Elektronski fakultet, Niš

**Stručna praksa**

**Izveštaj o izvedenoj praksi**

Teodora Novković 1426

DualSOFT.d.o.o

**Komisija: prof. dr Dragan Stojanović**

**prof. dr Milorad Tošić**

**prof. dr. Ivan Petković**

Niš, 2022

**Sadržaj**

[O kompaniji 2](#_Toc115668511)

[Zadaci stručne prakse 2](#_Toc115668512)

[Izveštaj 4](#_Toc115668513)

[Nedelja 1 – upoznavanje sa okruženjem i strukturom projekta 4](#_Toc115668514)

[Nedelja 2 – Rad na API-u za mobilnu aplikaciju i Ionic 5](#_Toc115668515)

[Nedelja 3-5 – izrada Blockbuster mobilne aplikacije 8](#_Toc115668516)

[Nedelja 6-8 – Kreiranje Blockbuster-auth API-a i back-office 12](#_Toc115668517)

[Nedelja 9-10 – RabbitMQ i Web socket 15](#_Toc115668518)

[Nedelja 11 – Deploy 16](#_Toc115668519)

[Zaključak 16](#_Toc115668520)

# O kompaniji

**DualSOFT** je srpska kompanija koja se bavi razvojem i integracijom softverskih sistema i koja okuplja talentovane i motivisane ljude. Za 20 godina postojanja specijalizovani su za informatičke sisteme sportskih kladionica i ostalih organizatora igara na sreću. U potpunosti su usmereni na to da klijentima pomognu u razvoju rešenja koja su jedinstvena na tržištu, pomoću znanja i iskustva u korišćenju savremenih tehnologija.

# Zadaci stručne prakse

Biće opisana osnovna struktura projekta i tehnologije koje su korišćene u izradi istog.

Praksa je održana u firmi Dualsoft u Nišu, i to u periodu od 01.04. do 01.07.

U toku prakse, rađena je web aplikacija “*Blockbuster”*. Delovi Blockbuster sistema su:

* Blockbuster-mobile - web aplikacija za iznajmljivanje filmova
* Blockbuster-back-office – web aplikacija za administraciju i interakciju s korisnicima
* Blockbuster – web api za mobilnu aplikaciju
* Blockbuster-auth – web api za back office aplikaciju
* Blockbuster-db – servis za interakciju sa bazom i mapiranje modela
* Blockbuster-model – implementacija servisa za web api

Kompletan kod aplikacije se nalazi na sledecem linku:

### https://gitlab.com/zlatkovnik/blockbuster

Aplikacija je rađena u timu od 4 člana.

### Koriscene su sledece tehnologije i alati:

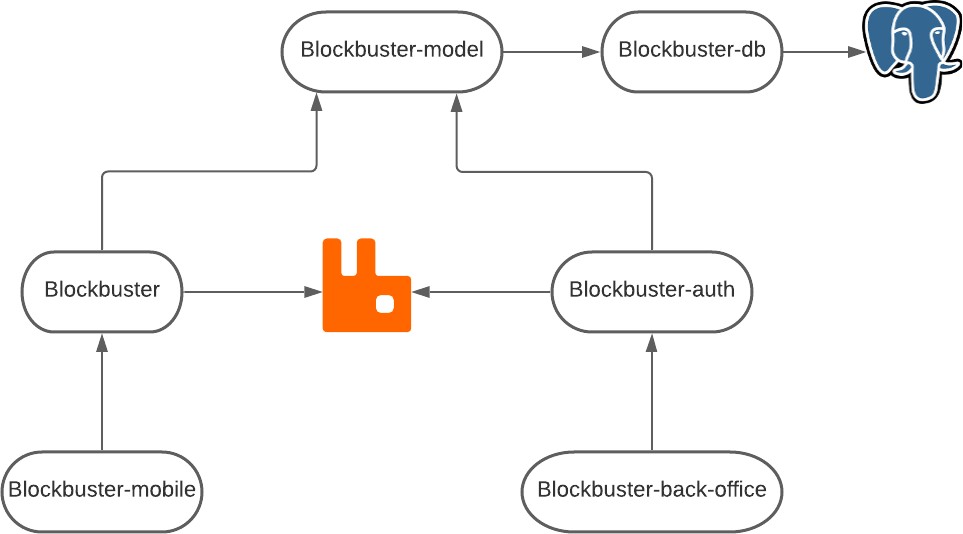
* Java Spring framework
* TypeScript Angular framework
* Ionic
* jOOQ
* PostgreSQL
* RabbitMQ
* Web socket

# Izveštaj

## Nedelja 1 – upoznavanje sa okruženjem i strukturom projekta

Prve nedelje smo se upoznali sa okruženjem i instalirali sve potrebne alate za izradu projekta. Zatim smo počeli sa učenjem tehnologija koje ćemo koristiti, na početku je fokus bio na PostgresSQL, Java String i jOOQ.

Nakon toga smo se upoznali sa strukturom projekta koja je prikazana na dijagramu sa slike. Na osnovu projektovanog modela, kreirali smo semu koriscenjem PostgreSQL baze podataka sa svim potrebnim tabelama za rad aplikacije. Neke od tabela su : CUSTOMER, FILM, RENTAL... Uneli smo inicijalne podatke i testirali upite kako bismo se upoznali sa dijalektom Postgres-a



Tokom izrade projekta koristili smo Maven za buildovanje i za regulisanje dependency-ja, pa smo deo vremena utrošili na upoznavanje sa ovim alatom. Nakon instaliranja Maven-a, kreirali smo parent projekat koji je bio zaduzen za regulisanje verzija svih biblioteka koje su koriscene u projektu, a onda započeli sa kreiranjem strukture Blockbuster web aplikacije.

## Nedelja 2 – Rad na API-u za mobilnu aplikaciju i Ionic

Druge nedelje smo poceli sa kreiranjem web API-a, koji treba da pokrije sve osnovne funkcionalnosti aplikacije. Korisnicima se treba omoguciti prikaz najnovijih, najbolje ocenjenih i preporucenih filmova, pregled svih filmova sa filtriranjima po nazivu i kategorijama i sortiranjima, detaljni pregled filma, pregled kategorija, pregled glumaca sa filtriranjem i sortiranjem, detaljni pregled glumaca, pregled prodavnica, registracija i logovanje, a logovani korisnici imaju mogucnost da upravljaju svojim nalogom, uplaćuju kredit kojim vrše plaćanje, kao i samo iznajmljivanje filmova.

Ova aplikacija je rađena koriscenjem Java Spring framework-a. Implementirali smo CRUD operacije za filmove, iznajmljivanja, glumce, prodavnice i transakcije. Za kreiranje upita iz Jave, koristili smo JOOQ biblioteku. JOOQ je objektno-relacioni maper koji implementira Active Record obrazac. Koristi Database-first pristup.

Primer JOOQ upita koji vraća sve filmove:

List<Film> films = context

.selectFrom(FILM)

.fetchInto(Film.class);

Najbolji i najnoviji se vrlo lako nalaze, tako da nema potrebe da dalje komentarisem. Medjutim preporuceni i najtrazeniji su malo zanimljiviji.

Najtrazenije filmove smo nalazili na sledeci nacin:

public List<FilmForSlider> getTopRented(int count) { List<FilmForSlider> films = context

.select(FILM.FILM\_ID, FILM.TITLE, FILM.RELEASE\_YEAR, FILM.RATING, FILM.DESCRIPTION, FILM.STAR\_RATING, FILM.POSTER\_URL)

.select(count(RENTAL.RENTAL\_ID))

.from(FILM)

.join(INVENTORY)

.on(FILM.FILM\_ID.eq(INVENTORY.FILM\_ID

.cast(Integer.class)))

.join(RENTAL)

.on(INVENTORY.INVENTORY\_ID.eq(RENTAL.INVENTORY\_ID))

.groupBy(FILM.FILM\_ID)

.orderBy(count(RENTAL.RENTAL\_ID).desc())

.limit(count)

.fetchInto(FilmForSlider.class); return films;

}

U sustini to su filmovi koji imaju najvise rental kolona, tj oni koji su najvise iznajmljivani. Preporuceni se nalaze na sledeci nacin:

List<CategoryCount> categories = context

.select(CATEGORY.NAME, count().as("rentCount"))

.from(CATEGORY)

.join(FILM\_CATEGORY)

.on(FILM\_CATEGORY.CATEGORY\_ID.cast(Integer.class)

.eq(CATEGORY.CATEGORY\_ID))

.join(INVENTORY)

.on(FILM\_CATEGORY.FILM\_ID.eq(INVENTORY.FILM\_ID))

.join(RENTAL)

.on(INVENTORY.INVENTORY\_ID

.eq(RENTAL.INVENTORY\_ID.cast(Integer.class)))

.where(RENTAL.CUSTOMER\_ID.cast(Integer.class).eq(customerId))

.groupBy(CATEGORY.NAME)

.orderBy(count().desc())

.limit(3)

.fetchInto(CategoryCount.class);

List<FilmForSlider> films = context

.select(FILM.FILM\_ID, FILM.TITLE, FILM.RELEASE\_YEAR, FILM.RATING, FILM.DESCRIPTION, FILM.STAR\_RATING, FILM.POSTER\_URL)

.from(FILM)

.join(FILM\_CATEGORY)

.on(FILM\_CATEGORY.FILM\_ID.cast(Integer.class)

.eq(FILM.FILM\_ID))

.join(CATEGORY)

.on(CATEGORY.CATEGORY\_ID

.eq(FILM\_CATEGORY.CATEGORY\_ID.cast(Integer.class)))

.where(CATEGORY.NAME.like(categories.get(0).getName()))

.or(CATEGORY.NAME.like(categories.get(1).getName()))

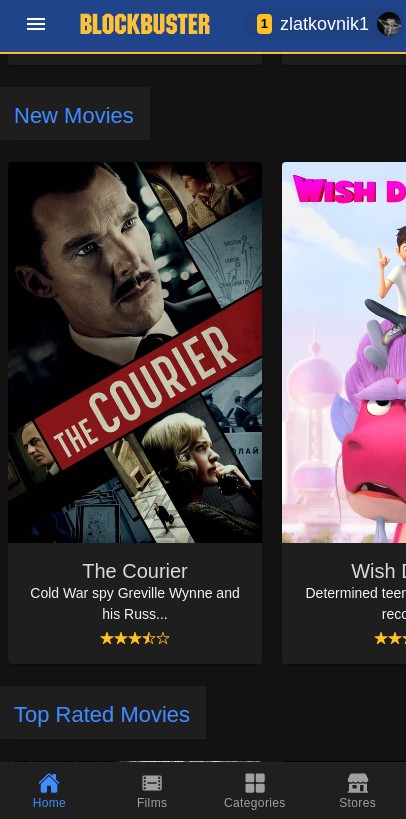
.or(CATEGORY.NAME.like(categories.get(2).getName()))

.limit(200)

.fetchInto(FilmForSlider.class);

Uzimaju se najiznajmljivaniji filmovi od korisnika i izvlace se njihove kategorije sortirane po broju filmova. Uzimaju se top tri kategorije i vracaju se filmovi sa tim kategorijama. Nakon toga se skaliraju filmovi koji se prikazuju korisniku po tome koja kategorija je najtrazenija.

## Nedelja 3-5 – izrada Blockbuster mobilne aplikacije

Ova front end aplikacija je rađena u Ionic framework-u.

### Home

Pregled najboljih, preporucenih, najtrazenijih i najnovijih filmova.

### Pretraga filmova

Implementirali smo sledeće funkcionalnosti: pregled filmova, sortiranje, pretraga, dinamicko ucitavanja, odabir i filtriranje po favoritima itd. Na pocetku se ucita neki broj filmova, a kada korisnik skroluje do kraja liste konkatenira se sledeci broj filmova po datom kriterijumu.

### Odabir omiljene prodavnice

Implementirali smo stranicu za odabir omiljene prodavnice, na toj stranici imamo integraciju open street maps resenja, za graficki prikaz prodavnica i korisnikove lokacije. Naravno kada korisnik odobri prikaz svoje lokacije. Tu se prikazuje lista svih prodavnica koje postoje i lista obliznjih prodavnica, gde se distanca racuna koristeci pitagorinu teoremu. Ako je distanca manja od zadate (50km) prodavnica ispunjava uslov. Tu je moguce prikazati datu prodavnicu na mapi i odabir prodavnice kao omiljene. Sto kasnije olaksava iznajmljivanje filma zato sto je u listi prodavnica uvek prva njegova odabrana.

### Stranica za detalje o filmu

Implementirali smo stranicu koja prikazuje detalje o odabranom filmu. Tu se nalazi naziv filma, godina premijere, rejting, duzina filma, zanr, jezik, dugme za dodavanje u listu favorita, opis filma, lista glumaca sa linkom do detalja o glumcu i koliko je dvd-eva ima u odabranoj prodavnici. Takodje pise sa koliko sredstava raspolazemo i koliko kosta da se iznajmi dvd. Klikom na rent otvara se forma za odabir prodavnice, nakon odabira prodavnice korisnik mora da potvrdi zahtev. Korisniku ce biti onemoguceno da ponovo posalje zahtev za isti film.

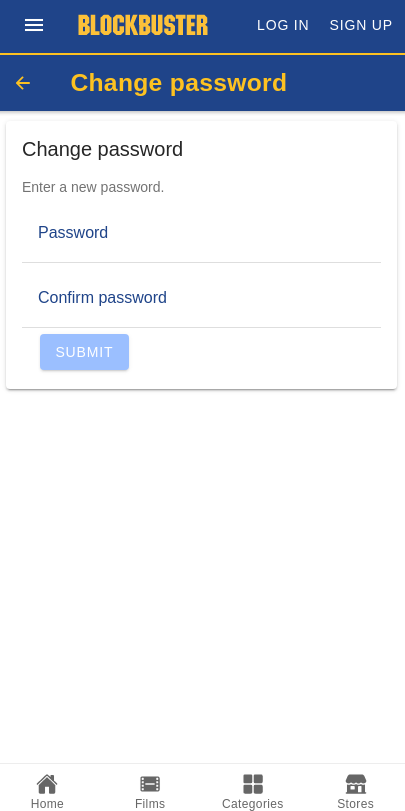
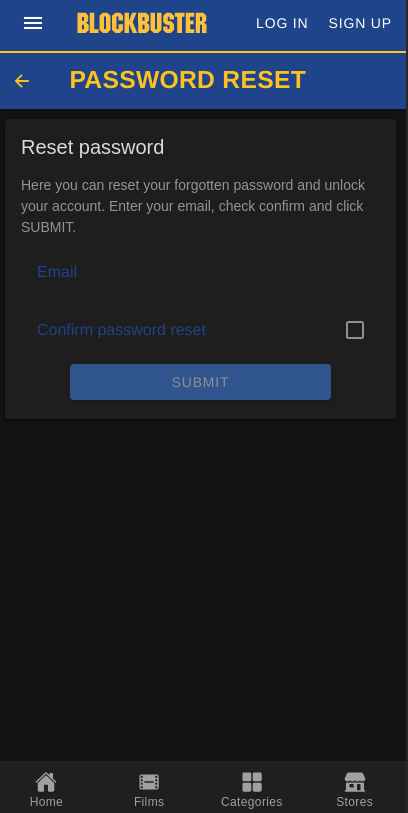
### Stranica za pregled zahteva za rentu

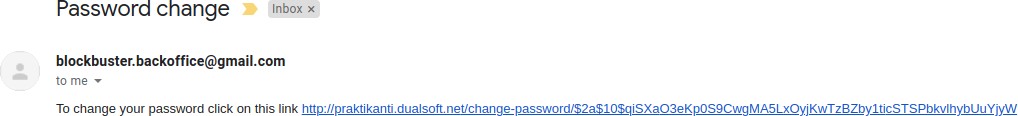
Korisnik ima uvid u svoje zahteve za rentu. Svi poslati a neprihvaceni zahtevi mogu biti obustavljeni, a svaki zahtev ne mora biti odobren. Naime back office moze da odluci da odbije zahtev iz bilo kog razloga. Na ovoj strani korisnik moze da filtrira zahteve po tome da li su nereseni, prihvaceni, odbijeni ili obustavljeni.

### Stranica za pregled iznajmljivanja

Stranica sa listom svih odobrenih iznajmljivanja, filterom za vreme i statusom, odnosno da li su vraceni ili ne. Iznajmljivanja sadrze sledece informacije: naslov filma, poster, kada je iznajmljen, kada je vracen, i ako jeste i u kojoj prodavnici.

### Sistem za izmenu lozinke

U slucaju da korisnik zaboravi sifru ima opciju da resetuje sifru tako sto se salje link za resetovanje sifre na email adresu. Nakon toga korisnik treba da se navigira na stranicu za resetovanje lozinke sa datim kodom za reset. Nakon toga unosom nove sifre menja se stara lozinka.

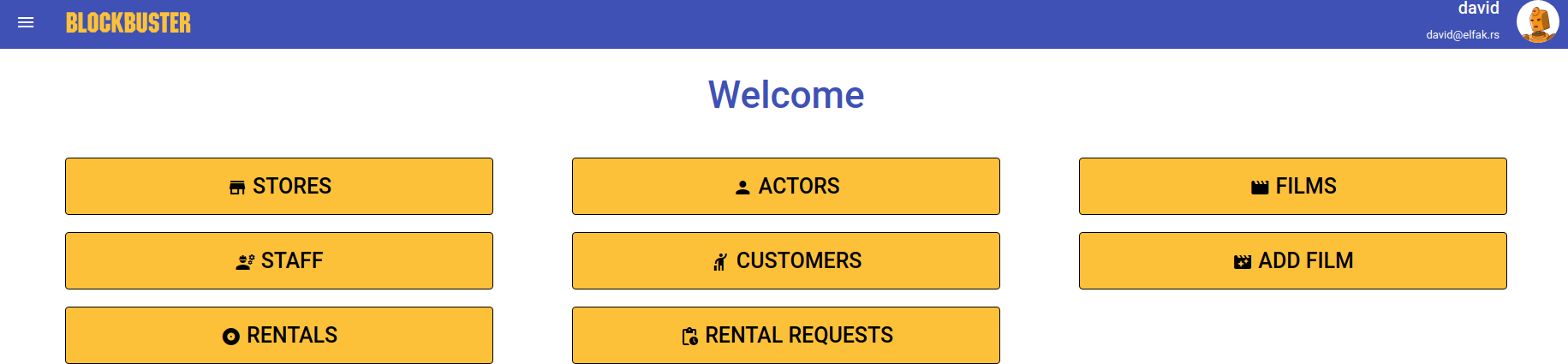


## Nedelja 6-8 – Kreiranje Blockbuster-auth API-a i back-office

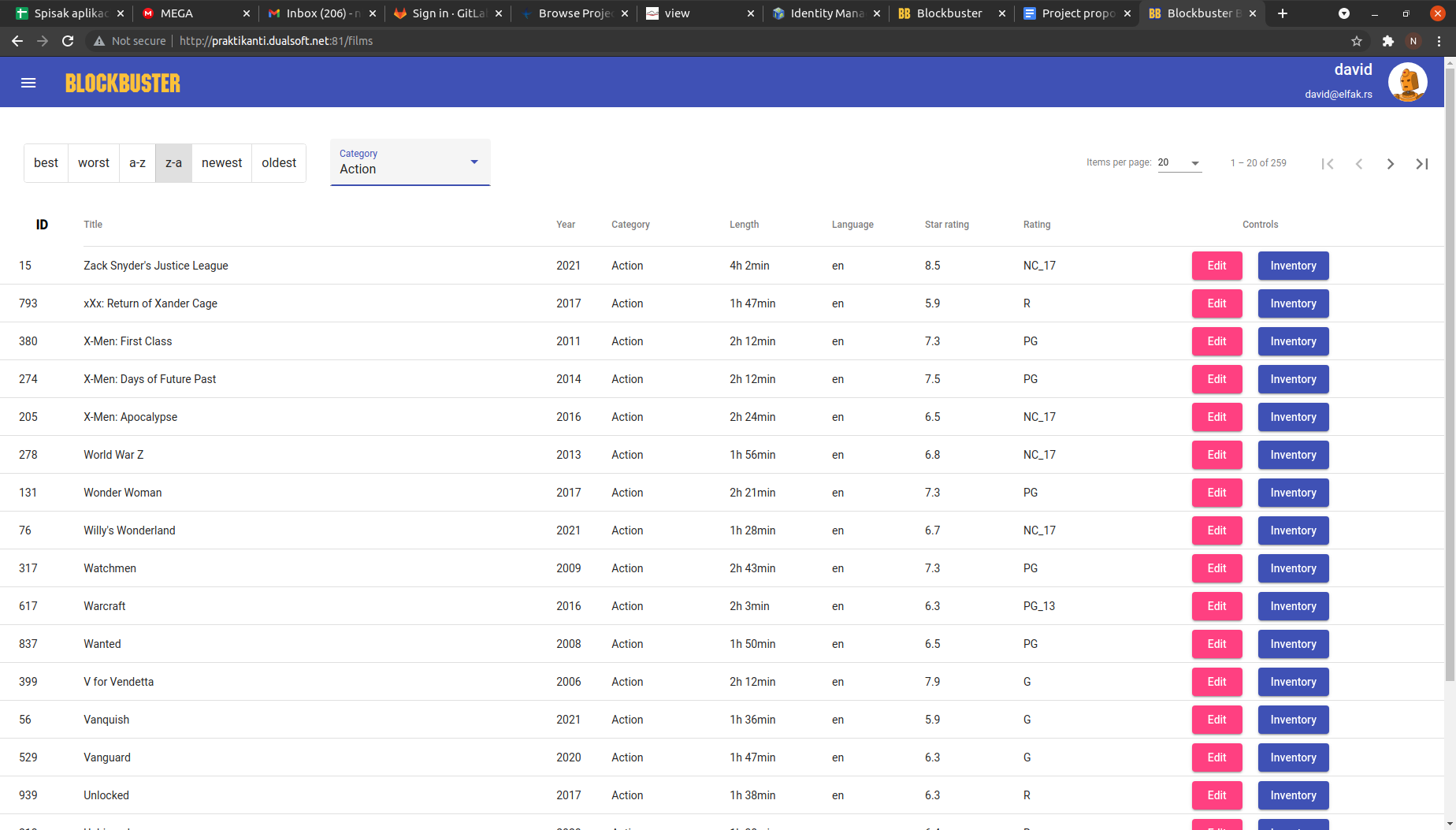
API će biti korišćen od strane back office aplikacije, koja je namenjena radnicima u prodavnicama i omogucava im celokupan menadzment baze i obradu zahteva za iznajmljivanje.

I ova aplikacija je radjena u Java Spring framework-u. Implementirali smo CRUD operacije za sve tabele. Za kreiranje upita iz Jave, koristili smo JOOQ biblioteku. JOOQ je objektno-relacioni maper koji implementira Active Record obrazac. Koristi Database-first pristup.

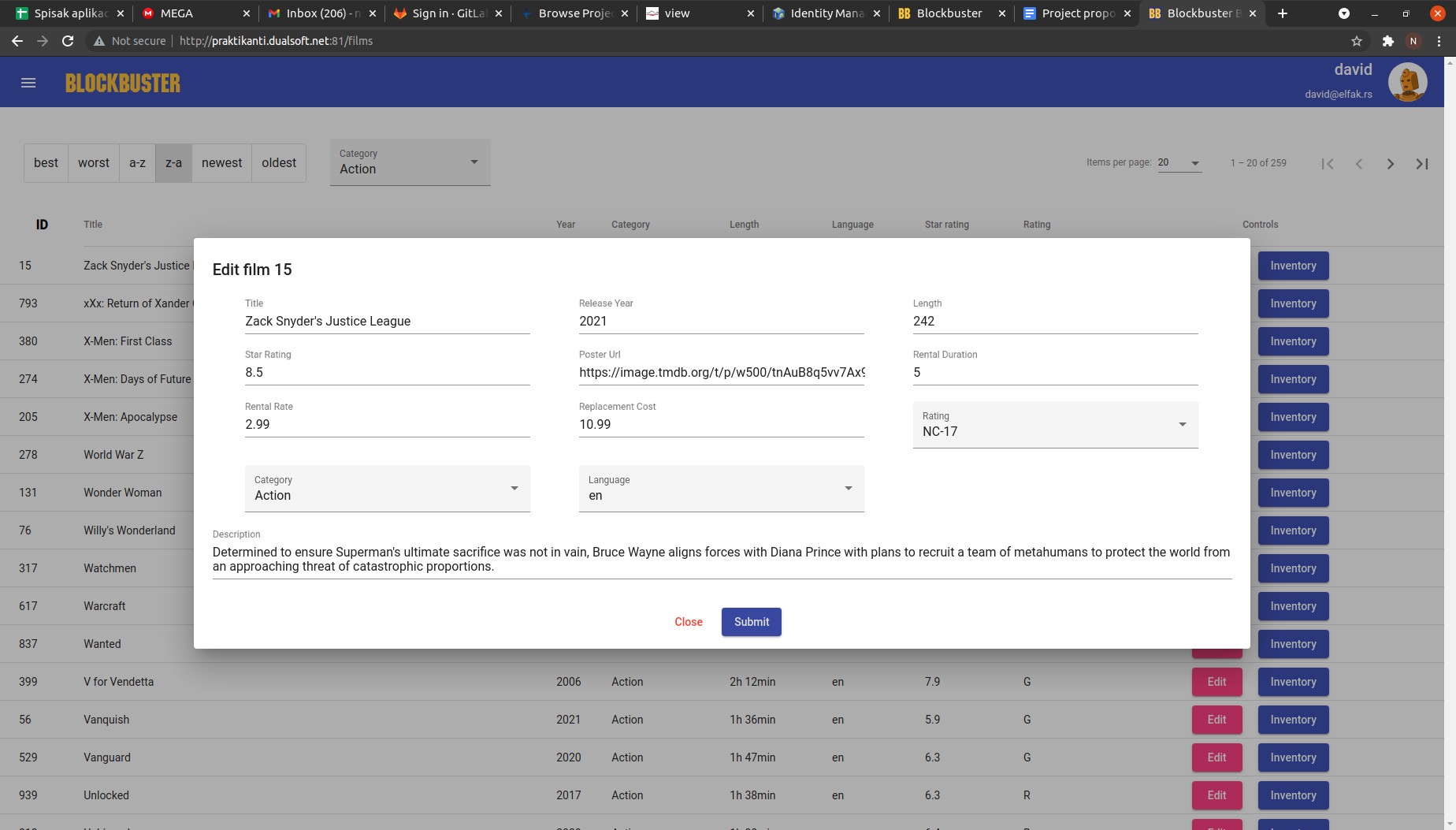
Na front end-u koristimo Angular, i aplikacija je namenjena iskljucivo za desktop i ima jednostavan dizajn i jednostavna je za korišćenje. Na sledećoj slici je osnovni meni aplikacije



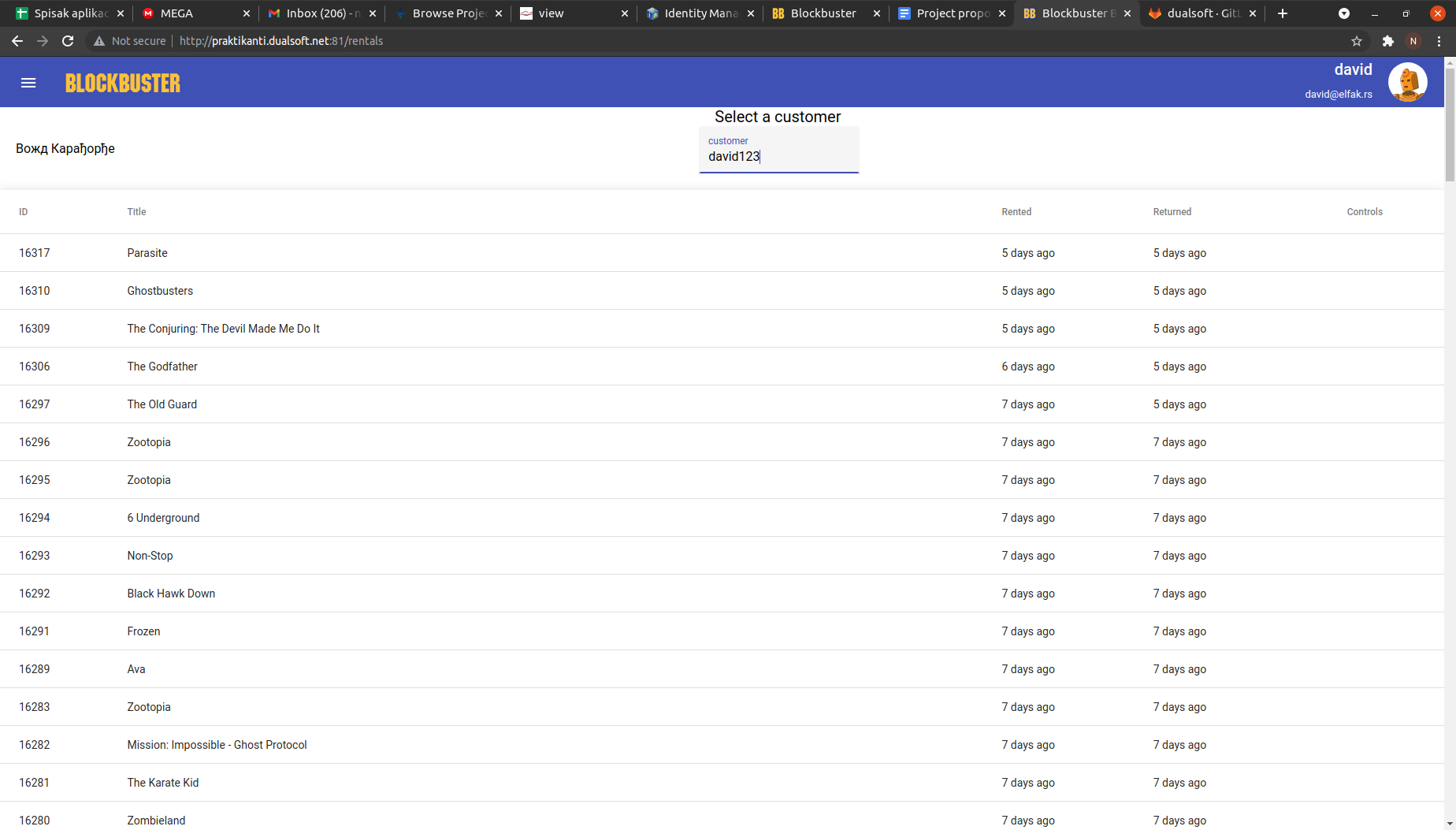
Kada se selektuje neka kartica dobija se tabelarni prikaz sa sortiranjem i filtriranjem:



Izgled kartice za edit filma:

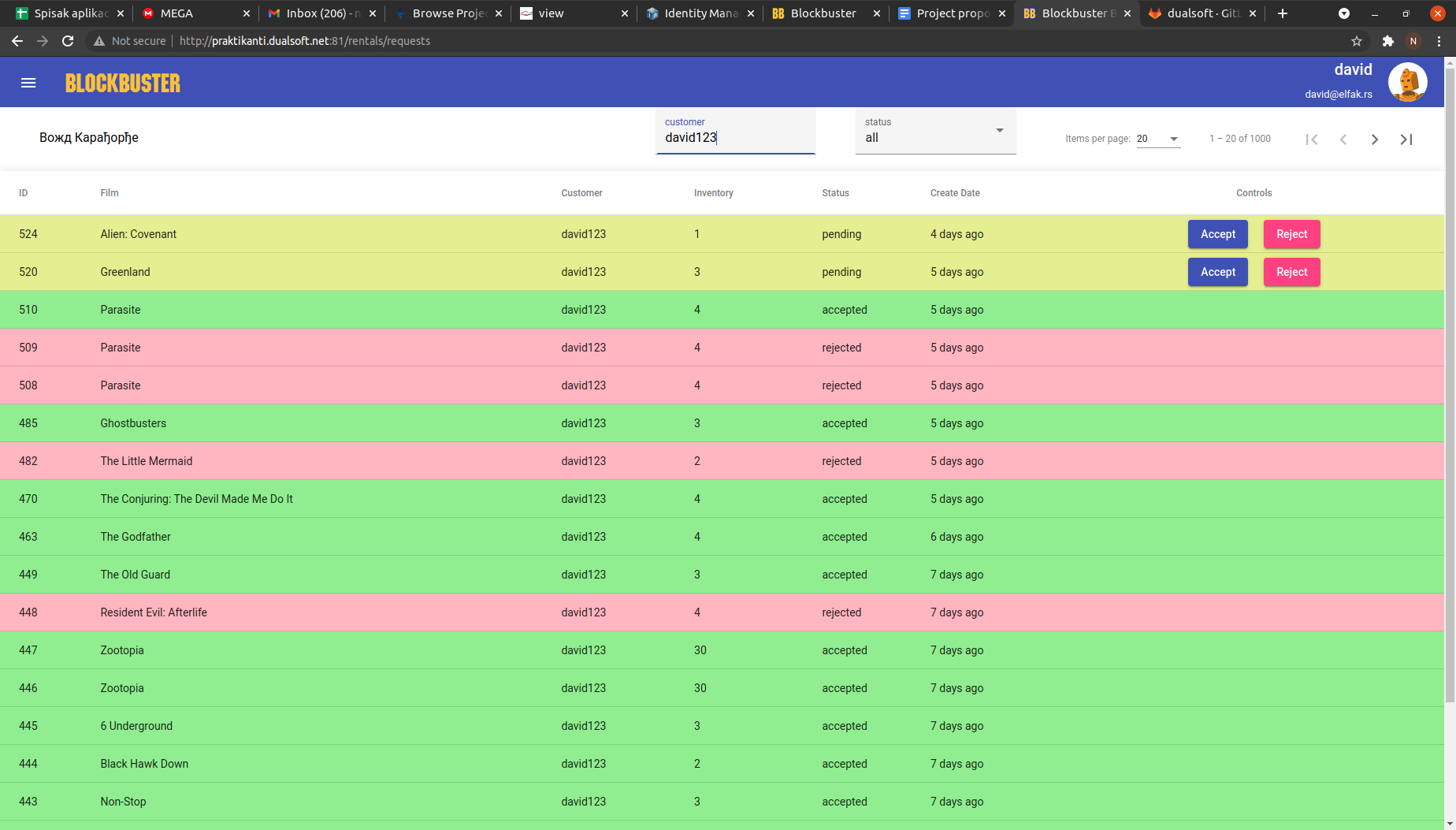


Stranica sa istorijom iznajmljivanja i opcijom da se vrati dvd u inventar, ovo koristi back office radnik koji prima kopiju od korisnika i zabelezi da je vraćena:



## Nedelja 9-10 – RabbitMQ i Web socket

Mobilni klijent i radnik u prodavnici koji koristi back office moraju biti u real time komunikaciji da bi ceo sistem zaživeo na pravi način i bez eventualnih nepoklapanja u podacima. Notifikacije koje se razmenjuju su vezane za tok iznajmljivanja filma. Prvo zainteresovani klijent zahteva neki film, obaveste se svi radnici iz prodavnice i odgovaraju na zahtev. Klijent moze da otkaze zahtev te je real time od krucijalne vaznosti, takođe moguć je slučaj da dva radnika istovremeno resavaju isti zahtev.



Za notifikacije koristimo web socket, na kom otvaramo sesiju sa svakim korisnikom ili prodavnicom i saljemo realtime notifikacije bez obzira na kom su fizickom serveru. Korisnika obavestavamo o tome sta se desilo sa njegovim zahtevom i njegovi zahtevi se azuriraju u pravom vremenu. Back office-u u prodavnici u kojoj klijent salje zahtev stize notifikacija o zahtevu i lista se azurira.

Medjuserverska komunikacija je implementirana rabbitMQ brokerom. Oba servera otvaraju po jedan exchange na koji se povezuju queue-ovi. Moguca je replikacija servera zarad boljih performansi, jer se za svaki server otvara poseban queue sa jedinstvenim identifikatorom, koji sluzi za razmenu poruka.

## Nedelja 11 – Deploy

Projekat smo podigli na CentOS server sa pripremljenom osnovnom konfiguracijom. Naš zadatak je bio da na Tomcat postavimo i pokrenemo naše servere, koristeći Nginx za rutiranje, dok su klijentske aplikacije podignute direktno na računaru, takođe uz pomoć Nginx-a.

Suočavali smo se sa raznim problemima tokom deploymenta, kao što su pakovanje Spring boot aplikacije u war fajl. Povezivanje na RabbitMQ broker koji je serviran na remote računaru. Podešavanje Firewalla da ne blokira naše zahteve. Podešavanje CORS-a za novu konfiguraciju. Izmenu SockJS konfiguracije da omogući realtime komunikaciju. Http limitirane funkcionalnosti aplikacije itd.

# Zaključak

Za kratak vremeniski period uspeli smo da zajedno praktično savladamo i naučimo dosta toga. Upoznali smo se se pravilnim arhitekturom projekta i sa JOOQ-om kao objektno relacionim maperom koji nam je omogucio da jednostavno od tabela napravimo gotove klase. Bavili smo se Angularom pre svega Ionic-om. Takođe, uspeli smo da podignemo projekat na server uz pomoc kolega u firmi.