FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Integrace KOS a Syllabus Plus
Student:	Jakub Dobrý
Vedoucí:	Ing. Ondřej Guth, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Proveďte analýzu současného procesu tvorby rozvrhu na Fakultě informačních technologií Českého vysokého učení technického v Praze. Na základě této analýzy navrhněte datové propojení KOS IS a v současnosti používané rozvrhovací aplikace Scientia Syllabus Plus s důrazem na co nejpohodlnější a nejefektivnější použití fakultním rozvrhářem. Navržené řešení implementujte jako prototyp a vhodnými prostředky otestujte.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 9. listopadu 2020 v Praze.

Bakalářská práce

INTEGRACE KOS A SYLLABUS PLUS

Jakub Dobrý

Fakulta informačních technologií ČVUT v Praze
Katedra softwarového inženýrství
Vedoucí: Ing. Ondřej Guth, Ph.D.
13. května 2021

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především vedoucímu své práce Ing. Ondřeji Guthovi, Ph.D., za pomoc, čas a trpělivost s odpovídáním na mé otázky. Chtěl bych také poděkovat Ing. Danielu Dombkovi, Ph.D., za čas věnovaný konzultacím procesu tvorby rozvrhu a následně i uživatelskému testování. Dále bych chtěl poděkovat svým kamarádům, kteří mě po celou dobu velmi podporovali. V neposlední řadě bych rád poděkoval své rodině za podporu a trpělivost při psaní této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 13. května 2021 .....

Abstrakt

Tato bakalářská práce řeší návrh, implementaci a testování aplikace pro datové spojení Studijního informačního systému KOS a softwaru pro tvorbu rozvrhu Scientia Syllabus Plus. Svého účelu dosahuje pomocí grafického rozhraní. To umí zobrazit data z obou systémů a rozdíly mezi nimi. Dále umožňuje částečné převody na úrovni jednotlivých položek. Výsledek práce poslouží fakultnímu rozvrháři při tvorbě rozvrhu na Fakultě informačních technologií.

Klíčová slova grafická aplikace, Scientia Syllabus Plus, Studijní informační systém (KOS), synchronizace dat, vizualizace rozdílů dat

Abstract

This bachelor thesis deals with the design, implementation, and testing of an application for data connection between the Study Information System KOS and the Scientia Syllabus Plus timetable software. It achieves its purpose through a graphical interface. It can display data from both systems and the differences between them. It also allows partial conversions at the level of individual items. The result of this work will be used by the faculty scheduler in creating the timetable at the Faculty of Information Technology.

Keywords graphical application, Scientia Syllabus Plus, Study Information System (KOS), data synchronization, data difference visualization

Seznam zkratek

ČVUT	České vysoké učení technické v Praze
FIT	Fakulta informačních technologií
KOS	Studijní informační systém
Syllabus Plus	Scientia Syllabus Plus
JVM	Java Virtual Machine
API	Application Programming Interface
XML	Extensible Markup Language
COM	Component Object Model
ODBC	Open Database Connectivity
SQL	Structured Query Language
ASCII	American Standard Code for Information Interchange
JSON	JavaScript Object Notation
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JAR	Java ARchive
MB	Megabyte
CSV	Comma-separated values
REST	Representational state transfer
TLB	Type Library
UEF	Unified Emulator Format
JRE	Java Runtime Environment

Kapitola 1
Úvod

Na Fakultě informačních technologií (FIT) je potřeba každý semestr vytvořit nový rozvrh. Na to je nutné zjistit například, jaké předměty budou vyučované, kolik paralelek je potřeba vypsat a kdo bude tyto předměty učit. Většina informací je zadávána do Studijního informačního systému KOS. Pro následné vytvoření rozvrhu je používána rozvrhovací aplikace Syllabus Plus. Je proto potřeba umožnit převádět data mezi těmito dvěma systémy.

Aby fakultní rozvrhář nemusel vše dělat ručně, existuje aktuálně řešení, které umí data jednorázově převést. Bohužel není dostatečně flexibilní a neumožňuje pokrýt velkou část požadavků. Kromě toho současné technické řešení není uživatelsky přívětivé, tedy rozvrhář neví, jaké konkrétní změny nastanou a jestli se mu neztratí již provedené úpravy. I proto řešení vyžaduje asistenci pro samotný běh.

Toto téma jsem si zvolil, protože si myslím, že jsem schopný nabídnout řešení, které odstraní hlavní problémy a celkově usnadní tvorbu rozvrhu. Nový program tak ušetří čas a starosti fakultnímu rozvrháři.

Práce je rozdělena na čtyři hlavní části, začíná analýzou, pokračuje návrhem a realizací a končí testováním.

V první části se zabývám analýzou procesu tvorby rozvrhu a poté sběrem problémů a případných požadavků od fakultního rozvrháře. Na základě získaných informací sestavím výsledné požadavky na aplikaci a případy užití.

V kapitole návrhu představuji nové řešení pomocí grafické aplikace, která dokáže data, ať už kompletní stav nebo jejich část, opakovaně převádět. Pro pohodlí uživatele, jednodušší převody a celkovou průhlednost fungování umí i přehledně vizualizovat data z jednotlivých systémů. Uživatel si tak dokáže lépe představit, která data se kde nachází a jaký převod potřebuje provést. Dále v této části popisuji vliv nového řešení na proces tvorby rozvrhu, strukturu dat systému KOS a softwaru Syllabus Plus a rozebírám možnosti práce s jednotlivými systémy.

V kapitole realizace se zabývám výběrem vhodné technologie, implementací navrženého řešení a popisu architektury. Zabývám se zde i vzniklými problémy během implementace a jejich řešením. Kapitola končí rozborem použitých licencí jednotlivých knihoven.

Testování řeším pomocí manuálních scénářů, které pokrývají požadavky aplikace a scénáře použití. Pro ověření pohodlnosti a efektivnosti práce s výslednou aplikací provádím uživatelské testování, které se zaměřuje na právě tento požadavek.

Kapitola 2
Cíle práce

Hlavním cílem je vytvořit aplikaci s grafickým uživatelským rozhraním, která zjednoduší a zefektivní převody dat mezi softwarem Syllabus Plus a systémem KOS pro fakultního rozvrháře. Toho se snaží dosáhnout pomocí vizualizace dat, zobrazení rozdílů, možnosti částečných převodů a samotným grafickým uživatelským rozhraním.

Cílem analýzy je zmapovat proces tvorby rozvrhu na FITu, jeho problémy a požadavky. Na základě zjištěných informací navrhnout řešení, které analyzované požadavky splní, a umožní fakultnímu rozvrháři převody dat mezi systémy v obou směrech.

V realizaci je cílem vytvoření grafické aplikace. Ta bude umět vizualizovat data jednotlivých systémů a provádět kompletní nebo částečné převody.

V kapitole testování je cílem ověřit funkčnost výsledné aplikace a provést uživatelské testování zaměřené na intuitivnost a přehlednost uživatelského rozhraní.

Výsledná práce bude přínosná jak pro aktuálního fakultního rozvrháře, tak i pro budoucí tím, že zjednoduší a zpřehlední převody potřebných dat mezi systémy a odstraní aktuální problémy.

Kapitola 3
Analýza

3.1 Struktura dat a pojmy

Na začátek práce představím data a položky, se kterými budu pracovat. Využívat budu primárně názvosloví, které je zadefinováno ve studijním informačním systému používaném na ČVUT, který se jmenuje KOS. V tomto systému je uchovávána většina dat o běhu fakulty, z kterých mě zajímají data potřebná pro tvorbu rozvrhu.

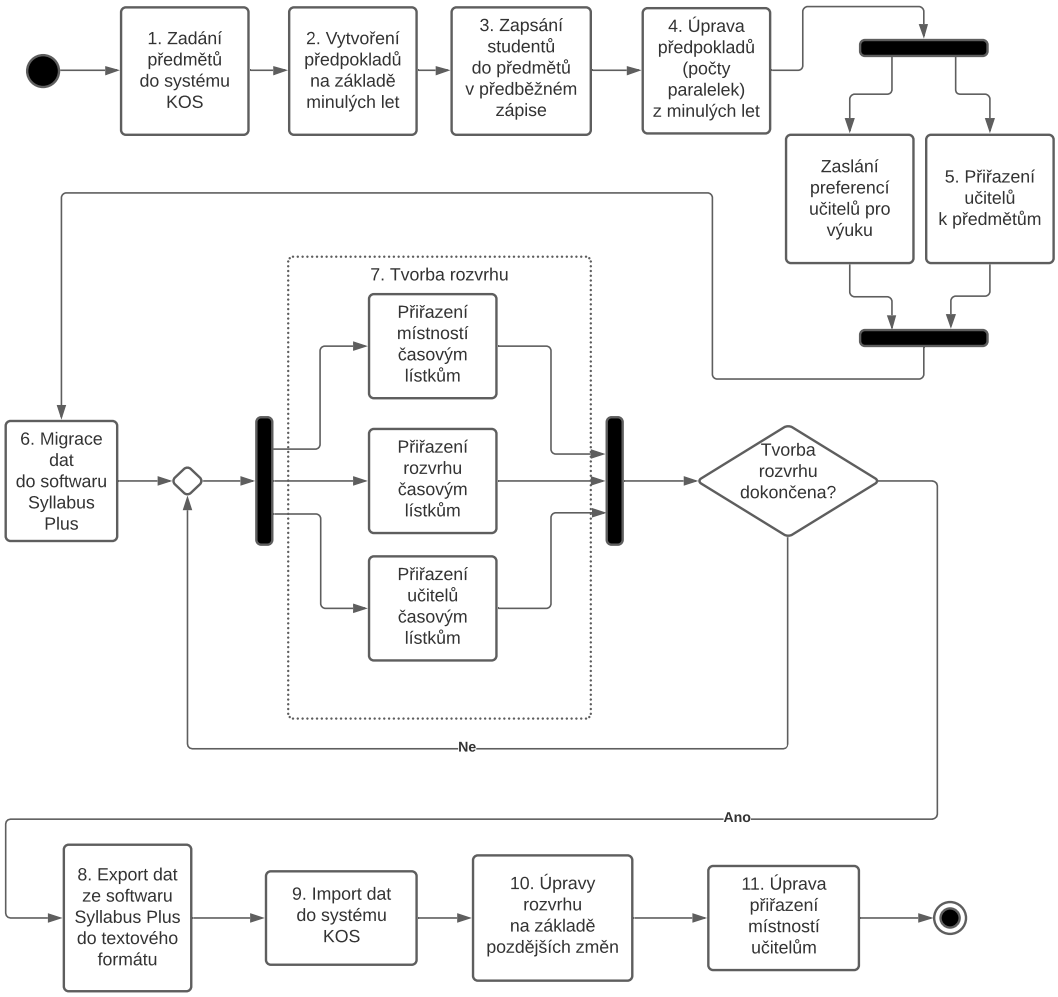
Jako nejvyšší organizační jednotky slouží na fakultě katedry. Každá katedra má pod sebou vypsané předměty. Každý předmět má své paralelky. Paralelka představuje vypsání přednášky, proseminář nebo cvičení daného předmětu. Většinou používá číselné označení a využívá se při tvorbě rozvrhu studenty. Každá paralelka je ale zároveň seskupením jednoho nebo více časových lístků. Časový lístek označuje konkrétní rozvrhové vypsání. Většina předmětů obsahuje pouze jeden časový lístek pro paralelku, nicméně jsou i výjimky. Jako příklad, kdy časový lístek neodpovídá samotné paralelce, může být předmět matematická analýza. V tomto předmětu je každý týden jedna přednáška a jednou za čtrnáct dní je k ní navíc ještě druhá. Při zápisu se však pro studenta chovají jako jeden celek a tvoří tak dohromady pouze jednu paralelku.

V rámci zjednodušení názvů budu používat pro převod dat ze systému KOS do softwaru Syllabus Plus slovo „import“. Pro převod dat zpátky ze softwaru Syllabus Plus do systému KOS pak „export“. Dále, pokud nebude uvedeno jinak, pod pojmem „rozvrhář“ myslím fakultního rozvrháře.

3.2 Proces tvorby rozvrhu

Tvorba rozvrhu je proces, který začíná kolem poloviny předchozího semestru a končí začátkem nového semestru, někdy ještě pár dní poté. Skládá se ze základních akcí:

1. Nastavení vyučovaných předmětů, zadání předmětů do systému KOS
2. Vytvoření předpokladů na základě minulých let
3. Zapsání studentů do předmětů v předběžném zápise
4. Úprava předpokladů z minulých let
5. Přiřazení učitelů k předmětům
6. Migrace dat do softwaru Syllabus Plus
7. Tvorba rozvrhu



Obrázek 3.1 Schéma procesu tvorby rozvrhu, čísla označují pořadí hlavních akcí

8. Export dat ze softwaru Syllabus Plus
9. Import dat do systému KOS
10. Zapracování dodatečných změn
11. Optimalizace přesunů vyučujících mezi místnostmi

Proces je znázorněn na obrázku 3.1.

Začíná zadáním vyučovaných předmětů pro daný semestr do systému KOS katederními rozvrháři. Většinou se předměty nechávají z minulých let, maximálně se upravuje pár jednotlivých.

Poté fakultní rozvrhář vytvoří předpoklady očekávaných kapacit jednotlivých předmětů na základě dat a zkušeností z minulých let.

Přibližně v půlce předchozího semestru se otevře studentům možnost si předběžně zapsat předměty. Tento zápis trvá kolem čtrnácti dní. Během této doby si mohou studenti prostřednictvím systému KOS zapsat předměty.

Po skončení předběžných zápisů upřesní katederní rozvrháři kapacity podle projeveného zájmu. Fakultní rozvrhář pak na základě dat vygeneruje časové lístky pro jednotlivé paralelky.

Dále katederní rozvrháři přiřadí vyučující k jednotlivým předmětům. Během toho učitelé zašlou své preference s časovými možnostmi. Občas své preference zapíší do systému KOS, většinou jsou však zasílány pomocí emailu přímo fakultnímu rozvrháři [1].

V tuto chvíli máme veškeré důležité informace v systému KOS. Následuje jednorázový převod dat ze systému KOS do softwaru Syllabus Plus. Převedou se informace o katedrách, předmětech, časových lístcích. Dále se přenesou učitelé, kteří jsou uspořádáni do skupin podle předmětů a jejich rolí (přednáška nebo cvičení) a nakonec místnosti, které jsou rozdělené do manuálně vytvořených skupin.

Začíná tvorba samotného rozvrhu, kterou provádí fakultní rozvrhář a během které je potřeba realizace spousty operací. Mezi ně patří přiřazení učitelů a místností časovému lístku nebo naplánování časového lístku v rámci týdne. Kromě informací získaných ze systému KOS jsou potřeba i další, jako skupiny studentů nastupujících do prvních ročníků nebo studijní plány. Tyto informace jsou buď fakultním rozvrhářem zadávány do softwaru Syllabus Plus manuálně, nebo jsou pouze ručně kontrolovány.

Po dokončení tvorby rozvrhu se data opět jednorázově převedou ze softwaru Syllabus Plus do systému KOS. Nepřevedou se ale všechna data, převedou se pouze informace o naplánování časového lístku, dvou přiřazených učitelích a místnosti.

Než je rozvrh plně připraven pro další semestr, probíhá ještě spousta úprav. Těmi jsou třeba nově vniklé časové lístky předmětů, případně jiné požadavky vyučujících. Změny tak fakultní rozvrhář zanáší do softwaru Syllabus Plus, ve kterém je naplánuje, a poté provádí stejné akce ještě v systému KOS, protože jednorázový převod dat již proběhl a během tvorby se znovu neopakuje.

Nicméně ne všechny změny je dobré dělat v softwaru Syllabus Plus. Mezi takové patří třeba optimalizace „přeběhů“ mezi předměty. Aby vyučující nemusel zbytečně cestovat po nejvzdálenějších učebnách fakulty, může například jen přejít do vedlejší nebo dokonce zůstat v aktuální. Tato optimalizace probíhá ale až na konci tvorby, většinou během posledního týdne těsně před začátkem nového semestru. [1]

Po všech změnách je rozvrh dokončen a připraven pro následující semestr.

3.3 Aktuální technické řešení

V rámci celého procesu se v této práci zabývám částí převodů dat mezi softwarem Syllabus Plus a systémem KOS.

Aktuální řešení funguje pomocí konzolové aplikace (spouštěné z příkazové řádky) napsané v jazyce Java a pomocné relační databáze PostgreSQL (dále „pomocná databáze“). Celé řešení může běžet lokálně u rozvrháře, nicméně to má svá úskalí. Rozvrhář by potřeboval nainstalovat databázový server PostgreSQL a běhové prostředí jazyka Java (JRE) ve verzi 11. Kromě instalace serveru nastává i problém s verzí JRE. Rozvrhář používá pro přístup k systému KOS aplikaci, která vyžaduje JRE ve verzi 8. Tím nastává komplikace správného nastavení souběhu různých verzí běhových prostředí.

Kvůli spouštění aplikace pomocí příkazové řádky, problémům s instalací databázového serveru a verzím JRE, je pomocná databáze hostována na serveru a řešení obsluhováno správcem softwaru.

V rámci přípravy je potřeba na straně rozvrháře nainstalování ODBC ovladače pro databázi PostgreSQL a nastavení pojmenovaného připojení k PostgreSQL serveru.

Hlavní použití vypadá tak, že nejdříve správce softwaru stáhne potřebná data ze systému KOS, převede je do potřebné struktury pro software Syllabus Plus a poté je uloží do pomocné databáze. V softwaru Syllabus Plus je následně potřeba importovat několik předpřipravených skriptů, které slouží pro stažení dat z pomocné databáze.

Jedná se o ODBC skripty (pojem z dokumentace softwaru Syllabus Plus), které využívají lokální databázový konektor a jazyk SQL pro dosažení patřičných změn v softwaru Syllabus Plus.

Nejedná se o čisté SQL, skripty mají i svá specifika, která jsou podrobněji popsána v dokumentaci softwaru. Pro import dat rozvrhář zavolá skripty manuálně přes speciální výběrové okno, které je přístupné z hlavní nabídky, a data se do softwaru stáhnou a zpřístupní pro práci.

Po dokončení úprav rozvrhu pak ve fázi exportu rozvrhář použije další předpřipravené skripty, které převedou data do pomocné databáze. Správce softwaru pak následně spustí konzolovou aplikaci, která načte data z databáze a převede je do formátu csv. Soubor pak předá rozvrháři, který ho načte do systému KOS.

3.3.1 Problémy

Při tvorbě rozvrhu, po prvotním převodu dat, přichází od vyučujících nebo katederních rozvrhářů další požadavky na změny. Těmi mohou být noví učitelé, kteří budou tento semestr učit, počet vypsaných paralelek nebo časových lístků pro daný předmět, případně, ale pouze zřídka, vytvoření celého nového předmětu. Všechny změny je potřeba zakomponovat do tvorby rozvrhu. Aktuální řešení sice umožňuje částečné převody provádět, ale pouze na úrovni například všech časových lístků, všech učitelů. Převod všech časových lístků by však mohl přepsat důležité informace a tím by se mohly ztratit rozpracované změny, případně hotové části rozvrhu. Rozvrhář tak v tomto případě vytváří položky manuálně v softwaru Syllabus Plus a poté je ručně přepisuje zase zpátky do systému KOS. [1]

Kromě problému se změnami je však nevýhodou i rozhraní, kterým je konzole. Ta není pro běžného uživatele přívětivá. Je proto potřeba asistence další osoby, která aplikaci spouští, nastavuje a dává do provozu. Software se tak stává poměrně značnou překážkou a slouží jen pro ty nejnutnější operace, konkrétně jednorázový import a export.

Dalším z častých problémů jsou změny mezi tím, jestli je časový lístek vyučován sudý nebo lichý týden. Soubor pro úpravu dat v systému KOS neumožňuje měnit paritu časového lístku, proto všechny takové změny musí být prováděné v systému KOS nezávisle na softwaru Syllabus Plus. Jsou proto zanechávány v softwaru Syllabus Plus i časové lístky na týdny, kdy se samotná hodina nekoná, aby v případě změny nemusely probíhat nové převody a manuální synchronizace. [1]

Kromě parity u lístků se také stává, že se u předpřípravy splete sudý a lichý týden. Taková situace má za následek špatné nastavení sudých a lichých týdnů v softwaru Syllabus Plus, což může vést ve výsledku až ke špatně navrženému rozvrhu, ve kterém jsou prohozené sudé a liché týdny. Velmi to pak zkomplikuje celou přípravu rozvrhu. [1]

Občas se stane, že si rozvrhář omylem v softwaru Syllabus Plus smaže časový lístek. Když by věděl, který si tak smazal, ušetřilo by mu to práci, protože aktuálně ho musí kromě nalezení i správně znovu vytvořit a nastavit. [1]

Transparentnost, která v aktuálním řešení chybí, je jedna z nejdůležitějších věcí. Bez ní není možné provádět částečné převody, jelikož by mohlo dojít k rozbití již hotových dat nebo nechtěnému přepsání informací od katederních rozvrhářů. [1]

Další problémy aktuálního řešení:

- Správná inicializace obrazu softwaru Syllabus Plus
  Pro inicializaci obrazu je potřeba několik informací, které nemusí být jednoduché dohledat. Samotná inicializace je ale klíčová pro správné fungování plánování a převod dat.

- Učitelé, kteří byli přiřazeni časovému lístku v systému KOS, ale nejsou nastaveni v softwaru Syllabus Plus, budou při exportu přemazáni [1]
  Může se stát, že katederní rozvrhář přiřadí časovému lístku vyučujícího v systému KOS. Fakultní rozvrhář ale tuto změnu nemusí zaregistrovat a zanést si ji taky do softwaru Syllabus Plus. Když pak následně fakultní rozvrhář provede export dat, bude v něm nastavený učitel chybět a v systému KOS se odstraní.

- Manuálně vytvořené časové lístky v softwaru Syllabus Plus nejdou exportovat
  Pokud fakultní rozvrhář vytvoří časový lístek v softwaru Syllabus Plus a poté ho chce exportovat do systému KOS, tak se mu to nepodaří. Tato funkcionalita není vůbec podporována, protože nově vytvořený časový lístek nemá přiřazený žádný identifikátor, pomocí kterého by bylo možné zjistit, ke kterému časovému lístku v systému KOS patří.

- Převod dat ze systému KOS převádí pouze základní údaje o časových lístcích
  Při prvotním převodu dat se zanesou pouze informace o časovém lístku. Pokud by byla v systému ještě nějaká další data, jako třeba časové naplánování rozvrhu, tak se tento údaj nepřevede.

- Ne vždy je potřeba importovat všechny místnosti
  V rámci systému KOS se nachází velké množství místností. Většinu ale není potřeba importovat, protože se v nich na FITu nevyučuje.

- Omylem smazaný časový lístek v softwaru Syllabus plus
  Pokud se rozvrháři stane, že si omylem odstraní časový lístek v softwaru Syllabus Plus, tak je pro něj velmi obtížné poté takový lístek dohledat. Kromě toho navíc ztratí možnost časový lístek po dokončení rozvrhu následně exportovat.

3.4 Požadavky

Hlavními požadavky fakultního rozvrháře je umožnění kromě jednorázového celkového importu a exportu i částečných převodů na úrovni jednotlivých časových lístků nebo jejich částí. Dále je důležitá přehlednost prováděných operací. Rozvrháři musí být jasné, která data se změní, aby nedocházelo k nechtěnému přepsání informací. S tím souvisí i to, že aplikace musí být obsluhovatelná bez asistence, tedy rozvrhář musí být schopen všechny převody provádět sám.

Na základě předem popsaných problémů a hlavních požadavků jsem sestavil funkční a nefunkční požadavky pokrývající požadovanou funkcionalitu aplikace.

3.4.1 Funkční

3.4.1.1 F1 – Pomocník pro inicializaci obrazu softwaru Syllabus Plus

Pro inicializaci obrazu softwaru Syllabus Plus a další práci s daty je potřeba zadat informace o daném semestru. Správné nastavení je klíčové pro korektní plánování časových lístků.

3.4.1.2 F2 – Převod všech časových lístků do softwaru Syllabus Plus

Pro prvotní načtení dat je potřeba převod všech časových lístků. S těmi souvisí i převod předmětů a kateder, pod které spadají.

3.4.1.3 F3 – Převod všech učitelů do softwaru Syllabus Plus

Pro prvotní načtení dat je potřeba převod všech učitelů. Učitele je nutné pro lepší práci organizovat do skupin podle předmětů a jejich rolí. Musí se proto převést i tyto skupiny.

3.4.1.4 F4 – Organizace místností do předdefinovaných skupin

Místnosti je pro lepší přehlednost a práci potřeba roztřídit do předem definovaných skupin. Skupiny jsou definované textovým názvem a množinou kódů místností.

3.4.1.5 F5 – Převod všech místností do softwaru Syllabus Plus

Pro prvotní načtení dat je potřeba převod všech místností. S tím souvisí i jejich předdefinované skupiny, které je nutné taky převést.

3.4.1.6 F6 – Částečný převod časových lístků

Kromě převodu všech časových lístků musí aplikace umožňovat i výběr a převod jakékoli podmnožiny.

3.4.1.7 F7 – Převod všech informací a potřebných nastavení časového lístku

V rámci převodu časového lístku je potřeba převést informace o rozvrhu, přiřazených učitelích a přiřazených místnostech. Pro správné nastavení a fungování je nutné převést ještě paritu týdne a typ (přednáška, proseminář, cvičení) časového lístku.

3.4.1.8 F8 – Částečný převod informací o časovém lístku

Uživatel by měl mít k dispozici výběr, zda chce vždy převést všechny informace o časovém lístku, nebo pouze jejich část. Částí je myšleno rozvrh, učitelé nebo místnosti.

3.4.1.9 F9 – Částečný převod učitelů

Kromě převodu všech učitelů musí aplikace umožňovat i výběr a převod jakékoli podmnožiny.

3.4.1.10 F10 – Částečný převod místností

Kromě převodu všech místností musí aplikace umožňovat i výběr a převod jakékoli podmnožiny.

3.4.1.11 F11 – Spárování záznamů mezi systémem KOS a softwarem Syllabus Plus

V případě, kdy se do systému Syllabus Plus dostane časový lístek, učitel nebo místnost jiným způsobem, než pomocí aplikace pro převod, je potřeba pro správné fungování položky spárovat. To znamená, že se přiřadí daným položkám identifikátory, které využívá systém KOS. Ty jsou následně používány při exportu dat.

3.4.1.12 F12 – Zobrazení informací před převodem do softwaru Syllabus Plus

Pro lepší přehlednost je potřeba upozornit uživatele na změny, které se chystá provést. Při převodu to je seznam položek, případně jejich vlastností, které budou změněny.

3.4.1.13 F13 – Upozornění na nekonzistenci dat při exportu do systému KOS

Při exportu časových lístků ze softwaru Syllabus Plus je potřeba upozornit uživatele na změny, které budou v systému KOS provedeny. Tento požadavek předchází nechtěnému přepsání informací.

3.4.1.14 F14 – Zobrazení rozdílů mezi položkami

Pro zjištění nekonzistence dat je požadováno zobrazit rozdíly mezi položkami v jednotlivých systémech. Uživatel by měl na první pohled zjistit, která data nejsou shodná a je potřeba je importovat nebo exportovat.

3.4.1.15 F15 – Zobrazení informací o časovém lístku

Každý časový lístek obsahuje informace o rozvrhu, učitelích a místnostech, které je potřeba vizualizovat v porovnatelné formě.

3.4.1.16 F16 – Zobrazení 1:1 dat ze softwaru Syllabus Plus a systému KOS

Zobrazení dat by mělo být porovnatelné jedna ku jedné. Zároveň by mělo zobrazení představovat tabulku nebo seznam, se kterými je fakultní rozvrhář zvyklý pracovat. [1]

3.4.1.17 F17 – Export všech časových lístků do systému KOS

Rozvrhář by měl mít možnost exportu všech časových lístků ze softwaru Syllabus Plus do souboru, který slouží pro změnu dat v systému KOS.

3.4.1.18 F18 – Export vybraných časových lístků do systému KOS

Rozvrhář by měl mít možnost exportu jakékoli podmnožiny časových lístků ze softwaru Syllabus Plus do souboru, který slouží pro změnu dat v systému KOS.

3.4.1.19 F19 – Výběr semestru pro zobrazení dat ze systému KOS

Rozvrhář by měl mít možnost výběru semestru, který se má použít pro načtení dat ze systému KOS.

3.4.2 Nefunkční

3.4.2.1 N1 – Aplikace s grafickým uživatelským rozhraním

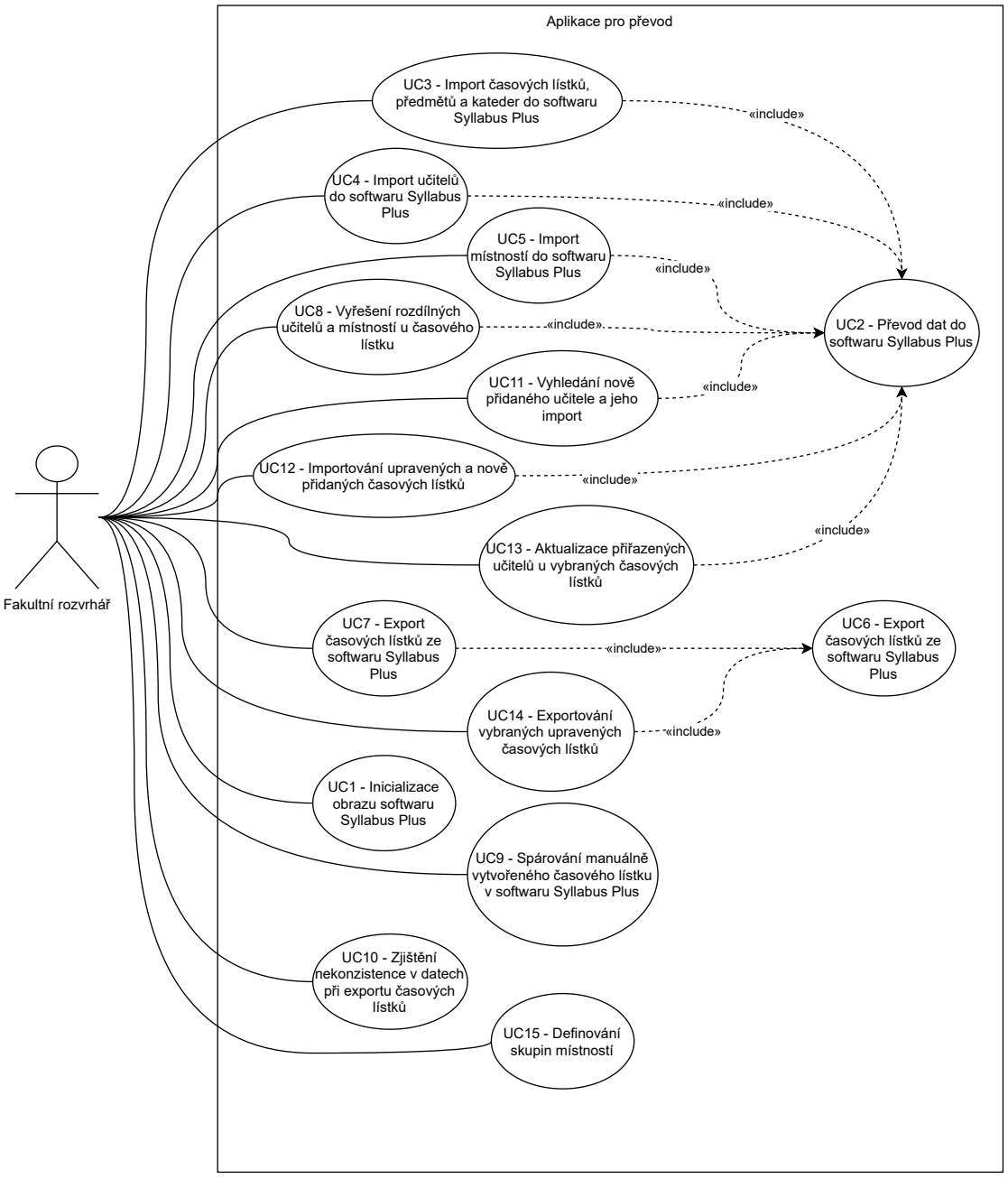
Aplikace by měla mít grafické uživatelské rozhraní pro práci s daty. Uživatel by tak měl být schopen obsluhovat aplikaci sám bez pomoci další osoby.

3.4.2.2 N2 – Filtrace zobrazených dat

Zobrazená data o časových lístcích, místnostech a učitelích by mělo být možné filtrovat.

3.5 Případy užití

Na základě požadavků a procesu tvorby rozvrhu jsem sestavil případy užití. Obrázek 3.2 zobrazuje jejich návaznosti a aktéry. V mém případě je ve všech případech užití jediný aktér, kterým je fakultní rozvrhář.



Obrázek 3.2 Diagram případů užití

3.5.1 UC1 – Inicializace obrazu softwaru Syllabus Plus

Rozvrhář spustí software Syllabus Plus a začne s inicializací nového obrazu. Během inicializace se mu zobrazí dialog pro nastavení začátku semestru, vyučované dny, začátek a konec hodin, počet hodin za den a počet týdnů na semestr. Rozvrhář otevře aplikaci pro převod a nastaví semestr, který se využije pro načítání dat ze systému KOS. Zobrazí si jakým způsobem má položky nastavit, aby byla data správně reprezentována (například při plánování časových lístků). Formulář vyplní podle získaných dat z aplikace a inicializuje obraz. V případě, že se rozvrhář splete, musí aktuální obraz smazat a začít vytvářet nový od začátku. Jinak je inicializace úspěšně dokončena.

3.5.2 UC2 – Převod dat do softwaru Syllabus Plus

Rozvrhář zvolí převod vybraných dat do softwaru Syllabus Plus. Zobrazí se mu dialog s přehledem všech položek, které chce převést. Rozvrhář klikne na tlačítko pro spuštění převodu. Během převodu může dojít k chybě. Tou může být například selhání spojení se softwarem Syllabus Plus nebo jiná výjimka. V takovém případě převod selže a rozvrhář se bude muset podívat, která data byla převedena a která stále chybí. Má pak tři možnosti jak pokračovat. První je opětovné označení všech položek a spuštění převodu. Druhá je nejdříve odstranění všech převedených položek a spuštění převodu znovu. Třetí je zjištění rozdílů mezi systémy a převedení chybějících položek. V případě úspěšného dokončení je převod dat dokončen a data ze softwaru Syllabus Plus se automaticky aktualizují.

3.5.3 UC3 – Import časových lístků, předmětů a kateder do softwaru Syllabus Plus

Rozvrhář začne výběrem všech časových lístků, předmětů a kateder. Poté pokračuje stejně jako v případě 3.5.2. Převod dat je dokončen.

3.5.4 UC4 – Import učitelů do softwaru Syllabus Plus

Rozvrhář začne výběrem všech učitelů a jejich skupin. Poté pokračuje stejně jako v případě 3.5.2. Převod dat je dokončen.

3.5.5 UC5 – Import místností do softwaru Syllabus Plus

Rozvrhář začne výběrem všech místností a jejich předdefinovaných skupin. Poté pokračuje stejně jako v případě 3.5.2. Převod dat je dokončen.

3.5.6 UC6 – Export časových lístků ze softwaru Syllabus Plus

Rozvrhář zvolí export a zobrazí se mu dialogové okno s přehledem všech položek, které se chystá exportovat, a údajů, které se změní. Rozvrhář zkontroluje, zda seznam souhlasí a zda chce zobrazené informace opravdu přepsat. Pokud informace nesouhlasí, tak převod přeruší a data opraví. Jinak zvolí export a program se ho zeptá, kam chce soubor uložit. Pokud klikne na zrušit, celý formulář se zavře a rozvrhář musí iniciovat export znovu (položky zůstanou vybrané). Jinak rozvrhář vybere lokaci souboru. Pokud soubor již existuje, program se zeptá, zda ho chce přepsat. Pokud zvolí že ne, převod dat se přeruší a rozvrhář musí iniciovat export znovu (položky zůstanou vybrané). Pokud zvolí ano či soubor ještě neexistuje, začne export dat do souboru.

V případě, že dojde k selhání při exportu, zobrazí se rozvrháři dialogové okno s položkami, které se nepodařilo exportovat. Pokud vše doběhne v pořádku, zobrazí se pouze notifikace o úspěchu a exportovaný soubor je připravený pro další práci. Tento soubor rozvrhář následně poskytne systému KOS. Export dat je dokončen.

3.5.7 UC7 – Export časových lístků ze softwaru Syllabus Plus

Rozvrhář vybere všechny časové lístky v softwaru Syllabus Plus. Poté pokračuje stejně jako v případě 3.5.6. Export časových lístků je dokončen.

3.5.8 UC8 – Vyřešení rozdílných učitelů a místností u časového lístku

V systému KOS a v softwaru Syllabus Plus se nachází časový lístek, který nemá shodné informace. Tento lístek by ale chtěl rozvrhář v softwaru Syllabus Plus aktualizovat. Zobrazí si data z obou systémů. Zjistí, o který časový lístek se jedná, a tento časový lístek vybere. Poté pokračuje stejně jako v případě 3.5.2. Aktualizace je dokončena.

3.5.9 UC9 – Spárování manuálně vytvořeného časového lístku v softwaru Syllabus Plus

Rozvrhář vytvořil v softwaru Syllabus Plus manuálně časový lístek. Nyní by chtěl tento časový lístek exportovat a spárovat s časovým lístkem v systému KOS, který tam také manuálně vytvořil. Spustí aplikaci pro převod, načte data z obou systémů a dohledá konkrétní časové lístky v obou systémech. Položky označí a klikne na tlačítko spárovat. V případě selhání se zobrazí notifikace o chybě. Jinak se položky úspěšně propojí a aktualizují se automaticky data ze softwaru Syllabus Plus. Po úspěšném propojení již bude správně fungovat export i porovnání změn. Spárování je dokončeno.

3.5.10 UC10 – Zjištění nekonzistence v datech při exportu časových lístků

Rozvrhář vybere časové lístky ze softwaru Syllabus Plus a zvolí export. Zobrazí se mu dialogové okno s přehledem dat a porovnání oproti datům ze systému KOS. Exportované časové lístky ale mají v systému KOS již nastavené učitele od katederních rozvrhářů, které se nenachází v softwaru Syllabus Plus. Systém zobrazí tento konflikt rozvrháři, který se pak může rozhodnout, zda dané změny chce opravdu přepsat, nebo může export zrušit a nejdříve data opravit. (Oprava dat může zahrnovat spárování manuálně vytvořeného časového lístku podle případu 3.5.9 nebo jakékoli jiné změny.)

3.5.11 UC11 – Vyhledání nově přidaného učitele a jeho import

Do systému KOS byl katederním rozvrhářem přidán nový učitel. Rozvrhář spustí aplikaci pro převod a načte data. Přepne na zobrazení učitelů a pomocí filtru vyhledá konkrétního učitele, kterého vybere. Poté pokračuje stejně jako v případě 3.5.2. Import je dokončen.

3.5.12 UC12 – Importování upravených a nově přidaných časových lístků

V systému KOS byly vytvořeny nové časové lístky a upraven rozvrh u několika stávajících. Rozvrhář spustí aplikaci pro převod a načte data. Ze zobrazení zjistí, které časové lístky jsou rozdílné, a následně je vybere pro převod. Poté pokračuje stejně jako v případě 3.5.2. Data jsou převedena do softwaru Syllabus Plus.

3.5.13 UC13 – Aktualizace přiřazených učitelů u vybraných časových lístků

Do systému KOS byli u několika časových lístků přiřazeni učitelé katederními rozvrháři. Fakultní rozvrhář spustí aplikaci pro převod, načte si data a na základě zobrazení rozdílů vybere časové lístky. Před samotným převodem zvolí pouze převod učitelů, aby se mu nerozházely ostatní informace například o konkrétním rozvrhu nebo místnostech. Poté pokračuje stejně jako v případě 3.5.2. Časové lístky v softwaru Syllabus Plus jsou aktualizované s novými učiteli.

3.5.14 UC14 – Exportování vybraných upravených časových lístků

Rozvrhář si načte data ze softwaru Syllabus Plus a vybere konkrétní časové lístky. Poté pokračuje stejně jako v případě 3.5.6. Export vybraných časových lístků je dokončen.

3.5.15 UC15 – Definování skupin místností

Rozvrhář nastaví skupiny místností. Poté aplikaci restartuje a po načtení se místnosti roztřídí do nově definovaných skupin.

3.6 Mapování požadavků na případy užití

Sestavil jsem tabulku mapování požadavků a případu užití (viz tabulka 3.1), abych si ověřil, že všechny požadavky jsou pokryty alespoň jedním případem užití a zároveň že žádný případ užití není zbytečný. Tabulka ukazuje, že jsou všechny požadavky pokryty.

	Případy užití														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F1	x														
F2			x												
F3				x											
F4															x
F5					x										
F6		x						x				x	x		
F7		x										x			
F8													x		
F9		x									x				
F10		x													
F11									x	x					
F12		x													
F13										x					
F14								x	x	x		x	x		
F15								x	x	x		x			
F16								x	x	x		x	x		
F17						x	x								
F18						x				x				x	
F19	x														

Tabulka 3.1 Mapování požadavků a případů užití

Kapitola 4
Návrh

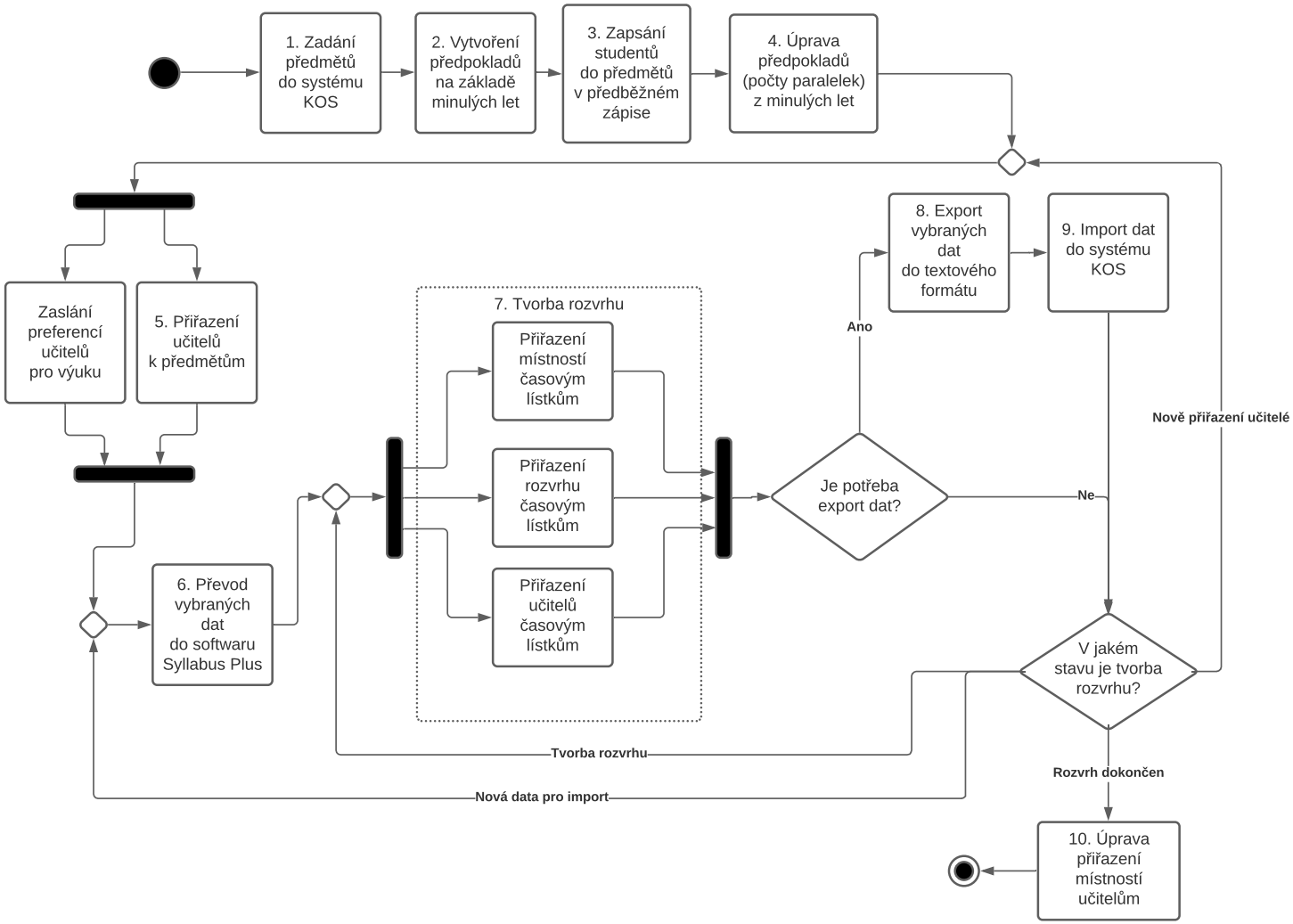
4.1 Návrh

Hlavním cílem nového řešení je grafická aplikace s uživatelským rozhraním a částečnými převody dat. Uživatelské rozhraní je velmi důležité pro to, aby uživatel měl jistotu a kontrolu nad tím, co aplikace s daty udělá. Při zacházení je zásadní, aby bylo v každém kroku jasné, se kterými daty se pracuje, případně co se změní nebo přepíše po provedení importu nebo exportu. Jako řešení, aby aplikace měla své využití opakovatelné a pomáhala tak mnohem více při tvorbě rozvrhu, bude sloužit import a export dat i po menších částech a to v jakémkoli čase během tvorby rozvrhu. Kromě typu pro převod (časové lístky, učitelé, místnosti) bude možné specifikovat i jednotlivé položky. Půjde tak nově převést například konkrétní časový lístek, jeden konkrétní učitel nebo třeba celý předmět.

4.2 Změna procesu tvorby rozvrhu

Na základě požadovaných vylepšení se proces tvorby rozvrhu změní. Oproti původnímu schématu přibude několik nových smyček a akce s úpravou rozvrhu po jeho samotném dokončení úplně zmizí. Celý upravený proces je znázorněn na obrázku 4.1.

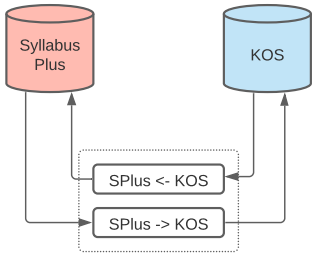
Smyčky vznikají po části tvorby rozvrhu, případně exportu dat. Následně se vrací před převod do softwaru Syllabus Plus, tvorbu rozvrhu nebo před přiřazení učitelů k předmětům. Tímto vzniká v procesu velké množství nových možností. Akci s úpravami rozvrhu mohu úplně odstranit proto, že vznikem nových smyček jsou situace s případnými požadavky na změny již v procesu zahrnuty.



Obrázek 4.1 Schéma průběhu tvorby rozvrhu – nové řešení

4.3 První návrh přístupu

Nyní se zaměřím na část převodů a práce s aplikací. Pro první návrh jsem vycházel z funkcionality aktuální aplikace a ze schématu nového procesu. Přidal jsem možnost načtení dat ze systému a výběr jakékoli části dat pro převod. Aplikace se spustí, uživatel si vybere data pro převod a po dokončení převodu se zase ukončí. Práce s daty je znázorněna na obrázku 4.2.



Obrázek 4.2 První návrh – práce s daty (nepoužito)

Následně jsem vytvořil tři hlavní scénáře používání s touto úpravou.

1. UCN1 – Import dat ze systému KOS do softwaru Syllabus Plus
2. UCN2 – Export dat ze softwaru Syllabus Plus do systému KOS
3. UCN3 – Import a export nových časových lístků ze systému KOS do softwaru Syllabus Plus a zase zpátky do systému KOS

První dva scénáře vypadají dobře, ale u třetího jsem zjistil problém. V rámci používání je potřeba si ukládat informace stranou. Protože se mi zdál tento krok zbytečný, přišel jsem s novým návrhem.

4.3.1 UCN1 – Import dat ze systému KOS do softwaru Syllabus Plus

Spustíme program, načteme si data ze systému KOS, vybereme všechna a importujeme do softwaru Syllabus Plus.

4.3.2 UCN2 – Export dat ze softwaru Syllabus Plus do systému KOS

Spustíme program, načteme si data ze softwaru Syllabus Plus, vybereme všechna a exportujeme do systému KOS.

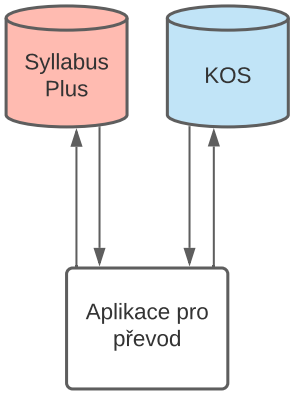
4.3.3 UCN3 – Import a export nových časových lístků ze systému KOS do softwaru Syllabus Plus a zase zpátky do systému KOS

Spustíme program, načteme si všechna data ze systému KOS, vybereme časové lístky, o kterých nám bylo oznámeno, že se změnily, a importujeme do softwaru Syllabus Plus. Nyní začneme s úpravou rozvrhu. Bohužel, jelikož nevíme, co všechno můžeme ovlivnit, musíme si poznamenávat

všechny provedené změny. Dále otevřeme zpátky program, vybereme všechny ovlivněné časové lístky a další položky, které jsme si poznamenali při úpravě rozvrhu, a poté zvolíme export do systému KOS.

4.4 Druhý návrh přístupu

Nový návrh je založen na tom, že aplikace po spuštění načte data ze systému KOS a ze softwaru Syllabus Plus najednou. Poté si vytvoří interně pro každý systém model a data odprezentuje uživateli v přehledné formě a zvýrazní rozdíly. Uživatel si tak bude moci prohlédnout, ve kterém systému jsou jaká data, a na základě těchto informací se může rozhodnout, které akce chce s daty provést. Práce s daty je znázorněna na obrázku 4.3. Můžeme si povšimnout, že v rámci aplikace pro převod není definováno, která data budou kam převedena. Je to tak proto, že aplikace zůstává po převodu dále spuštěná. Díky tomu může uživatel data importovat nebo exportovat vícekrát.



Obrázek 4.3 Druhý návrh – práce s daty

Zkusil jsem tento návrh na stejných scénářích. Můžeme si povšimnout velké změny u třetího, kdy importujeme pouze část. Díky porovnání změn si nemusí uživatel poznamenávat provedené změny. Na základě toho, že tento případ bude velmi častý a je to jeden z hlavních požadavků, zvolím koncept druhé verze návrhu.

4.4.1 UCN1 – Import dat ze systému KOS do softwaru Syllabus Plus

Spustíme program, načteme si data ze systému KOS a softwaru Syllabus Plus. Zjistíme, že v softwaru Syllabus Plus žádná ještě nejsou, vybereme proto všechna a importujeme do softwaru Syllabus Plus.

4.4.2 UCN2 – Export dat ze softwaru Syllabus Plus do systému KOS

Spustíme program, načteme si data ze systému KOS a softwaru Syllabus Plus. Zjistíme, že v systému KOS jsou jiná data, vybereme proto všechna a exportujeme do systému KOS.

4.4.3 UCN3 – Import a export nových časových lístků ze systému KOS do softwaru Syllabus Plus a zase zpátky do systému KOS

Spustíme program, načteme si data ze systému KOS a softwaru Syllabus Plus. Z vizualizace rozdílů zjistíme, že nám v softwaru Syllabus Plus chybí některé časové lístky. Zkontrolujeme si, zda souhlasí se seznamem změn, který jsme dostali, a importujeme je do softwaru Syllabus Plus. Díky rozdílovému zobrazení si nemusíme poznamenávat, které úpravy děláme, a můžeme tak rozvrh přeuspořádat. Půjdeme zpátky do programu, provedeme aktualizaci, zobrazíme si rozdíly a změněné položky exportujeme do systému KOS.

4.5 Práce s daty v systému KOS

Pro přístup k části dat systému KOS je přístupné KOS API vytvořené Jakubem Jirůtkou. Jedná se o rozhraní v podobě RESTful (implementující standard REST) webových služeb, které zprostředkovává přístup k vybrané části dat přímo z databáze systému KOS. Dokumentace je dostupná v [2]. Výstupem jsou data ve formátu XML a standardu Atom. [2]

Kromě KOS API existuje ještě Sirius API, které je využíváno pro webový rozvrh. Dokumentace je dostupná v [3]. Hlavním zdrojem informací pro toto API je KOS API. Data jsou pak převedena podle adekvátních pravidel do lokální databáze, ze které pak API získává informace. Výstupem jsou data ve formátu JSON. [4]

Pro úpravu dat v systému KOS slouží textový soubor ve formátu csv, který po načtení upraví odkazované časové lístky. Každá řádka v souboru reprezentuje jeden časový lístek a má formát viz obrázek 4.4.

<katedra_id> <listek_id> <mistnost_id> <ucitel1_id> <ucitel2_id> <cislo_dne> <cislo_hodiny> <souběžné vyúčtování>

Obrázek 4.4 Formát řádku textového souboru pro systém KOS

Souběžné vyúčtování je pole, které na FITu nevyužíváme. Jediná data, která jdou takto změnit, je tedy rozvrh, místnost a učitelé. Další omezení tohoto formátu je počet učitelů přiřazených k jednomu časovému lístku, který je stanoven na maximálně dva.

4.5.1 Data

Použitá část struktury systému KOS a Sirius API je zachycena na obrázku 4.5.

První rozdělení celků je podle středisek. Všechna střediska získám pomocí `/divisions`.

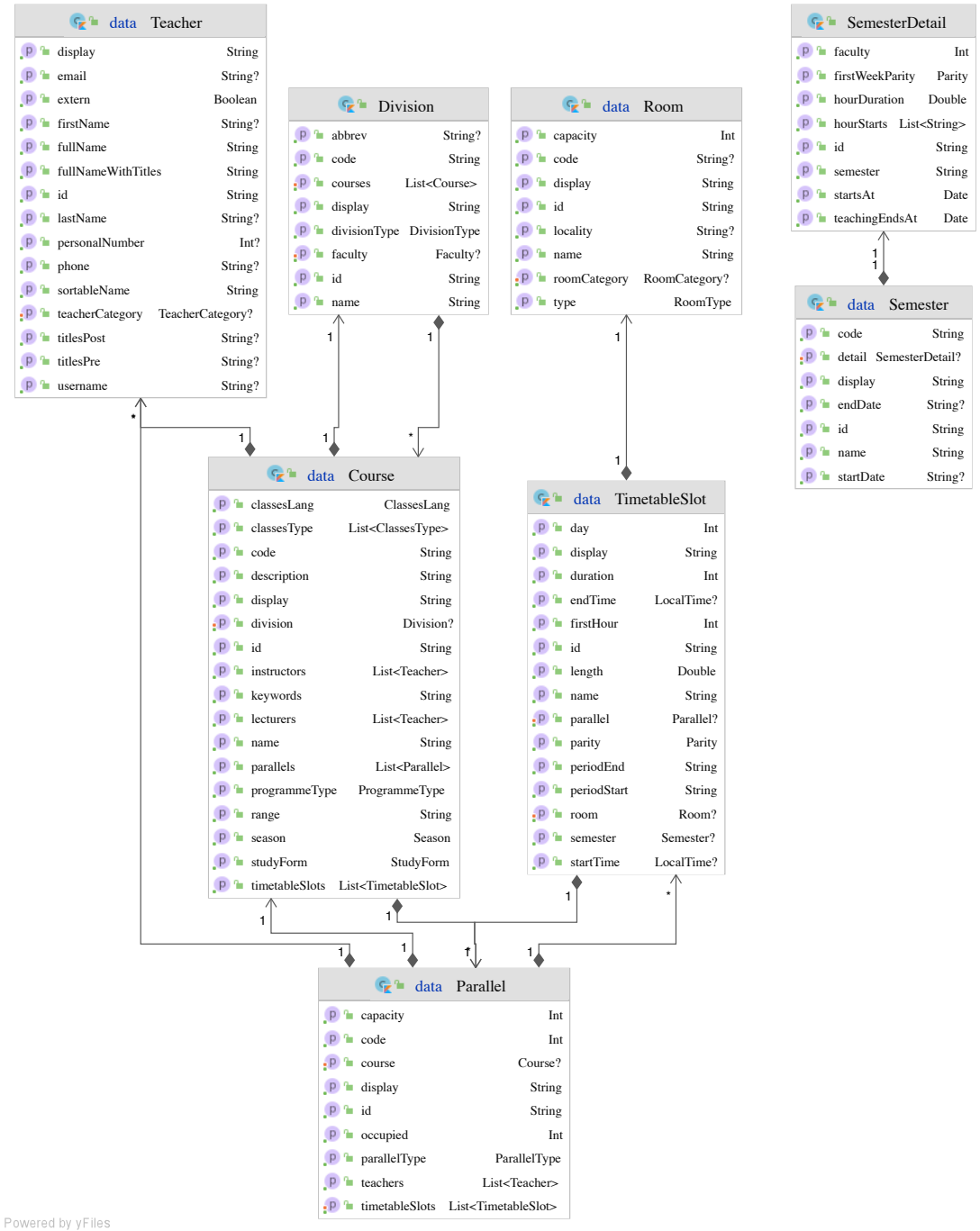
Součástí středisek je jedno hlavní (ČVUT), sdružující všechny fakulty. Poté jsou mezi středisky samotné fakulty a pod fakultami katedry.

Střediska, které jsou fakultami, mají své předměty. Předměty získám pomocí `/divisions/kód_střediska/courses`.

Každý předmět má své vypsané paralelky, které získám pomocí `/courses/kód_předmětu/parallels`. Paralelka poskytuje vlastnosti jako kapacitu, svůj kód, typ nebo učitele, kteří ji vyučují. V rámci paralelky mohou být dle dokumentace zapsáni maximálně čtyři vyučující. Každá paralelka má zároveň seznam časových lístků (TimetableSlot). Obsahuje informace o dni, délce, začátku, zda je vyučovaný týden sudý nebo lichý a ve které místnosti je vyučovaný.

Předměty získané podle střediska nejdou vyfiltrovat podle semestru, je proto nutné provést tuto operaci na úrovni aplikace.

Vyučující je možné získat buď všechny, podle středisek nebo třeba předmětů. Jelikož jich je v databázi přes třicet tisíc, je potřeba provést o něco náročnější filtraci. Učitele pro předmět



Obrázek 4.5 Diagram použité části datového modelu KOS API a Sirius API

získám pouze pomocí `/courses/kód_předmětu`, kde musím zároveň vyplnit i parametr `sem=kód_semestru`. Při vyplnění semestru tak dostanu ještě učitele, kteří jsou organizováni do skupin zkoušejících, garantů a vyučujících.

Místnosti jdou získat pouze všechny najednou pomocí `/rooms`. Další filtry tak musí být případně udělány mimo práci s KOS API.

Téměř všechna data se dají filtrovat podle semestru. Je tak možné získat pouze potřebná data a není nutné se zatěžovat například několik let starými vypsanými paralelkami. Dostupné semestry získám pomocí `/semesters`. Dalším důležitým bodem je získání aktuálního semestru pomocí `/semesters/current`. V rámci KOS API jsou dostupné pouze informace o začátku, konci a názvu semestru. Pro zjištění větších podrobností se tak hodí využít Sirius API. V tomto API získám po zavolání `/faculties/identifikátor_fakulty/semesters/kód_semestru` kromě původních údajů navíc i paritu prvního týdne, délku hodiny a začátky hodin (seznam, může se hodit pro pozdější zobrazení).

4.6 Práce s daty v softwaru Syllabus Plus

Pro práci s daty softwaru Syllabus Plus je dostupných několik možností. Těmi nejdůležitějšími jsou ale skripty a Automation Server.

4.6.1 Skripty

Skripty jsou objekty, které jsou načteny do aplikace, a obsahují posloupnost akcí [5]. Jejich definováním tak lze dosáhnout komplikovanějších operací, které se dají volat opakovaně [5]. Existuje pět typů skriptů. Těmi jsou

- Emailové skripty,
- Výměnné skripty pro načítání a ukládání dat v UEF formátu,
- ODBC skripty používající SQL jazyk pro komunikaci s ODBC databází,
- Výkazové skripty, které dokáží vypsat ASCII text do souboru,
- Šablonové skripty, které umí spouštět výkazy a ukládat je do souboru. [5]

Z těchto typů jsou vhodné výměnné skripty a ODBC skripty.

4.6.2 ODBC a výměnné skripty

Pomocí skriptů lze předdefinovat operace. To znamená, že je možné vytvořit skript, který převede všechna data do softwaru Syllabus Plus a který je zase dostane z něj. Lze vytvořit i skripty, které například převedou pouze část dat. Třeba pouze učitele, nebo pouze místnosti v jisté kategorii. Skripty dle dokumentace nepodporují argumenty [5]. Bez nich jsou možnosti částečných převodů omezené na předem definované celky.

4.6.3 Automation Server

Využití Syllabus Plus Automation Server a s tím spojené COM binární rozhraní je další možností pro práci s daty. COM rozhraní dovoluje externím aplikacím přímo komunikovat se softwarem Syllabus Plus. Na rozdíl od skriptů, které umí pouze import a export dat, lze pomocí rozhraní přistupovat přímo k vnitřním datům, funkcím nebo událostem aplikace. Nejde ale o technologii softwaru Syllabus Plus. Jedná se o standard v rámci platformy Microsoft Windows, umožňující komunikaci mezi různými aplikacemi. Součástí softwaru Syllabus Plus je typová knihovna, která

obsahuje informace o COM rozhraních pro práci s daty. Knihovna obsahuje přístupné atributy objektů, metody (například naplánování časového lístku) a události. [6]

V praxi to znamená, že má vývojář k dispozici kompletní rozhraní pro práci s téměř všemi daty softwaru Syllabus Plus a může je spravovat. Pro využití v mé práci to znamená, že si může uživatel vybrat, která data by chtěl převést bez nutnosti předem definovaných scénářů.

V rámci nastavení spojení je potřeba udělat pár kroků. Jako první je spuštění a inicializace obrazu softwaru Syllabus Plus. Poté v horní sekci v menu je položka COM. Jako možnosti jsou k dispozici nastavení identifikátoru a registrace samotného serveru. Je praktické si pro každý obraz nastavit vlastní identifikátor, aby nedocházelo zbytečně k nechtěným připojení do jiných obrazů. Změna identifikátoru je možná po kliknutí na položku „ProgID...“. Identifikátor se pak musí uložit do obrazu. Po uložení se software Syllabus Plus restartuje a pro zaregistrování serveru stačí jen kliknout na položku „Zaregistrovat server“.

Po úspěšném zaregistrování COM serveru poté může jakákoli aplikace přistupovat k datům pomocí nastaveného identifikátoru. V rámci dokumentace je uveden příklad s připojením v jazyce .NET Visual Basic pro Windows Word, kde hlavní podstatu tvoří nastavení unikátního identifikátoru *progId* a získání aktivní instituce (*ActiveCollege*) [6].

4.6.4 Data

Data softwaru Syllabus Plus mají použitou část struktury zachycenou na obrázku 4.6.

Při inicializaci obrazu je potřeba správně zadefinovat aktivní instituci. Důležité je nastavit začátek semestru a pracovní dny. Podle prvního dne semestru se určuje i den začátku týdne. Pokud by se tak nastavil začátek na středu, bude začínat každý týden ve středu. Dalším důležitým parametrem je začátek a konec výuky. Výuka na FIT je rozdělena do hodin dlouhých čtyřicet pět minut, kde je po každých dvou hodinách patnáctiminutová přestávka. Proto je potřeba využít malý trik při inicializaci a připočítat délku přestávky k jednotlivým hodinám.

Vznikne tak hodina dlouhá 45 + 15 / 2 = 52,5 minut. První hodina začíná v 7:30 (fakultní začátek výuky) a poslední hodina končí v 20:38 (na fakultě končí poslední hodina v 20:30, nicméně potřebujeme vykompenzovat délku hodiny i s přestávkou). Čas 20:38 je tam proto, že při inicializaci se nedefinuje délka hodiny, ale počet hodin během dne. Vezmu-li tak délku jedné hodiny jako 52,5 minuty a vynásobím patnácti (délka přestávky), dostanu 13 hodin a 7,5 minuty. Při přičtení k začátku výuky 13,125 hodiny, dostanu čas 20:37,5. Po zaokrouhlení tak dostávám 38 minut, což vychází na průměrnou délku hodiny 52,53 minuty.

Rozdíl 0,03 minuty se v rámci délek hodin nijak neprojeví, takže není potřeba jej více řešit. Napadla mě ještě možnost přidat jednu hodinu navíc, aby nevycházel výsledný počet desetinně, nicméně to by mělo dva důsledky. Prvním je ten, že by se dala naplánovat hodina na nevalidní čas (z pohledu hodin fakulty), což by mohlo způsobit problém. Druhým je zvyk rozvrháře pracovat s daným počtem v rámci rozhraní softwaru Syllabus Plus a změna by tak mohla způsobit zbytečnou chybovost.

V softwaru Syllabus Plus jsou jako hlavní celek (pod samotnou institucí) používány katedry. Katedra má pouze název a popis. Pod katedrami jsou předměty. Předměty mají název, popis a svou katedru.

Pod předměty jsou přímo „časové lístky“, kterým se říká aktivity. Jedna aktivita označuje konkrétní okno v rozvrhu. Má svůj název, popis, předmět, velikost, typ (přednáška, cvičení, proseminář) a délku. Pro naplánování aktivity jsou potřeba ale i další informace. První je parita. Tím je myšleno, zda se aktivita vyučuje sudý nebo lichý týden, případně každý týden. Po nastavení parity se nastaví, který den a hodinu by se měla aktivita vyučovat. Kromě časového plánu se dají ke každé aktivitě přiřadit i učitelé a místnosti. V rámci softwaru Syllabus Plus se dá jedné aktivitě přiřadit neomezené množství místností a učitelů. V rámci systému KOS však takové možnosti nejsou dostupné. Tam platí omezení na jednu místnost a dva učitele (k tomu se dostanu později).

Obrázek 4.6 Diagram použité části datového modelu softwaru Syllabus Plus

Vyučující jsou organizováni do skupin podle kombinace předmětu a přednášky, nebo cvičení. Mají svůj název a popis.

Místnosti jsou organizovány do předem definovaných skupin. Těmi jsou například počítačové místnosti, místnosti pro přednášky nebo laboratoře.

Samotné roztřídění do skupin u vyučujících a místností nemá žádnou funkci. Slouží však pro snazší orientaci při přiřazování těchto položek k aktivitám.

Kromě základních položek je potřeba ještě pár pomocných. Těmi jsou typy aktivit (laboratoř, přednáška, cvičení) a dostupnosti (sudý, lichý týden, oba týdny).

4.7 Design

Pro vytvoření designu je klíčových pár požadavků. Těmi jsou 1:1 srovnání, aby bylo možné porovnat data mezi systémy, poté možnost načtení všech dat na jedno kliknutí a možnost zobrazit data pro časové lístky, učitele a místnosti. Začal jsem proto navrhovat první verzi programu, nejprve pro časové lístky.

4.7.1 První verze návrhu

Nejdříve jsem rozdělil obrazovku na dvě části. Na levé straně se nachází data ze systému KOS a na pravé straně ze softwaru Syllabus Plus. Doprostřed, mezi jednotlivé sekce, jsem dal tlačítko pro načtení všech dat. Pod každou sekcí je pak tlačítko, které znázorňuje převody dat mezi systémy.

Pro návrh jsem vycházel z toho, že data jsou uspořádána do hierarchie, kde nejvýše jsou katedry, pod nimi předměty a pod předměty časové lístky. Jednotlivé sekce jsem podle toho rozvrhl jako rozbalovací nabídky, kde se nejdříve vybere katedra, poté předmět a na závěr časový lístek. Pod výběrem kategorií se pak ukáží informace k danému časovému lístku. Mockup návrhu je vidět na obrázku 4.7.

Po jeho vytvoření jsem zkusil projít některé scénáře a přišel jsem na několik problémů. Tím prvním a nejdůležitějším je to, že není možné vybírat časové lístky pro převod. Vlastně celé rozhraní nepodporuje výběr více než jedné katedry, jednoho předmětu a jednoho časového lístku. Dalším problémem je, že nejde jednoduše porovnat počet časových lístků a předmětů. Uživatel by tak musel vždycky rozbalit jednu nabídku a poté druhou, kdy mezitím by se mu první schovala.

Rozhodl jsem se proto udělat nový návrh.

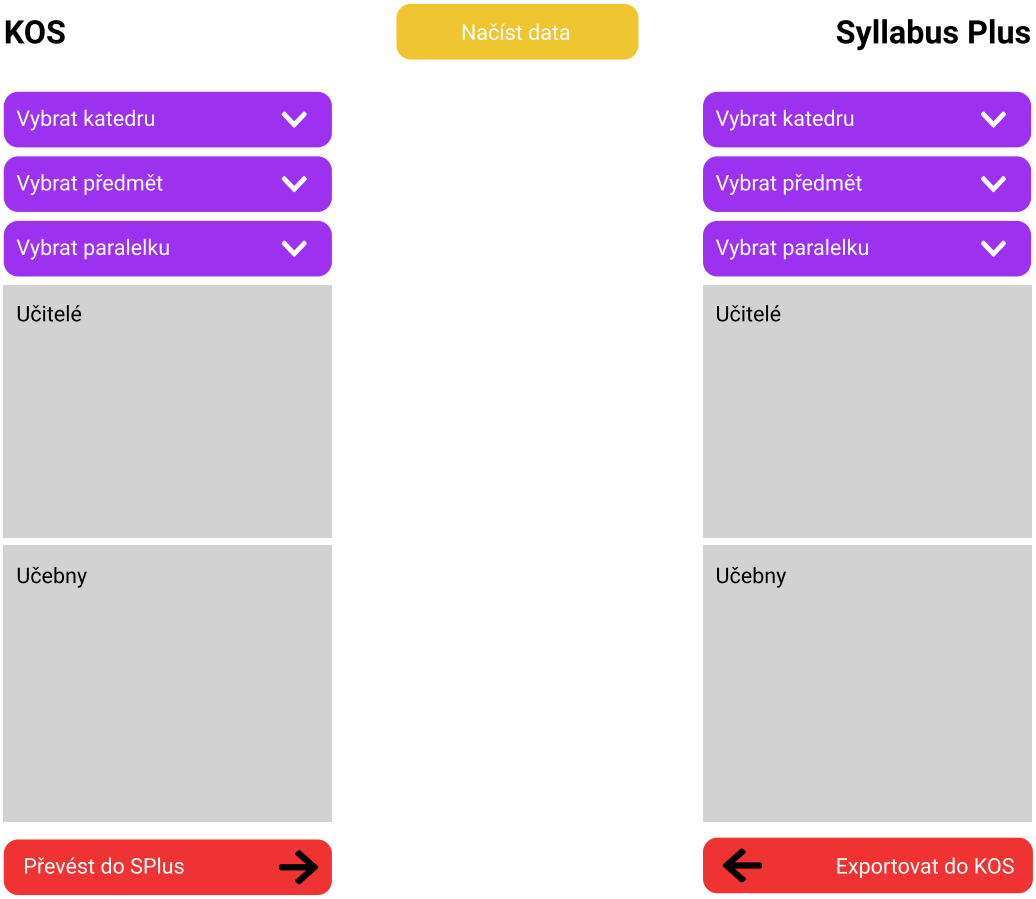
4.7.2 Finální návrh

Z původního řešení se mi líbilo rozdělení prostoru aplikace na dvě části, takže to jsem zachoval. Co jsem ale hned změnil, je zobrazení hierarchických dat. Místo rozbalovacích nabídek jsem udělal v každé sekci tři sloupce.

Jeden sloupec reprezentuje katedry, druhý předměty a třetí časové lístky. Aby se dalo v těchto sloupcích pohybovat, tak při vybrání katedry se zobrazí všechny její předměty a při vybrání předmětu se zobrazí všechny jeho časové lístky. Úplně stejné zobrazení jsem dal pro oba systémy, nicméně u softwaru Syllabus Plus jsem zpřeházel pořadí sloupců. Vyřešil jsem tak problém, kdy se nedalo jednoduše porovnat více dat.

Zároveň jsem si uvědomil, že detail časového lístku není vždycky nejdůležitější a uživatele bude zajímat spíše až v případě problému nebo rozdílu. Díky rozložení do sloupců jsem mohl přidat ještě další funkcionality z požadavků. Před každou položkou se nachází výběrové políčko, pomocí kterého si může uživatel označit záznamy, které by chtěl převést. Na pravé straně pak zobrazuji indikátor, který identifikuje, že se v rámci dané kategorie nachází rozdíl oproti datům v druhém systému.

Na závěr jsem přidal do prostřední části výběr mezi pohledy časových lístků, učitelů a místností. Pohledy pro učitele a místnosti budou také rozděleny na sloupce, samotné sloupce však

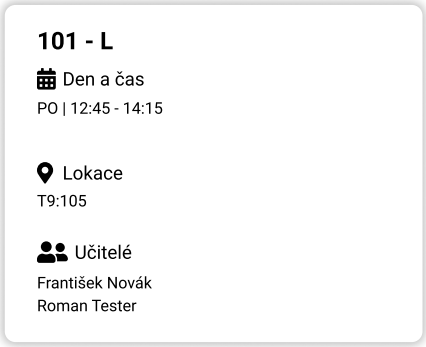


Obrázek 4.7 První návrh rozhraní – nepoužito

budou specifické pro dané kategorie. Pro místnosti budou v prvním sloupci předpřipravené kategorie a ve druhém místnosti. U učitelů bude v prvním sloupci předmět a typ (přednáška, cvičení) a v druhém sloupci bude učitel.

V levém horním rohu se nachází rozbalovací nabídka pro výběr semestru. Tento výběr se týká pouze dat ze systému KOS, proto se nenachází v prostřední části.

Samotný detail časového lístku jsem přesunul do vlastního dialogového okna. Jakmile uživatel klikne na jakýkoli časový lístek, otevře se mu dialogové okno s více informacemi. Návrh okna je na obrázku 4.8



Obrázek 4.8 Návrh dialogového okna s informacemi o časovém lístku

Rozhraní tak splňuje nejdůležitější požadavky a dává prostor pro doplnění všech ostatních funkcí. Jeho návrh je na obrázku 4.9. V dalších krocích už jsem jen přidával funkcionality na základě požadavků.

Do dolní části jsem k tlačítkům pro převody přidal i tlačítka pro vybrání a zrušení výběru všech položek. Tato tlačítka jsou obzvlášť důležitá při prvotním převodu všech dat. Zároveň přibylo ještě výběrové pole před názvem každé kategorie. Zaškrtnutím se pak vyberou všechny položky spadající do dané kategorie. U každé položky a za hlavičkou kategorie se pak nachází barevná tečka. Červená tečka znázorňuje chybějící položku v druhém systému a šedá tečka znázorňuje, že informace o dané položce se mezi systémy v některém z parametrů liší. Pod nadpisem každé kategorie se pak nachází vyhledávání.

Mezi sekcemi přibyla tlačítka pro aktualizaci dat v jednotlivých systémech odděleně a tlačítko pro manuální spárování položek.

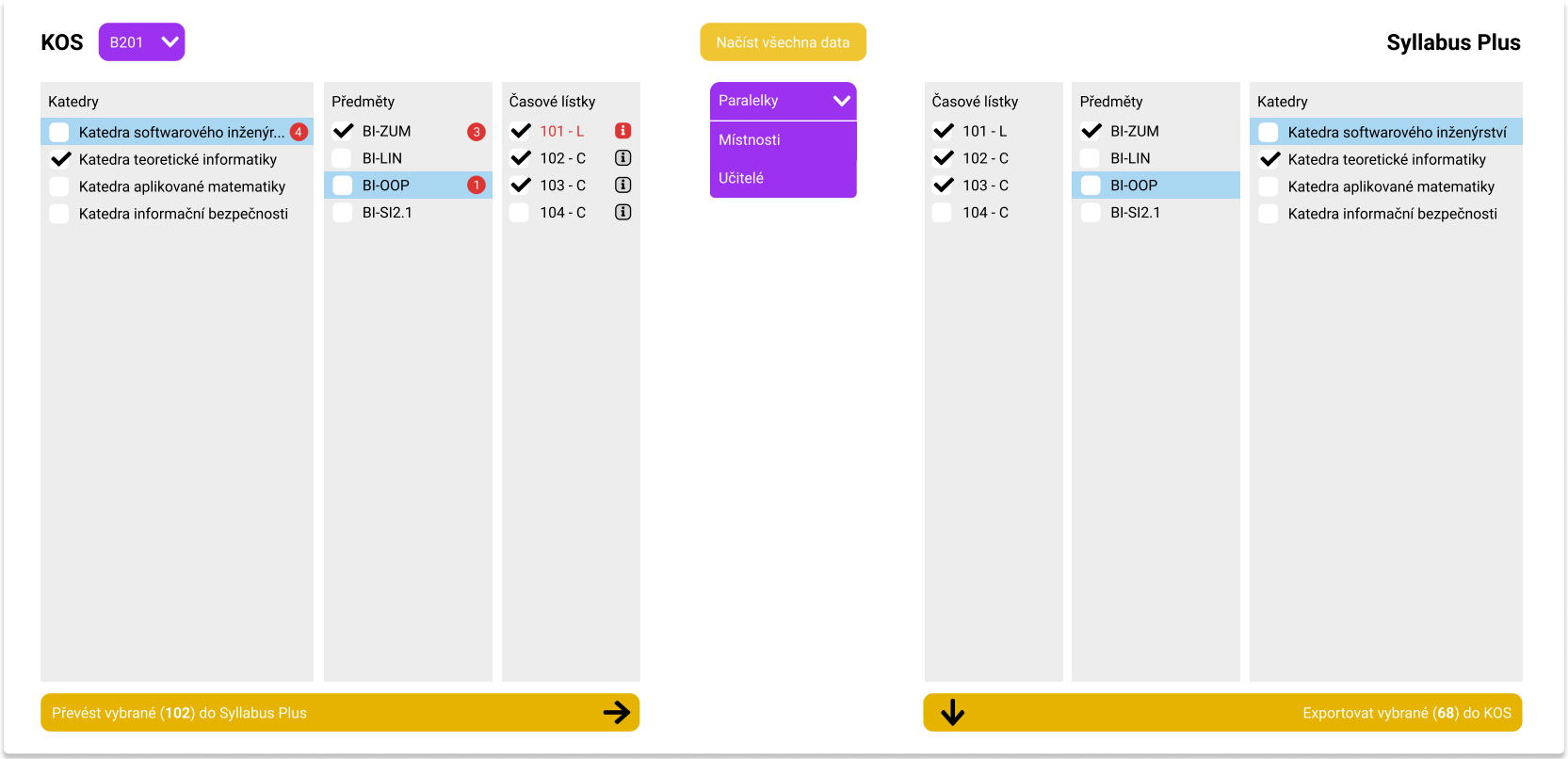
V horním pravém rohu jsou ještě dvě nové funkcionality pro software Syllabus Plus. Tou první je nastavení spojení. Po kliknutí na tlačítko s identifikátorem spojení zobrazí dialog pro případnou změnu identifikátoru a pokusu o znovunavázání spojení. V pravém rohu se nachází ještě ikonka, která zobrazí dialog pro prvotní nastavení obrazu softwaru Syllabus Plus.

Pro převody dat existuje série dialogových oken. Před každým převodem ze systému KOS do softwaru Syllabus Plus se zobrazí dialogové okno, ve kterém je seznam všech vybraných položek. U zobrazení časových lístků si může uživatel ještě vybrat, zda chce převést všechny informace, nebo třeba pouze údaje o místnostech či učitelích.

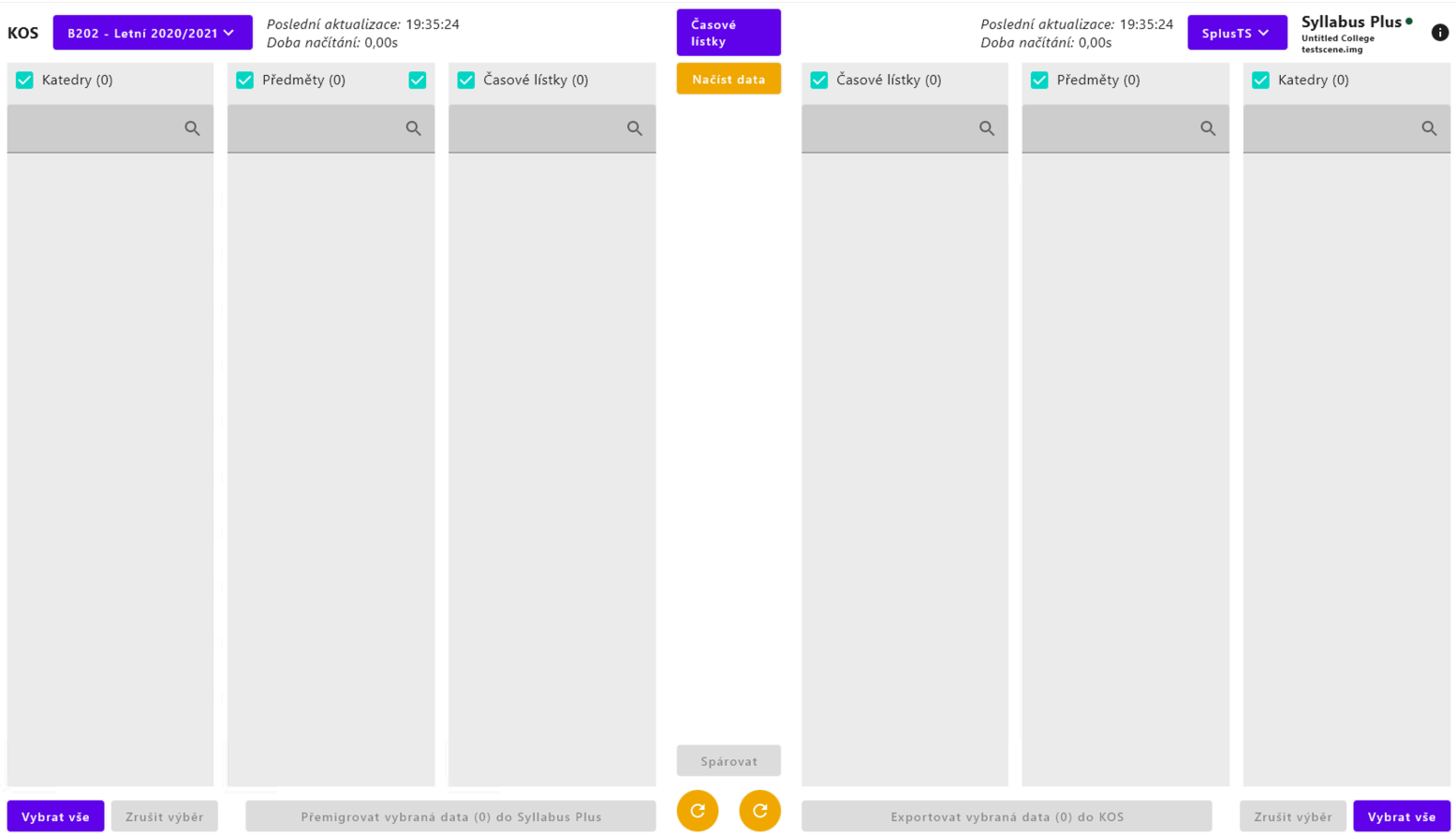
Před exportem dat ze softwaru Syllabus Plus se zobrazí dialogové okno, ve kterém se zobrazí všechny vybrané položky. U každé položky je napsáno, která data se konkrétně změní oproti datům, která se aktuálně nachází v systému KOS.

Po spuštění převodu dat se uzamknou všechny komponenty a zobrazí se lišta ukazující průběh převodu. V dolní části může uživatel převod kdykoli přerušit. Po dokončení převodu se v případě úspěchu zobrazí pouze oznámení v dolní části obrazovky. V případě chyby se ale zobrazí další dialogové okno, ve kterém jsou uvedeny všechny položky, u kterých převod selhal včetně důvodu.

Finální návrh je na obrázku 4.10.



Obrázek 4.9 Druhý návrh – podklad pro finální návrh



Obrázek 4.10 Finální návrh

4.8 Splnění požadavků

Samotné grafické rozhraní aplikace splňuje požadavek N1. Zobrazení dialogového okna s informacemi pro inicializaci obrazu softwaru Syllabus Plus splňuje požadavek F1. Uspořádání dat do sloupců a rozdělení obrazovky na dvě části splňuje požadavek F16. Libovolný výběr dat a částečné převody u všech pohledů splňuje požadavky F2, F3, F5, F6, F9, F10 a částečný export u pohledu časových lístků splňuje požadavky F17 a F18. Požadavek F4 je splněn pouhým přidáním sloupce navíc k pohledu místností, není nutné cokoli v návrhu měnit, jedná se pouze o implementaci. Požadavek F7 je implementačně závislý, nesouvisí s návrhem rozhraní. Možnost výběru informací u převodu časových lístků pokrývá požadavek F8. Tlačítko pro spárovaní položek splňuje požadavek F11. Dialogové okno před převodem dat do systému Syllabus Plus splňuje požadavek F12 a dialogové okno při exportu dat splňuje požadavek F13. Rozdíly mezi položkami zobrazené pomocí barevné tečky splňují požadavek F14. Dialogové okno zobrazené po kliknutí na časový lístek splňuje požadavek F15. Výběr semestru v levém horním rohu splňuje požadavek F19. Filtr u každého sloupce splňuje požadavek N2.

Tímto uživatelské rozhraní splňuje všechny požadavky.

Kapitola 5
Realizace

5.1 Výběr technologií

Hlavními rozhodujícími faktory pro výběr technologií jsou grafické uživatelské rozhraní, komunikace se systémem Syllabus Plus přes COM rozhraní a s tím se přímo váže i podpora operačního systému Windows. Potřebuji tak vybrat programovací jazyk, framework pro tvorbu uživatelského rozhraní a knihovnu umožňující potřebnou komunikaci se softwarem Syllabus Plus.

5.1.1 Programovací jazyk

Jako jazykové možnosti pro vývoj aplikace jsem zvolil Java, Kotlin a JavaScript (případně TypeScript). Jsou to jazyky, které znám a se kterými jsem již v minulosti pracoval na tvorbě uživatelského rozhraní.

Nejzajímavější uživatelská rozhraní se mi povedla v minulosti vytvořit pomocí programovacího jazyka JavaScript ve spojení s HTML a CSS. Ačkoliv bych rád tyto technologie využil i v této práci, tak bohužel kvůli potřebám těsné provázanosti s operačním systémem by mohly vznikat zbytečné komplikace při práci se systémovými soubory nebo při snaze o komunikaci přes COM rozhraní. Proto jsem tento jazyk nezvolil pro implementaci aplikace.

Java nebo Kotlin se tak staly hlavními kandidáty. Pro jazyk Kotlin se mi podařilo najít framework pro tvorbu uživatelských rozhraní Compose for Desktop. Tato knihovna mi přišla zajímavá, protože využívá podobných principů reaktivity, jako některé webové frameworky, a pomocí syntaxe jazyka Kotlin odstraňuje potřebu psát uživatelská rozhraní pomocí XML [7]. Navíc Kotlin je open-source staticky typovaný programovací jazyk, který dokáže být přeložen na JVM nebo nativní aplikaci [8]. Oproti jazyku Java je bezpečnější při práci s datovými typy díky podpoře typů s nulovou hodnotou a aplikace jsou tak méně náchylné na chyby s tím spojené [8].

Co mě dále zaujalo, je kompletní kompatibilita s programovacím jazykem Java [8]. To znamená, že kód napsaný v jazyce Kotlin se dá zavolat z jazyka Java a obráceně [8]. Kompatibilita na této úrovni není samozřejmostí, například kód napsaný v jazyce Scala jde sice také volat z jazyka Java, ale existují oblasti, ve kterých nastávají komplikace [9]. Těmi jsou funkcionality, které nemají na úrovni kódu ekvivalent v jazyce Java, jako například *trait* [9].

Protože jazyk Kotlin je ale mladý jazyk od společnosti JetBrains s první verzí z roku 2016, potřebuji najít způsob, pomocí kterého dokáži komunikovat přes COM rozhraní [8]. Jelikož je ale COM rozhraní poměrně staré, tak se mi nepovedlo dohledat žádnou dostupnou knihovnu. To však vůbec nevadí, protože mohu využít jakoukoli knihovnu pro JVM, konkrétně jsem se zaměřil na knihovny pro jazyk Java, pro který se mi jich několik podařilo nalézt.

Protože Compose for Desktop plní požadavek na framework pro tvorbu uživatelského rozhraní

a povedlo se mi najít několik knihoven pro práci s rozhraním COM (více v podkapitole 5.1.4 o COM knihovnách), rozhodl jsem se pro jazyk Kotlin. Navíc díky kompatibilitě se mi nestane, že bych přišel o případnou funkcionalitu, která je v jazyce Java dostupná, a budu moct využívat nové koncepty syntaxe. Mohu využít například generátor kódu, který existuje pouze pro jazyk Java, a poté volat vygenerovaný kód z jazyka Kotlin (více v podkapitole 5.1.4 o COM knihovnách).

5.1.2 Knihovna Coroutines

Velkou výhodou jazyka Kotlin je jeho syntax a tu využívající knihovna Coroutines.

Jedná se o knihovnu, která je úzce spjatá s vývojem jazyka a jeho návrhem. Obsahuje high-level abstrakci nad asynchronními operacemi. Kromě Coroutines existují obecně pro vyřešení problému asynchronní práce další způsoby. Těmi jsou třeba vlákna, callbacky nebo promise. Tato řešení mají však své problémy. U vláken je to například jejich vysoká cena pro správu, omezené množství nebo problémy při snaze o sdílení dat mezi nimi. U callbacků pak problém s vnořováním, kde při větším množství může jednoduše vzniknout těžce čitelný kód, nebo odchytáváním výjimek. U promisů je pak většinou potřeba využít speciální API a s tím se vážou i nové specifické metody. Navíc návratová hodnota takových funkcí je třeba datového typu *Promise*, což zbytečně oddaluje samotná data, která jsou důležitější než obalové třídy. [10]

Myšlenkou, na které jsou založené Coroutines, je pozastavitelný výpočet. To znamená, že vykonávání funkce může být kdykoli během svého běhu pozastaveno a spuštěno někdy později. Ve výsledku to znamená, že vývojář může psát neblokující kód v podstatě stejným způsobem, jako blokující kód. [10]

Coroutines nejsou ale novým konceptem. V programovacích jazycích jsou používány již nějakou dobu. Důležitý je ale způsob implementace v jazyce Kotlin, kde většina práce je delegována na knihovny, a ve většině případů pro vytvoření asynchronní operace stačí pouze přidat klíčové slovo *suspend* do předpisu funkce. [10]

V rámci mé práce mi pomohou jednoduše vytvořit neblokující operace, například při stahování dat ze systémů nebo převodech, a paralelizovat načítání dat pro zrychlení práce.

5.1.3 Framework pro uživatelské rozhraní

Pro grafické rozhraní existuje nově knihovna Compose for Desktop. Jedná se o framework pro jazyk Kotlin, který je založen na moderním balíku nástrojů od Googlu využívající reaktivní přístup [7]. Compose for Desktop zjednodušuje a urychluje vývoj uživatelských rozhraní pro desktopové aplikace [7].

Aktuálně je tento framework v aktivním vývoji a oficiálně je teprve vydána alfa verze. Jeho návrh mi ale přijde známý a zajímavý a navíc je plně zpětně kompatibilní s grafickou knihovnou Swing [11].

Swing je balíček komponent pro tvorbu grafických rozhraní, který je součástí jazyka Java [12]. Jedná se o část JFC (Java Foundation Classes), což je soubor funkcionalit pro tvorbu uživatelských rozhraní nebo přidání grafických funkcionalit do aplikací napsaných v jazyce Java [12].

Pokud by se stalo, že některá funkcionalita bude během vývoje ve frameworku Compose for Desktop chybět, mohu jednoduše využít tu z knihovny Swing.

5.1.4 Přístup ke COM rozhraní

V rámci softwaru Syllabus Plus je k dipozici knihovna TLB. TLB, neboli typová knihovna, je binární soubor, který uchovává informace o COM rozhraní [13]. Těmi jsou vlastnosti objektů a metod, které jsou za běhu přístupné ostatním aplikacím [13]. V rámci jazyka Java však takové definice nemohu jednoduše využít. Potřebuji proto kromě knihovny i konvertor, který dokáže vytvořit třídy z definic.

Jako první jsem našel knihovny JACOB project[1], j-Interop[2], TlbCodeGenerator[3] a com4j[4]. Knihovna JACOB umožňuje dle dokumentace volání funkcí přes COM rozhraní, avšak sama o sobě neobsahuje generátor pro vytvoření tříd a vlastností v jazyce Java.

Knihovna j-Interop dle dokumentace také umožňuje volání patřičných funkcí, ale už ne předgenerování tříd.

Knihovna TlbCodeGenerator slouží pro vygenerování tříd a potřebných souborů pro přístup ke COM rozhraní. Bohužel, tuto knihovnu se mi nepodařilo zprovoznit, takže jsem o ní dále již neuvažoval.

První dvě knihovny jsem si nechal pro případ, kdyby mi něco nefungovalo korektně (například přístup k některým metodám), protože jsem našel lepší knihovnu com4j, která umí obě požadované funkcionality. Knihovna com4j je sice již poměrně stará a oficiální aktualizace vyšla naposledy v roce 2016, nicméně umožňuje jak přístup ke COM rozhraní, tak i vygenerování všech potřebných metod [14]. Abych se vyhnul některým známým problémům, dohledal jsem rovnou její fork[5] od autora archiecobbs, který opravuje chyby a implementuje podporu pro použití try-with-resources přístupu. Poslední commit má tento fork v září roku 2017 a obsahuje navíc oproti oficiální verzi 17 commitů.

Základní použití je velmi jednoduché a krátce popsané v [15]. Pro vygenerování potřebných tříd tak stačilo zavolat příkaz 5.1. Kromě tříd pro všechny metody a vlastnosti byla vygenerována třída *ClassFactory*, která obsahuje metodu *createApplication*. Pomocí zavolání této metody se vytvoří spojení přes COM rozhraní se softwarem Syllabus Plus. Na základě menšího testu spojení a několika základních metod jsem se rozhodl tuto knihovnu využít, protože plně postačí mým potřebám.

Výpis kódu 5.1 Příkaz pro spuštění generace tříd, upraven pro software Syllabus Plus [15]

```
java -jar tlbimp.jar -o splus_com -p splus.com splus.tlb
```

5.1.5 Další knihovny

Kromě dříve uvedených a základních knihoven jazyka Kotlin jsem se rozhodl, že by se mi ještě hodila knihovna pro Dependency Injection, dále knihovna pro vytvoření HTTP spojení, knihovna pro přeložení XML a JSON, a knihovna pro jednodušší práci s datumy a časy.

Pro Dependency Injection se mi podařilo najít několik knihoven, nicméně můj záměr byl, aby podporovala v nějaké formě framework Compose for Desktop. Vybral jsem si tak knihovnu s názvem Koin[6], která by ho dle dokumentace měla podporovat. Její použití je zároveň dostatečně jednoduché. Ve výsledku jsem ale zjistil, že podpora není taková, jakou bych si představoval. Nedařilo se mi dle dokumentace zprovoznit získávání třídy pomocí metody *get*. Nakonec jsem si implementoval tuto metodu sám pomocí využití zpětné kompatibility frameworku s třídami v jazyce Java, a tím jsem problém vyřešil.

Na vytvoření HTTP spojení jsem využil knihovnu Ktor[7]. Jedná se o oficiální knihovnu a framework od společnosti JetBrains pro rychlou a jednoduchou tvorbu webových aplikací, HTTP služeb [16]. Ktor využívá abstrakce asynchronních procesů pomocí Coroutines, které jsou součástí jazyka Kotlin [16]. Framework obsahuje i server, třeba pro tvorbu webových aplikací, nicméně pro mé využití mi postačí pouze klient. U klienta budu potřebovat dvě knihovny pro zpracování dotazů ve dvou různých formátech. Jednou je knihovna pro JSON formát. Ta je součástí Ktor

[1] https://github.com/freemansoft/jacob-project
[2] http://j-interop.org/index.html
[3] https://github.com/matthiasblaesing/TlbCodeGenerator
[4] https://com4j.kohsuke.org/
[5] https://github.com/archiecobbs/com4j
[6] https://insert-koin.io/
[7] https://ktor.io/

knihovny a je možné ji získat jako volitelný balíček. Pro XML využiji knihovnu Jackson, která je také součástí knihovny Ktor, v kombinaci s XML parser[8].

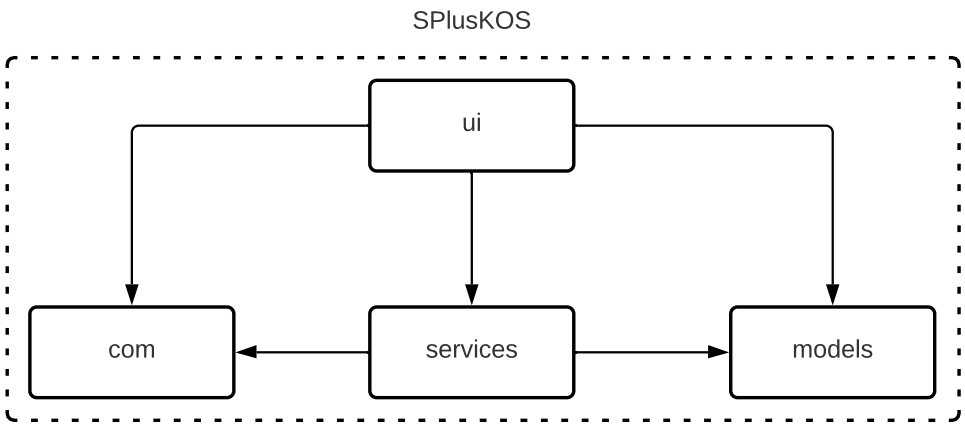
Na závěr ještě využiji drobnou knihovnu pro práci s datumy a časy. Usnadní mi primárně konverze mezi minutami, hodinami nebo třeba sekundami. Jedná se o knihovnu opět od společnosti JetBrains kotlinx-datetime[9]. Je tak plně kompatibilní s jazykem Kotlin.

5.2 Nastavení projektu

Jako nástroj pro kompilaci, vývoj a správu modulů je použitý nástroj Gradle[10]. Projekt aplikace se pak skládá ze čtyř modulů. Těmi jsou:

- com (modul pro přístup ke COM rozhraní včetně pomocných metod),
- models (definice tříd systémů a jejich vlastnosti),
- services (služby pro práci s modely a daty z obou systémů),
- ui (uživatelské rozhraní aplikace).

Návaznosti mezi moduly jsou znázorněny na obrázku 5.1.



Obrázek 5.1 Moduly projektu

Dále je vytvořen ještě modul s názvem *buildSrc*. Tento modul je předdefinovaný pro komplikovanější nastavení více modulů v rámci nástroje Gradle [17]. Mé využití je pro sdílení proměnných, které pak mohu využívat v souborech *build.gradle.kts* (jedná se o soubory sloužící pro nastavení kompilace jednotlivých modulů). Definuji zde verzi jednotlivých knihoven a verzi cíle pro JVM. Samotný modul se ale nijak nepropisuje do funkčnosti aplikace, proto ho v diagramu neuvádím.

5.2.1 Kompilace

Jelikož je aplikace napsaná v jazycích Kotlin a Java, její výstup je v základu ve formátu .jar, což je spustitelný soubor na JVM. Tento soubor lze získat pomocí spuštění Gradle úkolu *packageUberJarForCurrentOS*. Potřebná verze nainstalovaného prostředí jazyka Java je 15.

[8] `https://github.com/FasterXML/jackson-dataformat-xml`
[9] `https://github.com/Kotlin/kotlinx-datetime`
[10] `https://github.com/gradle/gradle`

Compose for Desktop ale umožňuje i lepší distribuci, než tento soubor [18]. Tou je kompilace do binárních souborů různých formátů (.dmg, .deb, .msi, .exe), mě zajímá nejvíce formát .exe [18]. V rámci konfigurace kompilace nastavím informace o spustitelné třídě a cílovém formátu a poté spustím Gradle úkol *packageExe*. Vytvoří se soubor, který končí příponou exe. Tento soubor obsahuje všechny potřebné soubory aplikace včetně prostředí jazyka Java pro samotný běh. Jedná se o instalátor, který při instalaci nakopíruje potřebné soubory do uživatelem definovaného umístění [18]. V případě odinstalace pak může uživatel postupovat stejně jako u většiny programů na systému Windows. V rámci ovládacího panelu si najde aplikaci a vybere odinstalaci.

Může nastat ale i situace, kdy se vyplatí více použít balíček JAR, kvůli úspoře místa na disku. Údaje o velikostech zobrazuje tabulka 5.1.

	Instalátor .exe	Balíček JAR
Distribuovaná aplikace	82 MB	34 MB
Nainstalovaná aplikace	170 MB	—
Běhové prostředí	—	72 MB [19]
Výsledná velikost na disku	**170 MB**	**106 MB**

Tabulka 5.1 Tabulka porovnání velikostí jednotlivých verzí

Vzhledem k přidaným funkcím, nezávislosti na nainstalovaném prostředí jazyka Java a kapacitě dnešních disků pohybující se v jednotkách terabytů, jsem se rozhodl pro použití instalátoru místo balíčku JAR.

Uživatel dostane jeden soubor, který obsahuje instalátor. Ten spustí, vybere si umístění a aplikaci nainstaluje. Po prvním spuštění se ve složce *%appdata%\SPlusKOS* vytvoří soubor pro nastavení a předdefinovaný seznam skupin místností. V souboru pro nastavení upraví uživatel potřebné údaje (*KOSAPIClientId* a *KOSAPIClientSecret*, viz podkapitola 5.4.2) a aplikaci restartuje. Po dokončení by mělo vše plně fungovat.

5.3 Datové modely

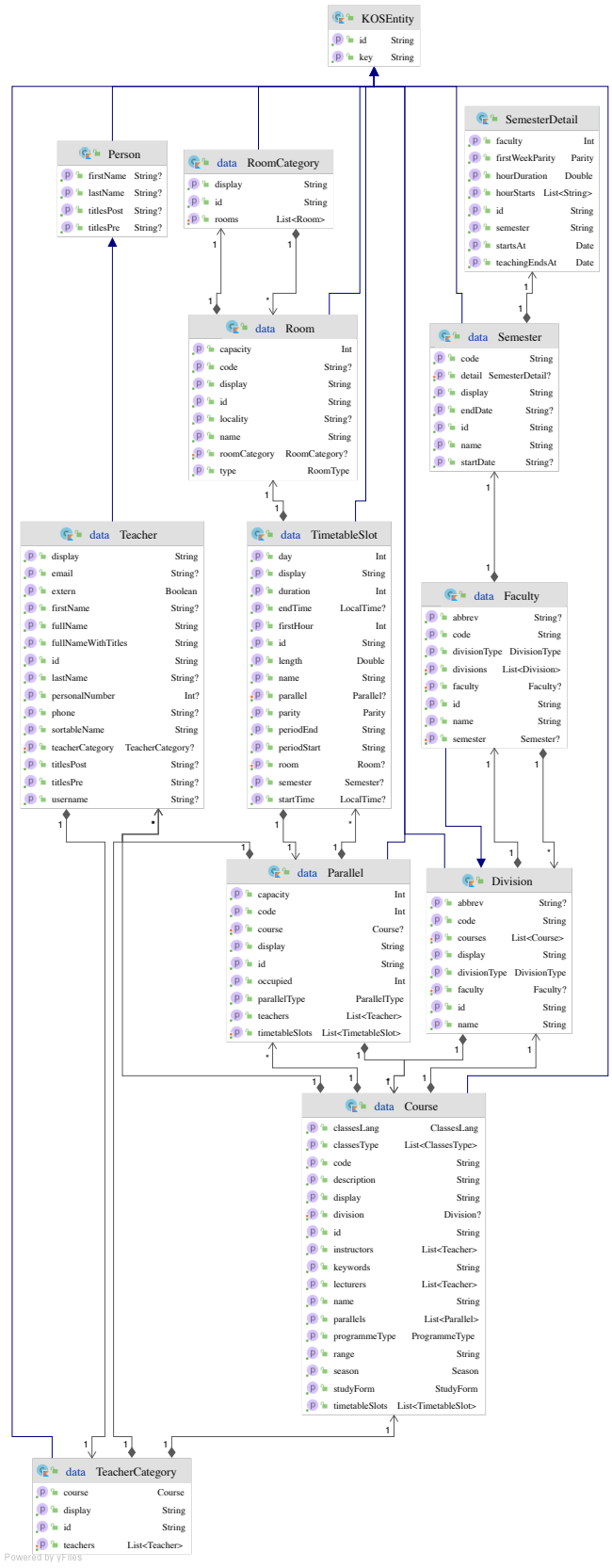
Na začátku implementace jsem nejdříve sestavil dva datové modely. Jeden pro systém KOS a jeden pro software Syllabus Plus. Každý model má jinou strukturu, ve které jsou potřebná data uložena, a také jiné názvosloví položek. Pro pozdější rozvoj a údržbu aplikace je důležité, aby tyto modely byly co nejvíce oddělené, protože pokud bych se rovnou pokusil na sebe data napasovat, nemohl bych provádět některé operace. Přišel bych hlavně o možnost porovnat data v jednotlivých modelech.

5.3.1 KOS

Kompletní diagram datového modelu systému KOS je zobrazen na obrázku 5.2.

Je založen na dvou hlavních rozhraních. Jedná se o *Keyable*, který slouží pro přiřazení unikátního identifikátoru, a *Sortable*, který slouží pro přiřazení klíče pro řazení a formátu zobrazení jako text. Obě rozhraní implementuje třída *KOSEntity*, která rovnou implementuje i metodu pro získání unikátního klíče. Unikátní klíč pro všechny entity v systému KOS je *id*. Z této abstraktní třídy dědí následující třídy:

- *Faculty* (fakulta),
- *Division* (organizační jednotka),
- *Room* (místnost),
- *Parallel* (paralelka),



Obrázek 5.2 Diagram datového modelu systému KOS

- *Teacher* (učitel),
- *TimetableSlot* (časový lístek),
- *Course* (předmět),
- *RoomCategory* (kategorie místností),
- *TeacherCategory* (kategorie učitelů),
- *Semestr* (semestr).

Třídy *RoomCategory* a *TeacherCategory* se v datech KOS API nenachází. Pro mé využití jsou ale potřeba kvůli organizaci místností do předdefinovaných skupin a učitelů podle předmětů a jejich rolí.

Kromě tříd dědících z *KOSEntity* se v modelu používá ještě třída *SemesterDetail*. Obsahuje podrobnější data o semestru, která se nedají získat přes KOS API, ale přes SIRIUS API. Mají tak jinou strukturu. Proto tato třída nedědí z abstraktní třídy *KOSEntity*.

Třídy jsou mezi sebou velmi provázané, nicméně popíši nejdůležitější strukturu návaznosti dat, které jsou používané napříč aplikací. Pro zobrazení časových lístků je důležitá následující struktura. Hlavní třída *Faculty* obsahuje seznam tříd *Division*. Tím se odkazuje na veškeré katedry, které se nachází pod danou fakultou. Každá katedra (*Division*) má seznam tříd *Course* (předmětů). Předmět obsahuje seznam paralelek (*parallels*). Paralelka (*Parallel*) obsahuje seznam časových lístků (*timetableSlots*) a učitelů (*teachers*).

Pro zobrazení místností je hlavní třídou *RoomCategory*. Ta obsahuje seznam místností (*rooms*). A pro zobrazení učitelů je hlavní třídou *TeacherCategory*, která obsahuje dva seznamy učitelů. Prvním jsou učitelé, kteří přednáší (*lecturers*) a druhým jsou učitelé, kteří cvičí předměty (*instructors*).

5.3.2 Syllabus Plus

Všechny entity softwaru Syllabus Plus mají jako předka abstraktní třídu *Entity* a jejich závislosti a hierarchie jsou znázorněny na obrázku 5.3.

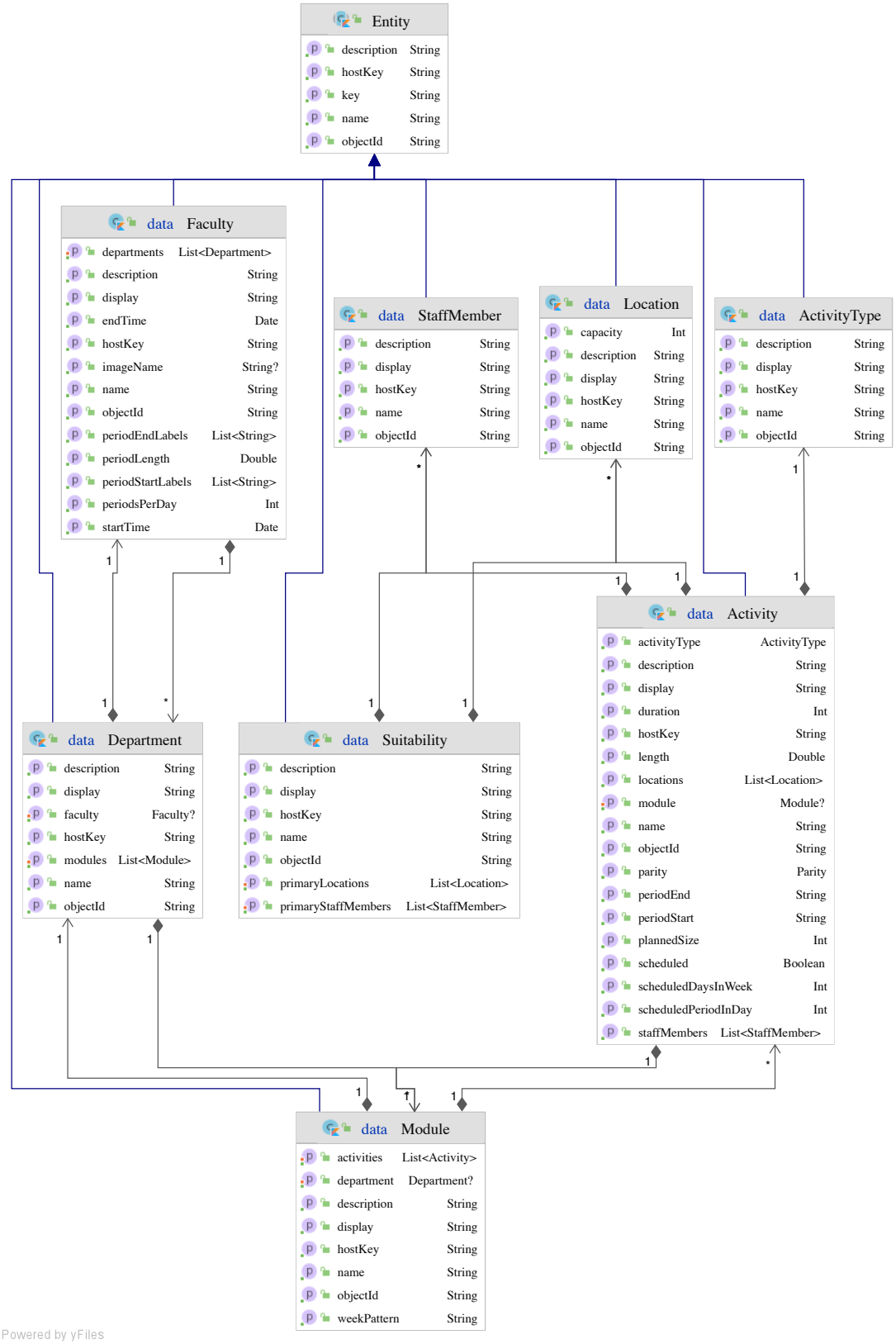
Tato třída implementuje také dvě rozhraní *Sortable* a *Keyable*. Dále definuje čtyři atributy, které obsahuje každá podtřída. Těmi jsou

- *objectId* (unikátní identifikátor v rámci softwaru Syllabus Plus),
- *hostKey* (unikátní identifikátor ze systémů třetích stran, pro mé využití bude daný atribut obsahovat nejčastěji identifikátory systému KOS),
- *name* (název entity),
- *description* (popis entity).

Zároveň definuje unikátní identifikátor rozhraní *Keyable* jako *hostKey*. Díky tomu tak unikátní identifikátor je v obou systémech stejný, což se bude hodit třeba při porovnávání změn.

Dalšími třídami tvořící model softwaru Syllabus Plus jsou

- *Faculty* (fakulta),
- *Department* (katedra),
- *Module* (předmět),
- *Activity* (časový lístek),
- *StaffMember* (učitel),



Obrázek 5.3 Diagram datového modelu softwaru Syllabus Plus

- *Location* (místnost),
- *Suitability* (tento typ je využíván pro vytvoření skupin učitelů a předmětů).

Důležitými strukturálními vztahy mezi entitami je pro zobrazení časových lístků ve třídě *Faculty* seznam kateder (*departments*). Katedra (*Department*) má seznam předmětů (*modules*) a předmět (*Module*) má seznam časových lístků (*activities*). V rámci časového lístku (*Activity*) se pak nachází dva seznamy. Prvním je seznam učitelů (*staffMembers*) a druhým seznam místností (*locations*). V rámci systému KOS jde sice přiřadit pouze jedna místnost, nicméně software Syllabus Plus jich podporuje i více, proto je potřeba pole.

Pro organizaci učitelů a místností do kategorií slouží třída *Suitability*. V ní jsou dva seznamy *primaryStaffMembers* a *primaryLocations*. Software Syllabus Plus umožňuje nastavit jedné třídě *Suitability* jak učitele, tak i lokace. V rámci pohledů však vždy použiji pouze jeden seznam pro jednu kategorii (učitelé, místnosti).

5.4 Přístupy k datům

V této části se zabývám přístupem k datům systému KOS a softwaru Syllabus Plus z pohledu implementace.

5.4.1 COM rozhraní pro přístup k softwaru Syllabus Plus

V modulu *com* jsou k dispozici knihovnou com4j vygenerované třídy a metody pro práci se softwarem Syllabus Plus. Bohužel, všechny předdefinované třídy nejsou plně funkční a existují případy, u kterých jsem musel provést několik změn.

První takovou je podpora předávání objektů z vygenerovaného rozhraní (obsahující vlastnosti pro komunikaci přes COM rozhraní) do vygenerovaných metod (například metoda objektu z jiného vygenerovaného rozhraní). Pokud vygenerovaná metoda přijímá jako svůj parametr objekt vygenerovaného typu, tak při zavolání má aplikace vyhodí výjimku a okno softwaru Syllabus Plus je okamžitě ukončeno. Řešení problému se mi nepodařilo nalézt jak v dokumentaci nebo ve známých a nahlášených chybách, tak ani na diskuzních fórech. Zkusil jsem se proto nad problémem zamyslet a napadlo mě, že příčinou by mohlo být předávání objektu s vazbou na COM rozhraní metodě, která má také vazbu na dané rozhraní. V momentě předání nepotřebuji totiž získávat informace o daném objektu nebo s ním provádět další operace, stačí mi pouze předat jeho aktuální stav. Vyzkoušel jsem proto změnit datový typ z vygenerovaného typu na *Object*, abych změnil instrukci a knihovna objekt dále nezkoušela překládat. Po spuštění vše zafungovalo. Bohužel tak přicházím o typovou kontrolu u těchto metod, nicméně jelikož funkčnost je důležitější, rozhodl jsem se použít danou změnu jako finální řešení.

Dalším problémem je indexace kolekcí. V rámci kolekcí získaných přes COM rozhraní pro software Syllabus Plus jsou položky indexované od jedničky. V jazyce Java (v tomto jazyce jsou vygenerované třídy) a Kotlin jsou však všechny kolekce indexované od nuly. Tento rozdíl způsobuje, že při použití vygenerovaného rozhraní pro kolekce, které je rozšířením iterátoru jazyka Java, dochází při dotazu na první prvek k výjimce. Vytvořil jsem si proto pomocnou metodu *map*, která získá velikost kolekce a poté prochází prvky od prvního indexu a získává jejich data pomocí volání předdefinované knihovní metody *item*. Protože všechny kolekce jsou v rámci vygenerovaných tříd zastřešovány vlastními třídami, musel jsem pro vytvoření obecné metody použít reflexi.

Třetím problémem je volání metod, které očekávají referenci na již existující instanci objektu. Bohužel, po důkladném zkoumání jsem zjistil, že se jedná o takzvanou akci *PropertyPutRef*, která není v základní knihovně naimplementovaná. Mezi jedním repozitářem vycházejícím z původní knihovny jsem ale našel implementaci. Jedná se o fork od autora jasperkrijgsman[11], kde se

[11] `https://github.com/jasperkrijgsman/com4j`

konkrétní úprava nachází v commitu „Support propputref methods dispinterface methods[12]“. Jedná se sice o malou změnu, ale přidává podporu pro anotaci *@PropPutRef*, kterou přesně potřebuji. Potřebné úpravy z commitu jsem tak použil v mnou použité již jednou upravené verzi, zkompiloval a použil výslednou knihovnu místo původní pro moji aplikaci. V předpisu metody jsem pak jen upravil anotaci z *@PropPut* na *@PropPutRef* a změnil typ argumentu na *Object* (kvůli problému popsanému výše). Jakmile jsem tyto změny provedl, tak vše již správně zafungovalo.

Tím jsem vyřešil všechny problémy, na které jsem narazil během práce s touto knihovnou.

S knihovnou se pracuje následovně. Pomocí zavolání tovární metody *createApplication(progID)* (progID je identifikátor pro navázání spojení) se vytvoří nové spojení. Následně pak mohu přistupovat k různým datům. Nejdůležitější je metoda *activeCollege* pro získání aktivní školy. Pod ní se potom dají získat nebo nastavit katedry, předměty, časové lístky, a tak dále.

Protože uživatel může mít najednou otevřených více oken softwaru Syllabus Plus, může být potřeba nastavit ProgID speciálně pro každý obraz. To se dá poměrně jednoduše v horní nabídce softwaru Syllabus Plus pod záložkou „COM“, kde se nachází možnost „ProgID...“. Po jejím kliknutí se zobrazí dialogové okno. Základní progID je nastaveno na hodnotu „Splus“. Nastavení tohoto identifikátoru se ukládá k aktivnímu obrazu, takže je možné ho nastavit pouze jednou a dále ho neměnit. Po případné změně je pak potřeba ve stejné nabídce zvolit ještě možnost „Zaregistrovat“.

Následně jde navázat spojení pomocí výše popsaných metod.

5.4.2 KOS API

Pro přístup k datům přes REST API jsem využil framework Ktor, jehož součástí je HTTP client. Vytvořil jsem dva klienty, protože data z KOS API jsou ve formátu XML, zatímco data získaná přes Sirius API jsou ve formátu JSON. Pro formát JSON jsou ve frameworku Ktor k dipozici parsery již v základu. Pro formát XML jsem vybral knihovnu Jackson. Kvůli formátu KOS API jsem musel upravit způsob získávání hodnot. V rámci odpovědí existují možnosti, kde potřebuji získat z XML elementu jak hodnotu z atributu, tak i textový obsah. Jako příklad je název, který je uveden pro více jazyků a specifikum jazyka je pouze v atributu a hodnota poté pod tagem. Zadefinoval jsem proto novou vlastní property s názvem *innerText*.

Dalším specifikem je způsob získávání listu výsledků. Maximální množství položek, které jde získat pomocí jednoho dotazu, je tisíc. Dále je potřeba dát si pozor na to, že samotné stránkování nezaručuje unikátnost dat [20]. Je proto potřeba data seřadit, v dokumentaci je doporučeno použít pro tento účel id [20]. Abych tak dostal všechny položky, implementoval jsem pomocnou metodu *fetchFeeds*, která vykoná první dotaz a pokud v odpovědi dostane odkaz na další stránku, zavolá další dotaz, dokud nezíská všechny výsledky.

Pro přihlášení je použitý protokol OAuth 2.0 a fakultní autorizační server [21]. Pomocí parametrů *ClientId* a *ClientSecret* získám po zavolání url `https://auth.fit.cvut.cz/oauth/token` token, který má platnost hodinu. Abych zbytečně nezatěžoval autorizační server, token ukládám do mezipaměti a na nový se doptávám až když jeho platnost končí. Více konkrétních informací je popsáno v [22], případně v samotné specifikaci protokolu OAuth 2.0[13].

Dat v systému KOS je mnoho (počet paralelek je přes sto tisíc). Abych nestahoval všechna data, volám jednotlivé metody filtrující třeba paralelky pro každý předmět. Ve výsledku je to mnoho dotazů, kde každý trvá nějaký čas. Pokud pustím načtení všech časových lístků synchronně, v jednom vlákně, tak tato operace trvá více než minutu. To mi přišlo jako příliš dlouho, jelikož taková operace se bude volat poměrně často. Využil jsem tak jednoduchost práce s Coroutines jazyka Kotlin a operaci paralelizoval. Povedlo se mi zkrátit čas načítání pod deset sekund.

Po implementaci klientů a paralelizace jsem narazil na problém, při kterém aplikace začala padat na nedostatek popisovačů. Povedlo se mi zjistit, že jsem špatně použil HTTP klienta. Místo

[12] `https://github.com/jasperkrijgsman/com4j/commit/87f9c41c47a8a8a4647b92feeb81e5a3c6bd8b2f`
[13] `https://tools.ietf.org/html/rfc6749`

toho, abych používal pouze jeden sdílený, vytvářel jsem pro každé spojení nový. Otevřít klienta je ale náročná operace jak časově, tak paměťově. Jelikož při načítání dat vytvářím hodně dotazů na server, aplikace přesáhla limit počtu popisovačů systému Windows a následně spadla. Opravil jsem problém tak, že jsem změnil práci s klientem a jeho instanci si nově ukládám a nevytvářím pokaždé novou. Samotnou instanci pak explicitně neuzavírám, protože existuje maximálně jedna na běh programu a při jeho vypnutí se automaticky ukončí.

Pro zápis dat využívám textový soubor, popsaný v kapitole návrhu v sekci 4.5.

5.5 Keš

Všechna data, se kterými pracuji, načítám do stromových struktur. Tím myslím to, že každý objekt má své potomky a rodiče. Tato struktura je výhodná pro zobrazování a přímou práci s daty, avšak se nehodí pro odkazování se napříč strukturou a vyhledávání primárně podle unikátního identifikátoru. Takové vyhledávání pak potřebuji hlavně při porovnávání změn mezi jednotlivými modely.

Proto jsem přidal novou strukturu, kterou jsem si nazval keš. Nazval jsem jí tak proto, že práce s ní je založena na principu klíč-hodnota a data jsou uložena v hashovací tabulce. Jako klíč jsem zvolil unikátní identifikátor ze systému KOS, protože ten znám v obou systémech. Mohu se pak jednoduše pomocí znalosti jednoho objektu a jeho identifikátoru odkázat na ten samý objekt v druhém systému.

Pro tyto účely slouží rozhraní *SPlusCache* a *KOSAPICache*. Obsahují jednoduché metody pro přidání nebo odebrání objektu z dočasného úložiště. Jakmile si uživatelské rozhraní vyžádá potřebná data pro zobrazení, projdu je (to mohu jednoduše díky stromové struktuře) a uložím ještě navíc do keše.

5.5.1 Porovnání

Nyní k samotnému porovnání. Rozhodl jsem se, že při porovnání změn mě zajímají tři stavy. Těmi jsou:

- položky jsou stejné,
- položky jsou rozdílné (v jakémkoli parametru),
- položka chybí v druhém systému.

Pak jsem vytvořil pro každou položku vlastní extension metodu *compare*, která je volána na dané entitě a jako parametr přijímá keš opačného systému, se kterou by měla pracovat. V prvním kroku porovnání zjistím, zda se položka vůbec nachází v druhém systému. Pokud ne, vrátím stav, že v druhém systému chybí. Jinak pokračuji dále a porovnávám položku s jejím protějškem v druhém systému pomocí extension metody *isSame*.

Extension metoda *isSame* je implementována pro každý typ položky, který chci porovnávat. Její cíl je porovnat strukturu položek na úrovni dat, která nás zajímají. U všech položek je to unikátní identifikátor, u předmětu i třeba katedra, u několika dalších je to více informací. Nejvíce informací se pak porovnává u časového lístku. Kromě výše zmíněného identifikátoru porovnává název, přiřazený předmět, místnost, přiřazené učitele, rozvrh nebo kapacitu. Celá implementace je navržena tak, aby v případě úpravy porovnávacích parametrů (ať už přidání nebo odebrání) nebylo potřeba měnit velké kusy kódu a stačilo změnit jednu metodu.

Jakmile dojde k porovnání konkrétních dvou položek, tak se pokračuje dále následovně. Pokud nejsou shodné, vrátím rozdílný stav. Jinak pokračuji dále v kontrole.

Jelikož se jedná o data ve stromové struktuře, rozhodl jsem se, že budu zobrazovat nejenom rozdíl týkající se dané položky, ale i všech jejích potomků. Proto při porovnání změn pokračuji

kontrolou všech potomků a jejich rozdílů. Pokud se v rámci jejich změn nachází alespoň jeden, který není stejný, vrátím výsledek porovnání aktuální porovnávané položky jako rozdílný stav.

Pokud jsem nezjistil během jednotlivých kontrol rozdíl, kontrolu kompletně ukončím a prohlásím položky za shodné v obou systémech.

5.6 Uživatelské rozhraní

Uživatelské rozhraní je založeno na jedné hlavní obrazovce a třech pohledech, které se skládají ze sloupců. Pro zachování posledního načteného stavu je implementováno řešení, které ponechává všechny pohledy v paměti, i když nejsou zrovna viditelné. To zaručuje, že se uživatel může přepínat mezi jednotlivými pohledy bez ztráty vybraných dat nebo pozice ve scrollování.

V horní části se nachází hlavičky jednotlivých systémů, výběrová tlačítka pro specifické funkce (výběr semestru, nastavení spojení) a ukazatel průběhu načítání/aktualizace dat. Po načtení dat je místo ukazatele zobrazen čas poslední aktualizace a doba, kterou poslední aktualizace zabrala. Funkcionalita této lišty se nachází v komponentě *Toolbar*.

5.6.1 Data

Pro uchování dat je vytvořena třída *Database*, která agreguje data pro systém KOS a software Syllabus Plus. Data jsou načítána asynchronně pomocí pomocné třídy *Data*, která umožňuje načíst data při spuštění aplikace nebo manuálně při zavolání metody *load*. Kromě samotných dat třída uchovává i informace, zda se data načítají, jestli došlo k chybě, poslední čas stažení a doba trvání stažení.

5.6.2 Navigace

V rámci řešení přepínání pohledů jsem vytvořil třídu *Router*, která obsluhuje zobrazení. Obsahuje aktuální vybranou obrazovku a seznam všech definovaných obrazovek.

5.6.3 Notifikace

Pro notifikace (viz obrázek 5.4) (například při dokončení převodu, nebo při jakékoli chybě) jsem využil již předdefinované notifikace, které se nazývají *Snackbar*. Dále jsem vytvořil kanál, přes který se dají posílat jednotlivé zprávy z jakéhokoli místa. Hlavní okno pak poslouchá kanál a zaslané zprávy postupně ve stylu první přijde, první se zobrazí, obsluhuje. Notifikace se zobrazí na čtyři sekundy, uživatel má ale možnost ji zavřít i dříve.

Obrázek 5.4 Ukázka notifikace

5.6.4 Výběr semestru

Pro systém KOS je potřeba zvolit semestr. V levém horním rohu se uživateli načte při spuštění aplikace seznam posledních dvanácti semestrů plus dva semestry do budoucna (tyto počty jsou případně jednoduše nastavitelné v rámci kódu programu, ve třídě *Config*). Změna semestru je aplikovaná vždy na všechny akce týkající se systému KOS.

5.6.5 Připojení k softwaru Syllabus Plus

V rámci navázání spojení se v základu pro hodnotu „progId“ použije *Splus*.

Pokud si chce uživatel identifikátor změnit, tak po kliknutí v pravém horním menu na tlačítko s aktuálním identifikátorem se zobrazí okno, kde se dá zvolit jiný identifikátor. Nachází se v komponentě *ChangeProgIdDialog*. Po kliknutí na tlačítko „Změnit“ se nový identifikátor uloží a zkusí se vytvořit nové spojení.

V případě, kdy se spojení nepodaří vytvořit, ale identifikátor je nastaven správně, stačí dialog pro změnu pouze zobrazit a kliknout na tlačítko „Změnit“ bez změny identifikátoru.

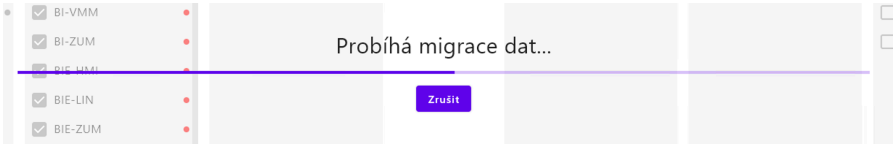
Jakmile je spojení úspěšně navázáno, tak se zobrazí v pravé části zelená tečka a informace o připojeném obrazu softwaru Syllabus Plus (viz obrázek 5.5). Těmi jsou název instituce/fakulty a název souboru obrazu pro případ, kdy název fakulty není nastaven (například „testscene.img“).

Obrázek 5.5 Navázané spojení se softwarem Syllabus Plus

V případě, kdy nastane konflikt a uživatel omylem zaregistruje více obrazů pod stejným identifikátorem, má možnost zjistit, ke které instanci softwaru Syllabus Plus je aplikace připojena. Po kliknutí na text „Syllabus Plus“ se uživateli otevře připojená instance softwaru Syllabus Plus. Pro lepší další používání však doporučuji změnit identifikátor druhé instance, protože neznám způsob, kterým se dá vybrat, která instance je zrovna zaregistrována v případě konfliktního indentifikátoru.

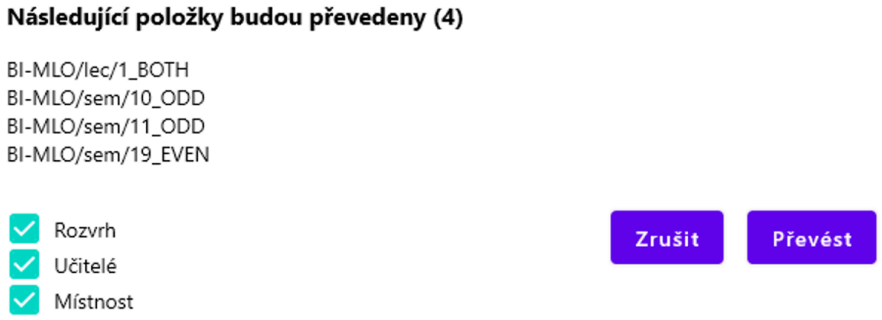
5.6.6 Převod dat

Převod dat je tvořen několika funkcionalitami. První je překryvná vrstva, která se zobrazí při spuštění jakéhokoliv převodu. Tato vrstva vypíná všechny aktivní prvky v rámci aplikace a zobrazuje postup převodu s možností přerušení (viz obrázek 5.6).



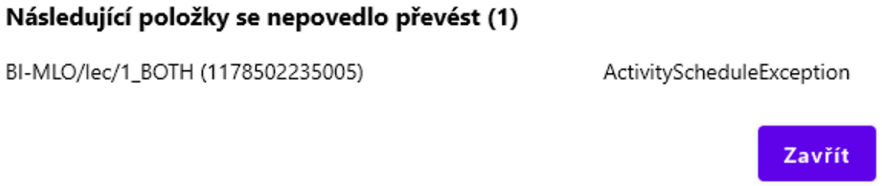
Obrázek 5.6 Překryvná vrstva při převodu dat

Dále jsem vytvořil společné dialogové okno (viz obrázek 5.7), které se zobrazí po zavolání metody *showMigrationConfirmDialog* pro všechny převody, a které zobrazuje seznam vybraných položek. V případě potřeby se dají definovat vybratelné části, které se zmigrují (tato funkcionalita je využita pouze pro převody časových lístků).



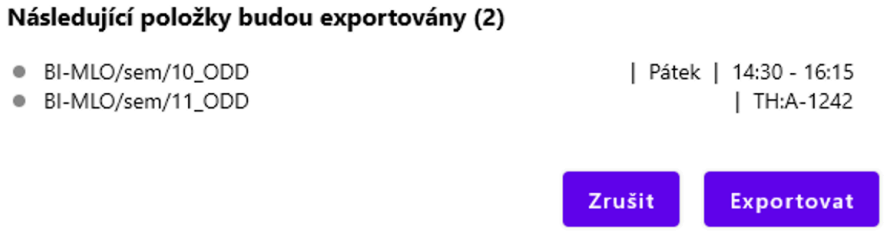
Obrázek 5.7 Potvrzující dialog pro časové lístky před převodem dat

Po zvolení začátku převodu se zavolá callback *onConfirm*, který zavolá patřičnou metodu. Definoval jsem metody a třídy pro převod časových lístků (*TimetableSlotsMigration*), učitelů (*StaffMembersMigration*), místností (*LocationsMigrations*) a potom pro export časových lístků ze softwaru Syllabus Plus (*KOSExportMigration*). Všechny třídy mají implementovanou spolupráci s postupem zobrazeným na překryvné vrstvě, možnost přerušení v průběhu převodu a seznam všech položek, které se nepodařilo převést. Po dokončení převodu se zobrazí dialog zobrazující seznam položek, u kterých převod selhal (viz obrázek 5.8). Tento dialog je opět stejný pro všechny typy převodů.



Obrázek 5.8 Dialog zobrazující položky, u kterých se nepovedl převod

V rámci exportu položek se pak zobrazí dialog upozorňující na rozdíly mezi daty pro export a aktuální verzí načtených dat ze systému KOS (viz obrázek 5.9).



Obrázek 5.9 Potvrzující dialog pro časové lístky před exportem

Dialog se zobrazí pomocí zavolání metody *showExportConfirmDialog*. Po odsouhlasení se zobrazí další dialog s výběrem lokace pro uložení souboru. Jelikož pro uložení souboru není dostupná komponenta v rámci knihovny Compose for Desktop, využil jsem dialogové okno z knihovny Swing.

V rámci tvorby souboru ale dochází ke dvěma změnám oproti datům v softwaru Syllabus Plus. První je, že formát souboru dovoluje pouze jednu místnost pro časový lístek. To ale není problém, protože časový lístek je vždy rozvrhován pouze do jedné místnosti. Druhá, a zásadnější,

je omezení pouze na dva učitele, protože existují časové lístky, pro které je potřeba mít přiřazeno více učitelů. V softwaru Syllabus Plus jich jde nastavit více, učitele tak musím oříznout. Všechny přiřazené učitele u časového lístku proto seřadím podle jejich unikátních identifikátorů vzestupně (aby při každém exportu bylo jejich pořadí stejné) a vezmu první dva, které pak propíši do výsledného souboru.

5.6.7 Pohledy

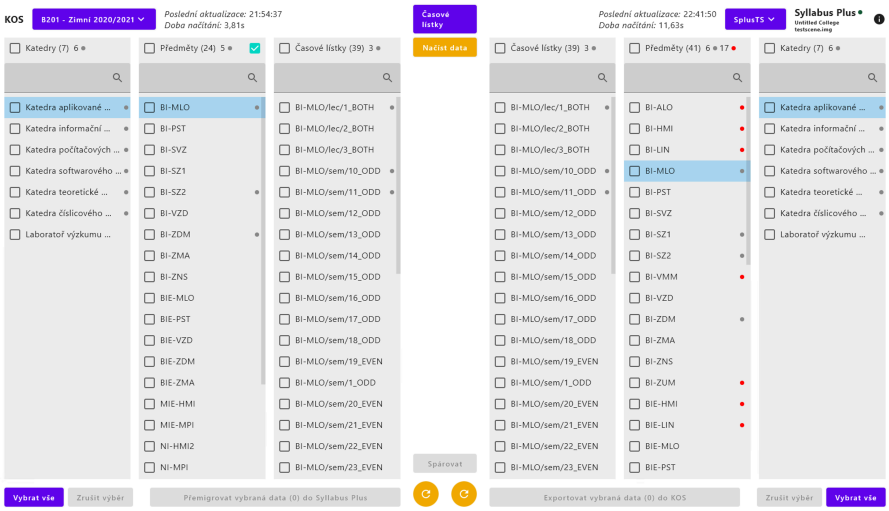
Všechny pohledy využívají komponentu *View*, která rozděluje okno na levou část, pravou část a střed. Samotný pohled pak po stranách používá sloupce a uprostřed jsou tlačítka pro načtení dat z obou systémů, z každého systému zvlášť a pro synchronizaci položek. Každý pohled může do této sekce přidat i své další komponenty.

V každé sekci se nachází alespoň dva sloupce, které jsou vedle sebe. Pro systém KOS pak obsahuje sekce ještě lištu s možnostmi pro převod dat do softwaru Syllabus Plus. V této liště se nachází tlačítko pro vybrání všech položek, zrušení výběru a pro samotné zahájení převodu. Sloupec je definován pomocí společné komponenty *DataColumn*. Každý sloupec má vyhledávání, označení jednotlivých, vybrání všech a zobrazení počtu položek. Dále se dá nastavit, že některé sloupce je možné sbalit, čímž je možné docílit odebrání nadřazené kategorie a zobrazení všech položek.

Funkce vyhledávání umí kromě textu viditelného pro uživatele prohledávat i další atributy jednotlivých položek. Těmi může být třeba u učitele kromě uživatelského jména i celé jméno a příjmení, u předmětu místo kódu jeho celý název.

5.6.7.1 Časové lístky

Zobrazení časových lístků se skládá ze tří sloupců (viz obrázek 5.10). Prvním sloupcem jsou katedry, druhým předměty a třetím časové lístky. Pokud uživatel vybere například katedru, v dalším sloupci se zobrazí předměty, které patří pod vybranou katedru. Dále když vybere předmět, zobrazí se jeho všechny časové lístky (v systému KOS přeskakuji paralelky, protože je pro práci nijak nepotřebuji).



Obrázek 5.10 Pohled časových lístků

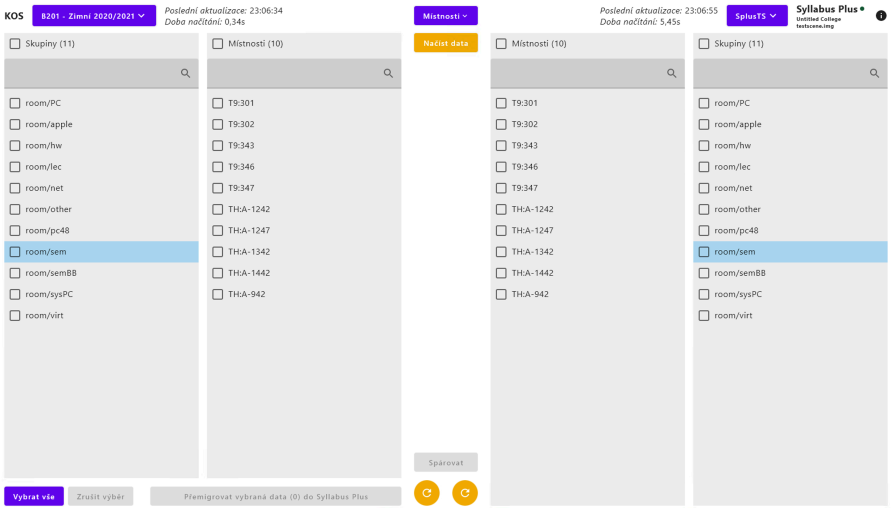
Protože ne vždy je šikovné zobrazovat předměty seskupené podle kateder, dá se tento sloupec sbalit. To znamená, že se zmenší jeho šířka a ve sloupci předmětů se zobrazí všechny, nezávisle

na katedře. Pro ostatní sloupce tato operace již udělat nejde, protože by zbytečně znepřehlednila práci s časovými lístky (bylo by jich příliš v jednom sloupci a uživatel by tak měl problémy s mnoha položkami pracovat).

Sekce časových lístků má ještě jednu specialitu oproti ostatním. Tou je lišta s možností exportu vybraných časových lístků ze softwaru Syllabus Plus do textového souboru. Lišta obsahuje opět tlačítka pro vybrání všech položek, zrušení výběru a zahájení exportu.

5.6.7.2 Místnosti

Zobrazení místností se skládá ze dvou sloupců (viz obrázek 5.11). Prvním jsou skupiny místností a druhým samotné místnosti. Skupiny místností jsou definované v textovém souboru, který je načten při spuštění aplikace. V rámci definic se přidá automaticky ještě jedna skupina, kterou jsem nazval „room/other“. Tato skupina obsahuje všechny ostatní místnosti, které nepatří do žádné jiné skupiny. Při vybrání skupiny se zobrazí místnosti pouze pro danou skupinu, nicméně uživatel opět může sloupec se skupinami sbalit. Poté se mu zobrazí seznam všech místností.



Obrázek 5.11 Pohled místností

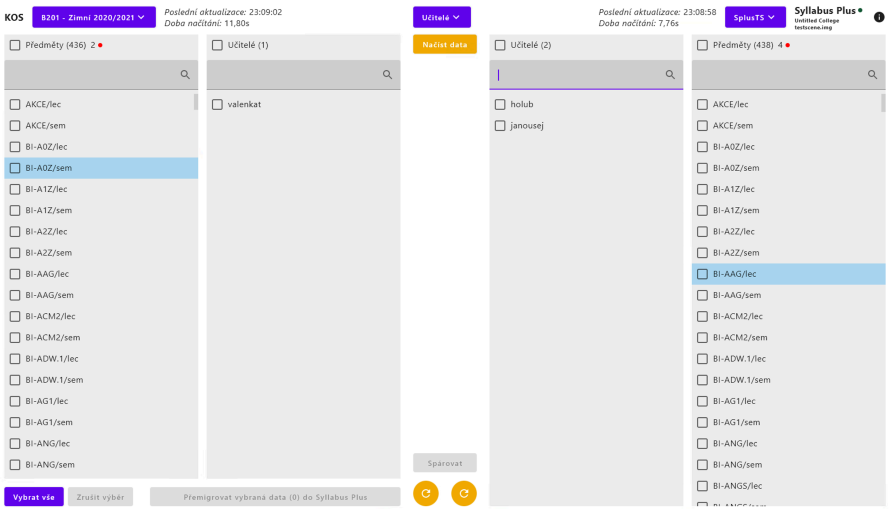
5.6.7.3 Učitelé

Zobrazení učitelů se skládá ze dvou sloupců (viz obrázek 5.12). Prvním jsou předměty a druhým učitelé. Předměty jsou ještě rozděleny na skupinu přednášejících a skupinu cvičících. Každý předmět se tak v seznamu nachází dvakrát. Pokud to uživatel bude potřebovat, může opět schovat sekci předmětů a zobrazí se mu pouze všichni uživatelé.

U tohoto pohledu je důležité podotknout, že nenačítám úplně všechny učitele na celém ČVUT. Jejich počet je příliš velký, proto načítám pouze ty učitele, kteří jsou přiřazení alespoň v jednom předmětu fakulty.

5.7 Známé problémy

Při vývoji jsem narazil na několik chyb, které se nachází v použitých knihovnách. První chybou je scrollování. Uživateli se vykreslí sloupec, nebo jiná komponenta, která by měla obsahovat scrollbar. Na základě obsahu ale žádný není potřeba. Scrollbar se tak nezobrazí, nicméně pokud uživatel najede myší do místa, kde by měl být, a klikne tam, tak celé uživatelské rozhraní zamrzne



Obrázek 5.12 Pohled učitelů

a nefungují již žádné další operace. Aplikaci je pak potřeba restartovat. Podařilo se mi najít tento problém nahlášený jako oficiální chybu ve frameworku Compose for Desktop pod číslem 304[14].

Další chyba, na kterou jsem narazil, souvisí s komunikací s COM rozhraním. Během převodu dat se mi občas stalo, že běh celé aplikace selhal na neodchytitelnou chybu paměťové ochrany. Dle dokumentace mnou použitá knihovna com4j využívá části kódu napsané v programovacím jazyce C [23]. Tyto části pak komunikují přes COM rozhraní. Pravděpodobně v rámci komunikace dojde ke kritické chybě a to má za následek kompletní pád aplikace. Chybu se mi však nepodařilo nikdy konzistentně nasimulovat, stává se velmi zřídka. Když nastane, tak stačí uživateli aplikaci pouze restartovat a pustit převod znovu. Nemá to ale žádný vliv na konzistentnost dat. Pád převod jen přeruší, všechna data do té doby převedená zůstanou a pouze se nepokračuje s dalšími položkami. Nevzniknou proto žádná nevalidní data a případně může uživatel kdykoli daný převod spustit znovu (vzhledem k propojení systémů se již převedená data pouze aktualizují, nevzniknou žádné duplicitní záznamy).

5.8 Licence

Pro vývoj jsem použil knihovny, které využívají licenci *BSD 2-Clause "Simplified" License*[15] nebo *Apache 2.0*[16]. Seznam knihoven a jejich licencí je zachycen v tabulce 5.3. Jejich práva a omezení potom zobrazuje tabulka 5.2.

Obě licence umožňují jak soukromé, tak komerční využití, včetně distribuce a případných modifikací. Jejich jediným požadavkem je zachování upozornění na licenci a autorská práva v rámci samotných knihoven. Použité knihovny včetně jejich licencí jdou zobrazit v samotné aplikaci pod položkou „O aplikaci“ v levé horní nabídce.

Protože jsem požadavky licencí splnil a zároveň mi jejich omezení umožňují bezplatné využití bez nutnosti zveřejnění zdrojových kódů nebo zachování licencí, mohu knihovny pro mou práci využít.

[14] `https://github.com/JetBrains/compose-jb/issues/304`
[15] `https://choosealicense.com/licenses/bsd-2-clause/`
[16] `https://choosealicense.com/licenses/apache-2.0/`

	BSD 2-Clause "Simplified" License	Apache 2.0
Povoluje		
Komerční použití	✓	✓
Modifikace	✓	✓
Distribuce	✓	✓
Použití pro patent	x	✓
Soukromé použití	✓	✓
Omezuje		
Používání ochranné známky	✓	x
Odpovědnost	✓	✓
Záruka	✓	✓

Tabulka 5.2 Přehled licencí a jejich omezení

Knihovna	**Licence**
com4j	BSD 2-Clause "Simplified" License[17]
com4j – oprava chyb a využití try-with-resources	BSD 2-Clause "Simplified" License[18]
com4j – úprava pro podporu referencí	BSD 2-Clause "Simplified" License[19]
Kotlin	Apache 2.0[20]
Kotlin Coroutines	Apache 2.0[21]
KotlinX DateTime	Apache 2.0[22]
Compose for Desktop	Apache 2.0[23]
Koin	Apache 2.0[24]
Ktor, HTTP Client, JSON	Apache 2.0[25]
Jackson (XML)	Apache 2.0[26]
Gradle	Apache 2.0[27]
Dokka – generátor dokumentace	Apache 2.0[28]

Tabulka 5.3 Seznam knihoven a jejich licencí

[17] https://github.com/kohsuke/com4j/blob/master/LICENSE.txt
[18] https://github.com/archiecobbs/com4j/blob/master/LICENSE.txt
[19] https://github.com/jasperkrijgsman/com4j/blob/master/LICENSE.txt
[20] https://github.com/JetBrains/kotlin/blob/master/license/README.md
[21] https://github.com/Kotlin/kotlinx.coroutines/blob/master/LICENSE.txt
[22] https://github.com/Kotlin/kotlinx-datetime/blob/master/LICENSE.txt
[23] https://github.com/JetBrains/compose-jb/blob/master/LICENSE.txt
[24] https://github.com/InsertKoinIO/koin/blob/master/LICENSE
[25] https://github.com/ktorio/ktor/blob/main/LICENSE
[26] https://github.com/FasterXML/jackson-dataformat-xml/blob/master/LICENSE
[27] https://github.com/gradle/gradle/blob/master/LICENSE
[28] https://github.com/Kotlin/dokka/blob/master/LICENSE

Kapitola 6
Testování

6.1 Manuální scénáře

Pro otestování funkcionality aplikace jsem sestavil scénáře, které pokrývají všechny případy užití. Součástí každého scénáře je

- počáteční stav,
- kroky,
- pokryté případy užití.

V jednotlivých krocích pak popisuji, jak dosáhnout výsledku definovaného v názvu testovacího scénáře.

6.1.1 T1 – Zobrazení okna pro inicializaci softwaru Syllabus Plus

Počáteční stav: Aplikace je spuštěná
Kroky:

1. V levém horním rohu kliknout na tlačítko s názvem aktuálně vybraného semestru
2. Vybrat semestr, pro který má inicializace probíhat – semestr se nastaví
3. V pravém horním rohu kliknout vedle nápisu „Syllabus Plus“ na ikonku s písmenem *i* – zobrazí se dialogové okno s předvyplněnými políčky pro inicializaci softwaru Syllabus Plus

Pokryté případy užití: UC1

6.1.2 T2 – Převod všech časových lístků do softwaru Syllabus Plus

Počáteční stav: Pohled s časovými lístky
Kroky:

1. V levém horním rohu kliknout na tlačítko s názvem aktuálně vybraného semestru
2. Vybrat semestr, ze kterého se mají získávat data – semestr se nastaví

3. V pravém horním rohu kliknout na fialové tlačítko s šipkou dolů – zobrazí se dialog
4. Do textového pole vyplnit nastavený identifikátor spojení aplikace Syllabus Plus
5. Kliknout na tlačítko změnit – úspěšné navázání spojení identifikováno zobrazením zelené tečky vedle nápisu „Syllabus Plus“
6. Kliknout na tlačítko „Načíst data“ – zobrazí se načítání
7. V levé části kliknout na tlačítko „Vybrat vše“ – vyberou se všechny položky, umožní se kliknutí na tlačítko „Přemigrovat vybraná data“
8. Kliknout na tlačítko „Přemigrovat vybraná data“ – zobrazí se dialog s potvrzením
9. Kliknout na tlačítko „Převést“ – zobrazí se průběh migrace dat
10. Po chvíli se zobrazí úspěšná notifikace nebo dialogové okno s položkami, které se nepodařilo převést
11. Po dokončení se spustí automaticky aktualizace dat ze softwaru Syllabus Plus

Pokryté případy užití: UC2, UC3

6.1.3 T3 – Převod vybraných časových lístků do softwaru Syllabus Plus

Počáteční stav: Pohled s časovými lístky
Kroky:

1. V levém horním rohu kliknout na tlačítko s názvem aktuálně vybraného semestru
2. Vybrat semestr, ze kterého se mají získávat data – semestr se nastaví
3. V pravém horním rohu kliknout na fialové tlačítko s šipkou dolů – zobrazí se dialog
4. Do textového pole vyplnit nastavený identifikátor spojení aplikace Syllabus Plus
5. Kliknout na tlačítko změnit – úspěšné navázání spojení identifikováno zobrazením zelené tečky vedle nápisu „Syllabus Plus“
6. Kliknout na tlačítko „Načíst data“ – zobrazí se načítání
7. V levé části označit položky k převodu kliknutím na checkbox – položky se vyberou a umožní se kliknutí na tlačítko „Přemigrovat vybraná data“
8. Kliknout na tlačítko „Přemigrovat vybraná data“ – zobrazí se dialog s potvrzením
9. Kliknout na tlačítko „Převést“ – zobrazí se průběh migrace dat
10. Po chvíli se zobrazí úspěšná notifikace nebo dialogové okno s položkami, které se nepodařilo převést
11. Po dokončení se spustí automaticky aktualizace dat ze softwaru Syllabus Plus

Pokryté případy užití: UC2

6.1.4 T4 – Porovnání změn mezi systémy – lišící se vlastnosti

Tento testovací scénář můžeme provést pro jakoukoli vlastnost a jakoukoli položku, včetně druhého směru s úpravou položky v systému KOS. Zde je popsán jako ukázka pro rozdílného učitele u časového lístku.
Počáteční stav: Pohled s časovými lístky
Kroky:

1. V levém horním rohu kliknout na tlačítko s názvem aktuálně vybraného semestru
2. Vybrat semestr, ze kterého se mají získávat data – semestr se nastaví
3. V pravém horním rohu kliknout na fialové tlačítko s šipkou dolů – zobrazí se dialog
4. Do textového pole vyplnit nastavený identifikátor spojení aplikace Syllabus Plus
5. Kliknout na tlačítko změnit – úspěšné navázání spojení identifikováno zobrazením zelené tečky vedle nápisu „Syllabus Plus“
6. Kliknout na tlačítko „Načíst data“ – zobrazí se načítání
7. Dohledat si položku, která se nachází v obou systémech
8. Najít danou položku v systému Syllabus Plus a změnit její přiřazené učitele
9. V dolní části kliknou na aktualizaci dat softwaru Syllabus Plus – zobrazí se načítání
10. Po dokončení načítání se zobrazí u nalezené položky šedá tečka, která signalizuje rozdíl

Pokryté případy užití: UC8, UC13

6.1.5 T5 – Porovnání změn mezi systémy – chybějící položka

Tento testovací scénář můžeme provést pro jakoukoli položku, včetně druhého směru s chybějící položkou v systému KOS. Zde je popsán jako ukázka pro chybějící časový lístek.
Počáteční stav: Pohled s časovými lístky
Kroky:

1. V levém horním rohu kliknout na tlačítko s názvem aktuálně vybraného semestru
2. Vybrat semestr, ze kterého se mají získávat data – semestr se nastaví
3. V pravém horním rohu kliknout na fialové tlačítko s šipkou dolů – zobrazí se dialog
4. Do textového pole vyplnit nastavený identifikátor spojení aplikace Syllabus Plus
5. Kliknout na tlačítko změnit – úspěšné navázání spojení identifikováno zobrazením zelené tečky vedle nápisu „Syllabus Plus“
6. Kliknout na tlačítko „Načíst data“ – zobrazí se načítání
7. Dohledat si položku, která se nachází v obou systémech
8. Najít danou položku v systému Syllabus Plus a smazat ji
9. V dolní části kliknou na aktualizaci dat softwaru Syllabus Plus – zobrazí se načítání
10. Po dokončení načítání se zobrazí i zmizí položka z pravé části a u položky v levé části se zobrazí červená tečka signalizující chybějící položku v druhém systému

Pokryté případy užití: UC11, UC12

6.1.6 T6 – Export všech časových lístků ze softwaru Syllabus Plus

Počáteční stav: Pohled s časovými lístky
Kroky:

1. V levém horním rohu kliknout na tlačítko s názvem aktuálně vybraného semestru
2. Vybrat semestr, ze kterého se mají získávat data – semestr se nastaví
3. V pravém horním rohu kliknout na fialové tlačítko s šipkou dolů – zobrazí se dialog
4. Do textového pole vyplnit nastavený identifikátor spojení aplikace Syllabus Plus
5. Kliknout na tlačítko změnit – úspěšné navázání spojení identifikováno zobrazením zelené tečky vedle nápisu „Syllabus Plus“
6. Kliknout na tlačítko „Načíst data“ – zobrazí se načítání
7. V pravé části kliknout na tlačítko „Vybrat vše“ – vyberou se všechny časové lístky, umožní se kliknutí na tlačítko „Exportovat vybraná data“
8. Kliknout na tlačítko „Exportovat vybraná data“ – zobrazí se dialog s potvrzením
9. Kliknout na tlačítko „Exportovat“ – zobrazí se dialogové okno s výběrem místa pro uložení souboru
10. Vybrat místo a název souboru
11. Kliknout na tlačítko „Save“ – zobrazí se průběh migrace dat
12. Po chvíli se zobrazí úspěšná notifikace nebo dialogové okno s položkami, které se nepodařilo exportovat

Pokryté případy užití: UC6, UC7, UC10

6.1.7 T7 – Export vybraných časových lístků ze softwaru Syllabus Plus

Počáteční stav: Pohled s časovými lístky
Kroky:

1. V levém horním rohu kliknout na tlačítko s názvem aktuálně vybraného semestru
2. Vybrat semestr, ze kterého se mají získávat data – semestr se nastaví
3. V pravém horním rohu kliknout na fialové tlačítko s šipkou dolů – zobrazí se dialog
4. Do textového pole vyplnit nastavený identifikátor spojení aplikace Syllabus Plus
5. Kliknout na tlačítko změnit – úspěšné navázání spojení identifikováno zobrazením zelené tečky vedle nápisu „Syllabus Plus“
6. Kliknout na tlačítko „Načíst data“ – zobrazí se načítání
7. V pravé části označit položky k převodu kliknutím na checkbox – položky se vyberou a umožní se kliknutí na tlačítko „Exportovat vybraná data“
8. Kliknout na tlačítko „Exportovat vybraná data“ – zobrazí se dialog s potvrzením

9. Kliknout na tlačítko „Exportovat“ – zobrazí se dialogové okno s výběrem místa pro uložení souboru
10. Vybrat místo a název souboru
11. Kliknout na tlačítko „Save“ – zobrazí se průběh migrace dat
12. Po chvíli se zobrazí úspěšná notifikace nebo dialogové okno s položkami, které se nepodařilo exportovat

Pokryté případy užití: UC6, UC10, UC14

6.1.8 T8 – Spárování položek mezi systémy

Počáteční stav: Pohled s časovými lístky, nebo místnostmi, nebo učiteli a načtenými daty
Kroky:

1. V levé části vybrat jednu položku
2. V pravé části vybrat jednu položku – tlačítko „Spárovat“ se stane aktivním
3. Kliknout na tlačítko „Spárovat“ – spustí se aktualizace dat systému Syllabus Plus, v případě chyby se zobrazí notifikace

Pokryté případy užití: UC9

6.1.9 T9 – Převod všech místností do softwaru Syllabus Plus

Počáteční stav: Pohled s časovými lístky
Kroky:

1. V levém horním rohu kliknout na tlačítko s názvem aktuálně vybraného semestru
2. Vybrat semestr, ze kterého se mají získávat data – semestr se nastaví
3. V pravém horním rohu kliknout na fialové tlačítko s šipkou dolů – zobrazí se dialog
4. Do textového pole vyplnit nastavený identifikátor spojení aplikace Syllabus Plus
5. Kliknout na tlačítko změnit – úspěšné navázání spojení identifikováno zobrazením zelené tečky vedle nápisu „Syllabus Plus“
6. Kliknout na rozbalovací nabídku a změnit pohled na místnosti – zobrazí se pohled místností
7. Kliknout na tlačítko „Načíst data“ – zobrazí se načítání
8. V levé části kliknout na tlačítko „Vybrat vše“ – vyberou se všechny položky, umožní se kliknutí na tlačítko „Přemigrovat vybraná data“
9. Kliknout na tlačítko „Přemigrovat vybraná data“ – zobrazí se dialog s potvrzením
10. Kliknout na tlačítko „Převést“ – zobrazí se průběh migrace dat
11. Po chvíli se zobrazí úspěšná notifikace nebo dialogové okno s položkami, které se nepodařilo převést
12. Po dokončení se spustí automaticky aktualizace dat ze softwaru Syllabus Plus

Pokryté případy užití: UC2, UC5

6.1.10 T10 – Převod vybraných místností do softwaru Syllabus Plus

Počáteční stav: Pohled s časovými lístky
Kroky:

1. V levém horním rohu kliknout na tlačítko s názvem aktuálně vybraného semestru
2. Vybrat semestr, ze kterého se mají získávat data – semestr se nastaví
3. V pravém horním rohu kliknout na fialové tlačítko s šipkou dolů – zobrazí se dialog
4. Do textového pole vyplnit nastavený identifikátor spojení aplikace Syllabus Plus
5. Kliknout na tlačítko změnit – úspěšné navázání spojení identifikováno zobrazením zelené tečky vedle nápisu „Syllabus Plus“
6. Kliknout na rozbalovací nabídku a změnit pohled na místnosti – zobrazí se pohled místností
7. Kliknout na tlačítko „Načíst data“ – zobrazí se načítání
8. V levé části označit položky k převodu kliknutím na checkbox – položky se vyberou a umožní se kliknutí na tlačítko „Přemigrovat vybraná data“
9. Kliknout na tlačítko „Přemigrovat vybraná data“ – zobrazí se dialog s potvrzením
10. Kliknout na tlačítko „Převést“ – zobrazí se průběh migrace dat
11. Po chvíli se zobrazí úspěšná notifikace nebo dialogové okno s položkami, které se nepodařilo převést
12. Po dokončení se spustí automaticky aktualizace dat ze softwaru Syllabus Plus

Pokryté případy užití: UC2

6.1.11 T11 – Převod všech učitelů do softwaru Syllabus Plus

Počáteční stav: Pohled s časovými lístky
Kroky:

1. V levém horním rohu kliknout na tlačítko s názvem aktuálně vybraného semestru
2. Vybrat semestr, ze kterého se mají získávat data – semestr se nastaví
3. V pravém horním rohu kliknout na fialové tlačítko s šipkou dolů – zobrazí se dialog
4. Do textového pole vyplnit nastavený identifikátor spojení aplikace Syllabus Plus
5. Kliknout na tlačítko změnit – úspěšné navázání spojení identifikováno zobrazením zelené tečky vedle nápisu „Syllabus Plus“
6. Kliknout na rozbalovací nabídku a změnit pohled na učitele – zobrazí se pohled učitelů
7. Kliknout na tlačítko „Načíst data“ – zobrazí se načítání
8. V levé části kliknout na tlačítko „Vybrat vše“ – vyberou se všechny položky, umožní se kliknutí na tlačítko „Přemigrovat vybraná data“

9. Kliknout na tlačítko „Přemigrovat vybraná data“ – zobrazí se dialog s potvrzením
10. Kliknout na tlačítko „Převést“ – zobrazí se průběh migrace dat
11. Po chvíli se zobrazí úspěšná notifikace nebo dialogové okno s položkami, které se nepodařilo převést
12. Po dokončení se spustí automaticky aktualizace dat ze softwaru Syllabus Plus

Pokryté případy užití: UC2, UC4

6.1.12 T12 – Převod vybraných učitelů do softwaru Syllabus Plus

Počáteční stav: Pohled s časovými lístky
Kroky:

1. V levém horním rohu kliknout na tlačítko s názvem aktuálně vybraného semestru
2. Vybrat semestr, ze kterého se mají získávat data – semestr se nastaví
3. V pravém horním rohu kliknout na fialové tlačítko s šipkou dolů – zobrazí se dialog
4. Do textového pole vyplnit nastavený identifikátor spojení aplikace Syllabus Plus
5. Kliknout na tlačítko změnit – úspěšné navázání spojení identifikováno zobrazením zelené tečky vedle nápisu „Syllabus Plus“
6. Kliknout na rozbalovací nabídku a změnit pohled na učitele – zobrazí se pohled učitelů
7. Kliknout na tlačítko „Načíst data“ – zobrazí se načítání
8. V levé části označit položky k převodu kliknutím na checkbox – položky se vyberou a umožní se kliknutí na tlačítko „Přemigrovat vybraná data“
9. Kliknout na tlačítko „Přemigrovat vybraná data“ – zobrazí se dialog s potvrzením
10. Kliknout na tlačítko „Převést“ – zobrazí se průběh migrace dat
11. Po chvíli se zobrazí úspěšná notifikace nebo dialogové okno s položkami, které se nepodařilo převést
12. Po dokončení se spustí automaticky aktualizace dat ze softwaru Syllabus Plus

Pokryté případy užití: UC2

6.1.13 T13 – Definování skupin místností

Počáteční stav: Pohled s časovými lístky
Kroky:

1. V levém horním menu vybrat položku *Aplikace* – zobrazí se podmenu
2. V podmenu zvolit položku *Nastavení* – otevře se okno prohlížeče souborů
3. Otevřít soubor *rooms.csv* a provést patřičné úpravy, soubor uložit
4. Aplikaci restartovat
5. Po restartování se nové skupiny místností automaticky načtou, lze ověřit postupem podle scénáře T10

Pokryté případy užití: UC15

6.2 Uživatelské testování

Jako cíl testování jsem zvolil zjištění intuitivnosti a přehlednosti používání aplikace, funkcionalitu aplikace v prostředí, ve kterém bude software používaný, a případné odhalení chyb. Jako uživatele pro testování jsem zvolil aktuálního fakultního rozvrháře Daniela Dombka (dále jen „tester“). Jedná se o jediného uživatele, který by s touto aplikací měl po dokončení pracovat, má nejbližší vztah k tvorbě rozvrhu a rozumí jeho problémům a požadavkům.

Pro uživatelské testování jsem vytvořil následující scénáře:

- Instalace programu
- Inicializace obrazu a navázání spojení se softwarem Syllabus Plus
- Převod všech časových lístků do softwaru Syllabus Plus
- Export všech časových lístků ze softwaru Syllabus Plus
- Oprava informací časového lístku ze systému KOS
- Oprava smazaného náhodného časového lístku v softwaru Syllabus Plus
- Vyhledání chybějícího učitele a jeho následný import do softwaru Syllabus Plus

Jedná se o jiné scénáře než pro otestování funkčnosti aplikace a zároveň nepokrývají celou funkcionalitu, protože cílem uživatelského testování je primárně intuitivnost a přehlednost aplikace. (Například z pohledu používání není potřeba testovat import všech dat pro jednotlivé pohledy, protože se vždy jedná o tu samou posloupnost kroků.)

Protože tester je dostatečně seznámen s problematikou tvorby rozvrhu, nemusel jsem vysvětlovat smysl a účel aplikace.

6.2.1 Instalace programu

Testerovi byl zaslán instalační soubor s aplikací a údaje k nastavení spojení s KOS API.

Poté byly zadány instrukce:

1. Spusťte instalační soubor
2. Projděte pomocníkem pro instalaci
3. Spusťte aplikaci
4. V levém horním menu zvolte „Nastavení“
5. Nastavte v souboru „settings.properties“ zaslané údaje
6. Restartujte aplikaci

Tester aplikaci pomocí těchto pokynů nainstaloval a nastavil spojení s KOS API. Během scénáře se nevyskytly žádné chyby ani nejasnosti.

6.2.2 Inicializace obrazu a navázání spojení se softwarem Syllabus Plus

Testerovi byly zadány následující instrukce:

1. Začněte vytvářet nový obraz softwaru Syllabus Plus a zastavte se na dialogu s nastavením instituce

2. Přejděte do aplikace pro převod a zobrazte informace o tom, jak nastavení instituce vyplnit *Tester by měl v pravém horním rohu kliknout na ikonku vedle textu Syllabus Plus*
3. Podle získaných informací vyplňte dialog a dokončete inicializaci obrazu
4. V softwaru Syllabus Plus zvolte položku „Com“ a pod ní položku „ProgID...“
5. Ponechte identifikátor a klikněte na tlačítko „Change“
6. Ve stejné nabídce „Com“ zvolte položku „Register Server (User)“
7. Spusťe aplikaci pro převod
8. V pravém horním rohu klikněte na rozbalovací nabídku a nastavte stejný identifikátor, jako v softwaru Syllabus Plus

Tester postupoval v rámci všech kroků podle pokynů bez problému, kromě zobrazení informací o nastavení obrazu. Nejdříve hledal tlačítko pro zobrazení dialogového okna v hlavním menu, nicméně po několika jednotkách sekund našel ikonu vedle nápisu Syllabus Plus.

V rámci nastavení bylo matoucí, který semestr se nastavuje, proto jsem musel poradit, jak semestr správně vybrat a případně změnit mimo dialogové okno.

6.2.3 Převod všech časových lístků do softwaru Syllabus Plus

Testerovi byla zadána situace, kdy začíná tvorba rozvrhu, a je potřeba převést všechny časové lístky do softwaru Syllabus Plus.

Ideální průchod by se měl skládat z těchto kroků:

1. Načtení všech dat ze systémů nebo aktualizace dat ze systému KOS
2. Výběr všech časových lístků v části systému KOS
3. Kliknutí na tlačítko pro migraci a následné potvrzení

Tester přišel na to, že je potřeba nejdříve načíst data. Poté ale místo výběru všech položek vybral všechny katedry a ty převedl. Nějakou chvíli byl zmatený, co znamená výběr všech položek, proto jsem musel objasnit, že původně převedl pouze katedry. Následně pak proběhl převod bez dalších problémů a asistence.

Vzhledem k vybranému semestru bylo zobrazeno několik časových lístků, které se nedařilo převést, nicméně bylo rozvrhářem objasněno, že se jedná o položky, které jsou vědomě plně kolizní s jinými časovými lístky.

6.2.4 Export všech časových lístků ze softwaru Syllabus Plus

Testerovi byla zadána situace, kdy je dokončena tvorba rozvrhu, a je potřeba exportovat všechny časové lístky ze softwaru Syllabus Plus.

Ideální průchod by se měl skládat z těchto kroků:

1. Načtení všech dat ze systémů nebo aktualizace dat ze softwaru Syllabus Plus
2. Výběr všech časových lístků v části softwaru Syllabus Plus
3. Kliknutí na tlačítko pro export a potvrzení položek

4. Vybrání cesty pro uložení souboru a zahájení exportu

Tester úspěšně aktualizoval data. Poté byl opět zmatený výběrem položek, nicméně po krátkém vysvětlení časové lístky správně označil. Prošel si v zobrazeném dialogu položky a zkontroloval, které jednotlivé údaje se změní. Opět v rámci vybraného semestru se mu zobrazily časové lístky, které nemají rozvrh, nicméně tento případ jsme již řešili, takže nebyl překvapením. Export proběhl v pořádku a soubor na první pohled obsahoval všechna data.

6.2.5 Oprava informací časového lístku ze systému KOS

Nejdříve jsem navedl na provedení změny (změna naplánování časového lístku) v softwaru Syllabus Plus a aktualizaci dat (cílem je nasimulovat změnu v systému KOS, proto je aktualizace dat součástí přípravy). Poté bylo zadáno, že konkrétní časový lístek se změnil a je potřeba zjistit, v čem je rozdíl, a následně provést opravu.

Ideální průchod by se měl skládat z těchto kroků:

1. Dohledání časového lístku v obou systémech
2. Zobrazení dialogového okna s detailem časového lístku a zjištění rozdílu
3. Označení časového lístku a provedení migrace

Tester dohledal daný časový lístek a zjistil, v čem se přesně liší. Při otevření dialogového okna s detailem časového lístku ale nenašel na první pohled, ze kterého systému jsou informace zobrazeny – protože ale věděl, které okno kdy otevřel, tak to nebyl problém. Následně provedl aktualizaci pomocí převodu (akci převodu jsem musel poradit, pravděpodobně proto, že do této chvíle nebyla podobná akce vůbec možná). Časový lístek se úspěšně opravil do stavu ze systému KOS.

6.2.6 Oprava smazaného náhodného časového lístku v softwaru Syllabus Plus

Nejdříve jsem navedl na provedení náhodného smazání časového lístku ze softwaru Syllabus Plus. Zadal jsem instrukci, že si tester omylem smazal časový lístek a je potřeba ho dohledat a opravit. Pro rychlejší dokončení scénáře jsem dovolil pracovat s informací, že se jednalo o lístek z předmětu BI-PAI.

Ideální průchod by se měl skládat z těchto kroků:

1. Načtení všech dat ze systémů nebo aktualizace dat ze softwaru Syllabus Plus
2. Dohledání časového lístku v části systému KOS pomocí červené tečky
3. Označení časového lístku a provedení migrace

Po chvíli rozvrhář bez problému zjistil, že u předmětu BI-PAI se v systému KOS nachází šedá tečka znázorňující rozdíl a po rozkliknutí předmětu našel i červenou tečku znázorňující chybějící položku v druhém systému. Po nalezení úspěšně provedl převod jednoho lístku opět bez komplikací.

6.2.7 Vyhledání chybějícího učitele a jeho následný import do softwaru Syllabus Plus

Nejdříve jsem navedl na provedení kompletního importu učitelů a poté smazání učitele „dombedan“ ze softwaru Syllabus Plus a aktualizaci dat (snažím se nasimulovat nového učitele v systému

KOS, proto je aktualizace dat součástí přípravy). Následně byla zadána situace, kdy do systému KOS přibyl nový učitel „dombedan“ a je potřeba ho importovat do softwaru Syllabus Plus.

Ideální průchod by se měl skládat z těchto kroků:

1. Sbalení sloupce s kategoriemi učitelů
2. Dohledání učitele „dombedan“ pomocí filtru
3. Označení učitele a provedení migrace

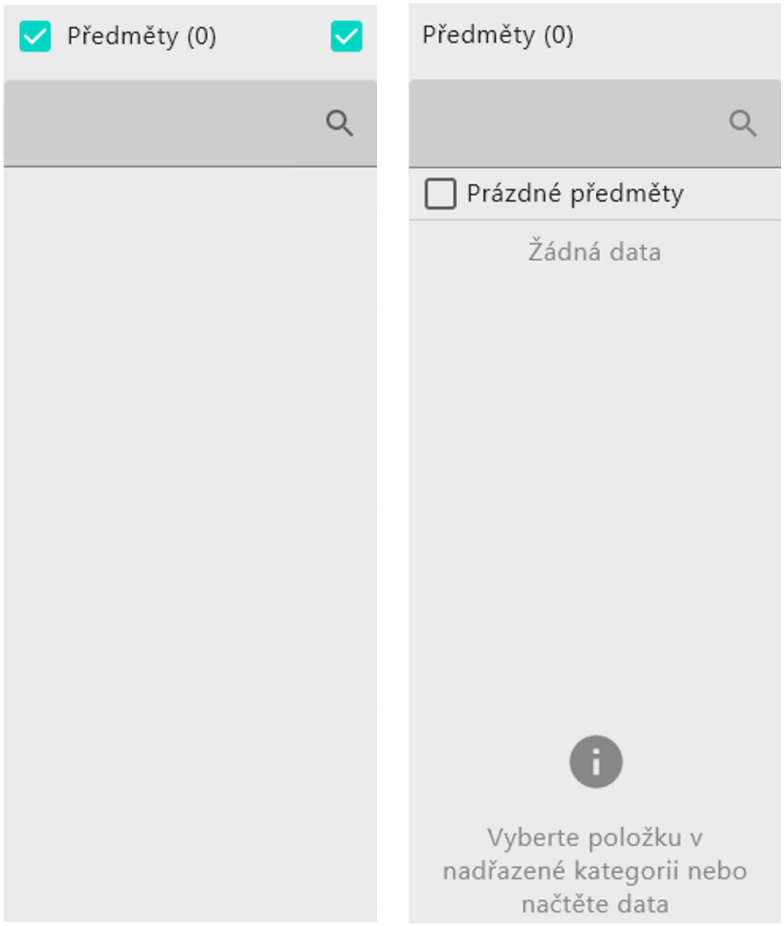
Tester byl zmatený při vyhledání učitele bez vybrané kategorie, kdy se pokusil vyhledat učitele v rámci prázdného sloupce. Nebylo zřejmé, zda aplikace funguje korektně, nebo proč se nic nemění. Pro vyhledání učitele jsem musel poradit, jak sbalit sloupec a zobrazit všechny učitele nezávisle na skupině. Poté proběhlo již vše v pořádku a učitel byl správně převeden do systému Syllabus Plus.

6.2.8 Zjištěné problémy a jejich řešení

- Pro vyřešení problému s nepřehledností vybraného semestru jsem do dialogového okna s informacemi o inicializaci semestru přidal stejný výběr semestru, jako v hlavním okně aplikace.
- Při převodu časových lístků (včetně rozvrhu) nepovoluji vznik kolizí, i když to v softwaru Syllabus Plus lze. Nicméně bylo mi řečeno, že je lepší, pokud se vyhodí výjimka (tak jako nyní), než aby program ignoroval omezení a potom vznikly nechtěné kolize.
- Tlačítka pro aktualizaci dat jednotlivých systémů byla hledána spíše v horní části programu, proto jsem je přesunul pod tlačítko s načtením všech dat.
- Pro zpřehlednění původu časového lístku jsem přidal do dialogového okna další text s informací o systému (nenachází se nadále pouze v hlavičce).
- Během testování bylo lehké zmatení mezi označenými položkami pro převod a otevřenými položkami pro zobrazení ve sloupcích. Dle zpětné vazby není jasné, kolik položek je označených. Jako řešení jsem navrhl přesunout informaci o počtu vybraných položek mimo tlačítko pro migraci. Změna je zachycena na obrázku 6.2.
- Během testování vyhledání učitele bylo zjištěno zmatení s prázdným sloupcem s učiteli při nevybrání kategorie. Navrhl jsem proto řešení, které v případě nevybrané kategorie sloupec znepřístupní a zobrazí text s instrukcí pro jeho aktivaci (pokyn k vybrání položky v nadřazené kategorii). Změna je zachycena na obrázku 6.1.
- Akce sbalení sloupce není jednoduše dohledatelná, nicméně při nahodilém klikání na ni přišel. Jako řešení jsem navrhl podtrhávání textu názvu sloupce při najetí.

Další poznámky, které se netýkaly přímo testovacích scénářů, ale v rámci práce na ně bylo poukázáno:

- zobrazení tlačítka pro vybrání všech položek ve sloupci, i když v daném sloupci lze vybrat maximálně jednu položku,
- chybějící zobrazení místností a učitelů, kteří nejsou přiřazeni do žádné skupiny,
- zaškrtávací políčko pro schování předmětů s prázdnými časovými lístky bylo v menší velikosti okna zablokováno jinými prvky (změna zaškrtávacího políčka je zachycena na obrázku 6.1).



(a) Původní zobrazení **(b)** Nové zobrazení

Obrázek 6.1 Upravené zobrazení sloupce (včetně jiné lokace filtru s prázdnými předměty)

(a) Původní zobrazení

(b) Nové zobrazení

Obrázek 6.2 Upravený panel s tlačítky pro migraci dat

6.2.9 Zhodnocení testování

Všechny akce uživatelského testování kromě zjištěných problémů byly na základě zpětné vazby intuitivní a dostatečně přehledné. Dále bylo během převodu oceněno okamžité propsání změn v softwaru Syllabus Plus.

Testování ukázalo, že výsledná aplikace je přehledná, funkční v prostředí, ve kterém bude software používaný, a neobsahuje chyby ovlivňující funkcionalitu. Po opravení zjištěných problémů (viz podkapitola 6.2.8) je intuitivnost aplikace splněna.

Jako nápad na vylepšení byla zmíněna možnost zobrazení detailu časových lístků pro více předmětů najednou a možnost výběru více položek pomocí klávesové zkratky „ctrl + shift“. Na základě krátké konverzace ale nebylo zřejmé, zda by to mělo reálné použití v tvorbě rozvrhu, proto bych nechal tento nápad spíše do případného vylepšení aplikace.

Manuální scénáře potvrdily splnění a funkčnost všech požadavků aplikace. Otestování správného výpisu jednotlivých položek s exportem časových lístků bylo provedeno odděleně s několika ukázkovými záznamy, kdy exportovaná data měla správný formát a patřičné změny se korektně provedly.

Kapitola 7
Závěr

V práci jsem se zabýval analýzou procesu tvorby rozvrhu, návrhem, implementací a testováním aplikace.

V rámci analýzy jsem zjistil, jak funguje proces tvorby rozvrhu na Fakultě informačních technologií. V procesu jsem se potom zaměřil na část s převody dat mezi systémem KOS a softwarem Syllabus Plus. Zjistil jsem, jak funguje aktuální řešení a analyzoval jeho problémy. Těmi hlavními jsou částečné opakované převody, přehlednost operací pro fakultního rozvrháře a uživatelské rozhraní. Na základě zjištěných informací jsem pak sestavil požadavky a případy užití.

V další kapitole jsem se zabýval návrhem řešení. Během návrhu jsem přišel na to, že nejlepší možností je mít všechna data v jedné aplikaci a umožnit uživateli rozhodnout, co s daty udělá, na základě vizualizace se zobrazením rozdílů položek jednotlivých systémů. Uživatel tak může identifikovat případné další nekonzistence, které mohly při práci vzniknout, a předcházet zbytečným chybám. Díky tomu také odpadá nutnost poznamenávání úprav stranou při doplnění nových informací do systému KOS. Povedlo se mi ale také vylepšit připojení k softwaru Syllabus Plus. Místo využití pomocné databáze se má aplikace umí připojit k softwaru Syllabus Plus na přímo. To mi umožnilo zjednodušit celý proces o jeden krok a zároveň jsem díky tomu získal více možností (například naplánování časového lístku).

V implementaci jsem pak dal návrhy dohromady a za použití několika knihoven jsem sestavil grafické rozhraní a plně funkční aplikaci. Bohužel, během této cesty jsem narazil na několik problémů. Většinu se mi podařilo vyřešit, ale dva v aplikaci zůstaly. Naštěstí jeden bude pravděpodobně opraven s aktualizací knihovny pro vykreslování uživatelského rozhraní a druhý se téměř neprojevuje a případně nemá zásadní dopad na funkčnost aplikace.

V rámci testování jsem sestavil scénáře, které pokrývají veškeré funkcionality a slouží pro otestování samotné aplikace. Dále jsem provedl uživatelské testování s fakultním rozvrhářem, které odhalilo několik drobnějších chyb. Jejich vyřešením jsem odstranil překážky v intuitivnosti.

Vytvořená aplikace splňuje všechny požadavky, řeší problémy aktuálního řešení a rozšiřuje možnosti při tvorbě rozvrhu.

7.1 Možnosti rozšíření do budoucna

V rámci dalšího vývoje by se dala aplikace rozšířit o dvě nové sekce. Jednou jsou studijní plány a druhou kruhy studentů prvních ročníků. Tyto informace jsou aktuálně manuálně definovány fakultním rozvrhářem v softwaru Syllabus Plus. Díky návrhu aplikace by mělo být možné případnou funkcionalitu vytvořit přidáním pohledů a zpracováním nových dat.

Dalším vylepšením by mohlo být převedení studentů a jejich zapsaných předmětů v daném semestru. Fakultní rozvrhář by tak mohl zahrnout do tvorby rozvrhu nové informace, které by

mohly mít za důsledek zlepšení výsledné podoby rozvrhu. Například by se v případě vzniklých kolizí více časových lístků dalo zredukovat množství situací, kdy si zapsaní studenti nemohou sestavit nekolizní rozvrh.

Na závěr dodám nápad na drobné vylepšení ovládání samotné aplikace, který byl vznesen během uživatelského testování. Jedná se o komplexnější práci s jednotlivými sloupci, kdy by mohlo být možné zobrazit například časové lístky pro více předmětů najednou.

Příloha A

Instalační příručka

Pro správnou instalaci stačí postupovat podle následujícího návodu.

1. Spustit instalační soubor
2. Vybrat místo pro instalaci souborů
3. Spustit a dokončit samotnou instalaci
4. Spustit aplikaci „SPlusKOS“ (lze vyhledat pomocí nabídky Start)
5. V levém horním menu pod položkou „Aplikace“ kliknout na položku „Nastavení“
6. Zobrazí se průzkumník souborů (pokud se nezobrazí, tak se do stejné složky dá dostat pomocí cesty „%appdata%\SPlusKos“)
7. Otevřít soubor „settings.properties“ v textovém prohlížeči
8. Nastavit položky *KOSAPIClientId* a *KOSAPIClientSecret*
   (potřebné údaje je možné získat na adrese `https://auth.fit.cvut.cz/manager`)
9. Uložit soubor
10. Restartovat aplikaci „SPlusKOS“

Po dokončení těchto kroků je aplikace úspěšně nainstalována a připravena k používání.

Pro odinstalaci aplikace stačí přejít do ovládacích panelů systému Windows. Mezi programy pak vyhledat aplikaci „SPlusKOS“ a kliknout na tlačítko „Odinstalovat“. Program se odstraní ze všech míst, kromě cesty „%appdata%\SPlusKos“ s konfiguračními soubory. Tu pak může uživatel ponechat (třeba v případě pouhé přeinstalace programu), nebo ji smazat.

Příloha B

Navázání spojení se softwarem Syllabus Plus

Pro korektní navázání spojení se softwarem Syllabus Plus může být potřeba provést několik kroků.

1. Otevřít obraz v softwaru Syllabus Plus
2. V levém horním menu pod položkou „Com“ zvolit položku „ProgID...“
3. Zobrazí se dialogové okno, do kterého se napíše unikátní identifikátor pro navázání spojení obrazu
4. Po změně je potřeba obraz uložit (software Syllabus Plus se automaticky restartuje)
5. V levém horním menu pod položkou „Com“ zvolit položku „Register Server (User)“
6. Spustit aplikaci pro převod
7. V pravém horním rohu kliknout na tlačítko s šipkou dolů, zobrazí se dialogové okno
8. Do textového pole napsat stejný unikátní identifikátor, který byl nastaven obrazu v softwaru Syllabus Plus
9. Potvrdit změnu

Při úspěšném navázání spojení se vedle textu „Syllabus Plus“ v pravém horním rohu objeví zelená tečka.

Bibliografie

1. DOMBEK, Daniel. *Fakultní rozvrhář* [online rozhovor]. 2021.
2. JIRŮTKA, Jakub. *O projektu – KOSapi* [online]. 2020 [cit. 2021-05-06]. Dostupné z: `https://kosapi.fit.cvut.cz/projects/kosapi/wiki`.
3. JIRŮTKA, Jakub. *CTU Sirius API API documentation* [online] [cit. 2021-05-06]. Dostupné z: `http://cvut.github.io/sirius/docs/api-v1.html`.
4. JIRŮTKA, Jakub. *Sirius: CTU Time management API* [online]. 2020 [cit. 2021-04-04]. Dostupné z: `https://github.com/cvut/sirius/`.
5. SCIENTIA, Ltd. *Agents Handbook*. 2007. [dokumentace softwaru Scientia Syllabus Plus].
6. SCIENTIA, Ltd. *Syllabus Plus Server*. 2007. [dokumentace softwaru Scientia Syllabus Plus].
7. JETBRAINS. *Compose for Desktop UI Framework* [online] [cit. 2021-01-30]. Dostupné z: `https://www.jetbrains.com/lp/compose`.
8. JETBRAINS. *FAQ – Kotlin Programming Language* [online]. 2021 [cit. 2021-01-30]. Dostupné z: `https://kotlinlang.org/docs/reference/faq.html`.
9. EPFL. *Frequently Asked Questions – Java Interoperability* [online]. 2012 [cit. 2021-05-03]. Dostupné z: `https://www.scala-lang.org/old/faq/4`.
10. JETBRAINS. *Coroutines guide* [online]. 2021 [cit. 2021-04-22]. Dostupné z: `https://kotlinlang.org/docs/coroutines-guide.html`.
11. JETBRAINS. *Integration Compose for Desktop into Swing-based application* [online]. 2021 [cit. 2021-04-21]. Dostupné z: `https://github.com/JetBrains/compose-jb/tree/master/tutorials/Swing_Integration`.
12. ORACLE. *About the JFC and Swing* [online] [cit. 2021-05-04]. Dostupné z: `https://docs.oracle.com/javase/tutorial/uiswing/start/about.html`.
13. MICROSOFT. *COM, DCOM, and Type Libraries* [online]. 2021 [cit. 2021-04-22]. Dostupné z: `https://docs.microsoft.com/en-us/windows/win32/midl/com-dcom-and-type-libraries`.
14. KAWAGUCHI, Kohsuke. *com4j* [online]. 2014 [cit. 2021-04-22]. Dostupné z: `https://com4j.kohsuke.org/`.
15. KAWAGUCHI, Kohsuke. *com4j tutorial* [online]. 2014 [cit. 2021-04-22]. Dostupné z: `https://com4j.kohsuke.org/tutorial.html`.
16. JETBRAINS. *Welcome | Ktor* [online]. 2020 [cit. 2021-04-22]. Dostupné z: `https://ktor.io/docs/welcome.html`.

17. SSCHRASS. *Gradle Kotlin DSL: Define Kotlin version in unique place – Stack Overflow* [online]. 2019 [cit. 2021-05-03]. Dostupné z: https://stackoverflow.com/questions/47588311/gradle-kotlin-dsl-define-kotlin-version-in-unique-place.
18. JETBRAINS. *Native distributions and local execution* [online]. 2021 [cit. 2021-04-28]. Dostupné z: https://github.com/JetBrains/compose-jb/tree/master/tutorials/Native_distributions_and_local_execution.
19. ORACLE. *Windows Disk Space Requirements for JDK and JRE* [online]. 2021 [cit. 2021-05-03]. Dostupné z: https://www.oracle.com/java/technologies/javase/windows-diskspace.html.
20. JIRŮTKA, Jakub. *URL Parametry – KOSapi* [online]. 2016 [cit. 2021-04-26]. Dostupné z: https://kosapi.fit.cvut.cz/projects/kosapi/wiki/URLParameters.
21. JIRŮTKA, Jakub. *Zabezpečení – KOSapi* [online]. 2015 [cit. 2021-04-26]. Dostupné z: https://kosapi.fit.cvut.cz/projects/kosapi/wiki/Security.
22. JIRŮTKA, Jakub. *OAuth 2.0* [online]. 2017 [cit. 2021-05-09]. Dostupné z: https://rozvoj.fit.cvut.cz/Main/oauth2.
23. KAWAGUCHI, Kohsuke. *kohsuke/com4j: Type-safe Java/COM binding* [online]. 2016 [cit. 2021-05-06]. Dostupné z: https://github.com/kohsuke/com4j.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `com4j` ..... zdrojové kódy upravené knihovny com4j
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF