- Arhitektura i Projektovanje Softvera -
- Faza 1 -

- Phoenix Fantasy -
- DSV Solution -

Dušan Maksimović 18731
Aleksandar Marinkov 18734
Darko Velinov 17584

1. Kontekst i cilj projekta

Phoenix Fantasy je web aplikacija koja implementira online igru fantasy u kosarkaskom domenu. Aplikacija je dostupna na razlicitim uredjajima i omogucava korisnicima koji vole kosarku da se dodatno zabave, a kosarkaskom takmicenju poveca popularnost i gledanost.

Korisnici mogu da kreiraju nalog. Registrovani korisnici mogu da kreiraju sopstvenu ligu ili da se pridruze postojecoj pomocu koda. Minimalni broj igraca za igranje igre je 2, a maksimalni broj je broj ekipa u takmicenju kako bi svi ucesnici imali po 10 igraca u ekipi. Nakon kreiranja lige ucesnici biraju igrace za svoje timove, i nakon toga liga postaje aktivna. U toku kola takmicenja prikuplja se live statistika igraca i racunaju se njihovi fantasy poeni na osnovu kojih se korisnici rangiraju na listi.

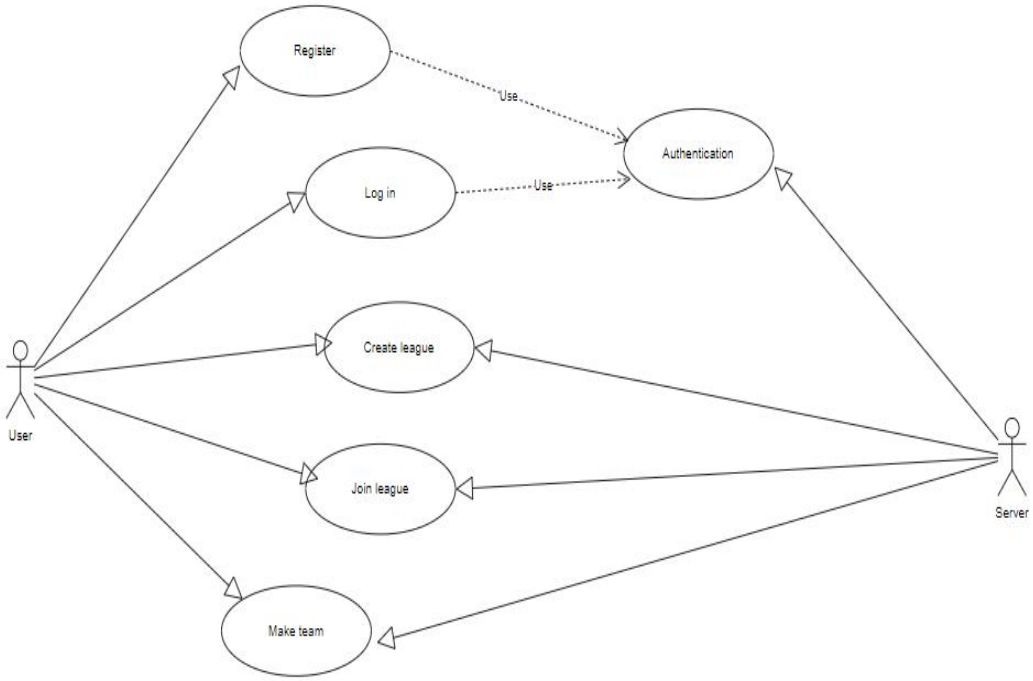
2. Arhitekturni zahtevi (zahtevi)

Ovde cemo definisati arhitekturno znacajne slucajeve koriscenja, glavne funkcionalne i ne-funkcionalne zahteve (atributi kvaliteta) i tehnicka i poslovna ogranicenja vezana za realizaciju projekta PhoenixFantasy.

2.1 Arhitekturno znacajni slucajevi koriscenja (glavni funkcionalni zahtevi)

Funkcionalni zahtevi web aplikacije PhoenixFantasy:

- Registracija korisnika
- Prijavljivanje korisnika
- Kreiranje lige
- Pridruzivanje ligi
- Pravljenje timova u ligi
- Razmena igraca izmedju ucesnika
- Razmena slobodnih igraca
- Razmena poruka izmedju ucesnika
- Unosenje live statistike igraca koja simulira kolo u takmicenju



2.2 Ne-funkcionalni zahtevi (atributi kvaliteta)

- **Pouzdanost i dostupnost:** Potrebno je omoguciti da aplikacija bude dovoljno pouzdana kako ne bi doslo do gubitka podataka u slucaju gubitka konekcije sa serverom. Treba omoguciti da aplikacija bude dostupna korisnicima 24h dnevno
- **Skalabilnost:** Potrebno je obezbediti adekvatan i efikasan rad centralizovane baze podataka sa povecanjem kolicine podataka koju skladisti, koja nastaje usled povecanja broja korisnika
- **Lakoca koriscenja:** Korisnicki interfejs treba da bude dovoljno intuitivan za korisnike sistema, kako bi se povecala korisnost aplikacije
- **Performanse:** aplikacija treba da obezbedi sto manje vreme odziva i najbolje performanse u zavisnosti od trenutnog broja korisnika
- **Sigurnost:** Obezbediti autentifikaciju i autorizaciju, kao i enkripciju osetljivih podataka. Klasicni SQL Injection ne sme probiti zastitu servera.

3.3 Tehnicka i poslovna ogranicenja

- **Optimizacija za web citace** – Google Chrome, Mozilla Firefox, Opera
- **Notifikacija dogadjaja** – obavestenje kada korisnik preduzme specificnu akciju koja se salju preko mejla

- **Apstrakcija podataka** – vazno je da interna organizacija baze podataka(sema) bude sakrivena od strane API-ja

3. Arhitekturni dizajn

3.1 Arhitekturni obrasci

Za projetovanje arhitekture PhoenixFantasy aplikacije bice koriseni arhitekturni obrasci: Layered, MVC, Repository i Publish/Subscribe.

Layered Arhitektura: Arhitekturni obrazac koji forsira podelu strukture aplikacije u slojeve. Svaki sloj u arhitekturi formira apstrakciju oko posla koji treba da se uradi da bi se zadovoljio odredjeni poslovni zahtev. Kod projektovanja aplikacije PhoenixFantasy bice koriscena troslojna arhitktura, standard za projektovanje web aplikacija. Tri sloja su: klijentski, serverski i sloj podataka(sloj baze podataka).

▪ Klijentski sloj: lokalni interfejs koji se koristi za kreiranje, modeliranje, analizu, predstavljanje, izvestavanje i distribuciju razlicitog sadrzaja. Ovaj sloj predstavljace sloj prikazan korisnicima u Web browseru preko kog oni mogu da interaguju sa ostalim slojevima aplikacije.

▪ Serverski sloj: Zahtevi primljeni sa klijentskog sloja se prihvataju i salju dalje odgovarajucem agentu. Ovaj sloj ce klijentu omoguciti sinhronu komunikaciju preko RESTful API poziva i asinhronu komunikaciju pomocu message brokera.

▪ Sloj baze podataka: Tokom rada ovako projektovane aplikacije (komunikacija klijenta i srvera) nastace odredjeni podaci koje ce biti neophodno ocuvati radi omogucavanja normalne funkcije programa. Ovaj sloj ce omoguciti trajnu perzistenciju ovako nastalih podataka.

MVC (Model-View-Controller): Arhitekturni obrazac koji odvaja aplikaciju na tri glavne logicke komponente: model, pogled i kontroler. Svaka od ovih komponenti rukuje specificnim razvojnim aspektima aplikacije. Ovaj obrazac u aplikaciji PhoenixFantasy ce biti upotrebljen za projektovanje RESTful API-ja (sinhrona komunikacija sa klijentom).

▪ Model: Centralna komponenta obrasca, dinamicka struktura podataka aplikacije, nezavisna od korisnickog interfejsa. Ova komponenta se bavi perzistencijom domenskih klasa.

▪ View: Bilo koji graficki prikaz podataka koji su projektovani u modelu. Ova komponenta u aplikaciji PhoenixFantasy nece biti implementirana tako da serverska strana kreira i vraca delove interfejsa, vec ce se interfejs implementirati koriscenjem JavaScript frameworka React, koji ce od servera dobijati podatke u JSON formatu.

▪ Controller: Dobijeni input pretvara u komande koje se koriste za model ili view. Ova komponenta obezbedjuje koriscenje usluga samog RESTful API-ja, cije funkcije poziva klijent, funkcija zatim vrsi obradu ulaznih podataka i podataka iz modela i tako kreira izlaz u vidu JSON objekata koje klijent lako moze da parsira i koristi za svoje potrebe.

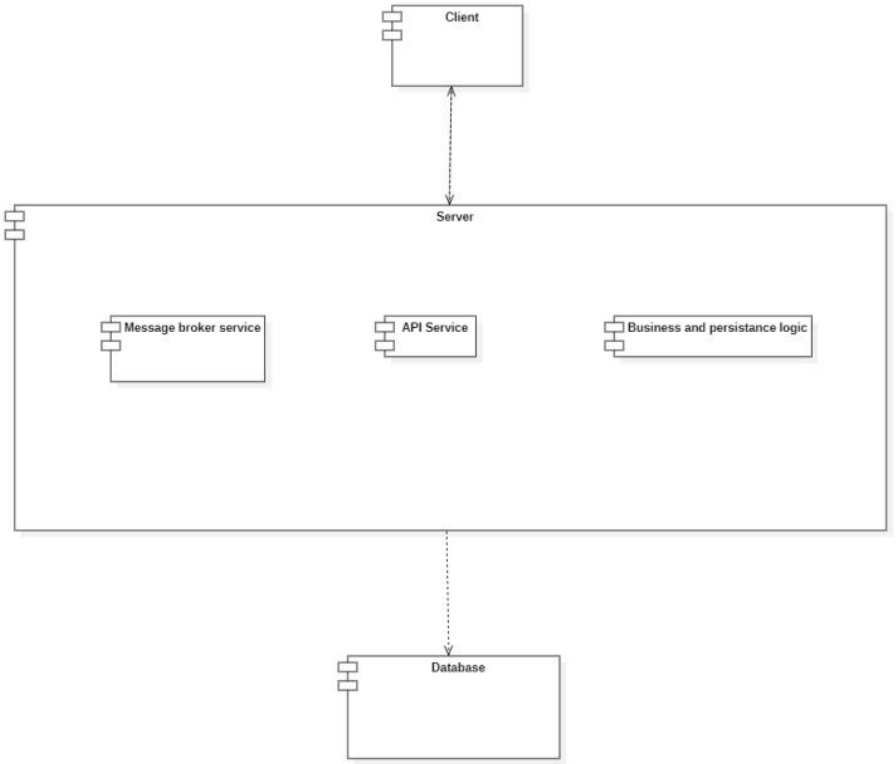
Repository: Arhitekturni obrazac koji aplikaciji omogucava podelu na repozitorijume, klase ili komponente koji obuhvataju logiku potrebnu za pristup izvorima podataka.Oni centralizuju zajednicku funkcionalnost pristupa podacima, obezbedjujuci bolju mogucnost odrzavanja i razdvajajuci infrastrukturu ili tehnologiju koja se koristi za pristup bazama podataka od sloja modela domena.
Ovaj obrazac bice koriscen radi postizanja postojanja interfejsa izmedju centralizovanog skladista (baza podataka) i mapiranja tih podataka(za koji ce biti koriscen Entity Framework ORM). Klijent ce graditi upite koje salje u skladiste za odgovore. Repozitorijumi ce enkaplsulirati skupove objekata iz baze podataka i operacije koje se mogu izvrsiti nad njima, obezbedjujuci put koji je blizi sloju perzistencije.

Publish/Subscribe: Arhitekturni obrazac koji obezbedjuje okvir za razmenu poruka izmedju onoga ko ih proizvodi (publisher) i pretplatnika (subscriber). Publisher i Subscriber se oslanjaju na message brokera, koji poruke prenosi od proizvodjaca do pretplatnika.

Ovaj obrazac bice koriscen radi predavanja asinhronih poruka klijentu od strane servera, putem message broker komponente.

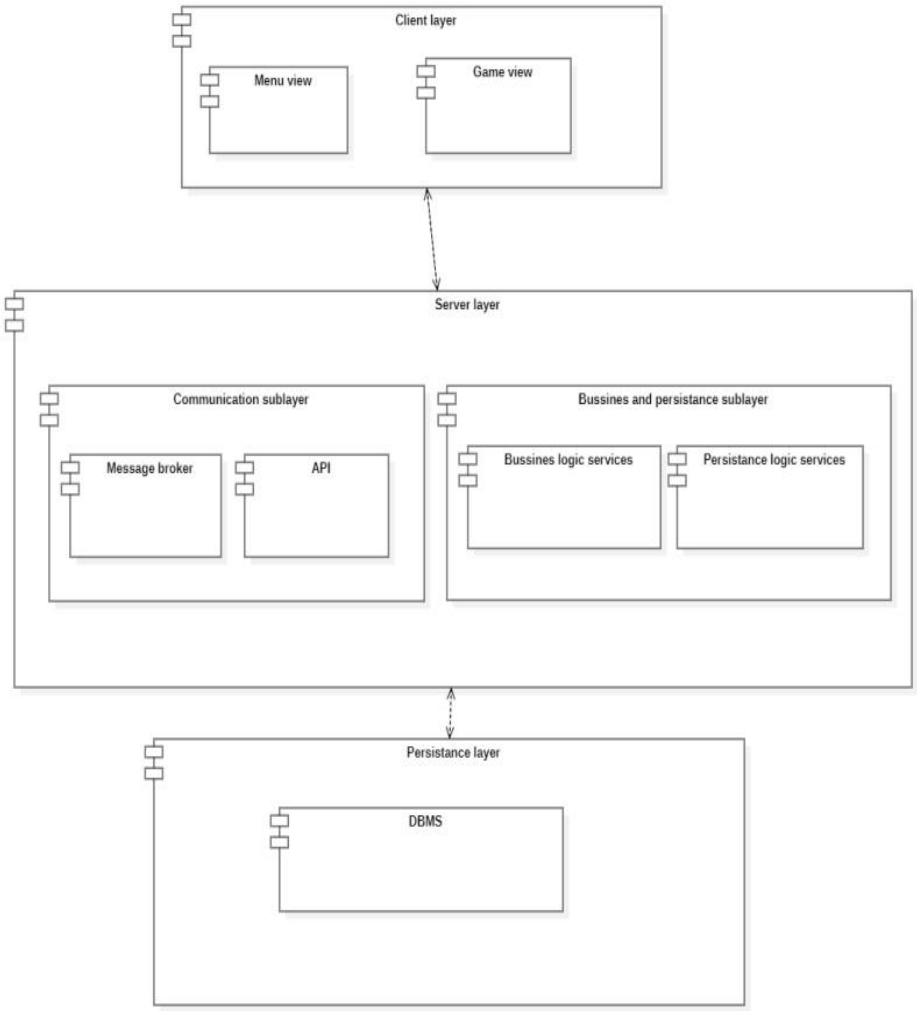
3.2 Generalna arhitektura (box-line)

Arhitektura sistema podrazumeva postojanje klijenta, servera i baze podataka u kojoj ce se cuvati informacije o korisnicima i njihovim igrama.



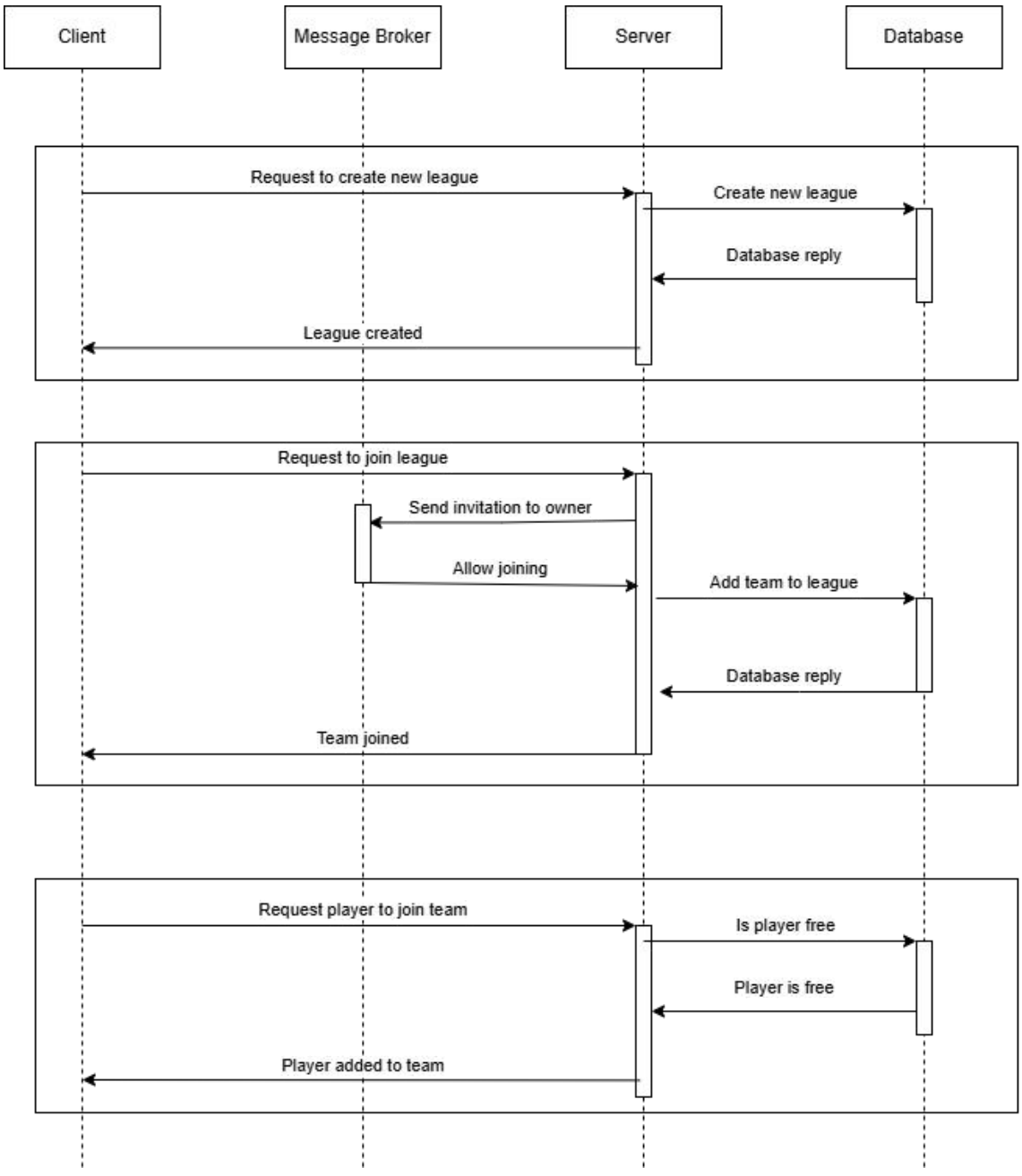
3.3 Strukturni pogledi

Dijagram prikazuje strukturu komponenti sistema i njihovu povezanost. Klijentski sloj, koji predstavlja deo aplikacije sa kojim korisnici interaguju, se sastoji od skripti, od kojih su neke zaduene za prikaz, dok neke ostvaruju komunikaciju sa serverom. Serverski sloj se sastoji od komunikacionog podsloja i podsloja za logiku igre tj. biznis logiku sa podslojem perzistencije. Komunikacioni podsloj obuhvata RESTful Web API za sinhronu komunikaciju i Message Broker za asinhronu komunikaciju sa klijentom. Na sloju perzistencije se nalazi DBMS kao konekcija sa bazom podataka.

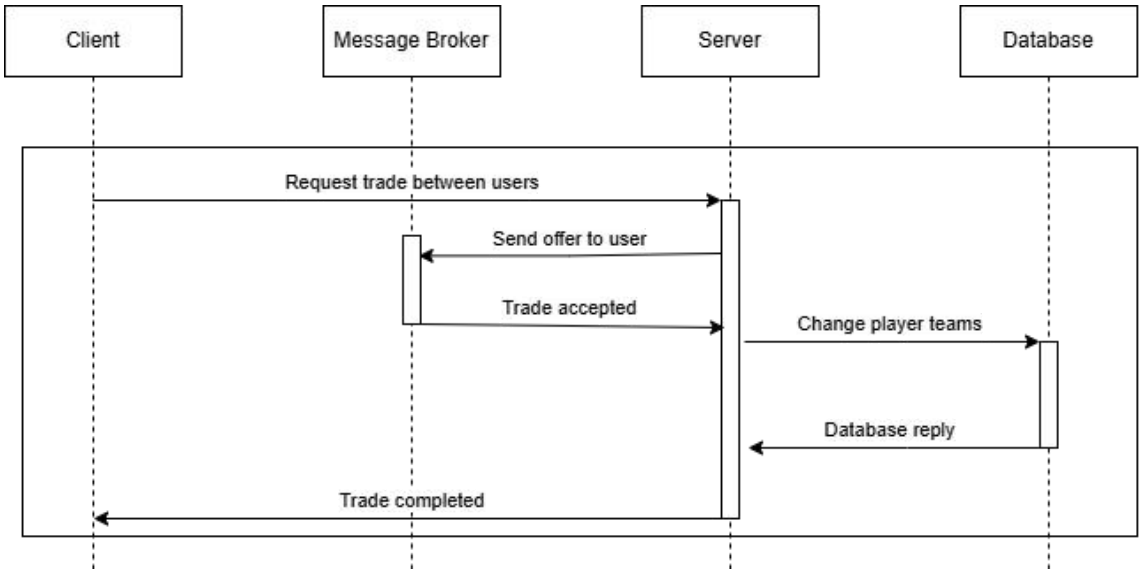

Client layer
Menu view
Game view
Server layer
Communication sublayer
Message broker
API
Bussines and persistance sublayer
Bussines logic services
Persistance logic services
Persistance layer
DBMS

3.4 Bihevioralni pogledi

- **Kreiranje nove lige**



- **Razmena igraca**



3.5 Implementaciona pitanja - biblioteke, komponente i okviri (framework-i) koji ce biti korisceni za implementaciju

- **React** - klijentska strana aplikacije
  Razvojna platforma, izgradjena na Typescript-u, ukljucuje kolekciju dobro integrisanih biblioteka koje pokrivaju sirok spektar funkcija, ukljucujuci rutiranje, upravljanje obrascima, komunikaciju klijent-server itd.
- **ASP.NET Core** - serverska strana aplikacije
  Open-source okvir za kreiranje modernih aplikacija. Dizajniran je tako da obezbedi optimizovani razvojni okvir za aplikacije koje se postavljaju u oblak ili se pokrecu lokalno.
- **MS SQL Server** - relaciona baza podataka
- **Signal R** - biblioteka za uspostavljanje real-time komunikacije
- **Entity Framework** - okvir za objektno-relaciono mapiranje

4. Analiza arhitekture

4.1 Potencijalni rizici u implementaciji i strategije prevazilazenja

Potencijalni rizici u implementaciji Phoenix Fantasy aplikacije ukljucuju probleme sa performansama sistema pod vecim opterecenjem, sigurnosne propuste kao sto su SQL injection, kao i mogucnost gubitka podataka usled prekida u radu ili gresaka. Takodje, moguci su izazovi sa kompatibilnoscu aplikacije na razlicitim uredjajima i preglednicima. Dodatni rizici ukljucuju kasnjenja u razvoju zbog neadekvatnog planiranja ili nepredvidjene slozenosti, kao i rizik od ispada sistema koji bi uticali na dostupnost aplikacije.

Za prevazilazenje ovih rizika, planirana je implementacija skalabilne arhitekture, optimizacija baza podataka i kontinuirano pracenje performansi sistema. Sigurnost ce se osigurati validacijom korisnickog unosa, enkripcijom osetljivih podataka i redovnim sigurnosnim testovima. Gubitak podataka ce se minimizirati automatizovanim rezervnim kopijama i transakcijama u bazi podataka. Kompatibilnost ce se osigurati testiranjem aplikacije na razlicitim uredjajima i preglednicima.