Dokumentacija

Student: Petrović Mladen

Predmet: Web programiranje ASP

Broj indeksa: 103/16

Sadržaj

Uvod

Korišćene tehnologije

.NET CORE, Entity Framework, Razor

Opis funkcionalnosti

Podela projekta

Projekat se nalazi u 6 fizički odvojenih projekata. Domain, DataAccess, Application, EfCommands, Api, WebApp.

Aplikacija je namenjena za vesti. Dakle ideja je da se dodaju vesti, kategorije, novinari, useri, i sl. Za svaku vest se selektuje niz novinara (journalist-a) koji su je pisali, kategorija, ime, opis, slika i sl. CRUD operacije takođe realizovane za uloge, polove, i ostale entitete.

Projekti su odvojeni, između ostalog zbog pridržavanja separation of concenrce principa. Arhitektura projekta, omogućava modularnost, jer generalno softver (engl soft), treba da bude dovoljno mekan, odnosno podložan promenama. Cilj ovakve vrste arhitektura je rešavanje zavisnosti.

Domen

Tu se nalaze sledeće klase: Category, Gender, Journalist, Log, Role, Story, User, StoryJournalist. Ove klase su se preko ORM-a mapirale u bazu podataka info_news.

Najbitniji delovi projekta su oni koje ne zavise ni od koga. U ovom slučaju to je Domain. U pitanju je klasna biblioteka, tako da nema Main izvršni program, jer se sam projekat nikad ni neće pokretati, već služi samo da ga ostale aplikacije koriste. On je najbitniji, zato što će se retko kad menjati. U ovom slučaju se radi

projekat za vesti, tako da koliko god se projekat menjao, neminovno je da će uvek postojati novinari, vesti, kategorije, i sl. Isto tako će biti, i za bilo koji drugi projekat. Tu se semantički nalazi domenska politika same aplikacije, a fizički se nalaze klase, sa čijim će objektima aplikacija da radi.

DataAccess

Context, Migracije, Konfiguracije. Za svaki entitet napisana konfiguracija, rešene veze...

Sledeći deo po bitnosti je DataAccess. U projektu se radio Code First pristup, odnosno napravile su se domenske klase na osnovu kojih se kreirala baza podataka. Pored Code First pristupa postoji i Database First, gde se prvo kreira baza pa onda objektni deo, a postoji i Model First, koji se skoro nikad ne koristi, gde se radi preko gui-ja, i na osnovu koga se dobiju i klase u c# ali i baza podataka.

Sam po sebi DataAccess nije najbitniji. Realno, uvek je moguće da se projekat prebaci na neki drugi vendor baze podataka. U nekom trenutku, teoretski aplikacija može da bude usporenija, zbog ogromnog broja korisnika, i zato postoji koncept Big Data. Kada postoji enormno veliki broj korisnika, primer instagram, nije realno da se svi podaci čuvaju na recimo sql serveru. Postoje ultra brze nerelacione baze, gde se vrlo brzo mogu dobiti traženi podaci.

Ukoliko u projektu ne postoji apstrakcija, tj usko je ograničen na recimo sql server, upada se u problem kada se želi prebacivanje na novi vendor b.p. Razlog tome, je što je, umesto apstrakcije, projekat usko vezan za implementaciju. U tim trenucima je najlakše da se ceo projekat počne ispočetka.

Koristeći ovu arhitekturu, apstrakcija se podiže interfejsima i apstraktnim klasama. Time se rešila sledeća stvar: Definišu se metode u interfejsu, koji će se implementirati u bilo kakvoj vrsti data accessa. Onda je potrebno samo da za novi pristup implementiraju metode, i zamene reference, tj da se kaže da se neće više koristiti sql server, nego npr. elastic search.

Application

U ovom delu se nalazi sva poslovna logika.

Exceptions: Tu su definisani domenski izuzeci koji će se bacati.

DataTransferObjects: Definisane DTO klase. Koncept DTO-a je uveden iz nekoliko razloga. Jedan od njih je što klijentu, ne trebaju da se vraćaju svi podaci (npr neće se vraćati lozinka određenog usera), drugi razlog je neotkrivanje strukture baze podataka, itd.

Interfaces:

Definisani interfejsi ICommand<TRequest>, ICommand<TRequest, TResponse>, IEmailSender.

Obzirom da ceo informacioni sistem, na najvišem nivou apstrakcije, može da se svede na ili slanje nekih podataka gde se ne dobija ništa nazad, ili slanje određenih podataka i dobijanja nečeg, uvedena su prva 2 interfejsa. Oba su generička, imaju metod Execute, s tim što prvi služi samo da od klijenta dobije nešto i izvrši neku poslovnu logiku nad njima (koristi se za insert, update, delete), dok drugi dobija nešto (u konkretnom primeru su bili samo Search objekti) i vraća nazad nešto (u projektu isfiltrirane podatke). IEmaISender propisuje svojstva za slanje maila.

Helpers: FileUpload. Interfejs koji sadrži pravila za ekstenzije fajla koji se šalje.

Searches: Klase u kojima se nalaze svojstva koja će se koristiti za pretragu.

Commands: Za svaki entitet definisani interfejsi, koji će se koristiti za CRUD operacije. Raspoređeni su po folderima.

U ovim interfejsima, se ne nalazi nijedno svojstvo niti metod. Oni služe isključivo zbog modularnosti, tj da, kada se definiše nekoliko implementacija u svakom trenutku može da se kaže “sada koristi ovu implementaciju”, a da ne mora da se ide po kodu i menjaju u svakom fajlu reference, logika, itd.

Responses: Tu se nalazi klasa PagedResponse koja će nam služiti za paginaciju.

Klijentu šaljemo ukupan broj resursa, broj trenutne strane, ukupan broj strana, kao i konkretne podatke.

EfCommands

Fajlovi organizovani po folderima. Za svaki entitet, definisane konkretne implementacije u Entity Frameworku za CRUD operacije. Tu se nalaze klase, koje implementiraju, svaka po određeni interfejs iz Application sloja.

Ovde omogućeno između ostalog i slanje Mail-a prilikom inserta novog usera.

Kada neko bude rešio da se prebaci na drugu bazu, taj neko će samo napisati posebnu implementaciju svakog od ovih interfejsa. Npr: ElasticSearchCommands

API

Sadrži kontrolere tj request handlere, za svaki entitet. U svakom metodu se poziva određena komanda i vraća odgovarajući status kod. Za svaki GET end point obezbeđena mogućnost pretrage, filtriranja, paginacije.

Koja se implementacija komande koristi definisano je u StartUp-u, tačnije u dependency injection containeru.

Manje bitan deo aplikacije, zavisi od ostalih. Slično kao web sajt samo bez guija. Služi da drugi programeri gađaju određene rute, tj metode u kontroleru, pri čemu će se neka poslovna logika odraditi, a isti će biti obavešteni putem statusnih kodova.

WebApp

Omogućene CRUD operacije za entitete Story i Journalist. Dakle za vesti i novinare. Tu se omogućava i upload slike, kao i validacija.

Konkretna aplikacija sa kojom će kranji korisnici imati fizički kontakt. Zavisi od svih ostalih, osim apija. Kontroleri su slični kao u apiju.

Dizajn baze podataka