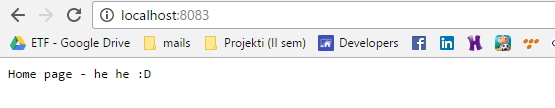
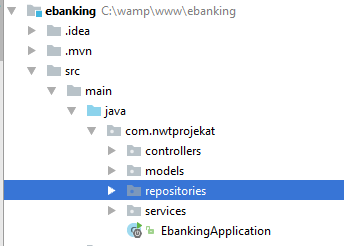
Nakon par sati guglanja(istraživanja) uspio sam uspostaviti Spring Boot projekat. Na stranici <https://start.spring.io/> se nalazi Spring Initializr koji nakon sto se odaberu generira pocetnu strukturu vec spremnu za rad. Kao IDE koristim IntelliJ.

Jedna od prvih stvari bila je podignuti aplikaciju na localhostu na portu 8083 (defaultni port 8080 mi je iz nekog razloga zauzet).

Root url localhost:8083/ mi vrati sljedeci view:



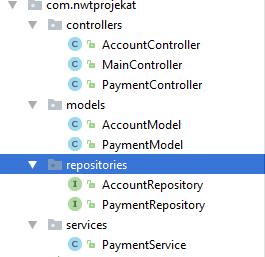
Do ovoga bilo je jednostavni doci, jer na spring.io ima dosta dobrih primjera. Po nekoj konvenciji u strukturi projekta je potrebno da imaju folderi controllers, models, services, repositories i naravno main java file.



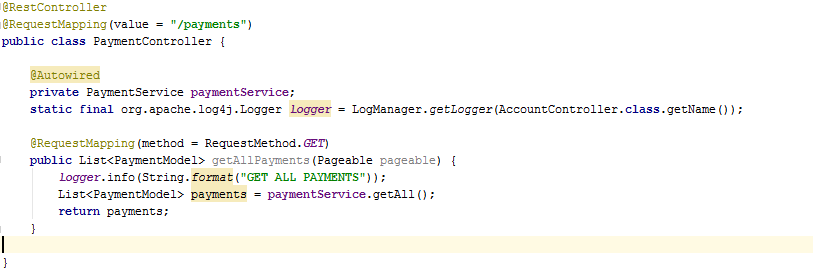
Izgled kontrolera koji je zaduzen za root url:

**package** com.nwtprojekat.controllers;  
  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.ResponseBody;  
@Controller  
**public class** MainController {  
 @RequestMapping(**"/"**)  
 @ResponseBody  
 **public** String index() {  
 **return "Home page - he he :D"**;  
 }  
}

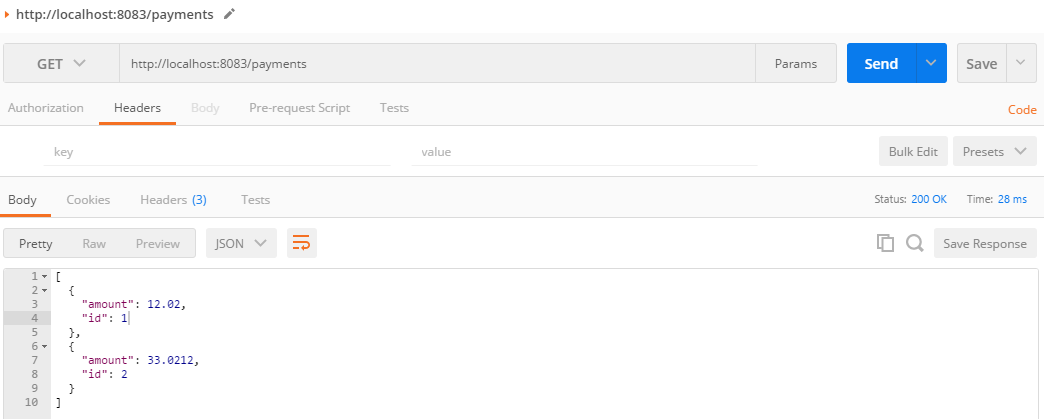
U samom projektu, ja sam zaduzen za modul Payments (Nalozi) tako da sam dodao PaymentController, PaymentModel, PaymentRepository, PaymentService



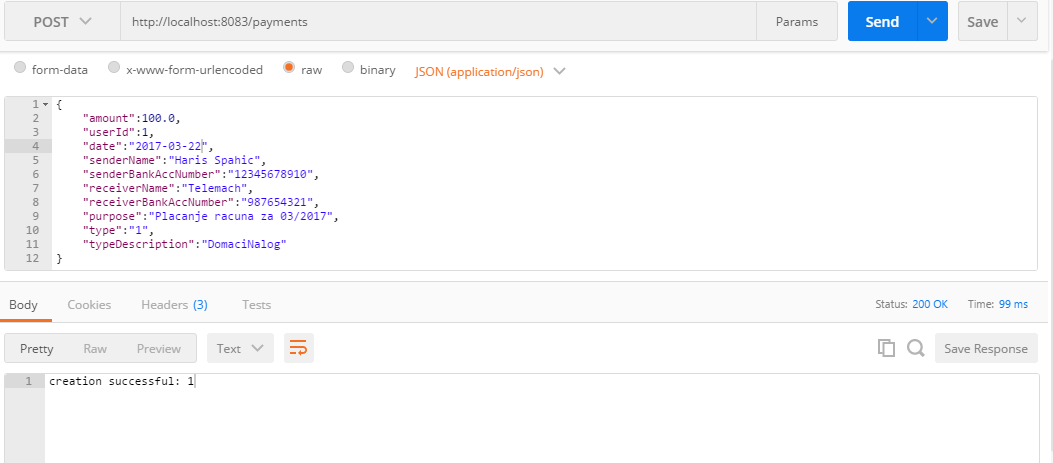
PaymentController izgleda ovako



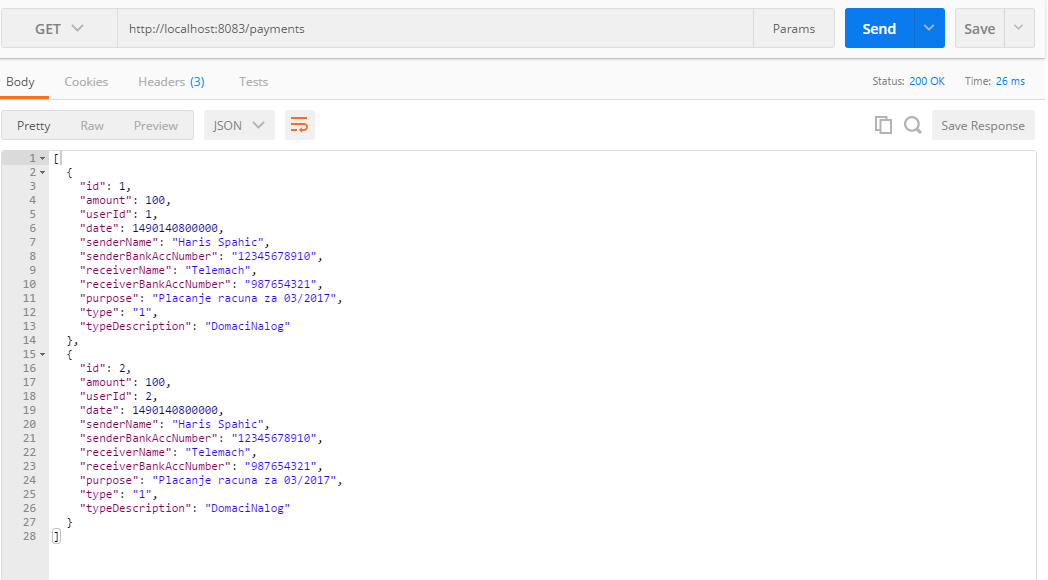
Ovo znaci da sam dodao jednu metodu koja vraca sve paymente iz baze a response izgleda ovako:



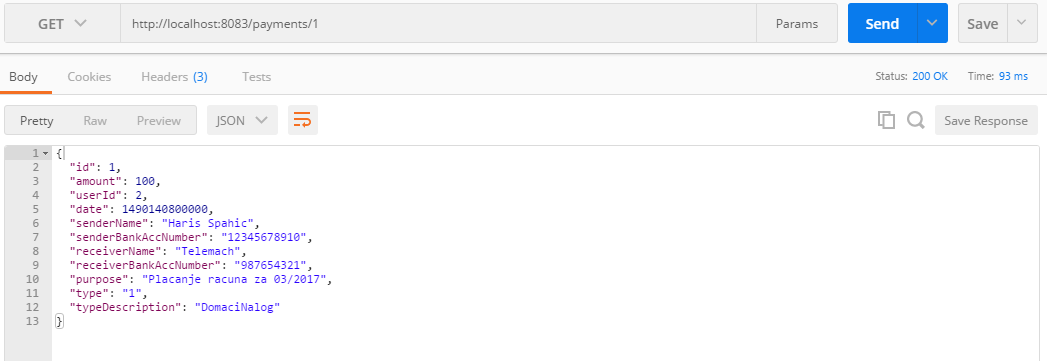
Insert u bazu ili POST metoda izgleda ovako



Nakon toga GET svih naloga:



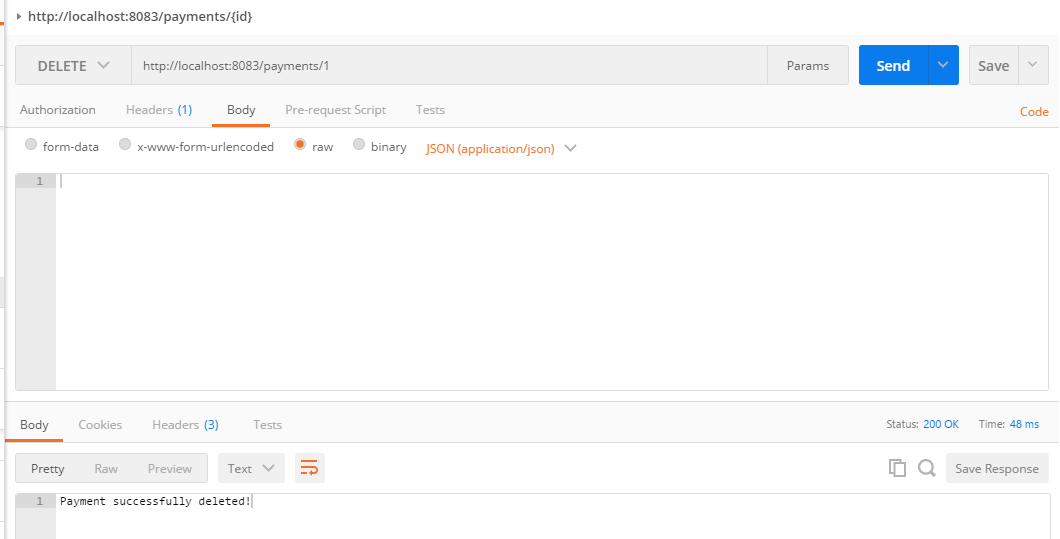
Dobavljanje naloga po ID:



PUT metoda:



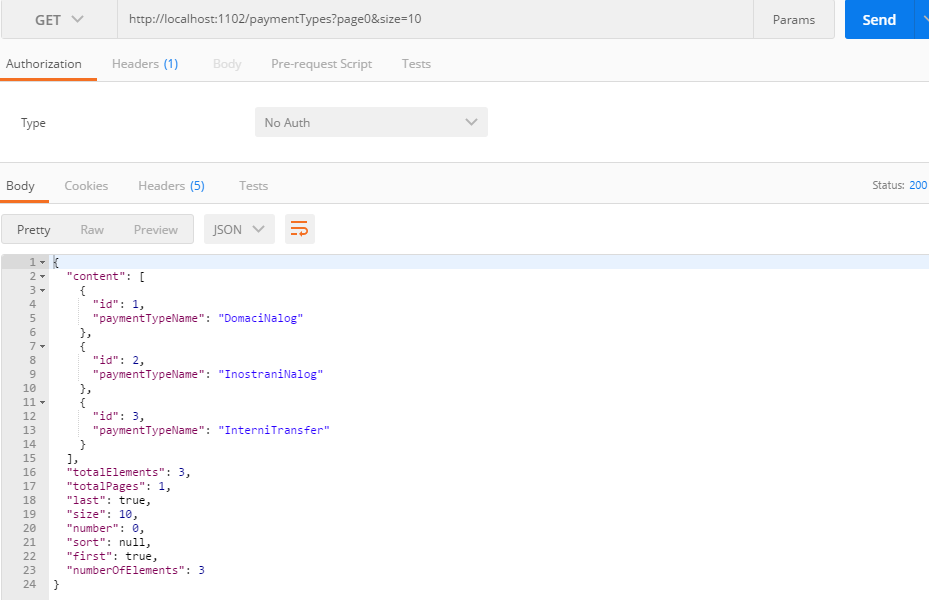
DELete metoda:



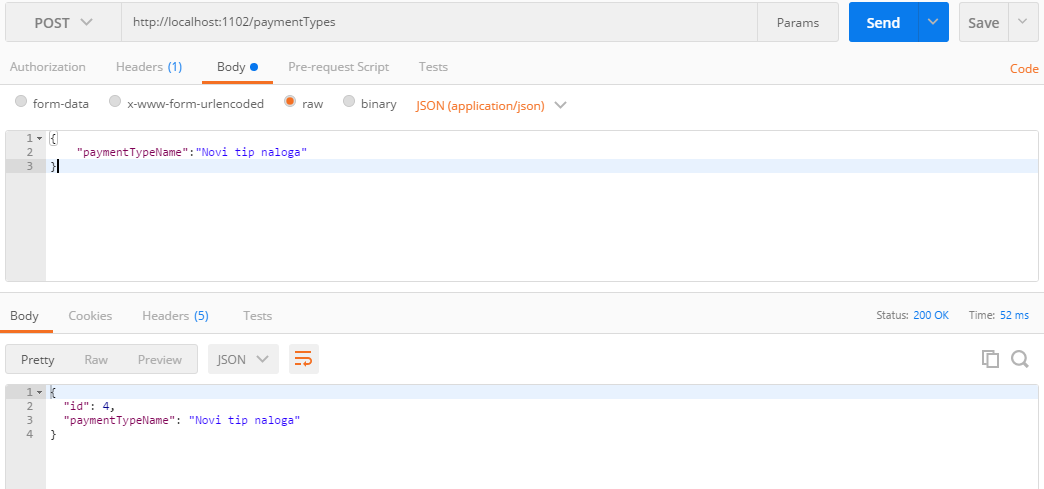
Dodan i EP paymentTypes

Bilo je izmijena u metodama GET, prije se vracao objekat List, sada se vraca objekat Page.

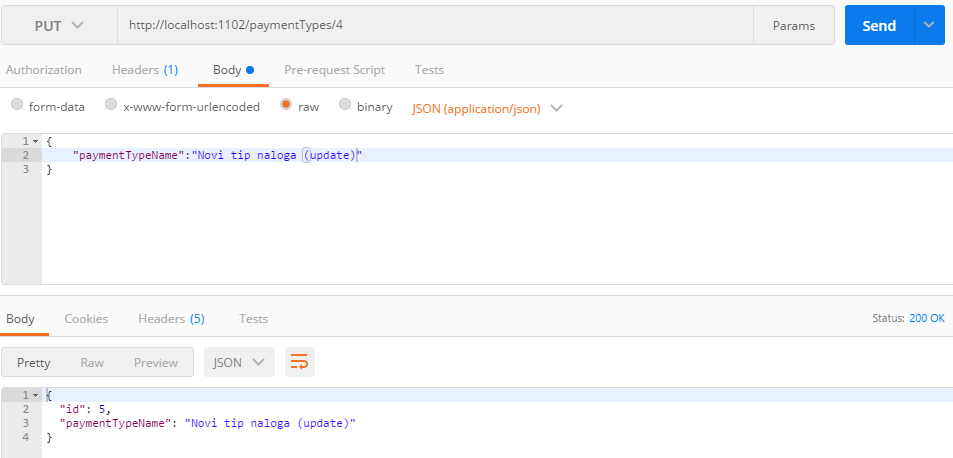
To sam iskoristio zbog JPARepository metode koja ima mogucnost citanja parametara ?page=0&size=10



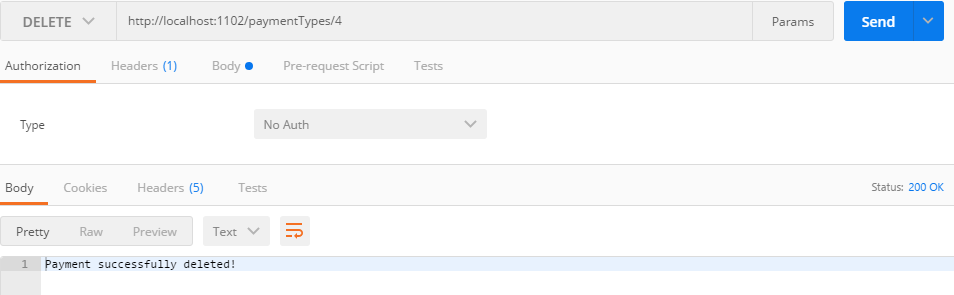
/paymentTypes POST metoda



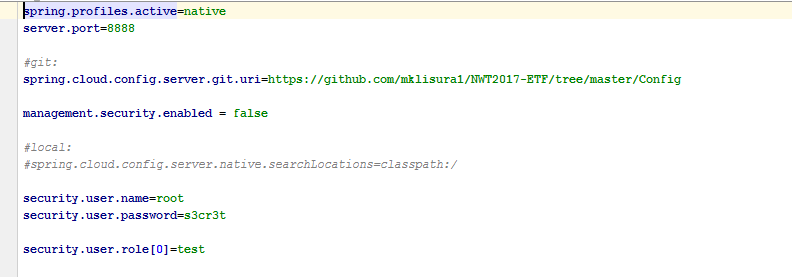
/paymentTypes PUT metoda



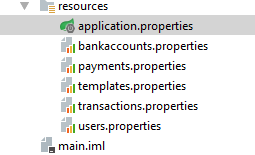
/paymentType DELETE metoda



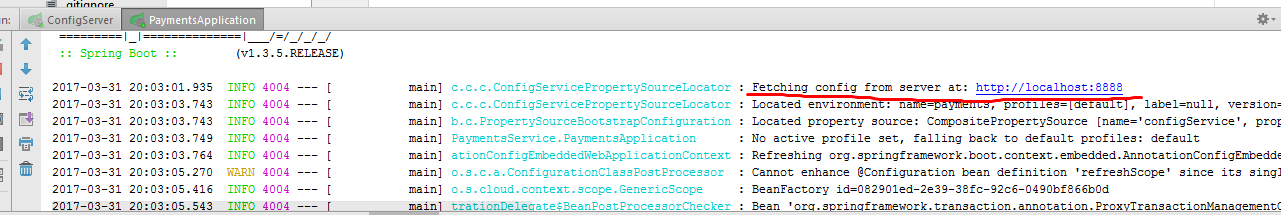
Osnovi podaci naseg Central Config servera:



Svaki modul na Centralnom serveru ima svoje config podatke

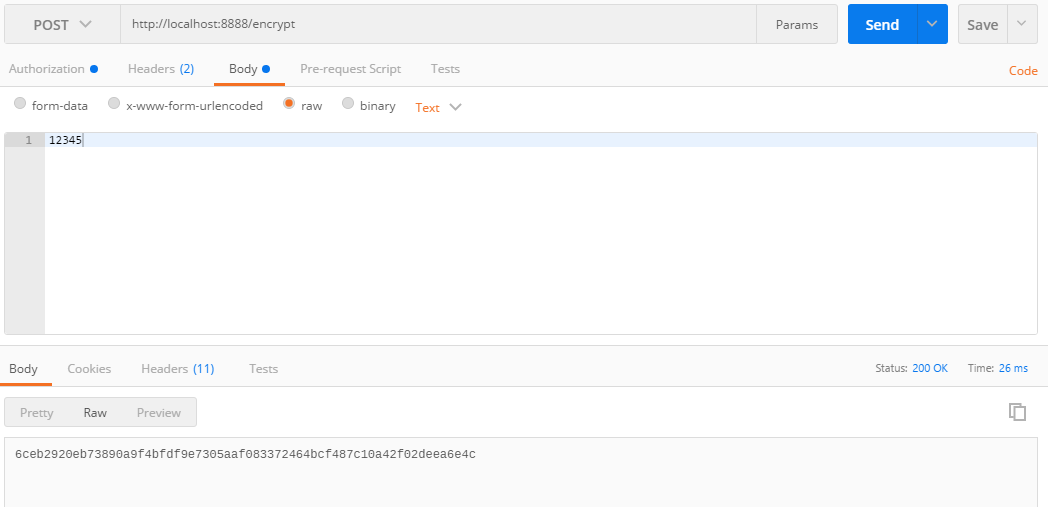


Kada se pokrece svaki pojednicani modul odnosno mikroservis vidimo da se kupe podaci sa Centralnog servisa:

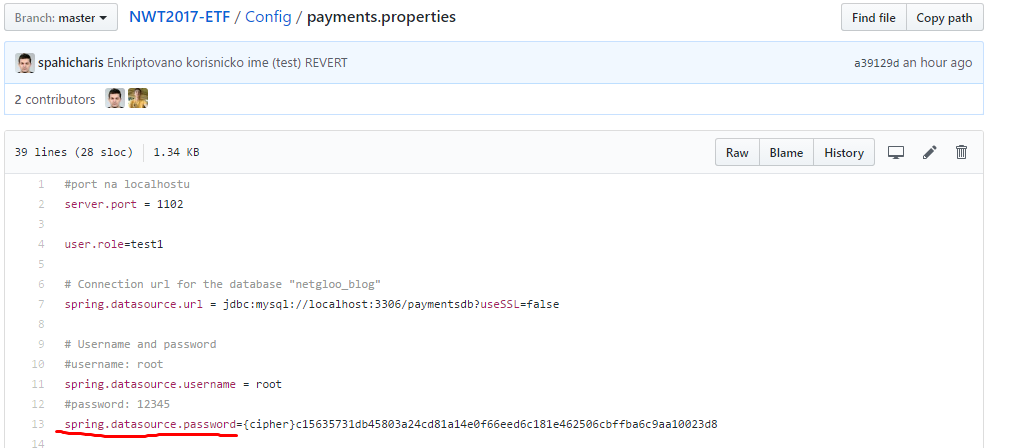


Dodatni zadaci kod zadace 2.

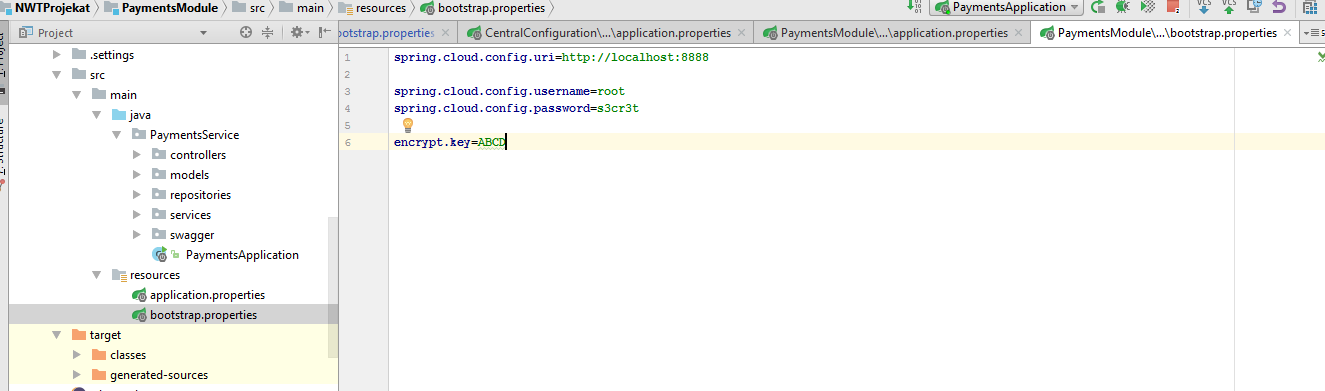
Generisanje enkriptovanog kljuca takodjer se radilo preko POST metode na Serveru:



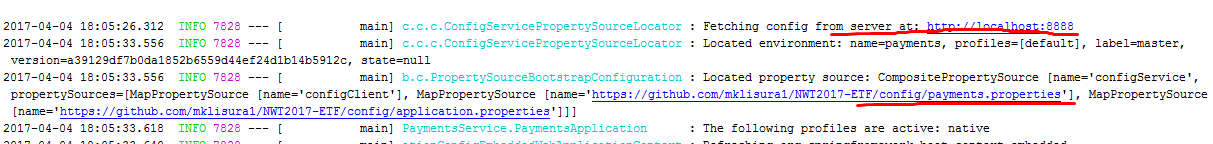
Ovdje je enkriptovan password za bazu podataka. Dobijeni hash string se stavio na mjesto do tad postavljenog stringa 12345:



Da bi klient ovo znao desifrovati potrebno je u bootstrap.properties ubaciti kljuc (ABCD)

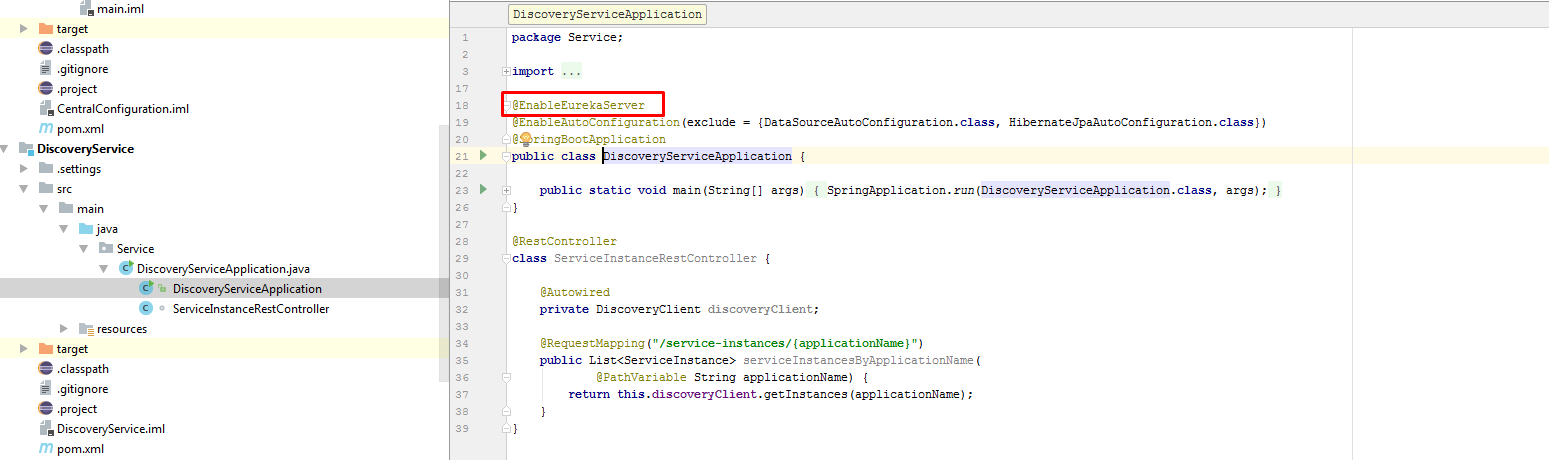


Tako da ce prilikom uspostave konekcije sa bazom client dobaviti hashiran password sa servera i uz key desifrovati i iskoristiti password za konekciju:

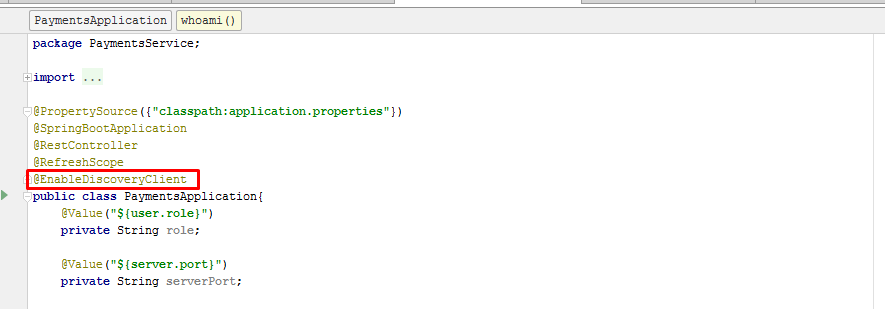


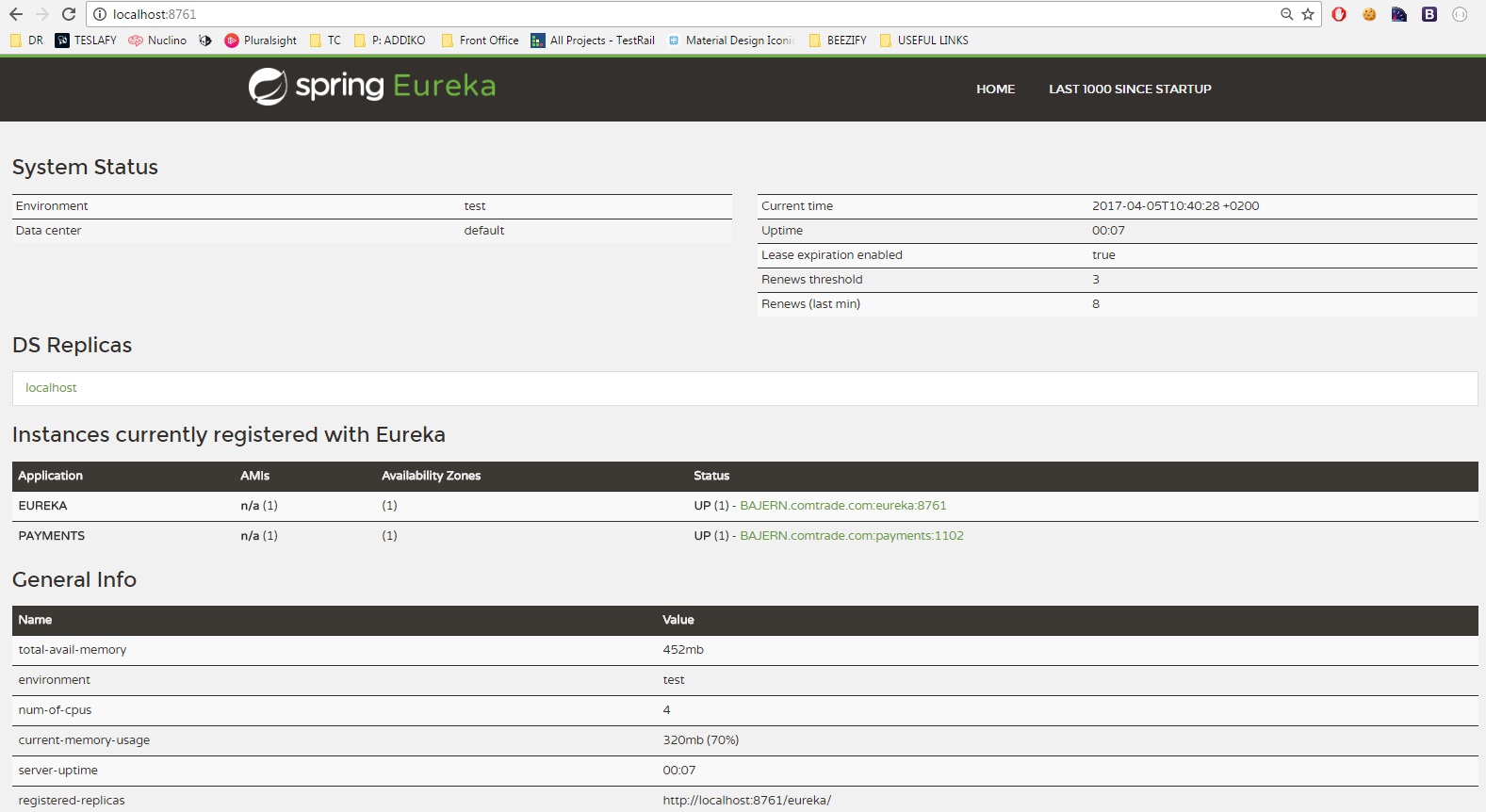
Zadatak 3 – EUREKA

Da bi se podesila eureka server, potrebno je dodati novi projekat(module) I main classu oznaciti notacijom EnableEurekaServer:



Zatim se podesi klijent, na nacin da se doda notacija EnableDiscoveryClient



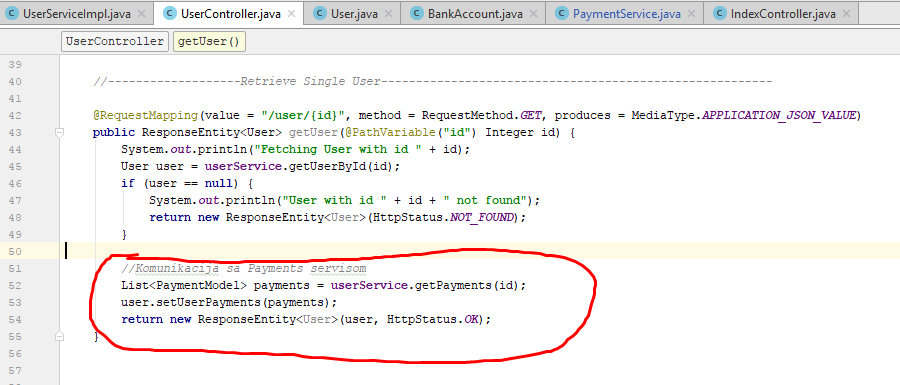
Da bi se provjerilo da je sve uredu, pokrenu se redom ConfigServer, EurekaServer I Client, u mom slucaju Client je PaymentsModule, te se nakon pokretanja svih servisa otvori localhost:8761:   


A na clientu kada se pozove metoda service-instances/{applicationName} u mom slucaju /payments:

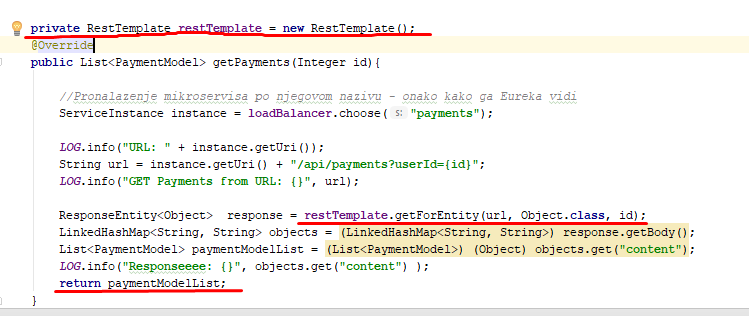


Ispod je prikazana komunikacija izmedju UsersService-a i PaymentsService-a.

Na slici ispod vidimo da se dohvataju svi nalozi vezani za id:



DiscoveryServis se brine da pronadje mikroservis po svome nazivu (“payments”):



Na kraju to izgleda ovako

