VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KLIENT PRO ZOBRAZOVÁNÍ INFORMACÍ Z INFORMAČNÍHO SYSTÉMU FIT

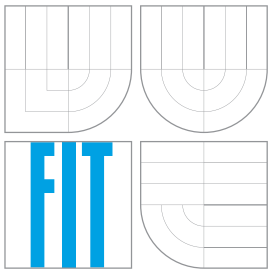
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB PELIKÁN

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KLIENT PRO ZOBRAZOVÁNÍ INFORMACÍ Z INFORMAČNÍHO SYSTÉMU FIT

CLIENT FOR DISPLAYING INFORMATION FROM THE FIT INFORMATION SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAKUB PELIKÁN
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR LAMPA
SUPERVISOR

BRNO 2015

Abstrakt

Cílem bakalářské práce je vytvořit grafickou aplikaci v programovacím jazyce Python 3, která bude studentům Fakulty informačních technologií VUT v Brně přehledně zobrazovat informace o zapsaných předmětech a automaticky je upozorňovat na vzniklé změny v systému. Informace jsou získávány z informačního systému prostřednictvím XML dokumentu, který aplikace validuje a zpracovává pomocí vlastního nástroje. Aplikace je určena pro platformy Linux, MacOS X a Windows.

Abstract

The aim of my bachelor thesis is to create the graphical application in programming language Python 3. This application will display to students of the Faculty of information technology VUT information about the courses and will automatically send notification to students about the system changes. Data will be extracted from information system through XML Document which will be validated and processed by the application internal tool. Application is compatible with Linux, MacOS X and Windows.

Klíčová slova

GUI, Informační systém FIT, VUT, Python 3, PyQt4, XML Dokument, XML Schéma

Keywords

GUI, FIT Information System, VUT, Python 3, PyQt4, XML Document, XML Schema

Citace

Jakub Pelikán: Klient pro zobrazování informací
z informačního systému FIT, bakalářská práce, Brno, FIT VUT v Brně, 2015

Klient pro zobrazování informací z informačního systému FIT

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Lampy. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Pelikán
6. května 2015

Poděkování

Rád bych poděkoval panu Ing. Petru Lampovi za poskytnuté konzultace a odborné vedení práce.

Obsah

Kapitola 1

Úvod

Studenti Fakulty informačních technologií VUT v Brně mají možnost přistupovat k informacím a dosaženým výsledkům v zapsaných předmětech prostřednictvím informačního systému, do kterého přistupují prostřednictvím webového rozhraní. V rámci této práce bude vytvořena aplikace v programovacím jazyce Python 3, která bude uživateli poskytovat kompletní proces získání a validace informací z informačního systému. Získané informace budou následně zobrazeny prostřednictvím přehledného grafického uživatelského rozhraní. Navíc aplikace bude umožňovat ve zvoleném časovém intervalu automatické stahování informací z informačního systému a vyhledávání změn v zapsaných předmětech. Uživatel bude o vzniklých změnách informován.

Práce se nejdříve obecně věnuje struktuře a validování dokumentů ve formátu XML, ve kterém jsou informace z informačního systému dostupné. Na kapitolu o formátu XML navazuje třetí kapitola, která se věnuje základní a rozšířené množině informací, které jsou k dispozici ve dvou XML dokumentech přístupných v informačním systému, a vytvoření chybějícího XML schématu pro dokument s rozšířenou množinou informací. V rámci čtvrté kapitoly je popsáno, jakým způsobem probíhá a je implementován proces validace a zpracování XSD dokumentu, který obsahuje XML schéma, a následná validace a zpracování XML dokumentu. Výsledkem zpracování XML dokumentu je stromová struktura informací o předmětech, která je zobrazována pomocí grafického uživatelského rozhraní.

Páta kapitola se zaměřuje na volbu grafické knihovny a její verze, dále na strukturu, popis a implementaci jednotlivých částí, ze kterých se skládá grafické uživatelské rozhraní. V šesté kapitole je popsán proces kontroly změn v informačním systému a implementace stahování, vyhodnocování a informování uživatele o změnách. Další kapitoly popisují, jakým způsobem si uživatel může přizpůsobit aplikaci svým požadavkům - implementace barevných schémat určujících vzhled aplikace, vytvoření jazykových verzí, uložení a zabezpečení osobního nastavení.

Desátá kapitola obsahuje možnosti vytvoření distribučních balíčků a instalace aplikace za pomocí implementovaného instalačního skriptu. V poslední kapitole je popsáno, jak probíhalo testování implementace validace a zpracování XML dokumentu a testování uživatelského rozhraní z hlediska multiplatformního použití.

Kapitola 2

Formát XML

Formát XML je rozšířitelný značkovací jazyk (*Extensible Markup Language*), který slouží pro popis struktury dat. Nepopisuje jejich zobrazení. Data jsou formátována pomocí tagů, které nejsou předdefinovány v žádném XML standardu, ale pro každý XML dokument jsou podle potřeby vymyšleny specifické tagy autorem XML dokumentu. Formátování dat pomocí XML se využívá pro snadné sdílení a přenos dat mezi rozdílnými aplikacemi a platformami [1].

2.1 Struktura XML

XML dokument obsahuje na prvním řádku deklaraci, po které následuje stromová struktura, která začíná `root` elementem a větví se pomocí `child` elementů.

2.1.1 Deklarace XML

Deklarace XML je uvedena na prvním řádku. Typicky obsahuje verzi XML a kódování dokumentu.

```
<?xml version="1.0" encoding="UTF-8"?>
```

2.1.2 Root element

Po deklaraci XML první tag je `root` element, který označuje, jaký typ dokumentu bude v souboru. Poslední tag je konec `root` elementu, který označuje konec dokumentu.

```
<person>
...
</person>
```

2.1.3 Child element

Po `root` elementu jsou v dokumentu `child` elementy. `Child` element je jakýkoliv element umístěný mezi počátečním a koncovým tagem `root` elementu.

```
<person>
  <firstname>Jakub</firstname>
  <lastname>Pelikan</lastname>
</person>
```

2.1.4 Jmenný prostor (*namespaces*)

Jmenný prostor umožňuje nezávisle na sobě používat několik druhů značkování v rámci jednoho dokumentu. Jmenný prostor je označován pomocí prefixu uvedeného před jménem elementu. Jmenný prostor je definován pomoci `xmlns` atributu, který je uveden v počátečním tagu elementu. Deklarace jmenného prostoru má syntaxi `xmlns:prefix="URI"` [2].

```
<root>
  <h:table xmlns:h="http://www.abc.org/TR/1">
    <h:firstname>Jakub</h:firstname>
  </h:table>
  <j:table xmlns:j="http://www.abc.org/TR/2">
    <j:commodity>paper</j:commodity>
  </j:table>
</root>
```

Jmenný prostor může být deklarován i v `root` elementu.

```
<root xmlns:h="http://www.abc.org/TR/1" xmlns:j="http://www.abc.org/TR/2">
  <h:table>
    <h:firstname>Jakub</h:firstname>
  </h:table>
  <j:table >
    <j:commodity>paper</j:commodity>
  </j:table>
</root>
```

2.2 Validování XML dokumentů

K validování XML dokumentů slouží XML schéma, ve kterém je popsaná struktura XML dokumentu. XML schéma definuje elementy a atributy, které se mohou vyskytovat v dokumentu, jejich hierarchii a datové typy. Dále definuje, které elementy jsou prázdné a které mohou obsahovat text [3]. XML schémata můžou být popsána například pomocí jazyka DTD nebo XSD.

2.2.1 DTD

DTD (*Document Type Definition*) je jazyk pro popis struktury XML dokumentů. DTD definuje strukturu dokumentu pomocí seznamu korektních elementů a atributů. K zápisu pravidel DTD používá vlastní syntaxi.

```
<!ELEMENT course (title,points,accr,grade)>
<!ELEMENT title (#PCDATA)>
...
```

2.2.2 XSD

XSD (*XML Schema Definition*) je jazyk pro popis struktury XML dokumentů. XSD je novější alternativa k DTD. K zápisu je na rozdíl od DTD použita syntaxe XML, což ulehčuje vytváření validačních souborů. XSD je v případě potřeby snadno rozšiřitelný, podporuje datové typy a jemné prostory. Celkově je bohatší a více využitelný než DTD.

Kapitola 3

Získávání informací z IS FIT

3.1 Informace ve formátu XML

V informačním systému FIT jsou k dispozici dva dokumenty ve formátu XML obsahující informace o předmětech zapsaných v aktuálním školním roce. První dokument `get-curses.php`[1] obsahuje základní informace o zapsaných předmětech v daném školním roce. XML schéma pro tento dokument je implementováno v XSD dokumentu `courses.xsd`[2]. Druhý XML dokument `get-cursesx.php`[3] obsahuje navíc oproti prvnímu dokumentu podrobné informace a informace o aktivitách v zapsaných předmětech. Tento dokument bude využíván implementovanou aplikací. Vzhledem k tomu, že neexistuje XML schéma potřebné k jeho validaci a zpracování, bude toto schéma vytvořeno v rámci této práce a popsáno v kapitole 3.2.

3.1.1 Množina základních zobrazovaných informací z IS FIT

Z XML dokumentu `get-courses.php` dostupného v informačním systému lze získat seznam všech zapsaných předmětů s jejich základními informacemi v českém jazyce.

- Id předmětu
- Zkratka předmětu
- Typ předmětu
- Způsob zakončení předmětu
- Počet aktuálně získaných bodů
- Počet kreditů
- Datum a čas poslední změny údajů
- Udělení zápočtu
- Datum poslední změny udělení zápočtu
- Jméno osoby, kterou byl udělen zápočet

[1] Dostupný online na `https://wis.fit.vutbr.cz/FIT/st/get-courses.php` [cit. 2014-11-20].
[2] Dostupný online na `http://www.fit.vutbr.cz/schemas/courses.xsd` [cit. 2014-11-20].
[3] Dostupný online na `https://wis.fit.vutbr.cz/FIT/st/get-coursesx.php` [cit. 2014-11-20].

3.1.2 Množina rozšířených zobrazovaných informací z IS FIT

XML dokument `get-coursesx.php` obsahuje základní množinu zobrazovaných informací popsanou v kapitole 3.1.1, která je rozšířena o podrobnější informace o předmětech a obsahuje informace o aktivitách, které v daném předmětu probíhají. Jsou to například projekty, půlsemestrální a semestrální zkoušky, cvičení a další aktivity spojené s výukou v daném předmětu. Ukázka z rozšířeného XML dokumentu se nachází v příloze C.

Rozšířené informace

- Csid předmětu
- Letní/Zimní semestr
- Název předmětu v českém a anglickém jazyce

Informace o aktivitách

- Název aktivity v českém jazyce
- Číslo aktivity
- Id aktivity
- Čas a datum zahájení a ukončení aktivity
- Čas a datum začátku a konce registrace aktivity
- Datum a čas poslední změny údajů o aktivitě
- Minimum, maximum a počet získaných bodů z dané aktivity
- Počet přihlášených studentů
- Jméno osoby, která udělila body za danou aktivitu
- Aktivita může obsahovat různé varianty, např. výběr z více zadání projektu nebo více termínů cvičení
- Způsob a podmínky registrace na aktivitu, např. zaregistrováni automaticky všichni nebo registruje se sám student, pokud dostal zápočet

3.1.3 Zhodnocení množiny zobrazovaných informací z IS FIT

Množina rozšířených zobrazovaných informací ve formátu XML obsahuje všechny základní informace o předmětu a aktivitách, které v něm probíhají. Tyto informace bych pouze rozšířil o časové údaje, ze kterých by bylo možno sestavit individuální rozvrh. Některé aktivity, například projekty, obsahují potřebné informace, ale většina aktivit je neobsahuje vůbec nebo v nedostatečné formě, například v názvu aktivity. Dále bych u aktivit probíhajících v prostorách školy doplnil i místo konání.

Časové informace (datum a čas začátku a konce)

- Pravidelných aktivit v daném předmětu, např. přednášky, pravidelná cvičení.
- Jednorázových nebo nepravidelných aktivit v daném předmětu, např. projekty, půlsemestrální a semestrální zkoušky, jednorázová cvičení nebo democvičení.
- Zimního, letního semestru a příslušných zkouškových období.

3.2 XML schéma pro XML get-coursesx.php

Při získávání informací z informačního systému využívám rozšířený XML dokument `get-coursesx.php`, ke kterému ale neexistuje XML schéma. Aby bylo možné při používání rozšířeného XML ověřit jeho validitu, implementoval jsem příslušné XML schéma a uložil jej do XSD souboru, jehož ukázka se nachází v příloze B. Při implementaci jsem vycházel z XML schématu pro XML dokument `get-courses.php`. XML dokument `get-coursesx.php` obsahuje množinu informací, které jsou v XML dokumentu `get-courses.php`, ale je rozšířena o podrobnější informace a informace o aktivitách v daném předmětu. Původní XML schéma jsem rozšířil o validaci informací, o které je rozšířené XML rozšířeno. Přidané validace jsou uvedeny v kapitole 3.2.1, na kterou navazuje kapitola 3.2.2, která popisuje textové komentáře, kterými jsou validace vhodně doplněny.

3.2.1 Přidané validace

- Povinně volitelných předmětů - skupina H,C,O,S,B,M,I,F,N,L,D,G,E
- Zimního a letního semestru
- Csid předmětu
- Aktivit
  - Způsob a podmínky registrace
  - Typ položky aktivity:
    - Výběr z jedné varianty
    - Výběr jedné z variant
    - Výběr více variant
    - Zápočet
  - Validace dalších podrobností o aktivitách, které jsou popsány v kapitole 3.1.2

3.2.2 Textové komentáře

Textové komentáře v XSD schématu jsou použity pro komentování významu zkratek, které jsou použity v XML. Všechny komentáře jsou v českém i anglickém jazyce.

```
<xsd:enumeration value="P">
  <xsd:annotation>
    <xsd:documentation xml:lang="cs">Povinný předmět</xsd:documentation>
    <xsd:documentation xml:lang="en">Compulsory courses</xsd:documentation>
  </xsd:annotation>
</xsd:enumeration>
```

Kapitola 4

Validace a zpracování informaci z IS FIT

Pro validaci a zpracování jsem vytvořil vlastní balíček, který kontroluje, zda XML dokument odpovídá XML schématu, které je načteno z příslušného XSD dokumentu. Nejprve se provede validace a zpracování XSD dokumentu a poté se pomocí vzniklé stromové struktury zpracovaného XML schématu validuje a zpracovává XML dokument. Data získaná z XML dokumentu budou poté zobrazena prostřednictvím grafického rozhraní uživateli. Pro načítání a zpracování zdrojových souborů jsem implementoval XML parser.

4.1 XML parser

Parser provádí lexikální analýzu vstupních souborů a rozděluje je na posloupnost tokenů. Při lexikální analýze je rozpoznáváno pět základních symbolů. Rozpoznané symboly jsou ukládány do proměnné typu slovník. Při inicializaci proměnné jsou definovány klíče, ke kterým jsou ukládány příslušné rozpoznané symboly.

{ `'startTag', 'singleTag', 'tagAttribute', 'endTag', 'text'` }

- Nepárový tag

  `<person/>`

  – `'startTag':'person', 'singleTag':True`

- Začátek párového tagu

  `<person>`

  – `'startTag':'person', 'singleTag':False`

- Konec párového tagu

  `</person>`

  – `'singleTag':False, 'endTag':'person'`

- Text mezi párovými tagy

```
<person>Text</person>
```

– `'text':'Text'`

- Atributy u párových a nepárových tagů

```
<person age="20"/>
```

– `'startTag':'person', 'singleTag':True, 'tagAttribute':'age':'20'`

```
<person age="20">
```

– `'startTag':'person', 'singleTag':False, 'tagAttribute':'age':'20'`

Pro práci s parserem jsou implementovány metody, které jsou invokovány podle potřeby.

- `getToken()` - metoda vrací naposledy zpracovaný token
- `getNewToken()` - metoda zpracuje a vrátí nový token

4.2 Validace a zpracování XSD dokumentu

XSD dokument obsahuje XML schéma, které je složeno z pevně definovaných elementů. Data načtená z tohoto dokumentu jsou zpracovávána pomocí implementovaného XML parsru. Jestliže je načtený token umístěn v počátečním nebo nepárovém tagu, tak se nejprve v seznamu validních elementů ověří, zda načtený element je v seznamu elementů. Po nalezení elementu jsou zpracovány jeho parametry. Poté se ověřuje, zda nadřazený prvek, který je umístěn na vrcholu zásobníku, může být rodičovským prvkem zpracovávaného elementu. Následně je do jeho seznamu `child` elementů přidán zpracovávaný element. Jestliže zpracovávaný element je načten z párového tagu, umístí se na vrchol zásobníku. V případě, že je zpracovávaný element v koncovém tagu, ověří se, zda na vrcholu zásobníku je umístěn příslušný počáteční element, který je poté odebrán z vrcholu zásobníku.

Po zpracování celého vstupního souboru je zkontrolován zásobník, zdali je prázdný. Poslední element, který byl odstraněn z vrcholu zásobníku, je `root` element, který je na vrcholu stromové struktury celého zpracovaného souboru a prostřednictvím kterého lze přistupovat ke všem elementům.

4.2.1 Implementace Elementů

XSD dokument má na rozdíl od XML dokumentu pevně definované elementy, které mají přesně vymezené nadřazené elementy, definované atributy a specifické chování. Při implementaci těchto elementů jsem využil dědičnosti tříd. Implementoval jsem hlavní třídu, která obsahuje metody pro zpracování jednotlivých atributů a metody, které obsahují sdílené chování většiny elementů. Pro každý element jsem implementoval vlastní třídu, která je potomkem hlavní třídy. Některé metody sdíleného chování jsem redefinoval podle potřeb daného elementu. Každá třída obsahuje seznam validních nadřazených elementů, seznam validních atributů a jednu metodu, která určuje specifické chování pro danou třídu. Při implementaci jsem vycházel ze specifikace elementů `W3C XML schema Definition Language (XSD) 1.1 Part 1: Structures`[1]. Specifikace obsahuje popis chování, defaultní hodnoty a seznam validních předků elementů.

[1] Dostupné online na http://www.w3.org/TR/xmlschema11-1/ [cit. 2014-11-20].

4.2.2 Datové typy

Datové typy specifikují, jakých hodnot můžou data nabývat. Datové typy jsem implementoval jako samostatnou třídu, která podle ověřovaného datového typu invokuje příslušnou metodu, která pomocí regulárních výrazů ověří, zda zadaná hodnota odpovídá danému datovému typu. Při implementaci jsem vycházel z `W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes`[2]. Rozlišuji 4 základní skupiny datových typů.

- `Date and Time Data Types` - Časové datové typy.
  - `Date, DateTime, Duration, GDay, GMonth, GMonthDay, GYear, GYearMonth, Time`
- `String Data Types` - Textové datové typy.
  - `ENTITIES, ENTITY, ID, IDREF, IDREFS, Language, Name, NCName, NMTOKEN, NMTOKENS, NormalizedString, QName, String, Token`
- `Numeric Data Types` - Číselné datové typy.
  - `Byte, Decimal, Int, Integer, Long, NegativeInteger, NonNegativeInteger, NonPositiveInteger, PositiveInteger, Short, UnsignedLong, UnsignedInt, UnsignedShort, UnsignedByte`
- `Miscellaneous Data Types` - Smíšené datové typy.
  - `AnyURI, Base64Binary, Boolean, Double, Float, HexBinary, NOTATION, QName`

4.2.3 Omezení pro datové typy

Omezení pro datové typy jsou speciální elementy, které omezují použití elementů datových typů. Tato omezení jsou implementována stejným způsobem jako implementace elementů. Každé omezení je implementováno v samostatné třídě, která dědí obecné vlastnosti od stejné třídy jako elementy.

4.2.4 Zásobník

Pro potřeby validace XSD dokumentu jsem implementoval zásobník, na který se ukládají zpracovávané elementy. Zásobník jsem implementoval formou třídy, která obsahuje metody pro práci se zásobníkem. Jednotlivé elementy jsou ukládány v uspořádaném seznamu, kdy nově vkládaný element je přidán na konec seznamu.

[2] Dostupné online na http://www.w3.org/TR/xmlschema11-2/ [cit. 2014-11-20].

4.3 Validace a zpracování XML dokumentu

Zpracovávání a validace XML dokumentu probíhá pomocí stromové struktury zpracovaného XSD dokumentu. Nejprve je inicializován `root` element, kterému se předá třída pro předávání hodnot. Poté je invokací příslušné metody spuštěna činnost `root` elementu, ve kterém začne zpracovávání XML dokumentu. Prochází se strom vytvořený z XSD dokumentu a jednotlivé třídy elementů vykonávají implementovanou činnost, při které zpracovávají tokeny, invokují metody jiných tříd uložených ve validačním stromu nebo ukládají zpracovaná data. Jestliže `root` element zpracuje úspěšně poslední token, proběhlo zpracování i validace XML souboru úspěšně.

4.3.1 Předávání hodnot mezi elementy

Pro uložení sdílených dat mezi elementy jsem implementoval třídu, do které lze uložit všechny potřebné hodnoty. Použití třídy významně omezí počet atributů, které se musí mezi elementy předávat při invokaci metod, a ulehčí přidávaní a upravování nových parametrů, které je potřeba předávat mezi elementy.

4.3.2 Rozhraní pro práci se získanými daty

Rozhraní pro práci se získanými daty umožňuje procházet a pracovat s daty, které obsahoval XML dokument. Jako první hodnota ve vytvořeném stromu je `root` element, jehož prostřednictvím lze přistupovat k celému stromu.

- `xmlTree.attrib` - obsahuje atributy elementu
- `xmlTree.child` - obsahuje seznam `child` elementu
- `xmlTree.text` - obsahuje text elementu
- `xmlTree.parrent` - obsahuje rodiče elementu
- `xmlTree.printTree()` - vypíše všechny elementy ve stromové struktuře, jejich atributy a text
- `xmlTree.returnChildTree()` - vrátí všechny elementy ve stromové struktuře, jejich atributy a text

Kapitola 5

Zobrazování zpracovaných informací z IS FIT

5.1 Výběr knihovny pro GUI

Při výběru knihovny pro tvorbu grafického uživatelského rozhraní (GUI) jsem se rozhodoval mezi dvěma nejpopulárnějšími multiplatformními knihovnami GTK a Qt. Obě knihovny splňovaly potřebné vlastnosti pro vytvoření GUI, které bude zobrazovat informace z IS FIT. Mezi hlavní požadované vlastnosti patřily existence jejich ekvivalentních verzí PyQt a PyGTK pro programovací jazyk Python 3 a možnost jejich použití na platformách typu Linux, Windows a MacOS X. Nejprve jsem pro implementaci zvolil knihovnu PyGTK, ale pro finální verzi jsem zvolil druhou knihovnu PyQt, jejíž použití mi přišlo intuitivnější a dostupná dokumentace přehlednější a rozsáhlejší.

5.1.1 Knihovna PyQt4

Pro implementaci GUI jsem zvolil knihovnu PyQt4, která vychází z knihovny Qt verze 4.8. PyQt4 jsem zvolil i přesto, že již existuje její novější verze PyQt5, jejíž použití se doporučuje pro vývoj nových aplikací. Starší verzi jsem zvolil, protože v novější verzi se vyskytují problémy se zobrazováním ikony aplikace. Ikona aplikace se v některých grafických prostředích zobrazuje na špatném místě nebo se nezobrazuje vůbec. Některé z těchto problémů již byly vyřešeny, ale některé se teprve řeší. Ikona aplikace bude popsána v kapitole 5.2.3.

5.2 Struktura a implementace GUI

Při prvním spuštění aplikace je zobrazena uvítací obrazovka a průvodce aplikací. Poté je spuštěna samotná aplikace, která umožňuje přes ikonu aplikace zobrazovat přehled předmětů, upravovat nastavení aplikace a provádět kontroly změn v informačním systému. O změnách je uživatel informován prostřednictvím upozornění.

5.2.1 Implementace okna aplikace

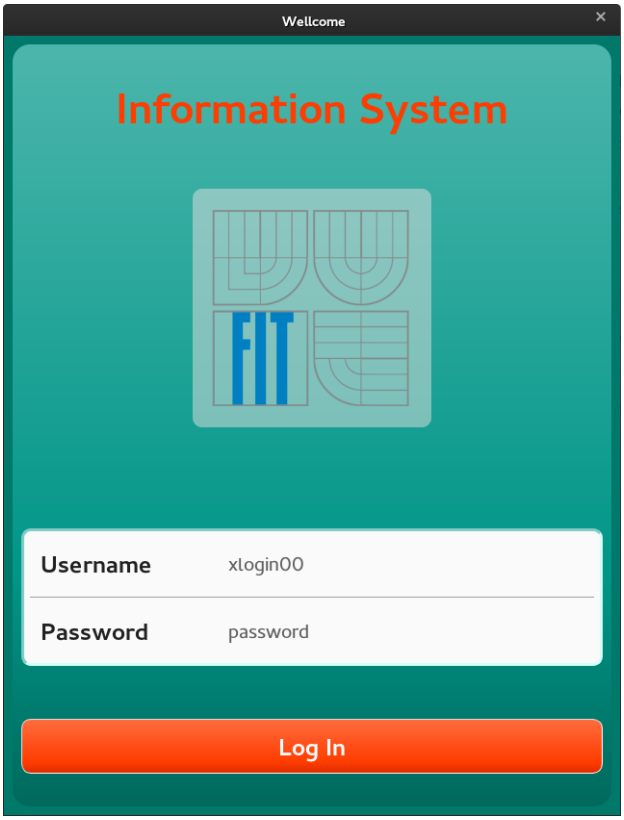
Hlavní okno aplikace, ve kterém se zobrazuje přehled předmětů nebo nastavení aplikace, je implementováno jako potomek třídy `QMainWindow`. Tato třída poskytuje framework pro vytváření aplikací s grafickým uživatelským rozhraním. Hlavní částí je centrální widget, bez kterého `QMainWindow` nelze vytvořit [9]. Centrální widget obsahuje instanci třídy `QWidget`

nebo její potomky. Třída `QWidget` je základní třídou, ze které vycházejí všechny objekty uživatelského rozhraní. Widget je základní prvek uživatelského rozhraní, přijímá od `window system` signály, například z myši nebo klávesnice, a zobrazuje sám sebe na obrazovce. Všechny widgety jsou obdélníky, které jsou připnuty na svého rodiče a na widgety před ním [12]. Pro zobrazování jednotlivých položek jsem implementoval potomka třídy `QWidget`, který zachytává signál stisknutí a následného uvolnění tlačítka myši, na základě kterého zobrazuje nebo skrývá widget obsahující podnabídku a příslušně volí mezi definovanými styly vzhledu. Položky jsou umístěny v `QScrollArea`, která v případě, že velikost v ní umístěných widgetů přesáhne její aktuální velikost, zobrazí posuvník a umožní posouvat svůj obsah. Ve spodní části hlavního okna je pod `QScroolArea` umístěn widget obsahující tlačítka určená k ovládání okna.

5.2.2 Uvítací obrazovka a průvodce aplikací

Uvítací obrazovka

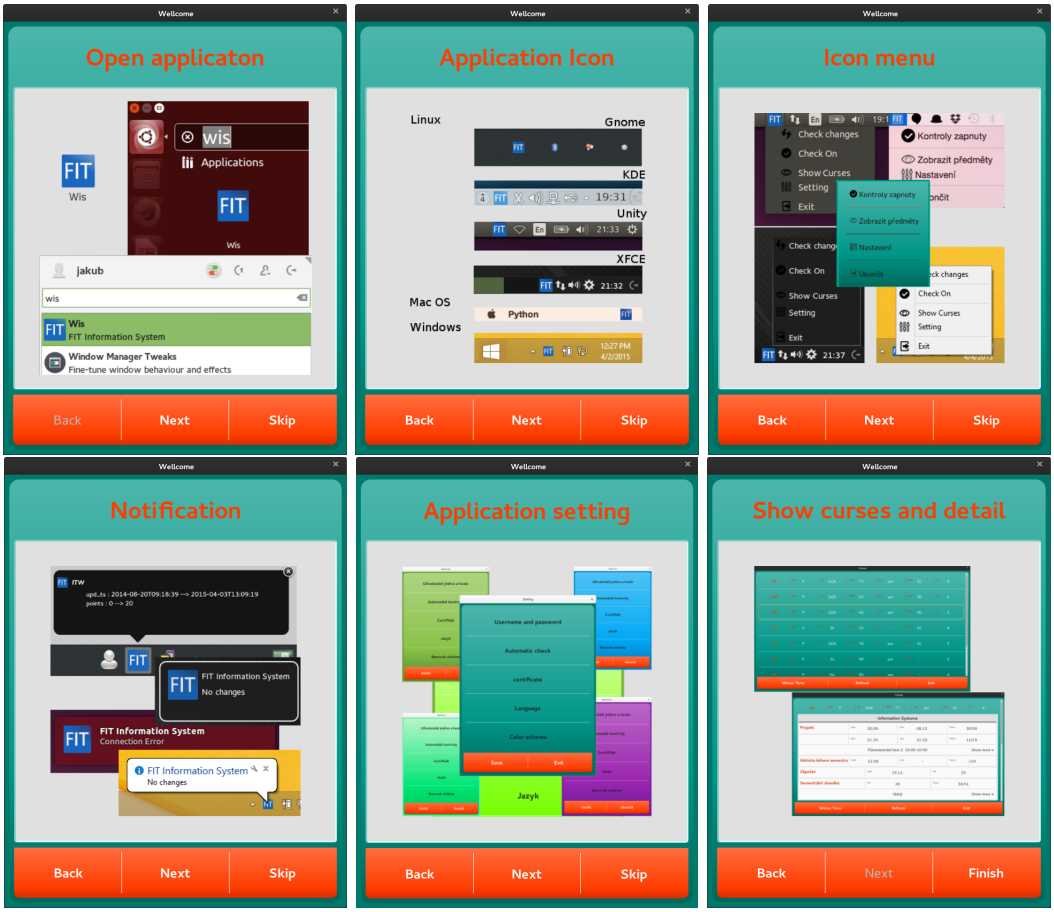
Při prvním spuštění aplikace se nejprve zobrazí uvítací obrazovka, která vyžaduje po uživateli zadání uživatelského jména a hesla pro přihlášení do informačního systému FIT. Uvítací obrazovka je zobrazena na obrázku 5.1. Po zadání přihlašovacích údajů se aplikace pokusí přihlásit do systému a tím ověřit platnost zadaných údajů. Proces přihlášení je popsán v kapitole 6.3. Po úspěšném přihlášení se vytvoří osobní nastavení, ve kterém jsou uloženy přihlašovací údaje a nastavení aplikace. Osobnímu nastavení se podrobně věnuje kapitola 9. Poté se zobrazí průvodce aplikací.



Obrázek 5.1: Přihlášení do IS FIT

Průvodce aplikací

Po prvním úspěšném přihlášení se uživateli zobrazí průvodce aplikací. Tento průvodce formou obrázku a textových popisků představuje jednotlivé části aplikace, prostřednictvím kterých je uživatel seznámen s chováním aplikace, s jejím použitím a možnostmi, jak aplikaci přizpůsobit svým požadavkům pomocí jejího nastavení. Jednotlivé kroky jsou zobrazeny na obrázku 5.2. Po dokončení průvodce aplikací se spustí samotná aplikace.



Obrázek 5.2: Průvodce aplikací

5.2.3 Ikona aplikace

Ikona aplikace je implementována jako potomek třídy `QSystemTrayIcon`, která umožňuje přidat ikonu aplikace do speciální oblasti na ploše nazývané oznamovací oblast nebo system tray, kde dlouhodobě běžící aplikace můžou zobrazovat ikony a zprávy [10]. Umístění a vzhled oznamovací oblasti se na jednotlivých operačních systémech a prostředích mírně liší. Zobrazení v rozdílných oznamovacích oblastech je zachycené na obrázku 5.3. Prostřednictvím ikony aplikace má uživatel možnost rychlého vstupu do aplikace nebo přímého použití některých funkcí, například zobrazení nastavení nebo ukončení aplikace.

- Inicializace potomka `QSystemTrayIcon`.

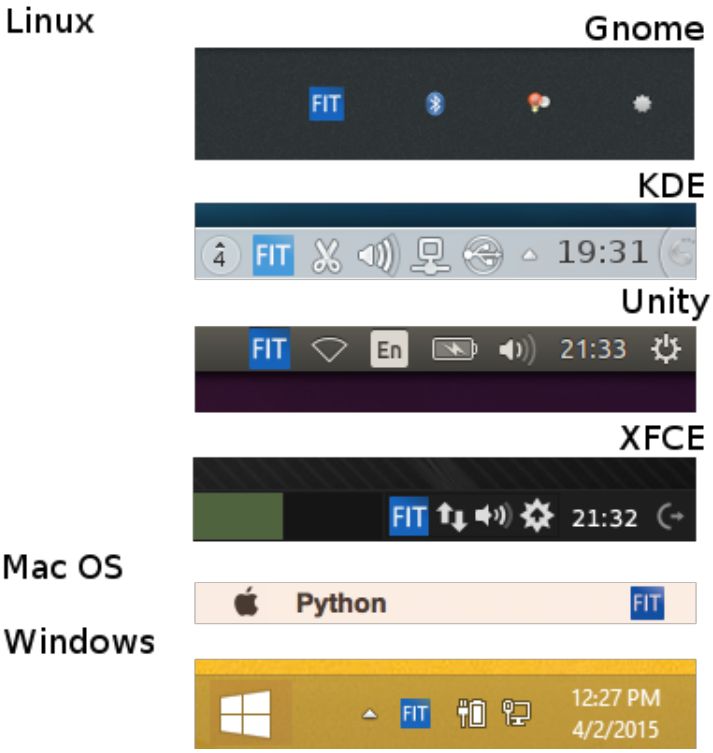
```
class SystemTrayIcon(QtGui.QSystemTrayIcon):
  def __init__(self, setting, parent=None):
    QtGui.QSystemTrayIcon.__init__(self, parent)
    ...
```

- Nastavení zobrazované ikony.

```
self.setIcon(QtGui.QIcon(self.setting['userSetting'].getImage('logo')))
```

- Přidání menu zobrazeného pravým kliknutím myší na ikonu.

```
self.setContextMenu(RightClickMenu(self.setting))
```



Obrázek 5.3: Zobrazení ikony aplikace v různých systémech a prostředích

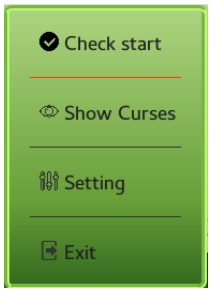
Akce levého kliku myší

Při kliknutí levým tlačítkem myši na ikonu aplikace se spustí vlákno kontroly změn v informačním systému. Kontrole změn se věnuje kapitola 6. Uživatel je o výsledku kontroly informován prostřednictvím upozornění, které je popsáno v kapitole 5.2.6. V případě, že byly nalezeny změny v IS, upozornění obsahuje zobrazení všech změn, jinak je uživatel upozorněn, že v systému neproběhly žádné změny.

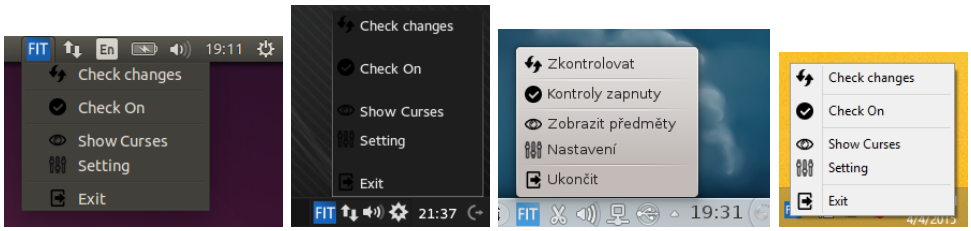
Akce pravého kliku myší

Při kliknutí pravým tlačítkem myši na ikonu aplikace se zobrazí menu s výběrem možných akcí. Menu je implementováno ve dvou vzhledových provedeních, vlastní a systémové. Jako výchozí je použito systémové provedení, které využívá systémový vzhled. Menu s vlastním vzhledem má definovaný vzhled pomocí `Qt Style Sheets` a odpovídá vzhledu aplikace. Definováním a používáním vlastního vzhledu se zabývá kapitola 7. Menu s vlastním vzhledem je zobrazeno na obrázku 5.4 a ukázky systémového provedení menu jsou zobrazeny na obrázku 5.5.

- Aktualizovat - Funkcionalita volby je totožná jako při levém kliknutí na ikonu aplikace.
- Kontroly Zapnuty/Vypnuty - Volba zapnutí a vypnutí automatické kontroly změn v systému. Automatické kontroly jsou popsány v kapitole 6.6. Položka zobrazuje uživateli aktuální stav automatických kontrol. Při najetí myši na položku se zobrazí stav kontrol, který by byl zvolen, jestliže by uživatel klikl na položku. Jestliže uživatel potvrdí volbu, zapne/vypne automatické kontroly, jinak se při opuštění myši z položky vrátí text do původního stavu.
- Zobrazit Předměty - Zobrazí okno obsahující přehled předmětů.
- Nastavení - Zobrazí okno obsahující nastavení aplikace.
- Ukončit - Ukončí aplikaci.



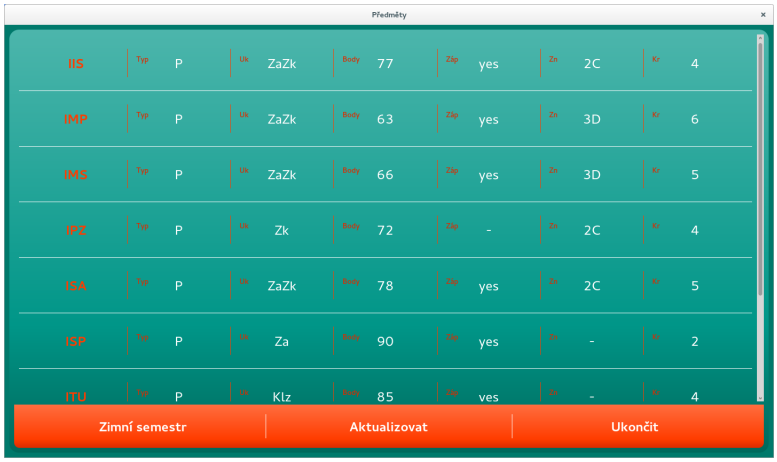
Obrázek 5.4: Menu s vlastním vzhledem



Obrázek 5.5: Menu se systémovým vzhledem

5.2.4 Přehled předmětů

V okně přehled předmětů je přehledně zobrazen seznam předmětů zimního nebo letního semestru se základními informacemi. Základní informace obsahují zkratku předmětu, typ předmětu, způsob ukončení, počet aktuálně získaných bodů, stav udělení zápočtu, aktuální známku a počet kreditů. Tyto informace jsou přehledně rozčleněny a doplněny o příslušné popisky (obrázek 5.6). Při kliknutí na řádek s příslušným předmětem se zobrazí podrobné informace o předmětu popsané v kapitole 3.1.2. Ukázka zobrazení podrobných informací je na obrázku 5.7. Vytváření položek s jednotlivými předměty a jejich detailními informacemi probíhá na základě zpracovaného XML dokumentu. Pomocí rozhraní pro práci se získanými daty, kterému se věnovala kapitola 4.3.2, je procházena vytvořená stromová struktura a podle elementů, které obsahuje, se vytváří příslušné položky. Uživatel má pomocí tlačítek umístěných ve spodní části okna možnost volby mezi zimním a letním semestrem, aktualizace zobrazených informací a zavření okna přehledu předmětů.



Obrázek 5.6: Přehled předmětů

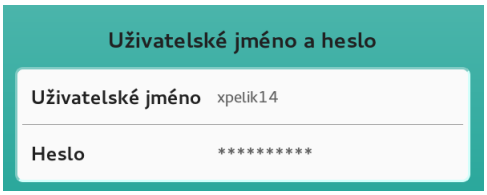
Obrázek 5.7: Podrobné informace o předmětu

5.2.5 Nastavení aplikace

Okno nastavení aplikace lze zobrazit pomocí položky v menu ikony aplikace. Umožňuje uživateli podrobnější nastavení a přizpůsobení aplikace.

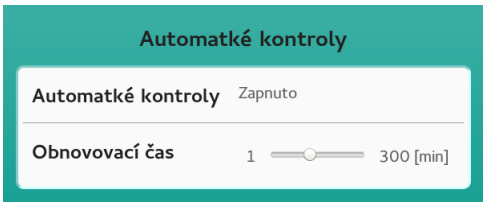
Obrázek 5.8: Nastavení aplikace

- Uživatelské jméno a heslo - Umožňuje změnu uživatelského jména a hesla.



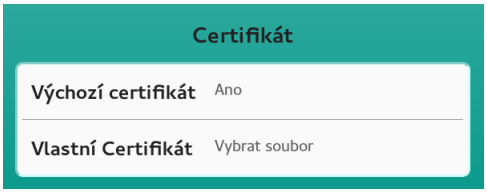
Obrázek 5.9: Nastavení uživatelského jména a hesla

- Automatické kontroly - Nastavení automatických kontrol a časového intervalu mezi jednotlivými kontrolami. Automatické kontroly budou dále popsány v kapitole 6.6.



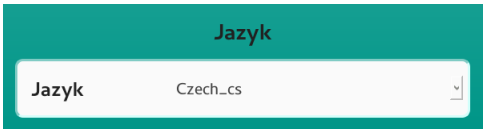
Obrázek 5.10: Nastavení automatických kontrol

- Certifikát - Nastavení použití výchozího certifikátu pro přihlášení do IS FIT nebo zvolení vlastního certifikátu.



Obrázek 5.11: Nastavení používaného certifikátu

- Jazyk - Nastavení jazyka aplikace. Dostupné jazykové verze a jejich implementace je popsána v kapitole 8.



Obrázek 5.12: Nastavení jazyka aplikace

- Barevná schémata - Výběr barevného schématu grafického uživatelského rozhraní a volba použití tohoto schématu i na menu, které je zobrazované u ikony aplikace. Tvorbě barevných schéma se věnuje kapitola 7.

Obrázek 5.13: Volba barevného schématu

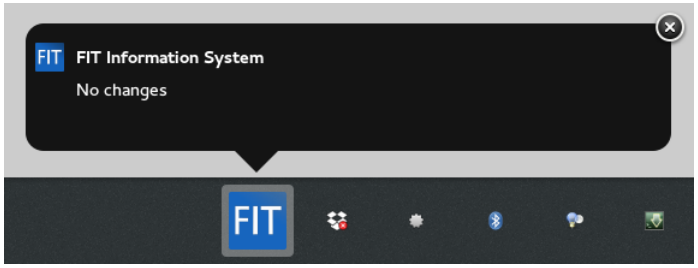
5.2.6 Upozornění

Prostřednictvím upozornění je uživatel informován o stavu změn v informačním systému a o případných chybách, které mohou nastat, např. chyba připojení k síti. Typ upozornění se liší podle použité platformy. Na platformě typu Linux jsou upozornění zobrazována primárně pomocí *Desktop notifications*, pokud systém toto upozornění nepodporuje nebo je aplikace spuštěna na platformě jiného typu, jsou použity standardní upozornění z knihovny Qt.

Desktop notifications

Desktop notification je malé pasivní okno, které upozorní uživatele na konkrétní událost [8]. Tento typ upozornění jsem zvolil na platformě typu Linux jako primární, protože okno upozornění používá výchozí vzhled systému, takže uživatel bude upozorněn takovým způsobem, jakým je zvyklý od ostatních aplikací. U upozornění můžeme definovat prioritu, dobu expirace, ikonu a text zprávy [14].

```
os.system("notify-send -u low \""+title+"\" \"\t"+msg+"\n\" --icon="+icon)
```

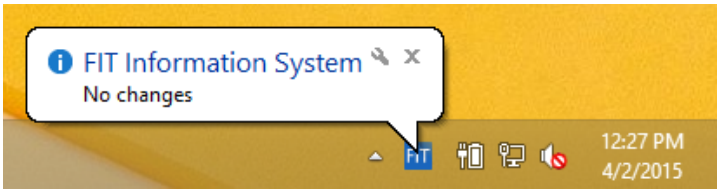


Obrázek 5.14: Desktop notifications (Gnome 3.14)

Qt standardní upozornění

Status notification message jsou standardní, tzv. balonové zprávy z knihovny Qt. Tyto zprávy se zobrazí k ikoně aplikace pomocí příkazu `showMessage()`. U balonových zpráv můžeme definovat název, zprávu, ikonu a dobu platnosti [11]. Standardní upozornění jsou kromě platformy typu Linux používána primárně a vzhledově na těchto platformách odpovídají upozorněním, které běžně používají i jiné aplikace.

```
self.tray.showMessage(title, msg, QtGui.QSystemTrayIcon.Information)
```



Obrázek 5.15: Qt standardní upozornění (Windows 8)

Kapitola 6

Kontrola změn v informačním systému

Třída, která obsahuje implementaci kontroly změn v informačním systému, je potomek třídy `QThread`. Třída je doplněna o implementaci stažení dat ze systému a jejich následné zpracování pomocí balíčku pro validaci a zpracování, vyhledání změn a informování uživatele na vzniklé změny pomocí upozornění. O dokončení kontroly změn je objekt, který spustil vlákno kontroly, informován pomocí signálů, které mohou být v objektu napojeny na sloty.

6.1 Třída QThread

Třída `QThread` umožňuje spravovat vlákna nezávisle na platformě. `QThread` objekt spravuje jedno vlákno, které pomocí příkazu `start()` zahájí svoji činnost invokací metody `run()`. Používání třídy `QThread` je možné více způsoby, například přidáním objektu `QObject` do vlákna pomocí příkazu `QObject.moveToThread()` nebo mnou použitým způsobem, kdy implementuji potomka třídy `QThread` a reimplementuji metodu `run()`.

- Vytvoření potomka třídy QThread.

```
class CourseData(QtCore.QThread):
  success = QtCore.pyqtSignal()
  loginFailed = QtCore.pyqtSignal()
  ...

  def __init__(self, setting, notifyNoChanges=False):
    QtCore.QThread.__init__(self,None)
    ...

  def run(self):
    ...
```

- Spuštění vlákna objektu potomka QThread.

```
self.thread = CourseData(self.setting)
self.thread.start()
```

6.2 Signály a Sloty

Signály a sloty se používají pro komunikaci mezi objekty. Při programování GUI, když změníme jeden widget, často chceme, aby jiný widget byl na změnu upozorněn. Jeden signál muže být připojen na více slotů a na jeden slot může být napojeno více signálu. Signály jsou vyslány, když objekt nějakým způsobem změní svůj stav. Slot je klasická funkce, na kterou je připojený signál. Když je signál vyslán, jsou zavolány všechny funkce, které jsou na něj připojeny [13].

6.2.1 Definování vlastních signálů

PyQt definuje všechny standardní signály, nové signály mohou být definovány jako třídní atributy pomocí `pyqtSignal()`. Widgety, které jsou ovlivňovány kontrolou změn v systému, mají podle potřeby implementovány sloty, které jsou napojeny na nové signály. Signály upozorňují widgety na úspěšné stažení a zpracování dat z informačního systému, případně na chybu přihlášení nebo chybu spojení.

- Definování nových signálů.

```
loginFailed = QtCore.pyqtSignal()
```

- Vyslání signálů.

```
self.loginFailed.emit()
```

- Připojení slotů na nové signály.

```
self.thread.loginFailed.connect(self.loginFailed)
```

6.3 Stahování informací z IS FIT

Jako zdroj informací používám rozšířený XML dokument popsaný v kapitole 3.1, který je dostupný v informačním systému. Ke stažení XML dokumentu z IS je potřeba uživatelské jméno, heslo a bezpečnostní certifikát. Vzhledem k použití bezpečnostního certifikátu jsem zvolil obsáhlou knihovnu `httplib2`, která umožňuje přihlášení uživatele za použíti bezpečnostního certifikátu.

- Nastavení cesty k certifikátu a časového limitu.

```
h=httplib2.Http(ca_certs = self.setting['userSetting'].cert['location'],
 timeout=60)
```

- Nastavení uživatelského jména a hesla.

```
h.add_credentials(username, password)
```

- Pokus o získání dat ze zadané adresy. Jestliže operace proběhne úspěšně, informace o operaci jsou uloženy v `resp` a získaná data v `content`. Při neúspěchu operace je vyvolána příslušná výjimka.

```
resp, content = h.request(xmlUrl, "POST", body="foobar")
```

- Ověření, zda bylo přihlášení úspěšné, probíhá pomocí ověření status kódu, který je k dispozici v `resp`. Při úspěšném přihlášení je status kód `OK 200` a při špatně zadaných přihlašovacích údajích `Unauthorized 401`. Ostatní status kódy nejsou rozlišovány a pokus o stažení informací je považován za neúspěšný.

```
if 'status' in resp and resp['status'] == '200':
  ...
elif 'status' in resp and resp['status'] == '401':
  ...
else:
  ...
```

6.4 Vyhledání změn

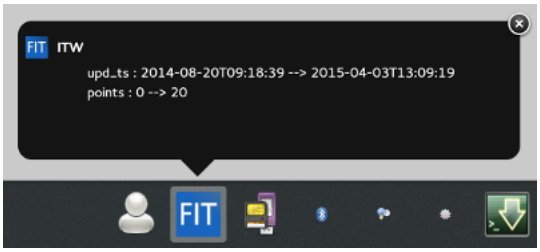
Při vyhledávání změn se porovnávají uložené a aktuálně stažené informace. Nejprve jsou porovnána nezpracovaná data, jestliže nově stažená data obsahují změnu, jsou zpracována a poté jsou pro jednotlivé předměty vyhledány změny, na které je uživatel upozorněn. Data obsahující změnu jsou uložena pro příští vyhledávání změn.

```
diff = []
for x in courses.attribute:
  if courses.attribute[x] != coursesOld.attribute[x]:
    diff.append(x)
```

6.5 Zobrazení změn

Porovnáním stávajících a nově stažených dat z informačního systému získáme pro každý předmět seznam změn. Jestliže je seznam změn neprázdný, vytvoříme text zprávy, který bude obsahovat všechny změny v daném předmětu. O vzniklých změnách informuje aplikace uživatele pomocí upozornění, které bylo popsáno v kapitole 5.2.6. Název předmětu, v kterém vznikly změny, bude použit jako nadpis a zpracovaný seznam změn bude obsahem zprávy upozornění. Ukázka upozornění na změnu je zobrazena na obrázku 6.1.

```
if diff != [] or diff1 != []:
  msg = '\t'
    for x in diff:
      msg = msg + x + ' : ' + coursesOld.attribute[x]
        + ' --> ' + courses.attribute[x] + '\n\t'
  self.notify.courseNotify(courses.attribute['abbrv'], msg)
```



Obrázek 6.1: Zobrazení změn pomocí upozornění

6.6 Automatická kontrola

Uživatel má možnost povolit automatické kontroly změn v informačním systému a nastavit interval mezi kontrolami. Automatické kontroly jsou spouštěny pomocí časovače `QTimer` jako samostatné vlákno kontroly změn. Při zapnutí automatických kontrol je nastaven časovač na časový interval zadaný uživatelem. Po uplynutí nastaveného intervalu je spuštěno vlákno kontroly a časovač zahájí nový odpočet. Pokud při uplynutí intervalu časovače je vlákno kontroly změn stále aktivní, je další kontrola vynechána. Při ukončení programu nebo vypnutí automatické kontroly je časovač zastaven.

- Spuštění nového procesu.

```
def autoCheckStart(self):
  self.autoCheckThreadStart()
  self.threadInterval = QtCore.QTimer(self)
  self.threadInterval.timeout.connect(self.autoCheckThreadStart)
  self.threadInterval.start(self.setting['userSetting'].autoCheck['time']
    *60000)
```

- Spuštění vlákna kontroly změn.

```
if self.thread == None or self.thread.isFinished():
  self.thread = CourseData(self.setting,self.notify)
  self.thread.notifySend.connect(self.notify.sendMessage)
  self.thread.start()
```

- Ukončení automatických kontrol.

```
def autoCheckStop(self):
  if self.threadInterval != None:
    self.threadInterval.stop()
    self.threadInterval = None
```

Kapitola 7

Barevná schémata

Pro definování vzhledu jednotlivých prvků, ze kterých se skládá GUI, jsem použil `Qt Style Sheets`. Tím jsem oddělil vzhled jednotlivých prvků od samotného kódu, což mi umožnilo jednoduše vytvořit více barevných vzhledů. Uživatel si může vybrat ten, který mu bude nejvíce vyhovovat, nebo si jednoduše vytvořit vlastní vzhled, který po přidání do složky `fitIS/gui/style` bude k dispozici v nastavení.

7.1 Qt Style Sheets

`Qt Style Sheets` je mocný mechanizmus, který umožňuje definovat vlastní vzhled widgetů. Koncept, terminologie a syntaxe `Qt Style Sheets` je inspirovaná `HTML Cascading Style Sheets`, ale přizpůsobená potřebám widgetů. Informace o `Qt Style Sheets` jsem čerpal z kapitoly Qt Style Sheets v Qt dokumentaci[1].

Vzhled může být nastaven pro:

- Celou aplikaci

```
QApplication.setStyleSheets('style_file')
```

- Na konkrétní widget

```
QWidget.setStyleSheets('style_file')
```

7.2 Syntaxe Qt Style Sheets

`Qt Style Sheets` jsou složeny ze sekvencí stylových pravidel. Tyto pravidla jsou složena ze selektoru a deklarace. Selektor specifikuje, které widgety jsou ovlivněny pravidlem. Deklarace specifikuje, které vlastnosti můžou být nastaveny widgetu.

- V příkladu stylového pravidla je `QComboBox` selektor a { `color: #FAFAFA` } deklarace.

```
QComboBox { color: #FAFAFA }
```

[1] Dostupné online na http://doc.qt.io/qt-4.8/stylesheet.html [cit. 2015-2-3].

Pravidlo může obsahovat více selektorů pro jednu deklaraci, tyto selektory jsou odděleny čárkou.

```
QComboBox, QLabel { background-color: #FAFAFA }
```

`Qt Style Sheets` podporuje všechny selektory, které jsou definované v CSS2. V Implementaci jsou využity:

- Jednoduchý selektor - odpovídá instanci třídy a všem podtřídám.

  ```
  QComboBox { color: #FAFAFA }
  ```

- Třídní selektor - odpovídá instanci třídy, ale neodpovídá podtřídám.

  ```
  .QComboBox { color: #FAFAFA }
  ```

- ID Selektor - odpovídá instanci třídy, která má odpovídající jméno objektu.

  ```
  #settingMainWidget { background-color: transparent }

  QWidget#settingMainWidget { background-color: red }
  ```

  Nastavení jména objektu:

  ```
  self.setStyleSheet('style_file')
  self.mainWidget = QtGui.QWidget()
  self.mainWidget.setObjectName('settingMainWidget')
  ```

- Selektor potomku - odpovídá instanci třídy `QComboBox`, která je potomkem `QWidget`.

  ```
  QWidget QComboBox { color: #FAFAFA }
  ```

- Dílčí prvky (*Sub-Controls*) - například tlačítko `drop-down` z `QComboBox`.

  ```
  QComboBox::drop-down { image: url(image/icon/more.png)}
  ```

- Pseudo-States - speciální stav widgetu, například myš je nad widgetem.

  ```
  QComboBox:hover{
    background-color: #DCEDC8;
    border-radius: 8;
  }
  ```

7.3 Výběr barev

Při výběru a kombinaci barev pro jednotlivá barevná schémata jsem využil doporučení a škálu barev pro Android, Web a iOS, které jsou dostupné na stránkách Google[7], odkud jsem převzal i obrázek 7.1 s ukázkami barevných palet. Každá barevná paleta obsahuje deset základních barev a čtyři doplňkové barvy, které se používají pro zdůraznění například hlavních tlačítek a komponent. Při výběru vlastní palety se doporučuje vybrat tři barvy ze základní palety a jednu barvu z doplňkové palety. Odstín barvy textu se ovlivňuje Alpha kanálem barvy, hodnota alpha se odvíjí od hierarchie informací, standardní text má hodnotu aplha 87%, sekundární text 54%, tipy pro uživatele a nepovolené možnosti 26%.

Obrázek 7.1: Ukázky barevných palet

7.4 Vytvořená schémata

Uživatel má možnost si vybrat ze šesti základních barevných schémat. Všechna schémata mají stejnou oranžovou doplňkovou paletu a dodržují doporučení pro barvu textu. Jednotlivá schémata se liší výběrem primární palety barev. Modré schéma je laděno do odstínů modré podobně jako vzhled informačního systému FIT. Teal (modrozelené) a zelené schéma se skládají z klidných odstínů barev. Světle zelené a modrozelené schéma se skládají ze svítivých výrazných odstínů. Fialové schéma se skládá z klidných odstínů fialové barvy, ale vzhledem k fialové barvě působí celkově velmi výrazně. Všechna barevná schémata jsou volena tak, aby aplikace působila atraktivním a moderním vzhledem.

Obrázek 7.2: Barevná schémata

Kapitola 8

Jazykové verze

V implementaci je oddělen zobrazovaný text od zdrojového kódu aplikace, což nejenže podporuje čistý programovací styl a umožňuje efektivně upravovat zobrazovaný text z celé aplikace na jednom místě, ale také umožňuje snadno vytvářet různé jazykové verze. Informace získané z informačního systému FIT jsou k dispozici v českém a anglickém jazyce, a proto jsem vytvořil pro aplikaci českou a anglickou verzi zobrazovaného textu. Výchozí jazyk je zvolen podle jazyka nastaveného operačnímu systému. Jestliže není jazyk operačního systému dostupný, je zvolen český jazyk. Pro tvorbu jazykových verzí jsem použil modul `gettext`, který umožňuje používání více jazykových verzí.

8.1 Module gettext

GNU gettext je multiplatformní nástroj patřící do projektu GNU určený pro psaní vícejazyčných verzí aplikací [5]. V aplikaci využívám modul `gettext` pro Python 3, který definuje API pro internacionalizaci. Umožňuje oddělit zprávy od zdrojového kódu, vytvořit katalog zpráv obsahující překlady a použít ho k vypisování zpráv uživateli za běhu programu [4].

8.2 Vytvoření a používání jazykových verzí

Proces vytvoření a použití překladu [4].

1. Označení textových řetězců v kódu.

```
import gettext
self.language = gettext.translation('languge_file_name',
    'language_file_location' , fallback=True)
self._ = self.language.gettext
print(self._('backButton'))
```

2. Získání seznamu zpráv, vytvoření `.pot` souboru.

```
xgettext -d wis -o lang.pot ./gui/*.py
```

   Ukázka `.pot` souboru.

```
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2015-03-02 23:25+0100\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: gui/willcomeGui.py:166
msgid "backButton"
msgstr ""
```

3. Vytvoření `.po` souboru a přeložení zpráv.

   ```
   cp lang.pot language/Czech_cs/LC_MESSAGES/lang.po
   ```

   Přeložení zpráv.

   ```
   #: gui/willcomeGui.py:166
   msgid "backButton"
   msgstr "Zpět"
   ```

4. Zkompilování katalogu zpráv z překladu, vytvoření binárního `.mo` souboru.

   ```
   msgfmt -o lang.mo lang.po
   ```

5. Načtení příslušného zkompilovaného katalogu zpráv a vypsání zpráv za běhu programu.

Kapitola 9

Osobní nastavení

Pro uložení a přístup k osobnímu nastavení jsem implementoval samostatnou třídu, která obsahuje veškeré údaje, které uživatel může za běhu aplikace nastavovat, a data ukládaná aplikací. Uživatelem nastavené údaje mohou být například uživatelské jméno a heslo, které je ukládáno v šifrované podobě. Příkladem aplikací ukládaných dat jsou aktuální verze dat stažených z informačního systému. Osobní nastavení se po každé změně ukládá pomocí modulu `pickle` jako serializovaný objekt a po spuštění aplikace je opět načteno.

9.1 Šifrované heslo

Přístup k instančnímu atributu, který obsahuje uživatelské heslo, je implementován promocí `properties`. Při ukládání se heslo nejprve zašifruje a poté uloží. Při přístupu k heslu se hodnota atributu dešifruje. Šifrování a dešifrování je implementováno pomocí standardu AES.

9.1.1 Property

V objektově orientovaných jazycích se často využívají metody `getter` pro načítání a `setter` pro ukládání dat, které zajišťují korektní přístup k privátním instančním atributům. V jazyce Python existuje jejich alternativa nazývaná `property`, která umožňuje přistupovat k privátním atributům stejným způsobem jako k veřejným a při tom definovat, jakým způsobem proběhne načítání a ukládáni dat [6].

- Definování `property`.

```
self.__password

@property
def password(self):
  return self.decrypt(self.__password)

@password.setter
def password(self, passwd):
  self.__password = self.encrypt(passwd)
```

- Přístup k instančnímu atributu.

```
print(self.password)
```

9.1.2 Standard AES

Standard AES je standardizovaný algoritmus používaný k šifrování dat, který šifruje a dešifruje data stejným klíčem [19]. Algoritmus je implementovaný v knihovně `Crypto.Cipher`. Algoritmus za pomocí šifrovacího klíče zašifruje nebo dešifruje heslo. Pro vytvoření šifrovacího klíče je použita funkce pro odvození klíče `PBKDF2`.

```
def encrypt(self,password):
  iv = os.urandom(16)
  key = PBKDF2(self.passphrase, self.salt).read(32)
  cipher = AES.new(key, AES.MODE_CBC, iv)
  return iv + cipher.encrypt(password + ' '*(16 - (len(password) % 16)))

def decrypt(self, ciphertext):
  key = PBKDF2(self.passphrase, self.salt).read(32)
  initVector = ciphertext[:16]
  ciphertext = ciphertext[16:]
  cipher = AES.new(key, AES.MODE_CBC, initVector)
  return cipher.decrypt(ciphertext).decode("utf-8").replace(' ','')
```

Funkce pro odvození klíče PBKDF2

Funkce pro odvození klíče je implementována v modulu `pbkdf2` a upravena pro použití se standardem AES[1]. Klíč je odvozen pomocí kryptografické soli a heslové fráze.

9.2 Uložení Osobního nastavení

Informace v osobním nastavení je potřeba uchovat uložené na disku, aby je bylo možné načíst při dalším spuštění aplikace. Pro ukládání informací jsem zvolil modul `pickle`, který umožňuje uložit a poté načíst objekt obsahující osobní nastavení. Osobní nastavení se načítá při startu aplikace a ukládá po každé změně nastavení.

```
def save(self):
  with open(self.saveFile['fileName'], 'wb') as file:
    pickle.dump(self,file)

def load(self):
  try:
    with open(self.saveFile['fileName'], 'rb') as file:
    return pickle.load(file)
  ...
```

9.2.1 Modul pickle

Modul `pickle` implementuje algoritmus pro serializaci a de-serializaci struktury objektu. Serializace je proces, kdy objektová hierarchie je převedena do bytového streamu, který je možno uložit. De-serializace je inverzní operace, kdy je bytový stream převeden zpátky do objektové hierarchie [20].

[1] Upravený algoritmus je dostupný online na http://stackoverflow.com/questions/7014953/i-need-to-securely-store-a-username-and-password-in-python-what-are-my-options [cit. 2015-2-3].

Kapitola 10

Distribuce a instalace aplikace

Pro sestavování a instalaci balíčků existuje pro jazyk Python 3 více nástrojů. Mezi hlavní nástroje patří například `distutils`, `setuptools` nebo `cx_freeze`. Základem pro všechny nástroje je instalační skript, pomocí kterého provádějí svoji činnost. Vytvořené balíčky je možno distribuovat pomocí Python package Index (PyPi).

10.1 Instalační skript

Instalační skript je centrem všech aktivit při sestavování, distribuci a instalaci python modulů. Hlavním účelem instalačního skriptu je popsat vytvořený modul [18]. Instalační skript se typicky skládá z volání funkce `setup()`, která je volána s parametry zadávanými formou pojmenovaných argumentů. Seznam argumentů se liší podle zvoleného nástroje, ale hlavní argumenty jsou pro všechny společné. Instalační skript se standardně pojmenovává `setup.py`. V implementaci je použit nástroj `setuptools`, který umožňuje pomocí argumentu zadat seznam balíčků, které jsou potřebné k běhu aplikace a jsou součástí PyPI. Tyto balíčky jsou během instalace automaticky staženy a nainstalovány. Pomocí argumentů se zadávají informace, které slouží k popisu balíčku, mezi které patří například jméno, verze, popis, autor, klíčová slova, klasifikace a další. Dále se zadávají argumenty, které jsou důležité pro správné sestavení a instalaci balíčku. Pomocí těchto argumentů se například zadává seznam balíčků, které mají být zahrnuty, vstupní bod, povolení zahrnutí dalších souborů uvedených v manifestu. Vstupní bod slouží při instalaci k vytvoření samostatného spustitelného souboru, přes který je možné spouštět zadanou funkci z vytvářeného balíčku. Typ souboru se automaticky zvolí podle platformy, na kterou je balíček instalován, například `.exe` soubor na platformě typu Windows. V implementaci jsem nastavil, aby vytvořený soubor spustil pomocí metody `main()` ze skriptu `__main__.py` grafické rozhraní pro zobrazování informací z informačního systému.

```
setup(
  name='Wis',
  version='1.0',
  description='Fit information system',
  author='Jakub Pelikan',
  author_email='xpelik14@stud.fit.vutbr.cz',
  keywords=['wis', 'fit', 'vutbr'],
  include_package_data=True,
  packages=['fitIS','fitIS.gui','fitIS.xmlParser'],
```

```
    entry_points = {'gui_scripts': [ 'wis = fitIS.__main__:main'] },
    install_requires=['pbkdf2', 'Crypto','httplib2','pyCrypto'],
)
```

10.1.1 Soubor s manifestem

Při vytváření instalačního balíčku jsou zahrnuty soubory `README, setup.py` a soubory s příponou `.py`, které jsou v modulech uvedených v seznamu parametrů atributu `packages`. K zahrnutí dalších souborů slouží soubor s manifestem `MANIFEST.in`, který je umístěn v kořenovém adresáři projektu. Obsahuje posloupnost příkazů, které umožňují zahrnovat nebo vyřazovat konkrétní soubory a adresáře. Příkazy manifestu zachovávají strukturu adresářů [15]. Pomocí souboru s manifestem jsou do distribuce zahrnuty složky, které obsahují ikony a obrázky, jazykové verze, barevné styly, certifikát pro přihlášení do informačního systému, XSD soubor potřebný k validaci XML a XML soubor pro ukázkové spuštění aplikace.

```
recursive-include fitIS/gui/image *
recursive-include fitIS/gui/lang *
...
```

10.2 Distribuční balíčky

Pomocí distribučních nástrojů lze vytvořit více různých typů balíčku. Základní balíček je distribuce zdrojů, který obsahuje zdrojové kódy, instalační skript, README a další zahrnuté soubory [15]. Distribuce zdrojů se vytvoří pomocí příkazu `python setup.py sdist`. Parametr `sdist` vytvoří archiv ve formátu, který je výchozí pro aktuální platformu. Pro Unix je vytvořen archiv `.tar.gz` a pro Windows `.zip`. Kromě základního balíčku lze vytvářet tzv. `built distribution`, které umožňují snadným způsobem nainstalovat náš balíček. Kromě distribučních balíčků lze například pomocí nástroje `cx_Freeze` vytvořit spustitelný soubor, který bude obsahovat vše potřebné pro běh aplikace.

10.2.1 Sestavené distribuce (Built distribution)

`Built distribution` je to, co je běžně považováno buď jako binární balík nebo instalátor, ale nemusí to být nutně binární balík, protože může obsahovat jenom zdrojové kódy v Pythonu. Pomocí `Built distribution` lze uživatelům na různých operačních systémech maximálně zjednodušit instalaci. Pro uživatele linuxových distribucí, které používají RPM a Debian balíčky, lze vytvořit přímo tyto balíčky a pro uživatele Windows spustitelné instalátory [16].

- Sestavení distribuce na aktuální platformu.

  ```
  python setup.py bdist
  ```

- Vytvoření RPM balíku, který používají linuxové distribuce Red Hat, SuSE, Mandrake a další.

  ```
  python setup.py bdist_rpm
  ```

- Vytvoření Windows instalátoru.

  ```
  python setup.py bdist_wininst
  ```

10.3 Python Package Index

Python Package Index (PyPI) je archiv softwaru pro programovací jazyk Python. Pro vložení nového balíčku je potřeba nejprve zaregistrovat nového uživatele přes registrační stránku PyPI. Poté pomocí distribučního nástroje, např. `distutils` nebo `setuptools`, lze zaregistrovat a nahrát vytvořený balíček do PyPI [17]. Nahraný balíček v PyPI je poté možné nahrazovat jeho novějšími verzemi.

- Registrace vytvořeného balíku.

```
python setup.py register
```

- Nahrání sestaveného balíku nebo balíků.

```
python setup.py sdist bdist_wininst upload
```

Kapitola 11

Testování

11.1 Testování zpracování a validace informací z IS FIT

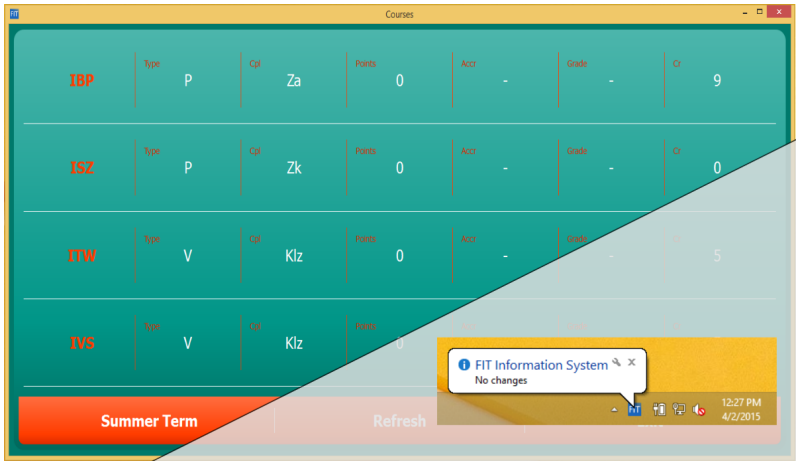
Při implementaci specifického chování tříd jednotlivých elementů XML schématu jsem pro každou implementovanou třídu napsal sadu testů, které měly za úkol otestovat korektní chováni aplikace při různých vstupech, které testují určitou třídu. Testoval jsem různé možnosti korektních, ale i nekorektních XML dokumentů. U korektních vstupů jsem testoval, zdali validace a zpracování proběhlo úspěšně, a dále jsem porovnával stromy hodnot vytvořené z XML dokumentu. Pro spuštění testů jsem implementoval automatický skript, který jsem spouštěl po implementaci chování nové třídy, abych nejen otestoval nově implementovanou třídu, ale také zda implementace neovlivnila ostatní třídy.

11.2 Testování uživatelského rozhraní

Testování uživatelského prostředí probíhalo zejména z hlediska multiplatformního použití a jeho zobrazování na různých platformách. Aplikace byla primárně vyvíjena na operačním systému Arch Linux s grafickým prostředím Gnome 3.14. Poté byla testována na dalších Linuxových distribucích a grafických prostředích a dále na platformách typu Windows a MacOS X. Při testování jsem se primárně zaměřil na to, zdali aplikace půjde nainstalovat, spustit a zdali lze vše zobrazit. Implementaci aplikace jsem podle potřeby upravil tak, aby ji bylo možno používat na všech vybraných platformách.

Platforma typu Windows

Po operačním systému, na kterém byla aplikace vyvíjena, jsem ji testoval na platformě typu Windows, na které jsem předpokládal, že spuštění aplikace bude nejvíce problematické. Pro testování jsem zvolil operační systém Windows 8. První problém se vyskytl v souvislosti s knihovnou `Crypto`, která sice pro Windows existuje, ovšem má rozdílnou implementaci. Ke knihovně jsem našel ekvivalentní knihovnu `pyCrypto`, která se importuje a používá stejným způsobem jako knihovna `Crypto`, takže ji stačilo pouze přidat do knihoven, které se automaticky stahují během instalace. Druhý problém byl s knihovnou `multiprocessing`, proto jsem upravil implementaci aplikace tak, aby místo procesů používala vlákna, která jsou přímo součástí knihovny `PyQt4`.

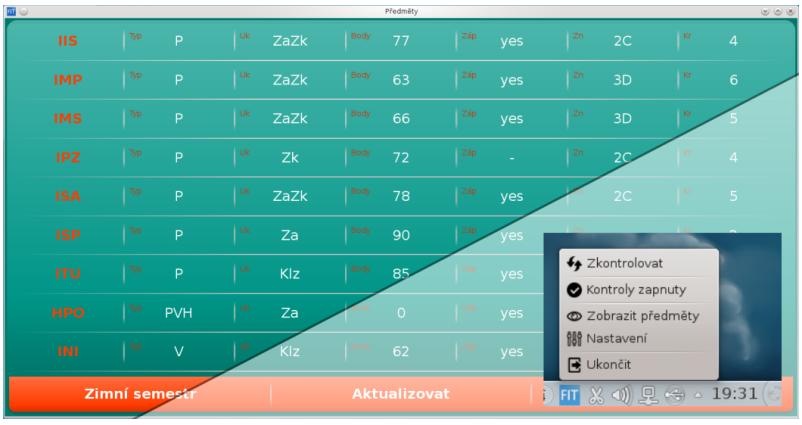


Obrázek 11.1: Aplikace na Windows 8

Platforma typu Linux

K testování jsem zvolil jedny z nejrozšířenějších distribucí Arch linux, Ubuntu 14.04, Fedora 21 a Debian v kombinaci s grafickými prostředími Gnome 3.14, XFCE, Unity, KDE. Pouze v grafickém prostředí Unity se nezobrazovaly položky v menu ikony aplikace, které měly definován vlastní styl. Vytvořil jsem proto nové menu aplikace, které jsem použil jako výchozí a využívá systémový vzhled. Pro použití menu s vlastním stylem na jiných prostředích je přidána do nastavení aplikace možnost volby mezi styly. Dále není rozlišeno mezi levým a pravým kliknutím myši na ikonu aplikace, kliknutí vždy zobrazí menu s nabídkou. Do nabídky jsem proto přidal položku, která má stejnou funkcionalitu jako levý klik na ikonu.

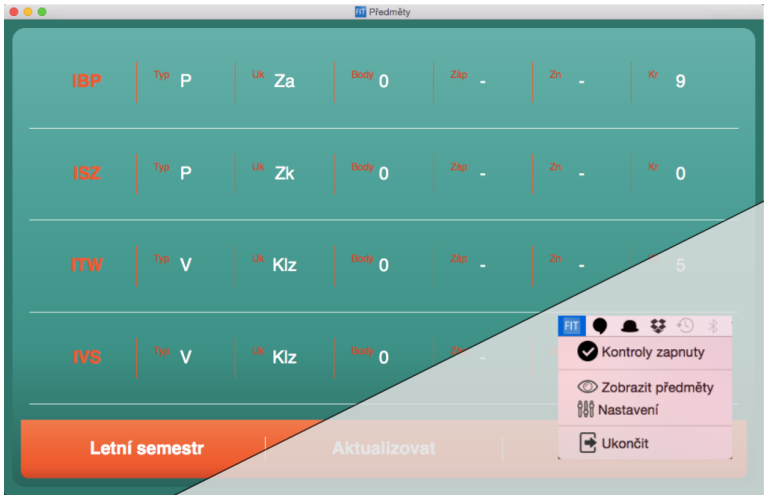
Vzhled na jednotlivých uživatelských prostředích se mírně liší. Například na grafickém prostředí KDE, které je založeno na Qt, nejsou oddělovače jednotlivých položek v celé linii stejné, ale na koncích přecházejí do ztracena.



Obrázek 11.2: Aplikace na Fedora s KDE

Platforma typu MacOS X

Aplikace jsem testoval na operačním systému OS X Yosemite. Při testování jsem narazil na stejný problém jako u grafického prostředí Unity, kdy není rozlišeno mezi pravým a levým kliknutím na ikonu aplikace. Dále se vyskytl problém se zobrazováním Qt standardních upozornění, které vyžadují instalaci speciální aplikace Growl.



Obrázek 11.3: Aplikace na OS X

Kapitola 12

Závěr

V rámci této práce jsem se seznámil s formátem XML a na základě získaných znalostí implementoval nástroj pro parsování a validaci XML dokumentů podle XML schématu. Pomocí tohoto nástroje a v rámci práce vytvořeného XML schématu zpracovávám XML dokument, který je dostupný v informačním systému Fakulty informačních technologií VUT v Brně a obsahuje informace o zapsaných předmětech. Pro zobrazení získaných informací jsem vytvořil grafické uživatelské rozhraní, jehož implementaci jsem rozložil na jednotlivé tématické části. S každou částí jsem se podrobně seznámil a pokusil se zvolit nejvhodnější způsob řešení daného problému, například zvolení modulu `gettext` pro oddělení zobrazovaného textu od programového kódu umožnilo vytvoření aplikace v českém a anglickém jazyce nebo vytvoření více barevných schémat definujících vzhled aplikace pomocí `Qt Style Sheets`. Po dokončení funkční části aplikace jsem se seznámil s možnostmi, jak lze aplikaci šířit a instalovat na různých platformách. Implementoval jsem instalační skript, který slouží jako hlavní vstup většiny nástrojů pro sestavování a distribuci balíčků v jazyce Python 3. Aplikaci jsem poté testoval a upravil tak, aby ji bylo možné používat na platformách typu Linux, Windows a MacOS X.

Tvorba této práce mi umožnila dosáhnout osobního rozvoje nejen v oblasti XML a tvorby GUI, ale také celkově ve tvorbě a distribuci aplikací a v možnostech, které přináší programovací jazyk Python 3.

Do budoucna se nabízí možnost řady zajímavých rozšíření. Do aplikace by bylo možné začlenit další části informačního systému, jako je emailový klient pro odesílaní a přijímání zpráv nebo video server se záznamy přednášek. Uživatel by mohl být automaticky, kromě dosavadního upozornění na změnu u předmětu, upozorněn také na nový email nebo na nově zveřejněné přednášky, které by bylo možné ihned stáhnout. Dále by bylo zajímavé rozšířit informace v XML dokumentu o časové údaje, ze kterých by bylo možné vytvořit osobní rozvrh obsahující přednášky, cvičení, registrace nebo termíny odevzdání projektů. Uživatel by si mohl zvolit, na které z těchto aktivit chce být upozorňován nebo které aktivity chce automaticky exportovat např. do systémového nebo Google kalendáře.

Literatura

[1] How Can XML be Used? [online]. http://www.w3schools.com/xml/xml_usedfor.asp, [cit. 2014-11-20].

[2] XML Namespaces [online]. http://www.w3schools.com/xml/xml_namespaces.asp, [cit. 2014-11-20].

[3] XML Schema Tutorial [online]. http://www.w3schools.com/schema/, [cit. 2014-11-20].

[4] gettext - Message Catalogs [online]. http://pymotw.com/2/gettext/, [cit. 2015-2-20].

[5] gettext [online]. https://www.gnu.org/software/gettext/, [cit. 2015-2-20].

[6] Properties vs. Getters and Setters [online]. http://www.python-course.eu/python3_properties.php, [cit. 2015-2-20].

[7] Style [online]. http://www.google.com/design/spec/style/color.html, [cit. 2015-2-20].

[8] Desktop notifications [online]. https://wiki.archlinux.org/index.php/Desktop_notifications, [cit. 2015-2-3].

[9] QMainWindow Class [online]. http://qt-project.org/doc/qt-4.8/qmainwindow.html, [cit. 2015-2-3].

[10] QSystemTrayIcon Class [online]. http://qt-project.org/doc/qt-4.8/qsystemtrayicon.html, [cit. 2015-2-3].

[11] QSystemTrayIcon Class Reference [online]. http://doc.qt.digia.com/4.6/qsystemtrayicon.html#showMessage, [cit. 2015-2-3].

[12] QWidget Class [online]. http://doc.qt.io/qt-4.8/qwidget.html, [cit. 2015-2-3].

[13] Signals & Slots [online]. http://qt-project.org/doc/qt-4.8/signalsandslots.html, [cit. 2015-2-3].

[14] ubuntu manuals notify-send [online]. http://manpages.ubuntu.com/manpages/hardy/man1/notify-send.1.html, [cit. 2015-2-3].

[15] Baleni pythonovskych knihoven [online]. http://diveintopython3.py.cz/packaging.html, [cit. 2015-3-16].

[16] Creating Built Distributions [online].
https://docs.python.org/3.4/distutils/builtdist.html, [cit. 2015-3-16].

[17] The Python Package Index (PyPI) [online].
https://docs.python.org/3.4/distutils/packageindex.html, [cit. 2015-3-16].

[18] Writing the Setup Script [online].
https://docs.python.org/3.4/distutils/setupscript.html, [cit. 2015-3-16].

[19] Advanced Encryption Standard [online].
http://cs.wikipedia.org/wiki/Advanced_Encryption_Standard, [cit. 2015-3-4].

[20] pickle - Python object serialization [online].
https://docs.python.org/2/library/pickle.html, [cit. 2015-3-4].

Příloha A

Obsah CD

- src/ - balíček se zdrojovými kódy aplikace, instalační skript, návod k instalaci a skript `run.py` pro spuštění aplikace bez instalace
- tex/ - zdrojové kódy textové části bakalářské práce pro LaTeX
- doc/ - programová dokumentace vytvořená pomocí doxygen
- xsd/ - XML schéma pro `get-coursesx.php`
- README - informace o bakalářské práci
- xpelik14-BP.pdf - textová část bakalářské práce

Příloha B

Ukázka z implementovaného XSD dokumentu

```
<?xml version="1.0" encoding="iso-8859-2"?>
<xsd:schema xmlns="http://www.fit.vutbr.cz/study/courses"
   targetNamespace="http://www.fit.vutbr.cz/study/courses"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="courses" type="CoursesType">
    <xsd:unique name="course_id">
      <xsd:selector xpath="course"/>
      <xsd:field xpath="@id"/>
    </xsd:unique>
  </xsd:element>
  <xsd:complexType name="CoursesType">
    <xsd:sequence>
      <xsd:element name="course" type="CourseType" minOccurs="0" maxOccurs="999"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CourseType">
    <xsd:sequence>
      <xsd:element name="title" minOccurs="0" maxOccurs="2" type="TitleType"/>
      <xsd:element name="accreditation" minOccurs="0" maxOccurs="1"
        type="CourseAccrType"/>
      <xsd:element name="examination" minOccurs="0" maxOccurs="1"
        type="CourseExamType"/>
      <xsd:element name="item" minOccurs="0" maxOccurs="999" type="ItemType"/>
    </xsd:sequence>
    <xsd:attribute name="abbrv" type="xsd:string"/>
    <xsd:attribute name="credits" type="xsd:nonNegativeInteger"/>
    <xsd:attribute name="type" type="CourseEntryType"/>
    ...
  </xsd:complexType>
  <xsd:complexType name="CourseAccrType">
    <xsd:sequence>
      <xsd:element name="teacher" type="xsd:string"/>
```

```
    </xsd:sequence>
    <xsd:attribute name="status" type="CourseAccrStatusType"/>
    <xsd:attribute name="date" type="xsd:date"/>
  </xsd:complexType>
  ...
  <xsd:simpleType name="CourseCompletionType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Za">
        <xsd:annotation>
          <xsd:documentation xml:lang="cs">Zápočet</xsd:documentation>
          <xsd:documentation xml:lang="en">Accreditation</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>
      <xsd:enumeration value="Zk">
        <xsd:annotation>
          <xsd:documentation xml:lang="cs">Zkouška</xsd:documentation>
          <xsd:documentation xml:lang="en">Examination</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>
      ...
    </xsd:restriction>
  </xsd:simpleType>
  ...
  <xsd:simpleType name="CourseGradeType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="1A">
        <xsd:annotation>
          <xsd:documentation>1.0</xsd:documentation>
        </xsd:annotation>
      </xsd:enumeration>
      ...
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="CoursePointsType">
    <xsd:restriction base="xsd:double">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="100"/>
    </xsd:restriction>
  </xsd:simpleType>
  ...
</xsd:schema>
```

Příloha C

Ukázka z rozšířeného XML dokumentu

```
<c:courses xmlns:c="http://www.fit.vutbr.cz/study/courses"
  ... http://www.fit.vutbr.cz/schemas/courses.xsd">
  <course id="9983" csid="550782" abbrv="IMS" type="P" completion="ZaZk"
    points="66" credits="5" sem="Z" upd_ts="2015-01-21T11:29:13">
    <title lang="cs">Modelování a simulace</title>
    <title lang="en">Modelling and Simulation</title>
    <accreditation status="yes" date="2015-01-15">
      <teacher>Hrubý Martin, Ing., Ph.D.</teacher>
    </accreditation>
    <examination grade="3D" date="2015-01-14">
      <teacher>Peringer Petr, Dr. Ing.</teacher>
    </examination>
    <item id="49534" num="0" type="single" start="2014-09-29" end="2014-10-31"
        registered="21" entry="login" reg_start="2014-09-29T11:33:56"
        reg_end="2014-10-30T00:00:00" upd_ts="2014-09-29T10:34:36">
      <title lang="cs">Uznání projektů z akad. roku 2013/14</title>
    </item>
    <item id="49779" num="1" type="single" start="2014-11-18" registered="436"
        max="10" entry="all" upd_ts="2014-10-21T16:31:57">
    <title lang="cs">Půlesemestrální písemka</title>
    <entry reg_type="all" reg_time="2014-10-21T16:31:42" points="7" who="hrubym"
        date="2014-12-03" upd_ts="2014-12-03T17:47:39"/>
    </item>
    <item id="49724" num="2" type="single" start="2014-10-07" end="2014-12-08"
        registered="436" max="20" entry="all" upd_ts="2014-11-30T20:16:42">
    <title lang="cs">Projekt</title>
    <entry reg_type="all" reg_time="2014-10-07T14:16:22" points="20" who="hrubym"
        date="2015-01-02" upd_ts="2015-01-02T13:56:00"/>
    </item>
    <item id="50996" num="3" type="accr" start="2015-01-15" registered="436"
        min="10" entry="all" upd_ts="2014-12-15T19:21:40">
    <title lang="cs">Zápočet</title>
```

```
    </item>
    <item id="49776" num="4" type="multi" min="30" max="70"
        upd_ts="2014-10-21T13:42:49">
      <title lang="cs">Zkouška</title>
      <variant id="50042" date="2015-01-14" entry="all" registered="436"
          upd_ts="2014-11-21T15:57:15">
        <title lang="cs">řádný (T1)</title>
        <entry reg_type="all" reg_time="2014-11-21T15:49:34" points="39"
            who="peringer" date="2015-01-21" upd_ts="2015-01-21T11:29:13"/>
      </variant>
      <variant id="50043" date="2015-01-27" entry="loginz" registered="207"
          upd_ts="2014-11-21T15:51:46">
        <title lang="cs">opravný (T2)</title>
      </variant>
      <variant id="50044" date="2015-02-04" entry="loginz" registered="121"
          upd_ts="2015-02-03T10:39:45">
        <title lang="cs">poslední opravný (T3)</title>
      </variant>
    </item>
  </course>
...
</c:courses>
```