Tema:

FirmInternalApp

-Documents module-

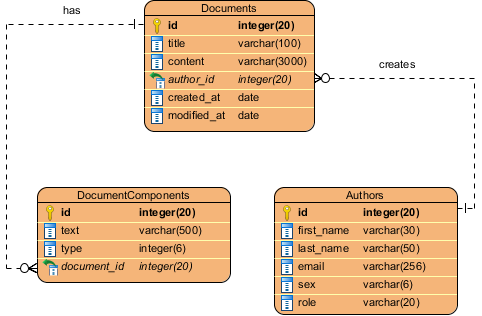
Nino Ćorović

OPIS TEME

Svaka ozbiljnija IT firma posjeduje aplikaciju koja se koristi interno za operacije poput upravljanja korisnicima, dokumentima i zahtjevima. FirmInternalApp je upravo primjer takve aplikacije, koja olakšava svakodnevni rad u firmi, omogućava pregled relevantnih dešavanja i novosti.

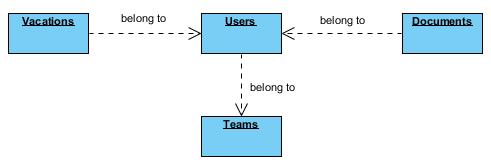
Aplikacija će raspolagati sa 4 modula, te će omogućiti sljedeće funkcionalne zahtjeve: upravljanje obavijestima, upravljanje godišnjim odmorima, upravljanje dokumentima, kao i dvije vrste korisnika i privilegije shodno dodijeljenom tipu. Ukoliko se radi o administratoru, dodatne privilegije bi omogućile brisanje i dodavanje dokumenata. Uposlenik HR tima će se razlikovati po mogućnostima odobravanja ili odbijanja zahtjeva za godišnji odmor. Svi ostali korisnici će imati pristup predaji zahtjeva, pregledu uposlenika po timovima, dodavanju novih timova i sl.

ERD DOCUMENTS MODULA



Slika 1. ERD modula dokumenti

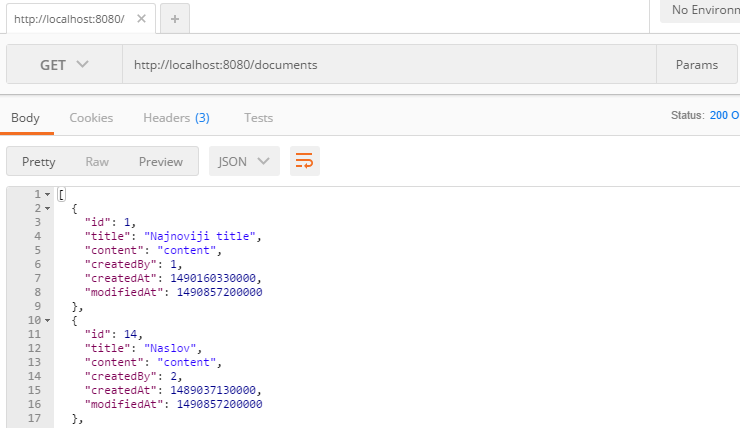
VEZE IZMEĐU MODULA



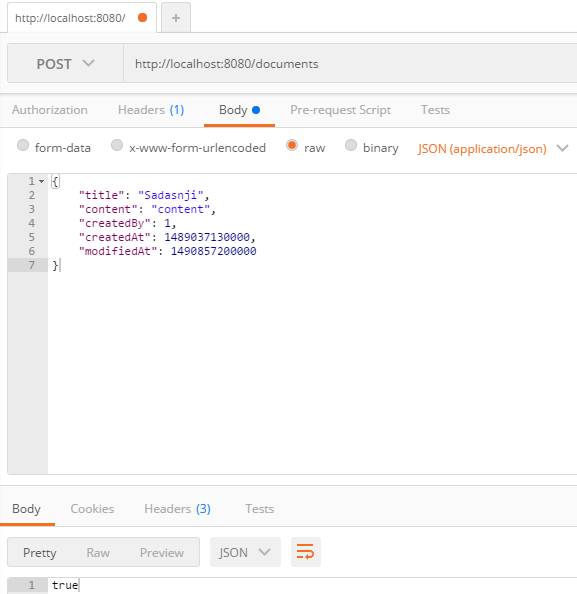
Slika 2. Dijagram komunikacije

POSTMAN REQUESTS

*Postman* HTTP zahtjevi na: <https://localhost:8080/documents>. U vrijeme slanja zahtjeva, modul bio na portu 8080, a sada je na 8084.

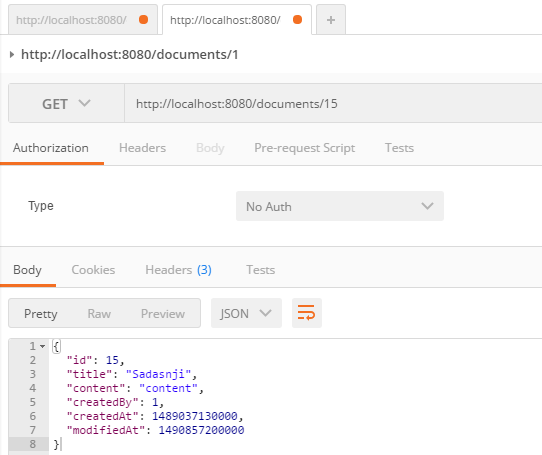


Slika 3. Get request za documents

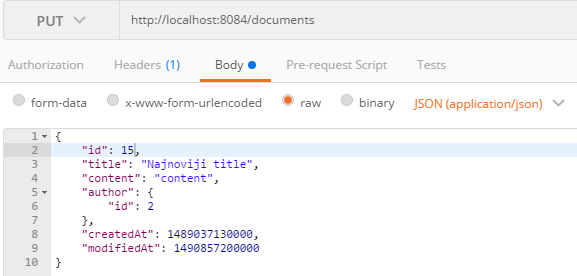




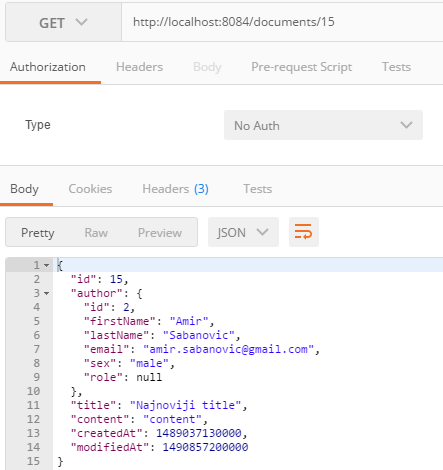
Slika 4. Post request za documents



Slika 5. Get request nakon Post requesta za documents( @ag id -> 15)



Slika 6. Put request za documents



Slika 7. Get request nakon Put requesta za documents id-a 15

Puni spisak svih requesta za postman može se pogledati na sljedećem linku:

<https://app.getpostman.com/run-collection/d89cbbe943f98cf16a5a>

DODATNE CUSTOM METODE ZA JPA REPOSITORY

U slučaju DocumentComponent entiteta omogućen je get zahtjev za komponente dokumenta, prema ID-u dokumenta kojem pripadaju:

@RepositoryRestResource

**public** **interface** DocumentComponentRepository **extends** JpaRepository<DocumentComponent, Long> {

**public** List<DocumentComponent> findByDocumentId(**long** id);

}

U slučaju Document entiteta omogućen je get zahtjev za dokumente, prema ID-u autora koji ih je kreirao:

@RepositoryRestResource

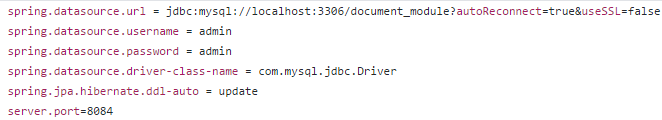
**public** **interface** DocumentRepository **extends** JpaRepository<Document, Long> {

**public** List<Document> findByAuthorId(**long** author\_id);

}

CONFIG

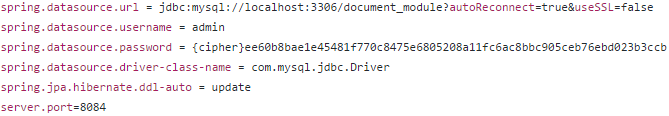
Config server je server koji je odgovoran za centralno čuvanje datoteka koje sadrže informacije potrebne za pristup bazi. Ovakva datoteka za documents modul naziva se “documents-client.properties” I prikazana je u nastavku.



Slika 8. Izgled datoteke “documents-client.properties” prije enkripcije

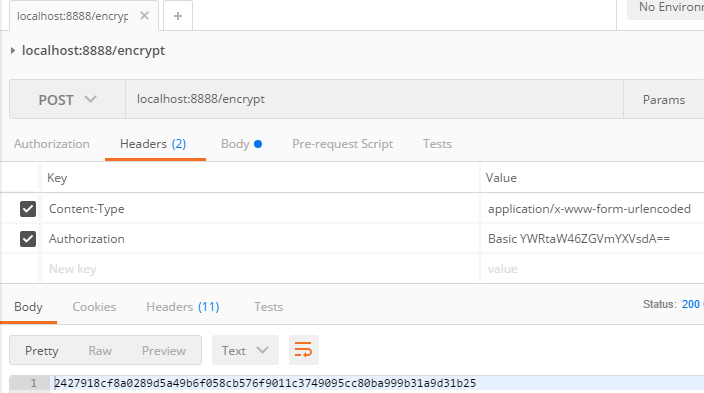
ENKRIPCIJA

Za enkripciju prethodno pomenutih datoteka korištena je simetrična enkripcija.

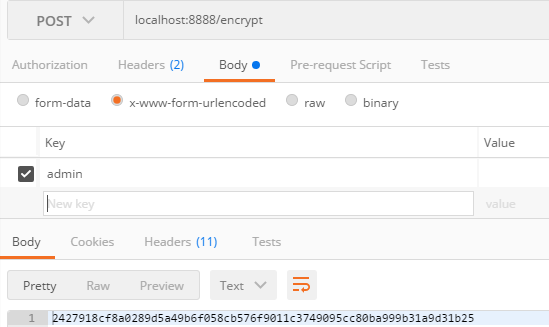


Slika 9. Izgled datoteke „documents-client.properties“ nakon enkripcije passworda

Dakle proces enkripcije passworda uz korištenje Postman-a prikazan je u nastavku:



Slika 10. Postman Post request za enkripciju (Headers)



Slika 11. Postman Post request za enkripciju (Body)

KOMUNIKACIJA TEAMS -> USERS

Komunikacija između pojedinačnih modula (mikroservisa) je ostvarena gdje god u sistemu postoji situacija da u tabeli jednog modula postoji instanca stranog ključa na tabelu koja se nalazi u nekom od preostalih modula. U tom modulu se zbog toga nalazi skraćena ili puna verzija te tabele u koju se onda vrše promjene svaki put kada dođe do promjene u odgovarajućoj tabeli iz drugog modula.

U slučaju TeamsModule i UsersModule, komunikacija koja se javlja je posljedica prisustva stranog ključa u tabeli Users, modula UsersModule, na tabelu Teams koja se nalazi u TeamsModule. Zbog toga u modulu UsersModule postoji tabela Teams koja predstavlja punu, kopiranu verziju tabele timova iz TeamsModule . Tako, svaki put kada se izvrši POST, PUT ili DELETE zahtjev prema tabeli timova TeamsModule, ova promjena će se propagirati i u tabelu timova u UsersModule.

Za implementaciju ove komunikacije koristi se RestTemplate klasa. Unutar klase TeamsService dodaje se privatna metoda restInit() koja vraća novu instancu RestTemplate klase:

@Service

**public** **class** TeamsService {

@Bean

@LoadBalanced

**private** RestTemplate restInit() {

**return** **new** RestTemplate();

}

Za propagaciju promjena uslovljenjim POST zahtjevom prema tabeli timova u TeamsModule, potrebno je da se u metodi addUser() TeamsService servisa pozove metoda postForObject() RestTemplate klase:

**public** **void** addTeam(Team team) {

teamsRepository.save(team);

String usersClient = sirc.getService("users-client");

String url = usersClient + "/teams";

RestTemplate rt = restInit();

rt.postForObject(url, team, Team.**class**);

}

Identičan proces vrši se i za PUT i DELETE zahtjeve (uz napomenu da se radi o put() i delete() metodama RestTemplate klase):

**public** **void** updateTeam(Long id, Team team) {

Team teamToBeUpdated = teamsRepository.findOne(id);

teamToBeUpdated.setHandle(team.getHandle());

teamToBeUpdated.setName(team.getName());

teamToBeUpdated.setInfo(team.getInfo());

teamsRepository.save(teamToBeUpdated);

String usersClient = sirc.getService("users-client");

String url = usersClient + "/teams/{id}";

RestTemplate restTemplate = restInit();

restTemplate.put(url, team, team.getId());

}

**public** **void** deleteUser(Long id) {

teamsRepository.delete(id);

String usersClient = sirc.getService("userss-client");

String url = usersClient + "/teams/{id}";

Map<String, String> params = **new** HashMap<String, String>();

params.put("id", String.*valueOf*(id));

RestTemplate restTemplate = **new** RestTemplate();

restTemplate.delete(url, params);

}