Mobilalkalmazás fejlesztése, felhő alapú szolgáltatások segítségével Problémadefiniálás, helyzetfeltárás

Kőházi Dániel
EQLO7M
kohazi.daniel@stud.u-szeged.hu

Témavezető: Dr. Németh Tamás

Problémadefiniálás, helyzetfeltárás

Manapság egy mobilalkalmazásfejlesztés során két opció áll a fejlesztői csapat előtt

1. Natív alkalmazás fejlesztése
2. Keresztplatformos technológia használata

Első opció

Általában az **első opció** a biztonságosabb választás, hisz ez a leg eszköz közelibb és legkevesebb limitációval járó megoldás. Viszont egy ilyen út bejárása rengeteg erőforással jár, hiszen gyakorlatilag kétszer le kell fejleszteni az applikációt hogy mindkét platformot támogassa (**iOS, Android**).

Második opció

AZ előbbi problémának a megoldására találták ki a **második opciót**. Ezek a technológiák már viszonylag régóta léteznek, például az első népszerűbb ilyen keretrendszer a PhoneGap volt ami már 2011-től elérhető volt, viszont nem lett egy bevett alternatíva, mert nagyon távol állt egy natív alkalmazástól. Ez annak köszönthető hogy egy olyan megvalósítást használ a **PhoneGap** hogy egy **WebView**-t (gyakorlatilag egy "keretek" nélküli webböngésző) használ, amiben egy **webalkalmazás** fut, azzal a különbséggel hogy ez a webalkalmazás képes **kommunikálni az Android API-val**. Viszont folyamatosan jelentek meg az újabb keretrendszerek amelyek egyre többet tudtak, mint például a React Native. Ezzel a keretrendszerrel szakmai gyakorlatom is van, személy szerint azt tapasztaltam hogy már eléggé közeli megoldásnak mondható, itt például már **natív UI** elemek generálódnak le, érzésre nagyon közelít a natív alkalmazásokhoz, viszont teljesítményben még mindig lemaradt a natív társaihoz képest.

Flutter

A korábban felsorolt gyengeségek miatt lehet érezni hogy ezek a keresztplatformos technológiák még gyermekcipőkban járnak. Viszont a **Google** ezekből a gyengeségekből tanulva, előrukkolt egy új keretrendszerrel, a Flutter-el. Maga egy keretrendszer **Dart** nyelvet használ, ami egy picit idegen lehet, viszont maga a nyelv nagyon hasonlít az **ES6**-ból már jól ismert elemekre. A Flutter segítségével villámgyors fejlesztési idő áll rendelkezésre, letisztult, gyors és szép **natív UI elemeket**, valamint **teljesítményben** is **kiemelkedő** eredmények. Erre rengeteg példát lehet találni: eBay motors, Google Ads, Philips Hue, valamint amit külön kiszeretnék emelni a Hamilton app.

Hamilton app fejlesztése a Flutter-ben

Ez az applikáció azért különleges hiszen ez az ékes példája annak hogy milyen **gyorsan** és mennyire **igényes** applikációt lehet fejleszteni a Flutter-ben. Magát az alkalmazást 2017-ben kezdték el fejleszteni szoros határidővel, viszont ezt a htáridőt bőven sikerült tartani, hiszen **3 hónap** alatt sikerült lefejleszteni a teljes értékű applikációt mindkét platformra. A fejlesztő csapat szerint a következő pontok segítették elő a sikert:

- **Egy fejlesztői csapat** - Csak egy csapatra volt szükség, akik egy kódbázison dolgoztak, ugyanabban a keretrendszerben, nyelvben, így magas volt a **kollaboráció** a fejlesztők között, valamint a **flexibilitás** is, bármikor tudtak egymásnak segíteni ha valaki kiesett a fejlesztésből betegség, vagy szabadság miatt.
- **Alkalmazás elrendezésének létrehozása** - Nagyon egyszerűen lehet olyan flexibilis UI elrendezést létrehozni amely jól éz ki iOS és Android oldalon is.
- **Prototípus készítés és dizájn iteráció** - Flutter-ban a UI prototípus készítés nagyon gyorsan végrehajtható, majd ezt a prototípust gyorsan tudjuk iterálni is.
- **Hot reload** - Az egyik legigéretesebb funkciója a Flutter-nek. Miután változtatunk valamit az applikáción, a Flutter **beinjektálja** az új kódrészeket az app-ba, így nem lesz szükség újraindítani, mint például egy natív Android alkalmazásnál ahol mindig új apk-t kell build-elni. Ez nagyon hasznos lehet például a debug-olásban, hiszen **nem történik állapotvesztés**, nem kell újra odanavigálni az adott screen-hez, valamint a korábban említett UI design iterálását is ez segíti előre.
- **IDEA** - A Flutter több IDEA-t is támogat fejlesztői csomaggal: **IntelliJ**, **Android Studio** és **Visual Studio Code**. Ez gyakorlatilag egy all in one csomag, mely tartalmaz debugging eszközöket, kódformázót, valamint megkönnyíti az importokat.
- **Animációk egyszerűsége** - Flutter-ben készült animációk sokkal **átláthatóbbak**, **egyszerűbben létrehozhatóak**, mint ha natív kódot használnánk.
- **Tesztelés** - Mivel egy kódbázis van, így azok a funkciók amik lelettek tesztelve **iOS**-en, nem kellet újra tesztelni őket **Android**-on.
- **Plugin** - A keresztplatformos alkalmazásoknál az egyik legnagyobb gond hogy rengeteg **OS specifikus** funkcióhoz nem tud hozzáférni. A **Flutter plugin-ok** támogatást biztosítanak a legtöbb ilyen funkcióhoz.
- **Visszatérés natív kódra** - Ha a korábban említett pluginok nem támogatnak egy adott funkciót, akkor bármikor visszalehet térni natív kódra, ami együtt működik a Flutter kóddal. Persze a cél hogy a jövőben egyre kevesebb ilyen helyzet alakuljon ki.

Összegezve körülbelül **75%-125%** közötti értékkel nőt a hatékonyság mintha külön-külön lett volna építve a két applikáció. Több mint **1 millió** ember töltötte le, és rengeteg **pozitív visszajelzést** kapott.

Következtetés

Tehát a szakdolgozatra készítendő mobilalkalmazásnál az elöző példában sorolt előnyöket mindenképp kiszeretném használni. Valamint a backend-et is felhő szolgáltatásokkal szeretném kivetelezni. Erre tökéletes lesz a **Firebase** felhő szolgáltatása, mely lényegesen lerövidíti a backend fejlesztését. A Firebase szerencsére nagyon jól müködik együttt a Flutter-el, rengeteg SDK csomag található amivel könnyen elérhetőek ezek a szolgáltatások.