Miskolci Egyetem

Gépészmérnöki és Informatikai Kar

Általános Informatikai Tanszék

**Adatmegosztó alkalmazás**

**Komplex feladat**

**Készítette**:

**Név**: Bodnár Renátó

**Neptunkód**: A4R0OD

**Szak**: Mérnök Informatikus Bsc

Webfejlesztés szakirány

Tartalomjegyzék

[Bevezetés 2](#_Toc40824308)

[Alapötlet 2](#_Toc40824309)

[Elvárások, működés 2](#_Toc40824310)

[Irodalom feldolgozás 4](#_Toc40824311)

[Okostelefon 4](#_Toc40824312)

[Android 5](#_Toc40824313)

[iOS 5](#_Toc40824314)

[Mobil fejlesztés 6](#_Toc40824315)

[Flutter 7](#_Toc40824316)

[GDPR 7](#_Toc40824317)

[Mi is ez? 7](#_Toc40824318)

[Személyes adat 8](#_Toc40824319)

[Adatkezelő, és feldolgozó 8](#_Toc40824320)

[Adatkezelés 8](#_Toc40824321)

[Adatkezelési tájékoztató 8](#_Toc40824322)

[Rendelkezésre álló technikák 10](#_Toc40824323)

[Flutter, Dart 10](#_Toc40824324)

[Clean Architecture 12](#_Toc40824325)

[Firebase 14](#_Toc40824326)

# Bevezetés

## Alapötlet

A feladat alapötlete a jelenleg is használatban lévő ügyfélkezelő megoldások forradalmasítása. Vegyük példának egy kormányablak működését, amikor a kliens megérkezik a helyszínre, egy jegykiadó automatánál meg kell adnia, hogy milyen ügyben szeretne eljárni. Ebben az esetben nem történik más, minthogy a rendszer felregisztrál egy megadott sorszámot, a megadott kérdéskör várakozó listájára, és ezt kinyomtatja az igénylőnek. Ekkor várakozásba kezd, és amint az ügyfélhívó monitoron megjelenik a sorszáma, bemehet az ügyintézőhöz. És itt azonosítania kell magát az adataival.

Ezt a procedúrát meg lehetne szüntetni, és egyben rengeteg időt megspórolni az ügyintézőnek, illetve a kliensnek egyaránt. Erre egy mobiltelefonos alkalmazás lehet a megoldás.

## Elvárások, működés

Ennek az applikációnak képesnek kell lennie a felhasználókat beregisztrálni, be- illetve kijelentkeztetni, és egyedi azonosítót rendelni hozzájuk. Ezen azonosító szerint a megadott adatokat eltárolni, biztonságos módon. Az eltárolni kívánt adatok csoportja egyéni, mindenki azokat mentheti el, amiket szeretne. Ugyanakkor az alkalmazásnak kell legyen egy szolgáltatói oldala is, amit az irodák, üzletek tudnak használni. Elsődleges funkciója egy QR kód létrehozása kell legyen, ami adatszinten képes továbbítani a leolvasónak az alapvető információkat:

* Az intézmény neve
* Annak egyedi azonosítója
* A bejelentkezéshez szükséges adatok, és prioritásuk: kötelező vagy opcionális

Ezt a kódot lehet menteni, amit kinyomtatva ki kell ragasztani jól látható helyre. Az ügyfél érkezését követően az alkalmazáson belül leolvassa azt, és a megjelenik egy képernyő a kért adatokkal. Minden sor mellett van egy checkbox, amivel beállítható, hogy az adatot oda kívánja-e adni a felhasználó. A kötelező mezőket nem lehet állítani. Az elküldés gomba bökve az rendszer küld egy értesítést a szolgáltatónak, ami tartalmazza a kért információkat, és beteszi a várakozó sorba az ügyfelet. Ebben sorban képesnek kell lenni az adatok olvasására, illetve a jelenleg aktív vendég lezárására.

A kliens oldali applikációba integrálni kell egy QR leolvasót, ami tudja olvasni a generált kódokat, és kinyerni belőle a szükséges adatok listáját. Ezt konvertálni kell egy megjeleníthető objektumhalmazzá, ami hiányzó szöveg esetén beviteli mezőt jelent, más esetekben egy Text fieldet, illetve egy checkboxot. Az adatok előzetes megadására és eltárolására szintén lehetőséget kell adnunk a felhasználónak, hogy bármikor elvégezhesse. Ehhez egy új oldalt kell létrehoznunk, a támogatott adatokkal, amik opcionálisan kitölthetők bármikor. A módosításokat menteni, és beolvasni a már létező adatokat.

# Irodalom feldolgozás

## Okostelefon

Pontos definíciója még nem született, hogy mit is jelent az okostelefon. Általában a PC-szerű funkcionalitást nyújtó mobiltelefonokat nevezzük így. Egyesek szerint az nevezhető smartphone-nak, ami szabványosított interface-eket és platformot nyújt a fejlesztők számára. De abban mindenképpen egyetérthetünk, hogy egy hordozható személyi számítógépről beszélünk, amivel telefonálni is lehet. Alapkövetelmény, hogy a készülék internetelérésre lehetőséget adjon, akár több módon is, Wifi-n, esetleg mobilinternet előfizetésen keresztül is. A készülék operációs rendszere és hardveres elemei pedig számtalan funkcionalitású eszközzé teszik a mai okostelefonokat. Emailezhetünk, böngészhetünk az interneten, fényképezni és videózni is lehet vele, játékok, személyi asszisztens szoftverek, munkát megkönnyítő lehetőségek. Határt csak a fejlesztők elképzelései tudnak szabni a funkcionalitásnak.

A technika robbanásszerű fejlődésével párhuzamosan a mobiltelefonok is egyre jobbak lettek, ma már nem ritka, hogy egy smartphone 8-10 magos processzorral, 6-10 GB memóriával, 256 – 512 GB háttértárral legyen felszerelve. Ez a specifikáció megfelel nagyjából egy 2013 körüli közepesen erős asztali számítógépnek. Eközben a kijelzők mérete is folyamatosan nőtt, az átlagos méret 6-7” közötti érték lett. Ezen a felületen már kényelmesen lehet olvasni, írni, és videókat is nézni. Ezen tulajdonságai miatt sok háztartásban leváltotta a régebben nagy sikernek örvendő tableteket, és laptopokat is.



*1.ábra: Okostelefon eladások*

Az 1.ábrán látható, mekkora mennyiségeket adnak el ezekből a telefonokból globálisan a világban. Ezek óriási számok, gyakorlatilag letarol bármi eddigi szórakoztató ipari eszközt, vagy telefonálásra alkalmasat. Ma már nagyon ritkán találkozni olyan emberrel, akinek ne lenne a zsebében egy példány valamelyik gyártótól, és ne a smartphone-ján intézné el a dolgait, amikhez nem feltétlenül kell egy laptop nagyobb kijelzője, vagy fizikai billentyűzete.

Ezt a mennyiséget már a szoftverfejlesztők sem hagyhatták figyelmen kívül.

## Android

A Google 2008 végén dobta piacra a mobil operációs rendszerét, Android néven. Az első ezt futtató készülék a HTC G1 volt, a T-Mobile forgalmazásában. A fejlesztés célja egy olyan nyílt forráskódú rendszer létrehozása volt, ami könnyen alakítható, és amihez a fejlesztőknek lehetőségük van írni úgynevezett managed kódot. Ez virtuális gépen feldogozott, nem pedig processzorban feldolgozott kódot jelent. Az alkalmazások kivétel nélkül hozzáférnek a telefonok minden erőforrásához, így mind egyenrangúak. A rendszer monolitikus Linux kernelen alapul.

A fejlesztést az *Open Handset Alliance* nevű cég-csoportosulás végezte, melynek több, mint 50 cég a tagja, köztük 11 készülékgyártó is. Az Android készülék és gyártófüggetlenül képes hasonló felhasználói élményt nyújtani, mint az iPhone, a felhasználói pedig belépőkategóriától a csúcs készülékekig megtalálhatják a számukra megfelelő készüléket.

Jelenlegi legújabb verzió az Android 10. A folyamatos fejlesztés során próbálják lekövetni az aktuális trendeket, olyan funkciókkal, mint például a rendszerbe épített sötét mód. Erre az Amoled kijelzők terjedése miatt volt szükség. De kapunk beépített asszisztenst is, aki folyamatosan figyel, és ha kimondjuk, hogy: OK Google, akkor az ezt követő utasításainkra reagál, és végrehajtja azokat. Annyi funkció, és olyan tudás jellemzi már a rendszert, hogy listába szedni is lehetetlenség lenne, vetekszik a laptopok és személyi számítógépek tudásával.

## iOS

Az iOS az Apple Inc. saját mobil operációs rendszere, melyet kizárólag a saját készülékeikre fejlesztettek. Az első iPhone szoftver 2007 júniusában jelent meg, ekkor a neve még iPhone OS volt, jelenleg is használt elnevezését az 5-ös verziótól kezdve használták. Ekkor a rendszer még rengeteg hiányossággal küzdött, nem lehet sem háttérképet változtatni, sem pedig szöveget másolni és beilleszteni, illetve alkalmazás bolt sem létezett még. Ekkor még a rendszer az érintéses vezérlés kezelésére koncentrált, ugyanis az elsők között volt az Apple, akik fizikai billentyűzet nélküli telefont gyártott. A legnagyobb ugrást az 5-ös verzió megjelenése hozta, ekkor keresztelték át is, és az itt megkapott funkciók a mai napig meghatározó részei a rendszernek. Bemutatkozott Siri, bár ekkor még csak értesítési központként üzemelt. Megjelent az iMessage, iCloud, olvasómódot kapott a böngésző, beépítésre került a Twitter, fejlődött a levelező és fényképező alkalmazás is. Az idő haladtával egyre jobban integrálódtak a rendszerbe a közösségi média alkalmazásai, és a felkapott szoftvereket, illetve más gyártók alkalmazásait saját megoldásokkal helyettesítették. Ily formán lekerült a telefonról a Google YouTube alkalmazása is, térkép és levelező kliense is.

A rendszer zárt, és a személyre szabásra sem sok lehetőséget ad, de amit gyárilag tud, azt megbízhatóan teljesíti is. Készülékek tekintetében nincsen alternatívája, az Apple kizárólag a saját eszközein teszi lehetővé a használatát, amik általában csúcskategóriás készülékek. A Google-lel ellentétben a fejlesztés is kicsit bonyolultabb az iOS rendszerre, ugyanis az Apple a fejlesztői környezetet is kizárólag a saját számítógépes rendszerükön tette elérhetővé, ezzel is erősítve a saját ökoszisztémájukat.

Jelenlegi formában a két rendszer funkcionalitása megegyezik, itt is van személyi asszisztens, követik a trendeket, és remek összhangban vannak a gyártó termékei egymással.

## Mobil fejlesztés

Amikor telefonos applikáció fejlesztésére kerül sor, 3 technológia áll rendelkezésünkre. Használhatjuk a gyártók natív programnyelveit, iOS esetén ez a Swift, Android esetén pedig a Kotlin. Ezek a nyelvek adják a jelenlegi legjobb teljesítményt és legtágabb funkcionalitást. Azonban használatukhoz ismerni kell a platformok működését, és magát a nyelvet is. Az így megírt kód nem használható újra a másik gyártó rendszerén, tehát két külön alkalmazás fejlesztésére van szükség.

Webfejlesztési előélettel kecsegtető lehetőség lehet a virtuális webapplikációk használata. Ilyen technológia például az Ionic. Ezek nagy előnye, hogy internetelérés nélkül is képesek futni, telepített formátumban. Működése megegyezik a böngészőben látható variánsukkal. Ezek a telefonon a rendszer beépített webnézegetőjén futnak, az ahhoz köthető elemek elrejtése mellett. Így azt a látszatot keltve, hogy egy valódi alkalmazást használunk.

Az utolsó csoportot a natív kódra forduló keretrendszerek alkotják. Használatukkal optimális esetben teljes kódmegosztás érhető el, így egyetlen technológia megismerésével fejleszthetjük az alkalmazásunkat mind a két operációs rendszerre. Felkapott webtechnológiát használnak alapul, amikben előre definiált eszközöket használhatunk. Ezeket az eszközöket fordítja át platformfüggő UI rendszerhívásokra, és jeleníti meg a képernyőre. Elterjedt rendszerek: Rect-Native, Xamarin-Forms, Native-Script.

A legfrissebb versenyző a Google legújabb keretrendszere, a Flutter, egy kicsit csavar még az előző elgondoláson.

## Flutter

Az UI elemeket többé nem a rendszertől kéri el, ezt kikerülve direktben a GPU-nak küldi ki az adatokat kirajzolásra. Menete:

* Layout Phase: a Flutter alapeleme a widget, ezek faszerkezetben írhatók fel. A gráf csúcseleméből a gyermekek felé kiküld egy kényszert, amit tiszteletben kell tartani, ennek tartalma általában a rendelkezésre álló hely magasság és szélesség adata. Amikor leellenőrizte, hogy belefér, generál egy új kényszert, és tovább küldi a saját gyermeke számára. Amikor leért a legalsó elemhez, összeállítja az elrendezés részleteit, és visszaküldi a szülőnek, aki a saját elrendezésével összefűzve szintén elküldi a saját szülőjének. A folyamaz végeztével mindennek meglesz a pontos helye.
* Painting Phase: létrehozza a kontextust, ami tartalmaz egy vásznat, erre festi fel a widgeteket. Lehetőség van új kontextus létrehozására is, ha átfedésben az előző fölé kellene rajzolni valamit.
* Compositing Phase: összeszedi az összes eddigi információt, és elküldi a GPU-nak feldolgozásra.
* Rasterizing Phase: Végső lépés, az alkalmazás megjelenik a kijelzőn, mint pixelek mátrixa.

Nagy előnye, hogy képes felvenni a versenyt a natív megoldásokkal, és használatával lefedhető mind a két cég mobilplatformja.

Úgy gondolom, hogy a minél jobb teljesítményt, és a legnagyobb elérhető felhasználói réteget figyelembe véve, a feladat megoldásához a Fluttert fogom használni.

## GDPR

### Mi is ez?

A GDPR a *General Data Protection Regulation* rövidítése. Egységes európai adatvédelmi rendelet, 2018 májusában lépett hatályba. Az Európai Unión belül egységessé teszi az adatvédelmet, és szabályozza az ügyféladatok kezelését. Minden olyan szervezetre vonatkozik, amely az EU-ban tartózkodó személyek adatait gyűjti, tárolja, vagy feldolgozza, akkor is, ha azok nem uniós állampolgárok, illetve a szervezet nem európai székhelyű. A be nem tartása súlyos bírságokat eredményezhet. A GDPR erősíti a bizalmat a szolgáltatók felé, növeli az adatbiztonságot, és javítja az adatkezelési folyamatokat.

A rendelet teljes szövege magyarul is elérhető az eur-les.europa.eu oldalán.

### Személyes adat

Minden olyan adat vagy információ, amely egy azonosított vagy azonosítható élő személlyel kapcsolatos, az személyes adatnak minősül. Illetve mindazon információk, amiket összegyűjtve egy bizonyos személy azonosítható, szintén személyes adatnak minősül.

Azok a titkosított vagy álnevesített személyes adatok, amelyeket felhasználva egy személy újra azonosítható, szintén az általános adatvédelem hatálya alá esik.

Az olyan személyes információk, amelyek anonimizáltak, és többé már nem azonosítható velük az érintett egyén, többé nem tekinthetőek személyes adatnak, de az anonimizálásnak visszafordíthatatlannak kell lennie.

Az általános adatvédelmi rendelet minden adatra vonatkozik, annak kezelése során használt technológiától függetlenül, IT-rendszerre épp úgy, mint biztonsági kamerás felvételekre.

### Adatkezelő, és feldolgozó

Az adatkezelő az a személy, aki az adatokat begyűjti. Tehát a valóságban, ha például elkérjük a felhasználó e-mail címét, amikor feliratkozik a hírlevélre. Ebben az esetben az üzemeltető az adatkezelő. Az ilyen esetekben begyűjtött adatok kikerülhetnek más vállalkozásokhoz is, például, ha a fent említett hírlevél küldéséhez egy online hírlevélszolgáltatót használunk. Ennek az üzemeltetője már adatfeldolgozó lesz, ugyanis ő nem gyűjtött be semmiféle adatot a felhasználóktól, megbízás teljesítéséhez kapott információkat.

### Adatkezelés

A rendelet előtti időkben a *Nemzeti Adatvédelmi és Információszabadság Hatóságtól* (NAIH) nyilvántartási számot kellett kérnie minden adatkezelőnek, és bejelentést kellett tenni róla. Ez a GDPR érvénybe lépésével megszűnt, ugyanakkor ha bármilyen incidens útján kikerülnek az általunk tárolt adatok, esetleg feltörik az oldalunkat, és kikerül az e-mail címek listája, ezt kötelesek vagyunk jelenteni a NAIH felé. Emiatt nagyon fontos, hogy a lehetőségekhez mérten mindent megtegyünk az álltalunk kezelt adatok biztonsága érdekében.

### Adatkezelési tájékoztató

A weboldalunk vagy alkalmazásunk látógatóit tájékoztatni kell az adatkezeléssel kapcsolatban. Ehhez el kell készíteni egy dokumentumot, ami részletesen, és legfőképpen érthető módon írja le, hogy milyen adatokat milyen célból, és meddig tárolunk el. Ehhez sok mintát találhatunk az interneten is, de segítségünkre lehet a NAIH személyes adatok kezeléséről szóló tájékoztatója is. Ennek értelmében tartalmaznia kell:

* Az adatkezelő személy vagy szervezet, illetve az érte felelős adatvédelmi tisztviselő adatait: neve, postacíme, e-mail címe, telefon, fax, honlap, székhely
* A személyes adatok kezelésének a célját, a jogalapját, tárolásának időtartamát, az adatok felsorolását és forrását
* Amennyiben a jogalap megjelölése jogos érdek, annak érthető bemutatását is
* Az adatok megadása szabad döntésen alapszik, abban az esetben, ha egy információra feltétlenül szükség van egy adott feladat elvégzéséhez, meg kell adni milyen következményekkel jár a nem megadása. Például: adatkérés biztonsági célból, következménye: belépés korlátozása, kizárás.
* Ha van profilalkotás az adatokból, vagy automatizált döntéshozatal, ennek részleteit, illetve a felhasználóra nézve a következményeinek ismertetése.
* Ha az adatihoz tartozik feldolgozó, annak megnevezése, az adatszolgáltatás céljának leírása.
* Azon személyek ismertetése, akik hozzáférhetnek a személyes adatokhoz. Illetve az okának ismertetése. Például: az Adatkezelő jogi képviselője, aki az érintett és az Adatkezelő közötti jogvita esetén ismerheti meg az adatokat.
* Az adatbiztonsági intézkedéseket, melyben ki van fejtve, hogy az Adatkezelő hol tárolja az adatokat, milyen szolgáltatást vesz igénybe ennek érdekében. Milyen intézkedésekkel gondoskodik arról, hogy illetéktelenek ne férhessenek hozzá a személyes adatokhoz.
* Ha harmadik ország vagy más nemzetközi szervezet felé történik adatszolgáltatás, annak megnevezése, annak megfelelőségi nyilatkozata, kötelező erejű vállalati szabályai, illetve különös helyzetekre vonatkozó eltérései.
* A használt cookie-k leírása
* A felhasználó jogainak ismertetése.
* A NAIH elérhetőségei, jogorvoslati panasz esetére

# Rendelkezésre álló technikák

## Flutter, Dart

A Flutter egy nyílt forráskódú, és ingyenes keretrendszer, a Google fejlesztésében jelent meg 2017 májusában. Natív alkalmazások sebességét megközelítő applikációk írását teszi lehetővé, egyetlen kódbázisból. Két fontos részből áll alapvetően:

* SDK (fejlesztői környezet): olyan eszközök gyűjteménye, ami segít az alkalmazás létrehozásában. Ez magán foglalja a kód natívra fordítását is.
* Keretrendszer: újrahasznosítható UI elemek gyűjteménye, amik az egyéni igényekre szabhatóak. Például: gombok, beviteli mezők, csúszkák...

Ahhoz, hogy Flutter alkalmazást tudjunk fejleszteni, a Google saját programozási nyelvét kell használnunk, a Dartot. 2011 októberében jelent meg, az idők folyamán sokat fejlődött. A front-end fejlesztésre koncentrál a nyelv, és mobil alkalmazások és weblapok létrehozásához egyaránt használható. A dart egy típusos objektum orientált programozási nyelv, JavaScripthez hasonló szintaxissal.

Támogatja a Hot-Reload funkciót, ezt webfejlesztők már ismerhetik, mobilos környezetben inkább csak próbálkozások voltak rá. Ez annyit tesz, hogy a kód írása közben valós időben láthatjuk a változtatásokat. Erős változtatások hatására, például egy új függőségből egy példány létrehozása, újra kell indítanunk az alkalmazást, de ennél kisebb módosításokat képes lekezelni.

A flutter.dev weboldalon remek dokumentáció érhető el, rengeteg példával, érthetően fogalmazva. A legtöbb problémánkra megoldást találhatunk rajta. Ezt úgy éreztem pozitív értelemben meg kell említenem, ugyanis más keretrendszerek keresése során nem a jól használhatóság volt a tendencia, mint inkább a labirintus szerű bolyongás.

A Flutter építő elemei a widget-ek. Ezek kombinálásával és egymásba ágyazásával epíthető fel az egész felhasználó felület. Minden widget definiál egy struktúrális elemet(gomb), egy stilisztikai elemet(szín), vagy egy elrendezési viselkedést(térköz).

Érdemes megemlíteni, hogy nem használja egyik platform natív elemeit sem, ugyanakkor a Flutter Library tartalmaz rengeteg használatra kész widgetet, amig úgy néznek ki mint az iOS illetve az Android által használtak. Ezek külön témákba szedve találhatóak meg, Material Design, és Cupertino néven. Természetesen ezektől eltérhetünk, és személyre lehet szabni mindent.

Az SDK futási időben fordítja a Dart kódot natív kóddá, így a többi rendszerrel ellentétben nincs szükség egy JavaScript hídra, emiatt nő a velük szembeni teljesítmény is. Ennek hozadékaként a platformmal való kommunikáció sebessége is nő, a híd megspórolásával, és a nyelvek közötti kontextusváltás hiányával. A Flutter jelenleg az egyetlen rendszer ami lehetővé teszi nélküle a reaktív felhasználói felületet.

A kirajzolási folyamatot már az előző fejezetben szemléltettem, de azok használatról ne esett szó. A Flutter frissíti a fát minden egyes új képkockánál. Hogy ezt megtehesse, sok objektumot készít, amig talán csak egyetlen kockán keresztül élnek. Ennek tisztán tartására egy generációs garbage collectort használ, ami nagyon effektíven működik ebben a rendszerben.

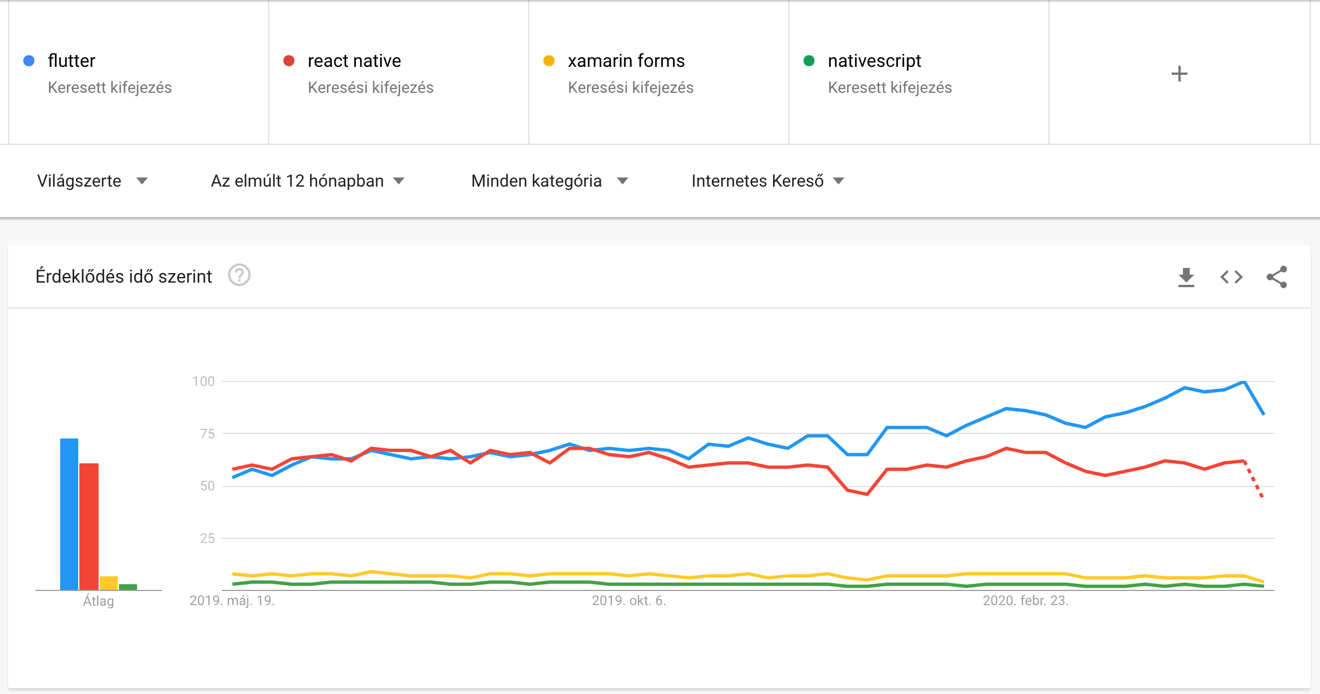
A képen képernyőkép látható

Automatikusan generált leírás

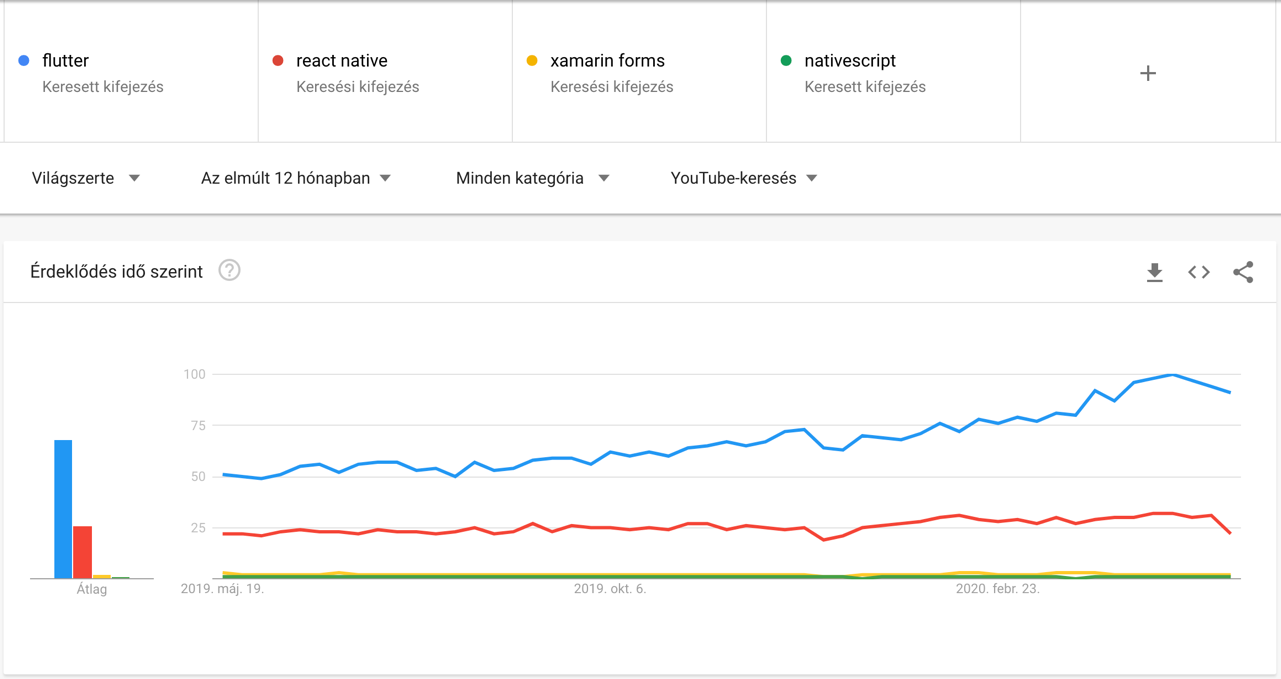
*2.ábra: Flutter részei*

Sok időt és pénzt tud megspórolni annak a vállalatnak, amelyik a használata mellett dönt. Két külön fejlesztőcsapat alkalmazása helyett elegendő egy, és ezzel le is van fedve mind a két nagy platform. A sebessége megegyezik a valódi natív alkalmazásokkal, már most is sok Flutter alkalmazás érhető el a Store-ban. A leginkább említésre méltó példa, a Xianyu app, amit az Alibaba fejlesztett, és több mint 50 millió aktív felhasználója van.

Jelenleg a legígéretesebb keretrendszer a piacon, ez meglátszik a népszerűségén is. A GitHub szoftver oldalak között a 11. helyen szerepel a csillagozások számában, illetve a Google keresések számában elért töretlen növekedés is figyelemre méltó. *Lásd 3. ábra*. Még szignifikánsabb eltéréseket vehetünk észre, ha ugyanezt a statisztikát leszűkítjük a YouTube-ra, ami mára az egyik legnagyobb információközponttá nőtte ki magát. Ha valaki kíváncsi egy technikára, először szinte mindig a neves videómegosztón keres az anyag után. *Lásd 4.ábra*. A ábrák a Google Trends oldalon készültek.



*3.ábra: Keresések száma a keretrendszerekre*



*4.ábra: Keresések száma a YouTube-on*

## Clean Architecture

A Clean Architecture Robert C. Martin alias Uncle Bob ötlete. Az elképzelés teljesen keretrendszer és programozási nyelv független. Bármilyen környezetben könnyen használható alkalmazás fejlesztésére. A kódot egymástól független részekre kell bontanunk, és ezek kapcsolatát absztrakcióval, nem pedig konkrét implementációval kell elérnünk. Az Uncle Bob féle diagrammot felhasználva, az alkalmazásunk a következő magasszintű elrendezést, és kommunikációt fogja alkalmazni:

A képen képernyőkép látható

Automatikusan generált leírás

*5.ábra: Clean Architecture*

Minden layer független a másiktól, és jól meghatározott feladata van:

* Data Layer: Az adatok begyűjtésével foglalkozik, a forrás típusától függetlenül, például: adatbázisszerver, lokális fájl, felhő, cache. Feladata még ezeknek a nyers adatoknak a modellé konvertálása. Tartalmazza még annak a logikáját, hogy melyik adatot honnan kell betölteni.
* Domain Layer: az üzleti logika magját tartalmazza, különböző Use Case-ekre elosztva. Vegyük észre, hogy a Domain Layerben már entitásokkal dolgozunk, nem pedig modellekkel. A modell valójában az entitás leszármazottja kell legyen, erre a különbözőségre azért van szükség, hogy ha kicseréljük a komplett Data Layert, és a konverzió a Domain Layerhez tartozott, veszélybe kerül a működés.
* Presentation Layer: ez a szint nem lehet teljesen Clean Code-ban megoldani, ugyanis a UI létrehozásához widgeteket kell használnunk, amik elindíthatnak bizonyos Use Caseket. Ennek megoldására egy státusz menedzselő megoldást kell használnunk, jelen esetben a BLoC, ami képes vezérelni a felhasználói felület státuszát, illetve lekezelni az onnan érkező feladatokat.

A reaktív felület létrehozásához nagy segítséget fog nyújtani a BLoC pattern (Business Logic Components). Ez a komponens tárolja, és vezérli a UI státuszát, és lekezéli az onnan érkező utasításokat (eventeket). Ezekhez streameket használ. A streamet úgy lehet elképzelni, mint egy cső, az egyik végén adatot lehet bele rakni, a másik végén pedig kiolvasni azt. Az eventeket is ilyen streameken keresztül lehet triggerelni, és ha annak eredményeképpen történnie kell változásnak a felületen is, az szintén így kap róla értesítést. Relatíve egyszerű a használat, mégis nagyon jó sebességet lehet elérni a használatával.

A jól átláthatóság, és könnyű fejlesztés érdekében az alkalmazásunkhoz ezt az architektúrát fogom használni.

## Firebase

A Firebase egy szolgáltatáscsomag a Google-től, használatával rengeteg kényelmi és hasznos funkciót is kapunk. Képes kiváltani egy komplett mobilalkalmazás backend szerverjét, és adatbázisszerverét egyaránt.

Önmagában nem egy konkrét termék, több kisebb-nagyobb szolgáltatás egy nagy integrált egésszé gyúrva. A jelenleg használható eszközök sora az alábbi képen látható:

A képen képernyőkép látható

Automatikusan generált leírás

*6.ábra: Firebase részei*

A szolgáltatások sora nagyon bőséges, és semmi sem kötelező. A fejlesztő azokat a modulokat használhatja, amire csak szüksége van. Megoldható, hogy csak a bejelentkezési rendszert használjuk, ha csak arra van szükség egy projecthez. Azonban az is látható, hogy még egy nagyobb, komplexebb alkalmazáshoz is bőven fel lehet építeni a platformra az egész back-end szolgáltatást, sőt, még többet is mint amire szükség lenne.

Fájlokat tárolhatunk a Hosting, és a Storage modulban is, de lehetőségünk van egy valós idejű adatbázist is használni. Bár ide leginkább az alkalmazás szintű adatokat érdemes tenni, esetleg olyan adatokat, amiknek az elérése és frissessége a fontos.

Az Authentication komponens lehetőséget ad felhasználók regisztrálására, bejelentkeztetésére. Alapból ismeri a nagy szolgáltatókat is, Facebook, Google, GitHub, és ezen accountainkkal is be tudunk lépni az alkalmazásba, ha azt felkészítjük előtte az ilyen bejelentkezésre.

Dynamic Links és Invites modul segítségével olyan hivatkozás hozható létre, amivel az appon belül egy konkrét helyre navigálható a felhasználó. Abban az esetben ha nem rendelkezik az alkalmazással, az áruházi oldalára navigál, és a telepítést követően az első megnyitás alkalmával is érvényben marad, és ott nyitja meg, ahová mutat.

A Firebase 3 csomagban érhető el:

* Spark: Ingyenes verzió, korlátozást leginkább az adatbázis olvasások limitálásában találni.
* Flame: Fix havidíjas verzió.
* Blaze: Legnagyobb csomag, limit nélkül, a felhasznált keretek után kell fizetni.

Az általam tervezett alkalmazáshoz a fenti funkciók, illetve az ingyenes Spark csomag használata elegendő kell legyen.

Források:

Irodalom feldolgozás:

<https://hu.wikipedia.org/wiki/Okostelefon>

<https://www.counterpointresearch.com/global-smartphone-share/>

<https://mobilarena.hu/teszt/android_fejlodestortenet/android_tortenelem.html>

<https://pcworld.hu/macworld/igy-fejlodott-az-ios-elso-resz-163108.html>

<https://pcworld.hu/mobil/igy-fejlodott-az-ios-masodik-resz-163418.html>

<https://medium.com/saugo360/flutters-rendering-engine-a-tutorial-part-1-e9eff68b825d>

<https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_hu>

<https://eur-lex.europa.eu/legal-content/HU/TXT/HTML/?uri=CELEX:32016R0679&from=HU>

<https://elemzeskozpont.hu/vallalkozas/gdpr-osszefoglalo-gdpr-lenyege-gyakorlati-tanacsok>

<http://gdpr.mvmh.hu/mvm/Tamogatas_igenylok_AT_v1.pdf>

<https://flutter.dev/>

<https://medium.com/@concisesoftware/what-is-flutter-here-is-everything-you-should-know-faed3836253f>

<https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/>

<https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>

<https://resocoder.com/2019/08/27/flutter-tdd-clean-architecture-course-1-explanation-project-structure/#Clean_Architecture_Flutter>

<https://www.raywenderlich.com/4074597-getting-started-with-the-bloc-pattern>

<https://medium.com/mito/kiprobaltuk-firebase-analytics-database-fabric-d6b208d6288f>