Multidiszciplináris tudományok, 10. kötet. (2020) 2 sz. pp. 519-524 **https://doi.org/10.35925/j.multi.2020.2.59**

ŰRLAPKITÖLTÉS? KATTINTSON!

Szűcs Miklós
mesteroktató, Miskolci Egyetem, Általános Informatikai Intézeti Tanszék
3515 Miskolc, Miskolc-Egyetemváros, e-mail: szucs@iit.uni-miskolc.hu

Bodnár Renátó
egyetemi hallgató, Miskolci Egyetem, Általános Informatikai Intézeti Tanszék
3515 Miskolc, Miskolc-Egyetemváros e-mail: bodnarreno@gmail.com

Absztrakt

A feladat alapötlete a jelenleg is használatban lévő ügyfélkezelő megoldások forradalmasítása, egy adatcserélő applikáció elkészítése. Az applikáció képes adatokat egyedi azonosítók alkalmazásával biztonságosan letárolni. Az azonosítók szabványosak, így adatcserére is alkalmasak. Az adatok biztonsági szintje szabályozható, így adatcsere esetén csak a megfelelő szintű adatok kerülnek átadásra. A kért adatok egy QR kód beolvasásával kerülnek azonosításra. A leolvasás után egy felületen az adatok ellenőrizhetők, szerkeszthetők. A kötelező adatok mindig átadásra kerülnek, az opcionális adatok csak a megfelelő beállítás esetén. Az adatok visszaküldése szintén QR kód segítségével történik. A kód eltárolható, így a küldött adatok utólag is ellenőrizhetők.

Kulcsszavak: *adatcsere, QR kód, adatbiztonság*

Abstract

The basic idea of the task is to revolutionize the customer management solutions that are still in use, to create a data exchange application. The application is able to store data securely using unique identifiers. The identifiers are standard, so they are also suitable for data exchange. The security level of the data can be regulated, so that only the appropriate level of data is transferred in the event of a data exchange. The requested data is identified by scanning a QR code. After reading, the data can be checked and edited on an interface. Mandatory data is always transmitted, optional data only if set correctly. Data is also returned using a QR code. The code can be stored so that the sent data can be checked afterwards.

Keywords: *data exchange, QR code, data security*

1. 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 regisztrál 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. A tényleges ügyintézés előtt viszont azonosítania kell magát az okmányaival.

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

2. A program működése

Az applikációnak képesnek kell lennie a felhasználókat regisztrálni, be- illetve kijelentkeztetni, és egyedi azonosítót rendelni hozzájuk. Az azonosító alapján a megadott adatokat biztonságos módon eltárolni. A tárolni kívánt adatok csoportja egyéni, mindenki azokat mentheti el, amiket szeretne (egy létező lista alapján). 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 QR kódot lehet menteni, amit kinyomtatva ki kell ragasztani jól látható helyre. Az ügyfél érkezését követően az alkalmazáson belül leolvassa azt, és a megjelenik egy képernyőn egy űrlap, a kért adatokkal.

Minden opcionális adat mellett van egy jelölőnégyzet, amivel beállítható, hogy az adatot át akarja-e adni a felhasználó. A kötelező mezőket nem lehet letiltani. Az elküldés gomba bökve az rendszer küld egy értesítést a szolgáltatónak, ami tartalmazza a kért információkat, és beteszi a várakozó sorba az ügyfelet. Ebben a sorban képesnek kell lenni az adatok olvasására, illetve a jelenleg aktív vendég lezárására.

A kliens oldali applikációba integrálni kell egy QR leolvasót, ami tudja értelmezni a generált kódokat, és kinyerni belőle a szükséges adatok listáját. Ezt konvertálni kell egy megjeleníthető objektum halmazzá, ami a kliensen még nem letárolt adat esetén üres beviteli mezőt jelent, más esetekben egy tartalommal megjelenő szövegmezőt, illetve az opcionális adatok mellett egy jelölőnégyzetet. 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 kell, és be kell tudni olvasni a már létező, mentett adatokat.

3. Felhasznált nyelvek, technológiák

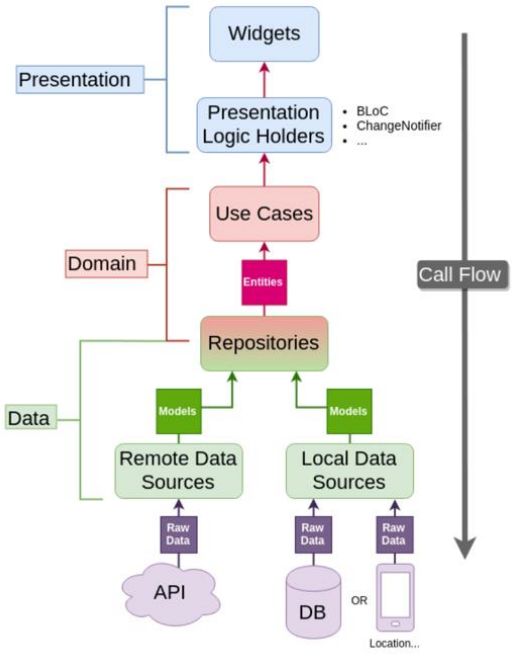
A Visual Studio Code [1] jelenleg az egyik legnépszerűbb fejlesztési környezet. Nagy előnye, hogy rendkívül kevés helyet foglal, az alap telepítés körülbelül 50 megabájt környékén van, ez persze különböző bővítmények telepítésével tovább nőhet. A nagy testvérének, a Visual Studio-nak csak a telepítője tízszer ekkora helyet foglal el. A Visual Studio Code használata teljesen ingyenes, nem kell hozzá sem előfizetés, sem egyszeri licenc vásárlása. Teljesen nyílt forráskódú, a fejlesztés menete is megtekinthető bárki számára, illetve elérhető az összes nagyobb platformra: MacOS, Windows, és Linux. Alapverzióban támogatja az IntelliSense funkciót, amivel automatikusan felismeri milyen nyelvben íródott a kód, amin éppen dolgozunk, és próbálja 'megérezni', hogy milyen szót vagy paran-

csot szeretnénk begépelni, ezen felül kiegészítési lehetőségeket kínál fel nekünk. Ebbe beletartoznak a változóink nevei is, ehhez figyelembe veszi a névtereket is.

A Flutter [2, 3] egy nyílt forráskódú, és ingyenes keretrendszer, a Google fejlesztésében jelent meg 2017 májusában. Natív alkalmazások sebességét megközelítő applikációk írását teszi lehetővé, egyetlen kódbázisból. Alapvetően két fontos részből áll:

- SDK (fejlesztői környezet): olyan eszközök gyűjteménye, ami segít az alkalmazás létrehozásában. Ez magában 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. Ezt Dartnak [4] hívják, 2011 októberében jelent meg, és azóta sokat fejlődött. A front-end fejlesztésre koncentrál a nyelv, mobil alkalmazások weblapok létrehozásához egyaránt használható. A Dart egy típusos objektumorientált programozási nyelv, JavaScripthez hasonló szintaxissal. A Flutter építőelemei a widgetek (grafikus űrlapelemek). Ezek kombinálásával és egymásba ágyazásával építhető fel az egész felhasználói felület. Érdemes megemlíteni, hogy nem használja egyik platform natív elemeit sem, ugyanakkor a Flutter Library tartalmaz rengeteg használatra kész widgetet, amik 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 hibrid rendszerekkel ellentétben nincs szükség 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. Az alkalmazást Robert C. Martin alias Uncle Bob ötlete, a Clean Architecture [5, 6] (1. ábra) alapján készítettük.



1. ábra. *Clean Architecture with BLoC*

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. Minden réteg független a másiktól, és jól meghatározott feladata van. Segítségével egy nagyon könnyen átlátható kódot kapunk, aminek a bővítése esetleg a szintek teljes cseréje is egyszerű.

A reaktív felület létrehozásához nagy segítséget nyújt a BLoC pattern (Business Logic Components). Ez a komponens tárolja, és vezérli a UI státuszát, és lekezeli az onnan érkező utasításokat (eventeket). Ezekhez streameket használ.

Backendnek a Firebaset [7] alkalmaztuk, ami egy Google fejlesztésű szolgáltatáscsomag. Használatával rengeteg kényelmi és hasznos funkciót is kapunk, pl. képes kiváltani egy komplett mobilalkalmazás szerverét. Önmagában nem egy konkrét termék, több kisebb-nagyobb szolgáltatás egy nagy integrált egésszé összegyúrva. A programban a felhasználók bejelentkezését, az adatok tárolását, és az alkalmazások közötti kommunikációt látja el.

4. Megoldás

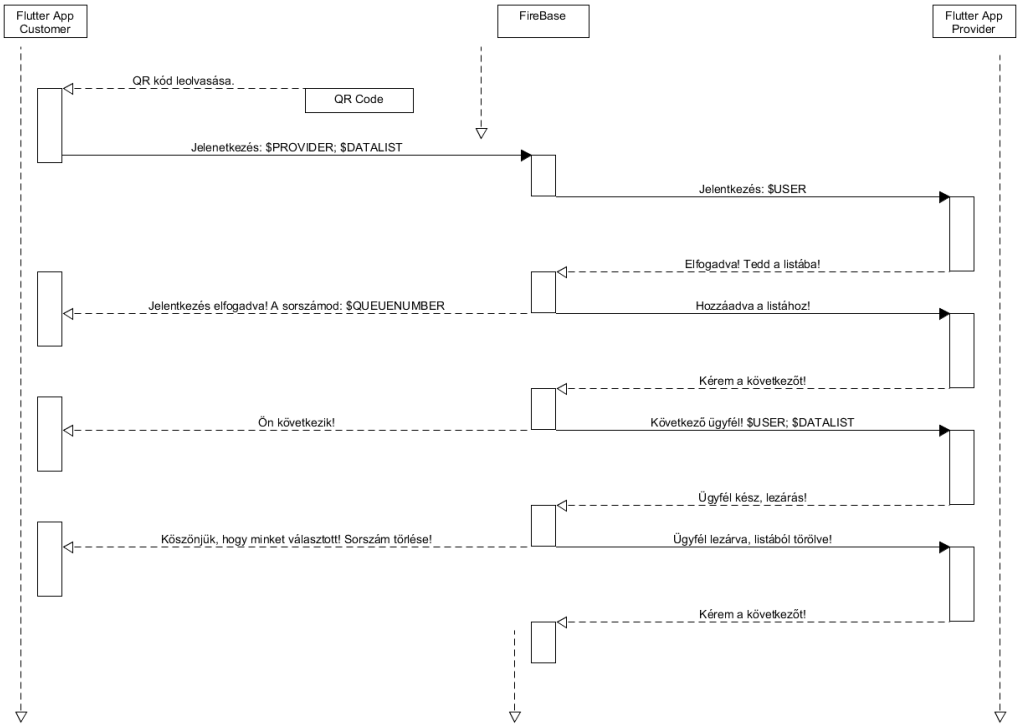
Az első feladat, a szerverként szolgáló Firebase project létrehozása. Ezt ingyenesen megteheti bárki, aki már rendelkezik Google felhasználói fiókkal, a console.firebase.google.com oldalon. A legalapvetőbb és legfontosabb dolog, hogy a nem regisztrált felhasználók ne legyenek képesek használni az alkalmazást.

A fejlesztést a bejelentkezés, és a regisztráció implementálásával kezdtük. Az előzetes tervek alapján létrehoztuk a BloC struktúrát, illetve egy egyszerű űrlapot, ahol meg lehet adni az e-mail címet és jelszót. A nagyon szoros Flutter-Firebase integráció miatt ez a művelet rendkívül egyszerű volt, egy függőség importálása után, annak a jó függvényét meghívva már meg is történt az autentikáció. Ez gyakorlatilag bármilyen backendhez köthető funkcionalitásnál ugyanígy megoldható.

A Firebase több lehetőséget is nyújt az adatok tárolására. A megoldásomhoz a Cloud Firestore elnevezésű NoSQL adatbáziskezelőt alkalmaztuk. Regisztráció során minden felhasználóhoz automatikusan létrejön egy FirebaseUser, amihez generálódik egy egyedi azonosító is. Az adatok biztos tárolására ezt az azonosítót használtuk, így nem történhetett meg, hogy hibás lekérésből adódóan bárki hozzáférjen mások adataihoz. Ezután létrehoztunk egy userData nevű kollekciót, amiben minden felhasználónak lesz egy saját dokumentuma, a neve pedig a generált azonosítója lesz. Az általános oldalak és űrlapok összerakása után a fejlesztés már az eltervezett workflow (munkamenet) (2. ábra) alapján haladt.

A munkamenet alapján a következő feladat a QR kód generálása, és annak beolvasása volt. Manapság megszámlálhatatlan oldal biztosít ingyenesen hozzáférhető végpontokat, aminek a segítségével elvégezhető a generálás. Ugyanakkor a telefonok már rendszer szinten is ismerik ezeket a metódusokat, így a natív API-t használó megoldás mellett döntöttünk, választásunk miatt a harmadik féltől való függés elkerülhető. A beolvasásra szintén lehetőséget nyújtanak a telefonok, így az egész folyamat menedzselhető bármilyen függőség nélkül. A következő, és a legnagyobb probléma az alkalmazások közötti kommunikáció. Ennek megoldására szükségünk volt a Firebase Cloud Messaging szolgáltatására, amit alapvetően az üzemeltetők szoktak használni alkalmazáson belüli értesítések küldésére. Ezen belül lehetőség van csoportokra is szűrni, mint például Android felhasználók, egy adott korcsoport és hasonlók. Egy bizonyos funkciójával képesek vagyunk egy másik felhasználónak is üzenetet küldeni, az előre regisztrált ID alapján. Ezt kihasználva küldhetők el a feliratkozáshoz, és a behíváshoz szükséges adatok. A beérkezett értesítések esetében az operációs rendszerek API-ján keresztül egy

Push notificationt (egy mobil alkalmazás által a telefon kezdő képernyőjére küldött üzenet) is megjelenik a felhasználók készülékén.



2. ábra. Workflow

Az elküldött adatok kódolása és dekódolása volt a legnehezebb feladat, ugyanis a JSON formátumba konvertált halmazt egyszerű szövegként továbbítva kissé nehezen alakítható vissza egy típussá a több szóból álló elnevezések miatt, mint például a First name. De egy minimális trükkel ez is orvosolható volt. Valójában miután minden lehetőséget bekötöttünk a megfelelő helyre, és az alkalmazás már készen is állt a használatra.

5. Felhasználhatóság

A tervekhez hűen a program szinte bármilyen ügyfelekkel foglalkozó egységben használható, ahol szükséges a sorban állás, ilyenek az okmányirodák, nagyobb posták, bankok. Ez a jelenlegi tilalmakkal teli helyzetünkben hatványozottan igaz, és szükséges is lenne. Gondoljunk csak egy étteremre, ahol az utcán sorban állva várakozunk a rendelés felvételére. Ezt nagyon egyszerűen ki lehetne küszöbölni az alkalmazás használatával, és csak annak a vendégnek kellene az ajtó közelében tartózkodni, aki valóban sorra is került. A probléma tökéletes megoldásához a programváltoztatások nélkül felhasználható.

Az applikáció elég generikus lett hozzá, hogy flexibilisen alkalmazható legyen bármilyen helyzetben, ennek csak a képzelet szab határt. Továbbfejlesztési lehetőséget kínál a további általánosítás. Ha

nemcsak adatokat, hanem formázott űrlapokat is tudna a program továbbítani, akkor az éttermi sorban állók már megkaphatnák az étlapot is, és az alapján a rendelés is megtörténhetne. Ezzel is rengeteg időt, és szükségtelen érintkezést tudnánk megspórolni. Az online oktatás világában a számonkéréshez is nagy segítséget tudna nyújtani. A szóbeli vizsga beugró kérdései személyre szabottan érkezhetnének meg a megerősítő üzenetben, és a válaszok beérkezésének sorrendjében az oktató egyből látná az eredményeket is, illetve be is hívhatná a diákokat.

Az eddig említett felhasználások egy-több kapcsolatot feltételeznek, tehát egy szolgáltató több kliens tud kezelni. Átalakításokkal ezt ki lehetne terjeszteni több-több kapcsolatra is, ebben az esetben egy cég alá beregisztrált összes ügyintéző megkaphatná a feliratkozásról az értesítéseket, és aki éppen felszabadul saját magához hívhatná a következő ügyfelet, a saját asztalának kódjával.

Technikai fejlesztési ötletünk lenne a kompatibilitás megnövelése. A cikk elkészítésekor nem foglalkoztunk webapplikáció létrehozásával, de a Flutter ezt már az újabb verziók megjelenésével támogatja, a Windows alkalmazásokkal együtt. Tehát teljesen megegyező kódbázisból, az informatikai világ szinte teljes egésze lefedhető. Ez hasznos újítás lehetne olyan irodák számára, akiknek tabletjeik nincsenek, de számítógépük már igen, ekkor az alkalmazottak egyszerűen csak megnyitnák a böngészőben a megfelelő oldalt, a felhasználók ezzel egyidőben továbbra is élvezhetnék a natív alkalmazás előnyeit.

6. Összefoglalás

Úgy gondoljuk, megszámlálhatatlan irány van még, ami nekünk nem jutott eszünkbe, de az olvasónak már az alapötlet bemutatás óta a fejében van. Ez egy remek project volt, ami rávilágított mekkora potenciál lehetne egy olyan egyszerű eszköz továbbgondolásában is, mint a sorszámhúzó gépek. Az okostelefonok már meghódították a világot, látszik a tendencia, hogy amihez nincsen applikáció, azt az emberek már nem szívesen használják. Ebben a világban egy remek belépési pont lehet a Flutter, és az adatcserélő applikáció.

Irodalom

[1] https://code.visualstudio.com/
[2] https://medium.com/saugo360/flutters-rendering-engine-a-tutorial-part-1-e9eff68b825d
[3] https://flutter.dev/
[4] https://dart.dev/
[5] https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
[6] https://www.raywenderlich.com/4074597-getting-started-with-the-bloc-pattern
[7] https://firebase.google.com/
[8] Bodnár Renátó: Adatmegosztó alkalmazás, Szakdolgozat, ME ÁIT, 2020