Testiranje i implementacija web servisa u Spring frameworku

SEMINARSKI RAD IZ PREDMETA NAPREDNE WEB TEHNOLOGIJE

Amila Borančić 17934/1705 DATUM, Sarajevo

# SADRŽAJ

Uvod

Implementacija i testiranje servisa u Springu

nesto?

Unit testovi

Integracijski testovi

Finance Tracker aplikacija

Popis funkcionalnosti i ERD

Setup projekta

Tabele i konekcija na bazu

Kontroler

Demonstracija rada aplikacije

Zaključak

Reference

Popis slika

Isječci koda

# Uvod

Spring Boot je framework koji je uveliko olakšao posao inžinjera koji se bave izgradnjom i razvojem web servisa za mikroservisnu arhitekturu. Spring je intuitivan framework napisan u programskom jeziku Java koji omogućava razvoj aplikacija uz minimalnu količinu ručne konfiguracije. Pored olakšane implementacije web servisa, Spring nudi i mnogo dependencija koji su namijenjeni za pisanje integracijskih i unit testova kako bi se osiguralo da servisi rade u skladu sa svojim namjenama.

U sklopu ovog rada se implementira Finance Tracker aplikacija u Spring Boot frameworku. Za ovu svrhu prvo će se definisati funkcionalnosti koje će ova aplikacija imati, kao i njen ERD. Biće obrazložene različite vrste testova za web servise, zašto su važni i kako se implementiraju u Springu na Finance Tracker aplikaciji.

# Implementacija i testiranje servisa u springu

U ovom poglavlju se daje kratki uvod u metodologiju implementiranja i testiranja web servisa pomoću Spring Boot-a. Prva podsekcija se dotiče ključnih koraka implementacije, dok naredne dvije služe da se izlože *unit* i *integration* testovi kao primjeri dvije najčešće vrste testiranja koje programer servisa vrši.

## Implementacija jednostavnog web servisa

Kako bi se web servis što uspješnije i brže implementirao, potrebno je na neki struktuiran način pristupiti njegovoj implementaciji.

Prvi korak ka tome je definicija osnovnih funkcionalnosti servisa, kao i kreiranje *entity-relationship* dijagrama na kome se jasno vidi kako će izgledati baza servisa. Drugi korak je postavljanje odgovarajuće strukture u projektu. Nešto više o organizaciji projekta biće rečeno prilikom osvrta na praktični primjer Finance Tracker aplikacije.

Idući korak je odabir baze koja će se koristiti, kao i inicijalizacija projekta pomoću Spring Initializr platforme, koja uveliko olakšava i skraćuje posao developera [1]. Zatim se prelazi na izgradnju model klasa na osnovu ranije nacrtanog ERD-a koji će se koristiti da se pomoću *code-first* pristupa na osnovu tih klasa izgenerišu tabele za bazu podataka [2]. Atribute modela je potrebno validirati pomoću anotacija [3].

Nakon što imamo kreirane model klase, potrebno je kreirati *service, repository* i *controller* klase koje sadrže gotovo pa sav funkcionalni dio koda. Detaljan opis ovih klasa i čemu one služe će biti dat u narednom poglavlju.

Što se tiče testiranja implementiranog servisa, Spring omogućava generisanje testnih klasa za svaku *service* i *controller* klasu, što naravno nudi uštedu vremena.

## Unit testovi

*Unit test* je test malog opsega koji developer piše kako bi bio siguran da neki određeni dio servisa radi ono za šta je namijenjen. Unit, odnosno jedinica koda koja se testira treba biti što je moguće manja i, ukoliko je moguće, izolirana od ostatka koda [4]. U svrhu unit testiranja, Spring Boot nudi mnoge dependencije, od kojih su dva najpopularnija JUnit i Mockito [5].

## Integracijski testovi

*Integracijski test* je test koji služi da se ustanovi da li različiti dijelovi aplikacije koja se gradi ispravno rade zajedno. Integracijski testovi u Springu se najčešće koristi da se ustanovi da li biznis sloj i *data persistence* sloj ispravno komuniciraju i da li je servis konektovan na odgovarajući bazu [6]. U svrhu integracijskog testiranja, Spring Boot nudi mnoge dependencije, od kojih je odabrana INSERT NAZIV DEPENDENCIJA za Finance Tracker aplikaciju.

# Finance Tracker

## Popis funkcionalnosti i ERD

Prvi korak prilikom izgradnje bilo kakve aplikacije je kreiranje popisa osnovnih funkcionalnosti, koji se kasnije može širiti ili modificirati po potrebi. Finance Tracker aplikacija treba da u sebi sadrži razne vrste transakcija, prihoda i rashoda korisnika.

Popis osnovnih funkcionalnosti:

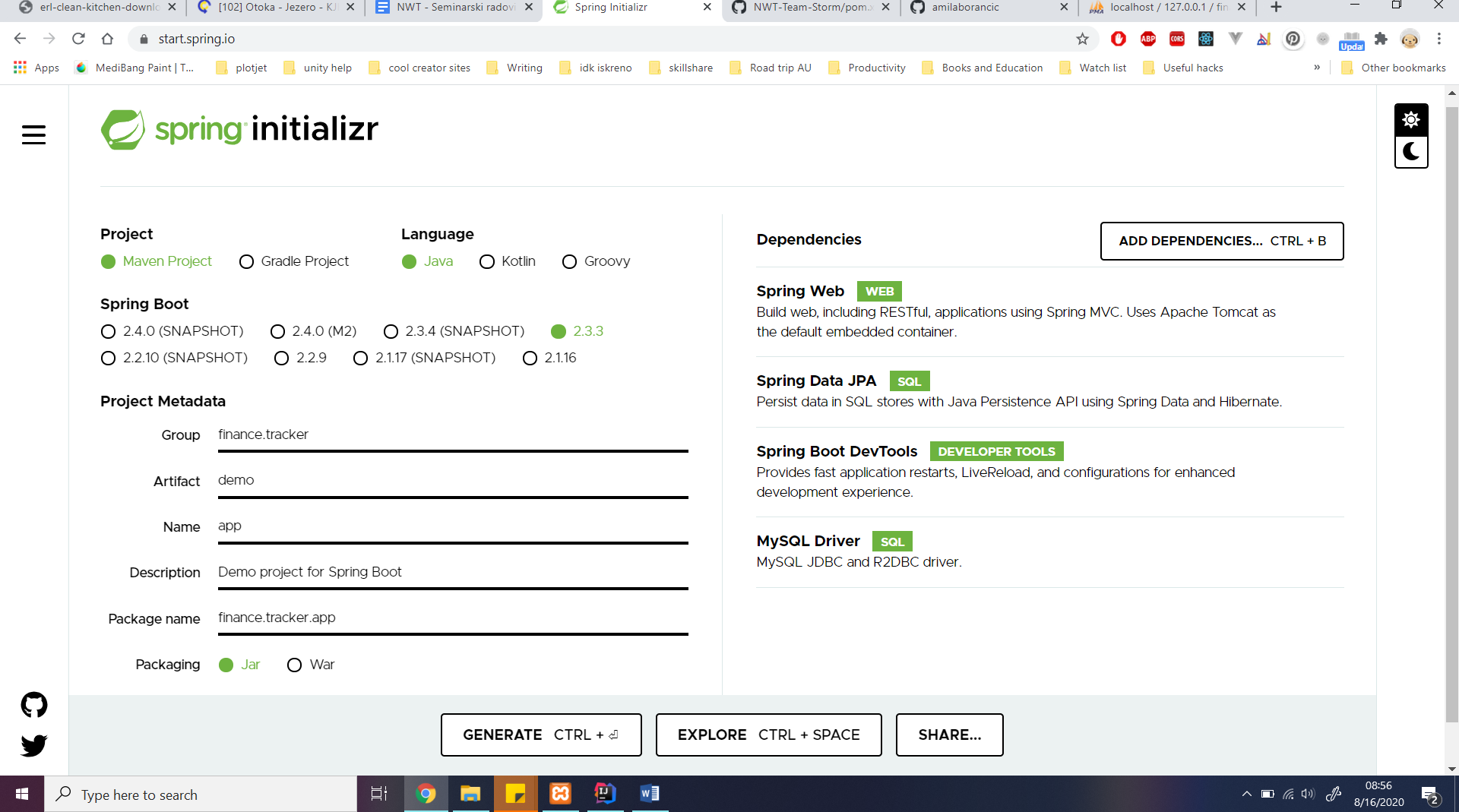
* kreiranje nove transakcije
* sortiranje transakcija po iznosu (rastuće i opadajuće)
* filtriranje transakcija po tipu (INDIVIDUALPAYMENT, REGULARPAYMENT, PURCHASE, INDIVIDUALINCOME, REGULARINCOME)

Svaka transakcija treba da ima svoj naziv, datum, iznos, tip (jedan od navedenih u tekstu iznad), opis kupljenog proizvoda (NULL ukoliko je tip transakcije neki od prihoda) i interval (broj dana nakon kojeg se transakcija REGULAR tipa ponavlja).

Svaki korisnički račun treba da ima trenutno novčano stanje računa.

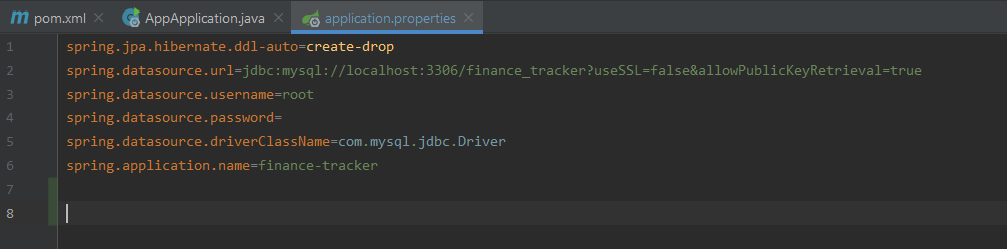
## Setup projekta pomoću Spring Initializr

Spring Initializr se koristi kako bi se izgenerisao početni projekat. Odaberu se početni dependencies, verzija Jave koja je podržana na mašini i daje se naziv projektu kao na slici ispod:



## Konekcija na bazu, kreiranje tabela i punjenje baze prilikom pokretanja aplikacije

Kako aplikacija ne bi radila sa hardkodiranim podacima, potrebno je odabrati način spajanja na bazu i bazu popuniti nekim testnim podacima. Spring nam nudi ugrađenu bazu podataka koja se može koristiti pomoću H2 ili HSQL dependencija [7] kao i driver za MySql bazu podataka, koji je odabran za ovaj projekat. U application.properties fajlu definišemo connection string, metod kojim će se puniti tabele u bazi prilikom svakom pokretanja (create-drop znači da će se prilikom svakog pokretanja tabele nanovo kreirati a prilikom svakom gašenja servisa brisati), kredencijale za pristup bazi i još par detalja.



Nakon što su završene postavke za pristup bazi, iduće što je potrebno uraditi je kreirati modele koji će predstavljati tabele za bazu. Modeli za Finance Tracker aplikaciju su Transaction i Account, kao što je prikazano na ERD-u ranije. Kako bi projekat bio dobro organizovan, preporučljivo je razdvojiti svu logiku aplikacije u zasebne foldere, odnosno *packages* u Spring Boot terminologiji, pa je to urađeno i u ovom projektu.

# Reference

[1] Spring Initializr: <https://freecontent.manning.com/wp-content/uploads/initializing-a-spring-boot-project-with-spring-initializr.pdf> [Posljednji pristup: 17.08.2020.]

[2] Code-first pristup: <https://www.tutorialspoint.com/entity_framework/entity_framework_code_first_approach.htm#:~:text=Code%20First%20modeling%20workflow%20targets,C%23%20or%20VB.Net%20classes.> [Posljednji pristup: 17.08.2020.]

[3] Validacija pomoću anotacija: <https://www.baeldung.com/spring-boot-bean-validation> [Posljednji pristup: 17.08.2020.]

[4] Unit testovi: <https://stackoverflow.com/questions/5357601/whats-the-difference-between-unit-tests-and-integration-tests> [Posljednji pristup: 15.08.2020.]

[5] JUnit i Mockito: <https://www.springboottutorial.com/spring-boot-unit-testing-and-mocking-with-mockito-and-junit> [Posljednji pristup: 15.08.2020.]

[6] Integracijski testovi: <https://reflectoring.io/spring-boot-test/> [Posljednji pristup: 15.08.2020.]

[7] Embedded baze podataka za Spring: <https://www.baeldung.com/spring-boot-h2-database> [Posljednji pristup: 16.08.2020.]