PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Klient Stagu pro platformu iOS

2014

Pavel Šamánek

Anotace

Cílem bakalářské práce je implementovat klienta informačního systému STAG pro platformu iOS. Uživatel bude moci jednoduchým a rychlým způsobem zobrazit základní informace ve své studijní agendě. Aplikace je určena pro studenty nebo vyučující Univerzity Palackého v Olomouci. V textové části jsou popsány témata týkající se programování pro platformu iOS a webových služeb.

Rád bych poděkoval mému vedoucímu práce, doc. RNDr. Michalu Krupkovi, Ph.D., za odbornou pomoc, cenné rady a připomínky při zpracování bakalářské práce.

Obsah

Seznam obrázků

Seznam tabulek

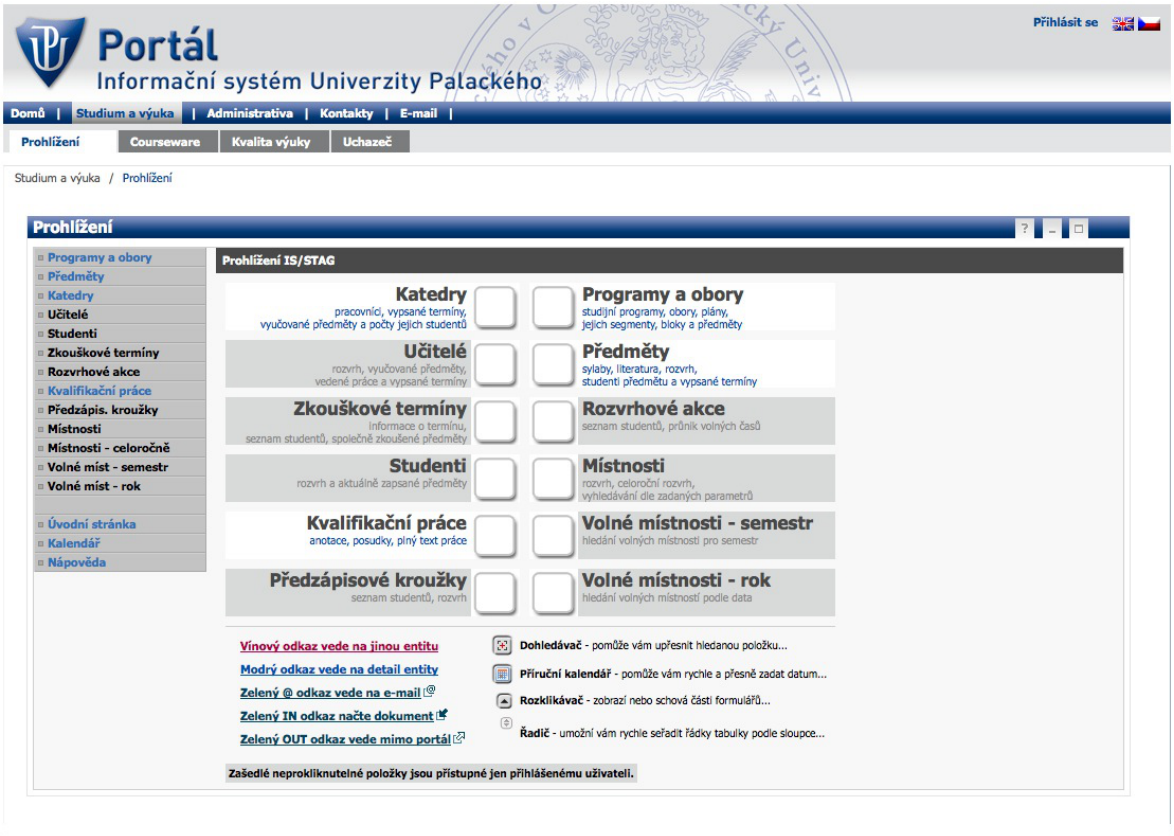
1. Úvod

Textová část bakalářské práce je zaměřena na stručný úvod do problematiky tvorby programů pro platformu iOS. Platformou iOS rozumíme mobilní zařízení od firmy Apple Inc. Nejrozšířenějším zařízením platformy iOS je mobilní telefon Apple iPhone. Další zařízení jsou Apple iPod Touch, Apple iPad, Apple iPad Mini a nejnovější Apple TV. Tyto zařízení operují s mobilním operačním systémem Apple iOS.

V práci je popsána platforma iOS, základy programovacího jazyka Objective-C, vybrané partie rozhraní Cocoa Touch a práce s webovými službami informačního systému STAG Univerzity Palackého v Olomouci.

Informační systém STAG (dále jako IS/STAG) je určen pro administraci studijních záležitostí vysoké školy. Poskytuje celou řadu funkcí pro studenty i vyučující. Obsahuje funkčnost od přihlášení do přijimacího řízení studijního programu až po elektronické odevzdání kvalifikační práce.

V současné době existuje k IS/STAG Univerzity Palackého pouze webové rozhraní, které je pro vyhledání a zobrazení základních informací příliš komplikované, viz obrázek 1. Speciální uživatelé IS/STAG (např.: správci, referenti a rozvrháři) mají k dispozici také klienta pro MS Windows. Aplikace StagClient, která je implementována v rámci bakalářské práce, zpřístupní základní informace jednodušším a rychlejším způsobem. Aplikace je určena pro studenty nebo vyučující Univerzity Palackého v Olomouci.



Obrázek 1. Webové rozhraní IS/STAG

2. Platforma iOS

V této kapitole je popsána platforma iOS, vznik tohoto systému a dostupné vývojové prostředky. Informace byly čerpány ze zdrojů [1] a [2].

2.1. Historie

Historie platformy iOS sahá až do roku 1989, ve kterém byl představen společností NeXT Computers, Inc. operační systém NeXTSTEP. Původně byl tento systém určen pouze pro počítače NeXT. Jádro je tvořeno Mach kernelem[1], který má objektově orientovanou architekturu a podporu multitaskingu. Základní služby operačního systému jsou poskytovány BSD Unixem[2]. Později se společnost NeXT Computers, vedená Stevem Jobsem, spojila s jeho dříve založenou společností Apple. Operační systém NeXTSTEP poskytl základy k vybudování operačního systému Mac OS X, jehož derivací vznikl systém iOS.

2.2. Operační systém iOS

Pojmenování iOS vzniklo z názvu iPhone OS. Jedná se o mobilní operační systém od firmy Apple, který byl představen společně s první verzí iPhone v roce 2007. Systém byl navržen pouze pro přístroje iPhone a iPod Touch, později byl rozšířen i pro zařízení iPad a Apple TV. Přibližně každý rok Apple vydává novou verzi, která poskytuje rozšířenou funkčnost.

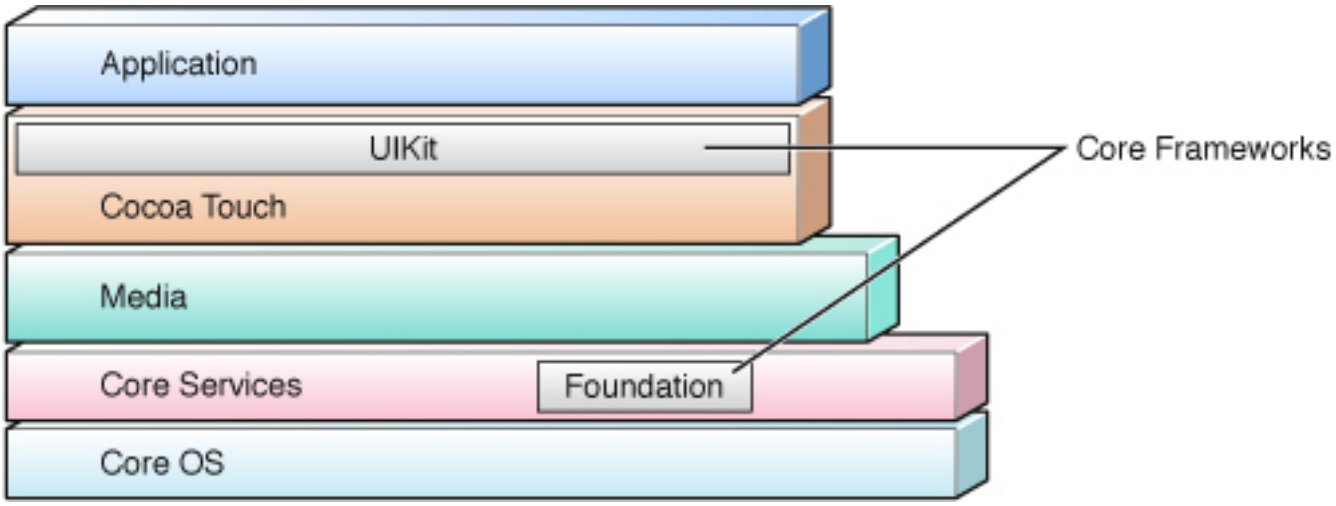
Podobně jako většina klasických desktopových systémů i iOS interaguje s uživatelem pomocí GUI[3], které však není založeno na vstupech z klávesnice a myši, ale na vstupech z dotykového displeje. Např.: aplikace se spouští jednoduchým dotykem místo dvojkliknutí tlačítka myši. Obrazovky mohou být přepínány přejetím prstů po dotykovém displeji místo překlikávání oken.

Systém iOS je založen na jednoduchosti, proto neobsahuje některé funkce, které lze nalézt v klasických operačních systémech. Nepracuje se se soubory, tak jako ve Windows nebo Mac OS X. Nastavení aplikací neprobíhá v samotné aplikaci, ale v systémovém nastavení, kde nalezneme veškeré nastavení zařízení iOS. Systém iOS podporuje běh více programů, avšak dokáže současně zobrazit pouze jeden program. Architektura iOS je tvořena čtyřmi vrstvami, které jsou popsány v další podkapitole.

[1] Mach kernel vznikl na Carnegie-Mellonově univerzitě jako nástroj pro výzkum operačních systémů, zejména pro paralelní a distribuované výpočty.

[2] Rozšíření původního Unixu, které vzniklo na univerzitě v Berkley.

[3] Graphical User Interface - grafické uživatelské rozhraní



Obrázek 2. Architektura systému iOS (převzato z [1])

2.3. Architektura iOS

Jednou z hlavních funkcí operačního systému je poskytnutí abstrakce hardwaru[4]. Tuto funkci samozřejmě plní i iOS, poskytuje abstrakci, která je rozděleno do několika vrstev. viz obrázek 2.

- **Core OS** - jedná se o základní vrstvu (jádro operačního systému). Najdeme zde nízkoúrovňové služby poskytované operačním systémem. Tato úroveň obsahuje jádro, souborový systém, síťovou architekturu, zabezpečení, správu napájení a velké množství systémových ovladačů. Také obsahuje knihovnu libSystem, která je kompatibilní s POSIX/BSD 4.4/C99 API specifikacemi. Pomocí této knihovny je možno přistupovat k nízkoúrovňovým funkcím. Knihovna je psaná v jazyku C.

- **Core Services** - tato vrstva obsahuje frameworky[5], které obstarávají základní práci s řetězci, kolekcemi, sítí, kontakty a nastavením. Také obsahuje služby, které pracují s GPS, kompasem, akcelerometrem a gyroskopem.

- **Media** - poskytuje grafické a multimediální služby vrstvě Cocoa Touch, zcela závisí na vrstvě Core Services.

[4]Jedná se o zjednodušený pohled, kdy jsou skryty detaily ovládání hardwaru v jednoduchých funkcích. Programátor pak nemusí znát detaily ovládání, ale využívá pouze funkce.

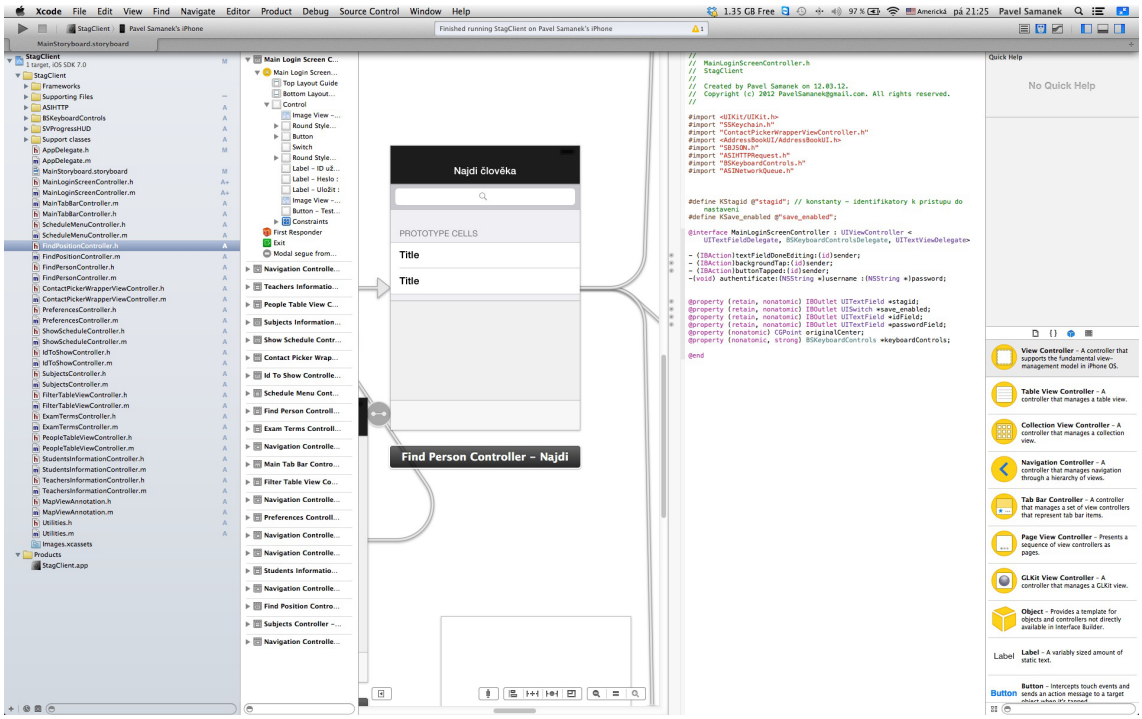
[5]Framework - jedná se o abstrakci, viz poznámka výše. Framework můžeme chápat jako zjednodušený pohled nad danou oblastí. Nemusíme znát detaily implementace jednotlivých akcí, pouze využíváme dostupné prostředky pro provedení námi požadovaných akcí.

- **Cocoa Touch** - vysokoúrovňová vrstva, která obsahuje klíčové frameworky pro tvorbu iOS aplikací. Obsahuje frameworky UIKit, GameKit, MapKit a další. Tato vrstva je popsána pordrobněji v další kapitole.

V celé této architektuře jsou dva frameworky, které jsou pro vývoj iOS aplikací klíčové. Jedná se o UIKit z vrstvy Cocoa Touch a Foundation z vrstvy Core Services.

UIKit poskytuje objekty grafického uživatelského rozhraní a definuje tyto základní chování aplikace - události, obsluhy událostí a vykreslování.

Druhý základní framework se nazývá **Foundation**. Tento framework definuje základní chování objektů, práci s nimi a poskytuje primitivní datové typy, kolekce a služby operačního systému.



Obrázek 3. Ukázka vývojového prostředí XCode IDE

2.4. Vývojové prostředky

Aplikace pro iOS jsou psány v programovacím jazyku Objective-C. Nejrozšířenější vývojové prostředí je Xcode IDE[6], viz obrázek 3. Vývojové prostředí neslouží pouze k vývoji aplikací pro iPhone, ale také pro iPad a Mac. Xcode je poskytován zdarma k operačnímu systému Mac OS X.

Xcode IDE v sobě osahuje prostředky k vývoji, ladění i optimalizaci. Obsahuje také simulátor iPhonu a iPadu, na kterém je možné testovat vytvořené programy. Dodávaný simulátor je skvělý nástroj pro urychlení testování, avšak nepodporuje hardwarově závislé funkce spojené s kamerou nebo akcelerometrem, proto aplikace využívající tyto funkce je nutné testovat na reálném přístroji.

K tomu, aby bylo možné otestovat program na reálném přístroji iOS, je nutné se přihlásit do některého z Apple vývojářských programů. Standardní vývojářský program, který stojí 99$ ročně, zajišťuje technickou podpou, distribuci programu přes App Store[7], a také možnost testovat a debugovat[8] program na reálném iOS

[6]Zkratka IDE označuje Integrated development environment, jedná se o vývojové prostředí pro tvorbu programů, které obsahuje editor zdrojového kódu, kompilátor nebo interpret a většinou také debugger.

[7]Jedná se o virtuální obchod s aplikacemi. Existuje jak pro platformu iOS, tak pro Mac. Jedná se o jedinou možnost, jak legálně zakoupit software pro iOS.

[8]Odladit - nalézat potencionální chyby

přístroji. Standardní program je omezen pouze na jednoho vývojáře. Enterprise vývojářský program, který stojí 299$ ročně, je určen pro firmy a projekty, které mají více vývojářů. Existuje také univerzitní vývojářský program, který je zcela zdarma. Univerzitní program je určen pouze pro vysoké školy a není zde možnost distribuce aplikací přes App Store.

2.5. Desktop versus iOS

Programování pro mobilní platformu se liší od programování pro klasický desktopový systém. V této podkapitole jsou popsány základní rozdíly tvorby programů pro klasický desktopový systém a mobilní systém. Vycházel jsem z literatury [4].

Prvním rozdílem je omezení pouze jedné aktivní a zobrazené aplikace v daný čas. Systém iOS od verze 4 umožňuje, aby se aplikace po stisku home tlačítka[9] přepnula do pozadí. Aplikace tedy může být aktivní v pozadí, ale není zobrazena.

Desktopové operační systémy dovolují běžícím aplikacím, aby nezávisle na sobě mohly vytvořit a kontrolovat více oken. Systém iOS poskytuje aplikacím pouze jediné okno. Jakákoliv interakce s aplikací probíha v tomto okně, jehož velikost je pevně daná velikostí obrazovky.

Programy na počítači mají přístup v podstatě ke všemu, k čemu má přístup uživatel spuštěné aplikace. Systém iOS tuto vlastnost striktně omezuje. Aplikace iOS může číst nebo zapisovat data pouze do vybrané části iOS souborového systému, která byla pro aplikaci vytvořena. Tato oblast se označuje jako sandbox[10] aplikace. Aplikace má několik dalších omezení v přístupu, není možné přistupovat k nízko číselným síťovým portům, které jsou již rezervovány. Není také možné provést žádnou operaci, která by vyžadovala práva administrátora.

Systém iOS očekává od aplikací, že její data budou po spuštění načtena a hlavní okno zobrazeno nejrychlejším možným způsobem - maximálně pár sekund. Očekává také, že pokud uživatel stiskne home tlačítko, aplikace bude schopna vše rychle uložit a skončit (případně se přepnout do pozadí). Pokud to aplikace neprovede do 5 sekund, bude systémem ukončena i v případě, že svá data nestihne uložit.

[9] Home tlačítko - Kruhové tlačítko na přístrojích iPhone a iPad. Základní funkce tlačítka je návrat z aplikace do hlavní nabídky.

[10] Sandbox je anglické slovo označující pískoviště.

Obrázek 4. Základní obrazovka - iPhone 5 a iPhone 4S

Je také nutné počítat s limitovanou velikostí displeje. Rozlišení 3.5 palcového retina[11] displeje u iPhone 4(S) je 640 x 960 pixelů a 4 palcového displeje u iPhone 5 640 x 1136 pixelů, starší displeje mají rozlišení pouze 320 x 480 pixelů. Retina displeje mají větší rozlišení, avšak více kontrolních prvků se zde nevejde. Situace je jiná u iPhonu 5, kde displej je o půl palce větší - řádově však můžeme z hlediska tvorby GUI považovat displeje iPhonu za totožné. Porovnání úvodních obrazovek u iPhonu 4S a iPhonu 5 najdeme na obrázku 4. Jen pro srovnání - největší současný Apple desktopový displej má 27 palců a rozlišení 2560 x 1440 pixelů. Tvorba GUI se tedy na platfomě iOS řídí zcela jinými pravidly než na klasickém desktopu.

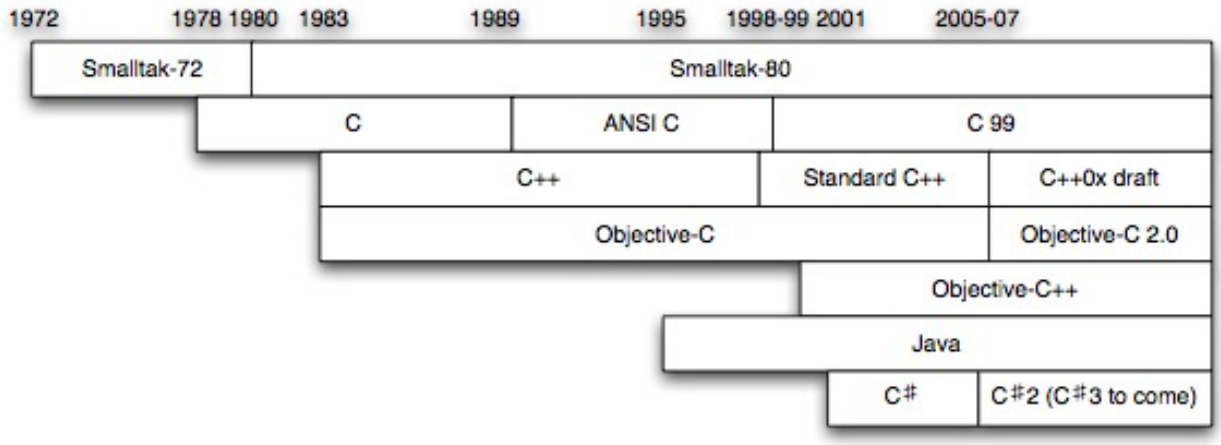
Přístroje iOS mají v současnosti 512 MB (iPhone 4, iPhone 4S, iPad 2) nebo

[11] Označení displeje od Apple s vysokým rozlišením

1024 MB (iPhone 5 a nový iPad) fyzické paměti RAM. Část paměti je vždy využita pro buffer obrazovky a ostatními systémovými procesy. Obvykle aplikacím není k dispozici více než polovina celkové paměti. Klasický desktopový operační systém využívá virtuální paměť[12]. Systém iOS nepoužívá virtuální paměť pro data aplikace, proto je k dispozici aplikacím pouze tolik paměti RAM, kolik je v danou chvíli volné. Je zde však zabudovaný mechanizmus pro rychlé uvolnění paměti. Tento mechanizmus aplikacím sdělí, kdy je potřeba uvolnit paměť. Jakmile tato chvíle nastane, aplikace musí uvolnit nepotřebnou paměť, v opačném případě hrozí její ukončení.

Výše uvedené rozdíly jsou spíše technického rázu. Je nutné brát v potaz zcela jinou formu vstupů než je tomu i klasického desktopu. Přístroje iOS nemají fyzickou myš ani klávesnici - interakce s uživatelem tedy probíha na jiné úrovni a tomu je třeba přizpůsobit stavbu a rozložení GUI.

[12]Virtuální paměť umožňuje ukládání nevyužívané paměti RAM na pevný disk. Aplikace pak mohou bezproblému fungovat dále i přesto, že si vyžádají více paměti RAM než je v danou chvíli dostupné.



Obrázek 5. Přibližná časová linie vzniku Objective-C a ostatních programovacích jazyků. (převzato z [5])

3. Programování pro iOS

V této kapitole jsou popsány základní aspekty programování aplikací pro iOS. Jsou zde stručně popsány základy programovacího jazyka Objective-C, rozebrány jednotlivé technologie Cocoa Touch a popsána architektura, kterou by měla splňovat každá aplikace. Vycházel jsem z literatury [1], [4] a [5].

3.1. Objective-C

Operační systém NextSTEP, jenž poskytl základy k Mac OS X a také iOS, využíval Objective-C, a proto jej dnes můžeme najít i v těchto systémech. Ačkoliv je systém iOS relativně nový, vznik Objective-C je datován do 80. let 20. století, přesně nelze říci, protože zde byly časové mezery mezi jednotlivými verzemi, standardizací a oficiálním vydáním. Hrubá historie Objective-C, jeho předchůdci a současníky je zobrazena na obrázku 5.

Objective-C je postaven na známém programovacím jazyku C a jedním z prvních objektově orientovaných programovacích jazyků - Smalltalk. C++ a Objective-C jsou dvě odlišné větve, které tvoří nadmnožinu jazyka C. Objective-C je velmi blízký k jazyku Smalltalk z hlediska syntaxe a dynamiky. C++ je mnohem více statický s cílem lepšího výkonu. Programovací jazyk C#, vyvíjen Microsoftem, je přímým konkurentem jazyka Objective-C.

Objective-C v jeho aktuální verzi 2.0 uvádí mnohá technická vylepšení jako je např.: Garbage collector[13], vlastnosti (properties), rychlou enumeraci, nová

[13]Nástroj programovacího jazyka, který provádí automatickou správu paměti.

klíčová slova a aktualizovanou funkcionalitu runtime knihoven[14].

3.1.1. Syntax

Objective-C je nadmnožinou jazyka C, proto program napsaný v jazyku C bude lehce zkompilovatelný jako Objective-C kód. Objektově orientovaný model Objective-C je založen na jazyce Smalltalk. Objective-C tedy přebírá syntaxi jazyka Smalltalk z hlediska objektově orientovaných operací. Syntax, která nesouvisí s objektově orientovaným modelem, je převzata z jazyka C.

Klíčová slova v Objective-C začínají znakem „@“, proto aby zde nevznikl konflikt s jazykem C. Příklad některých klíčových slov: `@class`, `@interface`, `@implementation`, `@public`, `@private`, `@try`, `@catch`, `@throw`, `@finally`, `@end`, `@protocol`, `@selector`, `@synchronized`, `@synthesize`.

3.1.2. Datové typy

Objective-C obsahuje všechny datové typy, které je možné nalézt v jazyce C. S textovými řetězci je lze rovněž pracovat stejně jako v jazyce C, ale většina aplikací využívá implementované třídy `NSString`, která s sebou nese rozšířenou funkčnost. Formát zápisu řetězců se neohraničuje pouze anglickými uvozovkami - `"řetězec"`, jak tomu bývá zvykem, ale - `@"řetězec"`. Logický datový typ lze nalézt pod označením `BOOL` s odpovídajícími hodnotami `YES` nebo `NO`. Každý objekt je typu `id`, což umožňuje využít výhody slabě typovaných systémů.

3.1.3. Objektově orientovaný model

Objective-C je založen na objektově orientovaném modelu, kde základní interakce mezi objekty probíhá pomocí zasílání zpráv. Stejně jako ve Smalltalku jednotlivé objekty si mohou zasílat zprávy bez toho aniž by věděli, zda daný objekt na zprávu odpoví. Výhodou je větší flexibilita pro programátora, kdy určité situace mohou být řešeny dynamicky - za běhu programu. Tento způsob se liší od C++ nebo C#, kde místo zasílání zpráv, jsou metody „volány“ a pokud daná metoda není implementována, není možné metodu použít. Pro srovnání je uveden příklad zaslání zprávy `udelej` objektu a volání metody `udelej` objektu:

```
[objekt udelej:argument]; // Objective-C
objekt.udelej(argument); //  C++, C#
```

Zaslání zpráv je v Objective-C ohraničeno hranatými závorkami a nepoužívá se tečková notace. Vidíme tedy, že zde není rozdíl pouze z hlediska sémantiky, ale také z hlediska syntaxe. Naproti tomu implementace i volání klasických funkcí je v Objective-C zcela totožné s jazykem C. V dalších oddílech textu je používán

[14]Knihovny nahrávané za běhu programu

termín „volání metody“, z hlediska Objective-C je tím myšleno zaslání zprávy objektu.

Objective-C definuje kořenovou třídu. Každá nová třída by měla být potomkem této třídy. V Cocoa se tato třída jmenuje `NSObject`. `NSObject` poskytuje běhovému prostředí širokou funkcionalitu. Ukazatel na kterýkoliv objekt může být definován jako ukazatel na třídu `NSObject` nebo, jak již bylo zmíněno, ukazatel na typ `id`. Oceňuji, že Objective-C přenechává programátorovi zodpovědnost a dává mu do rukou silný dynamický nástroj založený na dynamickém zasílání zpráv a slabě typovaném systému.
Příklad deklarace a implementace třídy Foo v Objective-C:

```
@interface Foo : NSObject
{
double x;
}
-(int)   f:(int)x;
-(float) g:(int)x :(int)y;
@end
@implementation Foo
 -(int)   f:(int)x {...}
 -(float) g:(int)x :(int)y {...}
@end
```

Deklarace atributů (dat instance) a metod (funkcí třídy) je uzavřena v bloku `@interface`, který končí `@end`. Atributy jsou navíc deklarovány v bloku uzavřeném závorkami, není tedy možné mísit deklaraci atributů a metod, jak to je možné v C++, či jiným jazycích. Implementace je uzavřena v bloku `@implementation`, končící `@end`.

Metody třídy začínají znakem „-“. Tento znak určuje, že se bude jednat o metodu instance. Je možné také použít znak „+“, který se používá, pokud budeme chtít definovat metodu třídy (statickou metodu). Typ návratové hodnoty a typy argumentů jsou uzavřeny v jednoduchých závorkách. Parametry jsou odděleny dvojtečkou „:“.

Na Objective-C se mi líbí možnost použití popisků u metod. Je to zajímavá věc, na kterou jsem u jiných programovacích jazyků nenarazil. Zpřehledňuje a zrychluje to práci zejména u neznámého zdrojového kódu. Samozřejmě existují propracované systémy automatického zobrazení části dokumentace a popisů funkcí. Tady je to však řešeno velice elegatním způsobem.

Příklad deklarace a použití metody bez popisku:

```
-(void) vlozitObjekt:(id)Objekt:(unsigned int)index
[policka vlozitObjekt:kniha:2]
// policka je instance třídy obsahující metodu vlozitObjekt::
```

Dvojtečka „:“ odděluje parametry a uvádí se také v názvu metody. Příklad deklarace a použití metody s popiskem:

```
-(void) vlozitObjekt:(id)Objekt naIndex:(unsigned int)index
[policka vlozitObjekt:kniha naIndex:2]
// policka je instance třídy obsahující metodu
// vlozitObjekt:naIndex:
```

Při přidání popisku se změní celé jméno metody. Názvy metod jsou delší, ale význam jednotlivých parametrů je jasnější.

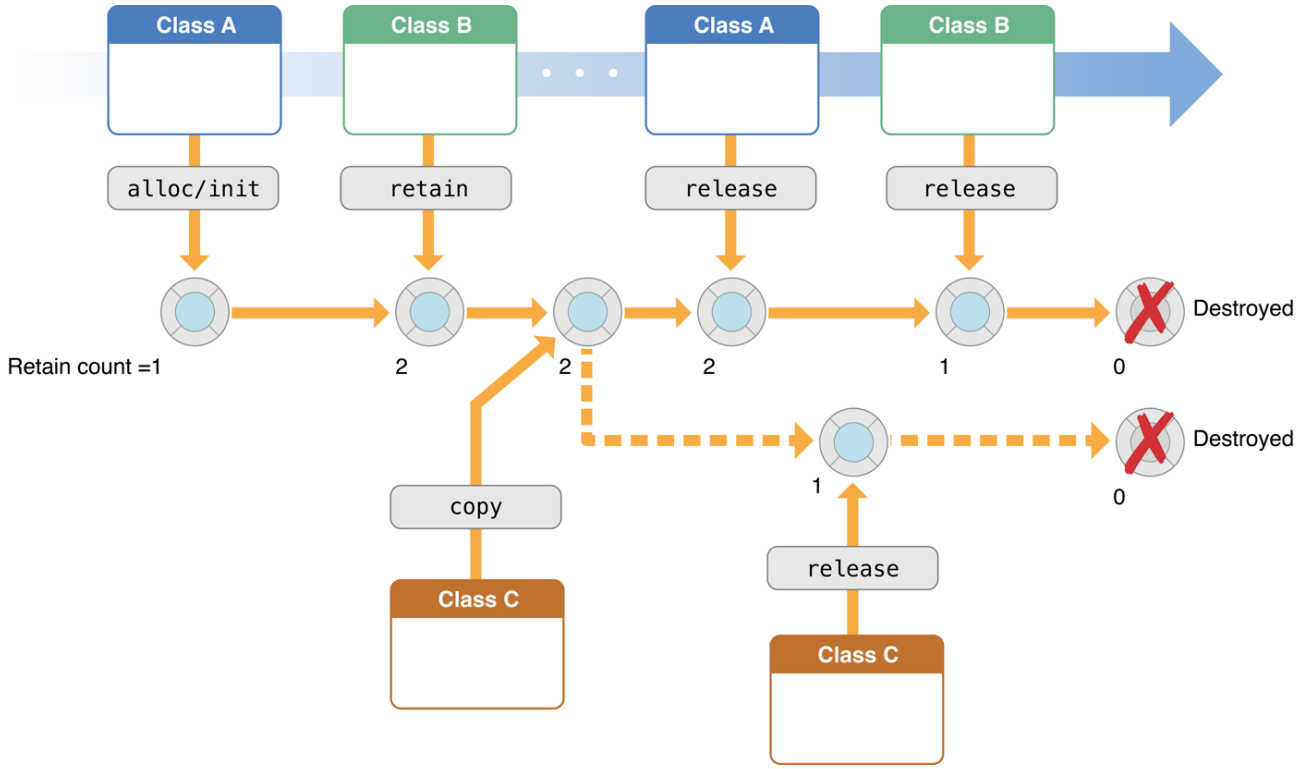
3.1.4. Správa paměti

V Objective-C je nutno pro nový objekt nejprve alokovat paměť a poté provést inicializaci. Jedná se o dva samostatné kroky, které jsou v jiných programovacích jazycích často provedeny v jednom kroku. Alokace paměti pro vytvoření objektu se provádí pomocí metody `alloc`. Nově alokovanému objektu je poté nunté zaslat zprávu `init`, čímž získáme inicializovaný objekt. Návratová hodnota zprávy `init` je také objekt, jehož atributy jsou nastaveny na konkrétní hodnoty. Zpráva `alloc` je tedy zasílána třídě a zpráva `init` je zasílána objektu třídy.

```
Trida *objekt = [[Trida alloc] init];
// Standardní postup alokace a~inicializace na jednom řádku
```

Alokovanou paměť je potřeba po jejím využití opět uvolnit. Každý objekt má přiřazen čítač odkazů, který po jeho alokaci/inicializaci je nastaven na hodnotu 1. Jestliže poté část programu potřebuje objekt dále využít, zašle mu zprávu `retain`, která zvedne čítač objektu o 1. Pokud program již objekt nepotřebuje, zašle mu zprávu `release`, která naopak čítač sníží o 1. Pokud je objekt zkopírován, nově vzniklý objekt má čítač nastaven opět na 1, bez ohledu na to, jakou hodnotu měl čítač původního objektu. Objekt může obdržet libovolný počet `retain` / `release` zpráv, dokud se čítač pohybuje v kladných číslech. Jakmile je čítač roven 0, je provedena dealokace objektu pomocí desktruktoru `dealloc`. Průběh životního cyklu objektu je zobrazena na obrázku 6. V Objective-C nalezneme dva základní přístupy ke správě paměti:

- **MMR (Manual Retain - Release)** je přístup, kdy jsou metody `retain/release` volány manuálně - tento přístup počítá s explicitní správou paměti. Programátor je zodpovědný za uvolnění svých zdrojů, případně zvýšení čítačů pro další využití zdrojů. Čítač odkazů je již implementován ve třídě NSObject, který je kořenovou třídou. Nalezneme jej tak u každého objektu.
- **ARC (Automatic Reference Counting)** je založen na stejném principu čítače odkazů jako MMR, ale při kompilaci programu automaticky umisťuje vhodné `retain/release` zprávy. Programátor tedy nemusí hlídat své zdroje a nepotřebuje nutně znát ani vnitřní mechanismy správy paměti.



Obrázek 6. Životní cyklus objektu z hlediska správy paměti (převzato z [1])

Objective-C ve své verzi 2.0 sice uvedl automatický Garbage collector, který po přidání ARC není Applem oficiálně dále podporován. Pro nové projekty je doporučováno použít ARC.

V aplikaci StagClient je využíváno obou způsobů, kdy některé části využívají automatickou správu a některé jsou spravovány manuálně. Tohoto je možné využít nastavením příznaku `-fobjc-arc` v nastavení kompilátoru u částí, které mají být spravovány automaticky.

3.2. Cocoa Touch

Cocoa Touch je derivací frameworku Cocoa, jehož původ je v operačním systému NeXTSTEP. Cocoa z tohoto systému převzala dva základní frameworky App Kit a Foundation Kit. Z těchto důvodů téměř všechny třídy v Cocoa začínají akronymem NS - NeXTSTEP (např.: NSString, NSArray). Jak již bylo zmíněno dříve, jedná se o vysokoúrovňovou objektově orientovanou vrstu operačního systém OS X, resp. iOS.

Cocoa Touch je základním nástrojem pro tvorbu aplikací pro iOS. Skládá se z několika klíčových frameworků, které jsou téměř totožné s původními frameworky z Cocoa. Cocoa Touch frameworky jsou však speciálně optimalizovány a uzpůsobeny pro dotyková rozhraní.

Mimo základní framework pro tvorbu uživatelského rozhraní UIKit, Cocoa Touch obsahuje další sadu nástrojů od tvorby 3D grafiky, práci se zvukem, sítí a také speciálními API[15] k zpřístupnění fotoaparátu nebo GPS zařízení. Jednotlivé technologie budou popsány podrobněji v následujícím textu.

3.2.1. Core Animation

Core Animation zpřístupňuje 2D, 3D grafiku a animace, které je možno nalézt v iOS. Tvorba animací a přechodů mezi jednotlivými obrazovkami je postavena na složení grafiky několika na sobě nezávislých vrstev. Animace jsou tvořeny pouze definováním základních kroků. Core animation následně dynamicky animace vygeneruje, mimo jiné i s možností přidání dalších speciálních efektů. Pod vrstvou Core Animation nalezneme Quartz 2D, který poskytuje další nástroje pro tvorbu 2D grafiky nebo OpenGL ES API, které zprostředkovává využití grafického akcelerátoru přístroje pro tvorbu 3D aplikací nebo her.

3.2.2. Core Audio

Core Audio poskytuje nástroje pro práci se zvuky na nižší úrovni. Implementuje OpenAL, což je API nezávislé na platfomě a je možné jej nalézt i u konkurenčního Androidu. Tato vrstva zajišťuje aplikacím přehrávaní jednoho nebo více zvuků současně, streamové přehrávání a záznam zvuku. Také obstarává přesměrování zvukového toku po připojení sluchátek do přístroje. Skrze OpenAL dodává mnoho zvukových efektů se kterými je možné operovat.

3.2.3. Core Data

Core Data je jednoduché, objektově orientované řešení, které poskytuje perzistentní uložení dat malého i velkého objemu. Aplikace mohou svá data sdílet pomocí URL, které funguje napříč platformou iOS. Webové aplikace mohou přistupovat k úložišti a uložit si svá cache data přes API, které je poskytováno. Programátor dostane do rukou přístup ke globálním datům přístroje např. k adresáři kontaktů, kalendáři nebo knihovně fotografií. Samozřejmě je možné si také vytvořit vlastní datový model bez nutnosti instalovat databázový systém, ten je již v této vrstve obsažen v podobě SQLite.

[15]Zkratka API označuje Application Programming Interaface, jedná se o programátorské rozhraní.

3.3. Architektura aplikací

Cocoa Touch je postavena na architektuře zvané **Model View Controller** (MVC), která logicky rozděluje program obsahující GUI. V dnešní době téměř všechny objektově orientované frameworky obsahují koncept MVC, ale pouze několik z nich se tímto řídí do takové míry jako Cocoa Touch. MVC architektura dělí třídy, komponenty dle role do tří kategorií:

- **Model** - třídy, které mají na starost data aplikace.
- **View** - tvořena z oken, kontrolními prvky a dalšími elementy, které uživatel vidí a může interagovat.
- **Controller** - logická část aplikace, která spojuje kategorii Model a View. Vyhodnocuje uživatelské vstupy a řídí aplikaci.

Cílem MVC je implementovat objekty tak, aby se řadili pouze do jedné z výše zmíněných kategorií. Například objekt implemetující tlačítko by neměl obsahovat také obsluhu události, která se vyvolá po stisku tlačítka.

MVC pomáhá udržovat znovupoužitelnost kódu. Třída implemetující obecné tlačítko může být použita v jakékoliv aplikaci. Pokud by tlačítko bylo implementováno i s konkrétní funkčností, nebylo by možné jej použít u jiné aplikace.

Xcode obsahuje editor pro tvorbu GUI, kterým lze primárně tvořit vzhled aplikace - **View** část modelu. Samozřejmě je také možné dynamicky upravovat nebo vytvářet vzhled přímo z kódu. Vzhled aplikace vytvořený v Xcode editoru je propojen s konkrétní třídou - částí **Controller**, která obsluhuje události vyvolané uživatelem a další řízení aplikace. Pokud aplikace potřebuje perzistentní uložiště, využije již zmíněnou část knihovny Cocoa Touch - Core Data a implementuje ji v rámci **Model** části aplikace.

Aplikace StagClient se drží konceptu MVC. View část je implementována pomocí relativně nového způsobu tvorby GUI - **Storyboard**. Storyboard umožňuje nejen vytvořit GUI jednotlivých obrazovek, ale také definuje jejich vzájemné propojení. K jednotlivým obrazovkám jsou vytvořeny ovládací části, které zpracovávají data a uživatelské vstupy. Podrobnější popis je možné nalézt v programátorské dokumentaci.

4. Webové služby

Webová služba je z pohledu klienta určité místo na webu, které má svoji URL a na které se provádí určitá operace. V rámci aplikace StagClient to bude například požadavek pro zaslání rozvrhu nebo seznamu studentů určitého předmětu. Webové služby IS/STAG jsou primárně určeny pro integraci s počítačovými systémy. Implementace dle standardů zajišťuje možné použití i pro platformu iOS. Aplikace StagClient tedy přistupuje k datům a akcím IS/STAG přes webové služby. Informace o webových službách IS/STAG byly čerpány z [6].

4.1. Komunikace

Existuje několik standardů pro komunikaci s webovými službami. Pro získání požadovaných dat a následnou práci s těmito daty na iOS zařízení plně dostačuje rozhraní REST (Representational State Transfer), kdy ke komunikaci s webovým serverem je využito metod protokolu HTTP (GET, POST, PUT, DELETE). Narozdíl od jiných rozhraních REST neobsahuje další komunikační protokol, ale jednotlivé požadavky jsou přesně určeny URL adresou.

Většina metod webových služeb IS/STAG obsahuje také volitelný parametr `outputFormat`, kterým je možné určit, jaký formát dat má server vrátit. Jedná se o formáty: XML, XLS, CSV, JSON také PDF. Formát JSON narozdíl od formátu XML přenáší při stejném množství informací méně dat, proto byl zvolen jako primární komunikační prostředek s aplikací. Příklad požadavku REST, který vrací seznam předmětů pro zimní semestr studenta s osobním číslem R11529, formát výstupu je zvolen JSON:

```
http://stagservices.upol.cz/ws/services/rest/predmety/
       getPredmetyByStudent?osCislo=R11529&semestr=ZS&outputFormat
       =JSON
```

Jednotlivé části adresy:

- **http://stagservices.upol.cz/ws/**
  - základní URL webových služeb IS/STAG
- **services/rest/predmety/**
  - adresa konkrétní služby
- **getPredmetyByStudent**
  - název metody
- **?osCislo=R11529&semestr=ZS&outputFormat=JSON**
  - parametry metody

Odpověď JSON výše uvedeného požadavku vypadá následovně :

```
[{
"predmetStudenta"
[{"zkratka":"BZP2","nazev":"Bezpečnostní předpisy v~chemii 2","
      katedra":"KFC","rok":"2013","kredity":0},
{"zkratka":"IM","nazev":"Instrumentální metody","katedra":"ACH","
      rok":"2013","kredity":6},
{"zkratka":"OS5","nazev":"Oborový seminář 5","katedra":"OCH","rok
      ":"2013","kredity":2},
{"zkratka":"SBP","nazev":"Seminář k~bakalářské práci","katedra":"
      OCH","rok":"2013","kredity":4},
{"zkratka":"SC","nazev":"Stereochemie","katedra":"OCH","rok":"
      2013","kredity":4},
{"zkratka":"ZBC","nazev":"Základy bioorganické chemie","katedra":
      "OCH","rok":"2013","kredity":4}]
}]
```

Formát JSON je lehce čitelný i pro člověka. Vrácený soubor JSON má strukturu `klíč:objekt (nebo hodnota)`, kdy ke klíči `predmetStudenta` je navázán seznam předmětů, se kterým následně můžeme operovat podobným stylem.

4.2. Poskytované služby

Webové služby IS/STAG zpřístupňují pouze část funkcionality, která je obsažena na webovém portálu. Pomocí webových služeb není například možné přihlašování/odhlašování ze zkouškových termínů nebo zapisování/odepisování předmětů. Systém tyto metody má implementovány, ale nejsou zveřejněny webovými službami (pro roli student). Webové služby IS/STAG jsou však neustále aktualizovány a je možné, že tyto metody bodou zpřístupněny v budoucnu.

Některé poskytované služby jsou k dispozici pouze ověřeným uživatelům. Autentizace klienta probíha pomocí standardní HTTP BASIC metody. Uživatelské jméno a heslo je zakódováno algoritmem Base64[16] a vloženo přímo do hlavičky HTTP požadavku. Komunikace se serverem je u těchto služeb možná pouze šifrovaně přes protokol HTTPS.

[16]Base64 je algoritmus pro zakódování binarních dat do řetězce ASCII znaků

5. Aplikace StagClient

V rámci této práce byla vytvořena aplikace StagClient pro platformu iOS. Aplikace je určena pro mobilní přístup k IS/STAG a je určena pro studenty nebo vyučující Univerzity Palackého.

V tuto chvíli je k dispozici pro přístup k IS/STAG pouze webové rozhraní, které je značně komplikované pro rychlé zobrazení základních informací. Webové rozhraní je primárně určeno pro přístup přes PC, není nijak optimalizováno pro mobilní zařízení. Aplikace implementovaná v rámci této práce zobrazí základní informace studijní agendy rychlejším a jednodušším způsobem právě přes mobilní zařízení.

Aplikace umožňuje jednoduchým způsobem zobrazit rozvrh přihlášeného uživatele, odeslat jej emailem nebo exportovat do kalendáře. Dále zobrazuje detaily jednotlivých rozvrhových akcí, tzn. jejich sylaby, seznam studentů, seznam učitelů. S těmito daty je možné dále pracovat. Seznam studentů může být zaslán emailem ve formátech XLS nebo CSV. Aplikace poskytuje zobrazení detailnějších informací o konkrétním studentovi, případně učiteli. Na základě těchto informací je možné vytvořit nový kontakt v adresáři, případně doplnit kontakt o tyto informace získané z IS/STAG. Také je možné zobrazit rozvrh vybraného studenta nebo učitele a opět provést export do kalendáře, případně zaslat na emailem.

Pomocí aplikace je možné rychle zjistit vypsané zkouškové termíny, respektive zkouškové termíny, na kterých je uživatel přihlášen. U jednotlivých zkouškových termínů jsou zobrazeny základní informace, tzn. datum, čas a kapacita termínu. Aplikace zobrazuje další informace o zkouškovém termínu. Zobrazí například seznam studentů nebo učitele vypsaného zkouškového termínu. Seznam studentů je opět možné zaslat emailem ve formátech XLS nebo CSV.

Aplikace umožňuje zobrazit na mapě pozici následující rozvrhové akce. Pozice je určena na základě adresy místnosti rozvrhové akce. Adresa místnosti je získána z IS/STAG. Přesné souřadnice pro zobrazení místa na mapě jsou zjištěny veřejným API Geocoding od Google. Google Geocoding vrátí zeměpisnou šířku a délku na základě adresy. Na mapě jsou také zobrazeny detaily rozvrhové akce, tzn. číslo místnosti a budovy, datum a čas začátku. Pro studenty tedy odpadá opakované dívání se do rozvrhu pro zjištění čísla místnosti daného předmětu na začátku každého semestru. Lze také spustit navigaci z aktuálního místa na adresu rozvrhové akce.

Aplikace poskytne uživateli seznam učitelů navštěvované katedry. Je zde možné rychle vyhledat požadovaného učitele a následně zobrazit jeho detailnější informace, vytvořit emailovou zprávu, zavolat či navštívit web učitele. Dále je možné zobrazit aktuální rozvrh, exportovat jej do kalendáře nebo odeslat emailem. Je možné také zobrazit učitelovu pozici následující rozvrhové akce a třeba spustit navigaci.

Aplikace obsahuje seznam kontaktů uživatele a automaticky detekuje možnost existence identity kontaktu v IS/STAG. Uživatel se může na základě své volby

rozhodnout, zda kontakt opravdu má identitu v IS/STAG (je hledáno na základě jména a přijmení), přiřadit kontaktu STAG ID a provádět podobné akce jako se studentem nebo učitelem.

Následuje programátorská dokumentace, kde jsou probrány detaily tvorby aplikace StagClient.

6. Programátorská dokumentace

V této kapitole je popsána architektura aplikace StagClient, jednotlivé třídy, ze kterých je složena a frameworky, které byly do projektu importovány.

6.1. Architektura aplikace

Aplikace je objektově orientovaná. Zdrojový kód je rozdělen do tříd a jak již bylo zmíněno, aplikace se drží konceptu MVC. V další podkapitole bude popsán způsob tvorby vzhledu aplikace, následně bude rozebrána logická část aplikace.

6.2. Grafické uživatelské rozhraní

Vzhled aplikace je tvořen jednotlivými obrazovkami, které se skládají z nakonfigurovaných grafických komponent. Tento vzhled je obsažen v jediném souboru - storyboard aplikace. Storyboard aplikace definuje nejen vzhledy jednotlivých obrazovek, ale také vztahy mezi jednotlivými obrazovkami. Je zde také možné nakonfigurovat přechody mezi obrazovkami. Na obrázku 7. je možné vidět část storyboardu aplikace.

Obrázek 7. Část grafické podoby storyboard u aplikace StagClient

Šipky, které je možné vidět na obrázku 7. určují, zda se jedná o přechod mezi obrazovkami (tzv. segue[17]) nebo o vztah mezi obrazkovkami. Počáteční šipka určuje obrazovku, kterou bude program inicializován.

[17]Segue - termín, kterým Apple označuje přechod mezi jednotlivými obrazovkami.

6.2.1. Přechod mezi obrazovkami

U obrazovky pod níž je nápis `Main Login Screen Controller` a `Main Tab Bar Controller` se jedná o přechod obrazovek. V aplikaci je tedy vytvořen přechod mezi těmito dvěmi obrazovkami. Každý přechod má unikátní identifikátor. V tomto případě je identifikátor roven textovému řetězci s hodnotou `loginSegue`. Pomocí identifikátoru je pak možné přechody vyvolat přímo z kódu. V tomto případě se přechod provede, pokud uživatel zadá správné uživatelské id a heslo. V aplikaci je využito dvou základních typů přechodů:

- **Modal** - označuje přechod mezi dvěmi obrazovkami, kdy druhá obrazovka kompletně překrývá první obrazovku. Uživatel nemůže manipulovat s původní obrazovkou, dokud není prezentovaná obrazovka uzavřena. Uzavření prezentované obrazovky je zodpovědností první obrazovky.
- **Push** - označuje přechod mezi dvěmi obrazovkami, kdy druhá obrazovka je prezentována na zásobník první obrazovky. V Cocoa Touch Existuje třída `UINavigationController`, která obsahuje tzv. navigační zásobník obrazovek. Na dnu zásobníku je základní obrazovka a na vrcholu zásobníku je aktuálně prezentovaná obrazovka. Uživatel může manipulovat s původní obrazovkou pomocí navigačního panelu, který standardně obsahuje také tlačítko `zpět` pro vrácení do předchozí obrazovky.

6.2.2. Vztah mezi obrazovkami

Mezi obrazovkami `Main Tab Bar Controller` a `Find Position Controller` je definován vztah. Pokud mezi dvěmi obrazovkami je definován vztah, znamená to, že první obrazovka obsahuje druhou. První obrazovka je tzv. „View Container“. První obrazovka je v tomto případě tvořena spodním panelem a místem pro prezentaci dalších obrazovek. Druhá obrazovka je prezentována nad tímto panelem. Dohromady obrazovky tvoří požadovaný vzhled aplikace, který je prezentován uživatelovi.

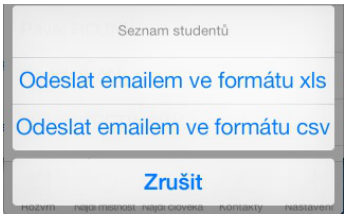
6.2.3. Automatické rozložení vzhledu

V aplikaci je využito nového způsobu umístění grafických komponent. Tento způsob se jmenuje „Auto Layout“ a byl uveden v iOS 6. Každé grafické komponentě jsou nakonfigurovány umísťující omezení. Tyto omezení se mohou vztahovat ke stranám displeje nebo jiné komponentě. Jakmile je nastavení omezení každé grafické komponenty kompletní, vzhled aplikace se automaticky přizpůsobí většímu displeji nebo natočení displeje. Omezení je implementováno třídou `NSLayoutConstraint`.

6.2.4. Uživatelská zpětná vazba

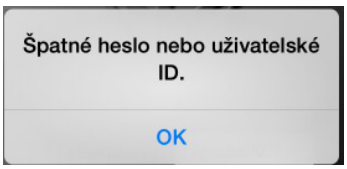
V aplikaci byly použity tyto třídy pro rychlé získání požadované informace od uživatele nebo informování uživatele o stavu aplikace:

- **UIActionSheet** je třída obsažena ve frameworku Apple UIKit, která prezentuje několik možností, jak pokračovat v dané úloze. Výsledek je pak zaslán zvolené třídě. V aplikaci je tato třída využita např. při volbě formátu seznamu studentů předmětu pro zaslání na email, viz obrázek 8.



Obrázek 8. Ukázka grafického vzhledu třídy UIActionSheet

- **UIAlertView** je třída obsažena ve frameworku Apple UIKit, která uživateli zobrazuje varovné dialogové okno se zprávou. Okno může obsahovat několik tlačítek. Uživatelská volba, podobně jako u `UIActionSheet`, je opět zaslána zvolené třídě. V aplikaci je tato třída využita např. při chybovém hlášení o špatně zadaných přihlašovacích údajích, viz obrázek 9.



Obrázek 9. Ukázka grafického vzhledu třídy UIAlertView

- **SVGProgressHUD** je třída, která informuje uživatele o probíhající úloze. Pokud úlohu není možné dokončit, je provedena signalizace uživateli. V aplikaci je tato třída použita k zpětné vazbě uživateli o stahování dat z IS STAG. Grafická podoba třídy je zobrazena na obrázku 10.

Obrázek 10. Ukázka grafického vzhledu třídy SVGProgressHud

6.3. Logická část aplikace

V první části jsou popsány třídy, které jsou propojeny s obrazovkami nakonfigurovanými ve storyboard. Tyto třídy se nazývají kontrolery. V další části jsou stručně popsány ostatní třídy a tabulka frameworků.

6.3.1. Kontrolery

Třídy, které jsou propojeny se vzhledem aplikace, se nazývají kontrolery. Kontrolery obsahují atribut `view`, kde je uložen odkaz na nakonfigurovaný grafický vzhled obrazovky aplikace. Atribut `view` může být nastaven pomocí již zmíněné storyboard nebo je možné jej nastavit až za běhu programu. Kontrolery vyhodnocují uživatelské vstupy, zpracovávají data a řídí přechody obrazovek. Jednotlivé kontrolery jsou odvozeny z tříd obsažených ve frameworku Apple UIKit. Aplikace obsahuje tyto hlavní kontrolery:

- **MainLoginScreenController** je kontroler úvodní přihlašovací obrazovky. Tato třída je potomkem třídy `UIViewController`, která poskytuje základní funkcionalitu pro řízení grafických komponent. Kontroler `MainLoginScreenController` řídí autentizaci a pomocí knihovny `SSKeychain` ukládá uživatelské jméno a heslo, pokud je tato volba zapnuta. V případě správně zadaných přihlašovacích údajů, proběhne přesun do obrazovky, kterou řídí `MainTabBarController`.

- **MainTabBarController** je kontroler obsahující řízení spodní lišty se záložkama. Pomocí záložek je možné přepínat obrazovky. Při úspěšném přihlášení je automaticky zobrazena první obrazovka `ScheduleMenuController`. `MainTabBarController` je potomkem třídy `UITabBarController`, která implementuje funkčnost spodního panelu se záložkami.

- **ScheduleMenuController** je kontroler řídící obrazovku s menu týkající se rozvrhu. Tato třída je potomkem třídy `UITableViewController`, která implementuje řízení obrazovek obsahující tabulky. V tomto případě je tabulka statická - předem nastavená ve storyboard aplikace. Tabulka obsahuje tři prvky: „Zobrazit rozvrh“, „Zobrazit zapsané předměty“ a „Zobrazit zkouškové termíny“. `ScheduleMenuController` řídí přechody obrazovek dle menu a tvoří dno navigačního zásobníku obrazovek. Následné procházení menu je pak jednoduše implementováno pomocí třídy `UINavigationController`.

- **FindPositionController** je kontroler řídící obrazovku s mapou, která zobrazuje pozici následné rozvrhové akce uživatele. Tato třída je potomkem `UIViewController`. Mapa je implementována pomocí třídy `MKMapView`,

která je obsažena ve frameworku Apple MapKit. Kontroler mimo jiné obsahuje také statickou metodu pro nalezení následující rozvrhové akce z parametrem předaného rozvrhu. Tato metoda je využita napříč aplikací. Rozvrh je získán pomocí asynchronního požadavku na webové služby IS/STAG třídou `ASIHTTPRequest`.

- **FindPersonController** je kontroler řídící obrazovku s tabulkou, která obsahuje učitele. Tato obrazovka je propojena s třídou `IdToShowControllerDynamic`, která je zobrazena po zvolení určitého učitele z tabulky. `IdToShowControllerDynamic` obrazovka je prezentována „push“ metodou na navigační zásobník obrazovek. `FindPersonController` je potomkem třídy `UITableViewController`. Tabulka je vytvořena dynamicky z dat získáných z webových služeb IS/STAG. Data jsou získána pomocí asynchronního požadavku třídy `ASIHTTPRequest`.

- **ContactPickerWrapperViewController** je kontroler řídící obrazovku s kontakty. Kontroler je potomkem třídy `UIViewController`. Kontakty jsou imlementovány třídou `ABPeoplePickerNavigationController`, kterou je možné nalézt ve frameworku Apple AddressBookUI. Po zvolení kontaktu je metodou „push“ prezentována obrazovka `ABPersonViewController`, která implementuje zobrazení detailů kontaktu.

- **PreferencesController** je kontroler řídící obrazovku s nastavením aplikace. Kontroler je potomkem třídy `UITableViewController` a obsahuje statickou tabulku s možnostmi nastavení aplikace.

V následujícím výpisu jsou uvedeny a stručně popsány další kontrolery aplikace. Tyto kontrolery řídí obrazovky, které jsou zpříštupněny pomocí výše uvedených kontrolerů. Jedná se o tyto kontrolery:

- **ShowScheduleController** je kontroler řídící obrazovku se zobrazením rozvrhu a implementuje zobrazení rozvrhu, odeslání rozvrhu emailem s využitím třídy `MFMailComposeViewController` a možnost exportování rozvrhu do kalendáře.

- **IdToShowControllerDynamic** je kontroler jejíž vzhled je nakonfigurován dynamicky z kódu. Tento kontroler zobrazuje na mapě následné rozvrhové akce učitele nebo studenta s daným STAG ID.

- **SubjectsController** je kontroler řídící obrazovku s tabulkou, která obsahuje předměty. Tento kontroler také obsluhuje uživatelovu volbu pro provedení akce s předmětem.

- **FilterTableViewController** je kontroler řídící obrazovku s tabulkou, která obsahuje volby filtrace předmětů.

- **ExamTermsController** je kontroler řídící obrazovku s tabulkou zkouškových termínů. Tento kontroler implementuje exportování zkouškových termínů do kalendáře a obsluhuje uživatelovu volbu pro provedení akce s termínem.
- **PeopleTableViewController** je kontroler řídící obrazovku s tabulkou studentů nebo učitelů. Tento kontroler implementuje odeslání dat z tabulky na email a obsluhuje uživatelovu volbu pro provedení akce s učitelem nebo studentem.
- **StudentsInformationController** je kontroler řídící obrazovku s tabulkou informací o studentovi.
- **TeachersInformationController** je kontroler řídící obrazovku s tabulkou informací o učitelovi.

6.3.2. Ostatní třídy

V následujícím výpisu jsou uvedeny ostatní třídy, které nejsou spojeny se vzhledem. Aplikace obsahuje tyto třídy:

- **ASIHTTPRequest** je třída nad Apple API `CFNetwork` pro zjednodušení práce s webovými službami. Implementuje také asynchronní provedení požadavků.
- **SBJson** je třída obsahující metody pro získání dat z formátu JSON.
- **MapViewAnnotation** je třída, která reprezentuje bod na mapě.
- **SSKeychain** je třída, která pracuje se systémovou iOS klíčenkou pro uchování uživatelského jména a hesla.
- **NSError** je třída, pomocí které dochází k ošetření chybových stavů, nastávajích zejména při výpadku připojení internetu.
- **BSKeyboardControls** je třída, která je vyjímkou mezi výše uvedenými třídami. Implementuje horní panel u klávesnice přihlašovací obrazovky. Pomocí tohoto panelu je možné přepínat mezi textovými poli pro zadání přihlašovacích údajů, případně klávesnici uzavřít.

6.3.3. Apple Frameworky

Následuje tabulka použitých Apple Frameworků se stručným popisem. Informace byly získány z [7].

Framework	**Prefix**	**Od iOS verze**	**Popis**
UIKit.framework	UI	2.0	Obsahuje třídy a metody pro řízení uživatelského rozhraní.
Foundation.framework	NS	2.0	Obsahuje rozhraní pro řízení datových typů.
Security.framework	Sec	2.0	Obsahuje rozhraní pro řízení zabezpečení.
QuartzCore.framework	CA	2.0	Obsahuje rozhraní Core Animation.
CoreText.framework	CT	3.2	Obsahuje funkce pro rozložení a vykreslení textu.
MessageUI.framework	MF	3.0	Obsahuje rozhraní pro vytvoření a zaslání emailových zpráv.
CoreLocation.framework	CL	2.0	Obsahuje rozhraní pro detekování polohy uživatele.
AddressBook.framework	AB	2.0	Obsahuje rozhraní pro přístup ke kontaktům.
AddressBookUI.framework	AB	2.0	Obsahuje třídy pro zobrazení kontaktů.
MapKit.framework	MK	3.0	Obsahuje třídy potřebné pro použití a zobrazení mapy v aplikaci.
CFNetwork.framework	CF	2.0	Obsahuje rozhraní pro přístup k síti.
CoreGraphics.framework	CG	2.0	Obsahuje rozhraní pro Quartz 2D.
MobileCoreServices.framework	UT	3.0	Definuje unifikovaný typ identifikátorů (UTI) podporovaných systémem.
SystemConfiguration.framework	SC	2.0	Obsahuje rozhraní pro zjištění síťového nastavení přístroje.

Tabulka 1. Tabulka použitých Apple Frameworků v aplikaci StagClient.

7. Uživatelská dokumentace

V této kapitole je popsána aplikace z pohledu uživatele. Jsou zde zobrazeny a popsány jednotlivé obrazovky včetně jednotlivých funkcí aplikace a také způsoby spuštění aplikace.

7.1. Instalace aplikace

Distribuce aplikací probíhá u platformy iOS přes virtuální obchod s aplikacemi App Store. Vznik aplikace byl umožněn univerzitním vývojářským programem, který Apple Inc. poskytuje bezplatně, avšak není zde žádná možnost distribuce aplikací. Z tohoto důvodu je možné aplikaci spustit pouze dvěma způsoby, které jsou popsány dále v textu.

V případě testování aplikace ve zkouškovém období, kdy ještě nejsou k dispozici konečné rozvrhy, jsou zobrazovány pouze prázdné rozvrhy se jménem studenta nebo učitele.

7.1.1. Spuštění aplikace v iOS simulátoru

Aplikace může být spuštěna v iOS simulátoru, který je součástí vývojového prostředí Apple Xcode IDE. Pro spuštění Xcode je nutné mít k dispozici systém Mac OS X. Pro účely testování dostačuje i virtualizovaná verze. Pokud Xcode není v systému nainstalován je možné jej nalézt ve virtuálním obchodě s aplikacemi, který se v Mac OS X nazývá App Store. Zde je možné vývojové prostředí zdarma stáhnout. Projekt byl vytvořen v Xcode verzi 5, proto doporučuji použít tuto nebo vyšší verzi.

Projekt aplikace se nachází v adesáři `CD/src/StagClient/` a je možné jej spustit pomocí souboru `CD/src/StagClient/StagClient.xcodeproj`.

Po otevření projektu je v horní liště Xcode možné nastavit typ simulovaného iOS zařízení. Aplikace vyžaduje jakoukoliv nabízenou verzi zařízení iPhone. Aplikaci je možné spustit pomocí volby Product/Run, případně klávesovou zkratkou Cmd+R.

Simulátor neobsahuje plnou podporu pro práci s mapou nebo export událostí do systémového kalendáře, proto byl také vytvořen screencast zobrazující práci s aplikací. Screencast je možné nalézt v cestě `CD/doc/screencast/`.

7.1.2. Spuštění aplikace v iOS zařízení

Prvním krokem pro spuštění aplikace na reálném zařízení iOS je registrace vývojářského účtu na webové adrese https://developer.apple.com. Dále je nutné se připojit do univerzitního vývojářského programu. Pro připojení do univerzitního vývojářského programu kontaktujte vedoucího vývojařského programu pro zaslání pozvánky. Lze jej kontaktovat na adrese `<michal.krupka@upol.cz>`.

Po připojení se do univerzitního vyvojářského programu je na webové adrese https://developer.apple.com povolen přístup do sekce „Certificates, Identifiers & Profiles“ (je nutné být přihlášen). Zde se nachází základní nastavení vývojářského programu. V této sekci zvolte záložku „Certificates“ a tlačítko „Add“. Zobrazí se nabídka typů certifikátů. Zvolte „iOS App Development“ a tlačítko „Continue“. Dále pokračujte dle pokynů na obrazovce.

Po vytvoření certifikátu prosím kontaktujte vedoucího vývojářského programu s UDID[18] zařízení iOS, na kterém bude aplikace testována. Dále také požádejte, aby iOS zařízení bylo přidáno do „Provisioning Profilu“ s názvem „STAG_Client_Samanek“.

Jakmile je profil aktualizován, stáhněte jej zvolením záložky „Provisioning Profiles“, profilu „STAG_Client_Samanek“ a tlačítka „Download“. V tuto chvíli je potřeba mít nainstalováné vývojové prostředí Apple Xcode IDE. Pokud jej ještě nemáte, postupujte prosím podle 7.1.1. Spuťte projekt aplikace pomocí souboru `CD/src/StagClient/StagClient.xcodeproj`. Otevře se aplikace Xcode s načteným projektem aplikace. Připojte iOS zařízení do počítače a spusťte stažený soubor s profilem.

V Xcode se přihlašte do Vašeho vývojářského účtu pomocí volby v horní liště Xcode/Preferences. Otevře se nastavení Xcode, kde v záložce „Accounts“ je možné spravovat vývojářské účty. Kliknutím na znak „+“ a zvolením „Add Apple ID“ se otevře nové dialogové okno pro přidání vyvojářského účtu.

V Xcode je také nutné povolit vývoj na iOS zařízení. Pro povolení vývoje na iOS zařízení zvolte v horní liště Window/Organizer, případně klávesovou zkratku Shift+Cmd+2. Otevře se „Organizer“, kde zvolte požadované zařízení iOS a klikněte na tlačítko „Use for Development“. Proběhne instalace profilu na zařízení iOS. Po nainstalování profilu na zařízení iOS je v horní liště Xcode možné zvolit kromě iOS simulátorů také spárované iOS zařízení. Instalace aplikace na iOS zařízení proběhne automaticky při jejím spuštění. Aplikaci je možné spustit pomocí volby Product/Run, případně klávesovou zkratkou Cmd+R.

V případě, že se vyskytnou problémy, mě prosím kontaktujte na adrese `<pavelsamanek@gmail.com>`. Podrobnější popis fungování a nastavení vývojařského programu je možné nalézt na webové adrese https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html.

[18] Jedinečný identifikátor Apple zařízení. Je možné jej zjistit např. v aplikaci Apple iTunes po připojení iOS zařízení do počítače.

7.2. Přihlášení

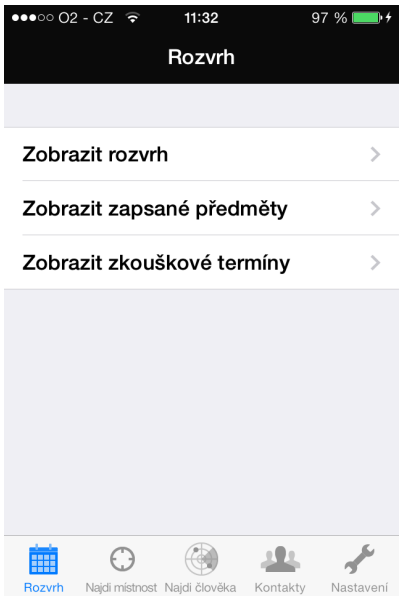
Po startu aplikace je zobrazena přihlašovací obrazovka, viz obrázek 11. Pro práci s aplikací je nutné přihlášení do IS/STAG. Přihlášení je možné pod stejnými přihlašovacími údaji, které jsou používány pro přihlášení do webového rozhraní IS/STAG - Portálu Univerzity Palackého.

Obrázek 11. Úvodní přihlašovací obrazovka aplikace StagClient

Vyplněné přihlašovací údaje lze uložit do systémové klíčenky pomocí přepínače „uložit“.

7.3. Rozvrh

Po úspěšném přihlášení je zobrazena první záložka - obrazovka s nabídkou týkající se rozvrhu. Tato obrazovka je zobrazena na obrázku 12.



Obrázek 12. Obrazovka s nabídkou týkající se rozvrhu

7.3.1. Zobrazit rozvrh

V případě zvolení první volby na obrazovce v první záložce je prezentována nová obrazovka v `landscape`[19] módu s rozvrhem přihlášeného uživatele. S rozvrhem je možné provádět další akce pomocí tlačítka umístěného v pravém horním rohu.

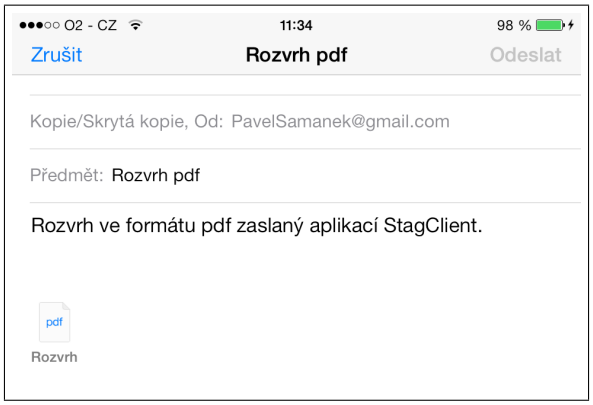
Rozvrh je možné zaslat na email ve formátu PDF nebo ICS. Existuje také možnost jej exportovat do kalendáře telefonu. Obrazovka se zobrazením rozvrhu a příslušnými akcemi je zobrazena na obrázku 13. Na obrázku 13. je zobrazen prázdný rozvrh, nejedná se však o chybu programu, pouze přihlášený uživatel nemá zapsány žádné rozvrhové akce.

V případě zvolení exportu rozvrhu do kalendáře je chod přesměrován do aplikace Apple Kalendář, kde jsou načteny a zobrazeny všechny uživatelovi rozvrhové akce s možností exportu do zvoleného kalendáře. Návrat do aplikace StagClient je možný pouze manuálně.

V případě zvolení odeslání emailem se automaticky chod programu přesune do aplikace Apple Mail s automaticky vytvořenou emailovou zprávou s přílohou v požadovaném formátu, viz obrázek 14. Po odeslání emailové zprávy nebo zrušení odeslání je chod přesunut zpět do aplikace StagClient, konkrétně do obrazovky s načteným rozvrhem. Odtud je možnost se vrátit zpět do hlavní nabídky první záložky programu.

[19] Zobrazení obrazovky v širokoúhlém pohledu

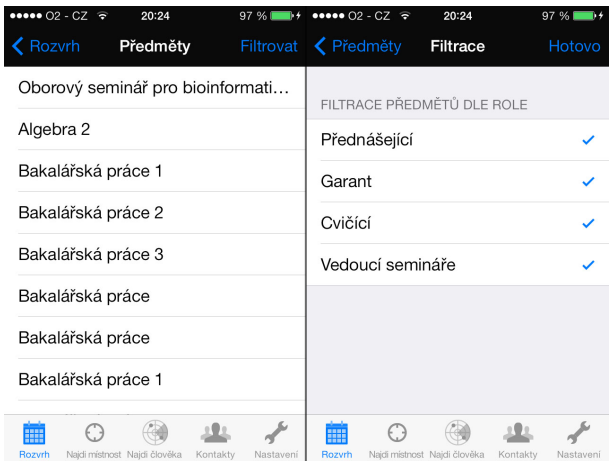
Obrázek 13. Obrazovka zobrazující rozvrh (nahoře) a nabídka s příslušnými akcemi (dole)



Obrázek 14. Automatické vytvoření emailové zprávy s přílohou

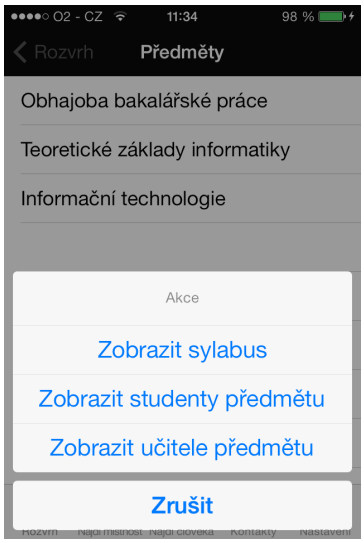
7.3.2. Zobrazit zapsané předměty

Druhá volba na obrazovce první záložky zobrazí seznam zapsaných předmětů (obrázek 15. vlevo). Pokud je přihlášený učitel, v pravém horním rohu se zobrazí tlačítko filtrovat. Zvolením tohoto tlačítka je prezentována nová obrazovka s tabulkou rolí, které je možné zaškrtávat (obrázek 15. vpravo). Filtrace předmětů je realizována na základě zaškrtnutých rolí přihlášeného učitele.



Obrázek 15. Obrazovka zobrazující seznam zapsaných předmětů (vlevo) a možnost filtrace (vpravo)

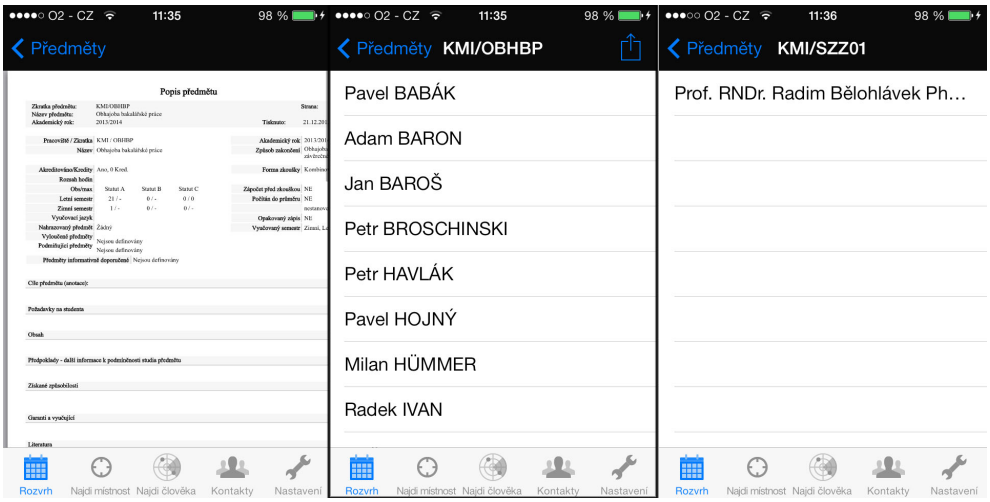
Zvolením předmětu je zobrazena nabídka akcí.



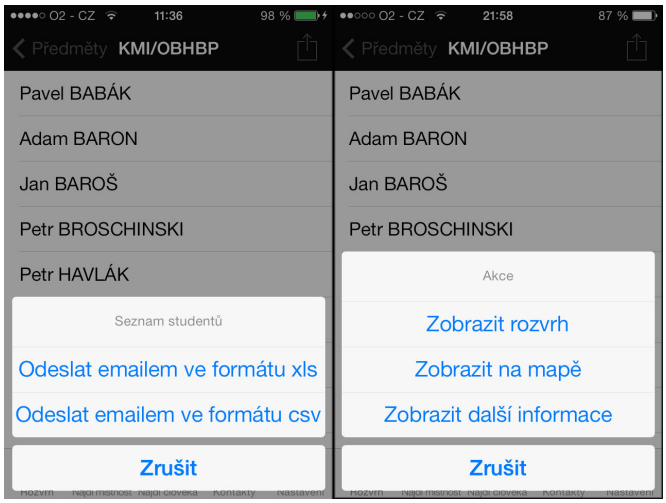
Obrázek 16. Obrazovka zobrazující nabídku akcí předmětu

Pokud je zvolena první možnost, je zobrazena obrazovka se sylabem předmětu (obrázek 17. vlevo). V případě zvolení zobrazení studentů daného předmětu je

prezentována nová obrazovka se seznamem studentů, kteří tento předmět mají zapsán (obrázek 17. uprostřed). Seznam studentů je možné odeslat emailem ve formátu XLS nebo CSV po zvolení tlačítka v pravém horním rohu (obrázek 18. vlevo). Třetí možnost je zobrazení učitele nebo učitelů daného předmětu (obrázek 17. vpravo).



Obrázek 17. Zobrazení sylabu předmětu (vlevo), zobrazení seznamu studentů (uprostřed) a zobrazení učitele daného předmětu (vpravo)



Obrázek 18. Nabídka odeslání seznamu studentů emailem (vlevo) a nabídka akcí, která je zobrazena po výběru studenta nebo učitele (vpravo)

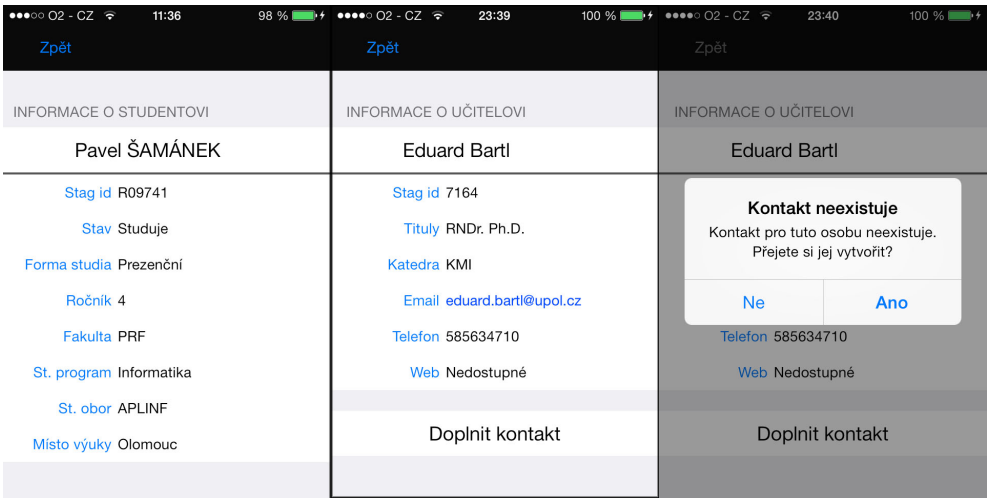
7.3.3. Nabídka akcí

Po zvolení studenta nebo učitele je zobrazena nabídka akcí (obrázek 18. vpravo).

V případě zvolení zobrazení rozvrhu je zobrazena obrazovka s rozvrhem zvoleného studenta nebo učitele. Rozvrh je lze odeslat na email nebo exportovat do kalendáře (podobně jako na obrázku 13.).

Zvolením druhé nabídky je prezentována obrazovka s mapou, kde je zobrazena adresa místnosti následující rozvrhové akce studenta nebo učitele. Na mapě je umístěn popisek obsahující název rozvrhové akce, číslo místnosti a čas začátku rozvrhové akce. V horní části obrazovky je umístěno tlačítko, které umožňuje přesun do aplikce Apple Mapy. V případě přesunu do této aplikace je načtena pozice rozvrhové akce s možností zahájit navigaci, případně provádět další akce. Návrat do aplikace StagClient je možný pouze manuálně. V případě, že žádná další rozvrhová akce neexistuje, je zobrazena aktuální pozice uživatele s popiskem „Volno“.

Třetí možnost zobrazí dostupné informace o studentovi nebo učitelovi z IS/STAG. Informační obrazovka je různá pro roli student a učitel (obrázek 19. vlevo a uprostřed). Informační obrazovka pro roli učitel obsahuje také tlačítko „Doplnit Kontakt“. Po zvolení tohoto tlačítka je možné existující kontakt v adresáři přístroje doplnit o informace z IS/STAG. Pokud kontakt neexistuje, je prezentováno dialogové okno s možností vytvoření nového kontaktu (obrázek 19. vpravo). Nový kontakt je vytvořen na základě informací z IS/STAG. Informační obrazovka dále umožňuje zavolat učiteli, vytvořit novou emailovou zprávu učiteli, případně přesměrovat na učitelovu webovou stránku po zvolení příslušného řádku s údajem. Je nutné, aby požadovaný údaj byl dostupný.



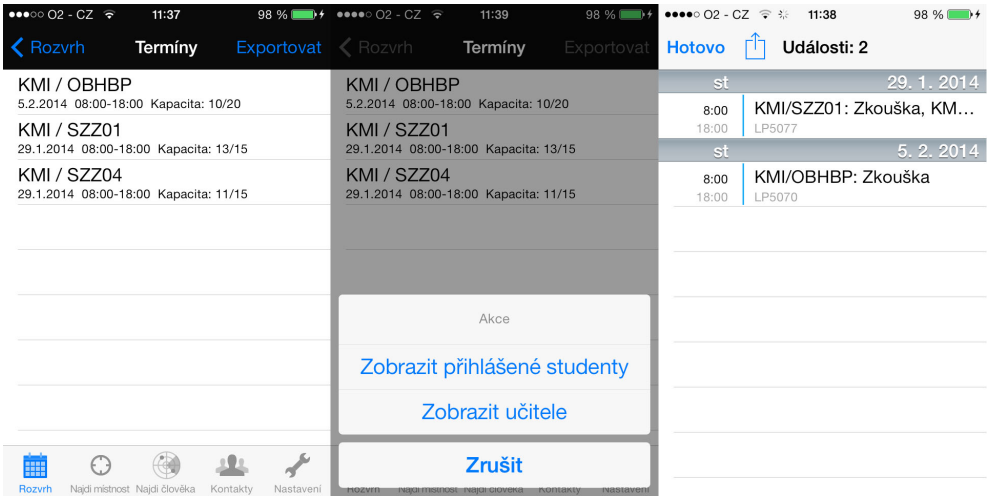
Obrázek 19. Informační obrazovka pro studenta (vlevo), informační obrazovka pro učitele (uprostřed) a možnost doplnit kontakt (vpravo)

7.3.4. Zobrazit zkouškové termíny

Třetí volba na obrazovce první záložky zobrazí seznam termínů zkoušek. Pokud je uživatel student, jedná se o seznam zkouškových termínů, na kterých je student přihlášen. (obrázek 20. vlevo). V případě, že se jedná o učitele, je zobrazen jeho seznam vypsaných termínů zkoušek.

Zkouškové termíny je možné exportovat do kalendáře telefonu po stisknutí tlačítka „Exportovat“ v pravém horním rohu. V případě exportu je chod přesunut do aplikace Apple Kalendář, kde jsou načteny a zobrazeny všechny uživatelovy zkouškové termíny s možností potvrzení exportu do zvoleného kalendáře (obrázek 20. vpravo). Návrat do aplikace StagClient je možný pouze manuálně.

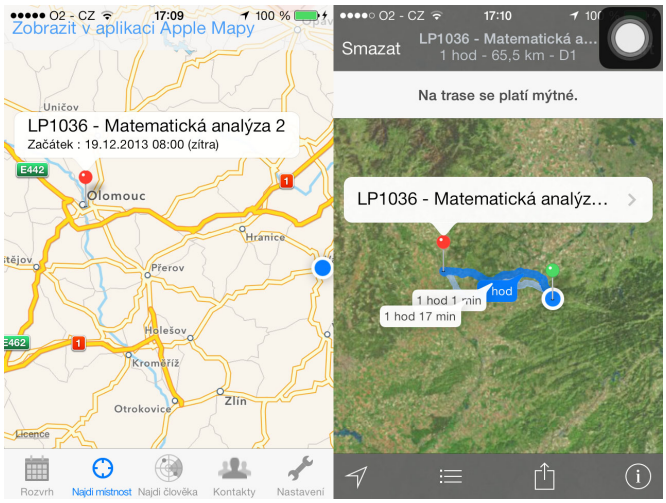
Po zvolení zkouškového termínu jsou zobrazeny akce, které je možné provést (obrázek 20. uprostřed).



Obrázek 20. Obrazovka se seznamem termínů (vlevo), seznam akcí po zvolení termínu (uprostřed) a export do kalendáře (vpravo)

7.4. Najdi místnost

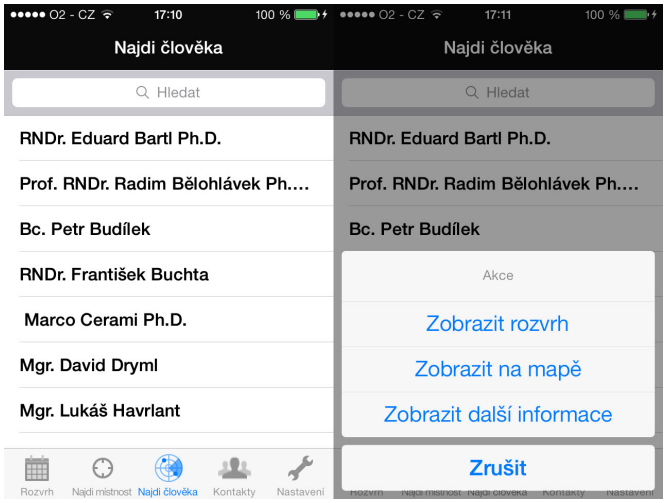
Druhá záložka aplikace obsahuje obrazovku s mapou, kde je zobrazena adresa místnosti uživatelovy následující rozvrhové akce a aktuální poloha uživatele. Na mapě je umístěn popisek obsahující název rozvrhové akce, číslo místnosti a čas začátku rozvrhové akce. V horní části obrazovky je umístěno tlačítko, které umožňuje přesun do aplikce Apple Mapy (obrázek 21. vlevo). V případě přesunu do této aplikace je načtena pozice rozvrhové akce s možností zahájit navigaci, případně provádět další akce (obrázek 21. vpravo). Návrat do aplikace StagClient je možný pouze manuálně. V případě, že žádná další rozvrhová akce neexistuje, je zobrazena aktuální pozice uživatele s popiskem „Volno“.



Obrázek 21. Obrazovka druhé záložky s mapou následující rozvrhové akce uživatele (vlevo) a otevřená aplikace Apple Mapy s pozicí rozvrhové akce (vpravo)

7.5. Najdi člověka

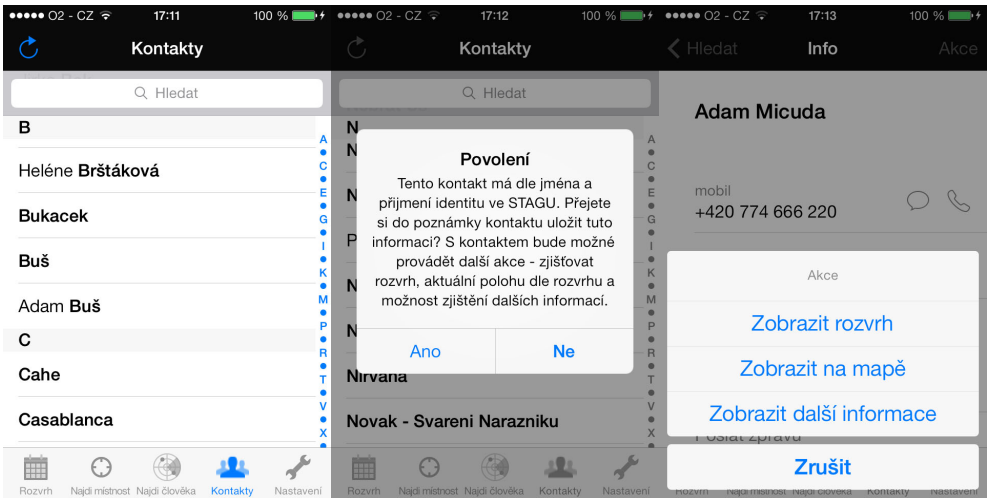
Třetí záložka aplikace obsahuje seznam učitelů z kateder všech zapsaných předmětů, pokud se jedná o studenta (obrázek 22.). Pokud se jedná o učitele, je zobrazen seznam učitelů stejné katedry. Po zvolení učitele ze zobrazen panel akcí popsaný v sekci 7.3.3.



Obrázek 22. Obrazovka třetí záložky se seznamem učitelů (vlevo) a příslušné akce (vpravo)

7.6. Kontakty

Čtvrtá záložka obsahuje adresář kontaktů z přístroje od uživatele (obrázek 23. vlevo). Aplikace po zvolení kontaktu na základě jména a příjmení zjistí, zda kontakt má identitu v IS/STAG. Pokud ano, je zobrazeno dialogové okno tázající se uživatele, zda si přeje doplnit STAG ID k tomuto kontaktu (obrázek 23. uprostřed). V případě kladné odpovědi je STAG ID uloženo do poznámky kontaktu a je prezentována obrazovka s dalšími informacemi o kontaktu. Tato obrazovka také poskytuje možnost provedení akcí s kontaktem stisknutím tlačítka v pravém horním rohu. Jednotlivé akce jsou popsány v sekci 7.3.3.



Obrázek 23. Obrazovka s kontakty (vlevo), dialogové okno po označení kontaktu, který má identitu v IS/STAG (uprostřed) a akce kontaktu, který má identitu v IS/STAG (vpravo)

7.7. Nastavení

Poslední záložka spíše vyhrazuje místo pro budoucí možné úpravy aplikace. V tuto chvíli obsahuje pouze možnost změny STAG ID.

Závěr

Cílem bakalářské práce bylo vyvinout mobilní aplikaci k IS/STAG Univerzity Palackého pro platformu iOS a stručně popsat základy programování pro tuto platformu.

V textové části bakalářské práce je popsána platforma iOS, základy programovacího jazyka Objective-C a základní prostředky pro vývoj aplikací na platformě iOS.

Vznikla aplikace StagClient, která zjednodušuje přístup ke studijní agendě. Aplikace je kompatibilní s mobilním telefonem Apple iPhone od verze 3GS a vyšší. Do budoucna se počítá s rozšířením pro Apple iPad. Funkčnost aplikace je závislá na webových službách IS/STAG. K získání dat byly použity již vytvořené webové služby. Byla také zahájena komunikace se správci IS/STAG Univerzity Palackého pro eventuální doplnění, případně vytvoření zcela nových přímo pro tuto aplikaci. V budoucích verzích aplikace se tedy očekává rozšířená funkčnost. Nabízí se také možnost distribuce aplikace přes virtuální obchod App Store. Není vyloučeno rozšíření aplikace pro ostatní univerzity používající IS/STAG.

Conclusions

The aim of this thesis was to develop a mobile application that interacts with IS/STAG Palacky University for the iOS platform and briefly describe the basics of programming for this platform.

In the text part of the thesis is described the platform iOS, basics of programming language Objective-C and basic tools for developing applications for the iOS platform.

Application StagClient was created. Its purpose is to simplify access to the university agenda. The application is compatible with Apple iPhone 3GS or newer. There is also planned an extension that would deliver Apple iPad compatibility. The functionality of the application is dependent on IS/STAG web services. Application uses only existing web services but communication with the administrators of IS/STAG Palacky University has been initiated to complete or create entirely new webservices. It is expected that new versions of the application will deliver more functions and things to do as webservices get update. There is a possibility of managing application's distribution via the App Store that is used to distribute applications for iOS platform. It is also possible that application might be extended to work with other universities that uses IS/STAG.

Reference

[1] *Cocoa Fundamentals Guide* [online], poslední úprava 13.12.2010 https://developer.apple.com/legacy/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html.

[2] N. Smyth: *Objective-C 2.0 Essentials,* CreateSpace Independent Publishing Platform, 2012, ISBN 978-1475256352.

[3] *Developer tools overview* [online], https://developer.apple.com/technologies/tools/.

[4] D. Mark, J. Nutting, J. LeMarche: *Beginning iOS 5 Development: Exploring the iOS SDK*, Apress, 2011, ISBN 978-1-4302-3606-1.

[5] Pierre Chatelier: *From C++ to Objective-C* [online], http://chachatelier.fr/programmation/fichiers/cpp-objc-en.pdf.

[6] *Webové služby nad IS/STAG* [online], http://stagservices.upol.cz/.

[7] *iOS Technology Overview* [online], https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/iPhoneOSFrameworks/iPhoneOSFrameworks.html.

A. Obsah přiloženého CD

Přiložené CD má následující adresářovou strukturu:

```
CD/
├── bin
│   └── StagClient.app ..... Sestavená aplikace
├── doc
│   ├── doc-src ..... Zdrojový text bakalářské práce v LaTeX
│   │   └── screenshots ..... Obrázky vložené do textové části
│   ├── screencast ..... Složka se screencastem použití aplikace
│   └── StagClient_Samanek.pdf ..... Textová část bakalářské práce
├── src ..... Zdrojové soubory aplikace (projekt Xcode IDE)
└── readme.txt ..... Instrukce pro spuštění aplikace
```

U veškerých odjinud převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro materiály, u kterých toto není splněno, je uveden jejich zdroj (webová adresa) v textu dokumentace práce nebo v souboru `readme.txt`.