Arhitekturni projekat

EasyFlow

Članovi tima:

Đorđe Pavlović
David Nikolić

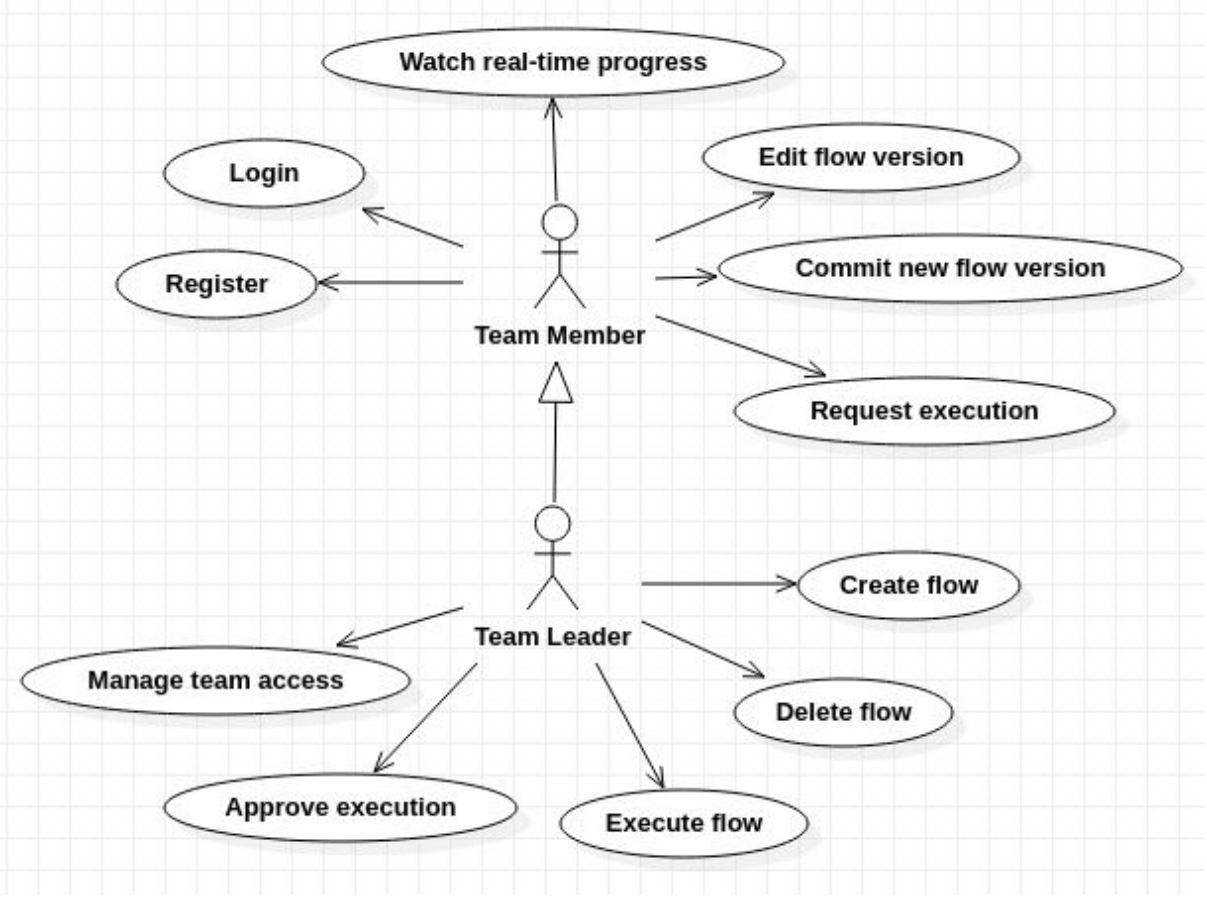
Kontekst i cilj projekta

Easyflow je framework namenjen za izradu pipeline-ova za Machine Learning projekte sa integrisanim Version Control sistemom. Korisnik skuplja tim oko sebe sa kojim vizuelno pravi tok algoritma i skladišti njegove različite verzije. Članovi tima mogu da predlažu promene i posmatraju izvršenje algoritma u realnom vremenu, a lider tima odobrava promene i pokreće izvršenje.

Arhitekturni zahtevi

Glavni funkcionalni zahtevi

- Kreiranje korisničkog naloga
- Logovanje sa postojećim nalogom
- Izmena toka algoritma
- Commit-ovanje nove verzije algoritma
- Zahtev za izvršenjem toka algoritma
- Praćenje izvrsenja toka u realnom vremenu
- Kreiranje tima (korisnik postaje Team Leader)
- Dodavanje novih članova tima (Team Leader)
- Kreiranje toka algoritma (Team Leader)
- Brisanje toka algoritma (Team Leader)
- Izvršenja toka algoritma (Team Leader)
- Odobravanje izvršenja toka algoritma (Team Leader)



Ne-funkcionalni zahtevi (atributi kvaliteta)

- Performanse - obaveštenja u realnom vremenu
- Pouzdanost - aplikacija neće izgubiti podatke
- Sigurnost - autentifikacija i autorizacija
- Jednostavnost - jednostavnost korišćenja i dobar korisnički doživljaj
- Skalabilnost - aplikacija treba da podrži veliki broj korisnika
- Modifikabilnost - promene u sistemu ne zahtevaju preteran napor
- Dostupnost - potrebno je da je aplikacija bude dostupna 24/7

Tehnička i poslovna ograničenja

- Ostvarivanje sinhrone i asinhrone komunikacije između klijenta i servera
- Sistem će biti dostupan 7 dana nedeljno, 24 sata dnevno
- Potrebno je da šema baze podataka bude skrivena
- Potrebno je omogućiti trenutnu notifikaciju o događajim

Arhitekturni dizajn

Arhitekturni obrasci

Layered

Sistem će se sastojati iz tri sloja. Prvi sloj je klijentska aplikacija. Drugi sloj je centralni server koji se sastoji od Web API-ja koji služi kao broker i koordinira sve korisničke zahteve prema više distribuiranih servera (serverom podataka i komputacionim serverima) i message brokera koji upravlja asinhronom komunikacijom između komputacionih servera i korisnika. Treći sloj je centralizovana baza podataka.

Repository

Sistem sadrži centralnu bazu podataka kojoj pristupaju nezavisne komponente.

Broker

Web API igra ulogu brokera i koordinira sve korisničke zahteve prema više distribuiranih servera (serverom podataka i komputacionim serverima). Ovo rešenje je jednostavno za dodavanje novih servera u aplikaciju. Jedino odstupanje je redis server koji je stavljen kao posebna komponenta iz razloga što će se korisnici direktno pretplaćivati na message broker i dobijati od komputacionih servera poruke za koje su pretplaćeni.

Pipe and Filter

Flow Manager upravlja podacima preko pipe and filter principa. Korisnici sami prave svoje tokove, tako sto redjaju filtere (operatore) na nacin koji im pogoduje. Pipe and filter nalaze da operatori budu nezavisni sto je u ovom slucaju najbitnija stavka.

Publish-Subscibe

Message broker implementira publish-subscribe arhitekturni obrazac. Klijenti ulaze u timove i subscribe-uju se na tokove tog tima. Kada se određeni tok izvršava i dođe do nekih promena korisnici primaju obaveštenja o tome.

MVC

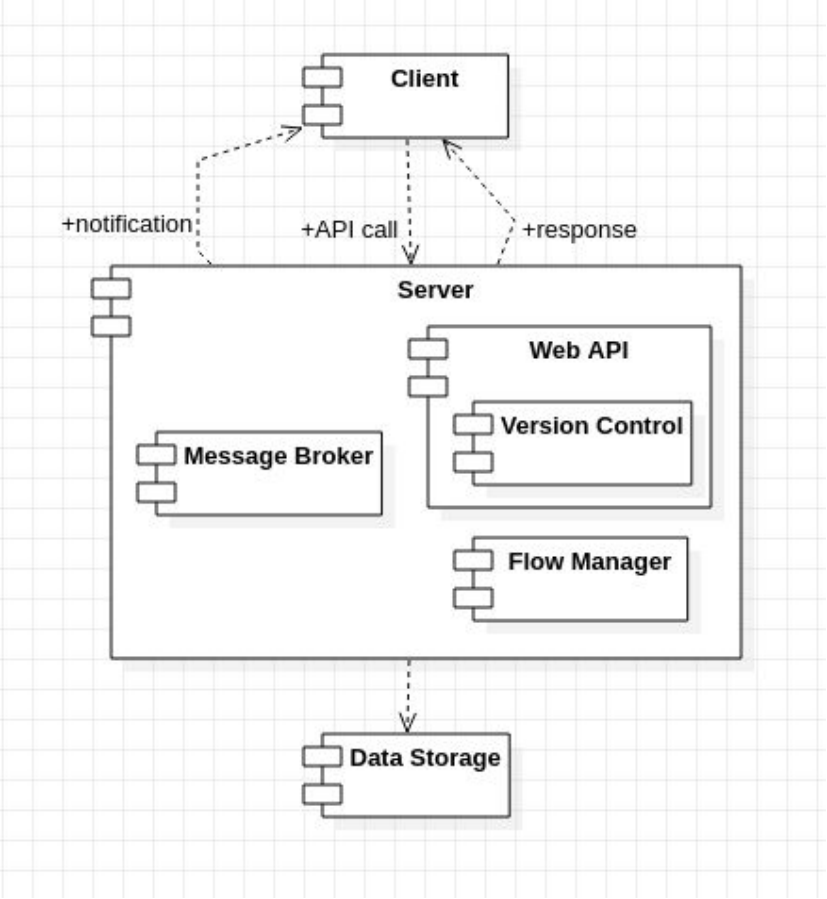
Django se zasniva na MVT (Model-View-Template) obrascu, koji ima sledeću analogiju sa MVC-om:

- Model mapira klase u bazu podataka (odgovara modelu iz MVC obrasca)
- View je zadužen za svu logiku, interakciom sa modelom, prima i odgovara na http zahteve i implementira endpoint-ove ka klijentu (odgovara Controller-u iz MVC obrasca)

- Template je zadužen za prezentacioni sloj, predstavlja osnovni html kod koji renderuje podatke

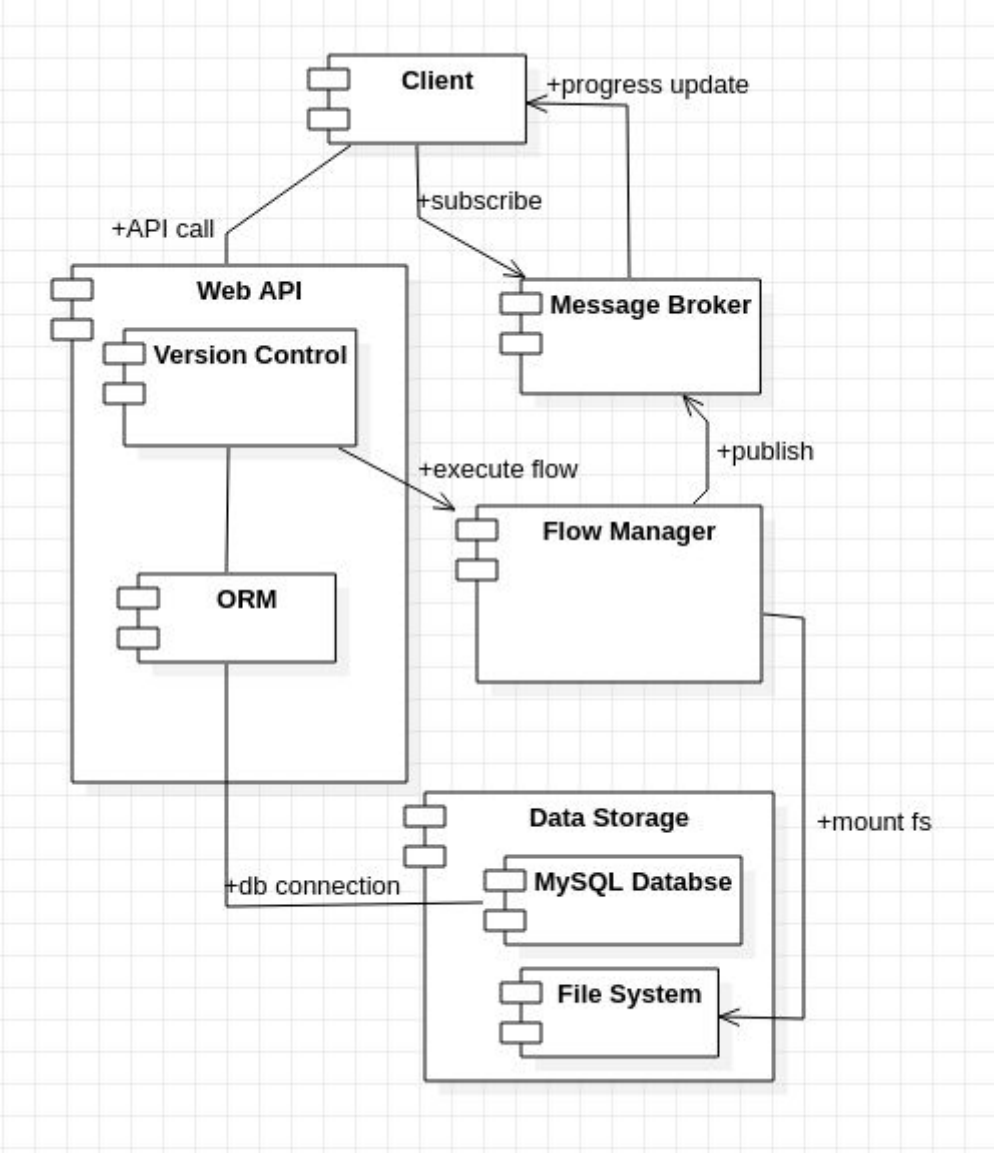
U našem sistemu za backend će se koristiti Django framework, tačnije jedan njegov deo Django REST Framework (DRF). On će implementirati Model i Controller sloj iz MVC arhitekture svojim Model i View slojevima, respektivno, dok se Template sloj eće koristiti. Na forntend-u će se implementirati View sloj iz MVC-a korišćenjem React.js frameworka, koji će upravljati prikazom podataka koje dobija od backenda.

Generalna arhitektura (box-line)



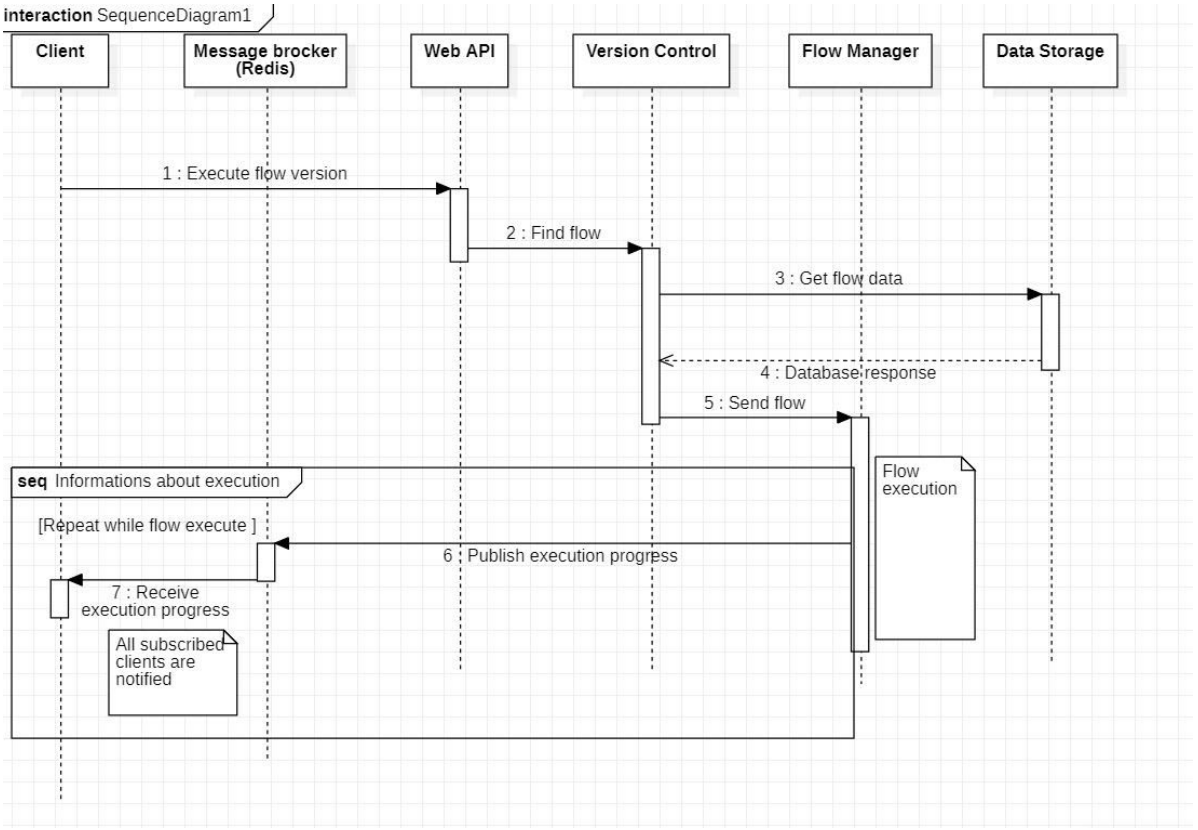
Strukturni pogledi

Navedeni dijagram ilustruje strukturu sistema navodeći komponente sistema kao i njihovu međusobnu povezanost. Glavne komponente koje čine ovaj sistem su klijent, message broker, web API sa version control-om i ORM-om ka bazi podataka koja se nalazi u centralizovanom skladištu podataka i flow manager koji izvršava tok.



Bihevioralni pogledi

Navedeni dijagram ilustruje bihevioralni pogled sistema prilikom pokretanja toka algoritma i praćenja njegovog izvršenja. Prilikom pokretanja nalazi se željeni tok u bazi podataka, koji se zatim pokreće na kompuzacionom serveru. Svi korisnici koji su se pretplatili dobijaju informacije o izvršavanju u realnom vremenu.



Dijagram rasporedjivanja

Navedeni dijagram ilustruje dijagram raspoređivanja sistema na čvorovima, odnosno računarima. Sistem se sastoji od sledecih čvorova:

- klijentski računari - pokreću web aplikaciju
- centralni server - upravlja svim sinhronim komunikacijama (komunikacija sa klijentom, komunikacija sa serverom podataka, komunikacija sa komputacionim serverima)
- komputacioni serveri - vrše obradu podataka (mogu biti lokalni računari ili platforme kao što su GCP, AWS, Azure, itd.)
- redis server - omogućava asinhronu komunikaciju izmedju klijenata i komputacionih servera
- server podataka - upravlja centralizovanom bazom podataka i omogućava fajl sisteme koji se mogu montirati na komputacionim serverima