Mobilní aplikace usnadňující hledání bydlení a spolubydlení

Filip Tomeš

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Filip Tomeš**
Osobní číslo:	**A19116**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Mobilní aplikace usnadňující hledání bydlení a spolubydlení**
Téma práce anglicky:	**Mobile Application Facilitating the Search for Housing and Roommates**

Zásady pro vypracování

1. Nastudujte a rozepište problematiku spojenou s nativním vývojem mobilních aplikací.
2. Vytvořte návrh aplikace a zvolte potřebné požadavky pro návrh aplikace dle tématu práce.
3. Vyberte vhodné technologie a prostředky pro implementaci aplikace ulehčující pro hledání ubytování studentů.
4. Vámi navrženou aplikaci implementujte.
5. Vytvořenou aplikaci vhodně otestujte.
6. Výsledky práce vhodně prezentujte a vyhodnoťte.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. ZAPATA, Belén Cruz a Antonio Hernández NIÑIROLA. Testing and securing Android Studio applications: debug and secure your Android applications with Android Studio. Birmingham: Packt Publishing, 2014, iv, 145 s. Community experience distilled. ISBN 9781783988808.
2. PHILLIPS, Bill, Chris STEWART, Brian HARDY a Kristin MARSICANO. Android programming: the Big Nerd Ranch guide. 2nd edition. Atlanta: Big Nerd Ranch, [2015], xxii, 618 s. ISBN 9780134171456.
3. LUMSDEN, Joanna. Emerging perspectives on the design, use, and evaluation of mobile and handheld devices. 1st edition. Hershey, Pennsylvania (701 E. Chocolate Avenue, Hershey, Pa., 17033, USA): IGI Global, 2015. ISBN 9781466685840.
4. MUKHERJEA, Sougata. Mobile application development, usability, and security. Hershey, Pennsylvania (701 E. Chocolate Avenue, Hershey, PA 17033, USA): IGI Global, 2017. ISBN 9781522509462.
5. BAHR, Benjamin. Prototyping of user interfaces for mobile applications. Cham, Switzerland: Springer, [2017], 1 online resource. T-labs series in telecommunication services. ISBN 9783319532103.

Vedoucí bakalářské práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů `zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 13.5.2022

Filip Tomeš, v.r.
podpis studenta

ABSTRAKT

Práca popisuje natívny vývoj mobilnej aplikácie pre operačný systém Android umožňujúcu zobrazenie, pridávanie či úpravu inzerátov so zameraním na obytné jednotky, ktorých údaje sú uložené pomocou cloudovej služby od Firebase. V prvej, teoretickej časti sa práca zaoberá porovnaním rôznych vývojových prístupov, získaním funkcionálnych požiadaviek, zostavenie návrhu a voľbou jednotlivých, aktuálne používaných technológií. Implementácia požiadaviek a tvorba užívateľského rozhrania s použitím najnovšieho súboru nástrojov Jetpack Compose je obsiahnutá v praktickej časti, rovnako ako aj testovanie komponentov a aplikácie ako hotového celku.

Klíčová slova: Android, Firebase, Jetpack Compose, Aplikácia, Manipulácia s inzerátmi

ABSTRACT

The work describes the native development of a mobile app for Android OS allowing showing, adding, or editing listed items focusing on residential units the data of which are saved using cloud service from Firebase. In the first theoretical part, the work deals with comparing various development attitudes, obtaining functional requirements, design creation, and choosing from different nowadays used technologies. The implementation of the chosen requirements and the creation of a user interface using the newest toolkit Jetpack Compose is obtained in the practical part of the work same as component and whole app testing.

Keywords: Android, Firebase, Jetpack Compose, Application, Listed items manipulation

Moja úprimná vďaka za venovaný čas, množstvom rád ohľadom textových úprav v obsahu a poskytnutých nápadov k zlepšeniu štruktúry celej práce patrí vedúcemu mojej bakalárskej práce Ing. Petrovi Žáčkovi, Ph.D.

Za rady a ochotu spojenú s vývojom aplikácie patria kolegovi Ing. Radovanovi Holíkovi, ktorého rady boli nápomocné už od samého začiatku.

Ani Fischerovej patrí špeciálna vďaka za samotný nápad a poskytnutý grafický návrh.

Rovnako by som rád poďakoval rodičom za ich chápavosť pri spracovávaní práce a kontrolu gramatiky.

Spolubývajúcim a kolegom ďakujem za príjemne strávené chvíle, ktorými som bol motivovaný k písaniu.

OBSAH

ÚVOD

Myšlienka tvoriť mobilnú aplikáciu namiesto webového rozhrania vznikla pretože každým rokom pribúda počet ľudí s dostupnosťou k elektrine, internetu a teda aj mobilných zariadení je pre mnohých jednoduchšie si kúpiť zariadenie v porovnateľne menšej cene ako keby si mali zadovážiť notebook alebo počítač, ktoré navyše ani nie sú také skladné a jednoducho prenosné. Väčšina užívateľov uskutočňuje badateľné množstvo úkonov pomocou ich telefónu, či už sa jedná o nakupovanie potravín, objednávanie rozličného tovaru, písanie si s rodinou a priateľmi a mnohé iné v relatívne blízkej minulosti nemožné činnosti.

Jednou z činností, ktorá minimálne v našich končinách študentom chýba je nájdenie si bývania na dlhší časový úsek. Existuje síce niekoľko webových portálov alebo aplikácií ponúkajúce takéto služby ale v oboch prípadoch sa jedná o buď zastarané systémy s nie optimálnym užívateľským rozhraním, alebo obsahujú inzeráty len od majiteľov daného objektu prípadne inzeráty od maklérov realitných kancelárií, ktorý často požadujú poplatky za sprostredkovanie. Zoznam týchto aplikácií je spomenutý v časti prieskum.

Týmto zbytočným poplatkom a takisto možnosť ponúknuť samotným nájomcom bytovej jednotky možnosť vytvoriť, upraviť alebo zmazať svoj inzerát, aby si dokázali jednoduchšou cestou nájsť nového spolubývajúceho je kľúčovou súčasťou konceptu aplikácie. Pre užívateľa, ktorý hľadá bývanie sú potom tieto inzeráty zobrazené ako príspevky na sociálnych sieťach. Autentizáciu užívateľov a databázu reálneho času pre inzeráty zabezpečuje cloudová platforma od Google – Firebase.

Práca sa v teoretickej časti zaoberá porovnaním natívneho vývoja s inými aktuálne dostupnými možnosťami vývoja na trhu a popísanie priebehu vývoja od prvotnej myšlienky až po publikáciu. Praktická časť je následne zameraná na zostavenie prvej verzie aplikácie branej ako koncept, ktorý slúži ako testovací model na vychytanie rôznych chýb na základe podnetov od vybraných ľudí, ktorým bude táto verzia poskytnutá.

I. TEORETICKÁ ČASŤ

1 PLÁNOVANIE A STRATÉGIA

Jednou z prvých fáz tvorby aplikácie je zaradená stratégia, toto spolu s plánovaním je popísané v tejto časti. Zozbieranie dát zo stávajúcich, konkurenčných aplikácií s podobnou tématikou je dôležité na vystihnutie určitých funkcionalít, ktoré by mohli byť súčasťou aplikácie a poslúžia teda ako forma inšpirácie.

1.1 Prieskum

Od prieskumu záleží, akým smerom sa bude uberať samotný vývoj. V tejto fáze je dôležité určiť si cieľovú skupinu užívateľov, prezrieť aktuálny stav trhu a zozbierať dáta konkurencie. [1]

1.1.1 Cieľová skupina

Vedieť, kto je budúcim zákazníkom je dôležitým prvkom, na ktorého základe sa stavia vizuálna identita, vývoj a marketingové stratégie. Táto skupina ľudí by mala zdieľať podobnosti v oblasti demografie, záujmov a problémov. [2]

1.1.2 Konkurenčný prieskum

Trh s mobilnými aplikáciami je miesto s miliónmi položkami dostupných k stiahnutiu, či už sa jedná o Google Play Store alebo Apple app store. Aby mal prvotný nápad šancu uspieť je nutné, aby samotná aplikácia dokázala byť konkurencie schopnou. K tomu je nutné spraviť analýzu aplikácií s podobnou myšlienkou. Výsledkom analýzy sú údaje, ktoré je možné využiť k zisteniu silných a slabých stránok súperov a ponúknuť tak analytický pohľad našej vízie potencionálnym investorom na demonštráciu pripravenosti na trh. [3]

- **Identifikácia konkurencie**

  K správne prevedenej analýze je potrebné jasne identifikovať divíziu, v ktorej sa nachádza nápad a teda na ňom založené aplikácie. Jednoduchým trikom je sa pozerať na trh očami bežného užívateľa, zadaním kľúčových slov do vyhľadávača v obchode s aplikáciami prehľadne zobrazí konkurentov z hľadanej oblasti. [3]
  Niektoré z najznámejších českých portálov, ktorých aplikácie sú často používané: Bezrealitky, Sreality.cz, Reality.cz a Realingo.

- **Unikátny aspekt**

  Je to hodnota aplikácie jedinečného charakteru, ktorá býva často kľúčom k úspechu. Nájdenie tohto aspektu napomáha k zabezpečeniu si lojálnosti užívateľov a pokrytie zo strany biznisu. Bez unikátnosti by mohla ľahko zapadnúť medzi ostatné aplikácie na trhu. [3]

1.2 Zostavenie požiadaviek a voľba funkcionalít

K udržaniu si dnešných, pomerne náročných užívateľov, je nutnosťou splniť určité štandardy v oblasti mobilných aplikácií a zvoliť také funkcie, aby pre nich mala aplikácia istú formu hodnoty.

Vývoju predchádza čo možno najdetailnejší popis všetkých požiadaviek na produkt, aby bolo popísané aké sú hlavné ciele, čo všetko má byť súčasťou a ako by mala vyzerať implementácia. [6] Nejasné požiadavky zvyšujú náklady a čas vývoja. [7]

V rannej fáze produktu sa odporúča sústrediť sa na jeden hlavný cieľ pre cieľovú skupinu a tým si vytvoriť základnú štruktúru do budúcnosti. [6]

1.2.1 Funkcionálne požiadavky

Predstavujú zoznam činností aplikácie a jej možné reakcie na rôzne vstupné informácie zadané užívateľom. [8, 9]

Keďže hlavným cieľom je navrhnúť aplikáciu zameranú na študentov hľadajúcich zväčša dlhodobý prenájom musia sa tomu prispôsobiť funkcionálne požiadavky.

1.2.2 Nefunkcionálne požiadavky

Existuje niekoľko odporúčaných vlastností, ktorých dodržanie pri vývoji vedie k väčšej šanci si udržať zákazníkov. Často sú zaraďované medzi nefunkcionálne požiadavky.

- ***Zrozumiteľné prostredie***

Strata záujmu z pohľadu užívateľa môže byť jednoducho spôsobená, pretože pozornosť ľudí nie je dokonalá a veľa z nás si už zvykla na určité opakujúce sa prvky v iných, viac známych aplikáciách. Je teda potrebné zachovať určité spôsoby rozloženia a nevytvárať zbytočne zložité a neprehľadné prostredie. [4]

- ***Výkon***

Rýchlosť a plynulosť je niečo, čo je dôležité takmer v každom odvetví. Aplikácie, ktoré trpia dlhým načítavaním obsahu nevytvárajú najlepší dojem na žiadneho používateľa, ktorý sa môže po takejto skúsenosti rozhodnúť hľadať inú, rýchlejšiu alternatívu. Je treba dbať na vhodný výber poskytovateľa hostingových služieb a vhodnú implementáciu získavania množstva dát, ktoré ovplyvňujú rýchlosť najčastejšie. [4]

- ***Bezpečnosť***

Vzhľadom na každoročne sa zvyšujúci počet útokov prevedených cez internet a teda aj samotné mobilné aplikácie. Mnohé z nich ukladajú citlivé informácie akými môžu byť údaje ku kreditným kartám, bankovým účtom, polohe užívateľa a iným. Z tohto hľadiska by mala byť bezpečnosť prvoradou záležitosťou pri vývoji, aby sa dokázalo v čo možno najväčšom množstve obmedziť stratu údajov. [4]

- ***Spätná väzba***

Veľkým pomocníkom už pri samotnom vývoji dokážu byť podnety od testerov. Pomáhajú odhaliť nedostatky skôr než sa aplikácia dostane do produkčného stavu. V produkcií sa ako testeri môžu tváriť užívatelia, kedy poskytujú ich názor na stávajúce funkcie a je to jedinečná príležitosť kedy sa vďaka samotnej cieľovej skupine aplikácia posúva vpred.

- ***Aktuálnosť***

Či už obsahová, bezpečnostná alebo vizuálna aktuálnosť, každý aspekt hrá rolu z dlhodobého hľadiska umiestnenia na trhu. Pravidelnými aktualizáciami sa riešia nahlásené problémy, nové grafické návrhy, technické vylepšenia a mnoho užívateľov potom naberá domnienku aktívneho vývoja a je menšia šanca, že by prechádzali ku konkurencii. [5]

1.3 Návrh

Účelom dizajnu je priniesť užívateľom príjemné prostredie, v ktorom sa budú ľahko orientovať, aby sa v nich zbudil záujem používať aplikáciu. Nesmie sa zabúdať na zásadu kedy musí aplikácia zostať intuitívna nezávisle na pestrosti grafického návrhu. [1]
Sekcia sa zaoberá len predbežným grafickým spracovaním, pre finálnu vizuálnu identitu budú využité služby študenta grafického dizajnu.

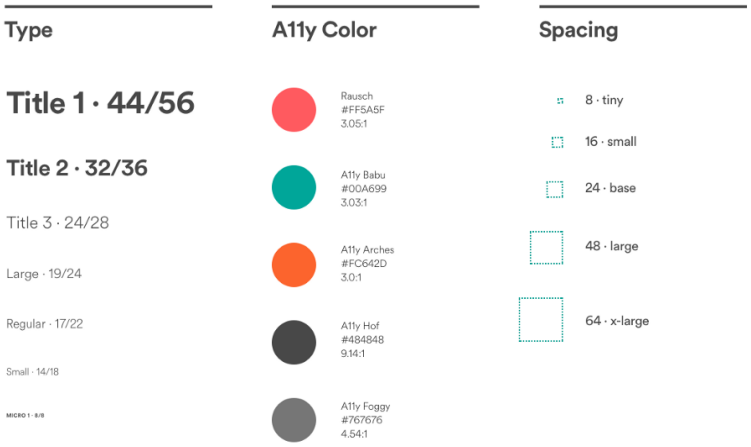
1.3.1 Wireframe

Ilustráciou zobrazené jednotlivé obrazovky a prechody medzi nimi. Takéto predbežné skice sa odohrávajú v skorých začiatkoch ešte pred implementáciou, časovo sa zefektívňuje vývoj. Úlohou nie je ukázať ako bude aplikácia vyzerať, ale to ako bude fungovať. Do jeho obsahu sa teda zahŕňajú len všeobecne orientované prvky a interakcie, nie špecifické elementy ako farby, fonty a obrázky. Aby sa dosiahlo čo najlepšieho užívateľského zážitku často sa wireframing robí v cykloch a robia sa v ňom dodatočné úpravy na základe odborných rád a odozvy od zákazníkov. [11]

1.3.2 Sprievodca štýlov

Sprievodca štýlov z angličtiny style guides sú dizajnové štandardy, ktoré pomáhajú dodržať konzistentnosť pre vyvíjaný produkt, bývajú určené vlastníckou spoločnosťou. Zahŕňajú napríklad farebné schémy, pravidlá fontov a podobné firmu reprezentujúce vizuálne prvky. [1]

Tvorbou nového sprievodcu štýlov, alebo vizuálnej reči sa zaoberal aj súčasný gigant v sprostredkovaní ubytovania na krátkodobé pobyty Airbnb. Ako základné dizajnové princípy si určili: jednotnosť, univerzálnosť, ukážkovosť a jednoduchosť smerom ku koncovým zákazníkom. [12]



Obrázok 1 – Airbnb style guide

1.3.3 Mockupy a prototypy

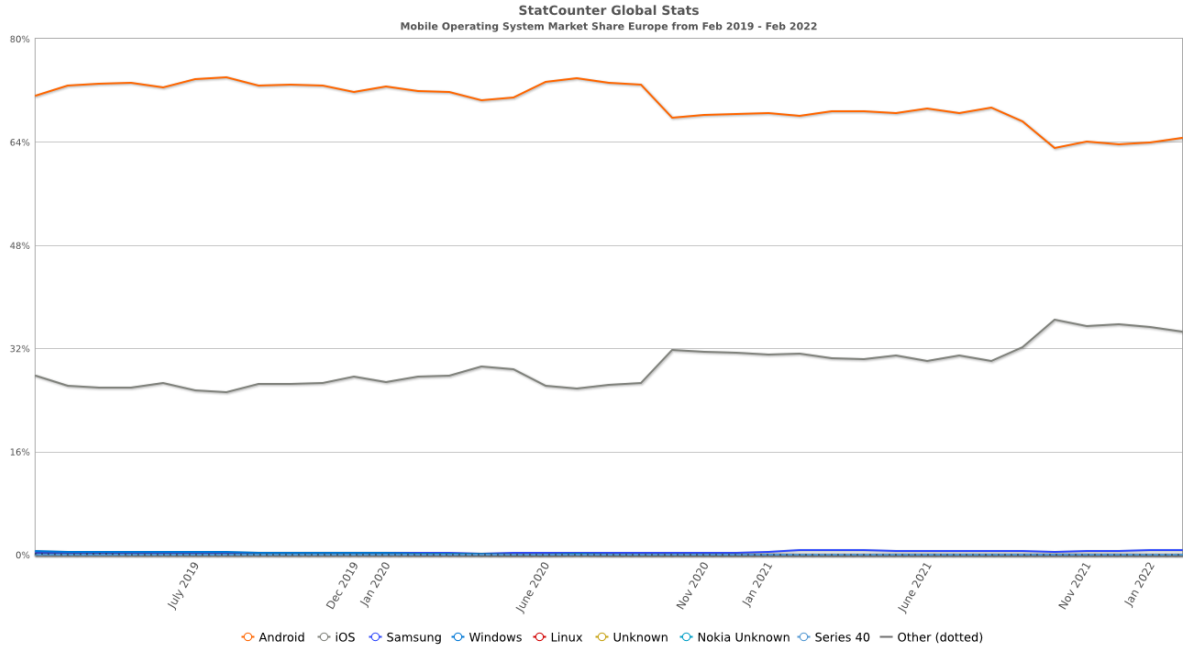
Mockupy sú vyobrazenia vizuálneho dizajnu, vytvárané aplikovaním sprievodcu štýlov na wireframy. Často sa menia v závislosti na pridávaných funkcionalitách a zmenách v celkovej architektúre. [1]

Zo statických návrhov k dynamickým, cez ktoré sa dá jednoduchým spôsobom preklikať sa používajú prototypovacie nástroje ako Figma alebo Invision. Užitočné sú najmä pri simulovaní užívateľa a jeho zážitku z aplikácie. [1]

2 PREHĽAD VÝVOJA MOBILNÝCH APLIKÁCIÍ

V tejto sekcií je opísaný natívny vývoj aplikácií pre oba aktuálne najrozšírenejšie mobilné operačné systémy. Sú tu spracované hlavné rozdiely a porovnanie z rôznych pohľadov medzi natívnym a multiplatformovým vývojom. Na základe zozbieraných dát je popísaná osobná preferencia vývoja pre systém Android.

Európskemu trhu s operačnými systémami pre mobilné telefóny za posledné tri roky dominuje Android s aktuálnym (ku dňu 13.2.2022) podielom 64,65% a iOS s 34,53% ako jediné dva konkurenčné systémy.



Obrázok 2 – Podiel trhu s operačnými systémami v Európe za obdobie troch rokov

2.1 Špecifiká natívneho vývoja

Natívne aplikácie sú vyvíjané pre jeden predom určený operačný systém, za poslednú dekádu sa jedná hlavne o operačné systémy Android alebo iOS. [13]

Sekcia sa zaoberá súhrnom kľúčových odvetví, v ktorých sú porovnávané jednotlivé vývojové prístupy.

- ***Rýchlosť***

Natívne aplikácie bežia priamo na operačnom systéme bez nutnosti využívania ďalšej medzivrstvy, ktorá by spomaľovala výpočty. Priamo používajú ich vlastné programovacie

jazyky a API služby. Výsledkom je optimalizované vykonávanie kódu a komunikácia cez API pre dosiahnutie čo najlepšej rýchlosti a efektivity. [14]

- ***Bezpečnosť***

Prostredníctvom natívnych funkcií zariadení je možné do aplikácie implementovať biometrickú alebo dvoj-faktorovú autentifikáciu pre užívateľa, ktoré sú najefektívnejším zabezpečením autorizačného overenia. [14]
Ďalšou výhodou natívnych aplikácií je podpora SLL certificate pinning, ktorá je efektívnejšou prevenciou útoku typu man-in-the-middle než SSL certifikáty používané webovými prehliadačmi a hybridnými aplikáciami. [14]

- ***Užívateľský zážitok***

Priniesť najlepší možný užívateľský zážitok by malo byť prioritou už pri samotnom navrhovaní, avšak aj prístup k vývoju má na tento dôležitý faktor vplyv.
Na každom detaile záleží, estetika natívnych aplikácií je užívateľovi daného zariadenia (operačného systému) intuitívnejšia na používanie. Výsledkom je rýchlejšie oboznámenie a osvojenie si prostredia aplikácie.
Taktiež existuje niekoľko rozdielov v integrácií určitých funkcií, vďaka ktorým sa iOS alebo Android od seba odlišujú. Aj napriek nepatrnosti týchto rozdielov výrazne pomáhajú k dosiahnutiu lepšieho zážitku z používania. [14]

- ***Konektivita***

Operačný systém umožňuje natívnym aplikáciám priamy prístup ku všetkým hardvérovým prvkom akými sú kamera, senzory, mikrofón, batéria a iné. Sprístupňujú tým veľa pridaných funkcií, z ktorých môže aplikácia získavať dáta a sprostredkovať ich užívateľovi.
Sú situácie kedy nie je internetové pripojenie k dispozícií, napríklad operátorom nepokryté územie, v takomto prípade je treba myslieť na to, že aplikácia závislá na prístupe k internetu môže byť nepoužiteľná ak ho stratí. Offline synchronizácia je riešením u natívnych aplikácií, ide o funkciu kde sa ukladajú všetky uskutočnené zmeny do lokálnej databáze zariadenia. V momente, keď sa pripojenie k internetu obnoví, aplikácia začne synchronizáciu lokálnych dát s tými na serveri. [14]

- ***Čas a cena***

Ak je obchodným plánom dané, že produkt má mať pokrytie na oboch platformách, musí sa dbať na cenové stropy a čas potrebný na doručenie pre oba operačné systémy. Nevýhodou natívneho vývoja je, že väčšina kódu nie je znovu použiteľná z Androidu na iOS alebo opačne, na rozdiel od hybridných aplikácií zdieľajúcich backend alebo cross-platform aplikácií so znovu použiteľnými základnými prvkami. [15]

2.2 Android

Open-source operačný systém spravovaný Googlom, ktorého cieľ je celú dobu rovnaký, a to vytvoriť operačný systém pre čo najviac zariadení. V úplných začiatkoch v roku 2003 mali byť týmito zariadeniami len fotoaparáty. Stal sa z neho operačný systém s oveľa širším spektrom použiteľnosti v rôznych typoch zariadení, primárne na podporu dotykových zariadení, najrozšírenejší v mobilných telefónoch a tabletoch, ale nájde uplatnenie aj v moderných domácich spotrebičoch. [33]
Java bola do roku 2019 hlavným jazykom pre tvorbu Android aplikácií, to sa však zmenilo s nástupom jazyku Kotlin, moderný programovací jazyk kooperujúci s Javou a aktuálne uprednostňovaný jazyk pre Android vývoj. [16]

2.2.1 Architektúra

Celý základ je postavený na jadre Linuxu aj z toho dôvodu je Android open-source. Týmto prístupom si Android berie kľúčové funkcionality a uľahčuje výrobcom zariadení vývoj hardvéru, pre už celosvetovo známe jadro. Úlohou jadra je pristupovať k jednotlivým hardvérovým prvkom zariadenia.
O poskytovanie rozhraní pre Java API framework a odhaľovaní hardvérových možností zariadenia sa stará Hardware Abstraction Layer (HAL). Každý z jeho modulov implementuje rozhranie pre určitý hardvérový komponent ako kameru, wifi alebo bluetooth. Každá bežiaca aplikácia vo verziách Androidu 5.0 a vyššie existuje vo vlastnom procese a vlastnej inštancií Android Runtime (ART). ART spracováva špeciálne, pre Android navrhnuté Dalvik Executable format (DEX) súbory v bytecode formáte pri tom ako spúšťa viaceré virtuálne stroje na zariadeniach s malou pamäťou. Kľúčovými vlastnosťami ART sú ahead-of-time (AOT) a just-in-time (JIT) kompilácia, opmimalizovaný garbage collector, konverzia DEX súborov do viac kompaktného strojového kódu od Android verzie 9.0

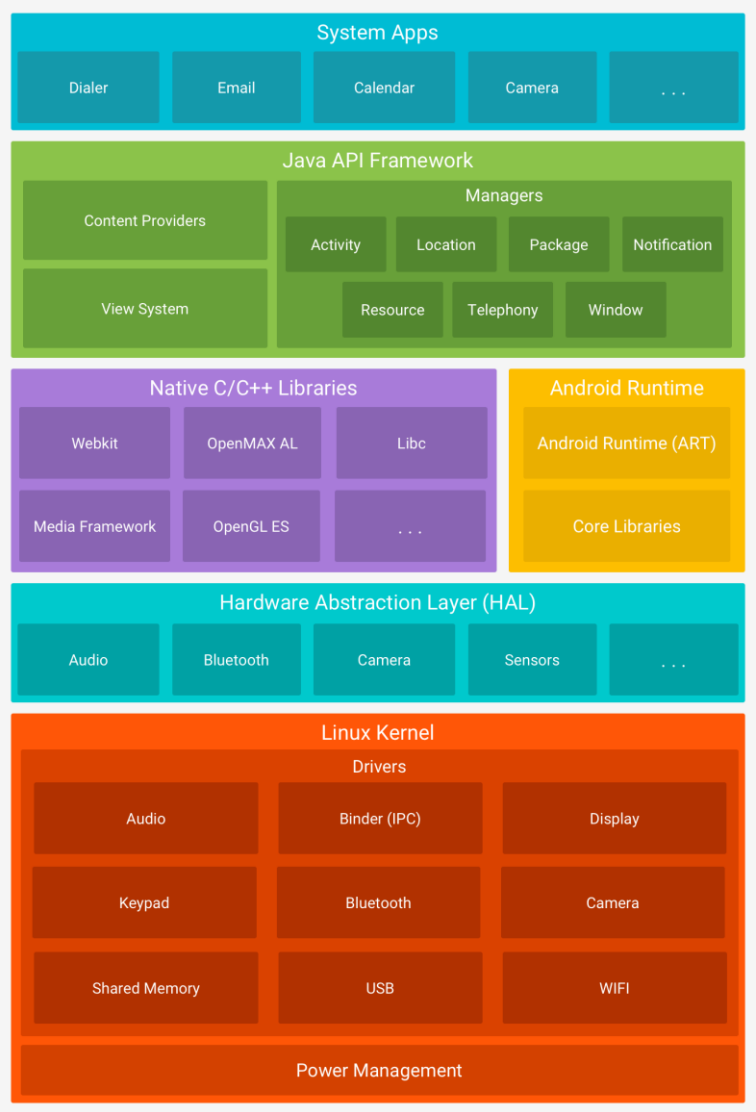
a z pohľadu vývoja lepšia podpora pri debuggovaní zahrnutím detailnejšej diagnostiky výnimiek.

Mnoho knižníc z jadra Androidu je písaných v jazyku C a C++, k niektorým z týchto knižníc je možné pristupovať priamo v kóde ak je aplikácia rovnako vyvíjaná v C alebo C++.

Všetky možnosti operačného systému Android sú dostupné cez aplikačné rozhrania napísané v jazyku Java. Vývojári majú plný a rovnaký prístup k Java API frameworku ako systémové aplikácie. Znovu použitím funkcií jadra a modulárneho systému komponentov je výrazným zjednodušením celkovej práce pri tvorbe aplikácie.

Android obsahuje už vstavané aplikácie pre základné služby ako email, SMS, prehliadač, kontakty, nastavenia a mnoho iných. Tieto aplikácie sú vstupným bodom pre užívateľa, vďaka ktorým je možné inštalovať aplikácie tretích strán.

[35]



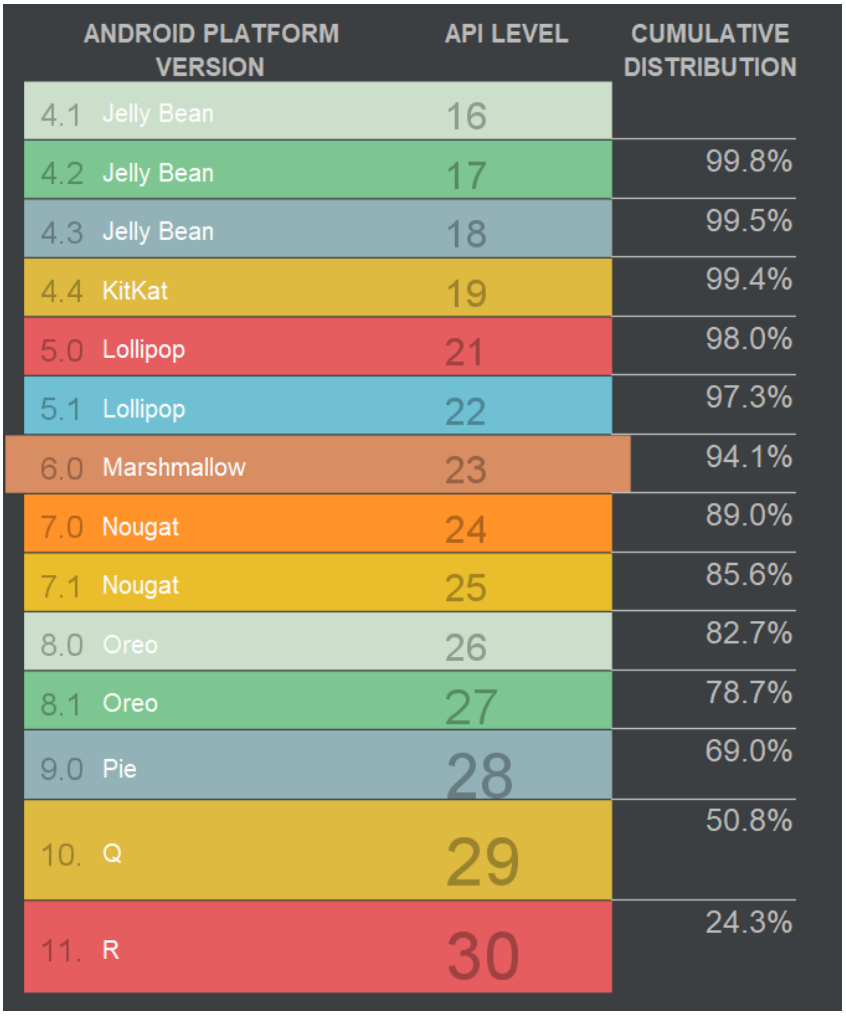
Obrázok 3 – Štruktúra softvéru celého Android systému

2.2.2 Verzie

Z historického hľadiska nie je Google autorom Androidu, ale v roku 2003 ho odkúpil a vznikla spoločnosť Android Inc. Prvý chytrý telefón prišiel na trh v októbri 2008, jednalo sa o HTC Dream (G1) s úplne prvou alpha verziou. Po roku dostal telefón update na Android 1.1 (beta verzia) a neskôr toho roku na produkčnú masovo vydanú verziu 1.5 s názvom Cupcake.

Počiatočné názvy podľa sladkostí, kde každá verzia niesla meno inej cukrovinky a išli za sebou v abecednom poradí začínajúcom písmenom C, pri predstavení verzie Cupcake v roku 2009. Google opustil túto dlho zaužívanú tradíciu verzie predstavením verzie 10.

Keď v apríli 2009 vyšla prvá verzia 1.5, nedisponoval zrovna ohurujúcimi funkciami, dokázal prehrávať a nahrávať videá vo formátoch MPEG-4 a 3GP, sledovanie YouTube, podpora virtuálnych klávesníc tretích strán, kopírovanie a vkladanie textu, jednoduché animácie medzi obrazovkami, na ktorých mohli byť umiesťované widgety. Výraznejší pokrok prišiel s verziami 2.0 až 2.3, tie sprístupnili funkcie ako Bluetootth 2.1, webový prehliadač s optimalizáciou pre JavasScript, Wi-Fi hotspot, podpora viacerých fotoaparátov, podpora Near Field Communication (NFC). Už v štvrtých verziách začala podpora multitaskingu, teda prepínania sa medzi spustenými aplikáciami. Pomerne veľká zmena prišla s nasadením verzie 5.0 Lolipop, v ktorej bol nahradený virtuálny stroj Dalvik, novým enginom ART. Malo to pozitívny vplyv na výdrž batérie a zároveň rýchlejšie spúšťanie aplikácií. S piatou verziou sa objavil nový design nahrádzajúci fyzické navigačné tlačidlá v mobilných zariadeniach, systémovou navigačnou lištou umiestnenou na dolnej strane. Vývojový smer Androidu je zrejmý, a to každý rok prichádzať s podporou nových hardvérových prvkov, prispôsobovať sa trendom v oblasti designu, optimalizovať výkon a spotrebovanú energiu. V auguste 2022 by mala byť oznámená nasledujúca verzia Android 13. [36]



Obrázok 4 – Aktuálna (11.5.2022) distribúcia Android platformy medzi jednotlivými API verziami

2.2.3 Základné bloky aplikácie

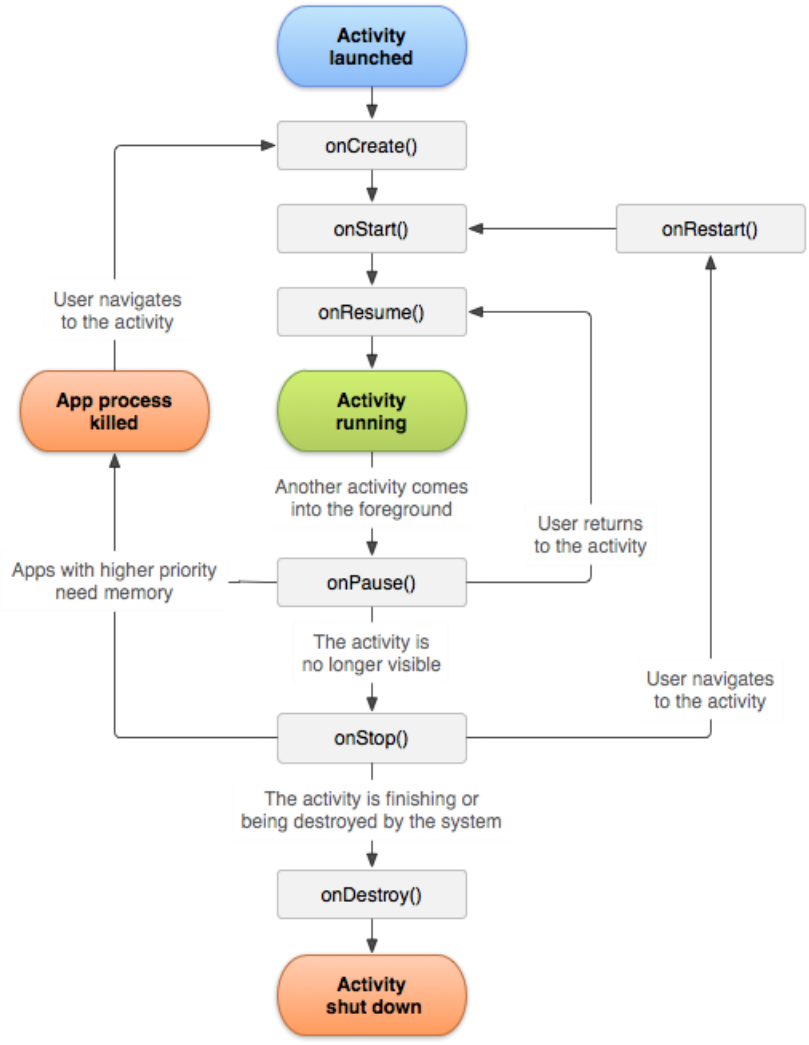
Stavebnými blokmi aplikácie pre Android sú v podstate len štyri piliere.

Aktivita (Activity) – jedna alebo viac tried medzi sebou si odovzdávajúcich údaje, hlavná aktivita je zobrazená užívateľovi ihneď po spustení aplikácie. V aktivite je možné implementovať úlohy rôznej komplexnosti, ktoré potom užívateľ svojimi vstupmi realizuje, jedná sa napríklad o prehliadanie položiek v zozname, vypĺňanie formuláru a iné interakcie. Jej návrh by mal podľa odporúčaní byť taký, aby umožňovala užívateľovi sa zamerať na jednu vec a aby ho zbytočne nerozptyľovala nesúvisiacimi prvkami.

Aktivita má tzv. životný cyklus, ten reaguje na zmeny v predefinovaných situáciách už zostavenými metódami, ktorých názvy evokujú a viacmenej jasne popisujú, kedy sú volané:

- onCreate() – aktivuje sa po spustení aktivity, obvykle sa v nej rieši inicializácia view prvkov, nastavenie premenných a celkového zavedeniu rozloženia elementov na obrazovke.

- onStart() a onResume() – aktivita prechádza do popredia, sú realizované činnosti potrebné k zobrazeniu uživateľského rozhrania danej aktivity. onResume() sa volá, keď aktivita prechádza z pozadia do popredia a užívateľ interaguje s danou aktivitou na rozdiel od onStart() zobrazujúcou prostredie.
- onPause() – keď je spustená nová aktivita, prechádza aktuálna aktivita do pozadia a nová prichádza do popredia. Operácie v tejto metóde by mali byť vykonávané len na atomickej úrovni, aby neboli časovo náročné.
- onStop() – volaná z dôvodu zastavenia aktivity, táto aktivita sa stále nachádza v BackStacku a je možné ju znova zavolať do popredia.
- onDestroy() – ukončuje životnosť aktivity, či už chcene alebo nechcene, pri nedostatku zdrojov môže byť totiž zavolaná operačným systémom.



Obrázok 5 – Schéma životného cyklu aktivity

Služby (Services) – operácie bežiace na pozadí bez potreby užívateľského prostredia. Umožňujú asynchrónne operácie s dlhšou dobou trvania. Prehrávanie hudby na pozadí, zatiaľ čo je užívateľ prepnutý do inej aplikácie, alebo má uzamknutý telefón je typickým príkladom.

Broadcast receivers – bežia na pozadí a neustálym počúvaním reagujú na udalosti zariadenia, keď takáto udalosť nastane, vysiela ju v podobe upozornenia užívateľovi. V momente, keď užívateľ obdrží správu SMS, telefón pomocou prijímaču príjme tento zámer (intent) a spustí priradenú službu. Prednastavená aplikácia potom zobrazí notifikáciu.

Poskytovatelia obsahu (Content providers) – sprostredkúvajú údaje medzi aplikáciami a procesmi. K týmto údajom potom môžu jednotlivé aplikácie pristupovať a zdieľať ich medzi sebou. [36]

2.3 iOS

Mobilné zariadenia od spoločnosti Apple sú vybavené operačným systémom iOS. Oproti Androidu, iOS je uzavretý a má štandardizované UI prvky s menšou možnosťou úprav, ale návrhové princípy Apple zaručujú estetické vizuálne prostredie. Programovací jazyk Swift je používaný pri vytváraní iOS aplikácií. [16]

2.4 Porovnanie natívnych platforiem

Prieskum vybraných oblastí a poukázanie na rozdiely medzi dvoma najrozšírenejšími platformami v odvetví mobilného vývoja.

2.4.1 Pracovné príležitosti

Na Českom trhu je počet pracovných príležitostí je približne taký, aký je stav trhu s týmito operačnými systémami. K mesiacu február 2022 je stav Android: 73,57% a iOS: 25,84%.
Mobile Operating System Market Share Czech Republic | Statcounter Global Stats
Pri hľadaní kľúčových slov „iOS vývojář“ na portáli Jobs.cz pre všetky kraje v Českej Republike, bolo k 20.3.2022 nájdených 22 ponúk oproti kľúčovým slovám „Android vývojář“, kde bolo nájdených 33 pracovných ponúk.

2.4.2 Rozpočet

Z hľadiska času a s ním spojených nákladov potrebných na vývoj aplikácie vyhráva iOS. Tým, že je Swift priamo prispôsobený pre iOS, tak dokáže vývojárom značne zjednodušiť prácu a ušetriť hodiny drahého času. Štruktúra iOSu je striktne uzavretá v Apple ekosystéme

a viacej tlačí na štandardizáciu určitých vývojových procesov, aby zaistila vhodnú a fungujúcu implementáciu funkcií aplikácie. Vďaka všetkým poskytnutým informáciám sa redukuje počet chýb a znižuje tak ďalšie náklady na ich odstránenie. [17]

2.4.3 Publikovanie

Viac prístupnou platformou na publikovanie aplikácií je určite Android, to je spôsobené najmä benevolentnejším akceptovaním aplikácií na Google Play v porovnaní s App Store. Ak sa dodržujú všetky politické ustanovenia spoločnosti, schválenie trvá maximálne sedem dní pre nových vývojárov, ktorí musia zaplatiť registračný poplatok 25 amerických dolárov. App Store je na druhú stranu obozretnejší a aplikácie prechádzajú zložitejším avšak aktuálne kratším schvaľovacím procesom, pri ktorom je avšak zamietnutie oveľa pravdepodobnejšie. Podobne ako Google aj Apple spoplatňuje publikáciu aplikácií na App Store, ich politika je odlišná v tom, že poplatok 99 amerických dolárov si vývojár musí platiť ročne. [17]

2.4.4 Výnosy

Napriek trhovej dominancií Androidu, iOS používatelia sú zvyknutí platiť za aplikácie a mikrotransakcie v nich a z dlhodobejšieho hľadiska sa iOS aplikácie stávajú lepším zdrojom príjmov.

2.4.5 Programovacie jazyky

Stavebným blokom vývoja aplikácie pri oboch operačných systémoch je programovací jazyk. Obe platformy majú na svojom historickom konte dva hlavné programovacie jazyky. V súčasnosti je oficiálne Androidom podporovaný jazyk Kotlin, pričom stále podporuje Javu. iOS umožňuje tvorbu aplikácií už len cez Swift a kompletne sa tak odpútal od Objective-C.

- Objective-C

  Vyvinutý v 80tych rokoch 20. storočia za účelom pozdvihnúť jazyk C na objektovo orientovaný. Apple používal tento jazyk ako ich hlavný programovací jazyk pre operačné systémy OS X a iOS až do roku 2014, kedy bol nahradený jazykom Swift. [19]

- Java

  Jeden z najznámejších objektovo orientovaných jazykov vôbec, s pôvodným názvom OAK určeným pre set-top boxy. Neskôr, v roku 1995 spoločnosťou Sun

Microsystems upravená a premenovaná Java pripravená vrhnúť sa do oblasti webového vývoja. Od roku 2009 kedy bola kúpená technologickým gigantom Oracle, je ním doteraz spravovaná a aktualizovaná, aby držala krok s aktuálnymi technológiami.

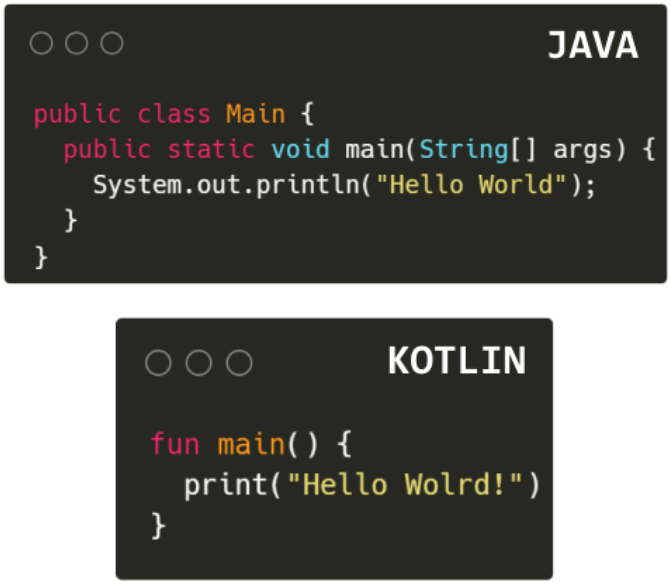
Ak počítač, na ktorom je kód spúšťaný obsahuje Java Runtime Environment (JRE) obsahujúci knižnice rôznych tried, je potom možné spustiť program. Na rozdiel od jazykov ako C alebo C++ , Java ako interpretovaný jazyk pri kompilácií prekladá zdrojový kód do byte kódu, robí to pomocou Java Virtual Machine (JVM), kedy je zdrojový kód programu prekonvertovaný do strojového kódu platformy, na ktorej zrovna beží. [20] Medzi ďalšie možnosti Javy patrí Garbage Collector na automatickú správu pamäte, to znamená, že vývojári sa nemusia manuálne starať o alokáciu a následnú dealokáciu použitých prostriedkov v kóde.

Tým, že môže byť spúšťaná na širokej škále platforiem, tak je Java používaná pre tvorbu backendu webových aplikácií, embedded systémov, systémov pre firmy, Android aplikácií, vyvíjané týmto jazykom od samotného začiatku Androidu a iných Javu podporujúcich systémov. [20]

- Kotlin

  Na prvý pohľad zdieľa s Javou veľa podobností ako staticky typovaný, multiplatformový jazyk taktiež bežiaci na JVM. V čom je ale zásadne lepší a odlišný od Javy? Jedná sa hlavne o jednoduchosť a podobnosť s inými modernými jazykmi.

  Nástupca Javy, aj takto sa prezýva Kotlinu od čias kedy bol v roku 2011 predstavený spoločnosťou sídliacou v Prahe ako lepšia alternatíva k Jave. Prvá stabilná verzia prišla na trh až v roku 2016. Google v roku 2019 oznámil oficiálnu podporu pre Kotlin ako nový hlavný jazyk určený pre vývoj. Stalo sa tak vďaka tomu, že neobsahuje také robustné množstvo kódu ako Java, jeho architektúra je viacej zameraná na škálovateľnosť a zníženie počtu chýb vďaka implementácií null safety – bez toho, aby bolo Kotlinu v kóde určené, že sa jedná o premennú s možnou hodnotou null, Kotlin neobsahuje null type.



Obrázok 6 – Java a Kotlin, porovnanie kódu výpisu do konzole

Obsahuje základné dátové typy ako integer, float, boolean, character, string. Premenné rozdeľuje do kategórií premenlivých (mutable) – je možné čítať, zapisovať a po zápise meniť hodnotu, a nemenných (immutable), u ktorých je možné zapísať hodnotu raz a potom ju len čítať. Ich deklarácia sa mení len v prvom slove, pri premenlivých je to „var“ a pri nemenných „val“, za nimi nasleduje názov premennej a priradenie určitej hodnoty. [40]

NullPointerException (NPE) je v mnohých jazykoch, vrátane Javy, bežnou výnimkou, ktorá vzniká keď sa pristupuje k premennej, ktorá obsahuje hodnotu null. Táto výnimka môže eskalovať do vážnych, ťažko dohľadateľných problémov. Tomuto problému sa v Kotline dá ľahko vyhnúť pretože väčšina výnimiek, kedy sa pristupuje k premenným s hodnotou null sú zachytené už pri kompilácií a vývojové štúdio na ne priebežne upozorňuje. Existuje však pár vzácnych výnimiek, kedy dôjde k NPE až po zostavení aplikácie, tie ale nastávajú len za vedomého označenia tzv. not-null assertion operátorom (!!) v kóde, alebo v iných, zriedkavých stavoch. [40]

Nastávajú prípady, kedy sa nevie aký typ bude naberať nenulová premenná a tá bude známa až pri vykonávaní kódu, riešením je deklarácia premennej použitím neskorá inicializácia (lateinit) modifikátorom. [40]

Zdieľa mnohé podobnosti s ostatnými jazykmi, ale napríklad tzv. sprievodné objekty (companion objects) sú menej známim konceptom v triedach, kedy tento objekt

obsahuje metódy a premenné, ktoré sú spoločné medzi všetkými vytvorenými inštanciami triedy. [40]

- Swift

  Podobne ako Kotlin, jedná sa o open-source jazyk. Apple vytvoril a v roku 2014 predstavil ako oficiálny jazyk pre vývoj aplikácií na platforme iOS. Swift má oproti Objective-C radu výhod. Používa generiku a funkcie vyššieho rádu (funkcia, ktorá dokáže zobrať funkciu ako argument), čím dosahuje lepšej čitateľnosti a znovu použiteľnosti kódu. Zlepšila sa bezpečnosť kódu, lepšia efektivita so správou pamäte a dokonca aj rýchlosť. [19]

2.5 Multiplatformový vývoj

Dnes jedna z najobľúbenejších foriem vývoja aplikácií, hlavne vďaka rýchlosti vývoja, jednoduchosti kódu a možnosť publikovať produkt na viaceré platformy, čo garantuje vyšší počet potencionálnych užívateľov a teda aj šancu vygenerovať z aplikácie väčšie zisky v kratšom čase.
Medzi najznámejšie a aktuálne najpoužívanejšie frameworky pre vývoj aplikácií na obe platformy – Android a iOS patria Flutter a React Native.

2.5.1 Flutter

Google vytvoril Flutter ako multiplatformový open-source SDK pre vývoj webových, desktopových a mobilných aplikácií. Od frameworku a knižníc sa líši tým, že má vlastné nástroje, knižnice, rozhrania pre programovanie aplikácií (API) a ostatné k vývoju potrebné funkcie, stále je však všeobecne označovaný ako framework. Má vlastný engine integrujúci základné API Flutteru. Pre vývoj sa používa objektovo orientovaný jazyk Dart, ktorý je taktiež od Googlu. Každý element užívateľského rozhrania je postavený na koncepte widgetu, sú to predpripravené stavebné bloky s možnosťou úpravy alebo tvorby vlastných. Kvôli tomu, že Dart kompiluje kód priamo do zariadenia, patrí medzi najrýchlejší framework.

2.5.2 React Native

Open-source framework od firmy Facebook vydaný v roku 2015, stavia na ich webovom frameworku React. Ako programovací jazyk používa z webového prostredia dobre známy Javascript, ktorý sa za posledné roky stal najpopulárnejším jazykom vo svete vývoja

webových aplikácií. Stal sa populárnym hlavne z dôvodu, že samotný React je široko používaný spoločnosťami a dobre ho pozná takmer každý webový vývojár. Medzi najznámejšie aplikácie postavené na tomto frameworku patria Facebook, Instagram, Shopify, Pinterest a Discord.

Rovnako ako ostatné multiplatformové frameworky, ani React Native nepodporuje všetky možnosti (zväčša senzory) danej platformy, moduly buď chýbajú alebo sú len čiastočne vyvinuté. Animácie a gestá sú zložité reprodukovať pre obe platformy za použitia rovnakých funkcií a preto nie je React Native vhodný na stavanie komplexného užívateľského prostredia. [23]

3 NÁSTROJE PRE TVORBU APLIKÁCIE

Na základe vytvoreného návrhu sa táto časť venuje prieskumu a výberu samotných nástrojov potrebných na tvorbu.

3.1 Vývojové prostredie

Jedná sa o program obsahujúci radu nástrojov pre vývoj softvéru, často označovaný anglickou skratkou IDE.

Vývojári v ňom väčšinou budujú softvér, pretože im uľahčuje ich prácu poskytnutím rôznych vstavaných nástrojov a možností pre efektívnejšiu prácu pri komplexnom vývoji aplikácie.

Základné nástroje ako editor, kompilátor a debugger by mali byť súčasťou každého IDE, niektoré z nich potom disponujú podporou verzovacieho systému s grafickým rozhraním alebo vizualizáciou údajov, ktoré sú považované za pokročilejšie funkcie. Vstavaný editor pomáha s organizáciou kódu a robí ho tak prehľadnejší zvýrazňovaním syntaxe daného jazyka. Vývoj zrýchľuje funkcia automatického dopĺňania kódu a používanie už vytvorených šablón s možnosťou ich úpravy použitím klávesových skratiek a kľúčových slov. Sledovanie už napísaného kódu a jeho analýza v reálnom čase patria tiež medzi funkcie, nielen skracujúce čas potrebný na písanie, ale dokonca vďaka nim robia vývojári menej preklepov, ktoré by nemuseli hneď zachytiť a po kompilácií tráviť čas na analyzovaní potencionálne nesprávne fungujúceho kódu. Veľa z prostredí umožňuje užívateľom pridávať ďalšie doplnky na zvýšenie produktivity. Tie bývajú vytvorené spoločnosťou vlastniacou vývojové prostredie alebo samotnými užívateľmi.

Trh s IDE je pomerne široký a z hľadiska podpory jazyka sú to prostredia podporujúce viacero programovacích jazykov alebo prostredia s podporou len jedného jazyka. Preto je dôležité sa rozhodnúť, aký jazyk bude pre vývoj aplikácie použitý.

Najznámejšími vývojovými prostrediami sú Eclipse – určené hlavne na vývoj Java aplikácií, Visual Studio od Microsoftu – primárne pre C#, IntelliJ IDEA – produkt od firmy JetBrains so špecifikáciou pre Javu. Pre natívny vývoj mobilných aplikácií existujú dve oficiálne vývojové prostredia Android Studio a Xcode, každé určené pre pokrytie ich vlastnej platformy. Takmer všetky vývojové prostredia sú voľne dostupne, avšak niekedy ak je nutné získať extra funkcie, je potrebné zaplatiť za tzv. pro verziu, ktorá je špeciálne navrhnutá pre profesionálne použitie. [25]

3.1.1 Android Studio

Špeciálne navrhnuté prostredie oficiálne určené pre vývoj Android aplikácií, vychádzajúce zo staršieho IntelliJ IDEA. Síce zdieľa podobne navrhnuté grafické rozloženie ako ostatné produkty spoločnosti JetBrains, ale od bežných IDE sa výrazne odlišuje ponukou vlastných možností. Zanecháva pocit jednoduchého používania a orientovania sa v ňom.
Pri prvom spustení sa zobrazí uvítacia obrazovka s ponukou týkajúcou sa otvorenia predošlého projektu, vytvorenie nového alebo stiahnutie z repozitára. Vytvorením alebo otvorením existujúceho projektu sa zobrazí hlavná obrazovka podobná ostatným IDE.

Komponenty hlavného okna:
Menu – nachádza sa v hornej časti, obsahuje viacero ponúk vykonávajúcich určité akcie
Panel s nástrojmi – často používané akcie ako výber zariadenia, spustenie projektu, pripojenie debuggeru, zapnutie profilovania a iné
Navigačná lišta – prehľadne zobrazuje zanorenie podľa aktuálne otvoreného súboru
Okno editácie – zobrazuje obsah zvoleného súboru, v hornej pravej časti je možnosť prepnúť tento editačný režim do zobrazenia kódu, designu alebo rozdelenia obrazovky, pri čom budú zobrazené oba.
Status bar – zobrazuje notifikácie, informačné správy o projekte a aktuálne vykonávaných procesoch spojených s vývojom aplikácie.
Okno projektového usporiadania – hierarchicky zobrazuje projektovú štruktúru

Užitočné nástroje a ich funkcie odkrývajúce ďalšie okná sú rozmiestnené po celom okne Android Studia. Kliknutím na niektoré z nich zobrazí okno s korešpondujúcimi nástrojmi. Toto okno je následne možné pripnúť k vyhovujúcej strane hlavného okna alebo presun a oddelenie mimo neho. Najčastejšie používanými nástrojmi sú:
Build Variants – možnosť rýchlej zmeny konfigurácie, na ktoré zariadenie (fyzické alebo emulátor) sa má projekt zostaviť
Database Inspector – užitočný hlavne pri debuggovaní databázových dotazov
TODO – položka zobrazujúca všetky komentáre v kóde obsahujúce kľúčové slovo TODO, tento zoznam sa dá ďalej filtrovať podľa zložky alebo súborov
Logcat – poskytuje prístup k monitorovaniu vybraných výstupov z aplikácie
Terminal – otvára okno s klasickým systémovým terminálom v IDE

Build – zobrazuje informácie o build procese počas kompilácie a v prípade chýb zobrazuje detailne popísané záznamy z nich

Run – zobrazí sa pri spustení a behu aplikácie , poskytuje záznamy z behu a umožňuje beh zastaviť alebo reštartovať

Event Log – správy a oznámenia týkajúce sa akcií uskutočnených v Android Studiu. Napríklad oznámenie o úspešnom zostavení projektu.

Gradle – prehľad o Gradle úlohách, ktoré konfigurujú zostavenie celého projektu

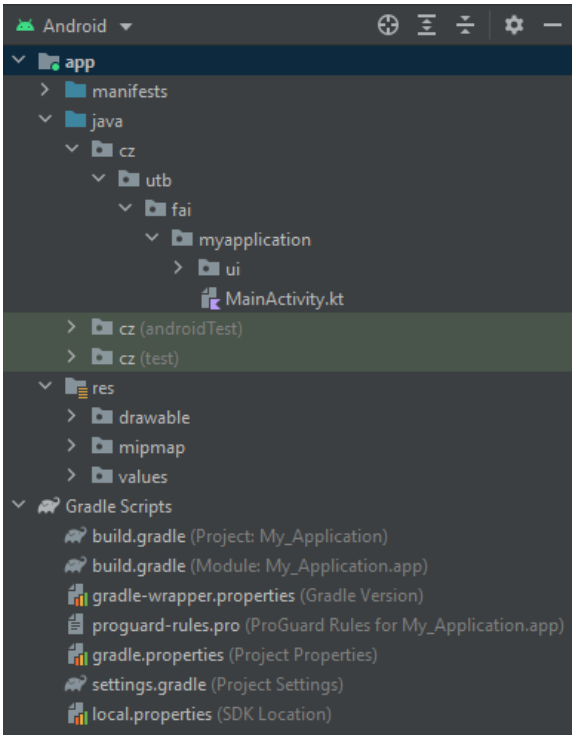
Profiler – nástroj poskytujúci analýzu a monitoring projektu v reálnom čase, hodí sa pri identifikovaní problémov spojených s využitím batérie, pamäte, procesoru a siete

Device File Explorer – okno s prehľadom a priamym prístupom do súborového systému zariadenia, na ktorom beží aplikácia

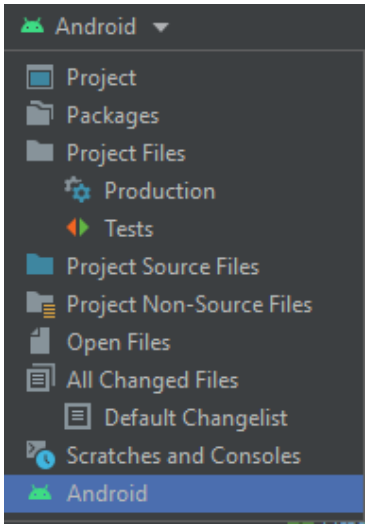
Resource Manager – nástroj na kontrolu zdrojov ako xml súbory, obrázky a iné digitálne aktíva (assets)

Layout Inspector – vizuálna reprezentácia komponentov uživateľského prostredia

Emulator – v prípade aktivácie Android Virtual Device (AVD) obsahuje zobrazenie tohto emulátora [40]



Obrázok 7 – Projektová štruktúra v Android Studio



Obrázok 8 – Android Studio, možnosti zobrazenia projektovej štruktúry

Dopĺňanie kódu je tu na vysokej úrovni, či už v zmysle automatických importov, chytrého dopĺňania s rôznymi variáciami pomocou klávesových skratiek alebo vytváranie skratiek obsahujúcich vlastné časti kódu.

Git, Github, CVS, Mercurial, Subversion a Google Cloud sú všetky podporované systémy na správu verzií, nie je tak nutné prechádzať z Android Studia do terminálu alebo iného externého programu.

Gradle je použitý ako systém k zostaveniu programu, obohatený o Android plugin for Gradle s viacerými potrebnými špecifickými schopnosťami pre Android. Nastavenia pre Gradle je možné ľubovoľne upravovať v súboroch pomenovaných *build.gradle*. V jeho dvoch konfiguračných súboroch projektu sa nachádzajú menovité závislosti balíkov, ktoré je možné využívať v projekte.

Debugger zobrazuje klasické veci ako aktuálne hodnoty premenných, návratové hodnoty metód, lambda výrazy a iné, z iných prostredí známe veci. Meraním výkonu aplikácie spustenej v emulátore alebo na fyzickom zariadení je možné sprostredkovať údaje o využití procesoru a pamäte zariadenia vizualizovanými dynamickými grafmi napomáha k odhaleniu únikov pamäte, optimalizovaniu grafického výkonu a analýze požiadavkou na sieť. Podrobné logy zo zariadenia, na ktorom je spustená aplikácia sú zobrazené v okne Logcat. [26]

Xcode

Vývojové prostredie od Apple určené pre platformy Apple, konkrétne iOS, macOS, watchOS a tvOS. S aktuálnou verziou 13 sa zlepšilo automatické dopĺňanie kódu, ktoré dokáže lepšie predpovedať, je rýchlejšie a navrhuje najpravdepodobnejšie výrazy. Editor bol obohatený o integráciu skratiek z textového editoru Vim. Pribudla šikovná funkcia do

debuggingu, vloženie break pointu na špecifický stĺpec kódu, ktorá sa nevidí v iných prostrediach tak často, ak vôbec. [27, 29]

3.2 Emulátor a Simulátor

Obe sú virtuálnym prostredím, v ktorom je spúšťaná aplikácia najmä počas vývoja a testovania. Hlavná odlišnosť emulátoru od simulátoru je, že emulátor virtualizuje všetky prvky reálneho fyzického zariadenia, čiže softvér a hardvér. Stále sa však nevyrovná fyzickému zariadeniu a môže dochádzať k miernym odchýlkam, ktoré nie je možné zachytiť vo virtuálnom prostredí. Oficiálne vývojové prostredia pre mobilný vývoj prichádzajú už so vstavanými virtuálnymi zariadeniami dostupnými hneď po inštalácií. V prípade Xcode sa jedná o simulátor a u Android Studia je to emulátor.

Výhodou využitia virtuálneho zariadenia je ušetrenie nákladov, ktoré by bolo nutné vynaložiť za nákup fyzických zariadení pri testovacích fázach, tieto náklady narastajú na veľkosti vývojárskeho tímu. Testovaniu takisto napomáha fakt, že tieto zariadenia nie je nutné pri každom spustení resetovať a nastavovať do určitého stavu, v ktorom je vzhľadom na stav vývoja potrebné dosiahnuť. V automatických testoch sa tým výrazne zlepšuje samotná spoľahlivosť testov. [28]

3.3 Fyzické zariadenie

Jedná sa o mobilný telefón podporujúci platformu, pre ktorú sa aplikácia vyvíja alebo testuje. Či už sa jedná o Apple alebo Android zariadenie, žiadny simulátor sa nedá nahradiť reálnym zariadením, na ktorom sa najlepšie testuje použiteľnosť a výkon aplikácie. Testovanie použiteľnosti je oproti virtuálnym zariadeniam lepšie kvôli reálnemu rozloženiu grafických prvkov a interakcií s klávesnicou prekrývajúcou určitý obsah. Xcode obsahuje len simulátor a tak je ďalšou výhodou overenia funkčnosti aplikácie voči potencionálnemu prepojeniu s fotoaparátom, senzormi a ostatnými hardvérovými prvkami zariadenia. V neposlednom rade vynikajú presnejšími a vernejšími údajmi z testovania výkonu. [28]

3.4 Systém správy verzií

K vysvetleniu, čo to je systém správy verzií by sa mohol zjednodušene prirovnať k záznamníku, pretože vďaka nemu je možné zaznamenávanie zmien spravených v kóde počas vývoja a tým poskytuje vhodný prehľad každej zmeny.

Načo používať verzovací systém, keď sa dá kód ukladať klasickým spôsobom? Pre samotného vývojára by sa to mohlo zdať ako akceptovateľná možnosť, avšak aj v tomto prípade je lepšie si zaznamenávať uskutočnený postup. Používanie systému správy verzií má ďalšiu radu výhod, ako už napovedá názov, je možné sa vracať do histórie a prezerať si každú jednu nahranú zmenu, ktorá sa nazýva commit. Skutočnú pridanú hodnotu dáva až tímom skladajúcich sa z viacerých vývojárov. Každý z nich môže pracovať nezávisle na tom druhom za použitia vetvenia (tzv. branch), to znamená, že vývojári majú vytvorené osobitné priestory označované ako branch, v ktorých môže byť vykonaná akákoľvek zmena. Vykonané zmeny nemajú žiadny vplyv na hlavnú vetvu (master branch) až pokiaľ vývojár nespojí svoju vetvu s vetvou hlavnou.

Bez cloudového poskytovateľa hostingu pre kód ukladaný do repozitárov nemá verzovací systém až taký ďalekosiahly dopad, keďže kód sa ukladá aj mimo lokálny disk a hrozí tak podstatne menšie riziko straty.

3.5 Autentizácia

Proces overenia totožnosti užívateľa pomocou určitých údajov a na základe ich právoplatnosti jeho vpustenie do aplikácie. Zvyčajne sa na overovanie používajú heslá v textovej alebo číselnej podobe, generovaným tokenom alebo použitím vlastných biometrických dát. Ďalšou dnes populárnou voľbou je prihlásenie sa cez už vytvorené účty na iných platformách ako Google, Facebook, Twitter a podobne. Kombinácia autentizačných údajov používaná úplne najčastejšie je meno a heslo. Autentizácia sa rozdeľuje do skupín jedno, dvoj, viacfaktorovej bezheslovej a autentizácií založenej na certifikátoch. [31]

3.5.1 Jedno-faktorová

Jediné čo požaduje je overiť len prihlasovacie údaje z jedného zdroja, aby bola overená identita. Typickým príkladom je vloženie emailu a hesla pri prihlasovaní sa do majority bežných stránok a aplikácií, ktoré si nevyžadujú ochranu na vyššej úrovni. [31]

3.5.2 Dvoj-faktorová a viac-faktorová

Na prihlásenie sa potrebuje užívateľ zadať dva alebo viac odlišných údajov, kedy už mu nestačí len heslo ale potrebuje získať a poskytnúť aj iné autentizačné údaje, aby sa dostal do aplikácie. Bežnými typmi týchto autentizácií sú kombinácie generovaných tokenov cez aplikácie tretích strán (maily, SMS, samotné aplikácie), hardvérových tokenov a biometriky.

Banky a iné spoločnosti vyžadujú dvoj-faktorovú autentizáciu, kvôli dôležitosti citlivých údajov a prístupu k peniazom užívateľa, ktoré by mohli byť jednoducho napadnuteľné a zneužité ak by útočník zistil heslo. [31]

3.5.3 Využívanie jedného účtu tretích strán

Registrovať a prihlásiť sa do väčšieho množstva aplikácií pomocou jedného účtu je dnes bežná prax a užívateľ si tak nemusí pamätať ďalšie heslá. Na autentizáciu využíva poskytovateľov centrálnej identity. Prihlasovanie pomocou Google účtu je bežnou praxou každého Android používateľa. Nevýhodou tohto prístupu môže byť skutočnosť, že v prípade odcudzenia jedného hlavného účtu, vzniká riziko odcudzenia údajov aj vo všetkých ostatných aplikáciách, v ktorých bol tento účet použitý. [31]

3.6 Databáza

Združená kolekcia systematicky uložených dát, ktoré môžu byť pridávané, upravované, mazané a inak manipulované. Bežný užívateľ aplikácie interaguje s databázami denne, už len keď sa jedná o obyčajné nahrávanie fotiek na sociálne siete, kedy nahráva dáta do databáze danej platformy. Dáta sú zaradené do kolekcie záznamov a pripravené na ďalšiu manipuláciu s nimi.

Základnými komponentmi databáze sú:

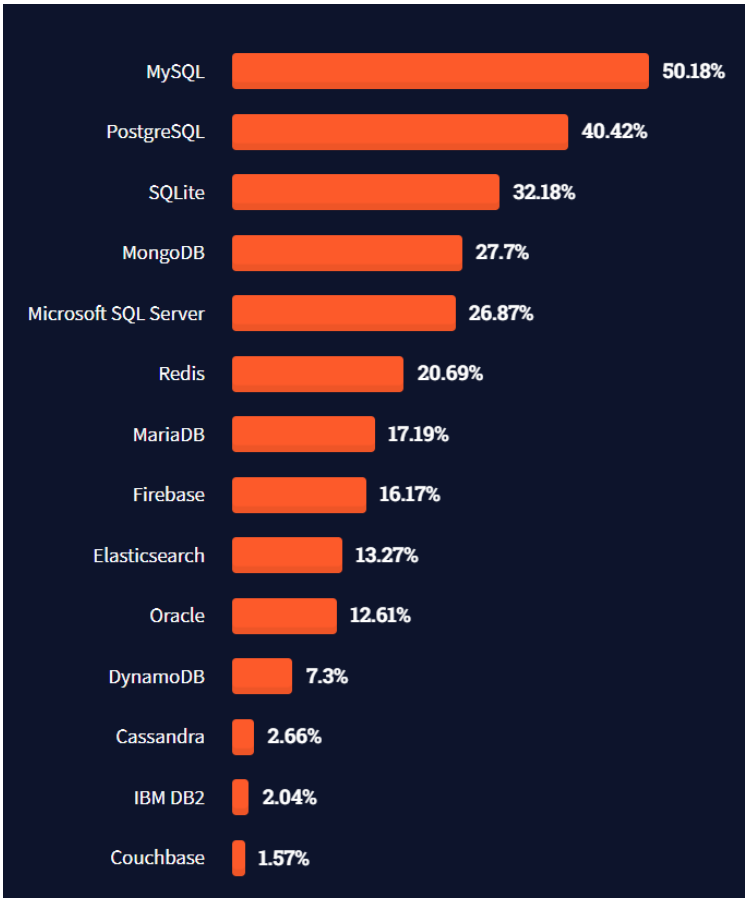
- dáta – typovo sa môže jednať o ľubovoľné entity ako text, čísla, obrázky, videá a iné,
- fyzické zariadenia, na ktorých sú dáta reálne ukladané,
- softvér manažujúci celý proces. Set programov vrátane operačného systému, databázového programu a aplikácií umožňujúcich prístup k samotnej databáze,
- databázový jazyk, označovaný v mnohých prípadoch aj ako dopytovací. Všetka komunikácia na manipuláciu s dátami sa uskutočňuje cez príkazy v databázovom jazyku,
- procedúra je inštrukčná sada inštrukcií a pravidiel napomáhajúcich s používaním systému pre správu databáz

[32]

Typy databáz:

- **Distribuovaná –** spravidla je rozmiestnená na viacerých, navzájom prepojených zariadeniach nachádzajúcich sa na jednom fyzickom mieste. Medzi týmito zariadeniami sú po častiach uložené všetky dáta.
- **Centralizovaná –** všetky údaje sú uložené na jednom centralizovanom mieste, ku ktorému môže byť pristupované z akýchkoľvek externých zdrojov a dostať sa k požadovaným informáciám.
- **Cloudová –** v podstate sa jedná o technológiu správajúcou sa ako centralizovaná databáza na internete. Disponuje možnosťou uloženia obrovského množstva údajov pri nepretržitej dostupnosti z akéhokoľvek miesta.
- **Relačná –** dáta sú ukladané do riadkov a stĺpcov v tabuľkách, za použitia cudzích kľúčov je možné ľahko vytvárať prepojenia medzi inak nezávislými tabuľkami.
- **NoSQL –** databáza bez jasne danej štruktúry, každý nahraný objekt môže mať iný tvar a veľkosť

[34]



Obrázok 9 – Najpoužívanejšie databázové systémy z prieskumu StackOverflow

3.7 Cloudové služby

V dnešnej dobe už len malé množstvo mobilných aplikácií funguje autonómne a nepotrebuje získavať údaje z externých zdrojov, využívať autentifikačné alebo iné serverové funkcie. [36]

Dve najrozšírenejšie cloudové platformy so základnými funkciami autorizácie, úložiskom, databázou, hostingom a analytiky pre mobilné zariadenia sú AWS a Firebase, rozšírené aj medzi webovými aplikáciami. [39]

3.7.1 Firebase

Pre natívne platformy Android a iOS, ale aj pre webové aplikácie poskytuje Software Development Kit (SDK). Čo sa týka ponuky služieb, je medzi nimi množstvo analytických nástrojov pre monitorovanie aplikácie, dve NoSQL databázy (Firestore a Realtime Database), ktoré sú bez nutnosti zložitej konfigurácie a volateľné trigger funkcie.

Bezplatný plán služieb obsahuje autentifikáciu užívateľov cez FirebaseAuth, notifikácie, do určitej miery užívania databázy, obmedzené úložisko a základnú analytika. Problém môže

nastať pri vyššej prevádzke po uvedení na trh a zle navrhnutými databázovými dotazmi nie je vysoký účet v rádoch tisícov dolárov žiadny problém.

Svojím prostredím pôsobí priamym a jednoduchým dojmom a nerobí problém ani front-end vývojárom, ktorí nemajú skúsenosti a vedomosti s konfiguráciou backendu. Všetku komplexitu berie Firebase na seba a je ideálnym riešením pre tvorbu real-time aplikácií. [39]

3.7.2 AWS

Riešením pre vývojárov k prepojeniu Android, iOS a React Native s AWS je nazývané AppSync. Dokáže zaujať nastavením rozličných prostredí – vývoj, testovanie a produkcia, integráciou s GraphQL a Apollo, výberom a plnou mocou nad databázou a množstvom služieb, ktoré svojou komplexnosťou pokrývajú takmer každý požiadavok.
Ceny za ponúkané služby nie sú nastavené príliš vysoko a v určitých prípadoch sú veľmi výhodné. Redukciou o hodnotu presahujúcou 80% dokázali prekonať mnohé iné cloud platformy.
Faktorom signifikantným pre vývoj je čas závislý na nastavovaní a spravovaní aplikácie a tým, že AWS ponúka desaťkrát viac služieb ako Firebase, je aj podstatne zložitejší na pochopenie a vyžaduje určitú skúsenosť alebo na niektoré pokročilé funkcie a služby aj expertízu. [39]

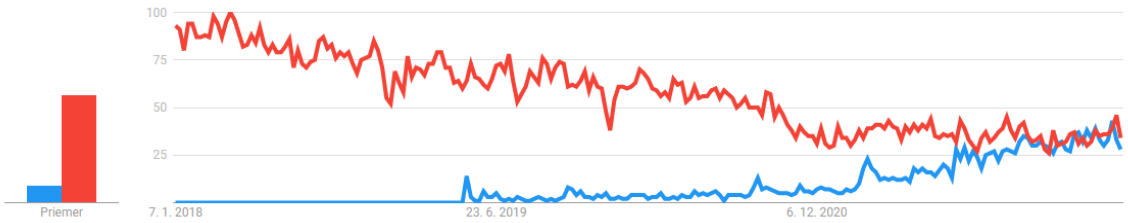
3.8 Užívateľské prostredie

Aplikácia, ktorá neobsahuje jej vizuálnu reprezentáciu a neponúka možnosť interakcie s jej prvkami, by nepôsobila na užívateľa prívetivým dojmom a nebola by veľmi obľúbená. V Androide sú aktuálne dve možnosti k budovaniu užívateľského prostredia: XML a Jetpack Compose, pričom je možná ich kombinácia.

3.8.1 Jetpack Compose

Deklaratívny prístup k budovaniu užívateľského prostredia je rozšírený medzi ostatnými platformami a Google umožnil tento prístup vytvorením moderného frameworku Jetpack Compose.

Už od samotného oznámenia a vydania alpha verzie tohto nástroja dennodenne narastá jeho popularita. Z dôvodu stále väčšej popularity a rastúcemu množstvu obsahu pre Jetpack Compose sa stáva z dlhodobého hľadiska jasným kandidátom kompletnej náhrady za XML.



Obrázok 10 – Porovnanie hľadaných výrazov jetpack compose a android XML v Google trends

Z technického hľadiska je prívetivejší k tvorbe komponentov a deklaratívnym spôsobom je podobný aj mnohým mobilným frameworkom alebo SwiftUI. Kód je čitateľnejší a prehľadnejší za využitia jeho potenciálu. Napríklad vytvorenie vlastného komponentu, ktorý je použitý naprieč celou aplikáciou, dokáže výrazne zmenšiť množstvo napísaného kódu. Použitím deklaratívneho prístupu, sa mení spôsob myslenia pri tvorbe vizuálnych komponentov. Namiesto opisovanie každého jedného kroku stačí k dosiahnutiu cieľa opísať očakávaný výsledok. Stačí teda opísať čo má byť použité a netreba vkladať extra kód na fungovanie prvku.

Funguje na princípe stavania blokov nazývajúcich sa Composable, jedná sa o obyčajnú funkciou označovanú dekorátorom @Composable, spájaním a vkladaním jednotlivých blokov do seba je potom možné vytvoriť ucelenú štruktúru a užívateľské prostredie. Mnoho z týchto Composables už je vstavaných a vývojári si ich prispôsobujú podľa potreby návrhu.

Pre udržanie si pozornosti užívateľa je nevyhnutné aktualizovať informácie na obrazovke. Composable funkcie sú obnovované zakaždým, keď obdržia nový súbor argumentov, ktoré sa s nimi spájajú. Tento proces obnovy sa nazýva rekompozícia a funguje na základe stavu. Stav ako taký, je hodnota alebo viaceré hodnoty priradené ku Composable funkcií. V čase zmeny takejto hodnoty sa spúšťa proces rekompozície a po jeho dokončení obdrží composable funkcia stav s už novou hodnotou.

Štýlovanie komponentov sa uskutočňuje použitím modifikátorov, ktoré sa porovnávajú s XML atribútmi. Úpravou vzhľadu, pridaním informácií o prvku pre postihnutých, reagovať na UI interakcie a ďalšími možnosťami sú modifikátori výrazne flexibilnejšou voľbou.

Navigácia medzi obrazovkami v jedinej aktivite nevyžaduje nutnosť použitia Fragmentov ako to je v prístupe bez použitia tohto frameworku. Stačí vytvoriť NavHost a definovať v ňom všetky cesty a priradiť k nim dané composable funkcie.

[37]

3.8.2 SwiftUI

Deklaratívny, uzavretý UI framework od spoločnosti Apple predstavený v roku 2019 na WWDC. Vývojárom sprostredkováva natívnu možnosť tvorby užívateľského prostredia pre aplikácie bežiace na operačných systémoch iOS, iPadOS, watchOS, tvOS alebo macOS. Od predošlých UiKit a Appkit frameworkov sa líši a preto je nutnosťou si zvyknúť na iný prístup pri jeho používaní. Značná časť všetkých dôležitých zmien a nových prístupov a princípov je zaznamenaná v oficiálnej dokumentácií od Apple.

Disponuje novým layout systémom podporujúcim písanie dynamického UI, ktorý sa prispôsobuje zariadeniu, na ktorom je zobrazovaný a náležite k nemu sa škáluje. Kontrola stavu (state management) je samozrejmosťou a SwiftUI obsahuje mnoho nástrojov od obaľujúcich anotácií (@State alebo @Binding) až po výkonný mechanizmus na správu závislostí – Environment API. Ďalšími dôležitými podporovanými funkciami sú asynchrónnosť, ktorá je často používaná pre získavanie údajov cez internet a neprerušila beh samotnej aplikácie a štýlovanie obsahu.

Značnou výhodou je spätná podpora a interoperabilita s predchádzajúcimi frameworkami AppKit a UIKit, vďaka ktorej nie je nutné prepisovať celý fungujúci kód aplikácie a ponúka vývojárom viac možností a flexibility ich spoločnou kombináciou. [38]

II. PRAKTICKÁ ČASŤ

4 ŠTRUKTÚRA APLIKÁCIE

Kapitola sa zaoberá špecifickým výberom požiadaviek a nástrojov, ktoré boli použité počas tvorby aplikácie. Tá je vo svojej prvej verzii označená ako prvá alpha verzia, nie je určená pre široké publikum verejnosti, ale pre individuálne privátne použitie, od toho sa odvíja celková voľba funkcionálnych a nefunkcionálnych požiadaviek rovnako ako aj výber jednotlivých nástrojov.

4.1 Funkcionálne požiadavky

Zo širokého spektra funkcionalít, ktoré ponúka daný typ aplikácie bolo rozhodnuté pre ukážku základného toku aplikácie a kľúčových požiadaviek k nemu.

- Registrácia – Systém umožňuje registráciu užívateľa do aplikácie
- Prihlásenie – Systém umožňuje prihlásenie užívateľa do aplikácie
- Odhlásenie – Systém umožňuje odhlásenie užívateľa z aplikácie
- Zobrazenie inzerátov – Systém umožňuje zobrazenie všetkých aktuálnych záznamov z databáze na hlavnú obrazovku
- Pridávanie inzerátov – Systém umožňuje pridávanie inzerátov na základe vložených údajov od užívateľa
- Úprava inzerátov – Systém umožňuje úpravu inzerátov za podmienky overenia vlastníka inzerátu
- Mazanie inzerátov – Systém umožňuje mazanie inzerátov za podmienky overenia vlastníka inzerátu
- Navigácia – Systém umožňuje prihlásenému užívateľovi navigovať sa v prostredí aplikácie

4.2 Nefunkcionálne požiadavky

- Veľkosť aplikácie – Inštalačný súbor aplikácie má pod 20 MB
- Užívateľská prívetivosť – Aplikácia má jasné a prehľadné užívateľské prostredie.

- Rýchlosť – Odozva jednotlivých akcií z pohľadu navigácie, načítania položiek alebo autentifikácie je minimálna.
- Lokalizácia – Prezentovaný jazyk a menové jednotky sú v súlade s cieľovým trhom (Česká republika)
- Kompatibilita – Mobilné zariadenia s operačným systémom Android po verziu 6.0 (Marshmallow)
- Bezpečnosť – Bezpečný autentifikačný proces.

5 TECHNOLÓGIE A PROSTRIEDKY PRE IMPLEMENTÁCIU APLIKÁCIE

Po spísaných požiadavkách na aplikáciu začína voľba vhodných nástrojov, vzhľadom k minimalistickému charakteru prvej verzii. Vybrané nástroje sú jednoduchšie na implementáciu, ale napriek tomu plne spĺňajú svoju funkciu.

5.1 Platforma - Android

Rozhodovanie sa pre akú platformu by malo zmysel vyvíjať tento koncept prebiehalo na základe určitých kritérií ako cieľový trh a skupina, osobných preferencií spojených s pracovnými príležitosťami.

Cieľovou skupinou aplikácie sú primárne študenti, mladí, či starí dospelí, ale v podstate každá osoba zháňajúca alebo ponúkajúca bývanie a nerobí jej problém používať moderný mobilný telefón. Na základe dát z prieskumu podielu trhu s operačnými systémami v Európe za posledné roky, je vidieť, že prevyšujú užívatelia s Android zariadeniami. Aj keď multiplatformový vývoj by mohol byť jasnou voľbou z pohľadu času vývoja a pokrytie oboch dôležitých platforiem, nastávali obavy z rýchlosti pri väčšom škálovaní v budúcnosti a nie najlepšia podpora natívnych možností. Vývoj pre iOS nepripadal príliš do úvahy nielen z dôvodu nízkej popularity na našom trhu, pretože tá z dlhodobého hľadiska stúpa, ale hlavne kvôli nízkej miere používania a nulovými vývojovými skúsenosťami. Avšak medzi jeho hlavné výhody z pohľadu vývoja patrí dokumentácia a predom určené pravidlá, ktorých sa musí vývojár držať – vzniká jasne ucelená štruktúra a vyhýba sa tak veľkému množstvu chýb a možnosť systémovej aktualizácie na starších zariadeniach a iPhone zariadenia z roku 2016 podporujú aktuálny iOS 15. 7-ročná podpora je snom každého Android užívateľa, ale hlavne vývojára. Výrazne sa tým totiž zbavujeme problémov s podporou aktuálnych funkcií, nie je nutné dbať uľahčuje práca vývojára, pretože nemusí dbať na prekážky s tým spojené. Osobné preferencie azda zohrali najväčšiu rolu pri výbere nakoľko záujem, viac pracovných príležitostí v Českej Republike, znalosť na používateľskej úrovni, predošlé skúsenosti s vývojom a dostupné fyzické prostriedky hrali v prospech vývoju špecificky pre Android platformu.

5.2 Android Studio

Po určení platformy bol proces uvažovania nad vývojovým prostredím veľmi krátky Android Studio jasnou voľbou, pretože výber nie je široký a žiadne IDE nie je tak prispôsobené pre natívny vývoj, tak ako Googlom odporúčané Android Studio.

Je vidieť, že všetko je prispôsobené k tvorbe Android aplikácií, počnúc už len takou bežnou bočnou lištou zobrazujúcu hierarchiu projektu, v ktorej je možné meniť spôsoby zobrazenia súborov, predvolený štýl zobrazenia je Android – ten zoskupí súbory do modulov a vytvorí tak jednoduchú a prehľadnú štruktúru projektu. Čo sa týka štruktúry, ďalším skvelým prvkom je dolná lišta s rôznorodými možnosťami spomenutými v časti 3.1.1. Nechýbajú ani schopnosti spustenia emulátora priamo v prostredí, alebo analytika výkonových štatistík aplikácie na spúšťanom zariadení.

5.3 Kotlin

Syntaxová výhoda – Kotlin dosiahne rovnakej funkcionality za použitia menšieho množstva kódu ako môžeme vidieť v krátkej ukážke na obrázku č.3 a jeho celková konštrukcia je oveľa jednoduchšia oproti Jave. Bodkočiarky (semicolons) sú voliteľné a preferovane sa vynechávajú úplne.

Tým, že je Kotlin stále pomerne novým jazykom a oproti Jave nie je tak vyspelý, občas je potrebné siahnuť po kóde z Javy, čo kotlinu nerobí žiadny problém vďaka jeho spätnej kompatibilite.

5.4 Jetpack Compose

Zvolený kvôli deklaratívnemu prístupu urýchľujúcemu tvorbu komponentov užívateľského prostredia, jeho rastúcej popularite medzi vývojármi zdieľajúcich svoje skúsenosti na fóra, sľubovanej rýchlosti na novších zaradeniach a náročnosti nájsť aktualizovanú dokumentáciu k tvorbe rozhrania pomocou XML.

5.5 Firebase

Využívanie cloudových služieb backendu dávalo najväčší zmysel a uprednostniť tak backend tretej strany oproti vlastnému, ktorý by bol síce ideálny pre použitie na produkčnej úrovni, kde náklady na údržbu sú s narastajúcim počtom užívateľov výrazne nižšie oproti Firebase. Pre aktuálne predvedenie funkcionálnych požiadaviek postačuje Firebase viac než dosť a nakoľko táto verzia nie je verejne publikovaná v obchode play, tak nie je očakávaný

žiadny nápor na užívateľov, ktorých interakcia s dátami by mohla viesť k zvýšeniu nákladov, tie by mali byť nulové pri ponúkanom free tier pláne.

Ako ďalšou výhodou je vnímaná nízka až takmer nulová odborná znalosť backendu, pretože množstvo komponentov už je predpripravených na použitie a stačí tak rýchle nahliadnutie do dokumentácie.

5.5.1 Autentifikácia

Bola využitá bezpečná a užívateľsky prívetivá forma autentifikácie pomocou Google účtu. Jedinou podmienkou z pohľadu užívateľa k takémuto typu prihlásenia je mať Google účet a táto podmienka je jednoducho pokrytá, pretože ten má každý Android používateľ už od prvotných nastavení zariadenia. Firebase takisto zjednodušuje samotný proces, v ktorom netreba rozlišovať medzi registráciou a prihlásením, pretože o všetko sa starajú už predpripravené služby.

5.5.2 Databáza

Na uchovávanie údajov sú ponúkané sú dve možnosti, Realtime Database a Firestore. Firebase obsahuje dokument, v ktorom opisuje hlavné rozdiely medzi týmito databázami a pomáha tak vybrať to správne riešenie. Na faktoroch ako rýchlosť, dostupnosť, štruktúra dát, bezpečnosť a aktuálnosť sa odvíjalo zvolenie cloudového riešenia Firestore, primárne odporúčaného pre najnovšie mobilné aplikácie.

5.6 Verzovanie a projektový plán

Kód aplikácie bol verzovaný pomocou najpoužívanejšieho verzovacieho nástroja Git, jeho voľba spočínala na jednoduchom fakte a tým je jeho znalosť a popularita medzi vývojármi. Ako cloudový repozitár verzií bol použitý BitBucket, ktorého výhody oproti populárnejšiemu Githubu, v pláne, ktorý je zdarma, sú väčšie úložisko a prepojenie so softvérom na správu projektu Jira spadajúcou pod rovnakú organizáciu Atlassian.

6 TVORBA NÁVRHU

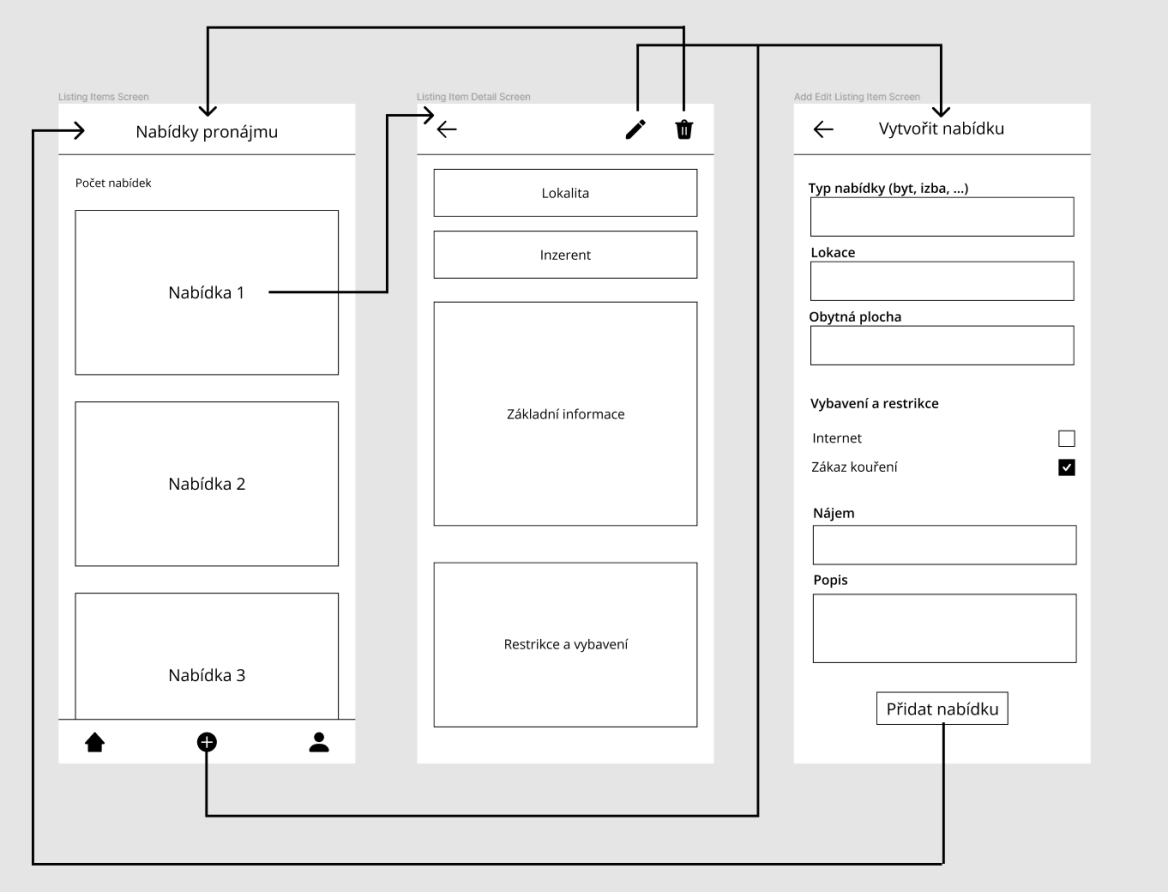
Celková vizuálna identita bola spracované študentkou Fakulty Multimediálnych Komunikácií v Zlíne. Z tohto dôvodu nebude tvorba grafického návrhu bližšie opísaná. Na základe tohto návrhu boli prevzatá farebnosť, výzor niektorých komponentov a určité prechody medzi obrazovkami súvisiace so zostavenými funkcionálnymi požiadavkami. Všetky prechody medzi obrazovkami boli zaznamenané do formy wireframe modelu k uľahčeniu práce pri samotnej integrácií.

6.1 Obrazovky

Príprava wireframe modelu bola realizovaná grafickým programom Figma. Zvolenými a pripravenými obrazovkami pre splnenie určených funkcionálnych požiadaviek sú:

- **Autorizačná obrazovka** – na základe voľby prihlasovania použitím Google účtu cez Firebase sa na nej nachádza jediné funkčné tlačidlo, ktoré po stlačení presmeruje užívateľa na obrazovku so zoznamom všetkých inzerátov.
- **Obrazovka zoznamu inzerátov** – hlavnou časťou sú všetky pridané inzeráty vo vertikálnom zozname s možnosťou rolovania po ose Y, zobrazuje počet aktuálne nahraných inzerátov a v spodnej časti je navigačné menu.
- **Detail obrazovka –** vo vertikálnom usporiadaní zobrazuje všetky detailné informácie poskytnuté pri jeho tvorbe rovnako ako aj kontakt na tvorcu inzerátu, vrchná časť obsahuje jedno až tri navigačné tlačidlá: tlačidlo späť, tlačidlo upraviť inzerát a tlačidlo vymazať inzerát. Tlačidlá upraviť a vymazať sú zobrazené len v prípade, že užívateľom zobrazený detail inzerátu je jeho vlastný.
- **Obrazovka na úpravu alebo pridanie inzerátu** – obrazovka je základom pre dve akcie (pridať a upraviť inzerát) z dôvodu zdieľania rovnakého dizajnu a zmena nastáva len v pred vyplnení vstupných polí v prípade editácie. Podstatou tejto obrazovky je poskytnúť užívateľovi možnosti k vytvoreniu podrobného inzerátu zadaním typu obydlia, presnej lokality, základných a voliteľných údajov k dosiahnutiu čo najlepšieho opisu. Až po vyplnení povinných vstupných hodnôt je dostupné tlačidlo k vytvoreniu inzerátu, ktoré je dovtedy viditeľné, ale neaktívne. Navigačné tlačidlo vo vrchnej časti slúži na vrátenie sa o krok dozadu, podľa toho z ktorej obrazovky bolo navigované.

- **Profilová obrazovka –** zobrazuje základné údaje o prihlásenom používateľovi ako fotku, emailovú adresu a meno. Užívateľ má možnosť odhlásiť sa jedným kliknutím na tlačidlo nachádzajúce sa v dolnej časti.



Obrázok 11 – Wireframe model pre komplexnejšie z obrazoviek

7 PROJEKTOVÁ PRÍPRAVA

Príprava projektu je nemalé množstvo krokov nutné vykonať k tomu, aby bol projekt nachystaný k implementácií všetkých prvkov. Aj keď sa zo začiatku môže príprava zdať ako bagatelizovateľná časť celkového vývoja aplikácie, keďže nemá dopad na jej fungovanie ale dobrým nastavením všetkých nástrojov dokáže ušetriť mnoho času.

7.1 Jira a Bitbucket repozitár

Začiatkom projektu bolo spísanie si všetkých požiadaviek do systému Jira, v ktorom sú prehľadne zobrazené. Výhodou tohto prístupu zapisovania je viditeľnosť odvedenej práce, jej analytika a v prípade, že sa v aplikácií počas testovania našla chyba, tak bola jednoducho zaznamenaná do Jiry medzi ostatné úlohy.

7.2 Android Studio – tvorba projektu a konfigurácia Firebase

Prvým krokom bolo spustenie nainštalovaného Android Studia, jednoduchou navigáciou sa zvolila možnosť vytvorenia nového projektu, nasledovaná voľbou šablóny obsahujúcou už základný predpripravený kód, možností je vskutku veľa, od prázdnej aktivity cez aktivitu prihlasovacieho formulára zložitejšiu aktivitu navigačnej lišty. Z dôvodu prístupu tvorby pomocou frameworku Jetpack Compose bola zvolená šablóna compose aktivity. K dokončeniu tvorby projektu je potrebné už len zadať názov aplikácie, v tomto prípade „Brloh“, zvoliť názov balíku, ktorý by mal predstavovať tvorcu – štúdio, firmu, či osobu, pričom sa dodržuje princíp v pomenovaní „lokalita“ „názov štúdia“ a „názov aplikácie“ každé oddelené bodkou, určí sa cesta, v ktorej je projekt uložený, programovací jazyk a minimálna podpora SDK, podľa požiadavkou ide o SDK verzie 23 (Android 6.0), v tejto časti šikovnou nápomocnou percentuálna hodnota vyjadrujúca množstvo zariadení, ktoré budú schopné spustiť aplikáciu. Finálnym krokom je kliknutie na tlačidlo dokončiť, po ktorom sa otvorí hlavné okno Android Studia, Gradle stiahne a nainštaluje všetky potrebné závislosti a v editačnom okne je pripravená hlavná trieda používajúca vzorovú composable funkciu a v dolnej časti s anotáciou @Preview je ukážka danej composable funkcie.

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            MyApplicationTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    Greeting( name: "Android")
                }
            }
        }
    }
}

@Composable
fun Greeting(name: String) {
    Text(text = "Hello $name!")
}

@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    MyApplicationTheme {
        Greeting( name: "Android")
    }
}
```

Obrázok 12 – Vygenerovaný kód po vytvorení nového projektu

7.2.1 Projektová štruktúra

Stále je však dobré si hierarchicky usporiadať celý projekt, aby s pribúdajúcou zložitosťou neprišiel o svoju prehľadnosť. Logicky usporiadaný projekt môže pomôcť aj v tímových ohľadoch, kedy sa každý člen tímu dokáže v priebehu pár hodín zorientovať a nepotrebuje k tomu dlhú dokumentáciu.

Vstupným bodom vlastnej štruktúry je vo vytvorenom balíku (com.loperdev.brloh) všetko mimo tohto balíku ostáva v základnom nastavení projektu. Zložkami pridanej štruktúry sú:

- common – obsahuje súbory spoločné a zdieľané naprieč projektom, jedná sa o konštanty, abstraktné triedy a iné, dôležitým aspektom je nutná nezávislosť týchto súborov na súboroch iných
- data – v tejto zložke sa nachádzajú všetky súbory manipulujúce so sieťovým tokom dát, ako napríklad prístup k databáze

- di – skratka dependency injection, obsahom je AppModule objekt poskytujúci singleton referencie na triedy
- domain – všetka biznis logika aplikácie, definujúca triedy, funkcie súvisiace s týmito triedami a rozhrania pre repozitáre
- presentation – reprezentuje všetky súbory úzko spojené s tým, čo je prezentované užívateľovi. Sú v nej zložky zastupujúce jednotlivé obrazovky, v ktorých sa nachádzajú súbory prislúchajúce k týmto obrazovkám.
- ui – zložka vytvorená pri spustení projektu, ktorej obsahom sú len veci korešpondujúce s nastavením témy
- utils – projektom zdieľaná zložka s rôznymi pomocnými funkciami

7.2.2 Material Theming

Android disponuje týmto uceleným štýlovaním od verzie 5.0, konfiguráciou hodnôt premenných je dosiahnuté k zjednodušeniu aplikovania chcených štylizačných modifikátorov.

Hodnoty všetkých súborov definujúce štýly sú zoskupené v hlavnom Theme.kt súbore, v ktorom je zostrojená všeobecná composable funkcia aplikujúca tieto definície do vstavanej MaterialTheme composable a jednoduchá logika určujúca, aká paleta farieb sa má aplikovať, je to buď paleta farieb pre tmavý alebo svetlý mód zariadenia.

```
5  val CosyOrange = Color( color: 0xFFFDA440)
6  val White1 = Color( color: 0xFFF7F7F7)
7  val Grey200 = Color( color: 0xFFDDDEDD)
8  val Grey500 = Color( color: 0xFFB4B4B4)
9  val Grey600 = Color( color: 0XFFADADAD)
10 val Black1 = Color( color: 0xFF1F1F1F)
```

Obrázok 13 – Definícia farieb v súbore Color.kt

```
12  private val DarkColorPalette = darkColors(
13      primary = CosyOrange,
14      onPrimary = Color.White,
15      secondary = Black1,
16      onSecondary = White1,
17      background = Color.Black,
18  )
19
20  private val LightColorPalette = lightColors(
21      primary = CosyOrange,
22      onPrimary = Color.White,
23      secondary = White1,
24      onSecondary = Grey500,
25      background = Color.White,
26  )
```

Obrázok 14 – Definícia palety farieb pre tmavý a svetlý režim

7.2.3 Firebase – založenie projektu

Využívanie služieb platformy Firebase je možné len s Google účtom, pre účely vývoja bol teda založený nový osobný účet. Nasledovalo vytvorenie samotného Firebase projektu navigovaním sa na stránku https://console.firebase.google.com/ s pochopiteľným rozhraním, kde je v strednej časti stránky zoznam projektov priradených k používanému účtu – v tomto prípade, keďže s účtom nie sú spojené žiadne projekty, je tam len tlačidlo pre pridanie projektu. Kliknutím na toto tlačidlo sa otvorí prehľadný sprievodca, na začiatku je zvolený názov projektu („Brloh“), Firebase k nemu automaticky vytvorí jedinečný identifikátor, po potvrdení podmienok o používaní je zobrazená druhá časť, v ktorej sú analytické možnosti ponechané a poslednou časťou je priradenie účtu k štatistikám z analýz a projekt je vytvorený do pár sekúnd po kliknutí na tlačidlo vytvoriť projekt.

8 INTEGRÁCIA FUNKCIONALÍT

Po všetkých prípravných fázach bolo možné začať pracovať na samotnej tvorbe aplikácie a integrovať tak jednotlivé funkcionálne požiadavky s použitím vybraných nástrojov pre platformu Android.

8.1 Autentifikácia

Prihlásením sa do Android Studia s rovnakým Gmail účtom k akému je priradený Firebase projekt, sa automaticky uskutoční prepojenie Android Studia s Firebase. Medzi nástrojmi v Android Studiu sa nachádza položka Firebase asistenta, v ktorom sú návody na aktiváciu ponúkaných služieb. Splnenie prvých troch požiadaviek – registrácia, prihlásenie a odhlásenie užívateľa je uskutočnená nasledovaním tohto návodu v časti „Authentication“.

8.2 Inzeráty

Základnou časťou celej aplikácie sú inzeráty uložené v databáze. Inzeráty sú prezentované užívateľovi vo forme, v ktorej s nimi môže interagovať štyrmi spôsobmi.
V zložke di je vytvorený objekt AppModule slúžiaci ako poskytovateľ referencií na Firestore databázu, konkrétnu kolekciu a repozitára implementujúceho všetky CRUD operácie.

```
@Module
@InstallIn(SingletonComponent::class)
object AppModule {

    @Provides
    @Singleton
    fun provideFirebaseFirestore() =  FirebaseFirestore.getInstance()

    @Provides
    @Singleton
    fun provideListedItemRef(db: FirebaseFirestore) =
        db.collection(LISTED_ITEMS_COLLECTION_ID)

    @Provides
    @Singleton
    fun provideListedItemList(listedItemRef: CollectionReference) : ListedItemRepository {
        return ListedItemRepositoryImpl(listedItemRef)
    }
}
```

Obrázok 15 – Ukážka kódu objektu AppModule

Pre vyhnutie sa písaniu opakovaciemu sa kódu je použitá knižnica Dagger-Hilt, ktorej účel je zjednodušenie implementácie architektonického vzoru dependency injection. Dagger-Hilt vnára závislosti v čase kompilácie na základe priradených anotácií s kľúčovými slovami

v kóde vygenerovaním tried pre reálne poskytnutie závislostí. Do Gradle súborov boli importované všetky závislosti týkajúce sa tejto knižnice ako je vidieť na obr. 15 a 16.

```
plugins {
    // base plugins
    id 'com.android.application'
    id 'kotlin-android'
    // kotlin annotation processing & dagger hilt
    id 'kotlin-kapt'
    id 'dagger.hilt.android.plugin'
    // google services
    id 'com.google.gms.google-services'
    id 'com.google.devtools.ksp' version '1.6.20-1.0.5'
}
```

Obrázok 16 – Použité pluginy v build.gradle (Module) súbore

```
// Dagger - Hilt
implementation 'com.google.dagger:hilt-android:2.41'
kapt 'com.google.dagger:hilt-compiler:2.41'
implementation "androidx.hilt:hilt-navigation-compose:1.0.0"
```

Obrázok 17 – importovanie Dagger-Hilt závislostí v build.grade (Module) súbore

Vstupným bodom pre túto knižnicu je hlavná trieda aplikácie, aby mala prístup ku všetkým závislostiam celej aplikácie. Obsahom hlavnej triedy je len jej názov, dedenie z triedy Application() a anotácia @HiltAndroidApp, názov triedy musí byť definovaný ešte v AndroidManifest.xml súbore.

Dôležitý prvok, ktorý by nemal chýbať je trieda inzerátu používaná naprieč celou aplikáciou, keďže jedinou úlohou tejto triedy je držať dáta typ triedy je dátová. Trieda obsahuje základné dátové typy, vlastné triedy ako User a OptionalCheckboxItem a špeciálnou anotáciou @ServerTimeStamp, ktorá Firebase poskytuje informáciu o tom, že sa jedná o TimeStamp typ. Znak otázniku, ktorý nasleduje za udaním typu premennej označuje, že daná premenná je nullable a teda môže naberať hodnoty null. Zaujímavosťou môže byť companion object spojený s touto triedou, jedná sa o vnorený objekt nachádzajúci sa v tejto triede a je k nemu možné pristupovať použitím dot notácie.

```
data class ListedItem(
    // basic info - autocompleted
    val id: String = "",
    @ServerTimestamp
    val createdDate: Timestamp? = null,
    val createdBy: User? = null,
    val totalMonthlyRent: Int = -1,
    // basic info - user's input
    val city: String = "",
    val street: String = "",
    val houseNumber: String? = "",
    val houseCategory: String = "",
    val desc: String? = null,
    val roomCount: Int = -1,
    val energiesMonthly: Int? = -1,
    val openSpace: Int = -1,
    val rent: Int = -1,
//     facilities
    val facilities: List<OptionalCheckboxItem>? = null,
    // restrictions
    val restrictions: List<OptionalCheckboxItem>? = null,
    val maxPersonCount: Int = -1,
) {
    companion object {
        val houseCategoryList = listOf("Byt", "Rodinný dúm", "Izba")
        val facilityIcons = mapOf(...)
        val restrictionIcons = mapOf(
            CheckboxIcon.CIGARETTE to R.drawable.ic_cigarette,
            CheckboxIcon.PETS to R.drawable.ic_pets,
        )
    }
}

class InvalidListedItemException(message: String): Exception(message)
```

Obrázok 18 – Dátová trieda pre inzerát s názvom ListedItem

Singleton trieda poskytuje implementáciu rozhrania pre inzeráty – ListedItemRepositoryImpl, jedná sa o triedu, ktorá má v konštruktore referenciu na kolekciu inzerátov a sú v nej implementované funkcie pracujúce s dátami pre získanie všetkých inzerátov, získanie len jedného inzerátu podľa identifikátoru, pridanie jedného inzerátu, aktualizáciu a vymazanie inzerátu.

```
interface ListedItemRepository {
    fun getListedItems(): Flow<Resource<List<ListedItem>>>

    fun getListedItemById(id: String): Flow<Resource<ListedItem>>

    suspend fun addListedItem(listedItem: ListedItem)

    suspend fun updateListedItem(id: String, listedItem: ListedItem)

    suspend fun deleteListedItem(id: String)

}
```

Obrázok 19 – Ukážka kódu rozhrania s definovanými metódami

V podstate sa jedná sa o implementovanie všetkých metód z rozhrania, ich volanie vo ViewModel triedach a vytvorenie obrazoviek pre vizuálnu reprezentáciu, pomocou ktorej užívateľ vidí výstupy reagujúce na jeho interakcie.

Každý definovaný stav, ktorý sa nachádza na obrazovke a je teda odkrytý vonkajším vplyvom, je rozdelený na stav súkromný viditeľný len vo VM a jeho kópiu uloženú vo verejne prístupnej premennej, s ktorou sa interaguje zo strany užívateľa. Niektoré stavy sú neprístupné z vonka VM, aby sa predišlo nechceným zmenám a dodržali sa základy objektového programovania.

```
private val _listedItemCity = mutableStateOf(
    ListedItemTextFieldState(placeholderText="Praha, Zlín, ...", labelText = "Město*", maxTextLength = 35)
)
val listedItemCity: State<ListedItemTextFieldState> = _listedItemCity

private val _listedItemStreet = mutableStateOf(
    ListedItemTextFieldState(placeholderText="Zahradní, Krátká, ...", labelText = "Ulice*", maxTextLength = 50)
)
val listedItemStreet: State<ListedItemTextFieldState> = _listedItemStreet

private val _listedItemHouseNumber = mutableStateOf(
    ListedItemTextFieldState(placeholderText="4732", labelText = "Popisné číslo", maxTextLength = 12)
)
val listedItemHouseNumber: State<ListedItemTextFieldState> = _listedItemHouseNumber

private val _listedItemHouseCategory = mutableStateOf(ListedItem.houseCategoryList[0])
val listedItemHouseCategory: MutableState<String> = _listedItemHouseCategory
private val _listedItemFacilityList = mutableStateListOf<OptionalCheckboxItem>(
    OptionalCheckboxItem(name = "internet", displayName = "Internet", icon = CheckboxIcon.WIFI),
    OptionalCheckboxItem(name = "parking", displayName = "Parkování", icon = CheckboxIcon.PARKING),
    OptionalCheckboxItem(name = "garden", displayName = "Zahrada", icon = CheckboxIcon.GARDEN),
    OptionalCheckboxItem(name = "elevator", displayName = "Výtah", icon = CheckboxIcon.ELEVATOR),
    OptionalCheckboxItem(name = "balcony", displayName = "Balkón", icon = CheckboxIcon.BALCONY),
)
val listedItemFacilityList: List<OptionalCheckboxItem> = _listedItemFacilityList
```

Obrázok 20 – Ukážka časti definovaných modifikovatelných stavov v AddEditListedItemViewModel triede

8.2.1 Pridávanie a editácia

Obe funkcie majú ako vstupný parameter objekt inzerátu a pre aktualizáciu je potrebné dodať ešte identifikátor inzerátu, aby sa dal nájsť v kolekcií. Tieto funkcie zdieľajú rovnaký adresár, pretože ich funkcionalita je takmer totožná a najväčší zmysel dáva mať pre ne totožnú obrazovku a ViewModel triedu.

ViewModel obsahuje definíciu modifikovateľných stavov (mutable state). Väčšinou sa jedná o stavy dátovej triedy ListedItemTextFieldState (obr. 21), ktorá drží stav jednotlivých textových polí použitých v prezentačnej časti na obrazovke.

```
data class ListedItemTextFieldState(
    val text: String = "",
    val placeholderText: String = "",
    val labelText: String = "",
    val maxTextLength: Int,
)
```

Obrázok 21 – Dátová trieda ListedItemTextFieldState

Každý argument dátovej triedy ListedItem, okrem argumentov typu Boolean má vlastný stav, pretože môže naberať rôznych hodnôt a pri používaní by bolo ťažké rozlíšiť, akému stavu sa má priradiť užívateľom zadaná hodnota S argumentmi definujúcimi dodatočné vlastnosti bytu ako vybavenie alebo reštrikcie ponúkanej nájomnej jednotky to bolo jednoduchšie vytvorením ďalšej dátovej triedy CheckboxItem, uchovávajúcej údaje o určitom vybavení alebo reštrikcií (obr. 22) a následne je použitý upravovateľný zoznam stavov (mutable state list).

```
data class OptionalCheckboxItem(
    val name: String = "",
    val displayName: String = "",
    val icon: CheckboxIcon = CheckboxIcon.NO_ICON,
    val selected: Boolean = false,
    val type: CheckboxItemType = CheckboxItemType.FACILITY
)
```

Obrázok 22 – Dátová trieda OptionalCheckboxItem

ViewModel reaguje na každú zmenu, ktorú na obrazovke vykoná užívateľ, preto je v ňom vytvorená funkcia s argumentom abstraktnej triedy AddEditListedItemEvent, v ktorej sú definované všetky udalosti. Tie sú zachytávané pomocou podmienkového výrazu when, ktorého vetvy sú všetky udalosti v triede AddEditListedItemEvent.

```
is AddEditListedItemEvent.EnteredRent -> {
    _listedItemRent.value = listedItemRent.value.copy(
        text = event.value.take(_listedItemRent.value.maxTextLength).filter { it.isDigit() }
    )
}
```

Obrázok 23 – Ukážka kódu reagujúceho na zmenu hodnoty v poli nájmu

Reakciou na udalosť ukladania inzerátu, je logika spúšťajúca sa v novom vlákne. Na jej začiatku sa nastaví stav nahrávania na kladnú hodnotu, do premennej sa inicializuje inzerát s vloženými hodnotami, prebehne jednoduchá kontrola, či sa jedná o už uložený inzerát a šlo

by o jeho aktualizáciu, alebo o nový (do funkcie nie je predaný argument identifikátoru) a podľa toho je zavolaná prislúchajúca funkcia z rozhrania repozitára kolekcie inzerátov. Ku koncu je na pevno nastavené držanie funkcie (delay) po dobu jeden a pol sekundy simulujúci dobu nahrávania, ktorá je inak okamžitá, keďže na server sú nahrávané len základné typy, stav nahrávania sa končí a na obrazovku sa vyšle udalosť o uložení. Ak sa uloženie nepodarí, na obrazovku sa pošle udalosť o tomto nezdare.

```
is AddEditListedItemEvent.SaveListedItem -> {
    viewModelScope.launch {  this: CoroutineScope
        _listedItemIsUploading.value = true
        val listedItem = createListedItem(currentListedItemId)
        try {
            if (currentListedItemId.isNullOrBlank()) {
                listedItemRepository.addListedItem(listedItem)
            } else {
                listedItemRepository.updateListedItem(
                    id = currentListedItemId!!,
                    listedItem = listedItem
                )
            }
            delay( timeMillis: 1500)
            _listedItemIsUploading.value = false
            currentListedItemId = null
            _eventFlow.emit(UiEvent.SaveListedItem)
        } catch (err: InvalidListedItemException) {
            _eventFlow.emit(
                UiEvent.ShowSnackbar(
                    message = err.localizedMessage ?: "Nejsem schopen uložit tvůj brloh"
                )
            )
        }
    }
}
```

Obrázok 24 – Ukážka kódu reagujúceho na udalosť o uložení inzerátu

Prípad aktualizácie záznamu sa líši len pri načítaní obrazovky. Inicializácia VM predchádza prezentovanie obrazovky, takže v jeho inicializačnom procese sa kontroluje, či obdržal identifikátor z predaných navigačných argumentov. Tento identifikátor je predaný funkcií getListedItemById() a v prípade nájdenia tohto inzerátu sa skopírujú jeho hodnoty do hodnoty polí vo VM.

Obrazovka pozostáva zo získania prístupu k stavom z VM, vytvorením bloku pre zachytávanie udalostí z VM a layout štruktúry, v ktorej vlastne navrhnuté komponenty v podobe navigačných tlačidiel, polí so zaoblenými rohmi určené na vstupy užívateľa

a tlačidlo vyvolávajúce udalosť na uloženie objektu, ktoré je stlačiteľné až po vyplnení povinných údajov.

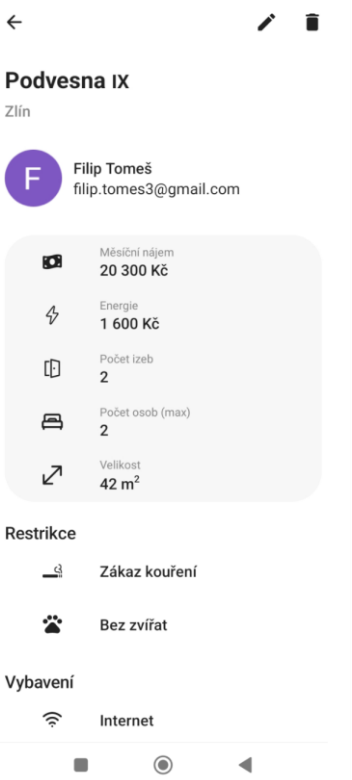
```
// rent
RoundedTextField(
    modifier = Modifier.fillMaxWidth( fraction: .55f),
    text = rentState.text,
    placeHolderText = rentState.placeholderText,
    onValueChange = { it: String
        addEditListedItemViewModel.onEvent(AddEditListedItemEvent.EnteredRent(it))
    },
    labelText = rentState.labelText,
    numeric = true
)
```

Obrázok 25 – Ukážka komponentu vstupného poľa určeného pre nájom

8.2.2 Zobrazenie

Inzeráty sú zobrazované vo forme zoznamu a po kliknutí na jednu z položiek je zobrazený detailná obrazovka tohto inzerátu.

Obrázok 26 – Ukážka obrazovky zobrazujúcej všetky inzeráty vo vertikálnom zozname



Obrázok 27 – Ukážka obrazovky zobrazujúcej detailné informácie o inzeráte

Repozitárom definované metódy (obr. 19) vracajú tok údajov (Flow), ktorý vyvoláva stavy v akých sa v danom čase nachádza, tie sú zachytávané abstraktnou triedou Resource zdieľanou naprieč projektom v zložke common.

```
sealed class Resource<T>(val data: T? = null, val message: String? = null) {
    class Success<T>(data: T) : Resource<T>(data)
    class Error<T>(message: String?, data: T? = null) : Resource<T>(data, message)
    class Loading<T>(data: T? = null) : Resource<T>(data)
}
```

Obrázok 28 – Ukážka kódu abstraktnej triedy Resource

Štruktúra tejto triedy je často používaná medzi rôznymi vývojármi a stala sa tak bežnou súčasťou mnohých projektov.

Funkcie fungujú na rovnakom princípe a tým je zo začiatku emitovať stav načítavania, získať objekty a každý získaný JSON objekt prekonvertovať na triedu ListedItem, rozdiel je iba v tom, že getListedItemById() konvertuje len prvý nájdený objekt, ktorého identifikátor sa stotožňuje s identifikátorom poskytnutým funkcií.

```
override fun getListedItems(): Flow<Resource<List<ListedItem>>> = flow {  this: FlowCollector<Resource<List<ListedItem>>>
    try {
        emit(Resource.Loading<List<ListedItem>>())
        val listedItemList = listedItemsRef.get().await().map { documents ->
            documents.toObject(ListedItem::class.java)
        }
        emit(Resource.Success<List<ListedItem>>(data = listedItemList))
    } catch (err: Exception) {
        emit(Resource.Error<List<ListedItem>>(message = err.localizedMessage ?: "Chyba v internetovém připojení"))
    }
}
```

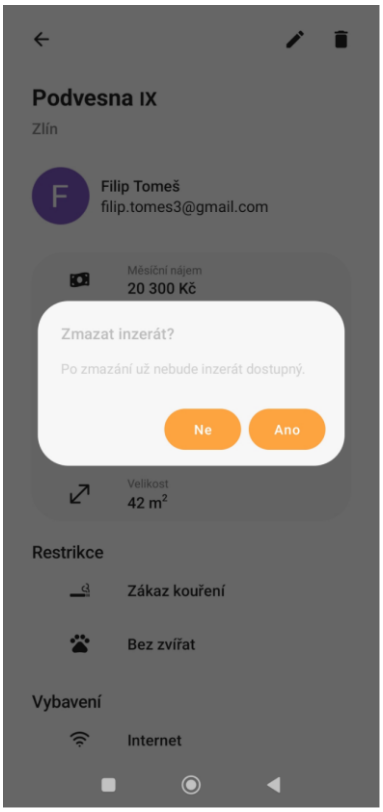
Obrázok 29 – Ukážka kódu funkcie získavajúcej všetky inzeráty

```
override fun getListedItemById(id: String): Flow<Resource<ListedItem>> = flow {  this: FlowCollector<Resource<ListedItem>>
    try {
        emit(Resource.Loading<ListedItem>())
        val listedItem = listedItemsRef.whereEqualTo( field: "id", id).get().await().map { documents ->
            documents.toObject(ListedItem::class.java)
        }.first()
        emit(Resource.Success<ListedItem>(data = listedItem))
    } catch (err: Exception) {
        emit(Resource.Error<ListedItem>(message = err.localizedMessage?: "Chyba v internetovém připojení"))
    }
}
```

Obrázok 30 – Ukážka kódu získavajúcej jeden inzerát na základe identifikátoru

8.2.3 Mazanie

Funkciu mazania je možné vyvolať len z obrazovky detailu inzerátu a to len za podmienky, že prihlásený užívateľ je zároveň vlastníkom tohto inzerátu. Jediným parametrom je identifikátor, podľa ktorého nájde a vymaže záznam z kolekcie. Stále je však potrebné dbať na použiteľnosť pre užívateľa a ten by určite nebol rád, keby po stlačení ikony na vymazanie bol inzerát hneď stratený. Často sa vyskytujúcim spôsobom alebo správaní sa, je zobrazenie potvrdzujúceho dialógového okna. (obr. 31).



Obrázok 31 – Ukážka dialógového okna, zobrazeného po kliknutí na ikonu kôš pre zmazanie inzerátu

Stlačením tlačidla s ikonou odpadkového koša vyvolá akciu na zobrazenie dialógového okna s dodatočným potvrdením tejto akcie.

```
IconButton(
    onClick = { listedItemConfirmDelete.value = true },
    modifier = Modifier.background(MaterialTheme.colors.background)
) {
    Icon(Icons.Rounded.Delete, contentDescription: "zmazat inzerát ikona")
}
```

Obrázok 32 – Ukážka kódu tlačidla pre vymazanie inzerátu

```
if (listedItemDetailViewModel.isUserOwner() && listedItemConfirmDelete.value) {
    ConfirmModal(
        title = "Zmazat inzerát?",
        contentText = "Po zmazání už nebude inzerát dostupný.",
        onDismiss = { listedItemConfirmDelete.value = false },
        onConfirm = {
            listedItemConfirmDelete.value = false
            listedItemDetailViewModel.onEvent(
                ListedItemDetailEvent.DeleteListedItem(listedItemDetailState.listedItem)
            )
        }
    )
}
```

Obrázok 33 – Ukážka kódu využitia vlastného ConfirmModal komponentu

8.3 Navigácia

Jetpack Compose má už vstavanú funkcionalitu navigácie medzi jednotlivými obrazovky. Základ tvorí composable funkcia Navigation, pomocou vytvorenej navController inštancie je možné sa navigovať medzi zvolenými cestami a predávať im argumenty . Vytvorenie ciest a priradenie k nim prislúchajúce composable funkcie obrazoviek je realizované zavolaním vstavanej composable funkcie NavHost, ktorej potrebným argumentom je navController ako voliteľný argument predávaný tejto funkcií je startDestination – konkrétne ide o destináciu k tzv. splash screen zobrazenej pri vstupe do aplikácie. Po tom čo uplynie pevne stanovená doba zobrazenia „Splash“ obrazovky, užívateľ je navigovaný podľa stavu prihlásenia – na Autentifikačnú obrazovku, ak je prihlásený, v opačnom prípade na obrazovku so všetkými inzerátmi.

```
@HiltViewModel
class NavigationViewModel @Inject constructor(

) : ViewModel() {

    private val _startDestinationRoute = mutableStateOf<String>(Screen.AuthUserScreen.route)
    val startDestinationRoute: State<String> = _startDestinationRoute

    private fun setStartDestinationRoute() = viewModelScope.launch { this: CoroutineScope
        Firebase.auth.currentUser.let { it: FirebaseUser?
            if (it != null) {
                _startDestinationRoute.value = Screen.ListedItemsScreen.route
            }
        }
    }

    init {
        setStartDestinationRoute()
    }
}
```

Obrázok 34 – Ukážka kódu navigačnej triedy ViewModel, rozhodujúcou o prvej obrazovke prezentovanej užívateľovi

Len v prípade ListedItemDetailScreen a AddEditListedItemScreen obrazoviek sa predávajú argumenty, keď sú volané. U ListedItemDetailScreen je to potrebný argument a v prípade AddEditListedItemScreen voliteľný, nakoľko sa môže jednať o pridávanie inzerátu kedy identifikátor ešte neexistuje (obr. 35). Ostatné obrazovky sú definované cestou (route) a composable funkciou obrazovky.

```
composable(
    route = Screen.ListedItemDetailScreen.route + "/{listedItemId}",
    arguments = listOf(navArgument(name = "listedItemId") { this: NavArgumentBuilder
        type = NavType.StringType
        nullable = false
    })
) { it: NavBackStackEntry
    ListedItemDetailScreen(navController = navController)
}
composable(
    route = Screen.AddEditListedItemScreen.route + "?listedItemId={listedItemId}",
    arguments = listOf(navArgument(name = "listedItemId") { this: NavArgumentBuilder
        type = NavType.StringType
        defaultValue = ""
    })
) { it: NavBackStackEntry
    AddEditListedItemScreen(navController = navController)
}
```

Obrázok 35 – Ukážka kódu definovania cesty, argumentov a priradenie composable funkcie pre obrazovky ListedItemDetailScreen a AddEditListedItemScreen

Samotný proces navigovania sa na určitú obrazovku je možné jednoduchým zavolaním funkcie navigate() na navController, do ktorej je vložená cesta. V niektorých prípadoch je ešte nutné dbať na backStack – zoznam obrazoviek, na ktoré bolo navigované sa ukladá do zoznamu v zmysle odstraňovania predošle otvorených navigačných ciest, napríklad keď sa užívateľ prepína medzi obrazovkami použitím spodnej navigačnej lišty. Nebolo by vhodné ukladať všetky cesty, jednak by to zbytočne zahlcovalo pamäť a za druhé by pri použití systémového tlačidla späť bol stále vracaný o jednu položku v backStacku naspäť, čo vplýva na zlý zážitok z aplikácie.

```
BottomNavigationItem(
    selected = currentRoute == bottomNavItem.screenRoute,
    onClick = {
        navController.navigate(bottomNavItem.screenRoute) { this: NavOptionsBuilder
            // pop up to the start destination of the graph
            popUpTo(Screen.ListedItemsScreen.route)
            launchSingleTop = true
        }
    },
    icon = {
        Icon(
            imageVector = bottomNavItem.icon,
            contentDescription = bottomNavItem.navItemTxt
        )
    },
    label = { Text(text = bottomNavItem.navItemTxt) },
    alwaysShowLabel = true,
    unselectedContentColor = LocalContentColor.current.copy(alpha = ContentAlpha.disabled)
)
```

Obrázok 36 – Ukážka časti kódu navigačného tlačidla, ktoré po navigovaní dodatočne vymaže celý backStack až po cestu obrazovky ListedItemsScreen

9 TESTOVANIE

Bez testovania by dnes nevznikol žiadny produkt a ak áno je veľká šanca, že sa v ňom nachádzajú nezachytené chyby pri vývoji. Testovací proces pozostával z manuálneho testovania implementovaných funkcionalít, komponentov a ich spoločného fungovania, pri narazení na spomalenia boli využité prostriedky Android Studia na analýzy vyťaženosti siete, pamäte RAM, procesoru a batérie.

9.1 Zariadenia

Testy boli vykonávané na fyzických zariadeniach s odlišnými verziami operačného systému Android, hlavne kvôli pokrytiu používateľov so staršími zariadeniami, ktoré už nemajú možnosť aktualizovať na najnovšiu verziu Androidu ale aj kvôli rôznym proporciám a rozlíšeniu obrazovky a výkonnostným rozdielom.

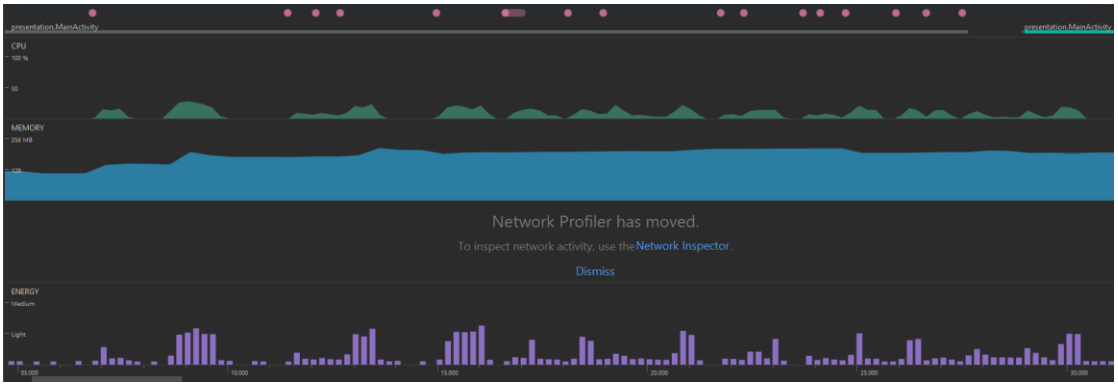
Hlavným zariadením používaným pri celom vývoji bolo zariadenie značky Xiaomi rady Poco X3 Pro s operačným systémom Android verzie 12.

Názov zariadenia	Android verzia	Rozlíšenie obrazovky (px)
Xiaomi Poco X3 Pro	12	1080x2400
Motorola One Vision	10	1080x2520
Xiaomi Redmi 6	8.1	720x1140

Tabuľka 1 – Zariadenia použité k testovaniu aplikácie

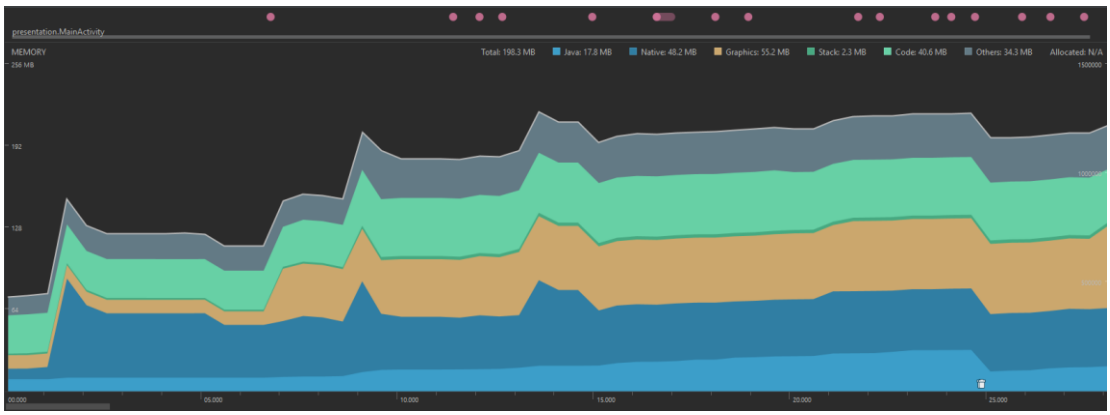
9.2 Výkonové parametre aplikácie

Nevyhnutnou testovacou časťou bolo profilovanie výkonu aplikácie, aby v určitých momentoch nevyužívala enormné množstvo pamäte, procesoru, čo by mohlo mať vplyv na spotrebu batérie, alebo aj kontrola sieťových operácií pri získavaní alebo nahrávaní dát do Firebase. Na toto bol využitý v Android Studiu integrovaný Android Profiler, ktorý detailne zobrazuje všetky potrebné údaje k analýze.



Obrázok 37 – Ukážka Android Profiler, v ktorom je analýza priebehu akcií aplikácie

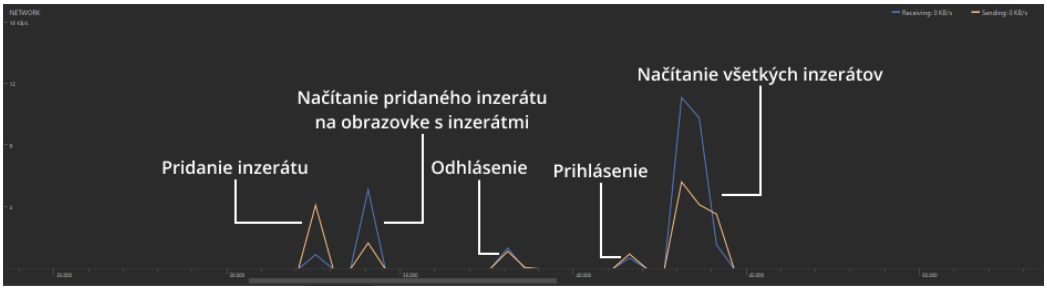
Profilér vo vrchnej časti červenými bodkami označuje spustené udalosti po dotyku – touch events. Ostatné výkonové parametre sú horizontálne oddelené a pre detailnejší pohľad je možné rozkliknúť každú jednu sekciu.



Obrázok 38 – Detailná ukážka z využitia pamäte RAM

Z priebehu využitia pamäte RAM na obr. 37 je zrejmé, že aplikácia zaberá viacero zdrojov po načítaní inzerátov, ale následne sa počas celého procesu navigovania medzi obrazovkami, pridávania, mazania a ostatných spúšťaných procesov v rámci aplikácie pohybuje v ustálených hodnotách s maximálnym celkovým nameraným vyťažením 198,3 MB.

Priebeh využitia siete závisí od aktuálne vykonávaného požiadavku. Najväčším vyťažením je získanie všetkých inzerátov, tento požiadavok je vykonaný len pri načítaní aplikácie, inak tento požiadavok načíta len aktualizované alebo pridané inzeráty v kolekcií.



Obrázok 39 – Detailná ukážka využitia siete

ZÁVER

Zameraním bakalárskej práce bola tvorba natívnej aplikácie pre ľudí s možnosťou inzerovať bytové priestory k prenájmu alebo pre tích, ktorí prenájom bytových priestorov naopak zháňajú. Od samého začiatku bolo jasné, že z časového hľadiska by tvorba tak komplexnej aplikácie nebola možná a tak bol vytvorený jej základný koncept.

Pod slovom tvorba nie je myslená len implementačná fáza, pretože tú predchádzajú rozobrané kľúčové procesy k získaniu údajov o prieskume trhu, určením si cieľovej skupiny používateľov, predstavenie a zhodnotenie možností prístupom k natívnemu, ale okrajovo aj multiplatformovému vývoju a porovnanie rôznych nástrojov spojených s vývojom. Z celkového prehľadu vývoja mobilných aplikácií by bolo z časového hľadiska výhodnejšie venovať úsilie do multiplatformového riešenia, avšak týmto spôsobom by nebolo dosiahnuté rýchlostných benefitov a osobných preferencií. Konceptuálne bolo dôležité získať ďalšie skúsenosti s čerstvo vydaným nástrojom pre tvorbu užívateľského prostredia plnohodnotným použitím namiesto už starnúceho prístupu tvorby pomocou XML.

Aplikácia je teda prispôsobená pre Android zariadenia využitím moderných nástrojov k tvorbe. O autentifikáciu pomocou Google účtu a poskytnutie databázových a analytických služieb sa stará cloudová platforma Firebase. Prostredie aplikácie poskytuje užívateľovi priamočiaru navigáciu a interakciu s komponentami a ako príjemný bonus sa prostredie prispôsobí svojím svetlým alebo tmavým motívom podľa nastavení zariadenia. Podľa získaných hodnôt nameraných pri výkonových testoch je bezproblémové použitie aj na zariadeniach so slabšími hardvérovými komponentami. Všetky funkcionálne požiadavky na koncept boli splnené a tak je momentálne zameraná aplikácia viac pre inzerenta, ten sa môže prihlásiť do aplikácie, po ktorom mu je prezentovaný zoznam všetkých inzerátov, použitím navigácie môže detailne opísať a pridať svoj inzerát a následne kedykoľvek v budúcnosti úpravami aktualizovať alebo zmazať.

Tvorba konceptu poskytla potrebné vedomosti k nadobudnutiu skúseností z oblastí potrebných pre tvorbu natívnych aplikácií a nasledovať budú kroky pre pridávanie funkcionalít v časti filtrovania inzerátov, hľadania podľa lokality, možnosť pridávania fotiek k inzerátom, možnosť textovej komunikácie medzi užívateľmi a integrovaním nových databázových riešení pre vkladanie fotiek k inzerátom. Až po vykonaní týchto dlhodobých plánov bude možné uviesť aplikáciu na trh.

ZOZNAM POUŽITEJ LITERATÚRY

[1] Mobile App Development Process: Step-by-Step Guide. Invonto [online]. Team Invonto, 21.4.2021 [cit. 2022-02-26]. Dostupné z: https://www.invonto.com/insights/mobile-app-development-process/

[2] How to Find Your Target Audience for Mobile Apps [online]. App Radar, 2019 [cit. 2022-02-26]. Dostupné z: https://appradar.com/blog/how-to-find-your-target-audience-for-mobile-apps

[3] How to Perform Competitive Analysis for your Mobile App Idea? [online]. Shivam Srivastava, 2021 [cit. 2022-02-26]. Dostupné z: https://appinventiv.com/blog/perform-competitive-analysis-mobile-app-idea/

[4] 10 Features That Make a Really Great Mobile App [online]. huree, 2018 [cit. 2022-02-26]. Dostupné z: https://blog.hurree.co/blog/10-features-that-make-a-really-great-mobile-app

[5] Mobile App Update: Why and How [online]. Ayo Oladele, 2021 [cit. 2022-02-26]. Dostupné z: https://www.velvetech.com/blog/mobile-app-update/

[6] How to Write an Effective Mobile App Product Requirements Document [online]. clearbridgemobile [cit. 2022-02-26]. Dostupné z: https://clearbridgemobile.com/how-to-build-a-mobile-app-requirements-document/

[7] Understanding Functional and Non-Functional Requirements in App Development [online]. Hiral Atha, 2021 [cit. 2022-02-26]. Dostupné z: https://www.moveoapps.com/blog/functional-and-non-functional-requirements-in-app-development/

[8] How to Write a Mobile App Product Requirements Document [online]. Kate Pismennaya, 2022 [cit. 2022-02-26]. Dostupné z: https://themindstudios.com/blog/mobile-app-requirements-document/

[9] Functional and Non-Functional Requirements for Mobile App: What's the Difference? [online]. Lvivity Team, 2021 [cit. 2022-02-26]. Dostupné z: https://lvivity.com/functional-and-non-functional-requirements

[10] Complete Guide to Creating Mobile App Wireframes [online]. Jill DaSilva, 2020 [cit. 2022-02-26]. Dostupné z: https://xd.adobe.com/ideas/process/wireframing/guide-to-creating-mobile-app-wireframes/

[11] How to Wireframe an App: Guide [online]. Roman Bord, c2017 [cit. 2022-02-26]. Dostupné z: https://stormotion.io/blog/why-is-wireframing-of-your-mobile-app-so-important/

[12] Building a Visual Language: Behind the scenes of our new design system [online]. Karri Saarinen, 2016 [cit. 2022-03-13]. Dostupné z: https://medium.com/airbnb-design/building-a-visual-language-behind-the-scenes-of-our-airbnb-design-system-224748775e4e

[13] The complete guide to mobile app development in 2022 [online]. John Adam, 2022 [cit. 2022-03-16]. Dostupné z: https://kruschecompany.com/mobile-app-development/

[14] Benefits of developing native apps [online]. Ivan Trogrlic, 2021 [cit. 2022-03-16]. Dostupné z: https://decode.agency/article/native-app-development-benefits/

[15] Native vs Hybrid vs Cross Platform – What to Choose in 2022? [online]. Native vs Hybrid vs Cross Platform – What to Choose in 2022?, 2021 [cit. 2022-03-16]. Dostupné z: https://www.netsolutions.com/insights/native-vs-hybrid-vs-cross-platform/

[16] IOS vs Android Development for Your Mobile App: Which is Better? [online]. Oskar Mieczkowski, 2021 [cit. 2022-03-19]. Dostupné z: https://www.monterail.com/blog/ios-vs-android-development

[17] ANDROID VS IOS DEVELOPMENT: WHICH PLATFORM SHOULD I DEVELOP FOR FIRST? [online]. Rami Anwa, 2020 [cit. 2022-03-31]. Dostupné z: https://easternpeak.com/blog/android-vs-ios-development-which-platform-first/

[18] 11 Best Android IDEs for Developers of 2022 [online]. Harvey, 2021 [cit. 2022-04-01]. Dostupné z: https://www.developer.com/mobile/top-android-ides-for-developers/

[19] Swift vs Objective-C: Which is Ideal for IOS App Development? [online]. Dharmik, 2021 [cit. 2022-04-02]. Dostupné z: https://www.intuz.com/blog/objective-c-v-s-swift-for-ios-app-development

[20] What is Java? Definition, Meaning & Features of Java Platforms [online]. Hartman, 2022 [cit. 2022-04-02]. Dostupné z: https://www.guru99.com/java-platform.html

[21] Google is adding Kotlin as an official programming language for Android development [online]. Miller, 2022 [cit. 2022-04-02]. Dostupné z: https://www.theverge.com/2017/5/17/15654988/google-jet-brains-kotlin-programming-language-android-development-io-2017

[22] Kotlin vs Java: Which is the Best Choice for Android App Development? [online]. Jhonson, 2017 [cit. 2022-04-02]. Dostupné z: https://medium.com/javarevisited/kotlin-vs-java-which-is-the-best-choice-for-android-app-development-7c9fc782d2c9

[23] WHY USE REACT NATIVE FOR MOBILE APP DEVELOPMENT [online]. nix, 2021 [cit. 2022-04-02]. Dostupné z: https://nix-united.com/blog/why-use-react-native-for-mobile-app-development/

[24] The Good and the Bad of Flutter App Development [online]. altexsoft, 2021 [cit. 2022-04-02]. Dostupné z: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-flutter-app-development/

[25] What Is an IDE? How It Helps Developers Code Faster [online]. Walker, 2021 [cit. 2022-04-02]. Dostupné z: https://www.g2.com/articles/ide

[26] Meet Android Studio [online]. Google, 2022 [cit. 2022-04-02]. Dostupné z: https://developer.android.com/studio/intro?utm_source=android-studio

[27] About an Xcode IDE for Apple application development [online]. Narayanan, 2018 [cit. 2022-04-02]. Dostupné z: https://medium.com/worst-ios-developer/about-an-xcode-ide-for-apple-application-development-68b161088e53

[28] Testing Apps on a Simulator vs. Emulator vs. Real Device [online]. Kinsbruner, 2020 [cit. 2022-04-02]. Dostupné z: https://www.perfecto.io/blog/simulator-vs-emulator

[29] What's New in Xcode 13? [online]. Dheeru N, 2020 [cit. 2022-04-02]. Dostupné z: https://betterprogramming.pub/whats-new-in-xcode-13-a48d5158f98

[30] Version Control Systems [online]. GeeksForGeeks, 2020 [cit. 2022-04-02]. Dostupné z: https://www.geeksforgeeks.org/version-control-systems/

[31] Authentication [online]. Beal, 2022 [cit. 2022-04-02]. Dostupné z: https://www.webopedia.com/definitions/authentication/

[32] What is a Database? Definition, Meaning, Types with Example [online]. Beal, 2022 [cit. 2022-04-02]. Dostupné z: https://www.guru99.com/introduction-to-database-sql.html

[33] What is Android? Here's everything you need to know [online]. Brown, 2022 [cit. 2022-04-02]. Dostupné z: https://www.androidauthority.com/what-is-android-328076/

[34] Best Databases For Mobile Apps 2022 – Choosing the Best One [online]. Patel, 2021 [cit. 2022-04-02]. Dostupné z: https://www.emizentech.com/blog/best-databases-for-mobile-apps.html

[35] Platform Architecture [online]. Google, 2021 [cit. 2022-04-02]. Dostupné z: https://developer.android.com/guide/platform

[36] LACKO, Ľuboslav. Mistrovství Android. Brno: Computer Press, 2017, 647 s. ISBN 9788025148754.

[37] Jetpack Compose Concepts Every Developer Should Know [online]. Dziedzic, 2021 [cit. 2022-04-02]. Dostupné z: https://medium.com/captech-corner/jetpack-compose-concepts-every-developer-should-know-5bcb47914542

[38] SwiftUI [online]. Sundell, 2022 [cit. 2022-04-02]. Dostupné z: https://www.swiftbysundell.com/discover/swiftui/

[39] Should You Use AWS or Firebase For the Back end of Your Mobile App?: Which is the right solution for your requirements? [online]. Trivedi, 2020 [cit. 2022-04-02]. Dostupné z: https://betterprogramming.pub/should-you-use-aws-or-firebase-for-the-back-end-of-your-mobile-app-a28f93540520

[40] SMYTH, Neil. Android Studio 4.2 Development Essentials – Kotlin Edition. Payload Media, 2021, 647 s. ISBN 978-1-951442-30-9.

ZOZNAM POUŽITÝCH SYMBOLOV A ZKRATIEK

HAL Hardware Abstraction Layer

ART Android Runtime

DEX Dalvik Executable format

AOT ahead-of-time

JIT just-in-time

NFC Near Field Communication

JRE Java Runtime Environment

JVM Java Virtual Machine

NPE NullPointerException

AVD Android Virtual Device

SDK Software Development Kit

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

Príloha P I: CD obsahujúce bakalársku prácu a zdrojové kódy tvorenej aplikácie