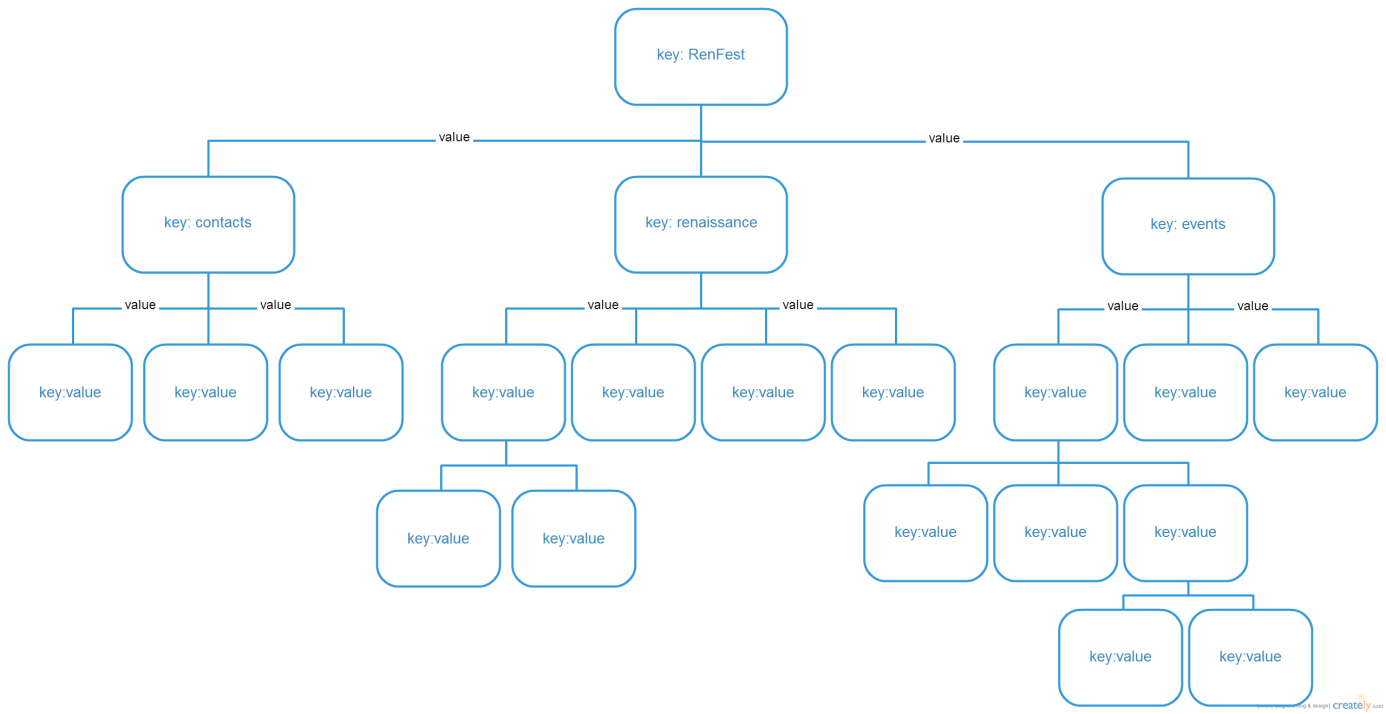
# Model podataka

Kod hibridnog mobilnog razvoja lokalnu pohranu podataka moguće je pohraniti pomoću cache, Web storage (localStorge i sessionStorage), WebSQL storage (SQLite) i IndexedDB storage. Za ovaj projekt odlučili smo se za *localStorage* pohranu podataka. *LocalStorage* omogućuje pohranu kolekcije podataka u obliku *key : value* parova na mobilni uređaj, pri tome key i value moraju biti u obliku stringova. *LocalStorage* nam se činio kao najbolji izbor zbog vrste podataka (velike količine teksta) koje cloud servis (Firebase) vraća, Firebase vraća kolekciju podataka u obliku key : value parova i to u json formatu. Prema tome moglo bi se reći da koristimo NoSQL. NoSQL je horizontalno skalabilan, nije relacijski oblik baze podataka i ima fleksibilnu shemu podataka.

Kako nismo koristili relacijski oblik baze podataka, ne možemo prikazati model podataka u obliku ERA modela, već na sljedeći način:



Slika xyz

Zanimalo nas je kako stvarno izgleda localStorage pohrana na mobilnom uređaju pa smo pomoću Android device monitora došli do localStorage datoteke na mobilnom uređaju (data/data/naša\_app/app\_webview/LocalStorage/ime\_datoteke.LocalStorage) i vidjeli smo sljedeće:

U localStorage postoji jedna tablica *Item table* koja ima atribute *key* i *value.*

Schema:

CREATE TABLE Item table

(

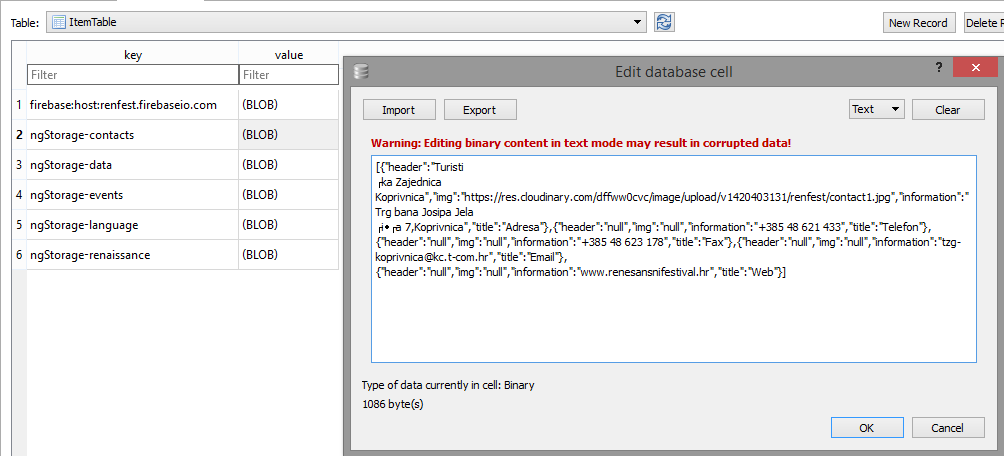
key TEXT UNIQUE ON CONFLICT REPLACE,

value BLOB NOT NULL ON CONFLICT FAIL

)

Slika xy

Na sljedećoj slici xyz možemo vidjeti vrijednosti pohranjene u atributima *key* i *value.* U atributima value pohranjena je kolekcija podataka (u ovom primjeru kontakt podataka) u obliku *key : value*.

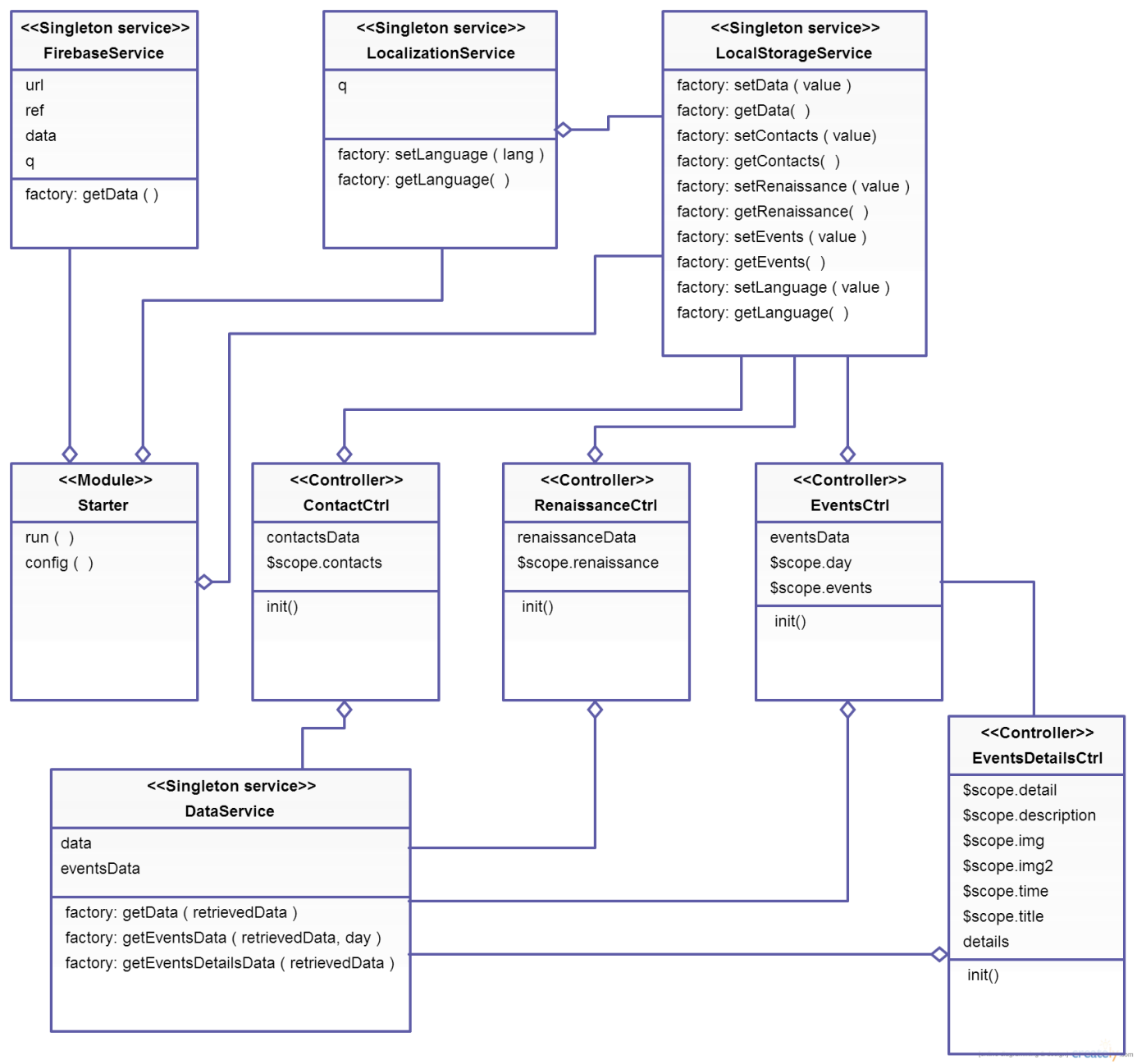


Slika xyz

# Class dijagram

Za razliku od Jave, AngularJs ima malo drugačiju sintaksu, neki svoj stil organizacije i pisanje koda. Kod AngularJs poslovna logika piše se u kontrolerima (eng. Controllers) i u servisima (eng. Services). U kontrolerima kao što je već spomenuto piše se poslovna logika koja se izvršava iza pogleda (eng View). Kontroleri uglavnom služe da prikažu podatke u određenim pogledima te da izvršavaju neke funkcije kao odgovor na korisnikovu interakciju s korisničkim sučeljem. Zatim poslovna logika može se s vremenom zakomplicirati i povećati kako projekt raste i kako se dodavaju nove funkcionalnosti, zato postoje servisi. Servisi također služe za pisanje poslovne logike, no oni se koriste kada se neki servis, neka njegova metoda može iskoristiti na više mjesta, u više kontrolera. Također da kontroleri ne bi bili nepregledni i ako vidimo da neki dio koda ponavljamo u više kontrolera tada taj kod izdvajamo i 'selimo' u servis, odnosno kreiramo metodu koja nam vraća taj isti rezultat, zatim tu metodu pozivamo u kontrolerima. AngularJs servisi su singleton i sve njegove metode su factory.

AngularaJs aplikacije koristi 'kontejnere' (eng. Module) gdje svaki modul sadrži odgovarajuće kontrolere i servise. Neka praksa je da se aplikacija podijeli na više modula, u našem slučaju po funkcionalnostima. Prema tome za neku funkcionalnost aplikacije postoji odgovarajući modul koji sadrži određene kontrolere, servise koji omogućuju da ta funkcionalnost funkcionira. Kada se ti svi moduli spoje dobimo finalni proizvod, tj. aplikaciju sa svim funkcionalnostima.



U nastavku slijedi opis dijagrama klasa.

**FirebaseService**

*FirebaseService* je servis koji služi dohvaćanje podataka s Firebase clouda. *FirebaseService* servis pomoću preuzete *AngularFire* biblioteke također omogućuje i sinkronizaciju podataka u realnom vremenu. Firebase cloud vraća podatke u json formatu.

|  |  |
| --- | --- |
| **Varijable** | **Opis** |
| url | Sadrži url putanju do pohranjenih podataka |
| ref | Sadrži Firebase konstruktor s *url* varijablom, služi kao Firebase referenca |
| data | Sadrži sinkronizirane podatke |
| q | Sadrži vrijednost koja omogućava asinkrono izvršavanje metoda |
| **Metode** | **Opis** |
| getData() | Metoda vraća kolekciju podataka u obliku *key:value* |

**LocalizationService**

LocalizationService je servis za lokalizaciju aplikacije, tj. da bude dostupna na više jezika. Servis uz pomoću cordova plugina ([*Apache Globalization Plugin*](http://docs.phonegap.com/en/3.0.0/cordova_globalization_globalization.md.html#Globalization)) saznaje koji jezik se koristi na smartphone.

|  |  |  |
| --- | --- | --- |
| **Varijable** | **Opis** | |
| q | Sadrži vrijednost koja omogućava asinkrono izvršavanje metoda | |
| **Metode** | | **Opis** |
| getLanguage() | | Metoda vraća odabrani jezik na mobitelu. |
| setLanguage(lang) | | Metoda kao ulazni parametar zaprima odabrani jezik na mobitelu u obliku stringa (tipa 'hr', 'en') i pohranjuje navedenu vrijednost u lokalnu bazu.  Koristi *setLanguage(lang)* metodu iz *LocalStorageService* servisa. |

**LocalStorageService**

LocalStorageService je servis za lokalnu pohranu podataka koristeći localStorage. LocalStorage omogućuje NoSQL pohranu podataka u obliku 'key:value'.

|  |  |
| --- | --- |
| **Metode** | **Opis** |
| setDat(value) | Metoda za pohranjivanje podataka u *localStorage*, ulazni parametar su podaci preuzeti s Firebase clouda. |
| getData() | Metoda vraća preuzete podatke s Firebase clouda iz *localStorage*. |
| setContacts(value) | Metoda za pohranjivanje podataka o 'kontaktima' u *localStorage*, ulazni parametar su podaci o kontaktima. |
| getContacts() | Metoda za dohvaćanje podataka o 'kontaktima' iz *localStorage*. |
| setRenaissance(value) | Metoda za pohranjivanje podataka o 'renesansi' u *localStorage*, ulazni parametar su podaci o renesansi. |
| getRenaissance() | Metoda za dohvaćanje podataka o 'renesansi' iz *localStorage*. |
| setEvents(value) | Metoda za pohranjivanje podataka o rasporedu programa u *localStorage*, ulazni parametar su podaci o rasporedu programa. |
| getEvents() | Metoda za dohvaćanje podataka o 'rasporedu programa' iz *localStorage*. |
| setLanguage(value) | Metoda za pohranjivanje podataka o jeziku na mobitelu u *localStorage*, ulazni parametar su podaci o jeziku korištenja. |
| getLanguage() | Metoda za dohvaćanje podataka o jeziku na mobitelu iz *localStorage*. |

**DataService**

DataService je servis za manipulaciju podacima. Ovaj servis je nastao iz razloga jer se u nekim kontrolerima ponavljao isti kod. (prolazako po podacima kroz for each petlju, push u polje i slično.)

|  |  |  |
| --- | --- | --- |
| Varijable | Opis | |
| data | Sadrži podatke koje vraća metoda. | |
| eventsData | Sadrži podatke o eventima za određeni dan. | |
| Metode | | Opis |
| getData(retrievedData) | | Metoda vraća polje podataka. Ulazni parametar su podaci koje je potrebno organizirati i staviti u polje. |
| getEventsData(retrievedData,day) | | Metoda vraća polje podataka. Ulazni parametar su podaci koje je potrebno organizirati i staviti u polje, te sljedeći parametar sadrži podatak o kojem danu renesansnog festivala se radi. |
| getEventsDetailsData(retrievedData) | | Metoda vraća polje podataka. Ulazni parametar su podaci koje je potrebno organizirati i staviti u polje. |

**Starter**

Starter je naziv modula koji sadrži sve ostale kreirane module. Starter modul sadrži dvije funkcije run() i config(). To su funkcije koje se izvršavaju tijekom 'bootstrap' procesa, odnosno u pozadini i brinu se da sve funkcionira.

|  |  |
| --- | --- |
| **Funkcije** | **Opis** |
| run() | Nakon što je platforma spremna, u pozadini se provjerava da li postoje podaci u *localStoragu*, ako ne postoje provjerava se *odabrani jezik korištenja* na mobitelu kako bi se preuzeli podaci prilagođeni tom jeziku. Zatim provjerava se da li postoji *internet konekcija* (pomoću *apache.cordova.network* plugina), ako postoji *preuzimaju se podaci s clouda i pohranjuju u localStorage*, a ako nema internetske veze upozorava se korisnika da uključi (bitno je tijekom prvog korištenja da bude uključen internet). U slučaju da postoje podaci i ako su podaci na cloudu izmijenjeni dolazi do *sinkronizacije podataka*.  Koristi sljedeće metode:  *LocalStorageService*: getData(), getLanguage(), setData(val), setContacts(val), setRennaisance(val), setEvents(val)  *LocalizationService*: getLanguage(), setLanguage()  *FirebaseService*: getData() |
| config() | Ovdje su konfigurirani state-ovi/route, gdje je navedeno kada se dođe na određenu putanju koji html template prikazati , koji kontroler koristiti i slično.  Također ovdje su konfigurirani stringovi (poput strings.xml u androidu), ujedno je konfigurirano kako se ti stringovi moraju prikazati na određenom jeziku mobitela. |

**ContactCtrl**

ContactCtrl je kontroler koji služi za prikaz kontakt podataka o Turističkoj agenciji grada Koprivnica.

|  |  |
| --- | --- |
| **Varijable** | **Opis** |
| contactsData | Sadrži kontakt podatke koji su dobiveni pomoću metode *getContacts()* iz *LocalStorageService* servisa. |
| $scope.contacts | Sadrži organizirane kontakt podatke dobivene pomoću metode *getData(contactsData)* iz *DataService* servisa. Podaci iz te varijable se prikazuju u *pogledu* (View). |
| **Metode** | **Opis** |
| Init() | Metoda koja preuzima kontakt podatke iz *localStorage*, prosljeđuje ih metodi za organizaciju podataka *getData(contactsData)* iz *DataService* servisa, te rezultat pohranjuje u *$scope.contacts* kako bi se podaci mogli prikazati korisniku u *pogledu* (View). |

**RenaissanceCtrl**

RenaissanceCtrl je kontroler koji služi za prikaz podataka o renesansi.

|  |  |
| --- | --- |
| **Varijable** | **Opis** |
| renaissanceData | Sadrži podatke o renesansi koji su dobiveni pomoću metode *getRenaissance()* iz *LocalStorageService* servisa. |
| $scope.renaissance | Sadrži organizirane podatke o renesansi dobivene pomoću metode *getData(renaissanceData)* iz *DataService* servisa. Podaci iz te varijable se prikazuju u *pogledu* (View). |
| **Metode** | **Opis** |
| Init() | Metoda koja preuzima podatke o renesansi iz *localStorage*, prosljeđuje ih metodi za organizaciju podataka *getData(*renaissanceData*)* iz *DataService* servisa, te rezultat pohranjuje u *$scope.*renaissance kako bi se podaci mogli prikazati korisniku u *pogledu* (View). |

**EventsCtrl**

EventsCtrl je kontroler koji služi za prikaz informacija o rasporedu programa renesansnog festivala.

|  |  |
| --- | --- |
| **Varijable** | **Opis** |
| eventsData | Sadrži podatke o rasporedu programa koji su dobiveni pomoću metode *getEvents()* iz *LocalStorageService* servisa. |
| $scope.day | Sadrži podatak o kojem se točno danu renesansnog festivala radi. Taj podatak se dobiva iz angular *$stateParams* servisa koji uzima parametar iz url putanje. |
| $scope.events | Sadrži organizirane podatke o rasporedu programa dobivene pomoću metode *getEventsData(eventsData, $stateParams.day)* iz *DataService* servisa. Podaci iz te varijable se prikazuju u *pogledu* (View). |
| **Metode** | **Opis** |
| Init() | Metoda koja preuzima podatke o rasporedu programa iz *localStorage*, prosljeđuje ih metodi za organizaciju podataka *getEventsData(eventsData, $stateParams.day)* iz *DataService* servisa, te rezultat pohranjuje u *$scope.* events kako bi se podaci mogli prikazati korisniku u *pogledu* (View). |

**EventsDetailsCtrl**

EventsDetailsCtrl je kontroler koji služi za prikaz dodatnih informacija, detalja o određenom programu renesansnog festivala ako postoje.

|  |  |
| --- | --- |
| **Varijable** | **Opis** |
| $scope.detail | Sadrži organizirane podatke koje sadrže detalje o određenom programu, dobivene pomoću metode *getEventsDetailsData(details)* iz *DataService* servisa. Podaci iz te varijable se prikazuju u *pogledu* (View) |
| $scope.description | Nasljeđeni podatak iz *EventsCtrl* kontrolera, sadrži opis određenog programa. |
| $scope.img | Nasljeđeni podatak iz *EventsCtrl* kontrolera, sadrži url putanju servisa gdje je pohranjena slika. |
| $scope.img2 | Nasljeđeni podatak iz *EventsCtrl* kontrolera, sadrži url putanju servisa gdje je pohranjena slika 2. |
| $scope.time | Nasljeđeni podatak iz *EventsCtrl* kontrolera, sadrži vrijeme određenog programa. |
| $scope.title | Nasljeđeni podatak iz *EventsCtrl* kontrolera, sadrži naslov određenog programa. |
| details | Nasljeđeni podatak iz *EventsCtrl* kontrolera, sadrži detalje određenog programa. |
| **Metode** | **Opis** |
| Init() | Metoda koja preuzima nasljeđene podatke iz *EventsCtrl* kontrolera, prosljeđuje varijablu *details* metodi za organizaciju podataka *getEventsDetailsData(details)* iz *DataService* servisa, te rezultat pohranjuje u *$scope.* detail kako bi se podaci mogli prikazati korisniku u *pogledu* (View). |