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

Diplomová práce

Simulace přesunů výuky za účelem její optimalizace

Plzeň 2018 Bc. Petra Volenová

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 27. června 2018

Bc. Petra Volenová

Poděkování

Ráda bych poděkovala Ing. Tomáši Kotoučovi za vedení mé diplomové práce, věcné připomínky a cenné rady, které mi pomohly tuto práci vypracovat.

Abstract

This thesis is focused on the development of an application that provides simulation of timetable events migration at the University. The theoretical part deals with the research of the problems of the timetable events migration. It also deals with the university courses timetabling problem. It presents basic algorithms that are used for timetabling. Finally, it examines technologies that are suatable for schedule and charts visualization in Java portal application. The practical part describes the implementation of the created application. It introduces the simulation of timetable events migration algorithm and the application structure. Finally, it describes the application deploytment on the information system portal and its testing.

Abstrakt

Tato práce je zaměřena na vývoj aplikace, která umožní simulaci přesunu výuky na univerzitě. Teoretická část práce se zabývá seznámením s problematikou přesunů výuky. Dále seznamuje s problematikou tvorby univerzitních rozvrhů. Představuje základní algoritmy, které se pro tvorbu univerzitních rozvrhů používají. Na závěr prozkoumává technologie, které jsou vhodné pro zobrazení rozvrhů a grafů v portálových Java aplikacích. Praktická část popisuje implementaci vytvořené aplikace. Seznamuje s algoritmem simulace přesunu výuky a strukturou aplikace. Na závěr popisuje postup nasazení na portál informačního systému IS/STAG a její otestování.

Obsah

1 Úvod

Místnosti na univerzitách často nejsou plně využívány, jelikož v nich výuka neprobíhá celou možnou dobu. Z tohoto důvodu univerzity používají více místností než potřebují. Také není plně využita kapacita místností a výuka probíhá ve větších prostorech než je potřeba. Provoz těchto místností však není levnou záležitostí, a proto je vhodné se tímto problémem zabývat.

Z tohoto důvodu se tato práce zabývá možností simulace přesunu výuky. Jedná se o navrhnutí aplikace, která umožní simulovat přesun vybraných rozvrhových akcí do vhodných místností.

Aplikaci bude sloužit k nasimulování nového umístění rozvrhových akcí. Díky tomu, že bude možné rozvrhové akce přesunout do menšího množství místností, může dojít ke snížení nákladu na jejich provoz. Dále v případě přesunutí výuky studentů na jedno místo, dojde k úspoře jejich času, který by strávili přesunem.

V první části práce seznamuje s problematikou přesunu výuky a důvody, proč je důležité se jí zabývat. Dále uvádí požadavky, které by aplikace měla splňovat. Následně se zabývá problematikou tvorby rozvrhů na univerzitě.

Jelikož má tvorba univerzitních rozvrhů společné znaky s přesunem výuky, jsou v další části jsou zkoumány algoritmy a optimalizační techniky, které je vhodné použít při tvorbě těchto rozvrhů. Poté jsou prozkoumávány technologie vhodné pro zobrazení rozvrhů a tvorbu grafů v portálových Java aplikací. Výběr technologií je ovlivněn tím, že výsledná aplikace bude součástí informačního systému IS/STAG.

Poté je popsána implementace zvoleného řešení s použitím prozkoumaných technologií. Jedná se o algoritmus, který provede simulaci stěhování výuky, tabulky v databázi i popis struktury aplikace. Na závěr uvádí postup sestavení a nasazení aplikace do informačního systému IS/STAG a seznamuje s provedeným otestováním aplikace.

2 Přesun výuky

Kapitola obsahuje základní úvod do problematiky přesunů výuky. Seznamuje s důvody, proč je tento problém důležité řešit, popisuje hlavní informace a shrnuje požadavky kladené na vytvářenou aplikaci.

2.1 Důvody přesunu výuky

Západočeskou univerzitu tvoří mnoho různých fakult a kateder. Jelikož se tato pracoviště nenacházejí na jednom místě, ale jsou rozprostřena po celé Plzni, neprobíhá jejich výuka centralizovaně, ale odehrává se v různých budovách a lokalitách. Část výuky se dokonce odehrává i v jiných městech.

Vzhledem k ubývajícímu množství studentů dochází k situacím, kdy se budovy univerzity nevyužívají naplno. Některé místnosti nejsou při výuce plně obsazeny, jiné se dokonce využívají jen minimálně.

Protože provoz těchto budov stojí univerzitu mnoho peněz, je vhodné výuku v některých budovách přesunout tak, aby docházelo k efektivnímu využívání místností. Nepoužívané místnosti se pak mohou dále pronajímat. Případně je možné celé prázdné budovy odprodat.

V nedávné době došlo k přesunutí výuky Ekonomické fakulty z centra města do univerzitního kampusu. Tímto krokem došlo k zefektivnění využívání budov v kampusu, zároveň univerzita mohla odprodat budovy v centru a snížit tím náklady na svůj provoz.

Přesun výuky do menšího množství budov nemusí mít jen ekonomický význam. Pokud mají studenti výuku v různých částech města, mohou mít problém s včasným přejetím přes celé město na další přednášku či cvičení. Z tohoto důvodu může být vhodné výuku studentům centralizovat do jednoho místa a tím jim usnadnit přesuny mezi hodinami.

V současné době neexistuje vhodný nástroj, který by umožnil simulovat přesun výuky mezi jednotlivými budovami univerzity. Proto je vhodné takovouto aplikaci vytvořit, jelikož bude mít mnoho využití. Bude ji možné použít pro optimalizaci a sestěhování výuky ve stejných budovách nebo naplánování přesunu do jiných

budov kvůli rekonstrukci či plánované výstavbě nových budov. Dále bude možné podle těchto nasimulovaných přesunů připravit podklady pro reálné stěhování.

2.2 Problematika přesunu výuky

Na Západočeské univerzitě je výuka reprezentována rozvrhovými akcemi (RA). Každá rozvrhová akce se odehrává v určitý den a hodinu, v konkrétní místnosti a budově. RA většinou vyučuje pouze jeden vyučující, avšak existují případy, kdy se vyučující během semestru na výuce střídají. Navštěvuje ji skupina studentů, která může být tvořena studenty z jedné nebo z více různých fakult.

2.2.1 Místnosti

Místnosti jsou označeny zkratkou budovy a číslem místnosti. Jejich vlastníkem může být konkrétní pracoviště univerzity nebo mohou patřit do společného fondu.

Účel místnosti definuje její typ. Podle vybavení a velikosti se místnosti dělí do různých kategorií, například to mohou být laboratoře, posluchárny nebo učebny.

Kapacita místnosti je určena počtem židlí, které se v ní nachází. Místnost nemusí být využívána pouze jejím vlastníkem, ale může být pronajímána i dalším pracovištím. Pronajímatelé pak vlastníkovi platí za počet židlí a čas, po který místnost používali pro výuku.

Na obrázku 2.1 se nachází ukázka jedné definované místnosti v systému IS/STAG spolu s jejími vlastnostmi.

2.2.2 Rozvrhové akce

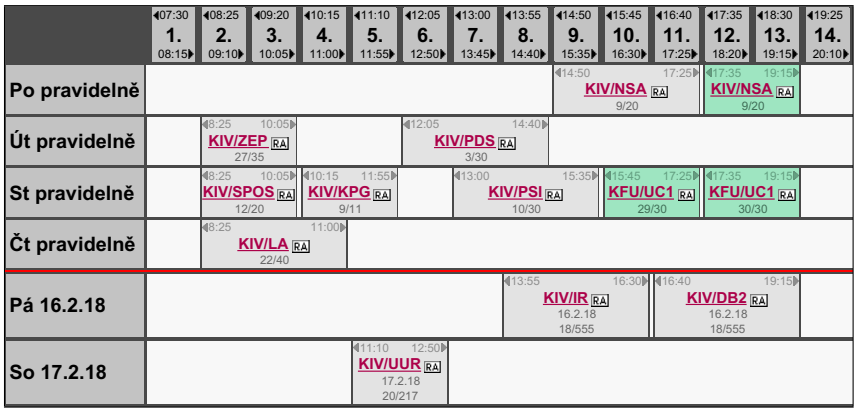
Rozvrhová akce neboli vyučovací hodina reprezentuje část výuky předmětu. Může to být přednáška, cvičení, nebo seminář. Některé předměty potřebují k výuce konkrétní RA speciální vybavení, a proto jsou jejich RA při rozvrhování zařazovány do místností podle typu místnosti.

Předměty mohou být vyučovány během zimního nebo letního semestru. RA proto obsahují informaci, pro který semestr jsou určeny. RA jsou dále přiřazeny na konkrétní den v týdnu. Některé RA se opakují pravidelně a jsou jsou vyučovány

Budova a místnost	UC-334
Adresa budovy	Technická 8, areál Bory - katedry FAV,2967, Plzeň
Pracoviště	Katedra informatiky a výpočetní techniky
Typ	Laboratoř
Společný fond	Ne
Určena pro výuku	Ano
Kapacita	20
Čísla dveří	-
Platnost od - do	01.08.2014
Identifikátor místnosti v externím systému	-
Identifikátor budovy v externím systému	WWW
Inventář	
Poznámka	tabule 200/100 cm keramická magnetická bílá

Obrázek 2.1: Ukázka místnosti[1]

v intervalu označeném konkrétními týdny. Proto jsou periodicky rozvrhovány na stejný den a čas po celý semestr. Tyto RA jsou zobrazeny v horní polovině obrázku 2.2 v řádcích označených dnem v týdnu. Jiné RA mají nepravidelné opakování a mohou být v průběhu semestru rozvrhovány na jiný den i čas, jak je zobrazeno ve spodní části obrázku 2.2 v řádcích označených konkretním datem.



Obrázek 2.2: Ukázka rozvrhových akcí[2]

Čas výuky RA je možné definovat dvěma způsoby. Nejčastěji je určen intervalem vyučovacích hodin dané časové řady, která je zobrazena v horní části rozvrhu na obrázku 2.2. Časová řada obsahuje seznam vyučovacích hodin a definuje jejich intervaly skutečného času. Druhý způsob definuje čas RA přímo skutečným časem, jehož intervaly většinou neodpovídají intervalům vyučovacích hodin časové řady. Tento způsob je použit například u rozvrhových akcí tělesné výchovy.

[1] https://portal.zcu.cz
[2] https://portal.zcu.cz

Počet studentů na RA je omezen maximálním počtem studentů, pro který je RA vypsána. Možná kapacita RA nemusí být vždy plně obsazena, jelikož se na ní může zapsat méně studentů než kolik povoluje. Ne všichni zapsaní studenti pak danou RA navštěvují. Proto jsou některé akce umístěny do místností s menší kapacitou, než je počet studentů, kteří se na ni mohou zapsat.

2.2.3 Simulace přesunu výuky

Simulace přesunu výuky umožňuje pro rozvrhové akce definovat nové místnosti, do kterých se může výuka přestěhovat. RA se mohou přesouvat do jiných místností na stejný čas i den nebo může dojít ke změně času či dne.

Jak se budou RA přesouvat určuje účel, kvůli kterému je výuka přesouvána. RA mohou být přesouvány do jiných budov, nebo mohou být ve stávající budovách pouze přeskupeny do méně místností a díky tomu optimalizovat využití i vytížení místností. Využití místností určuje, jak moc je místnost využívána pro výuku z pohledu obsazenosti rozvrhovými akcemi v rámci možné výuky během celého semestru. Vytížení místností definuje, jak moc je využitá kapacita místnosti v poměru s počtem studentů na RA.

RA se mohou přesouvat do prázdných místností nebo mohou být přesunuty mezi již existující výuku. Prázdné místnosti mohou být vytvořeny duplikováním již existujících místností nebo mohou být vytvořeny jako nové fiktivní místnosti, které reprezentují například novou výstavbu.

2.3 Požadavky na aplikaci

2.3.1 Funkce aplikace

Hlavní funkcí aplikace bude simulace přesunu výuky z jedné množiny místností do jiné. Uživatel aplikace bude mít možnost tyto množiny definovat a určit, které rozvrhové akce se budou přesouvat. Aplikace proto bude umožňovat naplánovat výuku do nových místností nebo ji ve stávajících optimalizovat. Dále bude umožňovat sestěhovat výuku předmětů dané katedry nebo sestěhovat výuku studentů vybrané fakulty.

Přesun výuky bude možné naplánovat ve fázích, kde pro každou fázi půjde defino-

vat množinu místností, do kterých se bude výuka dané fáze přesouvat, a množinu RA, která se bude přesouvat.

Aplikace bude přesouvat pouze standardní výuku. Zkouškové ani přijímačkové termíny nebudou předmětem jejího zájmu. Přesouvat se budou pouze výukové RA, které mají definovanou místnost a čas výuky a jsou na nich zapsáni studenti. Pokud se v jednu dobu učí více různých RA v jedné místnosti a tvoří tak skupinu RA, přesouvá se tato skupina celá do nové místnosti.

Pro simulaci přesunu bude vhodné použít starší data, například loňská, protože někteří letošní studenti ještě nemusí mít zapsanou veškerou výuku a data pro přesun by byla nekompletní.

Nová místnost je vybírána podle kapacity původní místnosti a počtu studentů, kteří na dané RA studovali. Pokud byla původní místnost větší než počet studentů na RA, bude se výuka stěhovat do místnosti o kapacitě počtu studentů. Pokud byla původní místnost menší než počet studentů, přestěhuje se RA do místnosti, která je stejně minimálně velká jako původní místnost.

Pro fázi stěhování bude možné definovat podmínky, podle kterých se vyberou RA na stěhování. Mezi těmito podmínkami budou například:

- stěhování výuky zvolené katedry,
- stěhování výuky, kde jsou zastoupeni studenti dané fakulty z $x\%$,
- stěhování výuky z vybraného typu místnosti,
- stěhování výuky z místností s určeným intervalem kapacit,
- stěhování výuky z konkrétního seznamu místností.

Dále bude možné určit podmínky, podle kterých se budou vybrané RA stěhovat. Budou to například:

- zachovat den i čas RA,
- umožnit změnit čas RA,
- umožnit změnit den RA,
- začít s nejobsazenějšími RA.

Aplikace bude zobrazovat přesunuté RA a umožní porovnat stav před a po přestěhování. K tomuto účelu bude zobrazovat množinu grafů, která ukáže, jak se po přestěhování změnilo využití a vytížení místností vůči stavu před stěhováním.

2.3.2 Možnosti uživatele

Do aplikace bude mít přístup pouze přihlášený uživatel, který bude mít uživatelskou roli *superrozvrhář*, *prorektor* nebo *management*.

V aplikaci bude mít možnost provádět následující činnosti:

- pojmenovat a uložit stěhování,
- navrhnout více stěhování,
- navrhnout fáze stěhování,
- vybrat množinu místností, jejichž RA se budou stěhovat,
- vybrat množinu místností, do které se budou RA stěhovat,
- navrhnout fiktivní místnosti, do kterých se bude stěhovat,
- vyfiltrovat rozvrhové akce pro stěhování,
- vybrat, v jakém pořadí se budou skupiny RA stěhovat,
- vybrat podmínky stěhování,
- zobrazit přestěhovanou výuku,
- zobrazit grafické výstupy.

2.3.3 Výstupy aplikace

Aplikace bude obsahovat základní statistické přehledy, které budou zobrazovat, jak se simulace stěhování povedla, zda byl změněn čas nebo den výuky a zda se některé rozvrhové akce nepovedlo přestěhovat. Dále bude umožňovat zobrazit grafy, které vykreslí hlavní ukazatele vytížení či využití místností. Tyto výstupy budou zobrazeny pro stavy před a po stěhování a budou umožňovat porovnání obou stavů.

Mezi základními grafy, které budou v aplikaci zobrazeny, se budou nacházet například:

- počet místností dle procentní vytíženosti,
- procento využití dle kapacity místnosti.

Dále bude aplikace zobrazovat, do jakých místností a na jaké dny a časy se jednotlivé RA přesunuly.

3 Problematika tvorby rozvrhů na univerzitě

Kapitola se zabývá tvorbou rozvrhů na univerzitě. Shrnuje základní problémy, které nastávají při tvorbě rozvrhů nastávají, a porovnává tvorbu nových rozvrhů s přesunem již existující výuky do jiných místností.

3.1 Problematika tvorby rozvrhů

Přesun výuky na univerzitě úzce souvisí s tvorbou rozvrhů. Oba problémy využívají podobné techniky, které vedou k vytvoření nových rozvrhů. Proto je vhodné tuto oblast prozkoumat a zjistit, na které problémy se při přesunu výuky zaměřit.

Problematika rozvrhování se zabývá přiřazením rozvrhových akcí do volných časových slotů a vhodných místností. Tyto RA jsou pak přiřazeny vyučujícím a skupinám studentů.

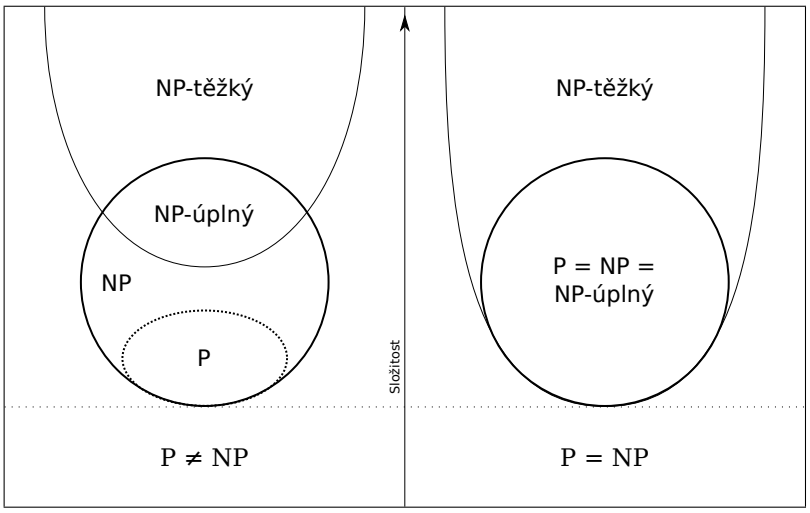
Rozvrhování je možné provádět manuálně. Rozvrhář definované RA ručně přesouvá do místností a určuje, ve který den a čas se bude jejich výuka konat. Tato možnost je časově náročná a vyžaduje zkušenosti s tvorbou rozvrhů. Druhou možností je automatická tvorba rozvrhů, která je vykonávána na základě vybraného algoritmu. Je podstatně rychlejší a může dosahovat lepších výsledků. Protože je však rozvrhování NP-úplný problém, může být toto automatické řešení problematické.

3.1.1 NP-úplný problém

NP-úplný (nedeteministický polynomiální) problém označuje třídu složitosti problémů, pro kterou neumíme vytvořit algoritmus, který by našel řešení v polynomiálním čase. Pokud ale řešení uhádneme, dokážeme v polynomiálním čase ověřit jeho správnost. [17]

Dále je definován tím, že všechny NP problémy lze na NP-úplný problém převést v polynomiálním čase. V současné době není jisté, zda se problémy z kategorie

P (problémy řešitelné v polynomiálním čase) rovnají, nebo nerovnají NP problémům. Na obrázku 3.1 jsou možné vztahy těchto tříd zobrazeny. Všeobecně se předpokládá, že $P \neq NP$, což znamená, že neexistuje žádný algoritmus, který by NP problém vyřešil v polynomiálním čase. [17] [29]



Obrázek 3.1: Vztah mezi P a NP problémy[1]

3.1.2 Typy

Existují tři základní typy, podle kterých lze definovat rozvrhování: [25]

- klasický školní rozvrh – definuje rozvrhování pro pravidelné týdenní hodiny pro jednu skupinu studentů, typickým příkladem je rozvrh na základní nebo střední škole,
- univerzitní výuka – definuje rozvrhování formou kurzů a snaží se o minimalizaci překryvů mezi jednotlivými hodinami, které mají společné studenty,
- zkouškové termíny – definuje rozvrhové akce s minimálním překryvem pro společné studenty, termíny stejného předmětu se snaží rozprostřít do celého zkouškového období.

Tato práce se zaměří pouze na rozvrhování univerzitní výuky, protože tento typ definuje výuku, u které bude prováděna simulace přesunu do nových místností.

[1] https://upload.wikimedia.org/wikipedia/commons/a/a0/P_np_np-complete_np-hard.svg

3.1.3 Definování omezení

Existují dva typy omezení, se kterými se při tvorbě rozvrhů můžeme setkat. Jedním typem jsou pevná omezení (hard constraints), která by neměla být porušena, protože by tím došlo k nesplnitelným situacím. Druhým typem jsou měkká omezení (soft constraints), která určují preference a jejich porušení není tak závažné.

Mezi pevnými omezeními mohou být například [27] [9]:

- vyučující nemůže učit dvě více rozvrhových akcí v jeden čas,
- kapacita místnosti je větší nebo rovno počtu studentů na RA,
- vlastnosti místnosti musí odpovídat potřebám výuky,
- student nenavštěvuje více než jednu RA v jeden čas,
- rozvrhová akce bude učena ve stejné místnosti.

Měkká omezení mohou být definována následujícím způsobem [27] [9]:

- studenti by neměli mít pouze jednu RA za den,
- studenti by neměli mít více než dvě RA po sobě,
- studenti by neměli mít RA rozvrhované na poslední časový slot,
- RA by měli být rozprostřeny v celém týdnu,
- vyučující si mohou určit, kdy chtějí nebo nechtějí učit,
- vzdálenost mezi místnostmi dvou následujících RA by měla být minimalizovaná.

Omezení nejsou vždy pro všechny případy stejná, ale jsou přizpůsobena aktuálním požadavků univerzity a jejím pravidlům definujícím výuku. Co může být pro jednu školu omezením, které by nemělo být porušeno, může být na jiné škole přípustnou situací. Například podle výše uvedených pevných omezeních by studenti neměli mít více než jednu RA v jeden čas. Tato podmínka není vždy splněna, jelikož si studenti sami vybírají předměty, které budou studovat a některé volitelné předměty jsou vyučovány ve stejnou dobu.

3.1.4 Vyhledávací i optimalizační problém

Tvorbu rozvrhů je možné definovat jako problém vyhledávání v případech, kdy řešení problému spočívá ve splnění všech definovaných omezení. Druhou možností je definovat tento problém jako optimalizační. Taková situace nastává v případě, kdy při řešení problému uspokojujeme pevná omezení a snaží se minimalizovat měkká omezení. [25]

Pro oba definované typy řešíme základní problém, u kterého v případě vyhledávání rozhodujeme, zda existuje řešení tohoto problému, a v případě optimalizace, zda existuje řešení, které má hodnotu cílové funkce. [25]

3.2 Porovnání se simulací přesunu výuky

Vytváření rozvrhů i přesun výuky mají mnoho společných znaků. Přesto se mezi nimi najdou rozdíly, které tyto problémy od sebe odlišují.

Hlavním rozdílem je fakt, že při přesunu výuky není rozvrh vytvářen celý od začátku, ale jeho struktura je již v určité formě definována. Učitelé i studenti jsou již zvyklí, v který den a hodinu se jejich výuka koná, a proto nemusí být žádoucí tuto dobu měnit. Z tohoto důvodu nemusí chtít uživatel při simulaci stěhování výuky celý rozvrh přeházet, ale může vyžadovat pouze změnu místností výuky. Tato situace s sebou nese riziko, že se nenajde volná místnost, ve které by mohla být rozvrhová akce vyučována.

Na druhou stranu se díky takovému požadavku problém zjednodušuje, jelikož se vyhledávají pouze místnosti s volným časem ve stejnou dobu a stejný den, jaký byl u RA již definován, a následkem toho složitost problému klesá. Také to bude s největší pravděpodobností nejčastější případ, který při simulaci stěhování výuky nastane.

Dalším rozdílem je to, že při vytváření nových rozvrhů místnosti neobsahují žádné rozvrhové akce. Naopak u přesunu výuky se může výuka přesouvat do místností s již existující výukou a tím se snažit zaplnit volná místa.

Z těchto důvodů není problematika obou případů stejná a při volbě vhodného algoritmu je důležité na tyto rozdíly nezapomenout.

4 Algoritmy a optimalizační techniky tvorby rozvrhů

Tato kapitola shrnuje základní algoritmy a optimalizační techniky, které se používají při tvorbě univerzitních rozvrhů. Ve stručnosti představuje jednotlivé algoritmy a jejich použití pro rozvrhování. V závěru jsou algoritmy shrnuty a je zhodnoceno, zda jsou vhodné pro simulaci přesunu výuky.

4.1 Seznámení s algoritmy

Mnoho autorů zkoumalo různé druhy algoritmů, které by mohly být vhodné pro tvorbu univerzitních rozvrhů. Ve svých pracích zjišťovali, zda jsou algoritmy vhodné a implementovali jejich navržená řešení nad reálnými daty univerzit.

Pro řešení NP-úplných problémů je možné použít aproximační algoritmy nebo heuristiky. Aproximační algoritmy poskytují aproximaci optimálního řešení v polynomiálním čase. Toto řešení je přibližné a často obsahuje přípustnou chybu. Heuristické algoritmy jsou založeny na vyhledávání všech možných řešení, avšak nezaručují nalezení optimálního řešení. Protože jsou robustní, jsou velmi populární a široce používané. [17]

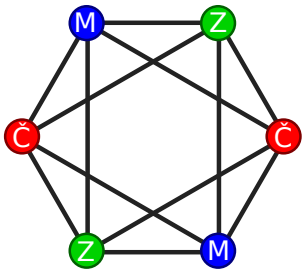
Nejčastěji zkoumané a používané algoritmy byly shrnuty ve vědeckých článcích i jiných pracích. Přístup k používaným algoritmům se časem vyvíjel. Dříve byly více zkoumané grafové algoritmy, v současnosti jsou populárnější algoritmy založené na evolučních přístupech či inspirované chováním zvířecích rojů a kolonií. [1] [5] [23] [25]

V následující části jsou některé algoritmy, které byly předmětem mnoha studií, představeny.

4.2 Obarvování grafu

Popis algoritmu

Problém obarvování grafu (Graph colouring) je jedním ze základních NP-úplných problémů. Pro neorientovaný graf $G = (V, E)$ problém spočívá v nalezení minimálního množství barev, které je potřeba použít pro obarvení všech vrcholů grafu tak, aby žádné dva sousední vrcholy neměly stejnou barvu. Na obrázku 4.1 je zobrazeno vhodné obarvení vrcholů minimálním počtem barev. Pro řešení tohoto problému bylo navrženo několik různých heuristických algoritmů. [25]



Obrázek 4.1: Ukázka možného obarvení grafu třemi barvami[1]

Použití pro rozvrhování

Metoda obarvování grafů byla pro rozvrhování výuky zkoumána v mnoha studiích. V jedné z nich byla použita pro rozvrhování kurzů se stejnou délkou (periodou). Pro každou rozvrhovou akci se vytvořil uzel. Všechny uzly rozvrhových akcí patřící jednomu kurzu se spojily hranou. Zároveň se vytvořily hrany mezi kurzy, které byli navštěvovány stejnými studenty. Realizovatelný rozvrh kurzu pak odpovídal obarvení uzlů tolika barvami, kolik bylo definováno period. Každý uzel byl obarven jednou barvou a žádné dva sousední uzly nesměly mít stejnou barvu. [6]

Metoda obarvování grafů byla dále vylepšována. Jedním z možných vylepšeních byl algoritmus, podle kterého bylo možné dělit vrcholy grafu podle pořadí tak, aby došlo ke snížení jeho chromatického čísla. Tímto způsobem byly specifikovány časy jednotlivých kurzů v rozvrzích fakulty. [26]

[1] https://upload.wikimedia.org/wikipedia/commons/b/b7/3-coloringEx.svg

4.3 Lineární programování

Popis algoritmu

Lineární programování (Linear Programming) specifikuje optimalizační problém, který je možné vyřešit v polynomiálním čase, zatímco 0/1-lineární programování (0/1-LP) a celočíselné lineární programování (ILP) jsou NP těžké problémy. Všechny tyto problémy mají společné omezení v podobě $A \cdot X = b$, kde A je matice a X a b jsou vektory. Jde o minimalizaci cílové funkce $X \cdot c^\top$, pro daný vektor c. Minimalizace je podle typu řešení prováděna buď na množině reálných čísel, $0, 1$, či celých čísel. Realizovatelné řešení je jakékoliv řešení, které splňuje všechna omezení. Mnoho NP-těžkých problémů je možné převést na 0/1-LP či ILP problém. Proto jsou problémy LP považovány za vzorové problémy, kterými se zabývá operační výzkum. [17] [29]

Použití pro rozvrhování

Lineární programování je možné použít pro rozvrhování univerzitních kurzů. Problém je rozložen na dvě části, první obsahuje rozvrhování časových period a druhá přidělení místností jednotlivým rozvrhovým akcím. Nejprve se vytvoří časový rozvrh. Jeho rozvrhové akce se následně přidělí jednotlivým místnostem. Pokud není možné najít vhodné umístění do místností, je původní rozvrh upraven a znovu přidělován do místností. [11]

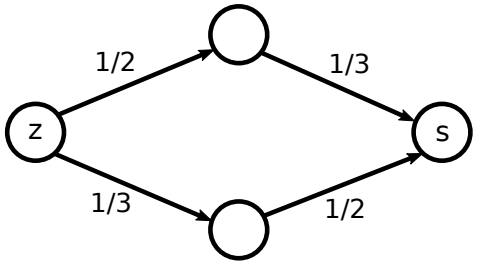
Rozvrhování bylo formulováno jako problém celočíselného lineárního programování i v další studii. Použitá metoda je založena na Lagrangeovské relaxaci spolu se subgradientní optimalizací. Podmínkou realizovatelného řešení umožňuje studentům navštěvovat pouze jeden předmět ve stejnou dobu. Studenti jsou rozděleni do skupin. V první fázi dojde k rozhodnutí, které skupiny studentů navštěvují konkrétní předmět. Ve druhé fázi se specifikuje potřebné zařízení pro předmět. Ve třetí fázi nastane určení místnosti pro výuku. [28]

4.4 Tok v síti

Popis algoritmu

Tok v síti (Network Flow) je algoritmus založený na vyhledávání v grafu. Graf $G = (V, E)$ s nezápornou kapacitní funkcí $u : E \to \mathbb{R}_\infty$ se nazývá síť. Tok v síti pak probíhá mezi dvěma odlišnými uzly, zdrojem z a stokem s. [13]

Na obrázku 4.2 je zobrazen tok v síti mezi zdrojem z a stokem s. Na hranách mezi vrcholy se nachází hodnota aktuálního toku a kapacity dané hrany.



Obrázek 4.2: Ukázka toku v síti

Použití pro rozvrhování

Algoritmus pro optimalizaci toku v síti byl použit pro řešení problému přidělení rozvrhových akcí do výukových místností. Navržené řešení obsahuje několik kroků. Nejprve jsou vygenerována potřebná data. Dále se vygeneruje síťový model, ve kterém proběhne aproximace řešení. Poté je řešení zhodnoceno a případně ručně modifikováno. Dokud není dosaženo uspokojivého řešení, některé kroky se opakují. [22]

Algoritmus byl dále použit při návrhu a implementaci systém pro podporu rozhodování (DDS) při tvorbě rozvrhů na univerzitě. Model síťového toku obsahuje tři úrovně. První úroveň *oddělení* obsahuje vrcholy reprezentující oddělení, které jsou napojeny na počáteční uzel. Druhá úroveň *fakulta/zaměstnanec* obsahuje vrcholy reprezentující zaměstnance oddělení, na které jsou tyto vrcholy napojeny. Třetí úroveň *velikost místnosti/čas* obsahuje vrcholy možných kombinací místností a časů. V případě, že místnost má dostatečnou kapacitu, je její vrchol napojen na uzel předchozí úrovně. Hrany, které vedou mezi druhou a třetí úrovní, reprezentují všechny možné výuky. [7]

4.5 Tabu vyhledávání

Popis algoritmu

Tabu vyhledávání (Tabu Search) je heuristika založená na lokálním vyhledávání. Algoritmy lokálního vyhledávání neprohledávají celý prostor proveditelných řešení, ale pouze jejich omezenou množinu. Nemají paměť, to znamená, že současné řešení nezávisí na celé historii vyhledávání, ale pouze na předchozím kroku. [17]

Řešení lokálního vyhledávání je založeno na jeho sousedech. Začíná z počátečního řešení, které může být získáno jinou technikou nebo náhodně vygenerováno. Ve smyčce přechází z jednoho řešení do řešení jeho souseda. V tabu vyhledávání algoritmus prohledává podmnožinu sousedů současného řešení. Soused s minimální hodnotou cílové funkce se stane novým řešením. Aby se zabránilo zacyklení, existuje tzv. *tabu seznam*, který obsahuje frontu posledních několika řešení, do kterých se algoritmus nesmí vracet. Finálního řešení je získáno po dosažení specifikovaného počtu cyklů nebo pokud cílová funkce dosáhne zadané hranice. [25]

Použití pro rozvrhování

Algoritmus tabu vyhledávání byl zkoumán pro použití při rozvrhování univerzitních kurzů. Na základě těchto výzkumů byl navrhnut model, který pracuje s pevnými i měkkými omezeními. Pro některá omezení dovoluje konflikty, aby tímto kompromisním přístupem došel k možnému řešení. [15] Tento algoritmus byl dále rozšířen o komplexnější případy, jako je například různá délka vyučovacích hodin. [16]

Dále byla pro rozvrhování kurzů použita hyper-heuristika tabu vyhledávání, která jako černá skříňka přijímala seznam nízkoúrovňových heuristik a vracela pro každou z nich zjištěné řešení. Tabu seznam nízkoúrovňových heuristik byl použit pro zakázání využití některých heuristik na určitou dobu. Pro evaluační funkci byla definována pevná i měkká omezení. [4]

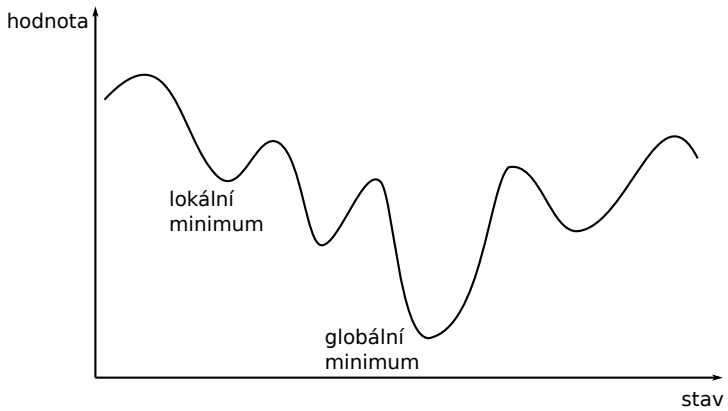
4.6 Simulované žíhání

Popis algoritmu

Simulované žíhání (Simulated Annealing) je založeno na lokálním vyhledávání, které je popsáno v předchozí podkapitole 4.5 v rámci algoritmu Tabu vyhledávání. Je rozšířeno o náhodné rozhodnutí, které umožňuje opuštění lokálního minima za účelem nalezení lepšího řešení. Algoritmus byl vytvořen podle skutečného fyzikálního procesu. Simuluje chladnutí horkých vibrujících atomů, které se po zahřátí na teplotu tání náhodně přeskupí a tím odstraní některé nedokonalosti materiálu. [17] [25]

Nejdůležitějšími prvky jsou počáteční teplota T_0, rychlost chlazení a a počet iterací. Nejprve je vytvořeno náhodné počáteční řešení a vhodně nastavena počáteční teplota T_0. V každé iteraci algoritmu se generuje náhodné sousední řešení, pro které je vypočtena cílová funkce. Pokud je rozdíl cílových funkcí současného a nového řešení $\Delta < 0$ je nové řešení akceptováno a stane se současným s pravděpodobností $e^{-\frac{\Delta}{T}}$, kde T je aktuální teplota. Teplota po pevně daném počtu iterací klesá. Nová teplota $T_n = a \times T_{n-1}$, kde $0 \leq a \leq 1$. Algoritmus končí, pokud se teplota limitně blíží 0 a žádné řešení, které by zvyšovalo cílovou funkci, již není akceptováno. [25]

Na obrázku 4.3 je zobrazen graf možných řešení při hledání minima algoritmem simulovaného žíhání. Při prohledávání se vypočtená řešení porovnávají mezi sebou a postupně konvergují k minimu zobrazenému na křivce, které je v ideálním případě globální.



Obrázek 4.3: Ukázka možných řešení algoritmu SA

Použití pro rozvrhování

Algoritmus simulovaného žíhání byl použit pro rozvrhování nad reálnými daty univerzity. Byl vylepšen zkombinováním s dalšími přístupy. Pro vygenerování počátečního řešení byl použit přístup založený na pravidlech, který předzpracoval data a tím poskytl dobré výchozí řešení. Dále bylo testováno použití geometrické i adaptivní chladicí a zahřívací funkce. Pro vyhodnocení vytvořeného řešení byla navrhnuta komplexní validační funkce. [9]

Pro rozvrhování univerzitních kurzů byl vytvořen třífázový algoritmus, v rámci kterého došlo ke zkombinování přístupu obarvování grafu, popsaného v podkapitole 4.2, s algoritmem simulovaného žíhání. V první fázi bylo algoritmem obarvování grafu vytvořeno počáteční řešení, které bylo ve druhé fázi vylepšeno algoritmem simulovaného žíhání. Třetí fáze spočívala ve finální optimalizaci prohazováním jednotlivých rozvrhových akcí algoritmem lokálního vyhledávání řízeného algoritmem simulovaného žíhání. [21]

4.7 Genetické algoritmy

Popis algoritmu

Genetický algoritmus (Genetic Algorithm) je založen na principu přírodního výběru. Tvoří ho tři základní operace: reprodukce, křížení a mutace. Jedinec je reprezentován řetězcem chromozomů (DNA). Během reprodukce jsou někteří jedinci zkopírováni do nové populace, při křížení dochází k prohození několika částí řetězce chromozomů mezi dvěma jedinci a mutace náhodně změní část DNA jednoho jedince. Tímto způsobem je vytvořena nová populace jedinců. [14] [17]

V tabulce 4.1 je zobrazeno vytvoření nové populace jedinců. Příklad ukazuje zkopírované jedince z předchozí populace, jejich vzájemné křížení a jejich mutaci.

Počáteční populace	1011 1001	1001 0010
Kopie	1011 1001	1001 0010
Křížení	1011 0010	1001 1001
Mutace	1010 1010	1001 1101

Tabulka 4.1: Příklad genetického algoritmu

Použití pro rozvrhování

Genetický algoritmus byl použit pro rozvrhování týdenních kurzů. Rozvrhování zahrnovalo navrhnutí rozvrhových akcí, vyučujících i místností a jejích přiřazení do jednotlivých časových slotů v průběhu týdne. Pro vyřešení tohoto problému byla nejprve definována pevná a měkká omezení. Během inicializace byla vytvořena náhodná populace vhodných řešení. Dále bylo vyhodnoceno porušení měkkých omezení. Poté následovala mutace, která přiřadila nové časové sloty nebo místnosti některým RA. Při křížení došlo k vytvoření potomka, který obsahoval RA svých rodičů. Algoritmus byl otestován na velké množství dat a poskytoval slibné výsledky. [10]

Dále byl genetický algoritmus použit pro přidělení kurzů do časových slotů a místností v situacích, ve kterých je potřeba se vypořádat s nevhodnými řešeními. Pro snížení složitosti problému byl zvolen sekvenční přístup. Každé RA byla nejprve přiřazena místnost a poté časový slot. Algoritmus začal od RA, které měli nejvíce studentů, jelikož bylo předpokládáno, že budou tyto akce problematické. Rozvrh byl vytvořen inkrementálně a následně byl ve dvou fázích ověřen. Nejprve bylo zkontrolováno, zda nedošlo k porušení pevných omezení a následně byla ověřena měkká omezení. [3]

4.8 Umělá inteligence hejna

Umělá inteligence hejna se zabývá interakcí jednotlivých členů společenství, tzv. agentů. Pracuje s modelováním chování včelích rojů, mravenčích kolonií nebo imunitních systémů. Staví na dvou základních principech, kterými jsou sebeorganizace a přizpůsobení se aktuálnímu prostředí. [20]

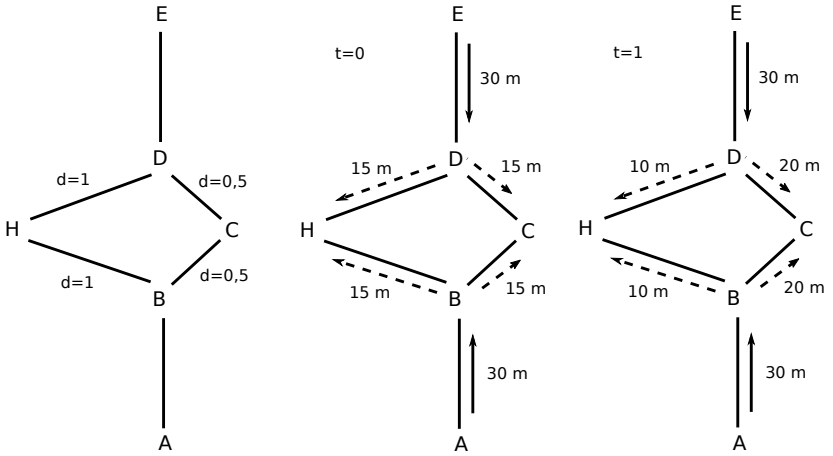
4.8.1 Optimalizace mravenčí kolonií

Popis algoritmu

Algoritmus optimalizace mravenčí kolonií (Ant Colony Optimization) funguje na principu napodobení chování reálných mravenců. Základem je pozitivní zpětná vazba, která je tvořena feromonovou stopou, jíž mravenci za sebou zanechávají. Čím více mravenců zvolenou cestou projde, tím je stopa silnější a pravděpodobnost zvolení této cesty dalším mravencem se zvyšuje. Tímto způsobem jsou mravenci

schopni nalézt nejkratší cestu k potravě. Na rozdíl od reálných mravenců mají mravenci v algoritmu částečnou paměť a nejsou úplně slepí. [8]

Na obrázku 4.4 je zobrazen způsob fungování algoritmu. První část obrázku zobrazuje vzdálenost jednotlivých cest. Pokud cesta není pokryta feromony, mravenci si cestu vybírají se stejnou pravděpodobností, jak je zobrazeno v druhé části. Třetí část ukazuje, že si stopa začne být silnější u kratší cesty a mravenci ji pak volí s větší pravděpodobností.



Obrázek 4.4: Ukázka algoritmu optimalizace mravenčí kolonií [8]

Použití pro rozvrhování

Pro rozvrhování byla použita MIN-MAX verze algoritmu optimalizace mravenčí kolonií. Nejprve se vygeneroval graf, ve kterém se hledala optimální cesta. Přiřazení kurzů bylo ovlivňováno množstvím feromonů, které se pohybovalo v zadaném intervalu. [27]

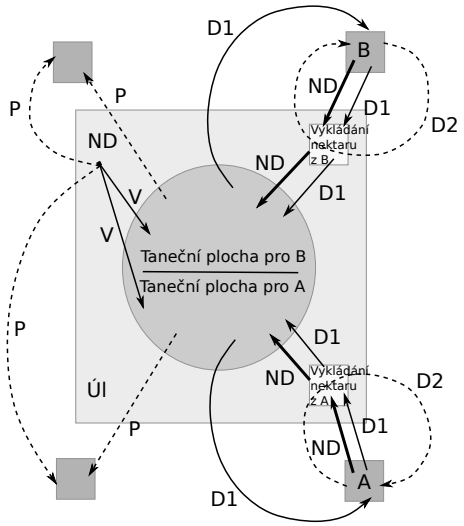
Algoritmus byl dále porovnáván i s jinými algoritmy, například algoritmem s simulovaného žíhání, či evolučním algoritmem. V této práci byl použitý algoritmus definován tak, že si mravenec zvolil jeden kurz z předem definovaného seznamu a na základě informace o porušení omezení a síle feromonů tento kurz umístil do rozvrhu. [24]

4.8.2 Umělý včelí roj

Popis algoritmu

Umělý včelí roj (Artificial Bee Colony) reprezentuje algoritmus, který je inspirován chování opravdového včelího roje. Základem jsou tři typy včel: dělnice, vyčkávající včely a průzkumnice. Dělnicím patří již nějaké existující řešení, které se snaží vylepšit lokálním vyhledáváním. Poté k sobě lákají vyčkávající včely, které si s vyšší pravděpodobností vyberou dělnici s lepším řešením. Vyčkávající včely se snaží dělnici vylepšit její řešení, a když se jim to povede, dělnice jejich řešení převezme. Pokud si dělnice několik iterací nezlepší své řešení, stane se průzkumníkem. Jako průzkumník si zvolí náhodně jiné řešení a znovu se začne chovat jako dělnice. [20]

Na obrázku 4.5 je zobrazeno chování včel, které shánějí potravu. Včela, která začíná jako nezaměstnaná dělnice (ND), se může stát průzkumníkem (P) nebo být vyčkávající včelou (V). Po obsazení nějakého zdroje dělnice láká další včely (D1) nebo nosí potravu do úlu (D2). Pokud je některý zdroj opuštěn, dělnice se jako nezaměstnaná včela (ND) stane průzkumníkem a hledá nový zdroj potravy. [20]



Obrázek 4.5: Ukázka chování včel [20]

Použití pro rozvrhování

Algoritmus umělého včelího roje byl použit pro rozvrhování. Obsahoval čtyři základní fáze: inicializační, fázi dělnic, vyčkávajících včel a průzkumnic. Inicializační

fáze vygenerovala náhodná řešení. V dalších fázích byly vyhledávány nové místnosti pro akce s porušenými omezeními. [19]

4.9 Zhodnocení algoritmů

Existuje mnoho různých přístupů, které byly pro tvorbu univerzitních rozvrhů prozkoumány. Vědecké práce se tomuto problému věnují již desítky let. Během této doby se vyprofilovalo několik ověřených základních algoritmů, které dokáží problém s určitou přesností vyřešit.

Hlavním problémem některých algoritmů bylo to, že při jejich návrhu nebylo počítáno s výukou, která by byla definována různě velkými časovými intervaly nebo dokonce neodpovídala vyučovacím hodinám. Taková výuka se na univerzitě vyskytuje a je nutné s ní počítat.

Dále algoritmy počítaly s prázdnými místnostmi, do kterých by bylo možné vytvořit celý nový rozvrh od začátku. Tato situace nemusí při stěhování výuky nastávat, protože se RA můžou přesouvat již mezi existující výuku a zaplňovat tak prázdná místa v rozvrzích místností.

Po prozkoumání výše uvedených algoritmů jsem došla k závěru, že jsou velmi komplexní a pro simulaci přesunu výuky zbytečně složité. Pokud bychom chtěli vytvářet nový rozvrh celé univerzity nebo některé její části úplně od začátku, bylo by vhodné některý z těchto algoritmů použít. V tomto případě se však RA přesouvají pouze do nových místností a optimalizace za cenu změny dne a času by byla spíše na škodu.

S ohledem na výše uvedené skutečnosti bude vhodnější v případě simulace přesunu výuky použít jednodušší algoritmus, který bude výuku přesouvat podle určených pravidel s důrazem na co největší zachování stejného dne a času přesouvané rozvrhové akce.

5 Tvorba rozvrhů a grafů

Kapitola se zabývá průzkumem vhodných technologií pro zobrazení univerzitních rozvrhů a tvorbu grafů.

5.1 Tvorba rozvrhů

Najít existující řešení pro zobrazování rozvrhů se ukázalo jako velmi těžký úkol. Nepodařilo se mi najít žádnou open-source knihovnu, která by se tomto problému věnovala. Většina vizualizačních nástrojů se věnovala zobrazení různému typů grafů a map.

Proto jsem došla k závěru, že nejlepší technologií pro zobrazování rozvrhů v portálových Java aplikacích bude spojení technologií HTML5[1], CSS3[2] a některé javascriptové knihovny, například JQuery[3].

Tyto technologie jsou použity již pro zobrazení rozvrhu v informačním systému IS/STAG, a proto je vhodné se tímto řešením inspirovat, případně je možné existující komponentu pro zobrazení rozvrhů použít.

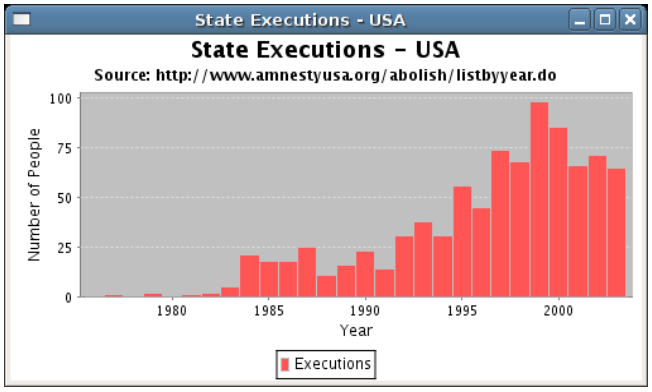
5.2 Tvorba grafů

Pro zobrazení výstupů ve formě grafů je vhodné použít knihovnu, která bude poskytovat zobrazení sloupcových a spojnicových grafů. Tato knihovna by měla být použitelná v Java portálové aplikaci. Následující knihovny umožňují takové grafy zobrazovat.

[1] https://www.w3.org/html/
[2] https://www.w3.org/Style/CSS/
[3] https://jquery.com/

5.2.1 JFreeChart

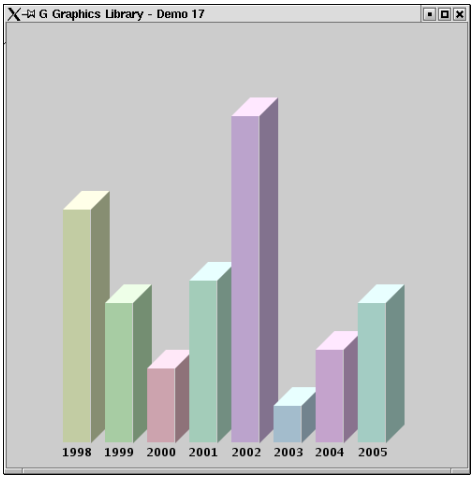
JFreeChart je open-source knihovna pro tvorbu grafů v Javě. Umožňuje vytvořit širokou škálu různých druhů grafů. Podporuje základní formáty výstupů (JPEG, PNG) i vektorovou grafiku (PDF, SVG). Ukázka je uvedena na obrázku 5.1. [18]



Obrázek 5.1: Ukázka sloupcového grafu vytvořeného v JFreeChart [18]

5.2.2 G

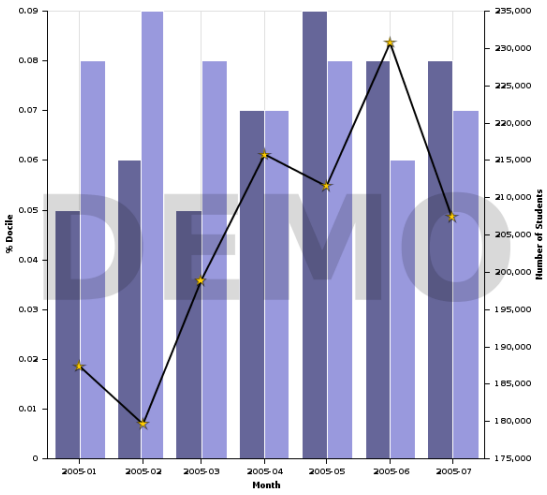
G je open-source grafická knihovna, která poskytuje možnost tvorby 2D grafů v Javě. Grafy jsou tvořeny objektově orientovaným způsobem. Umožňuje použití vrstvené grafiky a podporuje interakce. Výstupy tvoří základní rastrové obrázky, které je možné exportovat. Ukázka je uvedena na obrázku 5.2.[12]



Obrázek 5.2: Ukázka sloupcového grafu vytvořeného v G [12]

5.2.3 Big Faceless Graph Library

Big Faceless Graph Library je zpoplatněná knihovna, která umožňuje vytvořit různé typy grafů v Javě. Podporuje základní výstupní formáty (PNG, PDF, SVG). Funguje na principu webové služby a poskytuje rozhraní ve formě XML (eXtensible Markup Language)[4] nebo JSP Tag Library[5]. Ukázka je uvedena na obrázku 5.3. [2]



Obrázek 5.3: Ukázka sloupcového grafu vytvořeného v BFGL [2]

5.2.4 Výběr knihovny

Pro zobrazování grafů jsem zvolila knihovnu JFreeChart. Jejími výhodami jsou:

- open-source řešení
- podpora sloupcových grafů
- vhodná pro použití v Javě
- jednoduchá tvorba a úprava grafů
- již se v aplikaci IS/STAG používá

[4]https://www.w3.org/XML/
[5]https://docs.oracle.com/cd/A97630_01/java.920/a96657/taglibs.htm

6 Implementace

Kapitola popisuje praktickou část práce. Nejprve krátce seznamuje s informačním systémem IS/STAG, do kterého je aplikace zasazena. Poté představuje použitý algoritmus simulace přesunu výuky. Popisuje databázovou i aplikační část. Na závěr seznamuje s logováním aplikace.

6.1 Začlenění aplikace do IS/STAG

IS/STAG je informační systém studijní agendy vyvíjený na Západočeské univerzitě, který je zároveň používaný i na dalších dvanácti školách. Jedná se o portálovou Java aplikaci[1], která obsahuje mnoho různých portletů pro správu studia studentů. Uchazečům umožňuje zadávat přihlášky ke studiu, studenti si mohou zapisovat předměty a zobrazovat informace o studiu, vyučující mohou zapisovat známky a provádět mnoho dalších akcí.

Aplikace pro simulaci přesunu výuky byla vyvíjena jako součást tohoto informačního systému, a proto byly pro vývoj použity některé jeho knihovny. Konkrétní knihovny a jejich použití bude uvedeno dále v práci v podkapitole 6.4, která se zabývá strukturou webové části aplikace.

6.2 Simulační algoritmus

Algoritmus pro simulaci přesunu výuky je tvořen uloženou procedurou. Je volán zvlášť pro každou fázi stěhování. Nejprve se snaží přestěhovat RA, které jsou definovány skutečným časem a poté stěhuje RA definované vyučovací hodinou.

Pseudokód první části algoritmu se nachází v ukázce 1. Popisuje simulaci přesunu rozvrhových akcí, které jsou definovány skutečným časem, tj. neodpovídají vyučovacím hodinám, například výuka tělesné výchovy. U těchto RA se zachovává stejný čas.

[1] https://www.jcp.org/en/jsr/detail?id=168

Ukázka 1: Pseudokód algoritmu pro simulaci přesunu výuky

```
for d = 0 to p_den
    for i in
        select RA a jejich novy cas
        where
            urcene skutecnym casem
            jeste nebyly prestehovane
            odpovida id i faze stehovani
        order by nejvetsi kapacita

        for j in
            select mistnosti
            where
                mistnosti, do kterych se bude stehovat
                maji dostatecnou kapacitu
                maji volno v urceny novy cas a den
                odpovida id i faze stehovani
            order by kapacita mistnosti

                uloz do nejmensi vhodne mistnosti
                uloz i spolecne vyucovane akce
```

Do procedury nejprve přijdou parametry, které určí, jaký seznam uložených RA z jaké fáze se bude stěhovat. První cyklus určuje hodnotu pro výpočet posunu dne v týdnu. Pokud je parametr `p_den` roven nule, proběhne cyklus pouze jednou a den výuky se proto měnit nebude. Jinak se hodnota podle proměnné `d` posouvá o den dopředu nebo zpátky a pak se hledá volný čas v těchto dnech. Výuka v pracovním týdnu se přesouvá pouze na dny v pracovním týdnu, výuka o víkendech se přesouvá pouze na dny víkendu.

V dalším cyklu se prochází nalezené RA, pro které je vypočten den a čas, na který chceme RA přesunout. Pro každou takovou RA nalezneme seznam vhodných místností, které mají dostatečnou kapacitu a stejný typ a v cyklu pro ně ověříme zda mají ve vypočtený den a čas volné místo. Vybereme první nejmenší vhodnou místnost, do které RA přesuneme. Pokud je ve stejnou dobu ve stejné místnosti učena i další společná vyučovaná akce, přesuneme do nové místnosti i tuto RA.

Druhá část algoritmu funguje na stejném principu. Přesouvá RA, které jsou definované vyučovacími hodinami. Jeho pseudokód je uveden v ukázce 2. Na rozdíl od první části může změnit i čas. Pokud hodnota parametru `p_hodina` není rovna nule, může čas RA přesouvat na jiné vyučovací hodiny. V ostatních částech se algoritmus chová stejně jako jeho první část.

Ukázka 2: Pseudokód algoritmu pro simulaci přesunu výuky

```
for d = 0 to p_den
    for k = 0 to p_hodina
        for i in
            select RA a jejich novy cas
            where
                urcene vyucovaci hodinou
                jeste nebyly prestehovane
                odpovida id i faze stehovani
            order by nejvetsi kapacita

            for j in
                select mistnosti
                where
                    mistnosti, do kterych se bude stehovat
                    maji dostatecnou kapacitu
                    maji volno v urceny novy cas a den
                    odpovida id i faze stehovani
                order by kapacita mistnosti

                    uloz do nejmensi vhodne mistnosti
                    uloz i spolecne vyucovane akce
```

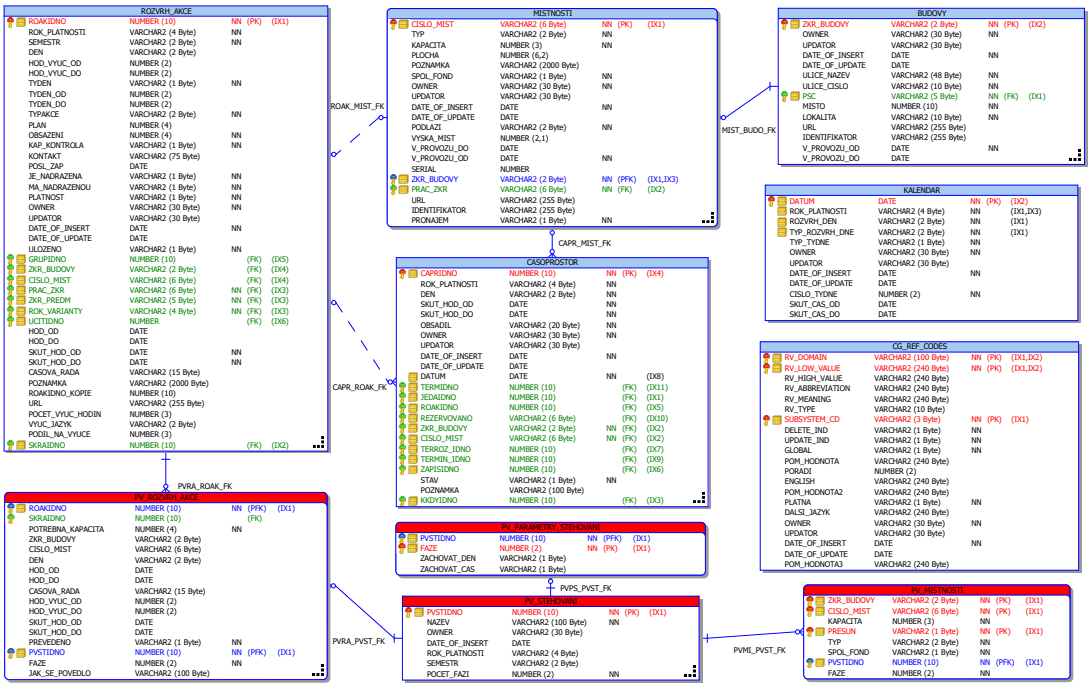
6.3 Databáze

Jelikož je aplikace integrovaná do informačního systému IS/STAG, používá stejnou databázi jako tento systém. Jedná se o databázi Oracle 12c[2], která je aktuálně ve verzi 12.1.0.2.

6.3.1 Použité tabulky

Aplikace používá nejen nové tabulky, které pro simulaci přesunu výuky potřebuje, ale využívá i některé stávající tabulky ze sytému IS/STAG, ze kterých čerpá data nebo je používá k ověřování dat při simulaci.

Na obrázku 6.1 se nachází zobrazení vztahů mezi jednotlivými použitými tabulkami. Nové tabulky, které byly vytvořeny pro potřeby simulace stěhování, jsou označeny prefixem `PV` a mají červené záhlaví. Větší verze obrázku se nachází v příloze A.



Obrázek 6.1: ERA diagram zobrazující strukturu tabulek v databázi

V následující části jsou popsány jednotlivé tabulky, které jsou zobrazeny na obrázeku 6.1. Zároveň je u nich vysvětlena jejich funkce, kterou zastávají v simulační

[2]http://www.oracle.com/us/corporate/features/database-12c/index.html

aplikaci.

CG_REF_CODES

Tabulka `CG_REF_CODES` je číselníkovou tabulkou, která obsahuje základní používané číselníky (domény). Pro jednotlivé domény vyjmenovává všechny možné hodnoty. V rámci této aplikace byly použity domény:

- `ANO_NE` – pro povinné položky, které mohou nabývat pouze hodnoty `"A"` nebo `"N"`,
- `DEN` – seznam jednotlivých dnů v týdnu,
- `FAKULTA` – seznam fakult univerzity,
- `LOKALITA` – seznam lokalit města, ve kterých se nacházejí budovy univerzity,
- `SEMESTR` – hodnota `"ZS"` pro zimní semestr a `"LS"` pro letní semestr,
- `TYP_UCEB` – seznam typů učeben univerzity, například laboratoř či posluchárna.

Rozsáhlejší číselníky, které obsahují mnoho položek, mají samostatné číselníkové tabulky. Příkladem takového číselníku je tabulka `CIS_PRACOVIST`, která uchovává seznam všech pracovišť univerzity.

KALENDAR

Tabulka `KALENDAR` obsahuje základní kalendářové informace akademického roku. V aplikaci je použita pro ověření, zda je ve zvolenou dobu ve vybrané místnosti volné místo pro stěhovanou RA. Algoritmus k tomu používá položky `CISLO_TYDNE`, `TYP_TYDNE`, `ROZVRH_DNE`, `DATUM` a `ROK_PLATNOSTI`, kterými vyhledává konkrétní datum pro ověření v časoprostoru.

BUDOVY

Tabulka `BUDOVY` obsahuje základní informace o budovách univerzity. Pro potřeby této aplikace je potřeba jen položka `LOKALITA` udržující polohu v rámci města, ve kterém se nachází. Je používána pro vyhledávání místností univerzity.

MISTNOSTI

Tabulka `MISTNOSTI` obsahuje potřebné informace o místnostech univerzity. Pro potřeby této aplikace je nutné získat pouze některé z těchto informací. Jedná se o položky `ZKR_BUDOVY`, `CISLO_MIST`, `KAPACITA`, `TYP` a `SPOL_FOND`. Z těchto položek jsou čerpány informace do stejně pojmenovaných položek v tabulce `PV_MISTNOSTI`, a proto mají tyto položky stejný význam.

ROZVRH_AKCE

Tabulka `ROZVRH_AKCE` obsahuje informace o rozvrhových akcích, které se na univerzitě vyučují. Obsahuje mnoho různých položek, ze kterých je pro aplikaci podstatná jen jejich podmnožina.

Zahrnuje základní položky, které se vztahují k RA a jsou stejné jako v tabulce `PV_ROZVRH_AKCE`, ale na rozdíl od nich obsahuje informace o původních nepřestěhovaných RA. To znamená, že všechny položky, které ve výše uvedené tabulce uvádí informace po přestěhování, v této tabulce obsahují data před přestěhováním.

Dále jsou navíc pro algoritmus stěhování zjišťovány informace z položek `TYDEN`, `TYDEN_OD` a `TYDEN_DO`, které pomáhají vyhledávat interval platnosti RA, ve kterém se vyučuje.

CASOPROSTOR

Tabulka `CASOPROSTOR` obsahuje informace o obsazených časech v místnostech univerzity. V aplikaci se používá pro ověření, zda je ve zvolenou dobu ve vybrané místnosti volné místo pro stěhovanou RA. Algoritmus proto pracuje s položkami `ZKR_BUDOVY`, `CISLO_MIST`, `DATUM`, `ROAKIDNO`, `SKUT_HOD_OD` a `SKUT_HOD_DO`, přes které je napojena na tabulky s místnostmi a rozvrhovými akcemi.

PV_STEHOVANI

Tabulka `PV_STEHOVANI` slouží k uložení základních informací o simulaci stěhování. Uživatel má možnost uložit si vytvořené stěhování a v budoucnu se k němu znovu vrátit.

Obsahuje položky:

- `PVSTIDNO` – ID záznamu, které je zároveň primárním klíčem a je automaticky vyplňováno ze sekvence `PVST_SEQ` triggrem `TR_PVST_BRI`,
- `NAZEV` – povinná položka, která obsahuje pojmenování simulace stěhování,
- `OWNER` – uživatel, který záznam o stěhování vytvořil,
- `DATE_OF_INSERT` – datum a čas založení stěhování,
- `SEMESTR` – hodnota semestru z číselníku `CG_REF_CODES` v doméně `SEMESTR`, jehož RA jsou plánovány pro simulaci přesunu,
- `ROK_PLATNOSTI` – rok platnosti RA, které budou přesouvány,
- `POCET_FAZI` – celkový počet fází, které jsou pro simulaci stěhování výuky vytvořeny.

PV_PARAMETRY_STEHOVANI

Tabulka `PV_PARAMETRY_STEHOVANI` ukládají položky, které ovlivňují jednotlivé fáze simulace stěhování výuky. Primární klíč je tvořen sloupečky `PVSTIDNO` a `FAZE`.

- `PVSTIDNO` – cizí klíč do tabulky `PV_STEHOVANI`, který parametry přiřazuje do konkrétní množiny uložených dat,
- `FAZE` – označuje fázi stěhování, kterou budou parametry ovlivňovat,
- `ZACHOVAT_DEN` – parametr pro zachování dne RA může mít nastavenou hodnotu podle domény `ANO_NE`, v případě hodnoty `"A"` se RA bude rozvrhovat na stejný den jako byla před přesunem,
- `ZACHOVAT_CAS` – parametr pro zachování času RA může mít nastavenou hodnotu podle domény `ANO_NE`, v případě hodnoty `"A"` se RA bude rozvrhovat na stejný čas jako byla před přesunem.

PV_MISTNOSTI

Tabulka `PV_MISTNOSTI` slouží pro uchování místností, ze kterých nebo do kterých se RA přesouvají. Všechny její položky jsou povinné. Informace jsou čerpány

z tabulky MISTNOSTI, do které je možné se přes cizí klíč ze sloupečků ZKR_BUDOVY a CISLO_MIST podívat na další informace o místnosti. Primární klíč tabulky je tvořen kombinací hodnot sloupečků ZKR_BUDOVY, CISLO_MIST, PRESUN a PVSTIDNO.

Tabulka obsahuje následující položky:

- ZKR_BUDOVY – informace o budově, ve které leží uvedená místnost,
- CISLO_MIST – číslo dveří konkrétní místnosti,
- KAPACITA – počet židlí v místnosti,
- PRESUN – hodnota "F" určuje, že se z místnosti RA přesouvají pryč, a hodnota "T" označuje, že se RA přesouvají do této místnosti,
- TYP – typ místnosti, například laboratoř nebo posluchárna, se bere z číselníku z tabulky CG_REF_CODES z domény TYP_UCEB,
- SPOL_FOND – hodnota "A"/"N" označuje, zda se místnost nachází ve společném fondu univerzity nebo patří konkrétnímu pracovišti,
- PVSTIDNO – cizí klíč do tabulky PV_STEHOVANI, který místnost přiřazuje ke konkrétní množině uložených dat,
- FAZE – pro místnosti, do kterých se bude výuka přesouvat označuje, v jaké fázi se má konkrétní místnost použít.

Oproti tabulce MISTNOSTI, lze do této tabulky přidat i fiktivní místnosti, do kterých pak mohou být RA přesouvány.

PV_ROZVRH_AKCE

Tabulka PV_ROZVRH_AKCE uchovává rozvrhové akce, které se vyučují v místnostech definovaných v tabulce PV_MISTNOSTI označené v položce PRESUN hodnotou "F". Tyto RA jsou určené pro simulaci přesunu do nových místností. Primárním klíčem v tabulce je skupina položek ROAKIDNO a PVSTIDNO, které stačí k identifikaci záznamu. Data jsou získána z tabulky ROZVRH_AKCE.

Položky, které se nachází v tabulce jsou:

- ROAKIDNO – ID rozvrhové akce, které je zároveň cizím klíčem do tabulky ROZVRH_AKCE,

- `SKRAIDNO` – ID skupiny RA,
- `POTREBNA_KAPACITA` – kapacita vypočtená jako minimum z:
  - kapacity RA v tabulce `ROZVRH_AKCE`,
  - obsazení studenty RA v tabulce `ROZVRH_AKCE`,
  - kapacity původní místnosti, ze které se RA stěhuje.
- `ZKR_BUDOVY` – nová budova, kam je RA přestěhována,
- `CISLO_MIST` – nová místnost, do které je RA přestěhována,
- `DEN` – den v týdnu, na který je RA přestěhována,
- `HOD_OD` – počáteční čas intervalu, na který je RA, přestěhována,
- `HOD_DO` – konečné čas intervalu, na který je RA, přestěhována,
- `CASOVA_RADA` – časová řada vyučovacích hodin,
- `HOD_VYUC_OD` – počáteční vyučovací hodina, na kterou je RA přestěhována,
- `HOD_VYUC_DO` – konečná vyučovací hodina, na kterou je RA přestěhována,
- `SKUT_HOD_OD` – začátek intervalu skutečného času, na který je RA, přestěhována, používá se pouze pro RA definované skutečným časem, které není možné naplánovat na vyučovací hodinu,
- `SKUT_HOD_DO` – konec intervalu skutečného času, na který je RA, přestěhována, používá se pouze pro RA definované skutečným časem, které není možné naplánovat na vyučovací hodinu,
- `PREVEDENO` – příznak z domény `ANO_NE`, zda již byla RA převedena do nové místnosti
- `PVSTIDNO` – cizí klíč do tabulky `PV_STEHOVANI`, který RA přiřazuje do konkrétní množiny uložených dat,
- `FAZE` – označuje fázi, ve které se bude RA přesouvat, pokud je nastavena na hodnotu `0`, fáze ještě nebyla přidělena, a proto není možné RA stěhovat,
- `JAK_SE_POVEDLO` – informace o tom, zda se při stěhování zachoval den nebo čas původní RA.

6.3.2 Další databázové objekty

Aplikace kromě tabulek v databázi používá i další objekty, které jsou uvedené v následující části.

Triggery a sekvence

Pro potřeby aplikace byl vytvořen trigger `TR_PVST_BRI`, který před vložením nového stěhování do tabulky `PV_STEHOVANI` načte ze sekvence `PVST_SEQ` další hodnotu a přidá ji do položky `PVSTIDNO` jako identifikátor záznamu.

Procedury

Do databáze byla uložena procedura `PR_PV_SIMULACE_STEHOVANI_VYUKY`, která přijímá parametry `P_PVSTIDNO`, `P_FAZE`, `P_DEN` a `P_HODINA` a simuluje přesun výuky. Její algoritmus je popsán v podkapitole 6.2.

6.4 Struktura aplikace

Tato část se věnuje popisu vytvořené aplikace. Seznamuje s její základní strukturou a uvádí její části, u kterých popisuje jejich funkci.

6.4.1 Základní stuktura

Portálová aplikace

Jedná se o portletovou webovou Java aplikaci, která běží ve webovém kontejneru Apache Tomcat[3]. Splňuje standard JSR-268 (Java Portlet Specification 2.0)[4], což je norma, která standardizuje portletový kontejner a tvorbu portletových aplikací. Proto může být nasazena na jakýkoliv portál, který tuto normu splňuje.

Portál je webová aplikace, která je tvořena portlety. Umožňuje personalizovaný, přizpůsobený, zabezpečený a jednotný přístup do systému, který agreguje různé

[3]http://tomcat.apache.org
[4]https://www.jcp.org/en/jsr/detail?id=286

aplikace do jednoho uživatelského rozhraní. Portlet je webová komponenta, která se nachází na portálové stránce a zobrazuje uživateli své informace. Nemusí být na stránce sama, ale může se na ní vyskytovat spolu s dalšími portlety, které mohou patřit i jiným aplikacím.

Java

Aplikace byla vyvíjena v jazyce Java verze 1.8[5], a proto používá jeho nové syntaktické konstrukce. Jedná se například o lambda výrazy, streamy nebo nové typy datumových a časových objektů.

MVC

Aplikace simulace přesunu výuky byla vytvořena podle softwarové architektury MVC (Model-View-Controller)[6]. Model pracuje s uchováním dat aplikace, view (pohled) zajišťuje interakci s uživatelem a controller vše řídí. Díky tomuto konceptu jsou jednotlivé části aplikace od sebe oddělené přes komunikační rozhraní, a proto při změně v jedné části nebude potřeba měnit ostatní.

Spring

Aplikace pro svůj běh používá Spring framework[7]. Pro propojení všech částí je použit Spring IoC kontejner, který je spravuje. IoC (Inversion of Control) je koncept, který místo volání knihovních metod nechává vytvoření objektů na frameworku a tím udržuje třídy aplikace na sobě nezávislé. Všechny potřebné informace Spring získává z XML souborů, které obsahují data o implementacích použitých rozhraní. Všechny tyto soubory jsou uvedeny v konfiguračním souboru `applicationContext.xml`, který Spring při svém běhu používá.

Pro připojení do databáze byl použit modul Spring JDBC, který zajišťuje otevření i zavření databázového spojení. Jedná se o nadstavbu, která poskytuje `JdbcTemplate`, díky které je pak možné komunikovat s databází.

[5]https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf

[6]https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture

[7]https://spring.io/

Knihovny IS/STAG

Jelikož je aplikace nasazena do portálu IS/STAG, používá některé jeho knihovny. Jedná se o modul pro tvorbu a validaci formulářů. Díky tomu je možné formulář nadefinovat v XML souboru a pomocí tag library pak zobrazit uživateli. Knihovna zároveň kontroluje, zda jsou data, která jsou v něm odeslána, validní a pokud nejsou, nedovolí data uložit a zobrazí uživateli formulář znovu k jejich editaci. Dále používá knihovnu pro práci s číselníky, které jsou uložené v databázi nebo nadefinované přímo v aplikaci.

6.4.2 Model

Model umožňuje spravovat data aplikace. Je možné ho rozdělit na několik samostatných částí. Jsou to datové beany, sloužící k uchovávání dat, DAO objekty, které přistupují do databáze, a managery, umožňující například cachování dat. Jejich jednotlivé části jsou popsány v následující sekci.

Datové beany

Datové beany neboli POJO (Plain Old Java Object) jsou třídy, jejichž instance slouží k uchování dat aplikace. Obsahují pouze konstruktor, gettery a settery, díky kterým je možné do nich data ukládat a později pak číst.

PresunVyukyStehovani Třída slouží k uchování základních informací o stěhování. Obsahuje informace, které se ukládají do tabulky `PV_STEHOVANI` popsané v podkapitole 6.3.1.

PresunVyukySearchBean Třída pomáhá při vyhledávání dat v databázi. Načtou se do ní informace z vyhledávacího filtru, podle kterých se pak z databáze vrací vhodná data.

PresunVyukyMistnost Třída slouží k uchování dat o místnostech. Jelikož mají tabulky `MISTNOSTI` a `PV_MISTNOSTI`, které byly popsány v podkapitole 6.3.1, stejnou základní strukturu dat, slouží tento objekt data z obou tabulek.

PresunVyukyRozvrhovaAkce Třída uchovává data z tabulek `ROZVRH_AKCE` a `PV_ROZVRH_AKCE`, které jsou popsané v podkapitole 6.3.1, jelikož slouží k načtení základních údajů o RA z původní tabulky do tabulky určené pro přesouvané rozvrhové akce.

PresunVyukyVolbaRA Do třídy jsou načítány informace z formuláře, kterým jsou nastavovány fáze stěhování u jednotlivých rozvrhových akcí. Proto obsahuje hodnoty vybraných parametrů spolu s číslem fáze, která má být nalezeným RA nastavena.

PresunVyukyParametryStehovani Třída obsahuje parametry, podle kterých má být určená fáze stěhována.

PresunVyukySimulaceStehovani Třída obsahuje parametry, které jsou použity při volání uložené procedury popsané v podkapitole 6.2.

PresunVyukyStatistickeUkazatele Třída obsahuje statistiky, které jsou zobrazovány uživateli a slouží ke zhodnocení výsledků proběhlé simulace přesunů výuky.

PresunVyukyDataGrafu Třída obsahuje data, která slouží k vykreslení grafů výsledných statistických ukazatelů.

DAO

DAO (Data Access Object) je objekt, který umožňuje získávat a ukládat data. Nemusí to být jen z databáze, ale data se mohou získávat i z jiných zdrojů. V případě této aplikace se data načítají pouze z databáze.

Připojení do databáze je zajištěno přes hikari[8] datasource, který obsahuje connection pool. Proto se při dotazu do databáze nemusí vytvářet nová připojení, ale berou se již vytvořená přímo z connection poolu a po použití se do něj zase vrací. Tento datasource je definován v konfiguračním souboru `datasources.xml`.

[8]https://brettwooldridge.github.io/HikariCP/

PresunVyukyDAO Rozhraní slouží pro komunikaci s dalšími vrstvami aplikace. Umožňuje nezávislost ostatních tříd na konkrétní implementaci DAO objektu. Obsahuje signatury metod, které jsou použity pro vyhledávání, ukládání, aktualizaci a odstraňování dat z úložiště. Implementace tohoto rozhraní je definována v konfiguračním souboru `dao.xml`.

PresunVyukyDAOImpl Třída implementuje rozhraní `PresunVyukyDAO` a zároveň dědí třídu `NamedParameterJdbcDaoSupport`, která obsahuje základní metody pro připojení do databáze. Pro komunikaci s databází se pak zavolá metoda `getNamedParameterJdbcTemplate()`, která vrací objekt, nad nímž je možné zavolat metodu `query()` pro vyhledávání dat nebo `update()` pro aktualizaci dat v databázi. Obě metody berou jako parametr řetězec SQL dotazu. Zároveň je možné jako další parametr předat instanci třídy `BeanPropertySqlParameterSource`, do které se nahrají parametry SQL dotazu. Předání parametrů tímto způsobem zabraňuje možnému SQL injection[9], ve kterém se útočník snaží přes formulář do databáze dostat vlastní kód a tím z ní data získat nebo je dokonce smazat.

Při ručním volání SQL dotazů jsou získaná data ručně mapována v objektu `RowMapper` do datových beanů. Načítána jsou z objektu `ResultSet`, ve kterém data vrací databáze.

PresunVyukyStoredProcedure Třída slouží k volání uložené procedury. Z tohoto důvodu dětí třídu `StoredProcedure`, která obsahuje základní metody pro volání procedur z databáze. V konstruktoru definuje parametry procedury. Zároveň obsahuje metodu `execute()`, ve které z objektu třídy `PresunVyukySimulaceStehovani` popsané v předchozí části načítá parametry, které pak v mapě předává při volání metody pro spuštění procedury.

Manager

Manager tvoří nadstavbu nad DAO objektem. Umožňuje práci s daty přes volání metod z DAO objektu a zároveň může získaná data dále upravovat nebo cachovat v paměti pro pozdější znovupoužití.

PresunVyukyManager Rozhraní, které obsahuje signatury metod pro komunikaci s vyššími vrstvami. Implementace tohoto rozhraní je definována v konfigu-

[9] https://www.w3schools.com/sql/sql_injection.asp

račním souboru `managers.xml`.

PresunVyukyManagerImpl Třída implementuje rozhraní `PresunVyukyManager`. Obsahuje odkaz na instanci implementace rozhraní `PresunVyukyDAO`, jejíž metody pak volá. Tuto instanci získává přes setter díky konceptu IoC, který byl popsán v podkapitole 6.4.1. Zároveň některá data ukládá do statické cache, ze které je v případě úpravy pak zase odstraňuje a tím zajišťuje jejich aktuálnost.

6.4.3 Controller

Základní funkce controlleru jsou obsluha požadavků od uživatele a řízení běhu aplikace. V souboru `web.xml` najde základní nastavení aplikace. Obsahuje umístění aplikačního kontextu, který byl uveden v podkapitole 6.4.1, definování servletu `PageDispatcherServet`, který přijímá a obsluhuje požadavky od uživatele, či seznam chybových stránek spolu s jejich číselnými kódy.

Portlety

Aplikace obsahuje portlet `PresunVyukyPortlet`, jehož vlastnosti jsou definovány v konfiguračním souboru `portlets.xml`.

Tento portlet dědí třídu `GenericPagesFormsPortlet`, která obsahuje základní metody pro portlety používající aplikační stránky a formuláře. Dále implementuje rozhraní `HelpCenterAware`, které umožňuje nastavit odkaz na nápovědu k aplikaci.

Portlet vytváří instanci session objektu třídy `PresunVyukySessionBean`, který slouží k dlouhodobějšímu uložení dat. Takto uložená data jsou přístupná delší dobu a nemusí být znovu načítána v každém requestu aplikace. Základními daty, která jsou v tomto objektu uložena jsou například informace o aktuálním otevřeném stěhování nebo nastavovaná fáze stěhování.

Formuláře

Formuláře v aplikaci slouží k vyhledávání, ukládání nebo modifikaci dat v databázi. Pro jejich vytváření je použit modul na tvorbu formulářů ze systému

IS/STAG, který tuto činnost usnadňuje. Položky formulářů, jejich popisky, datové typy a případné přednastavené hodnoty se pak definují v souboru `webForms.xml`.

Takto nadefinovaný formulář se pak propojí s obsluhující třídou v konfiguračním souboru `forms.xml`. Tyto třídy dědí třídu `WebForms`, která obsahuje metody pro práci s formuláři. Umožňuje například ve formuláři inicializovat data, upravit je před použitím, zpracovat formulář či vrátit formulář k úpravě v případě nevalidních dat.

Následující část obsahuje třídy, které tyto formuláře zpracovávají a seznamuje s jejich činností. Všechny dále zmíněné formuláře při svém zpracování ukládají data do databáze přes některou z metod definovanou v instanci implementace rozhraní `PresunVyukyManager`.

PresunVyukyStehovaniForm Třída zpracovává formulář, který ukládá nové stěhování. Na začátku inicializuje akademický rok stěhování na loňskou hodnotu, jelikož taková data budou v databázi určitě kompletní. Při zpracování data ukládá do databáze. Po úspěšném uložení z databáze získává ID stěhování, které uloží do session pro potřeby další práce s aplikací. Na závěr uživatele informuje, zda bylo uložení dat úspěšné.

PresunVyukyFiltrForm Třída definuje formulář pro vyhledávání místností, které bude chtít uživatel použít pro přesun. Existuje ve dvou instancích, kdy jedna vyhledává místnosti, ze kterých se budou rozvrhové akce stěhovat, a druhá umožňuje vyhledat místnosti, do kterých se budou RA stěhovat. Na začátku inicializuje příznak, který definuje, jaký z těchto dvou formulářů to konkrétně je. Nalezená data načítá do session k zobrazení uživateli.

PresunVyukyFiktivniMistnostForm Třída zpracovává formulář, přes který je možné definovat fiktivní místnosti. Jelikož se RA nemusí přesouvat pouze do existujících místností, ale uživatel může chtít simulovat přesun například do nové výstavby, je žádoucí tuto funkčnost uživateli poskytnout. Aplikace umožňuje nadefinovat fiktivní místnosti, do kterých se bude přesouvat. Místnosti, ze kterých se RA přesouvají, by neměly smysl, protože by v nich žádné RA nadefinovány nebyly.

Při zpracování formulář kontroluje, zda již takto pojmenovaná místnost v databázi neexistuje, protože pokud by ji pak uživatel chtěl pro přestěhování použít,

nastala by kolize. Na konci zpracování je uživatel informován, zda se uložení fiktivní místnosti povedlo.

PresunVyukyFiltrRAForm Třída definuje filtr, podle kterého jsou vybírány rozvrhové akce pro definovanou fázi stěhování. Pokud při zpracování formuláře v databázi odpovídající RA najde, nastaví jim zvolenou fázi stěhování. Na závěr uživatele informuje, zda se uložení povedlo a u kolika RA došlo k nastavení nové fáze, ve které se budou tyto RA stěhovat.

PresunVyukyParametryForm Třída obsahuje formulář, který nastavuje parametry pro jednotlivé fáze stěhování. V inicializační části se dívá do databáze, zda již tyto parametry byly pro danou fázi nastaveny, a pokud ano, přednastaví je pro zobrazení uživateli. Při zpracování formuláře zadané parametry ukládá s příslušnou fází stěhování do databáze. Na závěr uživatele informuje, zda byly parametry v pořádku uloženy.

Stránky aplikace

Stránky aplikace neboli Pages jsou třídy, které slouží pro zpracování a zobrazení dat. Dědí abstraktní třídu `PageJSP`, která definuje základní metody `doView()` a `doAction()`. První jmenovaná slouží ke zpracování dat v zobrazovacím režimu portletu, druhá pracuje v režimu akce.

Všechny stránky jsou definovány v konfiguračním souboru `pages.xml`. Ten obsahuje jejich název, přes který je možné na ně přistoupit a název JSP souboru, který má být po jejich zpracování zobrazen uživateli. Soubor obsahuje také seznam portletů a k nim přidělených stránek. Vstupní stránka aplikace je v uvedeném portletu nadefinována.

Následující část obsahuje třídy stránek, které se v aplikaci vyskytují.

PresunVyukyPage Třída reprezentuje úvodní stránku aplikace. V metodě `doView()` zjišťuje přes instanci implementace rozhraní `PresunVyukyManager` seznam uložených stěhování. Také umožňuje podle parametrů nastavit nově zvolené stěhování do session aplikace.

PresunVyukyVyberMistnostiPage Třída umožňuje v metodě `doAction()` uložit vybrané místnosti z nalezeného seznamu do databáze jako místnosti určené pro přesun výuky. Dále umožňuje duplikovat existující místnosti, do kterých je možné výuky přesouvat a které po této akci nebudou obsahovat žádné RA.

Pro práci s databází metoda obsahuje instanci implementace rozhraní `PresunVyukyManager`. Při ukládání vybraných nebo duplikovaných místností nejprve zjistí, zda již takové místnosti nejsou uložené v databázi. Poté uživatele informuje, zda bylo uložení úspěšné a kolik místností bylo do databáze uloženo.

PresunVyukyOdstraneniMistnostiPage Třída v metodě `doAction()` umožňuje odstranit vybranou místnost ze seznamu místností, které jsou určené pro přesun výuky. Pro tuto akci volá metodu z instance implementace rozhraní `PresunVyukyManager`. Na závěr informuje uživatele, jak se tato akce povedla.

PresunVyukyFiktivniMistnostPage Třída slouží k zobrazení výsledku po pokusu o vytvoření fiktivní místnost. V metodě `doView()` nastavuje atribut requestu podle přijatého parametru, zda se vytvoření povedlo.

PresunVyukyPripraveniRAPage Třída umožňuje v metodě `doAction()` připravit rozvrhové akce zvolených místností, ze kterých se bude výuka stěhovat, do seznamu vybraných RA. Pokud jsou všechny RA těchto místností správně připraveny, nastaví příznak, že je možné pokračovat v dalším nastavování stěhování. Na závěr uživatele informuje, zda se připravení RA povedlo.

PresunVyukyVyberFazePage Třída v metodě `doAction()` nastavuje aktuální fázi stěhování. Umožňuje založení nové fáze, nebo nastavuje fázi již existující. Tuto fázi pak ukládá do session aplikace. Po vytvoření nové fáze aktualizuje celkový počet fází v stěhování. Na závěr uživatele informuje, zda se vytvoření nové fáze povedlo.

PresunVyukyNastaveniStehovaniPage Třída v metodě `doView()` zjišťuje aktuální stav nastavování fází rozvrhových akcí. Načítá stav RA s nenastavenou fází, s nastavenou fází a s aktuálně nastavovanou fází. Tyto informace pak předává pro zobrazení uživateli.

PresunVyukySimulaceStehovaniPage Třída v metodě `doAction()` vytváří nové vlákno aplikace, které se pak postará o simulaci přesunu výuky. V metodě `doView()` pak kontroluje stav průběhu simulace, o kterém informuje uživatele, a na závěr ukončuje vlákno simulace.

PresunVyukySimulaceThread Třída dědí třídu `Thread`, která umožňuje vytvořit nové vlákno aplikace. V konstruktoru získává potřebné parametry, podle kterých bude následně řídit volání procedury pro simulaci stěhování. Překrývá metodu `run()`, ve které definuje a sleduje průběh spouštění simulace.

PresunVyukyGrafPage Třída vytváří v metodě `doView()` načítá data pro zvolený graf a vyváří jej spolu s dialogovým oknem, ve kterém bude graf zobrazován.

Pro vytvoření grafu je použita knihovna JFreeChart, která je popsaná v podkapitole 5.2. V tomto případě je použita tovární metoda `ChartFactory.createLineChart()`, která jako parametry nastavuje základní popisy os grafu a datový set v objektu třídy `CategoryDataset`.

6.4.4 View

View neboli pohled zajišťuje interakci s uživatelem. Poskytuje mu uživatelské rozhraní, přes které mu zobrazuje informace a zároveň přes něj přijímá další instrukce.

JSP

JSP (JavaServer Pages)[10] slouží k definování vzhledu stránky. Na straně serveru jsou `.jsp` soubory přeloženy a jako servlety generují dynamický obsah stránky. Většinou jsou tvořeny HTML (HyperText Markup Language) značkami, které pak následně doplňuje Java kód. Ten se může skládat z direktiv, deklarací, skriptletů a výrazů.

Dalšími možnými prvky mohou být akční elementy, které řídí generování stránky, Unified Expression Language (EL) usnadňující přístup k datovým objektům a tag libraries, které pomáhají k odstranění používání skriptletů. Standardní taglib,

[10]https://www.jcp.org/en/jsr/detail?id=245

která se používá nejčastěji, je JSTL (JSP Standard Tag Library). Tato knihovna umožňuje základní práci s daty, jako je například získání dat z requestu, použití cyklů a podmínek, či vypsání dat do stránky. Uživatel si může vytvořit i vlastní tag library, které mu následně usnadňují práci.

V rámci aplikace byly kromě standardní tabliby použity knihovny:

- `fmt` – slouží k lokalizaci dat, podle zadaného klíče najde lokalizovaný řetězec, který následně vypíše do stránky,
- `portlet` – umožňuje pracovat se základními objekty portletu,
- `val` – slouží k vytvoření formulářů, které jsou generovány knihovnou pro tvorbu a validaci formulářů z modulu IS/STAG, která je více popsaná v podkapitole 6.4.3 v části Formuláře,
- `pages` – umožňuje vytvářet záložky a odkazy na jiné stránky aplikace.

Data jsou získávána nejen z atributů requestu, ale také ze session objektu, který umožňuje uchovávat data delší dobu.

Vykreslování stránek aplikace je rozděleno do následujících `.jsp` souborů. Jednu stránku může tvořit i více vnořených souborů, které díky tomu mohou být znovu použity na více různých stránkách a nedochází tak k duplikaci kódu.

PresunVyuky.jsp Základní stránka tvoří hlavní záložky aplikace a stará se o zobrazování obsahu těchto záložek. Umožňuje mezi záložkami přepínat. Zároveň v případě potřeby může některé záložky zneplatnit.

Základními záložkami jsou:

- *Výběr stěhování* – záložka umožňuje vybrat si dříve uložené stěhování nebo založit nové,
- *Výběr místností* – záložka slouží k výběru místností, jejichž výuka se bude přesouvat, a místností, do kterých se přesouvat,
- *Nastavení stěhování* – záložka umožňuje nastavení fáze stěhování přesouvaným RA a také nastavení parametrů ovlivňujících stěhování,
- *Výsledky simulace* – záložka zobrazuje průběžný stav simulace přesunu, po přestěhování výuky zobrazí výsledné hodnoty, statistiky a grafy.

Stránka uživateli zobrazuje informace o úspěšně provedených akcích i chybové hlášky. Tyto hlášky jí ve formě zpráv nastavují aplikační stránky i formuláře aplikace.

VyberStehovani.jsp Stránka definuje obsah první záložky aplikace. Obsahuje formulář pro výběr dříve uložených stěhování. Dále poskytuje odkaz pro vytvoření nového stěhování výuky.

NoveStehovaniDialog.jsp Stránka se zobrazuje v dialogovém okně a obsahuje formulář pro založení nového stěhování.

NastaveniFaze.jsp Stránka obsahuje formulář se seznamem vytvořených fází stěhování a možností založení nové fáze. Je zobrazována na záložkách s přidáváním místností a nastavováním rozvrhových akcí.

NastaveniMistnosti.jsp Stránka generuje obsah záložky, která umožňuje vyhledávat místnosti. V prvním sloupci umožňuje vybírat místnosti, jejichž RA bude chtít uživatel přesouvat. Ve druhém sloupci naopak dovoluje vybrat místnoti, do kterých se bude výuka přesouvat.

Header.jsp Stránka obsahuje hlavičky, které označují, jaké místnosti uživatel právě vybírá.

Filtr.jsp Stránka obsahuje filtr, podle kterého jsou vyhledávány existující místnosti v databázi. Při vybírání místností, do kterých se bude výuka přesouvat navíc obsahuje odkaz na vytvoření fiktivní místnosti.

FiktivniMistnost.jsp Stránka obsahuje odkaz pro vytvoření fiktivní místnosti, který se následně zobrazuje v dialogovém okně.

FiktivniMistnostDialog.jsp Stránka se zobrazuje v dialogovém okně a obsahuje formulář, který umožňuje vytvoření nové fiktivní místnosti, do které se bude výuka stěhovat.

VyhledaneMistnosti.jsp Stránka zobrazuje tabulku vyhledaných místností, které se podle zadaného filtru nacházejí v databázi. Umožňuje je zkopírovat pro přesun výuky. V případě místností, do kterých se bude výuka stěhovat, navíc umožňuje jejich duplikování. To znamená, že se místnosti označí příponou `"x"` a budou se považovat za prázdné. Při přesunu se pak výuka nebude řadit mezi již existující RA, ale bude moci být naplánována tak, jako kdyby byla místnost aktuálně bez výuky.

SeznamMistnosti.jsp Stránka zobrazuje místnosti, které byly vybrány pro simulaci přesunu výuky. Umožňuje je z tohoto seznamu odstranit. V případě místností, ze kterých bude výuka stěhována, pak vyžaduje překopírování jejích RA, do seznamu připravených RA pro přesun výuky.

NastaveniStehovani.jsp Stránka zobrazuje obsah záložky, který dovoluje nastavovat stěhování. Informuje uživatele o aktuálním nastavení fází stěhování pro rozvrhové akce. Obsahuje stránky pro nastavení fáze k vybraným RA a nastavení parametrů, které ovlivňují stěhování. V případě, že všechny připravené RA mají nastavenou fázi, zobrazuje odkaz pro spuštění simulace stěhování.

NastaveniRA.jsp Stránka obsahuje formulář, který umožňuje vyhledat odpovídající rozvrhové akce a nastavuje jim příslušnou fázi stěhování. Umožňuje tak vybírat RA například podle typu místnosti nebo zastoupení studentů určité fakulty.

NastaveniParametruStehovani.jsp Stránka obsahuje formulář, který umožňuje nastavení parametrů stěhování pro každou fázi zvlášť. Formulář dříve uložené nastavení zobrazuje uživateli při svém načtení. Tyto parametry jsou pak použity pro příslušnou fázi během simulování stěhování výuky.

VysledkySimulace.jsp Stránka zobrazuje záložku, která informuje uživatele o průběhu jednotlivých fází simulace stěhování a na závěr informuje o výsledcích této simulace.

VyslednaData.jsp Stránka zobrazuje data po doběhnutí simulace přesunu výuky. Vypisuje základní statistiku úspěšnosti simulace, umožňuje prohlédnout grafy

se základními ukazateli využití a vytížení místností a zobrazuje, kam byly RA v simulaci přesunuty.

GrafSimulace.jsp Stránka vykresluje zvolený graf s daty statistického ukazatele, který je vytvořen knihovnou `JFreeChart`. Tato knihovna je popsaná v podkapitole 5.2. Zobrazený graf je umístěn do dialogového okna.

CSS a JS

Vzhled a chování JSP stránek je upraveno formátováním CSS styly (Cascading Style Sheets) a javascriptovými funkcemi.

Aplikace používá stejné styly, které jsou použity v informačním systému IS/STAG. Hlavním důvodem je, aby se udržel jednotný vzhled portálu, který všechny jeho aplikace zobrazuje uživatelům. V souboru `presun_vyuky_style.css` definuje rozšíření původních stylů, které dále pomáhá vytvářet vzhled aplikace.

Dále byly použity javascriptové funkce, které zajišťují práci s dialogovými okny nebo zaškrtávání více checkboxů najednou.

Lokalizace

Informační systém IS/STAG podporuje lokalizování textů v aplikaci. Přes tag library `fmt` dovoluje zadat lokalizační klíče, ke kterým je následně v `.properties` souboru definován řetězec s lokalizovaným textem. Tento způsob se používá v JSP souborech.

Druhou možností je nastavení lokalizovaného řetězce přímo v třídách aplikace. Z requestu je možné získat instanci objektu `ResourceInfo`, který obsahuje metodu `getString()`. Tato metoda bere jako parametr lokalizační klíč, podle kterého vyhledá lokalizovaný řetězec.

6.5 Logování

V rámci aplikace jsou všechny důležité úkony logovány. Jedná se zejména o ukládání informací do databáze nebo odchycené chyby.

Logování je nastaveno v souboru `logging.xml`. Je zajištěno knihovnou systému IS/STAG, která umožňuje ukládat informace v úrovních `DEBUG`, `INFO` a `ERROR`. Díky tomuto rozdělení je možné odlišit, co zalogovaný záznam znamená a rychle odhalit chyby v aplikaci.

Každý logovaný záznam obsahuje prefix *"Presun vyuky"*, aby bylo v seznamu logů celého informačního systému možné odlišit, které logy se této aplikace týkají.

7 Nasazení

Kapitola seznamuje s postupem sestavení a nasazení nové aplikace do systému IS/STAG. Popisuje umístění aplikace a uživatelské dokumentace.

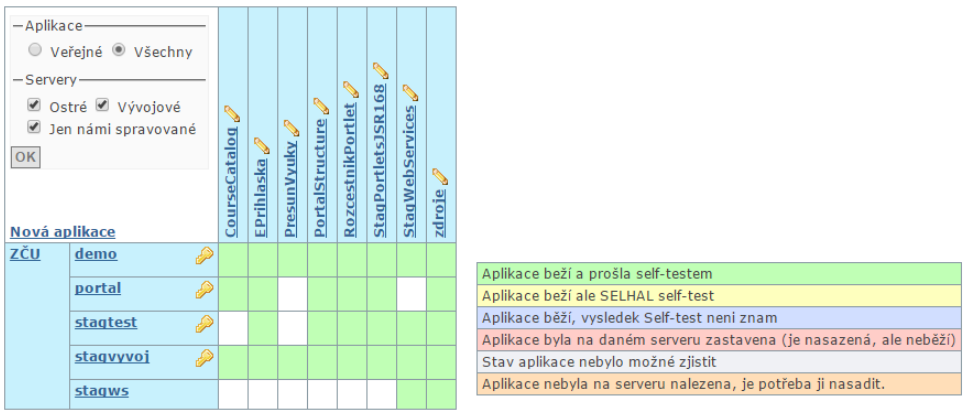
7.1 Sestavení aplikace

Pro sestavení aplikace byl použit nástroj Gradle[1], který vychází z Groovy[2]. Používá se pro automatizaci procesů. Pro sestavení aplikace byly vytvořeny buildovací skripty `build.gradle` a `settings.gradle`.

Skript `build.gradle` obsahuje návod na vytvoření `war` souboru. Obsahuje seznam modulu, na kterých je aplikace závislá. Dále uvádí adresáře zdrojových souborů, ze kterých má být aplikace sestavena.

V souboru `settings.gradle` jsou uvedeny moduly, které mají být do aplikace vloženy.

Aplikaci je možné sestavit ručně nainstalovaným nástrojem Gradle. Druhou možností je využití buildovacího nástroje, který se nachází na webu vývojářů informačního systému a je používám pro sestavování a nasazování aplikací informačního systému IS/STAG. Ukázka tohoto nástroje je zobrazena na obrázku 7.1.



Obrázek 7.1: Ukázka buildovacího nástroje s nasazenými aplikacemi

[1] https://docs.gradle.org/current/userguide/userguid.html
[2] http://groovy-lang.org/

7.2 Nasazení aplikace

Pokud byla aplikace sestavena výše zmíněným buildovacím nástroje, je již její nasazení do informačního systému IS/STAG jednoduché. Nástroj obsahuje seznam aplikací a serverů, na které je možné aplikace nasadit. Nejprve je aplikace přidána do nástroje. Poté je zadáno její sestavení a aplikace vytvoří záznam v jednom ze seznamů úspěšných nebo neúspěšných buildů. Pokud byla aplikace úspěšně sestavena, je možné její verzi nasadit na vybraný server.

Aby aplikace fungovala správně, musí server, na kterém běží databáze obsahovat všechny databázové změny, které byly pro aplikaci vytvořeny. Z tohoto důvodu je nutné do zvolené databáze vydistribuovat skript, který tyto změny v databázi provede. K tomuto účelu existuje další webová aplikace, která umožňuje do jednotlivých databází nahrávat skripty se změnami. Je přístupná vývojářům informačního systému IS/STAG. Druhou možností je spuštění skriptu přímo ve vybrané databázi po přihlášení.

7.3 Umístění aplikace

Pro účely této práce byla aplikace umístěna na demo portál, který se nachází na stránce `hhtps://stag-demo.zcu.cz/`. Tento portál slouží k demonstraci existujících aplikací informačního systému. Dovoluje uživatelům, aby se přihlásili pod libovolnou rolí a vyzkoušeli si jeho funkčnosti. Databáze, nad kterou běží, neobsahuje stejná data, jako jsou v provozní databázi, ale pouze demostrační podmnožinu, která je navíc modifikována a anonymizována.

Aplikace je přístupná uživatelským rolím *superrozvrhář*, *prorektor* a *management*. Proto je nutné se pod některou z těchto rolí do demo portálu přihlásit kliknutím na login uvedený u vybrané role. Aplikace je poté přístupná na záložce `IS/STAG` na podstránce `Přesun výuky`.

7.4 Uživatelská dokumentace

Uživatelská dokumentace aplikací informačního systému IS/STAG se nachází na webu `http://is-stag.zcu.cz`, kde se udržuje v aktualizované podobě. Proto je vhodné, aby se uživatelská dokumentace, která popisuje aplikaci vytvořenou

v rámci této práce, nacházela na stejném místě.

Dokumentace, která popisuje použití aplikací informačního systému, se vytváří ve formátu `DocBook`[3], ve kterém se značkovacími tagy definuje její formát a struktura. Změny v dokumentaci jsou pak nahrávány do verzovacího nástroje, ze kterého je každý den generovaná nová verze dokumentace. Ta je pak umístěna na webu ve formátu HTML a také je vygenerována do formátu PDF.

Dokumentace k této aplikaci je přístupná přes ikonu otazníku v portletu aplikace. Po kliknutí se zobrazí stránka s odkazem, který vede na poslední verzi dokumentace. Druhou možností, která vede na uvedenou dokumentaci, je zadání odkazu `http://is-stag.zcu.cz/napoveda/stag-v-portalu/is-stag_presun-vyuky.html` přímo do adresního řádku webového prohlížeče.

[3]https://docbook.org/whatis

8 Testování

Kapitola obsahuje popis základního uživatelského otestování aplikace. Dále seznamuje s výsledky testování simulačního algoritmu pro přesun výuky.

8.1 Uživatelské testy

Aplikace byla otestována nad reálnými daty Západočeské univerzity. Při testování byly používány místnosti i rozvrhové akce této univerzity, které byly platné v minulých letech.

8.1.1 Základní testování funkčnosti aplikace

Základní funkčnost aplikace byla otestována v nejpoužívanějších prohlížečích jako jsou Chrome, Firefox a Edge. Bylo postupováno podle uživatelské dokumentace, která je popsána v podkapitole 7.4.

Nejprve byla testována možnost založení nového stěhování, ve které byla vyzkoušena funkčnost formuláře pro jeho vytváření i na zadávání nevalidních vstupů. Dále došlo k vybrání jiného dříve vytvořeného stěhování.

Na další stránce byl testován formulář pro vyhledávání místností univerzity. Ověřovala se jeho funkčnost a seznam nalezených místností. Po vyhledání místností je možné tyto nalezené položky přidat pro stěhování. Bylo testováno přidávání po jedné i po více položkách najednou. Vybrané místnosti je možné odstranit, proto došlo k otestování i rušení výběru místnosti pro stěhování.

Dále bylo ověřeno duplikování místností, které vybere místnosti do kterých se bude výuka stěhovat a algoritmus stěhování je následně považuje za prázdné. Otestováno bylo i vytváření fiktivních místností. Do formuláře byly zadávány nevalidní vstupy a bylo sledováno chování aplikace. Pro nastavování místností bylo vyzkoušeno i nastavování jiné než první fáze stěhování.

Poté bylo otestováno připravení RA pro simulaci stěhování. Ve formuláři pro nastavování fáze stěhování k vybraným RA bylo vyzkoušeno zadávání různých pa-

rametrů, které následně vybíraly, jakým RA bude fáze přidělena. Dále došlo k otestování formuláře pro nastavování parametrů stěhování výuky, které jsou použity v algoritmu simulace stěhování.

Poté byl nad zvolenými daty spuštěn simulační algoritmus, který provedl přesun výuky do zvolených místností podle nastavených fází stěhování. Na závěr byla otestována správnost zobrazených statistických ukazatelů a grafů.

8.1.2 Výsledky testování

Všechny provedené testy proběhly s očekávaným výsledkem bez chyb způsobujících pád aplikace. Proto byla těmito testy prokázána základní funkčnost aplikace.

8.2 Test simulačního algoritmu

Vytvořený algoritmus byl otestován nad reálnými daty Západočeské univerzity. Byla ověřena jeho funkčnost při simulaci přesouvání výuky.

8.2.1 Testovací případy

Testování bylo provedeno na třech vzorových případech, které mohou v reálné situaci nastat. Jedná se o následující případy.

Testovací případ 1

- převod do prázdných místností
- otestováno nad místnostmi budovy UC

První testovací případ se věnoval simulaci přesunu výuky do nových nebo úplně prázdných místností. Tato situace nastává, pokud univerzita výuku stěhuje do již vystěhovaných místností nebo do nové výstavby.

V tomto případě se jedná o simulaci přesunu výuky v budově UC. Ve čtyřech fázích stěhování jsou postupně přesouvány RA do nových místností v jednotlivých patrech budovy.

Testovací případ 2

- převod do místností s již existující výukou
- testováno nad místnostmi budovy ST a EU

Druhý testovací případ simuloval stěhování do místností s existující výukou. Jedná se o situaci, kdy je například nutné přesunout rozvrhové akce z jiné lokality do univerzitního kampusu, ve kterém se nenacházejí volné místnosti.

Konkrétně se jedná o přesun rozvrhových akcí z budovy ST do EU + zachování data a času nepřesunuté akce, nekompatibilita místností (někdy vyžaduje moc velkou kapacitu, která ale neexistuje)

Testovací případ 3

- převod velkého množství rozvrhových akcí
- testováno nad laboratořemi a učebnami celé univerzity

Třetí testovací případ se zabývá stěhováním velkého množství rozvrhových akcí. Pokud se univerzita rozhodla, všechnu výuku přestěhovat do univerzitního kampusu, nastala by takováto situace.

Jedná se o stěhování místností typu laboratoř a učebna, které se ze všech budov univerzity přesouvají do duplikovaných prázdných místností budov UU a UL. Vše bylo převedeno v rámci jedné fáze stěhování bez nastavení preferencí místností určitým skupinám RA.

8.2.2 Výsledky testování

Testovací případy byly postupně spuštěny v simulační aplikaci. Simulační algoritmus pro všechny z nich doběhl do konce a následně zobrazil statistické údaje, které jsou uvedené v tabulce 8.1. Z těchto dat vyplývá, že algoritmus dokáže přesunout téměř veškerou výuku.

Například u druhého testovacího případu se nepřesunula výuka ze třech místností. Při procházení dat je ale zjevné, že se všechny místnosti přesunout nemohly, jelikož

Testovací případ	1	2	3
Doba běhu simulace (v s)	11	78	2 056
Počet stěhovaných RA	414	132	5 136
Počet neúspěšně převedených RA	2	3	88
Procento úspěšně převedených RA (v %)	99	97	98
Procento úspěšně převedených RA na			
stejný čas a stejný den (v %)	100	95	75
stejný čas a jiný den (v %)	0	3	19
jiný čas a stejný (v %)	0	0	0
jiný čas a jiný den (v %)	0	2	6

Tabulka 8.1: Výsledky simulačního algoritmu

v budově, ze které se výuka stěhuje, jsou místnosti s větší kapacitou, než která se nachází v druhé budově.

V případě přesouvání výuky do prázdných místností, algoritmus neměl problém zachovat jak čas tak i den v týdnu všech stěhovaných RA. Pokud však byla přesouvána výuka z velkého množství místností, byla úspěšnost zachování těchto údajů jen 75%, avšak při takovém množství rozvrhových akcí taková situace pochopitelná.

8.3 Závěr testování

Otestování aplikace i simulačního algoritmu prokázalo, že je vytvořená aplikace funkční.

9 Závěr

V rámci této práce byla představena základní problematika přesunu výuky a rozvrhování univerzitní výuky. Byly prozkoumány algoritmy, které umožňují optimalizovat vytváření nových rozvrhů. Bylo zjištěno, že tyto algoritmy jsou pro simulaci přesunu výuky nevhodné, a proto byl navržen nový algoritmus přesunu.

Dále byly prozkoumány technologie, které jsou vhodné pro zobrazování rozvrhů a grafů v portálových Java aplikacích.

Na základě těchto poznatků byla vytvořena aplikace, která simuluje přesun výuky vybraných rozvrhových akcí. Dovoluje mu vybrat, které rozvrhové akce se budou stěhovat a nastavit jednotlivé fáze stěhování. Po simulaci uživatele informuje o výsledcích této simulace a zobrazuje mu statistické výstupy.

Aplikace byla nasazena na portál informačního systému IS/STAG. Jako jeho součást bude dále vyvíjena a vylepšována. V rámci tohoto systému byla otestována nad reálnými daty univerzity a došlo k ověření funkčnosti simulačního algoritmu.

Seznam použitých zkratek

0/1-LP	0/1-lineární programování (0/1-linear programming)
BFGL	Big Faceless Graph Library
CSS	Cascading Style Sheets
DAO	Data Access Object
ECMA	European Computer Manufacturers Association
EL	Unified Expression Language
HTML	HyperText Markup Language
ILP	Celočíselné lineární programování (Integer Linear Programming)
IoC	Inversion of Control
IS/STAG	Informační systém studijní agendy
JDBC	Java Database Connectivity
JPEG	Joint Photographic Experts Group
JSP	JavaServer Pages
JSR-268	Java Portlet Specification 2.0
JSTL	JSP Standard Tag Library
LP	Lineární programování (Linear Programming)
MVC	Model-View-Controller
NP	Nedeterministický polynomiální čas (Nondeterministic polynomial time)
P	Polynomiální čas (Polynomial time)
PDF	Portable Document File
PNG	Portable Network Graphics
RA	Rozvrhová akce
SA	Simulované žíhání (Simulated Annealing)
SQL	Structured Query Language
SVG	Scalable Vector Graphics
XML	eXtensible Markup Language

Literatura

[1] Abdullah, S. – Malik, W. A. *Heuristic Approaches for University Timetabling Problems*. PhD thesis, The University of Nottingham, Červen 2006.

[2] BFGL. *Big Faceless Graph Library* [online]. Dostupné z: `https://bfo.com/products/graph/`.

[3] Blum, C. et al. A GA evolving instructions for a timetable builder. 10 2002.

[4] Burke, E. – Kendall, G. – Soubeiga, E. A Tabu-Search Hyperheuristic for Timetabling and Rostering. *Journal of Heuristics*. Dec 2003, 9, 6, s. 451–470. ISSN 1572-9397. doi: 10.1023/B:HEUR.0000012446.94732.b6. Dostupné z: `https://doi.org/10.1023/B:HEUR.0000012446.94732.b6`.

[5] Das, D. – Natarajan, S. A Structured Timing Itinerary Using an Augmented Swarm Intelligent Algorithm. In *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, s. 1221–1228, Dec 2015. doi: 10.1109/CICN.2015.235.

[6] de Werra, D. An introduction to timetabling. *European Journal of Operational Research*. 1985, 19, 2, s. 151 – 162. ISSN 0377-2217. doi: 10.1016/0377-2217(85)90167-5. Dostupné z: `https://doi.org/10.1016/0377-2217(85)90167-5`.

[7] Dinkel, J. J. – Mote, J. – Venkataramanan, M. A. An Efficient Decision Support Systems for Academic Course Scheduling. *Oper. Res.* October 1989, 37, 6, s. 853–864. ISSN 0030-364X. doi: 10.1287/opre.37.6.853. Dostupné z: `http://dx.doi.org/10.1287/opre.37.6.853`.

[8] Dorigo, M. – Maniezzo, V. – Colorni, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*. Feb 1996, 26, 1, s. 29–41. ISSN 1083-4419. doi: 10.1109/3477.484436.

[9] Elmohamed, M. A. S. – Coddington, P. D. – Fox, G. A Comparison of Annealing Techniques for Academic Course Scheduling. In *Selected Papers from the Second International Conference on Practice and Theory of Automated Timetabling II*, PATAT '97, s. 92–114, London, UK, UK, 1998. Springer-Verlag. doi: 10.1007/BFb0055883. Dostupné z: `http://dl.acm.org/citation.cfm?id=646430.692895`. ISBN 3-540-64979-4.

[10] Erben, W. – Keppler, J. A genetic algorithm solving a weekly course-timetabling problem. In Burke, E. – Ross, P. (Ed.) *Practice and Theory of Automated Timetabling*, s. 198–211, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. ISBN 978-3-540-70682-3.

[11] Ferland, J. A. – Roy, S. Timetabling problem for university as assignment of activities to resources. *Computers & Operations Research*. 1985, 12, 2, s. 207 – 218. ISSN 0305-0548. doi: https://doi.org/10.1016/0305-0548(85)90045-0. Dostupné z: http://www.sciencedirect.com/science/article/pii/0305054885900450.

[12] G. *G 2D Graphics Library and Rendering Engine For Java* [online]. Dostupné z: http://geosoft.no/graphics/.

[13] Goldberg, A. V. – Tardos, É. – Tarjan, R. Network flow algorithm. Technical report, Cornell University Operations Research and Industrial Engineering, 1989.

[14] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1st edition, 1989. ISBN 0201157675.

[15] Hertz, A. Tabu search for large scale timetabling problems. *European Journal of Operational Research*. 1991, 54, 1, s. 39 – 47. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(91)90321-L. Dostupné z: http://www.sciencedirect.com/science/article/pii/037722179190321L.

[16] Hertz, A. Finding a Feasible Course Schedule Using Tabu Search. *Discrete Appl. Math.* March 1992, 35, 3, s. 255–270. ISSN 0166-218X. doi: 10.1016/0166-218X(92)90248-9. Dostupné z: http://dx.doi.org/10.1016/0166-218X(92)90248-9.

[17] Hromkovic, J. *Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics*. Springer-Verlag, 2004. ISBN 3540441344.

[18] JFreeChart. *JFreeChart* [online]. Dostupné z: http://www.jfree.org/jfreechart/.

[19] Junaedi, D. – Maulidevi, N. U. Solving Curriculum-Based Course Timetabling Problem with Artificial Bee Colony Algorithm. In *2011 First International Conference on Informatics and Computational Intelligence*, s. 112–117, Dec 2011. doi: 10.1109/ICI.2011.28.

[20] Karaboga, D. An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report - TR06. 01 2005. doi: 10.1.1.714.4934.

[21] Kostuch, P. The University Course Timetabling Problem with a Three-Phase Approach. In Burke, E. – Trick, M. (Ed.) *Practice and Theory of Automated Timetabling V*, s. 109–125, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-32421-8.

[22] Mulvey, J. M. A classroom/time assignment model. *European Journal of Operational Research*. 1982, 9, 1, s. 64 – 70. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(82)90012-1. Dostupné z: http://www.sciencedirect.com/science/article/pii/0377221782900121.

[23] Pandey, J. – Sharma, A. K. Survey on University timetabling problem. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, s. 160–164, Březen 2016.

[24] Rossi-Doria, O. et al. A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem. In Burke, E. – De Causmaecker, P. (Ed.) *Practice and Theory of Automated Timetabling IV*, s. 329–351, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-45157-0.

[25] Schaerf, A. A Survey of Automated Timetabling. *Artificial Intelligence Review*. Duben 1999, 13, 2, s. 87–127. ISSN 1573-7462. doi: 10.1023/A:1006576209967. Dostupné z: https://doi.org/10.1023/A:1006576209967.

[26] Selim, S. M. Split Vertices in Vertex Colouring and Their Application in Developing a Solution to the Faculty Timetable Problem. *Comput. J.* February 1988, 31, 1, s. 76–82. ISSN 0010-4620. doi: 10.1093/comjnl/31.1.76. Dostupné z: http://dx.doi.org/10.1093/comjnl/31.1.76.

[27] Socha, K. – Knowles, J. – Sampels, M. A MAX-MIN Ant System for the University Course Timetabling Problem. In Dorigo, M. – Di Caro, G. – Sampels, M. (Ed.) *Ant Algorithms*, s. 1–13, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45724-4.

[28] Tripathy, A. School Timetabling-A Case in Large Binary Integer Linear Programming. *Manage. Sci.* December 1984, 30, 12, s. 1473–1489. ISSN 0025-1909. doi: 10.1287/mnsc.30.12.1473. Dostupné z: http://dx.doi.org/10.1287/mnsc.30.12.1473.

[29] Vazirani, V. V. *Approximation Algorithms*. Springer Publishing Company, Incorporated, 2003. ISBN 3540653678.

Seznam obrázků

Seznam tabulek

A ERA diagram