FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Mobilní klient pro aplikaci Journal
Student:	Martina Fukalová
Vedoucí:	Ing. Zdeněk Rybola, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2021/22

Pokyny pro vypracování

Cílem bakalářské práce je vytvoření mobilního klienta pro webovou aplikaci Journal, která poskytuje funkce pro ukládání záznamů o důležitých událostech, import z ICS kalendářů, zobrazení událostí na mapě, přikládání příloh a přehledné vyhledávání.

Pokyny:
- Analyzujte dostupné funkce webové aplikace Journal.
- Proveďte rešerši podobně zaměřených mobilních aplikací pro záznam událostí a deníkových záznamů.
- Specifikujte požadavky na mobilní aplikaci využívající REST rozhraní webové aplikace Journal.
- Navrhněte řešení mobilního klienta, implementujte podle návrhu a patřičně otestujte.
- Zdokumentujte řešení a připravte uživatelskou příručku a vývojářskou příručku.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 26. května 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Journal - mobilní aplikace

Martina Fukalová

Katedra Softwarového inženýrství
Vedoucí práce: Ing. Zdeněk Rybola, Ph.D.

30. července 2020

Poděkování

Děkuji vedoucímu mé bakalářské práce, Ing. Zdeňku Rybolovi, Ph.D., za ochotu vést mou bakalářskou práci, za cenné rady a za čas, který mi věnoval v průběhu zpracování bakalářské práce. Dále děkuji všem, kteří se podíleli na vývoji webové verze aplikace Journal, na kterou jsem navázala ve své práci. V neposlední řadě bych chtěla poděkovat rodině a přátelům, kteří mi byli oporou po celou dobu mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 30. července 2020 .....

Odkaz na tuto práci

Fukalová, Martina. *Journal - mobilní aplikace.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato práce se zabývá vytvořením mobilního klienta pro webovou aplikaci Journal, již dříve vytvořenou v rámci jiné bakalářské práce, na jejíž funkcionalitu tento mobilní klient navazuje. Tato mobilní aplikace, podobně jako její webová verze, poskytuje funkce pro vedení deníku v chytrém telefonu, tedy ukládání osobních záznamů o důležitých událostech, s možností jejich zobrazení na mapě či v kalendáři, přikládání příloh a přehledného vyhledávání. Součástí této bakalářské práce je rovněž analýza konkurenčních řešení se zhodnocením přínosu právě této mobilní aplikace.

Klíčová slova mobilní klient, webová aplikace, událost, deník, Android, Journal, Kotlin, REST, Mapy Google

Abstract

This bachelor thesis deals with a creation of mobile client for web application Journal which were created previously within another bachelor thesis, the functionality of which this mobile client follows and extends. This mobile application, like its web version, provides functions for keeping a diary in a smartphone, it means storing personal records of important events, with the possibility of displaying them on a map or calendar, attaching attachments and

clear search. A part of this bachelor's thesis is also an analysis of competing solutions with an evaluation of the benefits of this mobile application.

Keywords mobile client, web application, event, diary, Android, Journal, Kotlin, REST, Google Maps

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Lidé měli odpradávna potřebu zaznamenávat události, které prožívali nebo jichž se stali svědky. V historii se pro záznamy používalo mnoho materiálů: dřevěné či hliněné destičky, kůra stromů, papyrus či pergamen vyráběný z ovčí kůže. Ale teprve s rozšířením papíru nastal přelom v uchování informací, papír se totiž dal snadno vyrábět ve velkém množství a stával se postupně levnější a dostupnější pro širší vrstvy populace. Mezi lidmi se stala oblíbenou deníková metoda zápisků, tedy osobních záznamů vedených obvykle na denní bázi.

S nástupem informačních technologií začal elektronický způsob záznamu informací postupně vytlačovat papírovou podobu. Používání počítačů, internetu a později i chytrých telefonů se stalo každodenní součástí našich životů. Lidé se naučili využívat nové technologie nejen k práci, ale i pro zábavu. Někteří lidé chtějí své zážitky a myšlenky sdílet, což vedlo k raketovému startu obliby sociálních sítí a blogů, jiní si chtějí uchovat své vzpomínky jen pro svou vlastní potřebu a vedou si osobní deníky.

Prozkoumala jsem několik mobilních aplikací pro platformu Android, které jsou určeny k vedení osobního deníku. Některé z nich umožňují pouze prosté vedení textových záznamů na bázi kalendáře. Jiné poskytují i možnost přidání obrázkových příloh. Nenašla jsem však žádnou takovou aplikaci, která by umožňovala přidání souřadnic GPS a následné zobrazení události v mapě. Tato funkcionalita se mi osobně jeví jako velmi užitečná a žádaná.

Ve své bakalářské práci navazuji na již existující aplikaci Journal, která byla vyvíjena na Fakultě informačních technologií ČVUT Praha po dobu dvou semestrů pod vedením Ing. Zdeňka Ryboly, Ph.D., v předmětech BI-SP1 (Softwarový týmový projekt 1) a BI-SP2 (Softwarový týmový projekt 2). Tato aplikace byla vyvinuta za účelem online správy a uchovávání osobních záznamů o událostech, čímž nahrazuje papírovou formu deníku. Kromě toho disponuje tato aplikace oproti papírové verzi dalšími užitečnými funkcemi, jako například možností vyhledávání podle názvu, data nebo tagu přiřazenému k

události, přidávání lokací k událostem a možností jejich následného zobrazení na mapě nebo zobrazení událostí v kalendářním náhledu.

Vzhledem k dnešní oblibě mobilních zařízení (chytrých telefonů či tabletů) se jevilo jako přínosné převést webovou aplikaci Journal do mobilní verze. Tímto krokem se uživateli zpřístupnila možnost mít svůj osobní deník vždy po ruce a využít i další možnosti, které nabízí mobilní zařízení, tedy možnost být připojen na internet i pracovat offline, pokud internetové připojení není k dispozici. Rovněž vzhled a ovládání mobilní aplikace bylo přizpůsobeno možnostem displeje mobilního zařízení, na rozdíl od webové aplikace.

V první kapitole této práce se zabývám analýzou dosavadního vývoje a funkčnosti webové aplikace Journal. Zde rovněž shrnuji poznatky z průzkumu konkurenčních aplikací. Ve druhé kapitole specifikuji funkční a nefukční požadavky kladené na aplikaci. Třetí kapitola obsahuje návrh mobilní aplikace, tedy použité technologie, doménový model a uživatelské rozhraní. Ve čtvrté kapitole popisuji vlastní implementaci mobilního klienta. Poslední, pátá kapitola se zabývá testováním a její součástí je programátorská příručka.

Cílem této bakalářské práce je tedy zanalyzovat funkcionalitu webové aplikace Journal, navázat na její REST API a zpřístupnit uživatelům mobilních telefonů její mobilní verzi, přizpůsobenou možnostem chytrého telefonu. Součástí této bakalářské práce je rovněž vývojářská příručka k této mobilní aplikaci. V rešeršní části této práce se věnuji podobně zaměřeným mobilním aplikacím pro vedení deníkových záznamů, zhodnocuji jejich klady, které uživatelé v recenzích vyzdvihují, a případné nedostatky, kterým je nutno dále věnovat pozornost. V závěru své práce pak shrnuji výsledky uživatelského testování mnou vytvořeného mobilního klienta pro vedení deníkových záznamů.

Kapitola 1

Analýza

1.1 Dosavadní vývoj webové aplikace

Tato část kapitoly se zabývá historií dosavadního vývoje webové aplikace Journal. Vývoj webové aplikace Journal započal v zimním semestru akademického roku 2017/18 v předmětu BI-SP1 pod vedením Ing. Zdeňka Ryboly, Ph.D. Tým ve složení Vojtěch Kraus, Matyáš Herman, Martin Vatrt a Tomáš Voldřich tehdy vytvořil analytickou dokumentaci a první verzi webové aplikace.

Poté se v rámci předmětu BI-SP2 v letním semestru akademického roku 2017/18 práce na této webové aplikaci rozdělila na dvě části. První tým ve složení Vojtěch Kraus a Matyáš Herman pokračoval ve vývoji webové aplikace a to jak na frontendu, tak na backendu. Tento tým rovněž vytvořil REST API (Representational State Transfer Application Programming Interface) pro mobilní aplikaci.

Druh tým složený z Igora Tsaregorodtseva a Zhanybeka Sadvakassova zahájil práci na mobilní aplikaci pro operační systém Android. Tato mobilní verze webové aplikace Journal neměla obsahovat všechny funkce webové aplikace, ale měla zpřístupňovat hlavní funkcionalitu aplikace Journal na mobilním zařízení.

V zimním semestru akademického roku 2018/19 dokončil práci na webové aplikaci Journal Matyáš Herman jakožto svou bakalářskou práci. Práci na mobilním klientovi webové aplikace Journal jsem znovu zahájila já v letním semestru akademického roku 2019/20 v rámci mé bakalářské práce, přičemž jsem navázala na REST API pro mobilního klienta, vyvinuté prvním týmem ve složení Vojtěch Kraus a Matyáš Herman v letním semestru akademického roku 2017/18.

1.2 Analýza funkčnosti webové aplikace

Tato kapitola se zabývá analýzou funkčnosti webové aplikace, podle jejíhož vzoru bude navrhnuta aplikace mobilní.

1.2.1 Přihlášení a registrace

Uživatel se může do aplikace přihlásit po zadání přihlašovacích údajů. Budou mu pak zpřístupněny všechny funkce aplikace. Pokud uživatel nemá zřízený uživatelský účet, může jej zřídit registrací.

Uživatel může zobrazit svůj uživatelský profil, kde uvidí své jméno a příjmení, může nahrát profilový obrázek či změnit své přístupové heslo.

1.2.2 Události

Stěžejním bodem celé aplikace jsou události. U událostí se evidují následující atributy: název, datum, popis, tagy, osoby, lokace, přílohy, datum vytvoření a datum poslední úpravy.

Uživatel může události vytvářet, editovat a mazat. Událostem lze také přidávat podudálosti.

1.2.3 Seznam událostí

Na úvodní stránce aplikace je zobrazen seznam událostí seřazených podle data. U událostí jsou uvedeny vybrané atributy, jako je: datum, název, lokace, tagy a osoby. U každé položky seznamu je tlačítko pro zobrazení detailu události, upravení nebo smazání události, a přidání podudálosti.

Seznam událostí je možné filtrovat podle atributů: název, tagy, datum události a zúčastněné osoby.

1.2.4 Kalendář

Aplikace umožnuje zobrazit události ve formě kalendáře zobrazeného po měsících. Po kliknutí na událost vyznačenou v kalendáři se otevře dialog s názvem a datem události a s tlačítkem pro přechod na detail události.

1.2.5 Mapa

Na mapě jsou markerem označeny události, které mají přiřazenou lokaci. Markery událostí s lokací blízko sebe se shlukují do klastrů. V pravé části obrazovky se průběžně zobrazuje seznam událostí, jejichž markery jsou viditelné v aktuálním výřezu mapy.

Po kliknutí na marker je zobrazeno informační okno s názvem události, datem, adresou a tlačítkem pro přechod na detail události. Po kliknutí na klastr markerů je mapa přiblížena tak, že se klastr rozdělí na jednotlivé markery.

1.2.6 Seznam přátel

V aplikaci je možné udržovat seznam přátel a tyto přátele pak přidávat k jednotlivým událostem. U každého z přátel je možné uložit jméno a email, aplikace pak ukládá počet událostí, ke kterým je daný přítel přiřazen. U záznamu přítele lze měnit atributy, anebo ho smazat.

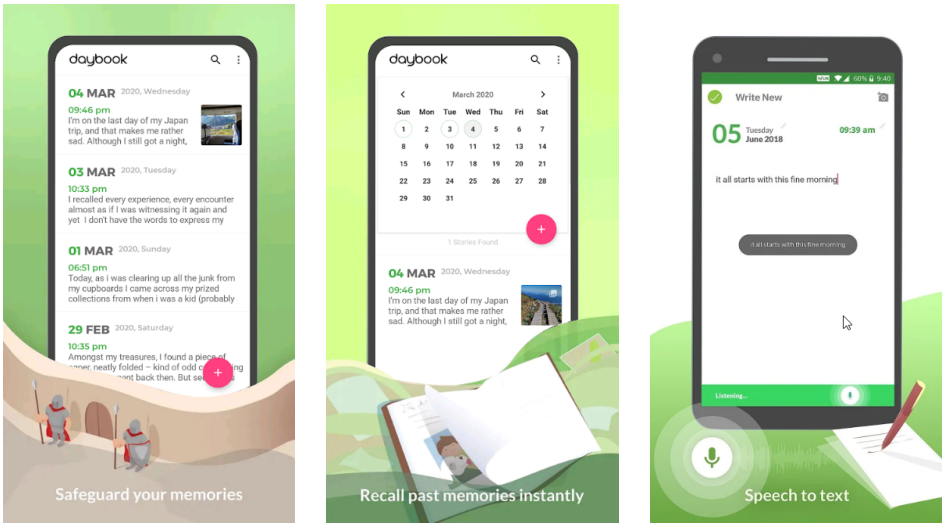
1.3 Konkurenční aplikace

V současnosti existuje na platformě Android řada dostupných mobilních aplikací pro vedení osobního deníku. Prozkoumala jsem funkcionalitu pěti nejoblíbenějších z nich [11], abych zjistila, zda nabízejí podobnou či dokonce stejnou funkcionalitu, jako mnou vyvíjená mobilní aplikace.

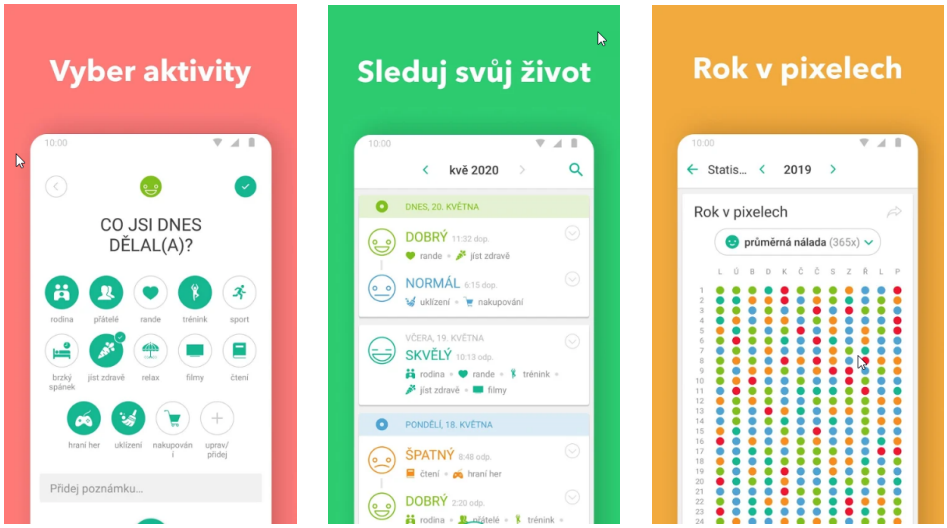
1.3.1 Daybook

Deníková aplikace Daybook[1] je uvedena na prvním místě seznamu doporučených aplikací pro vedení osobního deníku na webové stránce Android Authority. Uživatelské rozhraní je jednoduché a nerozptylující, možnosti formátování textu omezené, ale zato je zde dostupná možnost převádět hlasové záznamy na text a naopak, možnost předčítání textových záznamů (Obrázek 1.1).

Uživatelé si však stěžují na vysoké měsíční předplatné ve srovnání s jinými aplikacemi, které mají oproti Daybooku více funkcí, například tzv. moodtracker, tedy funkci pro sledování a grafické vyhodnocení průběhu nálad uživatele v uplynulém čase.



Obrázek 1.1: Snímky obrazovek aplikace Daybook



Obrázek 1.2: Snímky obrazovek aplikace Daylio

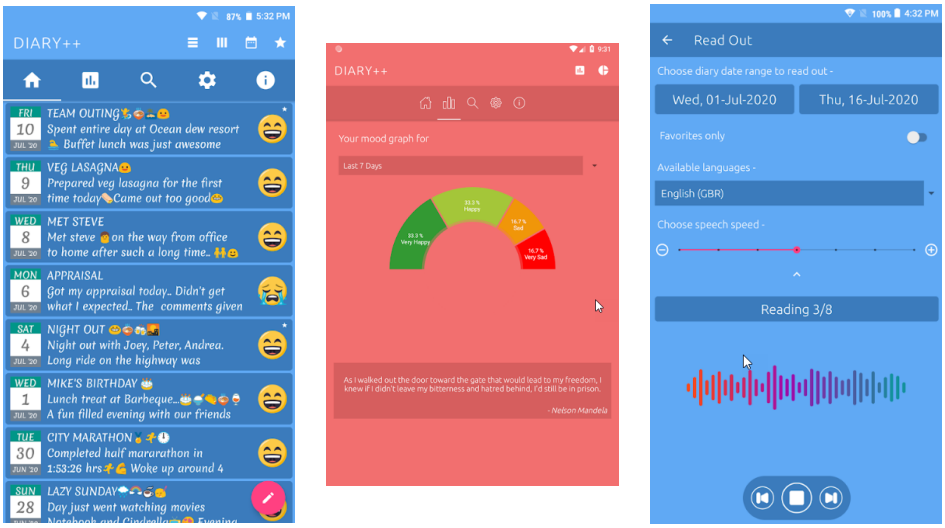
1.3.2 Daylio

Daylio[2] umožňuje vytvořit soukromý deník i bez psaní textových poznámek. Denní záznam se může skládat z několika druhů ikon vybraných z velké databáze předdefinovaných ikon, například ikony vyjadřující denní náladu, několika ikon pro aktivity uskutečněné daného dne, případně lze přidat další předdefinované i vlastní obrázky. Lze samozřejmě přidávat i klasické textové poznámky (Obrázek 1.2).

Uživatelé u této aplikace hodnotí kladně, že mohou sledovat průběh svých nálad za uplynulé časové období, například měsíc nebo rok. Tento formát jim může pomoci k lepšímu pochopení jejich zvyklostí. Rovněž jsou uživatelé spokojení s velice rychlým způsobem vytváření denních záznamů, které jim díky předdefinovaným ikonám nezabere příliš času, tento typ tzv. mikrodeníku je tedy vhodný pro velice zaneprázdněné uživatele.

1.3.3 Diary ++ (v české verzi Můj deník, v anglické verzi Diary with lock)

Diary ++[3] je další z mobilních aplikací pro vedení osobního deníku na platformě Android. Je dodávána s barevným a příjemným uživatelským rozhraním. Poskytuje běžné funkce pro psaní textových poznámek doplněných o velký výběr z dostupných emotikonů. Mezi další patří Grafické funkce pro sledování nálad, možnost nastavení připomenutí, přidávání obrázků a další. Zajímavým rysem je při neoprávněném pokusu o otevření deníku možnost vyfotit narušitele, pokud je nastaveno oprávnění přední kamery (Obrázek 1.3).



Obrázek 1.3: Snímky obrazovek aplikace Diary++

1.3.4 Diaro

Diaro[4] je rozšířená deníková aplikace, snadno použitelná, s jednoduchým intuitivním designem bez zbytečného rozptylování. Je to zatím jediná deníková aplikace, u které jsem našla geografické označování lokací událostí. Samozřejmostí je možnost přidávání obrázků jako příloh k událostem (Obrázek 1.4).

Z recenzí k této aplikaci lze vyčíst, že uživatelé dříve považovali tuto deníkovou aplikaci za špičku mezi konkurencí, ale nyní mají pocit, že v dnešní době poněkud zastarala. Uživatelům vadí synchronizace přes úložiště Dropbox, datově omezené v neplacené verzi na pouhé 2 MB. Dále uživatelé aplikaci vytýkají chudý výběr možností formátování textu. A v neposlední řadě by uživatelé po mnoha urgencích přivítali možnost nastavení připomínek, aby nemuseli vést deník souběžně i v jiné aplikaci, která toto umožňuje.

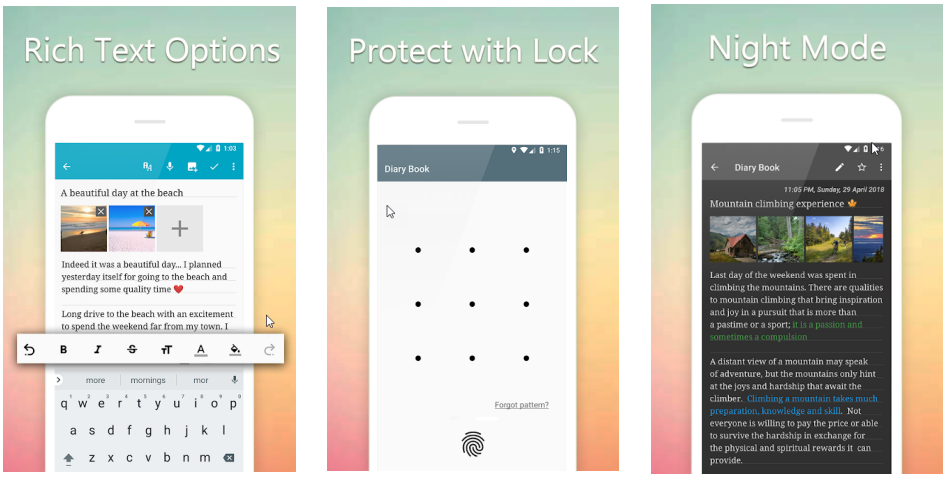
1.3.5 Diary Book

Diary Book[5] je další propracovaná a uživatelsky příjemná mobilní aplikace pro vedení osobního deníku. Zahrnuje obvyklý sortiment požadovaných funkcionalit, jako je možnost nastavení denní připomínky k psaní, dobré možnosti formátování textu, noční režim, cloudovou synchronizaci s Google Diskem a různé typy zámků. Rovněž oblíbená je možnost převodu textu na řeč ve zvoleném jazyce a režim automatického zámku (Obrázek 1.5).

1.3.6 Zhodnocení

Vybrala jsem si pět nejoblíbenějších konkurenčních deníkových aplikací a u nich jsem vyhodnocovala jejich dostupné funkcionality, na které uživatelé

Obrázek 1.4: Snímky obrazovek aplikace Diaro



Obrázek 1.5: Snímky obrazovek aplikace Diary Book

těchto aplikací kladou důraz. Snažila jsem se zjistit, která z těchto aplikací pokrývá nejvíce funkcionalit žádaných uživateli, a rovněž mě zajímalo, které z těchto aplikací umožňují geografické označování lokací událostí, což je funkcionalita, která zajímá mě.

Při analýze dostupných funkcionalit konkurenčních deníkových aplikací (Tabulka 1.1) jsem zjistila, že všechny aplikace z mého výběru nabízejí hruba stejné funkcionality, které uživatelé vyžadují. Pokud se některá z aplikací opozdí za konkurencí, uživatelé si toho velmi rychle všimnou a urychleně přecházejí ke konkurenci, která požadovanou funkcionalitu nabízí. Příkladem může být chybějící tzv. moodtracker (sledovač nálad) v deníkové aplikaci Daybook, který je ovšem plánovaný v příští aktualizaci. V takovém případě ovšem též svou roli hraje snadnost exportu uložených dat a kompatibilita s formátem

dat importovaných do jiných aplikací, protože uživatelé samozřejmě nechtějí přijít o své minulé deníkové záznamy.

Druhým kritériem je pořizovací cena aplikace, která přijde v potaz, když je uživatel dostatečně zlákán bezplatnou verzí aplikace a vidí, že za malý jednorázový poplatek získá mnohá užitečná rozšíření. V tomto případě opět prohrává aplikace Daybook, protože její měsíční předplatné je zhruba ve stejné výši, jako jednorázový poplatek u ostatních konkurenčních aplikací.

Třetím kritériem, podle kterého si uživatelé vybírají svou deníkovou aplikaci, je možnost neomezeného a bezplatného objemu zálohování dat. V tomto kritériu zaostává aplikace Diaro, která jediná používá poněkud zastaralou možnost zálohování dat na úložišti Dropbox, která je bezplatná jen do určitého objemu dat.

Ze všech těchto dostupných aplikací se poněkud vymyká aplikace Daylio, jejíž možnosti jsou sice velmi omezené, ale zato pořizování deníkových záznamů je tak rychlé, že nenáročným či velmi zaneprázdněným uživatelům tento způsob vedení tzv. mikrodeníku plně vyhovuje a hodnotí jej velmi kladně.

Zaujalo mě, že jsem nenašla žádnou recenci, ve které by si uživatelé stěžovali na chybějící možnost přidávání tagů k záznamům anebo geografické označování lokací událostí, či naopak je vyzdvihovali u dvou aplikací, které je nabízejí: tagování nabízí aplikace Diary Book, geografické označování lokací událostí aplikace Diaro. Tyto dvě funkcionality jsem implementovala ve svém mobilním klientovi, protože obě mi připadají velice zajímavé a přínosné.

Funkčnost	Daybook	Daylio	Diary ++	Diaro	Diary Book
Možnosti formátování textu	omezené	omezené	omezené	omezené	ANO
Nastavení připomínek	Nenašla jsem	ANO	ANO	Nenašla jsem	ANO
Kalendář	ANO	ANO	ANO	ANO	ANO
Prohledávání záznamů a filtrování	ANO	Nenašla jsem	ANO	ANO	ANO
Možnost přidání tagů k záznamům	Nenašla jsem	Nenašla jsem	Nenašla jsem	Nenašla jsem	ANO
Přidávání příloh k událostem	obrázky	Nenašla jsem	obrázky	obrázky	obrázky
Geografické označování lokací událostí	V příští aktualizaci	Nenašla jsem	Nenašla jsem	ANO	Nenašla jsem
Ochrana osobních údajů	PIN, otisk prstu	PIN	PIN, vzor (pattern), otisk prstu	PIN, vzor (pattern), otisk prstu	PIN, vzor (pattern), otisk prstu, heslo
Zálohování dat	Cloud	Google Disk	Google Disk	Dropbox	Google Disk
Moodtracker (funkce grafu pro sledování nálad)	V příští aktualizaci	ANO	ANO	ANO	ANO

Tabulka 1.1: Dostupné funkcionality konkurenčních deníkových aplikací

Kapitola 2

Specifikace požadavků

Tato kapitola specifikuje všechny funkční a nefunkční požadavky kladené na aplikaci.

2.1 Funkční požadavky

Funkční požadavky byly definovány na základě funkčnosti, která byla identifikována analýzou webové aplikace *Journal*. Funkční požadavky byly definovány s cílem zpřístupnit hlavní funkcionalitu webové aplikace v mobilním klientovi.

F1 Přihlášení Uživatel se bude moci do aplikace přihlásit s přihlašovacími údaji, které získal při registraci do webové aplikace. Pro větší pohodlí, uživatel zůstane přihlášen v mobilní aplikaci, dokud se sám neodhlásí. Pokud nový uživatel nemá ještě založený účet, bude odkázán na registrační stránku webové aplikace.

F2 Seznam událostí V aplikaci bude možné zobrazit všechny události uživatele ve formě seznamu. Každá položka bude obsahovat název události, výňatek popisu a datum události. Případně bude zobrazeno počáteční a koncové datum, pokud se jedná o vícedenní událost.

F3 Vyhledávání Události bude možné vyhledávat podle slov obsažených v jejich názvu nebo popisu.

F4 Mapa Události, které mají přiřazenou lokaci, bude možné zobrazit na mapě. Lokace událostí budou na mapě vyznačeny markery, které se pro lepší přehlednost budou shlukovat do klastrů, při velké hustotě událostí na jednom místě. Seznam událostí v dané lokalitě pak bude dostupný po kliknutí na příslušný klastr.

F5 Kalendář Události bude možné zobrazit ve formě kalendáře. Záznam v kalendáři bude obsahovat název události. A podle doby trvání události bude zasahovat do jednoho či více dnů.

F6 Detail události Každou událost bude možné zobrazit ve formě detailu. V detailu události bude uveden především název události, její popis, datum a pokud událost obsahuje také tagy, přílohy a lokace, budou zobrazeny i tyto nepovinné atributy.

F7 Vytvoření nové události V aplikaci bude uživateli umožněno přidat novou událost se všemi jejími atributy popsanými v detailu, přičemž název a datum budou jediné povinné atributy.

F8 Upravení události U události bude možné změnit libovolné atributy dříve definované uživatelem.

F9 Smazání události Vytvořenou událost bude možné smazat.

2.2 Nefunkční požadavky

Nefunkční požadavky, popsané v této kapitole, definují omezení kladená na aplikaci.

N1 REST API Aplikace bude komunikovat s REST rozhraním dodaným zadavatelem, ze kterého bude především získávat data o událostech a také posílat data při vytvoření nové události či její úpravě.

N2 Cache Data přijatá ze serveru bude aplikace ukládat do lokální databáze. Tato data pak budou zobrazena uživateli při nedostupnosti sítě nebo chybě serveru a zajistí tak uživateli lepší uživatelský zážitek.

N3 Přívětivý design Aplikace bude mít intuitivní design, který bude tématicky korespondovat se vzhledem webové aplikace.

N4 Aplikace pro Android Aplikace bude vytvořena pro uživatele mobilního operačního systému Android.

2.3 Případy užití

2.3.1 UC1 Přihlášení

Tento scénář popisuje první interakci uživatele s aplikací. Po provedení všech kroků bude uživatel přihlášen a bude tomu tak i při pozdějším znovuspuštění aplikace, pokud se sám neodhlásí.

Aktér Nepřihlášený uživatel

Hlavní scénář Přihlášení s přihlašovacími údaji.

1. Uživatel nainstaluje a otevře aplikaci.
2. Aplikace zobrazí přihlašovací obrazovku.
3. Uživatel vyplní přihlašovací údaje a stiskne tlačítko pro přihlášení.
4. Aplikace otevře obrazovku se seznamem událostí.

Alternativní scénář Přihlášení bez přihlašovacích údajů.

1. Scénář začíná v 2. kroku hlavního scénáře, pokud uživatel nemá přihlašovací údaje.
2. Uživatel klikne na odkaz *Register* na přihlašovací obrazovce.
3. Uživatel je přesměrován na registrační stránku webové aplikace.
4. Uživatel vyplní registrační údaje a odešle žádost o registraci.
5. Uživatel obdrží registrační email a potvrdí registraci.
6. Uživatel se vrátí na přihlašovací obrazovku mobilní aplikace.
7. Scénář pokračuje v bodě 3. hlavního scénáře.

2.3.2 UC2 Vyhledání události

Tento případ použití umožňuje uživateli vyhledat požadovanou událost a zobrazit její detail.

Aktér Přihlášený uživatel

Hlavní scénář Vyhledání události v seznamu

1. Uživatel otevře aplikaci a jako úvodní obrazovka se mu zobrazí seznam událostí.
   a) Alternativně, pokud uživatel nezná parametry vyhledávané události, skroluje seznamem a pokračuje ve 4. kroku tohoto scénáře.
2. Uživatel klikne na tlačítko vyhledávání v horní liště aplikace a zobrazí se mu políčko pro vyhledávání.

3. Uživatel zadává klíčové slovo, podle kterého chce vyhledávat, a aplikace v reálném čase vybírá události, jejichž název nebo popis obsahují toto klíčové slovo.
4. Uživatel vybere požadovanou událost a po kliknutí na její záznam se mu otevře stránka s detailem události.

Alternativní scénář Vyhledání události na mapě.

1. Uživatel otevře menu kliknutím na ikonu v horní liště aplikace.
2. Po kliknutí na položku menu *Map* se uživateli zobrazí mapa s markery označujícími lokace událostí.
   a) Alternativně, pokud mnoho událostí má svou lokaci na stejném místě, případně blízko sebe, zobrazí se uživateli na mapě markery klastrů, do kterých jsou události shromážděny.
   b) Uživatel klikne na klastr s markery a zobrazí se mu seznam událostí, které jsou v této lokaci.
   c) Po kliknutí na vybraný záznam v seznamu bude uživatel přesměrován na detail události.
3. Kliknutím na vybraný marker se uživateli zobrazí informační pop-up okno s název a datem události.
4. Po kliknutí na informační pop-up okno bude uživatel přesměrován na detail události.

2.3.3 UC3 Zobrazení události v kalendáři

Aktér Přihlášený uživatel

Hlavní scénář

1. Uživatel vybere v menu aplikace položku *Calendar*
2. Uživateli bude zobrazen kalendář, kde budou jednotlivé události vyznačeny podle svého data.

2.3.4 UC4 Smazání události

Tento případ užití umožňuje uživateli smazat vybranou událost.

Aktér Přihlášený uživatel

Hlavní scénář

1. Include(UC2 Vyhledání události)
2. V horní liště detailu události klikne uživatel na tlačítko *Smazat*.
3. Aplikace otevře dialog s dotazem, zda chce uživatel opravdu událost smazat.
4. Uživatel potvrdí smazání události.

2.3.5 UC5 Vytvoření nové události

Tento případ užití umožňuje uživateli přidat novou událost.

Aktér Přihlášený uživatel

Hlavní scénář

1. Na stránce seznamu klikne uživatel na Floating Action Button (dále jen FAB) označující hlavní akci obrazovky.
2. Aplikace zobrazí formulář pro přidání nové události.
3. Uživatel vyplní povinná pole: název a datum události, případně počáteční a koncové datum, jedná-li se o vícedenní událost.
4. Nepovinně pak uživatel může přidat k události tagy a popis události.
   Extend <Přidání lokace události>
   Extend <Přidání příloh k události>
5. Uživatel potvrdí vytvoření nové události kliknutím na tlačítko *Save* a je přesměrován na detail nově vytvořené události.

2.3.6 UC6 Upravení události

Aktér Přihlášený uživatel

Hlavní scénář

1. Include(UC2 Vyhledání události)
2. V horní liště detailu události klikne uživatel na tlačítko *Upravit*.
3. Aplikace otevře obrazovku s formulářem pro úpravu události.
4. Uživatel změní požadovaná pole. Povinná pole název a datum musí zůstat neprázdná.
   Extend <Přidání lokace události>
   Extend <Přidání příloh k události>
5. Uživatel potvrdí upravení události.

2.3.7 UC7 Přidání lokace k události

Tento případ užití umožní uživateli přidat či upravit lokaci u nově vytvářené nebo upravované události.

Aktér Přihlášený uživatel

Hlavní scénář Vyhledání lokace podle adresy

1. Ve formuláři pro přidání/upravení lokace uživatel klikne na tlačítko *Select location*.

2. Aplikace zobrazí dialog pro přidávání lokace.
3. Uživatel zadá název požadovaného místa do pole *Title* a klikne na tlačítko *Vyhledávat*.
4. Aplikace nalezne lokaci podle zadaného názvu, zobrazí její adresu v poli *Address* a na mapě nastaví její souřadnice.
   a) Pokud aplikace nenalezla žádnou adresu podle zadaného názvu, pokračuje uživatel bodem 3 tohoto scénáře, případně alternativním scénářem.
5. Uživatel klikne na tlačítko *Save* a uloží zvolenou lokaci.

Alternativní scénář Vyhledání lokace na mapě

1. Scénář začíná v bodě 2 hlavního scénáře.
2. Uživatel přesune marker na mapě na požadovanou pozici. Při posunu markeru jsou mu průběžne zobrazovány souřadnice aktuální pozice.
3. Zadá název vybrané lokace do pole *Title*
4. Scénář pokračuje v bodě 5 hlavního scénáře.

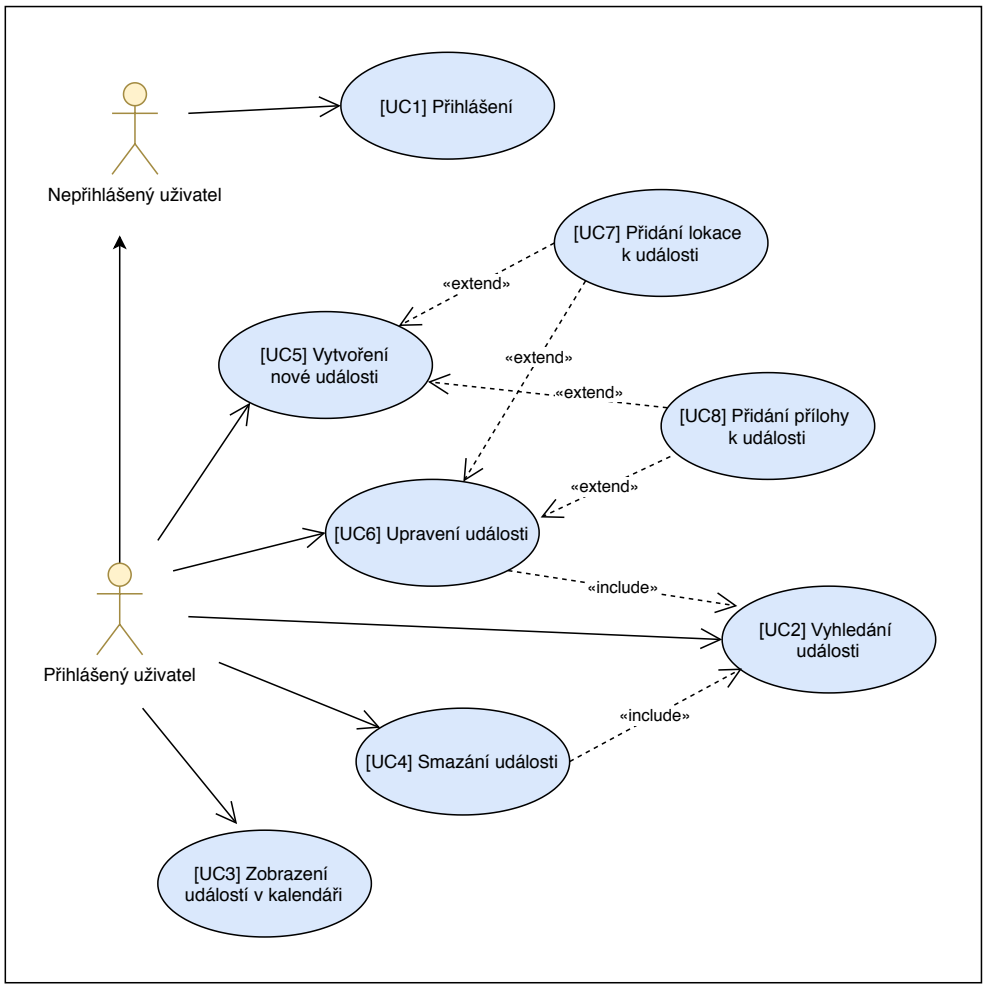
2.3.8 UC8 Přidání přílohy k události

Tento scénář umožní uživateli přidat přílohu k vytvářené či upravované události.

Aktér Přihlášený uživatel

Hlavní scénář

1. Ve formuláři pro přidání/upravení lokace uživatel klikne na tlačítko *Přidat přílohu*.
2. Aplikace zobrazí dialog požadující povolení k přístupu do datového úložiště zařízení.
3. Uživatel povolí aplikaci přístup kliknutím na tlačítko *Allow*.
4. Uživatel vybere v úložišti zařízení požadovanou přílohu a potvrdí její přidání.



Obrázek 2.1: Diagram případů užití

	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
F1	x							
F2		x		x		x		
F3		x		x		x		
F4		x		x		x		
F5			x	x				
F6		x		x		x		
F7					x		x	x
F8						x	x	x
F9				x				

Tabulka 2.1: Pokrytí funkčních požadavků případy užití

Kapitola 3

Návrh

3.1 Výběr programovacího jazyka

Programátor má na výběr několik programovací jazyků. Nejpopulárnějšími a nejrozšířenějšími z nich jsou Java a Kotlin. Oba jsou oficiálními jazyky pro vývoj aplikací.

Java byla dlouhou dobu jediným oficiálním jazykem pro vývoj Android aplikací. V roce 2017 byl Kotlin prohlášen "sekundárním"oficiálním jazykem a v roce 2019 byl dokonce oznámen jako doporučený jazyk pro vývoj Android aplikací [6], mimo jiné také kvůli vleklému soudnímu sporu společnosti Google se společností Oracle o vlastnická práva na jazyk Java [7]. Proto jsem si ho také zvolila pro vývoj mé aplikace.

Ve zbytku kapitoly si představíme některé významné charakteristiky Kotlinu.

3.1.1 Kotlin

Kotlin získal takovou popularitu mezi programátory, že pouhé dva roky po stanovení Kotlinu druhým oficiálním jazykem pro vývoj aplikací se stal Googlem doporučeným jazykem a "všechny nové projekty by měly být psány v Kotlinu". Mnoho populárních aplikací používá pro svůj vývoj Kotlin, mimo jiné Slack, Airbnb, Netflix, a další [8]. Následuje seznam některých jeho předností oproti Javě.

Stručnější syntaxe Kotlin poskytuje jednodušší a stručnější syntaxi v porovnání s výmluvnou Javou. Zároveň je ale s Javou kompatibilní a jeden projekt může obsahovat jak soubory v jazyce Kotlin, tak i v jazyce Java.

Nullpointer safety Kotlin implicitně nedovoluje přiřadit do proměnné hodnotu *null*, pokud není explicitně specifikováno, že proměnná je *Nullable*.

Funkční programování Kotlin poskytuje podporu pro lambda výrazy, která je v Javě dostupná až od verze 8.

Korutiny Korutiny poskytují způsob pro psaní asynchronního a neblokujícího kódu. Stejně jako vlákna mohou běžet paralelně a komunikovat spolu, oproti vláknům jsou však mnohem ”levnější”, přičemž mnoho korutin může být spuštěno v jednom vlákně.

Extension functions Kotlin umožňuje přidat metody dostupné ve třídě bez nutnosti z ní dědit.

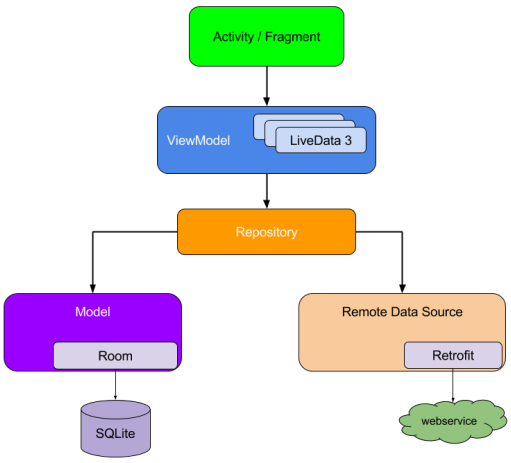
3.2 Architektura

Mezi nejrozšířenější návrhové vzory pro Android aplikace patří Model-View-Controler (MVC), Model-View-Presenter (MVP) a Model-View-ViewModel (MVVM). Všechny tyto návrhové vzory se snaží dodržovat třívrstvou architekturu pro lepší přehlednost a škálovatelnost aplikace. Dříve záleželo zcela na programátorovi, jakou architekturu zvolí. Nezávisle na architektuře, stále se vyskytovaly problémy se správou životního cyklu *aktivit*, což je problematika specifická pro mobilní aplikace.

Google představil v roce 2017 komponenty pracující s životním cyklem[9], které podporují MVVM návrhový vzor, a ten se tak stal doporučenou architekturou pro návrh Android aplikací [10].

3.2.1 MVVM

Model-View-ViewModel architektura se snaží dostát principu *Separation of concerns* rozdělením tříd do tří vrstev View, ViewModel a Model, které odpovídají vrstvám třívrstvé architektury: prezenční, bussiness a datová, v tomto pořadí (Obrázek 3.1).



Obrázek 3.1: MVVM architektura přizpůsobená pro Android

3.2.2 Android komponenty

V následující kapitole si představíme některé komponenty poskytnuté Googlem, které mají usnadnit implementaci Android aplikací, mimo jiné také podporou pro návrhový vzor MVVM.

Aktivity a Fragmenty Představují obrazovky aplikace, se kterými uživatel bude interagovat. Specificky pro potřeby mobilních aplikací procházejí několika fázemi svého životního cyklu (Obrázek 3.2).

LiveData LiveData je pozorovatelný data holder, který upozorní pozorovatele na změny. Na rozdíl od jiných observerů (typ návrhového vzoru) si je komponenta LiveData vědoma životního cyklu aktivity.

ViewModel ViewModel komponenta usnadňuje implementaci vrstvy ViewModel v MVVM architektuře. Na rozdíl od aktivity uchovává data přes všechny fáze životního cyklu a aktivita při spuštění dostane přiřazen vždy stejný ViewModel, jako měla předtím. Navíc více aktivit či fragmentů může sdílet stejný ViewModel a předávat si pomocí něj data mezi sebou.

3.3 Použité technologie a knihovny třetích stran

Následující kapitola stručně představí vybrané technologie pro usnadnění vývoje Android aplikací a případně uvede, jak budou užitečné pro tento projekt.

3.3.1 Retrofit

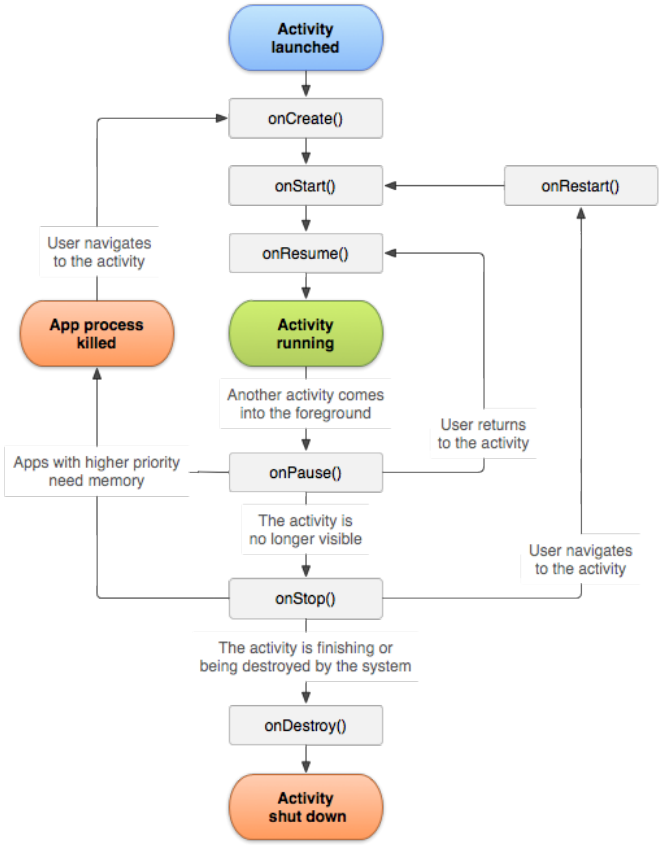
Retrofit[11] je HTTP klient pro Android a Javu, který se snaží usnadnit komunikaci s REST rozhraním. Jednoduchá práce s ním spočívá v definování rozhraní s metodami pro komunikaci se serverem a vytvoření objektu rozhraní pomocí builderu. Ukázka definice takovéto metody je na obrázku 3.3.

Retrofit je schopný serializovat JSON odpovědi přijaté ze serveru do předem definovaných Java tříd. K tomu potřebuje convertor - v našem případě je použit Gson.

Výhodou je možnost upravit návratové hodnoty metod posílajících požadavky na server tak, aby vracely LiveData, což přijde vhod při implementaci návrhového vzoru MVVM.

3.3.2 Room

Room[12] knihovna poskytuje vrstvu abstrakce nad databází SQLite a zjednodušuje tak práci s ní. Room se skládá ze tří částí (Obrázek 3.4):



Obrázek 3.2: Životní cyklus aktivity

```
@GET( value: "/rest/event")
fun getEvents(
    @Header( value: "Authorization") authHeader: String,
    @Header( value: "Authorization-Client") authClientHeader: String
): LiveData<ApiResponse<EventRequest>>
```

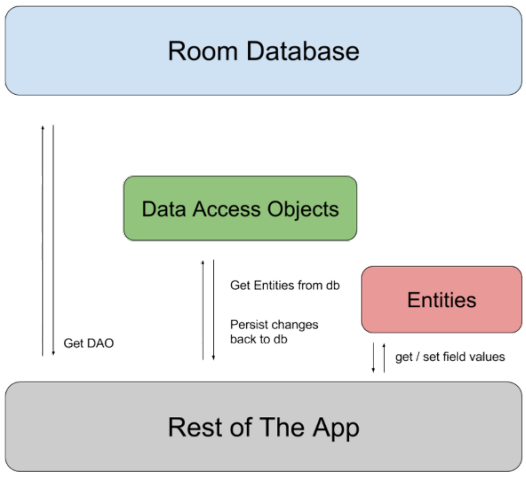
Obrázek 3.3: Retrofit - příklad signatury metody rozhraní

Databáze Abstraktní třída, která definuje entity databáze a metody pro získání DAO; bude pomocí Room implementována jako přístupový bod do databáze.

Entita Představuje tabulku v databázi. Je implementována jako POJO, která je označena anotacemi pro Room.

Data Access Object (DAO) Obsahuje metody umožňující získání entit z databáze a ukládání entit do databáze.

Naše aplikace bude používat Room databázi pro cachování dat přijatých ze serveru pro případ nedostupnosti internetového připojení.



Obrázek 3.4: Diagram architektury Room

3.3.3 Google Maps SDK for Android

Developerský balíček Google Maps[13] umožní přidávání map do aplikace. Mapy mají známý vzhled map z Google Maps, lze je ovládat ručními gesty a kromě jiného také přidávat markery na vybrané lokace.

3.3.4 Glide

Glide[14] je rychlý a efektivní framework pro načítání a správu médií na platformě Android. Pomocí jednoduchého rozhraní umožňuje dekódování, cachování nebo změnu velikosti obrázků stažených ze serveru. Stručnost potřebného kódu je ilustrována na obrázku 3.5.

```
Glide.with(requireActivity())
    .load(glideUrl)
    .into(image)
```

Obrázek 3.5: Použití Glide knihovny pro nahrání obrázku

3.3.5 Koin

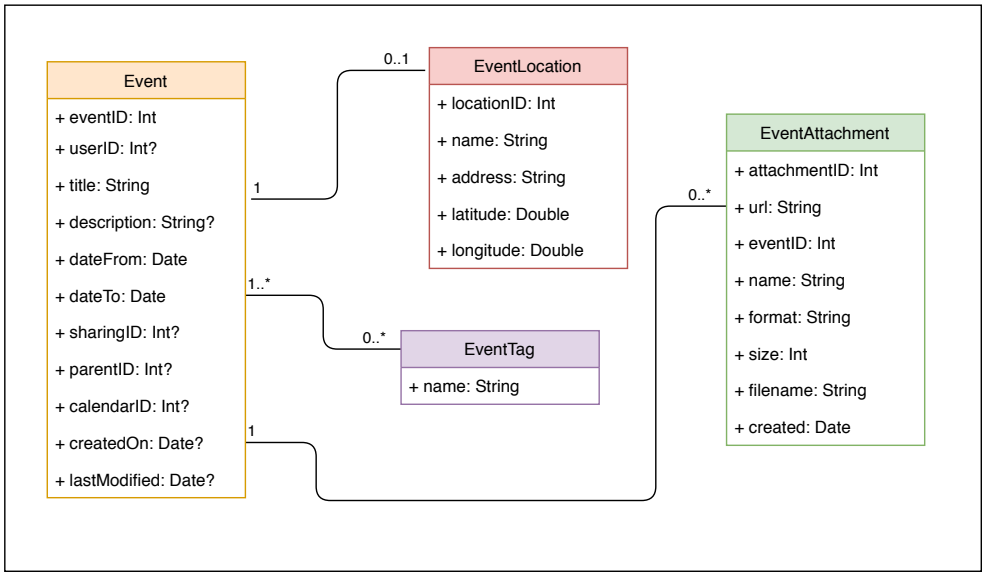
Koin[15] je Dependency Injection (DI) framework napsaný pouze v Kotlinu. V porovnání s komplexnějšími DI frameworky, jako je například Dagger2, je jednodušší na používání a zároveň pro potřeby naší aplikace dostačující.

V aplikaci je použit pro získávání závislostí na objektech, které implementují návrhový vzor Singleton, jako např. *Repository* nebo *Databáze*.

3.4 Doménový model

Doménový model zachycuje entity figurující v aplikaci, jejich atributy a vazby mezi nimi. V našem případě bude doménový model odpovídat databázovému modelu, neboť komponenta *Room* se postará o uložení objektu do databáze. Stěžejní entitou je *Event*, ostatní entity pak mají k této entitě vztah atributu.

Výsledný doménový model je k vidění na obrázku 3.6



Obrázek 3.6: Doménový model

Kapitola 4

Implementace

Následující kapitola představí jednotlivé komponenty aplikace, rozdělené do tří částí odpovídajících vrstvám MVVM architektury.

4.1 Model

Model představuje datovou vrstvu aplikace. Nalezneme zde tedy třídy a rozhraní pro práci s daty uloženými v lokální databázi či přijímanými ze serveru.

4.1.1 REST API

Rozhraní pro komunikaci se zadaným REST API je v naší aplikaci realizováno pomocí knihovny Retrofit. V rozhraní byly implementovány metody pro přijímání a odesílání událostí na server (Obrázek 4.1).

```
@GET( value: "/rest/event/{id}")
fun getEvent(
    @Header( value: "Authorization") authHeader: String,
    @Header( value: "Authorization-Client") authClientHeader: String,
    @Path( value: "id") eventId: Int
): LiveData<ApiResponse<Event>>

@POST( value: "/rest/event")
fun createEvent(
    @Header( value: "Authorization") authHeader: String,
    @Header( value: "Authorization-Client") authClientHeader: String,
    @Body event: OutcomingEvent
): LiveData<ApiResponse<Event>>
```

Obrázek 4.1: Ukázka Retrofit rozhraní

Retrofit call adapter Metody rozhraní byly přizpůsobeny tak, aby vracely objekt LiveData. Toho bylo dosaženo přidáním *Retrofit Call Adapteru* (Obrázek 4.2).

```
class LiveDataCallAdapter<R>(private val responseType: Type) :
    CallAdapter<R, LiveData<ApiResponse<R>>> {

    override fun responseType() = responseType

    override fun adapt(call: Call<R>): LiveData<ApiResponse<R>> {
        return object : LiveData<ApiResponse<R>>() {
            private var started = AtomicBoolean( initialValue: false)
            override fun onActive() {
                super.onActive()
                if (started.compareAndSet( expect: false,  update: true)) {
                    call.enqueue(object : Callback<R> {
                        override fun onResponse(call: Call<R>, response: Response<R>) {
                            postValue(ApiResponse.create(response))
                        }

                        override fun onFailure(call: Call<R>, throwable: Throwable) {
                            postValue(ApiResponse.create(throwable))
                        }
                    })
                }
            }
        }
    }
}
```

Obrázek 4.2: Retrofit call adapter pro transformaci dat přijatých ze serveru

4.1.2 Cache

Databáze pro lokální cachování dat přijatých ze serveru je v naší aplikaci realizována pomocí Room - abstrakční vrstvou nad SQLite databází.

To nám umožňuje specifikovat tabulky v databázi pouze pomocí tříd označených vhodnými *anotacemi* (Obrázek 4.3).

```
@Entity(tableName = "events")
data class Event(
    @PrimaryKey
    @ColumnInfo(name = "event_id")
    var eventId: Int,
    var userId: Int?,
    var title: String,
    var description: String,
```

Obrázek 4.3: Část entity

Data Access Object (DAO) umožňuje vyhledávat a vkládat entity do databáze. Dotazy do databáze jsou specifikovány v anotaci nad metodami DAO (Obrázek 4.4).

Poslední částí, kterou bylo třeba specifikovat, bylo rozhraní samotné databáze. Rozhraní muselo splňovat několik podmínek: dědit z rozhraní *RoomDatabase*, specifikovat *Entity* představující tabulky v databázi a definovat metodu, která vrací objekt DAO pro práci s databází (Obrázek 4.5).

```
@Insert(onConflict = OnConflictStrategy.IGNORE)
fun insertEvents(events: List<Event>): List<Long>

@Query( value: "SELECT * FROM events")
fun getAllEvents(): LiveData<List<Event>>

@Query( value: "SELECT * FROM events WHERE event_id = :id")
fun getEvent(id: Int): LiveData<Event>
```

Obrázek 4.4: Ukázka implementace data access objektu

```
@Database(entities = [Event::class], version = 1)
@TypeConverters(Converters::class)
abstract class EventDatabase : RoomDatabase() {
    abstract fun eventsDao(): EventsDao
}
```

Obrázek 4.5: Ukázka rozhraní databáze

Type Convertors Pro zrychlení aplikace při menším množství dotazů do databáze nejsou vztahy mezi jednotlivými entitami modelovány tradičně pomocí cizích klíčů. Namísto toho jsou objekty, na které událost odkazuje, deserializovány do tabulky událostí. Při dotazu na událost dostaneme tedy vždy kompletní objekt se všemi atributy. Serializace a deserializace je realizována pomocí *type convertorů* (Obrázek 4.6).

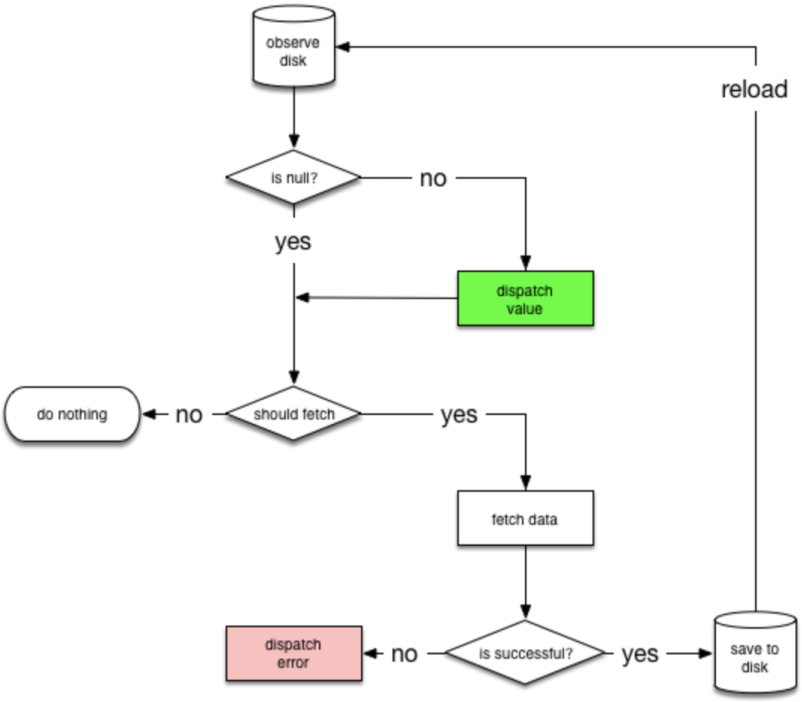
```
@TypeConverter
fun fromTimestamp(value: Long?): Date? {
    return value?.let { Date(it) }
}

@TypeConverter
fun dateToTimestamp(date: Date?): Long? {
    return date?.time?.toLong()
}

@TypeConverter
fun locationToJson(location: EventLocation?) : String? {
    return Gson().toJson(location)
}

@TypeConverter
fun jsonToLocation(locationJson: String?) : EventLocation? {
    return Gson().fromJson(locationJson, EventLocation::class.java)
}
```

Obrázek 4.6: Ukázka type convertorů pro serializaci a deserializaci objektů do databáze

4.1.3 Repository

Repozitář je komponenta, která rozhoduje o zdroji dat pro databázi. V závislosti na faktorech, především dostupnosti sítě, rozhoduje, zda budou uživateli zobrazena data ze serveru, nebo z lokální cache.

Network Bound Resource je třída usnaňující realizaci logiky pro rozhodování, odkud data zobrazit. Při dodržení *Single source* principu, třída pracuje následujícím způsobem: jako zdroj dat jsou vyslána data z lokální cache s informací, že probíhá načítání, ve stejnou dobu je poslán požadavek na server. Pokud požadavek úspěšně vrátí odpověď, nová data jsou zapsána do databáze a zobrazená data jsou aktualizována. Detailní popis její logiky je zřejmý z grafu (viz Obrázek 4.7).



Obrázek 4.7: Logika třídy Network Bound Resource

4.2 ViewModel

ViewModel je třída určená pro ukládání a spravování dat potřebných pro chod aplikace. V průběhu aplikace aktivity a fragmenty mohou byt zničeny a znovu vytvořeny frameworkem aplikace. Při skladování dat v některé UI komponentě by při takovýchto událostech byla data ztracena. ViewModel přečká všechny

fáze životního cyklu aplikace a při znovuvytvoření dostane aktivita či fragment stejný ViewModel se všemi daty, jako měla před zničením.

Mediator Live Data je komponenta, která umožňuje sledovat libovolné množství LiveData a měnit svou hodnotu vždy, když se změní hodnota některé z jejích závislostí.

V naší aplikaci je takto definována logika vyhledávání v seznamu. Data poslaná do UI pro zobrazení jsou závislá na dvou zdrojích - seznamu dat přijatých z repozitáře a klíčovém slovu z vyhledávání v seznamu (Obrázek 4.8).

```
filteredEvents = MediatorLiveData<Resource<List<Event>>>().apply { this: MediatorLiveData<Resource
    addSource(events) { it: Resource<List<Event>>!
        value = filteredEvents()
    }
    addSource(query) { it: String!
        value = filteredEvents()
    }
}
```

Obrázek 4.8: Využití třídy Mediator Live Data

4.3 View - Uživatelské rozhraní

Uživatelské rozhraní aplikace je rozděleno do dvou aktivit *LoginActivity*, která je první obrazovkou po otevření aplikace a stará se o autentizaci uživatele, a *MainActivity*, která obstarává veškerou hlavní funkčnost aplikace po přihlášení uživatele. Obsah hlavní aktivity je pak dělen na jednotlivé fragmenty, které se v ní střídají podle toho, jak uživatel naviguje aplikací.

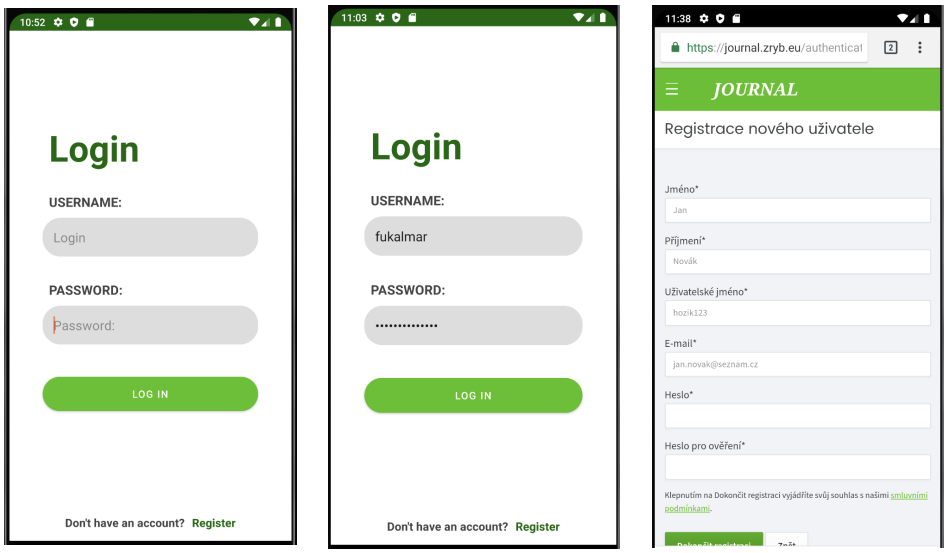
4.3.1 Login Activity

Login Aktivita poskytuje veškerou funkcionalitu spojenou s přihlašováním (Obrázek 4.9). Autentizace je nenutnou podmínkou pro přístup ke všem funkcím aplikace, přihlašovací obrazovka je tedy první, která se uživateli zobrazí.

Uživatel vyplní přihlašovací údaje a stikne tlačítko *Log in*. Aplikace pošle přihlašovací údaje na server k autentizaci, server vrátí odpověď, zda jsou přihlašovací údaje platné. Pokud ano, je třeba je posílat při každém požadavku na server.

Pokud uživatel nemá přihlašovací údaje do aplikace, může si zaregistrovat nový účet po kliknutí na odkaz *Register* na přihlašovací obrazovce. Pro pohodlnější používání aplikace uživatel zůstane přihlášen při každém dalším návratu do aplikace, dokud se sám neodhlásí.

Authentizace *Basic authentication* používaná v aplikaci je jednoduchý způsob autentizace využívající HTTP protokol. Uživatelské jméno a heslo je udržováno v aplikaci, v našem případě v *Shared Preferences*, a při každém požadavku na server jsou přihlašovací údaje posílány na server v podobě řetězce zakódovaného pomocí Base64.



(a) Přihlášení (b) Přihlášení (c) Registrace

Obrázek 4.9: Obrazovky spojené s přihlašováním

4.3.2 Main Activity

Hlavní aktivita slouží jako kontejner pro jednotlivé fragmenty, mezi kterými uživatel naviguje pomocí *Navigation component*. V následující kapitole budou představeny jednotlivé fragmenty a funkčnost, kterou aplikaci poskytují.

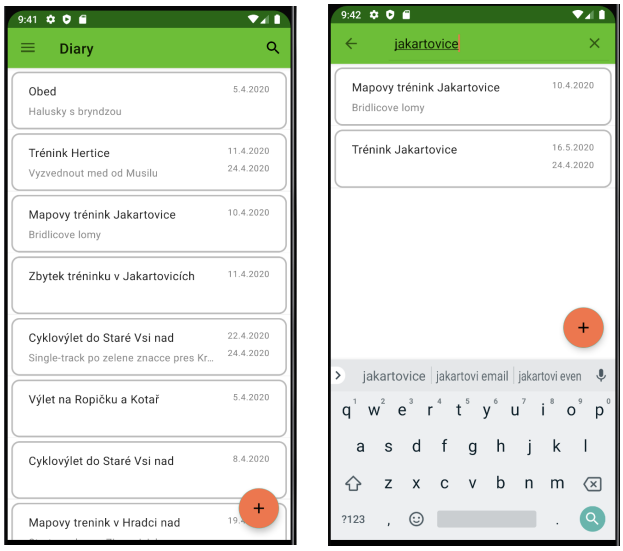
4.3.3 Seznam událostí

První obrazovka, která se uživateli zobrazí po přihlášení (nebo spuštění aplikace, pokud už je přihlášen), je seznam všech událostí (Obrázek 4.10). U každé události je uveden název, úryvek popisu a datum.

Po kliknutí na ikonu lupy v pravém horním rohu lze mezi událostmi vyhledávat (Obrázek 4.10b). Klíčové slovo zadané do vyhledávacího pole bude hledáno v názvech a popisech událostí.

4.3.3.1 Mapa

Na mapě se zobrazí markery pro události, které mají přiřazenou lokaci. Pokud je více markerů na stejném místě, shluknou se markery do klastrů (Obrázek



(a) Seznam (b) Vyhledávání v seznamu

Obrázek 4.10: Úvodní obrazovka se seznamem

4.11). Po kliknutí na samostatný marker se otevře *informační okno* s názvem a datem události, a po kliknutí na informační okno je uživatel přesměrován na detail události. V případě kliknutí na klastr se mapa přiblíží a klastr se rozloží na jednotlivé markery. Pokud již není možné mapu více přiblížit, je uživateli zobrazen seznam s událostmi v dané lokalitě.

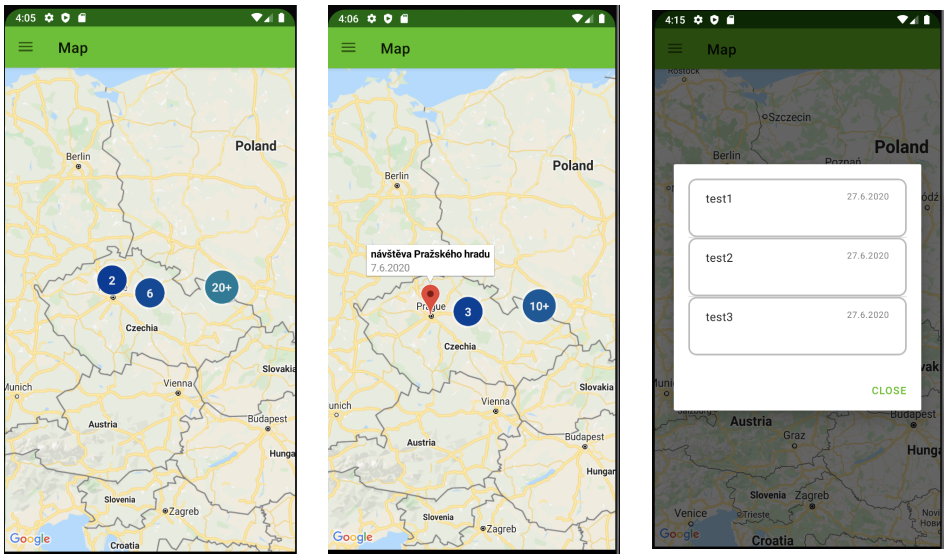
Klastrování markerů není implicitní chování Google Maps. Bylo třeba definovat vlastní cluster manager a přiřadit mu funkci pro chování při kliknutí na marker klastru (Obrázek 4.12).

4.3.4 Kalendář

Obrazovka kalendáře zobrazuje vždy jeden měsíc. Mezi měsíci lze přepínat pomocí gesta posunutí do strany. Toto chování je implementováno pomocí komponenty *View Pager*. Všechny prvky na obrazovce: linie rozdělující jednotlivé dny, čísla dnů v měsíci, názvy dnů v týdnu i události v kalendáři jsou vykreslovány přímo na *canvas* pohledu *(View)*.

4.3.5 Přidání, úprava a detail události

Detail události zobrazuje atributy události jako název, popis, tagy a datum, případně lokace a přílohy. K úpravě a vytvoření je použita stejná obrazovka, pouze v případě úpravy jsou pole formuláře vyplněna stávajícími hodnotami (Obrázek 4.13).



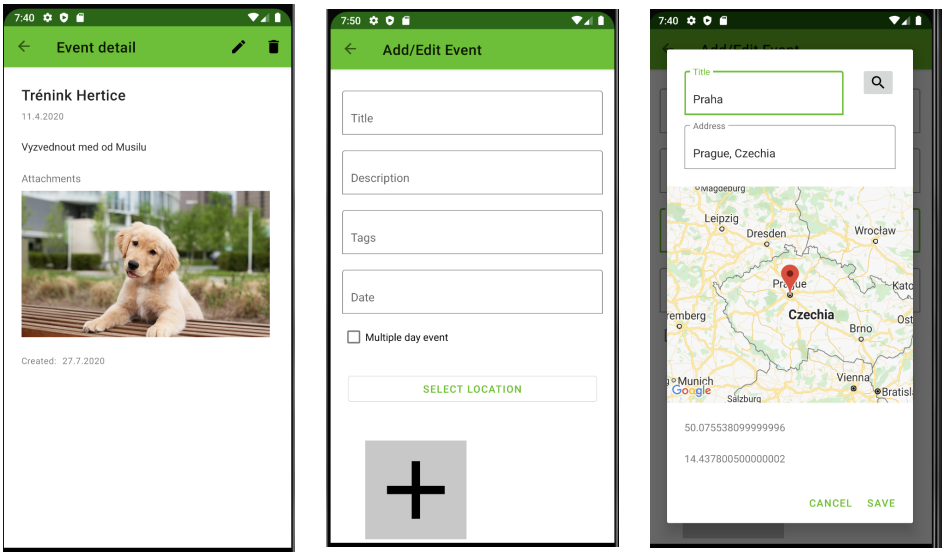
(a) Mapa (b) Informační okno (c) Seznam událostí klastru

Obrázek 4.11: Obrazovky mapy

```
override fun onClusterClick(cluster: Cluster<EventClusterItem>?): Boolean {
    cluster?.let { it: Cluster<EventClusterItem>
        val distinct : List<LatLng> = cluster.items.map { it.position }.distinct()
        if (distinct.size > 1) {
            val builder : LatLngBounds.Builder! = LatLngBounds.builder()
            distinct.forEach { builder.include(it) }
            val bounds : LatLngBounds! = builder.build()
            mMap?.moveCamera(
                CameraUpdateFactory.newLatLngBounds(bounds, 200)
            ) ^let
        } else {
            EventsListDialogFragment()
                .setList(cluster.items)
                .show(parentFragmentManager, tag: "dialog") ^let
        }
    }
    return true
}
```

Obrázek 4.12: Funkce definující chování při kliknutí na klastr markerů

Přidání lokace Po kliknutí na tlačítko *Select location* v editoru události se otevře dialog pro přidání lokace (Obrázek 4.13c). Lokaci v něm lze zadat dvěma způsoby. Uživatel může dlouze podržet marker zobrazený na mapě a ten se tak stane pohyblivým, nebo může zadat název lokace do pole *Název* a stisknout tlačítko *Hledat*, aplikace se pak pokusí najít adresu odpovídající lokace.



(a) Detail (b) Přidání/upravení (c) Dialog lokace

Obrázek 4.13: Obrazovky spojené s událostí

Kapitola 5

Testování

5.1 Scénáře uživatelského testování

V následující kapitole budou popsány úkony, které uživatel může chtít vykonat, a ideální průchod obrazovkami pro dosažení tohoto cíle.

5.1.1 UT1 Přihlášení a odhlášení

V tomto testu má uživatel za úkol se přihlásit a zase odhlásit.

Ideální průchod obrazovkami popisuje následující scénář a obrázek 5.1:

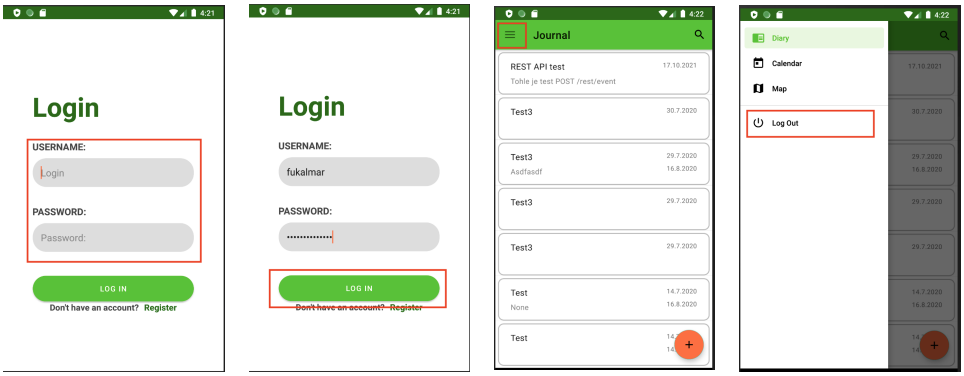
1. Tester vyplní přihlašovací údaje.
2. Tester stiskne tlačítko *Log In*.
3. Aplikace zobrazí úvodní stránku se seznamem událostí.
4. Tester stiskne tlačítko v levém horním rohu obrazovky pro zobrazení menu.
5. Aplikace otevře panel s bočním menu a uživatel stiskne tlačítko *Log out*.

5.1.2 UT2 Vyhledání lokace na mapě

V tomto scénáři má tester za úkol vyhledat na mapě událost v konkrétní lokaci a zobrazit její detail.

Ideální průchod obrazovkami popisuje následující scénář a obrázek 5.2:

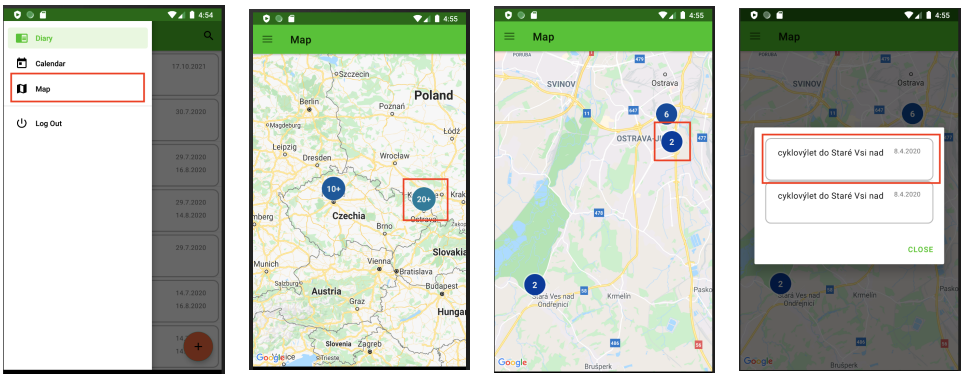
1. Tester vybere položku *Map* v levém menu.
2. Aplikace zobrazí obrazovku s mapou a markery označujícími lokace.
3. Tester klikne na marker klastru a ten se rozpadne na jednotlivé markery.



Obrázek 5.1: Ideální průchod obrazovkami pro UT1

4. Při kliknutí na marker klastru, kde všechny lokace jsou na jedno místě, zobrazí aplikace seznam událostí na této lokaci.

5. Tester klikne na vybranou událost ze seznamu a je přesměrován na detail události.



Obrázek 5.2: Ideální průchod obrazovkami pro UT2

5.1.3 UT3 Vyhledání události v seznamu

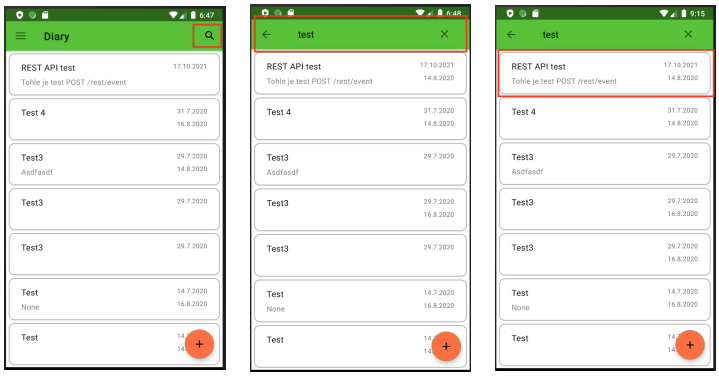
V tomto scénáři má uživatel za úkol vyhledat v seznamu událost s konkrétním názvem a zobrazit její detail.

Ideální průchod obrazovkami popisuje následující scénář a obrázek 5.3:

1. Tester stiskne tlačítko pro vyhledávání v pravém horním rohu obrazovky.

2. Vyplní požadované slovo do vyhledávacího políčka.

3. Ze seznamu vyfiltrovaných aplikací vybere požadovanou aplikaci a klikne na ni.



Obrázek 5.3: Ideální průchod obrazovkami pro UT3

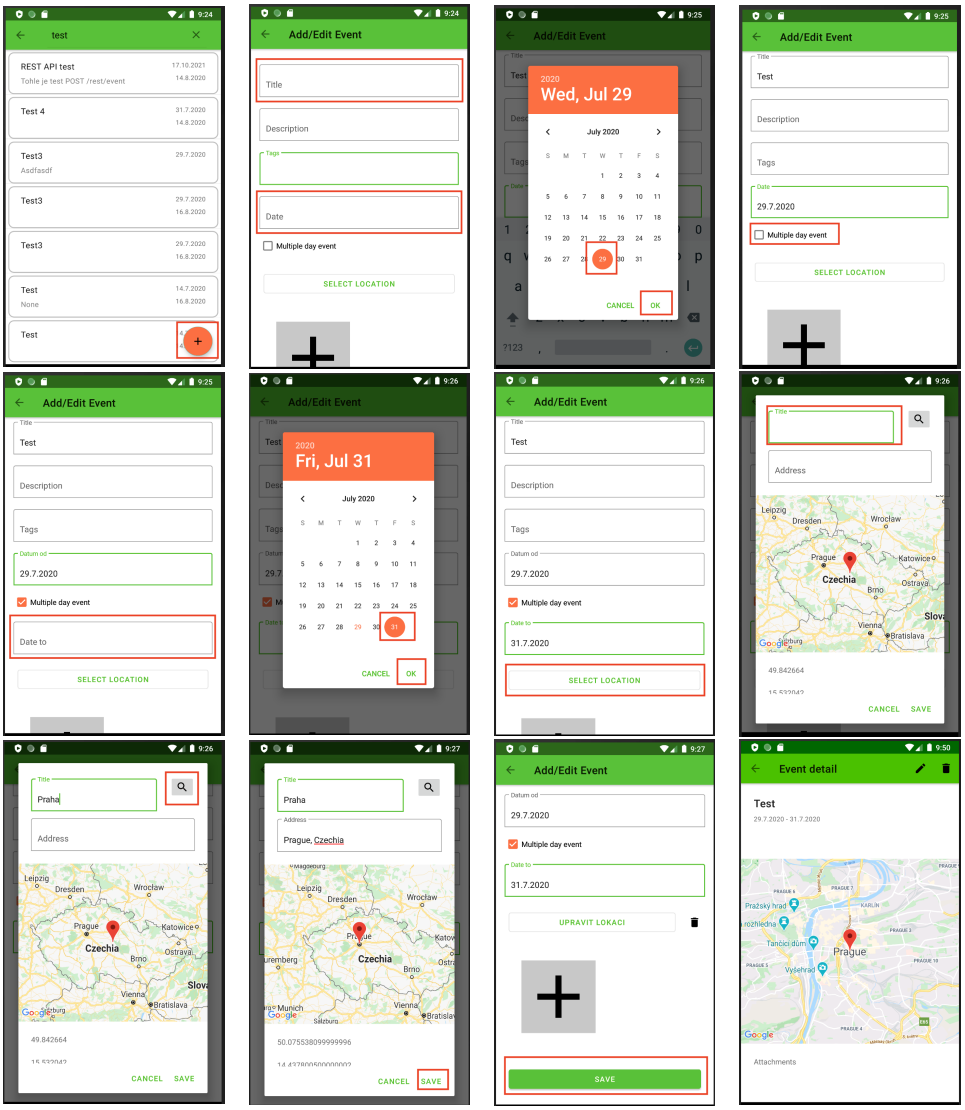
4. Aplikace zobrazí detail vybrané události.

5.1.4 UT4 přidání nové události

V tomto scénáři má uživatel za úkol vytvořit novou událost s názvem *Test*, datem 30.7. - 31.7.2020 a s lokací v Praze.

Ideální průchod obrazovkami popisuje následující scénář a obrázek 5.4:

1. Tester klikne na ikonu FAB v pravém dolním rohu obrazovky.
2. Vyplní název do pole *Title* a klikne na pole *Date*.
3. Aplikace zobrazí kalendář pro výběr data. Uživatel vybere požadované datum a stiskne tlačítko *OK*.
4. Na obrazovce přidávání události zaškrtne políčko *Multiple day event*.
5. Aplikace zobrazí pole pro přidání *Date to*.
6. Uživatel vybere koncové datum události a stiskne tlačítko *OK*.
7. Na obrazovce přidávání události stiskne uživatel tlačítko *Select location*.
8. Aplikace zobrazí dialogové okno pro přidání lokace.
9. Uživatel do pole pro název zadá slovo *Praha*
10. Uživatel stiskne tlačítko pro vyhledání adresy lokace.
11. Aplikace vyhledá požadovanou lokaci, zadá její adresu do pole *Address* a na mapě přesune marker na souřadnice nalezené lokace.
12. Uživatel stiskne tlačítko *Save* a editor lokace se zavře.



Obrázek 5.4: Ideální průchod obrazovkami pro UT4

13. Na obrazovce přidávání události uživatel stiskne tlačítko *Save*, lokace je uložena a uživatel je přesměrován na detail události.

5.2 Výsledky uživatelského testování

5.2.1 Tester 1

Tester č.1 je běžným uživatelem. Nemá žádné vzdělání v jakémkoliv technologickém oboru a jeho schopnost řešit problémy spojené s technologií je spíše

podprůměrná.

UT1 Tester měl problém se zadáváním přihlašovacích údajů, protože nebylo možné skrolovat přihlašovací obrazovkou a pole pro zadání údajů byla přístupná pouze po stisknutí klávesy *Next*, což mu přišlo příliš neintuitivní.

Na obrazovku byla přidána vrstva, která umožnila skrolování, a uživatel neměl již potíže.

UT2 Tester měl při prvním pokusu potíž přidat lokaci, protože při pokusu o vyplnění pole s názvem lokace se nevysunula klávesnice, což bylo chybou programátora.

UT3 Tester neměl problém se splněním úkolu.

UT4 Tester neměl problém se splněním úkolu.

5.2.2 Tester 2

Tester č.2 má vzdělání v oboru informačních technologií. Jeho zkušenost s Android aplikacemi je převážně uživatelská, v oboru programování Android aplikací je spíše začátečník.

UT1 Tester neměl problém se splněním úkolu.

UT2 Tester neměl problém se splněním úkolu.

UT3 Tester měl trochu problém s nastavením vícedenní události.

UT4 Tester neměl problém se splněním úkolu.

5.2.3 Tester 3

Tester č.3 má dlouholeté zkušenosti s programováním na profesionální úrovni, ale jeho znalost Android aplikací z uživatelského hlediska je spíše slabší. Zkušenosti s programováním Android aplikací pak nulové.

UT1 Tester neměl problém se splněním úkolu.

UT2 Tester měl problém přijít na to, že se na detail události dostane kliknutím na informační okno markeru.

UT3 Tester měl trochu problém s nastavením vícedenní události.

UT4 Tester měl problém s přidáním vícedenní události. A také s vyhledáním lokace v *Location editor*

5.3 Výsledky uživatelského testování

V průběhu testování se objevily některé drobné nedostatky aplikace, které mohly být průběžně opraveny.

Mezi závažnější nedostatky aplikace patří editor pro přidávání lokací. Většina uživatelů jej považovala za matoucí a v dalším vývoji aplikace by bylo vhodné přepracovat jeho design.

Závěr

Hlavním cílem této bakalářské práce bylo analyzovat funkcionalitu již existující webové aplikace Journal, vyvinuté na Katedře softwarového inženýrství, Fakultě informatiky ČVUT, v předchozích letech postupně v rámci několika vývojových týmů; navázat na REST API této webové aplikace a vytvořit mobilního klienta k této webové aplikaci, přizpůsobeného možnostem chytrého telefonu a implementujícího většinu dostupných funkcí původní webové aplikace Journal.

V kapitole věnované analytické části této práce jsem kromě analýzy výchozí webové aplikace Journal, na niž můj mobilní klient navazuje, provedla rovněž analýzu pěti konkurenčních mobilních aplikací pro vedení osobního deníku s výsledným porovnáním jejich dostupných funkcionalit a uživatelských preferencí. Výsledkem mého průzkumu bylo zjištění, že ač můj mobilní klient neobsahuje některé uživateli žádané fukcionality, jako je například moodtracker (sledovač nálad) nebo převod hlasových záznamů na text či obráceně, tak naopak nabízí některé jiné funkce, které dosud implementuje vždy jen jedna ze zkoumaných konkurenčních aplikací (a rovněž výchozí webová aplikace Journal) a které se mi přitom zdají zajímavé a užitečné.

V další části mé práce jsem provedla specifikaci požadavků a vytvořila jsem uživatelské scénáře, na základě kterých jsem následně vytvořila implementaci uživatelského rozhraní a také scénáře uživatelského testování vyvinuté aplikace. V kapitole popisující návrh aplikace vysvětluji svůj výběr architektury aplikace, použitých technologií a knihoven třetích stran pro každou ze tří vrstev aplikace.

Mým úkolem rovněž bylo vytvořit developerskou příručku k tomuto mobilnímu klientovi, což splňuje kapitola nazvaná Implementace, ve které popisuji krok za krokem implementaci uživatelských scénářů i použité technologie a ilustruji vybranými částmi zdrojového kódu.

Kapitola věnovaná testování pak slouží též jako uživatelská příručka, kde každý uživatelský scénář je doplněn adekvátními obrazovkami z mobilního klienta ukazujícími průchod daným scénářem.

Výsledkem této bakalářské práce je tedy funkční verze mobilního klienta webové aplikace Journal, zdokumentovaná po stránce analytické, návrhové, vývojářské i uživatelské, čímž bylo naplněno zadání této bakalářské práce. Jistěže tento mobilní klient nepokrývá veškeré funkcionality konkurenčních mobilních aplikací pro vedení deníkových záznamů, ale to ani nebylo náplní této práce. Je v tom však možno spatřovat prostor na možná vylepšení aplikace v budoucnu. Naopak nabízí funkcionality, které obsahuje jen málokterá z konkurenčních deníkových aplikací a které mi osobně připadají jako významné a zajímavé.

Bibliografie

1. BIGHEAD TECHIES. *Daybook - Diary, Journal, Note, Mood Tracker*. 2020. Dostupné také z: https://play.google.com/store/apps/details?id=com.bigheadtechies.diary&hl=cs cit. 2020-07-24.
2. HABITICS. *Daylio - Deník, Nálady, Poznámky, Deníček*. 2020. Dostupné také z: https://play.google.com/store/apps/details?id=net.daylio&hl=cs cit. 2020-07-27.
3. ARYA APPLICATIONS. *Můj Deník*. 2020. Dostupné také z: https://play.google.com/store/apps/details?id=diary.plus.plus&hl=cs cit. 2020-07-26.
4. PIXEL CRATER. *Diaro - deník, deník, poznámky, tracker nálad*. 2020. Dostupné také z: https://play.google.com/store/apps/details?id=com.pixelcrater.Diaro&hl=cs cit. 2020-07-26.
5. LUCIDIFY LABS. *Diary Book - Journal With Lock, Photos, Themes*. 2020. Dostupné také z: https://play.google.com/store/apps/details?id=in.lucidify.dairybook cit. 2020-07-26.
6. GOOGLE. *Android's Kotlin-first approach*. 2020. Dostupné také z: https://developer.android.com/kotlin/first cit. 2020-07-24.
7. COPYRIGHT ALIANCE. *Oracle v. Google*. 2020. Dostupné také z: https://copyrightalliance.org/copyright-law/copyright-cases/oracle-america-v-google/ cit. 2020-07-24.
8. *Kotlin*. 2020. Dostupné také z: https://developer.android.com/kotlin cit. 2020-07-24.
9. ANDROID DEVELOPERS. *Architecture Components - Introduction (Google I/O '17)*. 2020. Dostupné také z: https://www.youtube.com/watch?v=FrteWKKVyzI cit. 2020-07-24.
10. ANDROID DEVELOPERS. *Recomended guide to architecture*. 2020. Dostupné také z: https://developer.android.com/jetpack/guide cit. 2020-07-24.

11. SQUARE, INC. *Retrofit*. 2020. Dostupné také z: `https://square.github.io/retrofit/` cit. 2020-07-24.
12. ANDROID DEVELOPERS. *Room*. 2020. Dostupné také z: `https://developer.android.com/topic/libraries/architecture/room` cit. 2020-07-24.
13. ANDROID DEVELOPERS. *Google Maps Platform*. 2020. Dostupné také z: `https://developers.google.com/maps/documentation` cit. 2020-07-24.
14. BUMPTECH. *Glide*. 2020. Dostupné také z: `https://github.com/bumptech/glide` cit. 2020-07-24.
15. *Koin*. 2020. Dostupné také z: `https://insert-koin.io/` cit. 2020-07-24.

Příloha **A**

Seznam použitých zkratek

DAO Data Access Object

DI Dependency Injection

FAB Floating Action Button

GUI Graphical user interface

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

MVC Model-View-Controler

MVP Model-View-Presenter

MVVM Model-View-ViewModel

POJO Plain Old Java Object

REST Representational State Transfer

SDK Software Development Kit

XML Extensible markup language

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `apk` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF