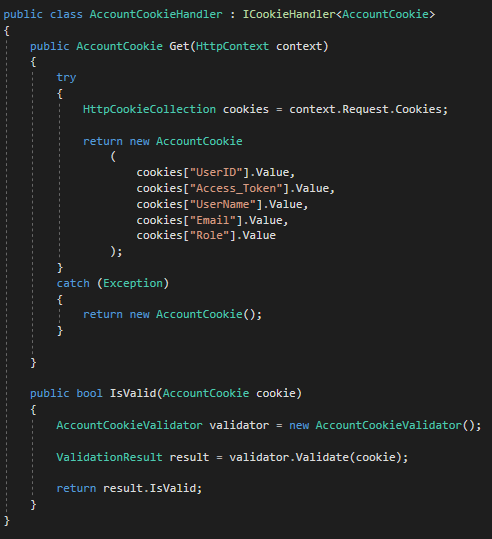
*Slika 33:* MVCRepository s Http klijentom

* 1. **Kolačići**

Kolačići (eng. Cookies) su fizička tekstualna datoteka koju sprema klijentov preglednik. U kolačiće se mogu spremati razni korisnički podaci koje preglednik trenutno treba, npr. korisničko ime, mail, uloga, ID i sl. U MVC-u kolačići se nalaze unutar *HttpContext* klase u obliku rječnika (eng. Dictionary). U ovom projektu kolačići se obrađuju pomoću *CookieHandler* klase koja sadrži metode za stvaranje, čitanje i brisanje kolačića. Uz *CookieHander* nalaze se i klase za pojedinačne modele kao što su *AccountCookieHandler* i *StoreCookieHandler* koji sadrže metode za čitanje i validaciju kolačića posebnih za pojedini model.



*Slika 34:* Klasa za rad s kolačićima



*Slika 35:* Klasa za rad s pojedinačnim kolačićima

1. **ZAKLJUČAK**

Ovaj završni rad prikazuje proces i sve potrebne alate za izradu API servisa i web aplikacije. Najveća prednost takvih aplikacija je njihova modularnost i slojevitost. Na isti API mogu se spojiti i mobilne i desktop aplikacije što ih čini dostupnima svima. Isto tako slojevitost pomaže u čistoći koda, lakšeg pronalaženja grešaka i boljih performansi. Aplikacije ovakvog formata uglavnom se koriste za organiziranje i korisne su za čuvanje informacija na jednom mjestu. Ova aplikacija imala je puno verzija dok nisam naišao na najbolje rješenje i naravno, još uvijek ima mjesta za nadogradnje kao što su dodavanje proizvoda u favorite, notifikacije i slično. Isprobavanjem naišao sam na puno načina rješavanja nekih problema kao što su validacija, bolja organizacija s UnitOfWork obrascem i sl. Završnim radom sam vrlo zadovoljan i mislim da je većina zamišljene funkcionalnosti ispunjena.

**LITERATURA**

1. MS SQL Server: <https://www.microsoft.com/en-us/sql-server/sql-server-2019>
2. .NET Framework <https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet-framework>
3. Visual Studio <https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019>
4. ASP.NET <https://dotnet.microsoft.com/apps/aspnet>
5. NuGet <https://docs.microsoft.com/en-us/nuget/what-is-nuget>
6. WebAPI <https://dotnet.microsoft.com/apps/aspnet/apis>
7. Entity Framework <https://docs.microsoft.com/en-us/ef/>
8. LINQ <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/>
9. Migracije <https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli>
10. Autentikacija <https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/authentication-and-authorization-in-aspnet-web-api>
11. OAuth2 <https://oauth.net/2/>
12. JWT <https://jwt.io/>
13. Fluent Validation <https://docs.fluentvalidation.net/en/latest/installation.html>
14. MVC <https://dotnet.microsoft.com/apps/aspnet/mvc>
15. HTTP klijent <https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclient?view=netcore-3.1>
16. Kolačići <https://docs.microsoft.com/en-us/aspnet/web-api/overview/advanced/http-cookies>

1. Model koji prikazuje entitete i njihove veze u bazi podataka [↑](#footnote-ref-1)
2. Odabirom Web API projekta automatski se u projekt uključuje i MVC predložak [↑](#footnote-ref-2)