Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Aplikace na měření nutričních hodnot pro Android

2019

Michal Václavek

Vedoucí práce: Mgr. Petr Krajča, Ph.D.

Studijní obor: Informatika, prezenční forma

Bibliografické údaje

Autor:	Michal Václavek
Název práce:	Aplikace na měření nutričních hodnot pro Android
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2019
Studijní obor:	Informatika, prezenční forma
Vedoucí práce:	Mgr. Petr Krajča, Ph.D.
Počet stran:	34
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliographic info

Author:	Michal Václavek
Title:	Nutrition Tracking Application for Android
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2019
Study field:	Computer Science, full-time form
Supervisor:	Mgr. Petr Krajča, Ph.D.
Page count:	34
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

V rámci bakalářské práce byla vyvinuta aplikace pro platformu Android, která pomáhá uživateli s měřením nutričních hodnot ze zkonzumovaných produktů. Aplikace nabízí uživateli upravitelnou databázi produktů a možnost tvoření vlastních jídel.

Synopsis

A product of this bachelor thesis is an Android application that helps user to track nutrition values of eaten products. Application provides an editable database of products and lets user create custom meals.

Klíčová slova: Android aplikace; aplikace na měření nutričních hodnot; programovací jazyk Kotlin

Keywords: Android application; nutrition tracking application; Kotlin programming language

Děkuji vedoucímu práce, Mgr. Petru Krajčovi, Ph.D., za cenné připomínky a vedení katedry za schválení vlastního tématu.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V dnešní době je čím dál tím více modernější vést zdravý životní styl, v rámci kterého jsou nejdůležitější dvě věci: pohyb a strava. Naštěstí nyní existuje spousta aplikací pro mobilní zařízení, které se snaží pomoci s dodržováním správné životosprávy.

V rámci této bakalářské práce byla podobná aplikace vyvinuta. Aplikace se zaměřuje na oblast stravy. Pomáhá uživateli se zaznamenáváním zkonzumovaných jídel a vypočítává z nich přijaté nutriční hodnoty. Uživatel má tedy přehled o příjmu makronutrientů z potravy za určité období.

Na začátku práce popíši požadavky aplikace. Proč takovou aplikaci vytvářet, když už existují jiná řešení. A také proč je dobré aplikaci používat. To vše je obsaženo v kapitole 2.

Dále popíši aplikace pro uživatele. Tedy z jakých částí se skládá, jakou funkcionalitu poskytuje a jak s ní má uživatel zacházet.

Ke konci se zaměřím na implementační detaily. Jak aplikace funguje pod povrchem. Zde zmíním jak je kód strukturován a následně jej popíši.

V průběhu práce jsou části, které nabízejí možné vylepšení aplikace.

2 Požadavky aplikace

Aplikace by měla uživateli umožnit zaznamenání si zkonzumovaných surovin/pokrmů. Ze zaznamenaných surovin poté vypočítat kolik kalorií a makronutrientů[1] uživatel zkonzumoval za určité období a tento údaj mu zobrazit. Dále by mu měla poskytnou databázi produktů. Do této databáze by uživatel měl mít možnost přidat produkt, upravit nebo také odebrat.

Dále se tato kapitola zaměření na to, proč se zabývat podobnou aplikací. A poté na existující řešení, jejich nedostatky a jak lze tyto nedostatky napravit.

2.1 Proč se tím zabývat

Na otázku, kterou pokládá tato podkapitola, můžeme pohledět dvěma způsoby. Jedním, z hlediska uživatele, proč by měl používat tuto aplikaci a druhým z hlediska tvůrce, proč by měl takovou aplikaci vytvářet.

Měli bychom dbát o své tělo, dodržovat takzvanou správnou životosprávu a právě správné stravování je důležitou částí správné životosprávy. Bohužel né každý se v této oblasti dobře orientuje. Naštěstí moderní technika může pomoci různými způsoby, například vyhledáním jídelníčku na internetu. Často ale nastává situace, kdy musíme přesněji vědět, kolik toho sníme.

I v tom nám může pomoci moderní technika, nejpohodlnější je zařízení, které má většina u sebe, mobilní telefon[2]. Nejde totiž jen o pouhé zapsání si požitých nutriřních hodnot, co člověk přes den zkonzumoval, ale i to, aby si byl vědom, co přesně se v daném jídle nachází za živiny. To může být náročné na zapamatování, opakovatelné hledání těchto informací na internetu, následného přepočtu a finálního součtu. Přesně v tomhle může pomoci moderní technika, mobilní telefon se specializovanou aplikací. Člověk si jednou vyplní informace o živinách, pokud jsou tyto informace dostupné v databázi, tak i tento krok není potřebný. A pak už zadá množství a o všechno ostatní se postará aplikace.

Z hlediska tvůrce už je odpověď na otázku, proč vytvářet takovou aplikaci, velmi jednoduchá, aby vyřešil výše popsaný problém. Dnes už existuje spousta aplikací, které se snaží přesně tento problém vyřešit. Nabízejí se tedy otázky. Proč vytvářet další aplikaci? Je s existujícími aplikacemi něco špatně? A další, které jsou hlavním tématem následující kapitoly 2.2.

2.2 Existující řešení

Ke konci předchozí kapitoly 2.1 jsem zmínil, že už existují podobné aplikace. Poté jsem vznesl pár otázek. Odpověď na tu první otázku, proč vytvářet další aplikace, je, že každý uživatel je jiný, co vyhovuje jednomu, nemusí vyhovovat druhému. Tudíž mít na výběr je z pohledu uživatele vítané.

[1] Základní živiny – sacharidy, bílkoviny, tuky

[2] V tomto případě myšleno chytrý telefon

Já sám jsem uživatel takové aplikace, dokonce jsem těch aplikací vyzkoušel několik. Bohužel žádná z nich nebyla schopna plně uspokojit mé potřeby. Proto jsem se rozhodl vytvořit si vlastní aplikaci podle mých představ. Dále zmíním, co mi na existujících aplikacích vadilo a poté i na možné nápravy.

2.2.1 Nedostatky

V této podkapitole zmiňuji problémy, které jsem měl s aplikacemi, které jsem používal. Pokud existuje aplikace, která daný problém nemá, tak má nějaký jiný z uvedených, nebo jsem na ni nenarazil.

Jedním z hlavních problémů je přílišná složitost aplikace. Snaží se nabídnout příliš mnoho funkcí, které se do aplikace, z mého pohledu, nehodí. Funkce receptů, kdy může uživatel vyhledávat recepty, psát k nim komentáře, přidávat do oblíbených apod. Ačkoliv recepty jsou spjaty s jídlem jako takovým, tak v rámci aplikace, ve které si chci přidat zkonzumované jídlo, působí, jakoby tam byly navíc, neboť nijak s hlavní funkcionalitou nesouvisí. Stejné je to pak s funkcí cvičení. Kde si člověk může vybrat, co za typ cvičení chce a aplikace navrhne cviky, které by uživatel měl provádět a jak dlouho. Opět to nemá nic společného s konzumací jídla. Důsledek těchto funkcí je i takový, že aplikace se neustále připomíná pomocí upozornění, že „je dostupný nový recept“, či „je čas si jít zacvičit“. Pokud tuto funkcionalitu nevyužívám, což ani v mém případě nechci, tak mě to jen otravuje a spíš to vede k tomu, že aplikaci odstraním.

Dalším problémem je způsob výdělku peněz. Některé aplikace nabízejí neúplné funkce a nutí uživatele, aby si připlatili za premium verzi, bez které nemají například přístup do databáze produktů. Je nutno uznat, že některé placené služby jsou kvalitní. A některé aplikce jsou bez nutnosti připlacení rovněž použitelné. Bohužel tím aplikace spadá do kategorie příliš složitých aplikací. Nebo jdou cestou reklam, které zhoršují kvalitu uživatelského rozhraní. Například, když projíždíte seznamem produktů a mezi nimi jsou vloženy reklamy. To velice znepříjemňuje manipulaci s produkty.

Dále mi chyběla možnost tvorby vlastních jídel. Aplikace vás nechá přidat si do databáze vlastní položku, ale já bych si rád přidal celé jídlo. Myšleno produkt, který obsahuje více existujících produktů. Například chleba se šunkou, který bych jednoduše složil z již dostupných produktů. V tomto případě chléb a šunka. Bohužel tuto funkcionalitu mi žádná aplikace nenabídla.

Náprava nedostatků

V předchozí podkapitole jsem zmínil několik problémů, které mají dostupné aplikace. V této části navrhnu způsob nápravy těchto problémů.

Složitost se dá vyřešit dvěma způsoby. Nemít zbytečnou funkcionalitu navíc, nebo ji dobře izolovat od ostatních, aby se funkcionality navzájem nevyrušovaly a neztěžovaly tak používání aplikace. V případě mé aplikace, jsem se rozhodl o první z těchto možností. Pokud bych v budoucnu chtěl přidat například jídelníčky, znatelně bych je oddělil od zbylé části aplikace. Ideu o tom, jak bych tak

učinil najdete v kapitole 3.4.

S finanční situací už je to poněkud složitější, pokud musíte zaplatit vývoj a údržbu serverů. Rozhodně není moudré zamykat důležitou funkcionalitu za poplatek, či „míchat“ reklamy mezi důležité prvky uživatelského rozhraní. Jelikož má aplikace nemusí, momentálně, generovat výdělek, tak je zcela zdarma, bez poplatků a reklam.

Ohledně možnosti přidání vlastního jídla je jen jedno možné řešení. Přidat tuto funkcionalitu, což jsem také tak učinil. Více v kapitole 3.3.

Obrázek 1: Navigační panel aplikace

3 Popis aplikace

Tato kapitola detailně popíše funkcionalitu aplikace a zároveň přiblíží, jak by ji měl uživatel používat.

Aplikace se dělí na tři hlavní části podle jejich funkcionality. Tyto části jsou dostupné ze spodní části obrazovky, která obsahuje navigační panel – obrázek 1.

První část, zleva, slouží k zobrazení zkonzumovaných produktů a nutričních hodnot, více v kapitole 3.1. Ze druhé části může uživatel přidat produkt, který zkonzumoval. Také tam spravuje produkty v databázi, více v kapitole 3.2. Poslední část slouží k tvorbě vlastních jídel a jejich správě, více v kapitole 3.3.

3.1 Zkonzumované produkty

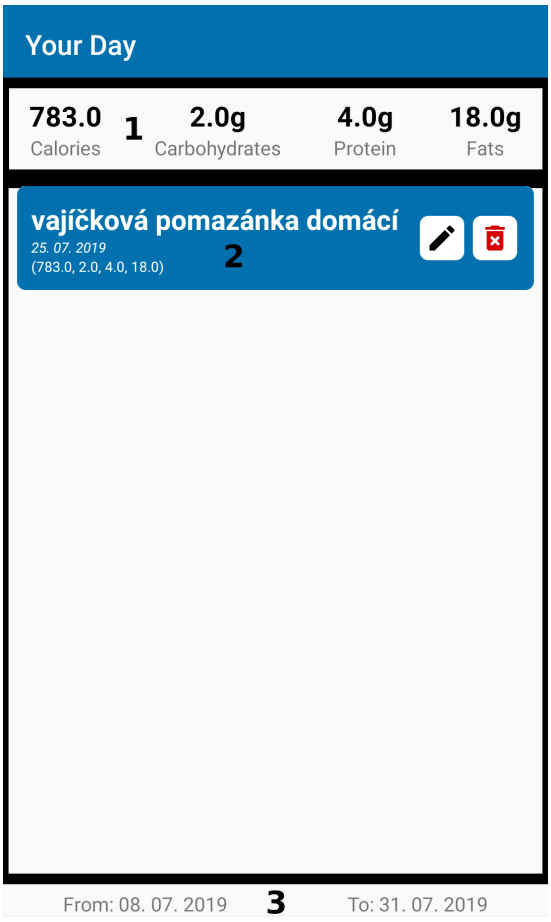
Tato část aplikace je vyobrazená na obrázku 2, který je rozdělen na tři sekce.

Sekce 1 Zde se zobrazují vypočítané hodnoty makronutrientů ze zkonzumovaných produktů – sekce 2. Tyto informace se automaticky aktualizují, pokud uživatel provede jedno z následujících:

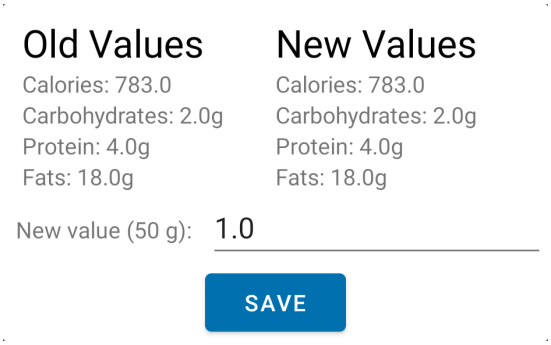
- přidá produkt jako zkonzumovaný
- upraví množství již zkonzumovaného produktu
- vymaže zkonzumovaný produkt
- změní údaj od-do – sekce 3

Sekce 2 Zobrazuje seznam zkonzumovaných produktů. Jednotlivá položka pak zobrazuje název produktu, datum, kdy byl produkt, a hodnoty makronutrientů. Také má ovládací prvky ve formě dvou tlačítek. Červený koš slouží k odstranění produktu ze seznamu. Černá tužka k upravení produktu. Po kliknutí na tlačítko úpravy se zobrazí dialog pro úpravu, který je zachycen na obrázku 3. Dialog nabízí možnost změnit množství produktu a tuto změnu uložit, čím se dialog zavře a hodnoty přepočítají. Dále dialog ukazuje dva sloupce hodnot. V levém jsou aktuální zkonzumované hodnoty. Pravý se mění při změně množství a dává tak uživateli představu o tom, jak se od sebe původní a potenciální hodnoty liší.

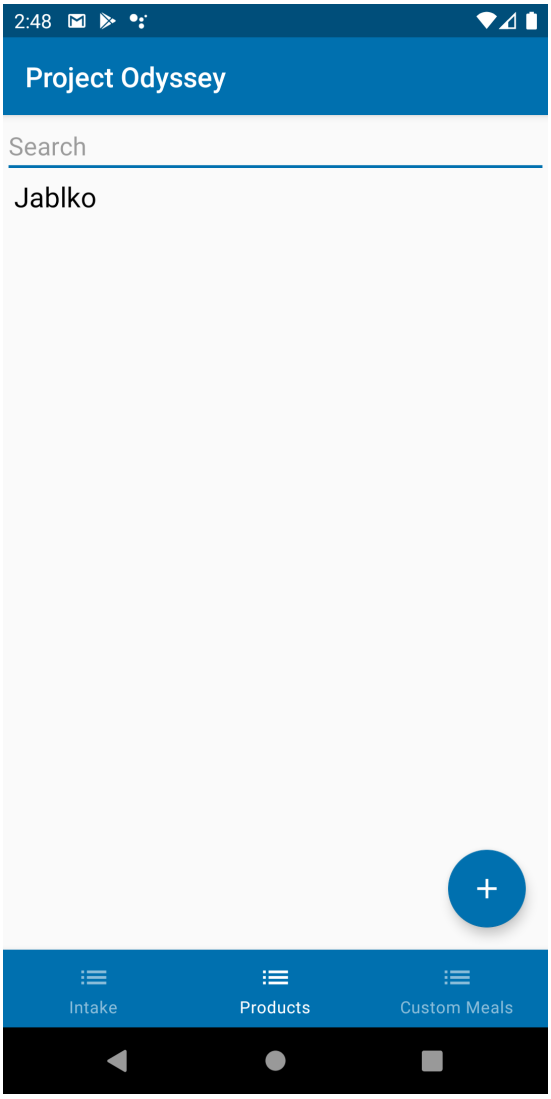
Sekce 3 Pod seznamem se nachází údaj o tom, v jakém časovém rozpětí má aplikace prezentovat zkonzumované produkty. Po kliknutí na datum se zobrazí kalendář, ve kterém si uživatel může zvolit jiné datum. Po změně dojde ke správnému aktualizování seznamu a přepočítání makronutrientů.



Obrázek 2: Seznam zkonzumovaných produktů



Obrázek 3: Editační dialog pro zkonzumovaný produkt



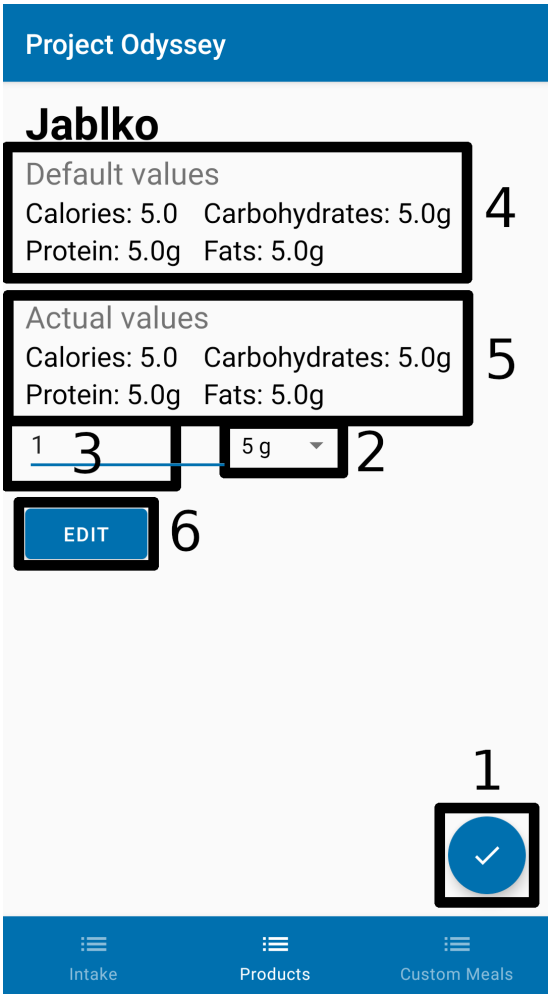
Obrázek 4: Seznam lokálně uložených produktů

3.2 Databáze produktů

Databáze produktů jsou ve skutečnosti dvě. Jedna online dostupná pro všechny a druhá lokální v zařízení uživatele. Rozdíl spočívá v tom, že online databáze je pouze pro čtení, kdežto lokální si může uživatel upravovat dle libosti. To napomáhá tomu, že informace z online databáze budou vždy ctít míru pravdivosti a uživatel jí tedy může důvěřovat.

Aplikace v této části zobrazuje seznam lokálně uložených produktů. Nabízí možnost vyhledání, přidání a úpravy produktů.

Seznam produktů Seznam produktů je vyobrazen na obrázku 4. Zde uživatel může vidět seznam lokálně uložených produktů, který zabírá většinu prostoru. Tento seznam slouží k výběru produktu, který chce uživatel zkonzumovat. To lze



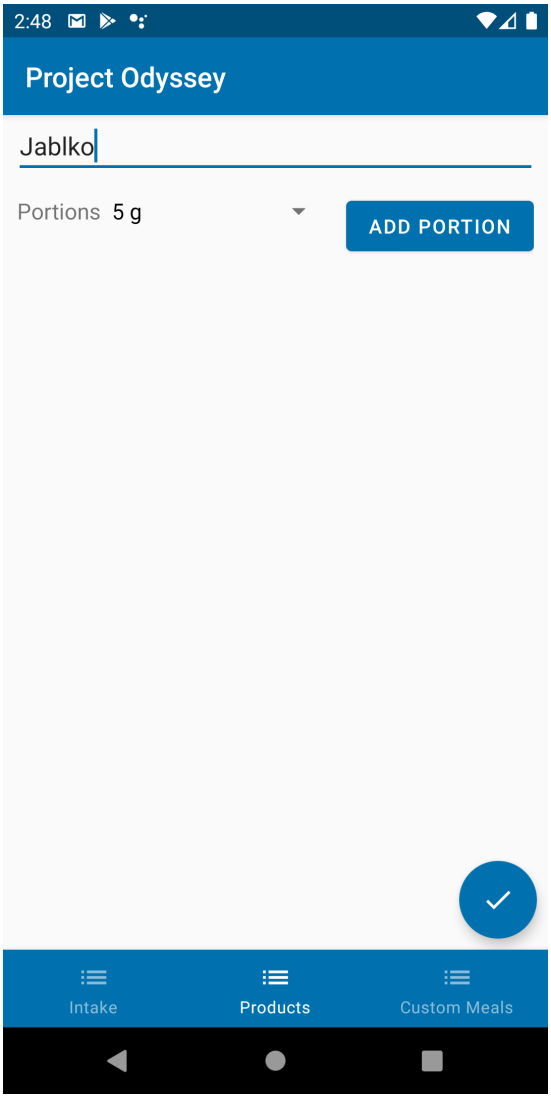
Obrázek 5: Detail produktu

posléze uskutečnit z detailu produktu. Není to ale jediný způsob, jak lze přidat zkonzumovaný produkt. Další možností jsou vlastní jídla, více v kapitole 3.3.

Zbývající prostor zabírá textové pole, umístěné nad seznamem, které slouží k vyhledávání. Při zadávání hodnot do textového pole se automaticky vyhledává v online databázi a v lokální databázi. Seznam produktů je průběžně aktualizován výsledky vyhledávání.

Detail produktu Při kliknutí na produkt v seznamu je uživatel přesměrován na jeho detail – obrázek 5. Zde může uživatel přidat zvolený produkt do zkonzumovaných produktů. Slouží k tomu tlačítko vpravo dole označené číslem 1.

Uživatel má možnost vybrat si z dostupných porcí – sekce 2. Sekce 3 slouží k určení množství zvolené porce. Sekce 4 a 5 ukazují původní hodnoty, které jsou obsaženy ve zvolené porci, a hodnoty, které budou skutečně zkonzumovány dle zadaného množství. Sekce 5 je tedy aktualizována vždy, když se změní hodnota v sekci 3.

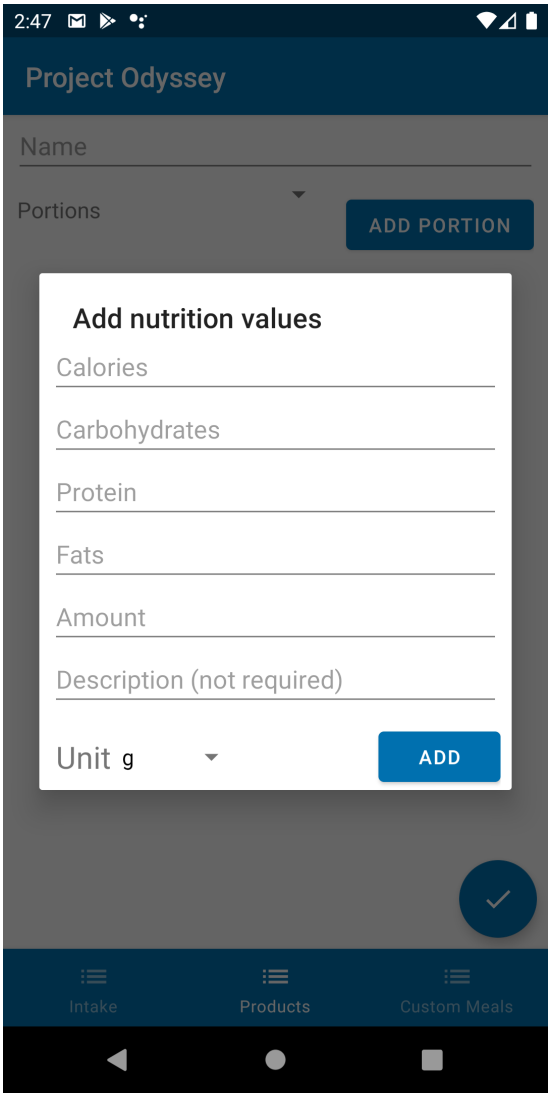


Obrázek 6: Přidání produktu

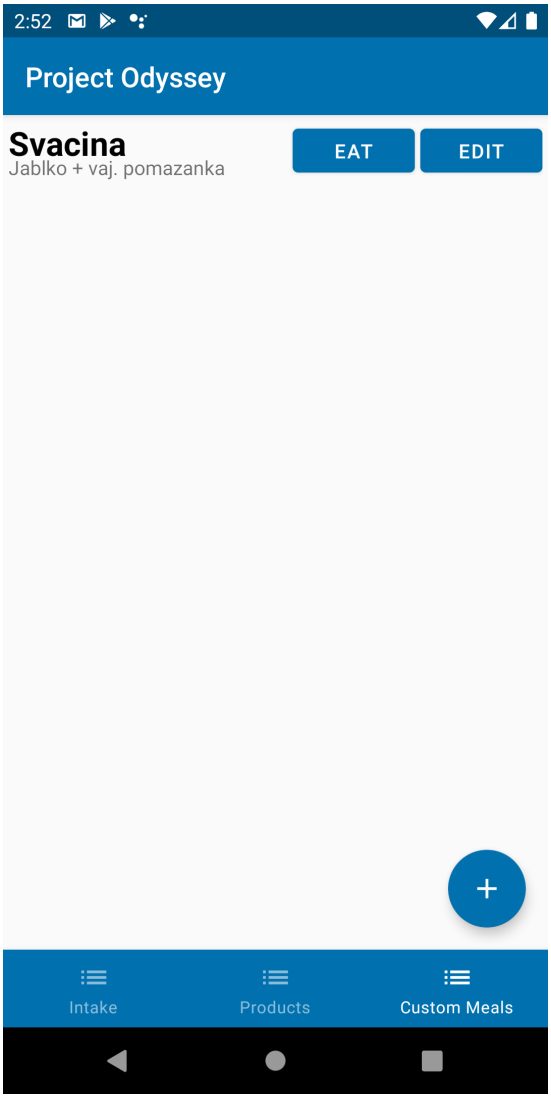
Tlačítko označené číslem 6 slouží k upravení produktu. Toto tlačítko je dostupné pouze pokud je produkt uložený lokálně. Po kliknutí se uživatel dostane na obrazovku upravení produktu, která je velice podobná obrazovce pro přidání produktu – obrázek 6. Avšak nabízí tlačítka pro úpravu a odstranění produktu.

Přidání produktu Obrazovka pro přidání produktu je vyobrazena na obrázku 6. Samotné přidání produktu je provedeno kliknutím na tlačítko vpravo dole. Předtím musí uživatel vyplnit název produktu do horního textového pole – na obrázku „Jablko“. A přidat alespoň jednu porci kliknutím na tlačítko „ADD PORTION“ – může se lišit lokalizací.

Po kliknutí na tlačítko přidání porce se zobrazí dialog pro přidání porce – obrázek 7. Zde uživatel zadá množství porce v poli „Amount“ a kolik daných makronutrientů se v ní nachází. Lze zvolit i jednotku, gramy, nebo mililitry. To



Obrázek 7: Dialog přidání makronutrientů k produktu



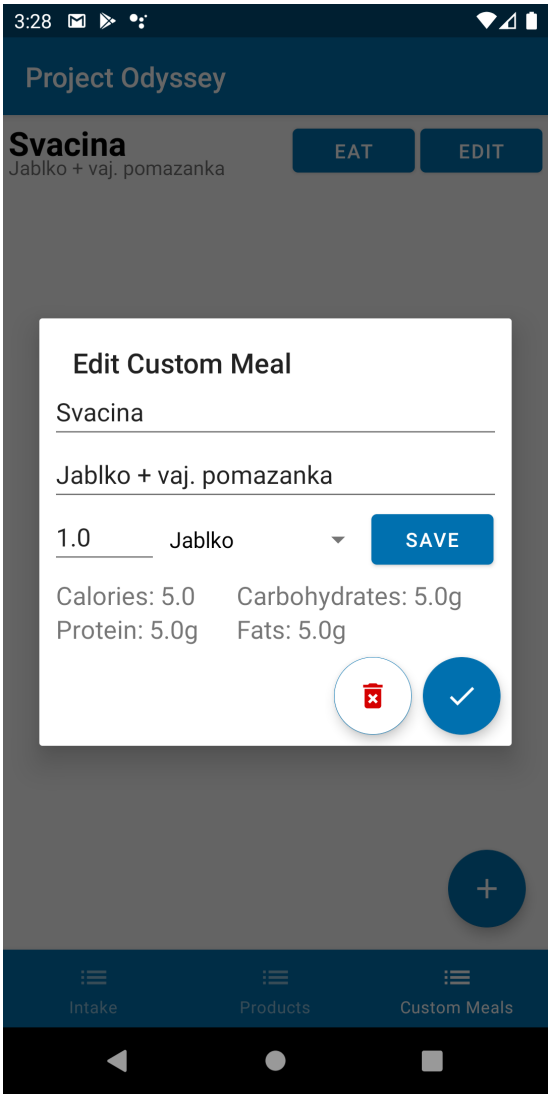
Obrázek 8: Seznam vlastních jídel

je nápocné, pokud uživatel zná hodnoty pro jednu z jednotek a nechce je přepočítávat na druhou. Uživatel má i možnost tuto porci pojmenovat zadáním hodnoty do pole „Description“. Pokud ji uživatel nepojmenuje, tak je automaticky vytvořena podle množství a jednotky.

3.3 Vlastní jídla

V kapitole 2.2 jsem zmínil, že mi v existujících řešeních chyběla možnost vlastního jídla, které by se dalo vytvořit přidáním produktů z databáze, nikoliv o přidání dalšího produktu. Výhoda tvorby vlastního jídla spočívá v tom, že uživatel nemusí nic přepočítávat a aplikace to udělá sama. Také jsem uvedl, že jsem tuto funkcionalitu přidal do mé aplikace a právě tato kapitola se na ni zaměří.

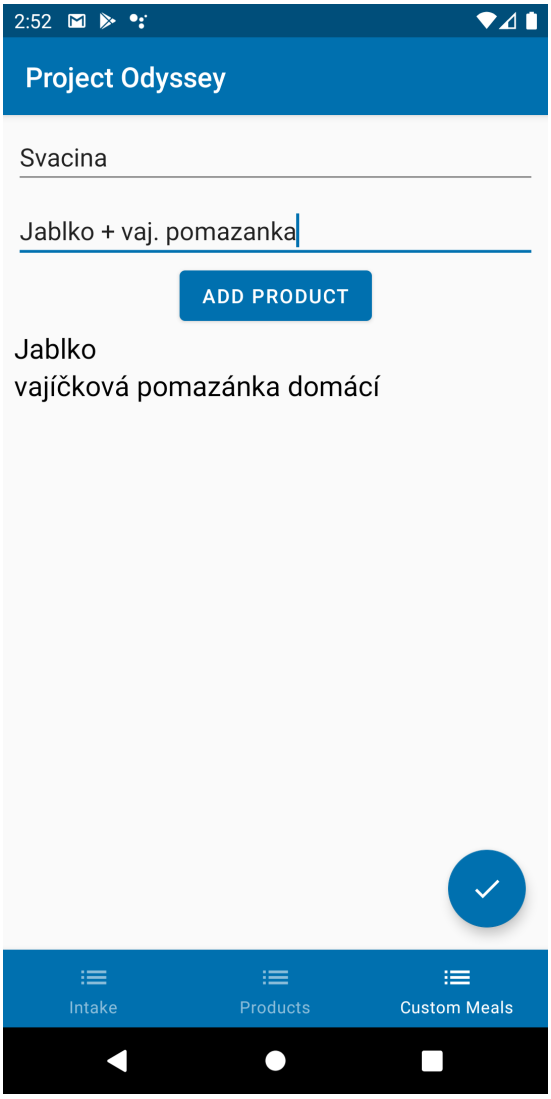
Na hlavní obrazovce této funkcionality je seznam vlastních jídel a tlačítko



Obrázek 9: Dialog úpravy vlastního jídla

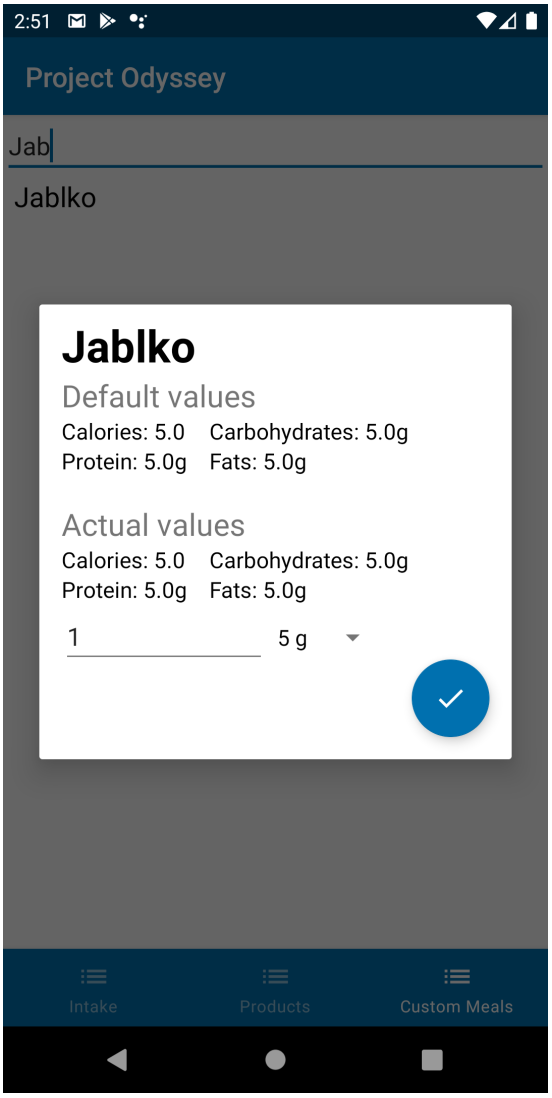
pro přidání nového jídla – obrázek 8. Položka vlastního jídla v seznamu obsahuje název, popis, tlačítko pro zkonzumování a tlačítko pro úpravu. Po kliknutí na tlačítko „Eat“ se přidají všechny produktu obsažené ve vlastním jídle do seznamu zkonzumovaných jídel. Následuje popis úpravy a přidání vlastního jídla.

Úprava vlastního jídla Po kliknutí na tlačítko úpravy se zobrazí dialog – obrázek 9. Dialog obsahuje název a popis jídla. Dále je zde možnost vybrat produkt, změnit mu množství a posléze potvrdit změnu kliknutím na tlačítko „Save“ po straně. Následuje část s hodnotami makronutrientů, které se automaticky aktualizují při změně množství. To je vhodné, když uživatel přemýšlí, zda-li nezačne snídat jablka dvě namísto jednoho. Stačí, aby změnil množství a hned vidí, jak by se hodnoty lišily. Ve spodním pravém rohu jsou tlačítka pro potvrzení změn a pro smazání vlastního jídla.



Obrázek 10: Přidání vlastního jídla

Přidání vlastního jídla Při přidání vlastního jídla je uživatel navigován na obrazovku 10. Zde uživatel musí vyplnit název jídla, popis jídla a přidat produkty, které chce, aby byly v jídle obsaženy. Poté potvrdí přidání jídla tlačítkem vpravo dole. Pro přidání produktu do jídla uživatel klikne na tlačítko „Add Product“. Poté se zobrazí obrazovka s textovým polem sloužící pro vyhledávání a seznamem produktů. Po zvolení požadovaného produktu se zobrazí dialog, kde si uživatel zvolí porci daného produktu a jeho množství. Toto je vyobrazeno na obrázku 11. Na pozadí lze vidět vyhledávací textové pole a seznam. Na popředí dialog pro přidání. Přidání uživatel potvrdí tlačítkem vpravo dole. Pro odebrání produktu ze seznamu stačí kliknout na jeho název.



Obrázek 11: Přidání produktu do vlastního jídla

3.4 Možná vylepšení

I když je aplikace funkční a dá se používat bez potíží, tak jsou zde určité aspekty, které by šly vylepšit. Hlavní problém spočívá v uživatelském rozhraní. Přesto, že jsem se snažil ho vytvořit co nejvíce přehledné a praktické, tak by šlo vylepšit. Nejlepší by bylo dát aplikaci vlastní styl, aby nepůsobila tak obyčejně. Při tomto „převlékání kabátku“ by se mohly vyřešit i další nedostatky jako špatné umístění řídících prvků. Dále by bylo dobré přidat animace na místa, kde by to dávalo smysl, například při odebírání produktů ze seznamu.

Z hlediska funkcionality by měla být možnost zadat datum při konzumaci produktu. Momentálně je datum vždy to „dnešní“. Dále by mohla být přidána funkce přidání čárového kódu k produktu a následného vyhledávání podle onoho kódu.

V kapitole 2.2 jsem zmínil, že existující řešení mají tu nevýhodu, že jsou příliš složité a matou uživatele. Také jsem uvedl, že kdybych měl přidat funkcionalitu, které přímo nesouvisí s měřením nutričních hodnot, tak bych se ji snažil náležitě oddělit od zbytku aplikace. Když budu chtít přidat funkcionalitu receptů. Tato sekce aplikace by sloužila k inspiraci a následné pomoci při přípravě pokrmů. Nijak zvlášť nepatří do zbytku aplikace. Jedno řešení, to horší, by bylo přidat další položku do menu na obrázku 1. V této pozici je tam tato položka zbytečně navíc a pouze zabírá místo a mate uživatele. Lepší řešení by bylo přidat hamburger menu[3]. Toto menu by pak obsahoval položku pro měření nutričních hodnot, receptů a dalších. Tímto způsobem jsou funkcionality bez většího vztahu odděleny a uživatel není zbytečně rozptylován.

[3]Hamburger menu je to, které vyjíždí z levé strany

4 Implementační detaily

V následující sekci se zaměřím na to, jak je aplikace naprogramovaná. Na začátek zmíním pár důležitých informací ohledně výběru verze Android SDK[4]. Poté zmíním knihovny, které jsem použil a krátce popíši, na co je která knihovna využita. Pak se ponořím do vlastního kódu. Přiblížím jeho základní strukturu a vybrané části proberu více do detailu. Na konec zmíním, jak by šel kód vylepšit.

4.1 Výběr verze Android SDK

Při výběru verze Android SDK jde o dvě hodnoty. Hodnotu takzvané minimální verze a hodnotu cílové verze.

Cílová verze je ta, pro kterou je aplikace určena. Z pravidla je to vždy nejnovější verze. Google dokonce požaduje používat novější verze, jinak aplikaci nebude možno přidat do Obchodu Play[5]. Zvolením novější verze má programátor přístup k novějším funkcím.

Minimální verze označuje nejnižší možnou verzi systému Android, na kterou půjde aplikace nainstalovat. Je lákavé nastavit tuto hodnotu na úplně první a podporovat tak všechny zařízení, ale udáním minimální verze programátor tvrdí, že aplikace bude fungovat i na této verzi. To znamená, že nějaká funkcionalita nemusí být přítomna. Proto špatný výběr minimální verze může vést k práci navíc.

Pro širokou veřejnost jsou Android verze známé jako například Android 5.0 Lollipop. Verze Android SDK jsou ale označovány pouhým číslem. V případě Android verze 5.0 je to pak číslo 21[6].

V mém případě jsem jako minimální verzi zvolil 21 a cílovou 29.

4.2 Knihovny

V této kapitole popíši knihovny, které jsem využil při vývoji aplikace. Jako knihovnu uvažuji všechny závislosti uvedené v souboru build.gradle. To znamená, že nejsou obsaženy v Android SDK a musejí být stáhnuty zvlášť.

Tabulka 1 ukazuje názvy knihoven a jejich verze v době psaní práce. Tabulka neobsahuje testovací knihovny. Pro knihovny z tabulky popíši jejich funkcionalitu a využití v aplikaci.

Constraint Layout Uživatelské rozhraní, které uživatel vidí na displeji svého zařízení, se skládá z různých komponent jako je například tlačítko. Tyto prvky jsou označeny jako View. View je nejzákladnější prvek uživatelského rozhraní a ostatní prvky, jako již zmíněné tlačítko, z něho dědí. Tyto prvky však musejí být na displeji nějak uspořádány. Takové uspořádání je práce takzvaných ViewGroup.

[4]Software Development Kit – vývojářské nástroje

[5]Obchod v zařízení, přes který se instalují aplikace

[6]Tabulku verzí lze najít na https://source.android.com/setup/start/build-numbers

Název knihovny	Verze
Constraint Layout	1.1.3
Recycler View	1.0.0
Kotlin extensions	1.2.0-alpha02
Coroutines	1.2.2
Coroutines Android	1.1.1
Material Theme 2.0	1.0.0
Firebase	17.0.1
Firebase Firestore	20.2.0
Navigation	1.0.0
Kodein	6.1.0
Lifecycle	2.2.0-alpha02
Room	2.1.0
Gson	2.8.5

Tabulka 1: Seznam knihoven

Například LinearLayout je ViewGroup, který uspořádává ostatní View v řadě za sebou. Android SDK nabízí další ViewGroup, ale když dojde k situaci, kdy rozložení prvků na displeji je složitější, tak je práce s nimi většinou nepraktická. Google tedy nabízí řešení tohoto problému v podobě Constraint Layout, které funguje na základě vztahů mezi komponenty uživatelského rozhraní a zbavuje se již zmiňované nepraktičnosti.

Recycler View Případ Recycler View je velice podobný Constraint Layout. Pro zobrazení seznamu prvků nabízí Android SDK ListView. Nevýhoda ListView je ale v tom, že při procházení seznamu dochází k tomu, že určité prvky mizejí a jiné se naopak zobrazují. Právě v tento moment je alokováno nové View, které se má zobrazit a staré, které nejde vidět, se označí Garbage Collectoru ke sběru. Tyto dva jevy mají za důsledek snížení výkonu. Řešením je právě Recycler View, který, jak už název napovídá, vezme View, které opouští displej a použije ho pro zobrazení dalšího prvku seznamu, který se právě dostává na displej. Jedna menší nevýhoda je to, že od programátora to vyžaduje větší spolupráci.

Kotlin extensions Ještě donedávna byl pro vývoj aplikací pro Android programátor nucen použit programovací jazyk Java. To se však změnilo, když se objevil jazyk Kotlin a byl Googlem přijat jako další oficiální možností, jak vytvářet aplikace. Kotlin, ve zkratce, se snaží vyřešit nepříjemnosti Javy. Ty problémy řeší například tak, že nabízí množství jazykových konstrukcí, které se nenacházejí v Javě. Jelikož je ale Android SDK napsáno v jazyce Java, tak tyto konstrukty nejsou dostupné. Ale dají se přidat právě pomocí Kotlin extensions.

Coroutines a Coroutines Android Další skvělý prvek jazyka Kotlin jsou korutiny[7]. V základu slouží k pozastavení vykonávání činnosti, dokud nějaká její část není dokončena, a to bez blokování vlákna, ve kterém je tato činnost vykonávána. Praktické využití je pak při asynchronním a paralelním programování, které je na Androidu striktně vyžadováno.

Material Theme 2.0 Material design je designový jazyk od firmy Google. Je přítomný v Androidu od verze 5.0 Lollipop – tedy API verze 21. Jeho druhá verze mírně upravuje specifikaci a přidává lepší možnosti jeho úpravy, aby byly aplikace originálnější na pohled.

Firebase a Firebase Firestore Firebase je cloudová platforma od firmy Google, která nabízí rozsáhlé množství produktů, ze kterých je v aplikaci využit produkt Firestore. Firestore je dokumentová cloudová databáze. Je spravována Googlem a tudíž nabízí snadný a jednoduchý přístup k uložení dat, aniž by se programátor musel starat o vlastní databázi a vlastní server. Dále v kapitole 4.4.

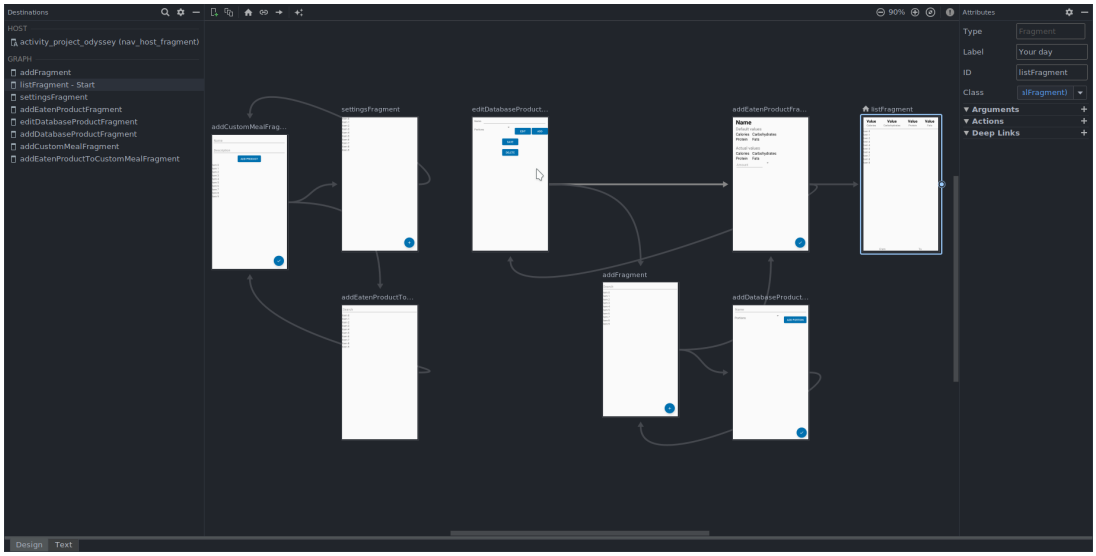
Navigation Jedna ze složitějších věcí při vývoji Android aplikace je navigace. Obrazovky aplikace jsou tvořeny pomocí tříd Activity a Fragment. Navigace mezi nimi je pak realizována pomocí třídy Intent, která představuje libovolnou operaci. V případě navigace je to operace navigace z jedné obrazovky na druhou. Tento systém je pak mírně nepřehledný, když se chceme navigovat z jedné obrazovky na druhou. Tato operace je znázorněna na zdrojovém kódu 1. Zde je malá výtka taková, že v objektově orientovaném jazyce by programátor čekal spíše něco jako `activity.navigate();`. Horší situace nastává, když programátor chce mezi obrazovkami komunikovat. Pokud chce pouze poslat data jedním směrem, tak před zavoláním funkce `startActivity;` přidá do Intentu požadované data pomocí `Intent.putExtra(String, Parcelable);`, kde první argument je identifikátor a druhý data. Druhá obrazovka pak musí data získat na základě indentifikátorů. Mnohem lepší by bylo data poslat/získat bez identifikátorů. Pokud by chtěla druhá obrazovka poslat data nazpátek, tak se nám situace stává ještě složitější. Tento případ tady nebudu rozebírat.

```
1        Intent i = new Intent(this, Activity.class);
2        startActivity(i);
```

Zdrojový kód 1: Navigace mezi obrazovkami - Intent

Výše zmíněné problémy řeší knihovna pro navigaci. Stačí si vytvořit soubor nav_graph.xml a v něm popsat obrazovky, jejich přesuny a data, které se mají poslat. K tomuto slouží i grafický nástroj – obrázek 12. Jde tedy vidět, že definice navigace je velice zjednodušená. V kombinaci s gradle pluginem „safe args“, který

[7]Správně označovány v informatice jako koprogramy



Obrázek 12: Grafický nástroj pro úpravu navigace

pro definované přechody vygeneruje metody, se stává navigace velice jednoduchá. Na straně volajícího – zdrojový kód 2. Lze vidět, že každá obrazovka má akce, které může provést, to jsou přechody. A akce mají argumenty, které mají datový typ definovaný v souboru nav_graph.xml. Na straně přijímající data pak jednoduše `val args by navArgs<EatenProductFragment>()`. Opět lze vidět správný datový typ.

```
1       val directions = CustomMealFragmentDirections.
            actionCustomMealFragmentToEatenProductFragment(products)
2       findNavController().navigate(directions)
```

Zdrojový kód 2: Navigace mezi obrazovkami - Safe Args

Kodein Při objektově orientovaném programování nastává často situace, kdy je potřeba, aby třída použila instanci jiné třídy. Většinou si onu instanci vytvoří sama. To bohužel vede k například horšímu testování třídy. Lepší způsob je, že si vyžádá, co potřebuje. Obvykle jako parametry konstruktoru. Požadované objekty pak přijdou „odněkud zvenčí“. Této technice se říká Dependency injection. A právě Kodein je k tomuto využit.

Lifecycle Již zmiňované Activity a Fragment, které tvoří obrazovky aplikace, prochází takzvaným životním cyklem. Ten se mění, když se obrazovka dostane do popředí, do pozadí či je aplikace systémem ukončena. Na tyto změny musí aplikace správně reagovat a to znamená více práce pro programátora. Práce, při které můžou vzniknout chyby. Kromě životního cyklu také aplikace mění konfigurace, například rotace zařízení. Na změny v konfiguraci se také musí reagovat.

Jelikož se při těchto změnách často vyskytují chyby, tak Google vydal právě tuto knihovnu, aby pomohl programátorům tyto chyby eliminovat.

Room Když nastane situace, kdy je potřeba uložit data do zařízení, tak Android poskytuje více možností. Jednou z nich je SQLite databáze, která je nejlepším řešením, když jsou data strukturována a je jich více. Jelikož práce s databází je činnost, ve které se vyskytuje hodně opakujícího se kódu navíc, tak Google vydal knihovnu Room, která tuto činnost značně zjednodušuje, více v kapitole 4.5.

Gson Opět další knihovna od Google. Tato knihovna slouží k serializaci a deserializaci JSONu.

4.3 Struktura kódu

Struktura kódu aplikace se dělí následovně:

- Modely
- Servisy
- Repozitáře
- Uživatelské rozhraní
- Pomocné části

Než popíši jednotlivé body ze seznamu, tak chci zmínit jednu třídu, která nezapadá nikam. Je to třída, která dědí ze třídy `Application`. Tato třída slouží k uložení globálních dat aplikace. V mém případě slouží pouze k definování pravidel pro knihovnu Kodein.

Modely

Modely jsou třídy, které logicky představují data v aplikaci. Existující modely v aplikaci jsou: `EatenProduct`, `DatabaseProduct`, `CustomMeal` a `NutritionValue`. `EatenProduct` představuje zkonzumovaný produkt. `DatabaseProduct` představuje produkt uložený v databázi a to jak online tak lokálně. `CustomMeal` představuje vlastní jídlo. `NutritionValue` představuje porci u produktu, její kalorie a makronutrienty.

Servisy

Servisy představují služby, které vykonávají specifickou činnost. `DatabaseProductServiceFirestoreImpl` slouží ke komunikaci s online databází Firestore. Tento servis dědí z rozhraní `DatabaseProductService`, aby bylo možné snadno změnit online databázi v budoucnu. Dále jsou servisy `EatenProductDao`, `DatabaseProductDao`, `CustomMealDao`, které slouží k ukládání dat do zařízení uživatele.

Repozitáře

Repozitáře představují hlavní zdroj pravdy ostatním částem aplikace. V mém případě existují tři repozitáře. Pro vlastní jídlo a zkonzumovaný produkt to jsou `CustomMealRepository`, `EatenProductRepository`, které pouze „obalují“ jejich příslušné servisy. V případě `DatabaseProductRepository` je pak změna, jelikož tento repozitář používá servisy dva. Jeden, který komunikuje s online databází a druhý, který komunikuje s lokální databází.

Uživatelské rozhraní

Je naprosto běžné, že v rámci aplikace existuje více Activit, jenomže při použití navigační knihovny to není potřeba. Tudíž má aplikace má pouze jednu Activitu a všechny obrazovky jsou tvořeny Fragmenty. Každý Fragment má příslušný .xml soubor, například frament_local.xml, který definuje uživatelské rozhraní dané obrazovky. Tento soubor existuje i pro onu jedinou Activitu, akorát je lehce odlišný. Není v něm definováno uživatelské rozhraní obrazovky. Místo toho je v něm definovaná kostra libovolné obrazovky. Dolní část navigačního menu a horní, většinová část, která je vyhrazena navigační knihovně a ta posléze ukazuje obrazovky dle navigace.

Do uživatelského rozhraní patří i soubory obsahující řetězce aplikace a jejich překlady. Soubory obsahující globální vzhled aplikace například barvy. A soubory, které jdou vykreslit, jako obrázky a vektorová grafika.

Pomocné části

Pomocné části aplikace obsahují kód nezávislý na ostatních částech. Obsahují zejména pomocné funkce pro zpříjemnění určitých úkonů.

4.4 Databáze produktů

Jak jsem již několikrát zmínil, databáze na ukládání produktů jsou dvě, online a lokální. V této kapitole se zaměřím na online databázi a lokální popíši v kapitole 4.5.

Aby bylo možné v budoucnu vyměnit databázi bez větších obtíží, tak existuje rozhraní `DatabaseProductService`, které je definováno na zdrojovém kódu 3. Toto rozhraní definuje základní metody potřebné pro komunikaci s databází. Teď jen stačí, aby třída, která chce komunikovat s databází, implementovala toto rozhraní. A přesně to dělá `DatabaseProductServiceFirestoreImpl`. I když není uživateli dovoleno upravovat online databázi, tak v budoucnu může existovat možnost admin verze aplikace, ze které by to bylo možné, takže implementovat tuto funkcionalitu není zbytečné. Také by mohla nastat situace, kdy by uživatelé měli uloženy produkty online a nikoliv v zařízení.

```
interface DatabaseProductService {
    suspend fun getAll(): List<DatabaseProduct>
    suspend fun getById(id: String): DatabaseProduct?
    suspend fun getByName(name: String): List<DatabaseProduct>?
    suspend fun add(product: DatabaseProduct)
    suspend fun update(product: DatabaseProduct)
    suspend fun delete(product: DatabaseProduct)
}
```

Zdrojový kód 3: Rozhraní DatabaseProductService

4.5 Ukládání dat v zařízení

V zařízení se ukládají tři typy dat. Zkonzumovaný produkt, vlastní jídlo a produkt, který uživatel přidá k následné konzumaci. Jak jsem již zmiňoval, ukládání dat v zařízení je prováděno pomocí knihovny Room.

Nyní popíši postup implementace pro `EatenProduct`, ale pro ostatní je to velice podobné. První důležitý krok je přidat anotaci `@Entity` ke třídě, jak lze vidět na zdrojovém kódu 4. Tato anotace označuje, že budeme do databáze ukládat instance této třídy. Dále je důležitá anotace `@Embedded` pro typy, které nejsou primitivní a nemají definovaný converter. A poslední důležitá anotace `@PrimaryKey`, která označuje primární klíč vytvořené tabulky.

```
@Entity
data class EatenProduct(
    var firestoreId: String,
    val name: String,
    @Embedded(prefix = "nv_")
    var portion: NutritionValue,
    var amount: Double,
    @PrimaryKey(autoGenerate = true)
    val id: Int = 0,
    val dateOfEating: Date
)
```

Zdrojový kód 4: Třída EatenProduct

Takto připravená třída potřebuje Data Access Object. To je objekt sloužící k interakci mezi databází a ostatním kódem. Zdrojový kód 5 znázorňuje DAO pro `EatenProduct`. Tento zdrojový kód není úplný, slouží pouze jako příklad. DAO je tedy rozhraní s anotací `@Dao` a metodami se specifickými anotacemi. Lze vidět, že pomocí anotace `@Query` lze provést libovolnou operaci nad databází.

Posledním krokem je definovat třídu, která bude komunikovat se samotnou databází. To je třída dědící z `RoomDatabase` – zdrojový kód 6. Opět je zdrojový kód neúplný. Databáze musí vědět, které objekty má ukládat, k tomu slouží anotace `@Database` a její parametr `entities`. Anotace `@TypeConverters` udává,

```
@Dao
interface EatenProductDao {
   @Query("SELECT * FROM EatenProduct")
   suspend fun getAll(): List<EatenProduct>

   @Query("SELECT * FROM EatenProduct WHERE id = :id")
   suspend fun getById(id: Int): EatenProduct

   @Insert
   suspend fun add(product: EatenProduct)

   @Update
   suspend fun update(product: EatenProduct)

   @Delete
   suspend fun delete(product: EatenProduct)
}
```

Zdrojový kód 5: Třída EatenProductDAO

které vlastní convertery má databáze použít. Converter je libovolná třídá, která má metody označené anotací `@TypeConverter`. V tomto případě je nutné mít converter pro třídu `Unit`, která je součástí třídy `NutritionValue`. Implementace tohoto converteru je na zdrojovém kódu 7. Databáze je plně připravena a posledním krokem je získat její instance. Získání instance je na zdrojovém kódu 8.

```
@Database(
   entities = [EatenProduct::class]
)
@TypeConverters(Converters::class)
   abstract class ProjectOdysseyDatabase : RoomDatabase() {
      abstract fun eatenProductDao(): EatenProductDao
}
```

Zdrojový kód 6: Třída ProjectOdysseyDatabase

4.6 Možná vylepšení

Jak jsem se zmínil v kapitole 3.4, aplikace funguje, ale je prostor ke zlepšení. Prvním krokem je použít Lifecycle knihovnu, a její komponenty ViewModel a LiveData, všude, kde je na to v aplikaci prostor. Odstranilo by to kód, který je tam momentálně navíc a zamezilo by to možnému výskytu chyb. Dále by se měl kód refaktorovat. Opět by se zamezilo chybám a vylepšila by se navigace kódem. Například tvorba dialogů je pro mě v nežádoucím stavu. Další problém je dokumentace. Prakticky celý kód není zdokumentován, to by mohlo do budoucna

```
class Converters {
   @TypeConverter
   fun fromUnit(unit: Unit?): Int? = if (unit == Unit.G) 0
      else 1

   @TypeConverter
   fun toUnit(num: Int?): Unit? = if (num == 0) Unit.G else
      Unit.ML
}
```

Zdrojový kód 7: Třída Converters

```
Room.databaseBuilder(
   context.applicationContext,
   ProjectOdysseyDatabase::class.java,
   "odyssey.db"
).build()
```

Zdrojový kód 8: Tvorba instance databáze

působit problémy. Dokonce během vývoje tento fakt působil problémy. A posledním větším problémem je testování. Aplikace není podrobně otestována, takže můžou být přítomny chyby, které jsem ručním testováním neodhalil.

Závěr

Cílem práce bylo vytvořit aplikaci pro Android, která by pomáhala uživateli se správou zkonzumovaných jídel. Vytvořená aplikace tedy umožňuje přidávat produkty, jako například jablko, do seznamu zkonzumovaných jídel. Z tohoto seznamu posléze vypočítá množství kalorií a makronutrientů za zvolené období. Dále poskytuje upravitelnou databázi produktů, do které může uživatel přidat další produkty, pokud je databáze již neobsahuje. V poslední řadě si může uživatel vytvořit takzvané vlastní jídlo. Vlastní jídlo se skládá z více produktů například jablko a chléb. Takto vytvořené vlastní jídlo pak může uživatel přidat do seznamu jako další produkt.

S aplikací, mimo rámec bakalářské práce, mám plány do budoucna. Tyto plány obsahují například lepší uživatelské rozhraní vytvořené ve spolupráci s designerem a přidání dalších funkcí jako je vyhledávání pomocí čárového kódu a možnost přidat obrázky k produktům.

Conclusions

The goal of this bachelor thesis was to develop an application that would help user manage consumed food. Final application therefor lets user add product to list of eaten products and then calculate amount of calories and macronutrients in the given time period. Additionally it provides editable database of products and allows user to add own product if there is none in the database. And lastly user can create custom meals that consist of multiple products. This custom meal can then be added to the list of eaten products as any other product can be.

I have plans with this application outside the scope of this bachelor thesis. First on the list is to create a better user interface in collaboration with a skilled designer. And then I would like to add some new functionality like barcode search and product images.

A Obsah přiloženého CD/DVD

bin/
: Obsahuje instalační soubor aplikace – NUTRITION-TRACKING.APK.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové kódy aplikace.

readme.txt
: Instrukce pro instalaci aplikace.

U veškerých cizích převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na CD/DVD, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `readme.txt`.

Literatura

[1] Bill Phillips Chris Stewart, Kristin Marsicano. *Android Programming: The Big Nerd Ranch Guide*. Third. 2017. ISBN 0134706056.

[2] Dmitry Jemerov, Svetlana Isakova. *Kotlin in Action*. 2017. ISBN 1617293296.

[3] *Documentation for app developers*. Google LLC. Dostupný z: ⟨https://developer.android.com/docs⟩.