1. Východisková kapitola

Prvá kapitola záverečnej práce bude mať deskriptívny charakter. Prostredníctvom údajov, ktoré budeme uvádzať, sa oboznámime s definíciou termínu "didaktický softvér" a taktiež aj s jednotlivými princípmi, ktoré musí daný softvér spĺňať. Ako vyplýva z obsahu termínu "didaktický", budeme sa sústrediť aj na spôsob, akým zmienený softvér prispieva k rozvíjaniu poznatkov jednotlivca. Úvodná kapitola nebude slúžiť len k vysvetleniu použitých technológií, ale jej podstatnými zložkami bude taktiež aj definovanie už existujúcich a dostupných technológií a už existujúcich a podobných analogických aplikácií. V jej obsahu sa zameriame aj na ciele práce definujúce motiváciu a výsledný zámer tvorby aplikácie. Jednotlivé uvedené termíny, ktorými sa budeme zaoberať a podrobne vysvetlíme princíp ich fungovania, sú nevyhnutnými zložkami, bez ktorých by bola realizácia aplikácie nemožná (prípadne by nespĺňala nami vytýčené parametre), a teda by mohla byť vo výsledku označená ako "nefunkčná" – nespĺňajúca jej primárny účel. Výsledná aplikácia, ktorá bude výstupom tento záverečnej práce, bude určená primárne pre operačný systém Android (avšak budúcou možnosťou rozšírenia aplikácie aj na operačný systém iOS). Jej užívanie bude primárne určené novým návštevníkom fitness centier, ktorým bude napomáhať v efektívnom využívaní času pre maximalizáciu ich cvičebných výsledkov.

1.1. Ciele práce

Cieľom bakalárskej práce je navrhnúť a implementovať interaktívne mobilné prostredie pre návštevníkov fitness centier. Prostredie poskytne registrovaným používateľom komplexný prehľad o cvičeniach (inštruktážne animácie a obrázky), umožní im tvoriť si časový plán vrátane push up notifikácií, poznámky k cvičeniam, generovať poradie cvičení, individuálny cvičebný plán a pod. Aplikácia bude uchovávať históriu cvičení a generovať štatistiky. Umožní prácu v offline režime a online zálohu pomocou Firebase. Aplikácia umožní registráciu / prihlásenie cez Google služby a bude realizovaná pomocou technológii / nástrojov: Flutter (Dart), Firebase (NoSQL).

1.2. Využité technológie

1.2.1. Dart

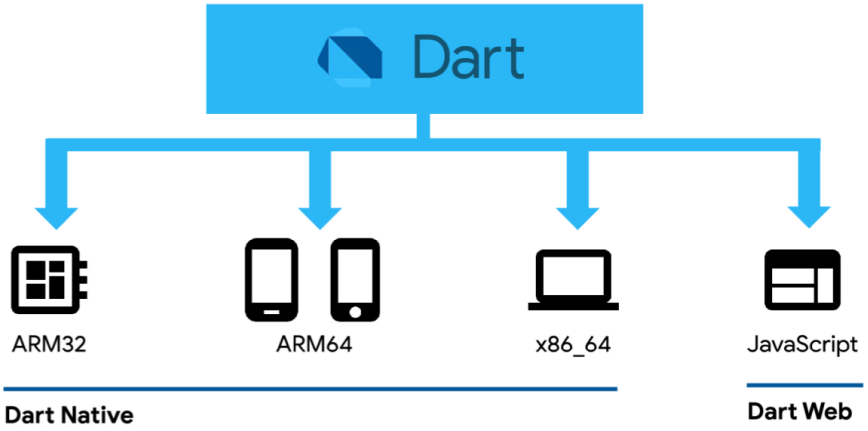
Dart je objektovo orientovaný programovací jazyk ktorý vznikol v roku 2011 a bol vyvinutý Googlom - nejedná sa teda z pohľadu programovacieho jazyka o prevratnú novinku ale o preverenú technológiu. Primárne bol vytvorený za účelom nahradiť JavaScript a slúži na tvorbu webových a mobilných aplikácii. V súčasnosti sa dá použiť aj na tvorbu desktopových aplikácii. Dart bol ovplyvnený jazykmi ako Java, JavaScript, C#, Kotlin a TypeScript. Obťažnosť naučenia sa tohoto jazyka tým pádom nie je náročná pre vývojára natívnych platforiem. Má podobnú syntax ako jazyky z rodiny C. Naša práca je Flutter mobilná aplikácia, Flutter aplikácie sú napísané v jazyku Dart.

1.2.2. Flutter

Flutter je framework a SDK, ktorý bol vydaný v roku 2017. Od svojho vydania patrí medzi najrýchlejšie rastúce crossplatformové riešenia.

Flutter vytvoril Google ako open source crossplatformovú knižnicu pre budovanie používateľského rozhrania, ktorá sa natívne kompiluje pre mobilné zariadenia, web a desktop aplikácie vytvorením jedného kódu. Flutter ako jediná crossplatformová technológia využíva na renderovanie vlastný engine, pomocou ktorého sa spúšťa beh celej aplikácie. Keďže je Flutter kompilovaný do natívneho kódu, k vývoju aplikácií nie je potrebná žiadna externá aplikácia alebo služba ako pri React Native.

Programovanie je tak vykonávané iba raz. V budúcnosti sa však počíta s Flutterom ako jediným prístupom k vývoju na platforme Google Fuchsia, ktorá by mala v budúcnosti nahradiť celý Android.

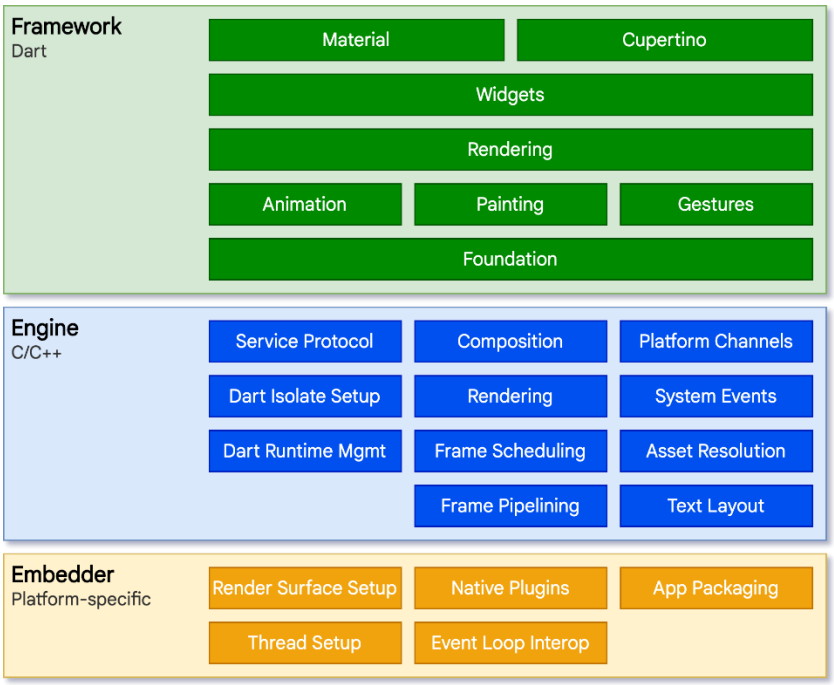


[1] Kompilácia programovacieho jazyka Dart do natívneho kódu

Pre natívne platformy (mobilné a desktop aplikácie) Dart obsahuje Dart VM s JIT (just-in-time) kompilátorom a zároveň AOT (ahead-oftime) kompilátorom, ktorý vytvorí natívny kód pre danú platformu. Pre webovú platformu Dart obsahuje dart2js kompilátor, ktorý preloží Dart kód priamo do JavaScriptu. Vďaka jazyku Dart sa Flutter môže kompilovať priamo do binárneho natívneho kódu platforiem iOS, Android, Fuchsia, macOS, Windows, Linux a web. Flutter teda nevyužíva natívne komponenty a SDK danej platformy, ale je skompilovaný do natívneho binárneho kódu platformy, čím má výhodu rýchlosti vykonávania inštrukcií oproti riešeniam ako napríklad React Native.

Aplikácie pomocou technológie Flutter je možné vyvíjať na všetkých desktopových operačných systémoch Windows, macOS a Linux. Keďže sa Flutter kompiluje do natívneho kódu, pre možný vývoj pre iOS je potrebný macOS. Bez macOS nie je možné spustiť iOS aplikáciu na simulátore alebo fyzickom zariadení. Pre Flutter vývoj je potrebné nainštalovať SDK a použiť jedno z vývojových prostredí Android Studio alebo Visual Studio Code, do ktorých je následné potrebné nainštalovať Flutter podporu. Flutter tím obe vývojové prostredia aktívne podporuje [2].

Flutter je technológia pre vytváranie používateľského rozhrania, ktorá umožňuje zdieľať kód medzi operačnými systémami a zároveň umožňuje komunikovať so službami danej platformy. Počas vývoja Flutter využíva Dart VM, ktorý umožňuje prejaviť zmeny v kóde pomocou technológie hot reload bez potreby kompilovať kód na novo [3]. Vyvíjanie aplikácie je tak pre vývojára rýchlejšie, pretože počas programovania nie je potrebné po zmenách kód kompilovať na novo a zmeny sa prejavia v bežiacej skompilovanej aplikácii.



[4] Prehľad Flutter architektúry

1.2.3. Widgety

Z vývojárskeho hľadiska je vo Flutteri všetko widget. Je to stavebný blok všetkého vo Flutteri. Widget slúži ako koncept pre vykresľovanie grafických elementov na obrazovku, rozloženie elementov (pozícia a veľkosť), používateľská interakcia, animácie, navigácia alebo kompletné spravovanie stavu aplikácie. Widgety v celej aplikácii vytvárajú strom. V natívnych knižniciach sa grafické elementy dedia od abstraktných tried, v ktorých sa následne upravuje vzhľad a správanie daného elementu. Vo Flutteri sa k riešeniu pristupuje naopak a používateľské rozhranie je tvorené kompozíciou widgetov. To znamená, že namiesto dedenia tried abstraktných elementov, sa widget skladá z viacero widgetov, medzi ktorými je vždy vzťah rodič-dieťa. Každý widget sa teda snaží vykonať iba jednu určitú funkcionalitu, ktorú je potom možné spojiť do väčšieho stromu vzťahov. Takýmto spôsobom je vytvorená celá obrazovka používateľského rozhrania. Vzhľadom na to, že syn widgetu je vždy widget, je jednoduché tvoriť dynamické elementy bez obmedzení. Tvorenie používateľského rozhrania je tak oveľa jednoduchšie a rýchlejšie ako pri potrebe dedenia určitých typov. Kompozícia tak dáva väčší zmysel pri vytváraní používateľského rozhrania ako objektovo orientované dedenie.

```
Widget build(BuildContext context) {
    return SafeArea(
        child: Scaffold(
            appBar: AppBar(
                title: Text('Nazov obrazovky'),
            ),
            body: Container(
                child: Center(
                    child: Text('Text v strede
                       obrazovky'),
                ),
            ),
        ),
    );
}
```

V ukážke Flutter kóduje vytvorená celá obrazovka, ktorá na vrchu obrazovky zobrazuje navigačný bar a v strede obrazovky je zobrazený text. SafeArea je widget, ktorý je zodpovedný za ohraničenie dostupného miesta obrazovky, kde sa môže vykresľovať grafické rozhranie. Scaffold je tiež widget, ktorý je zodpovedný za umiestnenie ďalších widgetov v podstrome, tvorí hlavnú kostru mobilnej aplikácie. AppBar widget vykresľuje navigačný bar v hornej časti obrazovky. Widget patrí medzi widgety, ktoré sú implementované tak, aby vyzerali natívne na platforme iOS aj Android. Center widget rieši umiestnenie v rámci dostupnej obrazovky na stred. Text je widget, ktorý na obrazovku vykresľuje zadaný text. Takýmto spôsobom je možné vytvoriť celú aplikáciu.

Vývojár si môže widget neimplementovať sám a znova používať v rôznych častiach stromov. Flutter už ale predpripravil veľké množstvo implementovaných widgetov, ktoré vývojár používa pre implementovanie grafického rozhrania. V architektúre Flutteru sa nachádzajú v blokoch Material a Cupertino. Material je sada widgetov, ktoré implementujú widget tak, aby vyzerali rovnako ako natívne grafické elementy pre Android a Cupertino je sada implementovaná pre iOS [5].

1.2.4. Firebase

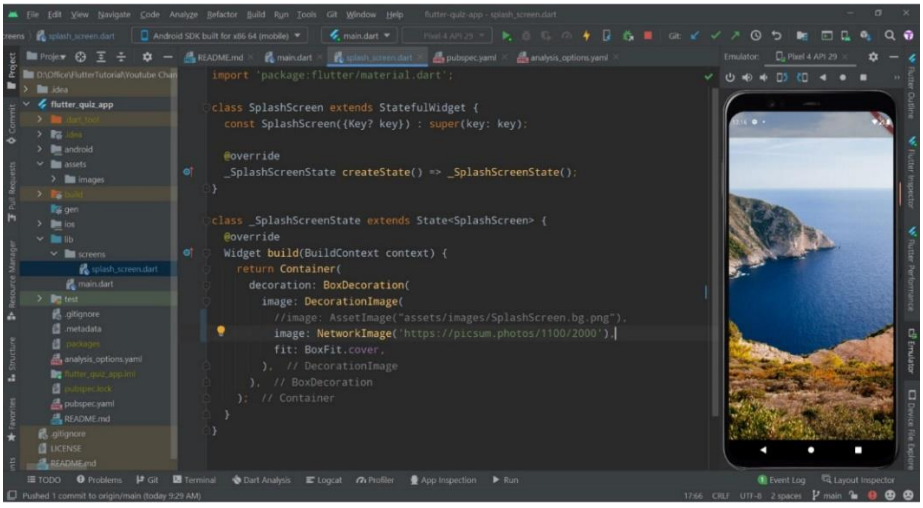
V bakalárskej práci sme sa rozhodli používať služby Firebase od spoločnosti Google. Firebase je vyvinutý na tvorbu mobilných a webových aplikácii a je dobrý nástroj na vytváranie, vylepšovanie a udržiavanie aplikácie, v ktorej je použitý. Firebase má predpripravené funkcionality, ktoré veľa developerov nerado programuje od základu, ale radšej by sa zamerali na aplikáciu ako celok. Tieto funkcionality sú: analytika,

autentifikácia, databáza, konfigurácia aplikácie, úložisko súborov atď.. V tejto práci používame autentifikáciu a databázu, ktoré sú súčasťou Cloud Firestore. Celý Firebase je hostovaný v Cloude čo uľahčuje komunikáciu aplikácie s Firebase. [6]

[7] Služby ktoré ponúka Firebase

1.2.5. Android Studio

Vyvíjať Flutter aplikácie je možné pomocou vývojových prostredí Android Studio a Visual Studio Code. Android Studio vyvíja Google, rovnako ako aj Flutter. Po stiahnutí Flutter nadstavby pre Android Studio sú dostupné všetky Flutter nástroje priamo vo vývojovom prostredí. Je možné vyvíjať na fyzické zariadenia a zároveň spúšťať všetky dostupné simulátory pre iOS a Android. Android Studio je tak vhodným vývojovým prostredím pre vytvorenie aplikácie.

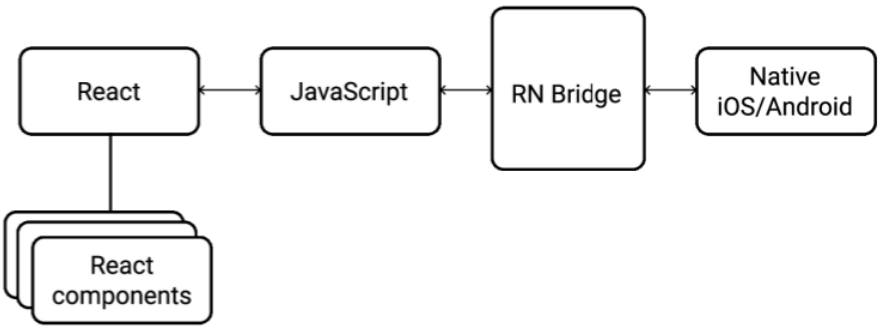


[8] Flutter v prostredí Android Studio

1.3. Podobné existujúce technológie

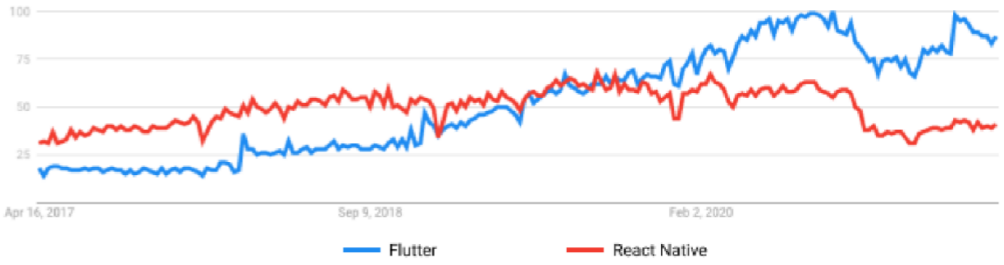
1.3.1. React Native

React Native je framework na budovanie aplikácií s použitím Reactu. Vydaný bol v roku 2015 spoločnosťou Facebook. Celý projekt je open source a je aktívne vyvíjaný komunitou a spoločnosťou Facebook. React Native využíva JavaScript a je založený na knižnici ReactJS, ktorá je najpoužívanejšou technológiou pre vývoj webových aplikácií.



[9] Architektúra React Native

React Native vytvára ďalšiu vrstvu nad natívnym prístupom. Využíva React komponenty [10], ktoré predstavujú natívne elementy z iOS a Android vytvorením jednotného API. Knižnica React Native implementuje RN Bridge [11], ktorý vytvára vrstvu medzi JavaScript a natívnym kódom daného operačného systému. RN Bridge následné volá príslušné metódy implementované v natívnom kóde s použitím SDK platformy. Väčšina grafických natívnych elementov teda majú svoju jednotnú implementáciu pre iOS a Android implementovanú pomocou React komponent. Výhodou tohoto prístupu je aplikácia, ktorá obsahuje natívne elementy a používateľ s aplikáciou pracuje rovnako ako s natívnymi aplikáciami. Nevýhodou je, že každá interakcia v aplikácii a všetky udalosti sa musia posielať a spracovávať cez ďalšiu vrstvu, čo v dôsledku spomaľuje výkon aplikácie a je potrebné aplikácie veľmi dobre optimalizovať. Aplikácie si navyše pri samotnom štarte aplikácie musia nainicializovať celú JavaScript vrstvu, čo má za dôsledok dlhší čas načítavania aplikácie až v sekundách navyše. [12]

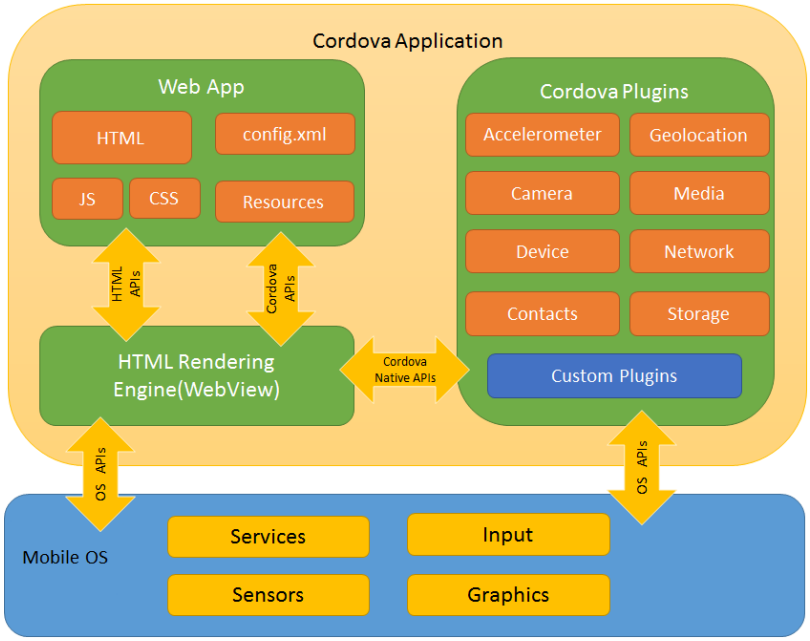


[13] Porovnanie popularity knižníc Flutter a React Native. Graf predstavuje počet vyhľadávaní v Google.

Prvá oficiálna stabilná verzia Flutter 1.0 bola vydaná v decembri 2018. Podľa grafu je vidieť ako popularita rýchlo rastie a v roku 2020 v raste prekonala React Native. Flutter je rovnako ako React Native crossplaformová technológia, ale k riešeniu pristupuje iným prístupom. Flutter nevytvára žiadny bridge a nevyužíva natívne komponenty. Flutter implementuje vlastný renderovací systém. Flutter je vytvorený pomocou programovacích jazykov C, C++, Dart a využíva 2D renderovací engine Skia, ktorý využíva aj webový prehliadač Chrome. Flutter teda nevyužíva natívne renderovacie knižnice a SDK, ale celé používateľské rozhranie natívnych grafických prvkov implementuje na novo. Flutter obsahuje sadu predimplementovaných elementov grafického rozhrania, ktoré simulujú natívne grafické elementy.

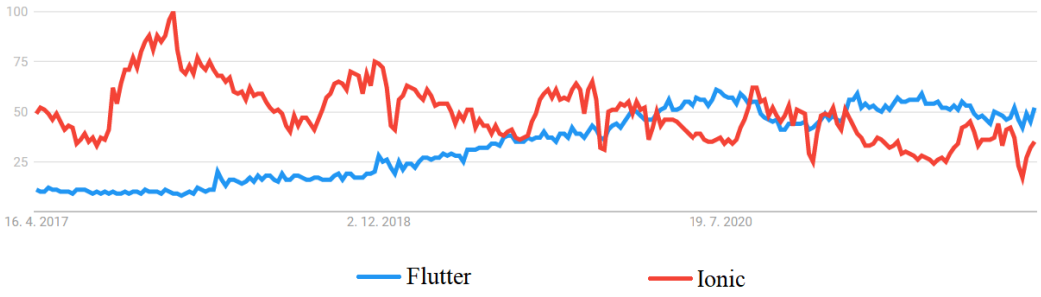
1.3.2. Ionic

Ionic je open-source nástroj na vývoj crossplatformových aplikácii za pomoci HTML5, CSS a JS. Pomocou Ionic-u vieme vyvíjať vysoko interaktívne mobilné aplikácie. Efektívny výkon Ionic-u spočíva v minimálnej DOM manipulácii a hardwarom urýchlené prechody. Ionic využíva Angular ako JS framwerok. Ionic má veľmi dobrú integráciu s Cordova's device API, čo prináša prístup API's zariadenia, takže môžeme použiť Ionic Native s Ionic komponentami na tvorbu UI. Ionic má vlastný CLI, ktorý slúži na konštrukciu, vývoj a testovanie Ionic aplikácií.



[14] Diagram Cordova aplikácie

Na natívne nasadenie využíva Cordova, ale beží v prehliadači ako progresívna webová aplikácia. Staršie vydania Ionicu boli pevne spojené s Angularom. Ionic bol prerobený, aby vedel fungovať ako samostatná knižnica web komponentov s integráciou najnovšieho JS frameworku ako Angular. Ionic vieme použiť s viacerými front-end frameworkami, ako napríklad React alebo Vue.js. Angular bol vždy hlavnou zložkou Ionicu, dokonca až 2/3 Ionicu sú prevzaté z Angularu, ako napríklad známa knižnica Angular Router. [15, 16]



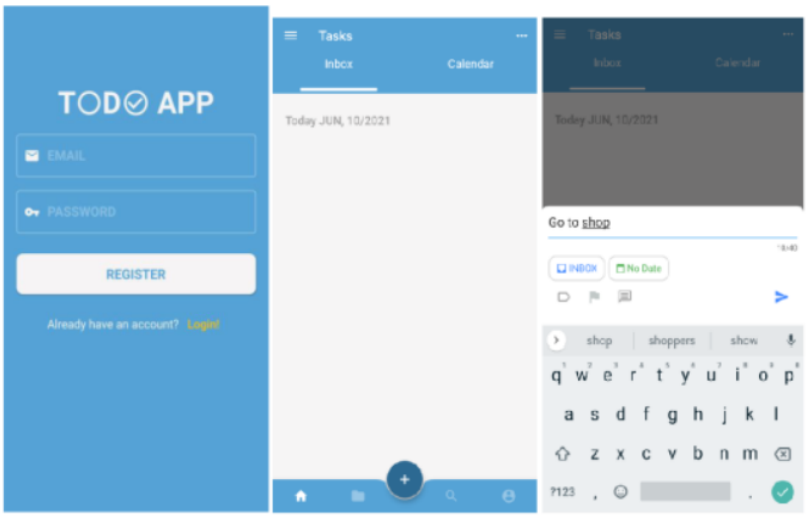
[17] Porovnanie popularity knižníc Flutter a Ionic. Graf predstavuje počet vyhľadávaní v Google.

Prvá oficiálna stabilná verzia Flutter 1.0 bola vydaná v decembri 2018. Podľa grafu je vidieť ako popularita rýchlo rastie a v roku 2020 v raste prekonala Ionic. Potom zase spadla a od roku 2021 Flutter znova vedie v popularite.

1.4. Podobné existujúce aplikácie

1.4.1. Todo aplikácia od Tamary

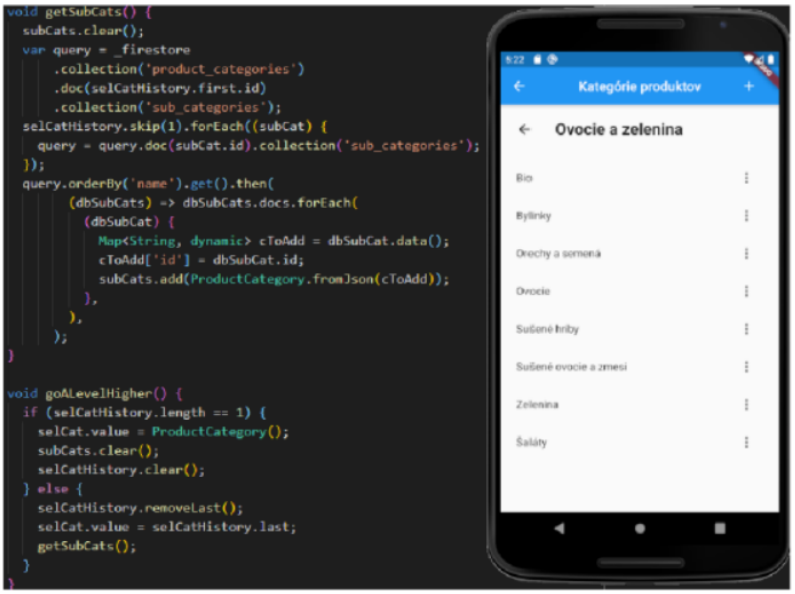
Táto Todo aplikácia [18] je určená na spracovanie plánov počas dňa. Aplikácia síce nesúvisí s Fitness a cvičením, ale za to je postavená na frameworku Flutter a využíva NoSQL databázu Firebase. Takže využíva rovnakú technológiu. Autorka mala zo začiatku problém napojiť sa na Firebase databázu, ale poučili sme sa z jej chýb a napojenie zvládli na prvý krát úspešne.



[18] Dizajn Todo aplikácie

1.4.2. Aplikácia Quick Shop od Samuela

Aplikácia Quick Shop [19] slúži na rýchlu orientáciu hlavne vo veľkých, ale aj malých obchodoch. Taktiež nijako nesúvisí s Fitness ale využíva rovnakú technológiu ako naša aplikácia. Vďaka autorovej práci sme mohli lepšie pochopiť prácu s databázou Firebase a asynchrónnym programovaným.



[19] Dizajn aplikácie Quick Shop

2. Požité zdroje:

1. Dostupné online na: https://dart.dev/overview
2. Set up an editor. Google, 2021. Dostupné online na: https://flutter.dev/docs/get-started/editor?tab=androidstudio
3. Measuring Performance. Apple , 2021. Dostupné online na: https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/MeasuringPerformance.html
4. Flutter architectural layers. Google, 2021. Dostupné online na: https://docs.flutter.dev/resources/architectural-overview#architectural-layers
5. Widgets catalog. Google, 2021. Dostupné online na: https://flutter.dev/docs/development/ui/widgets
6. Dostupné online na: https://www.freecodecamp.org/news/how-to-get-started-using-firebase-hosting-439d4bd45cb6/
7. Dostupné online na: https://dev.to/caelinsutch/what-is-firebase-1acj
8. Dostupné online na: https://fluttertutorial.net/category/how-to/
9. Dostupné online na: https://dev.to/goodpic/understanding-react-native-architecture-22hh
10. React Native components, reactnative, 2021. Dostupné online na: https://reactnative.dev/docs/componentsand-apis
11. React Native communication bridge, reactnative, 2021. Dostupné online na: https://reactnative.dev/docs/communication-ios
12. How we improved our react native cold start for android. mattermost, 2021. Dostupné online na: https://mattermost.com/blog/how-we-improved-our-react-native-cold-startfor-android/
13. Google Trends compare interest over time. Google, 2022. Dostupné online na: https://trends.google.com/trends/explore?date=2017-04-14%202022-01-23&q=Flutter,%2Fg%2F11h03gfxy9
14. Dostupné online na: https://cordova.apache.org/docs/en/10.x/guide/overview/
15. RAVULAVARU, Arvind. *Learning Ionic*. 2. vyd. Packt Publishing Ltd., 2017-01-04. ISBN 978-1-786546-605-1. Dostupné online na: https://www.packtpub.com/
16. NEZNÁMY (ed.). *Ionic Framework*. Ionic Framework. Dostupné online na: https://ionicframework.com/docs
17. Google Trends compare interest over time. Google, 2022. Dostupné online na: https://trends.google.com/trends/explore?date=2017-04-14%202022-01-23&q=flutter,ionic
18. PANAETOVA, Tamara. *Cross-platformová mobilná Todo aplikácia pomocou frameworku Flutter*. 2021. Dostupné online na: https://opac.crzp.sk/?fn=detailBiblioForm&sid=CF9E62B1F4156FE3B62E0884D7C8&seo=CRZP-detail-kniha
19. KOBERA, Samuel. *Aplikácia pre navigáciu kupujúcich v obchodoch*. 2021. Dostupné online na: https://opac.crzp.sk/?fn=detailBiblioForm&sid=38F43C5054BC4F7F29BCC2F31F28&seo=CRZP-detail-kniha