MÁV Plus

Programozói dokumentáció

Készítette:
Pfemeter Márton, Berente Bálint

Összefoglaló

Az alkalmazás célja a hivatalos MÁV applikáció funkcióinak biztosítása egy felhasználóbarátabb felületen. Ehhez többek között ugyanazt a szerver backend-et használja az app mint a hivatalos MÁV app. A fejlesztés iOS 16-ra történt, Swift programozói nyelvet, illetve főként SwiftUI UI keretrendszert használva, egy-két UIKit-et használó nézettel.

Fejlesztési Tapasztalatok

Általánosságban

Motiváló volt látni, ahogy az alkalmazás egyre gyorsabb, stabilabb és szebb lett az idők során. Mivel azzal az alapcéllal indult el a projekt, hogy a hivatalos alkalmazásnak egy újragondolt, letisztult és használható változatát készítsük el, úgy gondoljuk, sikerült lépéseket tenni a jó irányba. Habár a funkcionalitás sose lesz kész, hiszen új dolgok mindig akadnak, jönnek létre, használata már az első pillanatoktól kezdve élmény.

Térkép fejlesztése

A térképet először SwiftUI-t használva szerettük volna megvalósítani. Kézenfekvő megoldásnak tűnt, hiszen az alkalmazás többi része is SwiftUI-t használ, és itt is kész Map osztály, melyet lehet használni ilyen célra.
Sajnos azonban gyorsan kiderült, hogy a SwiftUI-os Map közel sem olyan gazdag funkcionalitásban mint a UIKit-es megfelelője. Előbbiből hiányzik többek között egy adott helyzet pozíció változásának animációja, a képernyőn látható térképrészleten lévő helyekre való szűrés, illetve az egymást fedő ikonok összevonása (clustering).
Ezek közül sikerült magunk megvalósítani a képernyőn látható térképrészletre való szűrést, illetve hogy az egymást fedő ikonok közül mindig csak egy maradjon meg. Ezek a megvalósítások viszont SwiftUI-al használva olyan súlyos teljesítményvonzattal rendelkeztek, hogy a térképen való nézelődést teljesen használhatatlanná tették. Ezért döntöttünk végül is a UIKit-es megvalósítás mellett.

Adatok forrása

MÁV API

A legfontosabb adatforrás talán. Az új Elvira / MÁV Jegyvásárlás oldal API-ját használja (https://jegy.mav.hu).
Az API végpontokat az apik.pdf tartalmazza.
Az API hívások Alamofire segítségével a Model/ApiRequests.swift fájlban találhatóak.

Térkép adatok

Az állomások pozíciói a MÁV és a BKK GTFS állományából lettek összeválogatva, majd név alapján hozzárendelve az állomásokhoz. Ennek eredményeként a legtöbb állomás (1000+) pozíciója ismert, de nem az összesé. Ezek egy Property List (.plist) fájlban vannak tárolva.
A vonatok pozíciói a hivatalos MÁV App által használt API, ami egyben küldi vissza az összes pozíciót, így abból kell vonat azonosító alapján keresni. Itt viszont azt kell megnézni, hogy a helymeghatározás API által visszaadott VonatID-ben megtalálható-e a fő API által adott vonat ID (mert össze van konkatenálva egy “_dátum” szöveggel)

Hírfolyam

A hírfolyam nézetben a hivatalos MÁVInform oldal tartalma jelenik meg, annak RSS feed-jét feldolgozva (https://www.mavcsoport.hu/mavinform/rss.xml). Ez a fő RSS feed a hírfolyam tartalmának címjeit, illetve elérési útvonalait (URL) tartalmazza. A különböző híreket már nem dolgozza fel az alkalmazás, azok egy böngészőablakban tekinthetők meg.

Tárolt adatok

Kedvenc állomásokat, legutóbbi kereséseket és útvonal előzményeket CoreData segítségével tárolunk. Erről bővebben az Architektúra részén lesz szó.

Architektúra

Model

Alamofire könyvtár segítségével a Model/ApiRequests.swift fájlban számos API hívást intéző függvény található. Nagy része ezeknek egyszer hívandó (pl. Állomások listája) az alkalmazás betöltésekor, ez az ApiRepository feladata.

Repository

Az MVVM mellé 3 Repository (tároló) tartozik. Összességében ezeknek a feladata kezelni az alsó model réteget, tárhelyet. Ezek singleton osztályok, és a shared static változóban érhető el az egyetlen példány.

ApiRepository

Az alkalmazás fő tárolója. Inicializálásnál betölti az alap adatokat az API-ról (pl. állomáslista). Emellett az egyes funkciók API hívásait is proxyzza.

StoreRepository

CoreData használatáért felelős tároló. Betölti a már tárolt adatokat, szinkronban elérhetővé teszi a VM-ek számára, miközben a mentési/törlési funkciókat is ellátja.

RssRepository

Az RSS feed feldolgozását végző, beépített XMLParserDelegate protokollt megvalósító osztály. Minden frissítésre (update metódus) az “Adatok forrása” című részben jelölt RSS feed elérési útvonaláról lekéri a hírfolyamot, majd feldolgozza azt az AlertsViewModel osztály által támogatott (és így közvetlenül megjeleníthető) formátumba.
Az RSS feed-ből annak feldolgozása során lényegében egyedül az egyes hírelemek címét, illetve elérési útvonalát (URL) tartja meg.

ViewModel

A Model-View-ViewModel (MVVM) mintát alkalmazva hoztunk létre minden nézetnek (View) egy ViewModel osztályt. E osztályoknak célja a megjelenítendő adatok kezelése. Amennyiben szorosan a nézethez tartozik a megjelenítendő adat, akkor közvetlenül a ViewModel tárolja azt (például útvonaltervezés utolsó lépésénél a kiválasztott utazási ajánlat - Offer - objektum), egyéb esetben pedig a megfelelő Repository objektummal kommunikál a ViewModel. Utóbbi esetben Combine keretrendszert használva a Repository-k értesítik a feliratkozott ViewModel-eket a változásokról. Ez alól kivétel a térkép (MapViewModel), mely adott időközönként (5 másodperc) frissíti a tárolt helyadatokat.

View

Ezek a felhasználói felületet megvalósító osztályok. SwiftUI-t használ szinte mindegyik nézet, illetve nézet komponens, kivétel ez alól egyedül a böngészőablak (WebView.swift, WebKit miatt) és a térkép (MapKitMap.swift, MapKit miatt) melyek UIKit-et használnak. Ezek az osztályok a saját (név szerinti) ViewModel-jükre hivatkoznak vagy tartalmazzák azt, abból nyerve minden megjelenítendő adatot.

Külső könyvtárak

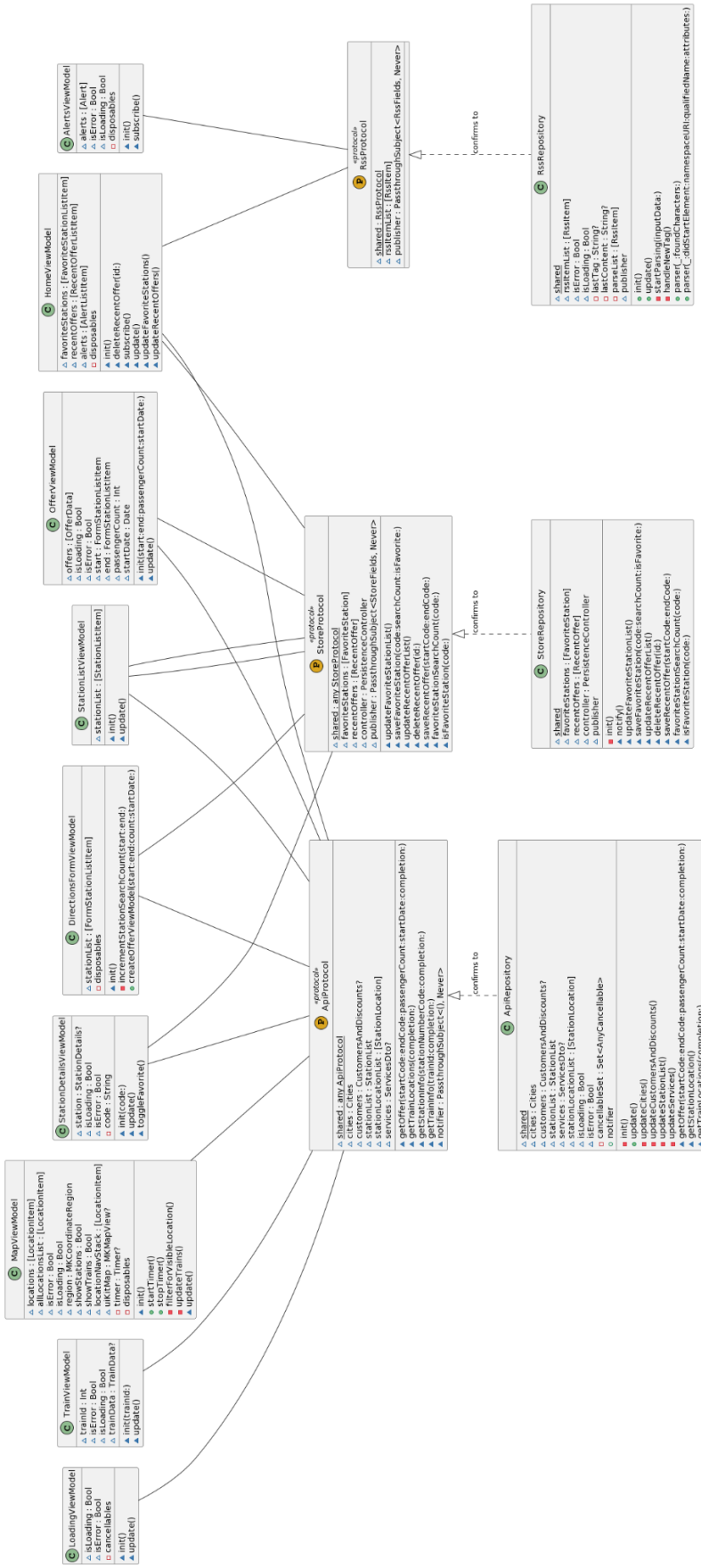
CocoaPods gyorstalpaló

A CocoaPods egy Ruby alapú csomagkezelő, melyet az XCode projektek is használnak. Mivel a projekt tartalmaz egy külső könyvtárat, az xcodeproj helyett az xcworkspace fájlt kell megnyitni. Másképpen nem fog lefordulni a projekt. Mindenekelőtt azonban telepíteni kell a pod-okat. Ezt a root-ba lépve “pod install” parancs kiadásával tehetjük meg (feltételezve, hogy a Cocoapods telepítve van a számítógépen)

Alamofire

Az Alamofire egy ipari sztenderd HTTP hívások használatához. Sokkal tömörebb és átláthatóbb kódot eredményez a használata, a JS-ben használt Axios könyvtárhoz tudnám hasonlítani. A körülményesen használható URLSession helyett használtuk ezt az implementációt.

Osztálydiagram



API Végpontok

Általános adatok - MÁV API

Az API-t a hivatalos és megújult https://jegy.mav.hu címen találtuk. Ez az API-t régebben NodeJS könytárnak is meg lett írva, de az API módosításokat nem követte le, és a típusok is még hiányoznak: https://github.com/berenteb/mav-api
Az erőforrást az oldal használatával tártuk fel és Swift típusokra képeztük le a következő oldal segítségével: https://quicktype.io/
A MÁV API-ja egészen sok dolgot támogat, el tudunk vele jutni akár a jegyek árához is, keresési szűréseket alkalmazhatunk és útvonalat is tervezhetünk, stb.
Nagy vonalakban a következő API végpontokat találtuk, két csoportjuk van. Vannak, amelyek jegyvásárláshoz köthetőek és amelyek információszerzéshez:

OfferRequestApi/GetOfferRequest

A legfontosabb végpont, amely a kiinduló állomásból, érkezési állomásból, indulási időből és szűrési, utas információkból készít egy “ajánlatot”, azaz útvonaltervet, árat és részleteket.

OfferRequestApi/GetCustomersAndDiscounts

Az előző végponthoz jön jól, lekéri az utas típusokat és kedvezményeket.

OfferRequestApi/GetServicesAndSearchServices

Ez kicsit haszontalanabb végpont, keresési feltételeket ad meg, pl 1., 2. kocsiosztály, helyfoglalás, stb.

OfferRequestApi/GetCitiesAndPostalCodes

Városok és irányítószámok végpontja.

OfferRequestApi/GetStationList

Ez hasznos, megállóhelyek listáját adja vissza, melyben benne vannak azok a kódok, amelyek a menetrendhez és útvonaltervezőhöz kellenek. Koordináták viszont nincsenek benne.

InformationApi/GetTimetable

Itt kettő fajta lekérés van, StationInfo és TrainInfo. Az első állomás indulási tábláját, menetrendjét adja vissza adott időpillanattól kezdve (ISO Dátum), a másik egy járat megállóit, adatait és a megállókhoz tartozó indulási időt.

InformationApi/GetWarningMessages

Weboldalon megjelenő értesítések, tapasztalat szerinti példák: nem működik a bankkártyás fizetés, maszkot kell hordani, stb.

Ezeken kívül van még 1-2 végpont, amelyek tisztázatlanok, vagy nincs rájuk szükségünk egyelőre. Vannak például árfolyamváltó, megye lista, dokumentum lista (ÁSZF és társai), közterület jellegek stb.
Elővárosi vonatokat a BKK Futár is tud kezelni, de úgy döntöttünk egy forrást használunk erre.

Vonat pozíció

Ez trükkösebb volt régen, mert XML-t használt a http://vonatinfo.mav-start.hu/, de áttértek JSON- ra, bár a HTTPS-ig már nem jutottak. Lejön a JSON, benne az összes vonat, ami a térképen szerepel. POST lekéréssel érjük el ezt, felküldeni pedig egy {“a":"TRAINS","jo": {"history":false,"id":false}} objektumot kell. Szerencsére, ha az id mezőben a false helyett beírunk egy vonat ID-t, akkor csak egyet küld le. Ezek azonban nem tűnnek kompatibilisnek az előző API kódjaival.
Ami viszont működik egy jó ideje, az a Mobil alkalmazásuk végpontja, ami trükkösebb, plusz ki kell szedni az egészből a nekünk érdekeset. Végpont url-je: http://vim.mav-start.hu/VIM/PR/ 20210220/MobileService.svc/rest/GetVonatok , ebben pedig nem ID, hanem vonatszám alapján lehet keresni. Itt visszajön a koordináta, ami alapján rá lehet szúrni a térképre a szerelvényt. Amire figyelni kell a lekérésnél, hogy be kell állítani jól a header-öket, és egy kamu UAID body mezőt is le kell küldeni, ami egyfajta session ID a mobil alkalmazáshoz, de ha nem változik az nem baj.

Hírek

Ez egy egyszerű RSS olvasás: https://www.mavcsoport.hu/mavinform/rss.xml, ami célzottan a MÁVINFORM híreit adja vissza.