Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Front-end studijní agendy základní školy pro benchmark testovacích nástrojů

Plzeň 2022 Martin Jakubašek

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin JAKUBAŠEK**
Osobní číslo:	**A19B0069P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Front-end studijní agendy základní školy pro benchmark testovacích nástrojů**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s typy dat, které je potřeba uchovávat ve studijní agendě základní školy a s daty uchovávanými back-endem studijní agendy základní školy.
2. Seznamte se s webovými službami.
3. Vytvořte front-end studijní agendy základní školy komunikující s back-endem přes webové služby. Při návrhu důsledně dbejte na zamýšlený účel aplikace, kterým bude benchmark testovacích nástrojů.
4. Navržený front-end implementujte. Dbejte na řádné oddělení jednotlivých vrstev/částí aplikace dle zvolené architektury.
5. Vytvořenou aplikaci důkladně otestujte různými typy testů a kriticky zhodnoťte pokrytí těmito testy.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Tomáš Potužák, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 23. června 2022

Martin Jakubašek

Poděkování

Tímto bych rád poděkoval vedoucímu mé bakalářské/diplomové práce Ing. Tomášovi Potužákovi, Ph.D. za všestrannou pomoc, množství cenných a inspirativních rad, podnětů, doporučení, připomínek a zároveň za velkou trpělivost s obdivuhodnou ochotou při konzultacích poskytnutých ke zpracování této práce.

Abstract

This bachelor thesis deals with the design and implementation of a front-end application of the study agenda of a primary school. The application communicates with the back-end via the REST interface. The main purpose of the application is to be part of the benchmark of testing tools. The application, therefore, emphasizes testing and eliminating as many errors as possible. The theoretical part of the thesis describes the study agenda, what data it must keep, back-end data, and web services. The practical part describes the implementation of the application and its testing.

Abstrakt

Tato bakalářská práce se zabývá návrhem a realizací front-endové aplikace studijní agendy základní školy. Aplikace komunikuje přes REST rozhraní s back-endem. Hlavním účelem aplikace je být součástí benchmarku testovacích nástrojů. Aplikace tedy klade důraz na testování a odstranění co největšího počtu chyb. Teoretická část práce popisuje studijní agendu, jaká data si musí uchovávat, data back-endu a webové služby. V praktické části je popsána implementace aplikace a její otestování.

Obsah

1 Úvod

Ve školním roce 2020/2021 byla úspěšně vytvořena a obhájena bakalářská práce „Aplikace pro studijní agendu základní školy s webovou službou“ [3]. Bakalářská práce se zabývala návrhem a implementací agendy základní školy, kde rozhraní aplikace bylo poskytováno webovou službou `REST`. Aplikace byla vytvořena za účelem použití v benchmarku testovacích nástrojů. Bakalářská práce tedy kladla důraz spíše na bezchybnou funkčnost než na implementaci veškerých funkcí potřebných pro studijní agendu. Přesto poskytuje většinu funkcionality pro studijní agendu základní školy [3].

Cílem této bakalářské práce je navázat na výše zmíněnou bakalářskou práci a vytvořit pro její back-end front-end uživatelské rozhraní. Uživatelské rozhraní bude jak textové (Command Line Interface – CLI), tak grafické (Graphical User Interface – GUI). Aplikace bude klást důraz spíše na CLI rozhraní, a to z důvodu použití práce primárně pro benchmark testovacích nástrojů, podobně jako back-end. I přes zaměření by aplikace měla obsahovat veškerou funkcionalitu poskytovanou back-endem a sloužit tedy v jisté, ale omezené podobě, i jako plnohodnotná aplikace.

V práci jsou nejprve diskutována potřebná data, která studijní agenda musí uchovávat. Dále jsou popsány webové služby a existující back-end studijní agendy základní školy. V praktické části je pak popsána analýza front-endu a následně jeho implementace. Následuje podrobný popis provedeného testování a závěr práce.

2 Studijní agenda základní školy

Studijní agenda se dá definovat jako informační systém určený pro základní, střední, vyšší odborné i vysoké školy. Za školní agendu tedy můžeme považovat takový informační systém, který slouží především pro potřeby vedení školy, a to konkrétně pro potřeby vedení školní dokumentace, školní matriky, vnitřní správu, inventuru majetku a komunikace škol mezi sebou, tak i se správními orgány nebo samotnými žáky/studenty a případně jejich zákonné zástupců [12, 13].

Vzhledem k povaze instituce, pro kterou je školní agenda určena, je její obsah zčásti určen zákonem č. 561/2004 Sb., o předškolním, základním, středním, vyšším odborným a jiném vzdělání (Školský zákon) [21]. V základní podobě vyhovující těmto podmínkám bychom si tedy měli představit školní agendu jako pod část nějakého informačního systému, umožnujícího vést školní dokumentaci a školní matriku v elektronické podobě. Jak školní dokumentace, tak i školní matrika se řídí právě zmíněným školským zákonem [12].

Podle zákony by tedy školní agenda měla umožňovat správu [12, 21]:

- *Školní matriky.* Ta obsahuje evidenci dětí, žáků nebo studentů.
- *Školní dokumentace*: Zde se jedná o povinnou dokumentací, kterou musí školy vést dle zákona č. 561/2004 Sb. a vyhlášky č. 364/2005 Sb., kde je také i její obsah.

Pro naše účely, kdy se zabýváme agendou určenou pro základní školy, můžeme definici školní agendy omezit, na školní agendu zabývající se jen správou základní školy [12].

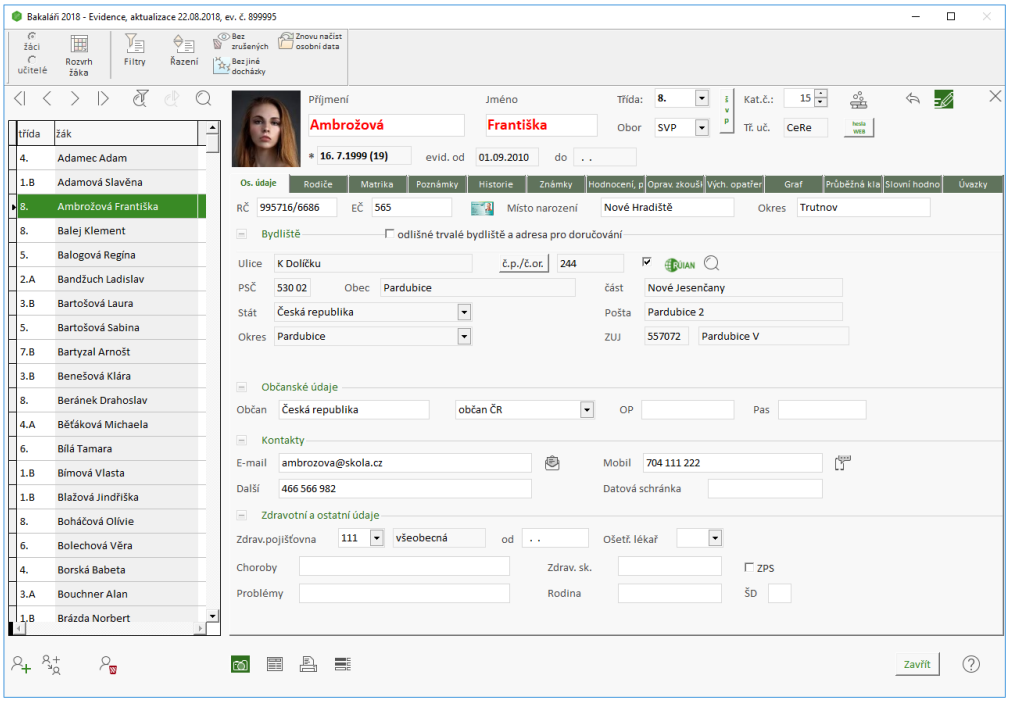
2.1 Existující systémy pro studijní agendu

Ze známých informačních systémů v České republice, splňující výše uvedené požadavky na studijní agendu, se jedná například o [13]:

- aSc Rozvrhy (Applied Software Consultants)
- Bakaláři (Bakaláři Software)

- dm Vysvědčení, dm Evidence, dm Knihovna (dm Software)
- iškola (Computer Media)
- RELAX KEŠ (Alis)
- SAS (MP-Soft)
- Škola OnLine (CCA Group)

Všechny tyto informační systémy nejen plní vlastnosti námi definované školní agendy, ale obsahují i další sadu užitečných funkcionalit. Většina takových informačních systémů bývá rozdělena na moduly, kde každý modul plní nějakou vlastnost studijní agendy, která může nebo nemusí být navázaná na další moduly. Na obrázku 2.1 můžeme vidět ukázku takového jednoho modulu. Tyto moduly nejen plní funkcionalitu danou zákonem, ale mimo jiné plní jak administrativní záležitosti, jako je správa majetku, inventury, správa rozvrhů, suplování, plány akcí školy, tak i zobrazení obsahu učitelům, žákům a rodičům a jejich možnou vzájemnou komunikaci a dále také komunikací se správními orgány [12, 13].



Obrázek 2.1: Ukázka modulu školní agendy Bakaláři pro správu žáků. Převzato z [1].

2.2 Typy dat uchovávané studijní agendou základní školy

Jak již bylo zmíněno v předchozí kapitole, studijní agenda pro základní školu musí obsahovat, podle zákoníku České republiky, vedení povinné školní dokumentace, která je definována v právních předpisech České republiky. Tyto legislativní požadavky mají tedy přímý vliv na data, které školní agendy musí uchovávat [12].

Vzdělání na základních školách se řídí dle § 2 zákona č. 561/2004 Sb., o předškolním, základním, středním, vyšším odborným a jiném vzdělání (školský zákon). Dále jsou dle zákona zákonní zástupci nezletilých žáků povinni dle § 22 školského zákona oznamovat údaje definované v § 28 školského zákona včetně dalších údajů podstatných pro průběh vzdělávání či bezpečnost žáka. Dále musejí dokládat dle § 21 školského zákona důvody nepřítomnosti žáka ve vyučování v souladu s podmínkami školního řádu. Nicméně školní agenda musí poskytovat dle § 21 školského zákona, žákům a jejich zákonným zástupcům informace o průběhu a výsledcích vzdělávání daného žáka a právo vyjadřovat se k rozhodnutí týkající se jeho vzdělávání [12, 21].

2.2.1 Dokumentace o průběhu vzdělání

Studijní agenda základní školy je tedy podle § 28 školského zákona povinna vést školní dokumentaci o průběhu vzdělaní, která musí obsahovat [21]:

a) rozhodnutí o zápisu do školského rejstříku a o jeho změnách a doklady uvedené v § 147,

b) evidenci dětí, žáků nebo studentů (dále jen „školní matrika“),

c) doklady o přijímání dětí, žáků, studentů a uchazečů ke vzdělávání, o průběhu vzdělávání a jeho ukončování,

d) vzdělávací programy podle § 4 až 6,

e) výroční zprávy o činnosti školy,

f) třídní knihu, která obsahuje průkazné údaje o poskytovaném vzdělávání a jeho průběhu,

g) školní řád nebo vnitřní řád, rozvrh vyučovacích hodin,

h) záznamy z pedagogických rad,

i) knihu úrazů a záznamy o úrazech dětí, žáků a studentů, popřípadě lékařské posudky,

j) protokoly a záznamy o provedených kontrolách a inspekční zprávy,

k) personální a mzdovou dokumentaci, hospodářskou dokumentaci a účetní evidenci a další dokumentaci stanovenou zvláštními právními předpisy.

2.2.2 Školní matrika

Mimo školní dokumentaci musí také školní agenda základní škola vést podle školského zákona záznam o školní matrice. Školní matrika je definována jako evidence údajů dětí, žáků a studentů, do které dle zákona smí zapisovat údaje jen škola samotná. Jiný obsah školní agendy však může upravovat i jiný subjekt, než škola samotná. Dle školského zákona může být veden elektronicky, ale i v listinné podobě. O datech ve školní matrice rozhoduje zákon č. 364/2005 Sb., o vedení dokumentace škol a školských zařízení. O způsobu vedení rozhoduje ředitel základní školy. Školní matrika je povinná si uchovávat [12, 21]:

a) jméno a příjmení, rodné číslo, popřípadě datum narození, nebylo-li rodné číslo dítěti, žákovi nebo studentovi přiděleno, dále státní občanství, místo narození a místo trvalého pobytu, popřípadě místo pobytu na území České republiky podle druhu pobytu cizince nebo místo pobytu v zahraničí, nepobývá-li dítě, žák nebo student na území České republiky,

b) údaje o předchozím vzdělávání, včetně dosaženého stupně vzdělání,

c) obor, formu a délku vzdělávání, jde-li o střední a vyšší odbornou školu,

d) datum zahájení vzdělávání ve škole,

e) údaje o průběhu a výsledcích vzdělávání ve škole, vyučovací jazyk,

f) údaje o znevýhodnění dítěte, žáka nebo studenta uvedeném v § 16, údaje o mimořádném nadání, údaje o podpůrných opatřeních poskytovaných dítěti, žákovi nebo studentovi školou v souladu s § 16, a o závěrech vyšetření uvedených v doporučení školského poradenského zařízení,

g) údaje o zdravotní způsobilosti ke vzdělávání a o zdravotních obtížích, které by mohly mít vliv na průběh vzdělávání,

h) datum ukončení vzdělávání ve škole; údaje o zkoušce, jíž bylo vzdělávání ve střední nebo vyšší odborné škole ukončeno,

i) jméno a příjmení zákonného zástupce, místo trvalého pobytu nebo bydliště, pokud nemá na území České republiky místo trvalého pobytu, a adresu pro doručování písemností, telefonické spojení.

j) jméno a příjmení, rodné číslo, popřípadě datum narození, nebylo-li rodné číslo dítěti, žákovi nebo studentovi přiděleno, dále státní občanství a místo trvalého pobytu, popřípadě místo pobytu na území České republiky podle druhu pobytu cizince nebo místo pobytu v zahraničí, nepobývá-li dítě, žák nebo student na území České republiky,

k) datum zahájení a ukončení školské služby nebo vzdělávání,

l) údaje o zdravotní způsobilosti, popřípadě o zdravotních obtížích, které by mohly mít vliv na poskytování školské služby nebo vzdělávání,

m) údaje o znevýhodnění dítěte, žáka nebo studenta uvedeném v § 16, údaje o mimořádném nadání, údaje o podpůrných opatřeních poskytovaných dítěti, žákovi nebo studentovi školským zařízením v souladu s § 16, a o závěrech vyšetření uvedených v doporučení školského poradenského zařízení,

n) označení školy, v níž se dítě, žák nebo student vzdělává,

o) jméno a příjmení zákonného zástupce, místo trvalého pobytu nebo bydliště, pokud nemá na území České republiky místo trvalého pobytu, a adresu pro doručování písemností, telefonické spojení.

2.2.3 Zákonné povinnosti s daty

Všechna výše uvedená data ze školní dokumentace a školní matriky musí být studijní agenda poté také schopna předat Ministerstvu školství, mládeže a tělovýchovy České republiky, pro další zpracování a uchovávání [12, 21].

Školní agenda základní školy si však nemusí ukládat jen data daná zákonem, ale v mnoha případech si uchovává i data jiná, která jsou nepovinná, ale svou povahou mohou být nápomocná ke správě školy. Mezi tyto data můžeme například řadit kategorie dat jako je správa a inventuru majetku, data pro lepší vizualizaci povinných údajů, rozvrhy, nebo například evidenci učebních knih nebo přijímacích zkoušek. Tato data však není školní agenda povinna předávat Ministerstvu školství, mládeže a tělovýchovy České republiky [12].

2.3 Data uchovávaná back-endem studijní agendy základní školy

Vzhledem k povaze bakalářské práce a jejímu úzkému vztahu s již vytvořenou bakalářskou prací, která měla za úkol vytvořit back-end studijní agendy základní školy, je potřeba si definovat jaká data si vlastně uchovává [3].

Back-end neobsahuje všechna data potřebná pro legitimní školní agendu, která obsahuje všechna povinná data, daná zákonem č. 561/2004 Sb, o předškolním, základním, středním, vyšším odborným a jiném vzdělání a zákonem č. 364/2005 Sb, o vedení dokumentace škol a školských zařízení. Data byla zmíněna v sekci 2.2.

Protože si ze všech povinných dat back-end uchovává jen jejich malý okruh, tak aby bylo možné simulovat školní agendu pro podmínku aplikace pro benchmark testování nástrojů.

2.3.1 Uchovávaná data daná zákonem

Z povinných údajů si back-end uchovává [3]:

1. Data žáka. Zde si uchovává jeho jméno, příjmení, datum narození, telefon, email a všechny jeho předměty.
2. Data žákovy absence.
3. Data žákova hodnocení.
4. Data předmětů. Zde si uchovává název předmětu.
5. Data tříd. Z třídy si uchovává jejich vyučované předměty, místnost, ve které se třída nachází a třídního učitele

2.3.2 Uchovávaná data nevyžádaná zákonem

Z dat nevyžádaných zákonem, ale důležitých pro správu základní školy obsahuje [3]:

1. Data učitele. Zde si uchovává jeho jméno, příjmení, datum narození, datum nástupu, adresu bydliště, telefon, email a jeho vyučované předměty.
2. Data o správě budov a místností. Z budov si uchovává jejich název. Z místnosti si uchovává jejich název, kapacitu a budovu, ve které se místnost nachází.

3. Data školní akce. Datum jejího konání, popisek, odpovědného učitele a třídu, která se akce týká.

4. Data školního rozvrhu. Čas zahájení a skončení, název, probíranou látku a konkrétního učitele, který akci vedl.

Jak můžeme z výčtu pozorovat, back–end obsahuje průřez dat typický pro školní agendu, a to v menší podobě oproti reálné školní agendě splňující všechny požadavky podle zákona a jiná doplňující data [3, 21].

3 Webové služby

Webová služba je sada protokolů a standardů pro přenos dat mezi různými aplikacemi přes internet. Webové služby mohou být použity různými platformami a programovacími jazyky pro přenos dat přes internet, tak aby umožňovaly komunikaci dvou různých programů stejně jako na jednom zařízení [23].

Za webové služby tedy můžeme považovat jakýkoliv software, který používá standardní webové protokoly pro přenos dat, jako např. HTTP nebo HTTPS, tak aby svým aplikacím poskytly potřebná data. Nejsou nijak omezeny platformou a programovacím jazykem. Pro svůj přenos dat využívají strukturované formáty dat, jako je např. XML nebo JSON. Tím nám umožní zajistit komunikaci dvou rozdílných aplikací, které se dokáží vzájemně domluvit přes výše zmíněné formáty [24].

Webové služby často bývají spojovány s architekturou SOA (Service–Oriented Architecture) [24].

3.1 SOA

SOA (Service–Oriented Architecture) je návrh aplikací zaměřený na služby. Kde služba reprezentuje část, nebo celou funkci, ke které můžeme přistupovat vzdáleně a chová se jako zcela samostatná jednotka. SOA pracuje na principu peer-to-peer. Charakteristiky služby jsou následující [7]:

- Je nezávislá na ostatních komponentech softwaru.
- Má jednoznačnou funkcionalitu.
- Klient nemusí vědět jak funguje, jen co poskytuje.
- Může být složena z více služeb.

Pro účely webových služeb se nejčastěji používají SOA implementace v podobě SOAP nebo REST protokolů, které jsou založeny na HTTP a HTTPS protokolech [7].

3.2 SOAP

SOAP (Simple Object Access Protocol) je protokol pro výměnu strukturovaných dat na webu pomocí formátu zpráv XML. Nejčastěji se používá v

kombinaci s HTTP protokolem. XML v SOAP protokolu můžeme jednoduše rozdělit na dvě části. První část XML definuje jen samotnou strukturu zprávy pomocí definovaných prvků, zatímco druhá část se skládá z ne–definovaných prvků a definuje obsah zprávy. Samotná struktura se poté skládá ze tří částí [18, 19, 23]:

- Obálky (envelope). Definuje samotnou strukturu posílané zprávy a jak jí přečíst. Celá zpráva je tedy obalená v obálce, která obsahuje hlavičku a tělo pro definování zprávy.
- Kódovací pravidla (encoding rules). Sada kódovacích pravidel pro dekódovaní a zakódování deklarovaných datových typů používaných v dané zprávě.
- Styl komunikace (communication styles). Definuje jak se zpráva přenáší, a to buď ve formě RPC (remote procedure call) nebo pomocí dokumentu.

Samotná zpráva se poté skládá z XML dokumentu, který má kořenový prvek: obálka (envelope), ve kterém se nachází prvky hlavička (header) a tělo (body) [19].

- Hlavička (header). Jedná se o nepovinný prvek. Slouží k posílání metadat přenášené zprávy.
- Tělo (body). Obsahuje veškeré definované prvky samotné aplikace.

Listing 3.1: Ukázka SOAP protokolu. Převzato z [20].

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV = "http://www.w3.org/2001/12/a"
    SOAP-ENV:encodingStyle = "1">

    <SOAP-ENV:Body xmlns:m = "http://www.xyz.org/">
        <m:GetQuotation>
            <m:QuotationsName>Name</m:QuotationsName>
        </m:GetQuotation>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.3 REST

REST (Representational State Transfer) není samotný protokol, ale jen styl návrhu přenášených zpráv pro webové služby. Na rozdíl od SOAP protokolu se nejedná o přímou definici protokolu, ale jen o definovanou skupinu pravidel, která by měly přenášené zprávy nad webovými službami dodržovat. Implementace, které splňují tyto pravidla, nazýváme RESTful, tj. REST API. Každý protokol vydávající se za RESTful, by měl dodržovat níže uvedených pět pravidel [5].

3.3.1 Jednotné rozhraní

Každé REST API by mělo dodržovat princip obecnosti. To dosáhne dodržením následujících pravidel [5]:

1. Vše musí být jednoznačně identifikováno, tak aby server/klient na první pohled poznal, o co se jedná.
2. Skupina stejných prvků musí být identifikována stejně.
3. Každý prvek musí nést tolik informace, aby jej bylo možné zpracovat na straně serveru/klienta.
4. Veškerá komunikace se serverem musí probíhat jen pomocí URI. Za URI myslíme jednoznačný identifikátor nějakého obsahu [22].

3.3.2 Klient–server

Rozhraní by mělo využívat datový vzor *klient-server*. To umožnujě lepší rozdělený a vývoj aplikace. Jediné co musí být konstantní je tedy jen jednotné RESTful rozhraní mezi klientem a serverem [5].

3.3.3 Bezstavovost

Každý požadavek klienta musí obsahovat všechny potřebné informace pro porozumění a dokončení požadavku. Server nesmí využívat ke zpracování požadavku data z předešlého požadavku [5].

3.3.4 Ukládání do mezipaměti

Každý požadavek musí nést informaci o jednoznačném rozhodnutí, zda je ho možné uložit do paměti, či nikoliv. Pokud je požadavek možné uložit do mezipaměti, mělo by být aplikace této vlastnosti využít, či nikoliv [5].

3.3.5 Hierarchické rozdělení architektury

Návrh a implementace RESTful API by měla umožňovat kompozici do hierarchické architektury. Tím je umožněno rozdělení rozhraní do jednotlivý navzájem nezávislých vrstev [5].

3.3.6 Přenositelnost kódu

Nepožadováno, avšak je dovoleno definovat standart pro přenos a vzdálené spuštění kódu. To poté umožňuje jednodušší definici rozhraní pro obsluhu takové webové služby, kdy většina kódu může být získána přímo ze serveru, namísto jen dat [5].

3.3.7 Kategorizace

REST tedy neklade omezení na použití formátu pro přenos zpráv. Nejčastěji se stejně setkáme s XML nebo JSON formátem [5]. Nejčastěji se RESTful žádost řadí do kategorií podle HTTP příkazu přenosu [5]:

1. GET. Získání specifického prvku.
2. PUT. Vytvoření, modifikace, změna prvku.
3. PATCH. Změna prvku.
4. POST. Vytvoření, změna prvku.
5. DELETE. Vymazání prvku.

Pro odpověď používá REST API klasické HTTP stavové kódy. Jedná se o kódy z kategorií REST [5]:

- 1xx – informace o přenosu
- 2xx – kódy úspěšného požadavku
- 3xx – kódy indikující nějakou akci navíc potřebnou od klienta pro úspěšný přenos
- 4xx – chybové kódy klienta
- 5xx – chybové kódy serveru

3.4 Formát přenášených zpráv

Historicky můžeme považovat jako jeden z prvních masově rozšířených formátů pro přenos dat po internetu za pomoci webové služby formát HTML. Postupně jak se internet vyvíjel, a s ním i technologie spojené s internetem, vznikla spousta nových formátů pro webové služby, z nichž většina vzala svou inspiraci právě z formátu HTML. Za ten nejdůležitější vycházející z HTML můžeme požadovat XML. Následně s rozšířením programovacího jazyka Javascript vznikl na návaznost XML formát JSON. Jak XML, tak i JSON můžeme dnes považovat za jedny z nejrozšířenějších používaných formátů přenášených zpráv pro účely datového přenosu u webových služeb [15].

3.4.1 HTML

HTML (HyperText Markup Language) si můžeme představit jako formát, který stál na počátku vzniklých formátů pro přenos dat pomocí webové služby. S postupným vyvíjením webových služeb a jejich formátu pro přenos dat však již dnes HTML nemůžeme považovat za univerzální nástroj pro přenos dat, ale spíše jako počáteční návrh a inspirací pro ostatní formáty. Dnes má HTML své využití jako přenos vizuálního obsahu pro webové stránky. Jedná se o značkovací jazyk, kde každý prvek je ohraničen závorkami a má specifický a pevně daný formát a strukturu. Dohromady takto tvoří hierarchický strom takových prvků [15].

Listing 3.2: Ukázka HTML formátu. Převzato z [6].

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

3.4.2 XML

XML (extensible markup language) je textový formát pro reprezentaci dat v lidsky čitelné, ale strukturované podobě, tak že i počítač ho dokáže jed-

noduše přečíst. XML se řadí mezi značkovací jazyky. To znamená, že každý jeho prvek musí být označen značkou, kdy značka narozdíl od HTML není pevně definovaná. Díky vlastnosti jednoduchého popisu, přečtení a zpracování se z něj stal jeden z nejčastěji používaných formátů pro přenos zpráv webových služeb [27].

Listing 3.3: Ukázka XML formátu. Převzato z [26].

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
    </book>
    <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Jak je z ukázky patrné XML, se podobá HTML. Nicméně narozdíl od HTML XML zprávu nelze přečíst s chybou. Pokud by zpráva obsahovala chybu, jediné, co se nám vrátí je zpráva o chybném stavu zprávy. To nám umožňuje spolehlivé zpracování dat, protože víme, že data budou bez syntaktické chyby [28]. Jako hlavní rozdíly oproti HTML můžeme považovat [28]:

- Všechny prvky musí být uzavřeny nebo označeny za prázdné.
- Prvek musí mít hodnotou vždy ohraničenou uvozovkami.
- HTML má přesně dané názvy prvků, zatímco XML ne.

Mezi hlavní výhody XML můžeme řadit nejen dostupnost nástrojů pro tvorbu a zpracování, ale i [28]:

- Detekci chyb. Každý prvek musí být jednoznačně uzavřen závorkami. To nám sice nám to prodlužuje velikost dokumentu, ale na druhou stranu nám to umožňuje jednoduchou a jasnou detekci chyb.
- Jednoduchou čitelnost. XML dokument je možné přečíst i lidským zrakem, což umožňuje jednoduché seznámení se s formátem.
- Příslib, že návrh XML umožňuje tu vlastnost, že ať už máme jakýkoliv nástroj na přečtení zprávy, tak právě návrh XML nám zaručuje že zprávu lze zpracovat a přečíst daným nástrojem.

3.4.3 JSON

JSON (JavaScript Object Notation) je textový formát pro reprezentaci dat v lidsky čitelném formátu, ale strukturované podobě tak, že i počítač ho dokáže jednoduše přečíst. JSON je založen na principech objektové syntaxe jazyka JavaScript. I když je založen na tomto jazyce, tak to neznamená ,že lze použít pro ten daný jazyk. JSON lze použít bez ohledu na zvolený programovací jazyk. JSON je také hluboce svázán s principy jazyka založeného na jazyce C [8, 25].

Listing 3.4: Ukázka JSON formátu. Převzato z [11].

```
{
    "book":[
        {
            "id":"444",
            "language":"C",
            "author": "Dennis Ritchie "
        },
        {
            "id":"555",
            "language":"Java",
            "author": "Bjarne Stroustrup"
        }
    ]
}
```

Jak je možné vidět z ukázky 3.4, JSON je založen na principu prvku, skládajícího se z klíče a hodnoty. Takto definovaný prvek umožňuje výsledným programovacím jazykům převést daný prvek na svojí vlastní určenou interní strukturu. Protože JSON je založen na principu JavaScriptu, umož-

ňuje, aby hodnota prvku nebyl jen obyčejný textový řetězec nebo číslo, ale i datová struktura jako je pole nebo objekt [25].

Za hlavní výhody JSON formátu tedy můžeme považovat jeho rozšířené používání a dále i [25]:

- JSON je hierarchický a jednoznačný, takže i aplikace co nezná formát původního objektu, dokáže prvek přečíst a nahradit svou definicí daného objektu.
- JSON je prostý text, což umožňuje jednoduché přečtení lidským zrakem, tak i počítačem, ale i další bezproblémový přenos, kdy se pro ostatní aplikace jeví jen jako textový řetězec.
- JSON kombinace klíče a hodnoty umožňuje menší velikost souboru oproti XML.

3.4.4 Porovnání XML a JSON

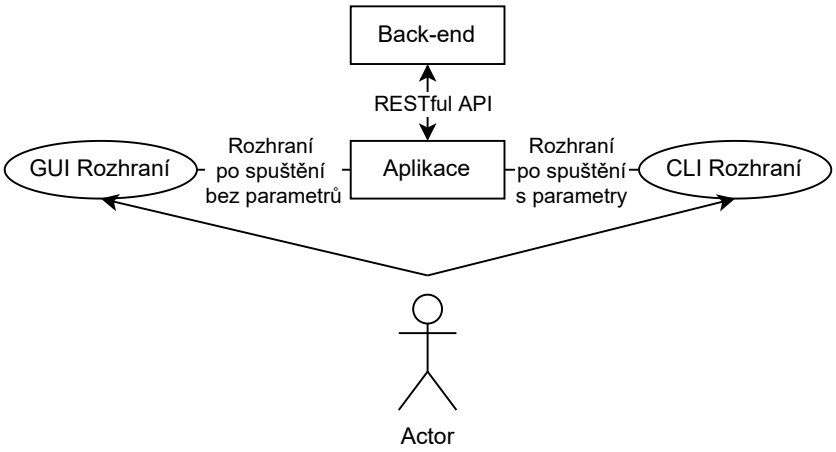
Oba formáty XML a JSON se ve webových službách používají, proto budou v této sekci porovnány jejich vlastnosti. Pro jednoduché aplikace JSON poskytuje dostatečně robustní řešení, které ale může odhalovat své nedostatky. Nedostatků si nejvíce můžeme všimnout při použití ve větších aplikacích, kdy JSON nijak neobaluje data, a tak vlastně ani neříká jakou strukturu data mají. To poté vede k implementaci těchto funkcí až na vyšších vrstvách aplikací. To může způsobovat vícero chyb při zpracování dat, protože poté každá koncová aplikace webové služby musí tato data interpretovat stejným způsobem [16].

Na rozdíl od jednoduššího JSON formátů XML za cenu složitější struktury přináší možnost robustnější kontroly dat pomocí XSD, ekvivalent tohoto u JSON chybí. Pokud bychom tedy měli vícero formátu dat stejného obsahu ve zprávě, XML za nás problém vyřeší již implementovanou strukturou dat, zatímco u JSON musíme definovat logiku zpracování a rozhodování, o který formát zprávy se vlastně jedná [17].

XML nám tedy poskytuje komplexnější definice prvků s možností použití široké škály nástrojů pro zpracování JSON zase umožňuje jednodušší definici a přehlednější definici prvků, za cenu menší samotné logiky přenosu dat [17].

4 Analýza

Úkolem bakalářské práce je vytvořit front–endové rozhraní na již existující back–end rozhraní aplikace pro studijní agendu [3], vytvořenou v rámci jiné bakalářské práce [3]. Back–end obsahuje webovou službu ve formě RESTful API, na které má aplikace této bakalářské práce navazovat, a vytvořit nad ní své vlastní, front–endové rozhraní komunikující s back-endem přes RESTful API a mající jak CLI, tak GUI verzi.



Obrázek 4.1: Schéma spolupráce back-endu a front-endu.

Jak můžeme vidět na obrázku 4.1, očekává se od nás vytvoření aplikace, která tvoří front–end nad již existujícím back–endem spojeným jeho rozhraním, ve formě jedné aplikace obsahující jak CLI, tak i GUI rozhraní.

4.1 Specifikace požadavků

Podle požadavků zadavatele má být taková aplikace vytvořena v programovacím jazyce JAVA. Po aplikaci se tedy očekává to, že by měla poskytovat veškerou funkcionalitu back-endu. Pod tím si můžeme představit obalení všech příkazů poskytovaných pomocí (RESTful API) back–endu, tak aby jejich výsledné spojení pomocí HTTP příkazů bylo skryto před uživatelem front–endové aplikace, ale umožňovalo jeho veškerou funkcionalitu.

Back-end nám pomocí všech svých příkazů umožňuje vytváření, modifikaci a mazání entit a jejich dat, která jsou jím definovaná a byla zde vypsána v kapitole zmiňující se o datech uchovávaný back-endem (2). O všech takových operacích nás poté informují příkazy návratovým kódem HTTP poža-

davku, který pojmenujeme jako stavový kód. Požadavek tedy může vracet jak kód pro označení správné vykonání operace, tak i kód pro opačný příkaz. U záporného výsledku operace ještě dostaneme i podrobný popis toho proč došlo k selhání operace, a tím i příkazu. Mimo stavový kód a při úspěšné operaci nám ještě vrací i struktury vytvořených nebo modifikovaných entit. Zde jsou nám všechna data vrácena pomocí formátu JSON. Předpokládá se tedy, že každý příkaz pro back-end nám vrátí stavový kód a případně i data o vytvořené nebo jinak manipulované entitě ve formě JSON. Případně nám vrátí chybovou hlášku.

Úkolem front-endu je tedy zakrýt webovou službu vlastním rozhraním, a tím zakrýt příkazy webové služby a poskytnout je, a to pomocí dvou takových rozhraní: CLI a GUI rozhraní. Obě rozhraní by tedy i přes zakrytí příkazů neměla nijak zakrývat informaci vrácenou pomocí těchto příkazů. Zároveň by měla být i dostatečně uživatelsky přívětivá, ale i tak by měla plnit veškerou funkcionalitu webového rozhraní.

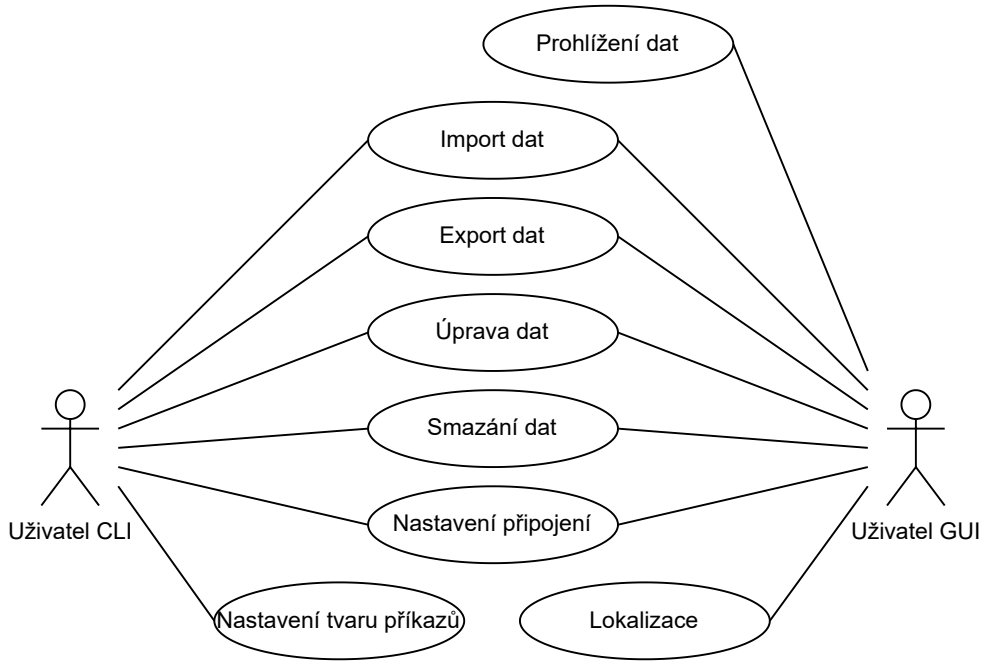
Jak již bylo zmíněno, jedním z hlavních požadavků na aplikaci je její napsání v programovacím jazyce JAVA. V návrhu aplikace by tedy měl být kladen důraz na její objektový návrh, a tím i rozdělení jednotlivých funkcionalit do modulů obsahující stejné nebo související funkcionality aplikace. Při návrhu by dále mělo být počítáno také s možností základní uživatelské modifikace konfigurace spojení s back-endem. Takto vytvořená aplikace nám splní požadavky na front-end, a to ten, že back-end může být přístupný z vícero míst a zároveň jeho implementace se může změnit, zatímco front–end bude stále fungovat. Dále se při návrhu nesmí zapomenout na požadavek vracení dat ve více formátech, a to konkrétně za pomoci JSON formátu a XML formátu. Další důležitou podmínkou poté samostatné aplikace je její dobré otestování. To znamená, že aplikace by měla být tedy důkladně otestována.

4.2 Případy užití

V následujících kapitolách jsou popsány případy užití aplikace.

4.2.1 Diagram případu užití

Na obrázku 4.3 je uveden diagram případu užití.



Obrázek 4.2: Diagram případu užití.

4.2.2 Popis případu užití

Na obrázku 4.3 můžeme vidět diagram případu užití. V zájmu zachování přehlednosti je v diagramu pouze uvedena obecná práce s daty, a ne s konkrétními daty. Jednotlivé případy užití jsou:

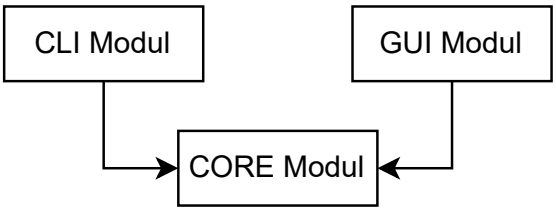
- Import dat. Aplikace by měla umožňovat nahrání dat na back-end, a to z XML, tak u JSON formátů.
- Export dat. Aplikace by měla umožňovat získávat data z back-endu, a to v podobě XML, tak i JSON formátu.
- Úprava dat. Aplikace by měla umožňovat upravovat data z back-endu, a takto upravená data mu poté vrátit.

- Smazání dat. Aplikace by měla umožňovat smazání dat z backendu.
- Nastavení připojení. Aplikace by měla umožňovat změnu připojení na jinou adresu back-endu.
- Nastavení tvaru příkazů. Aplikace by měla umožňovat změnit tvar zadávaných příkaz CLI.
- Lokalizace. Aplikace by měla umožňovat změnu lokalizace na anglickou a českou.
- Prohlížení dat. Aplikace by měla umožňovat zobrazování dat backendu.

4.3 Architektura aplikace

Podle výše zmíněného návrhu můžeme funkcionalitu aplikace rozdělit do tří modulů. Důležité je také zmínit, že pod modulem se myslí stejné pojmenování pro podobné funkcionality. Konkretně na modul pro:

- Obecné funkcionality pro připojení a manipulaci.
- Funkcionalitu `CLI` rozhraní.
- Funkcionalitu `GUI` rozhraní.

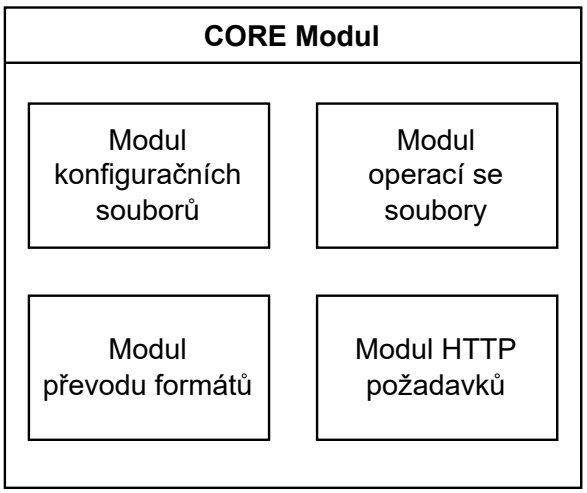


Obrázek 4.3: Ukázka modulů front–endové aplikace.

Jak můžeme vidět na obrázku (4.3), výše rozdělené moduly můžeme naznačit grafem, kdy za CORE modul bereme modul obsahující obecné funkcionality pro připojení a manipulaci, za CLI modul považujeme funkcionality CLI rozhraní a za GUI modul považujeme modul pro funkcionality GUI rozhraní. Z funkcí modulů by také měl vyplývat jejich vzájemný vztah, kdy CORE modul nám definuje základ, který budou používat obě rozhraní. Důležité je také ještě jednou zmínit, že pod modulem se myslí stejné pojmenování pro podobné funkcionality.

4.4 Funkcionality CORE modulu

Core modul se skládá ze 4 částí - modul konfiguračních souborů, modul operací se soubory, module převodu formátů a modul http požadavků. Jednotlivých částem jsou věnovány následující podkapitoly.



Obrázek 4.4: Ukázka CORE modulu front–endové aplikace.

4.4.1 Modul HTTP požadavků

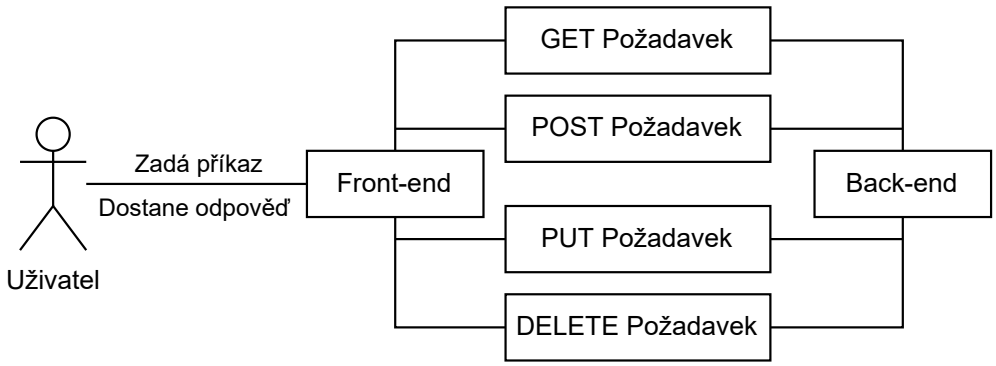
Protože back-end funguje na bázi webové služby implementované pomocí RESTful API, musí být front–end aplikace schopna posílat a přijímat HTTP požadavky. Pokud se podíváme na příkazy poskytované back–endem [3], zjistíme že je potřeba zpracovávat a posílat HTTP požadavky typu GET, PUT, POST a DELETE.

- GET. U požadavku je potřeba umět zpracovat přijatá data ve formátu JSON a stavový kód požadavku. Vzhledem k tomu, že požadavky GET jsou nejvíce rozmanité pro všechny entity, bude zapotřebí vytvořit funkcionalitu GET požadavků dostatečně obecně, aby byla schopna pokrýt vícero end–pointů, v různých provedeních pro každou entitu. To plyne z back–endových příkazů GET [3], které nám umožňují získat nejen všechny entity daného typu, ale i jejich pokročilé filtrování.
- POST. U požadavku je potřeba umět odeslat data ve formátu JSON podle specifikace dané jednotlivými příkazy [3]. Zde může být žádoucí provést předběžnou kontrolu dat a ušetřit odeslání zbytečných požadavků se špatnými daty. Díky svému účelu se neočekává větší množství uživatelů komunikujících pomocí front-endu s back-endem a tak

nehrozí přetížení back-endu požadavky se špatnými daty. Tím se i potenciálně vyhneme i špatné kontrole možných chyb, kdy takto stačí poté chybu detekovat a opravit pouze na back-endu. Ať už se zvolí jakákoliv možnost, bude zapotřebí umožnit vkládání i ve formátu XML, a to znamená, že bude potřeba modulu pro převod formátů. Po odeslání příkazů je nám vracen i stavový kód, který bude sloužit jako zpětná vazba po vložení. Vzhledem k povaze příkazů pro back-end, který umožňuje jen vložení jedné entity, je požadavkem pro front-end zakrýt tuto funkcionalitu možností vložení vícero prvků naráz. Jde nám tedy jen o to, narozdíl od back-endu, zakrýt vkládání entity po jedné, za vkládání vícero entit, jako jeden příkaz.

- PUT. U požadavku je potřeba umět vkládat modifikovanou entitu do JSON formátu, a tu následně odestlat jako příkaz. Zde je potřeba umět jen nadefinovat danou entitu stejně jako v POST požadavku s tím, že je možné mít entity, které neobsahují všechna svá data. Je také zapotřebí umět vkládat do adresy HTTP požadavku id modifikované entity.

- DELETE. V podstatě nejjednodušší příkaz na zpracování. Zde nic neposíláme, jen je potřeba správně nadefinovat adresu HTTP požadavku se správným id. Zde jediné, co je poté třeba zpracovat, je stavový kód.

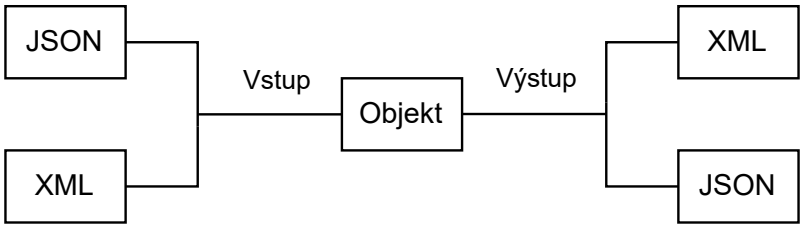


Obrázek 4.5: Ukázka modulu pro HTTP požadavky front–endové aplikace.

Z výše popsané analýzy nám vyplývá potřeba vytvoření modulu na zpracovávání a odesílání výše zmíněních HTTP požadavků. S tím, že požadavky přijímají jen data v JSON formátu, ale naše aplikace musí umět i přijímat data v XML formátu, je potřeba nadefinovat další modul, který nám bude umět převést data z jednoho formátu do druhého. Funkčnost takového modulu poté můžeme vidět na grafu modulu požadavku (4.5).

4.4.2 Modul převodu formátů

Dále tedy bude potřeba vytvořit modul pro převod datových formátů XML a JSON mezi sebou. Takový modul může využívat vlastnosti programovacího jazyka JAVA a převádět nám nejdříve datové formáty do objektu, až poté na druhý formát. Tím zajistíme i základní kontrolu integrity dat, ne však jejich obsahu. To můžeme již ponechat na back-endu. Funkcionalitu modulu poté můžeme vidět na grafu ukázky modulu (4.6).



Obrázek 4.6: Ukázka modulu pro převod formátů front-endové aplikace.

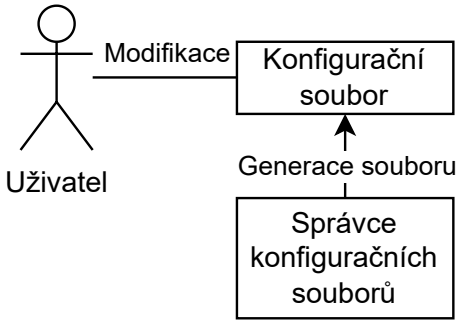
4.4.3 Modul operací se soubory

Protože ve většině případů bude naším zdrojem dat pro vkládání a modifikaci soubor, je potřeba také vytvořit modul pro načítání a vytváření souborů s vhodným datovým formátem, a to konkrétně XML a JSON formát. Zde je potřeba jen mít na vědomí správnou uživatelskou definici do daných formátů, které přijme back-end. Protože oba dva formáty jsou formáty textové, načítání není nijak potřeba složitě definovat, bude nám stačit jen načítání všech souborů do textového řetězce. Vše ostatní by nám už měly řešit výše zmíněné moduly. Pro vytváření nám vlastně stačí jen ukládat textový řetězec do souboru. Vizualizace nám poté může řešit externí program sloužící pro zobrazení těchto formátů, který je i zpřehledňuje (např. pomocí zvýraznění syntaxe) a zjednodušuje jejich přečtení lidmi.

4.4.4 Modul konfiguračních souborů

Poslední obecným modulem, který je třeba vytvořit, je modul pro správu konfiguračních souborů, a o konkrétně konfiguračního souboru pro specifikace připojení. Ten se skládá z adresy, portu, jména a hesla. Protože se jedná o aplikaci pro testovací účely, stačí takovýto soubor poskytovat ve formátu textovém, a to i s heslem. Pro vytvoření takového souboru můžeme využít klasického souboru pro JAVA specifické programy, a to properties souboru.

Ten se skládá z dvojice klíč a hodnota. Takový soubor je potřeba poté distribuovat s aplikací a umožnit i jeho editaci. Tento soubor by poté interně měl být využíván modulem pro obsluhu HTTP požadavků. Je třeba také myslet na to, že uživatel může soubor upravit do špatného formátu. Je tedy potřeba umožnit vygenerování souboru nového. Ukázku takového modulu můžeme poté vidět na ukázce schématu modulu (4.7).



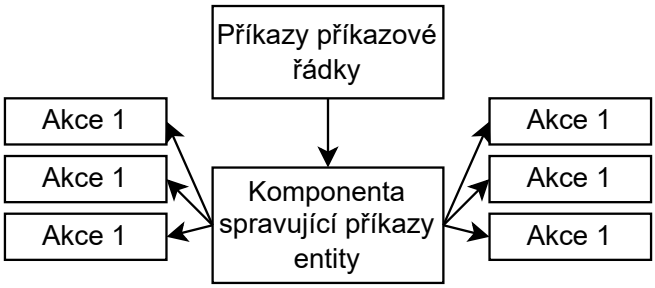
Obrázek 4.7: Ukázka modulu pro správu konfiguračních souborů front-endové aplikace.

4.5 Funkcionality CLI rozhraní

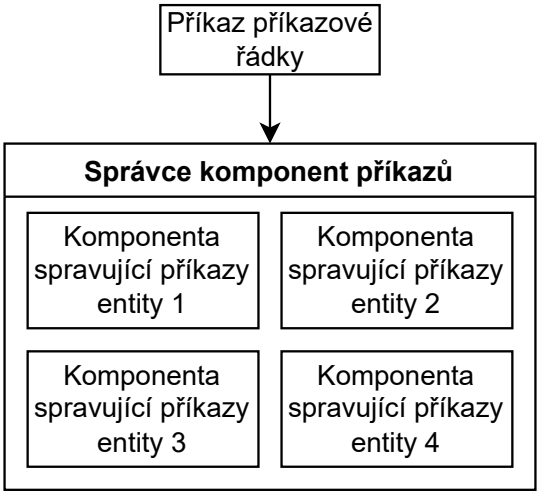
Pro modul rozhraní CLI je potřeba vytvořit příkazy v příkazové řádce a jejich interpreta. Příkazy nám poté vykonávají funkce příkazů HTTP RESTful API rozhraní webové služby back–endu. Každá entita má svojí vlastní specifickou sadu příkazů, které se liší, a mohou být tedy jen specifické pro danou entitu. Proto zde bude potřeba modul rozdělit na komponenty, kde každá komponenta by měla spravovat příkazy vlastní entity. Protože příkazy očekávají data, bude potřeba definovat způsob, jakým se do příkazu budou data vkládat. Podle specifikací však víme, že se data budou do příkazu dávat jen pomocí souborů. Bude tedy potřeba nad příkazy back–endové vytvořit obalení pro každý specifický formát. V našem případě se tedy jedná o formáty XML a JSON. Kromě tohoto obalení bude potřeba obalit POST příkazy, tak aby zvládly soubory s vícero entitami.

Znamená to tedy, že pro každou entitu bude potřeba vytvořit komponentu spravující její příkazy, tak jak je vyobrazeno na obrázku 4.8. Nad takovými komponentami poté bude vytvořit jednu, která nám bude komponenty vybírat a spravovat Takovou komponentu můžeme vidět na obrázku 4.9.

Protože příkazů bude velké množství, bylo by dobré vytvořit konfigurační soubor obsahující všechny komponenty příkazů, tak aby bylo možné



Obrázek 4.8: Ukázka komponenty spravující příkazy.



Obrázek 4.9: Ukázka správce komponent spravující příkazy entit.

uživatelsky modifikovat zadání příkazů stejně, jako u konfiguračního souboru připojení, který byl zmíněn výše. To může poté ulehčit testovatelnost, a také uživatelský komfort, kdy formu příkazů bude možné lehce změnit. Také pokud konfigurační soubor bude správně formátován, může sloužit i jako šablona všech možných příkazů a jako první rada, jak příkazy zadat.

Vzhledem k velkému množství příkazů bude také potřeba vytvoření správce těchto příkazů, který nám z příkazové řádky transformuje příkaz na námi zadaný, i se všemi jeho specifikacemi. Pokud se podíváme na tyto příkazy do hloubky zjistíme, že všechny mohou obsahovat malou množinu možných typů jednotlivých podpříkazů, a to:

- Podpříkaz definující část příkazu. Zde je definován textovým řetězcem určující typ příkazu, tj. pro kterou entitu, a jako její manipulaci bude provádět.
- Podpříkaz hodnoty. Zde bude definována hodnota pro příkaz, např. pro id manipulované entity.

- Podpříkaz definující typ souboru. Zde se bude jednat o podpříkaz definující jestli soubor s daty je typu XML nebo JSON.

```
Budova  Modifikuj  4  XML  mojeBudovy.xml
```

Obrázek 4.10: Ukázka možného příkazu příkazové řádky entity budova, modifikující budovu id 4, kde specifikace modifikovaných dat je dostupná v souboru XML mojeBudovy.xml.

Takto bude poté potřeba definovat všechny příkazy a jejich samotnou obsluhu, která nám transformuje příkaz v příkazové řádce na příkaz backendu. Ukázku možného příkazu můžeme spatřit na obrázku 4.10

4.6 Funkcionality GUI rozhraní

Pro modul rozhraní GUI bude potřeba vytvořit standardní GUI aplikaci. V podstatě se dá říct, že příkazy CLI rozhraní se nám transformují na tlačítka a formuláře. GUI by tedy mělo poskytovat vše nebo téměř vše co CLI rozhraní. Protože zde nebudeme vkládat entity jako soubory, ale na místo nich budou používány formuláře pro modifikaci a vkládání entit, bude zde tedy potřeba vytvořit seznam všech možných entit, a nad každou entitou takového seznamu umožnit její podrobnější editaci.

Editace každé entity by se poté měla skládat z možnosti výběru všech entit ze seznamu, nad kterým bude potřeba vytvořit nějakým způsobem filtraci entit, tak aby se co nejvíce podobala filtrování pomocí příkazů CLI aplikace. Poté je potřeba nad každou entitou umožnit její zobrazení, které bude formou formuláře, na rozdíl od souboru druhého rozhraní. To nám umožní i možnou modifikaci této entity. Nakonec bude potřeba umožnit vkládání nové entity, také pomocí formuláře.

I přesto že se jedná o aplikaci určenou k benchmarku testovacích nástrojů, je dobré zajistit i lokalizaci. Celé GUI bude potřeba i lokalizovat.

5 Implementace

Implementaci samotnou můžeme rozdělit do tří logických modulů, podle jejich funkcionality. Modulem se zde myslí jen pojmenování pro množinu obsahující stejné funkcionality. Moduly to jsou tedy následující:

- Modul pro obecné funkce aplikace a funkce pro HTTP dotazy na back–end. Tento modul označíme jako CORE modul
- Modul pro CLI rozhraní.
- Modul pro GUI rozhraní.

Taková implementace nám splňuje i počáteční návrh v analýze, kdy se aplikace dělila na tři moduly, a to tedy konkrétně modul CORE, modul CLI rozhraní a modul GUI rozhraní. Module v tomto případě se myslí sjednocení podobných funkcionalit. Celá aplikace se poté řídí tří–vrstvou architekturou, tj. má prezentační, aplikační a datovou vrstvu, v podobě popsané níže.

Jak již bylo zmíněno v analýze, aplikace je implementována v programovacím jazyce Java, a to konkrétně verze `17.0.2`. Pro správu vývoje aplikace byl zvolen nástroj Maven verze `4.0.0`. Ten zde hlavně zastává roli správce použitých knihoven, které budou vždy uvedeny u funkcionality, u které jsou použity, a také kvůli implementaci GUI rozhraní, které je vytvořeno nad JavaFX verze `17.0.1`. Samotný program byl poté napsán nad vývojovým prostředím Intellij IDEA. Struktura projektu a následně i samotná implementace se tedy řídí především zvyklostmi tohoto vývojového prostředí a jazyka `Java`. Aplikace je určena pro operační systém Windows.

Zdrojový kód aplikace je členěn do balíků, kdy obsah balíků odpovídá nějaké části výše zmíněných modulů. Všechny balíky se poté nachází v balíku `cz.zcu.fav.kiv.mjakubas.saf`. Konkrétně je aplikace rozdělena na balíky odpovídající modulům:

- Modul CORE: skládá se z balíků:
  - `application`
  - `config`
  - `entity`
  - `io`
  - `parser`

  - `request`

- Modul CLI: skládá se z balíku `command` a také používá primárně funkcionality z balíku `parser`
- Modul GUI: skládá se z balíku `gui`.

Dále bude uveden popis všech balíků. Podrobnější popis každé třídy a metody balíků je k dispozici v programátorské dokumentaci `javadoc`.

5.1 Balík `application` a `Main`

Základem balíku je rozhraní `IApplication` obsahující metodu `launch`. Takové rozhraní poté používá program pro definici startovacích bodů rozhraní CLI a GUI. Logiku toho, které rozhraní se poté spustí je možné nalézt v hlavním startovacím bodu aplikace, a ve třídě `Main`. To rozhodne o typu rozhraní podle zadaných argumentů programu. Pokud je program spuštěn bez argumentů, spustí se GUI. V opačném případě se spustí CLI rozhraní.

Listing 5.1: Hlavní bod aplikace. Ukázka logiky přepínání rozhraní při startu programu.

```
public static void main(String[] args) {
    IApplication application = new GUIApplication();
    if (args.length != 0)
        application = new CLIApplication();
    application.start(args);
}
```

5.2 Balík `io`

Balík slouží primárně pro načítaní a vytváření souborů s daty entit. Balík také dále obsahuje implementaci metod pro práci se soubory potřebnými pro chod programu. Balík je rozdělen do čtyř podbalíků, podle specifické funkcionality načítání/vytváření souborů a obecnou práci s nimi.

5.2.1 Podbalík `io.file`

Obsahuje funkcionalitu potřebnou pro vykopírování složek a souborů do souborového systému operačního systému, ze složky vytvořené nástrojem Maven pro zdroje dat a konfigurační soubory nacházející se uvnitř `jar` programu.

Složka se jmenuje `resources`. Toto je zásadní funkce pro správu konfiguračních souborů, která byla popsána v analýze a její podrobnější implementace bude následovat při popisu její implementace. V podstatě nám tedy umožňuje vzít jakoukoliv složku/soubor ze složky `resources` a vykopírovat ji do jakéhokoliv jiného adresáře souborového systému operačního systému.

5.2.2 Podbalík `io.resource`

Obsahuje funkcionalitu pro získání cesty k souboru/složce v `resources` složce. `Resources` složka je interní složka vytvořená nástrojem Maven pro zdroje dat a konfigurační soubory. Cesta je poté reprezentována pomocí objektu `Path` standardní knihovny Javy `java.nio`.

5.2.3 Podbalík `io.properties`

Obsahuje funkcionalitu pro obsluhu objektu `Properties` ze standartní knihovny Java `java.util`, který reprezentuje `properties` soubory. Podbalík nám nejdříve umožňuje získat a načíst `properties` soubor a poté i získávat hodnoty klíčů z daného souboru. Specifičtější funkcionalitou je poté získávání typu souboru (XML nebo JSON) ze souboru. To nám slouží poté pro konfigurační soubor pro příkazy v příkazové řádce. Implementace poté funguje tak, že známe obecný prefix pro klíč obsahující definici typu formátu souboru. Jedná se o konstantu `QUERY_SUB_TYPE` ve třídě podbalíku `PropertiesIO`. K té poté připojíme konkrétní typ souboru a to zkontrolujeme oproti textovým řetězcům reprezentující konkrétní typ formátu souboru. Pokud se shodují vrátíme konkrétní typ souboru. Ten reprezentujeme strukturou `enum FileType`, kterou nalezneme v balíku `parser`.

5.2.4 Podbalík `io.export`

Obsahuje funkcionalitu pro vytváření a nahrávání souborů dat entit ze souborového systému operačního systému. Podbalík nám umožňuje vzít entitu, zadanou jako objekt a transformovat ji na řetězec, který následně uloží do souboru. Také nám umožňuje načtení entity z textového souboru, a to jak formátu XML, tak i JSON, do objektu. Pro svou funkcionalitu potřebuje funkce z balíku `parser`, a to konkrétně podbalík `mapper`, který bude popsán podrobněji v popisu implementace příslušného balíku. Pro popis funkcionality současného balíku si funkci toho balíku můžeme představit jako převodník XML nebo JSON entity do objektu a naopak.

5.3 Balík `parser`

Balík slouží pro analýzu, rozbor a následný převod textového řetězce, na jiný formát určený konkrétním analyzátorem textového řetězce. Balík obsahuje pravě takové analyzátory. Balík je rozdělen do čtyř podbalíků.

5.3.1 Podbalík `parser.filetype`

Obsahuje definici podporovaných formátů souborů s kterými umí aplikace pracovat. Pro potřeby aplikace jsou podporovány dva, a to XML a JSON. Pro potřeby aplikace se v podstatě jedná o definici rozhodování s jakým typem formátu konkrétní metoda, nebo část kódu pracuje, kdykoliv se pracuje s formáty.

Listing 5.2: `Enum` definující formáty souborů podporovaných programem.

```
public enum FileType {
    XML,
    JSON
}
```

5.3.2 Podbalík `parser.mapper`

Obsahuje funkcionalitu popsanou v analýze jako převodník typů datových formátu XML na JSON a naopak. Základem je rozhraní `IMapper`, které nám definuje všechny metody, které musí převodník obsahovat.

Listing 5.3: Ukázka rozhraní definující převodník datových formátů.

```
public interface IMapper {

    public <T> String serialize(T[] entities)
    throws MapperException;

    public <T> String serialize(T entity)
    throws MapperException;

    public <T> T[] deserialize(String serialization,
    Class<T> entitiesClass) throws MapperException;

    public <T> T deserializeOne(String serialization,
    Class<T> entitiesClass) throws MapperException;
}
```

Z ukázky převodníku 5.3 nám převodník zastává dvě funkce, a to:

- serializaci entit do textové podoby daného formátu.
- deserializaci textového řetězce do objektu z daného formátu.

Pro potřeby aplikace jsou poté definovány dva takové převodníky, a to `JSONMApper` pro JSON a `XMLMapper` pro XML.

Pro potřeby samotného parsování formátů byla použita knihovna `Jackson` [9], která nám umožňuje převést objekt buď do formátu JSON nebo XML, a naopak. Jediným požadavkem poté na takový objekt, který má být převáděn je, aby obsahoval pro všechny své atributy metody GET a SET a obsahoval prázdný konstruktor. Definice takových objektů se poté nachází v balíku `entity`.

5.3.3 Podbalík `parser.tokenizer`

Obsahuje funkcionalitu pro `CLI` rozhraní. Jedná se o převodník textového řetězce na definované tokeny. Kdy pro naše potřeby nabývá `token` buď hodnoty obsahující textový řetězec nebo konkrétní třídu objektu `Javy`. Tuto funkcionalitu využíváme při příkazu v příkazové řádce, který před dalším zpracováním převedeme na tokeny.

Listing 5.4: Ukázka `tokenizeru`.

```
public List<Token> tokenize(@NotNull String[] args)
    throws TokenException {
    List<Token> tokenList = new ArrayList<>();
    for (String arg : args) {
        try {
            if (isClassValue(arg)) {
                tokenList.add(
                    new Token(processClassValue(arg)));
                continue;
            }
            if (isFileTypeValue(arg)) {
                tokenList.add(
                    new Token(processFileTypeValue()));
                continue;
            }

            tokenList.add(
                new Token(processStringValue(arg)));
        } catch (Exception e) {
            throw new TokenException(e);
        }
```

```
        }
        return tokenList;
    }
```

Na ukázce kódu 5.4 lze vidět základní logiku tokenizeru, který rozhoduje o hodnotě tokenu. Pokud je hodnota třída objektu `Java` nebo hodnota definovaná pro formát, bude token obsahovat hodnotu dané třídy objektu, jinak je hodnota textový řetězec.

5.3.4 Podbalík `parser.args`

Obsahuje funkcionalitu pro rozhodování, který příkaz byl zadán v příkazové řádce. Pro tuto funkcionalitu potřebuje transformovat příkaz na tokeny. Tokeny poté porovná s očekávanými tokeny příkazu a jejich hodnot. Pokud se shodují, ujistí nás o shodě příkazu a jeho interpretace, které se shodují.

5.4 Balík `entity`

Balík obsahuje definici všech entit, jako třídy, s kterými pracuje back–end. Všechny takto definované entity splňují výše zmíněný požadavek parseru `mapper`. Pro potřeby GUI dále jsou obohaceny o dědění z třídy `ASimpleEnity`. Ta nám sjednocuje vracení `id` entity, které by jinak každá entita měla pojmenované jinak, tak aby se její podoba shodovala s back–endem, a definuje metody podobající se `toString` metodě objektu, s tím, že pokaždé plní odlišný popis entity. Na závěr také definuje porovnávání entit `equals`, a to jen podle jejich `id` vracené jako metoda `ASimpleEntity`.

5.5 Balík `config`

Obsahuje funkcionalitu potřebnou pro správu konfiguračních souborů. Umožňuje jejich vygenerováním a následné načítání. Zároveň je poté interně poskytuje ostatním konzumentům. Základem balíku je třída `ConfigManager`, která umožňuje načítání, vygenerování a následné poskytování všech konfiguračních souborů aplikace. Pro potřeby aplikace jsou definovány tři třídy představující vždy jeden konfigurační soubor. Jejich definici nalezneme v podbalíku `configs`. Jedná se o:

- Konfigurační soubor pro konfiguraci připojení. Jedná se o třídu `NetConfig`. Definuje nám všechny potřebné hodnoty pro odesílání HTTP požadavků na back-end.

- Konfigurační soubor pro konfiguraci příkazů příkazové řádky. Jedná se o třídu `CommandConfig`. Definuje nám tvar příkazů příkazové řádky, které je potřeba zadat pro správnou interpretaci příkazu, tj. správnou tokenizaci, která byl popsána výše.
- Konfigurační soubor pro konfiguraci již konkrétního připojení se k back–endu. Jedná se o třídu `RequestConfig`. Definuje již tedy konkrétní přeparsovanou definici připojení, tak aby jí bylo možné použít při odesílání HTTP požadavků.

Konfigurační soubor pro konfiguraci připojení a pro příkazy příkazové řádky jsou uživatelsky modifikovatelné. Znamená to tedy že využívají balíku `io` a tyto dva konfigurační soubory, které mají svou defaultní definic v složce `resource` se nejdříve vykopírují do dané složky a až z ní se načtou a použijí. Pokud by soubory již byly vykopírovány, nekopírují se znovu, jen v případě pokud neexistují v určené externí složce. Znamená to tedy, že pokud by uživatel převedl jeden z těchto konfiguračních souborů do chybového stavu, stačí jej smazat a aplikace si je sama vytvoří nový.

5.6 Balík `request`

Obsahuje funkcionalitu potřebnou pro odesílání veškerých HTTP požadavků, to znamená požadavky GET, POST, PUT a DELETE. Základem je zde třída `EntitiesRequests`, která obsahuje právě všechny tyto požadavky. Pro požadavek je nutné mít konfigurační třídu `RequestConfig`, která byla již popsána výše, konkrétní cestu požadavku a případné posílané entity, jako objekty. Výsledkem každého takového požadavku je poté třída `EntitiesRequestResult`, která uchovává stavový kód požadavku a případně i vracené entity požadavku. Definice samotné logiky příkazů je implementovaná v podbalíku `basicrequest`.

5.6.1 Podbalík `request.basicrequest`

V podbalíku se nachází obecná definice HTTP požadavků, bez návaznosti na entitu, nebo jiné specifikace dané programem. Požadavky používají pro implementaci standartní knihovnu `Javy java.net`. Definice požadavku je potom definována abstraktní třídou `ABasicHttpRequest`.

Listing 5.5: Abstraktní třída `ABasicHttpRequest`.

```
public HttpResponseResult doRequest(String body)
    throws RequestErrorException {
    HttpRequest request;
    try {
        request = buildRequest(body);
    } catch (NullPointerException e) {
        throw new RequestErrorException(e);
    }
    HttpResponse<String> response;
    try {
        response = HttpClient.newBuilder()
                .authenticator(authenticator)
                .build()
                .send(request,
                    HttpResponse
                    .BodyHandlers.ofString());
    } catch (IOException | InterruptedException e) {
        throw new RequestErrorException(e);
    }
    return new HttpResponseResult(
        response.statusCode(), response.body());
}
```

Implementaci hlavní metody dané abstraktní třídy můžeme vidět na ukázce kódu 5.5. Zde nalezneme základní strukturu příkazu, kde každý příkaz jen musí vytvořit svoje tělo požadavku pomocí metody `buildRequest`. Zbytek implementace používá standartní postup z knihovny `java.net`. Požadavek poté vrací výsledek obalený v třídě `HttpResponseResult`. Implementace této abstraktní třídy pro každý příkaz je poté v podbalíku `requests`.

5.7 Balík `command`

Obsahuje implementaci obsluhy všech příkazů příkazové řádky. Příkaz je zde reprezentován anotovanou metodou, `CommandHandler`.

Listing 5.6: Ukázka anotace.

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface CommandHandler {
    String[] value();
}
```

Anotace 5.6 obsahuje pole textových řetězců, které reprezentuje tokeny, které identifikují příkaz metody. Každá metoda, tedy obsahuje unikátní kombinaci takových tokenů. Všechny takto anotované metody reprezentované anotací, je možné najít v podbalíku `handler`. Ten obsahuje třídu pro každou entitu s právě takovými anotovanými metodami.

Listing 5.7: Ukázka anotované metody.

```
@CommandHandler({"{query.absences}", "{httpMethod.get}",
"[query.sub.type]", "<java.lang.String>"})
public void handleGetAll(FileType fileType, String filePath)
    throws CommandException {
    try {
        Absence[] absences = (Absence[]) EntitiesRequests
            .get(config,
                RequestConfig.ABSENCE_ENTITIES_QUERY,
                Absence.class).getDeserializedEntities();
        EntityExporter.export(
            absences, fileType, filePath);
    } catch (Exception e) {
        throw new CommandException(e);
    }
    System.out.println("Absences successfully queried!");
}
```

Pro správu všech takových anotovaných metod poté slouží `CommandManager`, který rozhoduje o tom který příkaz se vykoná. Pro zaregistrování příkazů poté slouží `CommandRegistration`, který anotaci převede na hashmapu, kterou už je možné procházet a vyhledávat v ní příkaz standardním způsobem pro hashmapy.

5.8 Balík `gui`

Obsahuje kompletní implementaci GUI rozhraní aplikace. GUI je postaveno nad `JavaFX` [10]. Pro stylizaci prvků byla zvolena knihovna `BoostrapFX` [2]. Základem celého GUI rozhraní je třída GUI, která slouží jako vstupní bod GUI aplikace, a to konkrétně metoda `start`. Implementace jednotlivých funkcionalit samotného GUI je poté rozdělena do podbalíků.

5.8.1 Podbalík `gui.i18n`

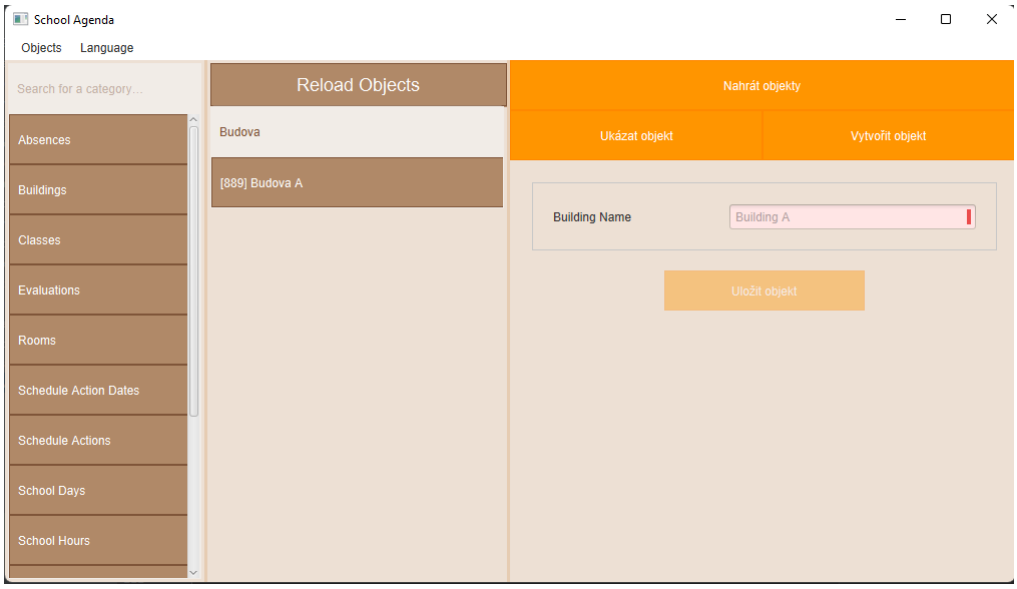
Obsahuje veškerou funkcionalitu pro lokalizaci textu aplikace. Implementací se jedná o `singleton`, který umožňuje bindovat lokalizovaný text z `Resource`

`bundle` definovaného v `resources` složce. Bundle obsahuje dvě lokalizace, a to českou a anglickou. Celá GUI aplikace tedy namísto pevných textových řetězců obsahuje klíče pro hodnoty překladu. Pokud by klíč neexistoval v lokalizaci, aplikace vypíše daný klíč a zaloguje chybu.

5.8.2 Podbalík `gui.view`

Obsahuje veškeré pohledy. Základem je zde rozhraní `ASearchableButtonPane`, které slouží jako základ pro vyhledávací seznam kategorií entit a vyhledávací seznam samotných entit. Seznam je zde implementován jako `VBox` tlačítek `ToggleButton`. Všechna taková tlačítka, která jsou v pohledu definována, se nacházejí v seznamu všech tlačítek. Filtrace je poté tvořena `InputFieldem`, který po zadání textu, projde seznam všech tlačítek, a podle jejich textu jej vyfiltruje a zobrazí jen odpovídající tlačítka.

Pro formuláře je zde poté třída `EntityFormsPane`, která se v základu skládá ze dvou tlačítek přepínající na dva různé pohledy, a to pohled pro vytváření a pohled pro modifikování entity. Následná implementace formulářů bude následovat později. Nad těmito dvěma tlačítky se poté nachází tlačítko pro import entit.



Obrázek 5.1: Ukázka implementace pohledů.

Na obrázku 5.1 můžeme vidět z levé strany dva sloupce seznamů tlačítek a na pravé straně potom pohled formulářů.

5.8.3 Podbalík gui.menu

Obsahuje menu aplikace, které se skládá z podmenu umožnující přepínaní lokalizace a podmenu pro import a spravování entit. Základem menu je poté abstraktní třída `AMenu`, které právě definují tyto dvě rozhraní. Samotné menu je poté sestaveno v GUI třídě rozhraní. Pohled menu lze poté vidět na horní liště na obrázku 5.1.

5.8.4 Podbalík gui.form

Skládá se ze dvou podbalíků.

- Podbalík nform. Obsahuje implementaci formulářů entit, pro které byla zvolena knihovna `FormsFX` [4]. Kostra každého formuláře entit je tvořena abstraktní třídou `AForm`, která umožňuje nadefinovat pro každou entitu formulář pro vytváření a modifikaci. Speciálně ještě může umožňovat vytvořit pod-formuláře těchto entit. Data formulářů potom definuje podbalík `nmodel`.
- Podbalík nmodel. Obsahuje implementaci modelů entit. Model slouží jak pro formuláře, tak i poté pro `controller`, který bude definován později. Základem každého modelu je poté rozhraní `IModel`, které definuje metody pro import a export entit z modelu. Každá entita poté obsahuje třídu, která implementuje toto rozhraní.

Listing 5.8: Ukázka části definice formuláře.

```
@Override
protected Form createPostForm
    (IModel<SchoolHour> entityModel) {
    SchoolHourModel model = (SchoolHourModel) entityModel;
        return Form.of(
            Group.of(
                Field.ofStringType(model
                .schoolHourNameProperty())
                    .label("form.label.school-hour-name")
                    .required("form.required")
                    .placeholder(
                    "form.placeholder.school-hour-name")
        )
    ).i18n(I18N.getLocalizationTool()
        .getResourceBundleService());
}
```

5.8.5 Podbalík gui.controller

Podbalík je rozdělen na `Controller`, pro jednotlivé kategorie a jednotlivé entity.

- Controller pro kategorie. Ovládá akce po stisku tlačítka pro kategorii entity.
- Controller pro entity. Ovládá veškeré dotazy na back–end. Každý `controller` entity zde poté dědí ze třídy `AEntityController`, která obecně definuje HTTP požadavky pro získávání entit a jejich následnou manipulaci. Definuje tedy pro každou kategorii entit GET, PUT, POST a DELETE dotazy. Každý `controller` také obsahuje buffer, který si ukládá data entit, tak aby se při dalším stisku tlačítka znovu nenahrávali z back–endu. Protože pro správné zobrazení a modifikaci entit jsou potřeba `id` entit, které ale nejsou vizuálně vhodné pro GUI aplikaci, je zde k dispozici třída `EntitiesController`, která spravuje všechny `controllery` a umožňuje jejich vzájemný přístup. To znamená, že jeden stisk tlačítka kategorie může vyvolat akci vícero `controllerů`. Pokud dojde k vyvolání příkazu DELETE, POST nebo PUT, dojde k smazání všech bufferů a entity se po dalším stisku kategorie nahrají znovu.

5.8.6 Podbalík gui.alert

Obsahuje implementace alertů `JavaFX` [10] pro stavové kódy při odeslání formulářů, jak pro modifikaci, tak i vytváření entit.

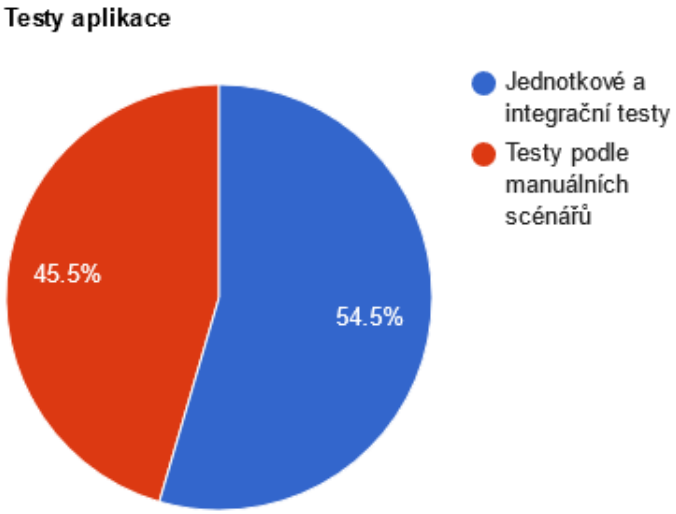
6 Testování

Protože bakalářská práce má sloužit jako nástroj pro benchmark testovacích nástrojů, je potřeba aby, aplikace obsahovala co nejmenší počet chyb. To nám zajistí její rozumnou použitelnost pro benchmark testovacích nástrojů. Aplikace jako taková je tedy otestována větším počtem testů a vícero druhů testů. Tím se má zajistit co nejmenší počet chyb výsledné aplikace, a tím pádem lepší kvalita pro benchmark testovacích nástrojů. Testy samotné aplikace lze poté řadit do čtyř základních druhů, a to:

- Unit testy. Cílem těchto testů bylo otestovat jednotkové části kódu, tj. nejmenší funkcionální úseky kódu. Kvůli rozdělení implementace kódu se tedy jedná primárně o testy jednotlivých metod.
- Integrační testy. Tento druh testů primárně sloužil pro ověření správné komunikace mezi back-endem a front-endem. Testy se tedy primárně zaobírali HTTP požadavky (GET, POST, PUT, DELETE) a jejich integrací s požadavky jednotlivých entit.
- Testy podle testovacích scénářů. Testovací scénáře zde byly využity hlavně pro složitě testovatelné funkcionality programu, pro které byla nejjednodušší ruční kontrola nebo ruční porovnání výsledků. V podstatě se jednalo o celé testování GUI pro které byly použité výhradně scénáře. Poté byla za pomoci scénářů nahrazena část integračních testů pro CLI rozhraní. Pro správnost HTTP dotazů byl použit program `Postman` [14], který umožnuje zasílání HTTP dotazů pomocí REST rozhraní a přehledné zobrazení jejich výsledků.
- Testy pro podpůrné funkcionality testů. Protože aplikace pro otestování potřebovala základní data a naplněnou databázi, kde data mohla být mazána, upravována a vytvářena bez žádných následků pro ostatní testy, obsahuje aplikace funkcionalitu pro tyto účely. V podstatě obsahují veškeré funkce pro tvorbu dat databáze a jejich spravování. Všechny tyto funkcionality byly poté otestovány vlastními testy.

6.1 Statistika testování

Celkové byla aplikace otestována `524` testy (jednotkové a integrační testy), a `437` manuálními scénaři. Manuální scénáře jsou dále rozděleny na scénáře



Obrázek 6.1: Procentuální podíl všech testů aplikace podle typu.

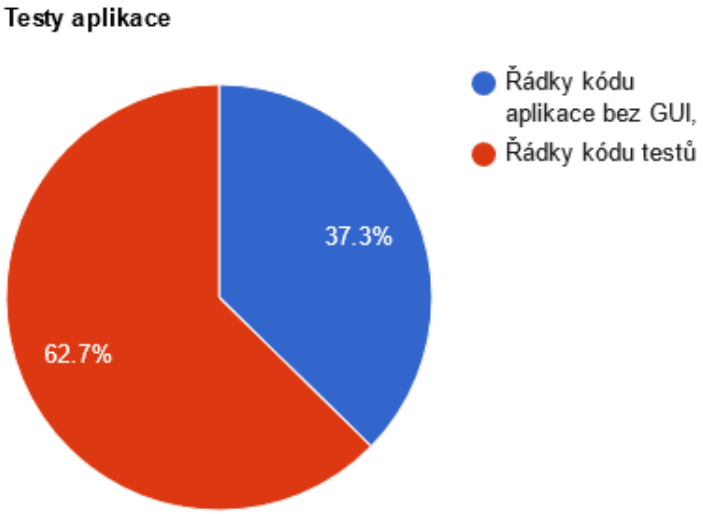
pro GUI (184 scénářů) a CLI (253) scénářů. Jak můžeme vidět na grafu obrázku 6.1 celkové aplikace obsahuje `961` testů, z toho `45,5%` tvoří testy podle manuálních scénářů a `54,5%` tvoří jenotkové a integrační testy.

Celkově bylo napsáno 6225 řádek kódu pro jednotkové a integrační testy, které měli za úkol pokrýt 3709 řádek kódu aplikace (bez GUI). Jak můžeme vidět na grafu obrázku 6.2 celkově s testy (bez GUI) obsahuje aplikace dohromady 9934 řádek kódu, kde je $62,7\%$ testů a $37,3\%$ kódu aplikace.

Tabulka 6.1: Statistika pokrytí jednotkovými a integračními testy

Pokrytí tříd	Pokrytí metod	Pokrytí řádek kódu
90%	68%	46%

Jak můžeme pozorovat na tabulce 6.1 celkově bylo jednotkovými a integračními testy pokryto 90% tříd, 68% metod a 46% řádek kódu. Zbylé pokrytí bylo dosaženo právě manuálními scénáři (253 scénářů). Do této statistiky nepočítáme testy GUI, ty jsou jen otestovány pomocí manuálních scénářů (184).



Obrázek 6.2: Procentuální podíl řádek kódu aplikace a testů, bez GUI.

6.2 Statistika testování jednotlivých balíků

V sekci je popsána podrobněji statistika testů jednotlivcích balíků.

6.2.1 Testování balíku application a třídy `Main`

Zde nebylo využito jednotkových testů ani testů integračních. Jedná se jen o definici spouštění aplikace. Ta byla manuálně otestována při testu spouštění aplikace v režimu CLI nebo GUI rozhraní.

6.2.2 Testování balíku `request`

Balík `request` je otestován jednotkovými a integračními testy. Testy se především zaměřují na správné připojení a komunikaci s back-endem. Balík sestává z `609` řádek kódu. Pro testování se kód skládá z `1630` řádek kódu. Celkem bylo provedeno `144` testů. Statistika pokrytí poté vyobrazena tabulkou 6.2. Tabulka pokrytí podbalíků je poté vyobrazena tabulkou 6.3.

Testy odhalily nedostatky v logice vracení výsledků HTTP požadavků. Bylo potřeba provést refaktorizaci vracení výsledků jednotlivých požadavků.

Tabulka 6.2: Tabulka statistik pokrytí balíku `request`

Pokrytí tříd	Pokrytí metod	Pokrytí řádek kódu
100%	93%	90%

Tabulka 6.3: Tabulka statistiky pokrytí podbalíků balíku `request`

Podbalík	Pokrytí tříd	Pokrytí metod	Pokrytí řádek kódu
request.basicrequest	100%	88%	90%

Ten se zvolil jako postup vyhození výjimky pokud je kód vraceného požadavku jiný než kód 200. Tím se zajistilo konzistentní chování požadavků.

6.2.3 Testování balíku `parser`

Balík `parser` je otestován jednotkovými testy. Jednotkové testy se primárně zaměřují na správnou funkčnost parserů. Balík sestává z `575` řádek kódu. Pro testování se kód skládá z `788` řádek kódu. Celkem bylo provedeno `63` testů. Statistika pokrytí poté vyobrazena tabulkou 6.4. Tabulka pokrytí podbalíků je poté vyobrazena tabulkou 6.5.

Tabulka 6.4: Tabulka statistik pokrytí balíku `parser`

Pokrytí tříd	Pokrytí metod	Pokrytí řádek kódu
100%	94%	89%

Tabulka 6.5: Tabulka statistky pokrytí podbalíků balíku `parser`

Podbalík	Pokrytí tříd	Pokrytí metod	Pokrytí řádek kódu
parser.args	100%	100%	94%
parser.filetype	100%	100%	100%
parser.mapper	100%	83%	76%
parser.tokenizer	100%	100%	100%

Testy odhalily nedostatky mapování objektu na XML nebo JSON, a to ten, že se serializovaly i nulové atributy entit. To způsobilo zavádějící a nepřehledné výpisy. Pro opravení byly použity prostředky knihovny `Jackson`.

6.2.4 Testování balíku `io`

Balík `io` je otestován jednotkovými testy. Jednotkové testy se primárně zaměřují na správný export a import entity a dále na bezchybné IO operace.

Balík sestává z 269 řádek kódu. Pro testování se kód skládá z 336 řádek kódu. Celkem bylo provedeno 30 testů. Statistika pokrytí poté vyobrazena tabulkou 6.6. Tabulka pokrytí podbalíků je poté vyobrazena tabulkou 6.7.

Tabulka 6.6: Tabulka statistik pokrytí balíku `io`

Pokrytí tříd	Pokrytí metod	Pokrytí řádek kódu
100%	94%	86%

Tabulka 6.7: Tabulka statistiky pokrytí podbalíků balíku `io`

Podbalík	Pokrytí tříd	Pokrytí metod	Pokrytí řádek kódu
io.export	100%	90%	82%
io.file	100%	100%	85%
io.properties	100%	100%	86%
io.resource	100%	100%	100%

Zde jednotkové testy neodhalily žádnou chybu, ale poté testování spuštění aplikace odhalilo chybu v čtení souborů z `resource` složky uvnitř `jar` souboru. Pro správnou funkčnost bylo potřeba refaktorovat čtení těchto souborů, a to z postupu:

```
this.class.getClassLoader().getResource(<cesta k souboru>)
```

na postup:

```
this.class.getClassLoader().getResourceAsStream(<cesta k
souboru>)
```

Tím se opravilo načítání souborů z `resource` složky. Po opravení byly testy upraveny a provedeny znovu.

6.2.5 Testování balíku `gui`

Pro provedení testů nebyly využity jednotkové ani integrační testy, ale testy pomocí scénářů. Celkově bylo pro vedeno pro otestování `gui` balíku, a tím i celého GUI `184` testů. Popis testů se nachází v příloze D. Jednalo se o testy pro lokalizaci, správnou funkčnost tlačítek, formulářů a prací s formuláři.

Testy odhalily:

- Chybějící lokalizace. Některým textovým polím chyběla lokalizace. Ta byla doplněna a testy zopakovány.

- Chybějící tlačítko kategorie entit pro budovy. Ta byla doplněna a příslušné testy byly zopakovány.
- Celkově nekorektní výpisy entit. Ty byly pozměněny a příslušné testy zopakovány.
- Chybějící kontrolu správného zadávání hodnot do formulářů. Ta byla doplněna a příslušné testy zopakovány.
- Špatné nahrávání a editace některých entit. To bylo opraveno a příslušné testy byly zopakovány.
- Špatná reakce formulářů na změnu lokalizace. Tato chyba vzhledem ke své netriviální opravě nebyla opravena. Pro změnu lokalizace formulářů je potřeba znovu kliknout na entitu/kategorii a tím vyvolat změnu lokalizace formuláře.
- Nefunkční nahrávání vícero-entit. To bylo nahrazeno a příslušné testy zopakovány.

6.2.6 Testování balíku `entity`

Balík se skládá jen z definice entit, tudíž testy zde nebyli potřeba. Třídy entit jsou poté použity v celém programu a v podstatě úspěšné splnění ostatních testů znamená, že i definice těchto tříd entit je v pořádku.

6.2.7 Testování balíku `config`

Balík `config` je otestován jednotkovými testy. Jednotkové testy se primárně zaměřují na správný tvar a získávání konfiguračních souborů. Balík sestává z `282` řádek kódu. Pro testování se kód skládá z `502` řádek kódu. Celkem bylo provedeno `63` testů. Statistika pokrytí poté vyobrazena tabulkou 6.8. Tabulka pokrytí podbalíků je poté vyobrazena tabulkou 6.9.

Tabulka 6.8: Tabulka statistik pokrytí balíku `config`

Pokrytí tříd	Pokrytí metod	Pokrytí řádek kódu
100%	100%	96%

Chyby odhalily špatné cesty pro komunikaci s back-endem pomocí REST API. Chyby také odhalily špatně nastavené počáteční značky CLI příkazů. Chyby byly opraveny a testy zopakovány.

Tabulka 6.9: Tabulka statistiky pokrytí podbalíků balíku `config`

Podbalík	Pokrytí tříd	Pokrytí metod	Pokrytí řádek kódu
config.configs	100%	100%	100%

6.2.8 Testování balíku `command`

Balík `command` je otestován pomocí integrační testů a testů pomocí scénářů. Integrační testy jsou použity pro testování HTTP požadavků POST, DELETE a základní GET pro všechny entity. Pro požadavky PUT a zbylé GET byly využity testy podle scénářů. Testy podle scénářů byly využity i pro ověření integračních testů. Balík je tedy spíše otestován testy podle scénářů. Balík sestává z `2583` řádek kódu. Pro testování se kód skládá z `2255` řádek kódu a `253` scénářů. Celkem zde bylo provedeno `182` testů. Popis scénářů se nachází v příloze C. Statistika pokrytí poté vyobrazena tabulkou 6.10.

Tabulka 6.10: Tabulka statistik pokrytí balíku `command`

Pokrytí tříd	Pokrytí metod	Pokrytí řádek kódu
83%	33%	24%

Jak je možné zpozorovat pokrytí testy je zde malé. Pro zbylé pokrytí bylo právě použitu `253` ověřovacích testů.

Odhalené chyby byly dvojího typu.

- Chyba použité HTTP metody u testu. Všechny takové chyby byly opraveny a příslušné testy zopakovány.
- Chyba přístupového bodu HTTP požadavku. Všechny takové chyby byly opraveny a příslušné testy zopakovány.

7 Závěr

V teoretické části této práce byla prostudována studijní agenda základní školy a požadavky, které musí splňovat, tak aby se dala považovat za studijní agendu. Byl prozkoumám zákon č. 561/2004 Sb., o předškolním, základním, středním, vyšším odborným a jiném vzdělání, a zákon č. 364/2005 Sb., o vedení dokumentace škol a školských zařízení, kterými se studijní agenda musí řídit. Dále se práce zabývala daty, který si aplikace musí uchovávat a daty back-endu. Byly ukázány i již existující studijní agendy.

Dále se práce zabývala webovými službami, kdy bylo vysvětleno, co je to webová služba. Poté následovaly ukázky existujících protokolů pro webové služby, a to SOAP a REST, které byly i podrobně popsány. Následoval popis formátů přenášených zpráv, a to konkrétně formátem XML a JSON. Ty byly také porovnány.

V praktické části práce byla popsána analýza návrhu aplikace. V analýze byly zmíněni všechny požadavky na aplikaci, poté byl uveden graf případu užití, a dále byly zmíněny všechny části aplikace, které budou potřeba. zmíněné části aplikace se skládaly z funkcionalit pro CLI a GUI rozhraní a poté ze samotného CLI a GUI rozhraní.

Následoval popis implementace. V popisu implementace programu v jazyce Java byla popsána struktura aplikace a struktura jednotlivých balíků. Je zde také popis komunikace aplikace s back-endem.

V poslední praktické části bylo zhodnocení testováni, kdy byly vyjmenovány jednotlivé druhy testů a jejich zhodnocení.

Výsledkem této bakalářské práce je funkční front-endová aplikace, které poskytuje veškerou funkcionalitu back-endového rozhraní pomocí REST API. Front-endové rozhraní je buď ve tvaru CLI nebo GUI. Dále se od aplikace očekává její použití pro benchmark testovacích nástrojů.

Literatura

[1] *Bakaláři* [online]. [cit. 4.6.2022]. Dostupné z: https://www.bakalari.cz/Home/Modules.

[2] *BootstrapFX* [online]. [cit. 4.6.2022]. Dostupné z: https://github.com/kordamp/bootstrapfx.

[3] Dub, M. *Aplikace pro studijní agendu základní školy s webovou službou.* Západočeská univerzita v Plzni, 2021.

[4] *FormsFX* [online]. [cit. 4.6.2022]. Dostupné z: https://github.com/dlsc-software-consulting-gmbh/FormsFX.

[5] Gupta, L. *What is REST* [online]. 27.8.2021. [cit. 26.1.2022]. Dostupné z: https://restfulapi.net/.

[6] *Ukázka HTML* [online]. [cit. 4.6.2022]. Dostupné z: https://www.w3schools.com/html/html_basic.asp.

[7] Hughes, A. *The Ins and Outs of a Service-Oriented Architecture (SOA)* [online]. cleo.com. [cit. 26.1.2022]. Dostupné z: https://www.cleo.com/blog/knowledge-base-soa-service-oriented-architecture.

[8] *Introducing JSON* [online]. [cit. 11.1.2022]. Dostupné z: https://www.json.org/json-en.html.

[9] *Jackson* [online]. [cit. 4.6.2022]. Dostupné z: https://github.com/FasterXML/jackson.

[10] *JavaFX* [online]. [cit. 4.6.2022]. Dostupné z: https://openjfx.io/.

[11] *Ukázka JSON* [online]. [cit. 4.6.2022]. Dostupné z: https://www.guru99.com/json-tutorial-example.html.

[12] Kopecká, K. *Analýza využití školního informačního systému Bakaláři na základních školách v Brně.* Masarykova univerzita, 2016.

[13] Neumajer, O. *Školní informační systémy* [online]. rvp.cz, 17.3.2010. [cit. 10.1.2022]. Dostupné z: https://clanky.rvp.cz/clanek/c/Z/8019/skolni-informacni-systemy.html.

[14] *program Postman* [online]. [cit. 20.6.2022]. Dostupné z: https://www.postman.com/.

[15] Safris, S. *A Deep Look at JSON vs. XML, Part 1: The History of Each Standard* [online]. toptal.com. [cit. 11.1.2022]. Dostupné z: https://www.toptal.com/web/json-vs-xml-part-1.

[16] Safris, S. *A Deep Look At JSON vs. XML, Part 2: The Strengths and Weaknesses of Both* [online]. toptal.com. [cit. 15.1.2022]. Dostupné z: https://www.toptal.com/web/json-vs-xml-part-2.

[17] Safris, S. *A Deep Look at JSON vs. XML, Part 3: XML and the Future of JSON* [online]. toptal.com. [cit. 15.1.2022]. Dostupné z: https://www.toptal.com/web/json-vs-xml-part-3.

[18] *Simple Object Access Protocol (SOAP) 1.1* [online]. World Wide Web Consortium, 8.5.2000. [cit. 24.1.2022]. Dostupné z: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383486.

[19] *SOAP* [online]. IBM, 16.5.2021. [cit. 26.1.2022]. Dostupné z: https://www.ibm.com/docs/en/wasdtfe?topic=applications-soap.

[20] *Ukázka SOAP protokolu* [online]. [cit. 4.6.2022]. Dostupné z: https://www.tutorialspoint.com/soap/soap_examples.htm.

[21] *Stejnopisy Sbírky zákonů* [online]. Ministerstvo vnitra České republiky, 17.3.2010. [cit. 10.1.2022]. Dostupné z: https://www.mvcr.cz/clanek/sbirka-zakonu.aspx.

[22] *Uniform Resource Identifier (URI)* [online]. 15.1.2005. [cit. 26.1.2022]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc3986#section-1.1.

[23] *What are Web Services?* [online]. geeksforgeeks.org, 14.6.2021. [cit. 24.1.2022]. Dostupné z: https://www.geeksforgeeks.org/what-are-web-services.

[24] *What Are Web Services?* [online]. cleo.com. [cit. 24.1.2022]. Dostupné z: https://www.cleo.com/blog/knowledge-base-web-services.

[25] *Working with JSON* [online]. Mozilla Corporation. [cit. 15.1.2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON.

[26] *Ukázka XML* [online]. [cit. 4.6.2022]. Dostupné z: https://www.javatpoint.com/xml-example.

[27] *XML introduction* [online]. Mozilla Corporation. [cit. 11.1.2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction.

[28] *XML Technology* [online]. World Wide Web Consortium. [cit. 11.1.2022]. Dostupné z: `https://www.w3.org/standards/xml/core`.

Přílohy

A Uživatelská příručka

Sestavení programu

Sestavení programu bylo otestováno pod systémem `Windows 11`. Pro sestavení programu je potřeba mít nastavené a nainstalované `Oracle OpenJDK` verze `17.0.2` a `Maven` verze `3.8.4`.

- Oracle JDK lze stáhnout zde:

  `https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html`

- Maven lze stáhnout zde:

  `https://archive.apache.org/dist/maven/maven-3/3.8.4`

Sestavení samotné lze poté uskutečnit pomocí příkazové řádky následovně:

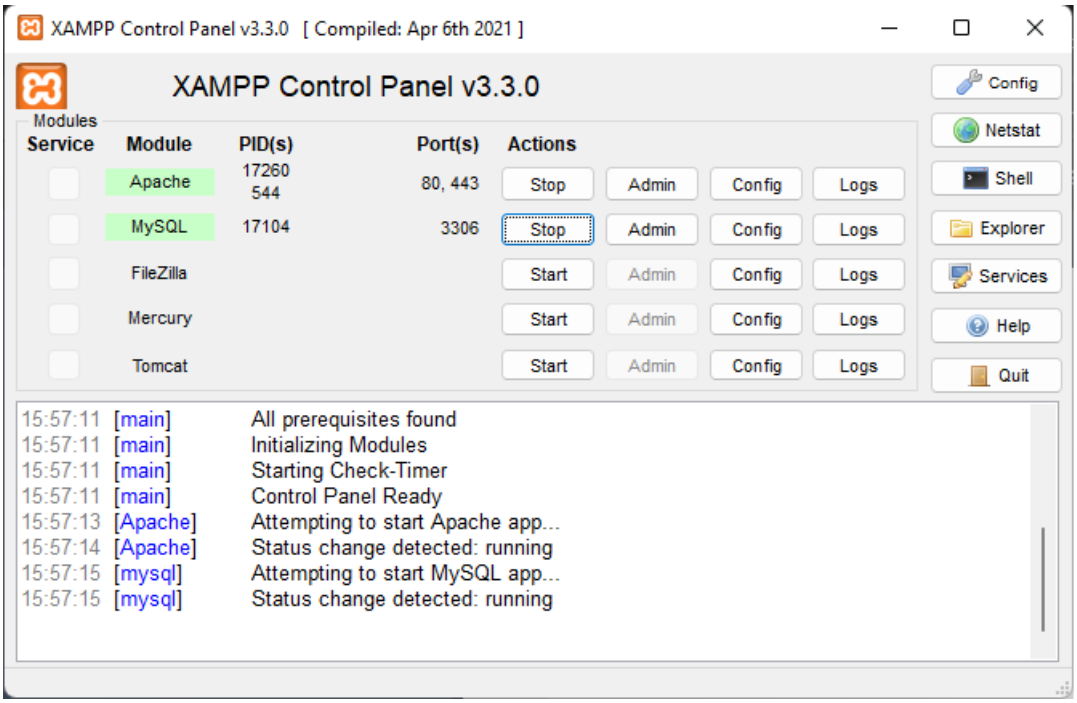
- V příkazové řádce je potřeba přepnout do adresáře se zdrojovými soubory a `pom.xml` souborem (`Aplikace_a _knihovny`/Zdrojove_soubory).
- Zadáme příkaz: `mvn install clean`

Pokud je vše správně nastaveno a nainstalováno vytvoří se ve složce `Aplikace_a_knihovny/Aplikace jar` soubor `school_agenda.jar`.

Back-end

Aplikace pro svou funkčnost potřebuje funkční back-end. Pokud je potřeba back-end spustit lokálně je zapotřebí nainstalovat si program `xampp`. Ten je dostupný zde: https://www.apachefriends.org/index.html. Po nainstalování programu je potřeba spustit moduly webového serveru `Apache` a databáze `MySQL`. Zapnuté moduly lze vidět na obrázku A.1.

Po zapnutí modulů je potřeba spustit administrátorské prostředí `MySQL` modulu. To provedeme kliknutím na tlačítko `Admin` u příslušného modulu. Zde vytvoříme novou databázi s názvem `schoolagenda` s kódováním `utf_-czech_ci`. Tu můžeme vytvořit pod tlačítkem `Databáze`. Pro přesný postup nastavení databáze je možné zkonzultovat návod obsažený v bakalářské práci back-endu [3]. Pokud chceme vložit i testovací data klikneme na tlačítko `Import` a naimportujeme zde do databáze soubor `db+data.sql`, který nalezneme ve složce `Vstupni_data`.



Obrázek A.1: Zapnuté moduly programu `xampp`

Před spuštěním aplikace

Než spustíme samotnou aplikace je dobré zkontrolovat konfiguraci připojení k back-endu. Ta se nachází ve složce `Aplikace_a_knihovny/Aplikace/config`, a to jako soubor `config.net.properties`. Zde můžeme nastavit podrobnosti připojení, a to přesně adresu, port a přihlašovací údaje. Soubor lze vidět na obrázku A.3. Pokud konfigurační soubor neexistuje postupujte na další sekci. Pokud se připojujete k back-endu lokálně, a beze změny back-endu, konfigurační soubor ponechejte tak jak je. V případě chybné editace, nebo rozbitého souboru, soubor smažte a postupujte na další sekci.

```
1  # Net configuration file
2  #
3  # Backend API address
4  address=http://localhost
5  # Backend API port
6  port=8080
7  # Backend API login
8  username=user
9  password=password
```

Obrázek A.2: Konfigurační souboru pro připojení k back-endu

Spuštění aplikace

Aplikaci lze spustit ve dvou provedeních, a to jako:

- Aplikací s CLI rozhraním.
- Aplikací s GUI rozhraním.

Před spuštěním aplikace zkontrolujte zda se ve stejné složce vyskytuje složka `configs` se soubory `config.cmd.properties` a `config.net.properties`. Aplikace se nachází ve složce `Aplikace_a_knihovny/Aplikace/`. Pokud zde aplikaci nevidíte přejděte na sekci A, a aplikaci sestavte. Pokud zde nevidíte výše zmíněné konfigurační soubory, pokračujte na další sekci, pokud je zde máte, následující sekci můžete přeskočit.

Vygenerování konfiguračních souborů

Pokud v jakékoliv fázi aplikace přijdete o konfigurační soubory nebo se nachází v chybném stavu, bude zapotřebí je znovu vygenerovat. Pokud jsou soubory chybné, smažte je. Poté zadejte příkaz ve složce `Aplikace_a_knihovny/Aplikace/`:

```
java -jar school_agenda.jar
```

Tím se spustí GUI rozhraní aplikace. S tím, ale nepotřebujeme pracovat. Po spuštění aplikaci zavřete. Tím se vygenerují nové konfigurační soubory, které můžete upravit.

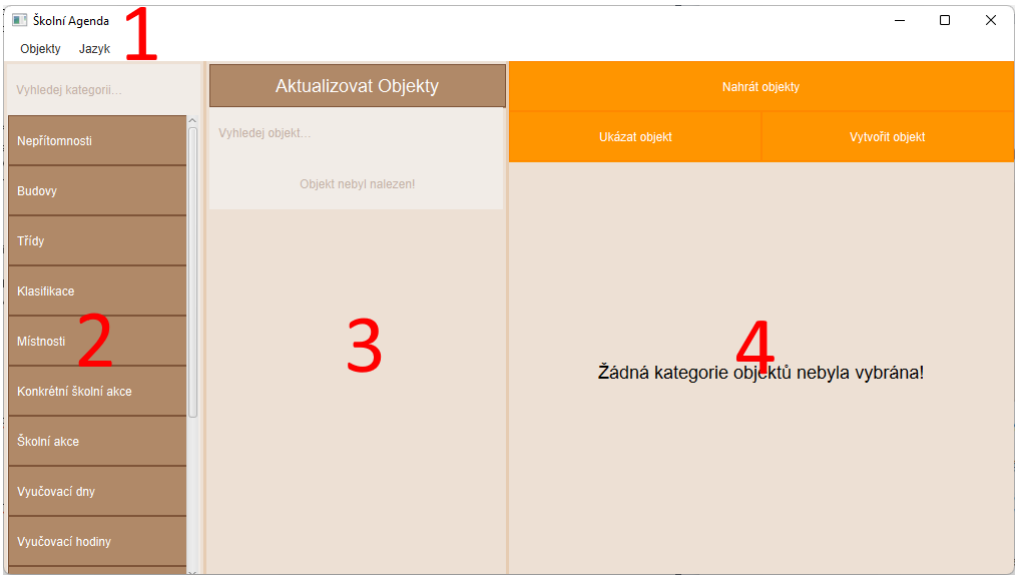
Spuštění a ovládání CLI rozhraní

CLI rozhraní je možné spustit přes příkazovou řádku. Aplikaci najdeme ve složce `Aplikace_a_knihovny/Aplikace/`, a to v podobě **jar** souboru. Obecně CLI rozhraní spustíme příkazem ve formátu:

```
java -jar school_agenda.jar [příkaz]
```

Příkazy a jejich popis poté najdeme v příloze B. Pro ukázky vkládání a modifikování entit je možné využít ukázkové soubory ve složce `Vstupni_data/Entity/`. Po zadání příkazu nám rozhraní vrátí informaci o výsledku operace.

Pro editaci tvaru příkazů je možné využít konfigurační soubor `config.cmd.properties`, kde je možno modifikovat tvar značek příkazů, tj. z vlajky pro označení XML souboru `-XML` je možné udělat vlajku `-x`.



Obrázek A.3: Pohled na aplikaci po spuštění.

Spuštění a ovládání GUI rozhraní

GUI rozhraní je možné spustit přes příkazovou řádku. Aplikaci najdeme ve složce `Aplikace_a_knihovny/Aplikace/`, a to v podobě **jar** souboru. GUI rozhraní spustíme příkazem:

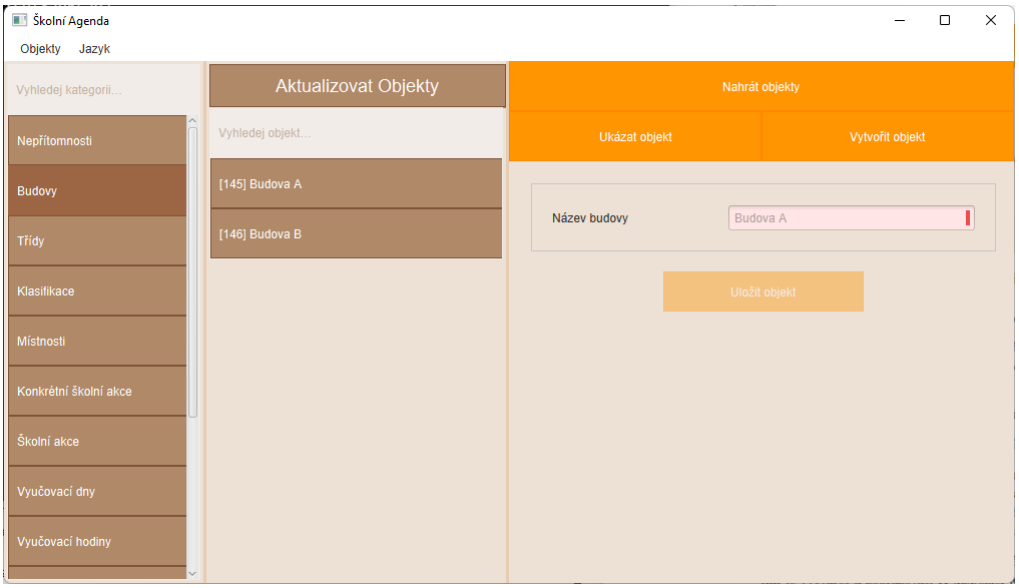
```
java -jar school_agenda.jar
```

GUI aplikace se skládá ze čtyř hlavních ovladacích prvků.

- Menu aplikace. Přes menu aplikace je možné změnit lokalizaci aplikace - menu `Jazyk`. Také je zde možné nahrát entity na back-end - `Objekty/Nahrát objekty`. Ukázkový soubor pro nahrávání entit lze nalézt ve složce `Vstupni_data/Entity/Multi/`.

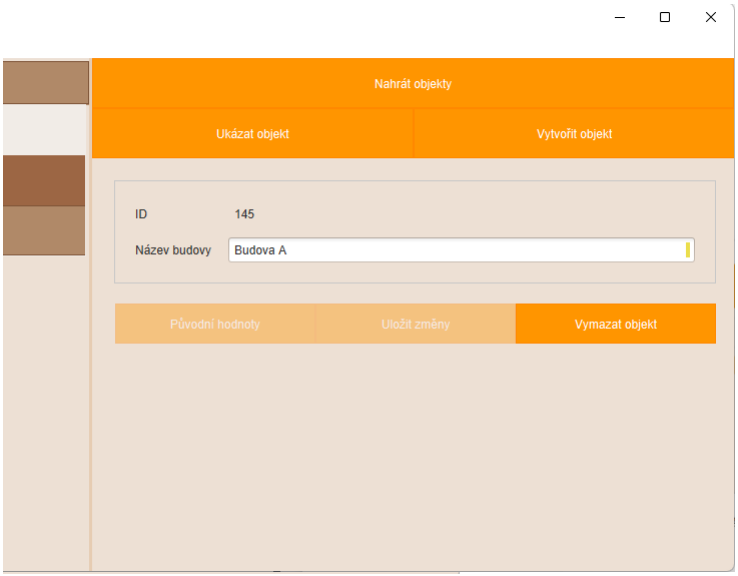
- Panel kategorií entit. Přes panel je možné vybírat kategorie entit, a tím je vytvářet a zobrazovat. K dispozici je také vyhledávací pole pro rychlé nalezení příslušné kategorie. Výběr kategorie ovlivňuje zbylé panely aplikace.

- Panel pro výběr konkrétní entity. Konkrétní tvar po výběru kategorii budov můžeme vidět na obrázku A.4. K rychlému výběru a nebo k filtrování entit je k dispozici vyhledávací pole nad seznamem entit. Pokud dojde k externí změně databáze, je možné vyvolat změnu a entity získat znovu tlačítkem `Aktualizovat Objekty`. Po výběru entity je možná i její editace na panelu pravém panelu, který lze vidět na obrázku A.5.



Obrázek A.4: Aplikace po vybrání kategorie budov.

- Panel pro vytváření a editaci entit. Mezi vytvářením a editací můžeme přepínat pomocí dvou tlačítek na panelu, a to tlačítka `Ukázat objekt` (editace entity) a `Vytvořit objekt` (vytvořit entitu). Pro vytvoření je nutné vyplnit příslušní formulář a kliknout na tlačítko `Uložit objekt`. Aplikace zobrazí výsledek vytvoření entity. Pro editaci je možné změnit jen určitou část formuláře. Zde jsou nám k dispozici tři tlačítka, a to tlačítko do vrácení původního stavu – `Původní hodnoty`, tlačítko pro odeslání editace – `Uložit změny` a tlačítko pro smazání entity – `Vymazat objekt`. O výsledku smazání a editace nás aplikace informuje. Posledním dostupným tlačítkem na panelu je tlačítko `Nahrát objekty`. To nám umožní nahrát libovolné entity, a o výsledku operace nás poté informuje. Ukázkový soubor pro nahrávání entit lze nalézt ve složce `Vstupni_data/entity/multi/`.



Obrázek A.5: Pohled na část aplikace s editací entity.

B Seznam všech příkazů CLI rozhraní pro defaultní značky

Následující stránky obsahují seznam všech možných příkazů pro nezměněné definici značek příkazů.

Příkazy pro absence

1. `absences --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů absencí.

2. `absences --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam absence s id.

3. `absences --get --student <id> --xml/--json <cesta k souboru>`
   Vrací seznam záznamů absencí podle studenta id.

4. `absences --get --student --name <jméno> --surname <příjmení> --xml/--json <cesta k souboru>`
   Vrací seznam záznamů absencí podle zadaného křestního a příjmení studenta.

5. `absences --get --scheduleActionDate <id> --xml/--json <cesta k souboru>`
   Vrací seznam záznamů absencí podle konkrétní rozvrhové akce id.

6. `absences --post --xml/--json <cesta k souboru>`
   Vytvoří záznam absence/absencí na základě parametrů v požadavku.

7. `absences --put <id> --xml/--json <cesta k souboru>`
   Upraví záznam absence s id, záznam je upraven podle parametrů v požadavku.

8. `absences --delete <id>`
   Odstraní záznam absence s id.

Příkazy pro budovy

1. `buildings --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů budov.

2. `buildings --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam budovy s id.

3. `buildings --get --name <jméno> --xml/--json <cesta k souboru>`
   Vrací seznam záznamů budov podle jména.

4. `buildings --get --room <id> --xml/--json <cesta k souboru>`
   Vrací záznam budovy podle id místnosti.

5. `buildings --post --xml/--json <cesta k souboru>`
   Vytvoří záznam budovy/budov na základě parametrů v požadavku.

6. `buildings --put <id> --xml/--json <cesta k souboru>`
   Upraví záznam budovy s id, záznam je upraven podle parametrů v požadavku.

7. `buildings --delete <id>`
   Odstraní záznam budovy s id.

Příkazy pro třídy

1. `classes --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů tříd.

2. `classes --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam třídy s id.

3. `classes --get --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů tříd podle jména.

4. `classes --get --student <id>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam třídy studentova id.

5. `classes --get`
   `--student --name <jméno> --surname <příjmení>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam třídy podle křestního jména a příjmení studenta.

6. `classes --get --teacher <id>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam třídy podle id učitele.

7. `classes --get --teacher`
   `--name <jméno> --surname <příjmení>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam třídy podle jména a příjmení učitele.

8. `classes --get --room <id> --xml/--json <cesta k souboru>`
   Vrací záznam třídy podle id místnosti.

9. `classes --get --room --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam třídy podle jména místnosti.

10. `classes --get --subject <id>`
    `--xml/--json <cesta k souboru>`
    Vrací seznam záznamů tříd podle id konkrétního předmětu.

11. `classes --get --subject --name <jméno>`
    `--xml/--json <cesta k souboru>`
    Vrací seznam záznamů tříd podle jména konkrétního předmětu.

12. `classes --post --xml/--json <cesta k souboru>`
    Vytvoří záznam třídy/tříd na základě parametrů v požadavku.

13. `classes --put <id> --xml/--json <cesta k souboru>`
    Upraví záznam třídy s id, záznam je upraven podle parametrů v požadavku.

14. `classes --put <id> --subject add`
    `--xml/--json <cesta k souboru>`
    Přidá třídě o id konkrétní předmět, který je uveden v těle požadavku.

15. `classes --put <id> --subject remove`
    `--xml/--json <cesta k souboru>`
    Odstraní z třídy o id, konkrétní předmět, který je uveden v těle požadavku.

16. `classes --delete <id> --xml/--json <cesta k souboru>`
    Odstraní záznam třídy s id.

Příkazy pro klasifikaci

1. `evaluations --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů hodnocení.

2. `evaluations --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam hodnocení s id.

3. `evaluations --get --studentsSubjects <hodnota>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam hodnocení podle id předmětu studenta.

4. `evaluations --get --studentsSubjects <hodnota> average`
   `--xml/--json <cesta k souboru>`
   Vrací vážený průměr předmětu podle id studenta.

5. `evaluations --post --xml/--json <cesta k souboru>`
   Vytvoří záznam hodnocení na základě parametrů v požadavku.

6. `evaluations --put <id> --xml/--json <cesta k souboru>`
   Upraví záznam hodnocení s id, záznam je upraven podle parametrů v požadavku.

7. `evaluations --delete <id>`
   Odstraní záznam s id.

Příkazy pro místnosti

1. `rooms --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů místností.

2. `rooms --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam místnosti s id.

3. `rooms --get --name <jméno> --xml/--json <cesta k souboru>`
   Vrací seznam záznamů místností podle jména.

4. `rooms --get --capacity <hodnota>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů místností podle kapacity.

5. `rooms --get --capacity <min> <max>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů místností, jejichž kapacita je v rozmezí.

6. `rooms --get --building <id>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů místností podle id budovy.

7. `rooms --get --building --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů místností podle jména budovy.

8. `rooms --get --class <id> --xml/--json <cesta k souboru>`
   Vrací záznam místnosti podle id třídy.

9. `rooms --get --class --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam místnosti podle jména třídy.

10. `rooms --post --xml/--json <cesta k souboru>`
    Vytvoří záznam místnosti/místností na základě parametrů v požadavku.

11. `rooms --put <id> --xml/--json <cesta k souboru>`
    Upraví záznam místnosti s id, záznam je upraven podle parametrů v požadavku.

12. `rooms --delete <id>`
    Odstraní záznam s id.

Příkazy pro pravidelné rozvrhové akce

1. `scheduleActions --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů pravidelných rozvrhových akcí.

2. `scheduleActions --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam pravidelné rozvrhové akce s id.

3. `scheduleActions --get --room <id>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů pravidelných rozvrhových podle id místnosti.

4. `scheduleActions --get --room --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů pravidelných rozvrhových akcí podle jména místnosti.

5. `scheduleActions --get --subject <id>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů pravidelných rozvrhových akcí id konkrétního předmětu.

6. `scheduleActions --get --subject --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů pravidelných rozvrhových akcí podle jména konkrétního předmětu.

7. `scheduleActions --post --xml/--json <cesta k souboru>`
   Vytvoří záznam pravidelné rozvrhové akce/akcí na základě parametrů v požadavku.

8. `scheduleActions --put <id> --xml/--json <cesta k souboru>`
   Upraví záznam pravidelné rozvrhové akce s id, záznam je upraven podle parametrů v požadavku.

9. `scheduleActions --delete <id>`
   Odstraní záznam s id.

Příkazy pro konkrétní rozvrhové akce

1. `scheduleActionsDates --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů konkrétních rozvrhových akcí.

2. `scheduleActionsDates --get <id>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam konkrétní rozvrhové akce s id.

3. `scheduleActionsDates --get --subject <id>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů konkrétních rozvrhových akcí podle id předmětu.

4. `scheduleActionsDates --get --substitutions`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů konkrétních rozvrhových akcí, které jsou suplované.

5. `scheduleActionsDates --get --substitutions <datum>`
   `--xml/--json <cesta k souboru>`

Vrací seznam záznamů konkrétních rozvrhových akcí podle data, které jsou suplované.

6. `scheduleActionsDates --get --substitutions <datum> --student <id> --xml/--json <cesta k souboru>`
   Vrací seznam záznamů konkrétních rozvrhových akcí podle data a id učitele, které jsou suplované.

7. `scheduleActionsDates --get --substitutions <datum> --teacher <id> --xml/--json <cesta k souboru>`
   Vrací seznam záznamů konkrétních rozvrhových akcí podle data a id studenta, které jsou suplované.

8. `scheduleActionsDates --post --xml/--json <cesta k souboru>`
   Vytvoří záznam konkrétní rozvrhové akce/akcí na základě parametrů v požadavku.

9. `scheduleActionsDates --delete <id>`
   Odstraní záznam s id.

Příkazy pro školní dny

1. `schoolDays --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů školních dnů.

2. `schoolDays --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam školního dne s id.

3. `schoolDays --get --name <jméno> --xml/--json <cesta k souboru>`
   Vrací seznam záznamů školních dnů podle zadaného jména.

4. `schoolDays --get --scheduleAction <id> --xml/--json <cesta k souboru>`
   Vrací záznam školního dne podle zadaného id pravidelné rozvrhové akce.

5. `schoolDays --get --scheduleActionDate <id> --xml/--json <cesta k souboru>`
   Vrací záznam školního dne podle id konkrétní rozvrhové akce.

6. `schoolDays --post --xml/--json <cesta k souboru>`
   Vytvoří záznam školního dne/dnů na základě parametrů v požadavku.

7. `schoolDays --put <id> --xml/--json <cesta k souboru>`
   Upraví záznam školního dne s id, záznam je upraven podle parametrů v požadavku.

8. `schoolDays --delete <id>`
   Odstraní záznam školního dne s id.

Příkazy pro školní hodiny

1. `schoolHours --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů školních hodin.

2. `schoolHours --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam školní hodiny s id.

3. `schoolHours --get --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů školních hodin podle zadaného jména.

4. `schoolHours --get --scheduleAction <id>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam školní hodiny podle id pravidelné rozvrhové akce.

5. `schoolHours --get --scheduleActionDate <id>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam školní hodiny podle zadaného id konkrétní rozvrhové akce.

6. `schoolHours --post --xml/--json <cesta k souboru>`
   Vytvoří záznam školní hodiny/hodin na základě parametrů v požadavku.

7. `schoolHours --put <id> --xml/--json <cesta k souboru>`
   Upraví záznam školní hodiny s id, záznam je upraven podle parametrů v požadavku.

8. `schoolHours --delete <id>`
   Odstraní záznam školní hodiny s id.

Příkazy pro studenty

1. `students --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů studentů.

2. `students --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam studenta s id.

3. `students --get --name <jméno> --surname <příjmení>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů studentů podle křestního jména a příjmení.

4. `students --get --class <id>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů studentů podle id třídy.

5. `students --get --class --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů studentů podle jména třídy.

6. `students --get --subject <id>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů studentů podle id konkrétního předmětu.

7. `students --get --subject --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů studentů podle jména konkrétního předmětu.

8. `students --post --xml/--json <cesta k souboru>`
   Vytvoří záznam studenta na základě parametrů v požadavku.

9. `students --put <id> --xml/--json <cesta k souboru>`
   Upraví záznam studenta s id, záznam je upraven podle parametrů v požadavku.

10. `students --put <id> --subject add`
    `--xml/--json <cesta k souboru>`
    Přidá studentovi s id konkrétní předmět, který je uveden v těle požadavku.

11. `students --put <id> --subject remove`
    `--xml/--json <cesta k souboru>`
    Odstraní studentovi s id konkrétní předmět, který je uveden v těle požadavku.

12. `students --delete <id>`
    Odstraní záznam s id.

Příkazy pro předměty studentů

1. `studentsSubjects --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů předmětů studentů.

2. `studentsSubjects --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam předmětu studenta s id.

3. `studentsSubjects --get --student <id> --subject <id> --xml/--json <cesta k souboru>`
   Vrací záznam předmětu studenta podle id studenta a id předmětu.

Příkazy pro předměty

1. `subjects --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů konkrétních předmětů.

2. `subjects --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam konkrétního předmětu s id.

3. `subjects --get --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů konkrétních předmětů jména.

4. `subjects --get --subjectList <id>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů konkrétních předmětů id předmětu.

5. `subjects --get --subjectList --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů konkrétních předmětů jména předmětu.

6. `subjects --get --class <id>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů konkrétních předmětů podle id třídy.

7. `subjects --get --class --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů konkrétních předmětů podle jména třídy.

8. `subjects --scheduleAction <id>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam konkrétního předmětu podle id rozvrhové akce.

9. `subjects --get --teacher <id>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam konkrétního předmětu podle id učitele.

10. `subjects --get --teacher --name <jméno>`
    `--surname <příjmení> --xml/--json <cesta k souboru>`
    Vrací záznam konkrétního předmětu podle křestního jména a příjmení učitele.

11. `subjects --get --student <id>`
    `--xml/--json <cesta k souboru>`
    Vrací seznam záznamů konkrétních předmětů podle id studenta.

12. `subjects --get --student --name <jméno>`
    `--surname <příjmení> --xml/--json <cesta k souboru>`
    Vrací seznam záznamů konkrétních předmětů podle křestního jména a příjmení studenta.

13. `subjects --post --xml/--json <cesta k souboru>`
    Vytvoří záznam konkrétního předmětu na základě parametrů v požadavku.

14. `subjects --put <id> --xml/--json <cesta k souboru>`
    Upraví záznam konkrétního předmětu s id, záznam je upraven podle parametrů v požadavku.

15. `subjects --delete <id> --xml/--json <cesta k souboru>`
    Odstraní záznam konkrétního předmětu s id.

Příkazy pro obecné předměty

1. `subjectLists --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů obecných předmětů.

2. `subjectLists --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam obecného předmětu s id.

3. `subjectLists --get --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů předmětů podle zadaného jména.

4. `subjectLists --get --subject <id>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam obecného předmětu podle zadaného id předmětu.

5. `subjectLists --get --subject --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam obecného předmětu podle zadaného jména konkrétního předmětu.

6. `subjectLists --get --teacher <id>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů obecných předmětů podle id učitele.

7. `subjectLists --get --teacher --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů obecných předmětů podle křestního jména a příjmení učitele.

8. `subjectLists --post --xml/--json <cesta k souboru>`
   Vytvoří záznam obecného předmětu na základě parametrů v požadavku.

9. `subjectLists --put <id> --xml/--json <cesta k souboru>`
   Upraví záznam obecného předmětu s id, záznam je upraven podle parametrů v požadavku.

10. `subjectLists --delete <id>`
    Odstraní záznam obecného předmětu s id.

Příkazy pro učitele

1. `teachers --get --xml/--json <cesta k souboru>`
   Vrací seznam všech záznamů učitelů.

2. `teachers --get <id> --xml/--json <cesta k souboru>`
   Vrací záznam učitele s id.

3. `teachers --get --name <jméno> --surname <příjmení>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů učitelů podle křestního jména a příjmení.

4. `teachers --get --subjectList <id>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů učitelů podle id předmětu.

5. `teachers --get --subjectList --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací seznam záznamů učitelů podle jména obecného předmětu.

6. `teachers --get --subject <id>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam učitele podle id konkrétního předmětu.

7. `teachers --get --subject --name <jméno>`
   `--xml/--json <cesta k souboru>`
   Vrací záznam učitele podle jména konkrétního předmětu

8. `teachers --post --xml/--json <cesta k souboru>`
   Vytvoří záznam učitele na základě parametrů v požadavku.

9. `teachers --put <id> --xml/--json <cesta k souboru>`
   Upraví záznam učitele s id, záznam je upraven podle parametrů v požadavku.

10. `teachers --put <id> --subjectList add`
    `--xml/--json <cesta k souboru>`
    Přidá učiteli s id obecný předmět, který je uveden v těle požadavku.

11. `teachers --put <id> --subjectList remove`
    `--xml/--json <cesta k souboru>`
    Odstraní učiteli s id obecný předmět, který je uveden v těle požadavku.

12. `teachers --delete <id>`
    Odstraní záznam učitele s id.

C Seznam testovacích scénářů CLI

Pro všechny testovací scénáře `CLI` se předpokládá validní připojení se k back-endu a databáze back-endu naplněná testovacími daty. Nejsou zde uvedeny testy pro JSON formát. Pro JSON formát byly vykonány stejné testy jako pro XML.

Absence

TS-Absences-01: Kontrola správného výpisu všech absencí, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
absences --get --xml "test/absences/all_absences.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Absences-02: Kontrola správného výpisu absence pomocí id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
absences --get 1 --xml "test/absences/absence_by_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Absences-03: Kontrola správného výpisu absence pomocí id studenta, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
absences --get --student 8 --xml
"test/absences/absence_by_student_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Absences-04: Kontrola správného výpisu absence pomocí jména a příjmení studenta, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
absences --get --student --name "Pavel"
--surname "Jaroš" --xml
"test/absences/absence_by_student_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Absences-05: Kontrola správného výpisu absence pomocí id konkrétní rozvrhové akce, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
absences --get --scheduleActionDate 3 --xml
"test/absences/absence_by_scheduleActionDate_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Absences-06: Kontrola správného vložení absencí z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
absences --post --xml "test/absences/post/post.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávaná entita.

TS-Absences-07: Kontrola správného upravení absence z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
absences --put 5 --xml "test/absences/put/put.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se upraví očekávaná entita.

TS-Absences-08: Kontrola správného odstraněné absence

Popis: Zadáme uvedený příkaz.
Příkaz:

```
absences --delete 5
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Budovy

TS-Buildings-01: Kontrola správného výpisu všech budov, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Buildings-02: Kontrola správného výpisu budovy podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
buildings --get 1 --xml "test/buildings/building_by_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Buildings-03: Kontrola správného výpisu budovy podle jména, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
buildings --get --name "Budova A" --xml
"test/buildings/building_by_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Buildings-04: Kontrola správného výpisu budovy podle id místnosti, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
buildings --get --room 1 --xml
"test/buildings/building_by_room_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Buildings-05: Kontrola správného vložení budovy z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
buildings --post --xml "test/buildings/post/post.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávaná entita.

TS-Buildings-06: Kontrola správného upravení budovy z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
buildings --put 3 --xml "test/buildings/put/put.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se upravila očekávaná entita.

TS-Buildings-07: Kontrola správného odstraněné budovy

Popis: Zadáme uvedený příkaz.
Příkaz:

```
buildings --delete 3
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Třídy

TS-Classes-01: Kontrola správného výpisu všech tříd, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --get --xml "test/classes/all_classes.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Classes-02: Kontrola správného výpisu třídy podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --get 1 --xml "test/classes/class_by_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Classes-03: Kontrola správného výpisu třídy podle jména, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --get --name "6.A" --xml
"test/classes/class_by_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Classes-04: Kontrola správného výpisu třídy podle id studenta, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --get --student 1 --xml
"test/classes/class_by_student_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Classes-05: Kontrola správného výpisu třídy podle jména a příjmení, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --get --student --name "Martin"
--surname "Merta" --xml
"test/classes/class_by_student_name_surname.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Classes-06: Kontrola správného výpisu třídy podle id učitele, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --get --teacher 1 --xml
"test/classes/class_by_teacher.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Classes-07: Kontrola správného výpisu třídy podle jména a příjmení učitele, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --get --teacher --name "Jan"
--surname "Zelený"  --xml
"test/classes/class_by_teacher_name_surname.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Classes-08: Kontrola správného výpisu třídy podle id místnosti, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --get --room 1 --xml
"test/classes/class_by_room_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity. **Popis:** Zadáme uvedený příkaz.
Příkaz:

```
classes --get --room 1 --json
"test/classes/class_by_room_id.json"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Classes-09: Kontrola správného výpisu třídy podle jména místnosti, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --get --room --name "Místnost 001A" --xml
"test/classes/class_by_room_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Classes-10: Kontrola správného výpisu třídy podle id předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --get --subject 1 --xml
"test/classes/class_by_subject_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Classes-11: Kontrola správného výpisu třídy podle jména předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --get --subject --name "Matematika Osm" --xml
"test/classes/class_by_subject_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávaná entita.

TS-Classes-12: Kontrola správného vložení třídy z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --post --xml "test/classes/post/post.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vloží očekávaná entita.

TS-Classes-13: Kontrola správného upravené třídy z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --put 3 --xml "test/classes/put/put.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se upraví očekávaná entita.

TS-Classes-14: Kontrola správného přidání předmětu třídě z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --put 3 --subject add --xml
"test/classes/put/putsubject.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se přidá předmět očekávané entity.

TS-Classes-15: Kontrola správného odstraní předmětu třídě z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --put 3 --subject remove --xml
"test/classes/put/putsubject.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní předmět očekávané entity.

TS-Classes-16: Kontrola správného odstranění třídy

Popis: Zadáme uvedený příkaz.
Příkaz:

```
classes --delete 3
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Klasifikace

TS-Evaluations-01: Kontrola správného výpisu všech klasifikací, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
evaluations --get --xml "test/evaluations/all_evaluations.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Evaluations-02: Kontrola správného výpisu klasifikace podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
evaluations --get 1 --xml
"test/evaluations/evaluation_by_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Evaluations-03: Kontrola správného výpisu klasifikace podle id studentova předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
evaluations --get --studentsSubjects 1 --xml
"test/evaluations/evaluation_by_ss_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Evaluations-04: Kontrola správného výpisu klasifikace podle id studentova předmětu a váženého průměru

Popis: Zadáme uvedený příkaz.
Příkaz:

```
evaluations --get --studentsSubjects average 1"
```

Očekávaný výsledek: Po zadání příslušného příkazu se průměrná hodnota entity.

TS-Evaluations-05: Kontrola správného vložení klasifikace z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
evaluations --post --xml "test/evaluations/post/post.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávaná entita.

TS-Evaluations-06: Kontrola správného upravení klasifikace z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
evaluations --put 25 --xml
"test/evaluations/put/put.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se upraví očekávaná entita.

TS-Evaluations-07: Kontrola správného odstranění klasifikace

Popis: Zadáme uvedený příkaz.
Příkaz:

```
evaluations --delete 25
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Místnosti

TS-Rooms-01: Kontrola správného výpisu všech místností, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --get --xml "test/rooms/all_rooms.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Rooms-02: Kontrola správného výpisu místnosti podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --get 1 --xml "test/rooms/room_by_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Rooms-03: Kontrola správného výpisu místnosti podle jména, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --get --name "Místnost 001A" --xml
"test/rooms/room_by_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Rooms-04: Kontrola správného výpisu místnosti podle kapacity, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --get --capacity 25 --xml
"test/rooms/room_by_capacity.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Rooms-05: Kontrola správného výpisu místnosti podle intervalu kapacity, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --get --capacity 25 40 --xml
"test/rooms/room_by_capacity_interval.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Rooms-06: Kontrola správného výpisu místnosti podle id budovy, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --get --building 1 --xml
"test/rooms/room_by_building_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Rooms-07: Kontrola správného výpisu místnosti podle budovy jména, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --get --building --name "Budova A" --xml
"test/rooms/room_by_building_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Rooms-08: Kontrola správného výpisu místnosti podle id třídy, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --get --class 1 --xml
"test/rooms/room_by_class_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Rooms-09: Kontrola správného výpisu místnosti podle jména třídy, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --get --class --name "6.A" --xml
"test/rooms/room_by_class_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Rooms-10: Kontrola správného vložení místnosti z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --post --xml "test/rooms/post/post.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávaná entita.

TS-Rooms-11: Kontrola správného upravení místnosti z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --put 5 --xml "test/rooms/put/put.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se upraví očekávaná entita.

TS-Rooms-12: Kontrola správného odstranění místnosti

Popis: Zadáme uvedený příkaz.
Příkaz:

```
rooms --delete 5
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Pravidelné rozvrhové akce

TS-ScheduleActions-01: Kontrola správného výpisu všech pravidelných rozvrhových akcí, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActions --get --xml
"test/scheduleactions/all_scheduleActions.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-ScheduleActions-02: Kontrola správného výpisu pravidelné rozvrhové akce podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActions --get 1 --xml
"test/scheduleactions/scheduleAction_by_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-ScheduleActions-03: Kontrola správného výpisu pravidelné rozvrhové akce podle id místnosti, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActions --room 1 --xml
"test/scheduleactions/scheduleAction_by_room_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-ScheduleActions-04: Kontrola správného výpisu pravidelné rozvrhové akce podle jména místnosti, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActions --room --name "Místnost 001A" --xml
"test/scheduleactions/scheduleAction_by_room_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-ScheduleActions-05: Kontrola správného výpisu pravidelné rozvrhové akce podle id předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActions --subject 1 --xml
"test/scheduleactions/room_by_subject_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-ScheduleActions-06: Kontrola správného výpisu pravidelné rozvrhové akce podle jména předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActions --subject --name "Matematika Osm" --xml
"test/scheduleactions/room_by_subject_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-ScheduleActions-07: Kontrola správného vložení pravidelné rozvrhové akce z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActions --post --xml
"test/scheduleactions/post/post.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávaná entita.

TS-ScheduleActions-08: Kontrola správného upravené pravidelné rozvrhové akce z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActions --put 11 --xml
"test/scheduleactions/put/put.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se upraví očekávaná entita.

TS-ScheduleActions-09: Kontrola správného odstranění pravidelné rozvrhové akce

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActions --delete 11
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Konkrétní rozvrhové akce

TS-scheduleActionDates-01: Kontrola správného výpisu všech konkrétních rozvrhových akcí, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActionsDates --get --xml
"test/scheduleactionsdates/all_sad.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-scheduleActionDates-02: Kontrola správného výpisu konkrétní rozvrhové akce podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActionsDates --get 1 --xml
"test/scheduleactionsdates/sad_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-scheduleActionDates-03: Kontrola správného výpisu konkrétní rozvrhové akce podle id předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActionsDates --get --subject 1 --xml
"test/scheduleactionsdates/sad_subject_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-scheduleActionDates-04: Kontrola správného výpisu konkrétní rozvrhové akce obsahující absence, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActionsDates --get --substitutions --xml
"test/scheduleactionsdates/sad_substitutions.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-scheduleActionDates-05: Kontrola správného výpisu konkrétní rozvrhové akce o datu obsahující absence, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActionsDates --get --substitutions "2020-01-01" --xml
"test/scheduleactionsdates/sad_substitutions_date.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-scheduleActionDates-06: Kontrola správného výpisu konkrétní rozvrhové akce o datu a studentova id obsahující absence, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActionsDates --get
--substitutions "2020-01-01" --student 1 --xml
"test/scheduleactionsdates/sad_substitutions_date.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-scheduleActionDates-07: Kontrola správného výpisu konkrétní rozvrhové akce o datu a učitelova id obsahující absence, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActionsDates --get
--substitutions "2020-01-01" --teacher 1 --xml
"test/scheduleactionsdates/sad_substitutions_date.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-scheduleActionDates-08: Kontrola správného vložení konkrétní rozvrhové akce z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActionsDates --post --xml
"test/scheduleactionsdates/post/post.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávaná entita.

TS-scheduleActionDates-09: Kontrola správného odstraněné konkrétní rozvrhové akce

Popis: Zadáme uvedený příkaz.
Příkaz:

```
scheduleActionsDates --delete 11
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Školní dny

TS-SchoolDays-01: Kontrola správného výpisu všech školních dnů, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolDays --get --xml "test/schooldays/all_schooldays.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SchoolDays-02: Kontrola správného výpisu školního dne podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolDays --get 1 --xml "test/schooldays/schoolday_by_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SchoolDays-03: Kontrola správného výpisu školního dne podle jména, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolDays --get --name "Pondělí" --xml
"test/schooldays/schoolday_by_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SchoolDays-04: Kontrola správného výpisu školního dne podle id rozvrhové akce, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolDays --get --scheduleAction 1 --xml
"test/schooldays/schoolday_by_sa_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SchoolDays-05: Kontrola správného výpisu školního dne podle id konkrétní rozvrhové akce, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolDays --get --scheduleActionDate 1 --xml
"test/schooldays/schoolday_by_sad_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SchoolDays-06: Kontrola správného vložené školního dne z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolDays --post --xml "test/schooldays/post/post.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávaná entita.

TS-SchoolDays-07: Kontrola správného upravení školního dne z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolDays --put 6 --xml "test/schooldays/put/put.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se upraví očekávaná entita.

TS-SchoolDays-08: Kontrola správného odstranění školního dne

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolDays --delete 6
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Školní hodiny

TS-SchoolHours-01: Kontrola správného výpisu všech školních hodin, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolHours --get --xml "test/schoolhours/all_schoolhours.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SchoolHours-02: Kontrola správného výpisu školní hodiny podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolHours --get 1 --xml
"test/schoolhours/schoolhours_by_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SchoolHours-03: Kontrola správného výpisu školní hodiny podle jména, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolHours --get --name "0. hodina" --xml
"test/schoolhours/schoolhours_by_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SchoolHours-04: Kontrola správného výpisu školní hodiny podle id rozvrhové akce, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolHours --get --scheduleAction 1 --xml
"test/schoolhours/schoolhours_by_sa_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SchoolHours-05: Kontrola správného upravení školní hodiny z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolHours --post --xml "test/schoolhours/post/post.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávané entity.

TS-SchoolHours-06: Kontrola správného odstranění školní hodiny

Popis: Zadáme uvedený příkaz.
Příkaz:

```
schoolHours --delete 6
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Studenti

TS-Students-01: Kontrola správného výpisu všech studentů, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --get --xml "test/students/all_students.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Students-02: Kontrola správného výpisu studenta podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --get 1 --xml "test/students/student_by_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Students-03: Kontrola správného výpisu studenta podle jména a příjmení, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --get --name "Tomáš"
--surname "Merta" --xml
"test/students/student_by_name_surname.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Students-04: Kontrola správného výpisu studenta podle id třídy, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --get --class 1 --xml
"test/students/student_by_class.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Students-05: Kontrola správného výpisu studenta podle jména třídy, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --get --class --name "6.A" --xml
"test/students/student_by_class_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Students-06: Kontrola správného výpisu studenta podle id předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --get --subject 1 --xml
"test/students/student_by_subject.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Students-07: Kontrola správného výpisu studenta podle jména předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --get --subject --name "Matematika Osm" --xml
"test/students/student_by_subject_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Students-08: Kontrola správného vložené studenta z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --post --xml "test/students/post/post.xml
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávané entity.

TS-Students-09: Kontrola správného upravení studenta z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --put 7 --xml "test/students/put/put.xml
```

Očekávaný výsledek: Po zadání příslušného příkazu se upraví očekávaná entita.

TS-Students-10: Kontrola správného přidání předmětu studenta z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --put 7 --subject add --xml
"test/students/put/putsubject.xml
```

Očekávaný výsledek: Po zadání příslušného příkazu se přidá předmět očekávané entity.

TS-Students-11: Kontrola správného odstranění předmětu studenta z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --put 7 --subject remove --xml
"test/students/put/putsubject.xml
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní předmět očekávané entity.

TS-Students-12: Kontrola správného odstranění studenta

Popis: Zadáme uvedený příkaz.
Příkaz:

```
students --delete 7
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Studentovi předměty

TS-StudentsSubjects-01: Kontrola správného výpisu všech předmětů studentů, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
studentsSubjects --get --xml
"test/studentssubjects/all_ss.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-StudentsSubjects-03: Kontrola správného výpisu předmětu studenta podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
studentsSubjects --get 1 --xml
"test/studentssubjects/ss_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-StudentsSubjects-05: Kontrola správného výpisu předmětu studenta podle id studenta a id předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
studentsSubjects --get --student 1 --subject 1 --xml
"test/studentssubjects/ss_student_id_subject_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

Předměty

TS-Subjects-01: Kontrola správného výpisu všech předmětů, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get --xml "test/subjects/all_subjects.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-02: Kontrola správného výpisu předmětu podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get 1 --xml "test/subjects/subject_by_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-03: Kontrola správného výpisu předmětu podle jména, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get --name "Matematika Osm" --xml
"test/subjects/subject_by_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-04: Kontrola správného výpisu předmětu podle id obecného předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get --subjectList 1 --xml
"test/subjects/subject_by_sl_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-05: Kontrola správného výpisu předmětu podle jména obecného předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get --subjectList --name "Matematika" --xml
"test/subjects/subject_by_sl_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-06: Kontrola správného výpisu předmětu podle id třídy, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get --class 1 --xml
"test/subjects/subject_by_class.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-07: Kontrola správného výpisu předmětu podle jména třídy, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get --class --name "6.A" --xml
"test/subjects/subject_by_class.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-08: Kontrola správného výpisu předmětu podle id rozvrhové akce, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get --scheduleAction 1 --xml
"test/subjects/subject_by_scheduleAction.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-09: Kontrola správného výpisu předmětu podle id učitele, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get --teacher 1 --xml
"test/subjects/subject_by_teacher.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-10: Kontrola správného výpisu předmětu podle jména a příjmení učitele, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get --teacher --name "Jan" --surname "Zelený"--xml
"test/subjects/subject_by_teacher_name_surname.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-11: Kontrola správného výpisu předmětu podle id studenta, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get --student 1 --xml
"test/subjects/subject_by_student.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-12: Kontrola správného výpisu předmětu podle jména a příjmení studenta, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --get --student --name "Tomáš" --surname "Mertl" --xml
"test/subjects/subject_by_student_name_surname.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Subjects-13: Kontrola správného vložení předmětu z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --post --xml "test/students/post/post.json"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávané entity.

TS-Subjects-14: Kontrola správného upravení předmětu z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --put 4 --xml "test/students/post/put.json"
```

Očekávaný výsledek: Po zadání příslušného příkazu se upraví očekávaná entita.

TS-Subjects-15: Kontrola správného odstranění předmětu

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjects --delete 4
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Obecné předměty

TS-SubjectLists-01: Kontrola správného výpisu všech obecných předmětů, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjectLists --get --xml
"test/subjectlists/all_subjectlists.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SubjectLists-02: Kontrola správného výpisu obecného předmětu podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjectLists --get 1 --xml "test/subjectlists/sl_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SubjectLists-03: Kontrola správného výpisu obecného předmětu podle jména, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjectLists --get --name "Matematika"
--xml "test/subjectlists/sl_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SubjectLists-04: Kontrola správného výpisu obecného předmětu podle id předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjectLists --get --subject 1 --xml
"test/subjectlists/sl_by_subject.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SubjectLists-05: Kontrola správného výpisu obecného předmětu podle jména předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjectLists --get --subject --name "Matematika Osm" --xml
"test/subjectlists/sl_by_subject_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SubjectLists-06: Kontrola správného výpisu obecného předmětu podle id učitele, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjectLists --get --teacher 1 --xml
"test/subjectlists/sl_by_teacher.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SubjectLists-07: Kontrola správného výpisu obecného předmětu podle jména učitele, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjectLists --get --teacher --name "Jan"
--surname "Zelený" --xml
"test/subjectlists/sl_by_teacher_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-SubjectLists-08: Kontrola správného vložení obecného předmětu z XML

TS-SubjectLists-09: Kontrola správného upravené obecného předmětu z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjectLists --put 1 --xml "test/subjectlists/put/put.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se upraví očekávaná entita.

TS-SubjectLists-10: Kontrola správného odstraněné obecného předmětu

Popis: Zadáme uvedený příkaz.
Příkaz:

```
subjectLists --delete 4
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

Učitelé

TS-Učitelé-01: Kontrola správného výpisu všech učitelů, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
teachers --get --xml "test/teachers/all_teachers.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Učitelé-02: Kontrola správného výpisu učitele podle id, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
teachers --get 1 --xml
"test/teachers/teacher_by_id.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Učitelé-03: Kontrola správného výpisu učitele podle jména a příjmení, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
teachers --get --name "Jan" --surname "Zelený" --xml
"test/teachers/teacher_by_name_surname.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Učitelé-04: Kontrola správného výpisu učitele podle id obecného předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
teachers --get --subjectList 1 --xml
"test/teachers/teacher_by_subjectlist.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Učitelé-05: Kontrola správného výpisu učitele podle jména obecného předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
teachers --get --subjectList --name "Matematika" --xml
"test/teachers/teacher_by_subjectlist_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Učitelé-06: Kontrola správného výpisu učitele podle id předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
teachers --get --subject 1 --xml
"test/teachers/teacher_by_subject.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Učitelé-07: Kontrola správného výpisu učitele podle jména předmětu, jako XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
teachers --get --subject --name "Matematika Osm" --xml
"test/teachers/teacher_by_subject_name.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vrátí všechny očekávané entity.

TS-Učitelé-08: Kontrola správného vložení učitele z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
teachers --post --xml "test/teachers/post/post.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se vytvoří očekávané entity.

TS-Učitelé-09: Kontrola správného přidání předmětu učitele z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
teachers --put 4 subjectList add --xml
"test/teachers/put/putsubject.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se přidá předmět očekávané entitě.

TS-Učitelé-10: Kontrola správného odstranění předmětu učitele z XML

Popis: Zadáme uvedený příkaz.
Příkaz:

```
teachers --put 4 subjectList remove --xml
"test/teachers/put/putsubject.xml"
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní předmět očekávané entitě.

TS-Učitelé-11: Kontrola správného odstranění učitele

Popis: Zadáme uvedený příkaz.
Příkaz:

```
teachers --delete 4
```

Očekávaný výsledek: Po zadání příslušného příkazu se odstraní očekávaná entita.

D Seznam testovacích scénářů GUI

Pro všechny testovací scénáře `GUI` se předpokládá validní připojení se k back-endu a databáze back-endu naplněná testovacími daty.

TS-GUI-01L: Kontrola anglické lokalizace titulku aplikace

Popis: po spuštění `GUI` se jako titulek aplikace má zobrazit: School Agenda.
Očekávaný výsledek: titulek aplikace se schoduje v názvu.

TS-GUI-02L: Kontrola české lokalizace titulku aplikace

Popis: po spuštění `GUI` se jako titulek aplikace má zobrazit: Školní Agenda.
Očekávaný výsledek: titulek aplikace se schoduje v názvu.

TS-GUI-03L: Kontrola anglické lokalizace tlačítek kategorií entit

Popis: po spuštění `GUI` se na levé straně aplikace zobrazí seznam kategorií entit. Očekávaný text tlačítek je: Absences, Buildings, Classes, Evaluations, Rooms, Schedule Action Dates, Schedule Actions, School Days, School Hours, Students, Students Subjects, Subjects, Subject Lists, Teachers.
Očekávaný výsledek: seznam kategorií entit obsahuje stejné pojmenování tlačítek

TS-GUI-04L: Kontrola české lokalizace tlačítek kategorií entit

Popis: po spuštění `GUI` se na levé straně aplikace zobrazí seznam kategorií entit. Očekávaný text tlačítek je: Nepřítomnosti, Budovy, Třídy, Klasifikace, Místnosti, Konkrétní školní akce, Školní akce, Vyučovací dny, Vyučovací hodiny, Žáci, Předměty žáků, Vyučované předměty, Předměty, Učitelé.
Očekávaný výsledek: seznam kategorií entit obsahuje stejné pojmenování tlačítek

TS-GUI-05L: Kontrola anglické lokalizace vyhledávacího pole pro kategorie entit

Popis: po spuštění `GUI` se na levé straně aplikace zobrazí seznam kategorií entit. Vyhledávací pole zde poté má mít popisek: Search fo a category...

Očekávaný výsledek: Text vyhledávacího pole se schoduje.

TS-GUI-06L: Kontrola české lokalizace vyhledávacího pole pro kategorie entit

Popis: po spuštění `GUI` se na levé straně aplikace zobrazí seznam kategorií entit. Vyhledávací pole zde poté má mít popisek: Vyhledej kategorii...
Očekávaný výsledek: Text vyhledávacího pole se schoduje.

TS-GUI-07L: Kontrola anglické lokalizace vyhledávacího pole pro kategorie entit

Popis: po spuštění `GUI` se na levé straně aplikace zobrazí seznam kategorií entit. Po vyhledání ve vyhledávacím poli kategorií entit neexistující kategorie se zobrazí: No category was found!.
Očekávaný výsledek: Text prázdného výskytu se schoduje.

TS-GUI-08L: Kontrola české lokalizace vyhledávacího pole pro kategorie entit

Popis: po spuštění `GUI` se na levé straně aplikace zobrazí seznam kategorií entit. Po vyhledání ve vyhledávacím poli kategorií entit neexistující kategorie se zobrazí: Kategorie nebyla nalezena!
Očekávaný výsledek: Text prázdného výskytu se schoduje.

TS-GUI-09L: Kontrola anglické lokalizace menu.

Popis: po spuštění `GUI` se na horní liště aplikace zobrazí menu. Menu má obsahovat kategorie: Objects, Language
Očekávaný výsledek: Text menu se schoduje.

TS-GUI-10L: Kontrola české lokalizace menu.

Popis: po spuštění `GUI` se na horní liště aplikace zobrazí menu. Menu má obsahovat kategorie: Objekty, Jazyk
Očekávaný výsledek: Text menu se schoduje.

TS-GUI-11L: Kontrola anglické lokalizace podmenu pro jazyk

Popis: po spuštění `GUI` se na horní liště aplikace zobrazí menu. Menu obsahuje podmenu s prvky: English, Czech.
Očekávaný výsledek: Prvky podmenu se schodují.

TS-GUI-12L: Kontrola české lokalizace podmenu pro jazyk

Popis: po spuštění `GUI` se na horní liště aplikace zobrazí menu. Menu obsahuje podmenu s prvky: Anglicky, Česky.
Očekávaný výsledek: Prvky podmenu se schodují.¨

TS-GUI-13L: Kontrola anglické lokalizace podmenu pro entity

Popis: po spuštění `GUI` se na horní liště aplikace zobrazí menu. Menu obsahuje podmenu s prvky: Upload Objects, Refresh Categories.
Očekávaný výsledek: Prvky podmenu se schodují.

TS-GUI-14L: Kontrola české lokalizace podmenu pro entity

Popis: po spuštění `GUI` se na horní liště aplikace zobrazí menu. Menu obsahuje podmenu s prvky: Nahrát Objekty, Aktualizovat Objekty.
Očekávaný výsledek: Prvky podmenu se schodují.

TS-GUI-15L: Kontrola anglické lokalizace pro seznam entity, prázdné.

Popis: po spuštění `GUI` se na ve středu aplikace zobrazí prázdný seznam entit. V tuto chvíli by měl obsahovat prvky s texty: Refresh Objects, Search fo an object..., No object was found!
Očekávaný výsledek: Texty prvků se schodují.

TS-GUI-16L: Kontrola české lokalizace pro seznam entity, prázdné.

Popis: po spuštění `GUI` se na ve středu aplikace zobrazí prázdný seznam entit. V tuto chvíli by měl obsahovat prvky s texty: Aktualizovat Objekty, Vyhledej objekt..., Objekt nebyl nalezen!
Očekávaný výsledek: Texty prvků se schodují.

TS-GUI-17L: Kontrola anglické lokalizace tlačítka na formulářovém panelu, prázdné.

Popis: po spuštění `GUI` se na pravé straně aplikace zobrazí prázdný formulářový panel. V tuto chvíli by měl obsahovat tři tlačítka, a to s texty: Import objects, Show object, Create object.
Očekávaný výsledek: Texty prvků se schodují.

TS-GUI-18L: Kontrola české lokalizace tlačítka na formulářovém panelu, prázdné.

Popis: po spuštění `GUI` se na pravé straně aplikace zobrazí prázdný formulářový panel. V tuto chvíli by měl obsahovat tři tlačítka, a to s texty: Nahrát objeky, Ukázat objekt, Vytvořit objekt.
Očekávaný výsledek: Texty prvků se schodují.

TS-GUI-01K: Klik na kategorii Nepřítomností

Popis: po kliku na kategorii nepřítomnosti se zobrazí čtyři testovací entity nepřítomností.
Očekávaný výsledek: Klik na kategorii zobrazil čtyři entity.

TS-GUI-02K: Klik na kategorii Budovy

Popis: po kliku na kategorii budovy se zobrazí dvě testovací entity budov.
Očekávaný výsledek: Klik na kategorii zobrazil dvě entity.

TS-GUI-03K: Klik na kategorii Třídy

Popis: po kliku na kategorii třídy se zobrazí dvě testovací entity tříd.
Očekávaný výsledek: Klik na kategorii zobrazil dvě entity.

TS-GUI-04K: Klik na kategorii Klasifikace

Popis: po kliku na kategorii třídy se zobrazí dvacet-čtyři testovacích entit klasifikací.
Očekávaný výsledek: Klik na kategorii zobrazil dvacet-čtyři entit.

TS-GUI-05K: Klik na kategorii Místnosti

Popis: po kliku na kategorii místnosti se zobrazí čtyři testovací entity místností.
Očekávaný výsledek: Klik na kategorii zobrazil čtyři entity.

TS-GUI-06K: Klik na kategorii Konkrétní školní akce

Popis: po kliku na kategorii třídy se zobrazí deset testovací entit konkrétních školních akcí.
Očekávaný výsledek: Klik na kategorii zobrazil deset entit.

TS-GUI-07K: Klik na kategorii Školní akce

Popis: po kliku na kategorii třídy se zobrazí deset testovací entit školní akce.
Očekávaný výsledek: Klik na kategorii zobrazil deset entit.

TS-GUI-08K: Klik na kategorii Vyučovací dny

Popis: po kliku na kategorii třídy se zobrazí pět testovací entit vyučovacího dne.
Očekávaný výsledek: Klik na kategorii zobrazil pět entit.

TS-GUI-09K: Klik na kategorii Vyučovací hodiny

Popis: po kliku na kategorii třídy se zobrazí šest testovacích entit vyučovací hodiny.
Očekávaný výsledek: Klik na kategorii zobrazil šest entit.

TS-GUI-10K: Klik na kategorii Žáci

Popis: po kliku na kategorii třídy se zobrazí osm testovací entit žák.
Očekávaný výsledek: Klik na kategorii zobrazil osm entit.

TS-GUI-11K: Klik na kategorii Předměty žáků

Popis: po kliku na kategorii předměty žáků se zobrazí čtyřicet-osm testovací entit předmětu žáka.
Očekávaný výsledek: Klik na kategorii zobrazil čtyřicet-osm entit.

TS-GUI-12K: Klik na kategorii Vyučované předměty

Popis: po kliku na kategorii třídy se zobrazí pět testovacích entit vyučovaných předmětů.
Očekávaný výsledek: Klik na kategorii zobrazil pět entit.

TS-GUI-13K: Klik na kategorii Předměty

Popis: po kliku na kategorii třídy se zobrazí tři testovací entity předmětu.
Očekávaný výsledek: Klik na kategorii zobrazil tři entity.

TS-GUI-14K: Klik na kategorii Učitelé

Popis: po kliku na kategorii třídy se zobrazí tři testovací entity učitelů.
Očekávaný výsledek: Klik na kategorii zobrazil tři entity.

TS-GUI-01TK: Klik na kategorii Nepřítomností, kontrola tvaru entity

Popis: po kliku na kategorii nepřítomnosti se zobrazí čtyři testovací entity nepřítomností ve správném tvaru.
Očekávaný výsledek: Klik na kategorii zobrazil čtyři entity, se správným popisem entit.

TS-GUI-02KT: Klik na kategorii Budovy, kontrola tvaru entity

Popis: po kliku na kategorii budovy se zobrazí dvě testovací entity budov ve správném tvaru.
Očekávaný výsledek: Klik na kategorii zobrazil dvě entity, se správným popisem entit.

TS-GUI-03KT: Klik na kategorii Třídy, kontrola tvaru entity

Popis: po kliku na kategorii třídy se zobrazí dvě testovací entity tříd ve správném tvaru.
Očekávaný výsledek: Klik na kategorii zobrazil dvě entity, se správným popisem entit ve správném tvaru.

TS-GUI-04KT: Klik na kategorii Klasifikace, kontrola tvaru entity

Popis: po kliku na kategorii třídy se zobrazí dvacet-čtyři testovacích entit klasifikací.
Očekávaný výsledek: Klik na kategorii zobrazil dvacet-čtyři entit, se správným popisem entit ve správném tvaru.

TS-GUI-05KT: Klik na kategorii Místnosti, kontrola tvaru entity

Popis: po kliku na kategorii místnosti se zobrazí čtyři testovací entity místností ve správném tvaru.
Očekávaný výsledek: Klik na kategorii zobrazil čtyři entity, se správným popisem entit.

TS-GUI-06KT: Klik na kategorii Konkrétní školní akce, kontrola tvaru entity

Popis: po kliku na kategorii třídy se zobrazí deset testovací entit konkrétních školních akcí ve správném tvaru.
Očekávaný výsledek: Klik na kategorii zobrazil deset entit, se správným popisem entit.

TS-GUI-07KT: Klik na kategorii Školní akce, kontrola tvaru entity

Popis: po kliku na kategorii třídy se zobrazí deset testovací entit školní akce ve správném tvaru.
Očekávaný výsledek: Klik na kategorii zobrazil deset entit, se správným popisem entit.

TS-GUI-08KT: Klik na kategorii Vyučovací dny, kontrola tvaru entity

Popis: po kliku na kategorii třídy se zobrazí pět testovací entit vyučovacího dne ve správném tvaru.
Očekávaný výsledek: Klik na kategorii zobrazil pět entit, se správným popisem entit.

TS-GUI-09KT: Klik na kategorii Vyučovací hodiny, kontrola tvaru entity

Popis: po kliku na kategorii třídy se zobrazí šest testovacích entit vyučovací hodiny ve správném tvaru.
Očekávaný výsledek: Klik na kategorii zobrazil šest entit, se správným popisem entit.

TS-GUI-10KT: Klik na kategorii Žáci, kontrola tvaru entity

Popis: po kliku na kategorii třídy se zobrazí osm testovací entit žák. ve správném tvaru.
Očekávaný výsledek: Klik na kategorii zobrazil osm entit, se správným popisem entit.

TS-GUI-11KT: Klik na kategorii Předměty žáků, kontrola tvaru entity

Popis: po kliku na kategorii předměty žáků se zobrazí čtyřicet-osm testovací entit předmětu žáka ve správném tvaru.
Očekávaný výsledek: Klik na kategorii zobrazil čtyřicet-osm entit, se správným popisem entit.

TS-GUI-12KT: Klik na kategorii Vyučované předměty, kontrola tvaru entity

Popis: po kliku na kategorii třídy se zobrazí pět testovacích entit vyučovaných předmětů ve správném tvaru.

Očekávaný výsledek: Klik na kategorii zobrazil pět entit, se správným popisem entit.

TS-GUI-13KT: Klik na kategorii Předměty, kontrola tvaru entity

Popis: po kliku na kategorii třídy se zobrazí tři testovací entity předmětu, ve správném tvaru, se správným popisem entit.
Očekávaný výsledek: Klik na kategorii zobrazil tři entity.

TS-GUI-14KT: Klik na kategorii Učitelé, kontrola tvaru entity

Popis: po kliku na kategorii třídy se zobrazí tři testovací entity učitelů ve správném tvaru.
Očekávaný výsledek: Klik na kategorii zobrazil tři entity, se správným popisem entit.

TS-GUI-01VH: Pozitivní kontrola vyhledávání kategorií

Popis: po vyhledání slova: školní akce, se zobrazí Konkrétní školní akce a školní akce kategorie entit.
Očekávaný výsledek: Vyhledání kategerií vyhledá dvě kategorie.

TS-GUI-02VH: Negativní kontrola vyhledávání kategorií

Popis: po vyhledání slova: mamut, se zobrazí text informující o nenalezení žádně takové kategorie.
Očekávaný výsledek: Vyhledání kategerií nevyhledá žádnou kategorii.

TS-GUI-03VH: Pozitivní kontrola vyhledávání entit kategorie

Popis: Po kliku na kategorii žáci a po vyhledání slova: Josef, se zobrazí konkrétní žák se stejným jménem.
Očekávaný výsledek: Vyhledání správné entity.

TS-GUI-04VH: Negativní kontrola vyhledávání entit kategorie

Popis: Po kliku na kategorii žáci a po vyhledání slova: Mamut, se nezobrazí žádný žák.
Očekávaný výsledek: Vyhledání nevyhledá žádnou entitu.

TS-GUI-01VL: Vytvořit nepřítomnost kontrola anglické lokalizace

Popis: Po kliku na kategorii Nepřítomnosti se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Students, Schedule action dates.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-02VL: Vytvořit nepřítomnost kontrola české lokalizace

Popis: Po kliku na kategorii Nepřítomnosti se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Studenti, Konkrétní školní akce.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-03VL: Vytvořit budovu kontrola anglické lokalizace

Popis: Po kliku na příslušnou kategorii se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Building name.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-03VL: Vytvořit budovu kontrola české lokalizace

Popis: Po kliku na příslušnou kategorii se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Název budovy.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-04VL: Vytvořit třídu kontrola anglické lokalizace

Popis: Po kliku na kategorii třída se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Class name, Rooms, Teachers.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-05VL: Vytvořit třídu kontrola české lokalizace

Popis: Po kliku na kategorii třída se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Název třídy, Místnosti a Učitelé.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-06VL: Vytvořit klasifikaci kontrola anglické lokalizace

Popis: Po kliku na příslušnou kategorii se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Test Name, Grade,

Weight, Students subjects.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-07VL: Vytvořit klasifikaci kontrola české lokalizace

Popis: Po kliku na příslušnou kategorii se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Název testu, Známka, Váha, Žákovy Předměty.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-08VL: Vytvořit místnost kontrola anglické lokalizace

Popis: Po kliku na kategorii místnost se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Room name, Capacity, Buildings.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-09VL: Vytvořit místnost kontrola české lokalizace

Popis: Po kliku na kategorii místnost se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Název místnosti, Kapacita, Budovy.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-10VL: Vytvořit konkrétní školní akce kontrola anglické lokalizace

Popis: Po kliku na kategorii konkrétní školní akce se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Date, Theme, Schedule actions.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-11VL: Vytvořit konkrétní školní akce kontrola české lokalizace

Popis: Po kliku na kategorii konkrétní školní akce se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Název místnosti, Kapacita, Budovy.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-12VL: Vytvořit školní akci kontrola anglické lokalizace

Popis: Po kliku na příslušnou kategorii se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Rooms, Subjects, School days a School hours.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-13VL: Vytvořit školní akci kontrola české lokalizace

Popis: Po kliku na kategorii školní akce se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Místnosti, Předměty, Vyučovací dny, Vyučovací hodiny.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-14VL: Vytvořit vyučovací dny kontrola anglické lokalizace

Popis: Po kliku na kategorii vyučovací dny akce se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: School day name.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-15VL: Vytvořit vyučovací dny kontrola české lokalizace

Popis: Po kliku na kategorii vyučovací dny se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Název vyučovacího dne.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-16VL: Vytvořit vyučovací hodiny kontrola anglické lokalizace

Popis: Po kliku na kategorii vyučovací hodiny se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: School hour name, School hour start time, School hour end time.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-17VL: Vytvořit vyučovací hodiny kontrola české lokalizace

Popis: Po kliku na kategorii vyučovací hodiny se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Název vyučovací hodiny, Začátek vyučovací hodiny, Konec vyučovací hodiny.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-18VL: Vytvořit žáka kontrola anglické lokalizace

Popis: Po kliku na kategorii žáci akce se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: First name, Last name, Birthdate, Residence, Phone, E-mail a classes.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-19VL: Vytvořit žáky kontrola české lokalizace

Popis: Po kliku na kategorii žáci se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Křestní Jméno, Rodné jméno, Datum narození, Adresa bydliště Mobil, E-mail, Třídy.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-22VL: Vytvořit vyučované předměty kontrola anglické lokalizace

Popis: Po kliku na příslušnou kategorii se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Subject name, Subjects a Teachers.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-23VL: Vytvořit vyučované předměty kontrola české lokalizace

Popis: Po kliku na příslušnou kategorii se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Název předmětu, předměty a učitele.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-24VL: Vytvořit předměty kontrola anglické lokalizace

Popis: Po kliku na kategorii žáci akce se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Subject name.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-25VL: Vytvořit předměty kontrola české lokalizace

Popis: Po kliku na příslušnou kategorii se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Název předmětu.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-26VL: Vytvořit učitele kontrola anglické lokalizace

Popis: Po kliku na kategorii žáci akce se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: First name, Last name, Birthdate, Residence, Phone, E-mail a commencement date.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-27VL: Vytvořit učitele kontrola české lokalizace

Popis: Po kliku na kategorii žáci se zobrazí formulář s vytvořením entity. Formulář obsahuje poté řádky označené jako: Křestní Jméno, Rodné jméno, Datum narození, Adresa bydliště Mobil, E-mail, Třídy.
Očekávaný výsledek: Všechny textové položky odpovídají.

TS-GUI-01KV: Kontrola polí formuláře pro vytváření nepřítomnosti

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: studenty, konkrétní školní akce.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-02KV: Kontrola polí formuláře pro vytváření budovy

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: název budovy.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-03KV: Kontrola polí formuláře pro vytváření třídy

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: název třídy, místnosti a učitele.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-04KV: Kontrola polí formuláře pro vytváření klasifikace

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: název testu, známka, váha a žákovy předměty.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-05KV: Kontrola polí formuláře pro vytváření místnosti

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: název místnosti, kapacitu a budovy.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-06KV: Kontrola polí formuláře pro vytváření konkrétní školní akce

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: datum, téma, školní akce.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-07KV: Kontrola polí formuláře pro vytváření školní akce

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: místnosti, předmětu, vyučovacího dne, vyučovací hodiny.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-08KV: Kontrola polí formuláře pro vytváření vyučovacího dne

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: název vyučovacího dne.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-09KV: Kontrola polí formuláře pro vytváření vyučovací hodiny

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: název vyučovací hodiny, začátek vyučovací hodiny a konec vyučovací hodiny.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-10KV: Kontrola polí formuláře pro vytváření žáků

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: křestní jméno, rodné jméno, datum narození, adresu bydliště, mobil, email a třídu.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-12KV: Kontrola polí formuláře pro vytváření vyučovaných předmětů

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: název předmětu, předměty a učitele.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-13KV: Kontrola polí formuláře pro vytváření předmětů

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: název předmětu.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-14KV: Kontrola polí formuláře pro vytváření učitelů

Popis: Po kliku na příslušnou kategorii entity se zobrazí formulář pro tvorbu její entity. Entita má obsahovat pole pro: křestní jméno, rodné jméno, datum narození, adresu bydliště, mobil, e-mail a datum nástupu.
Očekávaný výsledek: Všechny pole jsou přítomny a odpovídají si.

TS-GUI-01I: Kontrola integrity polí vytvářecích formulářů

Popis: Projdeme všechny vytvářecí formuláře kategorií entit. všechny pole položek by zde měli být označeny červenou barvu, až na hodnoty číselné, značící chybné vyplnění a pole by měla být prázdná.
Očekávaný výsledek: Všechna pole jsou prázdná a označena jako chybné.

TS-GUI-02I: Kontrola integrity seznamových polí: Studenti

Popis: Projdeme vytvářecí formuláře pro: Nepřítomnosti. Zkontrolujeme dané pole. Mělo by obsahovat příslušných osm entit.
Očekávaný výsledek: všechna konkrétní pole obsahují osm příslušných entit.

TS-GUI-03I: Kontrola integrity seznamových polí: Konkrétní školní akce

Popis: Projdeme vytvářecí formuláře pro: Nepřítomnosti. Zkontrolujeme pole konkrétní školní akce. Mělo by obsahovat deset příslušných entit.
Očekávaný výsledek: všechna konkrétní pole obsahují deset příslušných entit.

TS-GUI-04I: Kontrola integrity seznamových polí: Místnosti

Popis: Projdeme vytvářecí formuláře pro: Třídy, Školní akce. Zkontrolujeme pole místností. Mělo by obsahovat čtyři příslušné entity.
Očekávaný výsledek: všechna konkrétní pole obsahují čtyři příslušné entity.

TS-GUI-05I: Kontrola integrity seznamových polí: Učitelé

Popis: Projdeme vytvářecí formuláře pro: Třídy. Zkontrolujeme pole třídy. Mělo by obsahovat tři příslušné entity.
Očekávaný výsledek: všechna konkrétní pole obsahují tři příslušné entity.

TS-GUI-06I: Kontrola integrity seznamových polí: Žákovy předměty

Popis: Projdeme vytvářecí formuláře pro: Klasifikace. Zkontrolujeme pole třídy. Mělo by obsahovat dvacet-čtyři příslušné entity.
Očekávaný výsledek: všechna konkrétní pole obsahují dvacet-čtyři příslušných entit.

TS-GUI-07I: Kontrola integrity seznamových polí: Budovy

Popis: Projdeme vytvářecí formuláře pro: Místnosti. Zkontrolujeme pole místností. Mělo by obsahovat dvě příslušné entity.
Očekávaný výsledek: všechna konkrétní pole obsahují dvě příslušné entity.

TS-GUI-08I: Kontrola integrity seznamových polí: Školní akce

Popis: Projdeme vytvářecí formuláře pro: Konkrétní školní akce. Zkontrolujeme pole školní akce. Mělo by obsahovat deset příslušných entit.
Očekávaný výsledek: všechna konkrétní pole obsahují deset příslušných entit.

TS-GUI-09I: Kontrola integrity seznamových polí: Vyučovací dny

Popis: Projdeme vytvářecí formuláře pro: Školní akce. Zkontrolujeme pole vyučovací dny. Mělo by obsahovat pět příslušný entit.
Očekávaný výsledek: všechna konkrétní pole obsahují pět příslušných entit.

TS-GUI-09I: Kontrola integrity seznamových polí: Vyučovací hodiny

Popis: Projdeme vytvářecí formuláře pro: Školní akce. Zkontrolujeme pole vyučovací hodiny. Mělo by obsahovat šest příslušných entit.
Očekávaný výsledek: všechna konkrétní pole obsahují šest příslušných entit.

TS-GUI-10I: Kontrola integrity seznamových polí: Třídy

Popis: Projdeme vytvářecí formuláře pro: Žáky. Zkontrolujeme pole třídy. Mělo by obsahovat dvě příslušné entity.
Očekávaný výsledek: všechna konkrétní pole obsahují dvě příslušné entity.

TS-GUI-11I: Kontrola integrity seznamových polí: Předměty

Popis: Projdeme vytvářecí formuláře pro: Vyučované předměty. Zkontrolujeme pole předměty. Mělo by obsahovat tři příslušné entity.
Očekávaný výsledek: všechna konkrétní pole obsahují tři příslušné entity.

TS-GUI-12I: Kontrola integrity seznamových polí: Učitelé

Popis: Projdeme vytvářecí formuláře pro: Vyučované předměty. Zkontrolujeme pole učitelé. Mělo by obsahovat tři příslušné entity.
Očekávaný výsledek: všechna konkrétní pole obsahují tři příslušné entity.

TS-GUI-13I: Kontrola integrity číselného pole: Známka

Popis: Projdeme vytvářecí formuláře pro: Klasifikace. Zkontrolujeme pole známka. Měla by zde jít zadat libovolná nenulová číslice.
Očekávaný výsledek: pole se chová korektně.

TS-GUI-14I: Kontrola integrity číselného pole: Váha

Popis: Projdeme vytvářecí formuláře pro: Klasifikace. Zkontrolujeme pole váha. Měla by zde jít zadat libovolná nenulová číslice.
Očekávaný výsledek: pole se chová korektně.

TS-GUI-15I: Kontrola integrity číselného pole: Kapacita

Popis: Projdeme vytvářecí formuláře pro: Místnost. Zkontrolujeme pole kapacity. Měla by zde jít zadat libovolná nenulová číslice.
Očekávaný výsledek: pole se chová korektně.

TS-GUI-17I: Kontrola integrity: Datum

Popis: Projdeme vytvářecí formuláře pro: Konkrétní školní akce, Učitele, Žáky. Zkontrolujeme pole pro datum. Měla by de jít zadat libovolné datum.
Očekávaný výsledek: pole se chová korektně.

TS-GUI-18I: Kontrola integrity: Čas

Popis: Projdeme vytvářecí formuláře pro: vyučovací hodiny. Zde zkontrolujeme pole pro čas. Měla by zde jít zadat libovolný čas.
Očekávaný výsledek: pole se chová korektně.

TS-GUI-19I: Kontrola integrity: Email

Popis: Projdeme vytvářecí formuláře pro: Žáky, Učitele. Zkontrolujeme pole pro email. Měla by zde jít zadat korektní email.
Očekávaný výsledek: pole se chová korektně.

TS-GUI-20I: Kontrola integrity: Telefonní číslo

Popis: Projdeme vytvářecí formuláře pro: Žáky, Učitele. Zkontrolujeme pole pro telefonní čísla. Měla by zde jít zadat korektní telefonní číslo.
Očekávaný výsledek: pole se chová korektně.

TS-GUI-01V: Vytvoření nové entity: nepřítomnost

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-02V: Vytvoření nové entity: budova

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-03V: Vytvoření nové entity: třídy

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-04V: Vytvoření nové entity: klasifikace

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-05V: Vytvoření nové entity: místnosti

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-06V: Vytvoření nové entity: konkrétní školní akce

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-07V: Vytvoření nové entity: školní akce

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-08V: Vytvoření nové entity: vyučovací dny

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-09V: Vytvoření nové entity: vyučovací hodiny

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-10V: Vytvoření nové entity: žáci

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-11V: Vytvoření nové entity: předměty žáků

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-12V: Vytvoření nové entity: vyučovací předměty

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-13V: Vytvoření nové entity: předměty

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-14V: Vytvoření nové entity: učitelé

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu a otestujeme její úspěšné nahrání na back–end.
Očekávaný výsledek: entita by správně vytvořena.

TS-GUI-01Z: Správné zobrazení editované entity: Nepřítomnost

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-02Z: Správné zobrazení editované entity: Budova

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-03Z: Správné zobrazení editované entity: Třída

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě

zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-04Z: Správné zobrazení editované entity: Klasifikace

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-05Z: Správné zobrazení editované entity: Místnost

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-06Z: Správné zobrazení editované entity: Konkrétní školní akce

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-07Z: Správné zobrazení editované entity: Školní akce

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-08Z: Správné zobrazení editované entity: Vyučovací dny

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-09Z: Správné zobrazení editované entity: Vyučovací hodiny

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-10Z: Správné zobrazení editované entity: Žáci

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-11Z: Správné zobrazení editované entity: Předměty žáku

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-12Z: Správné zobrazení editované entity: Vyučované předměty

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-13Z: Správné zobrazení editované entity: Předměty

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-14Z: Správné zobrazení editované entity: Učitelé

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě zkontrolujeme všechny pole zda jsou stejné.
Očekávaný výsledek: entita má všechna pole správná.

TS-GUI-01R: Reset editované entity: Nepřítomnost

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-02R: Reset editované entity: Budova

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-03R: Reset editované entity: Třída

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-04R: Reset editované entity: Klasifikace

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-05R: Reset editované entity: Místnost

Popis: Projdeme vytvářecí formuláře pro: Třídy. Zkontrolujeme pole třídy. Mělo by **Popis**: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-06R: Reset editované entity: Konkrétní školní akce

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-07R: Reset editované entity: Školní akce

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-08R: Reset editované entity: Vyučovací dny

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-09R: Reset editované entity: Vyučovací hodiny

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-10R: Reset editované entity: Žáci

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-11R: Reset editované entity: Předměty žáku

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-12R: Reset editované entity: Vyučované předměty

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-13R: Reset editované entity: Předměty

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-14R: Reset editované entity: Učitelé

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole a klikneme na reset entity.
Očekávaný výsledek: úpravy entit byli vráceny.

TS-GUI-01U: Úprava editované entity: Nepřítomnost

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-02U: Úprava editované entity: Budova

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-03U: Úprava editované entity: Třída

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-04U: Úprava editované entity: Klasifikace

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.

Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-05U: Úprava editované entity: Místnost

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-06U: Úprava editované entity: Konkrétní školní akce

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-07U: Úprava editované entity: Školní akce

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-08U: Úprava editované entity: Vyučovací dny

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-09U: Úprava editované entity: Vyučovací hodiny

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-10U: Úprava editované entity: Žáci

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-11U: Úprava editované entity: Předměty žáku

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-12U: Úprava editované entity: Vyučované předměty

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-13U: Úprava editované entity: Předměty

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-14U: Úprava editované entity: Učitelé

Popis: Rozklikneme příslušnou kategorii entity. Vytvoříme novou entitu, kde otestovanost vytvoření je již zajištěna. Zvolíme vytvořenou entitu. Entitě upravíme všechny pole na validní.
Očekávaný výsledek: entita byla úspěšně upravena a všechna její pole byla změněna.

TS-GUI-01S: Smazání editované entity: Nepřítomnost

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-02S: Smazání editované entity: Budova

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-03S: Smazání editované entity: Třída

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-04S: Smazání editované entity: Klasifikace

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-05S: Smazání editované entity: Místnost

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-06S: Smazání editované entity: Konkrétní školní akce

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-07S: Smazání editované entity: Školní akce

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-08S: Smazání editované entity: Vyučovací dny

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-09S: Smazání editované entity: Vyučovací hodiny

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-10S: Smazání editované entity: Žáci

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-11S: Smazání editované entity: Předměty žáku

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-12S: Smazání editované entity: Vyučované předměty

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-13S: Smazání editované entity: Předměty

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.

TS-GUI-14S: Smazání editované entity: Učitelé

Popis: rozklikneme příslušnou kategorii entity, vytvoříme entitu, kde už jsme otestovali vytváření entit a poté danou entitu smažeme.
Očekávaný výsledek: entita byla úspěšně smazána.