PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Aplikace na objednávání jídel pro platformu Android

2012 Lukáš Novák

Anotace

V rámci bakalářské práce (na vlastní téma) byla vyvinuta aplikace pro mobilní platformu Android, která umožňuje správu stravovacího účtu v informačním systému Kredit. Tento informační systém využívá Správa kolejí a menz Univerzity Palackého. Primární použití výsledné aplikace je tak umožnit správu stravovacího účtu přes mobilní zařízení studentům Univerzity Palackého v Olomouci. Aplikace byla vyvinuta ve spolupráci se společností ANETE spol. s r.o., dodavatelem informačního systému Kredit, která k němu v průběhu práce vyvíjela webové služby.

Děkuji vedoucímu práce, Mgr. Petru Krajčovi, Ph.D., za cenné připomínky na konzultacích, společnosti ANETE spol. s r.o. za umožnění vzniku této práce a konzultantovi Ing. Karlu Královi za spolupráci při vývoji.

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Mobilní aplikace jsou dnes velmi populární. Čím více roste tržní podíl mobilních zařízení s pokročilým operačním systémem[1], tím je oblast vývoje pro tyto zařízení důležitější. Do skupiny pokročilých operačních systémů můžeme zařadit nejvýraznější „hráče“:

- **Android OS** je rozsáhlá svobodná platforma vyvinutá konsorciem Open Handset Alliance, do nějž spadá Google, Inc. Podrobněji je tento operační systém popsán v kapitole 4.
- **Apple iOS** je uzavřený OS vyvíjený společností Apple, Inc. Je nasazován na přístroje pouze této společnosti, např. Apple iPhone či Apple iPad.
- **Windows Phone** je nový operační systém vyvíjený společností Microsoft. Byl uveden na podzim roku 2010, díky čemuž prozatím není masově rozšířený.
- **Symbian OS** je operační systém vyvíjený společností Symbian Ltd., později výrobcem mobilních telefonů Nokia. V únoru 2011 však Nokia oznámila ústup od dalšího vývoje.

Právě první jmenovaný, Android OS, nahrazuje na trhu zařízení s jednoduchým operačním systémem. Je nasazován i do nejlevnějších zařízení, tzv. „low-endů“, proto člověk, který nehledá výhody pokročilého operačního systému, ani nemusí vědět, že je jeho mobilní zařízení tímto operačním systémem vybaveno. I tento uživatel ale může využít komfortu ve formě dalších doplňkových aplikací.

S rostoucím rozlišením a výkonem dnešních mobilních zařízení se objevuje stále více mobilních aplikací, které usnadňují uživatelům činnosti, jež by bez těchto zařízení museli vykonávat s pomocí osobního počítače.

Jednou z takových činností, které by bylo možné přesunout z osobních počítačů na mobilní zařízení, je i objednávání jídel v univerzitních menzách. Pro tuto činnost je zapotřebí počítač s přístupem na internet. K němu ale strávník nemusí mít přístup vždy, zatímco mobilní zařízení má s sebou téměř stále.

Vytvoření mobilní aplikace pro objednávání jídel v univerzitních menzách by tuto činnost jistě ulehčilo. Strávník by si takovým způsobem mohl objednat či odhlásit oběd v kteroukoliv dobu, a to bez nutnosti být u osobního počítače s přístupem na internet.

V rámci této bakalářské práce byla vyvinuta aplikace, pomocí které je možné kompletně spravovat stravovací účet. To znamená zobrazovat nabídku určeného

[1]Pokročilý operační systém se vyznačuje například aplikačním rozhraním, které poskytuje vývoj vlastních aplikací, pokročilou správou paměti, podporou běhu více procesů současně a podobně. Mobilní zařízení s takovým operačním systémem se nazývá smartphone, někdy také chytrý telefon.

data a výdejny, vytvářet objednávky, ale také například kontrolovat zůstatek na stravovacím účtu. V zadání práce je také zmínka o možnosti dalších doplňkových funkcí jako například zobrazení historie účtu. Tato i další funkce vyvinuté aplikace jsou popsány v kapitole 7.

2. Příprava práce

Tato kapitola popisuje situaci před vývojem aplikace a počátky spolupráce se společností ANETE spol. s r.o. Pro tvorbu aplikace bylo navázání spolupráce nezbytné. K informačnímu systému Kredit totiž nebylo žádné veřejné API, které by bylo možné pro komunikaci se serverem použít.

2.1. Současná situace

Pro objednávání jídel z univerzitních menz, kde je využíván informační systém Kredit, je možné využít internetovou aplikaci WebKredit, jejíž snímek obrazovky je na obrázku 1.. Ta je pro strávníky Univerzity Palackého dostupná na adrese `http://menza.upol.cz/`. Tato aplikace je sice přístupná na mobilních zařízeních přes běžný internetový prohlížeč, nicméně častému používání nevyhovuje z několika důvodů.

Za velkou nevýhodu lze považovat datovou náročnost, kdy při přihlášení, zobrazení nabídky a objednání jídla dojde ke stáhnutí dat o objemu 1 MB. Při použití přes běžné internetové připojení na osobním počítači jde o zanedbatelný objem dat, což ale neplatí při použití mobilního internetového připojení, kde je u většiny operátorů nastavený FUP[2].

S datovým objemem také souvisí rychlost internetového připojení. Ta není v současné době na všech místech České republiky dostačující pro přenos potřebného objemu dat, proto je práce se stravovacím účtem přes internetový prohlížeč velmi pomalá.

Další nevýhoda je neoptimalizované uživatelské rozhraní. Internetová aplikace není navržena pro mobilní zařízení, proto je práce s ní velmi zdlouhavá, nevhodná pro ovládání dotykem a nepřipravena pro současné rozlišení displejů mobilních zařízení. Jak na takovém rozlišení vypadá internetová aplikace je patrné z obrázku 2.

S aplikací navíc nelze při použití některých mobilních internetových prohlížečů pracovat z důvodu použití technologie AJAX[3]. Neustále probíhá automatické obnovování stránky, proto je nemožné provést jakoukoli akci. Běžného uživatele ale nemusí napadnout, že je možné tuto situaci vyřešit použitím jiného internetového prohlížeče.

[2] Akronym Fair Use Policy, díky kterému po vyčerpání určitého datového objemu dojde nejčastěji k rapidnímu omezení rychlosti internetového připojení. Limit FUP u běžných datových tarifů se dnes pohybuje kolem 150 MB.

[3] Akronym Asynchronous JavaScript and XML označuje technologie pro vývoj webových aplikací, kde probíhá interakce s uživatelem bez nutnosti opětovného načítání stránky.

Obrázek 1. Ukázka internetové aplikace WebKredit

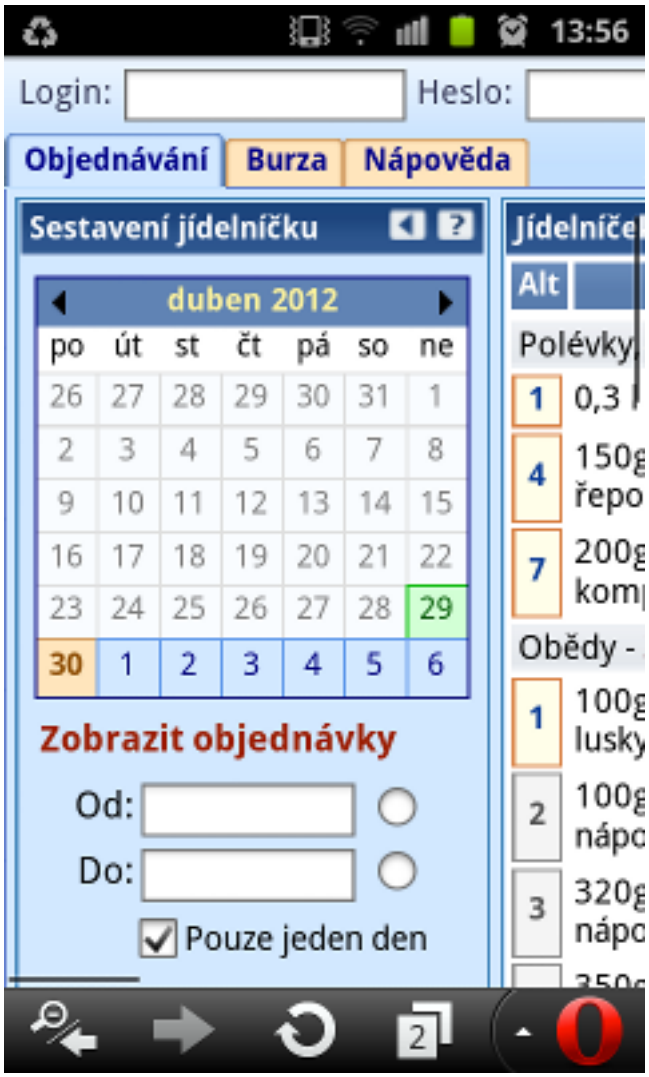
2.2. Podobné existující aplikace

Před vznikem aplikace bylo nutné zjistit, zda neexistují podobné aplikace pro správu stravovacích účtů českých menz. V obchodu s aplikacemi Google Play (více o Google Play v kapitole 8.2.) bylo v té době pět aplikací pro pouhé zobrazení jídelníčku na univerzitách ČVUT, VUT, Západočeské univerzitě v Plzni, Mendelově univerzitě v Brně a Masarykově univerzitě (seznam ke shlédnutí je na adrese: http://play.google.com/store/search?q=menza&c=apps).

Tyto aplikace jsou však určeny pouze pro zobrazení jídelníčku menz, popřípadě pro hodnocení jídel, nikoliv pro kompletní správu stravovacího účtu, což je hlavní funkcionalitou této aplikace. Pro Univerzitu Palackého však neexistovala ani taková aplikace. Již jen samotné zobrazení jídelníčku je jistě užitečná funkce, nicméně výsledek této bakalářské práce, zejména možnost vytvoření objednávek, posouvá interakci s menzami mnohem dál.

2.3. Komunikace s dodavatelem informačního systému

Pro vytvoření alternativy k internetové aplikaci bylo zapotřebí mít k dispozici programové rozhraní, tzv. API. Po kontaktování Správy kolejí a menz, která žádné API k k dispozici neměla, byla na její doporučení oslovena brněnská společnost ANETE spol. s r.o., která jim informační systém dodává.



Obrázek 2. Ukázka internetové aplikace WebKredit na mobilním zařízení

Na schůzce s vedením společnosti bylo zjištěno, že API mají pouze pro interní účely, nicméně myšlenka vytvořit aplikaci pro mobilní zařízení je natolik zaujala, že byli ochotni spolupracovat. Vývojem webových služeb byl pověřen zaměstnanec společnosti, Ing. Karel Král, který se tak stal konzultantem této bakalářské práce.

Vedení společnosti zpočátku uvažovalo nad jiným řešením, jak dostat správu stravovacího účtu na mobilní zařízení, a to vytvořit optimalizovanou internetovou aplikaci, která by bez problémů běžela na všech platformách. Po popisu výhod nativní aplikace souhlasili s předloženým návrhem.

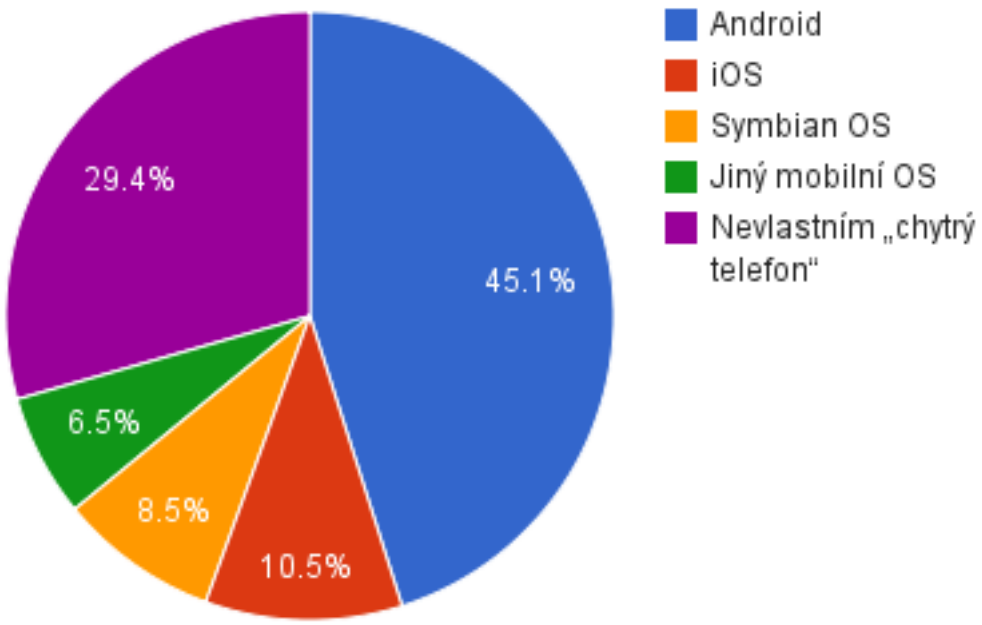
Mezi výhody nativní aplikace patří:

- Garantovaný stejný vzhled aplikace na všech zařízeních napříč různými roz-

lišeními. Lze využít systémové formulářové prvky, uživateli známý výběr data z kalendáře a podobně. Aplikace tak vypadá stejně jako zbytek systému a díky tomu by neměl být pro uživatele problém s aplikací pracovat.

- Nativní aplikace poskytuje lepší uživatelskou interakci pomocí dotykového displeje.
- Aplikaci je možné lépe integrovat do systému a použít funkce, které by ve webové aplikaci nemohly být k dispozici. Například použití systémových upozornění, přidávání položek do kalendáře či zobrazování tzv. „widgetů“ – miniaplikací, které lze vložit do domovské obrazovky systému.
- Nativní aplikaci je možné mnohem efektivněji distribuovat. Lze to provést pomocí integrovaných obchodů s aplikacemi, v případě Android OS jde o Google Play (dříve Android Market).

2.4. Průzkum rozšíření OS Android



Obrázek 3. Diagram zastoupení mobilních OS mezi studenty

I přes výhody uvedené v předchozí kapitole byla pro vedení společnosti ANETE spol. s r.o. důležitá informace, kolik strávníků vlastní telefon s OS Android a kolik z nich bude moci teoreticky aplikaci využívat.

Z tohoto důvodu byl proveden průzkum mezi studenty Univerzity Palackého, v němž bylo zjišťováno, jaký operační systém mají ve svém mobilním telefonu.

Pro účely zjištění zastoupení jednotlivých mobilních OS byl vytvořen dotazník pomocí webové aplikace Google Docs, který byl následně šířen pomocí sociálních sítí. Za pár dní se podařilo shromáždit téměř 500 odpovědí studentů Univerzity Palackého na otázku, jaký mobilní OS respondent používá. Z výsledku (obr. 3.) je patrné, že tři ze čtyř studentů Univerzity Palackého vlastní chytrý telefon, přičemž dva z nich na něm využívají OS Android. Dá se usuzovat, že se v dnešní době vyplatí tvořit aplikaci pro OS Android více než pro ostatní mobilní operační systémy, jakými jsou např. iOS, Symbian OS, Windows Phone 7 a jiné.

Dále byl mezi studenty prováděn ústní průzkum, zda by o takovou aplikaci měli zájem. Většina studentů by ji ocenila, vzhledem k tomu, že ne vždy je možné pro objednání jídla použít počítač.

2.5. Správa projektu

Pro zvýšení efektivity komunikace s konzultantem se nabízelo použít SW pro správu projektu, popř. pro správu verzí.

Pro správu projektu byl zvolen open-source[4] software Trac[5]. I přesto, že se nespolupracovalo na zdrojovém kódu, použití tohoto nástroje bylo výhodné. Byl použit pro hlášení změn, jak ve webových službách, tak v Android aplikaci, pro report chyb, pro poznámky ohledně dalšího vývoje a podobně.

V rámci nástroje Trac byl také využíván systém pro správu verzí. Vzhledem k tomu, že je nástrojem Trac implicitně podporovaný Subversion, byl použit právě tento software pro správu verzí. Nešlo o plné využití možností tohoto systému, sloužil spíše jako úložiště testovaných verzí či dokumentací webových služeb. Díky tomuto nástroji ale měly obě strany přehled o aktuálním vývoji i jeho historii.

[4]Označení počítačového softwaru s otevřeným zdrojovým kódem.

[5]Nástroj pro správu softwarových projektů s podporou hlášení chyb a způsobu dokumentace ve formě wiki (http://trac.edgewall.org/).

3. Cíle

Před vývojem aplikace byly vytyčeny cíle, které by aplikace měla splňovat. Jsou to bezpečnost, kompatibilita, přizpůsobitelnost, použitelnost, datová nenáročnost, robustnost, rozšiřitelnost a srozumitelnost. Jednotlivé cíle jsou popsány v následujících podkapitolách.

3.1. Bezpečnost

Jedním z důležitých cílů je bezpečnost. Aplikace bude propojena se serverem, na němž jsou účty tisíců strávníků. Za bezpečnost těchto účtů zodpovídá Správa kolejí a menz a dodavatelská společnost ANETE spol. s r.o. Tyto dva subjekty dbají na bezpečnost dat, proto bylo nutné se přizpůsobit.

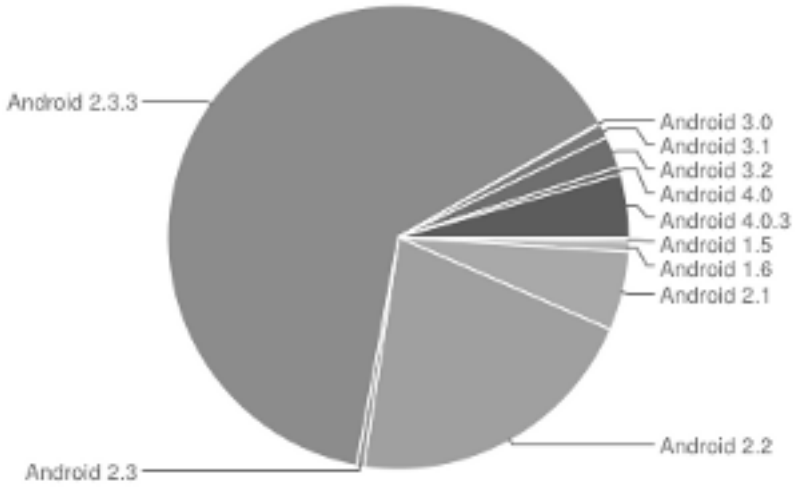
Základním prostředkem pro zabezpečení dat je použití šifrované komunikace se serverem pomocí protokolu HTTPS. To zajišťuje obranu před odposlechnutím komunikace. Navíc není možné získat odpověď na požadavek od nepřihlášeného klienta, jelikož v hlavičce každého požadavku jsou zasílány autentizační informace, které jsou před provedením akce vždy ověřeny.

3.2. Kompatibilita

Android OS je znám svou roztříštěností, jelikož je tento systém nasazen na velkém množství mobilních zařízení různých výrobců. Běží na zařízeních různých velikostí a rozlišení displejů, přičemž je žádoucí, aby jedna aplikace vypadala na všech zařízeních stejně, popřípadě aby na všech zařízeních plnila svůj účel. Navíc existuje více zpětně nekompatibilních verzí systému. Podle grafu na obrázku 4. znázorňujícího rozšíření jednotlivých verzí z 1. 5. 2012 je patrné, že největší podíl má Android OS verze 2.3.3. Na této verzi byla aplikace vyladěna, přitom ale byla vyvinuta a testována i pro ostatní verze.

Tohoto faktu si je vědom samotný tvůrce Android OS, proto dává vývojářům šanci vytvořit aplikaci tak, aby byla použitelná napříč různými zařízeními. Toho je docíleno například použitím virtuálních jednotek pro tvorbu uživatelského rozhraní nezávislého na rozlišení. Tento princip je popsán v [1].

K tomuto cíli bylo přihlédnuto v průběhu vývoje tak, aby byla aplikace použitelná na co nejvíce zařízeních. Během testování na fyzických i emulovaných zařízení, na kterých běží Android OS verze 2.1–4.0.4, nebyl zjištěn žádný problém. Vždy došlo ke správnému rozložení prvků uživatelského rozhraní a aplikace se tak přizpůsobila všem velikostem displeje, od nejmenšího na trhu až po tablety s Android OS.



Obrázek 4. Graf znázorňující rozšíření jednotlivých verzí Android OS

3.3. Přizpůsobitelnost

Jak bylo řečeno výše, aplikace bude používána na mnoha rozličných hardwarových konfiguracích určených zejména rozlišením a velikostí displejů. Proto má vývojář aplikace pro Android OS možnost rozdělit data do různých kategorií podle dané konfigurace. Jde například o grafické soubory, které jsou uložené v podadresářích podle rozlišení displeje. Pokud aplikace běží na displeji zařazeného do kategorie `ldpi` (nízké rozlišení), použijí se grafické soubory z adresáře tomu určenému. Další kategorie rozlišení displeje jsou `mdpi`, `hdpi` a `xhdpi`.

Podobně se zachází s řetězci. Ty jsou pro větší přizpůsobitelnost oddělené do samostatných XML souborů. Pokud bude v budoucnu aplikace používána na Slovensku, bude přeložení aplikace spočívat pouze v přidání dalšího XML souboru, namísto jiných, značně méně pohodlných řešení.

3.4. Použitelnost

Při tvorbě aplikace byla vyvinuta snaha o vytvoření uživatelsky přívětivého rozhraní, ve kterém se bude uživatel snadno orientovat. Byly použity standardní systémové komponenty, které by uživatel neměl mít problém ovládat. Byl dán velký důraz na UX[6] aplikace, tedy to, aby uživatel rád s aplikací pracoval a přesně věděl, co od aplikace očekávat.

[6]User Experience – překládáno jako „uživatelský prožitek“

3.5. Datová nenáročnost

Jak bylo již zmíněno v kapitole 2.1., u běžných datových tarifů je omezen datový objem. Proto je důležité, aby aplikace přenášela pouze nezbytný objem dat. Bylo potřeba docílit toho, aby aplikace přenášela řádově menší objem dat než ten, který je přenášen při objednávání přes internetovou aplikaci v běžném internetovém prohlížeči.

Proto, jak je uvedeno v kapitole 5.3., byl pro výměnu dat použit formát JSON namísto používanějšího XML. Z těchto důvodů jsou také veškeré požadavky a odpovědi zkomprimované pomocí softwaru Gzip.

3.6. Robustnost

Server je navržen tak, aby při každé očekávané či neočekávané chybě vrátil chybový HTTP kód. Aplikace na tyto kódy (konkrétně kódy 400 Bad Request a 500 Internal Server Error) umí reagovat a v případě chyby, kdy není možné dále pracovat s aplikací, dojde k odhlášení uživatele. V žádném případě aplikace nesmí spadnout při vzniku neočekávané výjimky.

Výjimka nastává, pokud server vrátí odpověď s chybovým kódem 400 Bad Request, ale zároveň speciální značkou indikující vypršení relace. V takovém případě se zobrazí dialog, který dá uživateli na výběr, zda se chce znovu přihlásit, či zda chce ukončit aplikaci.

3.7. Rozšiřitelnost

Serverová část i aplikace byla vyvíjena s předpokladem, že v budoucnu bude rozšířena o další klienty společnosti ANETE spol. s r.o. Jde zejména o podporu dalších univerzitních menz i jiných stravovacích zařízení. Předpokládá se, že aplikace by měla být schopná připojit se k serveru jakéhokoliv informačního systému Kredit. V aplikaci proto nejsou žádná data zabudována „natvrdo“, nýbrž se stahují ze serveru. Například druhy jídel (polévky, obědy, minutky, apod.) se stahují ze serveru, přičemž podle nich je posléze vykreslena nabídka.

Jedinou výjimkou jsou prozatím informace a fotky jednotlivých výdejen. Pro tato data se nestihly vytvořit webové služby, proto jsou zabudována v aplikaci. Vzhledem k záměru vyvinout obecnou aplikaci je však napravení této skutečnosti v plánu.

3.8. Srozumitelnost

Téměř každá třída i metoda je ve zdrojovém kódu aplikace popsána pomocí JavaDoc komentáře, který umožňuje jednoduchou tvorbu dokumentace. Slouží tak vývojáři například pro informaci, co která metoda vykonává, jakou má návratovou hodnotu a podobně.

4. Platforma Android

V této kapitole je popsána platforma Android, její historie a možnosti vývoje na této platformě. Informace vycházejí zejména z [2].

4.1. Historie

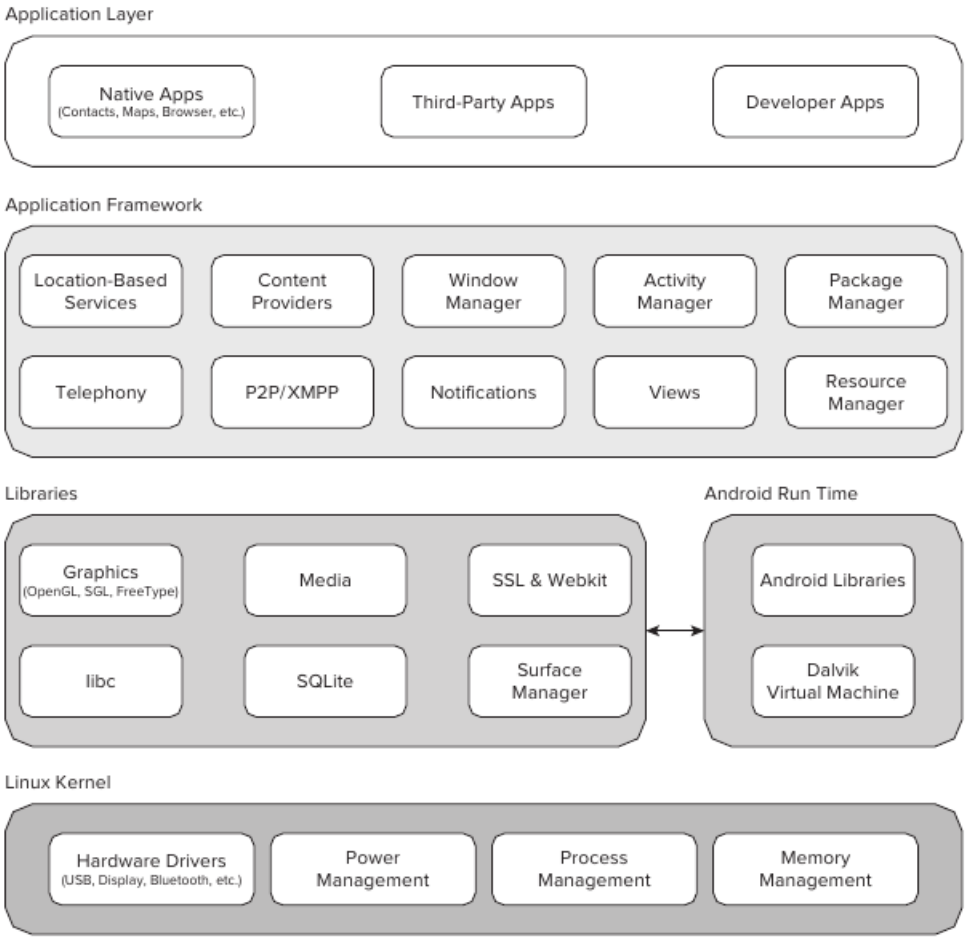
Počátky platformy Android sahají do roku 2003, kdy byla vytvořena v té době samostatná firma Android, Inc. Z této firmy udělal v roce 2005 dceřinou společnost Google Inc., který v té době hledal řešení pro mobilní trh. Později, v roce 2007, bylo sestaveno uskupení Open Handset Alliance. Toto uskupení se skládalo ze 34 (k dubnu 2012 bylo členů 84) výrobců mobilních zařízení, softwarových společností, mobilních operátorů a dalších. Tito členové v rámci Open Handset Alliance vytvořili společně produkt Android. Ten se skládá ze tří hlavních částí:

- Open-source operační systém pro mobilní zařízení.
- Open-source vývojářská platforma pro vývoj mobilních aplikací.
- Zařízení, na kterém běží operační systém Android a aplikace pro něj vytvořené.

4.2. Architektura systému

Architektura operačního systému Android OS je znázorněna na obrázku 5. Jednotlivé části jsou vypsány spolu se stručným popisem níže.

- **Linuxové jádro** poskytuje základní služby (hardwarové ovladače, správa paměti a procesů apod.). Jádro poskytuje abstraktní vrstvu mezi hardwarem a zbytkem architektury.
- **Knihovny** zahrnují např. libc, knihovny pro přehrávání audia a videa, OpenGL knihovny, SQLite, apod.
- **Běhové prostředí Android** je tím, díky čemu není Android pouze implementace linuxového systému pro mobilní zařízení.
  - Do této části patří **knihovny**, které jsou běžně dostupné Java vývojářům, ale také specifické knihovny určené pouze pro Android zařízení.
  - Patří sem zejména **Dalvik**, který je podobně jako Java Virtual Machine (dále JVM) určen pro zpracování zkompilovaného kódu (mezikódu), který není závislý na architektuře. Do tohoto mezikódu jsou zkompilovány vyvinuté aplikace, které jsou poté spuštěny ve virtuálním stroji Dalvik. Je zřejmé, že při běhu zařízení je aplikací v jeden okamžik spuštěno více. Proto je Dalvik naproti JVM navržen pro



Obrázek 5. Schéma architektury systému Android OS (převzato z [2])

efektivní běh více instancí virtuálního stroje zároveň. Dále je Dalvik naproti JVM optimalizovaný pro systémy, které jsou ve velké míře omezeny na paměť a rychlost procesoru. Je proto vhodný pro použití v embedded a mobilních zařízeních.

- **Aplikační rámec** poskytuje vývojářům prostředky pro hardwarovou abstrakci, tvorbu uživatelského rozhraní či přístup k datům.
- **Aplikační vrstva** na které jsou spouštěny jak nativní aplikace, tak aplikace třetích stran. Všechny aplikace jsou si rovnocenné, jelikož využívají stejné API.

4.3. Architektura aplikací

Hlavní myšlenka Android aplikací je znovupoužitelnost jednotlivých komponent. To je zajištěno pomocí sdílení jednotlivých částí aplikací. Aplikace se mohou

skládat z následujících částí:

- **Activity** je jednotlivá „obrazovka“ mající danou funkčnost. Hlavní prostředek pro interakci s uživatelem. Více informací o „aktivitách“ ve vývojářské příručce [3]. V textu je pro tento prostředek dále používán výraz „aktivita“.
- **Service** je aplikace pracující na pozadí, která neposkytuje uživatelské rozhraní.
- **Widget** je miniaplikace, kterou lze vložit do jiné aplikace. Nejčastěji jsou však widgety vkládány na domovskou obrazovku systému.

Díky principu sdílení aktivit si tak uživatel může vybrat, kterou aplikaci použije například pro zaslání e-mailové zprávy. Pokud vývojář vytvoří aplikaci s aktivitou, která může na takový požadavek reagovat, uživateli se při požadavku o odeslání e-mailu, zobrazí daná aktivita na výběr v nabídce. Tento princip byl použit pro propojení aplikace s aktivitou pro skenování QR kódů, které je popsané v kapitole 7.8.

4.4. Systémové komponenty

Pro běh aplikací se využívají následující systémové komponenty:

- **Activity Manager** řídí zásobník aktivit a jejich životní cyklus.
- **Views** jsou používány pro tvorbu uživatelského rozhraní. Spadají sem běžné komponenty, se kterými se lze setkat při programování uživatelského rozhraní na desktopu, např. textový popisek, editovatelné pole, tlačítko, a podobně.
- **Notification Manager** je správce systémových upozornění. Tato upozornění se zobrazují v upozorňovací liště v horní části obrazovky.
- **Content Providers** poskytují rozhraní pro ukládání a získávání dat z datové vrstvy. Jde například o telefonní seznam nebo obsah z databází typu SQLite.
- **Resource Manager** je správce zdrojů, který poskytuje aplikacím přístup k datům aplikace. Jde zejména o řetězce, obrazová data, audio data a podobně.

4.5. Vývoj software

Jako zdroj informací o vývoji software pro platformu Android OS byla použita publikace *Professional Android 2 Application Development* [2] a příručka pro vývojáře [4]. Google, Inc. poskytuje vývojářům balík nástrojů pojmenovaný souhrnně Android SDK[7], který se skládá z následujících částí:

- **Android API** jsou knihovny, které vývojáři poskytují přístup k funkcím systému. Stejné knihovny používá Google, Inc. pro vývoj nativních aplikací.
- **Vývojářské nástroje** pomocí kterých je možné zkompilovat a ladit aplikace pro platformu Android.
- **Správce virtuálních zařízení a emulátor** pomocí kterého je možné simulovat běh aplikace na různých konfiguracích hardwaru. Vývojář tak může zjistit, jak se aplikace chová na velkém počtu různých zařízení.
- **Dokumentace** ve které se nachází mnoho informací o vývoji aplikací společně s příklady použití. Velká část dokumentace je obsažena v internetové příručce pro vývojáře [4].

[7] Akronym anglického Software Development Kit se používá ve smyslu balíku nástrojů určených pro vývoj software.

5. Webové služby

Během vývoje aplikace konzultant pracoval na webových službách pro komunikaci aplikace (klienta) se serverem. Webové služby byly postupně testovány v aplikaci a průběžně upravovány pro efektivnější komunikaci. Konečný seznam vytvořených webových služeb je zobrazen v tabulce 1. Jelikož informační systém pro správu menzy běží na webovém serveru IIS[8], bylo pro webové služby použito řešení založené na WCF[9].

5.1. Princip

Jako rozhraní webových služeb bylo použito REST[10]. Každá webová služba má dané URI, přes který se ke službě přistupuje. Používají se k tomu metody HTTP GET, POST, PUT a UPDATE. Pro každý požadavek vrátí server odpověď ve vybraném formátu. Webový server podporuje odpovědi ve formátu XML nebo JSON.

Tabulka 1. Seznam použitých webových služeb

HTTP metoda	Jméno	Popis
POST	/login	Přihlásí uživatele
POST	/logout	Odhlásí uživatele
GET	/account/info	Získá informace o přihlášeném uživateli
GET	/account/balance	Získá zůstatky přihlášeného uživatele
GET	/account/history	Získá historii pro zadaný rozsah dat
GET	/orderingdates	Získá platná data pro objednání
GET	/canteens	Získá výdejny pro zadané datum
GET	/menu	Získá nabídku pro zadané datum a výdejnu
GET	/orders	Získá objednávky pro zadaný rozsah dat
PUT	/singleorder	Provede objednávku zadané položky

[8]Internet Information Services – webový server vytvořený společností Microsoft

[9]Windows Communication Foundation – API v Microsoft .NET Framework pro vytváření serverově-orientovaných aplikací

[10]Representational State Transfer – architektura webových aplikací pro jednoduchý přístup ke zdrojům dat

5.2. Příklad komunikace

Požadavek na webovou službu `/menu` pro získání nabídky jídel pro datum 26. 4. 2012 a výdejnu s identifikačním číslem 1:

```
GET /menu?date=2012-04-26T00:00:00&idCanteen=1 HTTP
```

Odpověď ve formátu JSON:

```
[{"HasSubsidy":true,"IdMenuItem":1,"MealAlt":1,
"MealAltDescription":"Polévka kuřecí s masem a těstovinami",
"MealKind":1,"MealKindDescription":"Oběd - polévka",
"Price":10.08},

...

{"HasSubsidy":true,"IdMenuItem":12,"MealAlt":2,
"MealAltDescription":"Hovězí maso na žampiónech",
"MealKind":4,"MealKindDescription":"Večeře - hlavní",
"Price":23.95}]
```

5.3. Formát odpovědi

Webovému serveru lze předat informace o tom, v jakém datovém formátu je požadována odpověď. Podporován je formát JSON nebo používanější XML. Následuje ukázka odpovědi na požadavek z předchozí části ve formátu XML:

```
<ArrayOfMenuItem>
  <MenuItem>
    <HasSubsidy>true</HasSubsidy>
    <IdMenuItem>1</IdMenuItem>
    <MealAlt>1</MealAlt>
    <MealAltDescription>
      Polévka kuřecí s masem a těstovinami
    </MealAltDescription>
    <MealKind>1</MealKind>
    <MealKindDescription>
      Oběd - polévka
```

```
    </MealKindDescription>
    <Price>10.08</Price>
  </MenuItem>

...

  <MenuItem>
    <HasSubsidy>true</HasSubsidy>
    <IdMenuItem>12</IdMenuItem>
    <MealAlt>2</MealAlt>
    <MealAltDescription>
      Hovězí maso na žampiónech
    </MealAltDescription>
    <MealKind>4</MealKind>
    <MealKindDescription>
      Večeře - hlavní
    </MealKindDescription>
    <Price>23.95</Price>
  </MenuItem>
</ArrayOfMenuItem>
```

Je patrné, že pomocí formátu JSON je možné přenést stejné informace pomocí menšího objemu dat. Zejména díky tomu, že název atributu se v každé položce vyskytuje pouze jednou. V případě XML se tento název vyskytuje v každé položce dvakrát, v otevírací značce i v uzavírací značce. Při srovnání formátů pro získání nabídky (webová služba `/menu`) se jedná zmenšení objemu přenesených dat až o 40 %. Proto byl pro výměnu informací se serverem použit právě formát JSON.

5.4. Způsob autentizace

Autentizace klienta probíhá pomocí tzv. „Basic“ autentizace popsané v RFC 2617 [5]. Zaslání přihlašovacích údajů je zajištěno přidáním položky `Authorization` do hlavičky HTTP požadavku. Hodnota této položky má formát `Basic base64(login:password)`. Funkce base64 znamená zakódování těla algoritmem Base64, který převádí binární data do podoby ASCII znaků, které je možné bez ztráty informací zaslat v HTTP hlavičce. Přihlašovací údaje jsou jednoduše zjistitelné, nicméně jelikož probíhá komunikace se serverem v šifrované podobě pomocí protokolu HTTPS, je možné si to dovolit.

6. Programátorská dokumentace

V této části je nastíněna struktura aplikace. Popíšeme si strukturu projektu a jednotlivé kategorie tříd. Také popíšeme, jak pracuje QR kódování a dekódování pro objednávání jídel pomocí oskenování QR kódu.

6.1. Struktura projektu

Adresářová struktura projektu se skládá z následujících položek:

- **src/** obsahuje třídy v jazyku Java rozdělené do podadresářů (balíčků) dle konvence jazyka Java.
- **gen/** obsahuje vygenerované informace mapující externí data na proměnné, které lze použít ve zdrojovém kódu.
- **assets/** obsahuje jediný soubor, a to `canteens.xml`, který obsahuje informace o jednotlivých výdejnách.
- **bin/** obsahuje do mezikódu zkompilovanou aplikaci s příponou `dex` (Dalvik Executable). Ta je ve formátu připraveném pro běh ve virtuálním stroji Dalvik (více o Dalvik v kapitole 4.2.). Dále se zde nachází soubor s příponou `apk` (Application Package File), což je archiv obsahující jak výše zmíněný soubor `.dex`, tak obrazová a další data potřebná pro běh aplikace. Tento formát je založený na formátu balíčků jazyka Java `.jar` (Java Archive) a je připravený pro instalaci na mobilním zařízení.
- **res/** obsahuje data potřebná pro běh aplikace. Mezi ně patří například obrazová data, definice stylů, řetězců a podobně. Více informací v kapitole 6.1.1.
- **AndroidManifest.xml** je základní konfigurační soubor projektu obsahující informace o názvu a verzi aplikace, minimální verzi API pro spuštění aplikace, seznam povolení přístupu k systémovým zdrojům a seznam jednotlivých aktivit, z kterých aplikace sestává.

6.1.1. Adresář /res

V adresáři /res se nalézají veškerá data potřebná pro běh aplikace. Jsou to obrazové soubory (zejména ikony, fotografie), definice uživatelského rozhraní ve formátu XML a další. Tato data jsou uspořádána do následující striktně dané struktury podadresářů:

- **drawable/** obsahuje obecné XML soubory, jejichž použití není závislé na aktuálním rozlišení. Jsou zde definovány grafické prvky, např. oddělovače

seznamů nebo vzhled části s informacemi o uživateli. Tyto grafické prvky jsou definovány samostatně pro jednotlivá témata (více informací v kapitole 7.9.).

- **drawable-hdpi/** obsahuje obrazová data, zejména ikony, použitá na zařízeních s vysokým rozlišením displeje.
- **drawable-ldpi/** obsahuje obrazová data, zejména ikony, použitá na zařízeních s nízkým rozlišením displeje.
- **drawable-mdpi/** obsahuje obrazová data, zejména ikony, použitá na zařízeních se středním rozlišením displeje.
- **drawable-xhdpi/** obsahuje obrazová data, zejména ikony, použitá na zařízeních s velmi vysokým rozlišením displeje (tablety).
- **layout/** obsahuje XML soubory definující uživatelské rozhraní jednotlivých aktivit a dialogů.
- **menu/** obsahuje XML soubory definující strukturu nabídek.
- **raw/** obsahuje tzv. „surová“ data (raw data). Zde se jedná o soubor certifikátu serveru pro šifrované připojení pomocí protokolu HTTPS.
- **values/** obsahuje XML soubory, jejichž účel je následující:
  - **arrays.xml** obsahuje definice polí ve formátu XML. Zde jsou definované dvě pole použité pro naplnění seznamu pro výběr tématu aplikace.
  - **attrs.xml** obsahuje definice vlastních atributů ve formátu XML. Ty lze použít při návrhu uživatelského rozhraní a poté přepsat v rámci jednotlivých témat aplikace v souboru *styles.xml*.
  - **colors.xml** obsahuje definice vlastních pojmenovaných barev ve formátu XML.
  - **strings.xml** obsahuje definice všech lokalizovatelných textů ve formátu XML. Ty jsou v rámci přizpůsobitelnosti pro lepší správu textů odděleny od zdrojových kódů. Pokud se tak bude v budoucnu přidávat podpora jiných jazyků (např. slovenštiny, kterou internetová aplikace WebKredit podporuje), stačí vytvořit další XML soubor s přeloženými texty, který aplikace automaticky použije v případě nastaveného systémového jazyka na slovenštinu.
  - **styles.xml** obsahuje definice jednotlivých témat aplikace. Jsou zde přepsány systémové atributy jako například barva pozadí, ale i vlastní atributy vytvořené v souboru *attrs.xml*.

- **xml/** obsahuje jediný soubor *preferences.xml*, jež definuje strukturu nabídky pro nastavení aplikace.

6.2. Rozdělení a popis tříd

Použitím objektového paradigmatu bylo možné jednoduše oddělit jednotlivé celky kódu do zhruba 40 tříd. Ty lze zařadit do následujících kategorií.

6.2.1. Aktivity

Aktivity představuji jednotlivé hlavní obrazovky aplikace, kterými jsou nabídka jídel, seznam objednávek, historie účtu a přihlašovací obrazovka. Tyto třídy jsou potomky třídy `android.app.Activity` z Android SDK. Následuje seznam aktivit a jejich stručný popis:

- **MainActivity** je hlavní aktivita obsahující záložky ve formě dalších aktivit, tj. `MenuListActivity`, `OrdersListActivity`, `HistoryListActivity`. Tato aktivita je spuštěná po celou dobu uživatelovy činnosti s přihlášeným stravovacím účtem. Proto je také pomocí této aktivity vykreslována lišta s informacemi o uživateli, která je tak zobrazena vždy, když je uživatel přihlášen.
- **LoginActivity** je přihlašovací obrazovka, pomocí které se po úspěšném přihlášení spouští aktivita `MainActivity`.
- **MenuListActivity** je aktivita pro zobrazení nabídky spouštějící se v záložce aktivity `MainActivity`.
- **OrdersListActivity** je aktivita pro zobrazení objednávek spouštějící se v záložce aktivity `MainActivity`.
- **HistoryListActivity** je aktivita pro zobrazení historie účtu spouštějící se v záložce aktivity `MainActivity`.

6.2.2. Dialogy

Pro zobrazení běžných upozorňovacích dialogů lze použít systémové funkce. Složitější dialogy bylo ale nutné vytvořit vlastní. Jedná se například o dialog výběru data, seznamu výdejen nebo dialog indikující vypršení relace. Většina dialogů dědí funkčnost z třídy `CancelableByClickDialog`, která umožňuje uzavřít dialog kliknutím mimo něj nebo do jeho obsahu. To je vhodné z hlediska uživatelského rozhraní. Následuje seznam dialogů a jejich stručný popis:

- **AboutDialog** je dialog zobrazující informace o aplikaci.

- **CancelableByClickDialog** je rodičovská aktivita většiny dialogů reprezentována abstraktní třídou. Jsou zde přepsány metody pro zavření dialogu dotykem mimo dialog nebo naopak do určené komponenty v dialogu.
- **CanteenInfoDialog** je dialog zobrazující informace o výdejně.
- **CanteenListDialog** je dialog určený pro výběr výdejny nebo pro otevření dialogu s informacemi o výdejně.
- **DateDialog** je dialog určený pro výběr data.
- **HistoryItemDialog** je dialog zobrazující detailní informace o položce ze seznamu historie.
- **LoginHelpDialog** je dialog zobrazující nápovědu „Jak se přihlásit“.
- **MenuItemDialog** je dialog zobrazující informace o položce z nabídky určený pro její objednání.
- **OrdersItemDialog** je dialog zobrazující detailní informace o položce ze seznamu objednávek.
- **PreferencesDialog** je dialog pro nastavení aplikace.
- **QRDialog** je dialog pro zobrazení informací o funkcionalitě skenování QR kódů, zjištění, zda je nainstalována potřebná aplikace a samotný start skenování QR kódu.
- **RangeDialog** je dialog určený pro nastavení rozmezí historie a objednávek.
- **SessionLostDialog** je dialog zobrazující se po ztrátě relace, která nastane po vypršení časového limitu nebo po přihlášení do účtu z jiného klienta. Uživatel má v takové situaci na výběr, zda se přihlásí do účtu znovu nebo zda aplikaci ukončí.
- **UserInfoDialog** je dialog zobrazující informace o uživateli. Pomocí tohoto dialogu je možné přepnout uživatele.

6.2.3. Datové třídy

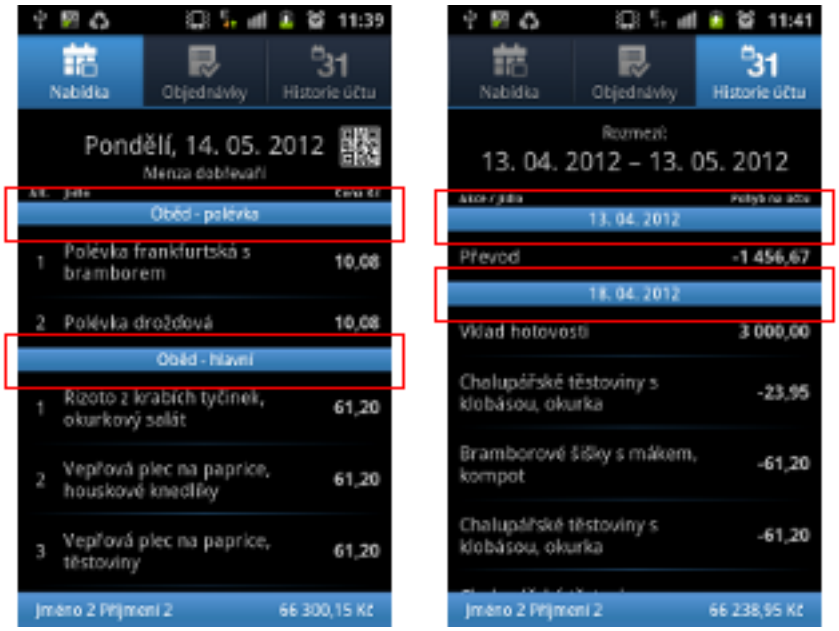
Datové třídy reprezentují jednotlivé datové celky, jejichž atributy jsou získány ze vzdáleného serveru. S těmito objekty se pracuje při vykreslení jednotlivých seznamů a při zpracování výběru jednotlivých položek. Následuje seznam datových tříd:

- **Canteen** reprezentuje výdejnu.
- **HistoryItem** reprezentuje položku historie.

- **MenuItem** reprezentuje položku v nabídce.
- **OrdersItem** reprezentuje položku objednávky.

6.2.4. Adaptéry

Adaptéry jsou třídy, jejichž instance se používá jako prostředník mezi kolekcí položek a UI komponentou reprezentující seznam. Jsou použity z důvodu potřeby vykreslovat jednotlivé položky kolekce pomocí vlastního rozložení prvků. Všechny adaptéry mimo adaptér pro seznam výdejen byly navíc použity pro vykreslení oddělovacích položek, které rozdělují položky seznamu do více sekcí. Tyto sekce jsou v seznamu nabídky za účelem oddělení druhů jídel a v seznamu objednávek a historie za účelem oddělení položek dle dat. Pro znázornění jsou sekce zobrazeny s červeným orámováním na obrázku 6. Následuje seznam adaptérů a jejich popis:



Obrázek 6. Znázornění sekcí v jednotlivých seznamech

- **CanteenListAdapter** propojuje kolekci výdejen se seznamem v dialogu pro výběr výdejny. Vytvořen z důvodu možnosti vykreslení tlačítka pro zobrazení dialogu s informacemi o výdejně.
- **MenuItemsAdapter** propojuje kolekci položek nabídky se seznamem. Vytvořen za účelem vykreslení položky nabídky dle vlastního návrhu zobrazujícího číslo alternativy, název alternativy a cenu.

- **OrdersItemAdapter** propojuje kolekci položek objednávek se seznamem. Vytvořen za účelem vykreslení položky objednávek dle vlastního návrhu zobrazujícího název objednávky, druh jídla a název výdejny.
- **HistoryItemsAdapter** propojuje kolekci položek historie účtu se seznamem. Vytvořen za účelem pro vykreslení položky historie účtu dle vlastního návrhu zobrazujícího název pohybu a částku.

6.2.5. Výjimky

Aplikace používá pro ošetření výjimečných stavů systém výjimek. Dvě třídy `BadRequestFaultException` a `InternalServerErrorException` jsou vytvořené výjimky reprezentující chybové kódy protokolu HTTP, konkrétně `400 Bad Request` a `500 Internal Server Error`. Server v takovém případě vrací více informací pro zjištění příčiny, proto nebylo vhodné použít běžnou třídu výjimky `java.lang.Exception`. Výjimka `BadRequestFaultException` uchovává tři chybové kódy, výjimka `InternalServerErrorException` uchovává dva chybové kódy.

Další třída byla použita pro pro ošetřování výjimek vzniklých v asynchronních úlohách. Třída `ExceptionsHandler` deklaruje abstraktní metodu, jejíž tělo je ošetřeno proti případným výjimkám tak, že vrací objekt nebo konstantu výjimku reprezentující. Důvodem pro vytvoření této třídy byla redundance kódu vzniklá několikerým opakováním obsáhlého příkazu `try ... catch`, pomocí kterého byl ukládán do proměnné kód vzniklé výjimky. Toto ukládání do proměnné bylo potřeba provést v těle několika asynchronních úloh. Vlákno, které tyto asynchronní úlohy vykonává nemá povoleno pracovat s uživatelským rozhraním, proto nebylo možné ošetřovat výjimky přímo. V tomto vlákně bylo zapotřebí vrátit kód vzniklé výjimky a následně ji ošetřit ve vlákně obsluhující uživatelské rozhraní. Použití třídy `ExceptionsHandler` spočívá ve vytvoření objektu, definování těla a spuštění vykonávání asynchronní úlohy, která je tak odolná proti případným výjimkám. Po vykonání tak dostaneme objekt nebo konstantu určující výsledek akce.

6.2.6. Asynchronní úlohy

Některé webové služby jsou datově i časově velmi náročné. Aby během vykonávání úloh reagovalo uživatelské rozhraní na činnost uživatele, bylo nutné použít pro komunikaci s webovými službami asynchronní zpracování úloh.

Většinu asynchronních úloh bylo třeba reprezentovat statickými třídami. Jedině tak je možné, aby prováděly činnost, tedy čekaly na odpověď serveru, i při restartování aplikace, které vzniká při změně konfigurace zařízení. To nastává například při otočení telefonu či vysunutí/zasunutí klávesnice. Toto chování je popsáno ve vývojářské příručce [6]. I přesto však bylo nutné vytvořit kód, který tyto asynchronní úlohy odpojuje od ukončující se aplikace a následně připojuje ke

startující se aplikaci. Tento princip zajišťuje třída `RotationAwareTask`, ze které všechny asynchronní úlohy dědí.

- **RotationAwareTask** je abstraktní třída, jež deklaruje metody pro použití asynchronních úloh, které jsou odolné proti změně konfigurace, tj. otočení telefonu, vysunutí/zasunutí klávesnice a podobně. K aktivitě je tak přistupováno pouze pokud je v asynchronní úloze dostupná. Pokud dostupná není (v okamžiku, kdy se aplikace následkem změny konfigurace restartuje), další práce s ní se odloží do chvíle, než bude k úloze po restartování opět připojena.
- **LoginTask** je statická třída reprezentující asynchronní úlohu přihlášení.
- **LogoutTask** je třída reprezentující asynchronní úlohu odhlášení.
- **LoadMenuTask** je statická třída reprezentující asynchronní úlohu získání nabídky.
- **MakeOrderTask** je statická třída reprezentující asynchronní úlohu provedení objednávky.
- **LoadOrdersTask** je statická třída reprezentující asynchronní úlohu získání seznamu objednávek.
- **LoadHistoryTask** je statická třída reprezentující asynchronní úlohu získání historie účtu.

6.2.7. Ostatní třídy

Tyto třídy slouží ke speciálnímu účelům, které jsou níže vysvětleny.

- **Authenticator** je třída uchovávající informace o přihlášeném uživateli, která obsahuje metody pro jeho přihlášení, odhlášení, kódování a dekódování přihlašovacích informací a podobně.
- **Base64Wrapper** je třída zajišťující výběr dostupné možnosti pro kódování do formátu Base64 a zpětné dekódování z formátu Base64. Při inicializaci je zjištěno, zda se v systému nachází třída `android.util.Base64`. Pokud ano, pro další práci se použije reflexe pro vyvolání nativních metod. Pokud ne, použije se třída `cz.novaklukas.droidkredit.Base64Compatibility` získaná z vyšších verzí Android API.
- **Base64Compatibility** je třída získaná z vyšších verzí Android API. Je použita v případě běhu aplikace na Android OS verze 2.1, kde není k dispozici systémová třída `android.util.Base64`.

- **Constants** je třída pro uchování konstant používaných v rámci celé aplikace.
- **DataSource** je třída, ve které jsou definovány statické metody pro komunikaci s webovými službami na vzdáleném serveru.
- **Functions** je třída obsahující statické metody, které jsou používány napříč celou aplikací.
- **HttpsClient** je třída reprezentující klienta pro připojení ke vzdálenému serveru pomocí HTTPS protokolu. Tento klient přijímá pouze certifikát předaný pomocí konstruktoru.
- **HttpsClientTrustAll** je třída reprezentující klienta pro připojení ke vzdálenému serveru pomocí HTTPS protokolu. Tento klient považuje všechny přijaté certifikáty serverů za důvěryhodné.
- **JSONParser** je třída, ve které jsou definovány statické metody určené pro získání dat z JSON formátu.

6.3. Popis QR kódování

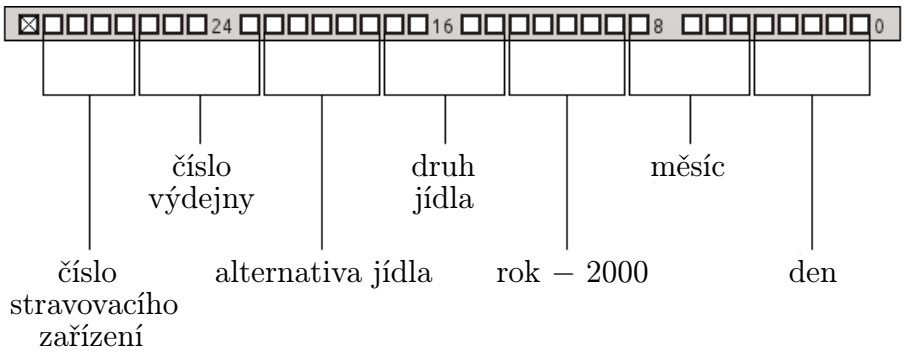
V aplikaci byla vytvořena funkce pro alternativní objednání pomocí oskenování QR kódu. Příklad QR kódu je znázorněn na obrázku 7. QR kód je dvojrozměrný kód zapisovaný do čtverce, který sestává z mnoha bodů dvou různých barev společně s prvky pro korektní zaměření tohoto kódu. V této kapitole si popíšeme, jak je kódování jídla do QR kódu řešeno.

QR kód umožňuje zakódovat řadu speciálních typů informací, od elektronické vizitky po internetovou adresu. Pro účely aplikace ale bylo vhodné pouze zakódování čistého textu. Existuje několik druhů velikostí QR kódu, nicméně vzhledem ke snímačům mobilních zařízení bylo vhodné použít ten nejmenší, který má rozlišení 21x21 bodů. Dále je možné využít různé úrovně korekce chyb, kvůli nimž je potřebný větší QR kód, ale na druhou stranu je výsledný QR kód odolnější proti poškození či chybě při skenování.

Bylo potřebné vytvořit takové kódování, aby bylo možné zakódovat sedm různých údajů, a to: den, měsíc, rok, druh jídla, alternativa jídla, identifikátor výdejny a identifikátor stravovacího zařízení. Všechny údaje lze zapsat pomocí číselné hodnoty. Tyto údaje by bylo možné společně s oddělovači zapsat jako řetězec. Tento řetězec by však byl velmi dlouhý, díky čemuž by nebylo možné použít QR kód s rozlišením 21x21 bodů.

Proto bylo zvoleno zakódování těchto údajů do 32 bitového čísla, které je v desítkové soustavě maximálně 10místné. Tak je možné využít zakódování do QR kódu s nejmenším rozlišením a s nejvyšší mírou korekce chyb. Kódování a dekódování je provedeno pomocí bitových posunů dle schématu na obrázku 8.

Obrázek 7. Ukázka QR kódu

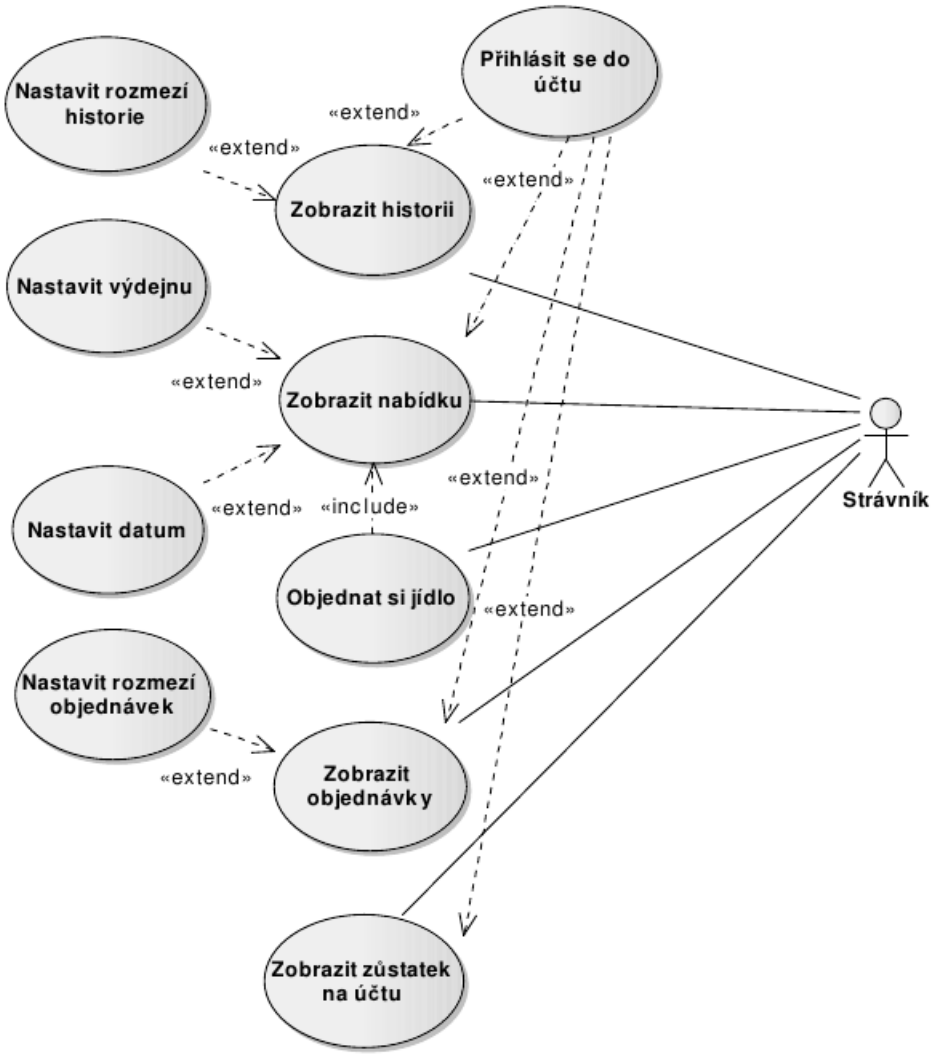


Obrázek 8. Schéma pro kódování a dekódování QR kódu

7. Uživatelská dokumentace

V této části naleznete uživatelskou dokumentaci aplikace, což je popis funkcí aplikace směřovaný k uživateli. V následujících kapitolách jsou popsány funkce aplikace spolu s odpovídajícími snímky obrazovek.

Na obrázku 9. je znázorněn diagram případů užití. V něm je přehledně zobrazeno, co aplikace uživateli nabízí.



Obrázek 9. Diagram případů užití aplikace

7.1. Přihlášení

Pro práci s aplikací je nutné se přihlásit do stravovacího účtu. Ihned po startu aplikace se zobrazí přihlašovací obrazovka (obr. 10.), kde je nutné vyplnit uživa-

telské jméno a heslo. Pomocí zaškrtávátka je možné aplikaci nechat zapamatovat přihlašovací údaje. V takovém případě se při dalším spuštění aplikace automaticky přihlásí.

Obrázek 10. Přihlašovací obrazovka aplikace

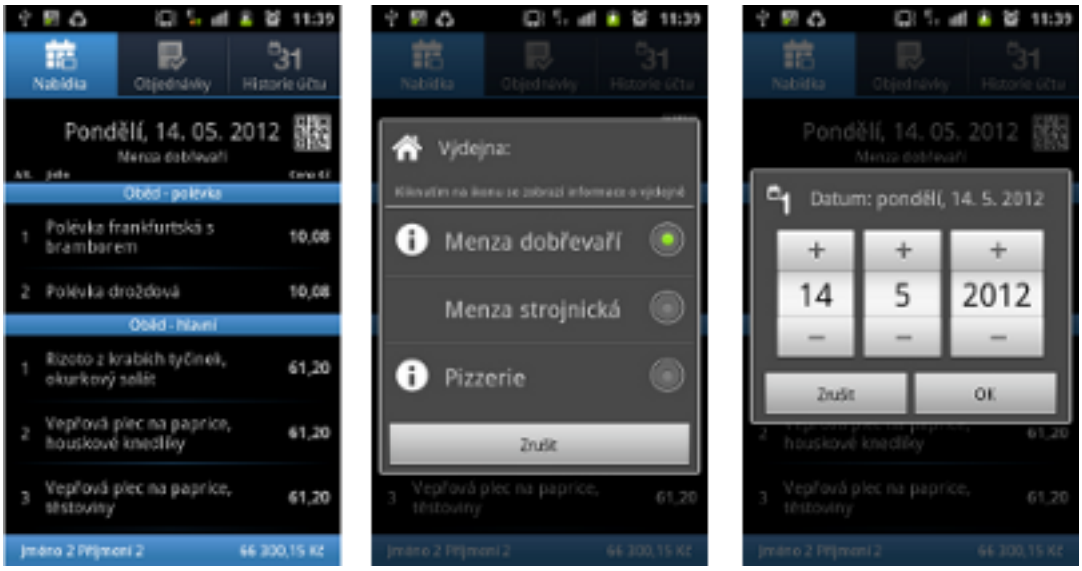
Většina strávníků využívá pro přístup ke správě stravovacího účtu univerzitní informační systém Portál Univerzity Palackého (dále jako Portál UP). Do tohoto systému je integrována internetová aplikace pro správu stravovacího účtu WebKredit, což má za důsledek, že uživatel přihlášený přes Portál UP se dostane do správy stravovacího účtu bez dalšího přihlášení. Jelikož by ale propojení s dalším informačním systémem bylo, pokud vůbec reálné, časově velmi náročné, bylo výhodnější využít v aplikaci standardní autentizaci společnosti ANETE spol. s r.o. Díky tomu je nutné se do aplikace přihlásit přihlašovacími údaji stejnými jako při přihlášení přes adresu `http://menza.upol.cz` – to znamená bez spolupráce s Portálem UP. Tyto přihlašovací údaje získal každý strávník při prvním převodu hotovosti na stravovací účet. Pokud takové údaje již nemá k dispozici, je možné si je na pokladně menzy UP opětovně vygenerovat. Tímto ale neztrácí přístup do správy stravovacího účtu přes Portál UP.

Díky tomuto (pro studenty nestandardnímu) způsobu přihlášení je po dvou a více neúspěšných pokusech otevřen dialog s těmito informacemi. Tento dialog lze také vyvolat výběrem položky v nabídce.

7.2. Zobrazení nabídky

Pomocí aplikace je možné si zobrazit nabídku vybraného data a výdejny (obr. 11. vlevo). Aplikace si po spuštění stáhne seznam platných dat pro přihlášeného uživatele. Implicitně je nastavené datum dle systémového času. Pokud toto datum není v seznamu platných dat, zobrazí se dialog (obr. 11. vpravo) pro jeho změnu. Tento dialog lze vyvolat posléze z nabídky.

Po potvrzení data se stáhne seznam platných výdejen pro vybrané datum. Pokud byla dříve použita výdejna, která je v seznamu platných výdejen, použije se. V opačném případě se použije implicitní výdejna uživatele, jejíž identifikátor se stáhne ze serveru. Změnit výdejnu lze posléze podobně jako datum pomocí vyvolání dialogu (obr. 11. uprostřed).

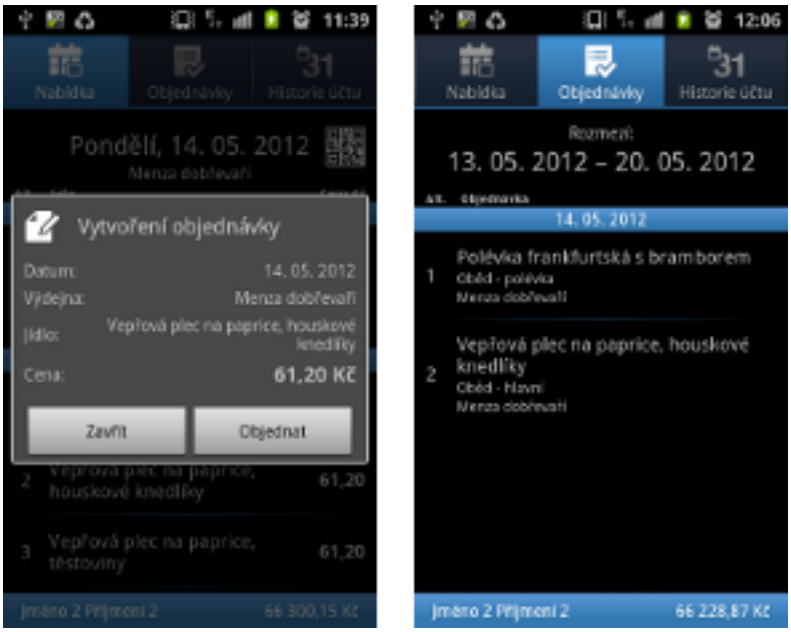


Obrázek 11. Obrazovka nabídky (vlevo) a dialogy pro výběr výdejny (uprostřed) a data (vpravo)

7.3. Objednání jídla

Až po výběru platného data a výdejny je zobrazena nabídka. Z této nabídky si může uživatel vybrat kterékoliv jídlo a následně ho objednat pomocí objednávacího dialogu (obr. 12. vlevo).

Na serverové straně je zajištěno, že v nabídce se budou zobrazovat pouze jídla, která jsou uživateli dostupná. Pokud se však během zobrazením nabídky a objednáním stane jídlo pro uživatele nedostupné (např. vyčerpáním porcí) zobrazí se chybový dialog s vysvětlující informací získanou ze serveru. Objednat jídlo lze také pomocí oskenování QR kódu, viz kapitolu 7.8.



Obrázek 12. Dialog pro vytvoření objednávky (vlevo) a obrazovka se seznamem objednávek (vpravo)

7.4. Zobrazení objednávek

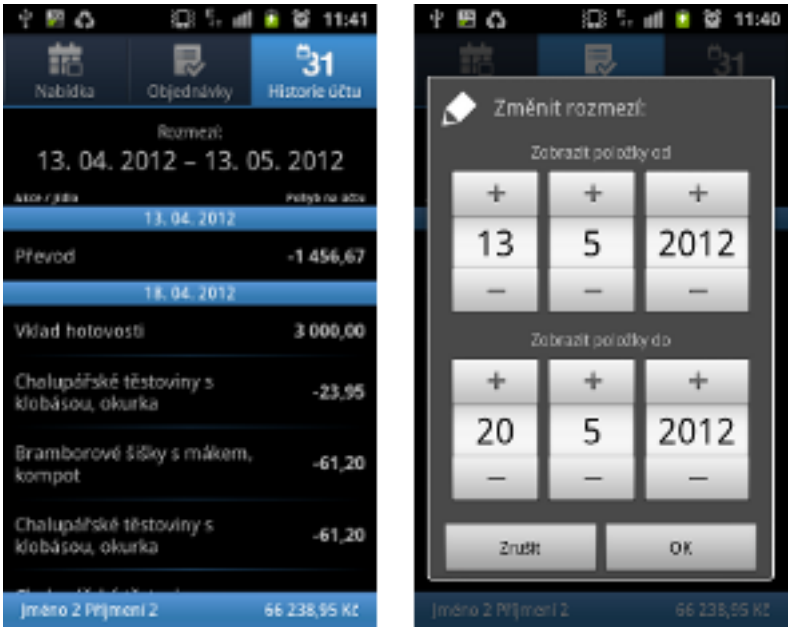
V aplikaci lze kliknutím na druhou záložku zobrazit seznam objednávek (obr. 12. vpravo), které prozatím nejsou vydány. Kliknutím na položku v seznamu objednávek se zobrazí dialog zobrazující detailní informace.

Tomuto seznamu lze nastavit rozmezí dat pomocí dialogu (obr. 13. vpravo) přístupného přes nabídku. Dialog obsahuje dvě komponenty pro výběr data, které jsou navzájem propojené tak, aby nebylo možné vybrat záporný počet dnů.

7.5. Zobrazení historie

Podobně jako zobrazení objednávek lze také zobrazit historii účtu (obr. 13. vlevo). Zde jsou uvedeny veškeré účetní operace, což jsou vklady na účet, převody mezi měsíci ale také jednotlivé objednané položky. Kliknutím na položku v historii se otevře dialog (obr. 14.) zobrazující detailní informace o položce.

Stejně jako v zobrazení objednávek je zde dostupný dialog (obr. 13. vpravo) pro výběr rozmezí, který také zabraňuje vybrat záporný počet dnů historie.



Obrázek 13. Obrazovka zobrazující historii účtu (vlevo) a dialog pro výběr rozmezí (vpravo)

Obrázek 14. Dialog zobrazující informace o položce historie

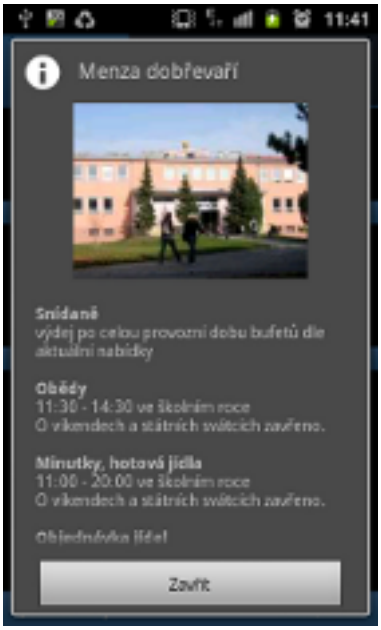
7.6. Zobrazení informací o uživateli

Napříč celou aplikací je u spodního okraje lišta zobrazující informace o přihlášeném uživateli. Tato lišta zobrazuje jméno a disponibilní zůstatek.

Po kliknutí na tuto lištu je vyvolán dialog „Informace o uživateli“, který navíc zobrazuje běžný zůstatek. Pomocí dialogu je taky možné přepnout uživatele, čímž dojde k odhlášení stávajícího a přechodu na přihlašovací obrazovku.

7.7. Zobrazení informací o výdejně

V dialogu pro výběr výdejny je u položky, u které jsou k dispozici informace, zobrazena ikona „i“. Pomocí této ikony lze vyvolat dialog (obr. 15.) s fotografií a informacemi o výdejně. Tyto informace se hodí například pro zjištění doby výdeje jídel, otevírací doby kanceláře či popisu, jak se k výdejně dostat.



Obrázek 15. Dialog s informacemi o výdejně

7.8. Získání objednávky oskenováním QR kódu

Mimo běžné objednání pomocí seznamu nabídky je v aplikaci přítomna funkce pro objednávání pomocí QR kódu. Tato funkce je připravena na to, aby byly ve výdejnách na jídelníčcích u jednotlivých jídel QR kódy. V takovém případě si nebude muset uživatel hledat jídlo v seznamu nabídky na malém displeji mobilního zařízení, ale jednoduše vyfotí QR kód a potvrdí objednávku. Od několika uživatelů bylo potvrzeno, že by běžné hledání v nabídce mohlo být zdlouhavé a právě tímto způsobem se dá zjednodušit.

Pro tuto funkcionalitu byl využit princip spolupráce aplikací v systému OS Android. Zjednodušeně řečeno lze z aplikace zavolat aktivitu jiné aplikace s určitým požadavkem, který má daná aktivita nadefinovaný. Volaná aktivita není nijak funkčně omezena, je možné v ní provést i komplikovanou akci. Před ukončením může navíc předat data volající aktivitě. V našem případě je zavolána aktivita SCAN, která vyvolá skenování obrazu (obr. 16. vpravo) a po oskenování QR kódu vrátí volající aktivitě data, která jsou v QR kódu zakódována.

Do módu skenování QR kódu se lze dostat kliknutím na ikonu QR kódu v pravém horním rohu obrazovky zobrazující nabídku. Tím se otevře dialog (obr. 16. vlevo) s informacemi o této funkcionalitě. Pokud má uživatel nainstalovanou aplikaci `Barcode Scanner`, která je ke skenování QR kódů v aplikaci použitá, zobrazí se tlačítko pro inicializaci skenování. V opačném případě se zobrazí upozornění o nutnosti nainstalovat aplikaci a tlačítkem pro spuštění produktové stránky v systémovém obchodě s aplikacemi Google Play (dříve Android Market).

Obrázek 16. Dialog pro inicializaci QR kódu (vlevo) a jeho následné skenování (vpravo)

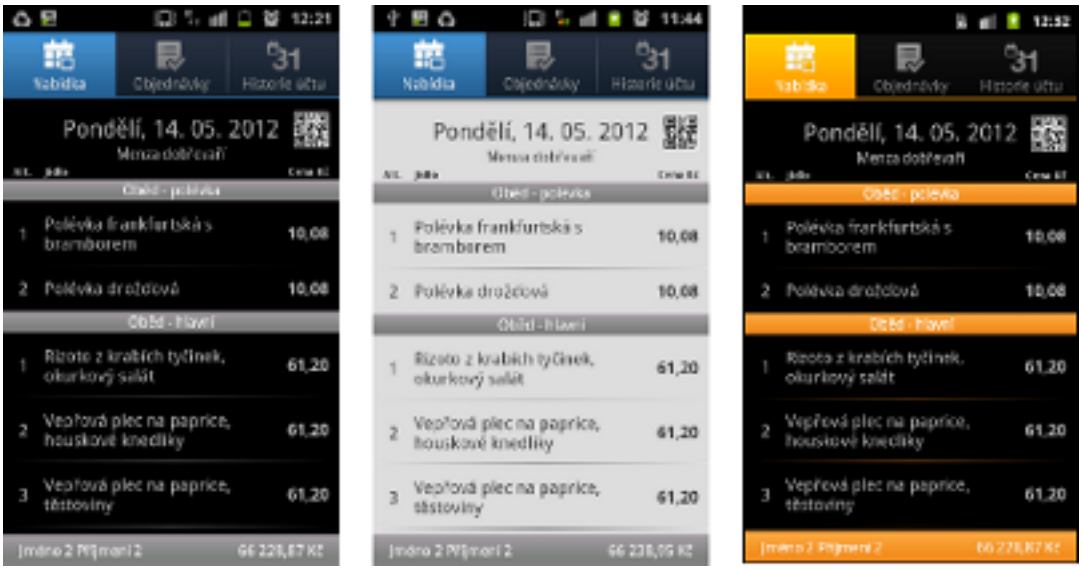
7.9. Změna tématu

Uživatelské rozhraní bylo vytvářeno od počátku vývoje pomocí systémových komponent. Těch, které uživatel zná z ostatních aplikací a se kterými by neměl nastat žádný problém. Vzhled těchto komponent lze sice razantně změnit se zachováním funkčnosti, nicméně je to náročná grafická práce a díky uživatelské přívětivosti systémových komponent bylo vhodné se tomuto kroku vyhnout.

Záměrem ale bylo udělat aplikaci líbivější, například namísto odstínů šedi použít jinou barvu, kterou by byly obarveny oddělovače seznamů, lištu s informacemi o uživateli a podobně. Problém nastal po zjištění, že různé instalace Android OS podle výrobců mají systém obarvený do různých barev. Většinou se jedná o podbarvení vstupů, barvu podsvícení položek seznamu při kliknutí na danou položku a podobně. Jedná se ale také o barvu záložkové lišty, která je v aplikaci stále viditelná. Tato barva však není programově zjistitelná. Jelikož se ale podle výrobců razantně liší (často používané barvy jsou oranžová, modrá, červená, zelená), nebylo možné pro prvky použít jinou barvu než odstíny šedi.

Z tohoto důvodu je umožněna uživateli změna tématu aplikace. Například pro oranžové systémové téma si tak může uživatel vybrat oranžové téma aplikace (obr. 17. vpravo). Nastavení tématu je dostupné přes nabídku, v níž je možné vybrat téma podle jména ze seznamu. Je to sice trochu nepraktické, ale v konečném důsledku je to lepší než „natvrdo“ zvolené téma v odstínech šedi (obr. 17. vlevo). To je uživateli implicitně vybráno po instalaci, jelikož se hodí ke všem barvám systémového tématu. V nastavení si však uživatel může později zvolit téma takové, které do systému lépe „zapadne“.

Navíc díky vytvořenému systému témat je možné použít téma, které nejen mění barvy prvků ale i pozadí aplikace. Někteří uživatelé totiž mohou být zvyklí spíše na aplikace se světlým pozadím, proto je jim umožněno nastavení světlého tématu (17. uprostřed). Pomocí principu témat by bylo možné aplikaci měnit razantněji, to však z důvodu jednotnosti uživatelského rozhraní není v plánu.

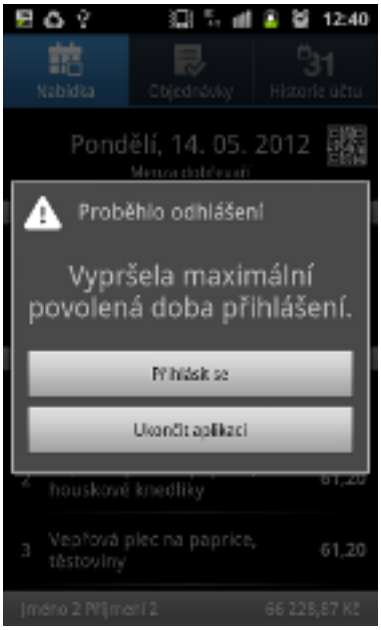


Obrázek 17. Srovnání jednotlivých témat aplikace

7.10. Ztráta relace

Pokud od poslední komunikace se serverem uběhne dostatečně dlouhá doba, může dojít ke ztrátě relace, což má za následek nutnost opětovného přihlášení pro další práci s aplikací. V takovém případě se při požadavku na server zobrazí dialog o ztrátě relace (obr. 18.), pomocí kterého si může uživatel vybrat, zda chce provést opětovné přihlášení, či zda chce aplikaci ukončit.

Tento dialog se navíc zobrazuje i v případě, pokud mezi přihlášením a požadavkem na server z jednoho zařízení dojde k druhému přihlášení na jiném zařízení. Tento princip vylučuje zasílání požadavků z více klientů zároveň, což by mohlo vést k nekonzistenci stravovacího účtu.



Obrázek 18. Dialog o ztrátě relace

7.11. Záložky

Pro řešení tří oddělených částí aplikace (zobrazení nabídky, objednávek a historie) bylo použito rozvržení založené na záložkách (obr. 19.). Ty jsou mimo přihlašovací obrazovku dostupné napříč celou aplikací, čímž je uživatelské rozhraní pro přepínání částí velmi přehledné.

Aktivní záložka je vždy zvýrazněna, díky čemuž má uživatel přehled, v jaké části se právě nachází. Bez využití záložek by bylo nutné provádět přepínání částí pomocí nabídky, což by bylo pravděpodobně značně matoucí.

Obrázek 19. Ukázka lišty se záložkami

7.12. Ikony

V aplikaci jsou použity ikony, které jsou používány v systémových aplikacích. Z tohoto důvodu nejen že vzhledově zapadnou do systému, ale také bude pro uživatele snadnější rozpoznat funkčnost, která se pod nimi skrývá. Tyto ikony jsou šířené pod licencí Apache licence, verze 2. Pro více informací, viz dodatek A. Další ikony v aplikaci, které nebyly v systému k dispozici, byly vytvořeny tak, aby co nejlépe zapadly do stejného vzhledu jako ikony ostatní.

Jako spouštěcí a hlavní ikonu aplikace bylo použito logo Univerzity Palackého společně s příborem (obr. 20.). Je tak zřejmé, že se jedná o aplikaci spojenou jak s univerzitou, tak se stravováním.

Obrázek 20. Spouštěcí ikona aplikace

8. Nasazení do reálného provozu

Po celou dobu vývoje aplikace byl jako vzdálený server použit testovací server ve společnosti ANETE spol. s r.o. Pro vývojáře klienta, je veřejné pouze rozhraní webových služeb, které tak může být implementováno odlišně na různých serverech.

8.1. Propojení se serverem Správy kolejí a menz

Před i během vývoje byla několikrát informována Správa kolejí a menz Univerzity Palackého (dále jako SKM) o průběhu vývoje. Vedení SKM se této myšlence nebránilo a vývoj takové aplikace vítalo.

Se SKM komunikovala i společnost ANETE spol. s r.o. ohledně nasazení webových služeb na jejich server. To se i přes technické problémy s nekompatibilní starší verzí informačního systému podařilo a aplikace je tak nyní připravena na komunikaci s menzou Univerzity Palackého. Aplikace je však v současné době v rámci interního testování společnosti ANETE spol. s r.o. propojena stále s testovacím serverem, nad jehož daty je možné si aplikaci vyzkoušet. Postup pro testování aplikace je popsán v dodatku C.

8.2. Distribuce přes obchod s aplikacemi

V pokročilých operačních systémech pro mobilní zařízení je většinou k dispozici služba pro distribuci aplikací. V případě Android OS je to Google Play, což je přejmenovaný dřívější Android Market. Tato služba poskytuje vývojářům bezplatnou i placenou distribuci aplikací, které jsou tak dostupné všem majitelům kompatibilních přístrojů. Pro publikování aplikace je možné přidat kromě názvu a popisu aplikace také mimo jiné snímky obrazovky, video zobrazující práci s aplikací a další propagační informace.

Vývojářský účet pro distribuci této aplikace byl již zřízen, nicméně aby bylo možné začít s distribucí a nasazením do reálného provozu, je nutné kompletní otestování aplikace ve společnosti ANETE spol. s r.o. Po publikaci bude aplikace k dispozici na internetové adrese `http://play.google.com/store/apps/details?id=cz.novaklukas.droidkredit`.
Bylo rozhodnuto, že aplikace bude z důvodu nabídnutí co největšímu počtu strávníků, ale také z důvodu tvorby v rámci bakalářské práce, nabízena bezplatně.

8.3. Propagace aplikace

Od chvíle, kdy bude aplikace publikována, bude potřebná co největší propagace v rámci Univerzity Palackého. Záměrem je, aby aplikace ulehčila objednávání jídel z univerzitní menzy co nejvíce studentům Univerzity Palackého.

Počátkem zimního semestru následujícího akademického roku by bylo vhodné vytvořit informační letáky, které budou k dispozici u jednotlivých objednacích míst v menzách UP.

8.4. Připomínky uživatelů

Google Play podporuje hodnocení a přidávání komentářů k aplikaci uživatelům, kteří si ji nainstalovali. Očekává se, že po publikaci bude k dispozici zpětná vazba uživatelů, kterou bude možné využít pro další vývoj aplikace. Informace o možnosti aplikaci hodnotit a přidávat komentáře byla zmíněna v dialogu „o aplikaci“, kde je pro tento případ možné využít tlačítko pro přímé přesměrování na stránku aplikace v obchodu Google Play.

9. Plány do budoucna

Rád bych pokračoval ve spolupráci se společností ANETE spol. s r.o., což umožní další vývoj této aplikace. Nyní má aplikace základní funkcionalitu pro správu stravovacího účtu, nicméně je k dispozici mnoho nápadů, ať už na přidání dalších funkcí nebo jen vylepšení stávajících. Aplikaci by bylo možné obohatit například o tyto funkce:

- Přidání dalších stravovacích zařízení, aby bylo možné aplikaci používat na dalších školách v České i Slovenské republice, které využívají stravovací systém společnosti ANETE spol. s r.o.
- Přesunutí informací a fotografií jednotlivých výdejen na vzdálený server. Zmenšila by se tak velikost balíčku aplikace.
- Vytvoření webové služby pro změnu hesla ke stravovacímu účtu. V tuto chvíli je to možné pouze přes internetovou aplikaci.
- Vytvoření webové služby pro odhlášení objednávky.
- Přepsání aplikace na nové uživatelské rozhraní doporučené v nových verzích Android OS, zejména od verze 4.0, označované *Ice Cream Sandwich*.
- Využití systémových upozornění na objednané jídlo, na otevření výdejny apod.
- Možnost přidání objednané položky do systémového kalendáře.
- Funkce, která by na pozadí hlídala, zda výdejna v následujících dnech nenabízí jídlo obsahující definované klíčové slovo. Pokud ano, byl by uživatel upozorněn pomocí systémových notifikací.
- Možnost spuštění aplikace bez přihlášení. To znamená dočasné ukládání objednávek a historie, tak aby byly dostupné bez internetového připojení.
- Vytvoření tzv. „widgetu“ – prvku, který se vkládá na domovskou obrazovku zařízení. Ten by například zobrazoval, jaké má uživatel na daný den objednané jídlo.
- Přidání sociálního prvku do aplikace v podobě ohodnocení jídel a podpory komentářů. Dále také okamžité nahrání fotografie jídla pomocí fotoaparátu a následné zobrazení uživatelům v aplikaci.
- Možnost sdílení pocitů po konzumaci jídla na sociálních sítích Facebook, Twitter, Foursquare.

Závěr

Cílem práce bylo vyvinout aplikaci na platformě Android pro správu stravovacího účtu v menze Univerzity Palackého. Vznikla tak aplikace, kterou budou moci využít strávníci Univerzity Palackého a která usnadní správu stravovacího účtu. Ta byla do této chvíle dostupná pouze přes internetovou aplikaci nebo přes objednací místa ve výdejnách. Pomocí aplikace si lze zobrazit nabídku pro vybrané datum a výdejnu, objednat jídlo, zobrazit objednávky, historii, informace o zůstatku či informace o vybrané výdejně. Je také možné si objednat jídlo pomocí oskenování QR kódu. Aplikace byla vyvinuta pro co největší počet zařízení s OS Android, čímž by měla být dostupná co největšímu počtu strávníků.

Aplikace byla vyvinuta ve spolupráci s brněnskou společností ANETE spol. s r.o., která v průběhu práce vyvíjela webové služby pro komunikaci se vzdáleným serverem. Při vývoji a testování byl jako tento server použit testovací server s nereálnými daty, avšak po skončení vývoje došlo k propojení se serverem Správy kolejí a menz Univerzity Palackého. Po otestování dojde k distribuci aplikace přes obchod s aplikacemi Google Play. Aplikace by tak měla co nejdříve sloužit strávníkům Univerzity Palackého.

I přesto, že byla aplikace primárně vyvíjena pro použití na Univerzitě Palackého, v budoucnu se plánuje její rozšíření i na další univerzity v České republice, které využívají stravovací informační systém společnosti ANETE spol. s r.o.

Reference

[1] *Supporting Multiple Screens : Android Developers* [online], poslední revize 9.5.2012 [cit. 2012-05-14]. Dostupné z : http://developer.android.com/guide/practices/screens_support.html.

[2] MEIER, Reto. *Professional Android™ 2 Application Development*. 1. vyd. Indianapolis (Indiana) : Wiley Publishing, Inc., 2010. 543 s. ISBN 978-0-470-56552-0.

[3] *Activities : Android Developers* [online], poslední revize 9.5.2012 [cit. 2012-05-14]. Dostupné z : http://developer.android.com/guide/topics/fundamentals/activities.html.

[4] *The Developer's Guide : Android Developers* [online], [cit. 2012-05-14]. Dostupné z : http://developer.android.com/guide/index.html.

[5] Network Working Group. *Request for Comments: 2617* [online], poslední revize 1999-06 [cit. 2012-05-14]. Dostupné z : http://www.ietf.org/rfc/rfc2617.txt.

[6] *Handling Runtime Changes : Android Developers* [online], poslední revize 9.5.2012 [cit. 2012-05-14]. Dostupné z : http://developer.android.com/guide/topics/resources/runtime-changes.html.

A. Licence použitých zdrojů

V aplikaci byla použita obrazová data a zdrojové kódy, které jsou licencované pod Apache License, verze 2.0. Soubor s informacemi o licenci `NOTICE.txt` je k dispozici na CD v kořenovém adresáři. Jsou to tato data:

- Obrazová data, která jsou v aplikaci použita pro ikony v menu a ikony dialogů a záložek. Ta jsou získána z distribuce Android SDK.
- Třídy z balíčku `com.google.zxing.integration.android`. Ty jsou použity pro propojení s aplikací `Barcode Scanner`, která umožňuje skenovat a dekódovat QR kódy. Podrobné informace o principu skenování QR kódů jsou uvedeny v části 7.8.
- Třída `cz.novaklukas.droidkredit.Base64Compatibility`, získaná z distribuce Android SDK, která je použita při běhu na zařízení se systémem Android verze 2.1. Android API podporuje práci s formátem Base64 až od verze 2.2, ve kterém je již třída `Base64` obsažena. Třída `cz.novaklukas.droidkredit.Base64Compatibility` je tak použita pro zajištění co největší kompatibility s Android přístroji.
- Třídy v balíčku `com.byarger.exchangeit`, které jsou použity pro připojení na testovací server a které umožňují připojení klienta k serveru pomocí protokolu HTTPS tak, že klient přijímá veškeré certifikáty. Toto řešení však není vhodné z hlediska bezpečnosti, proto je k dispozici i vlastní řešení, které přijímá pouze určený certifikát. To bude zřejmě použito při reálném nasazení.

B. Obsah přiloženého CD

Přiložené CD má následující strukturu:

- CD/
  - bin
    - android
      - DroidKredit.apk — Balíček pro OS Android
      - readme_android.txt — Návod pro spuštění aplikace
    - linux — Emulátor pro OS GNU/Linux
      - .android
      - android-sdk-linux
      - avd
      - DroidKredit.apk
      - readme_linux.txt — Návod pro spuštění emulátoru
      - run.sh — Skript pro spuštění emulátoru
    - windows — Emulátor pro OS Windows
      - .android
      - android-sdk-windows
      - avd
      - DroidKredit.apk
      - readme_windows.txt — Návod pro spuštění emulátoru
  - doc
    - doc-src — Zdrojový text dokumentace v LaTeX
    - screenshots — Snímky obrazovek
    - screencast.ogv — Screencast použití
    - AndroidMenzaUP_Novak.pdf — tato dokumentace
  - src — Zdrojové soubory aplikace (projekt Eclipse IDE)
  - util
    - Meal2QR — Utilita pro vygenerování QR kódu
      - Meal2QR.class
      - Meal2QR.java
  - NOTICE.txt — Licence použitých zdrojů
  - README.txt — Popis obsahu CD

C. Spuštění aplikace v emulátoru

Bylo vytvořeno několik způsobů jak spustit aplikaci, pokud není k dispozici mobilní zařízení s Android OS. Postup se liší podle použitého operačního systému.

Testovací server ve společnosti ANETE spol. s r.o. má občasné výpadky. Výpadek je v aplikaci při přihlášení signalizován hlášením „Server není dostupný“ nebo „Chyba komunikace serverem“. V případě výpadku mě prosím kontaktujte na <novakl@phoenix.inf.upol.cz>. Děkuji.

I přestože je nepravděpodobná situace, kdy by byl server nefunkční delší dobu, vytvořil jsem snímky obrazovek a screencast zobrazující práci s aplikací. Tyto soubory jsou uloženy na CD v adresáři `doc/`.

C.1. Postup na OS GNU/Linux

Pro OS GNU/Linux je připraven adresář s Android SDK, s obrazy virtuálního zařízení a nakonfigurovaným emulátorem. Vše je nastaveno tak, aby bylo spuštění emulátoru co nejjednodušší. Pro běh emulátoru je potřeba mít adresář `/bin/linux/` na zapisovatelném médiu.

Celý proces je navíc zautomatizován pomocí Bash skriptu, který se nachází v cestě:

```
/bin/linux/run.sh
```

Pokud by tento skript nepracoval korektně, lze použít manuální spuštění pomocí následujícího návodu:

1. V terminálu nastavte pracovní adresář na adresář `/bin/linux/`, který je na zapisovatelném médiu. Ověřte, zda mají, po kopii z CD, všechny soubory práva k zapisování.

2. Nastavte proměnnou cesty Android SDK na pracovní adresář příkazem:
   `export ANDROID_SDK_HOME=`pwd``

3. Spusťte emulátor s předpřipraveným obrazem systému příkazem:
   `./android-sdk-linux/tools/emulator -avd bp &`

4. Po nastartování systému a zobrazení uzamykací obrazovky odemkněte telefon „vysunutím“ lišty se zámkem a následně spusťte aplikaci pomocí ikony na domovské obrazovce.

5. Přihlaste se do aplikace pomocí následujících testovacích údajů: Uživatel: 2, Heslo: x

C.2. Postup na OS Windows

Pro OS Windows je situace o něco složitější než pro OS GNU/Linux nicméně byly provedeny takové kroky, aby bylo spuštění co nejjednodušší. Pro spuštění emulátoru na OS Windows je nutné postupovat podle následujícího návodu:

1. Zkopírujte adresář `bin\windows\` na zapisovatelné médium. Ověřte, zda mají, po kopii z CD, všechny soubory práva k zapisování.

2. Nastavte systémovou proměnnou ANDROID_SDK_HOME na adresář `bin\windows\`, který je na zapisovatelném médiu (Počítač – Vlastnosti – Upřesnit – Proměnné prostředí).

3. V souboru `windows\.android\avd\bp.ini` nastavte absolutní cestu k adresáři `windows\avd\bp.avd`

4. Spusťte aplikaci `windows\android-sdk-windows\AVD Manager.exe` a zkontrolujte, zda je virtuální zařízení s názvem bp správně nastaveno („fajfka“ – A valid Android Virtual Device).

5. Spusťte emulátor příkazem: `windows\android-sdk-windows\tools\emulator.exe -avd bp`

6. Po nastartování systému a zobrazení uzamykací obrazovky odemkněte telefon „vysunutím“ lišty se zámkem a následně spusťte aplikaci pomocí ikony na domovské obrazovce.

7. Přihlaste se do aplikace pomocí následujících testovacích údajů: Uživatel: 2, Heslo: x