

Univerzitet u Sarajevu
Elektrotehnički fakultet
Računarstvo i informatika

Projektna dokumentacija

Online platforma za učenje C++-a

Tasks Module

Predmet: Praktikum-Napredne web tehnologije

Džana Feratović
br.indeksa: 1161/16778
Tim 1

Sarajevo, mart 2017.

SADRŽAJ

1. Modul Zadaci	3
2. Osnovne funkcionalnosti modula	3
2.1. Unos novog zadatka	3
2.2. Pregled i rješavanje dostupnih zadataka	3
2.3. Pregled liste najboljih rješenja određenog zadatka	4
3. Entity-relationship dijagram modula	4
4. Package Explorer modula u Spring-u.....	5
5. Rezultati.....	5
HTTP GET metoda.....	5
HTTP POST metoda	10
HTTP PUT metoda	10
HTTP DELETE metoda	10
6. Implementacija Rest Controllera.....	11
7. Konfiguracija centraliziranog servera	16
8. Postavljanje Eureka klijenta	17
9. RestTemplate komunikacija	18
10. RabbitMQ komunikacija	24

1. Modul Zadaci

Modul Zadaci obuhvata osnovne funkcionalnosti koje se odnose na postavljanje i rješavanje zadataka kroz online platformu za učenje C++ programskog jezika. Kroz ovaj modul registrovanom i prijavljenom korisniku se omogućava rješavanje zadataka dostupnih na stranici, kao i pregled ostvarenog rezultata za već riješene zadatke. Također, ovaj modul omogućava korisnicima pregled liste najboljih rješenja za određeni zadatak. Zadatke koji se rješavaju kroz online konzolu mogu postaviti samo prijavljeni korisnici uz određene uvjete za objavljivanje zadatka. Opisane funkcionalnosti će nešto detaljnije biti obrađene kroz poglavlje koje slijedi. Također, u jednom od narednih poglavlja dat je i entity relationship dijagram prema kojem je implementirana baza ovog modula.

2. Osnovne funkcionalnosti modula

Osnovne funkcionalnosti modula Zadaci:

- Unos novog zadatka
- Pregled i rješavanje dostupnih zadataka
- Pregled liste najboljih rješenja određenog zadatka

2.1. Unos novog zadatka

Modul Zadaci omogućava prijavljenim korisnicima da unesu novi zadatak, pri čemu je potrebno da korisnik, osim tekstualne postavke zadatka, unese ulaze i očekivane izlaze u svrhu testiranja rješenja, kao i svoje, početno rješenje zadatka koje prolazi navedene testove. Nakon svih potrebnih unosa, zadatak postaje dostupan ostalim korisnicima za rješavanje. Korisnik koji je postavio zadatak može isti i obrisati, dok korisnici sa administratorskim privilegijama imaju mogućnost brisanja bilo kojeg od dostupnih zadataka.

2.2. Pregled i rješavanje dostupnih zadataka

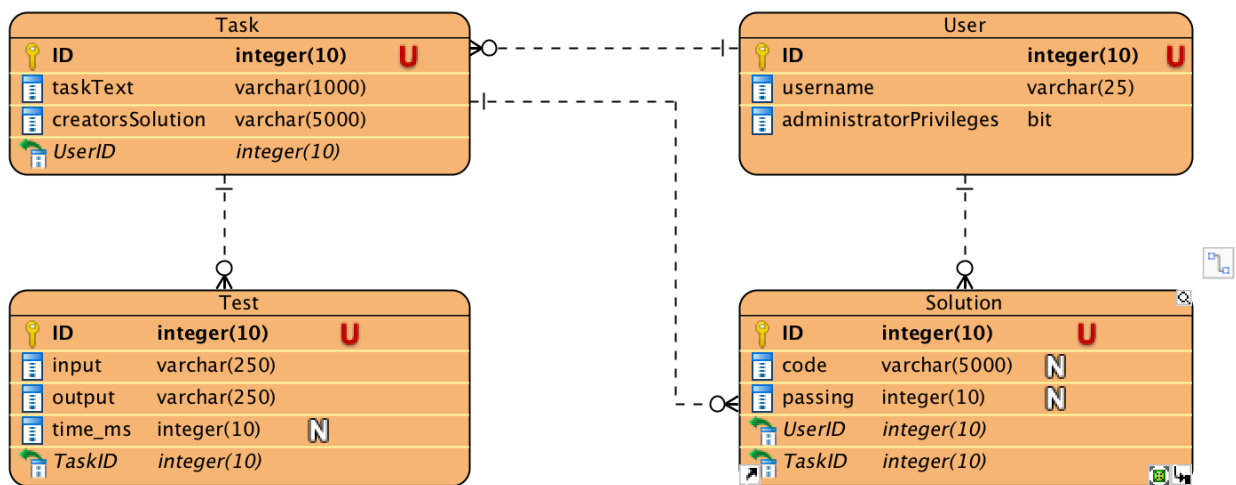
Jedna od funkcionalnosti modula Zadaci je i pregled zadataka dostupnih za rješavanje, kao i unos rješenja za odabrani zadatak. Prijavljeni korisnik, nakon odabira određenog zadatka sa stranice, unosi C++ programsko rješenje u dostupnu konzolu koje će dalje proći određene testove u svrhu utvrđivanja valjanosti rješenja.

Korisniku je omogućen pregled trenutnog rezultata (vrijeme izvršenja koda, broj testova koje je rješenje prošlo etc.) za riješeni zadatak.

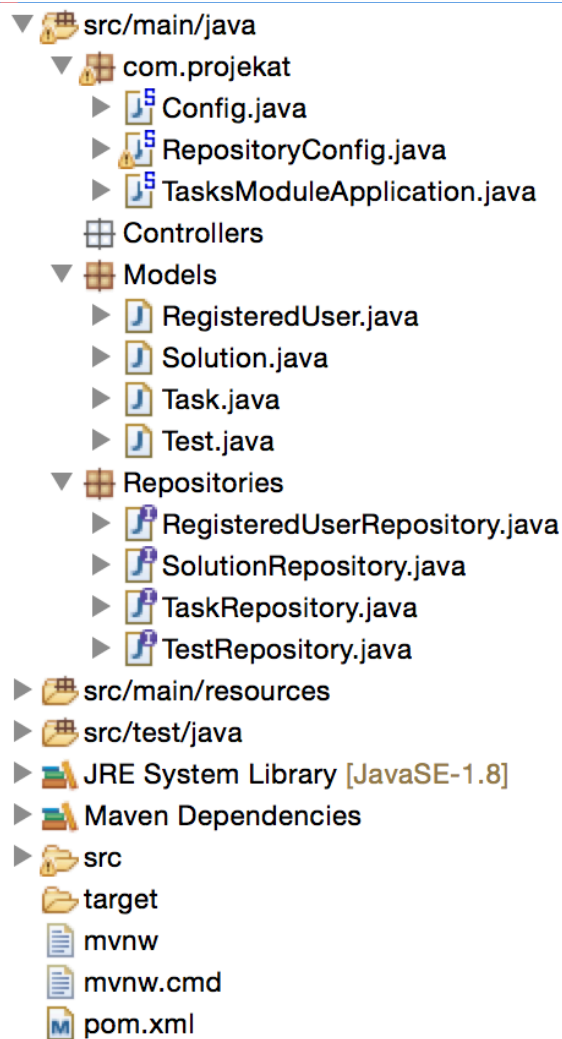
2.3. Pregled liste najboljih rješenja određenog zadatka

Prijavljenom korisniku je, kroz ovaj modul, omogućen i pregled korisnika i njihovih rješenja za odabrani zadatak koji su, po određenom kriteriju, najbolja rješenja.

3. Entity-relationship dijagram modula



4. Package Explorer modula u Spring-u



5. Rezultati

U nastavku će biti prikazan rezultat dosadašnje implementacije, te će se, korištenjem curl-a, testirati rad izgrađene baze.

HTTP GET metoda

Na nekoliko slika koje slijede bit će prikazan rezultat curl http GET metode. Na prvoj slici prikazan je GET svih entiteta User, dok je na drugoj slici prikazan GET specifičnog User-a po ID-u. Slično je prikazano za tabele Task, Test i Solution.

```
Dzana:~ Dzana$ curl http://localhost:8080/users
{
  "_embedded" : {
    "users" : [ {
      "id" : 1,
      "username" : "Dzana",
      "administratorPrivileges" : null,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/users/1"
        },
        "registeredUser" : {
          "href" : "http://localhost:8080/users/1"
        },
        "solutions" : {
          "href" : "http://localhost:8080/users/1/solutions"
        },
        "tasks" : {
          "href" : "http://localhost:8080/users/1/tasks"
        }
      }
    }, {
      "id" : 2,
      "username" : "Lala",
      "administratorPrivileges" : true,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/users/2"
        },
        "registeredUser" : {
          "href" : "http://localhost:8080/users/2"
        },
        "solutions" : {
```

```
}Dzana:~ Dzana$ curl http://localhost:8080/users/1
{
  "id" : 1,
  "username" : "Dzana",
  "administratorPrivileges" : null,
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/users/1"
    },
    "registeredUser" : {
      "href" : "http://localhost:8080/users/1"
    },
    "solutions" : {
      "href" : "http://localhost:8080/users/1/solutions"
    },
    "tasks" : {
      "href" : "http://localhost:8080/users/1/tasks"
    }
  }
}
```

```
Dzana:~ Dzana$ curl http://localhost:8080/solutions
{
  "_embedded" : {
    "solutions" : [ {
      "id" : 1,
      "code" : "gvruwvbrb",
      "passing" : 12,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/solutions/1"
        },
        "solution" : {
          "href" : "http://localhost:8080/solutions/1"
        },
        "task" : {
          "href" : "http://localhost:8080/solutions/1/task"
        },
        "user" : {
          "href" : "http://localhost:8080/solutions/1/user"
        }
      }
    }
  ]
},
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/solutions"
    },
    "profile" : {
      "href" : "http://localhost:8080/profile/solutions"
    }
  }
}
Dzana:~ Dzana$ █
```

```
}Dzana:~ Dzana$ curl http://localhost:8080/tests
{
  "_embedded" : {
    "tests" : [ {
      "id" : 1,
      "input" : "11",
      "output" : "22",
      "time_ms" : 12,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/tests/1"
        },
        "test" : {
          "href" : "http://localhost:8080/tests/1"
        },
        "task" : {
          "href" : "http://localhost:8080/tests/1/task"
        }
      }
    }, {
      "id" : 2,
      "input" : "99",
      "output" : "77",
      "time_ms" : 12,
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/tests/2"
        },
        "test" : {
          "href" : "http://localhost:8080/tests/2"
        },
        "task" : {
          "href" : "http://localhost:8080/tests/2/task"
        }
      }
    }
  ]
}
```



```
}Dzana:~ Dzana$ curl http://localhost:8080/tasks
{
  "_embedded" : {
    "tasks" : [ {
      "id" : 1,
      "taskText" : "nekiii zadatak",
      "creatorsSolution" : "lalalalalla",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/tasks/1"
        },
        "task" : {
          "href" : "http://localhost:8080/tasks/1"
        },
        "user" : {
          "href" : "http://localhost:8080/tasks/1/user"
        },
        "tests" : {
          "href" : "http://localhost:8080/tasks/1/tests"
        },
        "solutions" : {
          "href" : "http://localhost:8080/tasks/1/solutions"
        }
      }
    }, {
      "id" : 2,
      "taskText" : "drugi zadatak",
      "creatorsSolution" : "nakfsdbghsrbgj",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/tasks/2"
        },
        "task" : {
          "href" : "http://localhost:8080/tasks/2"
        }
      }
    }
  ]
}
```

```
}Dzana:~ Dzana$ curl http://localhost:8080/tasks/1
{
  "id" : 1,
  "taskText" : "nekiii zadatak",
  "creatorsSolution" : "lalalalalla",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/tasks/1"
    },
    "task" : {
      "href" : "http://localhost:8080/tasks/1"
    },
    "user" : {
      "href" : "http://localhost:8080/tasks/1/user"
    },
    "tests" : {
      "href" : "http://localhost:8080/tasks/1/tests"
    },
    "solutions" : {
      "href" : "http://localhost:8080/tasks/1/solutions"
    }
  }
}
```

HTTP POST metoda

Na slici ispod prikazan je rezultat http POST metode nad tabelom users korištenjem curl-a.

```
}Dzana:~ Dzana$ curl -X POST -H "Content-Type:application/json" http://localhost:8080/users/ -d '{"id":"","username":"irfanPra"}'
{
  "id" : 4,
  "username" : "irfanPra",
  "administratorPrivileges" : null,
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/users/4"
    },
    "registeredUser" : {
      "href" : "http://localhost:8080/users/4"
    },
    "solutions" : {
      "href" : "http://localhost:8080/users/4/solutions"
    },
    "tasks" : {
      "href" : "http://localhost:8080/users/4/tasks"
    }
  }
}
```

HTTP PUT metoda

Na slici ispod prikazan je rezultat http PUT metode nad kreiranim redom sa prethodne slike, korištenjem curl-a.

```
}Dzana:~ Dzana$ curl -X PUT -H "Content-Type:application/json" http://localhost:8080/users/4 -d '{"id":"","username":"irfanPraznaDopunjeno","administratorPrivileges":true}'
{
  "id" : 4,
  "username" : "irfanPrazinaDopunjeno",
  "administratorPrivileges" : true,
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/users/4"
    },
    "registeredUser" : {
      "href" : "http://localhost:8080/users/4"
    },
    "solutions" : {
      "href" : "http://localhost:8080/users/4/solutions"
    },
    "tasks" : {
      "href" : "http://localhost:8080/users/4/tasks"
    }
  }
}
```

HTTP DELETE metoda

Na slici ispod, korištenjem curl-a, prikazan je rezultat http DELETE metode nad user-om sa id-em 3.

```
}Dzana:~ Dzana$ curl -i -X DELETE http://localhost:8080/users/3
HTTP/1.1 204
Date: Wed, 22 Mar 2017 00:10:35 GMT
```

6. Implementacija Rest Controllera

Kako osnovne metode Crud repozitorija često nisu dovoljne za sve funkcionalnosti koje će biti implementirane u nastavku realizacije projektnih zadataka, u ovom dijelu bit će objašnjena implementacija dodatnih metoda Rest servisa unutar Rest kontrolera.

Na sljedećoj slici prikazana je implementacija TaskControllera, odnosno Rest kontrolera sa tri metode.

```
@RestController
@RequestMapping("/task")
public class TaskController {

    @Autowired
    private TaskRepository tr;

    @Autowired
    private TestRepository testr;

    @Autowired
    private SolutionRepository sr;

    @RequestMapping(value="/{id}/tests")
    public List<Test> getTaskTests(@PathVariable("id") long id)
    throws Exception
    {
        Task t=tr.findById(id);

        if(t.getTaskText()==null)
        {
            throw new Exception("Ne postoji taj task");
        }

        List<Test> testovi=testr.getAllTaskTests(id);

        if(testovi.isEmpty())
        {
            throw new Exception("Ne postoje testovi za zadatak");
        }

        return testovi;
    }

    @RequestMapping(value="/{id}/solutions")
    public List<Solution> getTaskSolutions(@PathVariable("id") long
id) throws Exception
    {
```

```

        Task t=tr.findById(id);

        if(t.getTaskText()==null)
        {
            throw new Exception("Ne postoji taj task");
        }

        List<Solution> solutions=sr.getAllTaskSolutions(id);

        if(solutions.isEmpty())
        {
            throw new Exception("Ne postoje rjesenja za zadatak");
        }

        return solutions;
    }

    @RequestMapping(value="/{id}/tenBestSolutions")
    public List<Solution> getTenBestSolutions(@PathVariable("id") long
id) throws Exception
    {
        Task t=tr.findById(id);

        if(t.getTaskText()==null)
        {
            throw new Exception("Ne postoji taj task");
        }

        List<Solution>
solutionsOrdered=sr.findAllTaskSolutionsOrderedByPassing(id);

        if(solutionsOrdered.isEmpty())
        {
            throw new Exception("Ne postoje rjesenja za zadatak");
        }

        if(solutionsOrdered.size()>10)
        {
            return solutionsOrdered.subList(0, 9);
        }

        return solutionsOrdered;
    }
}

```

TaskController

Za deklaraciju samog Rest kontrolera bilo je potrebno anotirati klasu sa `@RestController`. Također, anotacijom `@RequestMapping` podržani su zahtjevi za resursima korištenjem staze `"/task"`. Prva implementirana metoda, uz istu anotaciju omogućava da se korištenjem staze `"/task/{id}/tests"` pristupi svim testovima resursa sa unesenim id-em, koji je, korištenjem anotacije `@PathVariable` moguće poslati kao dio staze. Ova metoda koristi `TaskRepository` i `TestRepository` Crud repozitorije odnosno interfejsa u kojemu su implementirane metode `findById` i `getAllTaskTests`. Implementacija istih repozitorija sa metodama prikazana je na sljedećoj slici.

```
@RepositoryRestResource(path="tasks",collectionResourceRel="tasks")
public interface TaskRepository extends CrudRepository<Task, Long>{
    Task findById(@Param("id") long id);

    //vraca sve taskove koje je postavio user sa datim id-em
    @Query("select t from Task t, RegisteredUser ru where t.user=ru and
ru.id=:id")
    public List<Task> getAllUserTasks(@Param("id") long id);
}
```

TaskRepository

```
@RepositoryRestResource(path="tests",collectionResourceRel="tests")
public interface TestRepository extends CrudRepository<Test, Long>{
    Test findById(@Param("id") long id);

    @Query("select t from Test t, Task tt where t.task=tt and
tt.id=:id")
    public List<Test> getAllTaskTests(@Param("id") long id);
}
```

TestRepository

Druga metoda, slično, omogućava pristup svim `Solution` objektima u vidu liste, odnosno svim postavljenim rješenjima taska određen id-em, korištenjem staze `"/task/{id}/solutions"`. Ova metoda koristi, uz metode `TaskRepository`-a, i metode `SolutionRepository`, čija je implementacija prikazana na sljedećoj slici.

```

@RepositoryRestResource(path="solutions",collectionResourceRel="solution
s")
public interface SolutionRepository extends CrudRepository<Solution,
Long>{

    //vraca sva rjesenja za neki zadatak
    @Query("select s from Solution s, Task t where t.id=:id and
s.task=t")
    public List<Solution> getAllTaskSolutions(@Param("id") long id);

    //vraca rjesenja zadatka poredana po passing
    @Query("select s from Solution s, Task t where t.id=:id and
s.task=t ORDER BY s.passing DESC")
    public List<Solution>
findAllTaskSolutionsOrderedByPassing(@Param("id") long id);

    //vraca sva rjesenja koja je postavio korisnik na razl zadatke
    @Query("select s from Solution s, RegisteredUser ru where ru.id=:id
and s.user=ru")
    public List<Solution> getAllUserSolutions(@Param("id") long id);
}

```

SolutionRepository

Treća metoda implementiranog kontrolera vraća prvih 10 najboljih rješenja po passing polju za Task određen id-em u stazi `"/task/{id}/tenBestSolutions"` kao listu objekata tipa `Solution`, korištenjem metode iz `SolutionRepository`-a.

Uz opisane tri metode kontrolera `TaskController`, implementirana je i jedna metoda u `RegisteredUserController`-u i implementacija iste bit će opisana u nastavku.

```

@RestController
@RequestMapping(value="/user")
public class RegisteredUserController {

    @Autowired
    private RegisteredUserRepository rur;

    @Autowired
    private TaskRepository tr;

    @RequestMapping(value="/{id}/tasks")
    public List<Task> getUsersAddedTasks(@PathVariable("id") long id)
throws Exception
    {

```

```

        RegisteredUser r=rur.findById(id);

        if(r.getUsername()==null)
        {
            throw new Exception("Ne postoji user sa tim id-em");
        }

        List<Task> tasks=tr.getAllUserTasks(id);

        if(tasks.isEmpty())
        {
            throw new Exception("Nema taskova za tog usera");
        }

        return tasks;
    }
}

```

RegisteredUserController

Metoda getUsersAddedTasks, korištenjem sličnih principa kao i prethodno opisane metode, vraća listu zadataka koje je postavio korisnik sa id-em iz staze “/user/{id}/tasks”. Ova metoda koristi jednostavnu metodu RegisteredUserRepository Crud repozitorija, čija je implementacija prikazana na sljedećoj slici.

```

@RepositoryRestResource(path="users",collectionResourceRel="users")
public interface RegisteredUserRepository extends
CrudRepository<RegisteredUser, Long>{

    RegisteredUser findById(@Param("id") long id);

    //vraca usere koji su postavili rjesenje za zadatak
    @Query("select ru from RegisteredUser ru, Task t, Solution s where
s.user=ru and s.task=t and t.id=:id")
    public List<RegisteredUser> getAllUsersSolvedTask(@Param("id")
long id);
}

```

RegisteredUserRepository

7. Konfiguracija centraliziranog servera

U novokreirani Git repozitorij, u svrhe centralizovane konfiguracije za implementirane mikroservise, dodan je file tasks-client.properties čiji je sadržaj dat u nastavku.

```
server.port=8088
spring.jpa.database=POSTGRESQL
spring.datasource.platform=postgres
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.database.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/tasksmodule
spring.datasource.username=postgres
spring.datasource.password={cipher}AgBdiJMbK8a8aHgzkxkL6n+Y4wF/VgmQQxQxe
AgKRtG03vbWR9ZmkPZdRBzsGy3TDSd66/ry80mvD4RT9ndm5oBMIIlvHwmb8u8FbSaf9Gj7/
23eUiB6griErJXILAJtBu807po0S01SegwJNFzs/Sf2lAsX2oglym7k60EK7SYHL0yufRlA
I5tq1lEAdTquJB13ny96dBAXYGeyrGSD8u/dEt3FTW+nN7YXKAUyiqSboAoGgA7GczI2kF9U
MLdxOgx0wJP7uvh//vgMbEALsfbWmZvtHBkY3SGtmZp6+kMDI0q2iPHBQFf987KePigfzxCb
6D21X7vaMe02Q/TI4kykA4ofhmsjrlwbbAtTjAU10JyzoAfntYQs8SZuAGboG650gdIUaiof
ovVMtAT2Gd/kZltdhAi4qWdC90sqypaHJsgIrTAUBNRscqL+DPhQ+/h40lPqL0fmjzjSlWY
FEo6neWBJjJJicS+mUNpjvp0FEUFCIPX0bn4ecZeq7B30wNn4oyYN3vdALE1EBkyUBueli/Q
nkkQKx4AtSUL2AUCp9W9XHU7ttP/259Ynvhl5yFcPqK2CTHw0hofEkGf+bWDw3YRN0cjDTq7
cCj/1AGs9bF8NuLsC/ZfoAVP8zB2MLfpVjs0VN0Yh0eSEwR+Mn4Y78Dqb6tFdQgMIHg0sGla
jOAdwC9LVWSqrfnUbmeh7B+RotL8agDt4AvqQMhQtYq
```

tasks-client.properties

U ovom file-u se nalaze postavke aplikacije TasksModule. Port preko kojega se komunicira sa aplikacijom je 8088.

Također, kako bi aplikacija “znala” da postavke treba da preuzme sa nekog drugog repozitorija, odnosno da očekuje application.properties sa drugog servera, dodan je i file bootstrap.properties čiji je sadržaj dat u nastavku.

```
spring.application.name=tasks-client
spring.cloud.config.uri=http://localhost:8888
spring.cloud.config.username=root
spring.cloud.config.password=s3cr3t
```

bootstrap.properties

8. Postavljanje Eureka klijenta

Uz kreirani Eureka Service registar, koji se pokreće na 8761 port-u, aplikacija TasksModule postavljena je da radi kao Eureka klijent. Prvo što je modificirano u samom kodu aplikacije je otkrivanje aplikacije kao klijenta korištenjem Spring Cloud anotacije `@EnableDiscoveryClient` iznad main metode aplikacije. Main metoda aplikacije TasksModule nakon modifikacije prikazana je u nastavku.

```
@EnableDiscoveryClient
@SpringBootApplication
public class TasksModuleApplication {

    public static void main(String[] args) {
        SpringApplication.run(TasksModuleApplication.class, args);
    }
}
```

Main metoda TasksModule aplikacije

Također, dodan je i Eureka Client Dependency, te Dependency Management za izbjegavanje pisanja verzija dependency-a.

XML dodanog Eureka Client Dependency prikazan je ispod:

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-eureka-server</artifactId>
</dependency>
```

U projekat je dodan i novi Rest Controller: `ServiceInstanceRestController` koji vraća pobrojane sve registrovane instance servisa na `http://localhost:8088/service-instances/tasks-client`. Kod istog dat je u nastavku.

```
@RestController
class ServiceInstanceRestController {

    @Autowired
    private DiscoveryClient discoveryClient;

    @RequestMapping("/service-instances/{applicationName}")
    public List<ServiceInstance> serviceInstancesByApplicationName(
        @PathVariable String applicationName) {
        return this.discoveryClient.getInstances(applicationName);
    }
}
```

ServiceInstanceRestController

Kao test ispravnosti opisane implementacije, potrebno je pozvati se na `http://localhost:8088/service-instances/tasks-client`, te kao rezultat dobiti `ServiceInstance` objekat Eureka klijenta. Uspješni rezultati testa prikazani su, korištenjem `curl`-a, na sljedećoj slici.

```
Dzana:~ Dzana$ curl http://localhost:8088/service-instances/tasks-client
[{"host":"192.168.1.7","port":8088,"uri":"http://192.168.1.7:8088","serviceId":"TASKS-CLIENT","metadata":{}
,"instanceInfo":{"instanceId":"192.168.1.7:tasks-client:8088","app":"TASKS-CLIENT","appGroupName":null,"ipA
ddr":"192.168.1.7","sid":"na","homePageUrl":"http://192.168.1.7:8088/","statusPageUrl":"http://192.168.1.7:
8088/info","healthCheckUrl":"http://192.168.1.7:8088/health","secureHealthCheckUrl":null,"vipAddress":"task
s-client","secureVipAddress":"tasks-client","countryId":1,"dataCenterInfo":{"@class":"com.netflix.appinfo.I
nstanceInfo$DefaultDataCenterInfo","name":"MyOwn"},"hostName":"192.168.1.7","status":"UP","leaseInfo":{"ren
ewalIntervalInSecs":30,"durationInSecs":90,"registrDzana:~Dzana:~Dzana:~ Dzana$Dzana:~Dzana:~Dzana:~Dzana:~
```

9. RestTemplate komunikacija

Za jednostavnu komunikaciju između implementiranih modula korištenjem HTTP zahtjeva korišten je `RestTemplate` (Spring klasa za sinhronu klijent HTTP zahtjeve). Za loadbalancing i pretvaranje imena u url-ove korišten je `Ribbon` u kombinaciji sa Eureka serverom, te je u projekat dodan dependency za `Ribbon`:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-ribbon</artifactId>
</dependency>
```

Kako bi se omogućilo da `RestTemplate` resolve-a imena aplikacija prema kojima se upućuju zahtjevi, dodana je konfiguracijska klasa za `RestTemplate`:

```
@Configuration
public class RestTemplateConfigurationClass {

    @LoadBalanced
    @Bean
    RestTemplate restTemplate(){
        return new RestTemplate();
    }
}
```

`RestTemplateConfigurationClass.java`

Ovim je omogućena komunikacija `TasksModule` modula prema `UsersModule` korištenjem jednostavnih http zahtjeva tipa: `http://<users_client>/<putanja>`. Ovo je iskorišteno u metodi `isLogged` `RegisteredUserController` kontrolera, gdje `TasksModule` dobiva informaciju od `UsersModula` da li korisnik sa određenim username-om logovan na sistem.

```

@RequestMapping("/{username}/isLogged")
    public Boolean isLogged(@PathVariable("username") String username)
    {
        Boolean log=rt.getForObject("http://users-
client/user/logged?username="+username, Boolean.class);
        return log;
    }

```

isLogged metoda RegisteredUserController kontrolera

Ova metoda testirana je na sljedeći način:

```

Dzana:~ Dzana$ curl http://localhost:8088/user/dzana/isLogged
trueDzana:~ Dzana$ █

```

Slično, implementirana je i metoda isAdmin u RegisteredUserController kontroleru, za provjeru da li je korisnik unesenim username-om ima administratorske privilegije.

```

@RequestMapping("/{username}/isAdmin")
    public Boolean isAdmin(@PathVariable("username") String username)
    {
        List<String> roles=rt.getForObject("http://users-
client/user/roles?username="+username, List.class);
        return roles.contains("admin");
    }

```

isAdmin metoda RegisteredUserController kontrolera

```

Dzana:~ Dzana$ curl http://localhost:8088/user/dzana/isAdmin
trueDzana:~ Dzana$ █

```

U TaskController kontroleru dodane su tri metode čija će implementacija i testiranje biti prikazano u nastavku. Implementirane metode:

- addTask za unos novog zadatka, gdje se koristi informacija iz UsersModule da li je korisnik logovan na sistem

```

@RequestMapping("/addTask")
    public void addTask(@RequestBody TaskBody task) throws Exception
    {
        if(task.text==null || task.creatorsSolution==null ||
task.username==null)
        {
            throw new Exception("Polja za unos taska nisu
popunjena");
        }
    }

```

```

        Boolean log=rt.getForObject("http://users-
client/user/logged?username="+task.username,Boolean.class);

        if(!log)
        {
            throw new Exception("Korisnik sa tim username-om nije
logovan.");
        }

        RegisteredUser r=rur.findByUsername(task.username);

        if(r.getUsername()==null)
        {
            throw new Exception("Korisnik ne postoji?!");
        }

        Task novi=new Task();
        novi.setTaskText(task.text);
        novi.setCreatorsSolution(task.creatorsSolution);
        novi.setUser(r);

        tr.save(novi);
    }

    @SuppressWarnings("unused")
    private static class TaskBody{
        public String text;
        public String creatorsSolution;
        public String username;
    }



```

Metoda addTask TaskController kontrolera

```

Dzana:~ Dzana$ curl -X POST --data '{"text": "nekiiii novi task","creatorsSoluti
on":"aaaa","username":"dzana"}' -H "Content-Type:application/json" http://localh
ost:8088/task/addTask
Dzana:~ Dzana$ █

```

id	creators_solution	task_text	user_id
1	lalalalalla	nekiii zadatak	1 
10	aaaa	nekiiii novi task	1 

Testiranje addTask metode

- addTaskTest za unos novog testa za zadatak sa unesenim id-em u path-u, u kojoj je također iskorištena informacija iz UsersModule za provjeru da li je logovani korisnik postavio task za koji se unosi novi test

```
@RequestMapping("/{id}/addTest")
    public void addTaskTest(@PathVariable("id") long id, @RequestBody
    TestBody test) throws Exception
    {
        Task t=tr.findById(id);

        if(t.getTaskText()==null)
        {
            throw new Exception("Ne postoji taj task");
        }

        if(test.input==null || test.output==null ||
test.time_ms==null)
        {
            throw new Exception("Polja za unos testa nisu
popunjena");
        }

        Boolean log=rt.getForObject("http://users-
client/user/logged?username="+t.getUser().getUsername(),Boolean.class);

        if(!log)
        {
            throw new Exception("Korisnik koji je postavio taj
zadatak nije logovan.");
        }
        Test novi=new Test();
        novi.setInput(test.input);
        novi.setOutput(test.output);
        novi.setTime_ms(test.time_ms);
        novi.setTask(t);
        testr.save(novi);

        if(!t.getTests().add(novi))
        {
            testr.delete(novi.getId());
            throw new Exception("Nesto nije okej sa taskom?!");
        }
    }
```

```
@SuppressWarnings("unused")
    private static class TestBody{
        public String input;
        public String output;
        public Integer time_ms;
    }
```

addTaskTest metoda TaskController kontrolera

```
Dzana:~ Dzana$ curl -X POST --data '{"input": "testzatatask1","output":"aaaa","time_ms":12}' -H "Content-Type:application/json" http://localhost:8088/task/1/addTest
Dzana:~ Dzana$
```

id	input	output	time_ms	task_id
1	11	22	12	1 
12	testzatatask1	aaaa	12	1 

Test addTaskTest metode

- deleteTask metoda za brisanje taska sa unesenim id-em



```
@RequestMapping("/{id}/delete")
    public void deleteTask(@PathVariable("id") long id) throws
Exception
    {
        //provjeriti privilegije!!!!

        Task t=tr.findById(id);

        if(t.getTaskText()==null)
        {
            throw new Exception("Ne postoji taj task");
        }

        tr.delete(id);
    }
```

deleteTask metoda TaskController kontrolera

id	creators_solution	task_text	user_id
1	lalalalalla	nekiii zadatak	1 
10	aaaa	nekiiii novi task	1 

```
Dzana:~ Dzana$ curl http://localhost:8088/task/10/delete
Dzana:~ Dzana$
```

id	creators_solution	task_text	user_id
1	lalalalalla	nekiii zadatak	1

Test deleteTask metode

- addTaskSolution metoda kontrolera TaskController putem koje logovani korisnik (što se provjerava na isti način kao i u prethodnim metodama) unosi svoje rješenje za task sa id-em iz path-a

```
@RequestMapping("/{id}/addSolution")
public void addTaskSolution(@PathVariable("id") long id,
@RequestBody SolutionBody sb) throws Exception
{
    Task t=tr.findById(id);

    if(t.getTaskText()==null)
    {
        throw new Exception("Ne postoji taj task");
    }

    Boolean log=rt.getForObject("http://users-
client/user/logged?username="+t.getUser().getUsername(),Boolean.class);

    if(!log)
    {
        throw new Exception("Korisnik sa tim username-om nije
logovan.");
    }

    RegisteredUser
r=rur.findByUsername(t.getUser().getUsername());

    if(r.getUsername()==null)
    {
        throw new Exception("Korisnik ne postoji?!");
    }

    Solution novi=new Solution();
    novi.setCode(sb.code);
    novi.setUser(r);
    //ovo dodati uz compilermodule!!!!!!!!!!
    novi.setPassing(passing);
    sr.save(novi);
}
```

```

        if(!t.getSolutions().add(novi))
        {
            sr.delete(novi.getId());
            throw new Exception("Nesto nije okej sa taskom?!");
        }
    }

    @SuppressWarnings("unused")
    private static class SolutionBody{
        public String code;
        public String username;
    }





```

addTaskSolution metoda TaskController kontrolera

```

Dzana:~ Dzana$ curl -X POST --data '{"code":"kod za task 1","username":"dzana"}'
-H "Content-Type:application/json" http://localhost:8088/task/1/addSolution
Dzana:~ Dzana$ █

```

12	uyuib	1	1 	2 
13	kod za task 1	NULL	NULL 	1 

Test addTaskSolution metode

10. RabbitMQ komunikacija

Za prosljeđivanje poruka tj MessageQueuing između UsersModule i TasksModule korišten je RabbitMQ. Nakon instalacije samog RabbitMQ servera i pokretanja istog korištenjem komande *rabbitmq-server*, da bi TasksModule uspješno primao poruke o izmjenama u UsersModule modulu, dodan je dependency za Spring AMPQ.

```

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>

```

Također, da bi modul mogao ispravno primati poruke iz queue-a, implementirana je i klasa Receiver sa metodom *recvMessage*, koja prima poruku od UsersModula i, ukoliko je dio primljenog stringa *create*, kreira se novi *RegisteredUser* u TasksModule-u sa primljenim parametrima, a ukoliko je *delete*, isti se, ukoliko postoji, briše.


```

@Component
public class Receiver {

    @Autowired
    private UserRepository repository rur;

    @Autowired
    private RestTemplate rt;

    public void receiveMessage(String message) {
        System.out.println("Received <" + message + ">");
        String[] niz=message.split("\\;"); //poruka je Received
        <bbb;lalalla@lala.lala;create>

        if(niz[2].equals("create"))
        {
            RegisteredUser novi=new RegisteredUser();
            novi.setUsername(niz[0]);

            List<String> roles=rt.getForObject("http://users-
            client/user/roles?username="+niz[0],List.class);

            novi.setAdministratorPrivileges(roles.contains("admin"));
            rur.save(novi);
        }

        else if(niz[2].equals("delete"))
        {
            RegisteredUser novi=rur.findByUsername(niz[0]);
            if(novi.getUsername()!=null) rur.delete(novi.getId());
        }
    }
}

```

Receiver.java

Također, dodana je i konfiguracijska klasa za RabbitMQ.

```

@Configuration
public class RabbitmqConfigurationClass {

    final static String queueName="tasksQue";

    @Bean
    Queue queue() {
        return new Queue(queueName, false);
    }

    @Bean
    TopicExchange exchange() {
        return new TopicExchange("users-queue-exchange");
    }

    @Bean
    Binding binding(Queue queue, TopicExchange exchange) {
        return BindingBuilder.bind(queue).to(exchange).with("*.users");
    }

    @Bean
    SimpleMessageListenerContainer container(ConnectionFactory
connectionFactory,
        MessageListenerAdapter listenerAdapter) {
        SimpleMessageListenerContainer container = new
SimpleMessageListenerContainer();
        container.setConnectionFactory(connectionFactory);
        container.setQueueNames(queueName);
        container.setMessageListener(listenerAdapter);
        return container;
    }


    @Bean
    MessageListenerAdapter listenerAdapter(Receiver receiver) {
        return new MessageListenerAdapter(receiver, "receiveMessage");
    }
}


```


RabbitmqConfigurationClass.java

Na sljedećim slikama je prikazano testiranje rada RabbitMQ komunikacije između UsersModule i TasksModule-potrebno je unijeti novog korisnika kroz UsersModule, te bi se ta promjena trebala vidjeti i u TasksModule.

Izgled tabela prije unosa novok korisnika:

 nwt


 nwtusersmodule


 registered_user


Connected.

PostgreSQL 9.6.1

LOCAL







	id	email	password	username	verified
	1	lala	lala	ppp	TRUE
	2	lalalalal	ttttt	dzana	TRUE

tasksmodule	registered_user	Connected.	PostgreSQL 9.6.1	LOCAL
id	administrator_privileges	username	email	password
1	NULL	dzana	NULL	NULL
2	TRUE	Lala	NULL	NULL
8	FALSE	bbb	NULL	NULL

Unos novog korisnika u UsersModule-u:

```
Dzana:~ Dzana$ curl -X POST --data '{"username": "noviuser","password":"neki","email":"nekiemail@lala.lala"}' -H "Content-Type:application/json" http://localhost:8081/user/register
Dzana:~ Dzana$
```

Izgled tabela nakon unosa novog korisnika:

nwtusersmodule	registered_user	Connected.	PostgreSQL 9.6.1	LOCAL
id	email	password	username	verified
1	lala	lala	ppp	TRUE
2	lalalalal	ttttt	dzana	TRUE
8	nekiemail@lala.lala	\$2a\$12\$5AIAIMHELBnsxBirUWBEMuishLzsZ3C6CcFiwtqpgsu..32KCM4OW	noviuser	FALSE

tasksmodule	registered_user	Connected.	PostgreSQL 9.6.1	LOCAL
id	administrator_privileges	username	email	password
1	NULL	dzana	NULL	NULL
2	TRUE	Lala	NULL	NULL
8	FALSE	bbb	NULL	NULL
14	FALSE	noviuser	NULL	NULL