SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Marko Flajšek
Robert Pizek
Tarek Saghir
Karlo Šimunović

SPORT-MANAGER

TEHNIČKA DOKUMENTACIJA

Varaždin, 2017.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Tim: AIR1603

Članovi: Marko Flajšek, 45297/16-R

Robert Pizek, 45274/16-R

Tarek Saghir, 45256/16-R

Karlo Šimunović, 45273/16-R

GitHub: https://github.com/maxxis95/Sport-Manager

Scrum: https://app.scrumdesk.com/#/projects/19299/work/

SPORT-MANAGER

TEHNIČKA DOKUMENTACIJA

Mentor:

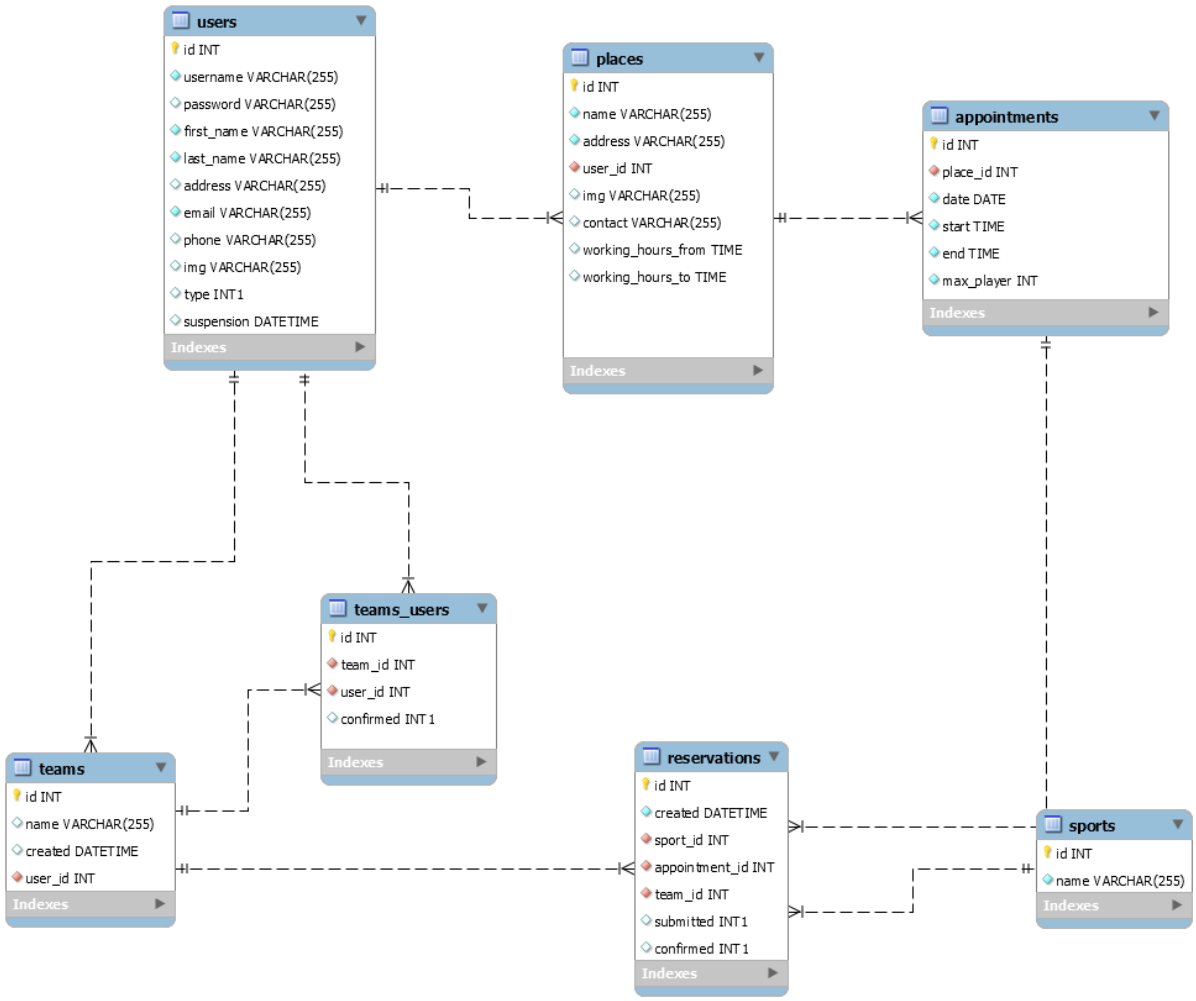
Doc.dr.sc. Zlatko Stapić

Varaždin, siječanj 2017

Sadržaj

1. Model podataka

Model podataka projekta Sport-Manager sastoji se od 7 tablica sa svojim pripadajućim atributima. Za širu sliku projekta i kreiranja *Product Backlog*-a izrađen je ERA dijagram cijele aplikacije koja nas usmjerava u budući razvoj projekta. Praćeno zadanom *Scrum* agilnom metodom je za drugu fazu bilo potrebno implementirati dijelove vezane uz rezervaciju termina, pregleda „mojih“ rezervacija, dodavanje novih termina, dodavanje novih objekata, pregled „mojih“ objekata, svih objekata itd. .Za te funkcionalnosti su se upotrebljavale tablice: „users“, „places“, „appointments“, „sports“. Na slici 1 je prikazan ERA model aplikacije Sport-Manager.



Slika 1. ERA dijagram

Popis te objašnjenje tablica i pridruženih atributa slijedi u nastavku:

- **Users**
  - atributi: id (INT1), username (VARCHAR), password (VARCHAR), first_name (VARCHAR), last_name (VARCHAR), address (VARCHAR), email (VARCHAR), phone (VARCHAR), img (VARCHAR), type (INT1), suspension (DATETIME)
- **Places**
  - atributi: id (INT), name (VARCHAR), address (VARCHAR), user_id (INT), img (VARCHAR), contact (VARCHAR), working_hours_from (TIME), working_hours_to (TIME)
- **Appointments**
  - atributi: id (INT), place_id (INT), date(DATE), start (TIME), end (TIME), max_player
- **Teams**
  - atributi: id (INT), name (VARCHAR), created (DATETIME), user_id (INT)
- **Teams_users**
  - atributi: id (INT), team_id (INT), user_id (INT), confirmed (INT1)
- **Sports**
  - atributi: id (INT), name (VARCHAR)
- **Reservations**
  - atributi: id (INT), created (DATETIME), sport_id (INT), appointment_id (INT), team_id (INT), submitted (INT1), confirmed (INT1)

Tablica „users“ pohranjuje podatke o svim korisnicima aplikacije. Bitno je napomenuti da atribut „password“ pohranjuje hash vrijednost lozinke (koristi se SHA-512 enkripcija). Od svih atributa je vjerojatno potrebno posebno objasniti atribut „type“ koji predstavlja jedan od tipa korisnika npr. običan korisnik, moderator ili administrator te atribut „suspension“ u kojeg se sprema datum suspenzije s obzirom na više neprovedenih rezervacija.

Tablica „places“ sadrži podatke o sportskim objektima koji su registrirani i u upotrebi u aplikaciji. Osim opisnih atributa, tablica sadrži i vanjski ključ „user_id“ se referencira na upravitelja objekta tj. vlasnika u tablici „users“.

Tablica „appointments“ sadrži informacije o mogućim terminima za neki od dodanih sportskih objekata. Tako da, osim opisnih atributa kao što je datum, početak i završetak termina te maksimalan broj korisnika, sadrži i vanjski ključ „place_id“ pomoću kojeg se termin referencira na pripadajući objekt u tablici „places“.

Tablica „teams“ je tablica u koju se spremaju nazivi timova koji će rezervirati neki sportski termin. Bitna je napomenuti atribut „user_id“ u kojeg se sprema ID korisnika koji je kreirao tim.

Tablica „teams_users“ je tablica koja je nastala vezom više-više između tablica „teams“ i „users“, a služi za dodavanje korisnika u timove pa zbog toga sadrži i vanjske ključeve „team_id“ i „user_id“ na tablice „teams“ i „users“.

Tablica „sports“ sadržava listu mogućih sportova koje će biti moguće odabrati na željenoj rezervaciji.

Tablica „reservations“ je povezana ternarnom vezom sa tablicama „appointments“, „teams“ i „sports“. U njoj se bilježe sve rezervacije sa informacijom o kreiranju i potvrdom popunjenosti termina te potvrdom održanosti termina. Sadrži vanjske ključeve „sport_id“, „appointment_id“ i „team_id“.

2. Web servis

Web servis aplikacije Sport-Manager služi za dohvaćanje i ažuriranje podataka vezanih za korisnike, sportske objekte, termine i druge podatke vezane uz funkcionalnosti Sport-Manager aplikacije. Vanjskim korisnicima se podaci prikazuju kroz sučelje mobilne aplikacije.

Komunikacija sa Web servisom omogućena je pomoću GET ili POST zahtjeva HTTP protokola. Nema razlike u njegovom korištenju kod obje metode. Izlazom Web servisa upravlja se pomoću parametara koju se šalju u sklopu GET ili POST zahtjeva. Web servis je dostupan na linku http://sportmanager.fitforev.lin25.host25.com.

U tablici 1 se nalazi popis korištenih tipova podataka.

Naziv tipa podatka	Opis
Integer	Prirodan broj npr. -1, 0, 1, 2, 3
Decimal	Decimalni broj npr. 12.4, 123.4, -2.5
String	Alfa-numerički niz znakova koje može sadržavati unicode znakove
Boolean	Može biti true (1) ili false (0)
DateTime	Datum i vrijeme zapisani u epoch/unix formatu
Object	Kompleksni tip podataka koji sadrži ostale tipove podataka u sebi
Object list	Kompleksni tip podataka koji sadrži listu drugih objekata

Tablica 1. Korišteni tipovi podataka

2.1. Korištenje Web servisa

Web servis prihvaća parametre koji se prosljeđuju zajedno sa zahtjevom u standardiziranom formatu POST parametara (application/x-www-form-urlencoded) ili u obliku upita kod GET poziva, a ispostavlja podatke iz baze podataka i potvrdne poruke ili poruke greške u JSON formatu. Metode zahtjeva su POST ili GET.

Omogućeno je lagano debugiranje (eng. *debugging*) svih zahtjeva pomoću log_file.txt datoteke u koju se zapisuju svi dobiveni parametri te. ako postoje, greške koje su se javile tijekom obrade zahtjeva.

Slijedi popis svih parametara:

- ***method*** - Sadrži naziv metode po kojoj Web servis obrađuje podatke i vraća rezultate
    - Primjer: *method=getData*
- ***table_name*** - Sadrži naziv tablice iz koje se dohvaćaju podaci
    - Primjer: *table_name=Users*
- ***search_by*** - Sadrži stupac ili stupce po kojima se pretražuju podaci u bazi podataka. Može sadržavati više vrijednosti odvojenih znakom „;“.
    - Primjer: *search_by = “place_id;date“, search_by=id*
- ***value*** - Sadrži vrijednost kojoj mora odgovarati stupac/stupci iz *search_by* parametra. Može sadržavati više vrijednosti odvojenih znakom „;“.
    - Primjer: *search_by=“1“, search_by=“1;1482950201“*
- ***data*** - Obavezno polje ukoliko se koristi *setData* metoda. Podaci koji se šalju na Web servis se onda i obrađuju te zapisuju u bazu. Šalju se u JSON formatu na način da se koristi *escape* za dvostruke navodnike „\“.
    - Primjer: *data = "[{\"id\": 4, \"username\":\"ivan55\",\"password\":\"d34kosi4msp1mf\",\"lastLogin\":1478360627}]"*

Moguće vrijednosti parametra *method*:

- ***getData*** - Vraća sve podatke iz baze podataka iz tablice „table_*name*“ koji odgovaraju uvjetima gdje je vrijednost stupca u „*search_by*“ jednaka vrijednosti parametra „*value*“.
    - Primjer: *method=getData&table_name=Users&search_by=id&value=4*
- ***getList*** - Vraća podatke u obliku liste ključ -> vrijednost gdje je ključ ID podatka, a vrijednost ono što se stavi u „*search_by*“ parametar.
    - Primjer: *method=getList&table_name=Users&search_by=email&value=*

- ***setData*** **-** Kod korištenja ove metode obavezno je korištenje „*data*“ parametra. Podaci koji se nalaze u „*data*“ parametru zapisati će se u tablicu „*table_name*“ kao novi zapis ili kao ažuriranje postojećeg zapisa, ovisno o tome postoji li on već u bazi ili ne.

    - Primjer: *method=setData&table_name=Users&data=“{\"id\": 4, \"username\":\"ivan55\",\"password\":\"d34kosi4msp1mf\",\"lastLogin\":1478360627}"*

U nastavku slijedi nekoliko primjera zahtjeva:

- Vraća korisnika s ID-em 5:

    - http://sportmanager.fitforev.lin25.host25.com?method=getData&table_name=Users&search_by=id&value=5

- Vraća sve sportske objekte, rezervacije i timove u jednom odgovoru:

    - http://sportmanager.fitforev.lin25.host25.com?method=getData&table_name=Places,Reservations,Teams

- Ažurira podatke o korisniku ID-a 4:

    - http://sportmanager.fitforev.lin25.host25.com?method=setData&table_name=Users&data="[{\"id\":4,\"username\":\"ivan55\",\"password\":\"d34kosi4msp1mf\",\"lastLogin\":1478360627}]"

Pregled odgovora:

- *User* entitet

NAZIV OBJEKTA	OPIS
User	Predstavlja jednog korisnika i podatke o njemu

Tablica 2. Opis *User* entiteta

- Svojstva *User* entiteta

NAZIV SVOJSTVA	TIP PODATAKA	OBAVEZAN	OPIS
id	Integer	Da	Jedinstveni identifikator korisnika
first_name	String	Da	Ime korisnika
last_name	String	Da	Prezime korisnika
address	String	Ne	Adresa stanovanja korisnika
email	String	Da	E-mail adresa korisnika
phone	String	Ne	Broj telefona korisnika
type	Integer	Ne	Broj koji označava razinu prava korisnika u sustavu
suspension	DateTime	Ne	Datum i vrijeme do kada je korisnik suspendiran od daljnjih rezervacija
places	Object List	Ne	Lista objekata Places entiteta. Sadrži sve sportske objekte s kojima je korisnik povezan
teams	Object List	Ne	Lista objekata Teams entiteta. Sadrži sve sportove s kojima je korisnik povezan

Tablica 3. Svojstva *User* entiteta

- Primjer *User* entiteta


```
{
        "id" : 1,
        "address" : "Pusta cesta 5",
        "email" : "mail@mail.com",
        "phone" : null,
        "type" : null,
        "suspension" : null,
        "places" : [{
                        "id" : 3,
                        "name" : "Sportska Dvorana Gimnazije",
                        "address" : "Gimnazijska 15",
                        "user_id" : 1,
                        "appointments" : [{
                                        "id" : 2,
                                        "place_id" : 3,
                                        "start" : "2016-11-07T13:00:00+00:00",
                                        "end" : "2016-11-07T14:00:00+00:00"
                                }
                        ]
                }, {
                        "id" : 4,
                        "name" : "Nogometno igrali\u0161te kod crkve",
```

```
                        "address" : "Nogometna 2",
                        "user_id" : 1,
                        "appointments" : [{
                                "id" : 1,
                                "place_id" : 4,
                                "start" : "2016-11-06T08:00:00+00:00",
                                "end" : "2016-11-06T12:00:00+00:00"
                        }
                        ]
                }
        ],
        "teams" : [{
                        "id" : 2,
                        "name" : "SuperTroopers",
                        "created" : "2016-11-06T00:00:00+00:00",
                        "user_id" : 1,
                        "_joinData" : {
                                "id" : 1,
                                "team_id" : 2,
                                "user_id" : 1,
                                "confirmed" : 1
                        },
                        "reservations" : [{
                                "id" : 1,
                                "created" : "2016-11-06T00:00:00+00:00",
                                "sport_id" : 1,
                                "appointment_id" : 1,
                                "team_id" : 2,
                                "submitted" : 1,
                                "confirmed" : 1,
                                "sport" : {
                                        "id" : 1,
                                        "name" : "Ko\u0161arka"
                                }
                        }, {
                                "id" : 2,
                                "created" : "2016-11-07T00:00:00+00:00",
                                "sport_id" : 2,
                                "appointment_id" : 2,
                                "team_id" : 2,
                                "submitted" : 1,
                                "confirmed" : 1,
                                "sport" : {
                                        "id" : 2,
                                        "name" : "Nogomet"
                                }
                        }
                        ]
                }
        ],
        "firstName" : "\u0160ljunko",
        "lastName" : "Kamen\u010di\u0107"
}
```

- Primjer poziva *setUserData* metode


```
{
	"method" : "setUserData",
	"data"                                                                                                  :
"{\"address\":\"marmar\",\"email\":\"marmar\",\"firstName\":\"marmar\",\"id\":0,\"l
astName\":\"marmar\",\"phone\":\"222\"}"
}
```


- Primjer odgovora *setUserData* metode


```
{
	"message" : "New user is successfully saved!",
	"statusCode" : 200
}
```


- Primjer odgovora koji sadrži grešku

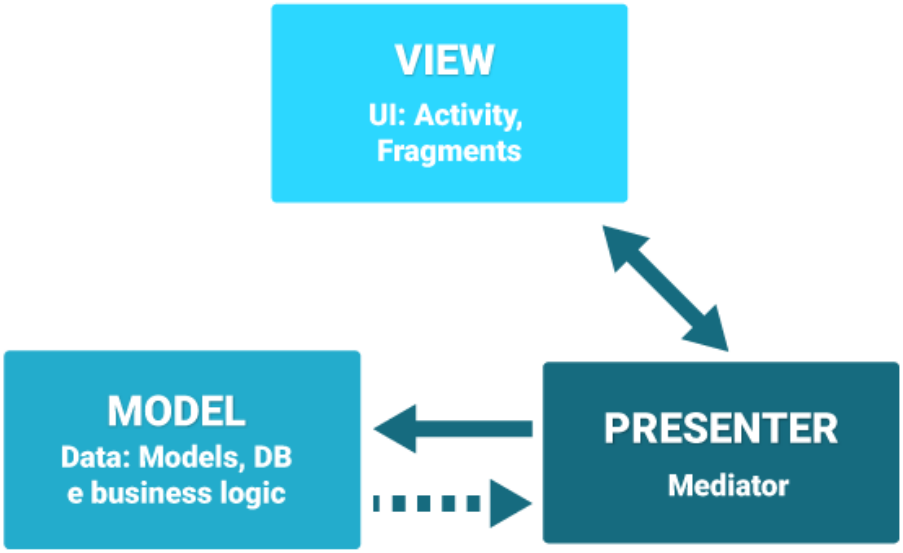

```
{
	"message" : "[ERROR] Saving new user failed!",
	"statusCode" : 404
}
```

3. Arhitektura aplikacije

Istraživanjem na internetu, konzultiranjem sa asistentima te dogovorom u timu odlučili smo na izradu aplikacije po uzoru na MVP arhitekturu. Jedan od razloga odabira upravo te arhitekture jest to što je trenutno iznimno popularna i raširena u Android zajednici. U nastavku je ukratko prikazan način na koji zapravo funkcionira MVP arhitektura.

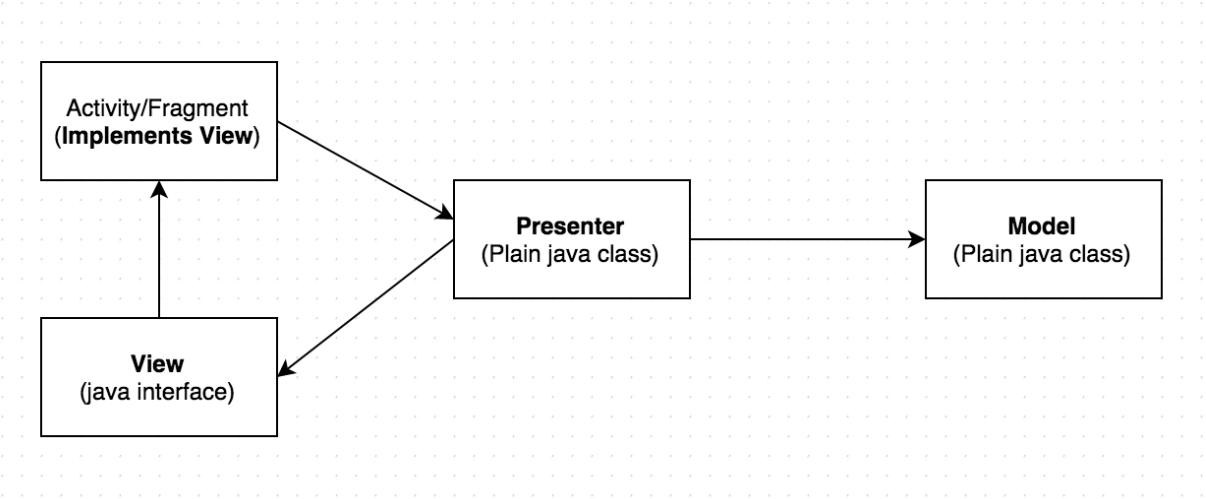
MVP je kratica koja objedinjuje 3 komponente (sloja):

1. ***Model***
   - Vrata prema poslovnoj logici
   - Pruža podatke koje prikazujemo na *View*-u
   - Komunicira sa Web servisom
2. ***View***
   - Sučelje koje implementira aktivnost
   - Sadrži reference prema *Presenter*-u
   - Zaslužan za kreiranje objekta od *Presenter*-a
   - Jedina zadaća jest pozivanje metode sa *Presenter*-om svaki puta kada se dogodi neka akcija (npr. klik na gumb)
3. ***Presenter***
   - Srednja poveznica između *View*-a i *Model*-a
   - Povlači podatke iz *Model*-a te ih formatirane vraća u *View*
   - Odlučuje što će se dogoditi kada se dogodi interakcija na *View*-u



Slika 2. Model MVP arhitekture (https://code.tutsplus.com/tutorials/an-introduction-to-model-view-presenter-on-android--cms-26162)

Odvajanje sučelja od logike u Android-u je teški zadatak, ali MVP svojom strukturom na neki način olakšava organizaciju našeg koda. Na prethodnoj slici je vizualno prikazan način komunikacije između spomenutih slojeva.



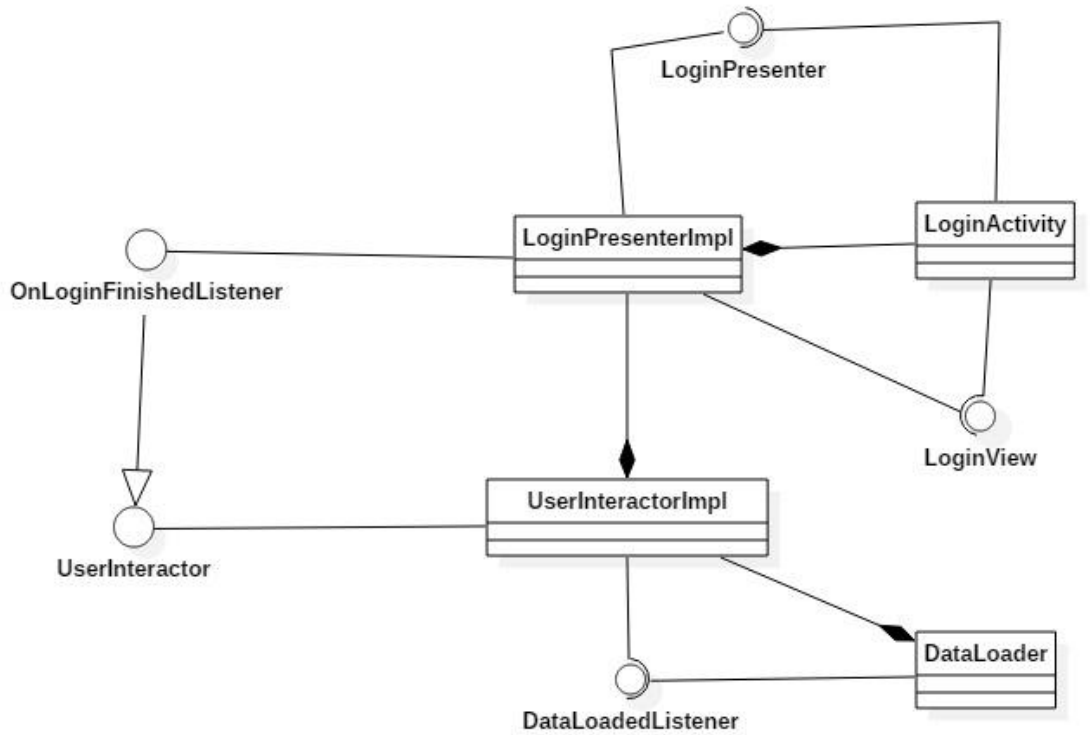
Slika 3. Jednostavan dijagram klasa MVP arhitekture (http://www.singhajit.com/mvp-in-android/)

Dijagram sa slike 3 nam prikazuje primjer implementacije MVP arhitekture u Java programskom jeziku.

U našem projektu koristimo se Web Servisom pa su naše klase u *Model*-u većinom nazvane *Interactor* jer komuniciraju sa metodama Web servisa kako bi u aplikaciji došlo do informacija iz baze podataka. Primjer takve klase je *UserInteractorImpl*.

4. Dijagram klasa

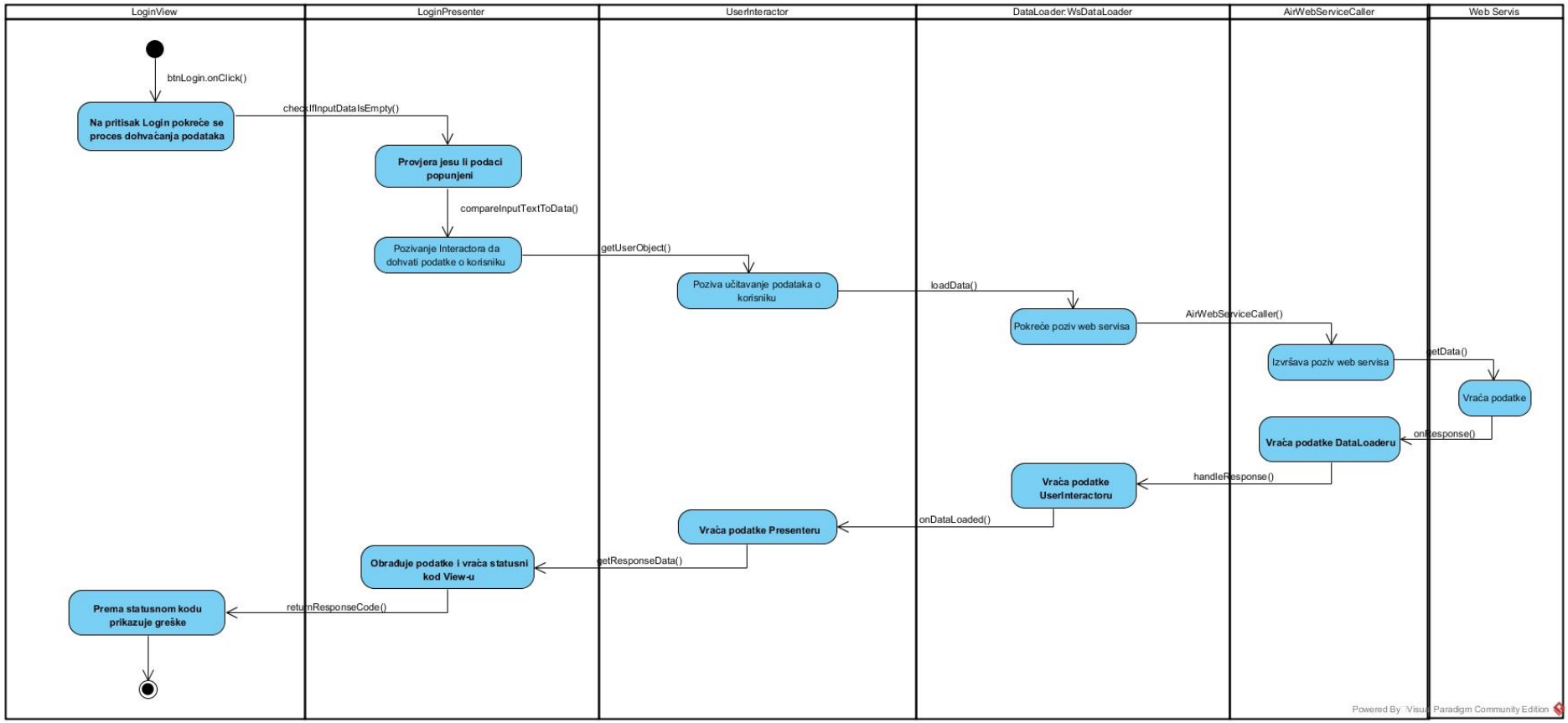
Na slici 2 je prikazan pojednostavljen prikaz dijagram klasa za dio arhitekture kako bi lakše uočili izvođenje komunikacija između slojeve u MVP arhitekturi. Klasa *LoginActivity* implementira sučelje *LoginView* gdje se nalaze metode preko kojih Presenter klasa dolazi do *View* sloja. Dakle, klasa *LoginActivity* i sučelje *LoginView* čine *View* sloj u odabranoj arhitekturi. Sljedeća klasa je *LoginPresenterImpl* koja implementira sučelje *LoginPresenter*, a to sučelje realizira klasa *LoginActivity*. Ovdje možemo uvidjeti prvu komunikaciju između dva sloja - *View*-a i *Presenter*-a. U klasi *LoginActivity* komuniciramo sa *Presenter* slojem preko *LoginPresenter* sučelja, dok s druge strane u klasi *LoginPresenterImpl* komuniciramo sa *View* slojem preko *LoginView* sučelja. Nadalje, dolazimo do *Model* sloja gdje je ovdje za primjer uzeta *UserInteractorImpl* klasu koja implementira sučelje *UserInteractor*. Kao što vidimo na dijagramu *UserInteractor* ima ugniježđeno sučelje *OnLoginFinishedListener* koje prosljeđujemo preko metode koja se nalazi u *UserInteractor* sučelju (*getUserObject*). Dakle način komunikacije između *Presenter* sloja i *Model* sloja se odvija također preko sučelja i njihovih metoda, samo što u ovoj situaciji sučelje kao referencu prebacujemo na *Model* sloj preko metode. Na kraju Imamo *DataLoader* apstraktnu klasu koja sadrži metodu za pozivanje Web servisa. Nju nasljeđuje *WsDataLoader* koji obavlja komunikaciju sa Web servisom.



Slika 2. Dijagram klasa komunikacije između slojeva MVP arhitekture

5. Dijagram aktivnosti

Na slici 3 je prikazan dijagram aktivnosti na primjeru prijave korisnika u aplikaciju. Početna akcija je pritisak gumba na login fragmentu koji pokreće proces dohvaćanja podataka gdje se na *LoginPresenter*-u provjerava jesu li svi podaci popunjeni te se poziva *Interactor* klasa koja komunicira sa Web servisom. U toj klasi se poziva učitavanje podataka o korisniku te se prelazi na *WsDataLoader* klasu gdje se pokreće poziv Web servisa. Nakon te klase prelazi se u *AirWebServiceCaller* klasu gdje se izvršava poziv Web servisa, sa kojeg se i čekaju podaci. Zatim se podaci vraćaju redom u *AirWebServiceCaller* klasu, nakon toga u *WsDataLoader* klasu, u *UserInteractor* klasu i sve do *LoginPresenter-a* koji vraća podatke u *View*, tj. *LoginView* gdje se vraćeni podaci u konačnici prikazuju na *LoginFragment*-u.



Slika 3. Dijagram aktivnosti komunikacije između slojeva i dohvaćanja podataka s Web servisa

6. Prava (privilegije) korisnika

U aplikaciji je kreirana logika koja prilikom prijave korisnika provjerava njegova prava u aplikaciji. Prava korisnika su podijeljena u tri razine:

1. **Registrirani korisnik**
   - Testni podaci:
     - Korisničko ime: *karlo*
     - Lozinka: *m*
   - Popis svih funkcionalnosti koje ovaj korisnik može koristiti je:
     - Pregled svih dodanih sportskih objekata
     - Pregled lokacije sportskog objekta na Google Maps karti
     - Rezerviranje dodanih termina
     - Odabir termina kao privatnog – vidljiv samo korisniku koji rezervira termin i ostalim korisnicima koje je pozvao
     - Odabir termina kao javnog – mogućnost da se bilo koji drugi korisnik pridruži terminu ukoliko ima slobodnih mjesta
     - Kreiranje timova registriranih korisnika
     - Pregled rezerviranih termina kategoriziranih po objektima te njihovo brisanje (kod registriranih korisnika i organizatora – nadležne osobe objekta)
     - Slanje obavijesti drugim korisnicima u timu kod rezervacije termina
     - Potvrda dolaska u neki termin putem NFC-a ili korisnikovim upisivanjem lozinke koju unaprijed pripremi organizator
     - Kreiranje i prijava putem korisničkih računa te mogućnost deaktivacije
     - Odabir sučelja aplikacije na hrvatskom i engleskom jeziku
2. **Moderator (vlasnik objekta ili nadležna osoba)**
   - Testni podaci:
     - Korisničko ime: *vlasnik*
     - Lozinka: *m*
   - Ovaj korisnik može koristit sve funkcionalnosti koje su dostupne registriranom korisniku uz nekoliko posebnih:
     - Dodavanje informacija o postojećim sportskim objektima
     - Dodavanje termina za rezervaciju za svaki sportski objekt

3. **Administrator**
   - Testni podaci:
     - Korisničko ime: *admin*
     - Lozinka: *m*
   - Ovaj korisnik ima sva prava registriranog korisnika i moderatora.

7. Projektni paketi

Već spomenuta arhitektura koje se koristi u ovoj projektu je MVP te su sukladno tome paketi u projektu posložili na slijedeći način:

Aplikacije sadrži dvije aktivnosti. *BaseActivity* i *MainActivity* (u nastavku osnovna i glavna aktivnost). U osnovnoj aktivnosti se nalazi logika za instanciranje *LoginFragment*-a. Glavna aktivnost se odnosi općenito na cijelu aplikaciju te sadrži logiku za *Navigation Drawer*, odnosno klizni izbornik koji se otvara kod pritiska na gumb od tri crte u gornjem lijevom uglu ekrana. Za bolju organizaciju MVP arhitekture su svi dijelovi koda vezani uz *View*, *Model* i *Presenter* smješteni u zasebne pakete.

Budući da u projektu koristimo *RecycleView widget* za koji možemo reći da je malo naprednija i fleksibilnija verzija *ListView*-a, u *adapters* paketu se nalaze klase koje su povezane uz taj *widget* te služe za prikaz podataka u njemu. Nadalje, u *entities* paketu nalaze se klase koje se instanciraju prilikom dohvaćanja podataka sa Web servisa te se pune pravim podacima koji se na kraju prikazuju u fragmentima. *Helper* paket sadrži pomoćne enumeracije koje služe za *TypeSafe* pisanje *switch* naredbe. Zadnji paket je *loaders* paket koji sadrži sučelja, apstraktne i obične klase koje su potrebne za komunikaciju sa Web servisom. Za međusobnu komunikaciju između klasa koriste se sučelja koja opisuju metode koje zadana klasa mora implementirati. Budući da su fragmenti dio *View*-a stavili smo ih u *View* kao pod paket naziva *fragments*.