10 oldalas beszámoló

TODO

- [x] előlap elkészítése
- [x] tartalomjegyzék elkészítése
- érdemi rész megírása (min. 10 oldal)
    - fejlesztéshez használt szoftverek és eszközök
        - [x] Visual Studio Code
            - [x] Fontosabb bővítmények
        - [x] Git
        - [x] GitHub
        - [x] React Native
        - [x] XAMPP
        - [x] Postman
        - [x] Xcode
        - [x] Android Studio
        - [x] Firebase
            - [x] FCM
    - fejlesztéshez használt programnyelvek
        - [x] JavaScript
        - [x] TypeScript
        - [x] Swift
        - [x] Kotlin
        - [x] PHP
    - [x] munka home office-ban
        - [x] idő követése
        - [x] fejlődés (péntekek)
        - [ ] kommunikáció
    - idealApp cross-platform alkalmazáskeret
        - [ ] specifikáció
        - [ ] feladataim
        - [ ] kliens
            - fontosabb használt könyvtárak
                - [ ] react-native-device-info
                - [ ] react-native-biometrics
        - [ ] szerver
            - [ ] cél
        - [ ] dokumentáció

Piszkozat

Használt szoftverek és eszközök

Visual Studio Code

A Visual Studio Code (vagy gyakran VSCode-ként emlegetett) a Microsoft népszerű, minden platformon elérhető kódszerkesztő programja. 2015-ben jelentették be és tették közzé a forráskódját. A funkciói között megtalálható a hibakeresés, szintaxis kiemelés, intelligens kód kiegészítés, beépített Git és bővíthetőség. A fejlesztők könnyen beállíthatják szinte bármilyen munkához a kiegészítők nagy választéka segítségével, így én is nagyon jól tudtam használni a számomra kiadott feladatok megoldásához.

Fontosabb bővítmények

- ESLint: Ez a bővítmény szintaxis és logikai hibákat ismer fel és emel ki fejlesztés közben.
- Path Intellisense: Ez a bővítmény az elérési utakat és import-okat ellenőrzi a kódban, hogy helyesen vannak-e megadva, illetve az írásnál segítséget biztosít a gyorsabb munkához.
- PHP Extension Pack: Ez a kiegészítő gyűjtemény a PHP programnyelvhez ad több segítő eszközt a hatékonyabb munkához, hibakereséshez. Gyakran hasznát vettem a projektem szerver oldalának fejlesztésénél.
- React Native Tools: Mivel a projektem kliens rész React Native technológiával készült, elengedhetetlen volt ennek a bővítménynek a használata, ami React Native projekt futtatásához, hibakereséséhez integrál eszközöket a VSCode-ba.

Git

A Git a legelterjettebb verziókezelő rendszer az iparban. Nyílt forráskódú, gyors, rengeteg programban megtalálható, mint beépített funkció és nagyban megkönnyíti a munkát biztonsági mentések és csapatmunka szempontjából. 2005-ben került kiadásra Linus Torvalds által. Már régóta használom az alkalmazás projektjeimhez, asztali programokhoz, szoftverfejlesztői versenyeknél és a technikusi szakdolgozatomnál is hatalmas segítségnek bizonyult.

GitHub

A GitHub egy weboldal / szolgáltatás, amely a Git-et használó projekteknek biztosít egy online helyet tárolásra, megosztásra és csapatmunkára. A gyakorlati helyemen már ezt használták mikor odakerültem, így könnyen ment a munka az otthonukból dolgozó kollégák között. Mivel már sok projektemnél használtam, nem kellett a rendszert megtanulnom, gördülékenyen be tudtam csatlakozni a GitHub-on létrehozott csoportba és az ott tárolt projektekbe.

React Native

A React Native egy olyan technológia mely lehetővé teszi, hogy a React keretrendszer segítségével natív alkalmazásokat fejleszthessünk JavaScript vagy TypeScript nyelven. Mivel a feladatom egy több platformon működő natív alkalmazás készítése, ez volt a tökéletes választás a megvalósításhoz. A kollégák számára is ismerős programnyelven tudtam dolgozni és a munkámat natív alkalmazás formájában tesztelni. A projekt könnyen fordítható iOS-re, Android-ra és Webre is, bár jelen esetben csak az első kettő platform volt a cél.

XAMPP

A XAMPP egy ingyenes szoftver csomag, mely szerverek fejlesztéséhez nyújt hasznos eszközöket. A munkám során használt szoftverek között ez a legidősebb, 2002-ben adták ki az első verzióját. Tartalmaz Apache HTTP szervert PHP támogatással, MariaDB adatbázist és egyéb szoftvereket. A projekt esetében a HTTP szervert használtam, hogy egy prototípus API-t készítsek a fejlesztett alkalmazáshoz.

Postman

A Postman egy API platform, amely segít egy API fejlesztésének minden lépésében. Nagyban leegyszerüsíti a tesztelést és csapatmunkát, gyakran használtam a PHP-ban írt prototípus API készítése közben, hogy ellenőrizzem, hogy megfelelően működik mielőtt az alkalmazást build-elem. Így gyorsan ki tudtam szűrni a hibákat és gyorsan ment a fejlesztés.

Xcode

Az Xcode az Apple saját készítésű integrált fejlesztői környezete, amit a macOS, iOS, iPadOS, watchOS és tvOS-re való szoftverek gyártásához használnak. Enélkül nem lehet iOS alkalmazást build-elni, ezért fontos volt, hogy a projektem iPhone készülékekre is meg tudjam valósítani. Viszont ezt a programot csak macOS-re adták ki és Windows számítógép és egy Ubuntu operációs rendszerű laptop birtokában kreatív megoldásokhoz kellett folyamodnom.

Android Studio

Az Android Studio a hivatalos integrált fejlesztői környezet a Google Android alapú eszközeihez. A JetBrains nevű cég fejlesztette az IntelliJ programjukra építve. Minden asztali operációs rendszeren elérhető. Tartalmazza az ADB-t (Android Debugging Bridge) és a Gradle build eszközöket, amelyek elengedhetetlenek voltak számomra a fejlesztés során, ha éles eszközön próbáltam ki a projektem. Android emulátort is tartalmaz, de jobban preferálom a fizikai eszközöket és direkt fejlesztési célre egy teszt eszközzel is rendelkezem. Az Android Studio környezetet és az Android rendszert már jól ismerem, mert hobbi szinten már 2015 óta készítek egyre komplexebb alkalmazásokat.

Firebase

A Firebase egy backend szolgáltatás, amely 2012-ben indult és a Google 2014-ben vásárolt fel. Segítségével több hasznos funkcióval lehet ellátni bármilyen alkalmazást, legyen az natív app, weboldal, asztali program vagy játék. A nyújtott szolgáltatások között van felhasználó kezelés, két féle adatbázis, fájl tárhely, hírdetések, statusztika gyűjtés, összeomlás napló, értesítés kezelés és még sok más. Jelen projektben csak az FCM nevű szolgáltatást vettem igénybe.

FCM

Az FCM (azaz Firebase Cloud Messaging) szolgáltatás képes értesítéseket küldeni egy API segítségével a mi alkalmazásunkat használó eszközökre, legyen az bármilyen készülék, bármilyen operációs rendszerrel. Ha egy fejlesztő egyszerűen szeretne értesítést küldeni felhasználóinak, ez a leg kézenfekvőbb módja, hogy megtegye. Android és iOS rendszereken is megoldható, hogy az alkalmazás akkor is tudjon üzeneteket fogadni, mikor az nincs az előtérben, Android-on mindezt plusz erőforrások használata nélkül, így tökéletes választás volt a projekthez.

Használt programnyelvek

JavaScript

A JavaSciprt a web programnyelve, de egyre több helyen használják a weben kívül is. Electron-nal asztali alkalmazások fejlesztésére, Node.js-sel vagy Deno-val szerver oldali logika készítésére és természetesen React

Native-vel natív alkalmazások fejlesztésére. A React Native egy új, szinte ismeretlen technológia volt még számomra. Mikor elkezdtem tanulni egyelőre az ismerősebb JavaScript nyelvvel ismertem meg az alapokat. Hamar megértettem, így át is váltottam a cél programnyelvre, amiben is a projektet készítettem el.

TypeScript

A TypeScipt egy 2012-ben a Microsoft által készített programnyelv, amit a legegyszerűbben úgy lehet leírni, hogy JavaScript típusokkal. A cégnél a TypeScript az elsődleges nyelv a frontend fejlesztésekhez, ezért én is ebben terveztem az alkalmazást megvalósítani. A React Native ezt is nagyon jól támogatja.

Swift

A Swift az Apple és a közösség által fejlesztett programnyelv, amit 2014-ben az Objective-C nyelv leváltására terveztek. Hogy a fejlesztőknek akadálymentes legyen az átmenet, az Xcode-ba az Apple olyan fordítót tett, amely lehetővé tette, hogy egy alkalmazásban C, Objectuve-C, C++ és Swift kód is legyen.

A React Native elsődlehes nyelve a JavaScript vagy a TypeScript, de ha az alkalmazásunkba szeretnénk operációs rendszerhez közelebbi natív funkciókat, akkor ezt sem árt ismerni. A projekt során nem kellett sokszor natív kódhoz nyúlni, de kísérleteztem vele, hogy bővítsem az ismereteim az Apple eszközökre való fejlesztés terén.

Kotlin

A Kotlin egy JetBrains cég által létrehozott programnyelv, ami a Szentpétervár közelében lévő Kotlin-szigetről kapta a nevét. Képes Java bytecode-ra és JavaScript-re is fordulni és képes együttműködni a Java programnyelvvel. 2011-ben jelent meg és később ez lett az Android alkalmazások fejlesztéséhez elsődlegesen ajánlott programnyelv. A projekt alatt ezzel sem találkoztam gyakran, viszont egy egyszerű prototípus elkészítéséhez tökéletes volt, mert már sok tapasztalatom van ezzel a nyelvvel, gyorsan össze tudok rakni bármilyen egyszerűbb alkalmazást. Az évek során ez lett a kedvenc programozási nyelvem.

PHP

A PHP (PHP: Hypertext Preprocessor) a JavaScipt-hez hasonlóan egy 1995-ös webhez készült nyelv. Elsődlegesen szerver oldali logikához használják a mai napig is. Bár a fiatal fejlesztők próbálják elkerülni és új technológiákat fejleszteni a helyére, mint a följebb említett Node.js és Deno, a PHP mostanság is nagyon népszerű sok weboldalon és rendszerben. Nekem is ezt a nyelvet kellett használnom, hogy elkészítsem az alkalmazáshoz tartozó szerver oldali logikát bizonyos funkciókhoz, mint például a biztonságos bejelentkezéshez biometrikus azonosítással.

munka home-office-ban

Az Idealap Kft. teljesen home-office-ban működik. Minden alkalmazott a saját otthonából dolgozik. Ennek sok előnye van, de felmerülhet a kérdés, hogy így hogyan ellenőrizhető a munka és hogyan lehet fenntartani a motivációt. Ezekre a kérdésekre a cég egész jó válaszokat adott. Például ahogy a használt eszközöknél említettem, az Idealapnál GitHub-ot használnak. A GitHub számon tartja, hogy ki, mikor és hány sor kódot módosított egy projekten. Viszont az elszámolást az órák alapján végzik, szóval a GitHub nem megfelelő erre a célra.

Idő követő rendszer

Az Idealapnak van egy házilag készített rendszere arra, hogy kövessük ki hány órát dolgozott egy feladaton. Ebben a rendszerben kaptam meg én is a feladataim, a specifikációt és sok egyéb információt a cégről, napi munka menetéről és néhány eszköz (GitHub és PHPStorm) beállításáról. Mikor a projekten dolgoztam, tervezgettem, kutattam vagy gondolkodtam, hogy bizonyos problémákat hogyan lehetne megoldani, mindig mértem az időt, amit a nap végén beírtam a rendszerbe. A meeting-eket és bemutatókat is ugyanígy számoltuk.

Fejlődés és motiváció fenntartása

Mikor az otthon és a munkahely ugyanaz az hely, nehéz motiváltnak maradni. Az Idealapnak van egy jó szokása ennek és a jó hangulatnak fenntartására, amivel még fejlődhetünk is a szakmában.

- A hónap minden péntekje "offline péntek", amikor mindenki a saját dolgával foglalkozik és nem vesz részt online egyeztetéseken. Ez alól csak a cégvezetés kivétel, ha a partner pénteken akar meetingelni. A szabad péntek szabad.
- Minden hónapban van egy "tanulós péntek", amikor valami olyat kell csinálni, amitől ügyesebbek és / vagy boldogabbak leszünk a munka vagy a szakma terén. A tanulós péntekről pár soros összefoglalót kell adni a kollégáknak a közös céges chat-en vagy emailben, vagy prezentálni online a többieknek.
- Illetve minden hónap utolsó péntekje "szabad péntek", amikor csak ügyeletet kell biztosítani, de nem kötelező dolgozni. Ez nem számít szabadságnak.
- Amikor ledolgozandó munkanap szombatra esik, az szabad pénteknek számít

Szerintem ezek nagyon jó szokások és mindig látszott, hogy jó a hangulat a csoportban.

kommunikáció

A cégnél elsődlegesen a Google szolgáltatásokat használják. Google Meet, Gmail, a Gmail-be épített Google Chat. Chat-en értem el a cég vezetőjét is, ahol megtervezett időközönként megbeszéltük a projekttel való haladást és egyéb kisebb dolgokat.

Hosszabb megbeszélésekhez vagy bemutatókhoz előre szervezett eseményt adtunk hozzá a közös Google Naptárhoz, amely egy Meet linket is tartalmazott. Ezeknél a bemutatóknál egyszerűen képernyő osztással körbe tudtam vezetni a vezetőt a projekt aktuális állapotáról.

idealApp cross-platform alkalmazáskeret

specifikáció

A cél egy olyan alkalmazás létrehozása React Native technológiával, amelybe bármilyen webalkalmazás beilleszthető és el tudja látni natív funkciókkal. Az elvárt funkciók:

- biztonságos bejelentkezés biometrikus azonosítás formájában az eszköz elérhető szenzoraival (iOS eszközök esetén TouchID vagy FaceID és Android esetén ujjlenyomat vagy arcfelismerés)
- értesítések fogadása egy szerverről még akkor is, amikor az alkalmazás nem fut az előtérben.

Fontos, hogy az alkalmazás mindkét fő telefonos operációs rendszeren ugyanúgy működjön és egy kódbázisból build-elhető legyen. React Native technológiát választottuk, hisz ez a leg kézenfekvőbb és legelterjetdebb módja egy cross-platform alkalmazás elkészítésének. A fő programnyelv pedig TypeScript a modern funkciói, típus biztonság és a kollégák tapasztalata miatt.

feladataim

A hosszútávú feladatom az alkalmazás elkészítése a följebb leírt specifikáció alapján. Feladataim közé tartozott még:

- elsajátítani az elvárt technológiákat és programnyelveket, hogy tudjak velük dolgozni
- kutatást végezni, hogy egyáltalán kivitelezhető-e a terv
    - ha igen, hogyan?
    - ha nem, miért nem és hogyan lehet az akadályt elkerülni?
- beszámolni bizonyos időközönként a haladásomról és a projekt aktuális állapotáról

A React Native és TypeScript nyelv tanulásához a cégtől segítséget is kaptam. Egy Udemy kurzust követtem az első hetek során. Így könnyen el tudtam kezdeni a projekt fejlesztését.

A kutatásokat és akadályokra megoldás keresését önállóan végeztem és szinte mindig megtaláltam a módot, hogy hogyan kivitelezhető az adott funkció. A megoldásaim mindig kipróbáltam egy prototípus alkalmazásban és a megbeszéléseken is mindig beszámoltam ezekről.

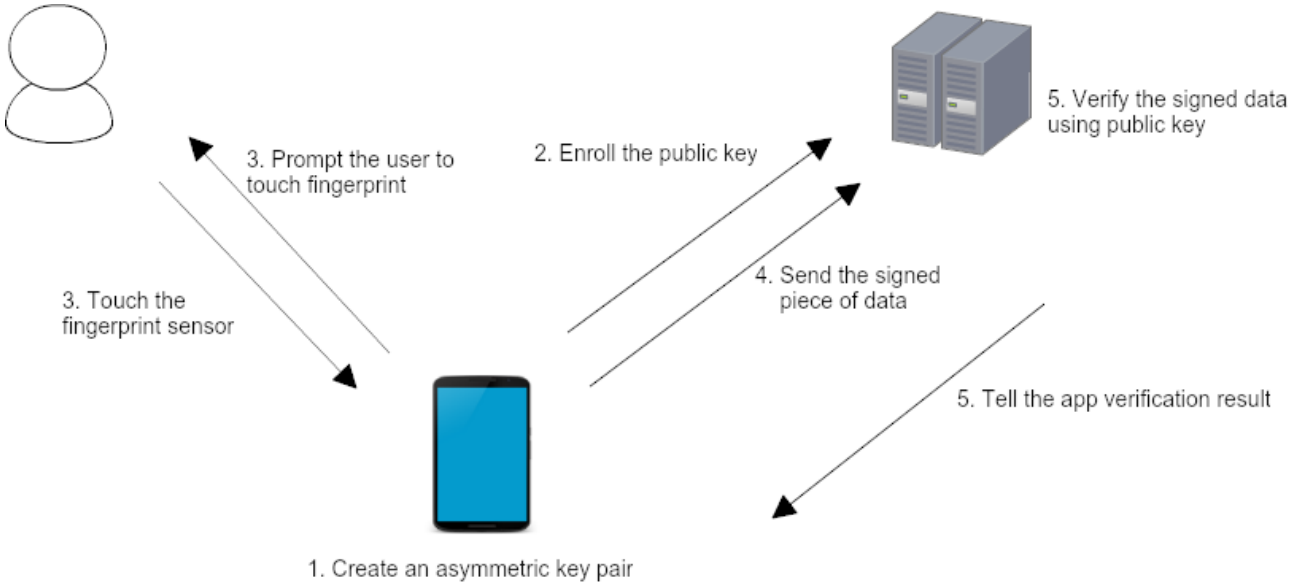
kliens

fontosabb használt könyvtárak

react-native-device-info

react-native-biometrics

szerver



Kép forrása: https://github.com/SelfLender/react-native-biometrics

cél

dokumentáció