Multiplatformní mobilní aplikace pro organizaci akcí

Bc. Lukáš Pevný

Diplomová práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

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

Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Lukáš Pevný**
Osobní číslo:	**A21164**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Multiplatformní mobilní aplikace pro organizaci akcí**
Téma práce anglicky:	**A Multi-platform Mobile Application for Organizing Events**

Zásady pro vypracování

1. Popište možnosti vývoje multiplatformních mobilních aplikací.
2. Shrňte základní strukturu frameworku Flutter, který bude využit pro implementaci praktické aplikace. Zaměřte se i na výhody a nevýhody.
3. Proveďte průzkum trhu s aplikacemi podobného typu a stručně je popište.
4. V praktické části práce stanovte funkcionální a nefunkcionální požadavky aplikace, shrňte služby použité na back-endu a věnujte se implementaci aplikace dle uvedených požadavků.
5. Implementovaný prototyp aplikace otestujte na reálném zařízení a popište zabezpečení aplikace.

Forma zpracování diplomové práce: **tištěná/elektronická**
Jazyk zpracování: **Slovenština**

Seznam doporučené literatury:

1. WINDMILL, Eric a Ray RISCHPATER. Flutter in action. Shelter Island, NY: Manning Publications Co., [2020]. ISBN 978--161-7296-147.
2. MIOLA, Alberto. Flutter Complete Reference: Create beautiful, fast and native apps for any device. Independently published, [2020]. ISBN 978-0141044804.
3. Flutter documentation [online]. [cit. 2022-11-29]. Dostupné z: https://docs.flutter.dev/
4. Cross platform mobile development [online]. [cit. 2022-11-29]. Dostupné z: https://kotlinlang.org/docs/cross-platform-mobile-development.html
5. BLoC state management library [online]. [cit. 2022-11-29]. Dostupné z: https://bloclibrary.dev/
6. Android developers [online]. [cit. 2022-11-29]. Dostupné z: https://developer.android.com/
7. Apple Developer Documentation [online]. [cit. 2022-11-29]. Dostupné z: https://developer.apple.com/documentation/
8. AWS Documentation [online]. [cit. 2022-11-29]. Dostupné z: https://docs.aws.amazon.com/

Vedoucí diplomové práce: **Ing. Radek Vala, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **2. prosince 2022**
Termín odevzdání diplomové práce: **26. května 2023**

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

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

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

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové 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 diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou 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 – diplomovou 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 diplomové 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 diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové 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 diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 15. 05. 2023

Lukáš Pevný, v. r.
podpis studenta

ABSTRAKT

Diplomová práca sa zameriava na vývoj multiplatformovej mobilnej aplikácie, pre organizovanie rôznych akcií v rámci uzavretej skupiny ľudí. Aplikácia poskytuje prehľadný a jednoduchý spôsob zobrazovania dôležitých informácií o jednotlivých akciách, vrátane možnosti ich tvorba a spravovania. Poskytne taktiež prostredie pre komunikáciu s ostatnými účastníkmi akcie prostredníctvom chatu. Ďalej poskytne archív pre uchovanie fotiek a videí zhotovených na akciách. V rámci práce bude spracovaný taktiež prieskum trhu a popis konkurenčných aplikácií a ďalej teoretický prehľad vývojových nástrojov pre vývoj multiplatformových aplikácií, ako je použitý framework Flutter.

Kľúčové slová:

multiplatformová aplikácia, akcie, organizácia, Flutter, AWS

ABSTRACT

The master's thesis focuses on the development of multiplatform mobile application for organizing various events within a close group of people. The application provides clear and simple way of displaying important information about individual events, including the possibility of creating and managing them. It will also provide and environment for communicating with other event participants via chat. Furthermore, the archive will provide a way to store photos and videos made on the events. The thesis will also include a market survey and description of competing applications, as well as theoretical overview of development tools for the development of multiplatform applications, such as the Flutter framework used.

Keywords:

multiplatform application, events, organization, Flutter, AWS

Rád by som sa poďakoval Ing. Radku Valovi, Ph.D. za jeho trpezlivosť, ochotu a vedenie počas celého procesu tvorby diplomovej práce. Taktiež by som sa rád poďakoval doc. Ing. Radku Šilhavému, Ph.D. za jeho čas a rady pri modelovaní prípadov užitia.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Organizovanie akcií v určitej skupine ľudí môže byť niekedy tvrdý oriešok. Mobilná aplikácia Akcošky Vám zabezpečí jednoduché organizovanie akcií, kde budete mať všetky potrebné informácie na jednom mieste. Vyhnete sa tak ignorancii či nezaujatiu Vašich priateľov a plány premeníte na skutočnosť.

Akcošky je mobilná aplikácia podporovaná operačným systémom Android a iOS, ktorou hlavnou úlohou je primárne – ako už z názvu vyplýva, organizovanie akcií. Pri vytváraní akcie je potrebné zadať základné informácie ohľadom názvu, typu akcie, mieste konania, dátumu či času. Užívateľ si priamo vyberá, pre ktorú skupinu je daná udalosť organizovaná. Všetkým pozvaným užívateľom následne príde notifikácia ohľadom vytvorenia tejto udalosti a v aplikácií následne vidia túto akciu a vidia všetky dôležité informácie o tejto akcii.

Taktiež vidia, kto všetko sa plánuje akcie zúčastniť, čo môže ovplyvniť ich konečné rozhodnutie, či zúčastnia alebo nie. V prípade, že sa aj tak akcia nepodarí zorganizovať je ju možné vymazať z databázy. Opačný prípad, úspešné zorganizovanie akcie, umožňuje akciu uschovať do archívu a všetky informácie, chat a galéria budú všetkým užívateľom prístupné aj po skončení.

Ako doplnok aplikácia obsahuje konverzáciu v podobe chatu pre dohadovanie a dolaďovanie detailov a podrobností, ako sa udalosť zorganizuje. Okrem toho obsahuje galériu, kde si užívatelia môžu uchovať zážitky a spomienky formou fotiek a videí. To môžu robiť ku každej akcii osobitne čo zabezpečuje prehľadnosť a poriadok.

Očakávaný výsledok tejto aplikácie je zlepšovanie organizovania a nárast počtu akcií, ktoré mali síce nápad, ale ich realizácia sa nepodarila.

I. TEORETICKÁ ČÁST

1 VÝVOJ MULTIPLATFORMOVÝCH MOBILNÝCH APLIKÁCIÍ

Vývoj multiplatformových mobilných aplikácií je prístup, ktorý nám umožňuje vytvoriť mobilnú aplikáciu, ktorá bude fungovať bez problémov na viacerých operačných systémoch. V multiplatformových mobilných aplikáciách, časti alebo aj celý zdrojový kód môže byť zdieľaný. Toto umožňuje vývojárom vytvárať aplikácie, ktoré budú fungovať na operačných systémoch Android a iOS bez nutnosti opätovného programovanie pre každú platformu zvlášť. [1]

1.1 Výhody

V tejto časti sú uvedené výhody multiplatformového vývoja mobilných aplikácií.

1.1.1 Znovu použiteľnosť kódu

Pri tomto prístupe nemusíme písať zvlášť kód pre každú platformu. Použitie jedného zdieľaného zdrojového kódu umožňuje skrátiť čas strávený vykonávaním opakujúcich sa úloh, ako API volania, ukladanie údajov, serializácia údajov a implementácia analýz. [1][2]

1.1.2 Šetrenie času

Vďaka znovu použiteľnosti kódu vývoj vyžaduje menej riadkov kódu. Čiže čas šetríme tým, že nemusíme písať toľko kódu. S menej riadkami kódu je menej miest, kde sa môžu objaviť chyby, čo má za následok menej času testovaním a údržbou kódu. [1][2]

1.1.3 Efektívne riadenie zdrojov

Vývoj separátnych aplikácií je drahý. Vďaka tomu, že máme jeden zdieľaný kód môžeme lepšie spravovať naše zdroje. Vývojové tímy pre Android aj iOS sa môžu naučiť písať a používať zdieľaný kód. [1][3]

1.1.4 Atraktívne príležitosti pre vývojárov

Mnohí vývojári mobilných aplikácií vnímajú moderné multiplatformové technológie, ako žiaduce prvky v technologickom zásobníku[1]. [1]

[1] Po anglicky tech stack, je to súbor technológií poskladaných dokopy na vytvorenie požadovanej aplikácie https://www.mongodb.com/basics/technology-stack

1.1.5 Príležitosť osloviť širšie publikum

Nemusíme si vyberať medzi rôznymi platformami. Aplikácia je kompatibilná s viacerými operačnými systémami, môžeme uspokojiť publikum so systémom Android aj iOS a maximalizovať svoj dosah. [1]

1.1.6 Rýchlejšie uvedenie na trh a prispôsobenie

Vzhľadom na to, že nemusíme vytvárať rôzne aplikácie pre rôzne operačné systémy, môžeme aplikáciu rýchlejšie vytvoriť a uviesť na trh. Ak bude treba aplikáciu nejako prispôsobiť alebo transformovať, bude jednoduchšie vykonať malé zmeny v konkrétnych častiach kódu. To nám taktiež umožní lepšie reagovať na spätnú väzbu používateľov. [3]

1.2 Nevýhody/Výzvy

V tejto časti sa nachádzajú nevýhody a výzvy s ktorými musia programátori počítať pri výbere programovania mobilných aplikácií multiplatformovo.

1.2.1 Nižší výkon

Výkon je jednou z najdôležitejších charakteristík aplikácie. Je to závislé na mnohých faktoroch, ale keď porovnáme aplikáciu, ktorá je natívna[2] s aplikáciou, ktorá je multiplatformová a oboje majú rovnaké funkcie tak natívna bude o niečo rýchlejšia. Každopádne tieto rozdiely vo výkone bývajú väčšinou malé, hlavne v prípade jednoduchých aplikácií. [4]

1.2.2 Slabé UI/UX

Nanešťastie, natívne aplikácie sú o animačných funkciách, 3D efektoch a nádhernej kombinácii grafiky vylepšenej hardvérovými funkciami. Väčšina multiplatformových aplikácií nemôže využiť všetky funkcie poskytované mobilnými zariadeniami čo vyústi v slabom UX. [4][5]

[2] Aplikácia, ktorá je vyvíjaná pre konkrétny operačný systém

1.2.3 Oneskorený prístup k najnovším funkciám

Vývojári multiplatformových aplikácií sú závislí na frameworku[3], ktorý si vybrali. A problém s tým je ten, že aktualizovať framework o nové funkcie bude nejakú chvíľu trvať. Zatiaľ čo Apple a Google pridávajú nové funkcie do iOSu a Androidu, frameworky musia adaptovať ich vývojárske nástroje a integrovať nové funkcie. [5]

1.2.4 Slabé prispôsobenie a integrácia navigácie

Frameworky nemusia podporovať všetky funkcie, ktoré by boli potrebné, predovšetkým tie čo sú viazané s hardwarom, lokálnymi nastaveniami zariadenia alebo zabudovaným prístupom k úložisku. [4][5]

1.2.5 Strata kódu pri prechode na nový framework

Vzhľadom na to, že každý framework môže používať iný jazyk a môže byť rôzne prispôsobený, prechod na nový framework môže znamenať neschopnosť použitia kódu z iného frameworku. [4]

[3] Sada vývojárskych nástrojov, ktorá pomáha a určuje, ako má programátor aplikáciu vyvíjať https://www.damidev.com/slovnik/framework

2 FRAMEWORKY PRE VÝVOJ MULTIPLATFORMOVÝCH MOBILNÝCH APLIKÁCIÍ

V tejto kapitole sa nachádza prehľad najznámejších frameworkov pre programovanie multiplatformových mobilných aplikácií.

2.1 Flutter

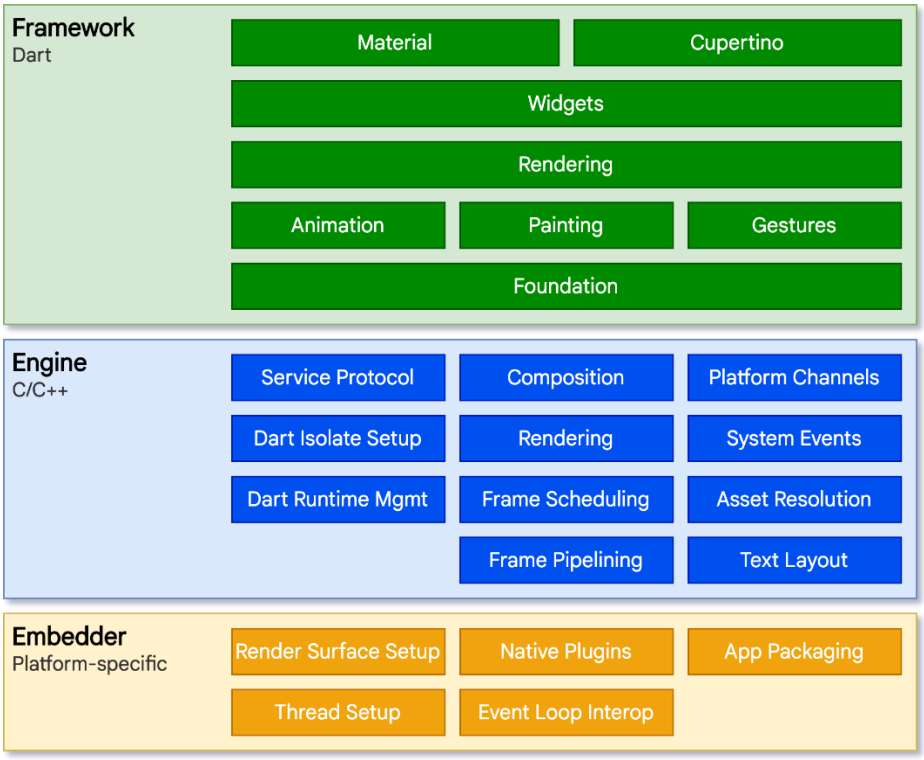
Flutter je prenosná sada nástrojov UI od spoločnosti Google na vytváranie natívne skompilovaných aplikácií pre mobilné zariadenia, web a počítače z jedného zdrojového kódu. Flutter funguje s existujúcim kódom, používajú ho vývojári a organizácie na celom svete a je bezplatný a funguje pod licenciou open source[4]. [6]

V porovnaní s ostatnými frameworkami, ktoré sa zaoberajú vývojom multiplatformových mobilných aplikácií je Flutter jedinečný v tom, že sa nespolieha na technológie webového prehliadača ani na súbor widgetov dodávaných s každým zariadením. Namiesto toho Flutter používa na vykresľovanie widgetov vlastné výkonné vykresľovacie jadro. Flutter sa navyše líši tým, že obsahuje len tenkú vrstvu C/C++ kódu. Flutter implementuje väčšinu svojho systému v jazyku Dart, ku ktorému môžu vývojári ľahko pristupovať, čítať, meniť, nahrádzať alebo odstraňovať. To dáva vývojárom obrovskú kontrolu nad systémom a taktiež znižuje latku prístupnosti pre väčšinu systému. [6]

2.1.1 Prehľad architektúry

Flutter je navrhnutý ako rozšíriteľný, viacvrstvový systém. Existuje ako súbor nezávislých knižníc, z ktorých každá knižnica závisí od základnej vrstvy. Žiadna vrstva nemá privilegovaný prístup k vrstve pod ňou a každá časť je navrhnutá tak aby bola voliteľná a nahraditeľná. [7]

[4] Licencia na voľné používanie, upravovanie a zdieľanie https://opensource.org/licenses



Obrázok 1 Flutter – vrstvy [7]

Pre operačný systém sú Flutter aplikácie zabalené rovnakým spôsobom ako všetky ostatné natívne aplikácie. Vkladací modul špecifický pre platformu poskytne vstupný bod, koordinuje s operačným systémom pre prístup k službám ako vykresľovanie povrchov, prístupnosť a vstup a riadi slučku udalostí. [7]

V jadre Flutteru je Flutter engine[5], ktorý je prevažne napísaný v jazyku C++ a podporuje základy potrebné na podporu všetkých Flutter aplikácií. Tento engine je zodpovedný za rastrovanie scén, vždy keď sa má vykresliť nový snímok. Poskytuje nízkoúrovňovú implementáciu základného API Flutteru vrátane grafiky (prostredníctvom Skia[6]), rozloženia textu, vstupy a výstupy súborov a siete, architektúru zásuvných modulov, Dart runtime a nástroje na kompiláciu[7]. [7]

[5] Jadro logiky počítačového programu https://en.wiktionary.org/wiki/software_engine
[6] Open source 2D grafická knižnica https://skia.org/
[7] Prevod zdrojového kódu do binárneho kódu https://www.techtarget.com/whatis/definition/compiler

Vývojári typicky pracujú s Flutterom prostredníctvom Flutter frameworku. Ten obsahuje bohatú sadu platformových a základových knižníc, ktoré sa skladajú z viacerých vrstviev. [7]

Zdola nahor máme:

- Základové triedy a základné bloky ako animácie, maľovanie a gestá
- Vykresľovacia vrstva poskytuje abstrakciu pri riešení layoutu
- Widget vrstva je abstrakciou zloženia. Každý vykresľujúci objekt vo vykresľovacej vrstve má korešpondujúcu triedu vo widget vrstve
- Material a Cupertino knižnice, ktoré poskytujú komplexné sady ovládacích prvkov, ktoré používajú primitívy widget vrstvy na implementovanie Material alebo iOS dizajnových jazykov

[7]

Flutter framework je relatívne malý, veľa funkcií vyššej úrovne bývajú implementované ako balíky, ktoré vychádzajú zo základných knižníc Dart a Flutter. [7]

2.1.2 Reaktívne používateľské rozhrania

Flutter je reaktívny, pseudodeklaratívny UI framework, v ktorom vývojár poskytuje mapovanie zo stavu aplikácie na stav rozhrania a framework preberá úlohu aktualizovať UI za behu, keď sa stav aplikácie zmení. Tento model je inšpirovaný prácou, ktorá prišla z Facebooku pre ich vlastný framework React, ktorý zahŕňa prehodnotenie mnohých tradičných prístupov dizajnu. [7]

Flutter ako aj mnohé ostatné reaktívne frameworky používa prístup, pri ktorom odstraňuje väzbu medzi UI a jeho základným stavom. Pomocou API React štýlu, vývojári vytvárajú popis UI a framework preberá zodpovednosť pre použitie tejto konfigurácie na vytvorenie a/alebo aktualizovanie UI v prípade potreby. [7]

Vo Flutteri widgety (podobne ako pri komponentoch v Reacte) sú reprezentované nemennými triedami, ktoré sú používané na konfiguráciu stromov objektov. Tieto widgety sú používané v separátnom strome objektov pre layout, ktorý je potom použitý na správu samostatného stromu objektov pre kompozíciu. [7]

Flutter je vo svojej podstate rad mechanizmov na efektívne prechádzanie zmenených častí stromov, prevod stromov objektov na stromy objektov nižšej úrovne a šírenie zmien v týchto stromoch. [7]

Widget deklaruje jeho UI preťažením *build()* metódy, čo je funkcia, ktorá konvertuje stav na UI. [7]

Obrázok 2 znázorňuje túto rovnicu.

Obrázok 2 Rovnica UI [8]

Táto *build()* metóda sa podľa návrhu vykonáva rýchlo a nemala by mať vedľajšie účinky aby bolo možné ju zavolať kedykoľvek keď je to potrebné. [7]

2.1.3 Widgety

Widgety sú stavebnými prvkami používateľského rozhrania aplikácie vytvorenej vo Flutteri a každý widget je nemenná časť používateľského rozhrania. Widgety tvoria hierarchiu založenú na kompozícií. Každý widget je vnorený do rodiča a môže získať kontext z rodiča. Táto štruktúra sa prenáša až ku koreňovému widgetu (kontajner, ktorý je hostiteľom aplikácie, zvyčajne MaterialApp alebo CupertinoApp). [7]

```
void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text('My Home Page'),
        ),
        body: Center(
          child: Builder(
            builder: (context) {
              return Column(
                children: [
                  const Text('Hello World'),
                  const SizedBox(height: 20),
                  ElevatedButton(
                    onPressed: () {
                      print('Click!');
                    },
                    child: const Text('A button'),
                  ),
                ],
              );
            },
          ),
        ),
      ),
    );
  }
}
```

Obrázok 3 Ukážka použitia widgetov [7]

Aplikácia aktualizuje používateľské rozhranie na základe udalosti (napr. užívateľská interakcia) tak, že povedia frameworku aby v hierarchií nahradil widget za iný. Framework porovná nový widget so starým a efektívne aktualizuje používateľské rozhranie. [7]

Flutter má vlastné implementácie každého ovládacieho prvku užívateľského rozhrania, namiesto toho aby sa spoliehal na tie čo ponúka systém. [7]

Tento prístup poskytuje niekoľko výhod:

- poskytuje neobmedzenú rozšíriteľnosť
- zabraňuje výraznému obmedzeniu výkonu tým, že Flutter nemusí prechádzať medzi kódom aplikácie a kódom platformy

[7]

Widgety sa zvyčajne skladajú z mnohých ďalších malých jednoúčelových widgetov, ktoré spolu vytvárajú silné efekty. [7]

2.1.3.1 Manažovanie stavov

Vo Flutteri existujú dva druhy widgetov: stateless a stateful. [7]

Widgety, ktoré nemajú žiadny stav ktorý treba meniť postupom času sa nazývajú stateless. [7]

Widgety, ktoré potrebujú aby sa menili na základe užívateľskej interakcie alebo ostatných faktoroch sa nazývajú stateful. Stateful widgety nemajú *build()* metódu, namiesto toho sú užívateľské rozhrania sú stavané cez *State* objekt. Keď zmeníme State objekt, musíme zavolať *setState()* aby sme dali signál frameworku aby aktualizoval užívateľské rozhranie zavolaním *State* build metódy. [7]

2.1.4 Vykresľovanie a layout

Táto sekcia popíše vykresľovaciu pipeline, čo je séria krokov, ktorá vykonáva premenu hierarchie widgetov na pixely vykreslené na obrazovke.

2.1.4.1 Vykresľovací model Flutteru

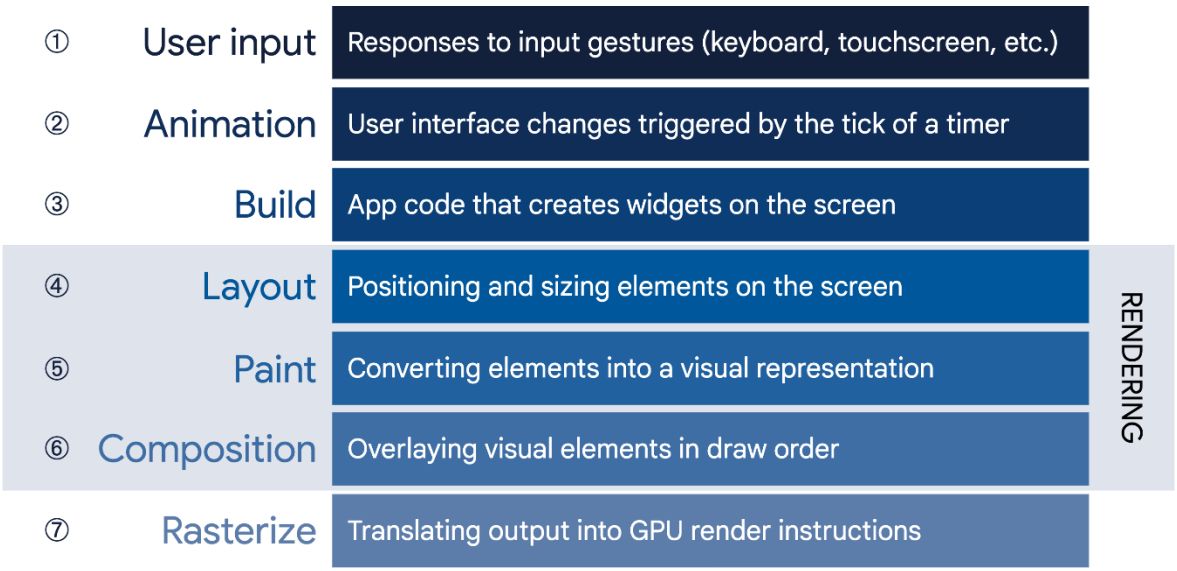
Multiplatformové mobilné frameworky typicky vytvárajú abstrakčnú vrstvu nad základnými natívnymi knižnicami UI pre Android a iOS, pričom sa snaží vyrovnať s nekonzistentnosťou jednotlivých platforiem. Natívne knižnice sú zodpovedné za kreslenie do Canvas objektu, ktorý potom môže Android vykresliť pomocou enginu Skia. [7]

Flutter minimalizuje tieto abstrakcie tak, že obchádza systémové knižnice pre UI v prospech vlastných widgetov. Dart zdrojový kód, ktorý vykresľuje vizuálne prvky Flutteru, je skompilovaný do natívneho kódu, ktorý na vykresľovanie používa grafický engine Skia. Flutter engine taktiež obsahuje vlastnú kópiu enginu Skia, čo umožňuje vývojárom aktualizovať svoju aplikáciu aby bola stále aktualizovaná aj keď telefón nebol aktualizovaný na najnovšiu

Android verziu. To platí aj pre ďalšie natívne platformy, ktorá Flutter podporuje - iOS, Windows a macOS. [7]

2.1.4.2 Fázy vykresľovania

Hlavnou zásadou Flutteru pri vykresľovaní je - jednoduché veci sú rýchle. Flutter má priamočiaru pipeline pre tok dát do systému ako je znázornené na Obrázok 4. [7]



Obrázok 4 Fázy vykresľovania [7]

Ďalej si podrobnejšie rozoberieme niektoré dôležité fázy.

Build: od Widgetu po Element

Považujme tento fragment kódu za demonštráciu hierarchie widgetov. [7]

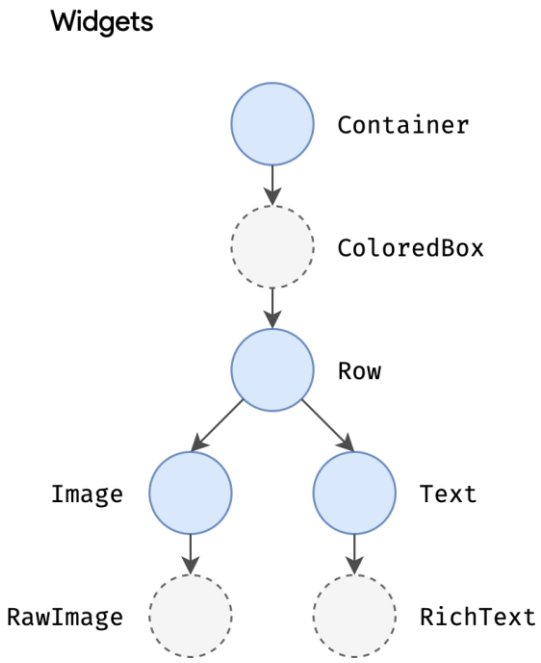
```
Container(
  color: Colors.blue,
  child: Row(
    children: [
      Image.network('https://www.example.com/1.png'),
      const Text('A'),
    ],
  ),
);
```

Obrázok 5 Ukážka hierarchie widgetov [7]

Keď Flutter potrebuje vykresliť tento fragment, zavolá *build()* metódu, ktorá vráti podstrom widgetov, ktorý vykresľuje UI na základe aktuálneho stavu aplikácie. Počas tohto procesu

build() metóda môže zaviesť ďalšie nové widgety ako potrebuje na základe ich stavu. Napríklad, v predošlom fragmente, Container má vlastnosti *color* a *child*. Container, v prípade, že *color* nie je null, vkladá widget ColoredBox, ktorý reprezentuje farbu. Takisto aj widgety Image a Text môžu vkladať nové widgety ako *RawImage* a *RichText* počas fázy zostavovania. Hierarchia widgetov potom môže byť hlbšia ako kód, ktorý ju reprezentuje. [7]

Obrázok 6 znázorňuje hierarchiu widgetov.

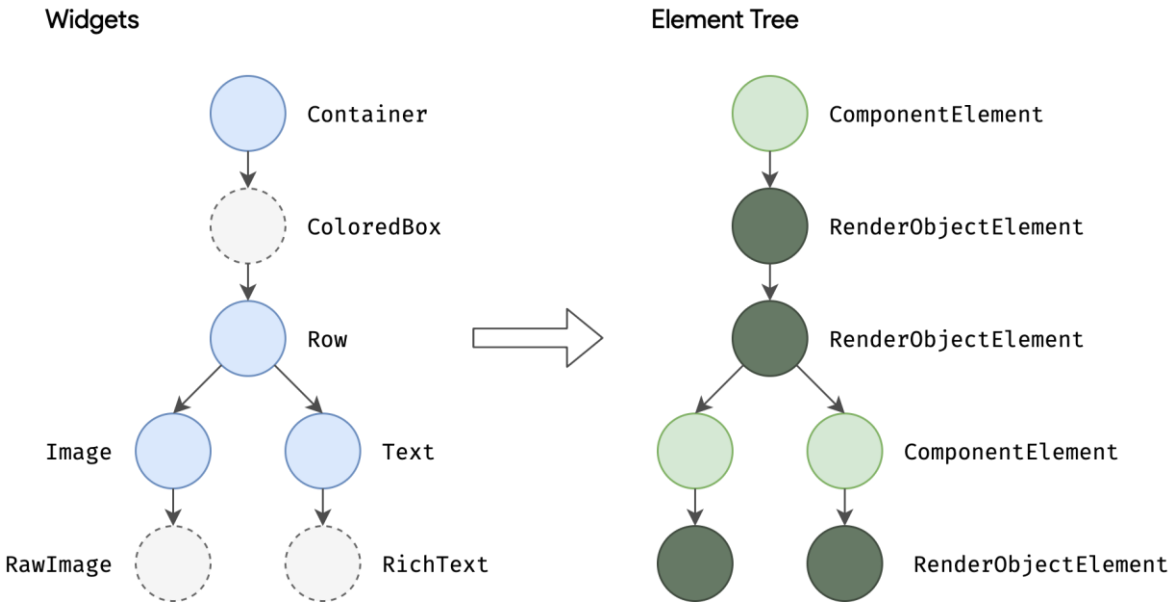


Obrázok 6 Hlbší strom widgetov [7]

Počas fázy zostavovania, Flutter prekladá widgety vyjadrené v kóde do korešpondujúceho stromu elementov kde jeden element je rovný jednému widgetu. Každý element reprezentuje špecifickú instanciu widgetu na danom mieste v stromovej hierarchii. [7]

Dva základné typy sú:

- *ComponentElement* – hostiteľ pre ostatné prvky
- *RenderObjectElement* – element, ktorý sa podieľa na fázach layoutu alebo vykresľovania



Obrázok 7 Zobrazenie vzťahu stromu elementov k stromu widgetov [7]

Vzhľadom na to, že widgety sú nemenné, vrátane vzťahu rodič/dieťa medzi uzlami, akákoľvek zmena v strome widgetov spôsobí vrátenie novej sady objektov widgetov. To ale neznamená, že základná reprezentácia musí byť prestavaná. Strom elementov je perzistentný od rámca k rámcu a zohráva rozhodujúcu úlohu z pohľadu výkonu, umožňuje Flutteru správať sa akoby je hierarchia widgetov jednorazová, a zároveň cachovať[8] jej základnú reprezentáciu. Tým, že Flutter prechádza len widgety ktoré sa zmenili môže znovu vystavať len tie časti, ktoré si vyžadujú rekonfiguráciu. [7]

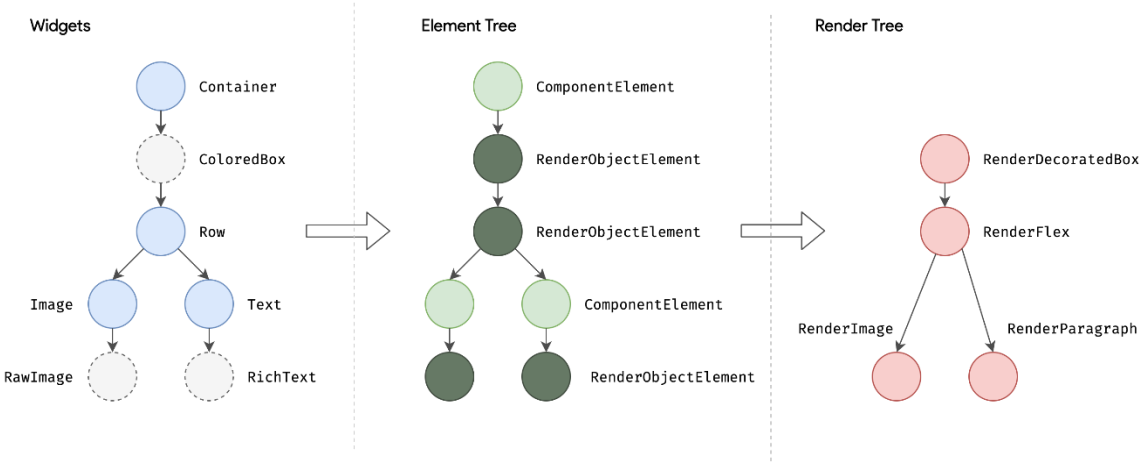
Rozloženie a vykresľovanie

Dôležitou časťou každého UI frameworku je schopnosť efektívne rozvrhnúť hierarchiu widgetov, určiť veľkosť a polohu každého prvku ešte pred ich vykreslením na obrazovku. [7]

Základnou triedou každého uzla vo vykresľovacom strome je *RenderObject*, ktorý nám definuje abstraktný model pre layout a vykreslenie. Tento model je všeobecný, neviaže sa na pevný počet dimenzií dokonca ani na karteziánsky súradnicový systém. Každý *RenderObject* pozná svojho rodiča, ale vie veľmi málo o potomkoch okrem toho ako k nim pristúpiť a ako sú obmedzené. Vďaka tomuto má *RenderObject* dostatočnú abstrakciu, aby bol schopný zvládnuť rôzne prípady užitia. Počas fázy zostavovania, Flutter vytvára alebo

[8] Ukladanie údajov do rýchlej vyrovnávacej pamäte z dôvodu, aby budúce prístupy boli rýchlejšie https://aws.amazon.com/caching/

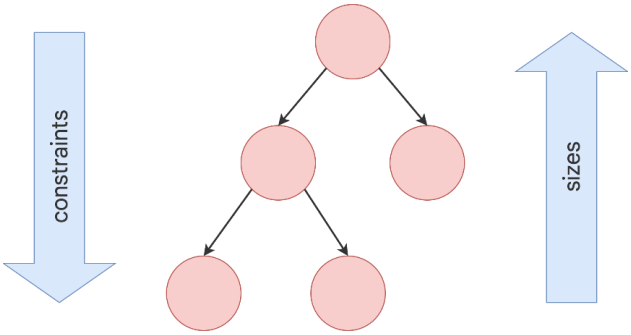
aktualizuje objekty, ktoré dedia od *RenderObject*, pre každý *RenderObjectElement* v strome elementov. *RenderObject*s sú veľmi jednoduché napr. *RenderParagraph* vykresľuje text, *RenderImage* vykresľuje obrázok a *RenderTransform* aplikuje transformáciu pred vykreslením potomka. [7]



Obrázok 8 Vzťah medzi stromom widgetov, stromom elementov a vykresľovacím stromom [7]

Väčšina Flutter widgetov je vykresľovaná objektom, ktorý dedí z podtriedy *RenderBox*, ktorá reprezentuje *RenderObject* pevnej veľkosti v 2D karteziánskom priestore. *RenderBox* poskytuje základ modelu obmedzenie boxu, ktorý stanovuje minimálnu a maximálnu šírku a výšku pre každý widget, ktorý má vykresliť. [7]

Na rozloženie widgetov Flutter pri prechádzaní vykresľovacieho stromu odovzdáva obmedzenia veľkosti z rodiča na potomka. Pri určovaní svojej veľkosti musí potomok rešpektovať obmedzenia, ktoré mu určil jeho rodič. Potomkovia reagujú odovzdaním veľkosti rodičovi v medziach, ktoré mu stanovil rodič. [7]



Obrázok 9 Odovzdanie obmedzení (constraints) a veľkosti (sizes) pri prechádzaní vykresľovacím stromom [7]

Na konci jediného prechodu stromov má každý objekt definovanú veľkosť v rámci obmedzení stanovených rodičom a je pripravený na vykreslenie metódou *paint()*. [7]

Tento model obmedzenia boxu je veľmi výkonný ako spôsob rozloženia objektov v čase O(n):

- Rodič môže diktovať veľkosť potomka nastavením maximálneho a minimálneho obmedzenia na rovnakú hodnotu. Potomkovia si môžu vybrať ako využiť toto miesto. Napríklad môžu len vycentrovať to, čo chcú vykresliť v obmedzeniach rodiča.
- Rodič môže určiť šírku potomka, ale poskytnúť mu flexibilitu, pokiaľ ide o výšku alebo určiť výšku a poskytnúť flexibilitu pri šírke. Príklad je flexibilný text, ktorý sa môže prispôsobiť horizontálnemu obmedzeniu, ale môže sa meniť vertikálne v závislosti od množstva textu.

[7]

Tento model funguje aj v prípade, že potomok potrebuje vedieť koľko miesta má k dispozícii aby sa rozhodol ako vykreslí svoj obsah. [7]

Rodičom všetkých *RenderObject*s je *RenderView*, ktorý reprezentuje celkový výstup z vykresľovacieho stromu. V prípade, že je potrebné vykresliť nový snímok zavolá sa metóda *compositeFrame()*, ktorá je súčasťou *RenderView*. Toto vytvorí *SceneBuilder* aby sa spustila aktualizácia scény. Keď je scéna aktualizovaná, *RenderView* zloženú scénu odovzdá metóde *Windows.render()*, ktorá odovzdá riadenie grafickému procesoru aby ju vykreslil. [7]

2.1.5 Zakomponovanie do platformy

Ako bolo spomenuté, používateľské rozhrania vo Flutteri nie sú preložené do ekvivalentných widgetov operačného systému, ale sú vytvorené, rozložené, zložené a vykreslené samotným Flutterom. Mechanizmus získavania textúr a zúčastňovanie sa v cykle aplikácie základného operačného systému sa líši v závislosti od jedinečných problémov danej platformy. Engine je platformo agnostický a predstavuje stabilné rozhranie ABI, ktoré poskytuje „platform embedder“ so spôsobom ako nastaviť a používať Flutter. [7]

Spomínaný „platform embedder“ je natívnou aplikáciou daného operačného systému ktorá hostí všetok Flutter obsah a funguje ako spojovací článok medzi Flutterom a operačným systémom hosta. Pri štarte Flutter aplikácie, embedder poskytuje vstupný bod, inicializuje Flutter engine, získava vlákna pre používateľské rozhranie a vykresľovanie a vytvorí textúru, do ktorej môže Flutter zapisovať. Embedder je taktiež zodpovedný za životný cyklus

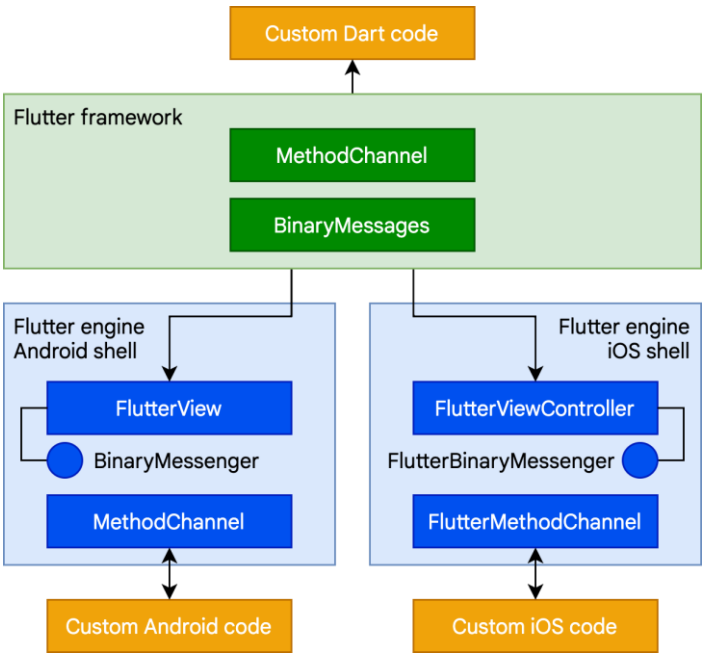
aplikácie vrátane vstupných gest, veľkosti okna, správy vlákien a správ platformy. Flutter obsahuje platform embedder pre Android, iOS, Windows, macOS a Linux, taktiež si môžeme vytvoriť vlastný embedder. [7]

2.1.6 Integrácia s iným kódom

Flutter umožňuje spoluprácu pomocou niekoľkých mechanizmov, či už pristupujeme kódu alebo API napísaným v jazykoch Kotlin alebo Swift, voláme natívne knižnice založené na jazyku C, vkladáme natívne ovládacie prvky do aplikácie v Flutteri alebo vkladáme Flutter do existujúcej aplikácie. [7]

2.1.6.1 Kanály platformy

Pre mobilné a počítačové aplikácie, Flutter dovoľuje volanie vlastného kódu prostredníctvom kanála platformy, čo je mechanizmus na komunikáciu medzi Dart kódom a kódom aplikácie, ktorý je špecifický pre danú platformu. Dáta sú serializované z Dart typu *Map* a následne deserializované do korešpondujúceho typu, v Kotline *HashMap* a vo Swifte *Dictionary*. [7]



Obrázok 10 Kanály platformy [7]

2.1.6.2 Rozhranie cudzích funkcií

Pre API rozhrania založené na jazyku C poskytuje Dart mechanizmus na prepojenie s natívnym kódom pomocou knižnice *dart:ffi*. Rozhranie cudzích funkcií (foreign function

interface - FFI) môže byť podstatne rýchlejšie ako kanály platformy pretože nie je potrebná žiadna serializácia na prenos údajov. Namiesto toho poskytuje Dart runtime možnosť alokovať pamäť na halde, ktorá je podporovaná Dart objektom, a uskutočňovať volania staticky alebo dynamicky linkovaných knižníc. [7]

2.1.6.3 Vykresľovanie natívnych ovládacích prvkov

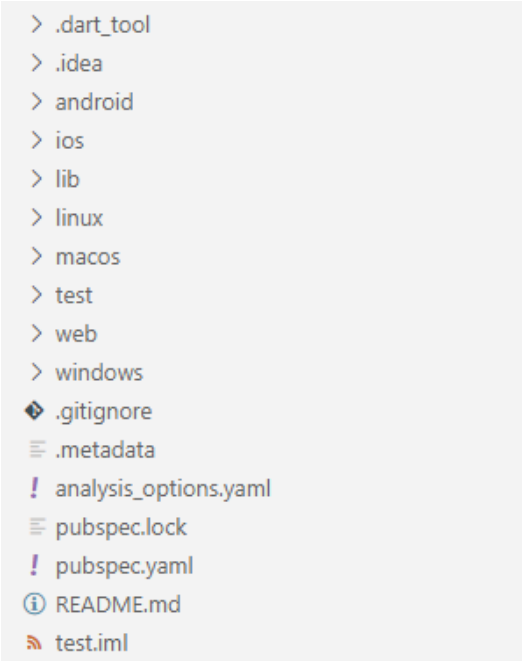
Vzhľadom na to, že Flutter vykresľuje textúry a strom widgetov interne, neexistuje miesto pre niečo ako Android view aby existoval v internom modeli. Flutter to rieši zavedením zobrazovacích widgetov zobrazenia platformy (*AndroidView* a *UiKitView*). Tieto widgety môžu byť integrované s limitami s ostatným Flutter obsahom. [7]

2.1.6.4 Hosťovanie Flutter obsahu v rodičovskej aplikácii

V prípade, že chceme vložiť Flutter widget do existujúcej aplikácie, tak novovytvorená Flutter aplikácia je vložená do Android activity alebo iOS UIViewController. [7]

2.1.7 Štruktúra Flutter projektu

Pri vytvorení nového Flutter projektu, štruktúra vyzerá ako na Obrázok 11.



Obrázok 11 Štruktúra nového projektu

.dart_tool

Priečinok .dart_tool slúži pre ukladanie súborov, ktoré sú používané rôznymi nástrojmi jazyka Dart. [9]

.idea

Priečinok .idea obsahuje nastavenia editora, v ktorom upravujeme kód. [10]

android

Priečinok android obsahuje nastavenia špecifické pre platformu Android, zdroje a kód. [10]

ios

Priečinok ios obsahuje nastavenia špecifické pre platformu iOS, zdroje a kód. [10]

lib

Priečinok lib je hlavný priečinok Flutter projektu. V tomto priečinku sa bude nachádzať všetok zdrojový kód spojený s frameworkom Flutter. Po vytvorení nového projektu sa v ňom nachádza jeden súbor, *main.dart*, ktorý slúži ako vstupný bod Flutter aplikácie. [10]

linux

V tomto priečinku sa nachádza zdrojový kód a všetky potrebné súbory pre platformu Linux.

macos

V tomto priečinku sa nachádza zdrojový kód a všetky súbory, ktoré sú špecifické pre platformu macOS.

test

Tento priečinok obsahuje kód, ktorý sa využíva k testovaniu aplikácie. [10]

web

Web priečinok obsahuje všetky potrebné zdrojové kódy a súbory pre podporu na zostavenie web aplikácie.

windows

Priečinok windows obsahuje špecifický zdrojový kód a zdroje pre platformu Windows.

.gitignore

Súbor verzovacieho systému git v ktorom sa nachádzajú cesty k súborom projektu, ktoré nemajú byť v git repozitári zahrnuté. [10]

.metadata

V tomto súbore sa nachádzajú metadata, ktoré Flutter nástroje využívajú. [10]

analysis_options.yaml

Tento súbor obsahuje analyzér, ktorý štatisticky analyzuje Dart kód, kontroluje chyby, varovania a linty. [11]

pubspeck.lock

Tento súbor je pomocný súbor pre pubspec.yaml. V tomto súbore sa nachádza zoznam závislostí aj s ich verziami. Pri každej závislosti sa nachádza popis, odkaz kde sa nachádza a hash daného balíka závislosti. [10]

pubspec.yaml

Súbor pubspec.yaml obsahuje špecifické metadáta a konfigurácie. V tomto súbore sa konfigurujú závislosti aplikácie ako externé balíčky, obrázky, fonty atď. [10]

README.md

V tomto súbore sa nachádza približný plán programovania prvej časti práce a opis projektu.

test.iml

IML znamená IntelliJ IDEA Module. Je to súbor, ktorý využíva vývojové prostredie IntelliJ IDEA. Tento súbor obsahuje informácie o vývojových moduloch ako je android, plugin, java alebo maven. [12]

2.1.8 Výhody

V tejto časti sa nachádzajú výhody frameworku Flutter, ktoré môžu rozhodovať pri výbere frameworku na programovanie multiplatformových mobilných aplikácií.

2.1.8.1 Hot Reload

Hot reload je funkcia, ktorá nám umožňuje rýchlo a jednoducho experimentovať, vytvárať používateľské rozhrania, pridávať nové funkcie a opravovať chyby. Hot reload funguje tak, že do bežiaceho Dart Virtual Machine (DVM) vloží aktualizované súbory. Po tom ako DVM

aktualizuje triedy novou verziou polí a funkcií, Flutter framework automaticky zrekonštruuje strom widgetov a umožní nám rýchlo zobraziť účinky našich zmien. [13][14]

2.1.8.2 Vysoký výkon

Frekvencia Flutteru je 60 snímkov za sekundu, čo nám umožňuje vidieť plynulý a jasný obraz. Pri tejto frekvencií snímkov dokáže ľudské oko identifikovať akékoľvek oneskorenie. [14][15]

2.1.8.3 Vlastné widgety pre programovanie užívateľského rozhrania

Flutter obsahuje pripravené vlastné widgety. Tieto widgety sa používajú na vytvorenie vynikajúceho rozhrania a vzhľadu. Každý objekt je v tomto frameworku widget – fonty, farebné schémy, menu, tlačidlá a taktiež padding. Skombinovaním týchto widgetov ich môžeme využívať na akejkoľvek úrovni prispôsobenia. [15]

2.1.8.4 Multiplatformový vykresľovací engine

Flutter má vysoko výkonný vykresľovací engine na vykresľovanie multiplatformových aplikácií bez zmien v užívateľskom rozhraní. [15]

2.1.9 Nevýhody

V tejto časti sa nachádzajú nevýhody frameworku Flutter, ktoré môžu taktiež rozhodovať či programátor použije tento framework alebo nie.

2.1.9.1 Veľké aplikácie

Aplikácie vytvorené vo Flutteri a zabalené s príslušnými nástrojmi sú prirodzene väčšie ako natívne aplikácie. Niektoré konkurenčné frameworky dokážu vytvoriť porovnateľné aplikácie s oveľa menšou veľkosťou. Dôvod tohto je, že vo Flutter aplikácií musí byť zahrnutý vykresľovací engine a widgety čo znamená, že do aplikačného balíka je toho zahrnutého viac. [14][16]

2.1.9.2 Limitovaný ekosystém

Relatívne mladý vek a špecifickosť frameworku Flutter spôsobujú, že ekosystém je pomerne obmedzený. Existujúce knižnice Javascript, ktoré môžu ostatné frameworky voľne používať nie je tak jednoduché implementovať do Flutteru. [14]

2.1.9.3 Limitovaná podpora komunity

Z mnoho rovnakých dôvodov, pre ktoré framework trpí obmedzeným ekosystémom knižníc a nástrojov tretích strán, trpí aj obmedzenou komunitou vývojárov. Bez potrebného času alebo širokého prijatia, ktoré by sa ešte mohlo rozrásť, sú oba tieto vplyvy nevyhnutným dôsledkom výberu akejkoľvek relatívne mladej technológie. [16]

2.1.9.4 Predpísané nástroje

Vzhľadom na komplexnú povahu Fluttera súvisiacich nástrojov môže táto technológia pripadať obmedzujúca vývojárom, ktorí sú zvyknutí zostavovať projekty z mnoho dostupných frameworkov a knižníc. [16]

2.1.9.5 Dart

Často uvádzaný ako najsilnejšia a najvýhodnejšia vlastnosť frameworku Flutter a zároveň aj najväčšia nevýhoda je Dart. Tento jazyk je výkonný, produktívny a prístupný – ale je to aj jazyk, ktorý sa za posledných 5 až 6 rokov používal len v obmedzenej miere a ešte v menšej miere sa rozšíril. [14][16]

2.2 React Native

React Native je populárny framework na vytváranie mobilných aplikácií založený na jazyku JavaScript, ktorý umožňuje vytvárať natívne vykreslené aplikácie pre iOS a Android. Tento framework umožňuje vytvárať aplikácie pre rôzne platformy z jedného kódu. [17]

Bol založený spoločnosťou Facebook v roku 2015 ako open-source projekt. Po pár rokoch sa stal jedným z najlepších riešení pre vytváranie mobilných aplikácií. Niekoľko popredných svetových aplikácií je napísaných v tomto frameworku, napríklad Instagram, Facebook a Skype. [17]

2.2.1 Dôvody úspechu React Native

Po prvé, použitím React Native môžu spoločnosti vytvoriť aplikáciu iba raz a z tohoto kódu nám vznikne aj iOS aj Android aplikácia. [17]

Po druhé, React Native je založený na knižnici React, ktorá už bola populárna v dobe vydania frameworku React Native. [17]

Po tretie, React Native umožnil vývojárom frontendových aplikácií, ktorí predtým pracovali iba s webovými technológiami, vytvárať robustné a aplikácie pripravené do produkcie pre mobilné platformy. [17]

2.2.2 Architektúra React Native

V tejto časti sa nachádza popis aktuálnej a novej architektúry React Native.

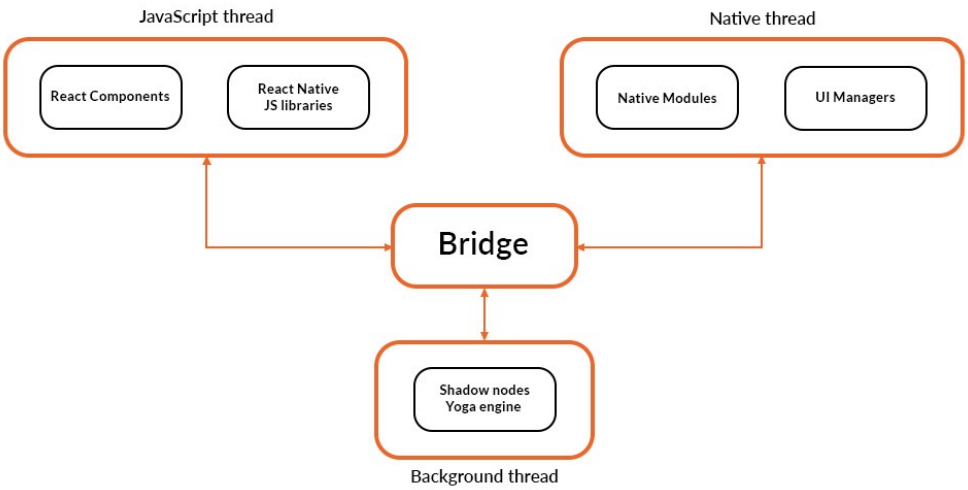
2.2.2.1 Aktuálna architektura

Proces vykresľovania aplikácie pomocou React Native zahŕňa tri druhy odlišných stromových štruktúr kde každá je zodpovedná za samostatnú časť vykresľovania:

- React Element Tree – obsahuje React elementy, JavaScript objekt, ktorý definuje ako by aplikácia alebo konkrétne zobrazenie malo vyzerať
- React Shadow Tree – obsahuje React Shadow Node, ktoré obsahujú informáciu z originálneho JavaScript elementu s pridanými údajmi o rozložení
- Host View Tree – skladá sa zo zobrazovacích prvkov špecifických pre konkrétnu platformu

[18]

Každý strom je asociovaný s korešpondujúcim vláknom – JavaScript vlákno pre React Element Tree, Shadow/Backgroud vlákno pre React Shadow Tree a UI/Main vlákno pre Host View Tree. [18]



Obrázok 12 Vlákna a komunikácia medzi nimi [18]

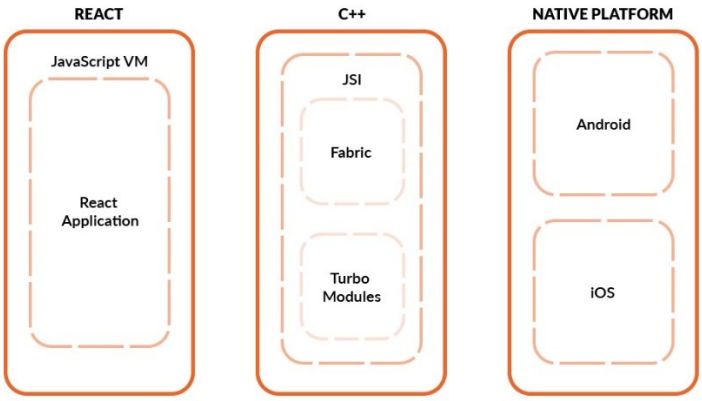
Komunikácia medzi strom/vlákno pármi prebieha pomocou Bridge – sprostredkovateľ správ, napísaný v jazyku C++, ktorý poskytuje obojsmernú komunikáciu s JavaScriptom a vrstvou v natívnej platforme pomocou správ v JSON formáte. [18]

S týmto prístupom súvisí niekoľko obmedzení. Najdôležitejšie to sú:

- Komunikácia môže prebiehať len asynchrónne – vzniká problém, napríklad pri používaní natívnych API komponentov, ktoré sa spoliehajú na synchrónnu komunikáciu
- Dáta nemôžu byť doručené pomocou zdieľanej pamäte, musia byť skopírované medzi vláknami, čo proces spomaľuje. To taktiež znamená, že dáta musia byť serializované.

2.2.2.2 Nová architektúra

Nová architektúra obsahuje tri kľúčové elementy, ktoré umožňujú prekonať obmedzenia súčasnej architektúry. [18]



Obrázok 13 Nová architektúra [18]

JavaScript Interfaces (JSI)

JavaScript Interfaces sú popisované ako odľahčené API pre vkladanie JavaScript engine do C++ aplikácie. V praxi to zabezpečuje priamu komunikáciu medzi JavaScriptom a natívnymi vrstvami použitím rozhrania medzi JavaScript a C++, bez použitia Bridge a JSON správ. JSI umožňuje napísať metódy kompletne v C++ a zaregistrovať ich v JavaScript runtime. [18]

Fabric

Fabric je nový vykresľovací systém, postavený na princípe zjednotenia väčšej časti logiky vykresľovania v C++ a zabezpečuje lepšiu spoluprácu s natívnymi platformami. Pomocou JSI, Fabric môže využívať výhody priamej komunikácie medzi JavaScriptom a natívnymi vrstvami, bez nutnosti spoliehať sa na Bridge. [18]

Turbo Native Modules

Aktuálna architektura používala pri prístupe k natívnym API Native modules. Tieto moduly poskytujú spôsob vystavenia natívnych tried pre JavaScript. [18]

Turbo Native Modules používa JSI čím sú tieto moduly oslobodené od hlavných obmedzení, ktoré predstavoval Bridge. Taktiež tieto moduly poskytujú väčšiu typovú bezpečnosť na rôznych platformách. [18]

Ďalšou výhodou Turbo Native Modules je používanie lazy loading pre moduly. To znamená, že moduly sa načítajú až keď sú potrebné. To vedie k rýchlejšiemu štartu aplikácie. [18]

2.2.3 Výhody

V tejto časti sa nachádzajú výhody frameworku React Native.

2.2.3.1 Veľká komunita vývojárov

React Native je open source čo umožňuje vývojárom prispieť svojimi vedomosťami k vývoju framework, ktorý je voľne dostupný pre všetkých. Ak má niekto problém pri vývoji, môžu sa obrátiť na komunitu pre podporu (v polovici roka 2020, sa na stránke StackOverflow nachádzalo 50 000 aktívnych prispievateľov pod tagom React Native). [17][19]

2.2.3.2 Fast refresh

Fast refresh umožňuje vývojárom spustiť aplikáciu a zároveň ju aktualizovať na nové verzie a modifikovať UI. Zmeny sú viditeľné ihneď a vývojár je odbremenený od znovu zostavenia aplikácie. [17]

2.2.3.3 Rýchle aplikácie

Niektorí tvrdia, že React Native kód môže mať negatívne vplyvy na výkon aplikácie. Napriek tomu, že JavaScript neposkytne taký výkon ako natívny kód, rozdiel je nepostrehnuteľný ľudským okom. [17][19]

2.2.3.4 Zabezpečená budúcnosť

Vzhľadom na tempo, akým tento framework ovládol trh a jeho jednoduchý prístup k riešeniu problémov pri vývoji, budúcnosť frameworku React Native pre vývoj multiplatformových aplikácií vyzerá jasne. [17]

2.2.4 Nevýhody

V tejto časti sa nachádzajú nevýhody frameworku React Native.

2.2.4.1 Absencia niektorých vlastných modulov

Napriek tomu, že React Native existuje už niekoľko rokov, niektoré vlastné moduly buď nechávajú miesto pre vylepšenie alebo vôbec neexistujú. To by znamenalo, že vývojár musí udržovať separátne kódy (React Native, iOS a Android) namiesto jedného. To ale nie je bežné pokiaľ sa vývojári nesnažia spraviť aplikáciu od nuly alebo upravujú nejakú existujúcu aplikáciu. [17][19]

2.2.4.2 Problémy s kompatibilitou a ladením

Napriek tomu, že je React Native používaný špičkovými technologickými firmami je stále v beta verzii. Vývojári môžu prísť do styku s rôznymi chybami v oblasti kompatibility balíčkov alebo s ladiacimi nástrojmi. Môže to mať negatívny vplyv na vývoj, pretože vývojári môžu stráviť dlhší čas zdĺhavým riešením problémov. [17]

2.2.4.3 Škálovateľnosť

Väčšinu času bude React Native pracovať dobre aj v prípade, že aplikácia narastie do veľmi sofistikovaných, komplexných riešení. Facebook a Skype používajú React Native niekoľko

rokov. Na druhej strane, Airbnb, sa rozhodlo použiť React Native ešte keď boli len začínajúci startup[9]. [17]

2.2.4.4 Pomoc vývojára natívnych aplikácií

Spôsobom akým funguje aktuálna architektúra React Native, pomoc vývojára natívnych aplikácií môže byť potrebná. V prípade, že vývojár zodpovedný za projekt nemá skúsenosti s natívnym vývojom bude mať ťažké zahrnúť natívny kód do kódu React Native. Časom sa ukázalo, že React Native nie je vhodný pre plány rastu spoločnosti a rozhodli sa pre vývoj dvoch natívnych aplikácií. Súčasným pokrokom React Native sa správnym výberom architektúry dá problémom so škálovateľnosťou ľahko predísť. [17][20]

2.3 Kotlin Multiplatform Mobile

Kotlin Multiplatform Mobile je SDK pre vývoj multiplatformových mobilných aplikácií. Môžeme vytvárať multiplatformové mobilné aplikácie a zdieľať časti aplikácie ako základné vrstvy, business logiku, prezentačnú logiku medzi operačnými systémami Android a iOS. [21]

Kotlin Multiplatform Mobile je podmožina Kotlin Multiplatform (KMP). Kotlin Multiplatform umožňuje vývoj počítačových a webových aplikácií. Kotlin Multiplatform bol vyvinutý spoločnosťou Jetbrains s využitím Kotlinu ako programovacieho jazyka. Kotlin vo verzií 1.2, ktorá bola vydaná v roku 2017 už obsahoval SDK Kotlin Multiplatform Mobile. Dôvod je snaha spoločnosti Jetbrains na urýchlenie a zjednodušenie vývoja aplikácií pre rôzne typy zariadení. [22]

Zatiaľ čo môže byť KMP označovaný ako framework, je to vlastne súbor technológií, ktorý umožňuje vývojárom zdieľať kód naprieč platformami a kombinuje výhody multiplatformového a natívneho prístupu vývoja. [22]

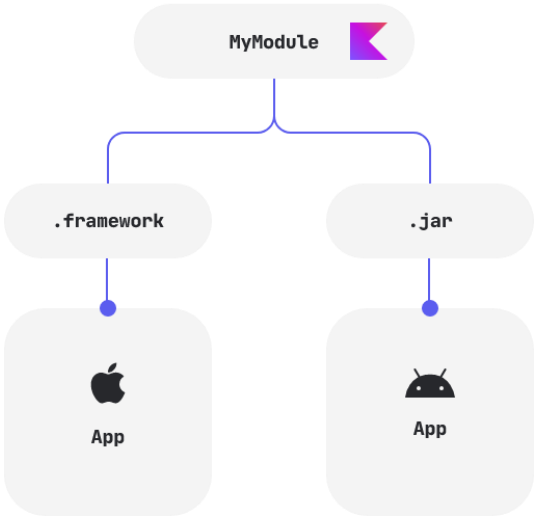
2.3.1 Kompilácia Kotlin kódu

Pre kompilovanie Kotlin kódu do natívnych binárnych súborov sa používa technológia Kotlin/Native, ktorá funguje bez použitia virtual machine. Používa backend založený na LLVM

[9] Spoločnosť založená jedným alebo viacerými podnikateľmi, ktorí chcú vyvinúť produkt alebo službu, po ktorých je podľa nich dopyt. https://www.investopedia.com/terms/s/startup.asp

kompilery pre Kotlin kompiler a natívnu implementáciu pre štandardnú knižnicu Kotlinu. Kotlin/Native je primárne používaný pre kompilovanie na platformy kde nie je možné využívať virtual machine alebo kde to nie je vhodné používať, ako sú napríklad vstavané zariadenia alebo iOS. Je vhodný vtedy, keď vývojár potrebuje vytvoriť program, ktorý nevyžaduje ďalší runtime alebo virtual machine. [21]

Zdieľaný kód napísaný v Kotline je skompilovaný do bitového kódu Java virtual machine pre Android pomocou Kotlin/JVM a do natívnych knižníc pre iOS pomocou Kotlin/Native. [21]



Obrázok 14 Kompilácie Kotlin kódu [21]

2.3.2 Výhody

V tejto časti sa nachádzajú výhody frameworku Kotlin Multiplatform Mobile.

2.3.2.1 Voliteľnosť v zdieľaní kódu

V porovnaní s ostatnými multiplatformovými frameworkami jeden z kľúčových rozlišovacích faktov Kotlin Multiplatform je vo voliteľnosti zdieľania kódu. Vývojárske tímy si môžu vybrať, ktoré časti, funkcie a moduly chcú zdieľať. [22]

Navyše v prípade, že už je aplikácia napísaná pre jednu platformu tak tento kód nemusíme zahodiť a začať odznova. Je možné prestavať existujúcu aplikáciu bit po bite, funkcia po funkcií, zatiaľ čo vyvíjať aplikáciu pre ďalšiu platformu. Toto môže pomôcť biznisom znížiť výskyt chýb a ostatných chýb pri úprave rovnakej aplikácie pre rôzne platformy. [22]

2.3.2.2 Možnosť rozšírenia a vylepšenia natívnych aplikácií

Biznisy môžu stále nasadiť KMP aj v prípade, že vývojárske tímy majú zhotovené separátne natívne aplikácie., ktoré nemajú spoločný kód. Pri budúcich iteráciách vývojového procesu môžu vývojári kedykoľvek začať používať Kotlin Multiplatform a zdieľať toľko kódu, koľko sa rozhodnú. [22]

2.3.2.3 Skrátenie času na vývoj a rýchlejšie uvedenie na trh

Použitie Kotlin Multiplatform na vývoj multiplatformových aplikácií môže ušetriť až 30-50% vývojového času. Okrem toho, pretože Android a iOS používajú tie isté riadky kódu pre business logiku, pridávanie a testovanie nových funkcií je jednoduchšie a rýchlejšie. [22][23]

2.3.2.4 Lepšia konzistencia medzi platformami

Použitím zdieľaného kódu vyústi v lepšej konzistencií architektúry a funkcií. Taktiež to pomáha vývojárskemu tímu odhaliť viac chýb v počiatku vývoja. [22][23]

2.3.2.5 Úspora nákladov na vývoj

Kotlin znižuje náklady na projekty pretože podniky nemusia vytvárať samostatné tímy pre každú platformu alebo najímať vývojárov, ktorí píšu tie isté riadky kódu viackrát. Zo skúseností KMP komunity, vývoj aplikácie na Android a iOS naraz, je lacnejší ako vývoj na každú platformu zvlášť. [22]

2.3.2.6 Rastúca komunity vývojárov

Tímy vývojárov po celom svete zvyšujú svoje schopnosti v programovacom jazyku Kotlin, vrátane KMM a KMP. Toto je čiastočne spôsobené rozhodnutím spoločnosti Google adaptovať Kotlin ako preferovaný jazyk pre Android. [22]

2.3.3 Nevýhody

V tejto časti sa nachádzajú nevýhody frameworku Kotlin Multiplatform Mobile.

2.3.3.1 Málo dostupných knižníc

KMP ponúka robustné knižnice pre kľúčové úlohy ako práca s lokálnym úložiskom, sieťové operácie, parsovanie JSONu, logovanie atď. Tieto knižnice alebo balíky nie sú tak dobre otestované v porovnaní so zavedenými frameworkami. Vývojári musia vytvárať wrappery

okolo natívnych riešení. Online zdrojov ohľadom KMP nie je tak veľa v porovnaní so zaužívanými frameworkami ale očakáva sa, že ich počet bude rásť spolu s rozširovaním ekosystému Kotlin a komunitou vývojárov. [22]

2.3.3.2 Chýbajúca podpora zdieľania logiky UI

Kotlin Multiplatform momentálne nepodporuje možnosť zdieľania UI logiky. Pri písaní kódu pre iOS vývojári nemajú inú možnosť ako napísať UI časť natívne. Používajú sa Apple frameworky jako SwiftUI a UIKit. Existuje však možnosť zdieľania UI kódu v budúcnosti s pomocou Jetpack Compose a Compose for Desktop. Compose multi platform zdieľa UI kód pre počítače, web a Android. To by doplnilo riešenie KMM, pretože to je tiež v jazyku Kotlin. [22]

2.3.4 Najvhodnejšie použitie

V tejto časti sa nachádzajú informácie o tom kedy sa zamyslieť nad použitím Kotlin Multiplatform Mobile pred ostatnými frameworkami.

2.3.4.1 Existujúca Android aplikácia s potrebou rozšírenia na iOS

Namiesto vývoja ďalšej natívnej aplikácie, v ktorej by bolo potrebné prispôsobovať alebo duplikovať celý kód, KMP prácu uľahčí. Práve na riešenie tejto situácie bol Kotlin Multiplatform navrhnutý. Pri rozširovaní aplikácie na iOS musí vývojársky tím pripraviť architektúru s ohľadom na viac platforiem. Nebude to fungovať out of the box[10]. [22]

2.3.4.2 Zdieľanie iba časti kódu medzi platformami

Vďaka voliteľnosti zdieľania iba časti kódu môžeme obmedziť použitie KMP na určité funkcie alebo moduly. KMM je ideálne riešenie pre vývoj multiplatformových aplikácií kde zdieľame logiku, ktorá nemôže byť presunutá na backend. V kontraste s ostatnými frameworkami ako React a Flutter, KMM dovoľuje použiť ich technológiu na zdieľanie iba časti kódu pričom zvyšok ostáva nedotknutý. [22]

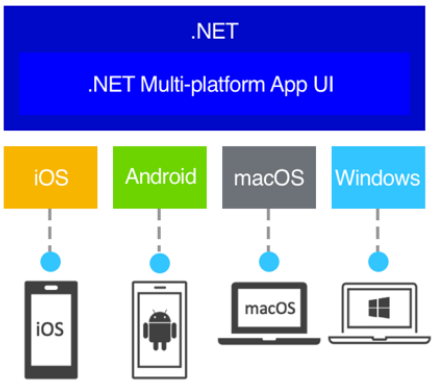
[10] Funkcia, ktorá je predpripravená na použitie https://en.wikipedia.org/wiki/Out_of_the_box_(feature)

2.3.4.3 Niektoré funkcie treba spraviť natívne

Napriek tomu že React Native a Flutter poskytujú vlastnosti a funkcie podobné natívnemu vývoju, stále existujú funkcie, ktoré je najlepšie spraviť natívne. S pomocou Kotlin Multiplatform Mobile, môžeme tieto funkcie zostaviť natívne a zvyšok kódu napísať pomocou KMM. [22]

2.4 .NET MAUI

.NET Multi-platform App UI (.NET MAUI), je multiplatformový framework pre vytváranie natívnych mobilných a počítačových aplikácií. Pomocou .NET MAUI môžeme vytvárať aplikácie ktoré budú fungovať na operačných systémoch Android, iOS, macOS a Windows z jedného zdieľaného kódu. [24]



Obrázok 15 .NET MAUI platformy [24]

.NET MAUI je pod licenciou open source a je evolúciou Xamarin.Forms. Je rozšírený z mobilných na desktopové aplikácie s UI prvkami, ktoré sú od základu prepracované na výkon a rozšíriteľnosť. Používaním .NET MAUI je možné vytvoriť multiplatformové aplikácie s použitím jedného projektu, ale môžeme pridať kód pre špecifickú platformu pokiaľ je to potrebné. Jedným z hlavných cieľov frameworku .NET MAUI je možnosť implementovať čo najviac UI a aplikačnej logiky z jedného kódu. [24]

.NET MAUI je pre vývojárov, ktorí chcú:

- Písať multiplatformové aplikácie pomocou XAML a C# z jedného kódu v prostredí Visual Studio
- Zdieľať UI layout a dizajn naprieč platformami
- Zdieľať kód, testy a business logiku naprieč platformami

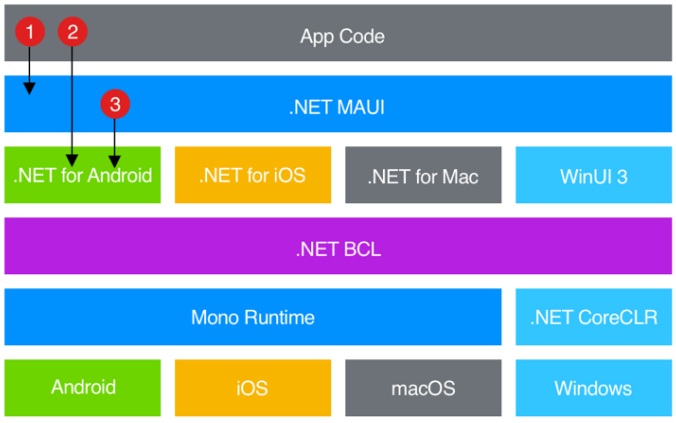
[24]

2.4.1 Ako funguje .NET MAUI

.NET MAUI zjednocuje Android, iOS, macOS a Windows aplikačné rozhrania do jedného aplikačního rozhrania, ktoré umožňuje „napíš raz, bež všade“ vývojársku skúsenosť, a zároveň poskytuje hlboký prístup ku každému aspektu natívnej platformy. [24]

.NET 6 a vyššie poskytuje niekoľko platformovo špecifických frameworkov: .NET pre Android, .NET pre iOS, .NET pre macOS a Windows UI 3 (WinUI 3). Všetky tieto frameworky majú prístup k tej istej .NET Base Class Library (BCL). Táto knižnica abstrahuje detaily základnej platformy mimo kódu. BCL závisí od .NET runtime, ktoré poskytuje rozhranie na realizáciu zdrojového kódu. Pre Android, iOS a macOS je toto prostredie implementované pomocou Mono. Na Windowse je toto prostredie implementované pomocou .NET CoreCLR. Zatiaľ čo BCL dovoľuje aplikáciám na rôznych platformách zdieľať business logiku, platformy majú rôzne spôsoby definovania používateľského rozhrania a poskytujú rôzne spôsoby ako jednotlivé prvky UI spolupracujú. Je možné vytvoriť UI pre každú platformu zvlášť pomocou frameworku pre konkrétnu platformu. Tento spôsob vyžaduje udržiavanie zdrojového kódu pre každú jednotlivú platformu. [24]

.NET MAUI poskytuje jeden framework pre vytvorenie UI pre mobilné a desktopové aplikácie. Tento diagram zobrazuje architektúru .NET MAUI. [24]



Obrázok 16 .NET MAUI architektúra [24]

V rámci aplikácie vyváranej vo frameworku .NET MAUI je písaný kód, ktorý priamo interaguje s .NET MAUI API (1). NET MAUI následne priamo využíva API natívnych platforiem (3). V prípade, že je to potrebné môže aplikačný kód komunikovať s API natívnych platforiem priamo (2). [24]

.NET MAUI aplikácie môžu byť vytvorené na PC alebo Mac a skompilované do balíčkov natívnych aplikácií:

- Android aplikácie sú skompilované z jazyka C# do intermediate language (IL), ktorý je skompilovaný just-in-time (JIT) do natívnej assembly pri spustení aplikácie.
- iOS aplikácie sú plne skompilované ahead-of-time (AOT) z C# do natívneho ARM assembly kódu.
- macOS aplikácie používajú Mac Catalyst, riešenie Apple, ktoré prenesie aplikáciu vytvorenú pomocou UIKit na počítač a podľa potreby ju rozšíri o AppKit a API platformy.
- Windows aplikácie používajú Windows UI 3 knižnicu na vytvorenie natívnych aplikácií.

[24]

2.4.2 .NET MAUI vs Xamarin.Forms

V tejto časti sa nachádza porovnanie frameworkov .NET MAUI a jeho predchodcu Xamarin.Forms.

2.4.2.1 Architektúra

Hlavná zmena v architektúre je tá, že .NET MAUI je integrované do .NET 6. [25]

2.4.2.2 Projektová štruktúra

V .NET MAUI je štruktúra, ktorá obsahuje jeden projekt pre všetky platformy. V Xamarin.Forms bol pre každú platformu zvlášť projekt. [25][26]

2.4.2.3 Renderer a Handler architektúry

Vo frameworku Xamarin sú UI prvky spravené pomocou renderer architektúra. Keď chcel vývojár zmeniť natívny UI prvok, musel vytvoriť vlastný renderer pre každú platformu. Tieto renderery sú náročné na výkon a veľkosť aplikácie. [25]

V .NET MAUI sa používa handler architektúra, ktorá je voľne viazaná na natívnu assembly. To vedie k odľahčenej aplikácií s lepším výkonom. [25]

2.4.2.4 Správa zdrojov

V .NET MAUI sa zmenila správa zdrojov, hlavne obrázkov. Už nie je potrebné mať pre každú platformu zvlášť obrázky. Jeden SVG obrázok je dostačujúci pre všetky platformy. Tento SVG obrázok je premenený na PNG tak, aby fungoval na všetkých platformách. [25]

2.4.2.5 Podpora Hot Reload

.NET MAUI obsahuje podporu pre C#. V podstate to znamená, že aj zmena na business vrstve sa reloadne. V Xamarine bol dostupný iba XAML hot reload. [25][26]

2.4.2.6 Grafické API

Vo frameworku Xamarin sa nenachádza žiadne API na kreslenie. .NET MAUI obsahuje grafické API, ktoré poskytuje plátno na kreslenie a maľovanie tvarov. [25]

2.4.2.7 Podporované vzory

Vo frameworku Xamarin sú podporované architektonické vzory MVVM a RxUI. V .NET MAUI sa tento zoznam rozšíril o Model-View-Update (MVU). [25]

2.5 Ionic Framework

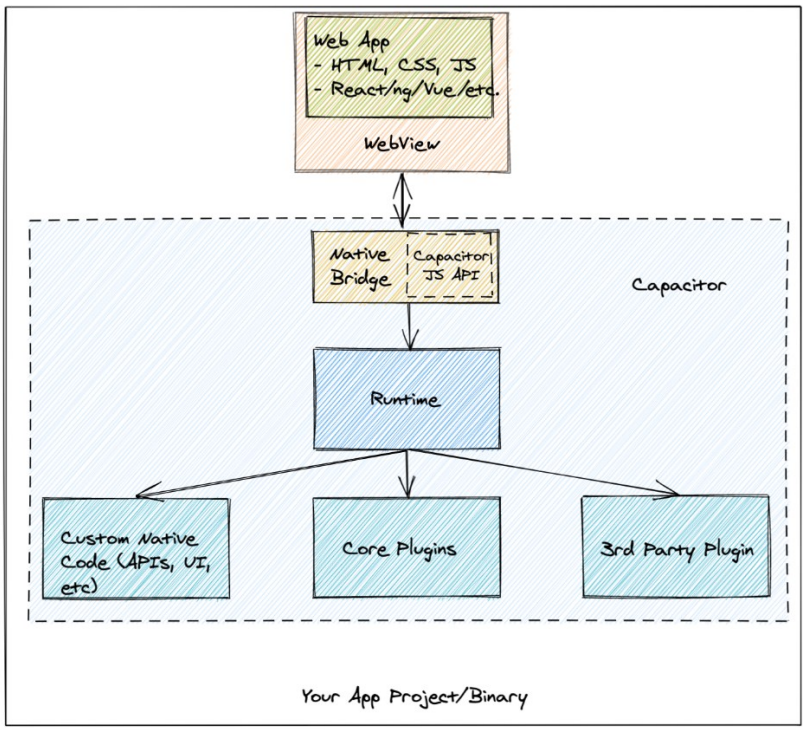
Ionic Framework bol vytvorený v roku 2013 ako open source SDK pre hybridné mobilné aplikácie. Doteraz bolo pomocou Ionic Frameworku vytvorených viac ako 5 miliónov aplikácií. Je známy pre poskytovanie platformovo špecifických UI prostredníctvom knižnice natívnych komponentov pre iOS a Android. Ionic Framework je v podstate npm balíček, ktorý potrebuje Node.js pre funkčnosť. [27]

Ionic Framework používa frontendové technológie ako HTML, CSS, JavaScript a Angular pre vývoj aplikácie. Použitím web technológií pomáha Ionic Frameworku zostaviť multiplatformové aplikácie z jedného zdrojového kódu. V podstate to umožňuje web vývojárom vytvoriť webové stránky, ktoré budú spustené v instancii webového prehliadača nazvaného WebView. WebView sa môže dodávať ako zásuvný modul a je to podstatný aplikačný komponent pre vykresľovanie web stránok a zobrazenie ich ako natívnu aplikáciu. [27]

Prvá verzia Ionic Frameworku bola založená na frontendovom frameworku Anglular, ktorý je používaný na zostavenie dynamických web stránok a progresívnych web stránok. Ionic Framework môže používať Angular Comman-Line Interface a komponenty pre vytvorenie plne funkčných mobilných aplikácií. [27]

Ionic Framework používa WebView a ten nemá prístup k hardwaru zariadenie v základe. [27]

Pristupovať k natívnym funkciám je možné pomocou natívneho runtime zvaného Capacitor. Capacitor ponúka prístup k funkciám ako sú súborový systém, kamera, lokalizačné služby a ďalšie. [28]



Obrázok 17 Architektúra Ionic aplikácie [29]

2.5.1 Výhody

V tejto časti sa nachádzajú výhody frameworku Ionic.

2.5.1.1 Jeden zdrojový kód

Ionic Framework umožňuje vytvoriť jeden zdrojový kód pre všetky platformy. Ionic Framework umožňuje vytvárať mobilné aplikácie bez nutnosti najímania natívnych vývojárov. Môže ho používať každý, kto sa vyzná vo webových technológiách a môže tak využívať svoje webové zručnosti na vytváranie plne funkčných mobilných aplikácií. [27]

2.5.1.2 Široká škála integračných možností a zásuvných modulov

Ionic Framework poskytuje možnosti integrovania aplikácie s mnohými nástrojmi. Tieto aplikácie môžu mať jednoduchý prístup k analytickým nástrojom, platobným systémom, bezpečnostným a testovacím nástrojom. [27][30]

2.5.1.3 Rozsiahly výber prvkov používateľského rozhrania a rýchle prototypovanie

Používanie hotových prvkov používateľského rozhrania pomáha vytvárať prototypy vašich budúcich aplikácií v pomerne krátkom čase. Na tento účel môžete použiť nástroj na vytváranie prototypov s názvom Ionic Creator. Spravuje ho tím Ionic a ponúka rozhranie drag & drop na konštrukciu interaktívnych prototypov, ale nemožno ho použiť na konštrukciu celej aplikácie. [27]

2.5.1.4 Pohodlnosť testovania

Vzhľadom na to, že Ionic aplikácie používajú WebView stačí nám na testovanie iba prehliadač zariadenia. Je to oveľa pohodlnejšie, pretože nemusíte používať testovacie zariadenie, aby ste sa uistili, že všetko beží v poriadku. [27]

2.5.1.5 Silná komunita

S viac ako 5 miliónmi vývojárov a neustálou aktivitou na fóre, vždy by sa vám malo podariť dostať odpoveď na vašu otázku, pokiaľ to nebolo zahrnuté v dokumentácií. [27]

2.5.2 Nevýhody

V tejto časti sa nachádzajú nevýhody frameworku Ionic.

2.5.2.1 Nedostatočný výkon natívnych aplikácií

Pokiaľ ide o výkon náročných aplikácií, Ionic nie je vhodný. Použitie WebView na vykresľovanie aplikácie vykazuje dobré výsledky pre bežné funkcie mobilnej aplikácie. V prípade aplikácií náročných na výkon, ako je napríklad Snapchat, ktorá využíva rozšírenú realitu prostredníctvom fotoaparátov smartfónu, alebo aplikácií náročných na grafiku spôsobí, že vaša aplikácia bude pomalá a bude reagovať spomalene. [27][30]

2.5.2.2 Systém závislý od zásuvných modulov

Na prístup k natívnym funkciám sa používajú zásuvné moduly. Vďaka množstve pripravených zásuvných modulov je jednoduché nájsť taký, ktorý obsahuje funkcionalitu, ktorú

potrebujeme. V prípade, že potrebujeme modul na nejakú špecifickú funkciu alebo prístup k neštandardnému hardwaru, je potrebné si zásuvný modul vytvoriť. [27]

2.5.2.3 Možné bezpečnostné problémy

Pri vytváraní hybridných aplikácií je bezpečnosť bežným problémom, pokiaľ sa vaša aplikácia dá spätne analyzovať. Existuje veľa spôsobov, ako ohroziť dianie vo vašej mobilnej aplikácii alebo PWA, napríklad útok typu man-in-the-middle. To je možné z toho dôvodu, že aplikácia Ionic je v podstate webová stránka, ktorá beží na zariadení. [27][30]

2.5.2.4 Veľkosť aplikácie

Písanie aplikácie pomocou HTML, CSS a JavaScriptu znamená oveľa viac kódu a ešte aj pridanie knižníc, zásuvných modulov a závislostí vyústi v aplikáciu, ktorá je oveľa väčšia ako aplikácia natívna. [27]

II. PRAKTICKÁ ČÁST

3 PRIESKUM TRHU S APLIKÁCIAMI S PODOBNOU TÉMATIKOU

V tejto kapitole sú popísané vybrané aplikácie, ktoré sa zaoberajú podobnou tematikou ako aplikácia, ktorá je výstupom tejto diplomovej práce.

3.1 Eventify

Eventify je platforma, ktorá vyzerá byť mierená na organizovanie rôznych konferencií a správu veľkých udalostí. V prípade, že sa užívateľ dostane na sekciu udalosti tak sa tu nachádza veľmi veľa informácií o vybranej udalosti. Niektoré z dostupných informácií o konkrétnej udalosti sú: harmonogram, rečníci, vystavovatelia, sponzori, novinky, sprievodca udalosťou a mnoho ďalšieho. Udalosti sa vyhľadávajú podľa mena a kódu udalosti.

Vyzerá to tak, že pre vytvorenie eventu a zaregistrovania sa ako poskytovateľ udalostí je potrebné kontaktovať priamo Eventify.

Možnosti prieskumu tejto aplikácie boli obmedzené z toho dôvodu, že formulár na registráciu nebol nájdený. Pri snahe dostať sa do sekcie aplikácie kde treba prihlásenie bolo na obrazovke potrebné zadať email. Po zadaní bola vypísaná hláška, že užívateľ nie je zaregistrovaný pre túto udalosť.

3.2 Meetup

Meetup je platforma, kde ľudia dokážu vyhľadávať udalosti na témy ktoré ich zaujímajú podľa kľúčových slov. Pri registrácií si človek vyberie oblasti ktoré ho zaujímajú a taktiež lokáciu v ktorej chce aby sa mu udalosti zobrazovali. Podľa týchto kritérií bude mať na hlavnej stránke udalosti, ktoré by užívateľa mohli zaujímať. Taktiež sa užívateľ môže pripájať do skupín, ktoré ho zaujímajú a následne sa udalosti z týchto skupín budú taktiež zobrazovať na hlavnej stránke.

V detaile akcií sa nachádzajú detailné informácie o konkrétnej akcii, čas a miesto konania, účastníci, fotky a pre členov skupiny v ktorej sa daná akcia koná je k dispozícií aj chat.

V detaile skupiny sa nachádzajú detailné informácie o skupine, nadchádzajúce a predošlé akcie, zoznam členov, fotky, diskusie a ďalšie.

V prípade, že by užívateľ chcel založiť vlastnú skupinu tak aj taká možnosť sa na tejto platforme nachádza. Táto funkcia je ale spoplatnená. Na základe toho aký plán si zvolíme tak od toho sa aj odvíjajú možnosti spravovania skupiny a akcií.

Prístup k tejto platforme je prostredníctvom webu www.meetup.com alebo prostredníctvom aplikácie na operačný systém Android a iOS.

3.3 Facebook

V tejto časti sa nachádzajú informácie ohľadom aplikácie Facebook. Tieto funkcie sú rozdelené pretože nie všetky spoločne spolupracujú.

3.3.1 Facebook Podujatia

Facebook Podujatia je súčasťou Facebooku, kde sa zobrazujú podujatia primárne podľa užívateľovej lokácie. Nachádza sa tu niekoľko možností filtrovania: podľa lokácie, podľa dátumu, podľa kategórie atď. V detaile podujatia sa nachádzajú všetky potrebné informácie ako sú popis, usporiadateľ, miesto konania, dĺžka trvania a počet ľudí, ktorí sa vyjadrili že sa akcie zúčastnia alebo iba majú záujem. Taktiež sa tu nachádza diskusia, ktorá môže slúžiť len pre usporiadateľa akcie na uverejňovania informácií alebo na otvorené diskutovanie. Záleží podľa nastavenie usporiadateľa.

Podujatie môže na Facebooku založiť ktokoľvek bez poplatku. Usporiadateľom môže byť súkromný profil alebo Facebook stránka. Taktiež sa dá zvoliť niekoľko typov podujatia z pohľadu možných účastníkov. Podujatia môže byť:

- súkromné – iba pre ľudí s pozvánkou
- verejné – to môže vidieť ktokoľvek na Facebooku aj mimo neho
- pre priateľov – vidieť podujatie budú iba priatelia usporiadateľa
- pre skupinu – vidieť podujatie môžu iba členovia zvolenej skupiny

3.3.2 Facebook Skupiny

Facebook Skupiny je časť Facebooku kde sa užívateľovi zobrazujú príspevky zo skupín, ktorých je členom. Tieto skupiny majú za cieľom spájať ľudí s podobnými záujmami, prácou atď. Záleží pod akým účelom bola konkrétna skupina vytvorená. Jednou z hlavných funkcií skupín je diskusia kde môžu členovia konkrétnej skupiny diskutovať. Funguje to tak, že člen skupiny vytvorí príspevok, ktorý môže obsahovať text, obrázky, videá, ankety, gif, súbor dokonca aj živé video. Ostatní členovia skupiny môžu v komentároch ku konkrétnemu

príspevku viesť diskusiu. Taktiež v skupine môžu existovať vytvorené témy pod ktorými môžu členovia vyhľadať konkrétne príspevky, ktoré ich zaujímajú. Každá téma vystupujú pod hashtagom s názvom témy a pre zahrnutie príspevku k vybranej téme treba tento hashtag s názvom témy zahrnúť v príspevku. Taktiež sa dajú prehliadať média a súbory samostatne bez príspevkov s nimi spojenými.

Skupinu môže vytvoriť ktokoľvek potrebuje k tomu zvoliť len názov skupiny a viditeľnosť skupiny. Tu sú 2 možnosti a to Verejná – členov skupiny a ich príspevky môže vidieť ktokoľvek a Súkromná – členov skupiny a ich príspevky budú vidieť iba ostatní členovia skupiny.

3.3.3 Messenger

Messenger patrí do Facebook ekosystému a na používanie je potrebné Facebook účet. Po vytvorení Facebook účtu je možné Facebook účet deaktivovať a používať ho iba pre Messenger. Messenger slúži na komunikáciu medzi jednotlivcami, prípadne je možné vytvárať chatové skupiny kde môže chatovať niekoľko členov medzi sebou. Forma komunikácie môže byť pomocou textových správ, hlasových správ, obrázkov, videí, gifov a emotikonov. Na každý typ správy je možné reagovať prostredníctvom emotikonov. Taktiež je možné prostredníctvom Messengeru spraviť aj hlasový, prípadne video hovor. Prostredníctvom Messenger aplikácie na operačným systéme Android a iOS je taktiež možné zdieľať aktuálnu polohu mobilného telefónu. V minulosti Messenger obsahoval viacero funkcií vrátane hier a hlasovaní v skupinovom chate, ale toto bolo z dôvodu pravidiel ochrany osobných údajov v Európe deaktivované.

3.4 Eventbrite

Eventbrite je platforma kde ľudia môžu vyhľadávať rôzne udalosti. Pri otvorení aplikácie (užívateľ nemusí byť prihlásený) sa aplikácia opýta užívateľa lokáciu v ktorej chce nájsť udalosti. Následne sú mu zobrazené všetky udalosti zoradené podľa dátumu udalosti. V prípade, že sa užívateľ prihlási tak si môže zvoliť aké typy udalostí chce, aby sa mu zobrazovali v zobrazení udalostí. Tieto udalosti môže užívateľ filtrovať rôznymi spôsobmi. Užívateľ môže začať sledovať jednotlivých organizátorov udalostí a tieto udalosti mu potom bude zobrazovať vo zobrazení udalosti úplne navrchu. Nachádzajú sa tu voľne dostupné, ale aj platené udalosti. Lístky na platené udalosti je možné zakúpiť priamo v aplikácií. V detaile

udalosti sa nachádzajú základné informácie k udalosti – dátum konania, miesto konania, prípadne cena lístka a informácie o udalosti.

Pre vytváranie udalostí má Eventbrite druhú aplikáciu s názvom Organizer. Tu môže akýkoľvek zaregistrovaný užívateľ vytvoriť udalosť. Pri vytváraní udalosti musí užívateľ vyplniť základné údaje o udalosti – názov, miesto konania, dátum konania, typ udalosti, vstupenky – zadarmo, platené, viditeľnosť – verejná, privátna. Organizátor tu taktiež vidí rôzne štatistiky zárobkov, udalosti ktoré aktuálne prebiehajú, minulé udalosti a koncepty.

V prípade, že udalosť bude platená tak si Eventbrite účtuje poplatky.

3.5 Porovnanie relevantných aplikácií

Tabuľka 1 zobrazuje porovnanie relevantných aplikácií.

Tabuľka 1 Porovnanie relevantných aplikácií

	Akcošky	Meetup	Facebook	Eventbrite
Zobrazovanie udalostí	✓	✓	✓	✓
Jednoduchý prístup k udalostiam	✓	✓	✓	✓
Verejné udalosti	✗	✓	✓	✓
Privátne udalosti	✓	✓	✓	✓
Systém pozývania na udalosti v rámci aplikácie	✓	?	✓	✗
End to end šifrovanie správ	✓	✗	NBD[11]	N/R[12]
Systém skupín	✓	✓	✓	✗
Zdieľanie médií medzi účastníkmi	✓	✓	✓	✗
Rozšírené funkcie v chate	TBD[13]	✗	✗	N/R[14]

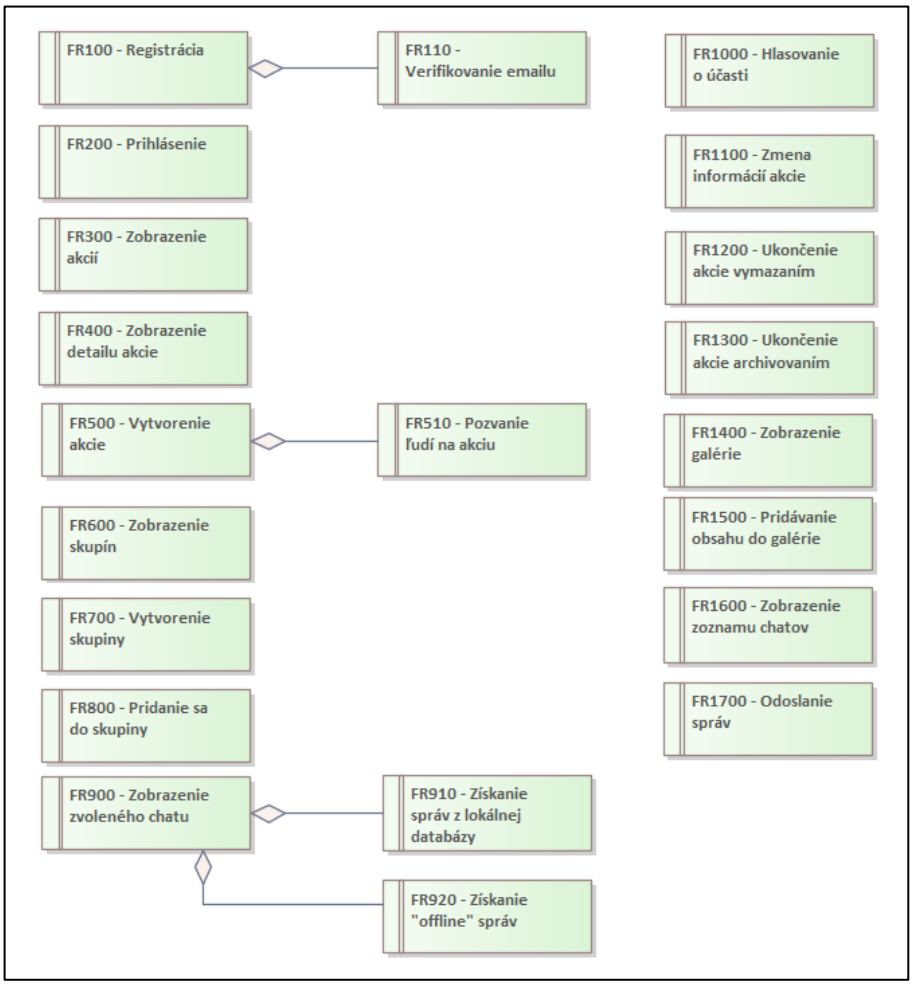
[11] Not by default – nie v základnom nastavení
[12] Not relevant – nie je relevantné
[13] To be determined – bude rozhodnuté
[14] Not relevant – nie je relevantné

4 MODELOVANIE SYSTÉMU

V tejto kapitole sa zameriame na modelovanie systému. Určíme si funkcionálne a nefunkcionálne požiadavky systému a taktiež sa pozrieme na konkrétne prípady užitia, scenáre a ako sú v systéme implementované.

4.1 Funkcionálne požiadavky

Obrázok 18 zobrazuje všetky funkcionálne požiadavky na systém.



Obrázok 18 Funkcionálne požiadavky

FR100 – Registrácia – Aplikácia umožní užívateľovi registráciu do systému.

FR110 – Verifikovanie emailu – Aplikácia pri registrácií vyžaduje overenie toho, že užívateľ ma prístup k emailu pod ktorým sa chce zaregistrovať.

FR200 – Prihlásenie – Aplikácia umožní užívateľovi prihlásenie do systému.

FR300 – Zobrazenie akcií – Aplikácia umožní užívateľovi zobraziť akcie na ktoré bol pozvaný.

FR400 – Zobrazenie detailu akcie – Aplikácia umožní užívateľovi zobraziť detail zvolenej akcie, kde sa nachádzajú všetky potrebné údaje ohľadom zvolenej akcie.

FR500 – Vytvorenie akcie – Aplikácia umožní užívateľovi vytvoriť novú akciu.

FR510 – Pozvanie ľudí na akciu – Aplikácia umožní užívateľovi pri vytváraní novej akcie pozvať ľudí z vybranej skupiny na akciu.

FR600 – Zobrazenie skupín – Aplikácia umožní užívateľovi zobraziť zoznam skupín v ktorých sa nachádza

FR700 – Vytvorenie skupiny – Aplikácia umožní užívateľovi vytvoriť novú skupinu.

FR800 – Pridanie sa do skupiny – Aplikácia umožní užívateľovi pripojiť sa do skupiny pomocou kódu na pripojenie.

FR900 – Zobrazenie zvoleného chatu – Aplikácia umožní užívateľovi zobraziť konverzáciu v chate.

FR910 – Získanie správ z lokálnej databázy – Aplikácia získa z lokálnej databázy správy, ktoré už užívateľ stiahol zo servera.

FR920 – Získanie „offline“ správ – Aplikácia stiahne zo serveru správy, ktoré užívateľovi prišli keď bol offline.

FR1000 – Hlasovanie o účasti – Aplikácia umožní užívateľovi zmeniť stav jeho účasti na akcií.

FR1100 - Zmena informácií akcie – Aplikácia umožní zakladateľovi akcie zmeniť jednotlivé informácie ku zvolenej akcií.

FR1200 – Ukončenie akcie vymazaním – Aplikácia umožní zakladateľovi akcie ukončiť akciu vymazaním akcie z databázy.

FR1300 – Ukončenie akcie archivovaním - Aplikácia umožní zakladateľovi akcie ukončiť akciu presunutím akcie do archívu.

FR1400 – Zobrazenie galérie - Aplikácia umožní užívateľovi zobraziť všetok mediálny obsah spojený so zvolenou akciou.

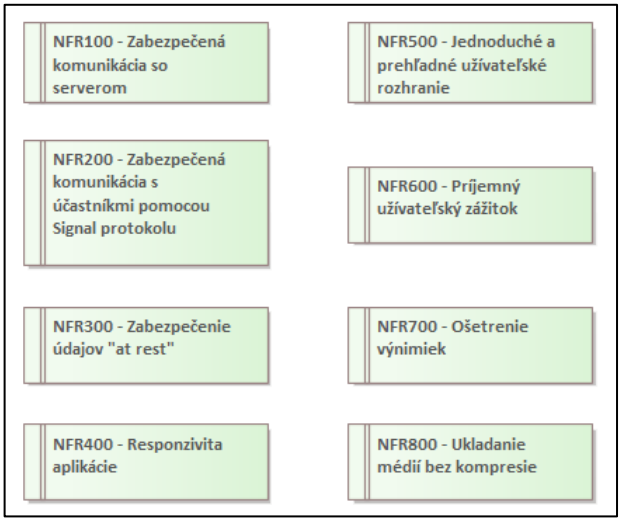
FR1500 – Pridávanie obsahu do galérie - Aplikácia umožní užívateľovi pridať mediálny obsah do galérie zvolenej akcie.

FR1600 – Zobrazenie zoznamu chatov – Aplikácia umožní užívateľovi zobraziť zoznam chatov ku všetkým aktuálnym akciám

FR1700 – Odoslanie správ – Aplikácia umožní užívateľovi odosielať správy v zvolenom chate

4.2 Nefunkcionálne požiadavky

Obrázok 19 zobrazuje všetky nefunkcionálne požiadavky na systém.



Obrázok 19 Nefunkcionálne požiadavky

NFR100 – Zabezpečená komunikácia so serverom – Komunikácia so serverom bude prebiehať prostredníctvom protokolu HTTPS

NFR200 – Zabezpečená komunikácia s účastníkmi pomocou Signal protokolu – Komunikácia formou chatu bude zabezpečená end to end šifrovaním pomocou Signal protokolu

NFR300 – Zabezpečenie údajov „at rest“ – Uložené správy lokálne v zariadení budú zabezpečené šifrovaním.

NFR400 – Responzivita aplikácie – Aplikácia sa bude prispôsobovať zariadeniu pričom nezáleží na veľkosti displeja.

NFR500 – Jednoduché a prehľadné užívateľské rozhranie – Aplikácia obsahuje užívateľské rozhranie, ktoré by malo byť pre užívateľa príjemné a jasné čo aká akcia robí.

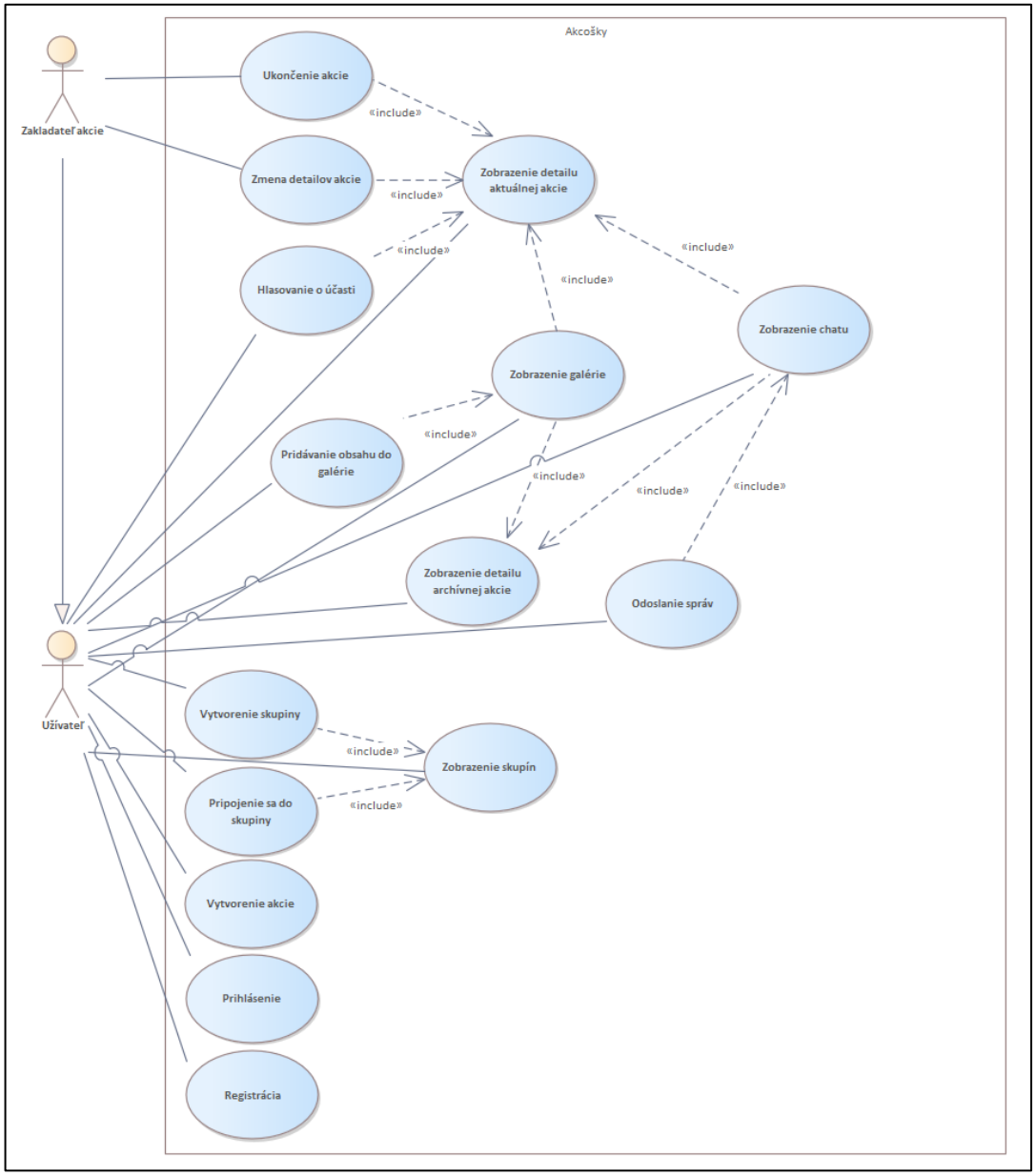
NFR600 – Príjemný užívateľský zážitok – Aplikácia bude rýchla, nebude zamŕzať a v častiach dlhšej komunikácie so serverom bude obsahovať animácie.

NFR700 – Ošetrenie výnimiek – Aplikácia bude ošetrovať výnimky spôsobom, aby užívateľ pochopil z akého dôvodu sa aplikácia dostala do takého stavu.

NFR800 – Ukladanie médií bez kompresie – Aplikácia bude ukladať média na server bez zbytočnej kompresie dát.

4.3 Model prípadov užitia

Obrázok 20 zobrazuje všetky prípady užitia.



Obrázok 20 Model prípadov užitia

4.3.1 Prihlásenie

Prihlásením sa užívateľ dostane do aplikácie. Prihlásením užívateľ overí svoju identitu a server mu na základe toho vie poskytnúť všetky potrebné informácie z databáze.

4.3.1.1 Scenár

1. Užívateľ zapol aplikáciu
2. Systém užívateľa presmeroval na prihlasovaciu stránku
3. Užívateľ vyplnil prihlasovacie meno a heslo
4. Systém overil zadané údaje
5. Systém presmeruje užívateľ-a na hlavnú stránku

Postcondition - Užívateľ je prihlásený

Výnimky

Zlé prihlasovacie údaje

Odklonenie v kroku 4

1. Zadané údaje systém vyhodnotil ako nesprávne
2. Systém užívateľovi zobrazil chybovú hlášku

Pokračovanie v kroku 3

4.3.1.2 Implementácia

Aplikácia overí, že všetky polia sú vyplnené. Následne vytiahne z prihlasovacieho formulára údaje zadané užívateľom. Tieto údaje sú odoslané na backend, ktorý heslo zahashuje a porovná s hashom čo je uložený v databáze. V prípade, že sa hash rovná tak je užívateľ autentifikovaný a backend mu vráti všetky potrebné údaje ako sú: informácie o akciách, informácie o skupinách v ktorých sa nachádza a informácie o účastníkoch/pozvaných na akciu. Taktiež je vrátený aj JWT token, ktorý sa v aplikácií používa ako verifikácia pri ďalších dotazoch na backend. V prípade, že sa hash nerovná tak backend vráti chybu, ktorá sa v aplikácií premietne do chybovej hlášky.

V prípade, že sa užívateľ sa prihlasuje prvý krát sú užívateľovi vygenerované kľúče pre správne fungovanie signal protokolu a tieto kľúče sú vložené do lokálnej databázy a verejné kľúče sú pomocou backendu vložené do databázy.

4.3.2 Registrácia

Aby mohol užívateľ aplikáciu plne využívať tak sa musí zaregistrovať.

4.3.2.1 Scenár

1. Užívateľ klikol na presmerovanie na obrazovku registrácie
2. Systém zobrazí užívateľovi obrazovku s registračným formulárom
3. Užívateľ vyplní všetky potrebné údaje
4. Systém overil rovnosť hesiel
5. Systém odošle na zadaný email užívateľa overovací kód
6. Užívateľ zadá overovací kód
7. Systém overí zadaný overovací kód
8. Systém dovolí užívateľovi dokončiť registráciu
9. Systém užívateľa zaregistruje

Výnimky

Nevyplnené všetky potrebné údaje

Odklonenie v kroku 3

1. Užívateľ nevyplnil všetky potrebné údaje
2. Systém užívateľovi zobrazil chybovú hlášku

Pokračovanie v kroku 3

Heslá sa nezhodujú

Odklonenie v kroku 4

1. Zadané heslá sa nezhodujú
2. Systém užívateľovi zobrazil chybovú hlášku

Neplatný overovací kód

Odklonenie v kroku 7

1. Zadaný kód nie je správny
2. Systém užívateľovi zobrazil chybovú hlášku

4.3.2.2 Implementácia

Aplikácia overí, že všetky polia sú vyplnené a že zadané užívateľské heslá sa zhodujú v oboch poliach. V ďalšom kroku je užívateľovi na email odoslaný overovací kód, že k zadanému emailu má naozaj prístup. Keď sa zadaný kód zhoduje s kódom, ktorý bol vygenerovaný aplikáciou tak sú tieto registračné údaje odoslané na server. Na serveri prebehne hashovanie hesla pomocou hashovacieho algoritmu Argon2. Následne sú všetky údaje uložené do databáze a užívateľ je oboznámený v aplikácií o úspešnej registrácií.

4.3.3 Zobrazenie skupín

Bez skupín toho veľa v aplikácií užívateľ nezmôže. V tomto use case sa užívateľovi zobrazí zoznam jeho skupín, formulár na pripojenie do skupiny a na vytvorenie skupiny.

4.3.3.1 Scenár

Precondition - Užívateľ je prihlásený

1. Užívateľ klikol na záložku skupín
2. Systém užívateľa presmeruje na obrazovku ohľadom skupín
3. Systém užívateľovi zobrazí zoznam skupín v ktorých je členom, formulár na pripojenie sa do skupiny a formulár na vytvorenie skupiny

Výnimky

Užívateľ sa v žiadnej skupine nenachádza

Odklonenie v kroku 4

1. Systém užívateľovi zobrazí informáciu o tom, že sa v žiadnej skupine nenachádza, formulár na pripojenie sa do skupiny a formulár na vytvorenie skupiny

4.3.3.2 Implementácia

Aplikácia užívateľovi vykreslí zoznam skupín v ktorých sa nachádza. Každá skupina tu je reprezentovaná názvom a kódom na pripojenie. V ďalšej časti aplikácia vykreslí formulár na pripojenie do skupiny a taktiež formulár na vytvorenie skupiny.

4.3.4 Vytvorenie skupiny

Skupiny sú potrebné na zaradenie užívateľa do okruhu priateľov. Z týchto skupín sú následne vyberaný užívatelia pri vytváraní akcie. Preto je potrebné aby nejaký užívateľ skupinu priateľov založil.

4.3.4.1 Scenár

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade vytvorenia skupiny
2. <include> Zobrazenie skupín
3. Užívateľ vyplnil názov skupiny
4. Systém overil dostupnosť zvoleného názvu skupiny
5. Systém skupinu vytvoril

Výnimky

Skupina s týmto názvom už existuje

Odklonenie v kroku 4

1. Skupina so zvoleným názvom už existuje
2. Užívateľovi je zobrazená chybová hláška

Pokračovanie v kroku 3

4.3.4.2 Implementácia

Aplikácia overí, že názov skupiny je vyplnený a odošle požiadavku na backend. Backend overí, že skupina s týmto názvom neexistuje. V prípade, že neexistuje tak skupinu vytvorí a backend odošle informáciu o tom, že skupina bola vytvorená a v aplikácií sa táto správa zobrazí formou Snackbaru. Taktiež sa aktualizuje zoznam skupín v ktorých je užívateľ členom. Pri vytvorení skupiny je zakladateľ skupiny automaticky pridaný do skupiny ako člen. V prípade, že existuje tak backend odošle informáciu o tom, že skupina s týmto názvom už existuje a užívateľa informuje chybovou hláškou.

4.3.5 Pripojenie sa do skupiny

Aby mohol byť užívateľ pozvaný na akciu tak sa musí nachádzať v danej skupine. Tam sa pripojí pomocou kódu na pripojenie.

4.3.5.1 Scenár

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade pripojenia do skupiny
2. <include> Zobrazenie skupín
3. Užívateľ vyplnil kód na pripojenie do skupiny
4. Systém overil či skupiny s takýmto kódom na pripojenie existuje
5. Systém užívateľa pripojil do skupiny

Výnimky

Neexistujúca skupina s týmto kódom na pripojenie

Odklonenie v kroku 4

1. Skupina s takýmto kódom na pripojenie neexistuje
2. Užívateľovi je zobrazená chybová hláška

Pokračovanie v kroku 3

4.3.5.2 Implementácia

Aplikácia overí, že kód na pripojenie do skupiny je vyplnený a odošle požiadavku na backend. Backend overí, že skupina s takýmto kódom na pripojenie existuje. V prípade, že existuje tak užívateľa do skupiny pripojí a backend odošle informáciu o tom, že užívateľ bol do skupiny pripojení a táto správa sa v aplikácií zobrazí formou Snackbaru. Taktiež sa aktualizuje zoznam skupín v ktorých je užívateľ členom. V prípade, že neexistuje tak backend odošle informáciu o tom, že skupina s takýmto kódom na pripojenie neexistuje.

4.3.6 Zobrazenie detailu aktuálnej akcie

V detaile o akcií užívateľ vidí všetky potrebné informácie o akcií ako názov, popis, dátum a čas konania, účastníkov a aj možnosť hlasovania o zúčastnení.

4.3.6.1 Scenár

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade zobrazenia detailu akcie
2. Užívateľ je účastníkom (alebo bol pozvaný) na akciu
3. Systém užívateľovi zobrazí zoznam akcií

4. Užívateľ klikol v zozname akcií na požadovanú akciu
5. Systém vráti užívateľovi obrazovku s detailnými informáciami ku konkrétnej akcií

Výnimky

Žiadne akcie

1. Užívateľ nie je účastníkom a ani nebol pozvaný na žiadnu akciu
2. Systém užívateľ-ovi zobrazí hlášku, že v žiadnej akcií sa nenachádza a navrhne mu, aby nejakú akciu vytvoril

4.3.6.2 Implementácia

Najskôr sa užívateľovi zobrazia akcie formou kariet v zozname v časovej osi, kde sa nachádza informácia o roku konania. Každá karta obsahuje najdôležitejšie informácie o konkrétnej akcií a to je názov, typ, miesto konania, počet účastníkov z celkového počtu pozvaných a dátum a čas konania. Časť karty kde sa nachádza dátum a čas konania sa môže meniť, pretože akcia môže byť jednodňová alebo viacdňová. V prípade viacdňovej akcie aplikácia vykreslí rozsah dátumov kedy sa akcia koná. Po kliknutí na požadovanú akciu následne aplikácia užívateľovi vykreslí obrazovku so všetkými informáciami prehľadnou formou. V prípade, že užívateľ konkrétnu akciu vytvoril tak pri každej informácií systém vykreslí tlačidlo pomocou ktorého môže konkrétnu informáciu editovať.

4.3.7 Ukončenie akcie

Po úspešnom ukončení akcie fyzicky alebo neúspešnom zorganizovaní akcie je treba akciu ukončiť v aplikácií. Ukončenie akcie môže prebehnúť dvomi spôsobmi: vymazaním akcie z databázy alebo presunutím akcie do archívu.

4.3.7.1 Scenár

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade podnetu na ukončenie akcie
2. Systém zobrazí obrazovku detailu akcie
3. <include> Zobrazenie detailu aktuálnej akcie
4. Zakladateľ akcie klikne na tlačidlo ukončenia akcie
5. Systém zobrazí vyskakovacie okno s typmi ukončenia akcie
6. Zakladateľ akcie zvolí možnosť ukončenia akcie "Presunutím do archívu"
7. Systém požiadavku spracuje

8. Systém vybranú akciu v UI presunie do archivovaných správ

Alternatíva

Ukončenie akcie vymazaním z databáze

Odklonenie v kroku 6

1. Zakladateľ akcie zvolí možnosť ukončenia akcie "Vymazaním z databázy"
2. Systém požiadavku spracuje
3. Systém vybranú akciu v UI zmaže z listu akcií

4.3.7.2 Implementácia

Po kliknutí na tlačidlo „Ukončiť akciu“ užívateľovi vyskočí vyskakovacie okno v ktorom sa môže rozhodnúť či chce akciu ukončiť vymazaním z databázy alebo presunutím do archívu.

Pri možnosti vymazania z databázy je na backend odoslaná požiadavky aby akciu vymazal z databázy. Backend vymaže akciu z databázy ako aj informácie o hlasovaní k tejto akcií. Taktiež vymaže všetky médiá ku zvolenej akcii zo servera v prípade, že sa tam nejaké nachádzajú. Následne je užívateľ presmerovaný na zoznam aktuálnych akcií a zvolená akcia je vymazaná aj z UI.

Pri možnosti archivovania akcie je na backend odoslaná požiadavka o uchovanie akcie. Backend označí akciu v databáze ako neaktívnu. Následne je užívateľ presmerovaný na zoznam aktuálnych akcií a zvolená akcia je presunutá do sekcie Archív.

4.3.8 Zmena detailov akcie

Zakladateľ má možnosť zmeniť detaily akcie keď to je potrebné.

4.3.8.1 Scenár

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade podnetu na zmenu detailov akcie
2. Systém zobrazí obrazovku detailu akcie
3. <include> Zobrazenie detailu aktuálnej akcie
4. Zakladateľ akcie klikne na možnosť editovania konkrétnej informácie o akcií
5. Systém aktualizuje rozhranie detailu pre možnosť zmeny informácie
6. Zakladateľ akcie aktualizuje zvolenú informáciu
7. Zakladateľ akcie odošle zmenu informácie/í

8. Systém požiadavku spracuje

Alternatíva

Zmena ďalšej informácie

Odklonenie v kroku 7

1. Zakladateľ akcie sa rozhodol pre zmenu ďalšej informácie

Pokračovanie v kroku 5

4.3.8.2 Implementácia

Keď užívateľ vyplní požadované polia na zmenu a odošle požiadavku tak aplikácia požiadavku spracuje a odošle ju na backend. Backend zmení požadované informácie v databáze. Následne sa požadované zmeny aktualizujú aj v aplikácií.

4.3.9 Hlasovanie o účasti

Pre informáciu užívateľa o tom, kto sa danej akcie zúčastní musí existovať spôsob aby ostatní užívatelia mohli zmeniť svoj status hlasovania.

4.3.9.1 Scenár

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade podnetu na zmenu hlasu o účasti na akcií
2. Systém zobrazí obrazovku detailu akcie
3. <include> Zobrazenie detailu aktuálnej akcie
4. Užívateľ klikne na tlačidlo na možnosť ÁNO/NIE v časti obrazovky ohľadom účasti na akcií
5. Systém požiadavku spracuje
6. Systém aktualizuje užívateľské rozhranie s aktualizovanou účasťou, ktorá bola zakliknutá

4.3.9.2 Implementácia

Aplikácia zaznamená užívateľovu žiadosť o zmenu hlasu v konkrétnej akcií. Aplikácia odošle požiadavku na zmenu hlasu na backend. Backend zmenu hlasu zaznamená do databáze. V prípade úspešnej zmeny sa aktualizuje užívateľské rozhranie so zvoleným hlasom.

4.3.10 Zobrazenie galérie

V galérií užívateľ vidí všetky fotografie a videá, ktoré pridali účastníci konkrétnej akcie.

4.3.10.1 Scenár

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade podnetu na zobrazenie galérie
2. Zvolená akcia je aktuálna
3. Systém zobrazí obrazovku detailu akcie
4. <include> Zobrazenie detailu aktuálnej akcie
5. Užívateľ klikol na tlačidlo galérie
6. Systém presmeruje užívateľa na obrazovku s galériou konkrétnej akcie
7. Systém získa zo servera všetky médiá spojené s konkrétnou akciou
8. Systém užívateľovi všetky získané média vykreslí formou galérie

Alternatíva

Zvolená akcia nie je aktuálna

Odklonenie v kroku 2

1. <include> Zobrazenie detailu archívnej akcie

Pokračovanie v kroku 5

4.3.10.2 Implementácia

Aplikácia odošle požiadavku na backend na získanie všetkých fotografií a videí ku zvolenej akcií. Backend vráti vygenerované URL adresy pomocou ktorých aplikácia stiahne priamo všetky médiá. V aplikácií sa spustí niekoľko paralelných úloh (maximálny počet jadier procesora mínus 2) a každý začne sťahovať médiá z úložiska pomocou URL adries, ktoré aplikácia dostala z backendu. Spočiatku sa vykreslia indikátory sťahovania a keď sa obrázok alebo video stiahne tak nahradí indikátor korešpondujúci s indexom obrázka/videa.

4.3.11 Pridávanie obsahu do galérie

Na to aby mohol užívateľ vidieť nejaké multimédia tak ich najskôr musí niekto pridať.

4.3.11.1 Scenár

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade podnetu pridania obsahu do galérie
2. Systém zobrazí obrazovku s galériou zvolenej akcie
3. <include> Zobrazenie galérie
4. Užívateľ klikol na tlačidlo pridania obsahu do galérie
5. Systém užívateľovi otvorí dialógové okno výberu podporovaného obsahu
6. Užívateľ zvolí obsah, ktorý chce nahrať
7. Systém spustí službu na popredí, ktorá nahrá zvolený obsah na server

4.3.11.2 Implementácia

Aplikácia zobrazí užívateľovi dialóg pre výber multimédií, ktoré chce nahrať. Po zvolení sa spustí služba na popredí. Využitie tejto služby je potrebné z toho dôvodu, že multimédiá ktoré chce užívateľ nahrať neprechádzajú žiadnou kompresiou a taktiež nie je limitovaný počet multimédií, ktoré môže užívateľ nahrať. Veľkosť týchto súborov preto môže byť dosť veľká. Pre vhodnú užívateľskú skúsenosť tieto multimédiá sa nebudú nahrávať prostredníctvom aplikácie pretože by musela byť zapnutá dokedy sa všetko neodošle. Pre tento účel slúži spomínaná služba na popredí. Keď táto služba na popredí spustí nahrávanie multimédií tak užívateľ môže pokojne aplikáciu vypnúť a nahrávanie bude pokračovať pretože táto služba na popredí pracuje na systémovej úrovni.

V prípade, že užívateľ nechá aplikáciu zapnutú tak postupne ako sa médiá nahrávajú tak sa mu zobrazujú v galérií.

4.3.12 Zobrazenie detailu archívnej akcie

Detail archívnej akcie obsahuje všetky informácie ako aj detail aktuálnej akcie s tým rozdielom, že v archívnej akcie je zakázané hlasovanie a taktiež nemôže zakladateľ akcie meniť žiadne informácie o akcií.

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade zobrazenia detailu archivovanej akcie
2. Užívateľ klikol na záložku archivovaných akcií
3. Systém získa všetky akcie ktorých sa užívateľ zúčastnil a už sú ukončené
4. Systém užívateľovi zobrazí zoznam týchto akcií
5. Užívateľ klikol v zozname akcií na požadovanú akciu
6. Systém vráti užívateľovi obrazovku s detailnými informáciami ku konkrétnej akcií

Výnimky

Žiadna akcia

Odklonenie v kroku 3

1. Užívateľ nebol účastníkom žiadnej akcie
2. Systém užívateľovi zobrazí hlášku, že v archíve sa žiadne akcie nenachádzajú

4.3.13 Vytvorenie akcie

V rámci vytvorenia akcie užívateľ vyplní všetky potrebné údaje ako sú názov, popis/plán, typ, miesto konania a dátum spolu s časom. V ďalšom kroku užívateľ vyberie skupinu a užívateľov so zvolenej skupiny ktorých chce na akciu pozvať. V poslednom kroku môže zvoliť voliteľné informácie ako ubytovanie, transport a odhadovaná cena akcie.

4.3.13.1 Scenár

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade podnetu vytvorenia novej akcie
2. Užívateľ klikol na záložku vytvorenia akcie
3. Systém užívateľa presmeruje na formulár vytvorenia akcie
4. Užívateľ vyplní všetky potrebné základné informácie
5. Užívateľ prejde na ďalší krok
6. Systém zobrazí užívateľovi obrazovku s pozvaním ľudí na akciu
7. Užívateľ zvolí skupinu z ktorej chce pozvať ľudí na akciu
8. Systém zobrazí užívateľovi ľudí z vybratej skupiny
9. Užívateľ vyberie ľudí, ktorých chce na akciu pozvať
10. Užívateľ odošle vytvorenie akcie
11. Systém spracuje požiadavku vytvorenia akcie
12. Systém presmeruje užívateľ-a na hlavnú obrazovku a aktualizuje zoznam akcií

Alternatíva

Dodatočné informácie

Odklonenie v kroku 10

1. Užívateľ doplní dodatočné informácie ohľadom akcie

Pokračovanie v kroku 11

4.3.13.2 Implementácia

Aplikácia overí správnosť vyplnenia požadovaných údajov o akcií. V prípade, že ich užívateľ nevyplní tak mu systém neumožní prejsť do ďalšieho kroku. Následne aplikácia získa z frontendu užívateľov ktorý majú byť pozvaný na túto akciu. V prípade, že v ďalšom kroku užívateľ vyplnil voliteľné polia tak ich aplikácia získa. V túto chvíľu má aplikácia všetky informácie o novej akcií a odošle požiadavku na backend. Backend akciu vytvorí v databáze a všetkým pozvaným užívateľom odošle informáciu o novo vytvorenej akciu formou push notifikácie.

4.3.14 Zobrazenie chatu

Pre zobrazenie správ a pre možnosť vytvorenia a odoslania správy sa najskôr musí konkrétny chat vykresliť.

4.3.14.1 Scenár

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade podnetu na zobrazenie chatu
2. Užívateľ klikol na záložku chatov
3. Systém užívateľovi zobrazí chat list so všetkými aktuálnym akciám
4. Užívateľ zvolí chatovú miestnosť, ktorú chce zobraziť
5. Systém užívateľa presmeruje na obrazovku chatu ku zvolenej akcií
6. Systém získa správy z databázy, ktoré užívateľovi prišli keď bol offline
7. Systém získa správy z lokálnej databázy
8. Systém má aktuálne správy a zobrazí ich užívateľovi

Alternatíva

Užívateľ klikol na tlačidlo Prejdi na chat v detaile aktuálnej akcie

Odklonenie v kroku 2

1. <include> Zobrazenie detailu aktuálnej akcie

Pokračovanie v kroku 5

Užívateľ klikol na tlačidlo Prejdi na chat v detaile archívnej akcie

Odklonenie v kroku 2

1. <include> Zobrazenie detailu archívnej akcie

Pokračovanie v kroku 5

Výnimky

Systém žiadne správy nemá

Odklonenie v kroku 8

1. Systém zobrazí užívateľovi predvolenú stránku chatu

4.3.14.2 Implementácia

Systém získa z databázy offline správy, ktoré užívateľ dostal keď bol nedostupný. Následne z lokálnej databázy vytiahne staré správy. Tieto správy vykreslí v poradí v akom prišli na obrazovku.

V neposlednom rade systém vytvorí end to end reláciu s každým účastníkom akcie pomocou Signal protokolu. Taktiež ak má užívateľ uložené relácie s ostatnými účastníkmi lokálne tak tieto načíta aby bolo end to end šifrovanie neporušené a aby to fungovalo v poriadku.

4.3.15 Odoslanie správ

Pre komunikáciu s ostatnými účastníkmi akcie je funkcia odosielania správ nutnosť.

4.3.15.1 Scenár

Precondition - Užívateľ je prihlásený

1. Scenár sa spustí v prípade podnetu na odoslanie správy
2. Systém zobrazí obrazovku zvoleného chatu
3. <include> Zobrazenie chatu
4. Užívateľ napíše správu a odošle ju
5. Systém požiadavku spracuje

4.3.15.2 Implementácia

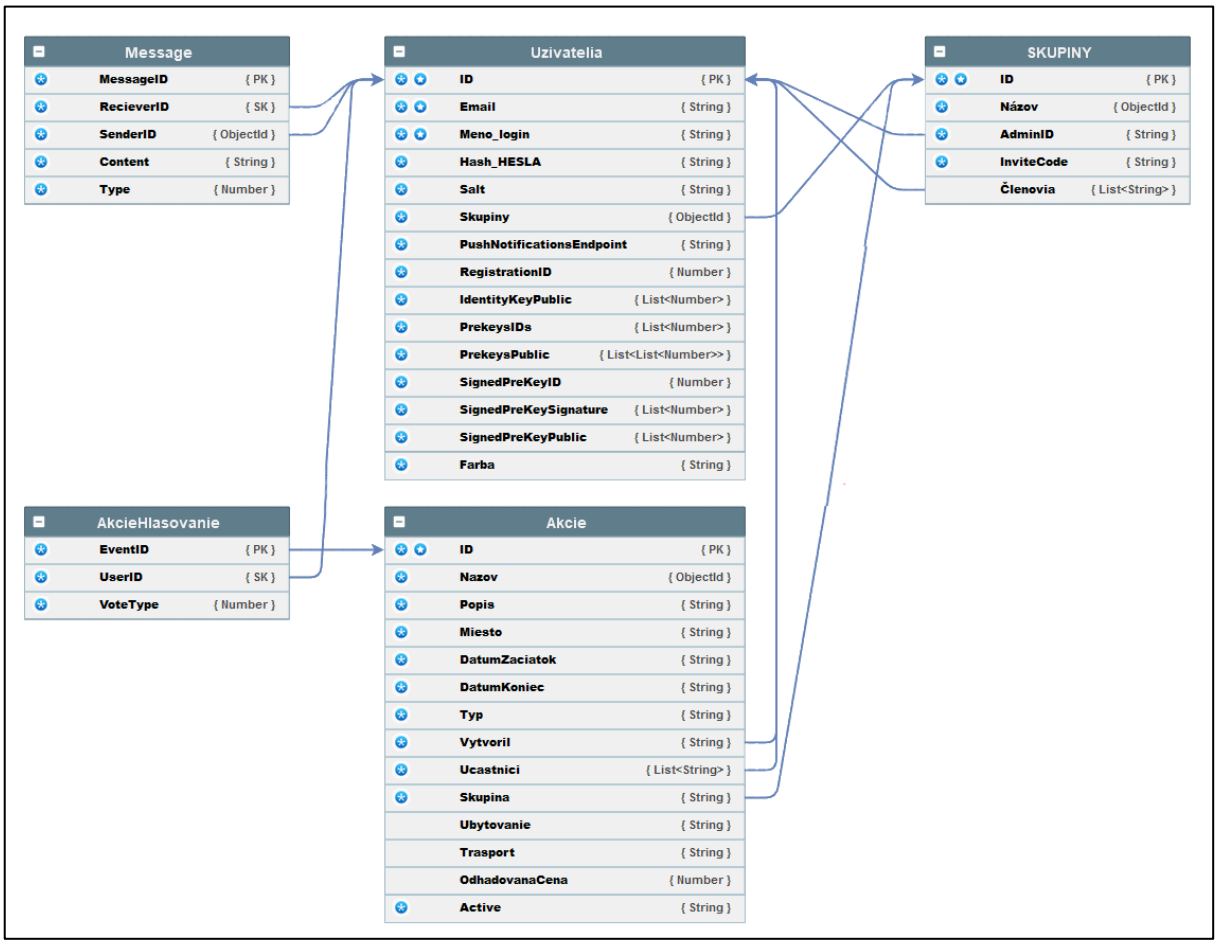
Aplikácia z formulára na odoslanie správy získa obsah správy. Pomocou existujúcich relácií aplikácia zašifruje a odošle správy všetkým účastníkom skupiny. Taktiež aplikácia aktualizuje lokálne uloženú informáciu o end to end relácií s konkrétnym účastníkom.

5 DATABÁZY

V tejto kapitole sa nachádzajú vyobrazené modely všetkých databáz s ktorými aplikácia pracuje.

5.1 DynamoDB

Obrázok 21 znázorňuje tabuľky v databáze DynamoDB a prepojenia medzi nimi. Viac informácií o databáze DynamoDB sa nachádza v kapitole 6.6 Amazon DynamoDB.



Obrázok 21 Model databáze DynamoDB

V tejto databáze sa používajú 2 typy primárnych kľúčov:

Partition key (PK) - Jednoduchý primárny kľúč, ktorý sa skladá z jedného atribútu známeho ako *partition key.* [31]

Partition key (PK) and sort key (SK) - Tento typ kľúča, označovaný ako zložený primárny kľúč, sa skladá z dvoch atribútov. Prvým atribútom je *partition key* a druhým atribútom je *sort key*. [31]

5.1.1 Tabuľka Užívatelia

Tabuľka Užívatelia slúži na ukladanie všetkých informácií o užívateľoch. Pole ID je jedinečný identifikátor každého užívateľa. Ďalej sa v tejto tabuľke nachádzajú základné údaje o užívateľovi ako sú *Email*, *Meno_login*, *Hash_HESLA* a *Salt*. List Skupiny obsahuje identifikátory skupín, v ktorých sa užívateľ nachádza. Pole *PushNotificationEndpoint* je *Amazon Resource Names (ARN),* ktorý slúži ako odkaz na konkrétny endpoint v službe Amazon SNS. Pole Farba obsahuje farbu v hexadecimálnom formáte, ktorou bude užívateľ mať v chate zvýraznené meno. Všetky zvyšné polia slúžia pre Signal protokol na vytváranie relácií medzi užívateľmi a následné šifrovanie správ.

5.1.2 Tabuľka Message

Tabuľka Message slúži na ukladanie chatovacích správ. Najdôležitejšie z tejto tabuľky je pole *Content* pretože obsahuje text správy, ktorá je zašifrovaná pomocou Signal protokolu. Pole *RecieverID* sa používajú pre doručenie správy správnej osobe a zaradenie do správnej a pole *SenderID* sa používa na identifikáciu odosielateľa pre použitie správnych kľúčov na odšifrovanie správy. Pole Type používa Signal protokol na určenie typu správy a zvolenie typu odšifrovania.

5.1.3 Tabuľka Skupiny

Tabuľka Skupiny slúži na ukladanie informácií o skupinách. Pole *ID*, ktoré slúži ako jedinečný identifikátor skupiny. Taktiež sa tu nachádzajú polia s informáciami o skupine ako sú *Názov*, *AdminID* – identifikátor Admina skupiny, *InviteCode* – kód, ktorý slúži na pripojenie užívateľa do skupiny. Posledné pole je *Členovia* a to slúži na informáciu, ktorí užívatelia sa v danej skupine nachádzajú.

5.1.4 Tabuľka Akcie

Tabuľka Akcie slúži na uloženie informácií o akciách. Obsahuje ID - jedinečný identifikátor akcie, Názov – názov akcie, Popis – podrobné informácie o akcie, Miesto – miesto kde sa akcia koná, DatumZaciatok – dátum začiatku akcie, DatumKoniec – v prípade viacdňových akcií obsahuje akcia aj dátum konca akcie, Typ – typ akcie slúži pre frontend pre vykreslenie ikony pre daný typ a Vytvoril – ktorý užívateľ akciu vytvoril.

Polia, ktoré sú nepovinné ale môžu upresniť organizáciu akcie sú Ubytovanie – ubytovanie kde budú účastníci akcie ubytovaní, Transport – akým spôsobom sa na akciu dopravia účastníci a OdhadovanaCena – na aký obnos peňazí môže akcia vyjsť.

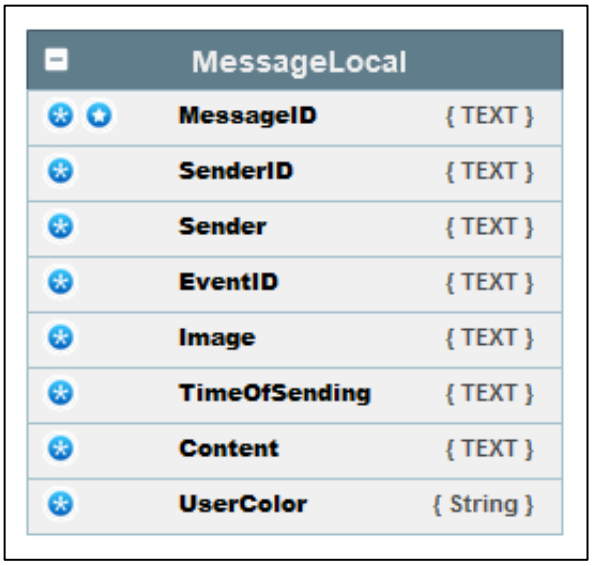
Taktiež sa tu nachádzajú polia *Ucastnici* a *Skupina*. *Ucastnici* obsahuje ID všetkých užívateľov, ktorí boli na akciu pozvaní a *Skupina* je ID skupiny s ktorou je akcia asociovaná.

5.1.5 Tabuľka AkcieHlasovanie

Tabuľka AkcieHlasovanie slúži na uloženie hlasovania o účasti na akcií. Tabuľka obsahuje *EventID*, *UserID* a *VoteType*. *EventID* obsahuje ID konkrétnej akcie, *UserID* je ID konkrétneho užívateľa a *VoteType* je informácia o tom ako pozvaný užívateľ zahlasoval či sa konkrétnej akcie zúčastní.

5.2 Sqlite

Obrázok 22 znázorňuje tabuľku v databáze Sqlite.

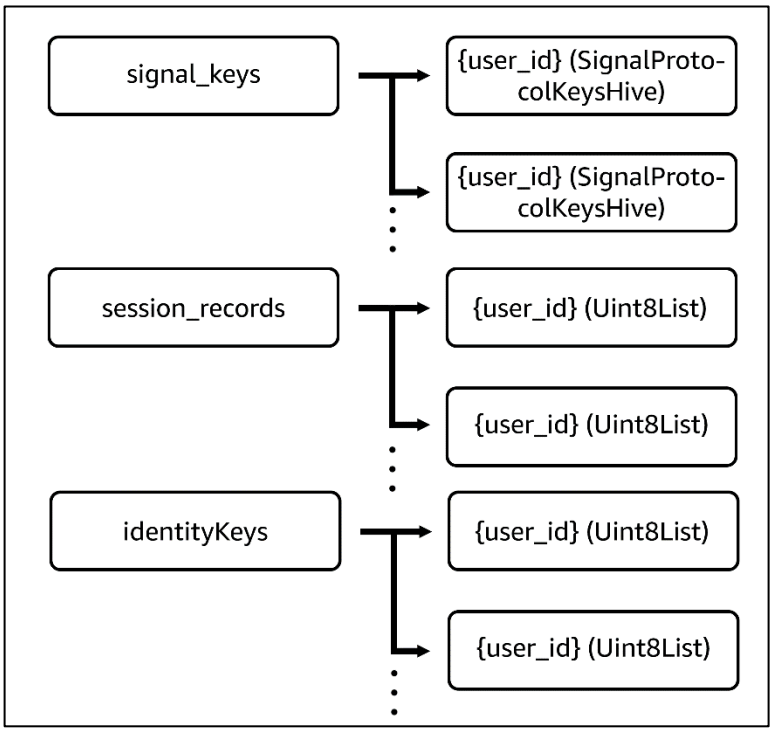


Obrázok 22 Model databáze Sqlite

Táto databáza je veľmi jednoduchí a slúži na ukladanie chatovacích správ lokálne v zariadení. Keď užívateľ stiahne správu z databázy DynamoDB táto správa je z tejto databáze odstránená, je dešifrovaná pomocou Signal protokolu a následne je vložená do tejto tabuľky MessageLocal v Sqlite. Táto databáza je šifrovaná pomocou AES 256.

5.3 Hive

Obrázok 23 znázorňuje akým spôsobom je využívaná databáza Hive lokálne v aplikácií.



Obrázok 23 Model databáze Hive

Hive je odľahčená a veľmi rýchla databáza kľúč-hodnota napísaná v jazyku Dart. [32]

Hive organizuje dáta do boxov. V tomto modeli sú využité dva boxy – signal_keys a session_records.

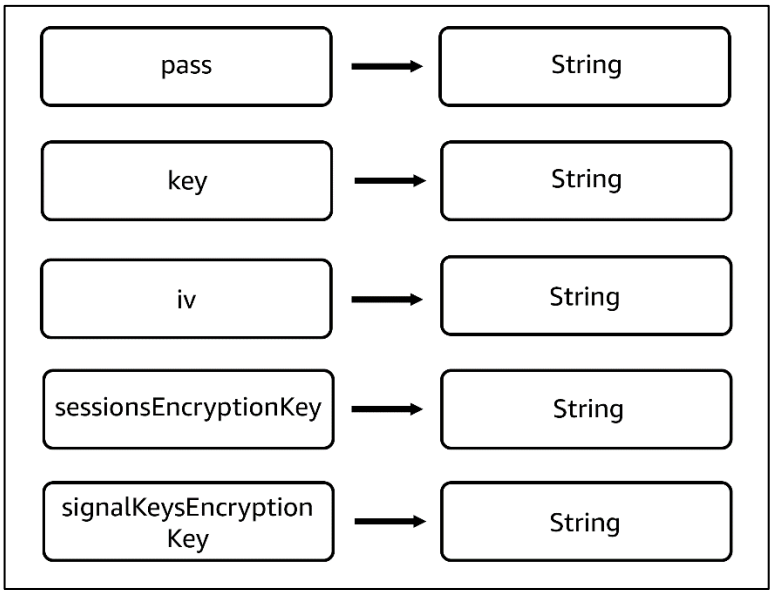
Signal_keys obsahuje lokálne uložené všetky potrebné kľúče s ktorými pracuje Signal protokol. V tomto boxe sa nachádzajú dáta typu SignalProtocolKeysHive – je to objekt. Pomocou adaptéru, ktorý sa vygeneruje je ale databáza Hive schopná uložiť aj takýto objekt.

Session_records obsahuje lokálne uložené informácie o reláciách, ktoré má užívateľ nadviazané s ostatnými používateľmi. Tieto informácie sa ukladajú aby Signal protokol vedel po prijatí novej správy pokračovať v nadviazanej relácií.

Tieto boxy sú taktiež samozrejme šifrované.

5.4 Flutter Secure Storage

Obrázok 24 znázorňuje akým spôsobom je v aplikácií využitý Flutter Secure Storage a aké konkrétne hodnoty sa v ňom ukladajú.



Obrázok 24 Model „databáze“ Flutter Secure Storage

Všetky hodnoty, ktoré sú vo Flutter Secure Storage uložené slúžia ako šifrovacie kľúče pre ostatné databázy. Pass, key a iv slúžia pre šifrovanie lokálnej databázy Sqlite, sessionsEncryptionKey slúži na šifrovanie Hive boxu kde sa nachádzajú relácie a signalKeysEncryptionKey slúži na šifrovanie Hive boxu kde sa nachádzajú Signal kľúče.

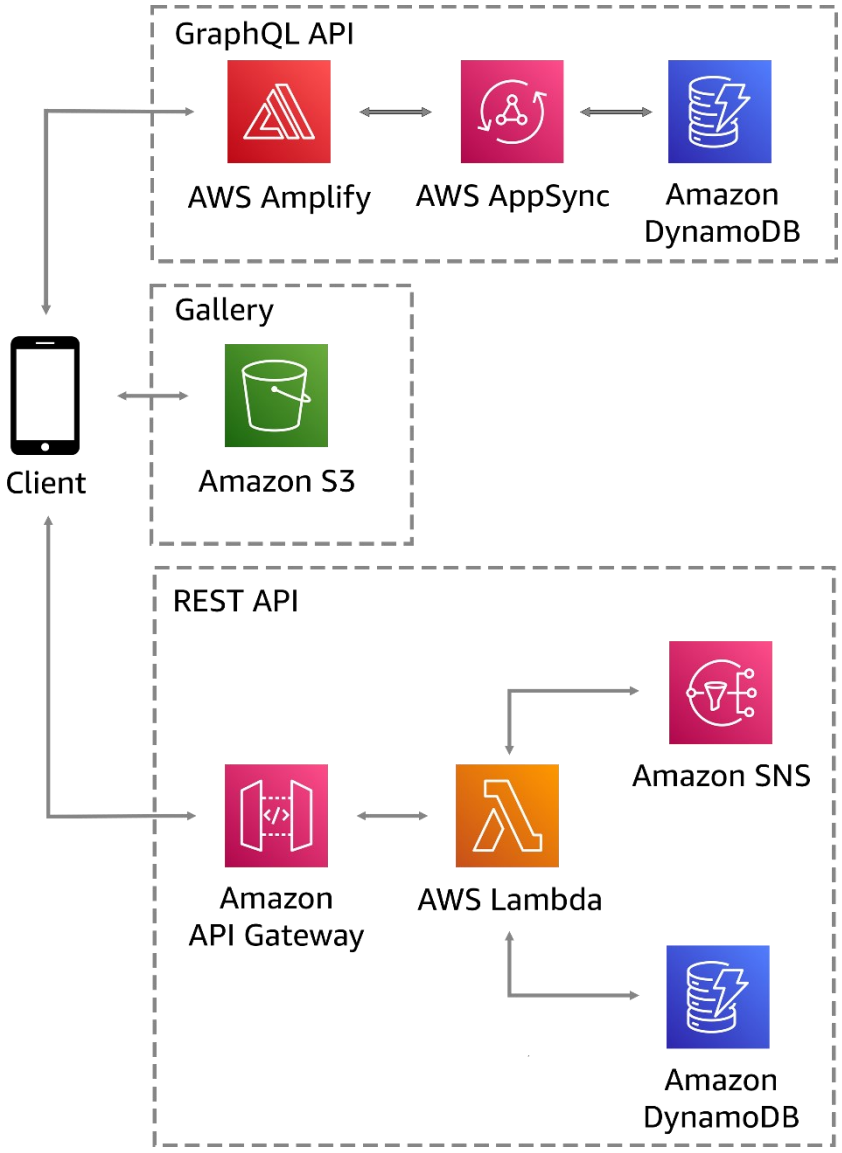
Dôvod výberu tohto riešenia pre ukladanie šifrovacích kľúčov pre ostatné databázy je jednoduchý a tým je bezpečnosť. Viac informácií o bezpečnosti Flutter Secure Storage sa nachádza v kapitole 9.1.1 Plugin Flutter Secure Storage.

6 SLUŽBY POUŽITÉ NA BACKENDE

Aplikácia je realizovaná formou tzv. serverless architektúry. Ako poskytovateľa služieb pre túto serverless architektúru bola vybraná platforma Amazon Web Services (AWS).

Serverless architektúra alebo architektúra bez servera je spôsob ako vytvárať a spúšťať aplikácie a služby bez nutnosti spravovať infraštruktúru. Aplikácia stále beží na serveroch, ale všetku správu serverov vykonáva AWS. Nie je potreba zabezpečovať, škálovať a udržiavať servery na prevádzku aplikácií, databáz a systémov ukladania. [33]

Obrázok 25 zobrazuje túto architektúru a konkrétne služby, ktoré boli použité.



Obrázok 25 AWS Architektúra

V ďalších riadkoch si rozoberieme charakteristiku jednotlivých služieb a taktiež to akým spôsobom jednotlivé služby využíva aplikácia.

6.1 AWS Amplify

AWS Amplify obsahuje súbor nástrojov a služieb na urýchlenie vývoja mobilných a webových aplikácií na platforme AWS. Amplify obsahuje sadu knižníc, UI prvky a príkazový riadok pre vytvorenie backendu aplikácie a jeho integráciu s aplikáciami pre iOS, Android, web, React Native a aj Flutter. [34]

Využitie

V tejto aplikácií bol dôvod použitia tejto služby komunikácia so službou AWS AppSync. Pretože komunikácia s touto platformou prostredníctvom frameworku Flutter mimo použitia Amplify nie je nijako zdokumentovaná, rozhodol som sa pre použitie tejto služby.

6.2 AWS AppSync

AWS AppSync poskytuje robustné, škálovateľné rozhranie GraphQL pre vývojárov aplikácií na kombinovanie údajov z viacerých zdrojov vrátane Amazon DynamoDB, AWS Lambda a HTTP API. [35]

Využitie

Dôvod využitia tejto služby je využitie GraphQL pre potreby funkcie chatu v aplikácií. Komunikácia v reálnom čase je realizovaná pomocou GraphQL, ktoré táto služba ponúka.

Užívateľ, ktorý odosiela správu využíva „mutation“ pre vytvorenie správy. GraphQL túto požiadavku spracuje a správu uloží do databázy. V prípade že užívateľ, ktorému posielame správu má zapnutú aplikáciu tak má vytvorený „subscription“ na GraphQL. Správu, ktorá mu bola odoslaná prijme ihneď v reálnom čase.

6.3 Amazon API Gateway

Amazon API Gateway je služba AWS na vytváranie, publikovanie, údržbu, monitorovanie a zabezpečenie rozhraní API REST, HTTP a WebSocket v ľubovoľnom rozsahu. Vývojári API môžu vytvárať API, ktoré pristupujú k AWS alebo iným webovým službám. [36]

Využitie

Aplikácia využíva túto službu pre komunikáciu s jednotlivými AWS Lambda funkciami, ktoré sú pre vonkajší svet vystavené práve prostredníctvom API Gateway.

6.4 AWS Lambda

AWS Lambda je výpočtová služba, ktorá umožňuje spúšťať kód bez poskytovania alebo správy serverov. Lambda spúšťa kód na vysoko dostupnej výpočtovej infraštruktúre a vykonáva všetku správu výpočtových zdrojov vrátane údržby serverov a operačných systémov, poskytovania kapacity a automatického škálovania a protokolovania. Pomocou Lambda je možné spustiť kód prakticky pre akýkoľvek typ aplikácie alebo backendovej služby. Stačí dodať kód v jednom z jazykov, ktoré Lambda podporuje. [37]

Využitie

AWS Lambda je jednou z najdôležitejších funkcií, ktorá je použitá v aplikácií. Celý backend kód je rozdelený do Lambda funkcií a tieto zabezpečujú chod aplikácie.

6.5 Amazon SNS

Amazon Simple Notification Service (Amazon SNS) je služba, ktorá zabezpečuje doručovanie správ od vydavateľov k účastníkom (známym aj ako producenti a konzumenti). Vydavatelia komunikujú s odberateľmi asynchrónne zasielaním správ na tému, ktorá je logickým prístupovým bodom a komunikačným kanálom. Odberatelia sa môžu prihlásiť k téme SNS a prijímať publikované správy pomocou podporovaného typu koncového bodu, napríklad Amazon Kinesis Data Firehose, Amazon Simple Queue Service, AWS Lambda, HTTP, e-mail, mobilné push notifikácie a mobilné textové správy (SMS). [38]

Využitie

Prostredníctvom služby Amazon SNS sú koncovým užívateľom doručované push notifikácie s rôznymi správami vtedy, kedy je to potrebné.

6.6 Amazon DynamoDB

Amazon DynamoDB je plne spravovaná NoSQL databáza, ktorá poskytuje rýchly a predvídateľný výkon s bezproblémovou škálovateľnosťou. DynamoDB umožňuje odbremenenie od administratívnej záťaže spojenej s prevádzkou a škálovaním distribuovanej databázy, takže nie je potreba sa starať o zabezpečenie hardvéru, nastavenie a konfiguráciu, replikáciu, opravu softvéru alebo škálovanie klastra. DynamoDB ponúka aj „encryption at rest“, ktoré eliminuje prevádzkovú záťaž a zložitosť spojenú s ochranou citlivých údajov. [39]

Využitie

Amazon DynamoDB je tiež veľmi dôležitá služba bez ktorej by chod aplikácie nebol možný pretože zabezpečuje ukladanie všetkých dát, s ktorými aplikácia pracuje.

6.7 Amazon S3

Služba Amazon Simple Storage Service (Amazon S3) je služba objektového úložiska, ktorá ponúka špičkovú škálovateľnosť, dostupnosť údajov, bezpečnosť a výkon. Zákazníci všetkých veľkostí a odvetví môžu využívať službu Amazon S3 na ukladanie a ochranu ľubovoľného množstva údajov pre rôzne prípady použitia, ako sú napríklad webové stránky, mobilné aplikácie, zálohovanie a obnova, archív, analýza veľkých objemov údajov atď. Amazon S3 poskytuje funkcie na manažovanie, takže je možné optimalizovať, organizovať a konfigurovať prístup k údajom tak, aby spĺňali špecifické obchodné, organizačné požiadavky. [40]

Využitie

Táto služba je využívaná hlavne pre ukladanie médií, ktoré užívatelia nahrajú ku konkrétnym akciám.

Taktiež je táto služby využívaná Lambda funkciami pretože ukladá celý backend s ktorým jednotlivé Lambda funkcie pracujú.

7 PREHĽAD UŽÍVATEĽSKÉHO ROZHRANIA APLIKÁCIE

Užívateľské rozhranie bude jednoduché a prehľadné. Taktiež všetky tlačidlá budú jasne označené a bude z neho jednoduché vydedukovať čo sa stane keď užívateľ naň klikne. Wireframy využité v nasledujúcich kapitolách slúžia ako základ užívateľského rozhrania a niektoré veci sa môžu zmeniť prípadne môžu byť ďalšie widgety pridané vzhľadom na zmenu požiadaviek počas písania práce.

7.1 Prihlásenie, Registrácia, Overenie

Tieto tri obrazovky obsahujú rovnaký jednoduchý dizajn, kde užívateľ musí vyplniť potrebné formulárové polia.

Tlačidlom na obrazovke Prihlásenie sa aplikácia pokúsi užívateľa prihlásiť. Pomocou linku „Vytvor si ho“ sa užívateľ z obrazovky Prihlásenie dostane na obrazovku Registrácia.

Tlačidlom na obrazovke Registrácia sa užívateľ dostane na obrazovku Overenie odkiaľ sa tlačidlom po úspešnom overení dostane späť na prihlásenie.

Wireframe k týmto obrazovkám sa nachádza na Obrázok 26.

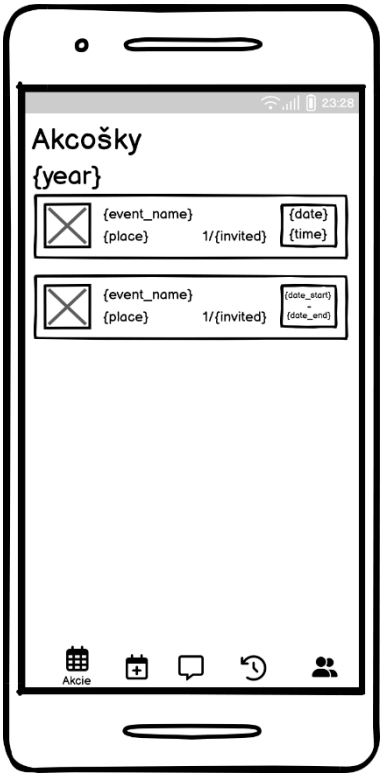
Obrázok 26 Wireframe pre Prihlásenie, Registráciu a Overenie

7.2 Akcie

Obrazovka Akcie je hlavnou obrazovkou aplikácie, ktorá sa užívateľovi zobrazí po prihlásení. Táto obrazovka obsahuje navigačné menu a zoznam akcií. Akcie sú zoskupené podľa roku v ktorom sa akcia koná. Jedna karta akcie obsahuje ikonu podľa typu akcie, názov akcie, miesto konania, ukazateľ koľko pozvaných ľudí sa akcie zúčastní (zahlasovalo áno) a informácie o dátume konania, ktorý môže byť v dvoch formátoch:

- Jednodňová akcia – zobrazuje dátum a čas akcie
- Viacdňová akcia – zobrazuje dátum začiatku a dátum konca akcie

Obrázok 27 zobrazuje wireframe k zoznamu akcií.



Obrázok 27 Wireframe Akcie

7.3 Detail akcie

Obrazovka Detail akcie je prístupná keď užívateľ klikne na vybranú akciu na obrazovke Akcie.

Užívateľovi sa zobrazí obrazovka, ktorá sa vo vrchnej časti skladá z:

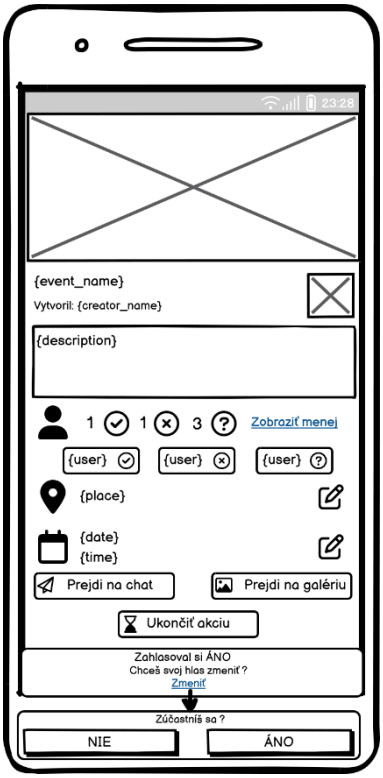
- Titulná fotka – titulná fotka je zobrazená na základe typu akcie
- Názov akcie

- Autor akcie
- Ikona akcie – na základe typu akcie
- Popis akcie – detailný popis alebo plán akcie

V spodnej časti obrazovky sa nachádza:

- Ukazateľ účastníkov – číselná reprezentácia či sa užívateľ zúčastní, nezúčastní alebo ešte nehlasoval
- Ukazateľ účastníkov detail – po rozkliknutí sa zobrazí ako konkrétny užívateľ hlasoval
- Ďalšie informácie o akcií - Obrázok 28 zobrazuje len miesto a dátum, ale môžu sa tu nachádzať ďalšie informácie ako sú ubytovanie, transport a odhadovaná cena. Autor akcie môže tieto hodnoty meniť po kliknutí na tlačidlo editovania.
- Prejdi na chat – tlačidlo po ktorom je užívateľ presmerovaný priamo na obrazovku chatu (7.6 Chaty, Chat Detail)
- Prejdi na galériu – tlačidlo po ktorom je užívateľ presmerovaná na obrazovku galérie (7.4 Galéria)
- Ukončiť akciu – tlačidlo, ktoré je dostupné iba pre autora akcie a tým ukončí akciu a presunie sa na obrazovku Archív (7.7 Archív)
- Hlasovanie – možnosť zahlasovanie o účasti na akcií tlačidlami ÁNO, NIE. V prípade, že užívateľ už zahlasoval tak vidí možnosť ako zahlasoval s možnosťou zmeny hlasu.

Obrázok 28 zobrazuje wireframe k detailu akcií.

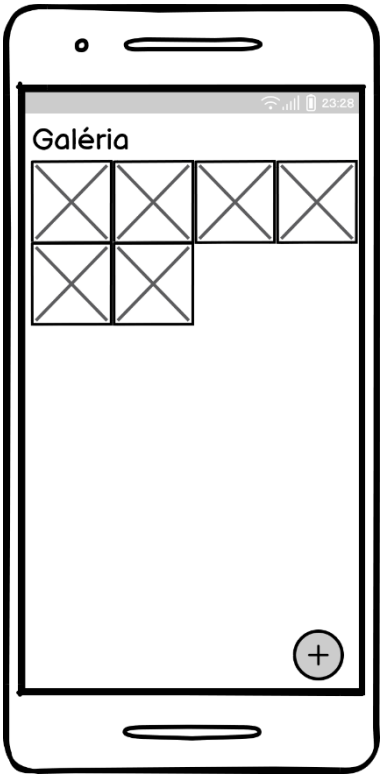


Obrázok 28 Wireframe Detail akcie

7.4 Galéria

Obrazovka Galéria je prístupná z obrazovky 7.3 Detail akcie. Galéria obsahuje jednoduché užívateľské rozhranie kde sú zobrazené náhľady pre fotky a videá. Po kliknutí na akýkoľvek obrázok alebo video je možné prechádzať všetky fotky alebo videá posúvaním doľava alebo doprava.

Wireframe pre Galériu je vyobrazený na Obrázok 29.



Obrázok 29 Wireframe Galéria

7.5 Vytvorenie akcie

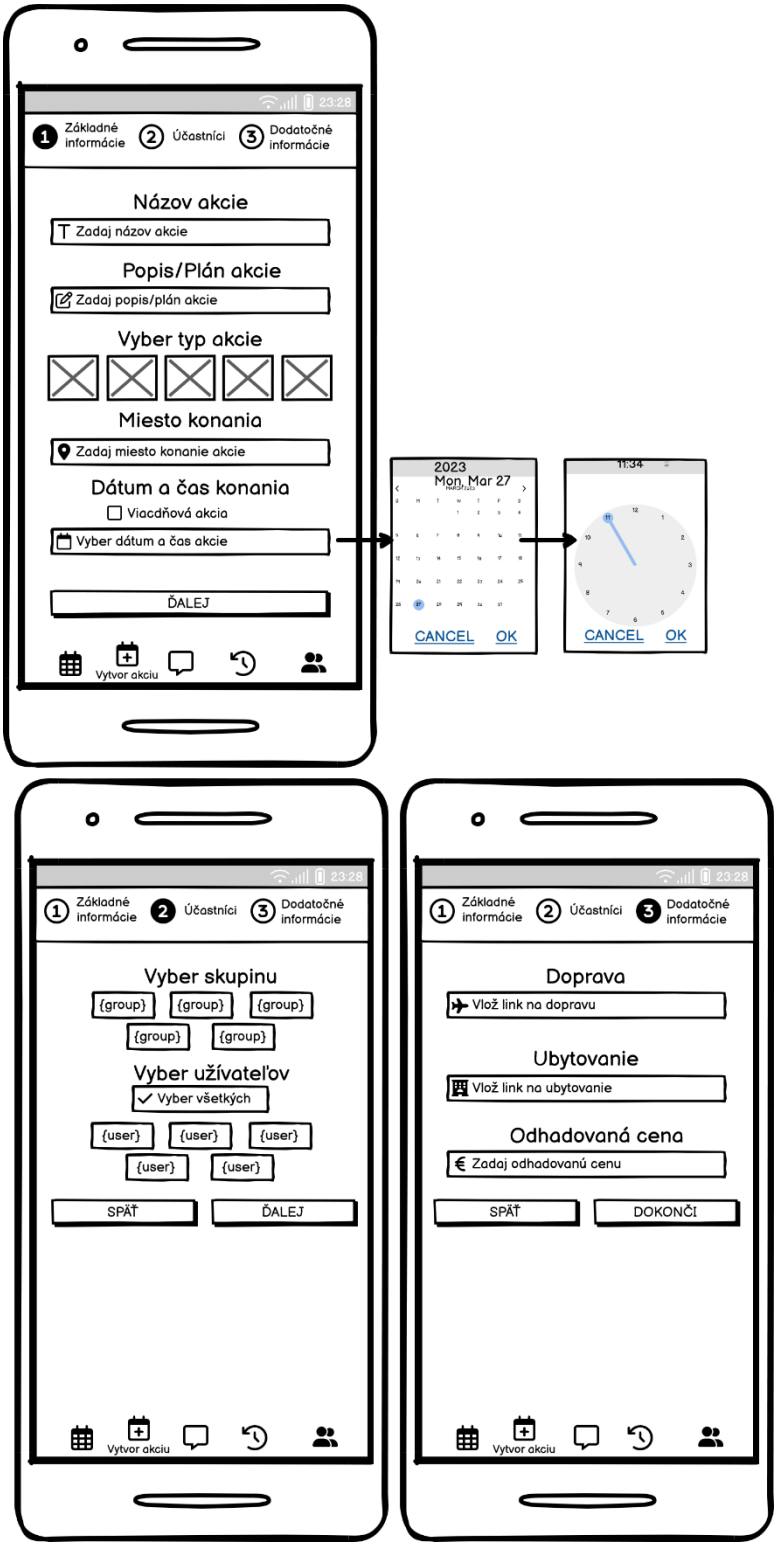
Obrazovka Vytvorenie akcie sa skladá z troch obrazoviek aby bol rozdelený proces vytvárania akcie na relatívne časti.

Prvá obrazovka sa skladá z povinných údajov pre vytvorenie akcie. Formulárové polia na vypísanie potrebných údajov sú názov akcie, popis/plán akcie a miesto konania akcie. Vybratie typu akcie je realizované pomocou posuvníka kde sa nachádzajú ikony, ktoré špecifikujú typ akcie o akú sa jedná. Dátum a čas konania je realizované tak, že po kliknutí sa zobrazí vyskakovacie okno na výber dátumu. Po výbere dátumu sa zobrazí vyskakovacie okno na výber času. V prípade, že je zakliknutý box Viacdňová akcia tak sa po kliknutí na výber dátumu zobrazí vyskakovacie okno kde užívateľ vyberie časový rozsah.

Druhá obrazovka slúži na výber ľudí, ktorých chce autor akcie pozvať na akciu. Najskôr vyberie skupinu z ktorej chce ľudí vybrať a následne vyberie konkrétnych ľudí.

Na poslednej obrazovke sa nachádzajú údaje, ktoré nie sú povinné ale pre niektoré typy akcií môžu byť užitočné. Tieto polia sú ubytovanie, transport a odhadovaná cena.

Obrázok 30 znázorňuje wireframe pre Vytvorenie akcie.



Obrázok 30 Wireframe Vytvor akciu

7.6 Chaty, Chat Detail

Tieto obrazovky spolu úzko súvisia pretože po kliknutí na kartu konkrétneho chatu je užívateľ presmerovaný na Chat Detail.

Chaty sa skladá z jednoduchého zoznamu chatov k akciám, ktoré sú aktuálne (tie, ktoré nie sú v archíve). V karte chatu sa nachádza ikona typu akcie, názov akcie, kto posledný v danom chate poslal správu, obsah správy a kedy bola táto správa odoslaná/prijatá.

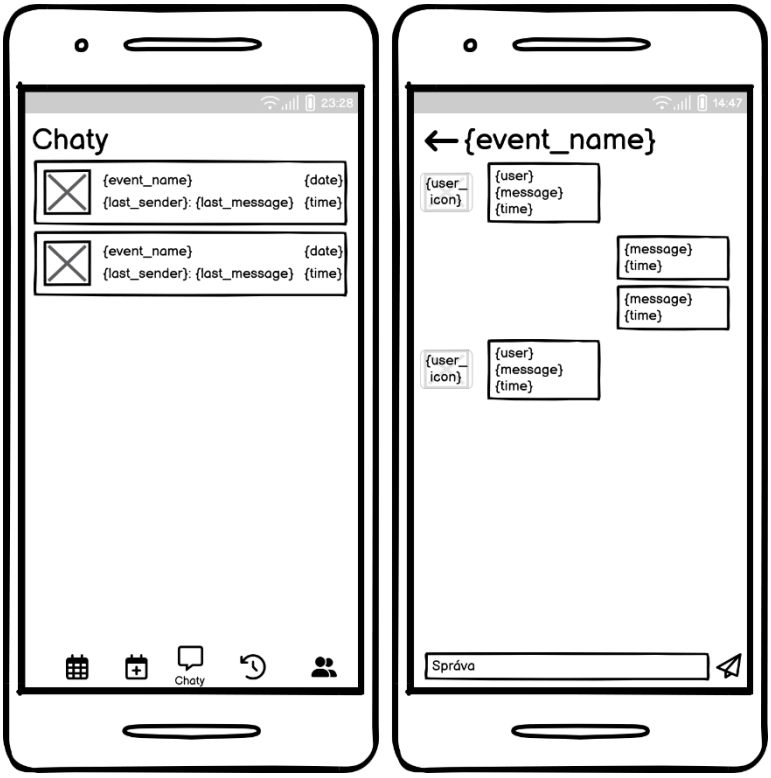
Chat Detail obsahuje zobrazovanie správ v konverzácií ku konkrétnej akcií. Vo vrchnej časti sa nachádza tlačidlo späť na chat list a názov akcie. Správy, ktoré užívateľ prijal sa nachádzajú v ľavej časti, tie ktoré odoslal sa nachádzajú v pravej časti.

Prijaté správy sa skladajú z avatara odosielateľa, užívateľského mena, obsahu správy a času odoslania.

Odoslané sú v podstate totožné, ale neobsahujú avatar odosielateľa.

Úplne dole na obrazovke sa nachádza formulár na napísanie správy a vedľa neho tlačidlo na odoslanie správy.

Obrázok 31 zobrazuje wireframe k týmto obrazovkám.

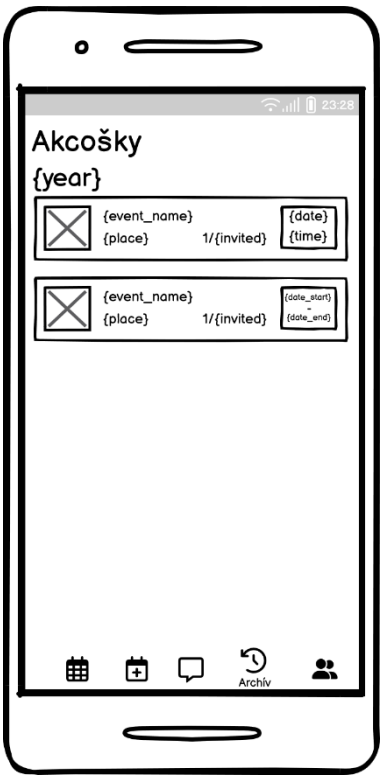


Obrázok 31 Wireframe Chaty, Chat Detail

7.7 Archív

Obrazovka Archív bude obsahovať identické informácie ako obrazovka 7.2 Akcie. Jediný rozdiel je v tom, že zoznam akcií v archíve bude obsahovať akcie zoradené od najnovšej po najstaršiu.

Wireframe pre Archív je zobrazená na Obrázok 32.



Obrázok 32 Wireframe Archív

7.8 Skupiny

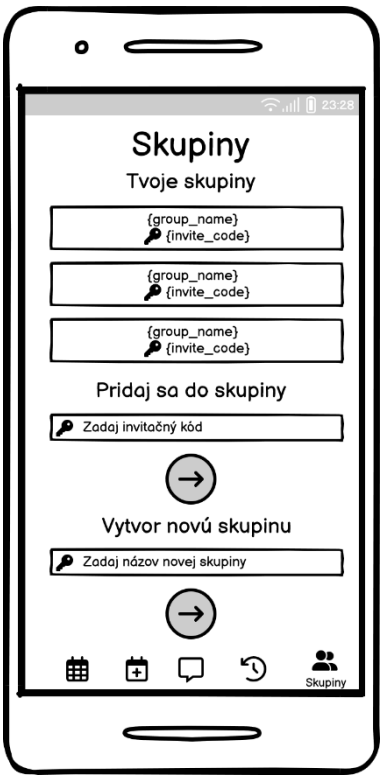
Obrazovka Skupiny sa delí na niekoľko častí – zoznam skupín, pripojenie do skupiny a vytvorenie novej skupiny.

Zoznam skupín obsahuje zoznam skupín v ktorých sa užívateľ nachádza. Karta skupiny obsahuje názov skupiny a kód, pomocou ktorého sa môžu ostatní užívatelia do tejto skupiny pripojiť.

Pridaj sa do skupiny obsahuje jednoduchý formulár na pripojenie do skupiny a tlačidlo na odoslanie formuláru. Jediná vec čo musí užívateľ vypísať je kód na pripojenie a následne stlačiť tlačidlo na odoslanie tejto požiadavky.

Vytvorenie do skupiny obsahuje taktiež ako aj pridanie do skupiny jednoduchý formulár na vytvorenie skupiny a tlačidlo na odoslanie formulára. Užívateľ musí vyplniť názov skupiny a následne odoslať požiadavku o vytvorenie skupiny prostredníctvom pripojeného tlačidla.

Obrázok 33 zobrazuje wireframe pre Skupiny.



Obrázok 33 Wireframe Skupiny

8 IMPLEMENTOVANÝ PROTOTYP APLIKÁCIE NA REÁLNOM ZARIADENÍ

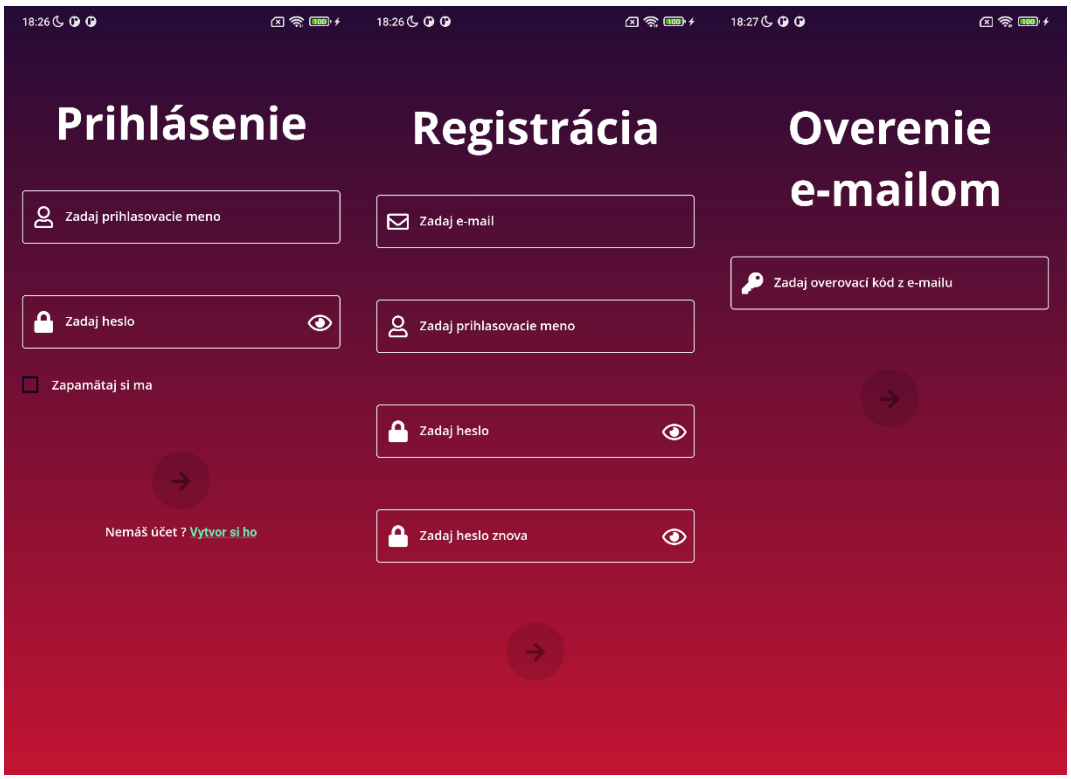
V tejto kapitole sa nachádzajú informácie ohľadom implementácie aplikácie na reálnom zariadení. Nachádzajú sa tu kapitoly na ukážku implementácie obrazoviek a informácie o testovaní aplikácie.

8.1 Implementácia obrazoviek aplikácie

V tejto kapitole sú vyobrazené jednotlivé obrazovky tak, ako ich bude vidieť koncový užívateľ na reálnom zariadení. Všetky obrázky boli vyhotovené na zariadení Xiaomi Redmi Note 8 Pro s rozlíšením 1080x2340, veľkosťou displeja 6.53" a operačným systémom Android 11.

8.1.1 Prihlásenie, Registrácia a Overenie

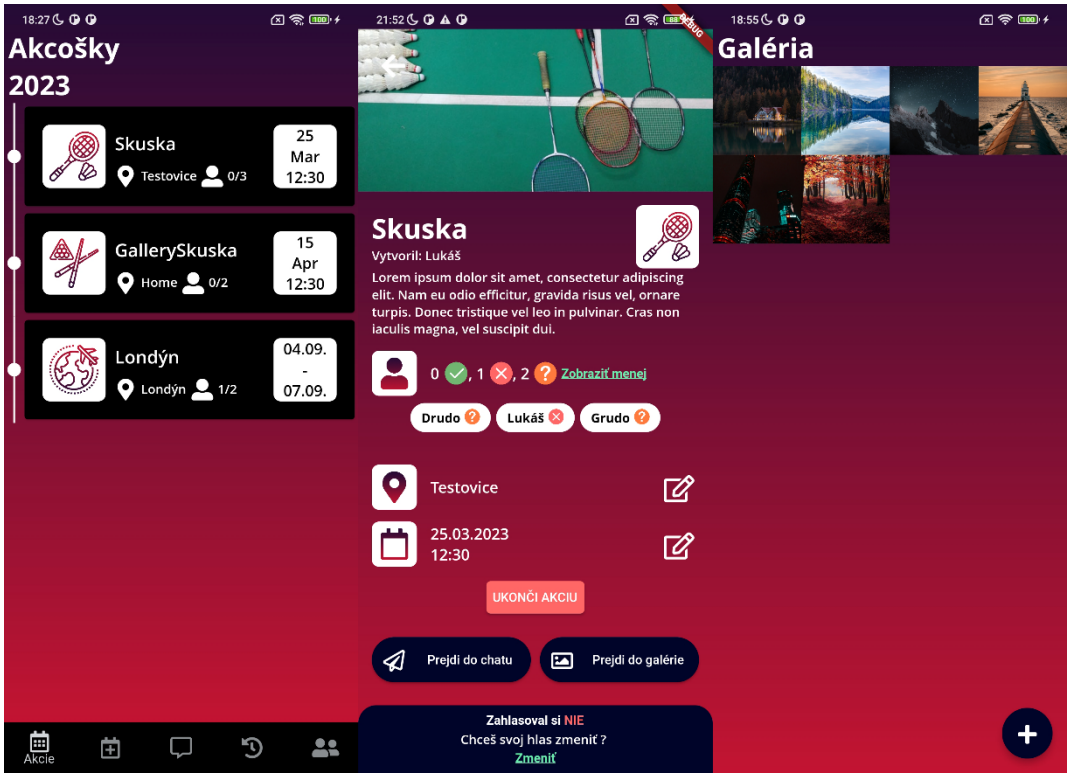
Obrázok 34 znázorňuje obrazovky Prihlásenie, Registrácia a Overenie na reálnom zariadení. V porovnaní s wireframom pre tieto obrazovky (Obrázok 26) môžeme vidieť, že rozdiely medzi wireframom a reálnou implementáciou sú minimálne.



Obrázok 34 Implementácia obrazovky pre Prihlásenie, Registráciu a Overenie

8.1.2 Akcie, Detail akcie a Galéria

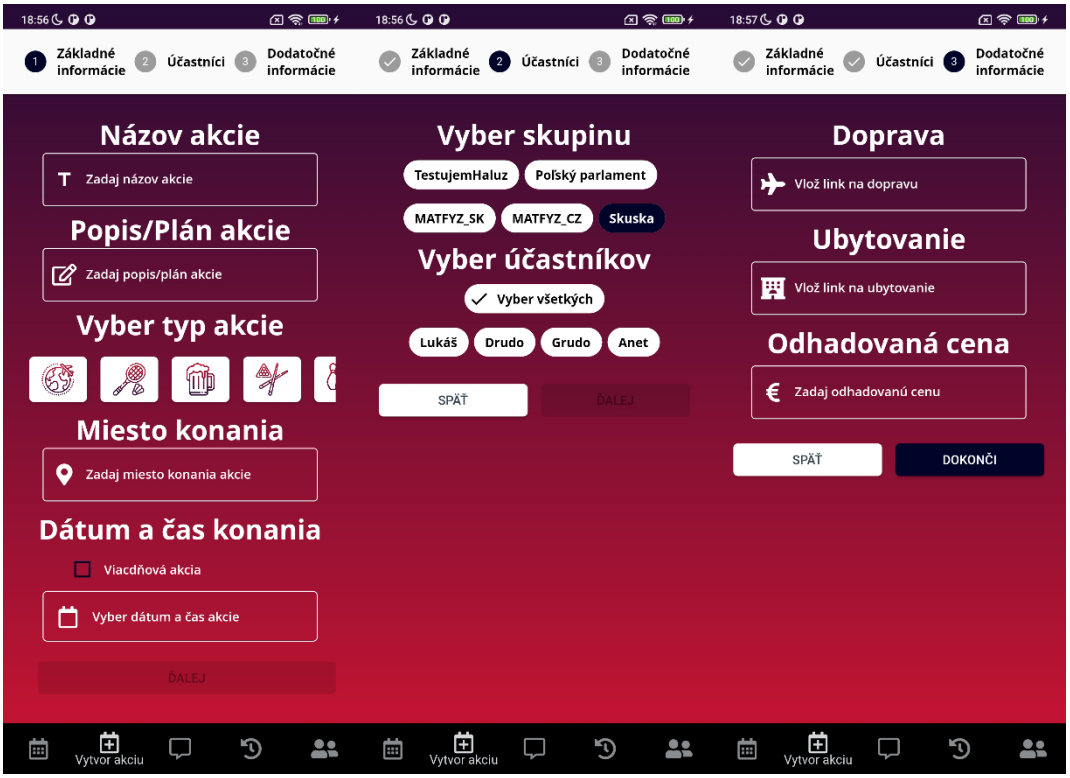
Obrázok 35 zobrazuje implementáciu obrazoviek Akcie, Detail a Galériu. Taktiež v porovnaní s wireframami (Obrázok 27, Obrázok 28, Obrázok 29) sa tu nachádzajú minimálne zmeny. Tlačidlo Ukončiť akciu sa nachádza pred tlačidlami Prejdi do chatu a Prejdi do galérie. Ďalšia zmena sa nachádza vo vyššie spomínaných tlačidlách vstupu do chatu a galérie. Tu je drobná zmena v syntaxe.



Obrázok 35 Implementácia obrazovky pre Akcie, Detail akcie a Galériu

8.1.3 Vytvorenie akcie

Obrázok 36 obsahuje implementáciu obrazovky pre Vytvorenie akcie na reálnom zariadení. Táto implementácia je v porovnaní s wireframom (Obrázok 36) jedna ku jednej.

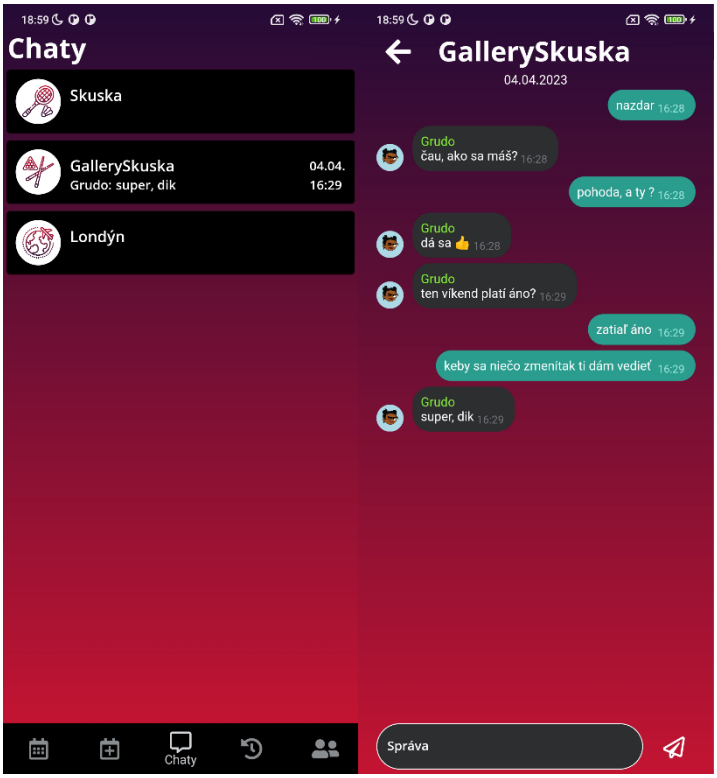


Obrázok 36 Implementácia obrazovky pre Vytvorenie akcie

8.1.4 Chaty a Chat Detail

Obrázok 37 zobrazuje implementáciu obrazoviek Chaty a Chat Detail.

Obrazovka Chaty obsahuje je v porovnaní s wireframom (Obrázok 31) identická. Obrazovka Chat Detail obsahuje niekoľko zmien. Pred prvou správou v novom dni sa na obrazovke nachádza dátum v ktorom dni bola správa odoslaná. Taktiež sa zmenila pozícia dátumu v správe z ľavého dolného rohu a zarovnania s odosielateľom a obsahom správy sa čas presunul do pravej strany chatovacej bubliny a so správou nie je zarovnaný, je trocha posunutý dole.



Obrázok 37 Implementácia obrazovky pre Chat a Chat Detail

8.1.5 Archív

Obrázok 38 obsahuje implementáciu obrazovky Archív. Táto implementácia nie je identická s wireframom (Obrázok 32), ale aj s implementáciou obrazovky Akcie (Obrázok 35). Jediný rozdiel je v tom, že na tejto obrazovke sú akcie zoradené od najnovšej po najstaršiu.

Obrázok 38 Implementácia obrazovky Archív

8.1.6 Skupiny

Obrázok 39 znázorňuje implementáciu obrazovky Skupiny na reálnom zariadení. Táto obrazovke je taktiež identická s wireframom pre túto obrazovku (Obrázok 33).

Obrázok 39 Implementácia obrazovky Skupiny

8.2 Testovanie na reálnom zariadení

Testovanie prebiehalo prostredníctvom niekoľkých zariadení s rôznymi rozlíšeniami a veľkosťami displeja. Tabuľka 2 zobrazuje konkrétne zariadenia a ich špecifikácia na ktorých bola aplikácia testovaná:

Tabuľka 2 Testovacie zariadenia aplikácie

Názov	**Verzia systému Android**	**Rozlíšenie**	**Veľkosť displeja**
Xiaomi Redmi Note 8 Pro	11	1080x2340	6,53"
Samsung Galaxy A51	13	1080x2400	6,5"
Xiaomi Mi A1	9	1080x1920	5,5"

Aplikácia bola testovaná manuálne. Vždy po naprogramovaní určitej časti aplikácie bola táto časť ihneď otestovaná na koncovom zariadení. Hlavným testovacím zariadením bol smartphone Xiaomi Redmi Note 8 Pro. Aplikácia na tomto zariadení funguje z pohľadu výkonu skvele a taktiež užívateľské rozhranie funguje a vyzerá vynikajúco.

Zvyšné dve testovacie zariadenia slúžili hlavne na testovanie funkčnosti posielania správ medzi zariadeniami prostredníctvom chatu. Na týchto zariadeniach bola taktiež aplikácia otestovaná, ale nie po každej naprogramovanej časti, ale ako celok. Aplikácia na týchto zariadeniach funguje skvelo aj napriek tomu, že zariadenie Xiaomi Mi A1 bolo vydané v roku 2017 a beží na Androide 9. Z pohľadu užívateľského rozhrania na zariadení Samsung Galaxy A51 je užívateľské rozhranie v podstate identické ako na zariadení Xiaomi Redmi Note 8 Pro z dôvodu podobného rozlíšenia aj podobnej veľkosti displeja. Testovanie užívateľského rozhrania na zariadení Xiaomi Mi A1 bolo kľúčové, pretože je o 1,03 palca menšie ako hlavné testovacie zariadenie. Užívateľské rozhranie ale bolo navrhnuté tak aby si dokázalo poradiť s rozdielnymi veľkosťami displeja. Toto sa testovaním na tomto zariadení potvrdilo, aj keď niekedy sa na menšie obrazovky zabudlo a bolo treba tieto časti užívateľského rozhrania zmeniť.

8.3 Testovanie nezávislými ľuďmi

Testovanie aplikácie prebehlo aj niekoľkými nezávislými ľuďmi. Počas testovanie aplikácie sa nestretli so žiadnou závažnou chybou iba odhalili jeden neošetrený vstup pri vytváraní akcie. Aplikáciu hodnotili kladne, nevyskytli sa žiadne problémy so zaobchádzaním s aplikáciou.

9 ZABEZPEČENIE APLIKÁCIE

V tejto kapitole budú rozobraté najdôležitejšie funkcie aplikácie, ktoré ju robia bezpečnú.

9.1 Ukladanie šifrovacích kľúčov

Aplikácia potrebuje bezpečný spôsob ukladania šifrovacích kľúčov ktorými sú zašifrované databázy. Bezpečné uloženie týchto kľúčov je možné vďaka nasledujúcemu pluginu.

9.1.1 Plugin Flutter Secure Storage

Na operačnom systéme Android používa tento plugin systém Android Keystore. [41]

9.1.1.1 Android Keystore systém

V prvom znižuje možnosť prístupu k citlivým údajom mimo operačného systému Android. Toto je riešené tým, že citlivé údaje nevstupujú do procesu aplikácie. Pri operáciách aplikácie sa citlivé údaje posielajú do systémového procesu, ktorý slúži na kryptografické operácie. Týmto docielime, že v prípade napadnutého procesu aplikácie útočník môže získať kľúče aplikácie, ale nezíska konkrétne citlivé údaje. Taktiež môžu byť citlivé údaje viazané na bezpečný hardware zariadenia. V tomto prípade nie sú citlivé údaje nikdy vystavené mimo tohto bezpečného hardware. V prípade napadnutia operačného systému útočník môže získať a použiť akékoľvek Android keystore kľúče na Android zariadení, ale nemôže ich extrahovať zo zariadenia. [42]

V druhom znižuje možnosť použitia neautorizovaných kľúčov tým, že Android Keystore nechá aplikácie špecifikovať spôsob autorizácie. Táto autorizácia je následne vyžadovaná pri každom použití kľúča. Spôsoby autorizácie sú kryptografia, časový interval platnosti a overenie užívateľa. [42]

Na operačnom systéme iOS sa používa systém Keychain. [41]

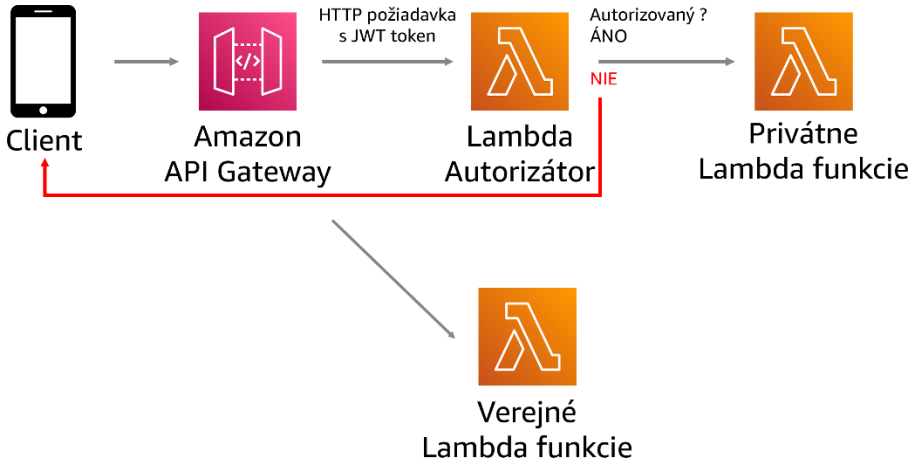
9.1.1.2 iOS Keychain Services

Keychain Services slúžia na uchovávania hesiel, kryptografických kľúčov, certifikátov, identít, poznámok a ďalších. Pri uchovaní citlivých údajov sú zabalené ako „keychain item“. Spolu s citlivými údajmi sú zabalené aj atribúty na kontrolovanie prístupu k dátam a aby ich bolo možné vyhľadať. [43]

Tento plugin obsahuje aj experimentálnu verziu uloženia údajov vo webovej verzií aplikácie pomocou WebCrypto API. Pri použití je veľmi dôležité dbať na správne použitie hlavičiek na odpovede a taktiež mať povolené HTTP Strict Forward Secrecy. To znamená, že aj v prípade odcudzenia privátneho kľúča používaného v aktuálnej relácií, nebude mať vďaka tomuto kľúču prístup k minulým alebo budúcim reláciám. [41]

9.2 Zabezpečená komunikácia s platformou Amazon Web Services

Klient so serverom komunikuje pomocou služby Amazon API Gateway, ktorá vystavuje jednotlivé časti backendu na prístup. Služba Lambda obsahuje endpointy, ktoré sú voľné prístupné napr. endpoint na vygenerovanie tokenu, a privátne, kam sa užívateľ nedostane, pokiaľ nebude overený či má na tieto endpointy prístup. Obrázok 40 znázorňuje túto funkcionalitu.



Obrázok 40 Komunikácia s platformou Amazon Web Services

Pri prihlásení do aplikácie sa užívateľove údaje odošlú na verejný endpoint kde sa overí, že užívateľov účet naozaj v databáze existuje a odošle sa mu JWT token, pomocou ktorého sa bude užívateľ preukazovať či je autorizovaný na jednotlivé privátne endpointy. Pri posielaní žiadosti na privátny endpoint sa v žiadosti zahrnie Authentication hlavička, do ktorej sa vloží tento JWT token. Služba Lambda Authorizer overí, že tento JWT token bol vydaný a je platný. V prípade že je platný tak užívateľova žiadosť je poslaná na príslušnú Lambda funkciu. V prípade, že užívateľ má neplatný token alebo žiaden token neuviedol je táto žiadosť vrátená na API Gateway a tá následne odošle užívateľovi odozvu, že jeho žiadosť bola zamietnutá.

9.3 Zabezpečené posielanie správ pomocou Signal protokolu

Na základe niekoľkých zdrojov [44] [45] je jednou z najbezpečnejších aplikácií na okamžité posielanie správ aplikácia Signal. Vzhľadom na to, že Signal je open-source a poskytujú tzv. Signal protocol vo viacerých programovacích jazykoch vo forme knižnice, rozhodol som sa tento komunikačný protokol implementovať aj v mojej aplikácií. Najdôležitejšie časti tohto protokolu sú 9.3.1 The X3DH Key Agreement Protocol a 9.3.2 Double Ratchet Algorithm.

9.3.1 The X3DH Key Agreement Protocol

Pomocou tohoto protokolu zriaďujeme zdieľaný tajný kľúč medzi dvomi stranami, ktoré sa overujú na základe verejných kľúčov. X3HD poskytuje forward secrecy a cryptographic deniability. X3HD je vytvorené za účelom kedy jeden používateľ (napr. Bob) je offline, ale poskytol nejaké údaje serveru. Druhý užívateľ (napr. Alica) chce použiť tieto údaje aby mohol poslať zašifrované údaje Bobovi a zároveň nadviazať spoločný tajný kľúč pre budúcu komunikáciu. [46]

Po počiatočnej dohode na kľúčoch pomocou X3DH nasleduje samotné posielanie zabezpečených správ. O toto sa stará algoritmus Double Ratchet. [46]

9.3.2 Double Ratchet Algorithm

Algoritmus Double Ratchet používajú dve strany na výmenu zašifrovaných správ na základe zdieľaného tajného kľúča. Zvyčajne strany použijú nejaký protokol dohody o kľúči (napríklad X3DH), aby sa dohodli na zdieľanom tajnom kľúči. Následne strany použijú Double Ratchet na odosielanie a prijímanie zašifrovaných správ. Strany odvodia nové kľúče pre každú správu Double Ratchet, aby predchádzajúce kľúče nemohli byť vypočítané z neskorších kľúčov. Strany si taktiež pošlú verejné hodnoty Diffie-Hellman kalkulácie pripojené k správam. Tieto hodnoty sú zmiešané do odvodených kľúčov aby neskoršie kľúče nemohli byť vypočítané z predošlých kľúčov. Tieto vlastnosti poskytujú určitú ochranu pre skoršie alebo neskoršie správy v prípade kompromitovania kľúčov niektorej strany. [47]

10 NÁVRH ĎALŠIEHO VÝVOJA

Medzi funkcie nad ktorými sa treba zamyslieť je implementácia Signal protocolu. Spôsob akým je Signal protokol implementovaný je formou komunikácie jedného s jedným. Toto je taktiež najbezpečnejší spôsob. V prípade, že by sa v skupine nachádzalo veľa účastníkov tak by odosielanie správy mohlo zabrať dlhší čas. Signal protocol obsahuje lepší spôsob šifrovania v skupinách pomocou tzv. Sender keys, ale na úkor bezpečnosti.

Jednou z ďalších funkcií, ktoré by bolo treba vylepšiť sa nachádza na serverovej časti. Konkrétne sa jedná o doby odozvy zo servera pri vykonávaní operácií v databáze DynamoDB. Inicializácia DynamoDB klienta spolu s prvou požiadavkou trvajú dlho.

Medzi nové funkcie, ktoré by aplikácia určite mohla obsahovať je viacej funkcií v chate. Napríklad hlasovanie na zvolenú otázku, nákupný zoznam kde by mohli všetci účastníci pridávať položky, zdieľanie lokality v reálnom čase atď.

Ďalším praktickým vylepšením, ktoré by zrýchlilo štart aplikácie je implementácia cache. V prípade, že užívateľ by aplikáciu otváral každých niekoľko desiatok minút aby odpísal na správu v chate a zoznam akcií spolu so zoznam skupín a užívateľmi sa v tomto časovom rozmedzí nemenil a bol by už dopredu nacachovaný, určite by to ocenil.

V prípade spustenia aplikácie pre verejnosť by bolo taktiež vhodné sa zamyslieť nad spôsobom monetizovania aplikácie. Jedným zo spôsobov je určite predplatné pre určitú skupinu. Toto predplatné by dávalo skupine možnosť nahrávať viac obsahu do galérie, prístup k unikátnym funkciám (ako napríklad vyššie spomínané hlasovanie v chate atď) a väčší počet užívateľov v jednej skupine. Tieto všetky funkcie a obmedzenia by museli byť do aplikácie pridané a taktiež by musel byť pridaný spôsob platby tohto predplatného.

ZÁVĚR

Cieľom diplomovej práce bolo vytvorenie prototypu aplikácie na organizovanie akcií. Frontend aplikácie bol realizovaný pomocou frameworku Flutter a backend pomocou jazyku C#, kde sa využívajú rôzne služby platformy Amazon Web Services.

Teoretická časť obsahuje popis vývoja multiplatformových aplikácií s hlavným zameraním na výhody a nevýhody. Taktiež obsahuje popis rôznych existujúcich frameworkov pre multiplatformové programovanie.

Praktická časť začína kapitolou ohľadom prieskumu trhu s aplikáciami, ktoré riešia podobnú problematiku. Ďalej sa v praktickej časti nachádza model systému, ktorý obsahuje funkcionálne a nefunkcionálne požiadavky a taktiež model prípadov užitia. Ku každému prípadu užitia je napísaný scenár a implementácia ako je daná funkcia naprogramovaná. Taktiež sa v tejto časti nachádza popis všetkých databáz s ktorými aplikácia pracuje. Dizajnu aplikácie sa venuje kapitola prehľad užívateľského rozhrania. Táto kapitola obsahuje jednotlivé wireframy pre všetky obrazovky aplikácie tak ako boli navrhnuté. Nasleduje kapitola, ktorá sa zaoberá implementáciou navrhnutého užívateľského rozhrania. V tejto kapitole sú zobrazené konkrétne obrazovky aplikácie na reálnom zariadení. Taktiež sú v tejto kapitole informácie o tom ako bola aplikácia testovaná. V poslednej časti sa nachádza kapitola ohľadom zabezpečenia aplikácie a návrh ďalšieho vývoja.

Pre nasadenie aplikácie na trh by bolo treba vyriešiť funkcie a problémy, ktoré sa nachádzajú v závere práce a taktiež spôsob monetizovania aplikácie. V budúcnosti by do aplikácie mohli pribúdať nové funkcie pre zlepšenie organizovania aplikácie na základe podnetov od užívateľov, ktorí aplikáciu používajú.

SEZNAM POUŽITÉ LITERATURY

[1] *What is cross-platform mobile development?* [online]. Kotlin Docs, 2023 [cit. 2023-01-30]. Dostupné z: https://kotlinlang.org/docs/cross-platform-mobile-development.html

[2] *What are the Benefits of Cross-platform Application Development?* [online]. Techmango [cit. 2023-01-30]. Dostupné z: https://www.techmango.net/what-are-the-benefits-of-cross-platform-application-development

[3] *Cross Platform Application Development: Benefits and Technology* [online]. Clariontech [cit. 2023-01-30]. Dostupné z: https://www.clariontech.com/blog/cross-platform-application-development-benefits-and-technology

[4] *Cross-Platform App Development Challanges & Solutions* [online]. Aglowid, 2021 [cit. 2023-01-30]. Dostupné z: https://aglowiditsolutions.com/blog/cross-platform-app-development-analysis/

[5] *The Pros and Cons of Cross-Platform Mobile App Development* [online]. AppsChopper [cit. 2023-01-30]. Dostupné z: https://www.appschopper.com/blog/pros-cons-cross-platform-mobile-app-development/

[6] *FAQ* [online]. Flutter Docs [cit. 2023-01-31]. Dostupné z: https://docs.flutter.dev/resources/faq

[7] *Flutter architectural overview* [online]. Flutter Docs [cit. 2023-01-31]. Dostupné z: https://docs.flutter.dev/resources/architectural-overview

[8] *Start thinking declaratively* [online]. Flutter Docs [cit. 2023-01-31]. Dostupné z: https://docs.flutter.dev/development/data-and-backend/state-mgmt/declarative

[9] *What not to commit* [online]. Dart dev [cit. 2023-02-03]. Dostupné z: https://dart.dev/guides/libraries/private-files#the-rules

[10] *Breaking Down Flutter Project File/Folders* [online]. Stack Secrets, 2019 [cit. 2023-02-03]. Dostupné z: https://stacksecrets.com/flutter/breaking-down-flutter-project-file-folders

[11] *Customizing static analysis* [online]. Dart dev [cit. 2023-02-03]. Dostupné z: https://dart.dev/guides/language/analysis-options

[12] *IML File* [online]. OpenMyFiles [cit. 2023-02-03]. Dostupné z: https://www.openmyfiles.com/iml-file/

[13] *Hot Reload* [online]. Flutter Docs [cit. 2023-02-04]. Dostupné z: https://docs.flutter.dev/development/tools/hot-reload

[14] *7 Benefits of Flutter: Why Use Flutter for Your Next App* [online]. INVERITA [cit. 2023-01-31]. Dostupné z: https://inveritasoft.com/blog/7-flutter-advantages-a-perfect-platform-for-your-next-app

[15] *What is Flutter App Development ? Advantages & Drawbacks of Flutter* [online]. The One Technologies, 2022 [cit. 2023-02-04]. Dostupné z: https://theonetechnologies.com/blog/post/flutter-mobile-application-development

[16] *Key Disadvantages of Flutter You Should Know About* [online]. PANGEA, 2023 [cit. 2023-02-04]. Dostupné z: https://www.pangea.ai/dev-flutter-development-resources/key-disadvantages-of-flutter/

[17] *What Is React Native? Complex Guide for 2022* [online]. netguru [cit. 2023-02-20]. Dostupné z: https://www.netguru.com/glossary/react-native

[18] *New React Native Architecture – An Overview* [online]. Software Mind, 2023 [cit. 2023-02-20]. Dostupné z: https://softwaremind.com/new-react-native-architecture-an-overview/

[19] *React Native Pros and Cons* [online]. pagepro [cit. 2023-01-31]. Dostupné z: https://pagepro.co/blog/react-native-pros-and-cons/

[20] *Top 10 React Native Disadvantages* [online]. back4app [cit. 2023-01-31]. Dostupné z: https://blog.back4app.com/react-native-disadvantages/

[21] *Multiplatform mobile FAQ* [online]. Kotlinlang docs [cit. 2023-02-16]. Dostupné z: https://kotlinlang.org/docs/multiplatform-mobile-faq.html

[22] *Kotlin Multiplatform Introduction* [online]. netguru [cit. 2023-02-16]. Dostupné z: https://www.netguru.com/blog/kotlin-multiplatform-introduction

[23] *Kotlin Multiplatform Pros and Cons for App Development* [online]. freshworks [cit. 2023-02-16]. Dostupné z: https://freshworks.io/kotlin-multiplatform-pros-and-cons-for-app-development/

[24] *What is .NET MAUI?* [online]. Microsoft Learn, 2023 [cit. 2023-02-17]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-7.0

[25] *Xamarin Versus .NET MAUI* [online]. Syncfusion blog [cit. 2023-02-19]. Dostupné z: https://www.syncfusion.com/blogs/post/xamarin-versus-net-maui.aspx

[26] *.NET MAUI vs Xamarin.Forms: A Comparison of Cross-Platform Frameworks* [online]. Grial [cit. 2023-02-19]. Dostupné z: https://grialkit.com/blog/learn-the-key-differences-between-net-maui-vs-xamarin-forms-for-cross-platform-mobile-and-desktop-development

[27] *The Good and the Bad of Ionic Mobile Development* [online]. altexsoft, 2019 [cit. 2023-02-20]. Dostupné z: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-ionic-mobile-development/

[28] *Capacitor* [online]. ionic docs [cit. 2023-02-20]. Dostupné z: https://ionicframework.com/docs/native

[29] *How Capacitor Works* [online]. [cit. 2023-02-20]. Dostupné z: https://capacitorjs.jp/blog/how-capacitor-works

[30] *What Is Ionic And How Does It Work?* [online]. [cit. 2023-02-20]. Dostupné z: https://appstronauts.co/blog/what-is-ionic-and-how-does-it-work/

[31] *Choosing the Right DynamoDB Partition Key* [online]. AWS Database Blog [cit. 2023-03-27]. Dostupné z: https://aws.amazon.com/blogs/database/choosing-the-right-dynamodb-partition-key/

[32] *Hive* [online]. pub.dev [cit. 2023-03-27]. Dostupné z: https://pub.dev/packages/hive

[33] *Building Applications with Serverless Architectures* [online]. AWS [cit. 2023-03-21]. Dostupné z: https://aws.amazon.com/lambda/serverless-architectures-learn-more/

[34] *AWS Amplify FAQs* [online]. AWS [cit. 2023-03-21]. Dostupné z: https://aws.amazon.com/amplify/faqs/

[35] *What is AWS AppSync?* [online]. AWS Docs [cit. 2023-03-21]. Dostupné z: https://docs.aws.amazon.com/appsync/latest/devguide/what-is-appsync.html

[36] *What is Amazon API Gateway?* [online]. AWS Docs [cit. 2023-03-21]. Dostupné z: https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

[37] *What is AWS Lambda?* [online]. AWS Docs [cit. 2023-03-21]. Dostupné z: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

[38] *What is Amazon SNS?* [online]. AWS Docs [cit. 2023-03-21]. Dostupné z: https://docs.aws.amazon.com/sns/latest/dg/welcome.html

[39] *What is Amazon DynamoDB?* [online]. AWS Docs [cit. 2023-03-21]. Dostupné z: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

[40] *What is Amazon S3?* [online]. AWS Docs [cit. 2023-03-21]. Dostupné z: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

[41] *Flutter secure storage* [online]. Dart and Flutter package repository [cit. 2023-04-05]. Dostupné z: https://pub.dev/packages/flutter_secure_storage

[42] *Android Keystore system* [online]. Android Developers [cit. 2023-104-05]. Dostupné z: https://developer.android.com/training/articles/keystore

[43] *Keychain services* [online]. Developer Apple [cit. 2023-04-05]. Dostupné z: https://developer.apple.com/documentation/security/keychain_services

[44] *The Most Secure Messaging Apps* [online]. AVG [cit. 2023-04-05]. Dostupné z: https://www.avg.com/en/signal/secure-message-apps

[45] *The best encrypted messaging apps in 2023* [online]. Tom's Guide, 2023 [cit. 2023-04-05]. Dostupné z: https://www.tomsguide.com/reference/best-encrypted-messaging-apps

[46] *The X3DH Key Agreement Protocol* [online]. Signal Specifications [cit. 2023-04-05]. Dostupné z: https://signal.org/docs/specifications/x3dh/

[47] *The Double Ratchet Algorithm* [online]. Signal Specifications [cit. 2023-04-05]. Dostupné z: https://signal.org/docs/specifications/doubleratchet/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface
UI	User Interface
UX	User Experience
SDK	Software Development Kit
KMP	Kotlin Multiplatform
XAML	Extensible Application Markup Language
BCL	.NET Base Class Library
PC	Personal Computer
SVG	Scalable Vector Graphics
PNG	Portable Network Graphics
MMVM	Model-View-Viewmodel
RxUI	ReactiveUI
MVU	Model-View-Update
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
N/R	Not relevant
NBD	Not by default
AWS	Amazon Web Services

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD-ROM

Priložené CD obsahuje:

- Zdrojové kódy aplikácie, backendu a build aplikácie vo formáte .zip
- Diplomovú prácu vo formáte .pdf