UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Mobilní aplikace podporující fitness lifestyle

Jindřich Mikule

Bakalářská práce
2015

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2014/2015

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jindřich Mikule**
Osobní číslo:	**I12185**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Mobilní aplikace podporující fitness live style**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit mobilní aplikaci určenou pro cílovou skupinu uživatelů, která se soustředí na trendy moderního životního stylu a to včetně moderních stravovacích návyků. Aplikace bude obsahovat databázi potravin, receptů a bude schopna dle požadavků uživatele vytvářet individuální stravovací plán. Individuální stravovací plán bude v rozšířené verzi obsahovat konkrétní recepty, seznam požadovaných surovin k nákupu atp., čímž bude podporovat selektivní spotřebu uživatele. Užité technologie a paradigmata: Databáze SQL lite, Android Studio, JAVA SE, XML, data mining.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

DARWIN, Ian F. Android cookbook. First edition. Beijing: O'Reilly, 2012, xix, 661 pages. ISBN 14-493-8841-8,
MEIER, Reto. Professional Android 4 application development. Updated for Android 4. Indianapolis: John Wiley, 2012, xlii, 817 pages. ISBN 978-111-8262-153,
PHILLIPS, Bill a Brian HARDY. Android programming: the Big Nerd Ranch guide. 1st ed. Atlanta, Ga.: Big Nerd Ranch, 2012, xxii, 602 pages. ISBN 03-218-0433-3

Vedoucí bakalářské práce: **Ing. Josef Brožek**
Katedra softwarových technologií

Datum zadání bakalářské práce: **20. prosince 2014**
Termín odevzdání bakalářské práce: **11. května 2015**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2015

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 04. 05. 2015 Jindřich Mikule

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce Ing. Josefovi Brožkovi za jeho odborné vedení, cenné rady, připomínky, podněty a veškerou poskytnutou pomoc.

Anotace

Bakalářská práce je věnovaná vývoji aplikace pro mobilní telefony. Soustřeďuje se na podporu skupiny lidí, která by chtěla začít žít zdravým životním stylem. Aplikace je navržená pro platformu Android. V mobilní aplikaci uživatel může využít návrhy individuálních stravovacích plánů a žít tak zdravěji. Pro ukládání dat slouží databáze SQLite. Aplikace je naprogramována v Javě.

Klíčová slova

Android, Java, SQLite, XML, kuchařka, individuální stravovací plán, mobilní aplikace, živiny

Title

Mobile application for fitness lifestyle

Annotation

This thesis is focused on creating application for mobile phones. It is focused on support group of people, which they would like to start live healthy lifestyle. Application is developed for the Android platform. At mobile application, user may accept suggestion of individual dietary plan and user may live healthier. For saving data is database SQLite. Application is developed by Java.

Keywords

Mobile application, Java, Android, SQLite individual dietary plan, cookbook, nutrients

Obsah

Seznam zkratek

ASCII	American Standard Code for Information Interchange.
BMR	Basal Metabolic Rate
BMI	Body Mass Index
ČSN	Česká technická norma (dříve Československá státní norma)
DB	Databáze
DP	Density-independent Pixel (DP nebo i DIP)
ER	Entity-relationship
ERD	Entity-relationship diagram
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
ICT	Information and Communication Technologies
IDE	Integrated Development Environment
IS	Informační systém
IT	Informační technologie
JDK	Java Development Kit
KT	Kalorické tabulky
MFP	MyFitnessPal
OOP	Objektově orientované programování
SE	Standard Edition
SDK	Software Development Kit
SQL	Structured Query Language
ISO	International Organization for Standardisation
PDA	Personal Digital Assistant
PHP	Personal Home Page (Hypertext Preprocessor)
UID	User IDentifier
UML	Unified Modeling Language
XML	Extensible Markup Language

Seznam obrázků

1 Úvod

1.1 Motivace

Jednou z klíčových motivací byla použitelnost programu pro laické uživatele. Můj program by neměl jen ležet v archivu univerzitní knihovny. Měl by mít při nejhorším aspoň jednoho spokojeného uživatele. Myslím si, že každý programátor si přeje, aby jeho software byl někdy používán. Motivace k tomu, proč jsem si vybral bakalářskou práci právě na téma: „Mobilní aplikace podporující fitness lifestyle“ a ne na úplně jiné téma, není vůbec snadná otázka, ale pokusím se popsat všechny myšlenky, které se mně honily hlavou, když jsem si práci mezi tématy vybíral.

Kladl jsem sám sobě otázku, chci programovat nějakou aplikaci pro desktopy, mobilní zařízení anebo bych raději navrhl webové stránky? Webové stránky? Šablona na stránky je v poměru s aplikací o dost rychleji napsána. Zdrojových kódů bude maximálně na 200 až 300 řádků. Značkovací jazyk HTML je o dost jednodušší, proto by toto nebylo tak náročné na přemýšlení. V případě databází bych přišel do styku maximálně se dvěma tabulkami. Jedna by byla pro uživatele stránek a druhá pro role, které v systému vykonávají. Na druhou stranu bych musel navrhovat celý design, který je klíčový. Desítky hodin učení se designu v grafickém editoru. Navíc by k navrhování dynamického webu muselo být použito PHP a JavaScript, které jsem si neoblíbil tak, jako Javu nebo C++. Nemám ze sebe pocit, že bych v současné době chtěl navrhovat šablonu. Ne, rozhodl jsem se, v současné době ze mě grafik ani webový kodér nebude.

Takže moje práce bude aplikace. Program pro desktopy by byla jistá cesta k úspěšnému konci. Semestrálních prací na toto téma jsem měl už dostatek a není vůbec nutné použít databázi. Proto bych ušetřil spoustu času při jejím návrhu. Ale mobilní aplikace je pro mě něco nového, neprozkoumaného a je pro mě zajímavější z hlediska budoucího uplatnění se na trhu práce. Věřím, že to bude výzvou.

Jakou skupinu aplikací zvolím? Bude to antivirový program? Ne, rozhodně ne. Grafický editor? To zní zajímavě. Možná jindy. Nebo hra? Ještě zajímavější. Asi v příští diplomové práci zkombinuji naposledy jmenované dvě skupiny, teď nemám dostatek zkušeností, potřebných k naprogramování jakékoliv zábavné hratelné hry. Textový, tabulkový editor? Ne. Moje řešení by nebylo lepší než konkurenční. Co takhle program zaměřený na zdraví, napadlo mě při procházení jednotlivých možností. No, to by snad mohlo být zajímavé. Mít uživatele, kterým aplikací nejen posloužím, ale taky pomůžu k zlepšení zdravotního stavu.

Teď už snad jen, na jaké téma bude moje aplikace. Nad tím jsem nemusel ani dlouho přemýšlet. Vždy jsem se chtěl seznámit s oborem výživového poradenství a fitness. Je to globální problém, se kterým se potýká hodně lidí. Nevědí, co si dát k jídlu nebo co si uvařit a tak si koupí nějaký uzeninový salát plný majonézy a konzumního salámu, který v sobě neobsahuje příliš toho, co člověk ke zdraví potřebuje. Zděsil jsem se, když jsem si uvědomil, jak nezdravě jím od doby, kdy jsem začal chodit na vysokou školu.

Javu jsem se učil jako první, proto věřím, že ji ovládám ze všech programovacích jazyků nejlépe. Nejrozsáhlejší platformou pro mobilní zařízení je Android. Ten hlavně podporuje výše zmíněný programovací jazyk.

2 Rešerše již existujících řešení

Při vývoji aplikace je dobré prohledat konkurenční trh. Najít, v čem jsou podobně provedené aplikace lepší anebo stejné či horší. V prvním případě zjistit, jestli jejich řešení nešlo nějakým způsobem vylepšit, ve druhém případě, zda provedení je dostatečně kvalitně zpracováno. V obou případech je dobré se inspirovat a přidat do aplikace žádané funkce, které aplikaci prospějí. I maličkosti při výběru aplikace mohou hrát roli. Pokud se jedná o vzhled aplikace, není příliš dobré se nechat přespříliš inspirovat. Uživatelé totiž nemají v oblibě „plagiátory". To, na co se nejvíce soustředí, je to, co vidí na obrazovkách svých mobilních zařízení: uživatelské rozhraní GUI. Například pokud zjistí, že tato aplikace vypadá podobně jako jiná, konkurenční, která se jim příliš nelíbila nebo nesplnila očekávání, mohou uživatelé udělat ukvapené závěry, že autorem je stejný vývojář či skupina vývojářů. Ověření stávajících řešení je popsáno přímo v této kapitole.

Průzkum se týká pouze verzí pro mobilní telefony s OS Android vyskytující se na Google play, obchodu s aplikacemi. Dále se týká pouze stabilních ukončených projektů a projektů se stále vycházejícími novými verzemi anebo updaty. Nehotové, nestabilní nebo nefunkční řešení není dobré zahrnout do tohoto průzkumu.

Android Market obsahuje nepřeberně velké množství mobilních aplikací pro lidi, kteří chtějí žít zdravým životním stylem. V drtivé většině případů však neobsahují češtinu a bývají různým způsobem zpoplatněny. Nejzajímavější android mobilní aplikace, zabývající se tímto tématem, jsou popsány v následujících pododdílech.

2.1 MyFitnessPal

Při prvním spuštění aplikace je možné vidět obrazovku sloužící k registraci nového uživatele. Lze ji provést přímo v aplikaci i na webových stránkách aplikace. K vytvoření účtu anebo při prvnímu přihlášení je potřeba připojení k internetu. Přihlášení k aplikaci lze provést za pomocí Facebook účtu nebo přes osobní e-mail. Aplikace je nabízena zdarma. Tato přednost je kompenzována umístěním množství reklam, které se zobrazují v dolní části aplikace.

Program nejenže umožní za pomoci různých výpočtů sledovat kalorie, ale stejně tak dobře může sledovat i cvičení a aktivity uživatele. V aplikaci je možné přidat si jakékoliv množství přátel a zobrazit jejich jídelníčky anebo jejich aktivity. Uživatelům je za pomoci rozhraní umožněno navzájem se povzbuzovat a tak podobně. Pokud dojde na přidávání potravin, aplikace má jejich obrovskou databázi. Developeři slibují, že by jich mělo být až pět milionů. Najít tedy potravinu, která není v knihovně přidána, je přinejmenším těžké. Pokud i přesto potravina v DB není lze jí přidat ručně nebo pomocí naskenování čárových kódů.

Aplikace umí sledovat váhu, podobně jako dává odhady toho, co si vyhodnotí jako ideální stav pro uživatele. Program je také podporován mnoha dalšími vývojáři a integruje se s mnoha jinými aplikacemi. Příkladem aplikací jsou Fitbit, Jawbone UP, Garmin, MapMyFitness, Runkeeper, Strava, Runtastic, Misfit, Withings, Healthkit.

Obrázek 1 - Ukázka MyFitnessPal

2.2 Noom Coach: Weight Loss Plan[1] [2]

Základ aplikace Noom je zcela zdarma. Pokud uživateli nestačí základní verze, může si přidat další nové vlastnosti aplikace. Platí se však za ně ne příliš přívětivou částkou. Základní licence začíná na třiceti dnech. Pokud vyprší, uživatel přijde o všechny získané funkce. Pokud chce uživatel získat funkce nazpět, musí zaplatit znovu a jeho předchozí funkce se opět obnoví. Jak už bylo výše zmíněno, platí se paušálně. Předplatné začíná na 10$, což je v přepočtu podle aktuálního kurzu 245,63 Kč za jeden měsíc, 20 amerických dolarů (491,30 Kč) za 3 měsíce a za jeden rok předplatného 40$ (982,60Kč).

Noom Weight Loss je aplikace, která pomáhá jejím uživatelům hubnout. Kombinuje funkce kardio trenéra a metod kalorických hubnutí do jednoho snadno použitelného balíčku. Po spuštění aplikace je to podobné jako s předchozí aplikací. Uživateli jsou nabídnuty dvě možnosti: přihlásit se nebo vytvořit nový účet. Registrací a celou aplikací uživatele provází usměvavý emotikon, který klade všetečné otázky. Krátce po registraci program nabídne sestavení jídelního rozvrhu na aktuální den. Po jeho vytvoření se na obrazovce zobrazí testovací podprogram krokoměru. Test spočívá v tom, že se má se zařízením nahoru a dolu zatřást. Skoro ke každé funkci jsou poskytovány informace i s názornými obrázky, o tom jak správně hubnout. Prezentace aktivit obsahuje například:

„Vytvořte si deficit kalorií. Docílíte tím úspěchu“,

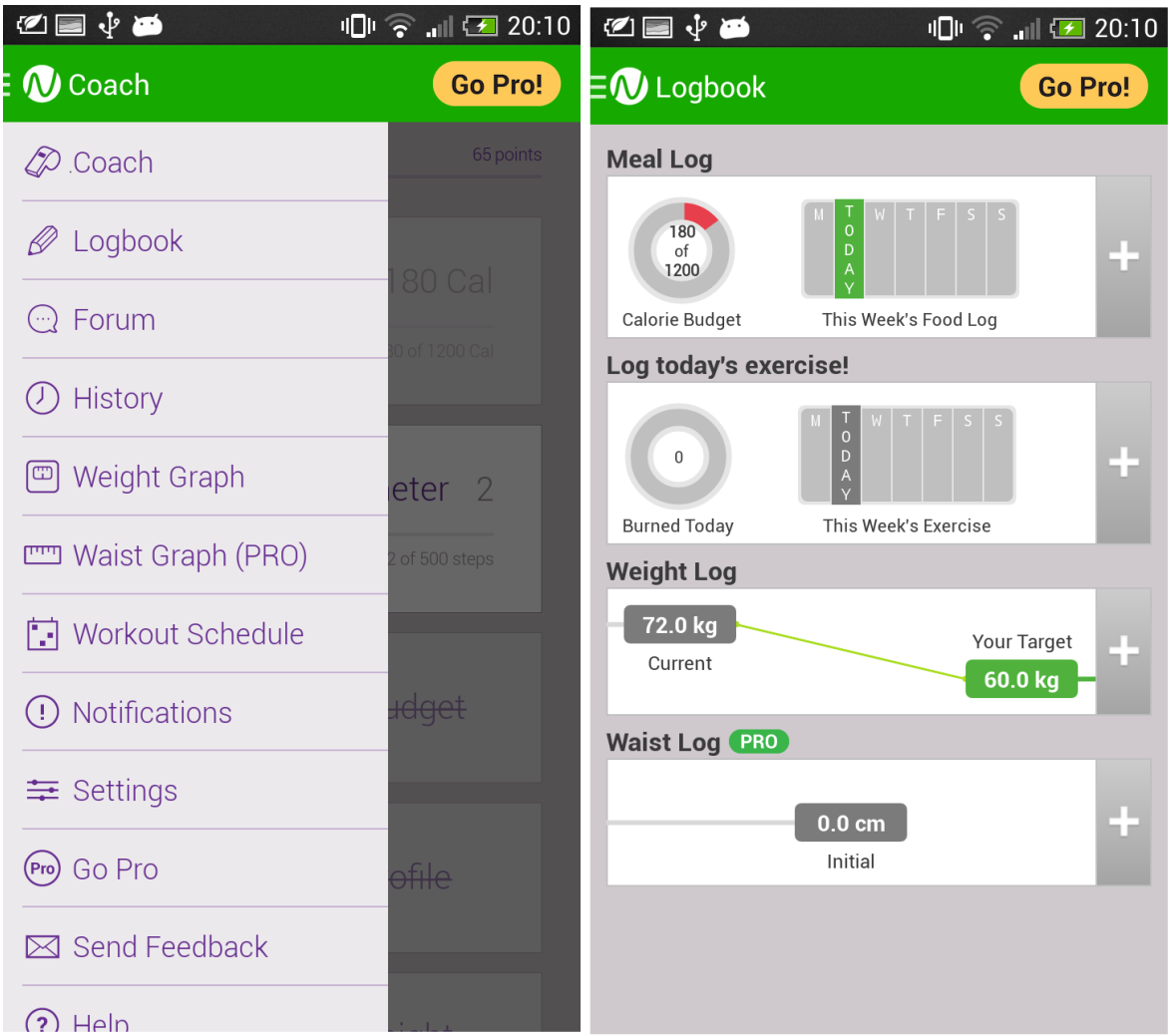
„Cvičení je těžké. Muffin cca s 400cal spálíme intenzivní chůzí na vzdálenost 6-ti kilometrů“

Po prezentaci následuje dotazník:

„Jaké jídla přes den obvykle jíte?“

Možnosti: *„snídaně, svačina, oběd, večeře“*

Program umí v základní verzi automaticky sledovat pohyb na mapě, počítat ušlé kroky, spočítat rozpočet kalorií na den, vykreslovat graf historie váhy anebo zobrazit historii jídelníčku. Pro motivaci uživatele používá takzvaného „achiev systému“, který za pomoci dosáhnutých úspěchů uživatele nutí plnit úlohy aplikace. V placené rozšířené verzi aplikace poskytne pár nových funkcionalit: spojení se s ostatními uživateli přes Facebook nebo Twitter, znázornění historie šířky pasu a dietní recepty. Program tedy umí sestavit plán hubnutí a taky dokáže sledovat pokrok. Všechny komponenty jsou intuitivně rozmístěny tak, aby se program co nejsnadněji ovládal.



Obrázek 2 - Ukázka programu Noom Coach[2]

2.3 Kalorické tabulky

Jako u předchozích dvou aplikací i u této je povinná registrace, při které je potřeba uvést základní informace jako je rok narození, výška, váha a pohlaví. Tyto skutečnosti umožní spočítat BMR metabolismu. Ten určuje, kolik tělo vydá energie v klidovém stavu.

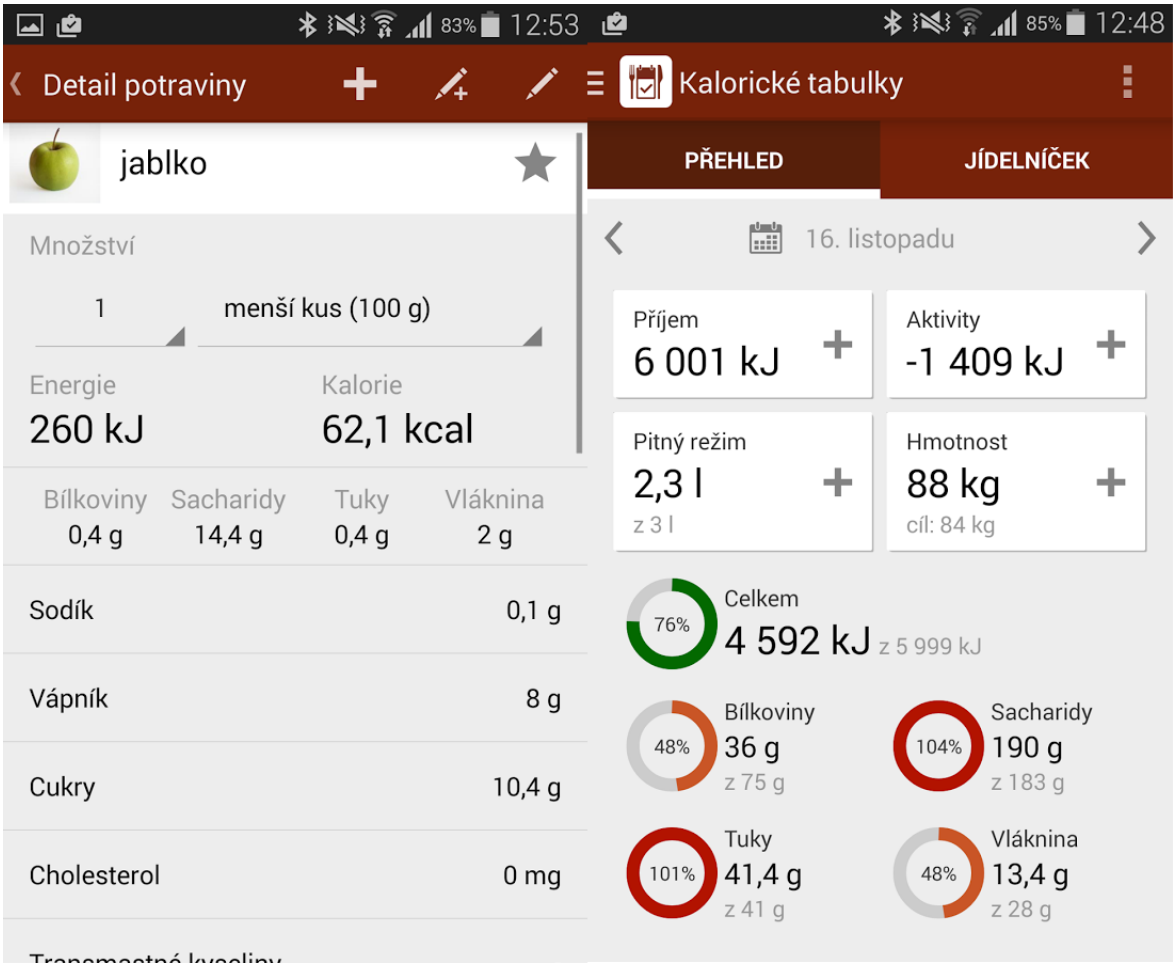
V případě, že se uživatel rozhodl hubnout, jednoduše ve Správě údajů nastaví cílovou hmotnost. Denní energetický příjem se automaticky přepočítá a sníží se tak, aby se dosáhlo optimálního hubnutí, to je přibližně půl kilogramu týdně. Pokud je některá potravina v databázi k nenalezení, lze navrhnout potravinu k uložení do databáze.

Důkladnější sledování vývoje hmotnosti nám umožní graf, který poskytuje informace o průběhu hubnutí. Stačí pravidelně zapisovat aktuální hmotnost ve Správě údajů. Dále aplikace zobrazuje hodnoty bazálního metabolismu a BMI. Nechybí ani informace o doporučených denních dávkách důležitých složek potravin.

Do jídelníčku se přidávají potraviny velmi snadno. Nejlepší je ale využít vyhledávání a našeptávače. Jednotlivé potraviny se dají označit jako Oblíbené anebo lze jednoduše ukládat celá jídla. Znovu je pak najdeme v sekci Oblíbené položky nebo Uložená jídla. Tím si usnadníme příští zadávání.

Dalším způsobem je použití skenování čárového kódu přímo z obalu potraviny. Do jídelníčku také vkládáme provedené aktivity, které nám odečtou energii z denního příjmu. Umožněno je i ukládání plánovaných nebo pravidelných aktivit. Kalendář nám umožní plánovat si jídelníčky dopředu a zároveň i ohlédnout se do předchozích dní. Uložené potraviny zůstanou přístupné, můžeme je různě měnit anebo sami smazat. V jídelníčku se nám zobrazuje i doporučený denní cíl pitného režimu. Započítávají se zde všechny nápoje, včetně alkoholických. Cílová hodnota se dá změnit podle konkrétní potřeby ve Správě údajů.

Všechny údaje jsou synchronizované s webovou verzí. I to je důvodem, proč aplikace Kalorické tabulky nefunguje offline bez internetového připojení.



Obrázek 3 - Ukázka aplikace Kalorické tabulky

2.4 Srovnání

Všechny aplikace nabízejí podobnou, až stejnou funkcionalitu. Dokážou vytvářet různé statistiky, které se týkají energetických hodnot zkonzumovaných potravin nebo tělesné váhy uživatele v časovém období. Dávají možnost dopředu si naplánovat jídelníček i zobrazit jejich historii. Dietní plán (jídelníček) stanovuje, jaké potraviny se budou v daný moment konzumovat. A lze na nich opakovaně nastavovat cílovou a aktuální váhu.

2.4.1 Kalorické tabulky

Kalorické tabulky mají mnohem lépe rozepsané informace o potravinách v porovnání s ostatními aplikacemi. Tabulky mají jako všechny aplikace energetickou hodnotu potraviny, bílkoviny, sacharidy, tuky. Navíc však obsahují i glykemický index GI, který udává rychlost využití glukózy tělem z určité potraviny. Dále uchovávají informace o vápníku, sodíku, vláknině, cholesterolu, transmastných kyselinách a o obsahu vitamínů a minerálů. To, čím se tabulky hlavně odlišují od jiných aplikací je zpráva o vlivu na zdraví a velmi užitečné praktické rady o potravině. Pro názornou ukázku by bylo dobré si jich pár ukázat.

Cituji praktické rady z aplikace Kalorických tabulek:

„Banán se dnes využívá také v kosmetice nejčastěji jako pleťová maska. Banány nikdy neskladujeme v lednici. Při výběru banánu dbejme na barvu slupky. Pokud je našedlá znamená to, že plody zrály při nízké teplotě a brzy se začnou kazit. Nedoporučuje se před vrcholným výkonem, protože má lehce tlumící účinky."

„Pokud vám trávení laktózy v kravském mléce způsobuje potíže a nechcete se jeho konzumace vzdát, zkuste je ve formě zakysaných mléčných výrobků. Pakliže chcete kravské mléko zcela nahradit, poohlédněte se ve zdravé výživě. Na výběr je dnes již řada produktů. Od mléka kozího, ovčího, někdy se podaří sehnat i mléko kobylí. Dále je možné koupit „mléko" sójové, mandlové, z kešu oříšků apod., které sice nejsou v pravém slova smyslu mléka, ale v kuchyni je využijete stejným způsobem. Hodí se však i k přímé konzumaci."

Cituji vlivy na zdraví z aplikace Kalorických tabulek:

„Blahodárně působí na mozek, nervovou soustavu a svalovou soustavu. Doporučuje se osobám s nemocným žaludkem, protože má zklidňující účinky. Banány jsou lehce stravitelné. Pro svůj vyšší obsah sacharidů a nízký obsah tuku je ocení zejména sportovci. Je možné je zařadit i do redukční diety, ale je vhodné střídat je i s méně sladkými druhy ovoce."

„Vápník, jehož zdrojem je i mléko, je základní součástí lidského těla. Nachází se v kostech, zubech, svalech, krvi a dalších tělesných tkáních. Pro lidskou stravu je tak vápník velmi podstatný. Jeho přísun je důležitý především u dětí (vývoj) a u těhotných žen. Také je nutno říci, že pokud se má v těle vápník zabudovat, je zároveň žádoucí dostatek vitamínu D. Při dlouhodobém deficitu vápníku (vit. D) v potravě, je větší riziko vyplavování vápníku z kostí a vzniku osteoporózy (týká se zejména starších osob, kterým se pak kosti lehce zlomí-např. i při prudkém zakašlání)."

Další předností tabulek je česká lokalizace oproti oběma následujícím řešením. Krásnou funkcí je i katalog aktivit, který spočítá přibližnou energetickou hodnotu, kterou při pohybu tělo spálilo za určitý čas nastavovaný v hodinách. Absolutně nejlepší vlastností aplikace je, že je distribuována pod volnou licencí bez omezení, která je úplně zdarma. Další věc, kterou používá je mojeID, která má nahrazovat registraci osobních údajů.

Naopak nedostatkem může být chybějící synchronizace se sociálními sítěmi a absence vlastního chatu mezi uživateli aplikace. Co některým uživatelům může připadat jako ztráta času, je vytvářet si nový účet a vyplňovat znovu ta samá data a potvrzovat aktivační e-mail. A právě kvůli chybějícím sociálním sítím se nedají mezi uživateli sdílet jejich aktivity ani jídelníčky jinak, nežli odkazem za pomoci webové aplikace, která vygeneruje odkaz v zadaném časovém rozmezí a ukáže všechny aktivity a jídla. Hlavním nedostatkem je, že aplikace nepodporuje uživatele, kteří nemají mobilní internet či připojení k němu, tudíž aplikace neumí plnohodnotně běžet v režimu offline. A je tu ještě poslední věc a tou jsou všude přítomné reklamy, nepříjemně zanimované tak, aby si jich každý uživatel povšiml.

2.4.2 Noom Coach: Weight Loss Plan

Noom Coach je velice interaktivní aplikací oproti srovnávaným konkurentům. Tu zapříčiňuje jakýsi průvodce. Ten říká, jak se má uživatel při hubnutí chovat správně pomocí několika po sobě jdoucích slajdů, obvykle končící dotazníkem o uživateli. Po stažení receptů na paměťovou kartu může aplikace fungovat i v offline režimu, což je výhodou pro ty uživatele, kteří rádi cestují nebo jsou mimo dosah mobilního internetu.

Třetí výhodou oproti Kalorickým tabulkám a MyFitnessPal je, že má Noom Coach vlastní krokoměr, který i dokáže spočítat ušlou vzdálenost a zaznamenat výkon na mapu. O interakci svědčí hlasový výstup programu, který informuje o ušlé vzdálenosti a době, za kterou se vzdálenost ujde a po skončení cvičeni oznámí, kolik se spálilo kalorií. Další funkci, kterou ostatní srovnávané aplikace nemají, je vlastní přehrávač hudby, zpříjemňující plnění tréninku. Ve srovnání s Kalorickými tabulkami má tato aplikace možnost vynechání registrace a přihlášení se pomocí sociálních sítí Twitter a Facebook. Při pohledu na srovnávané aplikace je uživatelské rozhraní nejlépe zpracováno, barvy jsou plné, výrazné a kontrastní. Dále registrace nepožaduje tolik informaci jako KT nebo MFP a proto má z nich nejpohodlnější průběh. Údaje se nastavují až v prezentacích.

Nedostatkem je určitě anglická lokalizace, která se nedá přepnout na češtinu. Krokoměr neumí počítat kroky při režimu spánku displeje, což je však chyba, kterou vývojáři pravděpodobně časem odstraní, protože je to stále aktivní projekt. Nemá inbox chat a nepodporuje možnost synchronizace s ostatními fitness aplikacemi kromě svých.

2.4.3 MyFitnessPal

Bezesporu hlavní předností aplikace je ohromná databáze potravin a receptů. I to, že je aplikace zcela zdarma, je nemalou výhodou. Ve srovnání s předchozími aplikacemi, je s obsahem 5 miliónu řádků v tomhle ohledu strčí do kapsy. Aplikace má jednoduchý, dvou až tříbarevný, nerušivý design ve srovnání s výše uvedenými řešeními. Je také nejlepší, co se týče provázanosti se sociálními sítěmi (dokáže sdílet a posílat statusy obsahující jídelníčky a právě provedené sportovní výkony) a propracovaností inbox zpráv. Další výhodu, kterou však lze snadno v nastavení vypnout, je upozorňování, které oznámí v určitou hodinu, že si uživatel měl dát předem zadané jídlo a předvolené množství tekutin. Zapomenout se nesmí ani na nejintuitivnější řešení aplikace v porovnání s programy popsané v kapitole 2.

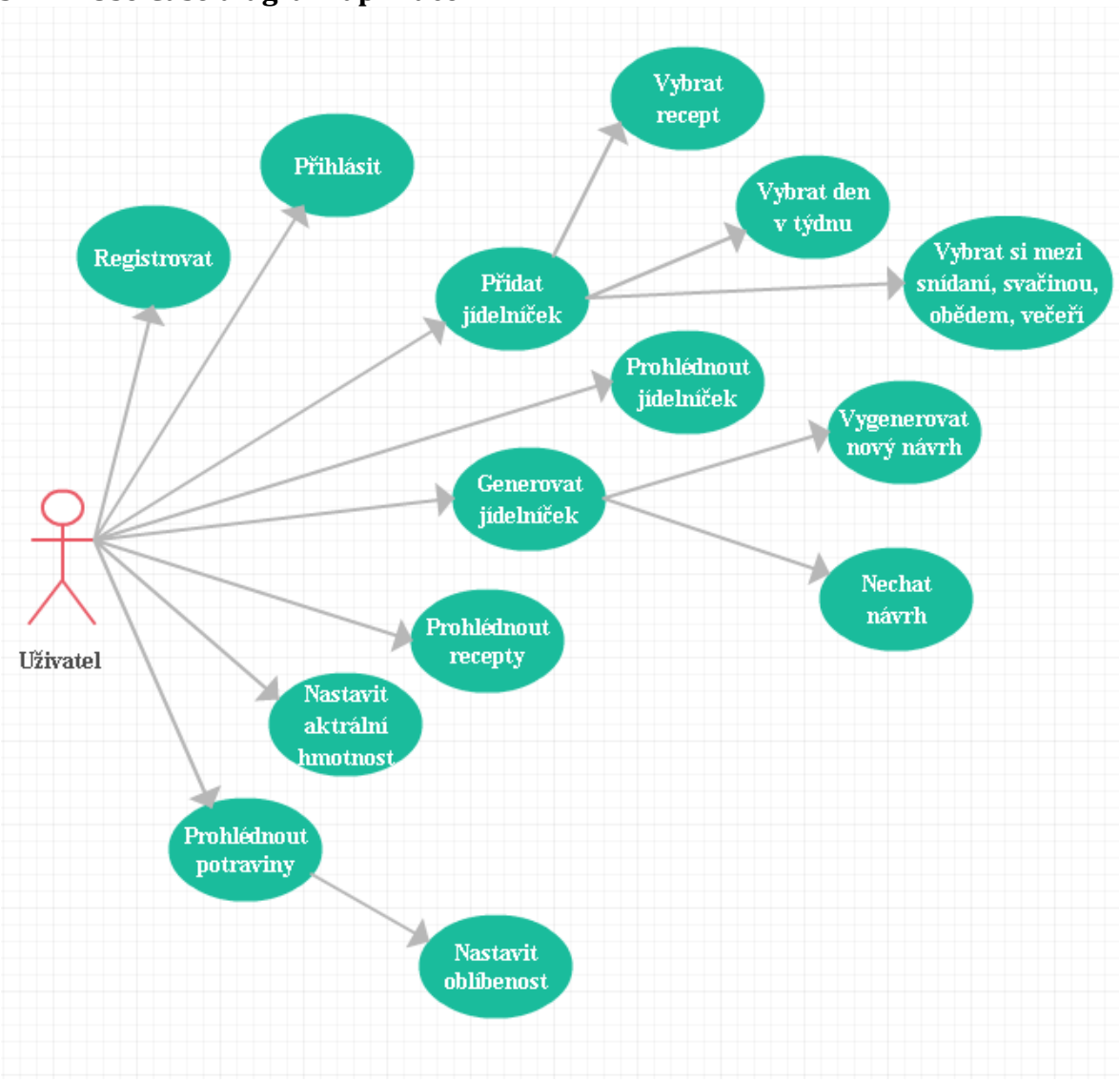
Zdlouhavá registrace, kterou MyFitnessPal disponuje, není příliš uspokojující stejně jako to, že se nelze registrovat bez přístupu na internet. Podobně jako registrace, tak i hledání receptů nefunguje bez připojení k internetu. Nemožná jsou i komerční sdělení, která se snaží znepříjemňovat jinak hladký chod aplikace. Absence české lokalizace, v poslední řadě, nejspíš rovněž zamrzí.

3 Analytická část

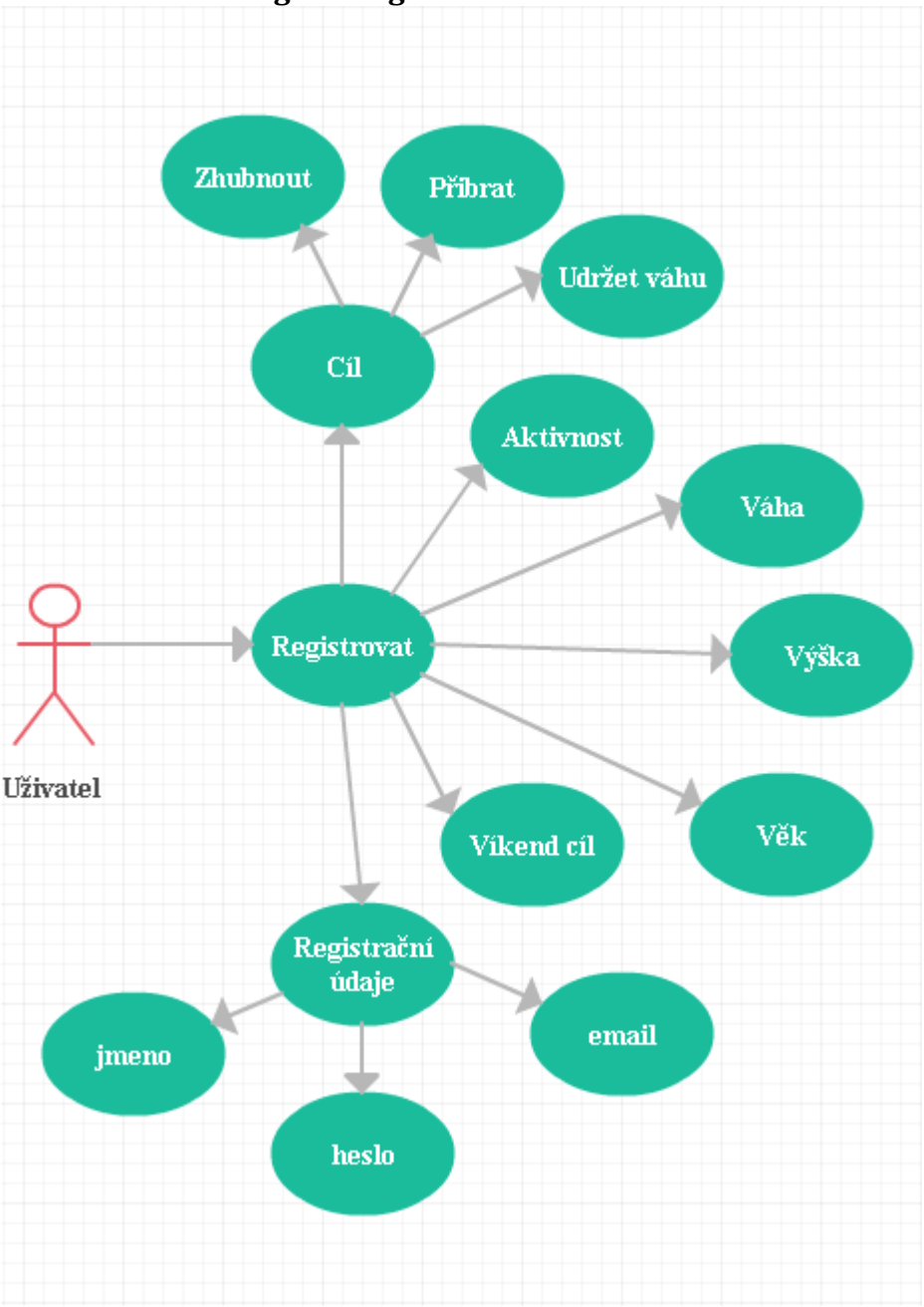
3.1 Požadavky[12]

Podle autora (Arlow, 2008, s. 112) je nejvhodnější modelovat případy užití, pokud má systém mnoho uživatelů, rozhraní a převládají funkční požadavky. Mobilní aplikaci může využívat v drtivé většině jeden nebo v některých případech i více uživatelů. Ačkoliv díky většímu množství funkčních požadavků je dobré, pro přehlednost, je zaznamenat. V tomto případě je zde aktérem uživatel mobilního zařízení. Dále v pododdílech jsou popsány jednotlivé požadavky, případy užití, které spouští aktér.

3.1.1 Use Case diagram aplikace



3.1.2 Use Case diagram registrace



3.1.3 Přidání jídelníčku

Začíná, když uživatel zvolí „Přidat jídelníček“. Systém zobrazí aktivitu s výběrem chodů, do kterého chce uživatel zadat recept.

Po kliknutí na „snídaně“, systém zobrazí formulář, ve kterém budou pouze recepty typu snídaně. Systém očekává recept, který uživatel zaškrtne ve formuláři. Systém zkontroluje správnost zadaných hodnot ve formuláři. Pokud jsou hodnoty špatné, zavře se formulář a nic se nestane, jinak systém recept správně uloží do zadaného chodu jídelníčku podle předchozího zadání a potom zavře formulář.

V případě že uživatel volí „oběd“, systém zobrazí formulář, ve kterém budou pouze recepty typu oběd. Systém očekává recept, který uživatel zaškrtne ve formuláři. Systém zkontroluje správnost zadaných hodnot ve formuláři. Pokud jsou hodnoty špatné, zavře se

formulář a nic se nestane, jinak systém recept správně uloží do zadaného chodu jídelníčku dle předchozího zadání a poté zavře formulář.

Pokud se uživatel rozhodne pro „večeře“, systém zobrazí formulář, ve kterém budou pouze recepty typu večeře. Systém očekává recept, který uživatel zaškrtne ve formuláři. Systém zkontroluje správnost zadaných hodnot ve formuláři. Pokud jsou hodnoty špatné, zavře se formulář a nic se nestane, jinak systém recept správně uloží do zadaného chodu jídelníčku podle předchozího zadání a potom zavře formulář.

Poslední případ, pro který se uživatel může rozhodnout je „svačina“. Systém zobrazí formulář, ve kterém budou pouze recepty typu svačina. Systém očekává recept, který uživatel zaškrtne ve formuláři. Systém zkontroluje správnost zadaných hodnot ve formuláři. Pokud jsou hodnoty špatné, zavře se formulář a nic se nestane, jinak systém recept uloží do zadaného chodu jídelníčku, podle předchozího zadání, a potom zavře formulář.

3.1.4 Generování jídelníčku

Požadavek je vyvolán, když uživatel zvolí v nabídce „Generovat jídelníček“. Systém zobrazí aktivitu s výběrem, na kolik dní počínaje dnešním si uživatel přeje jídelníček vygenerovat. Systém si zapamatuje počet dní, pro které má jídelníček vygenerovat. Potom začne prvním dnem.

Systém vybere postupně všechny recepty typu snídaně z databáze a uloží si je do seznamu. Později bude postupně číst ze seznamu receptu a zjišťovat, jakou energetickou hodnotu recept má. Pokud je energeticky hodnota receptu větší nebo rovna 20% cílového energetického výdaje uživatele, ale zároveň menší nebo rovna 25% cílového energetického výdaje uživatele, recept zůstane v seznamu, jinak je smazán. Systém opakuje tuto podmínku, dokud seznam není prázdný, nebo dokud se nedostane na konec seznamu. Systém ze zbývajících receptů v seznamu vybere zcela náhodně recept a uloží jej do jídelníčku v určitý den na místo jídla – snídaně.

Systém vybere postupně všechny recepty typu oběd z databáze a uloží si je do seznamu. Později bude postupně číst ze seznamu receptů a zjišťovat, jakou energetickou hodnotu recept má. Pokud je energeticky hodnota receptu větší nebo rovna 25% cílového energetického výdaje uživatele, ale zároveň menší nebo rovna 30% cílového energetického výdaje uživatele, recept zůstane v seznamu, jinak je smazán. Systém opakuje tuto podmínku, dokud seznam není prázdný nebo dokud se nedostane na konec seznamu. Systém ze zbývajících receptů v seznamu vybere zcela náhodně recept a uloží jej do jídelníčku v určitý den na místo jídla – oběd.

Systém vybere postupně všechny recepty typu večeře z databáze a uloží si je do seznamu. Později bude postupně číst ze seznamu receptů a zjišťovat, jakou energetickou hodnotu recept má. Pokud je energeticky hodnota receptu větší nebo rovna 20% cílového energetického výdaje uživatele, ale zároveň menší nebo rovna 25% cílového energetického výdaje uživatele, recept zůstane v seznamu, jinak je smazán. Systém opakuje tuto podmínku, dokud seznam není prázdný nebo dokud se nedostane na konec seznamu. Systém ze zbývajících receptů v seznamu vybere zcela náhodně recept a uloží jej do jídelníčku v určitý den na místo jídla – večeře.

Systém vybere postupně všechny recepty typu svačina z databáze a uloží si je do seznamu. Později bude postupně číst ze seznamu receptů a zjišťovat, jakou energetickou hodnotu

recept má. Pokud je energeticky hodnota receptu větší nebo rovna 5% cílového energetického výdaje uživatele, ale zároveň menší nebo rovna 10% cílového energetického výdaje uživatele, recept zůstane v seznamu, jinak je smazán. Systém opakuje tuto podmínku, dokud seznam není prázdný nebo dokud se nedostane na konec seznamu. Systém ze zbývajících receptů v seznamu vybere zcela náhodně recept a uloží jej do jídelníčku v určitý den na místo jídla – svačina.

Pokud uživatel zadal více než jeden den, proces se celý opakuje od vybrání snídaně až po svačinu do chvíle, než bude jídelníček s požadovaným počtem dní hotový.

3.1.5 Prohlížení jídelníčku

Požadavek uživatel vyvolá stisknutím tlačítka, „Prohlédnout jídelníček“. Systém vybere uživatelovy jídelníčky a zobrazí je v nové aktivitě. Pokud uživatel nemá vytvořené jídelníčky, požadavek skončí chybovou hláškou.

3.1.6 Nastavení aktuální hmotnosti uživatele

Uživatel zadá požadavek ke spuštění aktivity. Systém očekává číslo představující aktuální tělesnou hmotnost uživatele. Pokud systém nedostane správně zadané číslo, aktivita skončí.

3.1.7 Prohlédni recepty

Požadavek je spuštěn uživatelem stisknutím tlačítka, „Prohlédnout recepty“. Systém vybere recepty z databáze a zobrazí je v nové aktivitě.

3.1.8 Prohlédni potraviny

Případ užití nastává po výběru položky z menu. Systém vybere potraviny z databáze a zobrazí je v nové aktivitě. Pokud uživatel nemá vytvořené jídelníčky, požadavek skončí chybovou hláškou.

3.1.9 Nastav oblíbené potraviny

Systém nastaví oblíbené potraviny uživatele - po vyvolání položky z menu. Systém zobrazí seznam potravin, které dle výběru zaškrtne. Pokud je některá potravina zaškrtnuta, uloží se do databáze k oblíbeným potravinám, jinak aktivita skončí a vrátí se nazpět do navigačního menu.

3.1.10 Přihlásit se

Uživatel zadá přihlašovací údaje. Systém ověří údaje uživatele. V případě, že jsou správně, přesměruje uživatele na navigační aktivitu. Jinak systém zobrazí chybovou zprávu o nesprávně zadaných údajích.

3.1.11 Registrovat se

Systém vytvoří nový účet uživatele. Případ je spuštěn tlačítkem „Registrovat“. Dokud jsou přihlašovací údaje neplatné, systém žádá o jejich správné vyplnění. Nakonec systém vytvoří účet.

3.1.12 Nefunkční požadavky

OS android je jedním z hlavních nefunkčních požadavků, i když nad tím uživatel ani nepřemýšlí, programátor musí vědět, na jaká zařízení či platformu bude muset programovat. To, čeho si uživatelé nejvíce všímají, je design, ten je podobně jako OS android, nefunkčním požadavkem. Jeho špatný návrh má za následek například málo uživatelů, protože aplikaci neumějí ovládat. To platí hlavně v případě, že jsou neintuitivně rozmístěny komponenty na jednotlivých aktivitách.

Programovací jazyk je nedílnou součástí aplikace, proto jazyk Java hodnotím jako dalšího zástupce nefunkčních požadavků. Uživatele většinou nezajímá provedení, v jakém jazyce bude program napsán. Pro programátora je to jedno z důležitých rozhodnutí. Uživatele zajímá pouze cílový výsledek, to, co uvidí na své obrazovce.

3.2 Fitness lifestyle aplikace

Aplikace je zamýšlená jak pro jednoho, tak i pro více uživatelů na jednom mobilním zařízení. Uživatel se přihlásí pod svými přihlašovacími údaji. Pokud uživatel nemá zřízený účet, může se zaregistrovat. V registraci je po uživateli požadován věk, váha, cíl váhy, e-mail, heslo, jméno, pohlaví, typ diety. Tyto skutečnosti umožní spočítat BMR metabolismu. BMR určuje, kolik tělo vydá energie v klidovém stavu. Po úspěšném přihlášení do aplikace nebo po registraci se uživateli zobrazí navigační menu.

První co uživatel na navigaci uvidí, bude menu s tlačítky, což bude představovat hlavní menu. Uživatel si může vybrat, jestli si chce potraviny prohlížet, zadat aktuální hmotnost potravin a jiné možnosti. Nejdůležitější funkcí vůbec je generování jídelníčků a také jejich prohlížení. Generování vybírá uživateli zcela náhodné recepty, které filtruje tak, aby se hodily k jeho dietě.

Aplikace by mohla fungovat jako kuchařka, jelikož obsahuje všechny náležitosti, které mají kuchařky, jako například: název receptu, seznam potřebných surovin, popis postupu a tak dál.

Největší předností je plná podpora offline režimu. Po stažení aplikace již není potřeba mít připojení k internetu.

Aplikace jednoduše zpracovaná, komponenty jsou rozmístěny co nejblíže k palcům pro snadnější ovládání. Design je dvou až tříbarevný, proto je naprosto nerušivý.

3.3 Odlišení od konkurence

Srovnání proběhne mezi aplikacemi popisované v kapitole 2 a aplikací podporující zdravý životní styl.

Moje aplikace se liší od konkurenčních aplikací v několika zásadních věcech. Aplikace nemá implementovaný krokoměr. I když se to zdá být užitečná funkcionalita, má poměrně nespolehlivé měřící údaje, zapříčiněné používáním akcelerometru k počítání kroků a ještě nepřesnějším přepočítáváním kroků na metry. Neobsahuje tak rozsáhlou databázi potravin (obsahuje cca 1400 řádků). Ručním plněním databáze se nedá vyrovnat databázi MyFitnessPal za celý život, proto moje aplikace obsahuje pouze běžně dostupné potraviny. To, že není aplikace propojena se sociálními sítěmi, je pro některé uživatele výhodou a pro jiné nedostatkem.

Na druhé straně výhodou je práce v režimu offline, kdy po stažení aplikace uživatelé nepotřebují k jejímu používání internet. Další výhodou je svobodná distribuce aplikace, takže je úplně zdarma. Nevyužívá žádných komerčních sdělení ani bannerů. Asi největším přínosem je elegantní generování jídelníčků i na celý týden. To je také ta vlastnost, kterou se aplikace zcela liší od ostatních.

4 Architektura aplikace[6] [9]

Úplně na začátku, před samotnou implementací aplikace, je potřeba navrhnout a vytvořit diagramy tříd v UML. Diagram tříd nabízí vynikající možnost vizualizace struktury. Může zaznamenat základní vhled do struktury systému a tím zodpovědět otázku každého vývojáře, který se snaží začít pracovat na novém projektu: „Kde mám začít?“. Jeden přehledný obrázek diagramu vydá za několik odstavců psaného slovního popisu. Proto je níže v této kapitole umístěn.

Nedílnou návrhovou součástí aplikace je i databázový model, protože dobře navržený model je prvním správným krokem při vývoji aplikace. Díky tomu se vytvoří databáze, která splňuje všechny požadavky. Ta má přístup k aktuálním a přesným informacím, které by měly jít co nejsnadněji měnit.

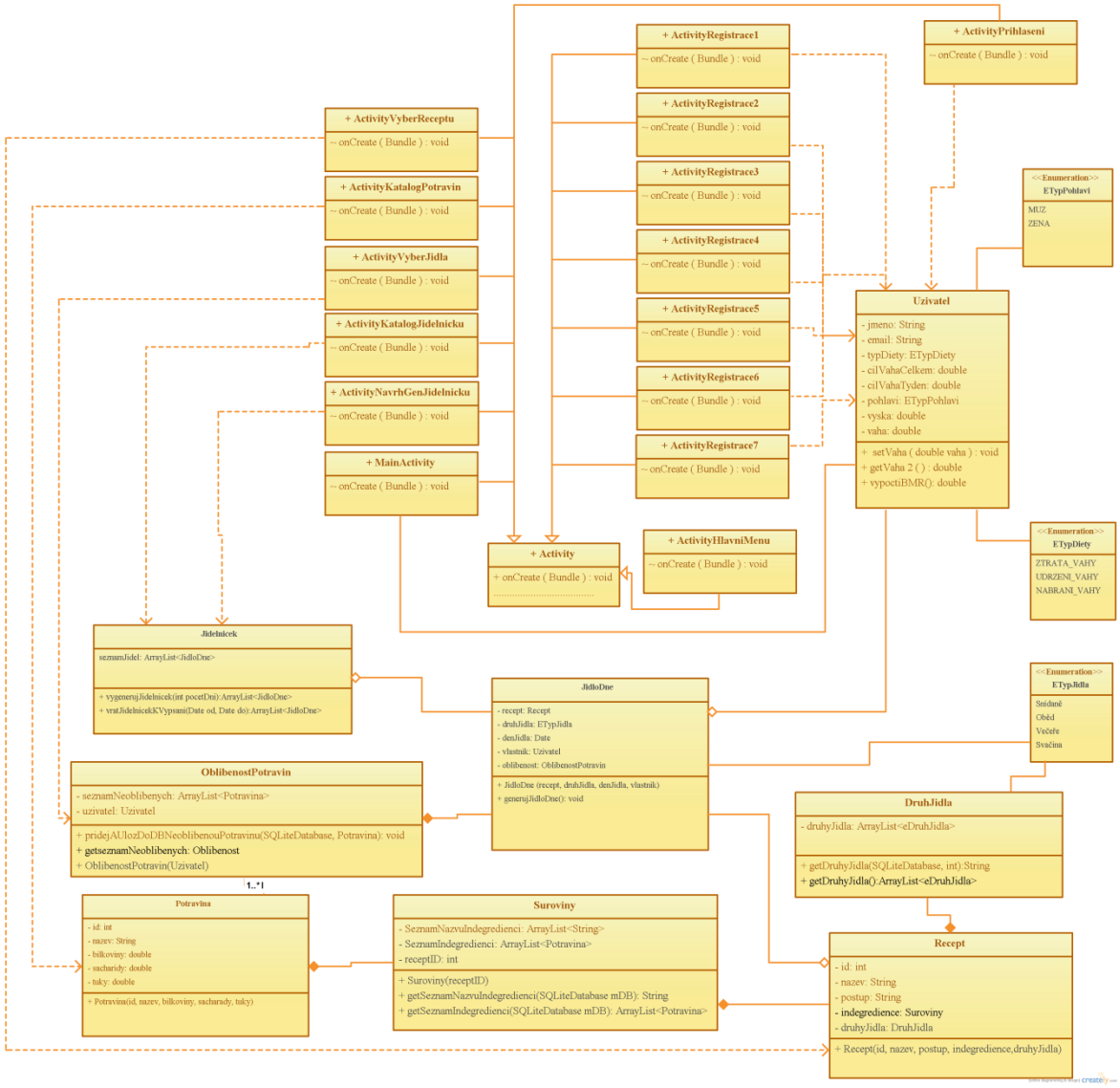
4.1 UML[6] [7]

Zkratka UML za sebou skrývá iniciály prvních písmen ve slovním spojení Unified Modeling Language. Ve volném překladu z anglického jazyka zkratka znamená unifikovaný modelovací jazyk. Na rozdíl od psaných programovacích jazyků zahrnuje grafickou syntaxi. To znamená, že má pravidla pro sestavování jednotlivých prvků jazyka do větších objektů. Vedle syntaxe UML obsahuje i grafickou sémantiku. To jsou jednoznačná pravidla říkající jednotlivým syntaktickým výrazům jejich význam.

Úplně největší nynější význam má jazyk při návrhu softwaru, protože v současné době se všechny jazyky zaměřují na objektově orientované programování (OOP). Objektově orientovaný návrh zcela každé komplikované aplikace je nutnou podmínkou pro její úspěšnou implementaci bez zbytečné ztráty času. Pro objektově orientovaný návrh není samozřejmě nezbytné používat UML, ale je možné použít jiné podpůrné prostředky, jako jiné, další typy diagramů. V UML je však podstatné, že přesně umí specifikovat, co má daný diagram obsahovat, což je velmi významné při sdílení informací s ostatními programátory. UML diagramů existuje hned několik typů, lišících se od sebe podle úlohy, která se zpracovává. Rozdíl je však pouze v řadě použitých značek, způsobu propojení entit.

Výhodou UML diagramů je existence otevřeného a rozšiřitelného standardu a univerzálnost použití na jakýkoliv objektově orientovaný jazyk. Pro taky mluví skutečnost, že je podporován celou řadou vývojových nástrojů určených specificky jen pro návrh UML a nebo integrovaným prostředím, které dokáže z kódu programovacího jazyka navrhnout diagram a i obráceně.

4.1.1 Diagram



4.2 DB[2] [11]

Databáze neboli datová základna je určitá uspořádaná množina informací, uložená na paměťovém médiu. Je určená k pojmutí i obrovského množství dat, ale to nevylučuje skutečnost, že existují i malé databáze.

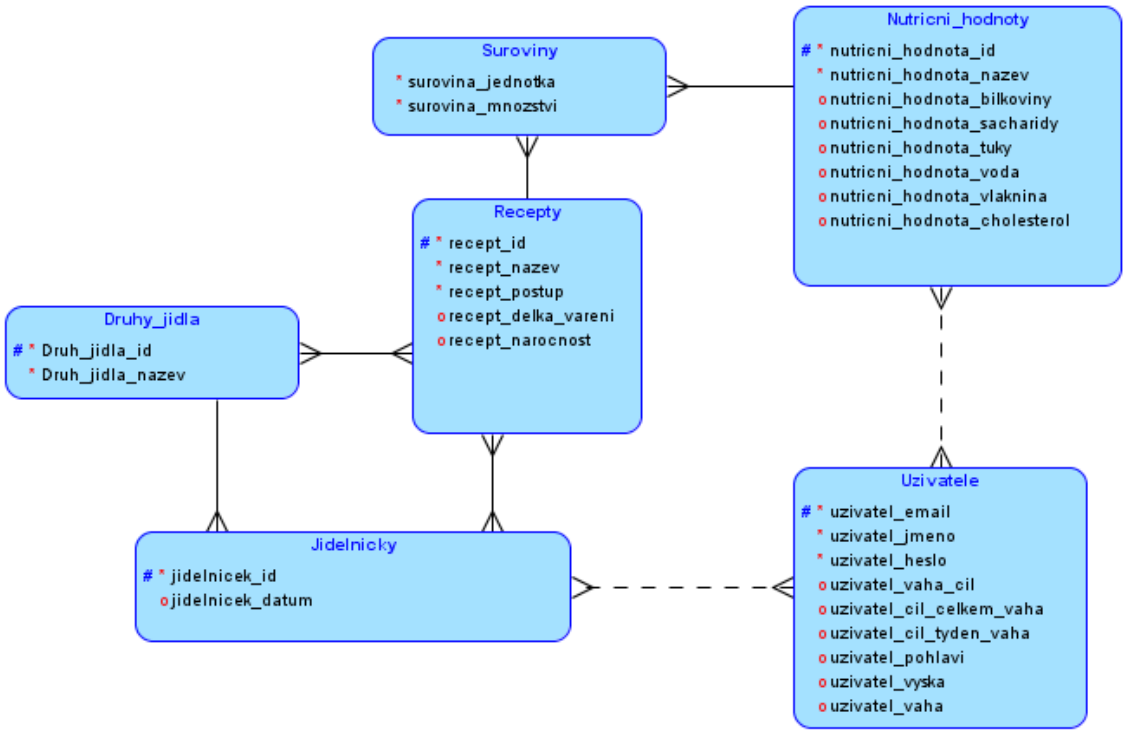
Na základě diagramů tříd UML šlo jednoznačně určit, jaké informace bude aplikace vyžadovat. Podle toho je žádoucí navrhnout konceptuální model. Ten se skládá z entit, instancí entit, atributů, relací a primární UID. Nejdříve je potřeba vytvořit entity a k nim přiřadit atributy a určit jejich primární UID. Nakonec stanovit relace mezi nimi.

Procesem návrhu databázového modelu se zabývá právě tato podkapitola.

4.2.1 Logický model

Z konceptuálního modelu se lze přesunout do prostředí relačních databází, konkrétně k logickému modelu. Ten čerpá veškeré informace pro logický návrh z ER modelu vytvořeného během konceptuálního modelování. Základem této fáze modelování je převod entit v tabulky, atribut ve sloupce a relace v cizí klíče. Strukturu každé takto vzniklé

tabulky je třeba pak zkontrolovat za pomoci normalizace, hlavně kvůli případné duplicitě sloupců. Logický model je třeba navrhovat před tím, než se ví, jaká databáze se bude používat.

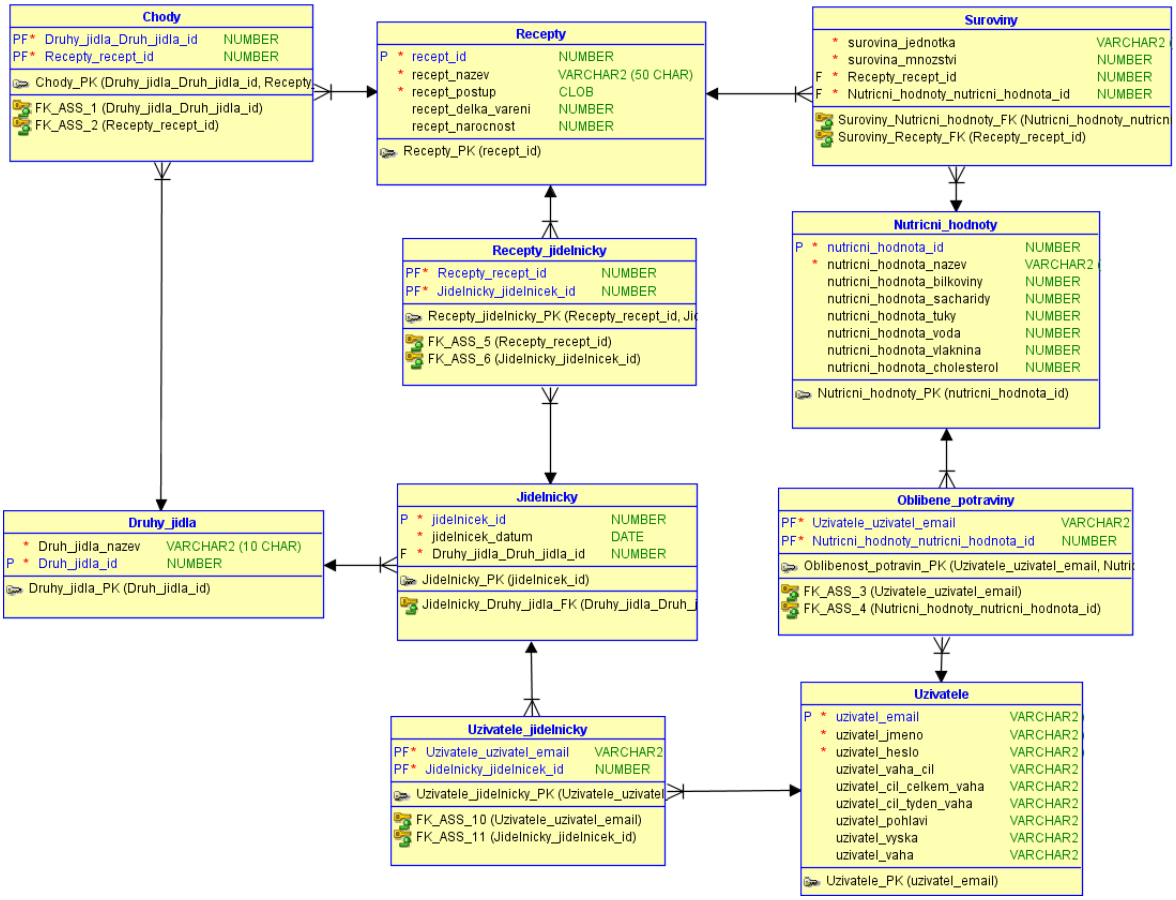


Obrázek 4 - Logický model mobilní aplikace

4.2.2 Relační model

Princip modelu dat je založen na teorii množin a predikátové logice prvního řádu. V relačním modelu jsou data přestavována za pomocí tabulek, jejich řádky reprezentují jednotlivé záznamy a sloupce jsou vlastnostmi záznamů. Relační model umožňuje vyjádřit všechny typy vztahů mezi záznamy (1 : 1,1:N, N: 1, N: M) a umožňuje data mezi sebou spojovat.

Výhodou relačního databázového modelu je flexibilita při vytváření vztahů mezi záznamy. Další předností by mohla být jednoduchost získávání dat nebo i snadná změna struktury databáze bez porušení integrity dat. Mezi nevýhody patřily, ještě před dvěma desetiletími, velké nároky na výkon počítače a kapacitu paměti. To je však v dnešní době, kdy běžný uživatel používá terabitové paměti a několika gigahertzové výpočetní jednotky, minulostí.



Obrázek 5 - Relační model mobilní aplikace

Výše uvedený obrázek představuje relační model aplikace. Všechny vazby mezi tabulkami jsou mandatory neboli povinné. To jasně značí nepřerušovaná plná čára. Všechny vazby také vyjadřují kardinalitu vztahu 1:N šipkou, kde rozvětvení u paty šipky znamená N vztah.

Tabulky UZIVATELE a OBLIBENE_POTRAVINY uchovávají údaje o uživateli a jejich oblíbených potravinách. V aplikaci se předpokládá, že uživatel si může časem nastavit svoje oblíbené potraviny. Je potřeba k vytváření oblíbené potraviny, které se budou brát v ohled, při generování jídelníčku, uchovávat je v samostatné tabulce OBLIBENE_POTRAVINY. Tím se docílí, že uživatel bude moci znát svoje oblíbené potraviny a nutriční hodnota společně i s názvem potraviny bude vědět, jakému uživateli patří. Zajímavostí je, že tabulka uživatel vlastní jediný primární klíč email, který není umělý.

Tabulka NUTRITRICNI_HODNOTY ukládá množství živin k jednotlivým surovinám. Na první pohled je vidět, že ne všechny atributy jsou povinné, je to zapříčeno právě z toho důvodu, že ze začátku tvorby aplikace nebylo zřetelné, které položky bude muset aplikace zpracovávat.

Tabulka SUROVINY představuje jednotlivé ingredience v receptu. Daný recept značí cizí klíč Recepty_recept_id. Právě při vaření podle receptu je nutné znát, z jakých všech surovin se bude vařit. Také v jakém množství bude surovina potřeba a v jaké bude jednotce. Například: „Mléko 100 ml“ nebo „Med 1 lžíce“. Aby aplikace dokázala spočítat,

jakou energetickou hodnotu dohromady recept má, je žádoucí znát jejich nutriční hodnoty z tabulky NUTRITRICNI_HODNOTY. Na ty se jednoznačně odkazuje pomocí cizího klíče Nutricni_hodnoty_nutricni_hodnota_id.

Konkrétní recept např. Banán s čokoládou lze najít v tabulce RECEPTY. Uživatel se díky této tabulce může dozvědět, jak recept uvařit, za jak dlouho nebo zdali je vaření těžké.

Tabulka CHODY Slouží jako spojovací tabulka mezi recepty a druhem jídla. Protože recept může být snídaní i večeří zároveň, tak v sobě uchovává dva cizí klíče. Recepty_recept_id a Druhy_jidla_druh_jidla_id.

DRUHY_JIDLA má za úkol informovat tabulku CHODY o tom, jestli je obědem, svačinou, snídaní, večeří. Zároveň to má povědět za pomocí id i tabulce JIDELNICKY.

Tabulka JIDELNICKY představuje jeden uživatelův chod v daný den. Zděděný cizí klíč Druhy_jidla_druh_jidla_id říká aplikaci, jakým druhem jídla je.

Klíčovou je spojovací tabulka RECEPTY_JIDELNICKY mezi JIDELNICKY a RECEPTY. Díky zděděným klíčům aplikace Recept_recept_id a Jidelnicky_jidelnicek_id přesně ví, jaký recept použije v jídelníčku. Při spojení obou tabulek aplikace kupříkladu vidí stravovací plán na určitý den.

UZIVATELE_JIDELNICKY rozlišuje, který jídelníček je kterého uživatele. To obsluhují dva zděděné klíče Jidelnicky_jidelnicek_id a Uzivatel_uzivatel_email;

4.3 Způsob napojení DB[10]

Pro uskladňování dat v databázi se nejčastěji používá databázový systém SQLite. Tuto oblíbenost si zasloužil nejen kvůli své jednoduchosti. Databáze nevyžaduje propojení se serverem, na rozdíl od ostatních databází založených na principu klient-server. To je právě hlavní důvod, proč se v drtivé většině mobilních aplikací používá právě SQLite. Aplikace by přestala fungovat pokaždé, kdyby neměla spojení se serverem, což je nepřípustné. Proto i tato aplikace podporující fitness lifestyle používá právě SQLite. Následující pododdíly pojednávají o relačním DB systému SQLite a způsobu napojení databáze.

4.3.1 SQLite

Databázový systém SQLite je jeden ze zástupců klasických databází využívající dialekt jazyka SQL. Dialekt používá standard SQL-92. Určité změny, lépe řečeno odlišnosti či vylepšení, vykazuje. Například RIGHT OUTER JOIN ani FULL OUTHER JOIN, neexistují. Pro spojení jedné a více tabulek, vývojáři nechali pouze dotazy LEFT OUTHER JOIN a INNER JOIN. Datový typ DATE, který obsahuje pouze datum, byl nahrazen DATETIME, který obohacuje navíc datum i o čas. Příkaz ALTER TABLE slouží pouze pro manipulaci se strukturou tabulky. Manipulace za pomocí tohoto příkazu, umožňuje již existujícím tabulkám přejmenovat tabulku za pomocí příkazu RENAME TO anebo přidat do tabulky celý sloupec ADD COLUMN. Poslední bod výčtu je významný, odlišuje se totiž od standardu SQL. Neexistuje zde kontrola datových typů při vkládání dat do tabulek. Prakticky tohle znamená, že je zcela možné vložit do sloupce typu STRING hodnotu typu INTEGER. Výjimku tvoří pouze sloupec INTEGER PRIMARY KEY, do kterého leze vložit pouze celočíselné hodnoty jako je to v jiných jazycích obvyklé.

Prostřednictvím SQLite lze využívat dotazy i skupinové dotazy pro výběr dat příkazem SELECT. Pro editaci již existujících dat v tabulkách příkaz UPDATE. Mazání dat

z tabulek příkaz DELETE. Aby se vytvořila tabulka, musí se zapsat do dotazu příkaz CREATE TABLE. SQLite nabízí zcela plnohodnotné prostředí pro práci s daty, včetně využívání takových základních prostředků jako jsou transakce a triggery.

4.3.2 Tvorba databáze

Pokud chce programátor používat v operačním systému Android SQLite databázi, musí ji vlastnoručně vyrobit. To lze dvěma způsoby. První způsob je použít oficiální distribuci aplikace od vývojářů databázového systému SQLite3 terminál nebo jiného softwarového prostředku, který dovoluje provést SQL dotazy pro práci s daty. Druhým postupem jak si vytvořit nebo případně aktualizovat databázi je tvorba Android aplikace, která v sobě bude obsahovat potomka třídy SQLiteOpenHelper. Ten zajišťuje celou logiku pro vytvoření a aktualizaci databáze. Po vytvoření vlastní třídy, která dědí z výše zmíněné třídy, je nutné implementovat tyto metody:

- onCreate() – metoda slouží právě pro vytvoření libovolného počtu tabulek v jedné databázi a jejich naplnění daty. Vykonání metody proběhne za pomoci instance třídy SQLiteDatabase.

- onUpgrade() – metoda slouží pro úpravu struktury databáze a tabulek. Za pomocí této metody je předán opět objekt SQLiteDatabase a staré číslo verze databáze a nové číslo verze databáze. Pokud je číslo nové verze databáze jiné, vyšší, než hodnota staré verze, provede se změna struktury právě za pomoci této metody.

Dále konstruktor potomka třídy SQLiteOpenHelper musí provést volání konstruktoru jeho rodiče prostřednictvím klíčového slova super. Parametry rodičova konstruktoru jsou v přesně daném pořadí:

- instance třídy Context, kterou má každý potomek třídy Activity,

- název databáze, který musí být deklarován jako String,

- instance třídy CursorFactory, která je používaná k vracení instance třídy Cursor v případě, kdy je volán SQL dotaz,

- celočíselná hodnota typu int představující verzi databáze.

Implementaci konstruktoru a zavoláním jej, za prostřednictví instance třídy potomka SQLiteOpenHelper, databáze nevznikne. Ta vznikne až po zavolání jedné ze zděděných metod od rodiče. Obě dvě metody vrací instance třídy SQLiteDatabase právě vytvořené databáze. Vyvoláním metody getReadableDatabase() umožní prohlížení databáze. Při volání druhé metody getWriteableDatabase() zpřístupní databázi určenou k editaci. Programátor si může vybrat režim databáze, se kterým bude pracovat.

Samotná databáze tvoří obal pro jednotlivé tabulky, které obsahují definovaná data. Tvorba tabulek se ve většině případů bude vytvářet výše popsanou metodou onCreate(). Ta bude muset používat instanci třídy SQLiteDatabase a volání její metody execSQL(String SQL). Do parametru této metody se vyplní String, který bude obsahovat SQLite dotazy, které jen manipulují s daty. Pro upřesnění dotaz obsahující jen příkaz SELECT nelze v metodě execSQL() provést.

Databáze se dá tímto způsobem tedy vytvořit. Co není úplně zřejmé, že se tímto způsobem nechá otevřít již existující soubor s databází. Pro uložení souboru s databází je nutno jej

umístit do specificky umístěné složky, která se dá vygenerovat v prostředí android studia klepnutím pravým tlačítkem myši na ikonu složky app těsně pod Android projektem. Po stisknutí tlačítka myši se zobrazí nabídka. Postup nabídkou je následující:

1. new
2. Folder
3. Assets Folder

Do této vygenerované složky Assets se umístí soubor s předem připravenou databází. Při nastavování konstruktoru instance SQLiteOpenHelper se musí nastavovaný název shodovat s názvem souboru včetně koncovky. Pro nastavení je nutné znát verze SDK. Pro verzi 17 a vyšší je adresář s daty umístěn přímo v adresáři s aplikací. Pro nižší verze se nalézá v /data/data/ adresáři. Tuto problematiku názorně popisuje kapitola 5.3.2

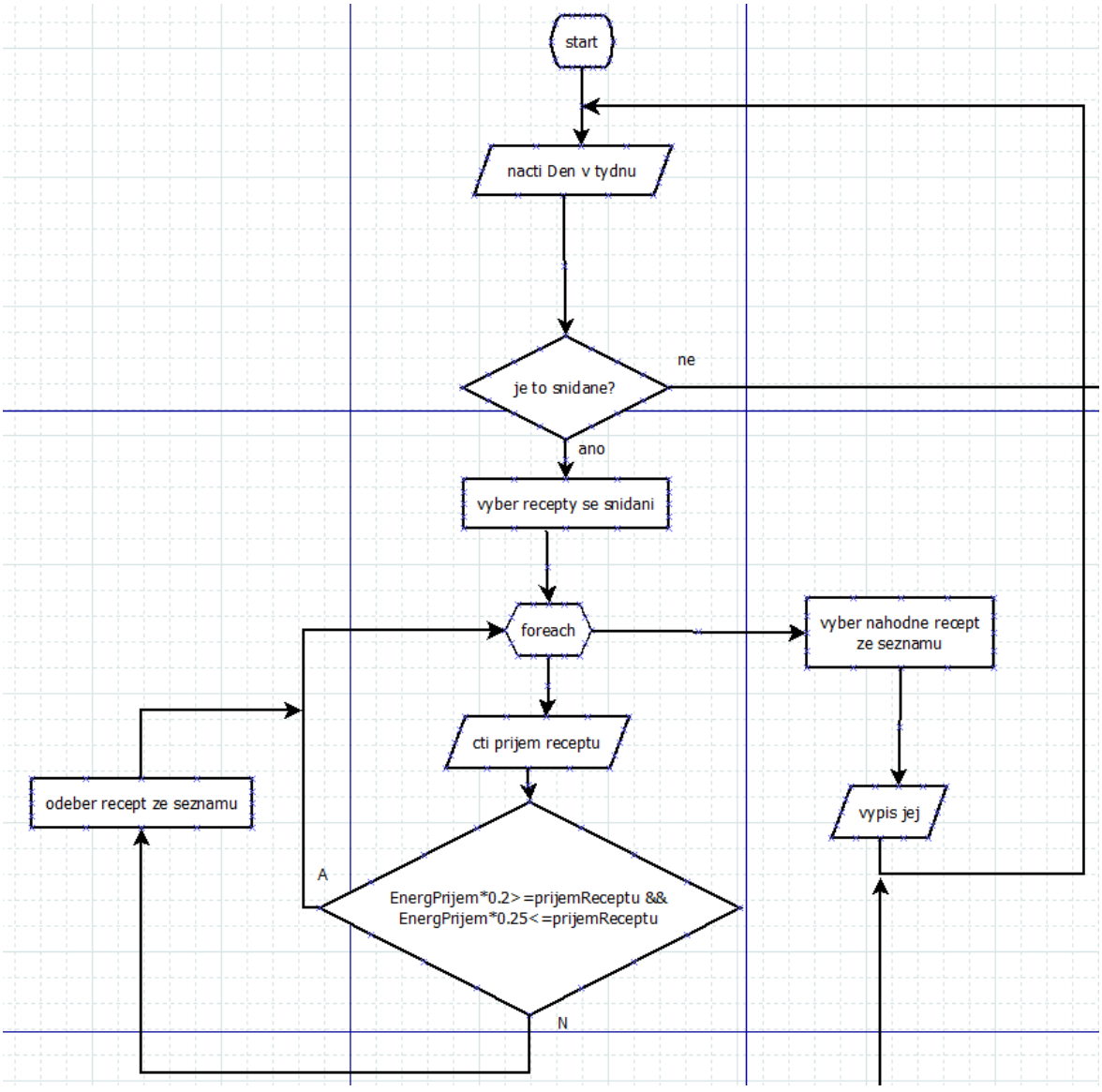
5 Vlastní řešení, implementace

V této kapitole jsou popsány nejdůležitější části tvorby mobilní aplikace. Nejprve jsou rozebrány hlavní algoritmy, bez kterých by tato aplikace nemohla vzniknout. Později softwary, bez kterých by se nebylo možné obejít při vývoji. Kapitolu zakončují klíčové problematiky aplikace.

5.1 Ukázky algoritmů

Algoritmy jsou přesné postupy, kterými lze danou úlohu vyřešit. S pojmem se setkáváme i v každodenním životě, když si chceme namazat housku máslem, vyprat prádlo nebo uvařit jídlo. U programování tomu není jinak. Proto tato podkapitola se bude jimi zabývat.

5.1.1 Algoritmus pro návrh individuálního stravovacího plánu



Obrázek 6 - Algoritmus pro generování jídelníčku

5.1.2 Algoritmus pro výpočet BMR

Jedním z důvodů proč se může člověk pohybovat, je, že má k tomu jistou energii. Neboli pokud člověk nepřijme z jakéhokoliv zdroje energii, přestane se hýbat, bude mrtvý. Takže je životně důležité, aby každý jednotlivec přijímal energii. Tu si tělo umí samo vyrobit konzumací jídla. Ale kolik každý potřebuje na den energie? Ideálně tolik energie, kolik za den stihne spotřebovat. Určení této hodnoty není jednoduché, každý je jedinečný a má jinou spotřebu. Právě k tomu slouží BMR. Pokud chce člověk zachovat váhu, musí příjem být roven výdeji energie. Hubnout člověk začne v případě, že má větší výdej než příjem energie.

Za zkratkou BMR se skrývá Basal Metabolic Rate, volně z anglického jazyka přeloženo jako Basální metabolická hodnota. Jednou z možností jak zjistit tuto ideální výživovou hodnotu v klidovém stavu, je výpočet basálního metabolického hodnocení. To představuje množství energie vydané v klidovém stavu v teplotně neutrálním prostředí na lačno. Nejpřesnější je tato hodnota pro člověka, který má průměrnou stavbu těla.

Na základě této hodnoty se tedy určí, kolik energie by měl uživatel v potravinách za den přijmout. Pokud uživatel například nepracuje v kanceláři, ale vykonává nějaké fyzicky náročné povolání, měl by si tuto hodnotu zvýšit. Nebo i například při plánovaném sportu je třeba BMR zvýšit.

Jedna z osob, která se tímto vztahem zabývá je Mifflin St Jeor. Ten vymyslel vztah, který má oproti ostatním malou odchylku 5%. Tento vzorec je založen na hmotnosti, výšce, pohlaví a věku jedince.

Pro výpočet se použije vzorec, který je podle studie americké dietologické asociace publikované v roce 2005 podle porovnání různých rovnic nejpřesnější.

$$10 * váha + 6{,}25 * výška - 5 * věk = BMR\ [kcal]$$

U mužů a žen se vzorec liší, pouze v přičtené konstantě. K BMR mužů se přičte číslo 5 a k BMR žen přičte číslo -161.

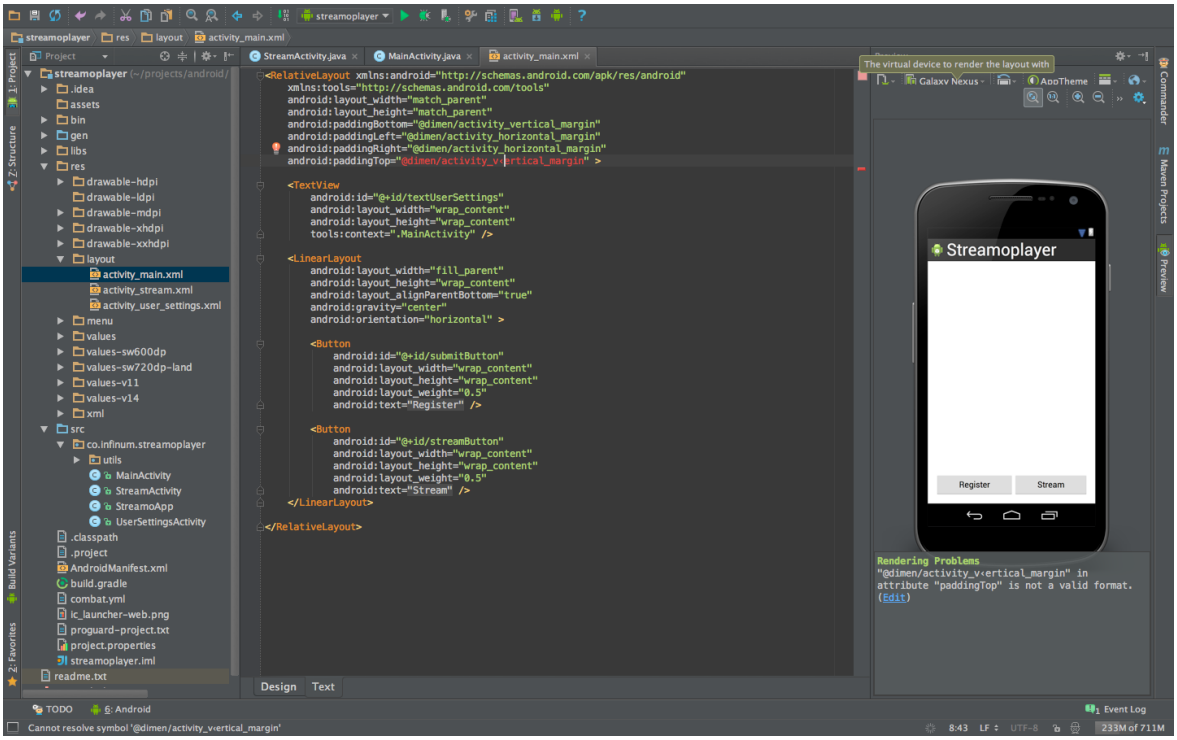
5.2 Použité softwary a vývojové prostředí

Úplně nejzásadnější výhodou používání IDE, a jiných softwarů je úspora času a energie. Programování v příkazovém řádku nebo v textovém editoru už jsou dávno minulostí. I když papír a tužka je stále nejspolehlivějším nástrojem programátora. V tomto oddíle se budu zabývat softwary, které jsem si k vývoji aplikace vybral.

5.2.1 Android Studio

Android studio je vývojové prostředí podporující programování pro OS Android. Je to vlastně jedno ze dvou možností IDE, kterou si lze na programování Androidů vybrat. Další variantou je Eclipse od Eclipse Foundation, které jsem si nevybral. Zajímalo mě, jak se s vývojovým prostředím umí vypořádat gigant jako je Google. Ze začátku jsem nebyl nadšený, ale po chvíli jsem si zvyknul. Rozhodně aplikace umí našeptávat, návrh grafického prostředí je, jednoduše řečeno, na výbornou.

Instalace Android studia, oproti Eclipse, by měla být jednoduchá. Stačí si stáhnout pro svůj počítač instalační balíček pro svůj systém a JDK. Je požadována originální Java od Sunu.



Obrázek 7 - Ukázka vývojového prostředí[3]

5.2.2 SQLite Expert Pro

SQLite expert pro je zajímavé řešení aplikace pro správu databázových systému SQLite.

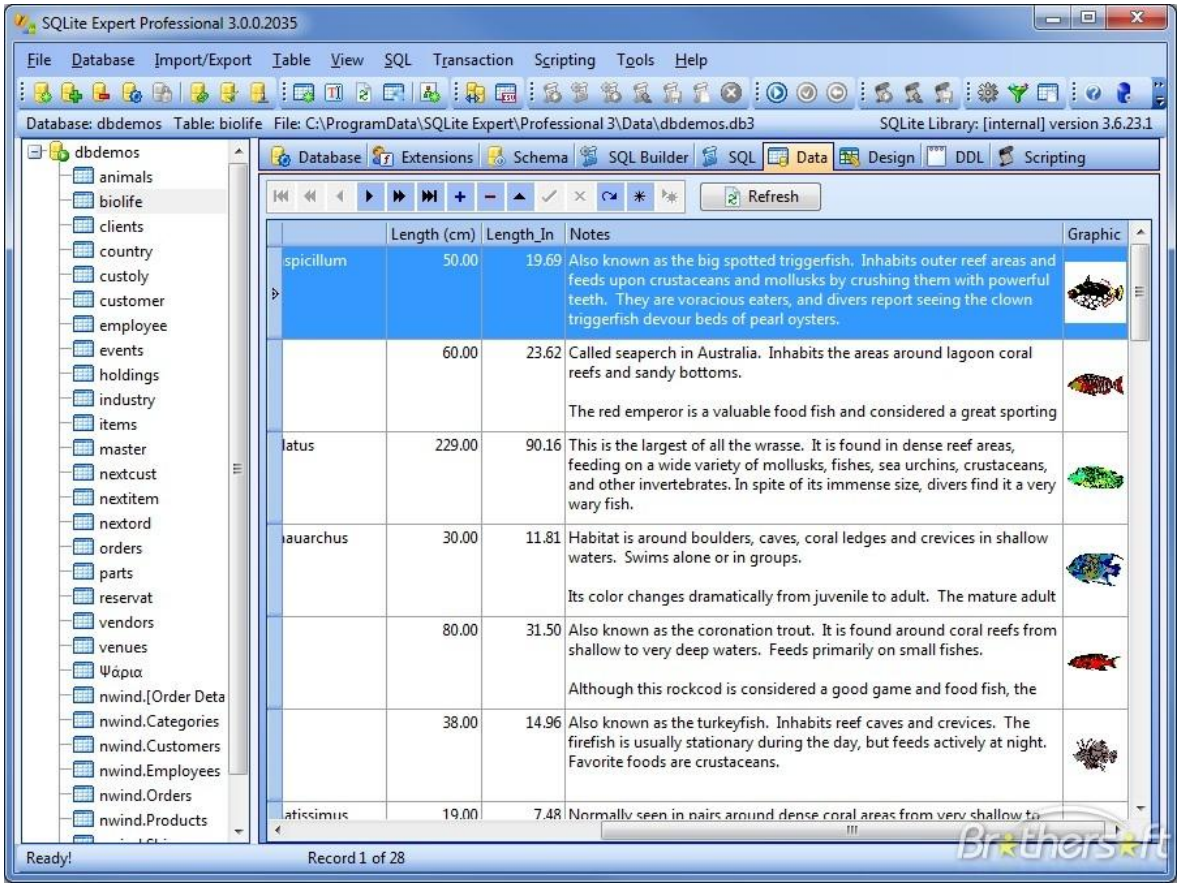
Co všechno aplikace umí?

- Otevřít .sqlite a .dll soubory s SQLite databází,
- vytvářet a plnit nové tabulky i stávající,
- spouštět jakékoliv SQL dotaz,
- procházet data, včetně podpory zobrazování obrázků a filtrování.

Má krásné uživatelské rozhraní s dostatečně velkými tlačítky a fonty. Je perfektní pro nováčky nebo lidi, kteří o SQL moc nevědí.

Jaké má limitace?

- Importování nebo exportování přes kopírovací schránku,
- občasné grafické chyby při konstrukci SQL dotazů,
- jisté základní funkce jsou z důvodu, že je licence zdarma, blokované (v placené verzi fungují bez problému).

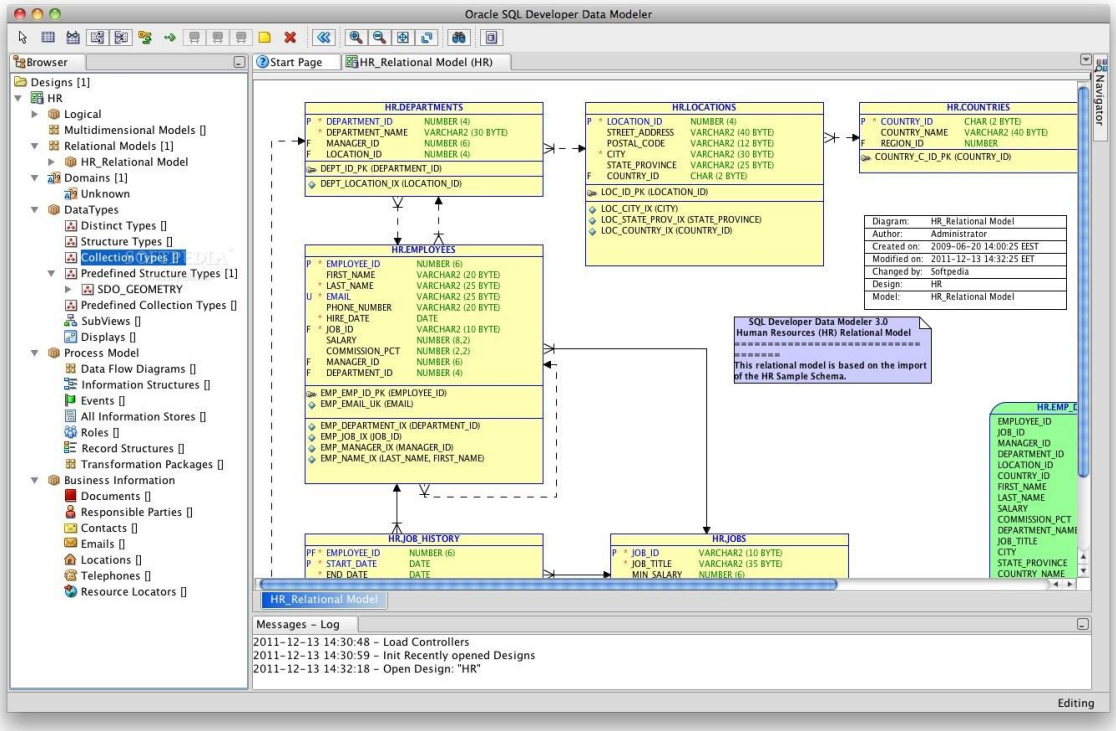


Obrázek 8 - Ukázka SQLite Expert Pro[4]

5.2.3 Oracle data modeler

Oracle data modeler je grafický nástroj pro tvorbu a návrh databázových modelů. Je to nástroj, který je zdarma a umí zlepšit produktivitu a zjednodušuje práci při veškerém modelování dat. Používáním uživatelé mohou vytvářet, procházet a editovat logické a relační modely. Po vytvoření relačního modelu si uživatel může jednoduše vygenerovat DDL dotazy, které mu pomohou vytvořit databázi fyzicky.

Uživatelé mají možnost vytvářet entity, atributy a přiřazovat jim název. Další funkcí, kterou program zahrnuje, je propojování entit a určení vztahu mezi nimi. Funkce engeneer to relation model, která umí z logického modelu vytvořit relační. Obsahuje i funkci, která převede opačně z relačního do logického modelu. Tisk modelů do formátu .pdf, je další užitečnou funkcí, kterou data modeler má.



Obrázek 9 - Ukázka Oracle data modeler[5]

5.3 Ukázka vlastního řešení

V tomto pododdíle je vysvětleno, jak se postupovalo při nastavování manifestu, co to jsou aktivity a jak se s nimi pracuje.

5.3.1 Manifest

Základem každé aplikace pro Android je soubor AndroidManifest.xml. Jak dobře popisuje autor (Murphy, 2011, s. 29-34), generuje se automaticky při založení nového projektu. Nachází se v kořeni projektového adresáře. Jsou v něm přihlášeny aktivity, služby, nastavují se různá povolení, hlavní aktivita projektu, kterou uvidí uživatel při spuštění aplikace jako první.

Kořenem tohoto souboru je níže uvedený element. Do atributu package se přiřazuje název balíčku, tím se definuje základ aplikace. Je užitečné, že vždy, když je třeba uložit v manifestu nějakou třídu, stačí pak nahradit balíček tečkou. Takže místo specifikace třídy com.example.hash.MainActivity se píše pouze .MainActivity.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
      xmlns:android="http://schemas.android.com/apk/res/android"
      package="hash.cookfit"
      android:versionCode="1"
      android:versionName="1.0" >
</manifest>
```

Také není špatné nastavit element <uses-sdk>. Povolí spustit aplikaci na zařízení s předem definovanou verzí Androidu. Používá pro to dva atributy minSdkVersion a targetSdkVersion. První atribut požaduje i google play, aby mohla být v obchodě vidět.

minSdkVersion znamená pro jakou verzi minimálně je aplikace určena. Druhy atribut určuje opak.

```
<uses-sdk
android:minSdkVersion="4"
android:targetSdkVersion="16" />
```

Element <appliaction> definuje obsah aplikace. Jeho atribut icon určuje základní ikonu pro všechny aktivity. Popisek aplikace se nastavuje atributem label.

V manifestu se musejí napsat všechny používané aktivity. Bez toho by program skončil chybou. Její přihlášení se dělá pomocí atributu <aktivity>. Do atributu name se napíše název aktivity, label vloží popisek.

```
<application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <activity
android:name=".data.ExpandableListMainActivity"></activity>
    </application>
```

5.3.2 Implementace tříd pracujících s databází

Mobilní aplikace využívá databáze SQLite. Aby mohla přistupovat k jednotlivým tabulkám této databáze, museli být vytvořeny speciální třídy. Ty obsahují instanci třídy DatabaseHelper, která je potomkem třídy SQLiteOpenHelper. Jedná se o pomocnou třídu starající se o otevírání databáze, pokud existuje. Pokud DB neexistuje, tak jí vytváří a pokud se vyskytne nová verze databáze, provede její aktualizaci. Databáze v aplikaci se neotevře nebo nevytvoří, dokud se nezavolá jedna z metod getReadableDatabase() nebo getWritableDatabase().

```
public class DatabaseHelper extends SQLiteOpenHelper {
…
@Override
public void onOpen(SQLiteDatabase db) {
super.onOpen(db);
}
}
```

Po vytvoření referenční proměnné této třídy se zavolá konstruktor. U starších verzí Androidu je nutné do cesty k souboru je nutné přidat před adresáře s aplikací přidat navíc cestu data/data/. Pro lepší názornost si dovolím ukázat otevření souboru s databází na následujícím příkladu:

```
String DB_NAME ="company.db";
SQLiteDatabase mDataBase;
public DataBaseHelper(Context context){
        super(context, DB_NAME, null, 1);
        if(android.os.Build.VERSION.SDK_INT >= 17){
            DB_PATH = context.getApplicationInfo().dataDir +
"/databases/";
        }
        else
        {
            DB_PATH = "/data/data/" + context.getPackageName() +
"/databases/";
        }
        this.mContext = context;
    }

public boolean openDataBase() throws SQLException
    {
        String mPath = DB_PATH + DB_NAME;
        mDataBase = SQLiteDatabase.openDatabase(mPath, null,
SQLiteDatabase.CREATE_IF_NECESSARY);
                return mDataBase != null;
    }
```

Následující metoda představuje načtení receptu. Dotaz z databáze nejprve spojí tabulku receptů společně s pomocnou tabulkou recepty_jidelnicky a napojí se na tabulku s jídelníčky. Filtrování probíhá na databázové úrovni za pomocí klausule where. Filtr používá hodnotu parametru jidelnicekID k nalezení receptu který jídelníček obsahuje.

```
SELECT
recept_id,
recept_nazev,
recept_delka_vareni,
recept_narocnost,
jidelnicek_datum,
druhy_jidla_druh_jidla_id ,
jidelnicek_id
FROM recepty
left join recepty_jidelnicky on recepty.recept_id =
recepty_jidelnicky.recepty_recept_id
left join jidelnicky on recepty_jidelnicky.jidelnicky_jidelnicek_id =
jidelnicky.jidelnicek_id
where jidelnicek_id= jidelnicekID;
```

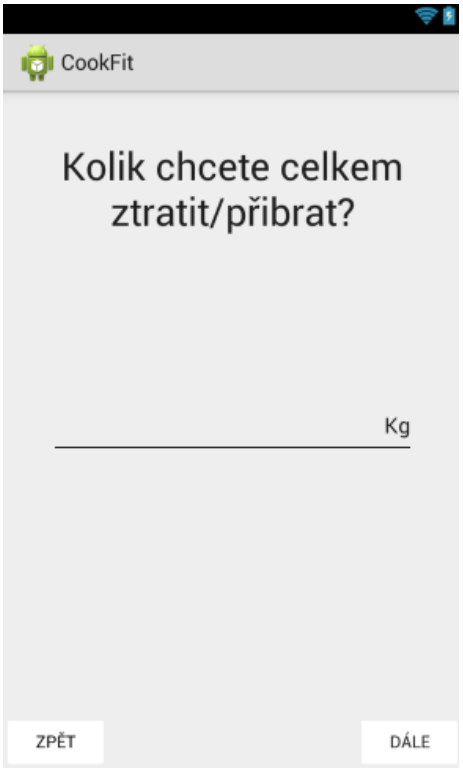
Cursor je třída která je nejvíce potřebná k používání selekčních příkazů SELECT. V kursoru se ukládají data ze selekčních příkazů, které se zadávají za pomocí funkce rawQuery(sql1, null). K tomu je nutné mít připravenou databázi v režimu čtení.

```
public Recept nactiRecept(int jidelnicekID) {
        String sql1 = "SELECT recept_id,recept_nazev,
recept_delka_vareni, recept_narocnost, jidelnicek_datum,
druhy_jidla_druh_jidla_id ,jidelnicek_id FROM recepty\n" +
```

```
                "left join recepty_jidelnicky on
recepty.recept_id=recepty_jidelnicky.recepty_recept_id\n" +
                "left join jidelnicky on
recepty_jidelnicky.jidelnicky_jidelnicek_id=jidelnicky.jidelnicek_id \n"
+
                "where jidelnicek_id=" + jidelnicekID + ";";
        final Cursor data = mDB.rawQuery(sql1, null);
        data.moveToFirst();

        Recept recept = new Recept();
        recept.setId(Integer.parseInt(data.getString(6)));
        recept.setNazev(data.getString(1));
        recept.setDelkaVareni(Integer.parseInt(data.getString(2)));
        recept.setNarocnost(Integer.parseInt(data.getString(3)));

        return recept;
    }
```

Tato metoda je v aktivitě registrace a pomáhá uložit do proměnné databáze údaje o uživateli, konkrétně v tomhle případě současnou tělesnou váhu, po vyplnění a stisknutí tlačítka hotovo na Android klávesnici se aktivita přesune na další registrační.

```
  editTextVahaCelkemCil.setOnKeyListener(new View.OnKeyListener() {
        public boolean onKey(View v, int keyCode, KeyEvent event) {
            if ((event.getAction() == KeyEvent.ACTION_DOWN)
            && (keyCode == KeyEvent.KEYCODE_ENTER)) {
                String s=editTextVahaCelkemCil.getText().toString();
                try {
                    vahaCelkemCil
=(NumberFormat.getInstance().parse(s)).doubleValue();
                    MainActivity.uzivatel.setCilCelkemVaha(i);
                } catch (ParseException e) {

Toast.makeText(getApplicationContext(),e.toString(),Toast.LENGTH_SHORT);
                }
                buttonRegistrace2Vpred.performClick();
                return true;
            }
            return false;
        }
  });
```



Obrázek 10 – Ukázka registrace

5.3.3 Implementace registrace

Prvním krokem, který musí uživatel udělat hned po prvním spuštění aplikace, je registrace. Ta začíná úvodní obrazovkou, která dává uživateli na výběr, jestli se chce přihlásit nebo opravdu registrovat. Tuto uvítací obrazovku je možné vidět na obrázku číslo 11. Při poklepání na tlačítko registrace je uživatel proveden několika různými obrazovkami, které se ho ptají na osobní otázky, které jsou nezbytné k odhadnutí jeho denních energetických výdajů. V OS Android jsou obrazovky provedeny za pomocí dvou souborů, bez kterých by se obrazovka nezměnila. Prvním souborem je XML soubor layout ressource file. Ten se v základu skládá pouze z jednoho layoutu. Ve většině případů výběru layoutu se bude nejspíš vybírat RelativeLayout nebo LinearLayout. Na tyto layouty se později pokládají jejich potomci, komponenty, které se na něm vykreslí. Potomkem layoutu může být mimo view i další layout. Java class, která musí dědit od třídy Activity, aby se dala přidat do manifest.xml, jak je výše v této kapitole popisováno.

LinearLayout zarovnává svoje potomky do několika stejně velkých řádků či sloupců (při připojení stejně rozměrných komponentů). Do řádky se komponenty přidávají od shora dolu. Při horizontálním přidávání se zase dodávají zleva doprava.

RelativeLayout pokládává svoje v sobě obsažené view relativně od vybraného nejbližšího komponentu, tím může být i okraj layoutu. Nastavení pozice view tedy probíhá na základě pozic ostatních komponent. Na příklad se dá říci, že pozice jednoho view má být nad druhým 20 pixelů a ten že má být horizontálně vycentrován s layoutem, ve kterém je právě umístěn. To znamená, že horní hrana prvního a spodní hrana druhého leží rovnoběžně s mezerou 20 pixelů. Na následujícím příkladu je popsána problematika tohoto pozicování.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="305dp"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Kolik chcete celkem ztratit/přibrat?"
        android:id="@+id/textViewNadpis"
        android:textSize="32dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="44dp"
        android:gravity="center" />
……
</RelativeLayout>
```

Element TextView na předchozím příkladu nastavuje několik atributů. Ten, který nastavuje pevnou šířku komponenty je layout_width. Rozměry se dají nastavit v několika základních jednotkách, zde je uveden příklad jedné z hodně používaných a tím je DP. DP je Density-independent Pixels, to znamená, že tento rozměr je na hustotě pixelů nezávislý. Tu má totiž každý display jinou. Další atributy jako `layout_height` nastavuje šířku, text nastavuje zobrazovaný text popisku, `textSize` zase výšku, `marginTop` představuje odsazení od horního okraje a `gravity` pracuje jako zarovnání textu v komponentě.

Obrázek 11 - ukázka aplikace

5.3.4 Způsob provedení oblíbenosti potravin

Každý z nás má určitě alespoň jednu potravinu, kterou si neoblíbil anebo ji nedokáže pozřít. A proto aplikace má implementovanou vlastnost, která dokáže tyto potraviny při generování jídelníčku odfiltrovat. Způsob, jakým je výběr neoblíbených potravin navržen, je jejich výpis do specielně navrženého seznamu. Odlišný od běžného listView je pravě v přítomnosti checkBoxu, který má představovat, zda potravina je neoblíbená (zaškrtnuté pole check boxu) nebo jestli uživatel potravinu jí (odškrtnuté pole). Pro řešení tohoto listu je nutné si vytvořit třídu, která je rozšířená ArrayAdapter<Potravina>. Adapter obsahuje atribut seznamu potravin ArrayListu<Potravina>. Ten v sobě uchovává položky, které jsou zobrazeny právě v listView. Pro zobrazení textu a check boxu je nezbytné vytvořit si xml soubor s jedním textView, checkBoxem a LinearLayoutem pro změnu barvy pozadí. Dále je klíčová přetížená metoda getView (int position, View convertView, ViewGroup parent) ta má za úkol právě nastavit, aby se v listu komponenty zobrazily hned po nastavení adaptéru za pomoci jeho instance a metody listu setAdapter().

```
@Override
      public View getView(int position, View convertView, ViewGroup
parent) {
            View v = convertView;

            if(v == null) {

            LayoutInflater inflater = (LayoutInflater)
            context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            v = inflater.inflate(R.layout.single_listview_item, null);

            TextView nazevPotraviny = (TextView)
            v.findViewById(R.id.name);
            CheckBox  chkBox= (CheckBox) v.findViewById(R.id.chk_box);

            }

            Potravina p = seznamPotravin.get(position);
            nazevPotraviny.setText(p.getName());
            holder.chkBox.setChecked(p.isSelected());
            holder.chkBox.setTag(p);

            return v;
      }
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
       xmlns:tools="http://schemas.android.com/tools"
       android:layout_width="match_parent"
       android:layout_height="match_parent" >
    <CheckBox
        android:id="@+id/chk_box"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true" />
     <TextView
        android:id="@+id/nazevPotraviny"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/chk_box"
        android:textStyle="bold" />
</RelativeLayout>
```

5.3.5 Používání selekcí dat

Selekce jsou hlavní součástí výběru dat a jejich filtrování. Používají se všude, kde potřebujeme z velkého množství informací získat pouze určitou malou omezenou část z nich. Právě databáze spadají přesně do této skupiny ohromného množství dat. Ty v sobě obsahují daný počet tabulek, ze kterých se za asistence databázového systému vybere požadovaná omezená část informací. Selekce se nedělá jinak než za použití dotazu začínajícího na klíčové slovo SELECT. Ten požaduje názvy sloupců, ze kterých má selektovat data. Následujícím příkazem FROM se výběr omezí pouze na jednu tabulku. Pro specifičtější výběr se použije podmínka pomocí WHERE příkazu pokračující názvem sloupce a příkazy IS LIKE, NOT LIKE, rovná se, apod. Dotaz se zakončí hodnotou a středníkem, v tomto případě. Vybírání dat v této mobilní aplikaci se provádí úplně stejně.

Pro zjištění jakého uživatele je jídelníček s id 5, se provede následující dotaz:

```
SELECT uzivatele_uzivatel_email, jidelnicky_jidelnicek_id from
uzivatele_jidelnicky WHERE jidelnicky_jidelnicek_id=5;
```

Aby systém aplikace věděl při výpisu uživatelských jídelníčků v jaký den si uživatel naplánoval konzumovat jaký jídelníček s id 5, spustí se dotaz.

```
SELECT recept_id, jidelnicek_datum, druhy_jidla_druh_jidla_id
,jidelnicek_id FROM recepty left join
recepty_jidelnicky on
recepty.recept_id=recepty_jidelnicky.recepty_recept_id left join
jidelnicky on
recepty_jidelnicky.jidelnicky_jidelnicek_id=jidelnicky.jidelnicek_id
WHERE jidelnicek_id=5;
```

Dalším příkladem je příkaz pro výběr všech druhů jídel (oběd, snídaně, večeře, svačina) pro zjištění, k jakému jídlu si můžeme recept s id 1 uvařit. Selekce se používá pro výpis katalogu receptů nebo při vybírání receptu do svého jídelníčku.

```
SELECT recept_id, druh_jidla_id FROM chody left join druhy_jidla on
chody.druhy_jidla_druh_jidla_id=druhy_jidla.druh_jidla_id left join
recepty on chody.recepty_recept_id=recepty.recept_id WHERE recept_id=1;
```

5.3.6 Styl získávání jídelníčku

Aplikace je určena pro všechny ty, kteří by chtěli svůj jídelníček upravit dle doporučených zásad zdravé výživy a být tak odměněni lepším zdravím. Nepotřebují kupovat drahé kuchařky, či si platit výživové poradce. Pro sestavení je potřeba obecný vzorec. První zásadou v používání aplikace, tak aby byla účinná, je pravidelně doplňovat tekutiny a dodržovat pravidelnost stravování. Další zásadou, aby si uživatel mohl naplánovat správně svůj stravovací plán, je požadavek správnosti vyplnění osobních informací, jinak by použitý algoritmus nebyl účinný a plány, které aplikace vytvoří, by nebyly funkční.

Bezprostředně po registraci je uživateli vypočítána hodnota bazálního metabolismu metodou v třídě uživatele, aby bylo možné vytvářet si jídelníčky. Hodnota se po každém zapnutí aplikace přepočítává.

```
public double vypoctiBMR(){
```

```
        double BMR=0;
        if(pohlavi==ETypPohlavi.MUZ){
              BMR=(10*vaha +6.25*vyska- 5*vek +5) *aktivnost;
        }else{
              BMR=(10*vaha +6.25*vyska- 5*vek -161)*aktivnost;
        }
        return BMR;
}
```

Když je známá hodnota BMR výdaje energie, je možné si snadno vygenerovat jídelníček. Například: Uživatel aplikace si zvolí dietu udržení váhy, jeho současná váha je 100kg a měří 200 centimetrů metrů, stáří mu nic neříká, protože je mu 20 let a je vysoce aktivní sportovec. Jaké bude mít BMR?

```
BRM=10*vaha +6.25*vyska- 5*vek +5) *aktivnost
BRM=(10*100 +6.25*200- 5*20+5)*2=4310 kcal
```

Protože je vysoce aktivní sportovec má proto o 100% větší spotřebu energie než člověk, který se vůbec nehýbe. Uživatel provedl registraci, a proto se mu provedl výpočet BMR. Stáhnul si aplikaci, protože nechtěl přemýšlet nad tím, co si má každý den vařit za jídla. Proto neváhá a vybere si v menu funkci vygenerovat stravovací plán. Po klepnutí na vybrané tlačítko se spustí EventHandler události OnClick. Ta se provede a otevře okno pro novou aktivitu, ve které je textView s nápisem „Na kolik dní mám vytvořit návrh stravovacího plánu?“ a pod ním umístěné počítadlo dní od 1 do x dnů a tlačítko s nápisem Start.

Uživatel si vybere 1 den, vykoná se metoda vygenerujJidelnicek() s počtem dní předanou jako parametr. Vždy si metoda vytvoří 4 instance třídy JidloDne a nastaví 4 různé druhy jídel a jeden den a přiřadí vlastníka návrhu. Jeden den má obvykle 4 jídla a to snídani, svačinu, oběd a večeři. Metoda používá generujJidloDne(), která se provede 4x pro každou instanci zvlášť.

Metoda generujJidloDne() projde databázi recept po receptu a naplní si lokální proměnou ArrayList<Recept> všemi recepty, které mají stejný druh jídla jako má nastavené třída JidloDne. Po naplnění seznamu se prochází znovu seznam, teď již naplněný a maže nevhodná jídla pro uživatele. Ty pozná tak, že celkový příjem z jídla není v rozmezí hodnot jeho bazálního metabolizmu BRM*0.2 až BRM*0.25 v případě snídaně. Oběd je 25-30% z denního příjmu potravin. Večeře zas 20-25% a svačina 5-10%. Při filtrování se zohledňují potraviny, které má uživatel neoblíbené. Recepty s touto potravinou se smažou ze seznamu. Metoda končí uložením receptu do jedné instance třídy JidloDne. Ta se později ukládá do seznamuJidel.

Množství dní, pro které se vygeneruje návrh stravovacího plánu do seznamuJidel obsluhuje for cyklus, který proběhne právě tolikrát, kolik je zadaná hodnota parametru metody. Po skončení cyklu metoda vyvolá speciální aktivitu, která vypíše jídelníčky a zobrazí dvě tlačítka pro přijmutí či odmítnutí návrhu. V případě odmítnutí se celý proces až doposud opakuje. Pokud uživatel jídelníček příjme, uloží jej systém jídelníček tohoto uživatele do databáze a začne znovu aktivita hlavního menu.

6 Závěr

Cílem bakalářské práce bylo vytvořit mobilní aplikaci podporující fitness lifestyle. Aplikace má vytvářet uživateli návrhy individuálních stravovacích plánů na libovolně dlouhou dobu. Při přijmutí návrhu by se měly uživateli jídelníčky uložit. Uživatel má být schopen si jídelníček prohlížet, stejně jako recepty, ze kterých je tvořen. Grafické rozhraní má být pro uživatele intuitivní, přehledné a srozumitelné. Pro tvorbu těchto aplikací byl výsledně zvolen jazyk Java. Pro ukládání dat slouží databáze SQLite. Splnění některých z těchto cílů mi způsobovalo nemalé potíže, ale nakonec se mi podařilo tyto překážky překonat a aplikace je připravena k používání. Protože si myslím, že provozovat aplikaci s připojením k databázovému serveru je pro studenta příliš nákladné, zvolil jsem cestu, která je méně nákladná a nepotřebuje ani připojení k internetu.

Ze začátku, při seznamování se s pro mě novým okruhem informačních technologií, jsem měl těžkosti. Takže nejdříve jsem se seznamoval s vývojovým prostředím. Android studio na začátku ukázalo, proč má na sobě štítek beta verze. Po stažení instalačního souboru z oficiálních stránek se mi ho po jeho rozbalení a instalaci nepodařilo spustit. Rozhodl jsem se správně, když jsem si stáhl o něco starší verzi. Ta se spustila se spoustou chybových hlášek. Ty byly způsobeny neaktuálně nainstalovanou verzí Java SE a její špatnou konfigurací v mém operačním systému.

Další problém se vyskytl po připojení testovacího zařízení do USB počítače. S tím se Android studio nedokázalo synchronizovat. Vývojové prostředí odmítalo testovací aplikaci „Hello world!“ zbuildovat bez připojeného zařízení nebo simulovaného virtuálního zařízení. Virtualizované zařízení potřebovalo přespříliš operační paměti, protože studio si bez problému alokuje po jeho spuštění jeden gigabite paměti.

Následující nesnází bylo, že jsem ještě neměl vůbec žádné zkušenosti s tvorbou aplikací pro Android. Zjistil jsem, že tvorba těchto aplikací není náročná, ale díky tomu, že to byla má úplně první aplikace, tvorba postupovala extrémně pomalu. Prováděl jsem testování od toho, jak se do listView přidávají položky, až po vytváření vlastních expandListView. Naštěstí vývoj pro Android není úplnou novinkou. O této problematice je napsáno ve spoustě užitečné literatury. Také nesmím zapomenout na množství příspěvků na vývojářském fóru stackoverflow.com, které mi téměř vždy pomohly.

Asi nejobtížnější záležitostí byla otázka, jak propojit moji databázi s mobilní aplikací. Knihy a materiály, na které jsem narážel při studiu této problematiky, byli již z části zastaralé a nefunkční.

Nynější informace o správných stravovacích návycích, které jsem získal studiem literatury, jsou pro mě velice cenné. Díky nim jsem si začal vařit a jíst zdravěji. Avšak chtěl bych upravit tuto aplikaci tak, aby zahrnovala i látky, které tělo nutně potřebuje. Mimo bílkoviny, sacharidy a tuky, bych si přál, aby se aplikace při generování jídelníčku řídila glykemickou zátěží potravin a individuálními denními dávkami pro vitamíny a minerály a dalšími látkami. Líbilo by se mi v budoucnu implementovat do aplikace průvodce. Ten za pomocí motivujícího systému, který je založený na plnění denních úkolů, dokáže v člověku probudit touhu používat aplikaci neustále. K tomu by bylo taky potřeba vylepšit vzhled aplikace, se kterým nejsem moc spokojen. Dosáhnout toho bohužel lze pouze tvorbou vlastních komponent, které sednou na míru jak barevně tak i funkčně. Je to zdlouhavá práce, která vyžaduje hodně trpělivosti a preciznosti.

Díky této bakalářské práci jsem mohl využít znalosti nabyté při bakalářském studiu a naučil jsem se i něco navíc, vývoj aplikace pro operační systém Android a UML. Při navrhování designu a při vývoji jsem se naučil pracovat i s XML, SQLite a mnoha dalšími technologiemi. Sice se v průběhu tvorby objevilo pár problémů, ale ty byly nakonec úspěšně překonány. Pevně věřím, že aplikace bude mít několik spokojených uživatelů a že aplikaci budou využívat každý den. Pokud budou v budoucnu aktivní uživatelé, budu moci rozšířit aplikaci o další z výše navrhovaných funkcí.

Literatura

1. 10 tipů pro sportovně založené uživatele s Androidem. *Svět Androida* [online]. 2013 [cit. 2015-05-17]. Dostupné z: http://www.svetandroida.cz/10-tipu-pro-sportovne-zalozene-uzivatele-s-androidem-201301

2. Noom loss coach review. *RiaFit* [online]. 2014 [cit. 2015-05-11]. Dostupné z: http://www.riafit.co.uk/2014/01/noom-weight-loss-coach-review.html

3. Android studio. 2013. *AWS* [online]. [cit. 2015-05-11]. Dostupné z: https://s3.amazonaws.com/infinum.web.production/repository_items/files/000/000/168/original/android-studio-3.png?1393599622

4. SQLite exper professional. 2012. *Full Program Indir* [online]. [cit. 2015-05-11]. Dostupné z: http://www.fullprogramlarindir.com/wp-content/uploads/2014/07/sqlite_expert_professional-447879-1300510547.jpeg

5. Oracle SQL Developer Data Modeler. 2011. *Softpedia* [online]. [cit. 2015-05-11]. Dostupné z: http://i1-mac.softpedia-static.com/screenshots/Oracle-SQL-Developer-Data-Modeler_1.jpg

6.UML: Diagramy tříd. 2010. *Miloš němec* [online]. [cit. 2015-05-11]. Dostupné z: http://www.milosnemec.cz/clanek.php?id=199

7 Nastroje pro tvorbu uml diagramu. 2012. *Root* [online]. [cit. 2015-05-17]. Dostupné z: http://www.root.cz/clanky/nastroje-pro-tvorbu-uml-diagramu/

8 Databáze. 2012. *Wikipedia otevřená encyklopedie* [online]. [cit. 2015-05-17]. Dostupné z: http://cs.wikipedia.org/wiki/Datab%C3%A1ze

9 FILIPENSKÁ, Jana a . *Mobilní aplikace pro výživové poradenství*. Pardubice, 2013. Signatura: D28796. Diplomová práce (.ing).

10 UJBÁNYAI, Miroslav. *Programujeme pro Android*. Vyd. 1. Praha: Grada, 2012, 187 s. Průvodce (Grada). ISBN 978-80-247-3995-3.

11 MURPHY, Mark. 2011. Android 2. Brno: Computer Press. ISBN 978-80-251-3194-7.

12 ARLOW, Jim. 2008. UML 2 a unifikovaný proces vývoje aplikací. Brno: Computer Press. ISBN 978-80-251-1503-9.

Příloha A – Fyzický model databáze

```
CREATE TABLE [Chody] (

  [Druhy_jidla_druh_jidla_id] INTEGER NOT NULL CONSTRAINT [FK_ASS_2] REFERENCES [Druhy_jidla]([druh_jidla_id]),

  [Recepty_recept_id] INTEGER NOT NULL CONSTRAINT [FK_ASS_3] REFERENCES [Recepty]([recept_id]),

  CONSTRAINT [sqlite_autoindex_Chody_1] PRIMARY KEY ([Druhy_jidla_druh_jidla_id], [Recepty_recept_id]))


CREATE TABLE [Druhy_jidla] (

  [druh_jidla_id] INTEGER NOT NULL PRIMARY KEY UNIQUE,

  [druh_jidla_nazev] [VARCHAR2 ] NOT NULL)


CREATE TABLE [Jidelnicky] (

  [jidelnicek_id] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

  [jidelnicek_datum] DATE,

  [druhy_jidla_druh_jidla_id] INTEGER CONSTRAINT [FK_ASS_8] REFERENCES [Druhy_jidla]([druh_jidla_id]))


CREATE TABLE [Nutricni_hodnoty] (

  [nutricni_hodnota_id] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

  [nutricni_hodnota_nazev] VARCHAR2(50) NOT NULL,

  [nutricni_hodnota_bilkoviny] NUMBER,

  [nutricni_hodnota_sacharidy] NUMBER,

  [nutricni_hodnota_tuky] NUMBER,

  [nutricni_hodnota_voda] NUMBER,

  [nutricni_hodnota_vlaknina] NUMBER,

  [nutricni_hodnota_cholesterol] NUMBER)


CREATE TABLE [Oblibenost_potravin] (
```

```
  [Uzivatele_uzivatel_email] VARCHAR2 NOT NULL CONSTRAINT [FK_ASS_9]
REFERENCES [Uzivatele]([uzivatel_email]),

  [Nutricni_hodnoty_nutricni_hodnota_id] NUMBER NOT NULL CONSTRAINT
[FK_ASS_10] REFERENCES [Nutricni_hodnoty]([nutricni_hodnota_id]))


CREATE TABLE [Recepty] (

  [recept_id] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

  [recept_nazev] [VARCHAR2 ] NOT NULL,

  [recept_postup] CLOB NOT NULL,

  [recept_delka_vareni] NUMBER,

  [recept_narocnost] NUMBER)


CREATE TABLE [Recepty_jidelnicky] (

  [Recepty_recept_id] INTEGER NOT NULL CONSTRAINT [FK_ASS_4]
REFERENCES [Recepty]([recept_id]),

  [Jidelnicky_jidelnicek_id] INTEGER NOT NULL CONSTRAINT [FK_ASS_5]
REFERENCES [Jidelnicky]([jidelnicek_id]),

  CONSTRAINT [] PRIMARY KEY ([Recepty_recept_id], [Jidelnicky_jidelnicek_id]))


CREATE TABLE [Suroviny] (

  [surovina_jednotka] [VARCHAR2 ] NOT NULL,

  [surovina_mnozstvi] NUMBER NOT NULL,

  [Recepty_recept_id] INTEGER NOT NULL CONSTRAINT [Suroviny_Recepty_FK]
REFERENCES [Recepty]([recept_id]),

  [Nutricni_hodnoty_nutricni_hodnota_id] INTEGER NOT NULL CONSTRAINT
[Suroviny_Nutricni_hodnoty_FK] REFERENCES
[Nutricni_hodnoty]([nutricni_hodnota_id]))


CREATE TABLE [Uzivatele] (

  [uzivatel_email] [VARCHAR2 ] NOT NULL UNIQUE,

  [uzivatel_jmeno] [VARCHAR2 ] NOT NULL,

  [uzivatel_heslo] [VARCHAR2 ] NOT NULL,
```

[uzivatel_vaha_cil] VARCHAR2(20) NOT NULL,

[uzivatel_cil_celkem_vaha] NUMBER,

[uzivatel_cil_tyden_vaha] NUMBER,

[uzivatel_pohlavi] VARCHAR2(10) NOT NULL,

[uzivatel_vyska] NUMBER NOT NULL,

[uzivatel_vaha] NUMBER NOT NULL,

CONSTRAINT [sqlite_autoindex_Uzivatele_1] PRIMARY KEY ([uzivatel_email]))

CREATE TABLE [Uzivatele_jidelnicky] (

[Uzivatele_uzivatel_email] [VARCHAR2] NOT NULL CONSTRAINT [FK_ASS_6] REFERENCES [Uzivatele]([uzivatel_email]),

[Jidelnicky_jidelnicek_id] INTEGER NOT NULL CONSTRAINT [FK_ASS_7] REFERENCES [Jidelnicky]([jidelnicek_id]),

CONSTRAINT [] PRIMARY KEY ([Uzivatele_uzivatel_email], [Jidelnicky_jidelnicek_id]))

Příloha B – Zdrojový kód souboru DatabaseHelper.java

```
public class DataBaseHelper extends SQLiteOpenHelper
{
    private static String TAG = "DataBaseHelper"; // Tag just for the
LogCat window
    //destination path (location) of our database on device
    private static String DB_PATH = "";
    private static String DB_NAME ="comp.db";// Database name
    private SQLiteDatabase mDataBase;
    private final Context mContext;

    public DataBaseHelper(Context context)
    {
        super(context, DB_NAME, null, 1);// 1? its Database Version
        if(android.os.Build.VERSION.SDK_INT >= 17){
            DB_PATH = context.getApplicationInfo().dataDir +
"/databases/";
        }
        else
        {
            DB_PATH = "/data/data/" + context.getPackageName() +
"/databases/";
        }
        this.mContext = context;
    }

    public void createDataBase() throws IOException
    {
        //If database not exists copy it from the assets

        boolean mDataBaseExist = checkDataBase();
        if(!mDataBaseExist)
        {
            this.getReadableDatabase();
            this.close();
            try
            {
                //Copy the database from assests
                copyDataBase();
                Log.e(TAG, "createDatabase database created");
            }
            catch (IOException mIOException)
            {
                throw new Error("ErrorCopyingDataBase");
            }
        }
    }
    //Check that the database exists here: /data/data/your
package/databases/Da Name
    private boolean checkDataBase()
    {
        File dbFile = new File(DB_PATH + DB_NAME);
        //Log.v("dbFile", dbFile + "   "+ dbFile.exists());
        return dbFile.exists();
    }

    //Copy the database from assets
    private void copyDataBase() throws IOException
    {
```

```
        InputStream mInput = mContext.getAssets().open(DB_NAME);
        String outFileName = DB_PATH + DB_NAME;
        OutputStream mOutput = new FileOutputStream(outFileName);
        byte[] mBuffer = new byte[1024];
        int mLength;
        while ((mLength = mInput.read(mBuffer))>0)
        {
            mOutput.write(mBuffer, 0, mLength);
        }
        mOutput.flush();
        mOutput.close();
        mInput.close();
    }

    //Open the database, so we can query it
    public boolean openDataBase() throws SQLException
    {
        String mPath = DB_PATH + DB_NAME;
        //Log.v("mPath", mPath);
        mDataBase = SQLiteDatabase.openDatabase(mPath, null,
SQLiteDatabase.CREATE_IF_NECESSARY);
        //mDataBase = SQLiteDatabase.openDatabase(mPath, null,
SQLiteDatabase.NO_LOCALIZED_COLLATORS);
        return mDataBase != null;
    }

    @Override
    public synchronized void close()
    {
        if(mDataBase != null)
            mDataBase.close();
        super.close();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {

    }

}
```