Arhitekturni projekat UMLDraw aplikacije

UMLDraw apikacija

UMLDraw je distribuirana aplikacija koja ima za cilj kolaborativno crtanje Use case i dijagrama klasa. Pruža korisnicima mogućnost kreiranja sopstvenog tima ili priključivanje nekom od već postojećih timova tako da korisnik može biti u ulozi crtača ili posmatrača.

Funkcionolnosti

Prilikom pristupa aplikaciji UMLDraw, korisnik će moći da izabere da li želi da se priključi vec kreiranim timovima ili da kreira svoj tim.

Ukoliko se priključi nekom od kreiranih timova, biće posmatrač, tj. gledaće crtanje UML dijagram od strane črtaca, pri čemu će prvo dobiti celokupni dotada nacrtani dijagram iz baze. U slučaju da kreira svoj tim, moći će da počne sa crtanjem i moći će da se drugi korisnici priključe njegovom timu i posmatraju crtanje.

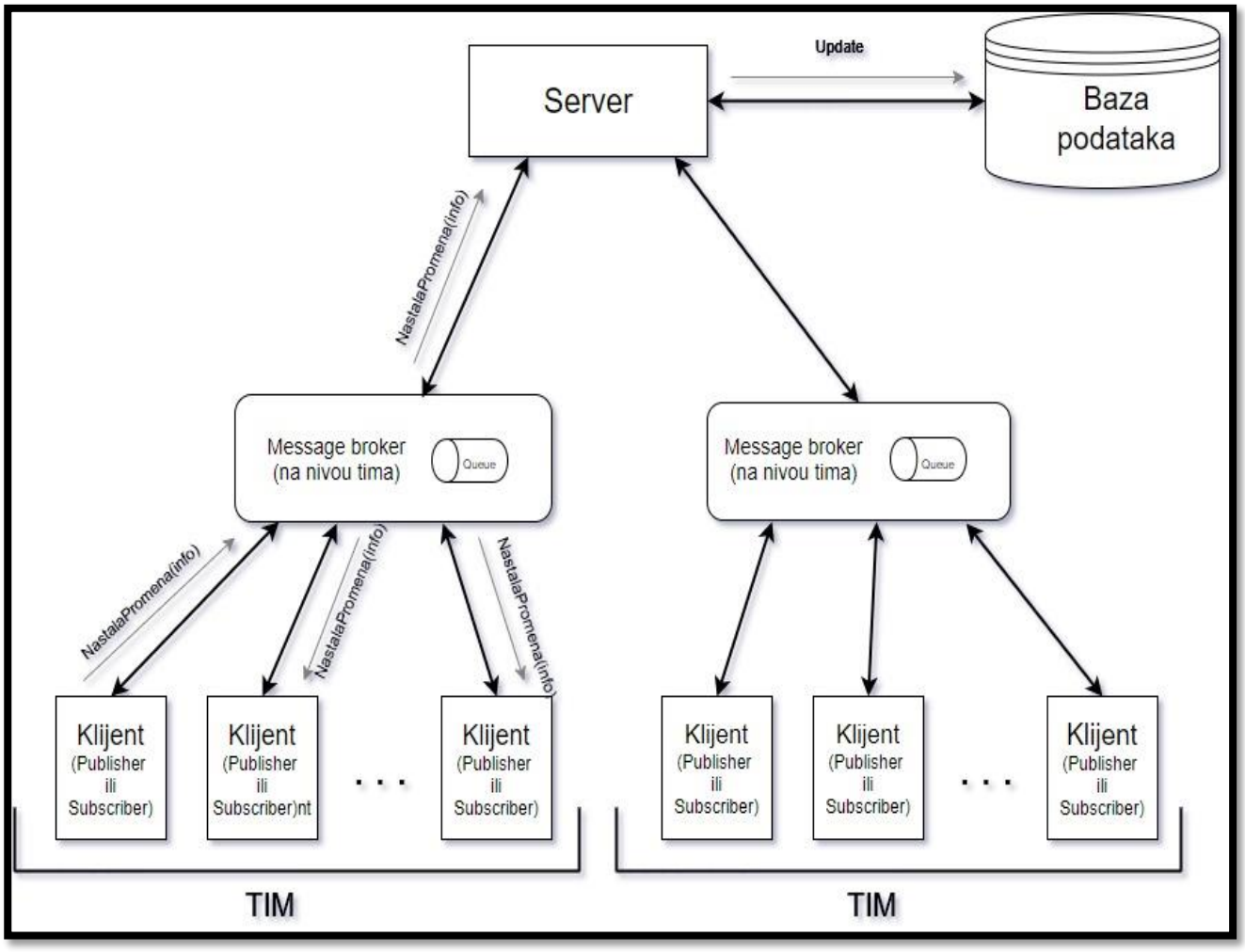
Kako ne bi doslo do blokiranja svih ostalih korisnika koji žele da crtaju, crtač će imati određeno vreme u kom ce moći da crta a kada istekne, mogućnost crtanja dobiće neko drugi. Takođe, korisnik koji trenutno crta može svojevoljno označiti da je završio sa crtanjem I pre isteka vremena, te time pravo na crtanje prelazi na drugog korisnika iz tima.

Tokom crtanja svim korisnicima će se u realnom vremenu prikazivati izmene na dijagramu i koliko je jos vremena preostalo crtacu da zavrsi crtez.

Arhitektura

Arhitektura DrawUML aplikacije se bazira na tri arhitekturna obrasca: Klijent server, Publisher Subscriber i Model View Controller. Klijenti predstavljaju aplikacije koje će vršiti konkretno crtanje i prikazivanje dijagrama korisniku. Server je zadužen za čuvanje informacija o trenutno aktivnim dijagramima, upravljanje timovima i njihovim članovima, kao i čuvanje dijagrama u bazi podataka. Komunikacija između servera i klijenta se obavlja preko Message brokera. Svi klijenti koji su članovi nekog tima su povezani na isti message broker. Svaki od klijenta pojedinačno može imati ulogu Publishera ili Subscribera, sa tim ograničenjem da u jednom trenutku samo jedan klijent uokviru tima može biti Publisher. Taj aktivni klijent jeste onaj koji u određenom trenutku vrši crtanje, dok svi ostali klijenti - članova tima prate rad aktivnog klijenta putem poruka o nastalim promenama koje dobijaju od Message Brokera. Takođe i server biva obavešten o nastalim promenama nad dijagramima, kako bi te promene mogao da evidentira u bazi podataka i time obezbedi perzistenciju podataka.

Šema arhitekture data je na sledećoj slici:



Slika 1 – Arhitektura DrawUML aplikacije

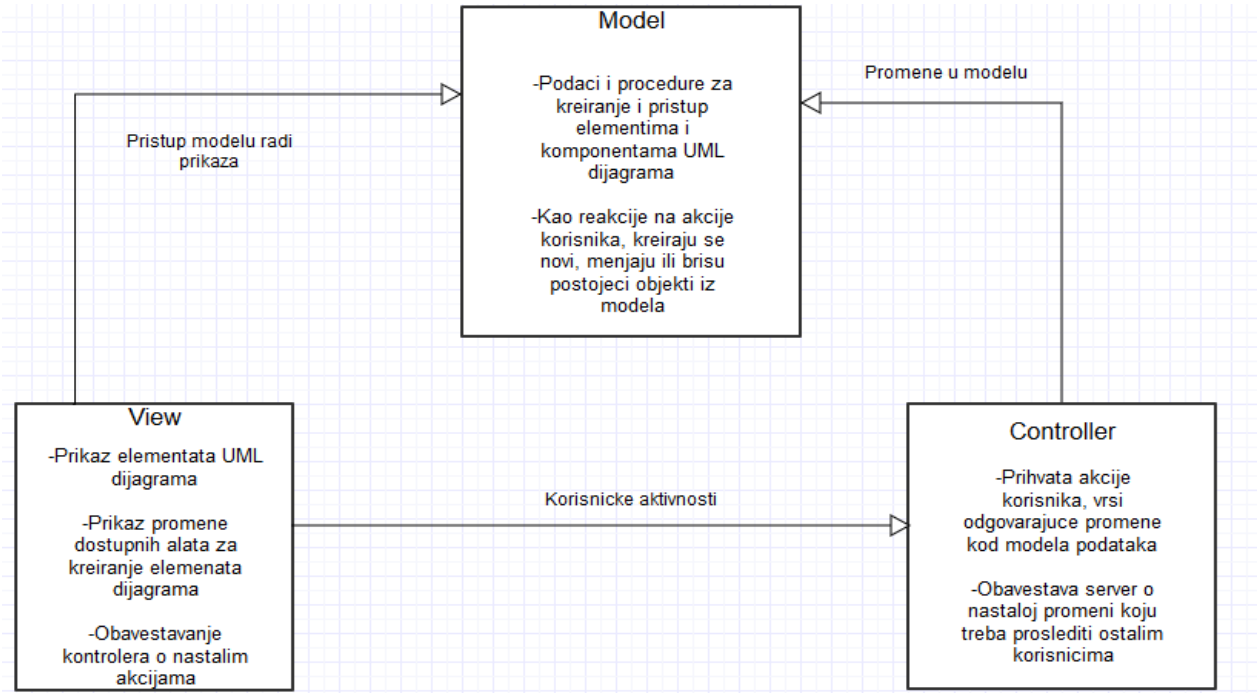
Sam klijent je zasnovan na Model View Controler arhitekturnom obrascu.

MVC obrazac sadrži tri komponente:

- Model- čuva podatke i procedure za kreiranje i pristup elementima UML dijagrama
- View-vrši prikaz elemenata UML dijagrama
- Controler-prihvata akcije korisnika

MVC arhitektura prikazana je na slici 2.

Programski jezik u kome će biti implementirana aplikacija jeste Java.

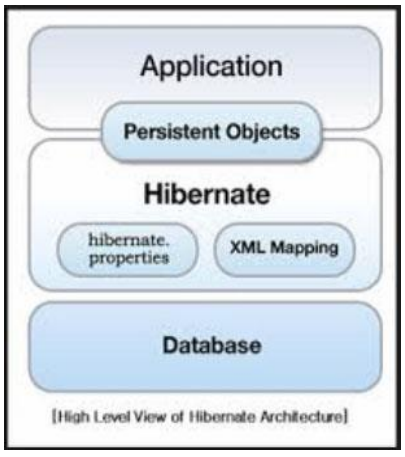


Slika 2 – Arhitektura Klijenta (MVC obrazac)

Hibernate

Hibernate je ORM biblioteka za Javu koja pruža framework za mapiranje objektno orijentisanog modela domena u tradicionalnu relacionu bazu. To nam omogućava da obezbedimo perzistentnost objekata sa kojima aplikacija radi bez potrebe da ulazimo u detalje same perzistencije, i pišemo kod koji će čuvati i pristupati objektima u relacionoj bazi korišćenjem SQL jezika.

Osnovna funkcija Hibernate framework-a jeste mapiranje objekata sa kojima radi aplikacija (objekata poslovne logike) na odgovarajuće tabele iz relacione baze koja se koristi za njihovo perzistiranje, kao i obavljanje CRUD (create, read, update, delete) operacije nad tim objektima. Predstavlja posrednika između same aplikacije i baze podataka, oslobađajući programere potrebe da vode računa o načinima na koje se njihovi objekti smeštaju u bazu.



Slika 3 – Arhitektura Hibernate

Hibernate koristi postojeći Java API, uključujući Java Database Connectivity (JDBC), Java Transaction API (JTA) i Java Naming and Directory Interface (JNDI). Pošto generiše JDBC kod, omogućava da se Hibernate koristi sa bilo kojom bazom za koju postoji JDBC drajver. JDNI i JTA omogućava korišćenje framework-a sa J2EE aplikacionim serverima.

Razlozi zbog čega smo se opredelili za Hibernate kao persistence framework:

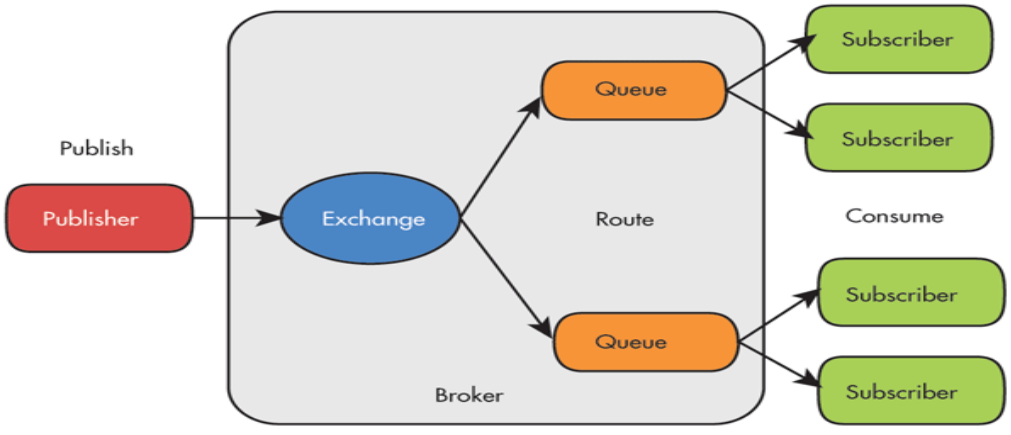
- Hibernate framework je open source (GNU Lesser General Public License (LGPL), te se može besplatno preuzeti i koristiti.
- Podržava mnoge relacione baze, među kojima i MySQL koju ćemo mi koristiti za projekat.
- Ne zahteva korišćenje posebnih interfejsa kako bi se obezbedila perzistentnost objekata klasa u aplikaciji, niti zahteva da aplikacija prati neku standardnu strukturu. Zbog toga se može lako integrisati sa različitim *J2SE/J2EE* aplikacijama i drugim framework-ovima.
- Mapiranje objekata na relacionu bazu je moguće definisati preko XML mapping fajla ili pisanjem Java anotacija direktno u kodu.
- Omogućava reverse engineering baze podataka.
- Hibernate framework je često korišćen kod Java aplikacija, tako da je dobro dokumentovan i ta dokumentacija je dostupna na internetu.

RabbitMQ

RabbitMQ predstavlja message Broker, softver čija je uloga da pojednostavi razmenu poruka među aplikacijama. Obezbeđuje aplikacijama jednostavnu platformu za slanje I primanje poruka, kao I sigurnost da će poruke biti isporučene primaocima. Poruke koje se razmenjuju mogu sadržati bilo kakve informacije relevantne za domen aplikacija koje komuniciraju. Message broker će poslate poruke smestiti u red I čuvati ih u redu sve dok primalac ne bude spreman da ih preuzme.

RabbitMQ pruža podršku za veliki broj operativnih sistema I programskih jezika. Implementira Advanced Message Queuing Protocol (AMQP), ali pored ovog pruža I podršku za druge protokole preko plugin-ova.

RabbitMQ, kao I većina message brokera, razmenu poruka obezbeđuje korišćenjem redova. Osnovna ahitektura RabbitMQ brokera je data na slici 4.



Slika 4 – Arhitektura RabbitMO bokera

Razlika od nekih drugih message brokera jeste u tome što publisher (aplikacija koja šalje poruke) nema direktnu komunikaciju sa redom. Umesto toga, publisher poruke šalje komponenti koja se

naziva Exchange. Exchange je zadužen za rutiranje i prosleđivanje poruka u odgovarajuće redove. Redovi se povezuju sa Exchange-om (bind) preko ključa za povezivanje – binding key. Sa druge strane, svaka poruka u sebi sadrži svoj ključ rutiranja – routing key. U koje redove će poruka biti prosleđena zavisi od para vrednosti ključeva binding key i routing key. Način na koji će ova dva ključa biti upoređivana zavisi od tipa Exchange-a. Postoje 5 tipa:

- Fanout
- Direct (*routing key = binding key*)
- Topic (*parcijalno poklapanje)*
- Header (*koristi zaglavlje poruke umesto routing key vrednosti*)
- Default (nameless) exchange (*routing key = queue name*)

Razlozi zašto smo se opredelili za korišćenje RabbitMQ message brokera:

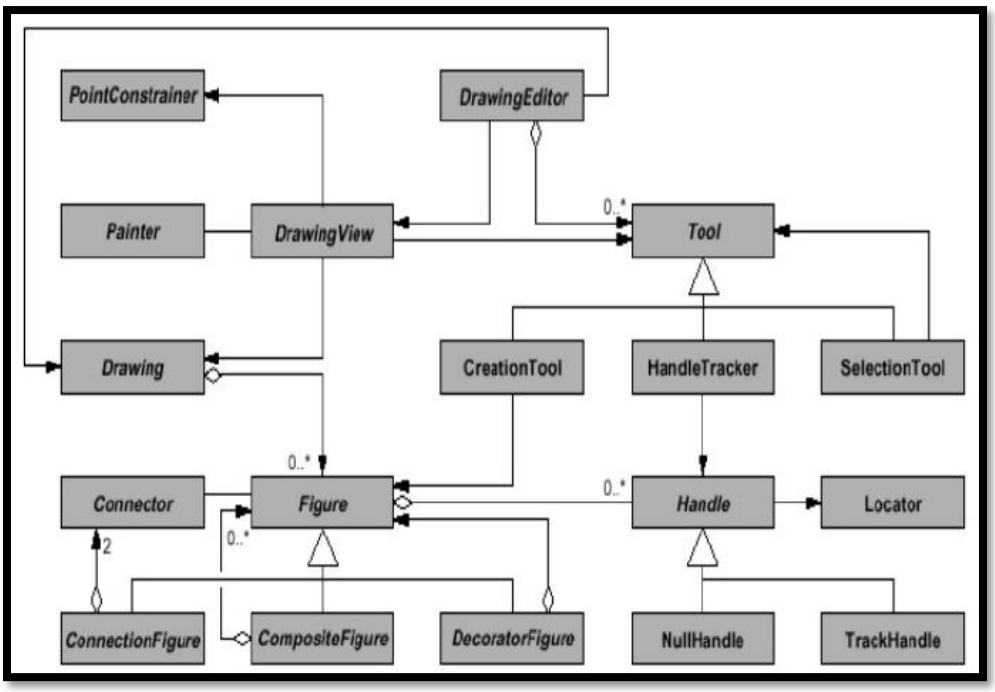
- RabbitMQ je open source message broker, pa se može besplatno preuzeti i koristiti.
- Moguće je preuzeti RabbitMQ server i instalirati ga lokalno na računaru, ili koristiti CloudAMQP, koji predstavlja RabbitMQ server hostovan na cloud-u I koji se može besplatno koristiti kreiranjem naloga.
- Pruža podršku za mnoge programske jezike, uključujući i Java programski jezik.
- Poruke se kodiraju kao bajtovi, pa je moguće preneti bilo koju vrstu podatka.
- Na zvaničnoj internet stranici je moguće naći dosta uputstava i informacija o načinu instaliranja, korišćenja kao i dosta primera konkretne upotrebe.

JHotDraw

JHotDraw je Java GUI framework za tehničku i struktuiranu grafiku. Iako je prvobitno razvijen kao dizajn vezba danas je to veoma moćan alat. Aplikacije koje kreira JHotDraw čuvaju kolekciju oblika koji su odvojeni jedni od drugih. Da bi povećao svoju fleksibilnost, JHotDraw prati principe šablona dizajna MVC-a tako što odvaja svoj model (podatke) iz svog pogleda (izgleda). Crtež se može smatrati modelom koji sadrži trenutno stanje aplikacije, posebna klasa nazvana DravingView predstavlja grafički izgled u GUI aplikacije.

On je dobar primer framework-a koji se sastoji od osnovnih dizajn obrazaca, kao što su Composite, State, Template, Factory method i Strategy. Poznavanje osnovnih pojmova iza tih obrazaca i njihovih aplikacija u okviru JHotDraw-a, pomaže da odlučite kako ga prilagoditi kako biste zadovoljili zahteve aplikacije.

Osnovna struktura JHotDraw framework-a dat je na slici 5.



Slika 5 – Arhitektura JHotDraw framework-a

Razlozi zašto smo se opredelili za korišćenje JHotDraw framework-a:

- JHotDraw framework je open source i dobro je dokumentovan
- JHotDraw je korišćen u mnogim poznatim aplikacijama kao što su: JARP, Joone, Renew,Netlogo i mnoge druge što govori o tome koliko su njegove mogućnosti velike i koje vrste aplikacija mogu da se naprave sa njim.
- JHotDraw je pogodan upravo za crtanje UML dijagrama