

Predmet:  
Mobilno poslovanje

Seminarski rad

Tema:  
Vremenska prognoza IOS app

Student:

Dejan Čolić 2013/0260

Beograd, Jun 2019

# Sadržaj

Sadržaj

[1. Sadržaj 2](#_Toc12476427)

[2. Rezime 3](#_Toc12476428)

[3. Uvod u aplikaciju 4](#_Toc12476429)

[4. Target server : Dark Sky 6](#_Toc12476430)

[5. Pregled strukture aplikacije 7](#_Toc12476431)

[5.1. Weather.swift 7](#_Toc12476432)

[5.2. Main Storyboard 11](#_Toc12476433)

[5.3. WeatherTableViewControler 12](#_Toc12476434)

[5.4. App delegate i assets 14](#_Toc12476435)

[6. Zaključak 15](#_Toc12476436)

# Rezime

’’ Vremenska prognoza’’ je jednostavna IOS aplikacija koja prima naziv grada i prikazuje vremensku prognozu za taj grad u narednih 7 dana. Cilj ovog projekta bio je prikazivanje rada sa web API-jem bez korišćenja framework-ova koji taj proces olakšavaju. Autor ovde napominje da je mnogo lakše, i preporučljivo, korišćenje SwiftyJson-a ili AlamoFire framework-a, ali ovde smo se bavili postupnom, korak po korak, komunikacijom sa web servisom, radi dubljeg razumevanja samog procesa.

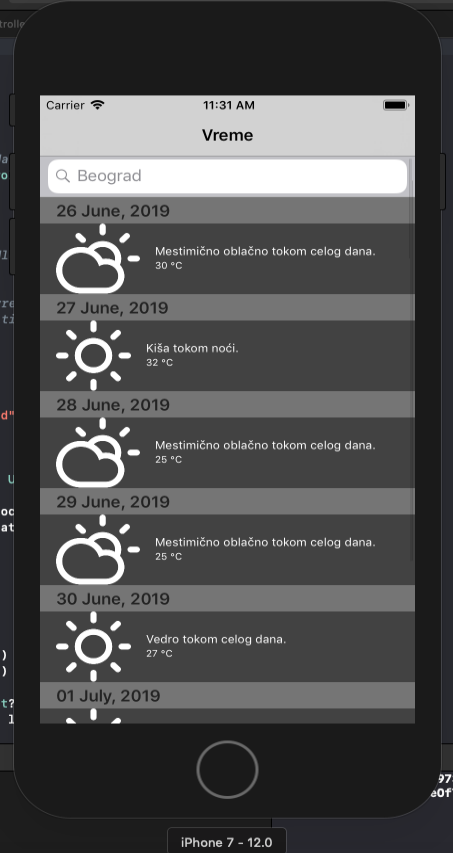
U ovom dokumentu prikazana je struktura aplikacije kroz detaljnjije objašnjavanje klasa koje je čine. Grafički interfejs aplikacije je veoma jednostavan i zato nećemo mnogo govoriti o njemu, već ćemo najveći deo posvetiti logici iza same aplikacije.

Aplikacija je pravljena u XCODE 10 okruženju, u programskom jeziku Swift 4. Namenjena je iPhone telefonima, i ima smanjene preformanse na tabletima.

# Uvod u aplikaciju

U ovom delu ćemo ukratko opisati aplikaciju Vremenska prognoza.

Pri pokretanju aplikacije pojavljuje se sledeći ekran

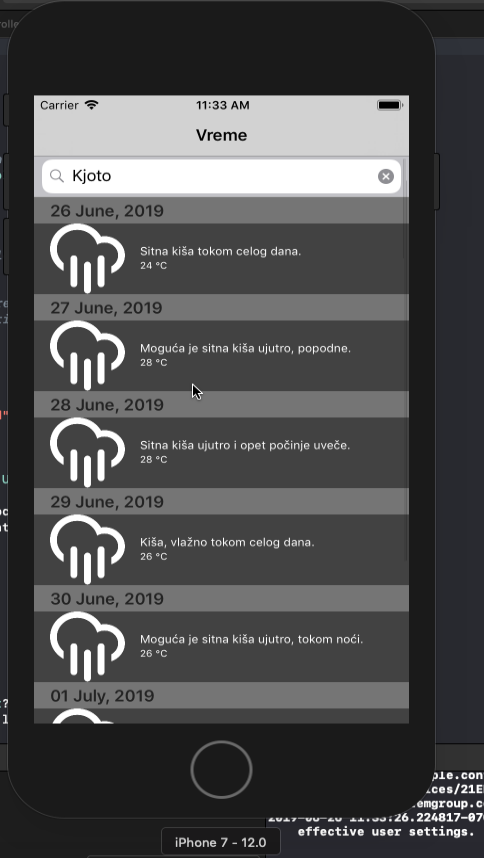


Slika 1

Na njemu vidimo jednostavan GUI koji se sastoji od table view-a , i search bara. Inicijalno, search bar je popunjen placeholder gradom (Beograd), i prikazuje podatke za taj grad odmah po pokretanju aplikacije. Ispod search bara se nalazi 8 ćelija, prva koja nosi datum tog dana kada se aplikacija koristi, a ostale ćelije prikazuju narednih 7 dana.

Ćelija se sastoji od opisa vremena za dati dan, maksimalne dnevne temperature u stepenima celzijusa, i sličice koja odgovara vremenu za taj dan.

Da bi dobili prognozu za neki drugi dan, u search bar unosimo naziv grada. Na primer, zanima nas vreme narednih 7 dana u Kjotu:



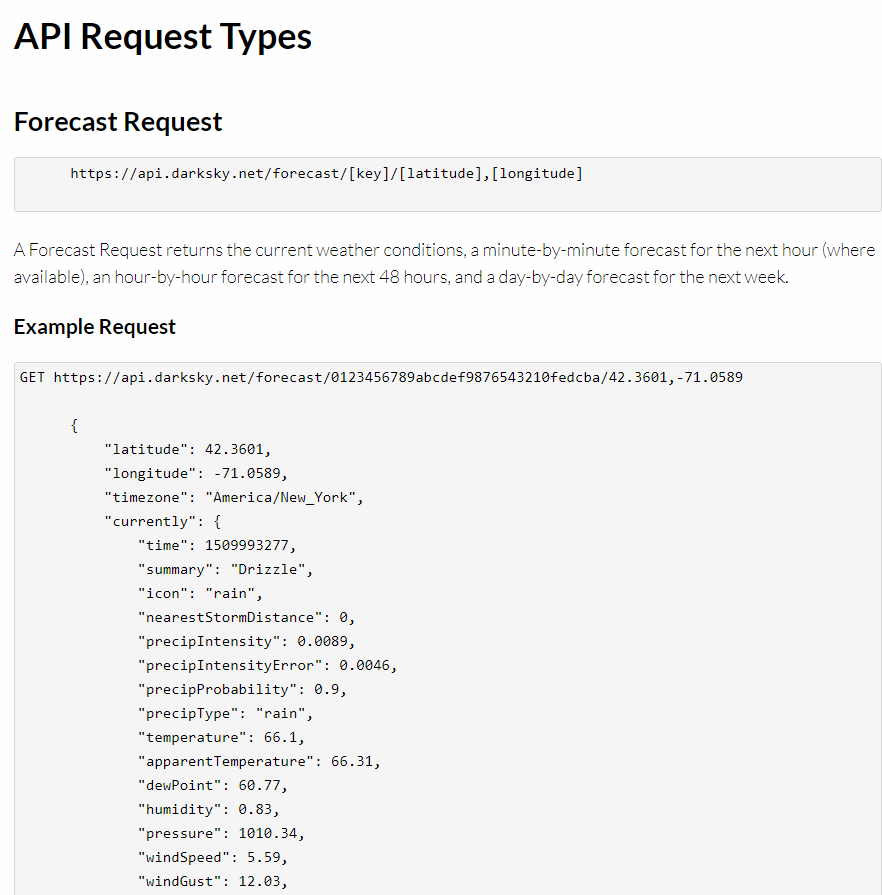
Slika 2

Aplikacija nam prikazuje odgovarajuće podatke za Kjoto , za narednih sedam dana.

# Target server : Dark Sky

Podatke o vremenu preuzimamo sa sajta Dark Sky, koji ima veoma pristupačan api.

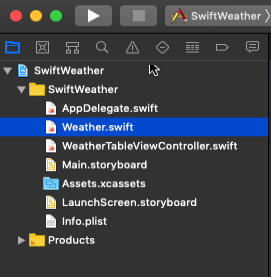
Potrebno je napraviti nalog na sajtu i dobijete pristup tajnom kodu koji koristite za HTTP request.



Slika 3

Kao što vidimo, da bi smo poslali zahtev api-u treba nam url sajta, ključ, i lokacija koja se sastoji od dvodimenzionalne koordinate.

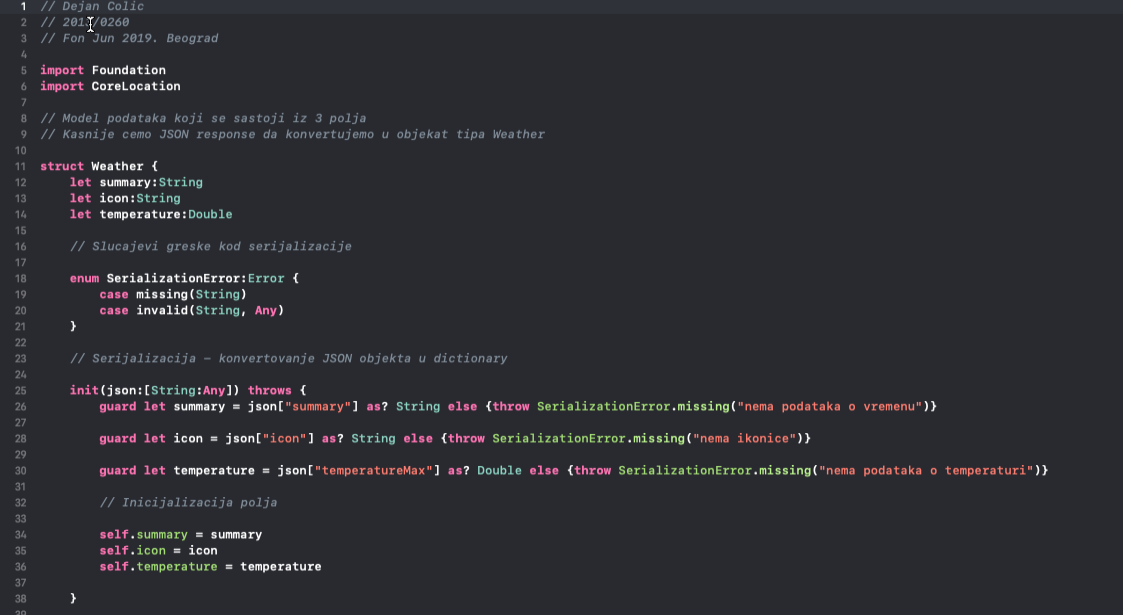
# Pregled strukture aplikacije



Slika 4

## Weather.swift

U ovom fajlu se odvija slanje request-a Dark Sky end point-u, primanje response-a , serijalizacija JSON objekata, i formiranje niza objekata tipa weather koje dalje koristimo u aplikaciji.



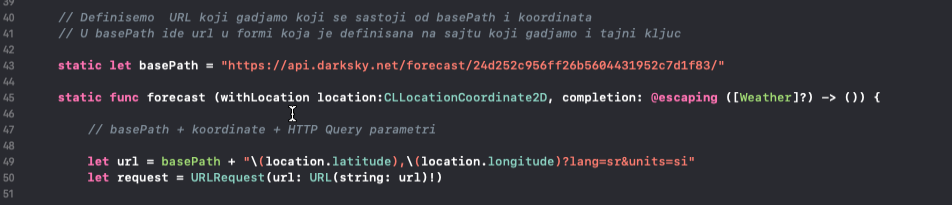
Slika 5

Prvo pravimo model podataka sa kojim ćemo raditi, tačnije strukturu Weather koja ima 3 polja: summary, icon i temperature. To su polja koja ćemo prikazivati u ćelijama aplikacije.

U drugom koraku kreiramo enum SerializationError preko koga definišemo slučajeve koji se obrađuju ako dođe do greške prilikom serijalizacije.

Serijalizacije je konvertovanje JSON objekata u dictionary tip podataka koji nama odgovara. Dictionary je tip koji se sastoji od ključa i vrednosti.

Nakon toga definisemo putanju koju gađamo

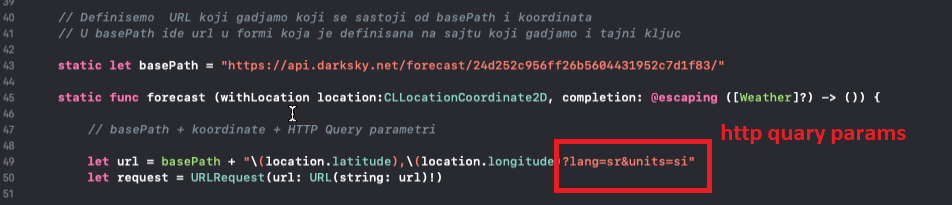


Slika 6

Vidimo da se konstanta request sastoji od URL-a sajta, našeg tajnog ključa, koordinata u vidu latitude i longitude. Request formulišemo po ugledu na primer koji nađemo na sajtu

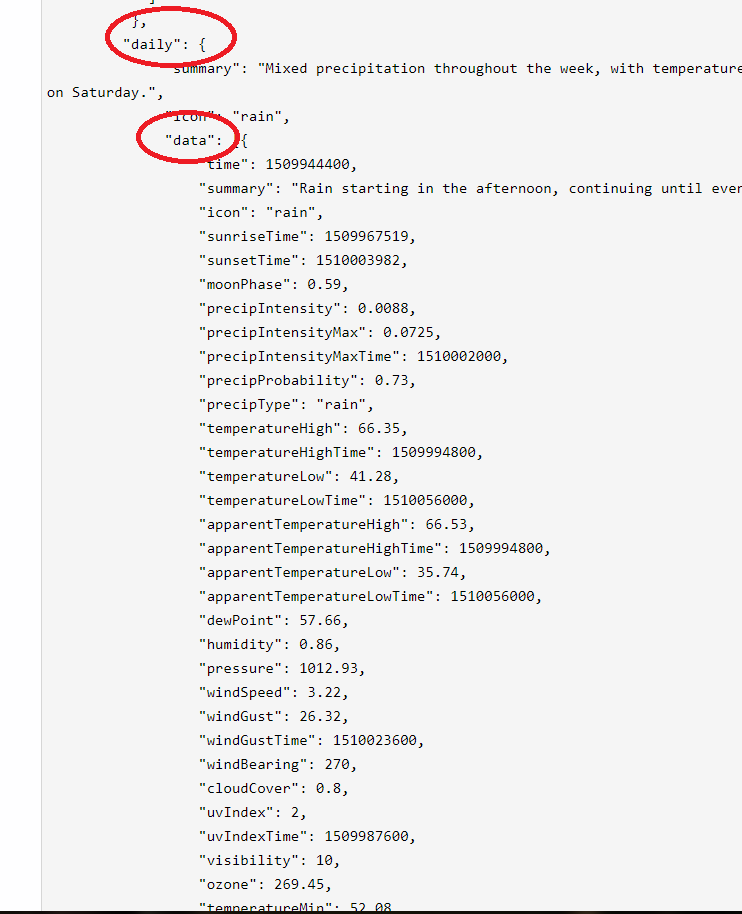
Slika 7

Da bi dobili rezultate na srpskom jeziku i u SI jedinicama ( celzijus umesto farenhajta i slično) na kraj url-a dodajemo HTTP quary parametre



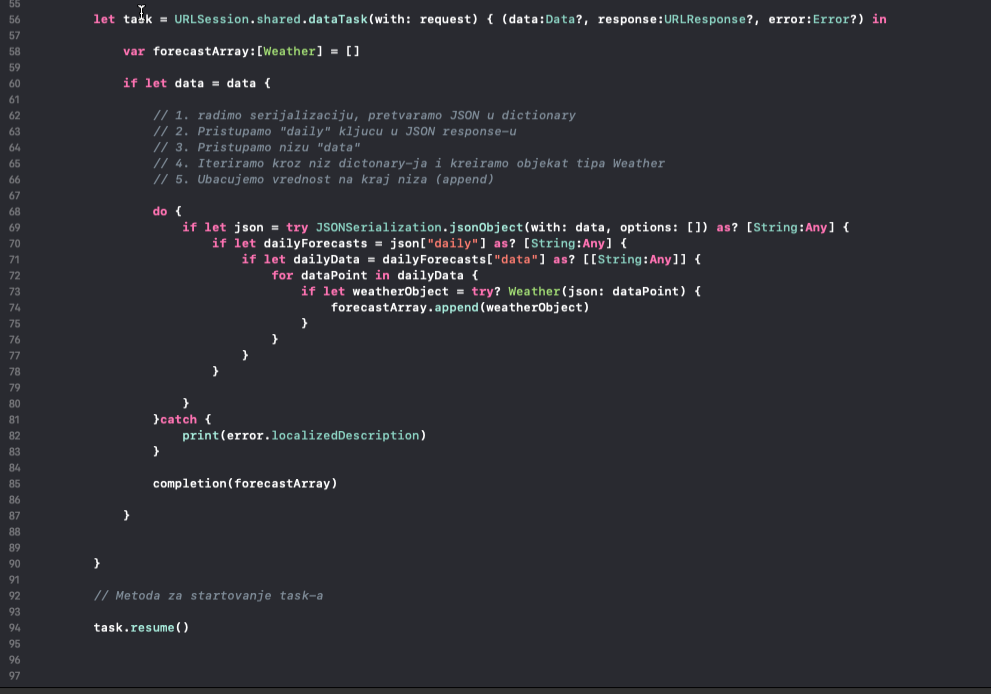
Slika 8

Sada gledamo kako izgleda odgovor Dark Sky-a da bi znali kako da definišemo serijalizaciju:



Slika 9

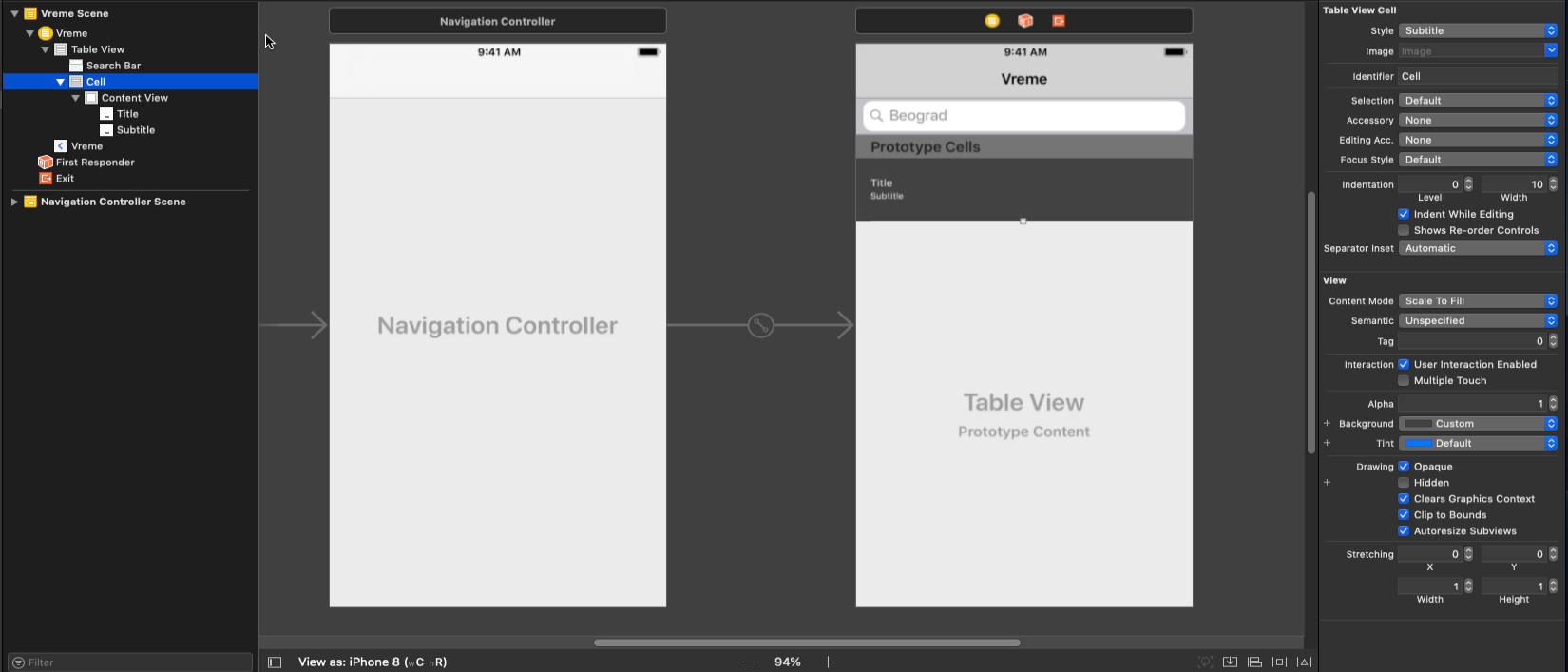
Interesuje nas dictionary ’’ daily ’’ zato što radimo dnevnu prognozu, i u njemu ’’ data ’’ skup dictionary-ja od kojih nama trebaju samo summary, icon i temperatureMax, koji odgovaraju poljima u strukturi Weather.



Slika 10

## Main Storyboard

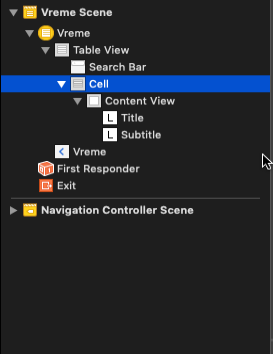
Main storyboard se sastoji od Navigation controlera koji deluje kao entry point za aplikaciju, i table view-a koji predstavlja glavni deo nase aplikacije.



Slika 11

Na table view smo stavili search bar da bi korisnik mogao da unosi potreban input.

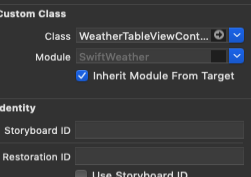
Hijerarhija elemenata na GUI-u izgleda ovako:



Slika 12

## WeatherTableViewControler

Nas ekran je povezan sa klasom WeatherTableViewControler da bi u njoj mogli da pišemo kod koji se odnosi na elemente GUI-a.



Slika 13

Stavljamo outlet za search bar, pravimo promenjivu koja će da prima rezultate serijalizacije, postavljamo firstResponder (tastatura da se vrati nakon unosa), i postavljamo početnu vrednost grada prilikom pokretanja aplikacija na Beograd.



Slika 14

Importujemo potrebne biblioteke i radimo geokodiranje



Slika 15

Dalje, formatiramo tableView tako da se sastoji od onoliko section-a koliko ima elemenata niza tipa Weather, i postavljamo da svaki section ima jedan row.

Popunjavamo sekcije formatiranim datumom i dodeljujemo vrednosti poljima objekta Weather.



Slika 16

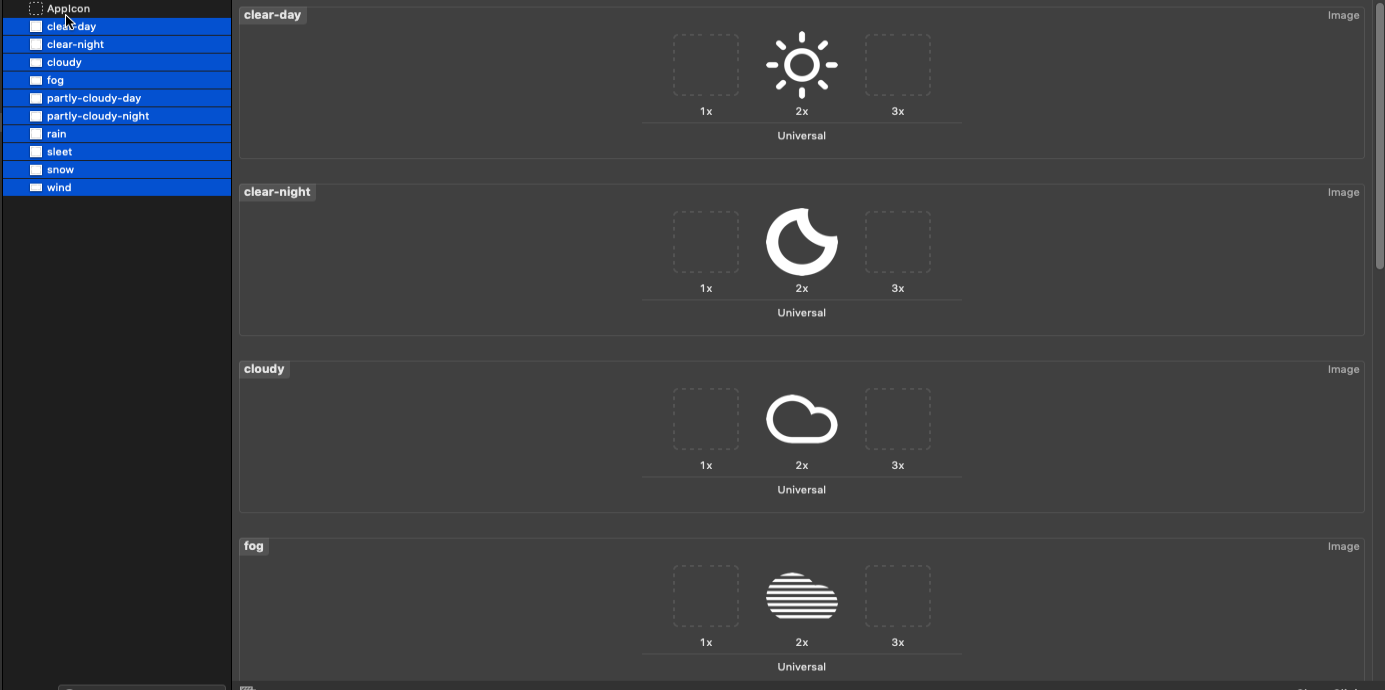
## App delegate i assets

App delegate je jedan od najvažnijih elemenata projekta i u njemu je definisano ponašanje aplikacije tokom različitih stanja kroz koja aplikacija prolazi



Slika 17

U fajlu assets se nalaze sličice koje koristimo za prikaz vremenskih prilika. Sve sličice su veličine 2x.



Slika 18

# Zaključak

Ovaj jednostavan projekat je prikazao kako se radi sa web API-em, kako se šalje zahtev servisu, kako se prima response i kako se parsira JSON u dictionary tip podataka.

Postoje mnogo framework-ovi koji olakšavaju čitavu proceduru, a među njima bih spomenuo AlamoFire, libary koji služi za elegantno obrađivanje HTTP konekcija, SwiftyJSON, libary za lako serijalizovanje JSON-a u Swift objekte. Postoje čak i biblioteke za povezivanje ova dva spomenuta. Takođe vredi spomenuti i Postman aplikaciju, koja omogućava testiranje funkcionalnosti apija bez pisanja mnoštva koda u samoj aplikaciji.