FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Výuková aplikace pro metodu PERT
Student:	Bc. Tomáš Sýkora
Vedoucí:	Ing. Petra Pavlíčková, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Výuková aplikace pro matematické modelování na základě metody PERT a její simulace. Cílem této aplikace je vylepšení výuky matematického modelování a to přímo metody PERT, se kterou studenti mají problémy. Dalším cílem je nástroj, který umožní zlepšení časových odhadů v rámci projektů a úspěšném dokončení projektů.

1. Určete procesy, které bude aplikace podporovat.
2. Namapujte jednotlivé procesy na funkční požadavky aplikace.
3. Navrhněte grafické uživatelské rozhraní aplikace.
4. Určete vhodné technologie, nad kterými bude vytvořena aplikace.
5. Připravte scénáře pro otestování základních průchodů aplikací.
6. Navrhněte relační model databáze.
7. Proveďte implementaci aplikace.
8. Otestujte aplikaci dle testovacích scénářů a opravte případné chyby.
9. Vytvořte uživatelskou příručku.
10. Vytvořte instalační příručku.
11. Proveďte ekonomické zhodnocení z hlediska nákladů i přínosů aplikace.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 20. prosince 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Výuková aplikace pro metodu PERT

Bc. Tomáš Sýkora

Katedra softwarového inženýrství
Vedoucí práce: Ing. Petra Pavlíčková, Ph.D.

6. května 2018

Poděkování

Děkuji vedoucí své práce za konzultace k funkcionalitě aplikace a hodnotné rady k obsahu práce. Také chci poděkovat všem, kteří se zúčastnili uživatelského testování, diskuzí o uživatelském rozhraní systému nebo revize textu. V neposlední řadě chci poděkovat své rodině, která mě během celého studia podporovala.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 6. května 2018 .....

Odkaz na tuto práci

Sýkora, Tomáš. *Výuková aplikace pro metodu PERT*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato diplomová práce se zabývá tvorbou webové aplikace pro výuku metody PERT a její simulace. Na začátku práce jsou nejdříve vysvětleny potřebné teoretické znalosti pro pochopení této metody. Následně uvádím návrh uživatelského rozhraní aplikace, analýzu a architekturu systému. Druhá polovina práce začíná implementací informačního systému. Poté práce obsahuje popis testování aplikace a shrnutí vytvářené dokumentace. V závěru práce ještě popisuji možná vylepšení systému, kterými je na vytvořený systém v budoucnu možné navázat.

Klíčová slova Projektové řízení, PERT, PERT simulace, odhad pracnosti, Java, Spring, MySQL

Abstract

This master thesis focuses on analysis, design and implementation of a web application for PERT and PERT simulation. In the beginning, the necessary knowledge for understanding PERT method is explained. After the theoretical part of the thesis, the focus shifts towards user interface draft, analysis and architecture of the system. In the second half of the thesis, the implementation of the information system is elaborated, followed by the testing description and a short summary of the documentation. The conclusion of the thesis points out the possible future improvements.

Keywords Project management, PERT, PERT simulation, estimated time, Java, Spring, MySQL

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Od určitého věku začneme každý řešit, jak si rozvrhnout svůj čas, abychom dokázali stihnout vše, co potřebujeme. Také se snažíme plánovat aktivity tak, abychom stihli i zábavu. Přesto, že o uspořádání existuje spousta studií, je člověk vynalézavý a s plánováním svého času si nějakým svým způsobem poradí.

Začneme-li plánovat větší celek, stává se plánování velmi zajímavou a problematickou činností. Zde je již vhodné využít vypracované techniky a postupy pro správu zdrojů. Navíc se jedná o problém, který lidstvo řeší již od svého počátku. Změnily se pouze plánované činnosti. Ve starověkém Egyptě se plánovala stavba pyramid. V současné době plánujeme spíše složité informační systémy. Ačkoliv se jedná o naprosto rozdílné aktivity, v samotném plánování jednotlivých činností narážíme na stejné problémy.

Této zajímavé problematice se věnuje projektové řízení. Tedy proces, který řeší alokaci zdrojů a koordinaci jednotlivých činností. Cílem projektového řízení je efektivně řídit práci lidí, kontrolovat možná rizika a zařídit dokončení projektu v daném termínu. Pro podporu tohoto procesu byla vynalezena spousta metod. Jednou z nich je i metoda PERT, které se věnuje tato práce.

Motivace

Studuji obor informační systémy a management. Během studia jsem absolvoval teoretické předměty a nyní jsem si mohl ověřit získané znalosti na praktickém příkladu. Také jsem měl poměrně volnou ruku ve výběru technologií a mohl jsem si pro implementační část práce zvolit technologie, které jsem považoval za zajímavé.

Zároveň věřím, že mnou navržená „Aplikace PERT“ dokáže plně nahradit současné materiály sloužící k výuce předmětu Metody a projektové řízení na České zemědělské univerzitě. Systém navíc nabídne výrazně jednodušší správu materiálů a bude sloužit jako hlavní komponenta ve výuce předmětu.

Myslím si, že na systém lze jednoduše navázat a implementovat další funkcionalitu. Poté má „Aplikace PERT“ ambice stát se nejen hlavní komponentou pro výuku uvedeného předmětu, ale zároveň sloužit také ve výuce jiných předmětů zabývajících se touto problematikou.

Struktura práce

Tuto práci je možné rozdělit do čtyř větších částí:

1. Teoretické podklady pro analýzu a implementaci systému,
2. upřesnění cílů práce,
3. praktická část práce,
4. závěr.

Teorie (2) obsahuje v první části podklady nutné pro pochopení metody PERT a její simulace. V druhé části se nachází popis technologií uvažovaných pro implementaci systému.

V kapitole 3 se nachází upřesnění cílů této práce a stručné shrnutí jednotlivých kapitol.

Třetí část práce je praktická a obsahuje několik kapitol. Kapitoly se mapují na standardní činnosti softwarového inženýrství a jsou podrobně popsány v kapitole 3. Praktická část ještě obsahuje možná vylepšení systému (10).

Kapitola 1

Cíle práce

Cílem této práce je vytvořit informační systém, jehož cílem je sloužit jako podpůrný nástroj pro výuku předmětu Metody a projektové řízení na České zemědělské univerzitě. Cíle vedoucí k vytvoření informačního systému je možné shrnout do následujících celků:

1. Seznámit se s technikou PERT a její simulací.
2. Provést analýzu procesů a určit požadavky kladené na systém.
3. Navrhnout GUI aplikace.
4. Zvolit vhodné technologie
5. Provést implementaci.
6. Moderovat testování systému.
7. Vytvořit dokumentaci systému.

Podrobný popis jednotlivých celků se nachází v kapitole 3.

Kapitola 2

Teorie

Pro analýzu a návrh systému je nutné mít teoretické znalosti z daných domén. Ty jsou popsány v této kapitole. Ta se dá rozdělit do tří větších částí. Nejdříve jsou v kapitole shrnuty podklady z oblasti matematiky. Zde je nutné seznámit čtenáře se základy pravděpodobnosti, statistiky a teorie grafů.

Dále se dostanu k hlavní doméně práce, metodám síťové analýzy projektového řízení. Nejdříve zde shrnu stručné informace o projektovém řízení. Poté jsou zde rozebrány metody pro řízení doby trvání projektů, metoda kritické cesty (dále CPM) a PERT (Program Evaluation and Review Technique). Obsahem této kapitoly je také jejich srovnání.

V závěrečné části se již dostávám k technologiím. Popisuji zde technologie, které jsem zvažoval použít pro implementaci webové aplikace. Uvádím zde pouze studie těchto technologií. Kapitola neobsahuje jejich srovnání. To popisuji až v praktické části práce v kapitole Implementace (6).

Dovolím si upozornit, že u některých částí této kapitoly nepopisuji dané disciplíny nebo domény do detailů. K tomuto důvodu jsem se rozhodl proto, že některé oblasti jsou příliš rozsáhlé a pro implementaci „Aplikace PERT“ není nutná detailní znalost problematiky.

2.1 Pravděpodobnost

V rámci této kapitoly popisuji základy pravděpodobnosti[1]. V kapitole se nesnažím popsat do hloubky tuto disciplínu, protože v dále uvedených metodách není potřeba. V systému se totiž pracuje hlavně se statistikou. Pro pochopení rozdílu mezi pravděpodobností a statistikou však tuto část matematiky také uvádím.

Jedná se o odvětví matematiky zabývající se činnostmi, u kterých nedokážeme určit, jak přesně dopadnou. Pro přesnější popis si vypomůžu praktickým příkladem. Předpokládejme, že házím kostkou. Vím, že mi může spadnout hodnota od 1 do 6. Množina výsledků je v této matematické disciplíně nazývána

prostorem elementárních jevů (Ω). Tato množina pokrývá všechny výsledky a rozlišuje je na základě podstatných detailů.

2.1.1 Náhodný jev

Náhodným jevem je míněna podmnožina prostoru elementárních jevů. Pro tyto podmnožiny je poté možné vypočítat pravděpodobnost, že daný náhodný jev nastane.

Vrátím se zpátky k uvedenému hodu kostkou. Za náhodný jev zde mohu považovat, že hozená hodnota bude větší než 5. Pravděpodobnost je poté počítána jako poměr mezi velikostí množiny náhodného jevu a všech elementárních jevů. Množina náhodného jevu obsahuje 2 hodnoty (5,6) a velikost prostoru elementárních jevů je 6. Pravděpodobnost je tedy $\frac{2}{6}$.

Jelikož se jedná o množiny, je možné s náhodnými jevy provádět všechny množinové operace. V seznamu uvádím několik příkladů.

- Náhodný jev $A = \{1\}$
- Náhodný jev $B = \{5, 6\}$
- Náhodný jev $C = \{1, 2, 3, 4, 5\}$
- Náhodný jev $D = \{3, 4, 5, 6\}$
- Nemožný jev $\emptyset$
- Sjednocení $A \bigcup B = \{1, 5, 6\}$
- Průnik $C \bigcap D = \{3, 4, 5\}$
- Doplněk $B^C = \{1, 2, 3, 4\}$
- Rozdíl $D \setminus B = \{3, 4\}$

2.1.2 σ-algebra

Jedná se o systém podmnožin prostoru Ω, který splňuje následující tři vlastnosti:

(i) $\emptyset \in \mathcal{F}$

(ii) Když $A \in \mathcal{F}$, tak $A^C \in \mathcal{F}$

(iii) Když $A_1, A_2, \ldots \in \mathcal{F}$, tak $\bigcup_{i=1}^{\inf} A_i \in \mathcal{F}$

2.1.3 Pravděpodobností míra

Pravděpodobnostní míra na $(\Omega, \mathcal{F})$ je funkce $P : \mathcal{F} \to \mathbb{R}$. Tato funkce bývá často nazývána pravděpodobnost a musí splňovat následující tři podmínky:

(i) $\forall A \ \in \mathcal{F}$ platí $P(A) > 0$

(ii) $P(\Omega) = 1$

(iii) Když jsou A_1, A_2 vzájemně disjunktní jevy $(A_i \bigcap A_j = \emptyset, \forall i, i \neq j)$ tak platí:

$$P(\bigcup_{i=1}^{\inf} A_i) = \sum_{i=1}^{\inf} P(A_i)$$

Trojice $(\Omega, \mathcal{F}, P)$ představuje pravděpodobnostní prostor.

2.2 Statistika

V první kapitole teorie jsem shrnul základy pravděpodobnosti. Tedy predikce jak dopadne sledovaný náhodný děj. Nyní se již dostávám ke statistice[2]. Ta proti pravděpodobnosti nepredikuje, ale pracuje s již naměřenými hodnotami. Z těchto hodnot jsou poté vypočteny vlastnosti náhodné veličiny.

Náhodná veličina se pohybuje na hraně pravděpodobnosti a statistiky, spíše patří ještě k pravděpodobnosti. Ale vzhledem k tomu, že ji v této práci využívám zejména u statistiky, uvádím ji až zde. V této části teorie se zaměřuji pouze na informace, které potřebuji pro výpočet PERT metody a její simulace.

2.2.1 Náhodná veličina

Náhodná veličina X na pravděpodobnostním prostoru $(\Omega, \mathcal{F}, P)$ je funkce, která přiřadí každému výsledku $\omega \in \Omega$ hodnotu $\mathrm{X}(\omega) \in \mathbb{R}$ a platí pro ni podmínka:

$$\{X \leq x\} = \{\omega \in \Omega : X(\omega) \leq x\} \in \mathcal{F}, \forall x \in \mathbb{R}$$

Tato podmínka se téměř v praxi nepoužívá a zjednodušeně říká, že $\{X \leq x\}$ je náhodný jev.

Definice náhodné veličiny je poměrně složitá, ale lze shrnout do následující věty. Náhodná veličina je funkce, která přiřadí každému výsledku experimentu ω reálnou hodnotu.

2.2.1.1 Doba trvání náhodné veličiny

U náhodné veličiny X se střední hodnotou μ a rozptylem σ^2 můžeme vypočítat s jakou pravděpodobností bude vstupní hodnota x nižší (2.1) nebo vyšší (2.2)

než hodnota získaná z náhodné veličiny. Pravděpodobnostní míru je možné vypočítat podle následujícího vzorce:

$$P(X < x) = F\left(\frac{x-\mu}{\sigma}\right) = \phi(u) \tag{2.1}$$

$$P(X > x) = 1 - \phi(u) \tag{2.2}$$

$\phi(u)$ představuje distribuční funkci normované náhodné veličiny s normálním rozdělením. Výhoda této distribuční funkce spočívá v tom, že její hodnoty jsou tabelované a lze je dohledat.

2.2.2 Výběrový průměr

Místo střední hodnoty je u náhodné veličiny možné pracovat s bodovým odhadem střední hodnoty získaných dat. Ta se vypočítá podle následujícího vzorce:

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

2.2.3 Výběrový rozptyl a směrodatná rozptylka

Odhad rozptylu na základě získaných dat se nazývá výběrový rozptyl a vypočítá se následovně:

$$s_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}_n\right)^2$$

Odhad směrodatné rozptylky se počítá podle vzorce:

$$s_n = \sqrt{s_n^2}$$

2.3 Teorie Grafů

Teorie grafů[3] představuje samostatný obor matematiky. Obor spadající do oblasti diskrétní matematiky. Za zakladatele oboru se považuje švýcarský matematik Leonhard Paul Euler. Známé je zejména jím publikované řešení problému nazvaného Sedm mostů města Königsbergu[4].

Jednalo se o to, zda je možné ve městě přejít každý z jeho sedmi mostů právě jednou a vrátit se do původního místa. Švýcarský matematik si tento problém převedl na graf, kdy břehy představovaly vrcholy a mosty hrany grafu. Pan Euler poté dokázal, že to možné není. Nyní o takovém grafu uvedeme, že není Eulerovský.

V úvodním odstavci naznačuji, čím se teorie grafů zabývá. Jedná se o poměrně velkou část matematiky. Pro pochopení PERT metody je však nutné znát pouze část teorie grafů. Tu popisuji v následujících odstavcích textu, jedná se o základní termíny: uzel, hrana a graf. Na těchto termínech poté definuji orientovaný a síťový graf.

2.3.1 Vrchol, hrana a graf

V grafu se snažíme znázornit propojení objektů. Tyto objekty zde nazýváme vrcholy. Množina vrcholů se obvykle značí V(vertices). Hranu si lze představit jako propojení dvou uzlů. Množina hran se obvykle značí E(edges). Přesná matematická definice neorientované hrany je :

$$E \subseteq \{\{u, v\} | u, v \in V, u \ \neq v \ \}$$

O hraně (u,v) můžeme říct, že se jedná o spojení mezi uzlem u a v. Neorientovaný graf G je poté definován následovně: $G = (V, E)$.

2.3.2 Orientovaný a síťový graf

Orientovaný graf je definován stejně jako graf neorientovaný. Liší se pouze definice množiny hran, zde již nepovažujeme hrany (u,v) a (v,u) za shodné. Množina hran je u orientovaného grafu definována následovně:

$$E \subseteq \{\{u, v\} | u, v \in V \ \times V \ \}$$

O hraně (u,v) můžeme poté říct, že vede z vrcholu u do vrcholu v.

Termíny uvedené v této kapitole již stačí k definici síťového grafu. Ten představuje orientovaný graf, na který jsou kladeny další podmínky:

1. Graf má právě jeden počáteční a koncový vrchol.
2. Každý vrchol, kromě počátečního, má příchozí hranu.
3. Každý vrchol, kromě koncového, má odchozí hranu.
4. Libovolné dva vrcholy nesmí spojovat více než jedna hrana.
5. K pojmenovávání vrcholů se používají přirozená čísla ($\mathbb{N}_0$). Hrana nesmí vést do uzlu s nižším číslem než má uzel zdrojový.
6. Graf neobsahuje cykly.

Všechny metody projektového řízení uvažované v této práci jsou navrženy pro síťové grafy typu AoN (Activity on Arrow). Tento typ grafu obsahuje informace o činnostech na hranách grafu. Vrcholy v grafu poté představují milníky v projektu.

2.4 Projektové řízení

Za jednoho z prvních průkopníků v oblasti projektového řízení[5] a také jeho zakladatele je považován Henry Laurence Gantt[6]. Pan Gantt tomuto odvětví výrazně přispěl vynalezením diagramu pro zachycení harmonogramu jednotlivých činností. Ganttův diagram znamenal posun od řízení lidí k řízení činností.

V současné době představuje projektové řízení klíčovou činnost pro úspěšné dokončení projektu a Ganttův digram je jedním z jeho základních nástrojů. Ale i když máte k dispozici špičkové analytiky a nejlepší programátory, bez kvalitního projektové řízení není možné efektivně organizovat tyto zdroje a přiřazovat je k daným činnostem.

Jak jsem již uvedl v předchozím odstavci, hlavním cílem projektového řízení je dokončení projektu v určeném termínu. Projekt představuje sled jedinečných aktivit, které mají určitý cíl. Ke splnění těchto aktivit máme přesně určené časové období a přidělené zdroje. Pro řízení těchto aktivit již existuje více nástrojů než pouhý Ganttův diagram. Mezi ně patří také metody PERT a CPM. Ty popisuji v následujících odstavcích této kapitoly.

Kromě zdrojů uvedených v jednotlivých kapitolách jsem čerpal informace z [7].

2.4.1 Metoda CPM

Metoda kritické cesty (critical path method)[8] je metoda síťové analýzy, která byla poprvé představena v roce 1959. Metoda se často využívá při řízení projektů a je založena na matematickém algoritmu, který popisuji v této kapitole.

Pro aplikaci CPM metody je nutné mít zpracovaný projekt ve formě síťového grafu. Vrcholy tohoto grafu představují milníky projektu. Hrany grafu jsou považovány za činnosti v projektu a obsahují dobu trvání jednotlivých činností.

Na graf splňující podmínky popsané v předchozím odstavci je poté aplikován algoritmus. Pro každý uzel existují v algoritmu následující hodnoty:

T_i^0 - Hodnota představuje nejdříve možnou dobu, kdy daná činnost může začít.

T_i^1 - Hodnota představuje nejdéle možnou dobu, kdy daná činnost musí začít, aby nedošlo ke zdržení projektu.

R_i - Interferenční rezerva vrcholu.

V první fázi algoritmu se začne od počátečního vrcholu grafu (2.3) a postupně se prochází všechny následující vrcholy (2.4). Těm je vypočtena hodnota nejdříve možného začátku činnosti (T_i^0). Algoritmus postupuje podle následujících vzorců (j je označení následujícího vrcholu po vrcholu i):

$$T_i^0 = 0 \tag{2.3}$$

$$T_j^0 = max(T_i^0 + t_{ij}) \tag{2.4}$$

Druhá fáze algoritmu začíná od koncového vrcholu grafu (2.5). V průběhu této fáze se postupně vracím grafem v obráceném směru hran k počátečnímu vrcholu grafu (2.6). V druhé fázi algoritmu je vypočtena hodnota nejdéle možné

doby začátku jednotlivých činností (T_i^1) podle následujících vzorců (i je označení předchůdce vrcholu j):

$$T_n^1 = T_n^0 \tag{2.5}$$

$$T_i^1 = min(T_j^1 - t_{ij}) \tag{2.6}$$

Po provedení obou fází algoritmu mohu vypočítat interferenční rezervu uzlu $R_i = T_i^1 - T_i^0$. Na základě této hodnoty mohu určit kritickou cestu projektu. Ta vede přes vrcholy grafu, které mají nulovou interferenční rezervu.

2.4.2 Metoda PERT

Metoda PERT[9][10] byla poprvé popsána námořnictvem Spojených států amerických v roce 1958[11]. Námořnictvo používalo metodu PERT pro hodnocení vlastních projektů.

V této práci jsem již uváděl metodu CPM, ta pracuje s přesnou hodnotou doby trvání. To však v praxi není vždy žádoucí nebo možné. Metoda PERT odstraňuje přesné hodnoty a přidává prvek náhody. Tato stochastická metoda obsahuje jako vstupní data místo jedné doby trvání hodnoty tři. Vstupní data jsou poté převedeny na střední hodnotu a rozptyl.

Vstupní data jsou standardně definována následovně:

a_{ij} - Odhad optimistické doby trvání činnosti mezi vrcholy i a j.

b_{ij} - Odhad pesimistické doby trvání činnosti mezi vrcholy i a j.

m_{ij} - Odhad očekávané doby trvání činnosti mezi vrcholy i a j.

Pro převedení vstupních dat činností na střední hodnotu (2.7) a rozptyl (2.8) je nutné znát vzorce pro transformaci dat na tyto hodnoty. Po aplikování vzorců získávám z třech vstupních parametrů dva, střední hodnotu a rozptyl. S těmito hodnoty poté pracuje lehce modifikovaná metoda CPM.

$$\mu(t_{ij}) = \frac{a_{ij} + 4m_{ij} + b_{ij}}{6} \tag{2.7}$$

$$\sigma^2(t_{ij}) = \left(\frac{b_{ij} - a_{ij}}{6}\right)^2 \tag{2.8}$$

Modifikace metody CPM spočívá v rozšíření výpočtu o rozptyly jednotlivých činností. Metoda CPM je provedena standardně, jak bylo popsáno v kapitole (2.4.1). Vstupní data představují střední hodnoty činností, ale také je nutné vypočítat rozptyl jednotlivých milníků (vrcholů grafu). Rozptyl pro danou cestu grafu je vypočítán jako součet rozptylu jednotlivých činností (hran), přes které tato cesta vede.

Po aplikování metody CPM mám v koncovém uzlu grafu vypočtenou očekávanou dobu trvání (střední hodnota). To je výsledek metody CPM. Poté si

dopočítám rozptyl kritické cesty (ta je určena stejně jako u metody CPM), tj. sečtu rozptyl všech činností, které kritická cesta obsahuje.

Ze získaného rozptylu je poté nutné vypočítat směrodatnou odchylku. Ta se vypočítá podle vzorce 2.9.

$$\sigma(t_{ij}) = \sqrt{\sigma^2(t_{ij})} \tag{2.9}$$

Po aplikování vzorce mohu pracovat s dobou trvání jako náhodnou veličinou, která je určena střední hodnotou a směrodatnou odchylku. Náhodná veličina slouží k vypočtení s jakou pravděpodobností daný projekt (ne)překročí požadovanou dobu trvání. Tento výpočet je popsán v kapitole zabývající se statistikou 2.2.1.1.

2.4.3 Simulace metody PERT

Výpočet PERT je také možné simulovat. Simulace pracuje s vygenerovanými hodnoty náhodného rozdělení. To je vytvořeno z vypočtených bodových odhadů střední hodnoty a směrodatné odchylky jednotlivých činností. Vlastnosti náhodného rozdělení jsou vypočteny podle stejných vzorců jako u metody PERT.

Na vygenerované hodnoty je poté možné aplikovat metody statistiky. Z těch lze zjistit minimální, maximální nebo střední dobu trvání projektu. Na získaných datech je také možné pozorovat stoupající přesnost výpočtů s rostoucím počtem vygenerovaných hodnot. Z těch je také možné určit kolikrát daná cesta byla kritická.

Vzhledem k tomu, že tyto metody nejsou specifické pouze pro simulaci PERT metody, popisuji je přímo v kapitole statistiky (2.2). Podklady pro tuto kapitolu jsem čerpal z [12].

2.5 Uvažované Technologie - Webová aplikace

Dostávám se již k závěrečné části teorie. V té popisuji jednotlivé technologie. Mohou zde být i popsány technologie, které jsem v implementaci nepoužil. Kapitola obsahuje pouze studie jednotlivých technologií. Důvody, které mě vedly ke zvolení jednotlivých technologií, jsou uvedeny v kapitole Implementace (6).

2.5.1 Java

Tato kapitole shrnuje informace o jazyku, který byl poprvé představen v roce 1995.[13] Jedná se o jazyk, který již existuje přes dvě desetiletí. To je v dynamicky rozvíjející se oblasti, kterou IT bezesporu je, výrazná doba. Přesto se však jedná stále o nejpopulárnější jazyk[14]. Od první verze Javy z roku 1995 proběhla spousta změn, nyní je již vyvíjen jazyk ve verzi 10. Nejvyšší produkční verzí v době psaní této práce zůstává Java 8.

Během vývoje přibyla do jazyka spousta nové funkcionality, která v první verzi chyběla. Vzhledem k tomu, že se jednalo o celkem osm hlavních verzí, docházelo ke změnám postupně. Nejvýraznější změny byly nejspíš obsaženy ve verzích 5 a 8. Ve verzi 5 došlo k přidání výčtových tříd a zejména generických typů[15]. V této verzi také byla obsažena implementace for-each iterace. Do této verze byly v jazyce využívány pro práci s kolekcí iterátory. K dalším velmi výrazným změnám došlo až ve verzi 8[16]. V této verzi došlo k přidání lambda funkcí. Ty přidávají další možnosti pro práci s kolekcí. Lambda funkcí je také možné nahradit vytvoření anonymní třídy.

Nyní se již dostávám k samotným principům jazyku. Jedná se o plně objektový jazyk s vlastní správou paměti. Java se řadí mezi interpretované jazyky. Zdrojový kód je nejprve zkompilován do platformně nezávislého byte kódu. Ten je poté spouštěn ve virtuálním stroji Javy (JVM). V něm je kód převeden interpretem jazyka do nativního strojového kódu cílového stroje. Vzhledem k těmto principům je zkompilovaný kód Javy platformě nezávislý.

Zatím všechny uvedené fakta platí jak pro Javu Standard Edition (SE), tak i pro Javu Enterprise Edition (EE). Mezi nimi je však obrovský rozdíl. Java SE poskytuje základní funkcionalitu jazyka jako jsou kolekce nebo konektor JDBC. Java EE je vybudována na těchto základech a nabízí možnosti pro vývoj produkčních aplikací. Mezi tuto funkcionalitu patří technologie pro vývoj webových aplikací: Java Server Pages (JSP) a JavaServer Faces (JSF). Mezi technologie poskytované Javou EE patří také Java Persitence API.

Další z důvodů, proč Java patří mezi nejoblíbenější jazyky, je množství frameworků a nástrojů, které během existence tohoto jazyka vznikly. Mezi tyto nástroje se mohou řadit frameworky řídící celé flow webové aplikace, ORM frameworky nebo sem také spadají nástroje pouze pro view vrstvu webových aplikací.

Podklady pro tuto kapitolu jsem nalezl v knižní publikaci Mistrovství v Javě[17].

2.5.2 JPA

Java persistence API je rozhraní pro mapování objektů z Javy do relačních databází. Jedná se o rozhraní nikoliv implementaci. Je to kontrakt, který musí implementovat ORM nástroje. JPA[18] představuje uznávaný standard pro ORM nástroje. Dále uvádím ukázky anotací a jejich význam.

- *Id* - jedná se o primární klíč tabulky.
- *Column(name = "graph_node_id")* - jedná se o sloupec v tabulce s uvedeným názvem.
- *Entity* - příznak, že se jedná o mapovací třídu.
- *ManyToOne* - Existuje více záznamů, které se mapují na jediný. Jedná se většinou o cizí klíče.

- *OneToMany* - Pro jeden záznam existuje v jiné tabulce více referencí. Často se jedná o anotaci položky pro cizí klíč u řídící tabulky.

2.5.3 Spring Framework

Jedná se o jeden z nejrozšířenějších frameworků pro tvorbu aplikací v Javě. Spring nejen nabízí prostředky pro implementaci běžných klientských aplikací, ale také rozšíření pro webové aplikace.

Framework je členěn do jednotlivých modulů. Tedy není nutné mít závislost na celý framework. Dále jsou v seznamu uvedeny příklady frameworků a jejich stručný popis.

Pro vytvoření této kapitoly jsem čerpal z knižní publikace Spring in Action[19].

Core Container
: Jedná se o základní funkcionalitu Springu. Do tohoto modulu spadají Bean a dependency injection.

Web
: Jedná se o rozšíření základního jádra o prvky pro tvorbu webové aplikace. Modul umožňuje mapování HTTP requestů na metody, které zajistí zpracování příchozího požadavku.

Data Access
: Modul pro přístup k datům. Ten obsahuje knihovny sloužící pro přístup k datům v databázi. Tento modul tedy umožňuje využití ORM nástroje. Sem také patří rozhraní JpaRepository, které je popsáno v kapitole 2.5.3.2.

Spring Security
: Modul pro zabezpečení aplikace. Přes Spring Security je možné omezit kompletní přístup k webu. Při příchodu vyskočí pop-up okno a bude nutné zadat heslo. Další z nabízených možností je správa uživatelů a přihlášení. V tomto modulu je možné přesně určit, odkud se berou data pro autentizaci a jaké stránky mají být zabezpečeny.

Test
: Modul pro otestování aplikace. Tento modul umožňuje vytvářet testy s dostupným aplikačním kontextem.

Z dříve uvedených informací vyplývá, že Spring nabízí spoustu možností pro návrh architektury aplikace. Mezi klíčové koncepty se řadí dependency injection. Tento koncept zajišťuje provázaní komponent v aplikaci.

Například máme v aplikaci servisní třídu, která je anotována anotací Service. Poté již stačí vytvořit třídní proměnnou této třídy, která je označena anotací Autowired. Pokud je v aplikaci povoleno skenování tříd, zajistí již Spring vytvoření instance dané servisní třídy.

Poté je možné přenechat spravování životního cyklu jednotlivých servisních tříd Springu. To přináší velkou výhodu. Ta spočívá v tom, že u jednotlivých tříd je možné definovat rozsah jejich životního cyklu. Například pokud mám data, která si chci udržovat pouze v rámci requestu. Mohu tyto data definovat jako komponentu s anotací Scope("request"). Spring poté zajistí vytvoření instance komponenty při začátku requestu a po jeho dokončení se postará o odstranění dat.

Tato struktura aplikace zjednodušuje i testování tříd. Pokud se testovaná třída skládá z komponent, je možné v rámci testů tyto komponenty ručně nastavit. Poté mohu otestovat třídu a místo databázové vrstvy mohu podstrčit svoji komponentu s mockovanými údaji. Zjednodušenou ukázku kódu s popisovanými principy je možné nalézt pod tímto textem.

```
@Repository
public interface CUserRepository extends JpaRepository<CUser,Integer>
    {
   CUser findByLogin(String login);

}

public class UserManageService {
   @Autowired
   private CUserRepository userRepository;

   public void setUserRepository(CUserRepository repository) {
      this.userRepository = userRepository;
   }

   public CUser findUserByLogin(String login) {
      return userRepository.findByLogin(login);
   }
}

public class CUserMock implements CUserRepository {
   public CUser findByLogin(String login) {
      CUser user = new CUser();
      user.setLogin("Test");
      return user;
   }
}

public class UserManageServiceTest {
   @Test
   public void test() {
      UserManageService userManageService = new UserManageService();
      userManageService.setUserRepository(new CUserMock());
      CUser user = userManageService.findUserByLogin("Test");
      Assert.assertEquals("Test", user.getLogin());
```

```
    }
}
```

2.5.3.1 Spring Boot

Spring je určitě užitečný framework. Pro vytvoření první webové stránky s použitím Springu je ale nutné nakonfigurovat spoustu konfiguračních částí frameworku. Tato konfigurace je často u většiny aplikací shodná a šla by tedy použít defaultní konfigurace. Výrazně by se tak usnadnilo vytvoření nového projektu vybudovaném na Springu.

Touto otázku se určitě zabývali programátoři, kteří tvoří tento oblíbený framework. Řešení této otázky bylo představeno 22. ledna 2014[20]. Jednalo se o modifikovaný framework Spring, který byl představen pod názvem Spring Boot.

Spring Boot umožňuje rychlé vytvoření nové aplikace postavené nad tímto frameworkem. Spolu se vznikem Spring Boot došlo také k vytvoření projeku Spring Initializr. Zde je možné zadat požadované závislosti na modulech a dalších Java knihovnách. Po vyplnění potřebných dat lze vygenerovat projekt. Ten obsahuje všechny závislosti a ukázkové Java třídy. Poté je již možné vzít vygenerovaný projekt a doplňovat vlastní implementaci.

Pokud je vyvíjena webová aplikace (nutný web modul), je možné spouštět aplikaci standardním příkazem java -jar app.jar, kdy app.jar představuje vyvíjenou aplikaci. Spring Boot již zajistí spuštění vestavěného serveru. Je tedy možné poměrně rychle vyvíjet webovou aplikaci bez starostí s instalací webového serveru a nasazováním aplikace.

2.5.3.2 JPA repository

Ve frameworku Spring se těší silné podpoře datová integrace. Jednu z možností jak získat data z databáze, popisuji v této kapitole. Spring nabízí rozhraní JpaRepository<T, ID extends Serializable>. To obsahuje dva generické typy. T je třída, která se přes ORM mapuje na relační databázi. ID je poté typ primárního klíče v této tabulce. Ukázku implementace tohoto rozhraní je možné shlédnout pod tímto textem.

```
@Repository
public interface CUserRepository extends JpaRepository<CUser,Integer>{

   CUser findByLogin(String login);

   CUser findByUserId(Integer id);

   List<CUser> findCUserByUserIdIsNot(Integer id);

}
```

Název metody	WHERE klauzule
findByUserId	user_id = {1}
findByUserIdAndName	user_id = {1} and name = {2}
findByUserIdOrName	user_id = {1} or name = {2}
findByNameLike	name like {2}
findByNameIn(Collection<String> names)	name in {1}
findCUserByUserIdIsNot	name <> {1}

Tabulka 2.1: Ukázka metod tvořených Springem

V ukázce jsou předvedeny výhody tohoto rozhraní. Není potřeba psát žádné SQL. To vytváří framework podle názvu metody. V případě, že se jedná o SQL dotaz s klauzulí WHERE, jsou do této podmínky dosazeny parametry metody. Vzhledem k tomu, že SQL příkaz je vytvářen podle názvu metody. Je nutné držet se u pojmenování metody specifikace frameworku.

Klíčovým prvkem pro pojmenovávání metod jsou názvy třídních proměnných v mapovacích Java třídách. Na ty je poté možné aplikovat různé modifikátory. Podmínky na jednotlivé proměnné lze spojovat logickými operacemi *and* a *or*. Na základě pojmenování poté vytvoří Spring SQL dotaz. V tabulce 2.1 uvádím různé příklady názvů a k nim příslušné WHERE klauzule.

Zatím jsem předvedl možnosti pro vytváření SQL dotazů přes Spring. Jak ale řešit, že je nutné provést složitější dotaz, například join přes dvě tabulky. Zde už je nutné psát SQL příkaz. Ten je přiřazen k funkci, kterou si můžeme pojmenovat dle libosti. Pro přiřazení je poté nutné uvést k funkci anotaci Query a samotný SQL dotaz.

Za další výhodu rozhraní JpaRepository je možné považovat to, že není nutné řešit otevření připojení k databázi a následně jeho uzavření. To je již řešeno na úrovni frameworku. Bez použití JpaRepository je nutné napsat spoustu výplňového kódu, který řeší tyto operace. Tento kód je navíc u všech SQL příkazů stejný. Níže uvádím ukázku vlastního SQL příkazu.

```
@Repository
public interface CUserRepository extends JpaRepository<CUser,Integer>{
@Query(select u.* from c_users u~join c_user_type t on u.type_id =
    t.type_id where t.type_name = ?1)
List<CUser> findCUserWithPermissionName(String name);

}
```

Uvedené rozhraní také umožňuje provádět odstranění a editaci databázových tabulek. U těchto operací jsou použity podobné principy jako je tomu u SQL

dotazů. Vzhledem k tomu, že u SQL dotazů jsem tyto principy podrobně popsal, neuvádím je znovu pro tyto operace.

2.5.4 Vaadin

Ve světě IT se jedná o poměrně mladý framework, který byl představen v roce 2001. V současné době je poslední verzí Vaadin 8.1 (25. 1. 2018). Ten byl vydán v lednu 2018.

Vaadin[21] je framework pro tvorbu webových aplikací v Javě. Jedná se o nástroj, který nabízí velké množství možnosti. Mezi ně patří připravené komponenty pro tvorbu webových stránek nebo podpora pro využití AJAXu.

Ve frameworku pravděpodobně najdete vše, co hledáte. Vaadin obsahuje základní komponenty jako jsou tlačítka, checkboxy nebo tabulky. Dále zde můžete nalézt různé varianty, jak vytvořit menu nebo využít vyskakovací dialogy. Vaadin také podporuje různé druhy grafů.

Také však do systému přidáte závislosti na knihovny, které vůbec nepotřebujete. Vaadin je totiž framework pro tvorbu velkých produkčních aplikací. Vaadin je primárně používán pro tvorbu GUI, ale poskytuje mnohem více funkcionality. Do té patří zpracování GET a POST requestů nebo autorizace uživatelů. Spousta možností nabízí výhody pro tvorbu velké aplikace, ale u menších aplikací to může představovat problém, protože použijete pouze zlomek funkcionality a nalézt tuto část nemusí být vždy jednoduché.

2.5.5 Thymeleaf

Jedná se o šablonový systém, který byl vyvinut španělským softwarovým inženýrem Danielem Fernándezem. Thymeleaf byl představen v červenci 2011 jako open-source software. V současné době je Thymeleaf vyvíjen skupinou The THYMELEAF team. Aktuální verze Thymeleaf je nyní 3.0.9., ta byla vydána v prosinci 2017.

Thymeleaf využívá šablony v šesti různých formátech. Ty uvádím v následujícím seznamu:

- HTML
- XML
- Text
- Javascript
- CSS
- RAW

Pro tvorbu šablon jsou zajímavé zejména formáty HTML a XML. Ty pracují s Thymeleaf syntaxí, která slouží pro vykreslení dat obdržených z WebContextu (Java třída) aplikace. Jedná se o interní třídu Thymeleaf, kterou je nutné vytvořit a naplnit daty v rámci backendu aplikace. Šablony také pracují se statickými texty vytaženými do konfiguračních souborů.

Pro tuto kapitolu jsem čerpal z oficiální dokumentace ThymeLeaf[22].

2.5.6 Primefaces

Jedná se o open source knihovnu pro tvorbu uživatelského rozhraní nad technologií JSF. První produkční verze této knihovny byla vydána v únoru 2010[23] tureckou společností PrimeTek. V současné době se za aktuální verzi považuje PrimeFaces 6.2.2 [24].

Integrace do projektu je jednoduchá. Stačí přidat závislost na jednu JAR knihovnu. Není třeba provádět žádnou konfiguraci. Jedinou další podmínku představuje to, že aplikace používá JSF. Poté si již stačí vybrat z mnoha předem připravených komponent. Knihovna zde nabízí velké množství různých tabulek, vstupní pole s kontrolou vstupních dat nebo je možné využít implementaci textového editoru. U Primefaces je nutné také uvést silnou podporu AJAXu. Ten je možné do aplikace integrovat formou listenerů nebo zpracováním událostí, které jsou z GUI vytvářeny.

Za zmínku určitě také stojí, že se jedná o dynamicky rozvíjející se projekt, který je využíván u velkých společností v různých odvětvích. Jedná se o leteckého dopravce Lufthansa, banku UniCredit, automobilovou značku Audi nebo výrobce grafických karet nVidia[25].

Primefaces se také těší výrazné podpoře komunity. Vzhledem k tomu, že se jedná o open source, může komunita výrazně přispívat k výsledné kvalitě knihovny. V neposlední řadě chci také uvést, že Primefaces disponují perfektní dokumentací s velkým množstvím ukázek jednotlivých komponent.

Kromě zdrojů uvedených přímo v kapitole, jsem pro vytvoření této kapitoly využil jako podklad dokumentaci knihovny[26].

2.6 Uvažované Technologie - Databáze

V této kapitole stručně popisuji technologie, které souvisejí s databází. Jedná se jak o samotný relační databázový systém, tak o technologii Liquibase, která umožňuje verzování datového modelu.

2.6.1 MySQL

Nejčastěji využívaná open source databáze v doméně relačních databází byla implementována švédskou firmou MySQL AB. V roce 2008 došlo k odkoupení MySQL společností Oracle[27]. Současnou verzí databáze vydanou společností

Název operace	Popis	Auto rollback
createTable	Vytvoření tabulky	Ano
addColumn	Přidání sloupce	Ano
addForeignKeyConstraint	Vytvoření cizího klíče	Ano
dropForeignKeyConstraint	Odstranění cizího klíče	Ne
createIndex	Vytvoření indexu	Ano
dropIndex	Odstranění indexu	Ne
createView	Vytvoření view	Ano
dropView	Odstranění view	Ne
renameTable	Přejmenování tabulky	Ano
delete	Odstranění tabulky	Ne

Tabulka 2.2: Popis operací Liquibase

Oracle je MySQL 8.0. Vzhledem k tomu, že se jedná o poměrně běžnou implementaci SQL, není zde v práci dále popisována.

2.6.2 Liquibase

Liquibase je nástroj pro správu databáze implementovaný v Javě. Nástroj byl poprvé vydán v roce 2007[28]. Poslední verzi Liquibase představuje verze 3.6.0 v dubnu 2018[29].

Liquibase slouží pro vytváření databázových objektů a podporuje většinu implementací SQL jazyka. Nástroj řeší častý problém verzovaní datového modelu. Úpravy modelu jsou ukládány do celků nazývaných changelogy. Ty je možné uchovávat ve formátech: XML, YAML, JSON nebo SQL. Zde se již jedná o textové soubory. Tedy lze je jednoduše verzovat.

Dané changelogy již stačí pouze spustit knihovnou. O samotné nasazení se poté postará Liquibase. Ta si udržuje ve vlastních tabulkách, co již bylo nasazeno. Na základě changelogů a uvedených tabulek vygeneruje knihovna rozdílové SQL skripty. Ty je poté možné nasadit do vývojové databáze nebo dodávat do produkce.

Liquibase také nabízí možnost rollbacku daných operací. Stačí pouze spustit skript s jinými parametry a ten následně vygeneruje nebo nasadí úpravy pro odstranění přidávaných změn. U většiny operací je rollback generován automaticky, u některých je ale nutné rollback operaci uvést explicitně. V tabulce 2.2 uvádím příklady SQL operací a zda podporují automatický rollback.

Pro vytvoření této kapitoly jsem použil dokumentaci Liquibase[30].

2.7 Hibernate

Vzhledem k tomu, že tato práce se zabývá tvorbou webové aplikace v Javě, pracuje se na aplikační úrovni s objekty. Data jsou však ukládána do relační databáze. Nástroj Hibernate pro mapování mezi objekty a relační databází popisuji v této kapitole.

Hibernate je ORM framework, který vyvinul programátor Gavin King v roce 2001. V současné době se o vývoj frameworku stará společnost Red Hat. Aktuální verzí knihovny je Hibernate OGM 5.3.1.Final [31]. Jedná se o nástroj vyvíjený v Javě a pro ni je také určený.

Framework nabízí dva formáty pro konfiguraci propojení databáze a aplikace: XML a anotace v Java třídách. Možnosti obou formátů jsou shodné. Jedná se pouze o odlišný formát, v XML se stejně odkazuje na Java třídy. U nových projektů je většinou preferována varianta anotací.

Po vytvoření mapovacích XML nebo přidání anotací, se již o mapování mezi aplikací a databází postará framework. Pro propojení s databází je nutné také nakonfigurovat informace o databázi (uživatel, heslo, adresa databáze...). Druhou možností je integrace Hibernate spolu s jiným frameworkem, například Spring. Poté již samotné připojení do databáze spravuje Spring a Hibernate zajišťuje pouze mapování mezi aplikací a databází.

Podklady pro tuto kapitolu jsem našel v [32].

2.8 Maven

Jedná se o nástroj od společnosti Apache Software Foundation pro usnadnění vytváření buildů aplikace. Maven byl poprvé vydán v roce 2006[33]. Současná verze knihovny je 3.5.3[34].

Hlavní konfigurační komponentou nástroje je Project Object Model (POM). V projektu je představován souborem pom.xml v kořenu aplikace. V XML souboru jsou popsány jednotlivé operace, které jsou nad projektem prováděny. Také obsahuje informace o vyvíjené aplikaci, například verzi. Operace z POMu jsou poté spouštěny přes Maven. Přes ten je možné řídit, jaké operace mají být provedeny.

Zatím popsané informace nepřináší výrazný rozdíl od jiného buildovacího nástroje Antu. V Antu je však nutné přidat všechny závislosti do projektu ručně. V Mavenu je možné definovat závislosti na jiné projekty. Ty jsou poté při kompilování aplikace staženy z repozitáře Mavenu. V těch se nachází téměř deset milionů různých aplikaci.

Jako podklady pro tuto kapitolu jsem použil[35].

2.9 Git

Systém pro verzování zdrojového kódu původně implementoval Linus Torvalds jako nástroj pro verzování jádra Linuxu, na kterém se podílel. Git původně nebyl zamýšlen jako běžný verzovací systém pro velké projekty. Přesto je však tento nástroj z roku 2005 aktuálně nejrozšířenější verzovací systém.

Git představuje distribuovaný verzovací systém. Tedy uživatel má lokální kopii repozitáře na svém stroji, na této kopii provádí operace. Ty následně musí doručit na vzdálený server.

Lokální repozitář Gitu je členěn do tří částí: Working directory, Staging area a Local repository. Working directory je adresář, ve kterém pracujete. Obsahuje všechny soubory ve složce. Ty nemusí být ani verzovány. Do Staging area si přidávám soubory z Working directory. Zde si modeluji a připravuji soubory ke commitu. Po commitnutí daných souborů se vytvoří commit, který spadá už do Local repository. Z něj je poté nutné doručit commity do vzdáleného repozitáře na serveru.

Pro tvorbu této kapitoly jsem čerpal z dokumentace Gitu[36] a knihy Pro Git[37].

Kapitola 3

Upřesnění zadání a cílů této práce

Jak jsem již uvedl v Úvodu této práce, zde se nachází upřesnění praktické části práce. Pořadí kapitol vyplynulo z vodopádového modelu vývoje, který poprvé představil pan Winston W. Royce v roce 1970 [38]. Tento model vývoje jsem považoval za vhodný proto, že u závěrečná práce mám předem určené zadání. Ze zadání je možné přesně určit, co by aplikace měla splňovat a jaké požadavky na ni budou kladeny ještě před počátkem implementace.

Jednotlivé kapitoly dále uvádím v seznamu. Každá uvedená kapitola poté obsahuje další seznam, kde jsou obsaženy jednotlivé kroky. Ty je pro její splnění nutné provést. V seznamu nemusí být všechny kapitoly, protože pokud kapitola obsahuje jedinou činnost, nevidím smysl uvádět ji v seznamu s podrobným rozpadem činností.

- Analýza a návrh systému
  1. Určit procesy, které aplikace bude podporovat.
  2. Určit funkční a nefunkční požadavky.
  3. Provést mapování mezi funkčními a nefunkčními požadavky.
  4. Určit způsob ukládání dat a případně navrhnout relační model databáze.
- Návrh uživatelského rozhraní aplikace
  1. Specifikovat potřebné obrazovky.
  2. Určit flow mezi obrazovky.
  3. Rozhodnout jak odlišit administrátorské prvky na úrovni GUI.
- Implementace

1. Určit použité technologie.
2. Navrhnout design aplikace.
3. Provést implementaci.

- Testování
  1. Implementace regresních testů v Javě.
  2. Vytvoření testovacích scénářů.
  3. Testování aplikace.
  4. Oprava chyb.
  5. Klasifikační testování.
- Dokumentace
  1. Kontrola existence Javadocu z implementační části práce.
  2. Vygenerování dokumentace a její popis.
  3. Vytvoření instalační příručky.
  4. Vytvoření uživatelské příručky.

Kapitola 4

Analýza a návrh systému

Nyní se již dostávám k analytické části práce. Na počátku analýzy proběhla diskuze s vedoucí mé práce o procesech, které má aplikace podporovat. Na základě těchto procesů jsem identifikoval jednotlivé funkční a nefunkční požadavky. Ty je v rámci návrhu a implementace systému nutné splnit. Kapitola také obsahuje podrobný popis požadavků a mapování mezi procesy a funkčními požadavky.

Vzhledem k tomu, že systém pracuje s daty, obsahuje kapitola také analýzu správy dat. V té nejdříve popisuji, proč jsem se v této doméně rozhodl pro zvolené řešení. Dále tato část obsahuje samotný návrh relačního modelu databáze.

4.1 Procesy

V této kapitole jsou popsány hlavní procesy, které bude „Aplikace PERT“ podporovat. Vzhledem k tomu, že systém primárně vzniká na podporu předmětu Metody a projektové řízení na ČZU, mapují procesy sylabus tohoto předmětu. V přiloženém seznamu jsou uvedeny procesy, které souvisí s výše uvedeným předmětem.

Druhý výrazně náročnější cíl, který nelze přímo mapovat na konkrétní procesy, je vytvořit dostatečně zajímavý nástroj pro simulaci odhadů pomocí metody PERT. Zajímavý je myšleno v tom smyslu, že dokáže zaujmout i jiné skupiny uživatelů než studenty. Pro ty slouží „Aplikace PERT“ jako doplněk v rámci jejich studia na univerzitě.

Věřím, že používání nastroje povede k častějšímu využití stochastické metody PERT. V praxi totiž momentálně dochází častěji k používání jednodušší metody CPM, která pracuje s fixní dobou trvání. Srovnání obou metod se nachází v teoretické části práce 2.4.2. Náhrada metody CPM metodou PERT povede k přesnějším odhadům nových systémů.

Dále uvádím výčet jednotlivých procesů:

1. Výuka teoretických znalostí pro projektové řízení.
2. Výuka teoretických znalostí pro metodu PERT.
3. Výuka teoretických znalostí pro simulaci PERT.
4. Výuka metody PERT (praktické ukázky).
5. Výuka simulace PERT (praktické ukázky).
6. Příprava materiálů pro výuku.

4.2 Funkční požadavky

Z uvedených procesů plynou funkční požadavky. Ty vyplynuly z analýzy procesů a diskuzí ohledně procesů s vedoucí mé práce. Požadavky dále člením dle oblastí, kam v rámci aplikace spadají.

4.2.1 Výpočty

1. Vizualizace výpočtů PERT.
2. Simulace PERT metody pomocí statistického rozdělení.
3. Umožnit u metody PERT upravovat parametry.
4. Umožnit u simulace metody PERT upravovat parametry.
5. Možnost editovat zadané projekty a uložit změny.
6. Přidání nového projektu a potřebných parametrů pro výpočty.
7. Odstranění projektu.

4.2.2 Teorie

1. Zobrazení teorie dle jednotlivých stránek.
2. Vytvoření nové stránky s teorií.
3. Editace textů v teoretické sekci aplikace.
4. Odstranění stránky teorie.

4.2.3 Autentizace

1. Autentizace a rozlišení uživatelských rolí.
2. Přidání uživatele.
3. Odstranění uživatele.
4. Editace uživatele

4.3 Nefunkční požadavky

V této kapitole uvádím nefunkční požadavky na aplikaci. Jedná se o omezení na počet uživatelů. To vyplývá z počtu studentů na cvičeních předmětu Metody a projektové řízení. Poté se zde ještě nachází dvě omezení na technologie.

1. Funkční pro minimálně 24 uživatelů.
2. Implementace v technologii Java.
3. Při využíti databázového stroje, použít technologii MySQL.

4.4 Mapování procesů na funkční požadavky - případy použití

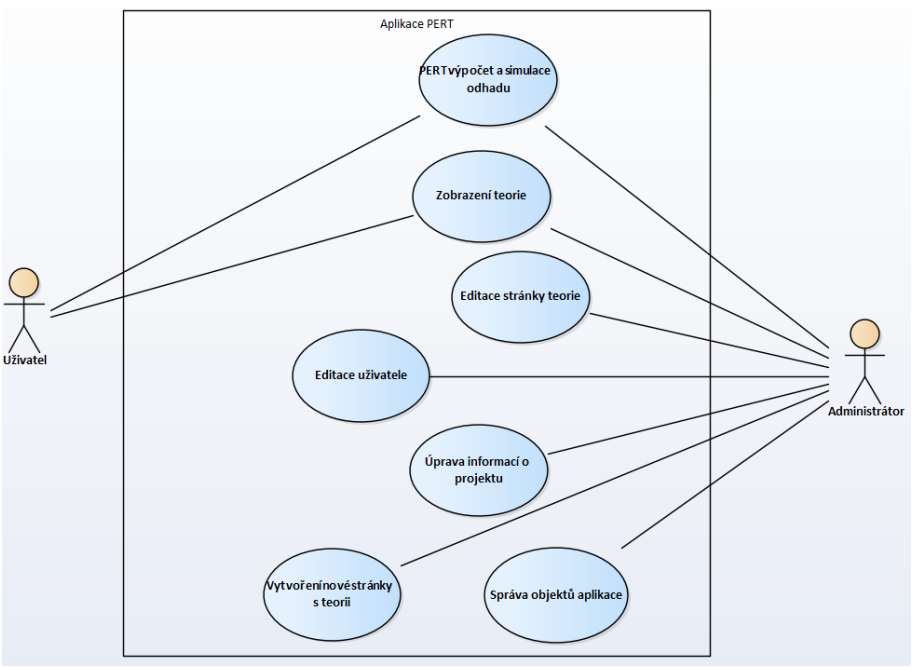
V práci jsem zatím uvedl procesy, které má systém podporovat. Následně uvádím funkční a nefunkční požadavky, které vyplynuly z analýzy procesů. Požadavky jsou v této práci ještě podrobně popsány, ale již zde uvedu mapování procesů na požadavky.

Mapování jsem se rozhodl zachytit v případech použití. Dále se nachází jednotlivé scénáře. Ty mají vždy stejnou strukturu, nejdříve popíši ke kterým procesům a funkčním požadavkům se daný scénář váže. Poté uvádím samotné kroky jednotlivých scénářů. Pokud při vykonání scénáře mohou nastat výjimky ze standardního průběhu, uvádím je u příslušného scénáře.

Případy užití uvedené v této kapitole pokrývají kompletní funkcionalitu Aplikace PERT. Pro větší přehlednost jsou případy členěny dle uživatelských rolí v systému. Digram pro případy užití je možné nalézt na obr. 4.1.

4.4.1 Běžný uživatel

Nejdříve uvádím scénáře použití pro běžné uživatele. Vzhledem k tomu, že portál pro běžné uživatele nenabízí konfiguraci vlastních projektů. Jsou pro běžné uživatele reálné pouze tři scénáře použití, kvůli velké shodě jsem navíc simulaci PERT metody a výpočet PERT spojil do jediného scénáře. Z těchto důvodů se nachází v této kapitole pouze dva scénáře použití.



Obrázek 4.1: Digram případů použití pro role běžného uživatele a administrátora

4.4.1.1 PERT výpočet a simulace odhadu

Jedná se o výpočty metodou PERT a simulaci těchto výpočtu pomocí statistického rozdělení na základě odhadnutých dob trvání (vstupní data). Tento scénář pokrývá procesy pro výuku PERT výpočtu a jeho simulaci. Funkční požadavky se zde mapují na první čtyři v sekci výpočtů (4.2.1).

1. Přejít na rozcestník PERT výpočtu nebo simulace.
2. Vybrat si konkrétní projekt pro výpočet.
3. Nastavit parametry jednotlivých činností (Odhady doby trvání).
4. Provést výpočet.
5. 4. a 5. krok se může několikrát opakovat.

Výjimky
3. a. Zadané shodné hodnoty optimistické a pesimistické doby trvání činnosti.

4.4.1.2 Zobrazení teorie

Pro samotné výpočty je nutné pochopení teorie, na které jsou výpočty postaveny. Vyhledání potřebné teorie je zachyceno v tomto scénáři. Ten se mapuje

na procesy pro výuku teorie a pokrývá první funkční požadavek v sekci teorie (4.2.2).

1. Přejít z úvodní obrazovky na stránku teorie.

2. Přepnout se na hledanou stránku v záložkách.

4.4.2 Administrátor

Nyní se již dostávám k samotné přípravě dat. Ta souvisí s nutností administrátorských oprávněních, tedy do aplikace je již nutné se přihlásit. K přípravě dat se vztahuje jediný proces a to „Příprava materiálů pro výuku“. Ten je pro všechny scénáře společný a proto ho již neuvádím u jednotlivých scénářů. Funkční požadavky jsou stále uváděny stejně jako v předchozí kapitole.

4.4.2.1 Vytvoření nového projektu

Scénář popisuje vytvoření samotného projektu, sestrojení jeho grafu s činnostmi a nastavení jejich doby trvání. Scénář se mapuje na požadavek vytvoření grafu v sekci výpočtů (4.2.1).

1. Přihlásit se do aplikace.

2. Přejít na správu aplikace.

3. Vyplnit název projektu.

4. Kliknout na položku „Vytvořit nový projekt“.

5. Přejít do sekce grafů a zvolit si vybraný graf.

6. Vytvořit strukturu grafu a uložit do databáze.

7. V detailu činností nastavit pracnosti činností.

8. Uložit data o činnostech.

Výjimky
1. a. Zadáno neexistující uživatelské jméno.
1. b. Zadáno špatné heslo.
4. a. Nevyplněn název projektu nebo vyplněn již existující název.
6. a. Zadán nevalidní graf.

4.4.2.2 Úprava informací o projektu

Zde se jedná o úpravy již existujícího projektu. Důvodem můžou být buď požadované změny v topologii grafu, například přidání nové činnosti. Nebo je nutné provést úpravu v datech jednotlivých činností. Tento scénář se mapuje na požadavek editace projektů v sekci výpočtů (4.2.1).

1. Na úvodní obrazovce se přihlásit.
2. Přejít na rozcestník Grafů (PERT/Simulace).
3. Vybrat graf, který chci editovat.
4. Upravit topologii grafu a uložit do databáze.
5. Nastavit data činností.
6. Uložit data o činnostech.

Výjimky
1. a. Zadáno neexistující uživatelské jméno.
1. b. Zadáno špatné heslo.
4. a. Zadán nevalidní graf.

4.4.2.3 Vytvoření nové stránky s teorii

Při provozu systému dojde také k situaci, kdy je žádoucí vytvářet nové stránky s teorií. Situace se váže na druhý požadavek v sekci teorie (4.2.2).

1. Na úvodní obrazovce se přihlásit.
2. Přejít do správy aplikace.
3. Vyplnit název nové stránky.
4. Uložit stránku do databáze.

Výjimky
1. a. Zadáno neexistující uživatelské jméno.
1. b. Zadáno špatné heslo.
3. a. Nevyplněn název stránky nebo vyplněn již existující název.

4.4.2.4 Editace stránky teorie

Určitě také nastanou situace, kdy bude nutné provést editaci současných textů teorie. To také souvisí s předposledním požadavkem v sekci teorie (4.2.2).

1. Na úvodní obrazovce se přihlásit.
2. Přejít na část aplikace s teorií.
3. Vybrat si požadovanou stránku a kliknout na editaci.
4. V editoru provést požadované změny (styl písma, barvy, nadpisy...).
5. Zobrazit si náhled.
6. Provést uložení.

Výjimky
1. a. Zadáno neexistující uživatelské jméno.
1. b. Zadáno špatné heslo.

4.4.2.5 Správa objektů aplikace

V systému je nejenom nutné vytvářet nové objekty, ale také bude docházet k odstraňování již zastaralých údajů. Postup odstraňování je popsán v následujícím scénáři. Ten se váže k požadavkům na odstranění dat v příslušných sekcích požadavků.

1. Na úvodní obrazovce se přihlásit.
2. Přejít do správy aplikace.
3. Vybrat si ze seznamu hledaný objekt.
4. Provést odstranění.

Výjimky
1. a. Zadáno neexistující uživatelské jméno.
1. b. Zadáno špatné heslo.
4. a. Objekt neexistuje.

4.4.2.6 Vytvoření uživatele

Následující scénář popisuje postup, jak vytvořit nového uživatele. Scénář souvisí s druhým požadavkem v sekci autentizace (4.2.3).

1. Přihlásit se.
2. Přejít do správy aplikace.
3. Kliknout na tlačítko nový uživatel.
4. Na nové stránce vyplnit potřebné údaje o uživateli.
5. Provést uložení.

Výjimky
1. a. Zadáno neexistující uživatelské jméno.
1. b. Zadáno špatné heslo.
5. a. Nezadány všechny údaje o uživateli.
5. b. Zadán již existující login.
5. c. Heslo a jeho potvrzení neobsahují shodné hodnoty.

4.4.2.7 Editace uživatele

Také se může stát, že je nutné změnit již existující údaje uživatele. Jedná se o poslední požadavek v sekci autentizace (4.2.3).

1. Přihlásit se.
2. Přejít do detailu uživatele.
3. Provést požadované změny.
4. Změnit heslo na samostatné obrazovce.
5. Uložit změny.

Výjimky
1. a. Zadáno neexistující uživatelské jméno.
1. b. Zadáno špatné heslo.
4. a. Neodpovídá aktuální heslo.
4. b. Heslo a jeho potvrzení neobsahují shodné hodnoty.
5. a. Zadán již existující login.

4.5 Podrobnější popis požadavků

Do této části práce jsou požadavky uvedeny pouze bodově a k nim uvedené scénáře použití. Z Krátkého bodového shrnutí nemusí být vždy zřejmé, co k tomuto požadavku vedlo. Ze shrnutí také není možné poznat, co se za tímto požadavkem skrývá. Kvůli těmto důvodům jsem tyto body v této kapitole podrobněji popsal.

4.5.1 Vizualizace výpočtů PERT

Systém nabídne přehlednější a názornější vizualizaci výpočtů PERT metody než současné materiály. Ty jsou zachyceny ve formě sešitů aplikace Excel. Tyto příklady jsou pro studenty těžké k pochopení a nenabízejí dostatečně přehlednou ukázku výpočtů.

4.5.2 Simulace PERT metody pomocí statistického rozdělení

V praxi je výpočet PERT metody možné simulovat využitím normálního rozdělení. Na základě hodnot doby trvání projektu je vypočtena střední hodnota a rozptyl rozdělení.

Systém poté z vypočtených dat (střední hodnota a rozptyl projektu) generuje na základě normálního rozdělení doby trvání těchto činností. Z těchto dat je poté možné určit kritickou cestu projektu. To je také přehledně prezentováno, aby studenti dokázali tuto problematiku pochopit.

4.5.3 Umožnit u metody PERT a její simulace upravovat parametry

V současných materiálech nemají studenti žádnou reálnou možnost, jak připravené grafy upravovat. Je sice možné upravit v několika listech Excelu parametry, ale spíše se student v podkladech ztratí. Aplikace nabídne jednoduchou editaci činností. Ta dokáže tuto činnost snížit z desítek minut na méně než minutu.

Pro studenty jsou v systému vytvořené grafy s určitým předem připraveným nastavením parametrů. Studenti poté mohou s těmito daty různě manipulovat a opakovaně provádět výpočty nebo simulaci.

Zejména sledování změněných vypočtených výstupů, umožní lépe porozumět dané problematice. V první verzi aplikace jsou zatím uvažovány pouze změny parametrů jednotlivých činností. Studenti nemají možnost upravit si samotný graf projektu, ani si upravené parametry činností uložit.

4.5.4 Zobrazení teorie dle jednotlivých stránek

Systém se snaží primárně znázornit výpočty metody PERT a také její simulaci. Pro pochopení těchto výpočtů je však nutná znalost potřebných teoretických

principů, nad kterými jsou tyto výpočty realizovány. Na to primárně míří tento funkční požadavek.

Teorie je prezentována formou strukturovaného textu. Vytvoření tohoto textu je možné v jednoduchém textovém editoru, který je do systému integrován. Samotné kapitoly teorie jsou dále shlukovány do jednotlivých stránek. Ty je možné přepínat v záložkách na stránce teorie.

4.5.5 Správa projektů

Vzhledem k tomu, že běžní uživatelé mají možnost přistupovat pouze k předem připraveným grafům. Je nutné, aby administrátoři systému mohli tyto grafy připravovat a případně editovat. V této kapitole tedy popíšu následující funkční požadavky.

1. Umožnit u metody PERT upravovat parametry.
2. Umožnit u simulace metody PERT upravovat parametry.
3. Možnost editovat zadané projekty a uložit změny.
4. Přidání nového projektu a potřebných parametrů pro výpočty.
5. Odstranění projektu

K editaci projektů jsou použity běžně dostupné struktury systému. Rozdíl však představují ovládací panely. Ty pro běžné uživatele nejsou zobrazené a tedy ani přístupné. U správy projektu se jedná o ovládací panel pro manipulaci se samotným grafem projektu (topologie grafu, názvy uzlů...). Na obrazovce jsou pro administrátory také dostupná tlačítka pro uložení dat do databáze.

Vytvoření nebo odstranění projektu je realizováno přes správu aplikace, která je dostupná po přihlášení uživatele. V té je možné vyplnit nový unikátní název projektu a ten vytvořit. Druhou možností pro správu projektů na této obrazovce je odstranění grafu. Pro to je nutné vybrat si projekt z nabízeného seznamu již existujících projektů a ten poté odstranit.

4.5.6 Správa Teorie

Hlavní cíl požadavků týkajících se správy teorie, je nabídnout intuitivní a použitelné GUI pro vytváření nutné teorie pro studenty předmětu Metody a projektové řízení na ČZU.

Vytváření a odstraňování jednotlivých stránek teorie je založeno na stejném principu, jaký je již použit pro shodné operace u projektů. Tedy ve správě aplikace se nachází sekce pro teorii. Editaci textací je nutné provést v integrovaném textovém editoru na samostatné obrazovce. Ten je podrobně popsán v analýze datového modelu a proto si dovolím pouze odkázat na tuto analýzu 4.6.2.

4.5.7 Autentizace a rozlišení uživatelských rolí

V systému je nutné rozlišit různé role uživatelů. Ty se v systému nacházejí dvě, běžný uživatel a administrátor. Běžný uživatel je každý, kdo využívá „Aplikaci PERT“, není potřeba žádné přihlášení. U administrátorské role je už nutné se přihlásit. Po přihlášení jsou dostupné ovládací panely, editace přihlášeného uživatele a přístup do správy aplikace.

4.5.8 Správa uživatelů

Tento požadavek skrývá jak správu přihlášeného uživatele, tak i správu všech ostatních uživatelů. To je způsobeno tím, že přihlášený uživatel má vždy roli správce systému. V systému je možné editovat údaje o přihlášeném uživateli, vytvářet nové uživatele nebo odstranit uživatele již existujícího.

4.5.9 Funkční pro minimálně 24 uživatelů

Aplikace slouží primárně pro praktické ukázky na cvičeních předmětu Metody a projektové řízení. Cvičení jsou dimenzovaná pro čtyřiadvacet studentů. Vzhledem k tomu, že v první verzi „Aplikace PERT“ není uvažována datová část pro uživatele, nevidím výrazné omezení počtu souběžně fungujících uživatelů.

Toto omezení však může nastat v případě, že dojde k přidání datové části (viz 10.3) a přihlašování uživatelů (viz 10.4). Za této konstelace spatřuji potencionální úzké hrdlo při ukládání upravovaných grafů do databáze. Zde je nutné mít k dispozici výkonný databázový server, který poskytne dostatečné množství připojení do databáze. V rámci aplikace je možné počet připojení do databáze konfigurovat. O jaké parametry se jedná je přesně uvedeno v instalační příručce aplikace.

4.6 Návrh relačního modelu databáze a související analýza

V rámci systému „Aplikace PERT“ je potřeba ukládat větší množství dat a tyto data zde také editovat. Fakta uvedená v předchozí větě tedy vylučují využít statickou inicializaci v rámci aplikace. Data by se znovu načetla při každém restartu systému. Vzhledem k tomu, že není možné vyloučit další navazující vývoj, nepovažuji za vhodné ani řešení „memory“ databázi (například H2). Na základě známých informací jsem se rozhodl využít standardní databázový server. V rámci analýzy se mi povedlo určit tři samostatné oblasti v databázi, které nejsou spolu provázány. Jedná se o data pro následující oblasti:

- Projekty - grafy,
- teorie,

Název sloupce	Datový typ	Popis sloupce
project_id	Int	Primární klíč záznamu.
name	Varchar(100)	Název projektu

Tabulka 4.1: Popis sloupců tabulky c_project

- autentizace.

V této kapitole jsem také řešil, jak ukládat strukturu grafů. Jedním z uvažovaných řešení bylo vytvořit aplikaci nad typově rozdílnými databázemi, relační a grafovou.

Nakonec zde převážilo to, že grafové databáze jsou poměrně nová technologie, která prochází stále výrazným vývojem. Tohle je jeden z důvodů, proč jsem vsadil na již stálou technologii relační databáze.

Druhý a rozhodně neméně důležitý důvod je pro mě skutečnost, že aplikace pak musí komunikovat s dvěma databázovými servery. Toto by ztížilo instalaci představitelům ICT z ČZU. Také by tím přibyl další prvek, který může způsobit nedostupnost aplikace.

Na základě informací uvedených v předchozích odstavcích textu jsem se tedy rozhodl vybudovat aplikaci nad čistě relační databází.

Návrhy jednotlivých částí databáze a popis tabulek uvádím v samostatných kapitolách.

4.6.1 Grafy - projekty

Část databáze sloužící pro ukládání projektů v systému. Nad projekty jsou poté prováděny jednotlivé metody výpočtu, PERT metoda a její simulace. V této části návrhu se řeší provázání jednotlivých projektů. Tuto strukturu jsem se rozhodl ukládat do tří tabulek. Vazby těchto tabulek jsou vidět na obr. 4.2. Tabulky jsou podrobně rozepsány v dalších podkapitolách.

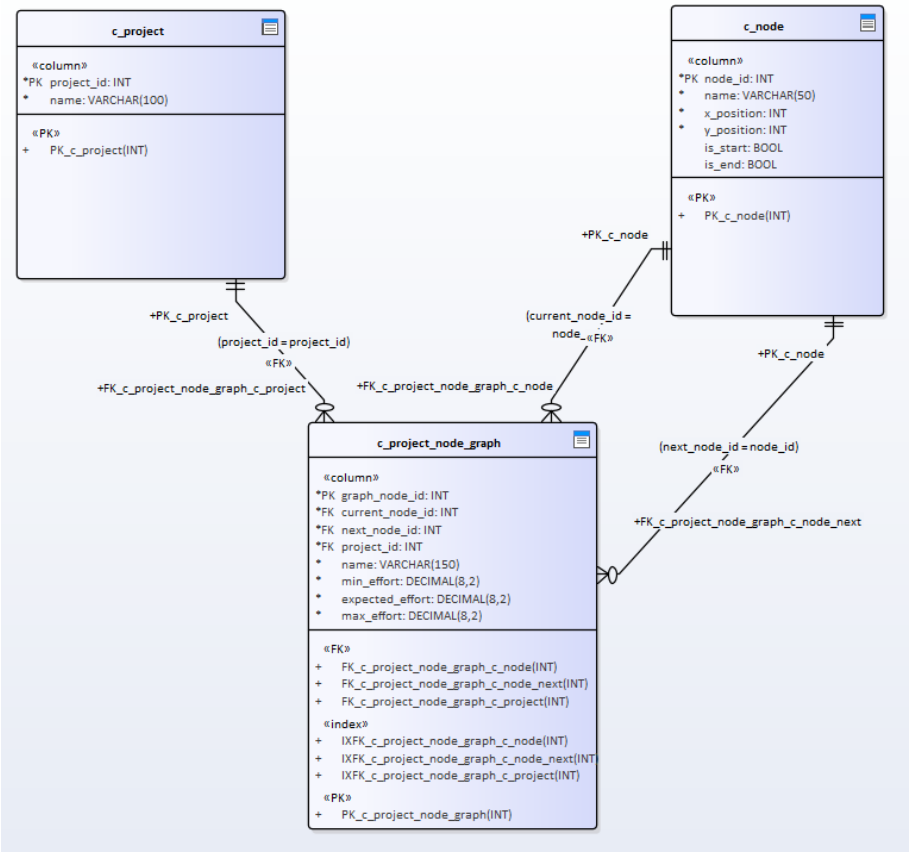
4.6.1.1 c_project

Tato tabulka zastřešuje obecné informace o projektu. V tabulce je evidován pouze název projektu. Posloupnost činností, z kterých se daný projekt skládá, je zachycena ve struktuře grafu, který je uložen v dalších tabulkách této oblasti databáze.

Popis samotných sloupců je možné najít v tabulce 4.1.

4.6.1.2 c_node

V této struktuře se ukládají informace o konkrétních uzlech. Tabulka obsahuje data s umístěním uzlů v rámci grafu. Je zde také uloženo, zda se jedná o počáteční nebo koncový uzel. Jednotlivé sloupce jsou popsány v tabulce 4.2.



Obrázek 4.2: Relační model grafů

Název sloupce	Datový typ	Popis sloupce
node_id	Int	Primární klíč záznamu.
name	Varchar(50)	Název činnosti.
is_start	Bool	Příznak zda se jedná o počáteční uzel grafu.
is_end	Bool	Příznak zda se jedná o konečnou činnost projektu.

Tabulka 4.2: Popis sloupců tabulky c_node

Název sloupce	Datový typ	Popis sloupce
graph_node_id	Int	Primární klíč záznamu.
node_id	Int	Cizí klíč do tabulky c_node, odkaz na informace o aktuálním uzlu .
next_node	Int	Cizí klíč do tabulky c_node, odkaz na potomka uzlu. Tedy kam z aktuálního uzlu povede hrana.
project_id	Int	Cizí klíč do tabulky c_project. Určuje příslušnost hrany ke konkrétnímu projektu. Na základě toho klíče patří k projektu také krajní uzly hrany.
min_effort	Decimal(8,2)	Minimální pracnost činnosti.
expected_effort	Decimal(8,2)	Očekávaná pracnost činnosti.
max_effort	Decimal(8,2)	Maximální pracnost činnosti.

Tabulka 4.3: Popis sloupců tabulky c_project_node_graph

4.6.1.3 c_project_node_graph

Jedná se o vazební tabulku mezi uzly a grafem. Tabulka tedy slouží k zachycení topologie grafu. Toho je dosaženo uložením informací o současném uzlu a jeho následovníku. Záznamy také obsahují data pro výpočty. Jedná se o optimistickou, očekávanou a pesimistickou dobu trvání činnosti. Podrobný popis jednotlivých sloupců se nachází v tabulce 4.3.

4.6.2 Teorie

„Aplikace PERT“ také poskytne teoretické podklady pro výpočty pomocí jednotlivých metod. Zde jsem čelil dvěma problémům, jak uložit strukturovaný text a jak umožnit tento text editovat. Zobrazení nepředstavovalo problém, došlo by k nastylování jednotlivých úrovní nadpisů. Mnohem větší problém však představovalo vytvoření stránky. Tuto problematiku dále popisuji v následujících odstavcích.

První navržené řešení počítalo s postupným budováním textu od nejnižších struktur. Vznikla by tak posloupnost struktur od odstavce, přes jednotlivé nadpisy, až po celé stránky teorie. Tato struktura by se mapovala na 6 tabulek. Tento návrh databázové části je možné prostudovat na obr. 4.3.

Zde uvádím seznam tabulek a jejich stručný popis:

- c_page - Stránka teorie.
- c_chapter - Jednotlivé kapitoly v rámci stránky.
- c_page_chapter - Vazební tabulka mezi stránkou a kapitolou.
- c_level - Úroveň nadpisu.
- c_text - Odstavec textu.
- c_chapter_text - Provázání textu a kapitoly.

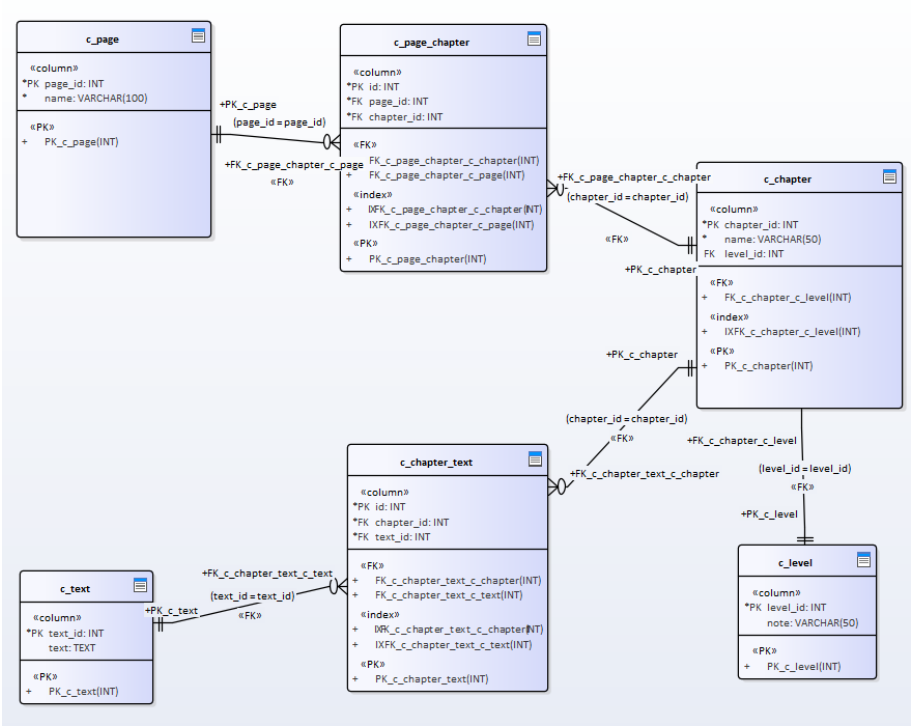
U tohoto řešení vystávaly problémy, jak udělat jednoduché GUI pro editaci struktury textu. Jak například poznat, že se jedná o odstavec textu a má být provázán s touto kapitolou? Nebo jak nastavovat úroveň nadpisů? Určitě se jedná o řešitelné problémy.

Mezi možná řešení patří rozbalovací menu s úrovní pro nadpis kapitoly nebo pevná struktura webové stránky, která si vynutí přiřazení textu ke kapitolám. Domnívám se však, že tato implementace editace textů je příliš složitá a vedla by spíše k tomu, že teoretické materiály by existovaly mimo „Aplikaci PERT“.

Proto jsem tuto variantu již v analýze vyřadil a snažil se nalézt jinou alternativu. Tu představuje textový editor z frameworku Primefaces. Ten podporuje také standardní zkratky, například Ctrl + B pro tučné písmo. U editoru jsem se rozhodl pro využití pouze části funkcionality. Jednotlivé možnosti uvádím v následujícím seznamu:

- Tučný text,
- kurzíva,
- podtržené písmo,
- škrtnuté písmo,
- barva textu a pozadí,
- nadpisy,
- číslovaný seznam,
- bodový seznam.

Zvolením této varianty jsem výrazně zredukoval datový model. Z původní struktury jsem použil pouze jedinou tabulku. Ta představuje kombinaci původních tabulek c_page a c_text, protože místo struktury je momentálně ukládán pouze text. Editor totiž vytváří jeho strukturu tak, že do textu přidává HTML tagy. Na základě těchto tagů poté dochází k vykreslení textu.



Obrázek 4.3: Relační model grafů

Název sloupce	Datový typ	Popis sloupce
page_id	Int	Primární klíč záznamu.
name	Varchar(100)	Název stránky s teorií.
text	Text	Samostatný text kapitoly.

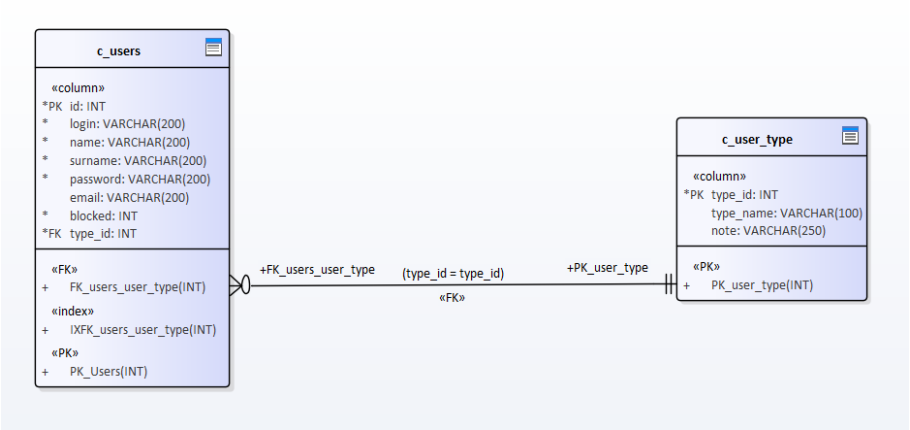
Tabulka 4.4: Popis sloupců tabulky c_page

4.6.2.1 c_page

V této tabulce se nachází jednotlivé stránky. Tabulka obsahuje název stránky a text rozšířený o HTML tagy. Na základě těchto tagů je poté text vykreslen. Popis sloupců je možné najít v tabulce 4.4.

4.6.3 Autentizace

V systému připravená data je třeba spravovat. Proto musí systém rozlišit různé uživatelské role. Systém ve své první verzi uvažuje pouze rozlišení běžného uživatele (nepřihlášeného) a administrátora. Pro implementaci této funkcionality jsem identifikoval nutnost dvou tabulek: c_users a c_user_type. Graf relačního modelu autentizace je zachycen na obr. 4.4. Podrobný popis tabulek a



Obrázek 4.4: Relační model grafů

Název sloupce	Datový typ	Popis sloupce
id	int	Primární klíč záznamu
login	Varchar(200)	Přihlašovací jméno uživatele
name	Varchar(200)	Jméno uživatele
surname	Varchar(200)	Příjmení uživatele
password	Varchar(200)	Zahešované heslo uživatele metodou SHA-512
type_id	int	Jedná se o cizí klíč do tabulky c_user_typ, určuje typ uživatelského účtu.

Tabulka 4.5: Popis sloupců tabulky c_users

jejich sloupců se nachází v následujících kapitolách.

4.6.3.1 c_users

Tabulka obsahuje seznam uživatelů. V tabulce jsou uloženy jejich údaje a zahešované heslo uživatele. Podrobnější popis sloupců poskytuje tabulka 4.5.

4.6.3.2 c_user_type

Tabulka eviduje typy uživatelského účtu. Její sloupce jsou popsané v tabulce 4.6.

Název sloupce	Datový typ	Popis sloupce
type_id	int	Primární klíč záznamu
type_name	Varchar(100)	Název oprávnění, například Administrátor.

Tabulka 4.6: Popis sloupců tabulky c_user_type

Kapitola 5

Návrh uživatelského rozhraní aplikace

Jelikož výuková aplikace „Aplikace PERT“ nenavazuje na žádný stávající systém a do budoucna ani není uvažována integrace na žádný již existující informační systém, mám v rámci návrhu uživatelského rozhraní volnou ruku. Rozhodl jsem se pro minimalistický styl, který bude nabízet intuitivní a uživatelsky přívětivé rozhraní systému. Návrh a tvorbu uživatelského rozhraní popisuji v dalších částech této kapitoly.

Na základě požadavků kladených na aplikaci, jsem si nejdříve určil, jaké obrazovky jsou potřeba pro jejich splnění. Následně jsem si vytvořil graf přechodů mezi obrazovky. Po vytvoření této kostry obrazovek, jsem začal pracovat na samotných obrazovkách a rozpracoval jejich náhledy. Systém je rozdělený tak, aby jednotlivé obrazovky působily přehledně a nenabízely zbytečnou funkcionalitu.

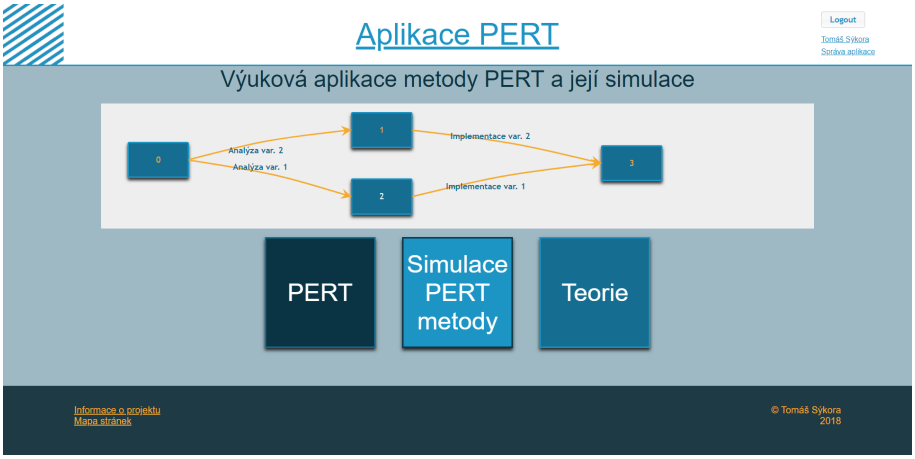
V rámci tvorby GUI jsem musel zohlednit, že systém je určený pro dvě uživatelské role, administrátora a běžného uživatele. Vzhledem k těmto faktům jsou další sekce této kapitoly členěny podle těchto rolí.

5.1 Uživatelská část

V této kapitole jsou popsány obrazovky, které slouží pro obě uživatelské role systému. Komponenty na těchto obrazovkách jsou pro obě role shodné, proto je popisuji společně. V kapitole jsem pro popis každé obrazovky zvolil shodnou strukturu, nejdříve jsou obrazovky textově popsány a následně je v kapitole její náhled.

5.1.1 Úvodní rozcestník

Stránka slouží jako úvodní obrazovka aplikace. Z této obrazovky jsou dostupné všechny akce a funkcionality. Pro běžného uživatele se v této verzi aplikace



Obrázek 5.1: Náhled GUI - Úvodní obrazovky

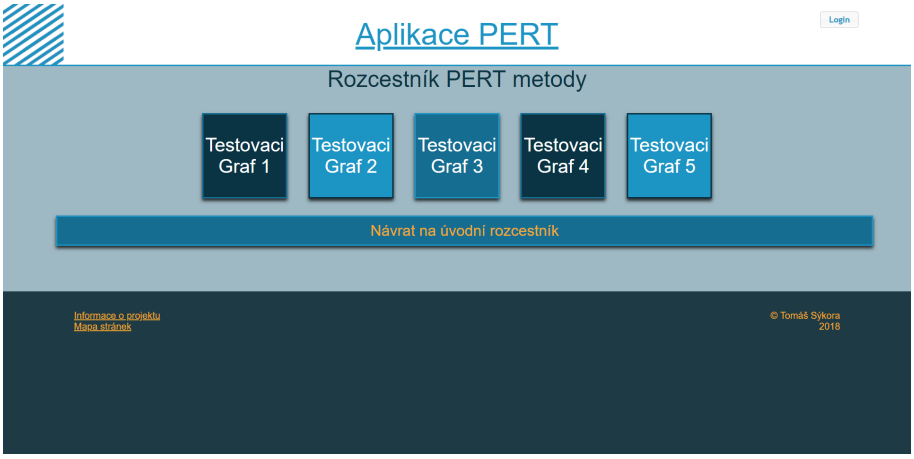
Obrázek 5.2: Náhled GUI - Teorie

jedná o přístup k teorii, výpočtu PERT metody a její simulaci.

U administrátorské role je k dispozici ještě nastavení profilu a přístup k administraci aplikace (správa projektů, teorie a uživatelů). Náhled této obrazovky je možné vidět na obr. 5.1.

5.1.2 Teorie

Na obrazovce se zobrazuje kompletní teorie v systému. Teorie je strukturována do jednotlivých stránek. Návrh obrazovky je provedený tak, že zobrazena je vždy jediná stránka teorie. Mezi jednotlivými stránkami lze přepínat na hlavním panelu obrazovky. Stránky se zde zobrazují jako jednotlivé záložky. Náhled obrazovky je možné najít na obr. 5.2.



Obrázek 5.3: Náhled GUI - Rozcestník

5.1.3 Rozcestník pro výpočet PERT a simulaci

Obrazovka slouží jako propojení mezi úvodním rozcestníkem a obrazovkou pro samotný výpočet. Na této obrazovce systém zobrazí dostupné projekty a uživatel si pro přechod na detail vybere z nabízených projektů. Projekty jsou znázorněny formou dlaždic. Ty jsou zobrazeny ve stejném stylu jako dlaždice na úvodním rozcestníku. Náhled této obrazovky je vidět na obr. 5.3.

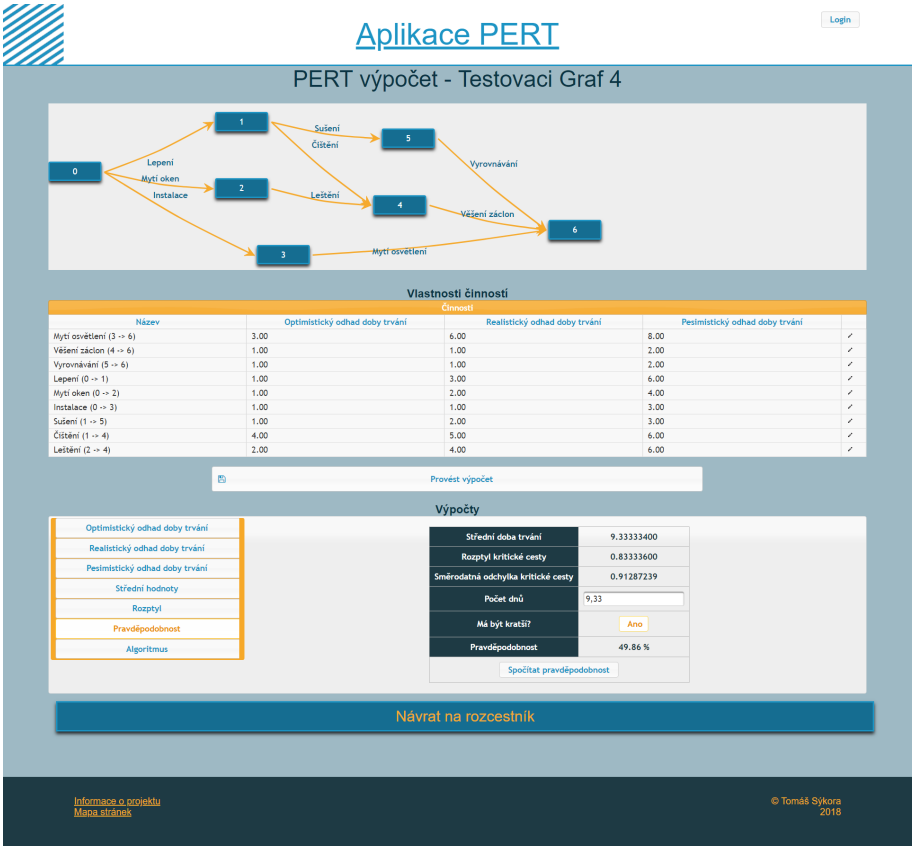
5.1.4 PERT - výpočet

Cílem obrazovky je znázornit výpočty PERT metody. Obrazovka se skládá ze tři rozdílných sekcí: graf projektu, data o činnostech a sekce výpočtů. Běžný uživatel smí pouze editovat data činností. V systému není možnost tyto úpravy uložit. Dále je pro běžného uživatele možné provést výpočet PERT metody. Po provedení výpočtu dojde k zobrazení dat v sekci výpočtů. Na základě vypočtených dat je poté možné provádět výpočet pravděpodobnosti pro různé délky trvání projektu. Obrazovku si je možné prohlédnout na obr. 5.4

5.1.5 PERT - simulace

Obrazovka obsahuje velmi podobné rozložení jako obrazovka pro výpočty metodou PERT. Tedy obrazovka znovu obsahuje tři části: graf projektu, data o činnostech a sekci simulací. Jelikož se jedná o simulaci, obsahuje obrazovka také další vstupní element pro počet kroků simulace.

V sekci simulace se poté nachází nasimulované hodnoty. Na základě těchto hodnot jsou určené četnosti jednotlivých cest a statistické hodnoty. Nad nimi je poté možné provést výpočet pravděpodobnosti. Ten probíhá nad kolekcí nejdéle trvajících cest v rámci jednotlivých kol simulace.



Obrázek 5.4: Náhled GUI - PERT

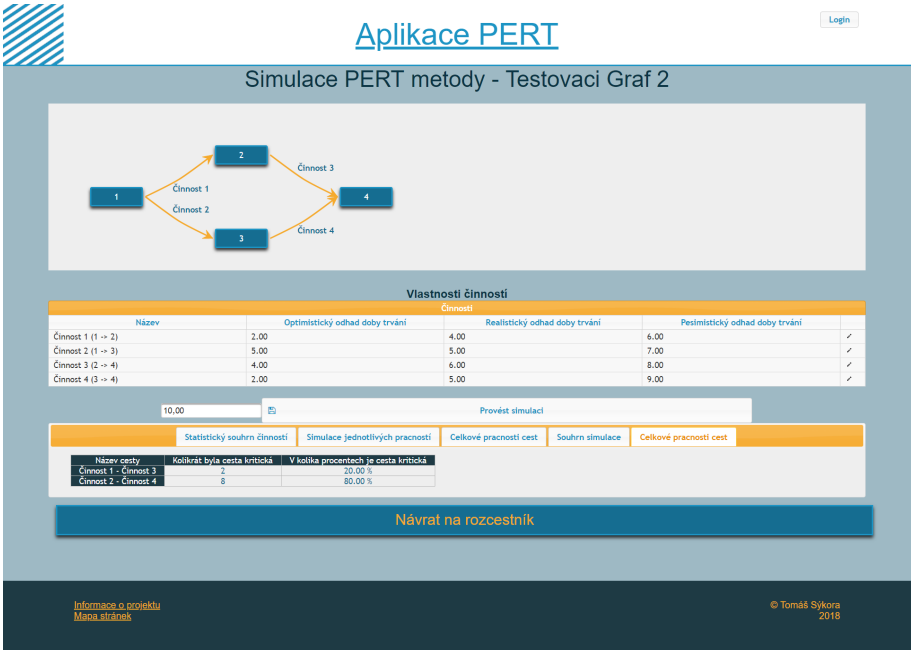
V první verzi systému zatím není pro běžné uživatele uvažováno ukládání změn, všechny provedené změny jsou prováděny v rámci obrazovky. Obrazovka je zachycena na obr. 5.5.

5.2 Administrátorská část

V další části této kapitoly popisuji obrazovky, které již nejsou pro běžného uživatele dostupné. Jsou zde popsány obrazovky pro správu informačního systému. Tedy ty části, jež umožní editovat připravené grafy a obsah teorie. Také sem spadá správa uživatelů.

5.2.1 Přihlašovací obrazovka

Obrazovka slouží pouze k přihlášení. V první verzi aplikace systém nabízí pouze jeden typ účtu, administrátorský. Stránka neobsahuje žádnou funkcionalitu jako je resetování hesla. Postup obnovení hesla je popsán v instalační



Obrázek 5.5: Náhled GUI - Simulace PERT metody

Obrázek 5.6: Náhled GUI - Přihlášení

příručce a je ho nutné provést přímo v databázi. Přihlašovací obrazovka je vidět na obr. 5.6.

5.2.2 Editor teorie

Editace textu probíhá na samostatné obrazovce. V té je integrován textový editor se základními funkcemi pro strukturování textu. Na obrazovce si také může uživatel zobrazit náhled upravovaného textu a následně ho uložit. Ob-

Obrázek 5.7: Náhled GUI - Editace teorie

razovka je zobrazena na obr. 5.7.

5.2.3 Úprava projektu

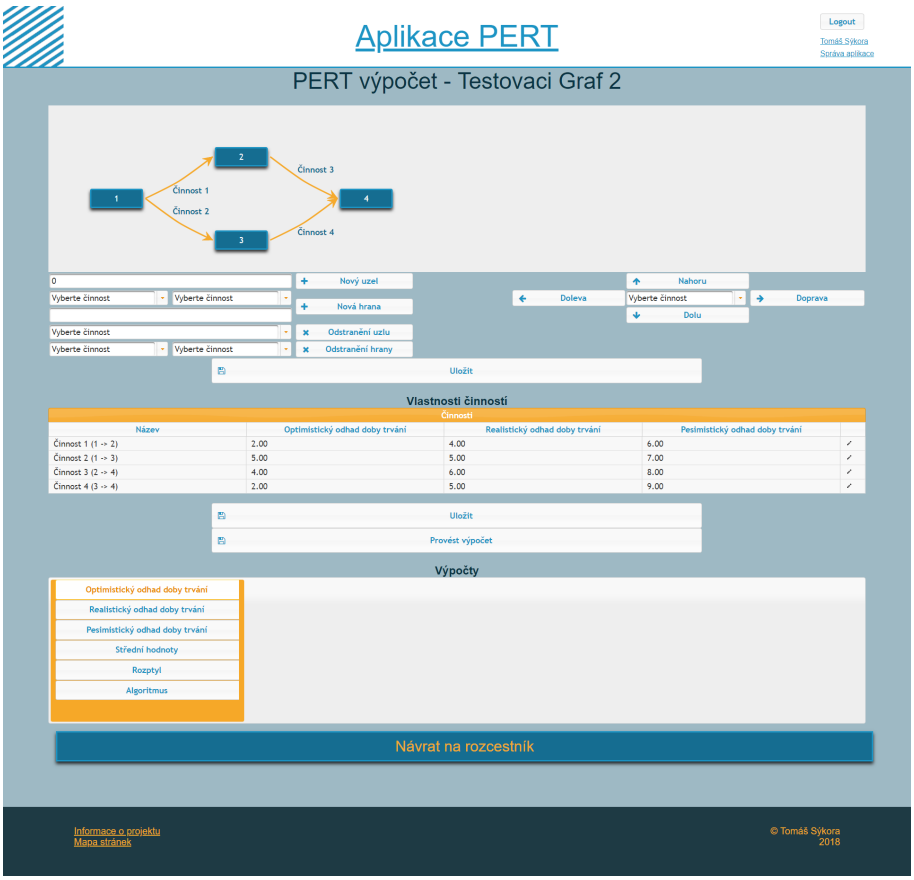
Obrazovka určená pro úpravu klíčových dat aplikace, tj. projektů. Pro jednoduchý a intuitivní přístup k editaci projektů, jsem se rozhodl využít stejné obrazovky jako mají dostupné běžní uživatelé. V případě administrátory systému se na obrazovce ještě nachází ovládací panely. Ty slouží k manipulaci se samotným grafem projektu a jeho daty. Na obrazovce se také nachází tlačítko uložit, které perzistentně uloží provedené změny do databáze. Náhled obrazovky se nachází na obr. 5.8.

5.2.4 Správa aplikace

V rámci systému bylo nutné vyřešit, jak přidávat a odstraňovat nová data. Byla zde možnost znovu vytvořit ovládací panely a ty zobrazovat v rámci příslušných obrazovek. Toto řešení mi přišlo nevhodné, proto jsem se rozhodl vytvořit souhrnnou obrazovku. Ta nabízí správu projektů, teorie i uživatelů. V rámci obrazovky je možné vytvářet a mazat tato data. Obrazovka je zachycena na obr. 5.9.

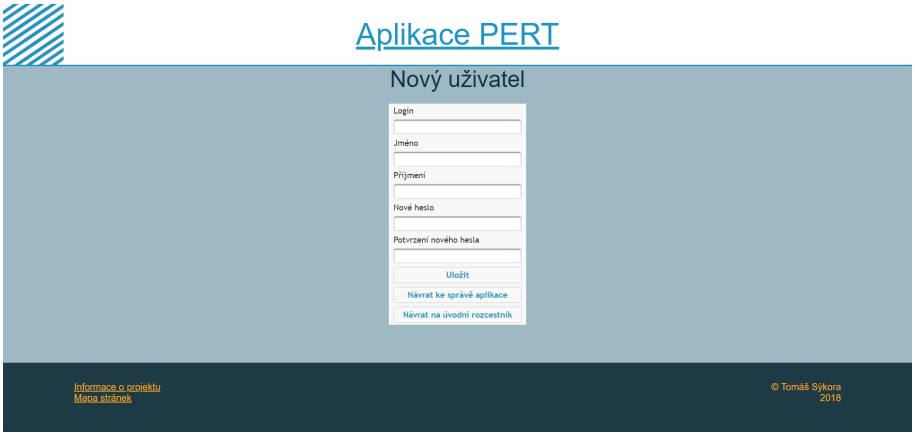
5.2.5 Vytvoření uživatele

Na obrazovku se lze dostat ze správy aplikace. Obrazovka obsahuje jednoduchý formulář pro vyplnění údajů o uživateli. Mezi data patří jméno, příjmení, login a heslo uživatele. Typ nového účtu je vždy nastaven jako administrátor systému. Obrazovku si je možné prohlédnout na obr. 5.10.

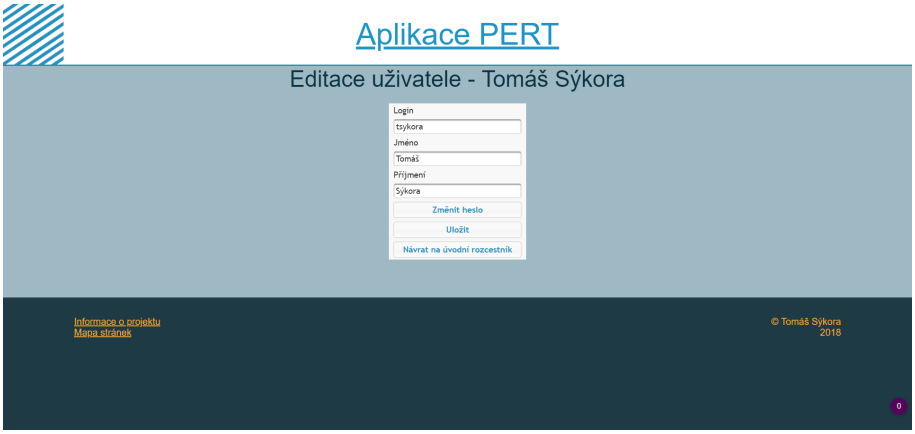


Obrázek 5.8: Náhled GUI - Úvodní obrazovky

Obrázek 5.9: Náhled GUI - Správa aplikace



Obrázek 5.10: Náhled GUI - Nový uživatel



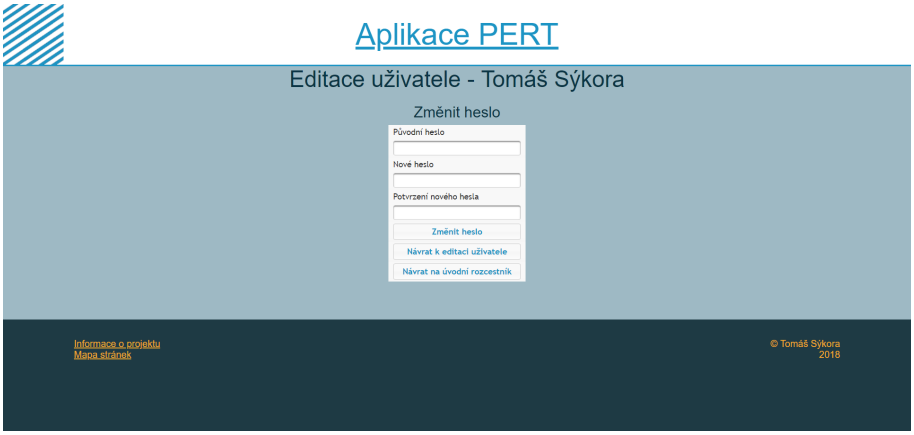
Obrázek 5.11: Náhled GUI - Editace uživatele

5.2.6 Editace uživatele

Přihlášený uživatel si zde může upravit své údaje a případně přejít na další obrazovku pro změnu hesla. Obrazovka pro editaci uživatele je dostupná z hlavičky aplikace. Náhled obrazovky se nachází na obr. 5.11.

5.2.7 Změna hesla

Na popisovanou obrazovku je možné přejít z editace uživatele. Obrazovka obsahuje formulář se třemi textovými vstupy pro aktuální heslo, nové heslo a jeho potvrzení. Obrazovka se nachází na obr. 5.12.



Obrázek 5.12: Náhled GUI - Změna hesla

Kapitola 6

Implementace

Zatím byla v práci popisována pouze analýza a návrhy GUI aplikace. Náplní této práce ale není pouze zpracování analýzy systému, její součástí je i implementace. Ta je popsána v této kapitole.

Nejdříve zde uvedu použité technologie a důvody proč jsem se rozhodl zvolit v dané doméně zrovna tuto variantu. Tedy jedná se částečně ještě o analytickou kapitolu, ale analýzu již tak úzce související se samotnou implementací, že si dovolím tuto část popsat až zde.

Poté již následuje samotný design aplikace a ukázka zajímavých nebo důležitých fragmentů kódu.

Jedním z nefunkčních požadavků je podmínka na programovací jazyk Java. Tedy následující diskuze se vztahují k frameworkům a knihovnám tohoto jazyka.

6.1 Použité technologie

Jak jsem již uvedl v úvodu této kapitoly, tak zde jsou uvedeny použité technologie a důvody, proč jsem se rozhodl vybudovat systém kolem nich. Pokud jsem pro danou oblast uvažoval o použití i jiné technologie, tak zde samozřejmě uvádím i důvody, proč jsem shledal tuto technologii pro informační systém nevhodnou.

6.1.1 Databáze

Jako každý systém i „Aplikace PERT“ pracuje s daty. Proto v této kapitole uvádím technologie, které slouží pro samotné vytvoření databáze a následně pro perzistentní uložení dat. Do této kapitoly jsem také zahrnul nástroj pro objektově relační mapování. Ten by spíše patřil přímo do technologií backendu aplikace, ale vzhledem k souvislosti s databází si dovolím uvést tuto technologii v této kapitole.

6.1.1.1 Způsob uložení dat

Na začátku analýzy jsem zvažoval tři varianty:

- Staticky uložená data v aplikační logice - tedy data přímo ve třídách v Javě,
- memory databázi typu H2 nebo Derby,
- plnohodnotný databázový server.

Ke zvolení technologie pro uložení dat jsem nejdříve potřeboval vyřešit objem dat a zda při provozu nebude docházet k úpravám konfigurace. Tato analytická část je již popsána v kapitole 4.6. Z analýzy v referované kapitole vyplynula nutnost použít databázový server. V tomto případě je technologie serveru určena nefunkčním požadavkem a aplikace je postavena na MySQL databázi.

Jen připomenu, že staticky uložená data nejsou reálná, protože při používání systému mohou nastávat změny v konfiguraci (přidání nového grafu, úprava teorie). Vestavěná databáze zase není vhodná proto, že systém má zároveň sloužit více uživatelům a na to memory databáze není vhodná. Stejně tak není ideální pro další vývoj systému a úpravy datového modelu.

6.1.1.2 Verzování datového modelu

Po volbě plnohodnotného SQL serveru jsem musel řešit, jak vytvářet a případně verzovat datový model databáze. Tento problém se řeší na mnoha projektech a často dokáže způsobit nemalé problémy. Zde jsem se stejně jako u volby samotné perzistence dat rozhodoval mezi třemi možnostmi.

Jako první z možností jsem zvažoval využít pokročilé možnosti ORM nástroje a nechat si vygenerovat model databáze na základě datových tříd v Javě. Tato možnost se mi nelíbila proto, že v tu chvíli databáze funguje formou černé krabičky, jejíž správu zajišťuje zvolený ORM nástroj. Touto volbou se tedy můžu připravit o kontrolu nad databází a nemám možnost přímo určovat cizí klíče a indexy tabulek.

Další možností je napsat si SQL skripty pro vytvoření databáze a ty následně verzovat. Zde si myslím, že se může jednat o ideální řešení, pokud je možné vyloučit další budoucí vývoj. Ale sám v kapitole 10 navrhuji další možná vylepšení systému. Na základě těchto informací není možné vyloučit další vývoj systému.

Změny v databázi plynoucí z navazujícího vývoje, je možné řešit dalšími SQL skripty. Zde však může nastat problém, pokud se na vytvořený systém bude neustále navazovat. Nově příchozí vývojář je nucený identifikovat všechny potřebné skripty upravující datový model. Tyto skripty poté musí spustit přesně v pořadí jak vznikaly. Tento postup nepovažuji za ideální a je silně vázaný na kvalitní dokumentaci postupu nasazování.

Do této chvíle jsem v kapitole uváděl pouze postupy, ve kterých jsem vždy našel nedostatky. Ty více či méně vylučovaly použití pro vývoj „Aplikace PERT“. Pro eliminaci zatím uvedených problémů jsem se rozhodl využít open source knihovnu Liquibase. Možnosti této knihovny podrobně popisuji v kapitole 2.6.2.

Pro zachycení jednotlivých změn datového modelu, jsem se rozhodl použít doporučené best practices Liquibase [39]. Ty spočívají v tom, že jednotlivé change logy jsou uspořádané podle vývoje (například podle verzí aplikace) a propojeny jediným dokumentem. Ten obsahuje reference na samostatné change logy.

Využitím Liquibase jsem vyřešil problém černé krabičky, který by nastal při automatickém generování datového modelu přes ORM nástroj. Také se tím řeší problém roztříštěnosti skriptů a nutnosti evidovat jejich pořadí. To je poté evidováno ve strukturovaném formátu. Ten zároveň obsahuje úpravy datového modelu.

Jako formát pro uložení změn jsem se rozhodl použít XML. Jedná se však pouze o osobní preference, protože všechny formáty podporované Liquibase nabízí stejné možnosti databázových úprav.

Ukázkovou (testovací) konfiguraci tabulek jsem se rozhodl držet v SQL skriptech. To zejména proto, že tato data slouží primárně k testování. Při používání systému vzniknou produkční data, která se budou pravidelně upravovat v aplikaci. Data v SQL skriptech je nutné udržet konzistentní vůči datovému modelu. Poté je možné získat poslední verzi databáze jediným příkazem. Datový model je vygenerován přes Liquibase a data obsahuje SQL skript.

6.1.1.3 ORM Nástroj

Zatím jsou v analýze řešeny pouze technologie pro uložení dat v databázi a způsob tvorby datového modelu. Neméně důležitá je však komunikace mezi databází a samotnou aplikací. Pro tu jsem zvažoval použít dvě varianty, přímé volání SQL a různé alternace nebo ORM nástroj.

Do alternací SQL volání řadím i různé formy JDBC šablon, které nabízí framework Spring, nebo nepřímé volání formou uložených procedur. Tento přístup však uvádím pouze jako jednu z variant, která patří spíše do minulosti a nepatří již do návrhu moderních systémů. Zejména kvůli spoustě nevýhod, jež tento postup datové integrace přináší. Mezi tyto nevýhody například patří:

- Duplikace informací o datovém modelu - SQL procedury a Java třídy.
- SQL fragmenty přímo ve zdrojových kódech Javy.

Obě uvedené nevýhody sdílejí to, že ztěžují úpravy datového modelu. Vždy je nutné provést změnu na dvou místech, jak přímo v SQL tak také v třídách datového modelu v Javě.

Vzhledem k výše popsaným problémům, jsem od začátku uvažoval jen nad využitím ORM. Jak jsem již uvedl v kapitole 6.1.1.2, nástroj používám pouze pro datové propojení nikoliv pro vytvoření datového modelu databáze. Pro zajištění objektově relačního mapování jsem uvažoval nad dvěma frameworky, MyBatis a Hibernate.

Nejdříve jsem nechtěl využít Hibernate a plánoval použít výrazně menší MyBatis. Zde se mi však příliš nelíbilo, že stále je nutné držet SQL šablony v mapperech knihovny. Tedy stále jsou informace o datovém modelu evidovány na více místech. Mapper je navíc možné konfigurovat pouze přes XML. To jsem se snažil v aplikaci omezovat a provádět konfiguraci v rámci Javy, případně konfiguračních properties souborů. Dalším jazýčkem na vahách pro mě představuje i to, že pokud ve fázi implementace narazím na funkcionalitu, jenž nebude MyBatis obsahovat, může pro mě být migrace na Hibernate náročnější než daný framework použít na začátku projektu. Další důvod pro mě představovala i podrobnější a kvalitnější dokumentace pro spojení Spring a Hibernate než integraci mezi Springem a ORM nástrojem MyBatis.

Na základě předchozích odstavců textu je celkem jasné, že jsem jako ORM nástroj využil Hibernate. Vzhledem k tomu, že jsem se snažil omezovat XML konfiguraci, využil jsem pro konfiguraci mapování anotace JPA přímo v Java třídách. Anotace zde blíže nepopisuji, protože jejich popis se již nachází v kapitole 2.5.2. Ukázku kódu s využitím anotací je možné najít níže. Upozorňuji však, že se jedná pouze o ilustrativní ukázku, která neobsahuje všechny proměnné, importy a metody dané třídy.

```
@Entity
@Table(name = "c_project_node_graph")
public class CProjectNodeGraph {

   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   @Column(name = "graph_node_id")
   private Integer graphNodeId;
   @ManyToOne(fetch = FetchType.EAGER)
   @JoinColumn(name = "current_node_id", referencedColumnName =
       "node_id")
   private CNode node;

   public Integer getGraphNodeId() {
      return graphNodeId;
   }

   public void setGraphNodeId(Integer graphNodeId) {
      this.graphNodeId = graphNodeId;
   }

   public CNode getNode() {
      return node;
```

```
}

public void setNode(CNode node) {
   if (!node.equals(this.node)) {
      node.getGraphs().remove(this);
      this.node = node;
      node.getGraphs().add(this);
   }
}
```

Po integraci ORM nástroje a anotaci třídních proměnných v Java třídách datového modelu. Jsem ještě řešil samotné propojení s databází. Zde jsem se znovu výrazně opřel o možnosti frameworku Spring. Ten nabízí dostatečnou podporu pro databáze. Stačí pouze nakonfigurovat parametry uvedené v následujícím seznamu a poté se již stará o připojení do databáze samotný framework. Seznam parametrů a jejich popis:

spring.datasource.url
: URL adresa běžící databáze.

spring.datasource.driver-class-name
: Název driveru pro připojení do databáze.

spring.datasource.username
: Uživatelské jméno, které bude použito pro přihlášení do databáze.

spring.datasource.password
: Heslo, které bude využíváno pro přihlášení do databáze.

spring.jpa.properties.hibernate.dialect
: Dialekt SQL, který je používán pro komunikaci s DB.

S využitím JPA anotací získám výhodu, že pro samotný přenos dat mohu využít metody nabízené frameworkem Spring. Ty jsou založeny na rozhraní pro přenos dat JpaRepository<T, ID extends Serializable>. Prvním parametrem je ukládaný objekt, druhý parametr představuje datový typ ID daného objektu. Při využití tohoto konceptu mám k dispozici metody pro manipulaci s daty, například pro:

- Vytvoření objektu,
- modifikaci objektu,
- mazání dat,
- Vyhledání dat podle ID.

Zde se jedná o základní dotazy. Spring však nabízí mnoho možností, jak vytvářet metody pro tvorbu SQL dotazů. Framework vytváří SQL dotazy podle názvu metody v dané třídě. Třída musí splňovat podmínku, že dědí od uvedeného rozhraní JpaRepository<T, ID extends Serializable>. Například metoda findByProjectId(Integer projectId) poté povede na dotaz, který obsahuje where klauzuli pro id projektu. Daný koncept podrobně popisuji v sekci teorie 2.5.3.2.

6.1.2 Backend aplikace

Při popisování ORM technologie, jsem již uvedl rozhraní JpaRepository. To poskytuje framework Spring. V uvedené kapitole je framework zmíněn pouze jako prostředek pro přístup k datům. Spring však představuje v systému základní komponentu. Ta funguje jako integrační prvek všech ostatních technologií. Framework zajistí v aplikaci propojení jednotlivých vrstev aplikace, prezentační, aplikační a datovou.

Framework taky zajišťuje zpracovávání HTTP požadavků. Pro tuto oblast jsem nejdříve plánoval použít nástroje Rewrite od společnosti OCPsoft. Ten nabízí mapování URL adres na JSF šablony. URL poté neobsahuje název šablony ale pouze mapovací adresu. Ta se na tuto šablonu mapuje v Javě. Při použití nástroje Rewrite je problematické předávání parametrů v URL. Tento problém je poměrně složitý a proto nakonec používám řešení přes framework Spring, který také nabízí URL mapování.

Tato kapitola je poměrně krátká, ale rozhodně to není proto, že jsem podcenil analýzu kolem backendu aplikace. Dané je způsobeno tím, že datovou integraci jsem popsal v kapitole 6.1.1.3 a zpracování HTTP požadavku a design aplikace je popsán v kapitole 6.2.

6.1.3 Frontend aplikace

Vzhledem k tomu, že se blížím ke konci kapitoly implementace, dostávám se již k prezentační vrstvě aplikace. Zde jsem od začátku uvažoval nad následujícími frameworky:

- Vaadin
- Thymeleaf
- Primefaces

Využití Vaadinu jsem považoval od začátku za příležitost, naučit se novou a moderní technologii. Po analýze a začátku implementace, jsem však měl o této variantě pochybnosti. K nim mě zejména vedlo, že pro vyvíjený systém této velikosti, jsem Vaadin považoval za zbytečně obrovský framework. Z něj využiji jen zlomek jeho možností. Tyto pochybnosti vedly k tomu, že jsem se začal ohlížet po alternativách.

Jako další variantu jsem chtěl využít Thymeleaf. Tedy už se nejedná o framework ale lehký šablonový systém. Navíc Spring obsahuje přímou integraci na Thymeleaf. Dokonce v publikaci Spring in Action [19] je to doporučovaná kombinace. Zde však nejsem spokojený s tím, že se jedná o nástroj bez jakékoliv nadstavby a tedy všechny složitější webové prvky je nutné napsat a nastylovat.

Zatím jsem pro vytvoření uživatelského rozhraní uvedl dva nástroje, U obou jsem nalezl určité nevýhody a výhody. Pokud spojím jejich výhody dohromady, dostanu se ke třetí možnosti, Primefaces. Jedná se o relativně menší framework (v porovnání s Vaadinem). Ten však nabízí velké množství připravených grafických komponent. Také je možné využít kolem čtyřiceti různých grafických vzhledů. Další obrovskou výhodu pro mě představuje výrazná podpora AJAXu, kdy framework s tímto řešením počítá a nabízí spoustu variant jak AJAX do webové aplikace integrovat.

6.2 Design aplikace - zpracování příchozího požadavku

V této kapitole bylo zatím řešeno jaké technologie použiji a důvody, proč jsem se je rozhodl využít. Zde popisuji samostatný design aplikace.

Jak je již z uvedených informací v této práci patrné, popisovaný systém představuje webovou aplikaci. Samotná architektura je dělena standardně do tří vrstev, prezentační, aplikační a datové. Nejedná se však o běžně uváděnou strukturu postupně na sebe navazujících vrstev. To souvisí s použitím frameworku Spring, který obstarává zpracování příchozích požadavků a vytvoření odpovědi pro obdržený požadavek.

Ve frameworku je zpracování požadavku implementováno přidáním dvou vrstev. Celá architektura zpracování je řízena „Dispatcher Servletem“. Jedná se o interní třídu Springu a také o jediný komunikační bod mezi klientem (webový prohlížeč) a aplikací. Třída zařídí zpracování příchozího requestu a navazující logiky, následně po získání potřebných dat, odešle klientovi odpověď. V aplikaci používám defaultní konfiguraci třídy, tedy není nutné nic konfigurovat.

Po obdržení požadavku nalezne „Dispatcher Servlet“ přes třídu Springu „Handler Mapping“ handler pro zpracování dat na základě zadané URL adresy. K dohledání dochází dle Springovských anotací u daných metod případně tříd. Níže uvádím ukázkový kód pro zpracování požadavku.

```
@Controller
public class PertDetailController extends AbstractGraphController {

   @RequestMapping(value = "pert/{graphId}")
   public String load(@PathVariable Integer graphId) {
      init();
```

```
        loadGraphData(graphId);
        GraphService.createNodes(nodes.values(), model);
        GraphService.createConnections(connections.values(), model);
        return "pertdetail";
    }
}
```

Pokud handler pro danou URL adresu neexistuje, dojde automaticky k přesměrování na chybovou stránku. Jestli je nalezen patřičný handler, je na něj přeposlán obdržený request. V rámci handleru dojde poté k aplikování business logiky a případně zavolání DAO vrstvy, pokud je nutné získat data z databáze. „Dispatcher Servlet“ poté od handleru obdrží model s daty a název šablony pro zobrazení. Název se poté předá do „View Resolveru“. Ten na základě konfigurace frameworku obalí název prefixem a sufixem. Z toho poté vznikne adresa šablony, kterou obdrží zpět „Dispatcher Servlet“. Níže je ukázka konfigurace „View Resolveru“.

```
@Configuration
public class ServletInitializer extends SpringBootServletInitializer {

    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver resolver = new
            InternalResourceViewResolver();
        resolver.setPrefix("/template/");
        resolver.setSuffix(".xhtml");
        resolver.setExposeContextBeansAsAttributes(true);

        return resolver;
    }
}
```

Dispatcher poté do získané šablony zašle data obdržená z handleru. V odpovědi poté obdrží data pro klienta. Ty poté pouze přepošle klientovi, který se již postará o vykreslení webové stránky.

Kapitola 7

Testování

Po implementaci se již logicky dostávám k testování jako následují fázi vývoje softwaru. Provedené testování lze rozdělit do dvou odlišných částí.

První z těchto fází má nejblíže k uživatelskému testování. Nejedná se však o klasické uživatelské testování v laboratoři použitelnosti. Testování většinou probíhalo přímo na strojích testerů a ani nebylo vždy moderováno. Tímto postupem jsem se snažil odladit nedostatky aplikace. Uživatelé prováděli v systému různé posloupnosti kroků. Ty vycházely ze scénářů použití. Tato část testování je dále popsána v kapitolách 7.3 a 7.4.

Druhá fáze testování představovala ověření funkčnosti aplikace. To proběhlo úspěšnou exekucí testovací scénářů, ty jsou popsány v kapitole 7.2. Toto testování nejvíce odpovídá klasifikačním testům, jenž jsou provedeny před dodáním aplikace zákazníkovi.

7.1 Automatické testy

V rámci aplikace došlo k implementaci nejen kódu s aplikační logikou, ale také k vytvoření jednotkových testů, které tuto logiku pokrývají. Samozřejmě tyto testy testují, že logika je správně implementována. Mají však také význam regresních testů. Pokud dojde k dalším úpravám systému, můžou v rámci implementace vzniknout regresní chyby. Těm nejde při vývoji zabránit, ale při pravidelném spouštění testů je možné tyto chyby brzo indikovat a okamžitě opravit.

7.2 Testovací scénáře

Jak jsem již uvedl na začátku této kapitoly, zde se nachází testovací scénáře pro aplikaci. Scénáře vychází z případů užití a pokrývají všechny funkční požadavky. Scénáře jsou psány formou kroků, které tester musí provést. Ke kaž-

dému kroku poté existuje očekávaný stav systému po jeho vykonání. Všechny scénáře, kromě prvních tří, mají prerekvizitu ve formě přihlášeného uživatele.

7.2.1 Výpočet PERT metody a pravděpodobnosti

Scénář k otestování PERT výpočtů a výpočtu pravděpodobnosti na vypočtených datech.

Krok k exekuci	Očekávaný stav systému
Přejít na rozcestník PERT.	Nacházím se na rozcestníku PERT.
Přejít na konkrétní projekt.	Nacházím se na projektu a nejsou zobrazeny ovládací panely.
Provést výpočet.	Výpočet se provedl.
Změnit parametry činností.	Parametry lze změnit.
Provést výpočet.	Výpočet se aktualizoval na základě změněných dat.
V části výpočtů přejít do sekce pravděpodobnosti.	Sekce pravděpodobnosti se zobrazila.
Vyplnit počet dnů a provést výpočet pravděpodobnosti.	Výpočet se provedl.
Otočit podmínku, zda je délka trvání kratší.	Podmínku lze otočit.
Provést výpočet pravděpodobnosti.	Výpočet se provedl a součet pravděpodobností se rovná 100 % s odchylkou setiny procenta.

Tabulka 7.1: TC1 - Výpočet PERT a pravděpodobnosti

7.2.2 Simulace PERT metody a výpočet pravděpodobnosti

Slouží k otestování simulace PERT metody a výpočtu pravděpodobnosti na nasimulovaných datech.

Krok k exekuci	Očekávaný stav systému
Přejít na rozcestník simulací.	Nacházím se na rozcestníku simulací.
Přejít na konkrétní projekt.	Nacházím se na projektu a nejsou zobrazeny ovládací panely.
Provést simulaci.	Simulace se provedla.
Změnit parametry činností.	Parametry lze změnit.
Provést simulaci.	Simulace se aktualizovala na základě změněných dat.
Změnit počet kroků simulace a provést znovu simulaci.	Parametry lze změnit a simulace je provedena s novým počtem kroků.

V části simulace přejít na záložku pravděpodobnosti.	Sekce pravděpodobnosti se zobrazila.
Provést výpočet pravděpodobnosti.	Výpočet se provedl.
Otočit podmínku, zda je délka trvání kratší.	Podmínku lze otočit.
Provést výpočet pravděpodobnosti.	Výpočet se provedl a součet pravděpodobností se rovná 100 % s odchylkou setiny procenta.

Tabulka 7.2: TC2 - Simulace PERT metody a výpočet pravděpodobnosti

7.2.3 Zobrazení stránky teorie

Scénář slouží k otestování zobrazení teorie a přepínání jednotlivých stránek na této obrazovce.

Krok k exekuci	Očekávaný stav systému
Přejít na stránku teorie.	Nacházím se na stránce teorie a je zobrazena první záložka.
Přepnout na jinou záložku (stránku).	Mezi záložky lze přepínat a zobrazí se nový text.
Vrátit se na původní záložku.	Zobrazil se původní text.

Tabulka 7.3: TC3 - Zobrazení stránky teorie

7.2.4 Přidání nového projektu

Cílem je otestovat vytvoření nového projektu a nastavení jeho dat (graf a doba trvání činností).

Krok k exekuci	Očekávaný stav systému
Přejít do správy aplikace.	Nacházím se ve správě aplikace.
Zkusím přidat nový projekt kliknutím na tlačítko „Nový projekt“.	Projekt se nepodaří vytvořit a je vrácena chyba „Název projektu je prázdný“.
Vyplnit název projektu a přidat projekt.	Projekt byl vytvořen.
Přejít zpátky na úvodní rozcestník.	Nacházím se na úvodním rozcestníku.
Přejít na rozcestník PERT metody.	Nacházím se na rozcestníku PERT metody a existuje zde nový projekt.

Přejít na nově vytvořený projekt.	Jsem na obrazovce pro detail projektu.
Vytvořit graf projektu.	Graf lze vytvořit.
Uložit graf.	Graf se uložil a je vyplněna tabulka s činnostmi, data odpovídají grafu.
Vyplnit data o činnostech a uložit.	Data lze vyplnit a v pořádku se uložila.
Provést výpočet.	Výpočet se provedl.

Tabulka 7.4: TC4 - Přidání nového projektu

7.2.5 Editace projektu - nevalidní graf

Scénář testuje, že nejde uložit nevalidní graf. Za nevalidní graf se považuje takový, který má více než jeden uzel bez vstupní hrany. Druhou možností je existence více uzlů bez odchozí hrany. Poslední z možností nastává, pokud graf obsahuje uzel bez hran.

Krok k exekuci	Očekávaný stav systému
Přes rozcestník PERT přejít na detail některého projektu.	Jsem na detailu projektu.
Provést editaci grafu, tak že existují dva uzly bez vstupní hrany a zkusit tento graf uložit.	Graf se nepodaří uložit a je vrácena chyba „Na základě grafu není možné rozhodnout o počátečním uzlu“.
Provést editaci grafu, tak že existují dva uzly bez odchozí hrany a zkusit tento graf uložit.	Graf se nepodaří uložit a je vrácena chyba „Na základě grafu není možné rozhodnout o koncovém uzlu“.
Převést graf na validní a zkusit uložit.	Graf se podařilo uložit.
Přidat uzel nepropojený se zbytkem grafu.	Graf se nepodaří uložit a je vrácena chyba „Existuje uzel bez hrany“.
Odstranit problematický uzel a uložit graf.	Uzel šlo odstranit a graf se uložil.

Tabulka 7.5: TC5 - Editace projektu - nevalidní graf

7.2.6 Editace projektu - nepovolené operace

Tento testovací scénář testuje blokování nepovolených operací již na úrovni editace grafu.

Krok k exekuci	Očekávaný stav systému
Přes rozcestník PERT přejít na detail některého projektu.	Jsem na detailu projektu.
Pokusit se vytvořit činnost bez jména.	Hrana nepůjde vytvořit a dojde k zobrazení chybové hlášky „Jméno činnosti je prázdné“.
Vyplnit jméno a nevybrat okrajové body.	Hrana nepůjde vytvořit a dojde k zobrazení chybové hlášky „Neexistují krajní elementy“.
Pokusit se přidat hranu do uzlu z nižším číslem než je uzel zdrojový.	Hrana nepůjde vytvořit a dojde k zobrazení chybové hlášky „Hrana nesmí vést do nižšího uzlu“.
Nevybrat uzel pro odstranění a pokusit se provést odstranění.	Nelze uzel odstranit, dojde k vypsání chyby v textaci „Zadaný uzel neexistuje“.
Nevybrat krajní elementy pro odstranění hrany a pokusit se hranu odstranit.	Nepůjde hranu odstranit, dojde k vypsání chybové hlášky „Neexistují krajní elementy“.

Tabulka 7.6: TC6 - Editace projektu - nepovolené operace

7.2.7 Odstranění projektu

Testovací scénář pokrývá odstranění projektu ze systému.

Krok k exekuci	Očekávaný stav systému
Přejít do správy aplikace.	Nacházím se ve správě aplikace.
Kliknout na tlačítko „Smazat“, aniž bych vybral název projektu.	Projekt nepůjde odstranit a dojde k vypsání chybové hlášky „Projekt neexistuje“.
Vybrat projekt a provést odstranění.	Proběhlo odstranění projektu.

Tabulka 7.7: TC7 - Editace projektu - odstranění projektu

7.2.8 Přidání uživatele

Testovací scénář pro přidání nového uživatele. Ve scénáři jsou obsaženy i negativní varianty, kdy už existuje zadaný login, případně nejsou zadána shodná hesla.

Krok k exekuci	Očekávaný stav systému
Přejít do správy aplikace.	Jsem ve správě aplikace.

Kliknout na tlačítko „Nový uživatel“.	Nacházím se na nové obrazovce s formulářem pro vytvoření uživatele.
Provést uložení.	Uložení selže a dojde k zobrazení chyby „Nejsou vyplněny potřebné údaje“.
Vyplnit údaje a zadat login, který již v systému existuje.	Uložení selže a dojde k zobrazení chyby v textaci. „Tento login již existuje“.
Zadat rozdílná hesla do položek „Nové heslo“ a „Potvrzení nového hesla“. Zkusit uložit uživatele.	Uživatel nelze uložit, dojde k vypsání chyby v textaci „Nové heslo a potvrzení není shodné“.
Zadat shodná hesla a uložit uživatele.	Uživatel je vytvořen.

Tabulka 7.8: TC8 - Přidání uživatele

7.2.9 Editace uživatele

Otestování editace přihlášeného uživatele. Ve scénáři je také pokryto, že nelze změnit login na již existující.

Krok k exekuci	Očekávaný stav systému
Přejít do detailu uživatele (proklik přes jméno přihlášeného uživatele).	Nacházím se v detailu uživatele.
Provést změnu loginu na již existující a uložit změnu.	Uložení nebude provedeno, dojde k zobrazení chyby v textaci „Tento login již existuje“.
Vrátit login na původní a změnit jméno uživatele. Uložit změnu.	Uložení proběhne v pořádku.
Odhlásit se ze systému.	Odhlášení proběhlo v pořádku.
Znovu se přihlásit.	Na hlavní obrazovce je v prokliku na detail uživatele již viditelné upravené jméno.

Tabulka 7.9: TC9 - Editace uživatele

7.2.10 Změna hesla

Scénář slouží k otestování změny hesla pro přihlášeného uživatele.

Krok k exekuci	Očekávaný stav systému
Přejít do detailu uživatele (proklik přes jméno přihlášeného uživatele).	Nacházím se v detailu uživatele.

Přejít na obrazovku s formulářem pro změnu hesla.	Nacházím se na nové obrazovce.
Do formuláře vyplnit neplatné aktuální heslo, zadat heslo nové a zkusit provést změnu.	Změnu nelze zadat. Je zobrazena chyba „Bylo zadáno špatné heslo“.
Vyplnit správné aktuální heslo a zadat rozdílné nové heslo a jeho potvrzení.	Změnu nelze provést. Je zobrazena chybová hláška „Nové heslo a potvrzení není shodné“.
Zadat všechny údaje správné a provést změnu.	Změnu se podařilo provést.
Odhlásit se.	Odhlášení proběhlo v pořádku.

Tabulka 7.10: TC10 - Změna hesla

7.2.11 Odstranění uživatele

Scénář pro odstranění uživatele včetně pokusu o odstranění neexistujícího uživatele. Scénář také obsahuje kontrolu, že není možné odstranit přihlášeného uživatele.

Krok k exekuci	Očekávaný stav systému
Přejít do správy aplikace.	Nacházím se ve správě aplikace.
Kliknout na tlačítko „Smazat“, aniž bych vybral uživatele.	Uživatel nepůjde odstranit a dojde k vypsání chybové hlášky „Uživatel neexistuje“.
Zkontrolovat jednotlivé položky uživatelů.	V seznamu není aktuálně přihlášený uživatel.
Vybrat uživatele a provést odstranění.	Proběhlo odstranění uživatele.

Tabulka 7.11: TC11 - Odstranění uživatele

7.2.12 Přidání nové stránky teorie

Testovací scénář pro vytvoření nové stránky teorie.

Krok k exekuci	Očekávaný stav systému
Přejít do správy aplikace.	Nacházím se ve správě aplikace.
Kliknout na tlačítko „Nová stránka teorie“ v sekci správy teorie, aniž bych vyplnil její název.	Stránku nelze vytvořit a dojde k vypsání chybové hlášky „Název stránky je prázdný“.
Vyplnit název stránky a provést vytvoření.	Stránka se vytvořila.
Vrátit se na úvodní rozcestník.	Nacházím se na úvodním rozcestníku.

Přejít na obrazovku s teorií.	Je zobrazena obrazovka teorie. V záložkách se nachází nově vytvořená stránka.

Tabulka 7.12: TC12 - Vytvoření nové stránky teorie

7.2.13 Editace textu

Testovací scénář, jenž otestuje využití textového editoru u editace teorie.

Krok k exekuci	Očekávaný stav systému
Z úvodní obrazovky přejít na stránku s teorií.	Nacházím se na obrazovce teorie.
Vybrat si jednu ze záložek a přejít do editace.	Nacházím se na nové obrazovce s textovým editorem.
Provést úpravy v textovém editoru.	Fungují všechny operace v editoru.
Zobrazit si text v náhledu.	Náhled byl zobrazen a odpovídá textu v editoru.
Uložit upravený text.	Text se podařilo uložit.
Vrátit se na stránku teorie.	Stránka obsahuje aktualizovaný text.

Tabulka 7.13: TC13 - Editace textu

7.2.14 Odstranění stránky

Otestování odstranění stránky v sekci teorie. Scénář obsahuje i negativní variantu, kdy není vyplněn název stránky k odstranění.

Krok k exekuci	Očekávaný stav systému
Přejít do správy aplikace.	Nacházím se ve správě aplikace.
Kliknout na tlačítko „Smazat“, aniž bych vybral stránku.	Stránku nepůjde odstranit a dojde k vypsání chybové hlášky „Stránka neexistuje“.
Vybrat stránku a provést odstranění.	Proběhlo odstranění stránky.

Tabulka 7.14: TC14 - Odstranění stránky

7.3 Testování aplikace s uživateli

V této části textu popisuji samotné testování s uživateli. Vzhledem k tomu, že v době dokončení práce ještě nebyly k dispozici zdroje, kde později aplikace bude nasazena, musel jsem vyřešit, jak distribuovat aplikaci k samotným

uživatelům.

Pokud bych v této práci analyzoval a vyvíjel desktopovou aplikaci, nejednalo by se o žádný problém a uživatelé by si nainstalovali aplikaci na koncová zařízení. Jak to však řešit u webové aplikace s podporou databáze, která je určená pro nasazení na server a ideálně nalézt řešení, které co nejméně zatíží uživatele? Ti si určitě kvůli testování nechtějí instalovat lokální aplikační server nebo server databázový. Navíc u některých uživatelů mohou tyto požadavky dokonce překračovat jejich uživatelské schopnosti.

Pro řešení této problematiky jsem se nakonec rozhodl využít možností technologie Spring Boot, kterou jsem již v aplikaci používal. Tím se vyřešila nutnost lokálního serveru, protože tato technologie spustí aplikaci na vlastním vestavěném serveru Tomcat. Aplikace je poté spouštěna jako běžná Java aplikace.

Stále však zůstávala nutnost existence databáze. Tu bylo nutné nějak „dostat“ k uživatelům. Zde jsem nalezl řešení v „memory“ databázi H2. Ta podporuje všechny použité technologie v „Aplikaci PERT“. Pro vygenerování datového modelu, jsem zde využil schopnosti Hibernate. Poté již stačilo doplnit inicální SQL skript s daty, která jsou vždy do aplikace při jejím spuštění nasazena.

Toto řešení databáze však nepřineslo jen pozitiva, v některých oblastech jsem musel proti plnohodnotnému serveru ustoupit. Například není možné ukládat řetězce delší než 256 znaků. I přesto bylo možné H2 databázi využít, protože žádný kompromisů nebyl blokující pro otestování funkčnosti aplikace.

Z nutnosti instalace webového serveru, lokální databáze a následně také nasazení WARka aplikace na nainstalovaný server, jsem se dokázal dostat pouze k nutnosti mít nainstalovanou Javu. Tu navíc většina testerů již měla nainstalovanou. Na testerech tedy zbývalo pouze pustit samotný WAR soubor aplikace. Pro tyto účely jsem navíc k aplikaci vždy přidal skript, který toto spuštění provedl.

7.4 Oprava chyb - Chyby

V této kapitole shrnuji nalezené chyby z fáze testování a jejich následnou opravu. V rámci testování došlo k identifikování osmnácti chyb v sytému. Ukázky chyb jsou uvedeny v následujícím seznamu, zejména první čtyři chyby byly kritické a blokovaly by používání aplikace. U ostatních chyb se jednalo o špatné textace nebo neintuitivní uživatelské rozhraní.

- Při uložení grafu došlo k nastavení doby trvání činností na hodnotu nula.
- Při odstraňování stránek teorie nedošlo k aktualizaci seznamu a šlo odstranit stránku dvakrát. Druhé odstranění selhalo a zablokovalo aplikaci.
- Šlo vytvořit uživatele s již existujícím loginem. Poté pro tohoto uživatele nefungovalo přihlášení, protože se nepodařilo určit heslo.

- Na správu aplikace bylo možné přejít zadáním URL adresy a poté provést editaci bez oprávnění.
- U teorie se špatně zarovnával text při použití seznamu.
- Uživatelé špatně vyhledávali element pro návrat na rozcestník. Vyřešeno umožněním návratu přes nadpis aplikace.
- Doby trvání obsahovaly špatné textace.
- Informace o projektu vedly na neexistující stránku.
- Pro příliš mnoho cest v grafu u simulace, docházelo k rozpadnutí GUI. Vyřešeno omezením velikosti tabulek. Při větším množství cest je poté nutné scrollovat v tabulce.
- Po vytvoření grafu se neaktualizoval seznam grafů.
- Špatná chybová hláška u simulace, pokud se rovnala optimistická a pesimistická doba trvání.

Kapitola 8

Dokumentace

V této kapitole popisuji dokumentaci vytvořeného systému. Dokumentace vznikala pro tři různé skupiny osob. Jednalo se o dokumentaci pro programátory. Instalační příručku pro osoby, které nasadí a budou udržovat „Aplikaci PERT“ v provozu. A příručka pro samotné uživatele systému.

8.1 Zdrojové kódy

Jedná se o dokumentaci zdrojových kódů pro kterou jsem využil Javadoc. Jedná o nástroj, který projde zdrojové kódy a na základě speciálních znaků (definovaných v nástroji) vybírá komentáře ze zdrojového kódu a z těchto fragmentů poté vygeneruje interaktivní dokumentaci projektu ve formě webových stránek.

Tato dokumentace je primárně určena programátorům a slouží k orientaci v projektu, pokud dojde k dalšímu vývoji aplikace. Jelikož je dokumentace ve formátu webových stránek není přiložena v textu této práce, ale dokumentaci je možné nalézt na přiloženém médiu.

8.2 Instalační příručka

V této kapitole popisuji obsah instalační příručky. Ta slouží pro nasazení aplikace a následně její provoz. Celý dokument uvádím v této kapitole. Dokument je součástí instalačního setu.

8.2.1 Úvod

Dokument slouží k instalaci a provozu „Aplikace PERT“. V dokumentu jsou obsaženy prerekvizity nasazení a popsán samotný postup nasazení aplikace. Dokument také obsahuje postup pro resetování hesla uživatele.

Název parametru	Popis parametru
logging.file	Soubor, kam se budou zapisovat logy aplikace.
logging.level.*	Úroveň logování v rámci aplikace.
spring.datasource.username	Jméno uživatele pro přihlášení k databázi.
spring.datasource.password	Heslo uživatele pro přihlášení k databázi.
spring.datasource.url	URL adresa databáze.
spring.datasource.dbcp2.initial-size	Iniciální velikost poolu připojení do DB.
spring.datasource.dbcp2.max-total	Maximální velikost poolu připojení do DB.

Tabulka 8.1: Popis konfiguračních parametrů aplikace

8.2.2 Prerekvizity

- WildFly 12 Final nebo Tomcat 8.5.30.
- Java 8
- MySQL databáze s existujícím schématem.

8.2.3 Postup nasazení

1. Do databáze nasadit skript ze složky **db/dm.sql** pro vytvoření datového modelu
2. Do databáze nasadit skript ze složky **db/data.sql**. Ten obsahuje základní data aplikace.
3. Nastavit parametry aplikace v **WEB-INF/classes/application.properties** dle tabulky 8.1.
4. Nasadit WARko(**app/estimation-1.0.1-SNAPSHOT.war**) aplikace na server.
5. Spustit server.
6. Přihlásit se za účet admin/admin a změnit heslo. Nebo vytvořit vlastní účet a admin účet odstranit.

8.2.4 Resetování uživatelského hesla

První verze aplikace nedisponuje funkcionalitou pro resetování uživatelského hesla. Proto je nutné resetování hesla provést operativním zásahem do databáze. Resetování je možné provést následujícím SQL příkazem: *update c_users set password = 'HASH_HESLA' where login = 'USER_LOGIN'*.

Do řetězce HASH_HESLA je nutné doplnit nové heslo rozšířené o řetezec „salt“ a zahashované metodou SHA512. USER_LOGIN poté představuje login uživatele.

8.3 Uživatelská příručka

Zde se jedná o dokumentaci sloužící samotným uživatelům systému. V dokumentaci jsou popsané jednotlivé obrazovky aplikace a obsahuje také jejich náhledy. Jedná se o obsah kapitoly s návrhem GUI aplikace (5). Dokument obsahuje navíc pouze úvod, kde je vysvětleno k čemu tento dokument slouží. Proto tento dokument v textu práce neuvádím a je možné ho nalézt v elektronické podobě na přiloženém médiu.

Kapitola 9

Ekonomické zhodnocení

V rámci této práce jsem provedl analýzu a následně i implementaci aplikace pro výuku odhadů s metodikou PERT a její simulace. Cílem této aplikace je ulehčit výuku předmětu Metody a projektové řízení na ČZU. Aplikace také nabídne pomocný nástroj každému, kdo o využití „Aplikace PERT“ bude mít zájem.

V této kapitole nechci uvádět přesná čísla, že uživatelé, kteří použijí „Aplikaci PERT“ dokáží vytvářet přesnější odhady o desítky procent a budou vracet přesné MD. To kolikrát není u nových projektů ani možné, protože je zde spousta neznámých, které se musí během projektu vyřešit.

Rozptyl se v odhadech pracnosti vyskytoval vždy a není možné ho vymýtit. Kolikrát je vytvářen nepřesnostmi v zadání, případně jako ochrana dodavatele při problematických úpravách. V rozumné míře tedy dává smysl.

Myslím si, že uživatele, jenž využijí mnou navržený systém, může aplikace donutit zamyslet se nad poměrem střední hodnoty a rozptylu. Například pokud je střední hodnota menší než rozptyl, tak to může indikovat, že odhad není vypracován kvalitně. Nemusí to být nutně samotný odhad, ale může se také například jednat o nekompletní zadání. Poté je nejspíš na místě zamyslet se, zda má takový projekt smysl a místo zisku nepřinese tento projekt spíše problémy.

Dané si dovolím ilustrovat na malé ukázce, kdy budu mít tři činnosti, které na sebe navazují. V rámci zjednodušení jsem neuvažoval více paralelních činností. Také nepočítám s vícečlenným týmem, ale k dispozici je jeden analytik, programátor a tester.

Činnosti uvádím v seznamu ve formě název činnosti a doba trvání. Doba trvání je uvedena v jednotce man-day (MD), jedná se o 8 hodin.

1. Analýza (5 MD)
2. Implementace (10 MD)
3. Testování (4 MD)

V prvním příkladu nepoužívám žádný rozptyl. Odhad je vypracován s fixní dobou trvání. I tato forma se ještě používá a je obrovsky riziková. Tento formát odhadu si můžu dovolit, pokud odhadovaný systém dokonale znám a zadání je připravené přesně a nenechává prostor k diskuzím.

Zde mám odhady pro činnosti odhadnuté přesnou hodnotou. Uvažujme však, že analýza trvá 12 MD, protože jsem podcenil tvorbu odhadu a neexistuje kvalitní zadání. Jak teď řešit alokaci zdrojů? Počítal jsem s tím, že analýza bude trvat 5 MD, po tomto období mám již nasmlouvaného programátora. Najednou však mám analýzu delší o 7 MD a jedná se o více jak polovinu implementace, kdy platím programátora a nemám pro něj práci.

U implementace mám najednou z 10 MD k dispozici pouze 3 MD. Navíc to nemusí být jediný problém, co když nemám již alokované zdroje programátora po fázi implementace. Během implementace musím shánět nového programátora. Ten musí nejdřív pochopit navrženou architekturu systému, nějakou dobu také potrvá než se dokáže orientovat v cizím kódu.

Všechny popsané komplikace zase vedou k navýšení celkové pracnosti. Znovu je nutné upravovat začátek a konec další fáze, opět může nastat problém s alokací zdrojů. U navazujících činností se již zdržení akumulovalo a celková doba trvání projektu se zvýšila. Všechno vyplynulo z špatně vytvořeného odhadu s fixní hodnotou doby trvání.

Při dalším příkladu již odhad pracuje s rozptylem a doba trvání je ve formě optimistická - očekávaná - pesimistická.

1. Analýza (2 MD - 5 MD - 12 MD)
2. Implementace (10 MD - 12 MD - 16 MD)
3. Testování (4 MD - 6 MD - 8 MD)

Zde však mám obrovský rozptyl pro analýzu. U té je spodní a horní hranice šestinásobná. Jak řešit alokaci zdrojů v této situaci? Můžu se řídit očekávanou dobou trvání a počítat s drobnou rezervou. Počítám s tím, že analýza bude trvat maximálně 7 MD a po tomto období mám již domluveného programátora. Znovu se dostávám do situace, kdy přicházím o část implementace. Nejedná se již o tak výrazný zásah jako u fixní hodnoty doby trvání, přesto se však jedná o poměrně významnou část činnosti.

Znovu dojde k navýšení celkové pracnosti. V uvažované situaci všechno vzniklo obrovským rozptylem u první činnosti. U dalších činností se znovu zdržení pouze akumulovalo. Takhle obrovský rozptyl by měl již upozornit, že je něco špatně a měl by vést k otázkám. Mám dostatečné zadání? Čeho se obávám, že jsem se rozhodl použít takto velký rozptyl?

Nyní vezmu stejné činnosti, jen disponuji přesnějším zadáním a odhaduji analýzu na 8 MD - 10 MD - 12 MD. Zde už rozptyl není tak výrazný a mám přesnější odhad doby trvání. Znovu uvažujeme nejhorší variantu pro analýzu, najednou se jedná o zdržení pouze 2 MD. Samozřejmě zase se jedná

o zásah do implementace, ale zásluhou výrazně přesnějšího odhadu analýzy již posun nenese takové riziko. Navíc se s menším rozptylem dá mnohem lépe pracovat a mohu si připravit harmonogram tak, abych dokázal dopady na projekt minimalizovat.

Zatím jsem řešil pouze interní problémy dodavatele. Je však nutné si uvědomit, že výše popsané problémy mají dopad i na zákazníka. Ti nejsou příliš rádi, pokud se jim posouvají termíny. Na dodání aplikace mohou mít navázané další integrační činnosti nebo reklamní akce. Poté se může na straně dodavatele řešit, že aplikace se nějakým způsobem dokončí a je nutné dodržet termíny. Toto řešení přinese dopad na kvalitu, nějakým způsobem se provede implementace, ale kvalita bude velmi „pofidérní“. Navíc jakékoliv zdržení projektu je pro dodavatele finančně nevýhodné. Činnost, realizovaná mimo původně odhadovaný harmonogram, bývá výrazně dražší.

V této kapitole jsem shrnul následky špatně vypracovaného odhadu. Doufám, že uživatelé, jenž se setkají s „aplikací PERT“, se při vysokém rozptylu zamyslí nad takovým odhadem. V aplikaci je chování vysokého rozptylu zejména viditelné v simulaci, kdy mi doby trvání jednotlivých činností „skáčou“ v obrovském rozsahu.

Kapitola 10

Možná další vylepšení

Cílem této diplomové práce je vytvořit nástroj, který usnadní výuku předmětu Metody a projektové řízení na ČZU. Jedná se o první verzi tohoto systému. Je zde tedy prostor pro mnoho dalších vylepšeních. Případně je možné integrovat „Aplikaci PERT“ do systému univerzity. Tyto možná vylepšení popisuji v dalších částech této kapitoly.

10.1 Napojení na LDAP

První vylepšení se týká výraznější integrace systému do univerzitní architektury IT. Od začátku vznikal tento systém jako separátní aplikace a nebylo rozhodnuto, kde bude provozována a jestli vůbec ve vnitřní doméně ČZU. Z těchto důvodů postrádá aplikace jakoukoliv integraci se současnými systémy ČZU.

Momentálně v „Aplikaci PERT“ existují pouze administrátorské účty, které se ověřují proti vlastní databázi systému. Kvůli chybějící integraci na emailový server, není možné provádět resetování hesla. To je u první verze systému nutné řešit operativním zásahem přímo v databázi (použitý algoritmus šifrování je popsán v instalační příručce).

Tento problém je možné v dalších verzích vyřešit napojením na fakultní LDAP a přihlašováním přes univerzitní přihlašovací údaje. Databáze v aplikaci poté bude sloužit pouze k přiřazení role v rámci Aplikace PERT.

10.2 Napojení na emailový server

Pokud bude aplikace provozována v rámci interní sítě univerzity, je možné zvážit, zda neintegrovat aplikaci na univerzitní e-mailový server. V Aplikaci poté může být implementována logika pro rozesílání různých notifikací nebo upozornění.

Rozesílání emailů také může sloužit k resetování hesla. Na základě požadavku na resetování hesla, dojde k odeslání e-mailu s odkazem do aplikace. Na odkazu si poté vlastník účtu (adresy) nastaví nové heslo do aplikace.

10.3 Přidání datové části

První verze systému je navržena jako náhrada současných materiálů ve formě Excelu. V Excelu je velmi pracné vytvořit nový projekt, protože tabulky a vzorce jsou specifické pro každou topologii uzlů. Vzhledem k tomu, že Excely spravují vyučující, je v první verzi aplikace dostupná konfigurační část pouze pro administrátory.

Tato logika je také použita proto, že systém momentálně neumožňuje resetování hesla. Tato překážka může být odstraněna implementací 10.1 nebo 10.2.

Poté je možné rozšířit databázovou strukturu o datové tabulky. Ty umožní správu vlastních grafů a případně textů i běžně přihlášeným uživatelům.

10.4 Umožnění přístupu běžným uživatelům

Tato kapitola úzce souvisí s kapitolou předchozí. Bez přidání datové části nedává smysl rozšířit aplikaci o přístupové údaje běžných uživatelů. Po jejich doplnění je v aplikaci možné vytvářet projekty a editovat data s vazbou k přihlášenému uživateli. Poté může dojít k doplnění čistě uživatelské sekce, například poznámek.

Pro přidání běžných uživatelů je nutné implementovat 10.1 nebo 10.2. Dané úpravy jsou nutné pro možnost resetování hesla. Pokud dojde k napojení na LDAP, je možné, že systém pouze přebere přihlášené uživatele a přiřadí jim roli v systému. U integrace na e-mailový server dojde pouze k doplnění komunikačního kanálu pro resetování hesla, ale samotné přihlášení zajistí „Aplikace PERT“.

10.5 Testy z teorie

Tato kapitola má předpoklad v tom, že v rámci systému je možné identifikovat uživatele. Proto je pro realizaci těchto úprav nutné implementovat 10.4 a jeho prerekvizity. Poté již nebrání nic implementaci testů do systému.

Ty mohou být realizovány například formou otázek a výběru odpovědi. V této variantě je možné i testy okamžitě ohodnotit. Pokud testy mají obsahovat i „otevřené“ otázky, je nutné ohodnocení těchto otázek hodnotícím.

10.6 Integrace na kreslící nástroj

Při analýze procesů a funkčních požadavků bylo domluveno, že zachycení uzlů a hran grafu je možné realizovat pouze ilustrací a separátním zadáním uzlů a hran. Toto řešení se mi příliš nelíbilo a snažil jsem se najít jiné alternativy. Nakonec jsem se rozhodl pro využití frameworku Primefaces, který jsem již v aplikaci měl k dispozici. Framework obsahuje grafové komponenty. Ty primárně slouží k zobrazení dat a nejsou interaktivní. Proto je v aplikaci nyní využívána logika s grafovou komponentou, která je spravována přes ovládací panel. Ten poté obsluhuje jakoukoliv manipulaci s grafem. Pro provedení změn je využita technologie AJAX, která zajistí zobrazení změn v GUI bez nutnosti přenačtení celé stránky.

Výše popsané neduhy můžou být v dalších verzích „Aplikace PERT“ vyřešeny integrací samostatného kreslícího nástroje do aplikace. Tento nástroj musí splňovat následující body:

- Manipulace s uzly (vytváření, mazání, posouvání).
- Manipulace s hrany (vytváření, mazání).
- Získání souřadnic uzlů z grafu.
- Získání informací o hranách z grafu.

Závěr

Cílem této diplomové práce bylo vytvořit webovou aplikace pro výuku metody PERT a její simulace. Aplikace slouží primárně pro výuku předmětu Metody a projektové řízení na České zemědělské univerzitě.

Pro vytvoření aplikace jsem nejdříve musel nastudovat teoretické základy metody PERT. Mezi tyto základy patří podklady o pravděpodobnosti, statistice a samotných metodách síťové analýzy, CPM a PERT. V rešeršní části také uvádím studie uvažovaných technologiích.

Poté již následuje praktická část práce. V té jsem nejdříve provedl analýzu procesů a určil funkční i nefunkční požadavky, které musí aplikace splňovat. Následně uvádím mapování mezi procesy a požadavky. V analytické části jsem také vypracoval podrobnější popis funkčních požadavků. Také jsem zde navrhl relační model databáze a uživatelské rozhraní aplikace.

Po analýze již navazuji popisem samotné implementace informačního systému. V rámci implementace jsem nejdříve musel zvolit jaké technologie pro vybudování systému využiji. Na zvolených technologiích jsem poté vybudoval webovou aplikaci. Po implementaci jsem otestoval aplikaci na reálných uživatelích, z tohoto testování vyplynuly výhrady k uživatelskému rozhraní aplikace. Po zapracování připomínek jsem podrobil aplikaci akceptačnímu testování. To bylo provedeno na základě testovacích scénářů, které jsem vypracoval v této práci.

V poslední části práce jsem zpracoval další možná vylepšení systému. Ty se zaměřují zejména na budoucí rozšíření aplikace. Dále zde také popisuji dokumentaci systému. Tu jsem vypracoval na třech úrovních. Jedná se o dokumentaci zdrojových kódů, uživatelskou a instalační příručku. Každá tato dokumentace je určená jiné skupině osob.

Věřím, že aplikace navržená v této práci se stane pravidelnou součástí výuky předmětu Metody a projektové řízení a usnadní pochopení metody PERT. Také doufám, že uživatelé, kteří se s aplikací setkají, se při tvorbě odhadu zamyslí nad tím, co vše může způsobit vysoký rozptyl u doby trvání projektu a pokusí se ho minimalizovat.

Literatura

[1] Grinstead, C. M.; Snell, J. L.: *Introduction to Probability.* American Mathematical Society, 1997, ISBN 0821807498.

[2] Stern, H.: Introduction to Probability and Statistics for Computer Science. [online prezentace][cit. 1. 5. 2018]. Dostupné z: `http://www.ics.uci.edu/~sternh/courses/67/stat67slides.pdf`

[3] Trudeau, R. J.: *Introduction to Graph Theory.* Longman, 1993, ISBN 0-582-24993-7.

[4] Shields, R.: Cultural Topology: The Seven Bridges of Königsburg, 1736. *Sage Publishing*, 2012, [online][cit. 25. 4. 2018]. Dostupné z: `http://journals.sagepub.com/doi/pdf/10.1177/0263276412451161`

[5] Management Mania: *Řízení projektů (Project Management).* [online][cit. 24. 4. 2018]. Dostupné z: `https://managementmania.com/cs/metody-rizeni-projektu`

[6] Weaver, P.: Henry L Gantt, 1861 - 1919, Debunking the myths, a retrospective view of his work. *PM World Journal*, 2012, [online][cit. 25. 4. 2018]. Dostupné z: `https://pmworldjournal.net/wp-content/uploads/2012/11/PMWJ5-Dec2012-WEAVER-Henry-Gantt-Debunking-Myths-Featured-Paper.pdf`

[7] Šubrt, T.; Langrová, P.: *Projektové řízeni I.* Česká zemědělská univerzita v Praze, 2013, ISBN 978-80-213-1194-7.

[8] Kelley, J. E.; Walker, M. R.: Critical-Path Planning and Scheduling. 1959, [online][cit. 25. 4. 2018]. Dostupné z: `https://www.computer.org/csdl/proceedings/afips/1959/5055/00/50550160.pdf`

[9] PM Knowledge Center: *The Program Evaluation and Review Technique (PERT): Incorporating activity time variability in a project schedule.* [online][cit. 24. 4. 2018]. Dostupné z: `http://www.pmknowledgecenter.com/`

dynamic_scheduling/baseline/program-evaluation-and-review-technique-pert-incorporating-activity-time-variability

[10] *Project Managment Body of Knowledge*. Project Management Institute, 2013, ISBN 1-880410-22-2.

[11] Program Evaluation Research Task, Summary Report, Phase 1. *Defence Technical Information Center*, 1958, [online][cit. 25. 4. 2018]. Dostupné z: http://www.dtic.mil/dtic/tr/fulltext/u2/735902.pdf

[12] Badiru, A. B.: A simulation approach to PERT network analysis. *Sage Publishing*, 1991, [online][cit. 25. 4. 2018]. Dostupné z: http://journals.sagepub.com/doi/pdf/10.1177/003754979105700409

[13] Binstock, A.: Java's 20 Years Of Innovation. [online][cit. 24. 4. 2018]. Dostupné z: https://www.forbes.com/sites/oracle/2015/05/20/javas-20-years-of-innovation

[14] TIOBE - The Software Quality Company: *TIOBE Index*. [online][cit. 24. 4. 2018]. Dostupné z: https://www.tiobe.com/tiobe-index/

[15] Oracle: *Java 5 Release Notes*. [online][cit. 24. 4. 2018]. Dostupné z: http://www.oracle.com/technetwork/java/javase/releasenotes-142123.html

[16] Oracle: *Java 8 Release Notes*. [online][cit. 24. 4. 2018]. Dostupné z: http://www.oracle.com/technetwork/java/javase/8all-relnotes-2226344.html

[17] Schildt, H.: *Mistrovství - Java*. Computer press, 2014, ISBN 9788025141458.

[18] Oracle: *JSR 338: Java Persistence API, Version 2.1*. [online][cit. 24. 4. 2018]. Dostupné z: http://download.oracle.com/otn-pub/jcp/persistence-2_1-fr-eval-spec/JavaPersistence.pdf

[19] Walls, C.: *Spring in Action*. Manning, 2015, ISBN 978-1-61729-120-3.

[20] Webb, P.: Spring Boot 1.0.0.RC1 Released. leden 2014, [online][cit. 25. 4. 2018]. Dostupné z: https://spring.io/blog/2014/01/22/spring-boot-1-0-0-rc1-released

[21] Holaň, J.; Kvasnovský, O.: *Vaadin 7 Cookbook*. Packt Publishing, 2013, ISBN 978-1-84951-880-2.

[22] The THYMELEAF team: *Tutorial: Using Thymeleaf*. [online][cit. 24. 4. 2018]. Dostupné z: https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html

[23] Çağatay Çivici: PrimeFaces 1.0.0 And 2.0.0 Released. únor 2012, [online][cit. 25. 4. 2018]. Dostupné z: https://www.primefaces.org/primefaces-1-0-0-and-2-0-0-released/

[24] Sincan, M.: PrimeFaces 6.2.2 Released. březen 2018, [online][cit. 25. 4. 2018]. Dostupné z: https://www.primefaces.org/primefaces-6-2-2-released/

[25] PrimeTek: *Who Uses PrimeFaces*. [online][cit. 24. 4. 2018]. Dostupné z: https://www.primefaces.org/whouses/

[26] PrimeTek: *PrimeFaces User Guide 6.2*. [online][cit. 24. 4. 2018]. Dostupné z: https://www.primefaces.org/docs/guide/primefaces_user_guide_6_2.pdf

[27] Arrington, M.: Sun Picks Up MySQL For one Billion. leden 2008, [online][cit. 25. 4. 2018]. Dostupné z: https://techcrunch.com/2008/01/16/sun-picks-up-mysql-for-1-billion-open-source-is-a-legitimate-business-model/

[28] Voxland, N.: LIQUIBASE 1.0 RELEASED. červen 2007, [online][cit. 25. 4. 2018]. Dostupné z: http://www.liquibase.org/2007/06/liquibase-10-released.html

[29] Voxland, N.: LIQUIBASE 3.6.0 RELEASED. duben 2018, [online][cit. 25. 4. 2018]. Dostupné z: https://www.liquibase.org/2018/04/liquibase-3-6-0-released.html

[30] Liquibase: *Liquibase Documentation*. [online][cit. 24. 4. 2018]. Dostupné z: http://www.liquibase.org/documentation/index.html

[31] Ercoli, F. M.: Hibernate OGM 5.3.1.Final is out. *Red Hat*, březen 2018, [online][cit. 25. 4. 2018]. Dostupné z: http://in.relation.to/2018/03/29/hibernate-ogm-5-3-1-Final-released/

[32] Bauer, C.; King, G.: *Hibernate in Action*. Manning, 2007, ISBN 9781932394153.

[33] Apache Software Foundation: *Maven 1.x News*. [online][cit. 24. 4. 2018]. Dostupné z: http://maven.apache.org/archives/maven-1.x/news.html

[34] Apache Software Foundation: *Release Notes Maven 3.5.3*. [online][cit. 24. 4. 2018]. Dostupné z: https://maven.apache.org/docs/3.5.3/release-notes.html

[35] Srirangan: *Apache Maven 3 cookbook*. Packt Pub., 2011, ISBN 978-1-849512-44-2.

[36] Git: *Reference*. [online][cit. 24. 4. 2018]. Dostupné z: `https://git-scm.com/docs`

[37] Chacon, S.; Straub, B.: *Pro Git*. Apress, 2014, ISBN 978-1484200773.

[38] Royce, W. W.: MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS. 1970, [online][cit. 25. 4. 2018]. Dostupné z: `http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf`

[39] Liquibase: *Liquibase Best Practices*. [online][cit. 24. 4. 2018]. Dostupné z: `http://www.liquibase.org/bestpractices.html`

Příloha **A**

Seznam použitých zkratek

GUI Grafické uživatelské rozhraní

ORM Objektově relační mapování

XML Extensible markup language

URL Uniform Resource Locator

JDBC Java Database Connectivity

JPA Java Persistence API

JVM Java virtual machine

AJAX Asynchronous JavaScipt and XML

HTTP Hypertext Transfer Protocol

LDAP Lightweight Directory Access Protocol

WAR Web Application Resource

MD Man-day

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `install` ..... adresář s instalační částí práce
  - `db` ..... skripty pro vytvoření dadatabáze
    - `dm.sql` ..... skript pro vytvoření datového modelu
    - `data.sql` ..... skript s ukázkovou konfigurací
  - `war` ..... adresář s war archivy aplikace
  - `dokumentace` ..... adresář s dokumentací
    - `Instalační příručka.pdf` ..... instalační příručka
    - `Uživatelská příručka.pdf` ..... uživatelská příručka
- `src` ..... zdrojové kódy
  - `impl` ..... zdrojové kódy implementace
    - `estimation` ..... zdrojové kódy aplikace
    - `db` ..... zdrojové kódy k databázi
    - `javadoc` ..... Javadoc aplikace
  - `thesis` ..... zdrojové kódy textu práce
- `text` ..... text práce
  - `DP_Sýkora_Tomáš_2018.pdf` ..... text práce ve formátu PDF