-Arhitekturno rešenje-

Stefan Sentić, 15359 Marko Vučković, 15473

Sadržaj

Funkcionalnosti

Comet je aplikacija koja služi kao kolaborativni *text/code* editor. Dokumenti sa kojima radi mogu biti tekstualni fajlovi ili *source* kod pisan u nekom programskom jeziku. Omogućava kolaborativni rad nad jednim ili više takvih fajlova. Funkcionalnosti aplikacije *Comet* su:

- Kreiranje tekstualnih fajlova ili *source* fajlova u nekom od programskih jezika
- Editovanje i pregled već postojećih tekstualnih ili *source* fajlova.
- Kolaboraciju nad takvim fajlovima u realnom vremenu
- Komunikacija u realnom vremenu između ljudi koji rade na istom fajlu/dokumentu
- Ukoliko se radi sa *source* fajlovima, *Comet* omogućava pokretanje konzole i izvršavanje tih fajlova u konzoli.
- Dodavanje kolaboratora za određeni dokument od strane *owner*-a dokumenta, uz mogućnost dodeljivanja različitih vrsta privilegija istima (*read, write..)*
- Prijavljivanje korisnika za ulogu kolaboratora na nekim od postojiećih dokumenata
- Čuvanje više različitih verzija fajla za vreme izrade istog, uz mogućnost vraćanja fajla na neku od prethodnih verzija
- Mogućnost da se aplikacija koristi i *offline* pri čemu bi radila kao i svaki drugi tekst/code editor
- Mogućnost čuvanja lokalnih kopija fajlova od strane kolaboratora

Pošto je *Comet* i distribuirana aplikacija, treba da omogući i:

- Pregled liste dostupnih fajlova na kojima je moguće raditi
- Pristup nekom od postojećih fajlova
- Registraciju korisnika i upravljanje korisničkim nalozima

Prilikom projektovanja aplikacije neophodno je obratiti pažnju i na funkcionalne zahteve zadate od strane profesora. Ti zahtevi su:

- Snimanje podataka u relacionu bazu podataka korišćenjem Hibernate (ili nekog drugog) ORM alata.

- Podrška za istovremeni rad više klijenata na jednom dokumentu (crtežu, partiji igre itd.) pri čemu prvi korisnik koji je otvorio dokument može da unosi izmene, a svi ostali (obavezno podržati više) mogu u realnom vremenu da prate izmene. Ukoliko prvi korisnik (onaj koji unosi izmene) zatvori dokument, prvi sledeći korisnik koji je imao read-only pogled dobija read-write privilegiju.
- Implementirati sinhronizaciju dokumenta kod svih klijenata korišćenjem neke message-passing biblioteke (jeromq).
- Omogućiti istovremeni rad na više dokumenata (većeg broja grupa korisnika).
- Omogućiti adekvatnu vizuelizaciju većeg broja različitih elemenata iz modela podataka pri čemu je moguće svakom elementu dodeliti tekstualnu labelu.
- Omogućiti iscrtavanje veza između objekata modela podataka.
- Uvesti različita ograničenja prilikom uvođenja veza u model (nije moguće povezati bilo koji objekat sa bilo kojim drugim, ograničiti broj veza po objektu itd.).
- Prilikom implementacija na odgovarajućim mestima iskoristiti barem 3 različita projektna obrasca.

Nefunkcionalni zahtevi i ograničenja

Prilikom projektovanja *Comet* aplikacije, ustanovljena je potreba za sledećim nefunkcionalnim zahtevima:

- Proširljivost (*extensibility*)
- Reupotrebljivost (*reusability*)
- Pouzdanost (*reliability*)
- Lakoća testiranja (*testability*)
- Lakoća korišćenja (*usability*)
- Dokumentacija
- Performanse
- Otpornost na greške (*fault tolerance*)

- Emocionalni faktori (*emotional factors* (*fun, educational*))
- Podrška za različite operativne sisteme (*cross-platform*)
- Lako održavanje (*maintainability*)
- Otvoren kod (*open source*)

Arhitekturni zahtevi

Na osnovu navedenih nefunkcionalnih zahteva (odnosno atributa kvaliteta) i ograničenja, imamo sledeće arhitekturne zahteve:

- Proširljivost - Arhitektura aplikacije mora da bude takva da omogući lako dodavanje novih funkcionalnosti na bazi trenutne aplikacije.
- Reupotrebljivost - Komponente aplikacije treba da budu projektovane tako da mogu lako da se iskoriste u budućim projektima slične funkcionalnosti.
- Pouzdanost - Aplikacija ne sme korisnicima da prikaže nevalidne modifikacije fajla nad kojim se kolaborativno radi. Takođe ne sme doći do gubljenja poruka koje se razmenjuju između korisnika i servera.
- Lakoća testiranja - Komponente sistema treba projektovati tako da je njihovo testiranje lako, odnosno komponente treba da su u slaboj međusobnoj sprezi.
- Lakoća korišćenja - Korisnički interfejs treba da bude jednostavan i lak za korišćenje. Korisnik ne treba da provede više od pola minuta u razmišljanju o tome koji alat čemu služi.
- Dokumentacija - Kod treba da bude dobro dokumentavan.
- Performanse - Odziv aplikacije pri normalnim uslovima korišćenja (sinhronizacija verzija fajlova nad kojima se radi) ne sme da bude veći od 3 sekunde (za sve klijente).
- Otpornost na greške - Prilikom nedostupnosti servera, aplikacija treba da bude u stanju da pruži funkcionalnosti na nivo jednog korisnika.
- Lako održavanje - Komponente sistema treba da imaju jednostavna i lako razumljiva imena koja opisuju njihovu svrhu u cilju lakšeg održavanja koda u budućnosti i lakog integrisanja novih članova tima u ceo proces.

- Emocionalni faktori - Aplikacija treba da ostavi jak emotivni utisak učenja na korisnike, kako bi oni lakše prihvatili i stekli potrebno znanje.
- Otvoren kod - Kod aplikacije treba da bude dostupan na javnom repozitorijumu na GitHub-u.
- Podrška za različite operativne sisteme - Rad u aplikaciji treba da je moguć na bilo kom operativnom sistemu (cross-platform)

Arhitektura aplikacije

Ovaj odeljak prikazuje logičku arhitekturu sistema i sadrži opis najznačajnijih klasa, njihovu organizaciju u paktete i podsisteme, i organizaciju podsistema u slojeve.

Arhitekturni pattern-i

Layered pattern

Na najvišem nivou primenjen je slojeviti (*layered*) arhitekturni obrazac. *Comet – collaborative text editor* obuhvata:

- Serverski sloj u kome je realizovana logika aplikacije, model podataka i skup klasa za pristup podacima iz *MySQL* baze podataka.
- Klijentski sloj u kome je realizovana aplikacija koja pruža grafički korisnički interfejs (GUI) pomoću koga korisnici sistema komuniciraju sa Serverskim slojem, odnosno poslovnom logikom i modelom podataka. Pored grafičkog korisničkog interfejsa klijentski sloj sadrži logiku za predstavljanje i obradu podataka i skup klasa za rad sa podacima koji se čuvaju lokalno (konfiguracioni podaci, podaci o trenutno aktivnim dokumentima, podaci o korisniku, itd...).

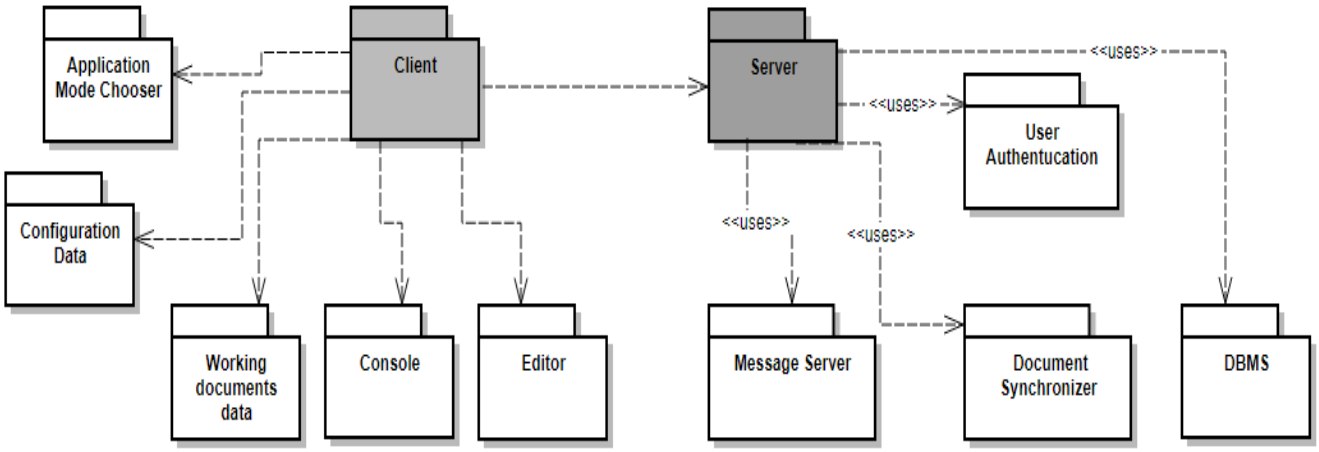
Ova arhitektura je jako pogodna za sisteme gde se deo obrade vrši i na klijentskom sloju.

Model View Controller pattern

U okviru klijentskog sloja primenjen je *Model View Controller* arhitekturni obrazac. Klijentska aplikacija sadrži model podataka, dva različita korisnička pogleda (*View*-a) i njima pridružene kontrolere za dva različita slučaja korišćenja. Aplikacija se može koristiti „*offline*“ i tada radi kao običan tekst *editor*, i može se koristiti „*online*“ i tada pruža dodatnu funkcionalnost u vidu alata za kolaboraciju u realnom vremenu.

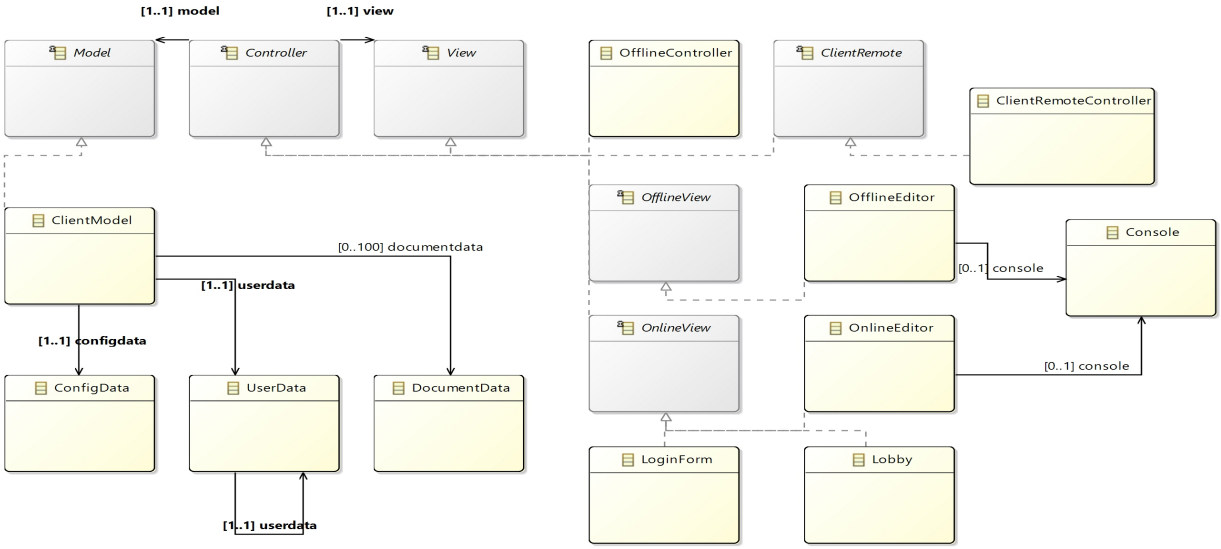
Komponente sistema

Na slici 1.1 dat je prikaz najvažnijih komponenti sistema *Comet* na klijentskom i serverskom sloju.



Slika 1.1

Na slici 1.2 dat je prikaz dijagrama klasa klijentskog sloja sistema *Comet.*

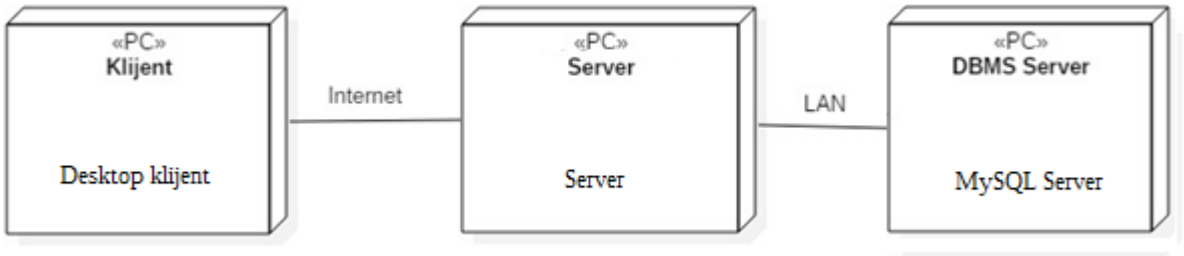


Slika 1.2

Fizički pogled

Fizički pogled sistema prikazuje različite fizičke čvorove za najopštiju konfiguraciju sistema.

Na slici 1.3 dat je dijagram raspoređenja sistema *Comet*.



Slika 1.3

Pogled na procese

U ovom odeljku je dat pregled najvažnijih procesa uključenih u izvršavanje *Comet* sistema.

Procesi

MySQL server

> MySQL Server je proces koji izvršava funkcionalnost MySQL sistema za upravljanje bazama podataka. Ovaj proces može konkurentno da prihvati određen broj upita, izvrši ih nad bazom podataka i vrati rezultate Comet serveru.

Comet server

> *Comet* server je proces koji izvršava funkcionalnost opsluživanja zahteva prispelih od više *Comet* klijenata. Ukoliko korisnik zahteva da se uključi u rad na nekom dokumentu, vlasniku dokumenta stiže zahtev za odobrenje. Takođe je moguće upućivati zahteve za preuzimanje verzija dokumenata na kojima se radi kao i zahteve za dodavanjem nove verzije dokumenta u bazu. *Comet* server omogućava razmenu poruka između korisnika koji rade na istom dokumentu. Razmena poruka je moguća i između klijenata koji čekaju u *lobby*-ju i vlasnika dokumenata.

Comet client

> *Comet* klijent je proces koji izvršava funkcionalnosti aplikacije za prikaz korisničkog interfejsa i izvršava funkcionalnosti *text editor*-a. Takođe pruža interfejs za prikaz postojećih i kreiranje novih kolaborativnih dokumenata. U okviru *online* režima aplikacije prisutan je interfejs za razmenu poruka sa ostalim kolaboratorima.

Konzola

> Konzola je proces koji se pokreće s ciljem kompajliranja i izvršenja dokumenta koji sadrži *source* kod pisan u nekom programskom jeziku. Neophodno je imati odgovarajuće softverske alate za dati programski jezik. U zavisnosti od operativnog sistema prisutnog na računaru, biće pokrenuta i odgovarajuća konzola (*Command Prompt* za *Windows*, *Terminal* za *Linux* i *Mac OS*).

Korišćene tehnologije

Za izradu klijenta i servera sistema *Comet* korišćene su *Java* tehnologije.

Za izradu modela aplikacije korišćen je *Eclipse Modeling Framework, framework* koji pruža alate za crtanje dijagrama klasa i generisanje koda na osnovu dijagrama klasa. Takođe pruža i alate koji omogućavaju *CRUD* operacije nad modelom podataka. Za izradu interfejsa klijentske aplikacije korišćeni su alati *Swing Designer i WindowBuilder*. Za asinhronu komunikaciju između klijenata i servera korišćen je *Java Remote Method Invocation.* Za rad sa bazom koritiće se *ORM* alat *Hibernate,* koji će olakšati perzistenciju podataka u *MySQL* bazu podataka.