České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Projektová kancelář na bázi volně dostupných nástrojů

Bakalářská práce

Dominik Kundrát

Studijní program: Softwarové inženýrství a technologie
Specializace: Business informatics
Vedoucí práce: Ing. Pavel Náplava, Ph.D.

Květen 2024

Vedoucí práce:
Ing. Pavel Náplava, Ph.D.
Centrum znalostního managmentu
Fakulta elektronická
České vysoké učení technické v Praze
Technická 2
160 00 Praha 6
Česká republika

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kundrát** Jméno: **Dominik** Osobní číslo: **507323**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

Specializace: **Business informatics**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Projektová kancelář na bázi volně dostupných nástrojů

Název bakalářské práce anglicky:

A project office based on freely available tools

Pokyny pro vypracování:

Prozkoumejte existující volně dostupné nástroje, které je možné využít pro plánování a řízení menších projektů. Konkrétně WBS a Ganttův diagram. Následně vybrané nástroje propojte do jednoho celku, aby nebylo nutné komplikovaně mezi nimi přepisovat data. Postupujte následovně:
- Definujte pojmy projekt, projektové řízení, WBS, Ganttův diagram atd.
- Analyzujte strukturu WBS a Ganttova diagramu. Porovnejte je a prozkoumejte, zda pro ně existují standardizované datové formáty.
- Prozkoumejte existující, volně dostupné nástroje pro podporu tvorby WBS a Ganttova diagramu.
- Vyberte takové nástroje, které pracují s takovými formáty, že je možné je jednoduše zpracovat a použít pro propojení nástrojů.
- Definujte nejčastější scénáře projektového manažera (tvorba a úprava plánů, konverze mezi WBS a Ganttovým diagramem atd.).
- Navrhněte a vytvořte takový nástroj, který bude minimálně definované scénáře podporovat.
- Vytvořený nástroj uživatelsky otestujte.

Seznam doporučené literatury:

Roy, U. K. (2015). Advanced Java Programming [e-kniha]. Oxford University Press. Dostupné z: https://app.knovel.com/hotlink/toc/id:kpAJP00001/advanced-java-programming/advanced-java-programming
Campesato, O. (2022). Java for Developers - Pocket Primer [e-kniha]. Mercury Learning and Information. Dostupné z: https://app.knovel.com/hotlink/toc/id:kpJDPP0001/java-developers-pocket/java-developers-pocket
Banzal, S. (2020). XML Basics [e-kniha]. Mercury Learning and Information. Dostupné z: https://app.knovel.com/hotlink/toc/id:kpXMLB0007/xml-basics/xml-basics
Project Management Institute, Inc. (PMI). (2019). Practice Standard for Work Breakdown Structures (3rd Edition) [e-kniha]. Project Management Institute, Inc. (PMI). Dostupné z: https://app.knovel.com/hotlink/toc/id:kpPSWBSE11/practice-standard-work/practice-standard-work
Project Management Institute, Inc. (PMI). (2013). A Guide to the Project Management Body of Knowledge (PMBOK® Guide) [e-kniha]. Project Management Institute, Inc. (PMI). Dostupné z: https://www.academia.edu/43812564/PMBOK_Guide_Fifth_Edition

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Náplava, Ph.D. Centrum znalostního managementu FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **07.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

Ing. Pavel Náplava, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studenta

Prohlášení

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

V Praze, 24. května 2024

.....
Dominik Kundrát

Poděkování

Nejprve bych rád poděkoval vedoucímu této bakalářské práce, panu Ing. Pavlu Náplavovi Ph.D., za jeho užitečné rady, bleskurychlé odpovědi a pravidelné konzultace, které byly klíčové pro úspěšné dokončení této práce. Dále bych chtěl poděkovat rodině, přítelkyni a kamarádům za jejich podporu v průběhu celého studia i tvorby této bakalářské práce.

Abstrakt

Tato bakalářská práce se věnuje návrhu projektové kanceláře využívající volně dostupné nástroje. Cílem je integrovat existující nástroje do jednotného systému pro podporu plánování a řízení menších projektů, konkrétně těch, které využívají vodopádovou metodu a jsou založeny na Work Breakdown Structure (WBS) a Ganttově diagramu.

Výstupem práce je kombinace dvou existujících nástrojů doplněná o nový konvertor, umožňující převod mezi soubory těchto nástrojů. Tímto způsobem je projektovým manažerům poskytnuto komplexní řešení bez potřeby investice do nákladných a často nadměrně složitých nástrojů.

Práce obsahuje analýzu volně dostupných nástrojů, výběr nejvhodnějších kandidátů, vývoj konvertoru pro převod mezi WBS a Ganttovým diagramem, a jeho následné testování. Testovací fáze zahrnuje jak systémové, tak uživatelské testy.

Výsledky uživatelského testování potvrdily, že vyvinutá aplikace má potenciál zlepšit efektivitu plánování a řízení menších projektů.

Klíčová slova: Work Breakdown Structure (WBS), Ganttův diagram, projektová kancelář, konverze, konvertor, projektové plánování, projektové řízení, volně dostupné nástroje.

Vedoucí práce: Ing. Pavel Náplava, Ph.D.

Abstract

This bachelor thesis is about the design of a project office using freely available tools. The aim is to integrate existing tools into a unified system to support the planning and management of smaller projects, specifically those using the waterfall method and based on the Work Breakdown Structure (WBS) and the Gantt chart.

The output of this work is a combination of two existing tools complemented by a new converter, allowing conversion between these tool files. In this way, a comprehensive solution is provided to project managers without the need to invest in expensive and often overly complex tools.

The work includes an analysis of freely available tools, the selection of the most suitable candidates, the development of a converter for conversion between WBS and Gantt chart, and its subsequent testing. The testing phase includes both system and user tests.

The results of user testing confirmed that the developed application has potential to improve the efficiency of planning and management of smaller projects.

Keywords: Work Breakdown Structure (WBS), Gantt chart, project office, conversion, converter, project planning, project management, freely available tools.

Title translation: A project office based on freely available tools

Seznam obrázků

Seznam tabulek

Seznam ukázek kódu

Seznam zkratek

API Application programming Interface. 28, 29, 38, 47

BE Backend. 27, 28, 31, 32, 34, 48

CSS Cascading Style Sheets. 30, 39

DOM Document Object Model. 28, 39

DRY Don't Repeat Yourself. 29

Dto Data transfer object. 34, 35, 37, 46

E2E End-to-end. 47

FE Frontend. 27, 30, 31, 34, 37, 39, 48

FRQ Functional Requirements. 23

HTML Hypertext Markup Language. 30, 39

JAXB Java Architecture for XML Binding. 28

JSON JavaScript Object Notation. 29, 37

MVC Model-View-Controller. 31, 32, 39

NFRQ Non-Functional Requirements. 23

PDF Portable Document Format. 27

PMBOK Project Management Body of Knowledge. 10

PMO Project Management Office. 13

POJO Plain Old Java Object. 29, 34

PP&C Production Planning and Control. 7

REST Representational State Transfer. 33, 38

SLF4J Simple Logging Facade for Java. 37

UI User Interface. 30, 38, 42

W3C World Wide Web Consortium. 28

WBS Work Breakdown Structure. 1–3, 5–8, 10–16, 18–24, 35, 37, 41, 42, 44–46, 48–50

XHTML Extensible Hypertext Markup Language. 44, 46

XML Extensible Markup Language. 16, 17, 20, 28, 29, 34, 35, 37, 43, 44

Obsah

Kapitola 1

Úvod

Projektové plánování je jednou z nejdůležitějších fází každého projektu. Pokud je plánování provedeno efektivně a pracovníci se účastní vývoje plánu, šance na úspěch se výrazně zvyšují. [1, s. 185]

Existuje celá řada menších projektů, u kterých je lákavé přeskočit plánovací fázi a přejít rovnou na jejich realizaci bez adekvátního řízení. Toto chování může vést k přehlédnutí kritických kroků, narušení časového plánu a potenciálně k nákladným chybám. [2]

1.1 Motivace

Pro zefektivnění projektového řízení a plánování se vyskytuje velké množství placených nástrojů. Menší projekty, které pro plánování a řízení volí vodopádovou metodu a pracují s WBS a Ganttovým diagramem, však ve většině případů nepotřebují a zároveň ani nechtějí investovat do drahého softwaru, který je zpravidla navíc zbytečně komplikovaný. Proto nastává otázka, zda nelze takové projekty podpořit již existujícími volně dostupnými nástroji.

Tato práce se zaměřuje na tvorbu tzv. projektové kanceláře, jejímž základem je sada vzájemně propojených nástrojů a jejich propojení je jedním z hlavních výstupů této práce.

Inspirací pro vytvoření takové projektové kanceláře byly zkušenosti z různých školních projektů, kde byla využita kombinace WBS a Ganttova diagramu, stejně jako z jiných menších projektů, které by mohly těžit z takového řešení.

1.2 Cíl práce

Hlavním cílem této práce je navrhnout a realizovat projektovou kancelář na bázi volně dostupných nástrojů. Tato kancelář bude sloužit jako centralizovaný systém pro podporu procesů souvisejících s plánováním a řízením menších projektů, s cílem zlepšit jejich efektivitu a produktivitu.

Nejprve se definují klíčové pojmy a stanoví se požadavky na projektovou kancelář. To zahrnuje identifikaci potřebných funkcionalit a charakteristik, které musí kancelář splňovat, aby byla schopna podporovat různé aspekty plánování a řízení malých projektů.

Poté proběhne analýza existujících volně dostupných nástrojů, které mohou být využity pro vytvoření projektové kanceláře. Na základě této analýzy bude vybrána optimální kombinace nástrojů, která nejlépe vyhovuje stanoveným požadavkům.

Následně dojde ke spojení vybraných nástrojů do jednoho celku, čímž vznikne jednoduchá projektová kancelář. Tento proces zahrnuje technickou integraci jednotlivých nástrojů tak, aby společně vytvořily koherentní a snadno použitelný systém.

Nakonec bude ověřena funkčnost nově vytvořené projektové kanceláře prostřednictvím systémových a uživatelských testů. Tyto testy zajistí, že kancelář správně funguje a splňuje všechny stanovené požadavky.

1.3 Použité nástroje a metody

Při vypracování této bakalářské práce byly použity různé nástroje a metody k zajištění přesnosti a efektivity. Zejména byly využity následující:

- **ChatGPT**[1] – Pokročilý jazykový model vyvinutý společností OpenAI[2], který poskytl podporu při formulaci textu, návrhu struktury některých kapitol a řešení technických dotazů.
- **Phind**[3] – Specializovaný vyhledávač zaměřený na technické dotazy, který byl použit pro vyhledávání relevantních informací a zdrojů.
- **GitHub Copilot**[4] – Nástroj integrovaný v IntelliJ IDEA[5], který využívá umělou inteligenci k asistenci při psaní kódu, čímž zvyšuje produktivitu a snižuje množství chyb.
- **DeepL**[6] – Překladatelský nástroj, který byl použit pro překlady technické dokumentace a zdrojů využitých v práci.

Využití těchto nástrojů umožnilo efektivní a rychlé získávání informací a řešení problémů, které se v průběhu práce objevily.

1.4 Struktura práce

V 1. kapitole je stručně popsáno, o čem práce pojednává, jaká je její motivace, co je jejím cílem a jaké nástroje byly využity k jejímu vypracování. Kapitola 2 definuje základní pojmy, které jsou zapotřebí k dobrému chápání práce. 3. kapitola popisuje parametry WBS a Ganttova diagramu a vzájemně je porovnává. 4. kapitola seznamuje s definicí a fungováním projektové kanceláře, které je tato práce věnována, prací projektového manažera a analyzuje nástroje pro projektové plánování a řízení. Kapitola 5 obsahuje rešerši formátů jednotlivých nástrojů vybraných v předešlé kapitole. 6. kapitola zkoumá, zda již neexistuje volně dostupné řešení převodu dat mezi WBS a Ganttovým diagramem. V 7. kapitole je rozebrán návrh aplikace, která umožňuje propojit vybrané nástroje do jednoho celku. Kapitola 8 poskytuje analýzu a výběr nástrojů a technologií pro implementaci této práce. V kapitole 9 je představena implementace aplikace. V 10. kapitole je detailně popsán průběh konverze a jeho možné zlepšení do budoucna. Předposlední kapitola 11 se zabývá systémovým a uživatelským testováním aplikace a jeho vyhodnocením. Nakonec shrnuje celou práci 12. kapitola.

[1] `https://chatgpt.com`
[2] `https://openai.com`
[3] `https://www.phind.com`
[4] `https://github.com/features/copilot`
[5] `https://www.jetbrains.com/idea`
[6] `https://www.deepl.com/translator`

Kapitola 2

Definice potřebných pojmů

V této kapitole jsou definovány pojmy spojené s tématem práce. Konkrétně se jedná o pojmy projekt, projektové řízení, Work Breakdown Structure, Ganttův diagram a konverze dat. Porozumění těchto pojmů (hlavně WBS a Ganttova diagramu) bude nedílnou součástí této práce. U definic WBS a Ganttova diagramu jsou nejprve uvedeny definice z českého a poté z anglického zdroje. Obě jsou následně shrnuty společně s jejich ukázkou a důvodem použití.

2.1 Projekt

Projekt je dočasné úsilí vyvinuté k vytvoření unikátního produktu, služby nebo výsledku. Dočasná povaha projektů naznačuje, že projekt má určitý začátek a konec. Konce je dosaženo, když jsou splněny cíle projektu nebo když je projekt ukončen, protože jeho cíle nebudou nebo nemohou být dosaženy, nebo když ho již není potřeba. Slovo dočásný obvykle neplatí pro produkt, službu nebo výsledek vytvořený projektem. Většina projektů je podniknuta za účelem vytvoření trvalého výsledku. [3, s. 3]

Každý projekt je **jediněčný**. I když se v některých činnostech projektu mohou vyskytovat opakující se prvky, tato opakování nemění základní a unikátní charakteristiky práce na projektu. [3, s. 3]

2.2 Projektové řízení

Projektové řízení je aplikace znalostí, dovedností, nástrojů a technik na projektové činnosti s cílem splnit požadavky projektu. Řízení projektu se uskutečňuje prostřednictvím vhodného použití a integrace těchto pěti skupin procesů:

- Zahájení.
- Plánování.
- Provedení.
- Monitorování a řízení.
- Uzavření. [3, s. 5]

Řízení projektu obvykle mimo jiné zahrnuje:

- Identifikaci požadavků.

- Adresování různých potřeb, obav a očekávání stakeholderů[1] při plánování a provádění projektu.
- Zakládání, udržování a provádění komunikace mezi stakeholdery, která je aktivní, efektivní a spolupracujícího charakteru.
- Řízení stakeholderů s cílem splnit požadavky projektu a vytvořit výstupy projektu.
- Vyvažování konkurenčních omezení projektu, mezi něž patří mimo jiné:
  - Rozsah.
  - Kvalita.
  - Harmonogram.
  - Rozpočet.
  - Zdroje.
  - Rizika. [3, s. 6]

Specifické charakteristiky a okolnosti projektu mohou ovlivnit omezení, na která se musí tým pro řízení projektu zaměřit. [3, s. 6]

Vztah mezi těmito faktory je takový, že pokud se změní kterýkoli z nich, je pravděpodobné, že bude ovlivněn alespoň jeden další faktor. Například pokud se zkrátí harmonogram, často je třeba navýšit rozpočet, aby bylo možné přidat další zdroje a dokončit stejný objem práce v kratším čase. Pokud navýšení rozpočtu není možné, může dojít ke snížení rozsahu nebo cílové kvality, aby byl konečný výsledek projektu dosažen v kratším čase při stejné výši rozpočtu. [3, s. 6]

Agile (agilní metodika) a Waterfall (vodopádová metodika) jsou dvě dobře známé metodiky řízení projektů. Obě jsou oblíbené při vývoji softwaru, ale každá z nich se hodí pro jiné typy projektů. Hlavní rozdíl spočívá v tom, že Waterfall je lineární systém práce, který vyžaduje, aby tým dokončil každou fázi projektu předtím, než přejde k další, zatímco Agile podporuje tým, aby pracoval současně na různých fázích projektu. [4]

2.2.1 Vodopádová metodika řízení projektu

Vodopádová metodika je lineární forma řízení projektů, která je ideální pro projekty, u nichž je konečný výsledek jasně stanoven již na začátku projektu. Očekávání od projektu a výstupy každé fáze jsou jasné a jsou nutné pro postup do další fáze. [4]

2.2.2 Agilní metodika řízení projektu

Agilní metodika vznikla jako reakce na rigidnější strukturu vodopádové metodiky. Díky tomu je mnohem plynulejší formou řízení projektů. Dokončení projektu vývoje softwaru může trvat roky a technologie se během této doby může výrazně změnit. Agilní metodika byla vyvinuta jako flexibilní metoda, která vítá začlenění změn směru i v pozdních fázích procesu a také zohlednění zpětné vazby zúčastněných stran v průběhu celého procesu. [4]

Při agilní metodě tým pracuje na jednotlivých fázích projektu souběžně, často s krátkodobými termíny. Navíc směr projektu určuje tým, nikoli projektový manažer. To může posílit motivaci týmu a zvýšit jeho produktivitu, ale také to vyžaduje, aby se tým více řídil sám. [4]

[1] Stakeholder, neboli zainteresovaná strana, je jednotlivec, skupina nebo organizace, která může ovlivnit nebo být ovlivněna rozhodnutím, činností nebo výsledkem projektu. Stakeholdeři mohou být aktivně zapojeny do projektu nebo mohou mít zájmy, které mohou být pozitivně nebo negativně ovlivněny provedením nebo dokončením projektu. [3, s. 30]

2.2.3 Porovnání agilního a vodopádového přístupu k řízení projektů

Hlavní rozdíly mezi agilní a vodopádovou metodikou spočívají v aspektech uvedených v tabulce 2.1. [4]

Parametr	Waterfall	Agile
Práce na projektech	Lineární, jedna fáze po druhé.	Současné práce na různých fázích.
Rozpočet	Pevný od začátku projektu.	Flexibilní, mění se během projektu.
Změny a adaptabilita	Nízká flexibilita, vyžaduje dokončení každé fáze.	Vysoká flexibilita, změny povolovány i v pozdějších fázích.
Zapojení klienta	Omezená, jen při kontrolních bodech nebo předání výstupů.	Častá, průběžná zpětná vazba během celého procesu.
Časový rámec	Pevný od začátku, méně přizpůsobivý.	Flexibilní, přizpůsobený průběhu projektu.

Tabulka 2.1: Porovnání parametrů vodopádové a agilní metodiky.

Tyto rozdíly znamenají, že Waterfall je vhodný pro projekty s jasně definovaným cílem a malou pravděpodobností změn, zatímco Agile je lepší volbou pro projekty, které mohou vyžadovat časté úpravy a experimentování s různými směry. [4]

2.3 Work Breakdown Structure (WBS)

WBS je nedílnou součástí životního cyklu a časového plánu projektu. Jedná se o hierarchii úkolů a úrovní (viz obrázek 2.1), vizualizovanou (nejlépe) pomocí stromového diagramu, jež pomáhá projektovému manažerovi rozpadnout projekt na menší měřitelné části. [5, s. 683][6]

2.3.1 Definice

Na webové stránce ManagementMania se nachází tato definice: „*WBS (Work Breakdown Structure), překládá se jako rozpad, rozpis práce nebo jako osnova rozpisu práce, často se používá zkratka WBS. Jedná se o jednoduchou analytickou techniku, jejímž cílem je rozložit projekt na jednotlivé činnosti až do takové úrovně podrobnosti, aby k nim bylo možné přiřadit odpovědnosti, pracnost a časový horizont.*“ [7]

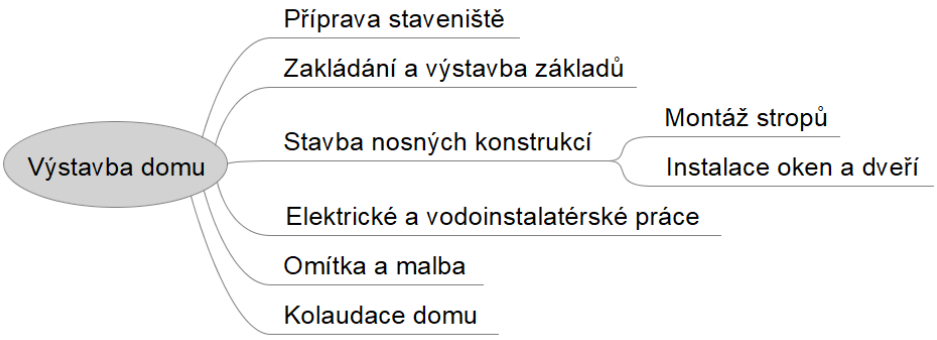
V knize Practice Standard for Work Breakdown Structures (3rd Edition) je uvedeno: „*A WBS is a hierarchical decomposition of the total scope of work to be carried out by the project team to accomplish the project objectives and create the required deliverables. Whereas the project scope statement describes the project scope and its major deliverables, assumptions, and constraints, the WBS elaborates on this description by defining, and hierarchically organizing, the total scope of the project. The WBS represents the entirety of the work specified in the current approved project scope.*“ [8, s. 3]

Z obou definic vyplývá, že WBS je dekompozice celkového rozsahu prací, které má projektový tým provést, aby dosáhl cílů projektu a vytvořil požadované výstupy. Díky WBS je možné přiřadit odpovědnosti, pracnost a časový horizont jednotlivým úkolům na nejnižší úrovni.

Zjednodušeně řečeno, WBS pomáhá převést celý rozsah projektu na řadu menších složek nazývaných „work packages“, které již mohly být řešeny v minulosti, což usnadňuje jejich realizaci, měřitelnost a říditelnost. [8, s. 3]

Hlavní odpovědnost za vytvoření WBS má projektový manažer. [5, s. 683]

WBS se nejčastěji zobrazuje pomocí stromového diagramu [6]. Dalo by se také říci, že z pohledu rozpadu je WBS speciálním typem myšlenkové mapy[2] (viz obrázek 2.1).



Obrázek 2.1: Ukázka WBS formou myšlenkové mapy.

Na obrázku 2.1 je vidět hierarchický rozpad projektu zleva doprava, to je uzpůsobeno vizualizací WBS pomocí myšlenkové mapy. Je však nutné podotknout, že ve většině případů se rozpad zobrazuje ze shora dolů.

2.3.2 Důvod použití

WBS pomáhá s přesnou organizací projektů. Vypomáhá s rozdělováním povinností. Jasně udává kontrolní body a milníky projektu. Díky WBS se lépe plánují náklady, rizika a časové rozpětí projektu. Celkově slouží k dobrému naplánování a lepší zvladatelnosti projektu. [5, s. 683]

2.4 Ganttův diagram

Ganttův diagram je druh sloupcového grafu pojmenovaný po H.L. Ganttovi, který tento diagram propagoval v průběhu první světové války. Slouží k vizualizaci posobějdoucích činností v projektu v čase. [9, s. 131]

2.4.1 Definice

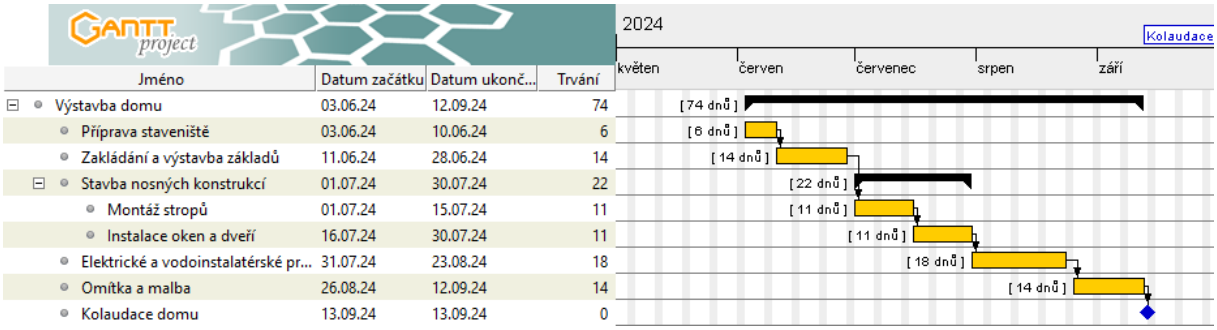
Na webové stránce ManagmentMania pojem definují jako: „*Ganttův diagram (Gantt Chart) je prakticky synonymem pro grafické znázornění naplánované posloupnosti činností v čase, které se využívá při řízení projektů nebo programů. Duchovním otcem je Henry Laurence Gantt. Ganttův diagram zobrazuje ve sloupcích (horizontálně) časové období ve kterém se plánuje. Podle délky plánovaného projektu se zobrazuje období v odpovídající podrobnosti (roky, měsíce, týdny, dny). V řádcích (vertikálně) se pak zobrazují dílčí aktivity (někdy nazývány jako úkoly) – tedy kroky, činnosti nebo podprojekty a to v takovém pořadí, které odpovídá jejich logickému sledu v plánovaném projektu. Délka trvání dané aktivity je pak vztažena k časovému období.*“ [11]

V knize Production Planning and Control – A Comprehensive Approach je Ganttův diagram popsán takto: „*Gantt charts are bar charts that illustrate the start and finish dates of the terminal elements and summary elements of a project showing activities (tasks or events) displayed against time. In a Gantt chart, the activities are listed one below the other on the left, and each activity is represented by a bar on a time scale with the position and length of the bar reflecting the start date, duration, and end date of the activity.*“ [12, s. 363]

[2] Myšlenková mapa je grafické znázornění úkolů, nápadů nebo myšlenek. Skládá se z ústředního úkolu, na který navazují další větve, které se rozšiřují na související dílčí úkoly. Každou větev lze dále rozdělit na menší podvětve, čímž vzniká hierarchická struktura, která umožňuje snadnou organizaci úkolů nebo myšlenek. [10]

Z obou definic je vidět, že Ganttův diagram je grafické znázornění plánovaných činností v čase, používané při řízení projektů. Vodorovné sloupce reprezentují časová období, zatímco vertikální řádky zobrazují dílčí aktivity v logickém sledu. Aktivity jsou znázorněny pruhy na časové ose, přičemž jejich délka znázorňuje jejich začátek, konec a plánovanou délku trvání (viz obrázek 2.2).

Ve většině případů se Ganttův diagram využívá pro plánování a řízení projektu, kde jsou uvedeny všechny plánované činnosti a data jejich začátku a dokončení. Ganttův diagram lze ale také využít pro plánování a kontrolu výroby (PP&C), kde jsou vyzobrazeny jednotlivé produkty a jejich počet. [12, s. 364]



Obrázek 2.2: Ukázka Ganttova diagramu.

2.4.2 Důvod použití

Ganttův diagram pomáhá projektovému manažerovi s plánováním a řízením projektů. Díky prvotnímu odhadu je poté možné sledovat, jak projekt probíhá a jak se drží definovaného plánu. Ganttův diagram umožňuje vizualizovat řadu věcí:

- Jednotlivé aktivity.
- Hierarchii aktivit.
- Kdy jednotlivé aktivity začínají a kdy končí.
- Plánovanou délku trvání jednotlivých aktivit.
- Kde a jak moc se aktivity překrývají s ostatními aktivitami.
- Datum začátku projektu a datum konce projektu. [12, s. 363-364]

2.4.3 Vazba mezi WBS a Ganttovým diagramem

WBS a Ganttův diagram hrají obě klíčové role v plánování projektu. WBS ukazuje, co projekt dosáhne nebo postaví, včetně rozsahu projektu, zatímco Ganttův diagram vizuálně ilustruje časovou osu aktivit projektu pomocí pruhů. Oba nástroje jsou důležitými součástmi úspěšného dokončení projektu, protože úspěch závisí na jasné definici cílů a termínu dokončení. Cíle a harmonogram projektu ovlivňují náklady na projekt a potřeby pracovních sil, takže je důležité je správně stanovit. [13]

Jak WBS přispívá k tvorbě Ganttova diagramu

Při začátku plánování projektu slouží WBS jako výchozí bod pro tvorbu Ganttova diagramu. Tyto dva nástroje pomáhají manažerům vytvořit projektový plán. [13]

WBS rozděluje práci hierarchicky, což je základem pro vytvoření Ganttova diagramu. Každý prvek WBS může být převeden na úkol v Ganttově diagramu. [13]

Rozdíl mezi WBS a Ganttovým diagramem

Hlavní rozdíly a vzájemné porovnání WBS a Ganttova diagramu je popsáno a ukázáno v následující kapitole (viz kapitola 3).

2.4.4 Odůvodnění výběru Ganttova diagramu

Výběr Ganttova diagramu oproti Kanban desce (*Kanban board*)[3] pro tuto práci je odůvodněn několika klíčovými faktory. Prvním z nich je, že výsledná projektová kancelář se zaměřuje na menší projekty řízené vodopádovou metodou. Tato metoda řízení projektů předpokládá, že každý fázový průchod musí být dokončen předtím, než se přesune do následující fáze (viz definice v podsekci 2.2.1). Ganttův diagram je ideální pro takové projekty, protože umožňuje vizualizovat časový rámec a závislosti mezi různými úkoly, což je klíčové pro efektivní plánování a sledování pokroku v rámci vodopádových projektů. [14]

Dalším důvodem je, že Ganttův diagram podporuje pohled na projekt jako celek, nikoli jen na jednotlivé úkoly. To je důležité pro projekty, které vyžadují koordinaci souvisejících úkolů a kde je nezbytné mít jasný přehled o tom, jak tyto úkoly navzájem souvisí a jaké jsou jejich časové závislosti. Díky tomu je možné lépe plánovat a řídit projekty, zejména ty, které vyžadují pečlivé řízení časových limitů. [14]

Zatímco Kanban desky jsou excelentní pro sledování a řízení jednotlivých úkolů v agilních projektech, kde je hlavní důraz kladen na rychlou iteraci a adaptivitu (viz definice v podsekci 2.2.2), Ganttův diagram nabízí unikátní výhodu v kontextu vodopádových projektů a projektů vyžadujících celkový náhled. Jeho schopnost zobrazovat časový rámec a závislosti mezi úkoly umožňuje týmům lépe rozumět struktuře projektu a efektivněji plánovat a koordinovat svou práci. [14]

Takže, i když Kanban desky mohou být velmi užitečné pro některé typy projektů, volba Ganttova diagramu pro menší projekty řízené vodopádovou metodou a vyžadující celkový náhled na projekt je logická a odůvodněná.

2.5 Konverze dat

Datová konverze je proces transformace dat z jednoho formátu do druhého tak, aby byl kompatibilní s cílovým systémem nebo aplikací. Často se provádí jako součást většího projektu, jako je migrace dat nebo integrace. Proces zahrnuje extrakci dat ze zdroje, jako je databáze, soubor nebo webová služba, jejich transformaci a načtení do požadovaného cílového systému nebo souboru. [15]

Proces konverze dat je jedinečný a závisí na konkrétních potřebách projektu. Na základě počtu a složitosti použitých datových formátů mohou být některé převody dat přímočaré a relativně jednoduché, zatímco jiné mohou být složitější. Konkrétní operace a transformace se také výrazně liší projekt od projektu. [15]

Komplexní a efektivní proces konverze dat by měl:

- Transformovat data do kompatibilního formátu pro cílový systém nebo aplikaci.
- Minimalizovat ztrátu dat během přenosu.
- Udržovat kvalitu dat, čitelnost a integritu.
- Zajišťovat konzistenci napříč všemi systémy. [15]

[3] Kanban (v překladu „vývěsní štít“ nebo „nástěnka“) je agilní metoda pro správu a zlepšování práce, která využívá vizualizaci pracovních položek na nástěnce (Kanbanu) k efektivnímu řízení toku práce a zlepšení efektivnosti procesů. [21]

2.5.1 Konvertor

Vzhledem k tomu, že nebyl nalezen zdroj popisující, co je konvertor, bylo po konzultaci s vedoucím práce sestavena následující definice: Konvertor je nástroj nebo software, který umožňuje konverzi dat (viz sekce 2.5). Správný konvertor by měl být bezestavový, tj. konverze dat probíhá v reálném čase, bez nutnosti uložení dat do mezipaměti nebo jiného typu úložiště před konverzí. Mezi hlavní výhody bezestavové aplikace patří:

- **Škálovatelnost** – Bezestavové aplikace jsou obecně více škálovatelné, protože každý požadavek je nezávislý a může být zpracován jakýmkoli dostupným serverem.
- **Nižší využití zdrojů** – Bezestavové aplikace často mají nižší využití zdrojů, protože není potřeba ukládat a spravovat data vzniklá při relaci.
- **Odolnost vůči chybám** – Bezestavové aplikace mohou být odolnější vůči chybám, protože ztráta serveru neovlivňuje uživatelské relace. [16]

Kapitola 3

Porovnání WBS a Ganttova diagramu

V minulé kapitole (viz sekce 2.3 a 2.4) bylo naznačeno, že jádrem naší projektové kanceláře jsou WBS a Ganttův diagram, proto si jejich vazbu v této kapitole popíšeme podrobněji – představíme si jejich základní parametry a následně si je porovnáme. Z důvodu nenalezení žádného zdroje, který by přímo stanovoval pravidla podoby alespoň jednoho diagramu, byly tyto parametry vybrány a spojeny z různých zdrojů a obohaceny o vlastní znalosti.

3.1 Parametry WBS

Formát WBS není jednoznačně definován. Nejdůležitějším prvkem tohoto diagramu jsou tzv. dodávky (*deliverables*) – jednotlivé prvky WBS. Dodávky rozpadáme tak dlouho, dokud úkoly na nejnižší úrovni neobsahují fyzicky předatelné výstupy. Tyto úkoly obvykle nazýváme pracovními balíky a lze je následně věrohodně ocenit (dá se odhadnout práce nutná na jejich realizaci, náklady, čas atd.) [17].

WBS je nejčastěji znázorňován jako stromový diagram, resp. myšlenková mapa, (viz definice v sekci 2.3) s kořenem představujícím celý projekt a větvemi reprezentujícími jednotlivé úkoly a podúkoly. Každý úkol v rámci WBS je jasně zařazen do hierarchické struktury, kde každý prvek má své specifické místo v rámci celkového schématu projektu. Tyto úkoly jsou systematicky propojeny, což znamená, že existuje jasná linie návaznosti mezi rodičovskými a dceřinými úkoly. Navíc, každý úkol je přiřazen k jedné a pouze jedné úrovni v této hierarchii, což zajišťuje jeho jednoznačnou pozici a roli v rámci celkového plánu projektu. Jeden úkol může být podřízen pouze jednomu nadřazenému úkolu (rodiči), avšak může mít libovolný počet podřízených úkolů (potomků).

3.2 Parametry Ganttova diagramu

V pátém vydání PMBOK[1], popisují Ganttův diagram jako sloupcový graf, který má jednotlivé aktivity na vertikální ose, data jsou zobrazeny na ose vodorovné a doby trvání jednotlivých aktivit jsou znázorněny jako vodorovné obdélníkové pruhy. Levá hrana obdélníku popisuje začátek, pravá konec a obsah obdélníku délku trvání aktivity. [3, s. 542]

Spojením této definice s obsahem webové stránky *Základní průvodce Ganttovými diagramy* v sekci *Obchodní přehledy a nápady* od Microsoftu, kde je popsána struktura Ganttova diagramu, [18] a znalostmi nabitými při studiu vzniká seznam parametrů, které by měl každý plnohodnotný Ganttův diagram obsahovat:

- Počátek projektu

[1] Mezinárodně uznávaný standard řízení projektů, který vydává institut PMI. [19]

- Úkoly – s počátkem a koncem nebo případně délkou trvání
- Milníky
- Seskupení úkolů – úkol, který pod sebou má další úkoly
- Logické závislosti mezi úkoly
- Zdroje

Úplný Ganttův diagram by měl také mít zastřešující úkol celého projektu, ze kterého se dá jednoduše zjistit jeho začátek, konec a délka trvání.

3.2.1 Logické závislosti mezi úkoly

Na rozdíl od WBS je možné stromovou strukturu a návaznosti mezi úkoly lépe strukturovat. Slouží k tomu tzv. logické závislosti. V Ganttově diagramu se udávají čtyři základní typy logických závislostí:

- **Finish to Start (Konec na začátek)** – Navazující úkol nemůže začít, dokud předcházející úkol neskončí.
- **Start to Start (Začátek na začátek)** – Navazující úkol nemůže začít, dokud předcházející úkol nezačne.
- **Finish to Finish (Konec na konec)** – Navazující úkol nemůže skončit, dokud předcházející úkol neskončí.
- **Start to Finish (Začátek na konec)** – Předchozí úkol může být dokončen pouze poté, co navazující úkol začal. [20]

3.3 Vzájemné porovnání diagramů

WBS i Ganttův diagram jsou nezbytné pro efektivní plánování a řízení projektů. WBS pomáhá definovat strukturu a rozsah projektu, zatímco Ganttův diagram umožňuje sledování a řízení časového rámce projektu. Spolupráce těchto dvou nástrojů zajišťuje, že projekt bude dobře naplánován a úspěšně realizován. [13]

WBS se během projektu obvykle významně nemění, zatímco Ganttův diagram je pravidelně aktualizován na základě aktuálního pokroku projektu. [13]

Tabulka 3.1 ilustruje hlavní rozdíly mezi WBS a Ganttovým diagramem. [13]

Parametr	WBS	Ganttův diagram
Účel	Rozdělení celkového rozsahu projektu na menší, snadno spravovatelné části. Ukazuje, co projekt vytvoří, včetně rozsahu projektu.	Vizualizace časového rámce projektu. Ukazuje, kdy budou jednotlivé úkoly prováděny, včetně jejich závislostí a pořadí.
Struktura	Stromová struktura s kořenem představujícím celý projekt, který se postupně dělí na menší úkoly a podúkoly.	Horizontální pruhy na časové ose zleva doprava pro každý úkol.
Závislosti	Nejsou explicitně zobrazeny.	Zobrazuje logické závislosti mezi úkoly.
Použití	Nástroj pro plánování projektu, který pomáhá definovat cíl a rozsah projektu.	Nástroj pro plánování a řízení projektu, který umožňuje projektovému manažerovi a týmu sledovat aktuální stav projektu v porovnání s původním plánem.

Tabulka 3.1: Vzájemné porovnání Ganttova diagramu a WBS.

Na obrázku 3.1 jsou pod sebou ukázány oba diagramy (WBS nahoře, Ganttův diagram dole). Ganttův diagram na ukázce vznikl převedením z WBS. Je vidět, že převod zachoval jména úkolů a jejich hierarchickou strukturu, byly však dodány další informace, jako je časové rozpětí jednotlivých úkolů i projektu, vizualizace projektu v čase, přidání logických závislostí a přetvoření úkolů na milníky. Červené šipky označené čísly znázorňují hlavní rozdíly mezi diagramy:

- **1** – Z kořene stromového diagramu, resp. myšlenkové mapy, WBS se stal zastřešující prvek celého projektu v Ganttově diagramu. Dá se z něho jednoduše vyčíst jeho začátek, konec a délka trvání.
- **2**, **3** – Označené úkoly poukazují na zachování hierarchické struktury. Ke každému úkolu bylo doplněno datum začátku, datum konce, jeho délka trvání a logické závislosti mezi úkoly. Z úkolu, který pod sebou má další úkoly (šipka 2), se stal zastřešující prvek. Jeho datum začátku je shodné s datem začátku nejdříve započatého úkolu v něm a jeho datum konce je naopak shodné s datem konce nejzazšího úkolu v něm.
- **4** – Z úkolu označeného 4. šipkou se stal milník, který má nulovou dobu trvání a označuje významný bod v projektu.

Obrázek 3.1: Vzájemné porovnání WBS a Ganttova diagramu.

Obě struktury a jejich vzájemná provázanost jsou základem námi tvořené projektové kanceláře a nástrojů, jak je popsáno v dalších kapitolách.

Kapitola 4

Projektová kancelář a její role

Projektová kancelář (Project Management Office (PMO)) je organizační jednotka, která poskytuje podporu, metodologie, nástroje a techniky pro řízení projektů v organizaci. Hlavním cílem PMO je zajistit, aby projekty byly prováděny efektivně a byly v souladu se strategickými cíli organizace. Projektová kancelář podporuje projektové manažery při plánování a řízení projektů. [3, s. 10]

Tato práce se zaměřuje na návrh a implementaci projektové kanceláře, která bude optimalizována pro menší projekty využívající vodopádovou metodu, které chtějí minimalizovat náklady na jejich plánování a řízení, a proto používají volně dostupné nástroje. Hlavním cílem této projektové kanceláře bude poskytnout nástroj podporující WBS a Ganttův diagram a přechod mezi nimi.

4.1 Role a zodpovědnosti projektového manažera

Projektový manažer je člověk, který zodpovídá za plánování, realizaci a uzavření projektu. Klíčové činnosti zahrnují definici rozsahu projektu, vytváření harmonogramu, řízení rizik, komunikaci s týmem a zainteresovanými stranami, sledování pokroku a zajištění dodržení termínů a rozpočtu. K tomu mu mimo jiné pomáhají projektové kanceláře. [3, s. 16-17]

4.2 Typy projektové kanceláře

Existují tři hlavní typy projektových kanceláří v organizacích, které se liší mírou kontroly a vlivu na projekty v rámci organizace:

- **Podpůrné (Supportive)** – Poskytuje konzultativní roli pro projekty tím, že dodává šablony, osvědčené postupy, školení, přístup k informacím a poznatky získané z jiných projektů. Tento typ PMO slouží jako úložiště projektových informací. Míra kontroly poskytovaná touto PMO je nízká.
- **Kontrolní (Controlling)** – Poskytuje podporu a zároveň vyžaduje dodržování určitých pravidel a postupů. Dodržování může zahrnovat přijetí projektových metodik, používání specifických šablon, formulářů a nástrojů nebo dodržování řídících předpisů. Míra kontroly poskytovaná touto PMO je střední.
- **Řídící (Directive)** – Přebírá kontrolu nad projekty tím, že je přímo řídí. Míra kontroly poskytovaná touto PMO je vysoká. [3, s. 10-11]

Výstup této práce bude spadat do podpůrné kategorie.

4.3 Nástroje projektového řízení

Pro efektivní řízení projektů pomocí vodopádové metody jsou využívány nejrůznější nástroje. Dvojicí z nich jsou Work Breakdown Structure (WBS) a Ganttův diagram, které pomáhají s rozpadem projektu na menší úkoly a při plánování, sledování a koordinaci jednotlivých úkolů a zdrojů.

Protože se tato práce zaměřuje na projektovou kancelář pro menší projekty, analyzujeme a popisujeme si v této sekci **volně dostupné** nástroje pro podporu projektového řízení s důrazem na zmíněné WBS a Ganttův diagram.

4.3.1 Analýza nástrojů pro podporu projektového řízení

V této podsekci si uvedeme volně dostupné nástroje pro plánování a řízení projektů, které využívají WBS a Ganttův diagram, nalezené při analýze trhu s těmito nástroji. Uvedené nástroje umožňují plánování a sledování projektů bez nutnosti vysokých investic do softwaru, což je náš základní požadavek.

Analýza byla provedena 16. května 2024.

Porovnání a vyhodnocení nástrojů

V následujících bodech jsou uvedeny tři bezplatné nebo open-source nástroje, které nejlépe splňují naše požadavky a které mohou pomoci s plánováním a řízením menších projektů vodopádovou metodou:

- **GanttProject**[1] – Bezplatná desktopová aplikace pro plánování a řízení projektů, která běží na Windows, Linux a macOS. Umožňuje jednoduchou práci s Ganttovým diagramem a zdroji projektu. Bohužel nepodporuje práci s WBS. Existuje i cloudová verze této aplikace, která je ale placená.

- **ProjectLibre**[2] – Projekt, který o sobě říká, že je nejlepší alternativou Microsoft Project[3] a je s ním i kompatibilní. Stejně jako GanttProject poskytuje aplikaci pro tvorbu Ganttova diagramu, která je dostupná jak v desktopové verzi (pro Windows, Linux a macOS), tak ve verzi cloudové, ta je bohužel placená. Z vytvořeného Ganttova diagramu následně dokáže udělat WBS, vypočítat náklady na jednotlivé pracovníky i celkové náklady na projekt.

- **Microsoft Excel šablony pro tvorbu Ganttova diagramu**[4] – Tyto šablony umožňují tvorbu a vizualizaci projektu ve formátu Ganttova diagramu v nástroji Microsoft Excel[5]. Oproti již jmenovaným nástrojům neposkytuje prakticky žádné funkce a jedinou odlišností je nástroj, který využívá.

Další nástroje jako TeamGantt[6], Redbooth[7] nebo Smartsheet[8] poskytují ve směs stejnou službu. Jedná se o cloudové nástroje, u kterých existují i bezplatné verze. Ty však mají zásadní nevýhody oproti již zmíněným nástrojům – je u nich velmi omezen počet nabízených vlastností.

Po důkladné analýze trhu byly nalezeny nástroje, které k plánování projektu využívají tvorbu a správu Ganttova diagramu, WBS však nikoli. Proto bylo potřeba najít alternativní nástroj, který jeho tvorbu umožňuje a ze kterého pak bude možné přenést vytvořené úkoly do některého z již zmíněných nástrojů a tím zautomatizovat manuální přechod z WBS do Ganttova diagramu.

[1] `https://www.ganttproject.biz`
[2] `https://www.projectlibre.com`
[3] `https://www.microsoft.com/cs-cz/microsoft-365/project/project-management-software`
[4] `https://create.microsoft.com/en-us/templates/gantt-charts`
[5] `https://www.microsoft.com/cs-cz/microsoft-365/excel`
[6] `https://www.teamgantt.com`
[7] `https://redbooth.com`
[8] `https://smartsheet.com`

Vzhledem k tomu, že WBS nemá jednoznačně určený formát (viz sekce 3.1), dá se k jeho reprezentaci využít myšlenková mapa. Hledání bezplatných nástrojů poskytujících tvorbu a správu myšlenkových map dosáhlo většího úspěchu než při předešlé rešerši. Uvedeme a popíšeme si tři nástroje nalezené na trhu s nimi, které splňují naše požadavky:

- **FreeMind**[9] – Bezplatná desktopová aplikace, která běží na Javě, s širokou uživatelskou bází. FreeMind je dostupný na Windows, Linux a macOS. Nevýhodou může být občasná nestabilita při dlouhodobém používání a starší vzhled.
- **WiseMapping**[10] – Online open-source nástroj. Uživatelé WiseMapping mohou spolupracovat na vytváření myšlenkových map v reálném čase a sdílet je s ostatními. Má však omezenou funkčnost, když je uživatel neregistrován. Hlavní výhodou je dostupnost.
- **MindMeister**[11] – MindMeister umožňuje uživatelům vytvářet interaktivní myšlenkové mapy, přidávat odkazy, obrázky a další obsah. V neplacené verzi dovoluje vytvořit maximálně tři mapy.

4.3.2 Výsledek analýzy

Na základě analýzy byly vybrány tři volně dostupné nástroje, které je možné využít pro plánování a následné řízení menších projektů využívajících vodopádovou metodu. Nebyl však nalezen nástroj, který by zároveň umožňoval vytvářet Ganttův diagram a WBS. Proto se analýza ještě zaměřila na nástroje, které tvoří myšlenkové mapy, protože myšlenkové mapy představují specifickou formu WBS a mohou být použity pro jeho tvorbu. Od těchto nástrojů byly rovněž vybrány tři volně dostupné.

Abychom mohli určit nástroje, které bude námi tvořená projektová kancelář využívat a spojí je do jednoho celku, budeme muset vybrané kandidáty dále analyzovat, konkrétně formát jimi vytvářených souborů. Jejich analýze se věnuje další kapitola (viz kapitola 5).

[9] `https://freemind.sourceforge.io/wiki/index.php/Main_Page`
[10] `https://www.wisemapping.com`
[11] `https://www.mindmeister.com`

Kapitola 5

Analýza vybraných nástrojů

V minulé kapitole (viz sekce 4.3) byly předvybrány tři nástroje pro plánování a řízení projektů pomocí Ganttova diagramu a tři pomocí WBS. Po důkladné analýze jimi tvořených souborů, jejich přípon a možností převodu mezi nimi, byl v této kapitole tento výběr zúžen na jeden nástroj od každého. Z těchto dvou vybraných nástrojů se bude skládat naše projektová kancelář.

U jednotlivých nástrojů se zaměříme hlavně na to, aby nástroj uměl soubory vytvářet a číst ve formátu, který bude dobře strojově zpracovatelný. Celé soubory v textové podobě, jejichž zkrácené ukázky jsou ukázány v této kapitole, jsou přiloženy v elektronické příloze. Rešerše byla provedena 18. května 2024.

5.1 Nástroje využívající Ganttův diagram

5.1.1 GanttProject

Desktopový nástroj GanttProject[1] používá na vytváření a ukládání Ganttova diagramu XML soubor s příponou `.gan`. Soubor je dobře strojově čitelný, zobrazuje hierarchii, není obsáhlý a na první pohled je jasné, co jednotlivé značky znamenají.

Zkrácený příklad souboru tvořeného nástrojem GanttProject je zobrazen na ukázce 5.1.

```
<project name="..." view-date="2024-05-01" version="3.1.3100" locale="cs_CZ">
    <view id="gantt-chart">...</view>
    <view id="resource-table">...</view>
    <calendars>
        <day-types>...</day-types>
    </calendars>
    <tasks>
        <taskproperties>...</taskproperties>
        <task id="1" name="..." start="2024-06-03" duration="...">
            <task id="4" name="..." start="2024-07-01" duration="...">
                <task id="6" name="..." start="2024-07-16" duration="..."/>
            </task>
        </task>
    </tasks>
</project>
```

Ukázka kódu 5.1: Ukázka souboru vytvořeného nástrojem GanttProject.

5.1.2 Project Libre

Následuje desktopový nástroj Project Libre[2], který na vytváření a ukládání Ganttova diagramu využívá XML soubor s příponou `.pod`. Soubor však oproti souboru tvořeného GanttProjectem

[1] https://www.ganttproject.biz/
[2] https://www.projectlibre.com

není příliš čitelný a je obsáhlý i při velmi malém počtu vytvořených úkolů. I přes velký počet elementů jsou jednotlivé elementy dobře popsané.

Na ukázce 5.2 je ukázán zkrácený příklad souboru tvořeného nástrojem Project Libre. Nástroj Project Libre na rozdíl od nástroje GanttProject využívá místo atributů u jednotlivých elementů samostatné elementy. To je hlavní faktor, který dělá soubor tvořený nástrojem Project Libre o tolik obsáhlejší než soubor tvořený nástrojem GanttProject.

```
... long encoded text ...
<Project xmlns="http://schemas.microsoft.com/project">
    ...
    <Calendars>
        <Calendar>
            <UID>1</UID>
            <Name>Standard</Name>
            <IsBaseCalendar>1</IsBaseCalendar>
            <WeekDays>...</WeekDays>
        </Calendar>
        <Calendar>...</Calendar>
        <Calendar>...</Calendar>
    </Calendars>
    <Tasks>
        <Task><UID>1</UID>
            <ID>1</ID>
            <Name>...</Name>
            <Type>0</Type>
            ... lots of parameters ...
        </Task>
        <Task>... lots of parameters ...</Task>
        <Task>... lots of parameters ...</Task>
    </Tasks>
    ...
</Project>
```

Ukázka kódu 5.2: Ukázka souboru vytvořeného nástrojem Project Libre.

5.1.3 Miscrosoft Excel šablony pro tvorbu Ganttova diagramu

Ganttův diagram vytvořený pomocí šablony v Miscrosoft Excel[3] je vytvářen a ukládán do `.xlsx` souboru. Vzhledem k tomu, že tento typ souboru není na bázi XML, je obtížné ho analyzovat a dál s ním pracovat.

5.1.4 Finální výběr nástroje

Prvním volně dostupným nástrojem pro aplikaci projektové kanceláře byl zvolen GanttProject, který vytváří soubor s příponou `.gan`. Hlavním důvodem jeho výběru byla schopnost vytvářet snadno srozumitelné soubory, které nejsou příliš rozsáhlé a lze je snadno strojově zpracovat.

5.2 Nástroje využívající WBS

5.2.1 FreeMind

Desktopový nástroj FreeMind[4] používá na vytváření a ukládání XML soubor s příponou `.mm`, který je velice dobře čitelný. Obsahuje pouze nejnutnější elementy a není proto obsáhlý. Další výhodou nástroje je velmi dobře zpracovaná dokumentace.

Zkrácený příklad souboru tvořeného nástrojem FreeMind je zobrazen na ukázce 5.3.

[3] `https://create.microsoft.com/en-us/templates/gantt-charts`
[4] `https://freemind.sourceforge.io/wiki/index.php/Main_Page`

```
<map version="1.0.1">
    <node ID="ID_1541775419" TEXT="...">
        <node ID="ID_762113780" POSITION="right" TEXT="..."/>
        <node ID="ID_1561781730" POSITION="right" TEXT="...">
            <node ID="ID_1561781732" POSITION="right" STYLE="fork" TEXT="..."/>
        </node>
    </node>
</map>
```

Ukázka kódu 5.3: Ukázka souboru vytvořeného nástrojem FreeMind.

5.2.2 WiseMapping

Dalo by se říct, že online nástroj WiseMapping[5] je na bázi FreeMindu. Na to poukazuje i podobné ovládání nástroje a možnost importu a exportu stejného typu souboru s příponou `.mm`. Proto se dá tento nástroj využít pro online vytváření a úpravu FreeMind souborů.

5.2.3 MindMeister

Online nástroj MindMeister[6] v bezplatné verzi umožňuje pouze export do `.rtf` formátu a nedovoluje import. Tento nástroj tedy nedává, kvůli nesplnění základních požadavků, smysl dále analyzovat.

5.2.4 Finální výběr nástroje

Druhý volně dostupný nástroj, který bude použit pro aplikaci projektové kanceláře, je FreeMind vytvářející soubor s příponou `.mm`. Hlavním důvodem jeho výběru je dobrá čitelnost a jednoduchost souboru. Nástroj WiseMapping dovoluje pracovat se stejným typem souboru, proto se může rovněž využívat. Aby se udržela desktopová rovina, bude se v této práci dále pracovat pouze s nástrojem FreeMind.

5.3 Výsledek analýzy

Analýza formátů souborů vybraných volně dostupných nástrojů pro Ganttův diagram a WBS vedla k výběru dvou nástrojů. Z těchto nástrojů se bude skládat naše projektová kancelář, spojí je do jednoho celku a usnadní tak konverzi dat mezi nimi. Pro tvorbu a ukládání Ganttova diagramu byl vybrán GanttProject díky jeho schopnosti vytvářet snadno srozumitelné soubory. Pro tvorbu a ukládání WBS byl vybrán FreeMind pro svou dobrou čitelnost a jednoduchost souboru.

Výsledkem této a předchozí kapitoly (viz kapitola 4) je skutečnost, že se bude výsledná projektová kancelář skládat ze dvou vzájemně propojených nástrojů. Jejich propojení, resp. aplikace konvertoru, bude založeno na konverzi formátů souborů vybraných nástrojů v této kapitole. Podrobný rozbor, jak bude výsledné řešení vypadat, je popsán v kapitole 7. Nejprve však budeme muset prozkoumat, zdali již neexistuje nástroj, který umožňuje konverzi mezi WBS a Ganttovým diagramem a kterým bychom se mohli inspirovat. Touto analýzou se zabývá následující kapitola (viz kapitola 6).

[5] https://www.wisemapping.com/en
[6] https://www.mindmeister.com

Kapitola 6

Analýza existujících řešení konverze

Tato kapitola se zaměřuje na hledání nástroje pro konverzi z WBS do Ganttova diagramu[1], který je volně dostupný a umožňuje import a export souborů. Výzkum byl proveden prostřednictvím různých vyhledávačů a databází uvedených v následujících sekcích.

6.1 Google Chrome

Bylo zapotřebí vyhledat již existující aplikace, proto byl zvolen webový prohlížeč Google Chrome[2] jako prvotní vyhledávací nástroj. Rešerše byla provedena dne 19. listopadu 2023.

První dotaz byl zvolen velmi přesný – „Převodník z WBS do Ganttova diagramu“. Tento dotaz však našel pouze velké množství placených aplikací na tvoření Ganttova diagramu. Když byla v dotazu vynechána „výplňová“ slova – „Převodník WBS Ganttův diagram“, byl výsledek bohužel prakticky identický a tedy nepřínosný. Nakonec byl použit dotaz „WBS převodník“. I v tomto případě však nebyly získány žádné použitelné zdroje. Další zobecňování dotazu vzhledem k účelům analýzy nedávalo smysl.

Dotaz v anglickém jazyce „WBS to Gantt chart convertor“ přinesl již uspokojivější výsledky. V Lucidchart[3] Product Questions na otázku, jak konvertovat WBS na Ganttův diagram, bylo odpovězeno, že to v aplikaci aktuálně nelze. V blogu, kde porovnávají WBS s Ganttovým diagramem, byla nalezena otázka „How to convert a WBS to a Gantt chart?“. Došlo k zjištění, že nástroj GanttPRO[4] umí importovat WBS soubory z Microsoft Excel. Připravený soubor však musí být v přesně daném formátu a obsahovat všechna potřebná data. Uživatel si pak musí každý sloupec synchronizovat, tzn. nastavit datový typ sloupce. Aplikace je navíc placená. [22] Další zajímavá aplikace, na kterou se narazilo, byla PlantUML[5], která pomocí jednoduché syntaxe umožňuje vizualizaci poznámek. Aplikace je zdarma, převod z WBS na Ganttův diagram je však zcela manuální a pro tuto práci tedy nepřínosný. [23]

Při hledání nebyl nalezen žádný nástroj, který by byl volně dostupný a ulehčoval by převod mezi diagramy.

6.2 Vyhledávač Summon

Analogicky bylo postupováno i u Vyhledávače Summon[6]. Zde však nebyl nalezen žádný validní výsledek. Rešerše byla provedena 3. prosince 2023.

[1] Opačný směr konverze není z implementačního hlediska příliš zajímavý, protože se jedná pouze o přenesení hierarchické struktury a jmen úkolů.

[2] `https://www.google.cz`

[3] `https://www.lucidchart.com`

[4] `https://www.ganttpro.com`

[5] `https://plantuml.com/`

[6] `https://knihovna.cvut.cz/katalogy-a-databaze/hledat-v/vyhledavac-summon`

6.3 Google Scholar

U Google Scholaru[7] bylo postupováno obdobně jako v podsekci 6.1. Byly zadány stejné dotazy ale v tomto případě byly získány mnohem méně validní výsledky. Rešerše byla provedena dne 3. prosince 2023.

Byl nalezen pouze jeden článek, ve kterém je ukázán prototyp, který umožňuje definovat různé faktory, jako jsou kombinace zdrojů, rizikové faktory, minimální kvalifikace pro úkoly, časování aktivit a rozdělení úkolů na paralelní dílčí úkoly. Dále umožňuje automatické přiřazení úkolů a zdrojů a nastavení závislostí mezi úkoly. Obsahuje také XML parser, identifikuje objekty v modelech a analyzuje jejich metadata. Ganttův diagram je pak následně generován pomocí Microsoft Project[8], který automaticky uspořádává parametry, generuje diagram, provádí vyrovnání zdrojů a ukládá výsledky. Tento článek pro tuto práci však není příliš relevantní a ještě k tomu je Microsoft Project placený. [24]

6.4 Phind

V neposlední řadě byl využit online vyhledávač, který využívá umělou inteligenci, Phind[9]. Rešerše byla provedena 3. prosince 2023.

Protože nástroj využívá umělou inteligenci, byl učiněn pokus o co nejpřesnější dotaz – „Existuje nějaký otevřený software pro převodník z WBS do Ganttova diagramu?“. Vyhledávač poukázal na nástroj ProjectLibre[10] a tvrdil, že hledaný převod dokáže. Po stažení a vyzkoušení volně dostupné verze však bylo zjištěno, že nástroj umí vytvářet pouze Ganttův diagram a pak ho konvertovat do WBS, tzn. nedokáže WBS vytvářet ani ho převádět do Ganttova diagramu. Nástroj je ještě k tomu poměrně neintuitivní a dovoluje převod diagramů vytvořených pouze přímo v aplikaci s příponou `.pod`. Druhýh navrženým nástrojem byl GanttProject[11]. Údajně by měl tento nástroj podporovat WBS i Ganttův diagram. Opak byl však pravdou. Nástroj podporoval pouze tvorbu Ganttova diagramu, ale převod mezi ním a WBS už ne. [25]

S dotazem v angličtině „Is there an open-source WBS to Gantt chart convertor?“ byla úspěšnost podobná. Phind poradil jeden již doporučovaný nástroj (GanttProject[11]), dva placené (OpenProject[12], GanttPRO[13]) a jeden, který není podporovaný na Windows (LibrePlan[14]). [26] Tyto nástroje nemají v této práci využití, protože nesplňují stanovené požadavky analýzy.

Díky online vyhledávači Phind byl nalezen zajímavý nástroj na tvorbu a převod Ganttova diagramu do WBS. Pro účely této bakalářské práce však nemá užití z výše zmíněných důvodů.

6.5 Výsledek analýzy

Cílem této kapitoly bylo najít nástroj, který by splňoval stanovené požadavky – umožňoval by konverzi z WBS do Ganttova diagramu, byl by volně dostupný a dovoloval by import a export vytvořených souborů. Žádný takový nástroj však nalezen nebyl a proto se návrh aplikace konvertoru v příští kapitole (viz kapitola 7) bude zabývat aplikací, která je vyvíjena na zelené louce.

[7] `https://scholar.google.com`
[8] `https://www.microsoft.com/cs-cz/microsoft-365/project/project-management-software`
[9] `https://www.phind.com`
[10] `https://www.projectlibre.com`
[11] `https://www.ganttproject.biz`
[12] `https://www.openproject.org`
[13] `https://ganttpro.com`
[14] `https://www.libreplan.dev`

Kapitola 7

Návrh řešení konverze souborů

Na konci 5. kapitoly jsme stanovili, že se naše projektová kancelář bude skládat ze dvou volně dostupných nástrojů, které propojí pomocí aplikace konvertoru. V minulé kapitole (viz kapitola 6) jsme zjistili, že žádná taková aplikace neexistuje a v této kapitole se zaměříme na její návrh. Nejdříve si uvedeme nejčastější scénáře projektového manažera, který by pro plánování a řízení menších projektů využíval naši projektovou kancelář a tím nastíníme její výslednou funkcionalitu. Dále si ukážeme několik diagramů, které poslouží k návrhu aplikace konvertoru uvnitř projektové kanceláře – diagram aktérů, diagram případů užití, funkční a nefunkční požadavky, diagram tříd a diagram nasazení. Na závěr si ukážeme, jak by mohla výsledná aplikace vypadat.

7.1 Scénáře projektového manažera využívajícího projektovou kancelář

V následujících podsekcích jsou uvedeny nejčastější scénáře projektového manažera, který by využíval pro plánování a řízení malých projektů projektovou kancelář vyvíjenou v této práci. Podsekce jsou seřazeny chronologicky podle průběhu projektu.

7.1.1 Tvorba WBS

Projektový manažer využívá pro rozpadnutí projektu na menší, lépe zvladatelné části WBS. Tento přístup pomáhá odhadnout čas a náklady projektu, alokovat zdroje efektivně a stanovit závislosti mezi různými částmi projektu. Díky WBS může projektový manažer také vizualizovat rozsah projektu, což usnadňuje plánování a vývoj harmonogramu. Přiřazování odpovědností se stává snazším a přesnějším, protože každý úkol je jasně definován v rámci WBS. Tímto způsobem lze sledovat pokrok projektu, identifikovat milníky a kontrolní body, a zajistit, že žádná práce nebude zdvojena ani přehlédnuta.

7.1.2 Konverze WBS do Ganttova diagramu

Konverze WBS do Ganttova diagramu umožňuje projektovému manažerovi převést jednotlivé úkoly a hierarchickou strukturu WBS na vizuální reprezentaci, která zobrazuje časové rámce a logické závislosti mezi úkoly. Ganttův diagram je efektivní pro plánování a řízení projektů založených na vodopádové metodice, protože umožňuje snadné přidělování zdrojů, stanovení časových rámců, sledování závislostí mezi úkoly a jejich následné plnění. Projektový manažer tak může lépe řídit a koordinovat práci týmu.

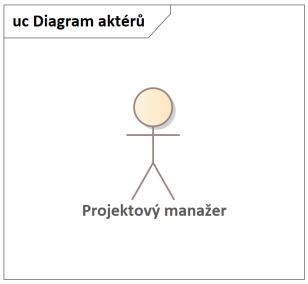
7.1.3 Konverze Ganttova diagramu zpět do WBS

V případě, že během realizace projektu dochází k úpravám v Ganttově diagramu, například přidávání nových úkolů nebo změnám názvů úkolů, projektový manažer může tyto úpravy

převést zpět do formátu WBS. Toto zjednodušení pomáhá udržet přehled o struktuře projektu a poskytnout vyššímu managementu jasný a stručný přehled o aktuálním stavu projektu. Aktualizace WBS na základě změn v Ganttově diagramu zajišťuje, že projektová dokumentace zůstává aktuální a relevantní pro potřeby řízení projektu.

7.2 Diagram aktérů

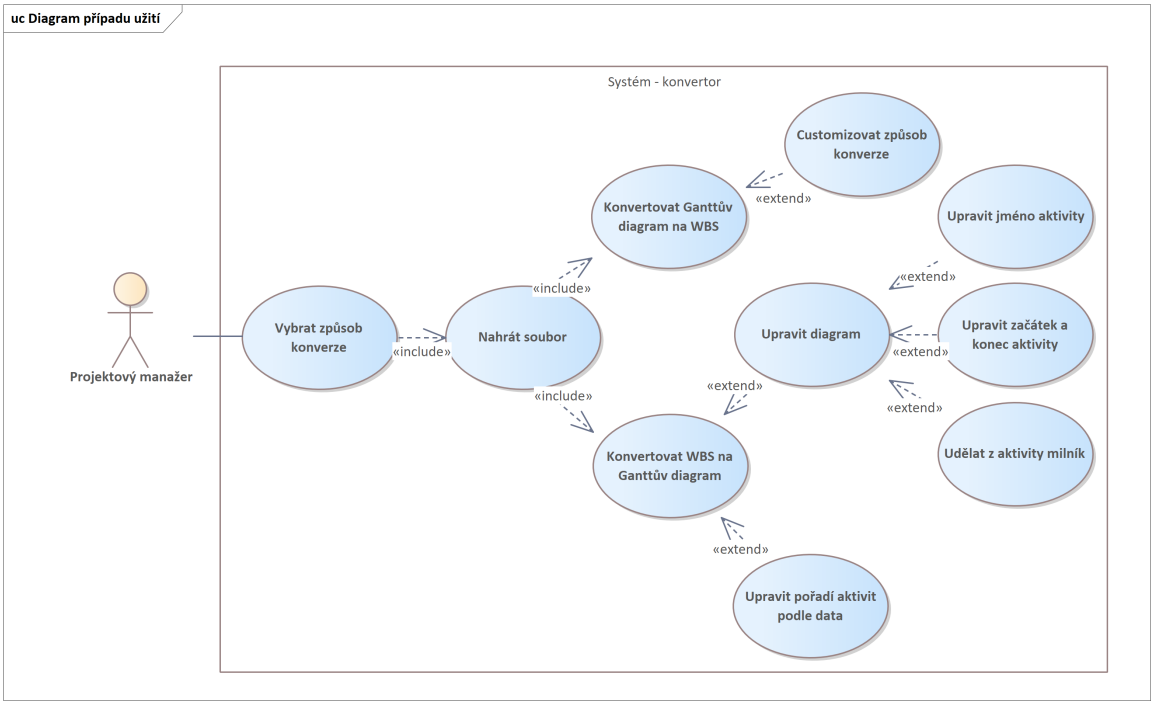
Protože výslednou aplikací této práce bude konvertor pro projektové manažery, skládá se diagram aktérů (viz obrázek 7.1) pouze z jednoho aktéra – projektového manažera. Projektový manažer má v systému neomezená práva.



Obrázek 7.1: Diagram aktérů.

7.3 Diagram případů užití

Diagram případů užití (viz obrázek 7.2) ukazuje všechny aktivity, které může projektový manažer v systému vykonávat. Projektový manažer si bude moct vybrat dva způsoby konverze, následně bude moct nahrát soubor, který chce konvertovat. Hlavními aktivitami pro projektového manažere jsou konverze z WBS na Ganttův diagram a naopak. Obě aktivity ještě obnáší dodatečné volitelné aktivity, které může projektový manažer při konverzi využít jako je např. customizace způsobu konverze nebo úprava převáděného diagramu.



Obrázek 7.2: Diagram případu užití.

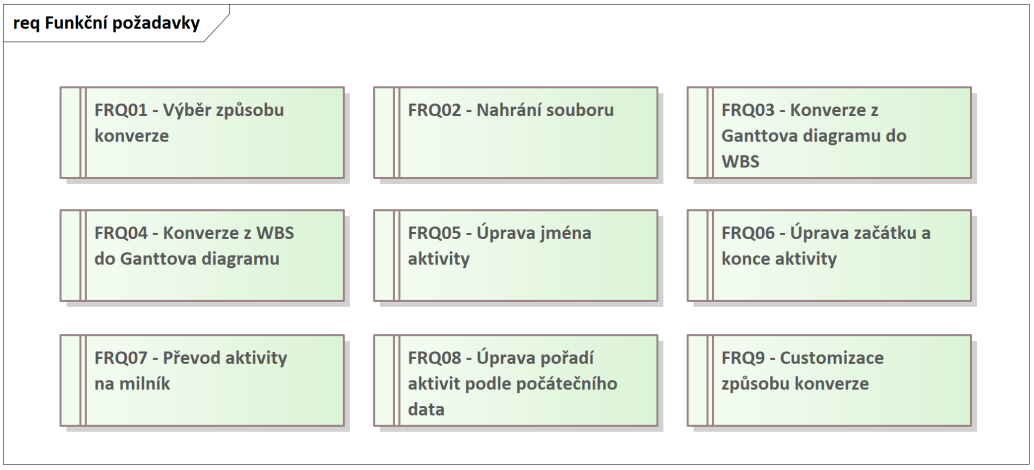
7.4 Systémové požadavky

Systémové požadavky pomáhají definovat chování vyvíjeného systému. Dělí se na funkční (FRQ), které definují, co bude systém uživatelům umožňovat, a nefunkční (NFRQ), které systému kladou kvalitativní nároky a různá omezení.

7.4.1 Funkční požadavky

Funkční požadavky znázorněné na obrázku 7.3 si popíšeme pomocí šablony:

```
Systém umožňuje <CO?> (<KOMU?>, <ZA PODMÍNEK?>, <PROČ?>).
```



Obrázek 7.3: Funkční požadavky.

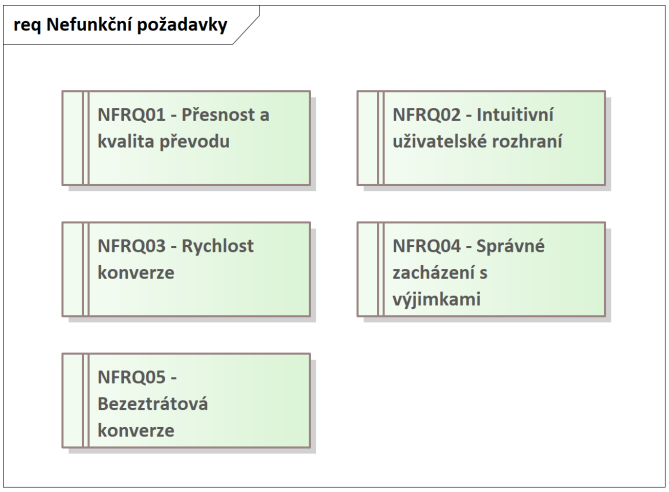
- **FRQ01 – Výběr způsobu konverze** – Systém umožňuje uživateli vybrat způsob konverze. Vybírá mezi konverzí z WBS do Ganttova diagramu a naopak.
- **FRQ02 – Nahrání souboru** – Systém umožňuje uživateli nahrát soubor ze svého zařízení.
- **FRQ03 – Konverze z Ganttova diagramu do WBS** – Systém umožňuje uživateli konverzi ze souboru vytvořeným nástrojem GanttProject[1] (přípona `.gan`) na soubor, který se dá otevřít nástrojem FreeMind[2] (přípona .mm).
- **FRQ04 – Konverze z WBS do Ganttova diagramu** – Systém umožňuje uživateli konverzi ze souboru vytvořeným nástrojem FreeMind[2] (přípona .mm) na soubor, který se dá otevřít nástrojem GanttProject[1](přípona `.gan`).
- **FRQ05 – Úprava jména aktivity** – Systém umožňuje uživateli při konverzi z WBS do Ganttova diagramu úpravu jména všech aktivit.
- **FRQ06 – Úprava začátku a konce aktivity** – Systém umožňuje uživateli při konverzi z WBS do Ganttova diagramu úpravu začátku a konce koncových aktivit[3]. Systém poté automaticky rekurzivně aktualizuje začátek a konec u rodičovských aktivit. Zároveň z těchto dvou údajů systém automaticky vypočítá délku trvání aktivity.

[1] `https://www.ganttproject.biz`
[2] `https://freemind.sourceforge.io/wiki/index.php/Main_Page`
[3] Aktivita, která nemá žádné potomky.

- **FRQ07 – Převod aktivity na milník** – Systém umožňuje uživateli při konverzi z WBS do Ganttova diagramu převést koncové aktivity[3] na milníky[4].
- **FRQ08 – Úprava pořadí aktivit podle počátečního data** – Systém umožňuje uživateli při konverzi z WBS do Ganttova diagramu zvolit, zdali chce, aby byly aktivity ve výsledném souboru seřazené podle počátečního data.
- **FRQ09 – Customizace způsobu konverze** – Systém dovoluje uživateli při konverzi z Ganttova diagramu do WBS možnost volby mezi ztrátovým a bezeztrátovým převodem a zda mají být ve výsledném souboru zahrnuty milníky.

7.4.2 Nefunkční požadavky

Nefunkční požadavky na obrázku 7.4 stanovují vlastnosti systému, které nejsou přímo spojeny s jeho funkcionalitou, ale ovlivňují jeho celkovou kvalitu, výkon, spolehlivost nebo uživatelskou přívětivost.



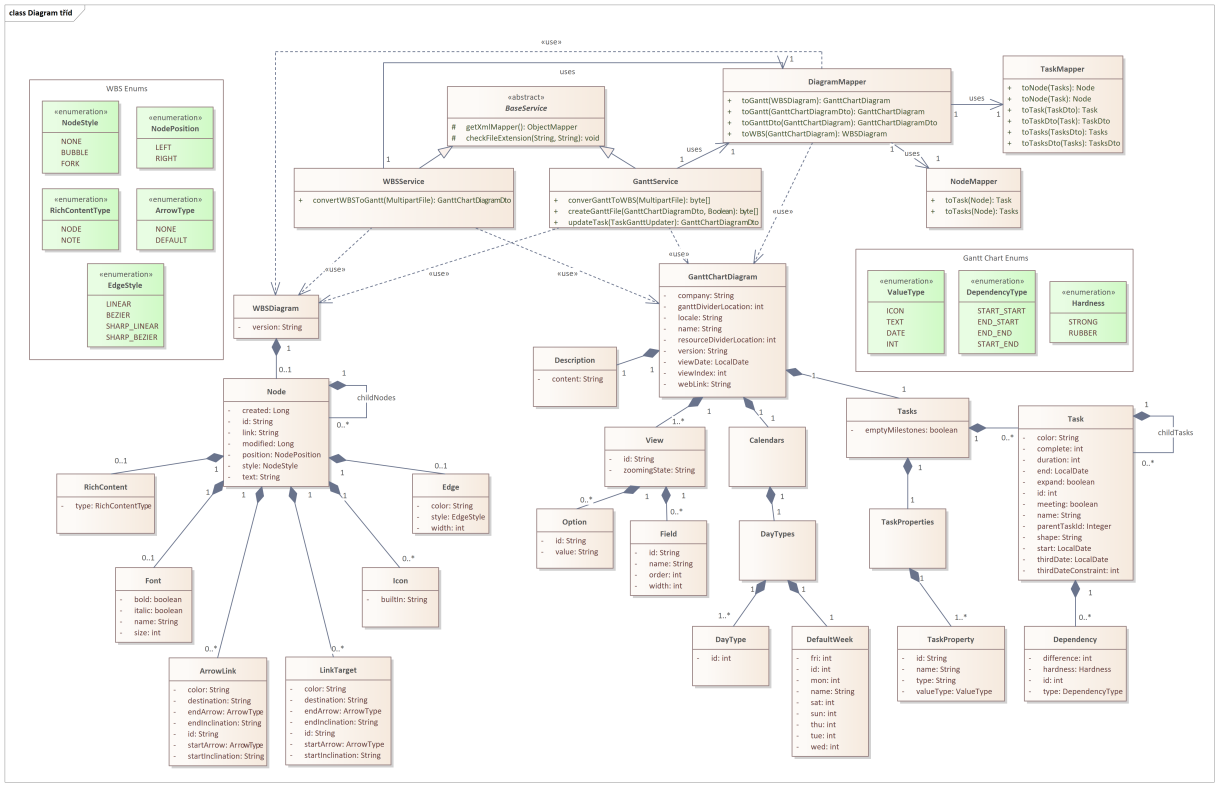
Obrázek 7.4: Nefunkční požadavky.

7.5 Diagram tříd

Diagram tříd na obrázku 7.5 znázorňuje strukturu tříd v aplikaci. Protože aplikace využívá dva specifické formáty souborů (viz sekce 5.3), mapuje diagram tříd jejich strukturu a datové typy jednotlivých atributů. Jelikož nad aplikacemi, které soubory tvoří, nemáme kontrolu, musíme se jim přizpůsobit. Kvůli tomu se v diagramu objevují následující atributy s nestandardními datovými typy, které však vývoj aplikace nijak zásadně neovlivní:

- `id: String (Node, ArrowLink, LinkTarget, View, Option, Field, TaskProperty)` – správně by měl být typu `int` nebo `long`.
- `color: String (Node, Task, Edge, ArrowLink, LinkTarget)` – správně by měl být typu `enum` nebo vestavěného barevného typu

[4]Milník označuje důležitý bod v projektu, který má nulovou délku trvání. Vizuálně zvýrazňuje klíčové body, jako je například dokončení určité fáze.

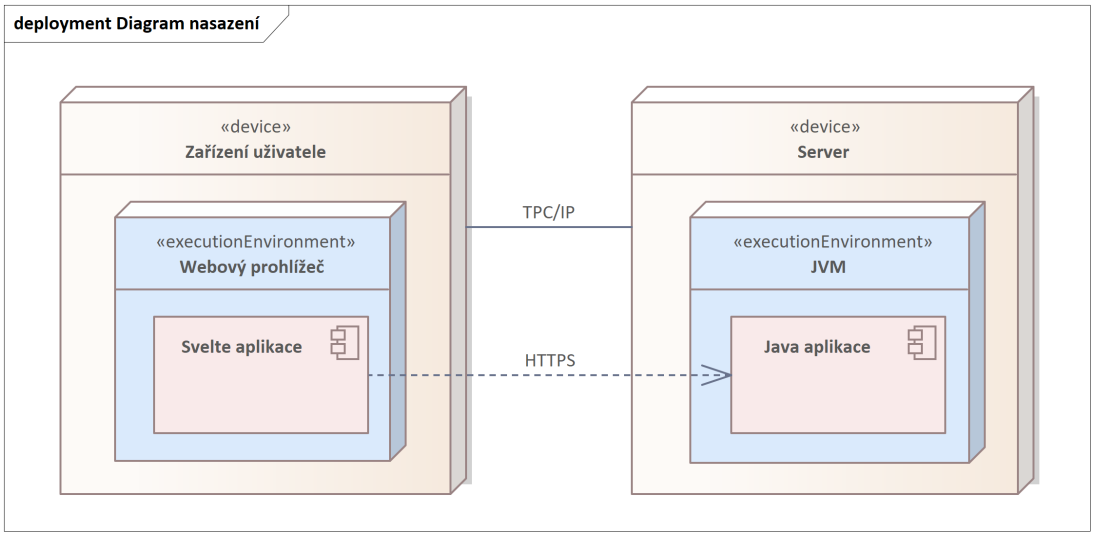


Obrázek 7.5: Diagram tříd.

Diagram tříd v plném rozlišení je přiložen v elektronické příloze.

7.6 Diagram nasazení

Obrázek 7.6 znázorňuje strukturu nasazení aplikace do produkce.



Obrázek 7.6: Diagram nasazení.

7.7 Grafický návrh

Grafický návrh slouží jako vzor pro implementaci frontendové části aplikace. Jeho hlavním cílem je předběžně navrhnout rozložení jednotlivých komponent. Konečná podoba aplikace se však může velmi lišit. Obrázek 7.7 zobrazuje návrh úvodní obrazovky výsledné aplikace. Její celý grafický návrh je ukázán v příloze A.

Návrh byl vytvořen v programu Figma[5].

Obrázek 7.7: Úvodní obrazovka aplikace.

Na základě návrhu byla aplikace realizována (viz kapitola 9). Vzhledem k tomu, že byla aplikace navržena s ohledem na uživatelskou přívětivost, bylo rozhodnuto, že bude implementována jako aplikace webová s minimalistickým uživatelským rozhraním. Tato volba byla učiněna s cílem zjednodušit a zintenzivnit interakci mezi uživatelem a aplikací. Před samotnou implementací bylo nezbytné zvolit vhodné technologie a nástroje, které budou v aplikaci využívány. Tato volba technologií a nástrojů je podrobněji popsána v následující kapitole (viz kapitola 8).

[5]https://www.figma.com

Kapitola 8

Analýza nástrojů a technologií

V této kapitole se zaměříme na výběr nástrojů a technologií pro realizaci aplikace konvertoru. Jak jsme se v minulé kapitole dozvěděli (viz kapitola 7) bude vyvíjena webová aplikace s jednoduchým uživatelským rozhraní. Proto je tato kapitola rozdělena na backendovou (viz sekce 8.1) a frontendovou (viz sekce 8.2) část. Aplikace bude bezestavová (viz podsekce 2.5.1) vzhledem k tomu, že výstupem realizace bude konvertor, nedává tedy smysl v aplikaci používat databázi ani řešit autentizaci a autorizaci uživatelů.

8.1 Backend (BE)

Backend je serverová strana webové aplikace. Ve většině případech ukládá a uspořádává data, tato data poté připravuje a poskytuje pro FE část aplikace. Uživatel s ní nikdy přímo neinteraguje. [27]

8.1.1 Výběr jazyka pro backend

Následující seznam jazyků pro backend byl sepsán pomocí článku *Frontend vs Backend* na webové stránce Geeks for Geeks[1], kde se popisuje a porovnává frontend a backend společně s uvedenými nejpopulárnějšími jazyky a frameworky v obou částech webové aplikace. [27]

Uvedené tři jazyky byly vyselektovány hlavně díky dřívějším zkušenostem s nimi. Popis jednotlivých jazyků byl sestaven z jejich oficiálních stránek.

PHP

PHP[2] je univerzální skriptovací jazyk zaměřený především na serverovou stranu webových aplikací. Umožňuje zpracovávat formulářová data, generovat dynamický obsah stránek a mnoho dalších. Jeho použití se rozděluje do tří hlavních oblastí: serverového skriptování, skriptování z příkazové řádky a vývoje desktopových aplikací. [28]

PHP je kompatibilní s mnoha operačními systémy a webovými servery a umožňuje volbu mezi procedurálním a objektově orientovaným programováním. Kromě běžného generování HTML obsahu zvládá PHP i práci s obrázky, PDF soubory, šifrování dat, odesílání emailů a zpracování různých formátů dat. PHP také nabízí bohatou podporu pro práci s databázemi a komunikaci s dalšími službami pomocí různých protokolů. Je vybaveno rozsáhlou sadou rozšíření, která umožňují rozmanité funkcionality. [28]

[1] `https://www.geeksforgeeks.org`
[2] `https://www.php.net`

Java

Java[3] je široce používaný objektově orientovaný programovací jazyk a softwarová platforma, běžící na miliardách zařízení, včetně notebooků, mobilních zařízení, herních konzolí, zdravotních přístrojů a dalších. Jeho pravidla a syntaxe jsou založeny na jazycích C a C++. Hlavní výhodou vývoje softwaru v Javě je jeho přenositelnost - kód napsaný pro Java program na notebooku lze snadno přesunout na mobilní zařízení. [29]

Java je stále nejpopulárnějším jazykem pro vývoj softwaru, ačkoliv na trhu přicházejí nové a vylepšené vývojové nástroje. Je důležitá pro konkurenceschopnost na trhu práce a zůstává žádaná pro svou stabilitu, bezpečnost a širokou podporu. [29]

Python

Python[4] je interpretovaný, objektově orientovaný, vysokoúrovňový programovací jazyk s dynamickou sémantikou. Jeho vestavěné datové struktury, spolu s dynamickým typováním a dynamickým vázáním, ho činí velmi atraktivním pro „Rapid Application Development“, stejně jako pro použití jako skriptovací nebo „glue“ jazyk ke spojení existujících komponent dohromady. Jednoduchá a snadno naučitelná syntaxe Pythonu, pomáhá s čitelností a tím snižuje náklady na údržbu programu. Python podporuje moduly a balíčky, což podporuje modularitu programu a opakované použití kódu. [30]

Finální výběr technologie

Pro vývoj BE části aplikace byl zvolen jazyk **Java**. A to hlavně díky předchozím zkušenostem s vývojem v tomto jazyce. Navíc Java podporuje velké množství knihoven a frameworků, které vývoj BE části webové aplikace velmi usnadní.

8.1.2 Výběr knihovny na práci s XML

Vzhledem k výběru jazyka Java v předchozí podsekci 8.1.1 byl vybrán seznam knihovny pomocí článku *A Guide to XML in Java* na webové stránce Baeldung[5], kde jsou uvedeny a popsány nejběžnější Java knihovny pracující s XML. [31]

Následující tři knihovny byly vybrány z důvodu jejich široké podpory a efektivnosti v rámci ekosystému Javy. Popis jednotlivých knihoven byl sestaven z jejich oficiálních stránek.

Java DOM Parser

Document Object Model (DOM) je oficiálním doporučením World Wide Web Consortium (W3C). Definuje rozhraní, které umožňuje programům přistupovat a aktualizovat styl, strukturu a obsah XML dokumentů. Po zparsování XML dokumentu pomocí DOM parsovače se získá stromová struktura, která obsahuje všechny prvky daného dokumentu. DOM poté poskytuje funkce, které slouží k procházení načteného souboru. [32][33]

JAXB

Java Architecture for XML Binding (JAXB)[6] poskytuje API a nástroje, které automatizují mapování mezi XML dokumenty a Java objekty. JAXB umožňuje vývojářům provádět následující operace: deserializovat obsah XML do reprezentace v Javě, přistupovat a aktualizovat reprezentaci v Javě, serializovat reprezentaci v Javě obsahu XML do obsahu XML. [34]

[3] `https://www.java.com`
[4] `https://www.python.org`
[5] `https://www.baeldung.com`
[6] `https://javaee.github.io/jaxb-v2`

Jackson XML

Knihovna Jackson je známá jako „nejlepší JSON parsovač pro Javu“. Jackson je sada nástrojů pro zpracování dat pro Javu, zahrnující hlavní streamingový JSON parsovač/generátor knihovny (serializace z POJOs do JSON a naopak). Jackson XML[7] je nástavba na knihovnu Jackson a slouží k serializaci a deserializaci mezi POJOs a XML. [35]

Finální výběr knihovny

Jako knihovna pro práci s XML byl zvolen **Jackson XML** vzhledem k aktivnímu přístupu autorů a elegantnímu a jednoduchému použití v kódu.

8.1.3 Výběr technologie pro mapování

Nástroje určené k mapování hrají jednu z hlavních rolí v logice aplikace konvertoru. Proto je potřeba vybrat kvalitní a spolehlivý nástroj, který její vývoj usnadní. Opět je výběr omezen pouze na nástroje, které jsou podporovány Javou. [36]

Seznam nástrojů byl vybrán pomocí článku *Performance of Java Mapping Frameworks* na webové stránce Baeldung[5], který porovnává výkon různých mapovacích frameworků v Javě. [36]

Uvedené tři nástroje byly vybrány z důvodu jejich výkonu, jednoduchosti použití a schopnosti efektivně řešit různé aspekty mapování mezi různými datovými modely, což je klíčové pro úspěch aplikace konvertoru. Popis jednotlivých technologií byl sestaven z jejich oficiálních stránek.

MapStruct

MapStruct[8] je generátor kódu, který výrazně zjednodušuje implementaci mapování mezi typy Java beanů na základě přístupu konvence nad konfigurací.

Generovaný kód mapování používá prosté volání metod a je tedy rychlý, typově bezpečný a snadno srozumitelný. [37]

JMapper

JMapper[9] je framework, který klade důraz na to poskytnout snadno použitelné, vysokovýkonné mapování mezi Java Bean objekty. Aplikuje princip DRY pomocí anotací a relačního mapování. Framework umožňuje různé způsoby konfigurace: založené na anotacích, XML nebo API. [38]

ModelMapper

ModelMapper[10] je framework, který automaticky určuje, jak se objektové modely mapují na základě konvencí, což usnadňuje proces mapování. Je inteligentní, typově bezpečný a k mapování používá předem stanovené konvence. ModelMapper podporuje integraci s jakýmkoli typek datového modelu (od JavaBeans a JSON stromů až po databázové záznamy). Tímto způsobem zjednodušuje a zabezpečuje mapování mezi různými datovými modely. [39]

Finální výběr technologie

Jako mapovací nástroj pro tuto práci byl zvolen **MapStruct**, vzhledem k jednoduchosti implementace, generování velké části logiky a předchozím zkušenostem s nástrojem.

[7] `https://github.com/FasterXML/jackson-dataformat-xml`
[8] `https://mapstruct.org`
[9] `https://jmapper-framework.github.io/jmapper-core`
[10] `https://modelmapper.org`

8.2 Frontend (FE)

Frontend, také nazývaný jako "klientská část aplikace", je část webové stránky, se kterou uživatel přímo interaguje. Zahrnuje veškerý obsah, který uživatelé přímo vnímají: barevné a stylové nastavení textu, obrázky, grafy a tabulky, tlačítka, barvy a navigační menu. Pro vývoj FE se používají následující jazyky: HTML, CSS a JavaScript. Responsivita a výkon jsou hlavními cíli FE vývoje. Vývojář musí zajistit, že stránka je responsivní[11] a žádná část webové stránky by neměla chovat abnormálně bez ohledu na velikost obrazovky. [27]

8.2.1 Výběr technologie pro frontend

Následující seznam technologií pro frontend byl sepsán na základě stejného článku jako u výběru jazyka pro backend (*Frontend vs Backend* na webové stránce Geeks for Geeks[1]). [27]

Uvedené tři technologie byly vybrány zejména na základě doporučení. Popis jednotlivých technologií byl sestaven z jejich oficiálních stránek.

React

React[12] je výkonná JavaScriptová knihovna pro tvorbu dynamických a interaktivních uživatelských rozhraní (User Interface (UI)). React je známý svou architekturou založenou na komponentách, která umožňuje vytvářet opakovaně použitelné UI prvky, čímž usnadňuje správu a údržbu komplexních webových aplikací. Je vyvíjen společností Facebook. [40]

Svelte

Svelte[13] je nástroj pro tvorbu rychlých webových aplikací. Převádí kód aplikace do optimálního JavaScriptu již při sestavení, místo interpretace kódu za běhu. Díky tomu se nemusí platit výkonová daň za abstrakce frameworku a při prvním načtení aplikace. [41]

Ve Svelte je aplikace složena z jedné nebo více komponent. Komponenta je opakovatelný samostatný blok kódu, který zahrnuje HTML, CSS a JavaScript, které spolu souvisejí. [41]

Angular

Angular[14] je komponentový framework pro tvorbu škálovatelných webových aplikací. Slouží jako sbírka dobře integrovaných knihoven, které pokrývají širokou škálu funkcí, včetně routování, správy formulářů, komunikace klient-server a dalších. Má v sobě sestavu vývojářských nástrojů, které pomáhají vyvíjet, sestavovat, testovat a aktualizovat aplikační kód. [42]

Angular je velmi škálovatelný. Může posloužit pro malé projekty až po aplikace velkých firem. Navíc ekosystém Angularu tvoří rozmanitá skupina více než 1,7 milionu vývojářů, autorů knihoven a tvůrců obsahu. [42]

Finální výběr technologie

Pro vývoj FE části aplikace byl vybrán framework **Svelte**, zejména díky jednoduchosti a rychlosti vývoje. Vzhledem k tomu, že Svelte odstraňuje běhovou režii frameworku a generuje optimalizovaný kód, vede vývoj aplikace k menší výsledné velikosti.

Všechny vybrané technologie v této kapitole budou použity k vývoji aplikace konvertoru, který je popsán v příští kapitole (viz kapitola 9).

[11] Zobrazuje se správně na zařízeních různých velikostí.
[12] `https://react.dev`
[13] `https://svelte.dev`
[14] `https://angular.io`

Kapitola 9

Implementace aplikace

V této kapitole se zaměříme na postup implementace aplikace konvertoru. Implementace byla realizována na základě provedené analýzy (viz kapitola 5 a 6), návrhu aplikace (viz kapitola 7) a výběru technologií (viz kapitola 8).

Pro vývoj BE i FE části aplikace bylo použito vývojové prostředí IntelliJ[1] od společnosti JetBrains. Toto prostředí bylo vybráno díky skvělé podpoře pro všechny potřebné jazyky a frameworky a předešlé zkušenosti. Disponuje širokou škálou užitečných nástrojů pro vývoj webových aplikací, v případě naší aplikace byl využit Svelte plugin[2]. K tomu všemu nabízí inteligentní návrhy, dynamické zvýrazňování chyb, možnosti refaktoringu a bohaté navigační funkce, což přispívá k lepšímu a rychlejšímu vývoji.

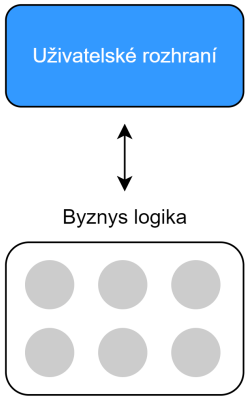
9.1 Architektura

Aplikace je postavená formou monolitické architektury a využívá Model-View-Controller (MVC) jako návrhový vzor.

9.1.1 Monolitická architektura

Monolitická architektura je tradiční model softwarového programu, kde je celá aplikace vyvíjena jako jedna jediná a velká aplikace. Všechny komponenty jsou integrovány do jednoho celku. V monolitu je pak snadná správa kódu a snižuje se kognitivní náročnost i náročnost nasazení do produkce. [43]

Tato architektura byla zvolena zejména kvůli menší velikosti a jednoduchosti aplikace. Znázornění monolitické architektury pro tuto aplikaci je ukázané na obrázku 9.1.



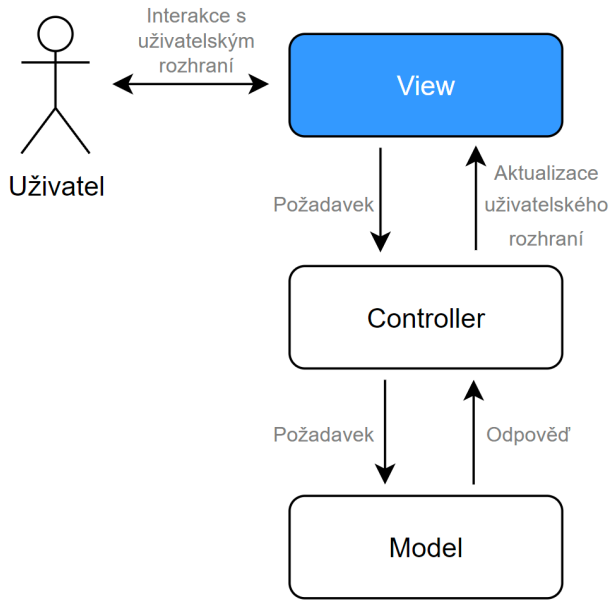
Obrázek 9.1: Ukázka monolitické architektury.

[1] https://www.jetbrains.com/idea
[2] https://plugins.jetbrains.com/plugin/12375-svelte

9.1.2 MVC

Model-View-Controller (MVC) je návrhový vzor, který rozděluje aplikaci do tří nezávislých komponent: datového modelu (Model), uživatelského rozhraní (View) a řídící logiky (Controller). Každá komponenta je zodpovědná za konkrétní aspekt funkcionality aplikace. Toto rozdělení zjednodušuje údržbu a rozšiřování aplikace, protože změny v jedné komponentě nevyžadují změny v ostatních komponentách. [44]

Na obrázku 9.2 je zobrazena ukázka MVC.



Obrázek 9.2: Ukázka MVC.

9.2 Backend

Jak jsme se již dozvěděli v předešlé kapitole (podsekce 8.1.1) byl pro vývoj BE části aplikace vybrán programovací jazyk Java. Pro usnadnění vývoje byl zvolen framework Spring Boot[3] společně s knihovnou Project Lombok[4]. Projekt je postaven na Apache Mavenu[5].

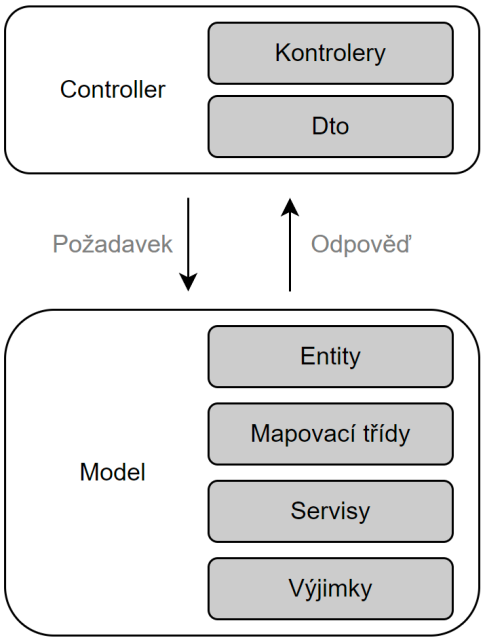
Na začátku minulé kapitoly (viz kapitola 8) jsme si také řekli, že postrádá smysl implementovat zabezpečení a databázi vzhledem k bezestavovosti výsledné aplikace.

V této sekci se zaměříme na strukturu BE, popíšeme si jeho jednotlivé části a ukážeme si k nim úryvky kódu.

9.2.1 Struktura backendové části aplikace

Jak již bylo zmíněno (viz podsekce 9.1.2), pro rozdělení aplikace konvertoru je použit návrhový vzor MVC. Na obrázku 9.3 je dále rozdělena část datového modelu a řídící logiky. V dalších podsekcích jsou jednotlivé části popsány a ukázány formou úryvku kódu.

[3] `https://spring.io`
[4] `https://projectlombok.org`
[5] `https://maven.apache.org`



Obrázek 9.3: Ukázka struktury backendové části aplikace.

9.2.2 Kontrolery

Kontroler je komponenta, která přijímá vstupy od uživatele a rozhoduje o následném provedení odpovídajících akcí. Funguje jako prostředník mezi uživatelským rozhraní a datovou vrstvou. Implementuje obchodní logiku a rozhoduje, jak budou data zpracována a jaká data budou zobrazena uživateli.

V projektu se pro kontrolery používá Representational State Transfer (REST), protože poskytuje konzistentní a standardní způsob návrhu rozhraní pro webové služby. Hlavním důvodem a výhodou použití REST pro výslednou aplikaci je jednoduchá práce se soubory.

Na ukázce 9.1 je uvedena zkrácená verze kontroleru zobrazující metodu pro aktualizaci úkolu v reprezentaci Ganttova diagramu. Tato metoda je namapovaná na `/gantt/updateTask` endpoint. Anotace `@Valid` zařizuje kaskádovou validaci označeného objektu pomocí Jakarta Validation[6]. V tomto případě se jedná o třídu `TaskGanttUpdater`.

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/gantt")
public class GanttController {

    private final GanttService ganttService;

    @ResponseStatus(HttpStatus.OK)
    @PostMapping(
            value = "/updateTask",
            consumes = MediaType.APPLICATION_JSON_VALUE,
            produces = MediaType.APPLICATION_JSON_VALUE
    )
    public ResponseEntity<GanttChartDiagramDto> updateTask(
            @RequestBody @Valid TaskGanttUpdater taskGanttUpdater
    ) {
        return ResponseEntity.ok(ganttService.updateTask(taskGanttUpdater));
    }
}
```

Ukázka kódu 9.1: Příklad kontroleru v aplikaci.

[6] https://beanvalidation.org

9.2.3 Dto

Data transfer object (Dto) slouží jako prostředek pro přenos dat mezi FE a BE. Po obdržení Dto objektů jsou data mapována na entitní objekty, se kterými se následně v aplikaci pracuje.

V aplikaci jsou všechny Dto třídy typu `record`[7]. Objekty vytvořené pomocí recordu jsou neměnné a proměnné jsou přístupné bez nutnosti definovat gettery, což je pro definici a tvorbu Dto vhodné. [45]

Na ukázce 9.2 jsou některé atributy recordu `TaskDto` validovány pomocí Jakarta validation[6].

```
public record TaskDto(
        @NotNull Integer id,
        @NotBlank String name,
        String color,
        String shape,
        Boolean meeting,
        @NotBlank String start,
        @PositiveOrZero Integer duration,
        Integer complete,
        String thirdDate,
        Integer thirdDateConstraint,
        Boolean expand,
        List<DependencyDto> dependencies,
        List<TaskDto> childTasks,
        @NotBlank String end,
        Integer parentTaskId
) {
}
```

Ukázka kódu 9.2: Příklad Dto v aplikaci.

9.2.4 Entity

Entity jsou pro aplikaci konvertoru nedílnou součástí. Díky knihovně Jackson XML (viz podsekce 8.1.2) pomáhají jednotlivé entity s deserializací XML souboru do POJOs a naopak. Všechny entity v aplikaci jsou ukázány na diagramu tříd (viz obrázek 7.5).

Ukázka 9.3 popisuje třídu `Tasks`. Kromě anotací knihovny Project Lombok[4] (`@Getter` a `@Setter`) slouží všechny ostatní anotace jako konfigurace pro knihovnu Jackson XML:

- `@JsonIgnoreProperties(ignoreUnknown = true)` označuje, že by Jackson měl ignorovat všechny neznámé prvky při deserializaci XML do této třídy.
- `@JacksonXmlProperty(isAttribute = true, localName = "empty-milestones")` určuje, že `emptyMilestones` má být interpretován jako atribut XML elementu s názvem `empty-milestones`.
- `@JacksonXmlProperty(localName = "taskproperties")` říká knihovně, že element (výchozí hodnota `isAttribute` je `false`) `taskProperties` má být mapován na XML element s názvem `taskproperties`.
- `@JsonSetter(nulls = Nulls.AS_EMPTY)` specifikuje, že Jackson by měl nastavit hodnotu atributu na prázdný řetězec, pokud je hodnota `null`. Toto nastavení zajišťuje, že nenastane nečekané chování – seznam prvků je roven `null`.
- `@JacksonXmlElementWrapper(useWrapping = false)` informuje Jackson, že by kolekce `taskList` neměla být obalena do dalšího elementu při serializaci do XML.

[7] Record je typem třídy v jazyce Java, který usnadňuje modelování jednoduchých datových struktur bez zbytečné složitosti běžných tříd.

Knihovna Project Lombok se automaticky integruje do editoru a buildovacích nástrojů a usnadní díky tomu práci [46]. Minimalizuje nebo odstraňuje tzv. boilerplate kód. V ukázce 9.3 pomocí dvou anotací všem atributům funkce nastavuje gettery a settery.

```
@Getter
@Setter
@JsonIgnoreProperties(ignoreUnknown = true)
public class Tasks {

        @JacksonXmlProperty(isAttribute=true, localName="empty-milestones")
        private boolean emptyMilestones;

        @JacksonXmlProperty(localName = "taskproperties")
        private TaskProperties taskProperties;

        @JsonSetter(nulls = Nulls.AS_EMPTY)
        @JacksonXmlElementWrapper(useWrapping = false)
        @JacksonXmlProperty(localName = "task")
        private List<Task> taskList;
}
```

Ukázka kódu 9.3: Příklad entity v aplikaci.

Jackson XML je pak schopen deserializovat XML obsah ukázaný na ukázce 9.4 do třídy `Tasks` z ukázky 9.3. Je vidět, že atributové prvky se mapují na základní typy (`String`, `int`, `boolean` atd.) a pro ostatní je potřeba definovat další třídy, které jsou obdobou třídy `Tasks`.

```
<tasks empty-milestones="true">
        <taskproperties>
            ...
        </taskproperties>
        <task ...>
        <task ...>
</tasks>
```

Ukázka kódu 9.4: Příklad elementu tasks v XML souboru.

9.2.5 Mapovací třídy

Další nedílnou součástí konvertoru jsou mapovací třídy. Slouží k mapování mezi dvěma objekty různých typů (např. Dto a entita nebo WBS entita a GanttChart entita). MapStruct (viz podsekce 8.1.3) generuje třídy, které implementují (popř. extendují) rozhraní (popř. abstraktní třídy) označené anotací `@Mapper`.

Ukázka 9.5 popisuje mapovací rozhraní `NodeMapper`. Toto rozhraní specifikuje v anotaci `@Mapper` konfigurační možnosti pro mapování objektů.

- `componentModel = "spring"` určuje, že vygenerovaná třída bude komponentou.
- `imports = {LocalDate.class}` importuje třídu `LocalDate`, která je použita při mapování.
- `unmappedTargetPolicy = ReportingPolicy.IGNORE` nastavuje strategii pro nezmapované cílové atributy na jejich ignorování.

Metoda `toTask` mapuje objekt typu `Node` na objekt typu `Task`. V ukázce 9.5 jsou uvedeny možnosti mapování pro jednotlivé atributy třídy.

- Atribut `id` je ignorován pomocí `ignore = true`.
- Atribut `childNodes` je mapován na atribut `childTasks`.

- Konstantní hodnota „1“ u atributu `duration` je nastavena pomocí `constant = "1"`.
- U atributu `start` je využita možnost definice konstanty pomocí Java výrazu díky `expression = java(LocalDate.now())`.
- Hodnota atributu `color` je získána z objektu `node` pomocí specifikované metody `defaultColor`, která je označena anotací `@Named("defaultColor")`.

```
@Mapper(
        componentModel = "spring",
        imports = {LocalDate.class},
        unmappedTargetPolicy = ReportingPolicy.IGNORE
)
public interface NodeMapper {

    @Mapping(target = "id", ignore = true)
    @Mapping(target = "childTasks", source = "childNodes")
    @Mapping(target = "duration", constant = "1")
    @Mapping(target = "start", expression = "java(LocalDate.now())")
    @Mapping(target ="color", source ="node", qualifiedByName="defaultColor")
    Task toTask(Node node);

    @Named("defaultColor")
    default String defaultColor(Node node) {
        if (node.getChildNodes() == null || node.getChildNodes().isEmpty()) {
            return "#ffcc00";
        } else {
            return "#000000";
        }
    }
}
```

Ukázka kódu 9.5: Příklad mapovacího rozhraní.

MapStruct při Maven kompilaci automaticky vygeneruje třídu (viz ukázka 9.6), která implementuje rozhraní uvedené v ukázce 9.5. Tato vygenerovaná třída automaticky provádí kontrolu `null` hodnoty a pokud mají některé atributy ze zdrojového a cílového objektu shodující se názvy a nejsou určeny k ignorování, automaticky je také mapuje.

```
@Component
public class NodeMapperImpl implements NodeMapper {

    @Override
    public Task toTask(Node node) {
        if ( node == null ) { return null; }
        Task task = new Task();
        task.setChildTasks( nodeListToTaskList( node.getChildNodes() ) );
        task.setColor( defaultColor( node ) );
        task.setDuration( 1 );
        task.setStart( LocalDate.now() );
        return task;
    }
    protected List<Task> nodeListToTaskList(List<Node> list) {
        if ( list == null ) { return null; }
        List<Task> list1 = new ArrayList<Task>( list.size() );
        for ( Node node : list ) {
            list1.add( toTask( node ) );
        }
        return list1;
    }
}
```

Ukázka kódu 9.6: Příklad vygenerované mapovací třídy.

Z ukázky 9.6 je vidět, že MapStruct umí generovat i rekurzivní metody (viz metoda `nodeListToTaskList`).

9.2.6 Servisy

Servisy zastřešují veškerou logiku a zpracování dat aplikace. Poskytují rozhraní pro manipulaci s daty a provádějí nad nimi operace.

Metody v servisách reflektují metody definované v kontrolerech a využívají všechny technologie definované v modelové části (viz podsekce 9.2.1).

Ukázka kódu 9.7 představuje metodu, která konvertuje WBS do Ganttova diagramu. Tato metoda provádí několik operací, včetně validace souboru, čtení dat ze souboru, zpracování dat a následné konverze do formátu vhodného pro zobrazení Ganttova diagramu.

Anotace `@Slf4j` od Project Lombok slouží k loggování. Využívá k tomu Simple Logging Facade for Java (SLF4J)[8].

K načítání XML souboru slouží `ObjectMapper` od Jackson XML.

`DiagramMapper` má za úkol mapování načtené třídy `WBSDiagram` do třídy `GanttChartDiagram` a následně převést výsledný diagram do Dto formátu, který je předán na FE aplikace.

```
1  @Service
2  @RequiredArgsConstructor
3  @Slf4j
4  public class WBSService extends BaseService {
5
6      private final DiagramMapper diagramMapper;
7
8      public GanttChartDiagramDto convertWBSToGantt(MultipartFile file) {
9          try {
10             checkFileExtension(file.getOriginalFilename(), ".mm");
11             ObjectMapper xmlMapper = getXmlMapper();
12             WBSDiagram wbs = xmlMapper.readValue(file.getBytes(), WBSDiagram.class);
13             checkNode(wbs.getNode());
14             GanttChartDiagram gantt = diagramMapper.toGantt(wbs, new Context());
15             defaultSetting(gantt);
16             log.info("Returned Gantt Chart successfully.");
17             return diagramMapper.toGanttDto(gantt);
18         } catch (...) { throw ... }
19     }
20 }
```

Ukázka kódu 9.7: Příklad servisní metody pro konverzi XML souboru obsahující WBS na Dto Ganttova diagramu.

9.2.7 Výjimky

Výjimky v systému slouží k zachycení a řízení neočekávaných situací nebo chyb, které mohou v programu nastat. Hlavním účelem takové výjimky je informovat uživatele o jejím nastání.

V ukázce 9.8 je ukázána třída `ConverterExceptionAdvice` označena anotací `@RestControllerAdvice`. Třída slouží pro globální zachytávání a následné zpracování výjimek v rámci aplikace. Jsou v ní definovány metody pro zpracování jednotlivých typů výjimek, které mohou nastat v průběhu běhu aplikace.

Metoda `handleInvalidGanttException` je označena anotací `@ExceptionHandler`, která umožňuje specifikovat a zpracovávat výjimku, v tomto případě se jedná o výjimku typu `InvalidGanttException`. Při zachycení této výjimky je vytvořen Dto `ExceptionDto`, který obsahuje informace o výjimce, a následně je vrácen klientovi v JSON formě odpovědi s chybovým stavem 400 (Bad Request).

[8] `https://www.slf4j.org`

Tento přístup umožňuje centralizované zpracování výjimek v aplikaci a poskytuje jednotný mechanismus pro jejich řešení a komunikaci s klientem.

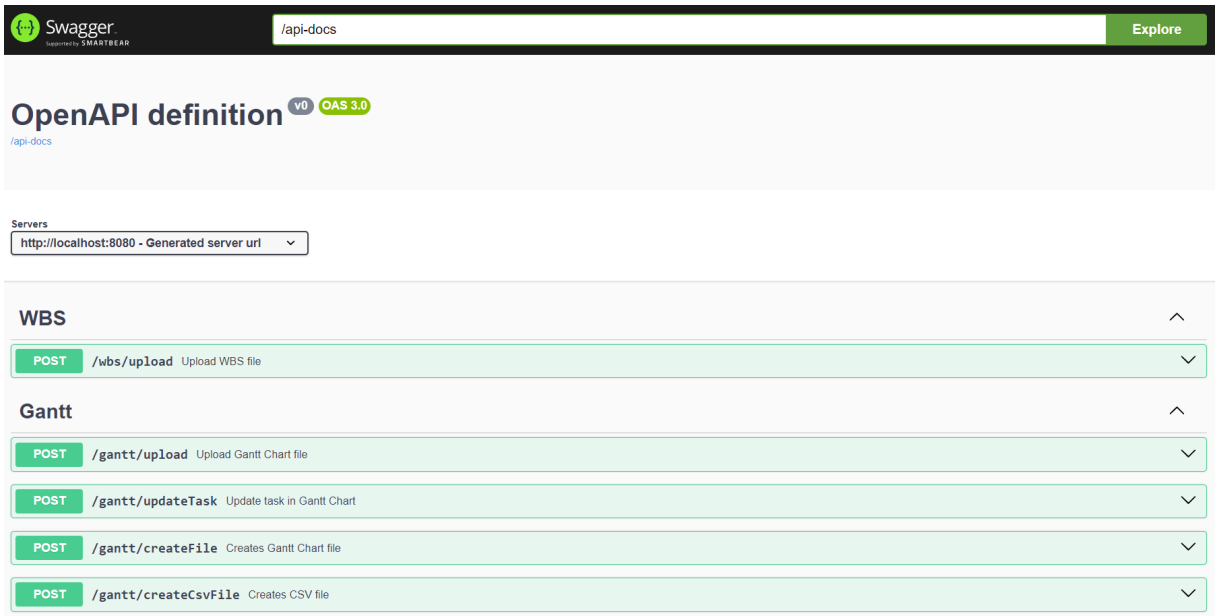
```java
@RestControllerAdvice
@Slf4j
public class ConverterExceptionAdvice {

    @ExceptionHandler(InvalidGanttException.class)
    public final ResponseEntity<ExceptionDto> handleInvalidGanttException(
        InvalidGanttException ex) {
        ExceptionDto exception = buildException("Invalid Gantt - INVALID GANTT EXCEPTION",
            ex);
        log.error(ex.getMessage());

        return ResponseEntity
                .badRequest()
                .contentType(MediaType.APPLICATION_JSON)
                .body(exception);
    }
}
```

Ukázka kódu 9.8: Příklad třídy na globální zachytávání výjimek.

9.2.8 Dokumentace

K dokumentaci se využívá Swagger OpenAPI[9]. Swagger OpenAPI je projekt používaný k popisu a dokumentaci RESTful API (endpointů). Definuje sadu souborů potřebných k popisu takového API. Tyto soubory mohou být poté použity projektem Swagger-UI k zobrazení API a Swagger-Codegen k generování klientů v různých jazycích. Další nástroje mohou rovněž využít výsledných souborů, jako jsou nástroje pro testování. [47]

K nastavení dokumentace ukázané na obrázku 9.4 je zapotřebí u každé metody kontroleru specifikovat dvě anotace: `@Operation` a `@Parameters`. Do první zmíněné se vkládají všechny potřebné informace k metodě jako je název, možné odpovědi s jejich kódem a obsahem. Do druhé se definují všechny parametry endpointu.



Obrázek 9.4: Ukázka dokumentace endpointů v aplikaci.

[9] `https://swagger.io/specification/v2`

9.3 Frontend

Z minulé kapitoly (viz podsekce 8.2.1) vyplývá, že pro vývoj FE části aplikace byl vybrán framework Svelte. Vzhledem k minimálním zkušenostem s FE vývojem a faktem, že FE část nebyla hlavním cílem této práce, byl zvolen jednoduchý grafický design (viz příloha A). Jak to již u frameworků bývá, vyžaduje vývoj ve Svelte alespoň základní znalosti HTML, CSS a JavaScriptu.

V této sekci si nejprve povíme něco o Svelte komponentách, poté si ukážeme strukturu FE části aplikace a nakonec si jednotlivé části popíšeme.

9.3.1 Svelte komponenty

Komponenty jsou základními stavebními kameny Svelte aplikací. Jsou psány do souborů s příponou `.svelte` a obsahují tři sekce. Všechny sekce – script, style a markup – jsou volitelné.

`<script>`

Blok `<script>` obsahuje JavaScript, který se spouští při vytvoření instance komponenty. Proměnné deklarované (nebo importované) v komponentě[10] jsou „viditelné“ a použitelné v její markup (HTML) sekci.

Svelte používá klíčové slovo `export` k označení proměnné, která se má stát přístupnou pro ostatní komponenty. Tímto způsobem mohou ostatní části kódu, které používají tuto komponentu, získávat přístup k této proměnné.

Jakýkoli top-level proměnná[10] může být označena jako reaktivní díky JavaScriptové značce `$`. Kdykoli se změní hodnota stanovená uvnitř reaktivní proměnné, aktualizuje se i hodnota proměnné. Vyhodnocení probíhá až po zpracování ostatního skriptového kódu ale předtím, než je vykresleno HTML stránky. [48]

`<style>`

CSS uvnitř `<style>` bloku je omezeno na komponentu, ve které je definované. Následně nedochází ke konfliktu jmen díky přidání speciální Svelte třídy k ovlivněným prvkům, která je odvozena z hashe stylů komponenty (např. svelte-123xyz). [48]

Markup sekce

Malé písmeno na začátku značky, jako je `<div>`, označuje běžný HTML prvek. Velké počáteční písmeno u značky, jako je `<Widget>`, indikuje komponentu.

Když je JavaScriptový výraz v markup sekci obklopen složenými závorkami, je jeho hodnota interpretována jako text. V elementu `<h1>Hello {name}!</h1>` se za proměnnou `name` doplní její hodnota.

Svelte ve své markup sekci také poskytuje možnost logického bloku – `{#if ...}`, `{#each ...}`, `{#await ...}`, `{#key ...}`. Blok `{#key expression}...{key}` zajišťuje přegenerování prvků uvnitř jeho bloku, kdykoli se změní hodnota `expression`.

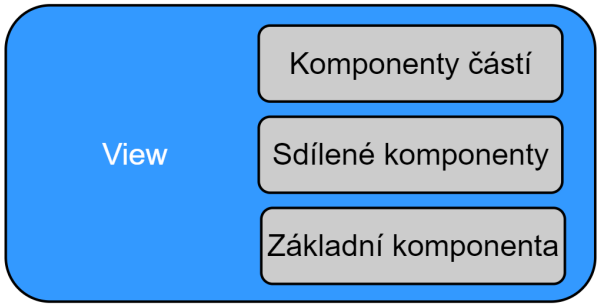
V neposlední řadě mohou mít jednotlivé elementy *direktivy*, které nějakým způsobem ovlivňují jejich chování. Patří k nim například `on:eventname`, který naslouchá DOM událostem. [49]

9.3.2 Struktura frontendové části aplikace

Jak již bylo zmíněno v podsekci 9.1.2, pro rozdělení výsledné aplikace konvertoru je použit návrhový vzor MVC. Na obrázku 9.5 je dále rozdělena část uživatelského rozhraní. V dalších podsekcích si jednotlivé části popíšeme a ukážeme si k nim ukázky kódu.

[10] tj. proměnné, které nejsou deklarované uvnitř funkce nebo logického bloku

Aplikace vytvořená pomocí Svelte se skládá z komponent (viz podsekce 9.3.1). Komponenty se poté importují a používají v jiných komponentách. V této aplikaci jsou komponenty rozděleny do komponent částí, které tvoří sekce stránek, sdílených komponent, které se používají ve více komponentech částí a základní komponenta, což je základní stavební kamen celé aplikace.



Obrázek 9.5: Ukázka struktury frontendové části aplikace.

9.3.3 Komponenty částí

Komponenty částí obsahují logiku a vizuál větších sekcí stránek, které se poté importují buď to samy do sebe nebo do základní komponenty (`App.svelte`, viz podsekce 9.3.5).

Na ukázce 9.9 je ukázána velmi zkrácená verze Svelte komponenty, která zajišťuje nahrání souboru s příponou `.gan`, resp. Ganttova diagramu.

```
<script>
    ...
</script>

<FileUpload {files} inputAccepted=".gan" on:files={(e) => files = e.detail}>
</FileUpload>
{#if files && files.length > 0}
    <div class="gantt-checkboxes">
        <Checkbox ...>...
            <ToolTip ...>...</ToolTip>
        </Checkbox>
    </div>
{/if}

<style>
    ...
</style>
```

Ukázka kódu 9.9: Příklad Svelte komponenty pro nahrání souboru s Ganttovým diagramem.

9.3.4 Sdílené komponenty

Sdílené komponenty jsou opakovaně používané prvky, které poskytují určitou funkcionalitu nebo vizuální prvek a používají se v různých částech aplikace. Tyto komponenty jsou navrženy tak, aby byly znovupoužitelné, což usnadňuje správu a údržbu kódu.

Na ukázce 9.10 je příklad zjednodušené Svelte komponenty pro zobrazování popisku u elementu, např. tlačítka.

Do `<slot>` sekce se při použití v jiné komponentě dá vložit jakákoli další markup sekce (popsáno v podsekci 9.3.1).

```
<script>
    export let show;
    export let x = 0;
    export let y = 0;
```

```
</script>

{#if (show)}
    <div class="tooltip" style="transform: translate({x}%,{y}%)"><slot></slot></div>
{/if}

<style>
    ...
</style>
```

Ukázka kódu 9.10: Příklad sdílené Svelte komponenty pro popisek.

Komponenta, zobrazená na ukázce 9.10, může být snadno importována a použita v různých částech aplikace, což zlepšuje konzistenci a snižuje duplikaci kódu.

9.3.5 Základní komponenta

Základní komponenta (často nazývaná jako `App.svelte`) je hlavní komponentou aplikace. Obsahuje celkovou strukturu a kostru rozložení stránek v aplikaci. Tato komponenta obvykle obsahuje globální prvky, jako jsou záhlaví, zápatí a další komponenty částí aplikace.

Na ukázce 9.11 je příklad použití komponent částí. Je vidět, že se pomocí logického bloku rozhoduje, která komponenta bude použita.

```
<script>
    ...
</script>

<Header/>
<main ...>
    <Tabs {activeItem} {items} on:tabChange={tabChange}/>
    {#if activeItem === 'WBS -> Gantt'}
        <WBSUpload {files} errorMessage=""/>
    {:else if activeItem === 'Gantt -> WBS'}
        <GanttUpload {files} errorMessage=""/>
    {/if}
</main>
<Footer/>

<style>
    ...
</style>
```

Ukázka kódu 9.11: Ukázka importu a použití komponent částí v App.svelte.

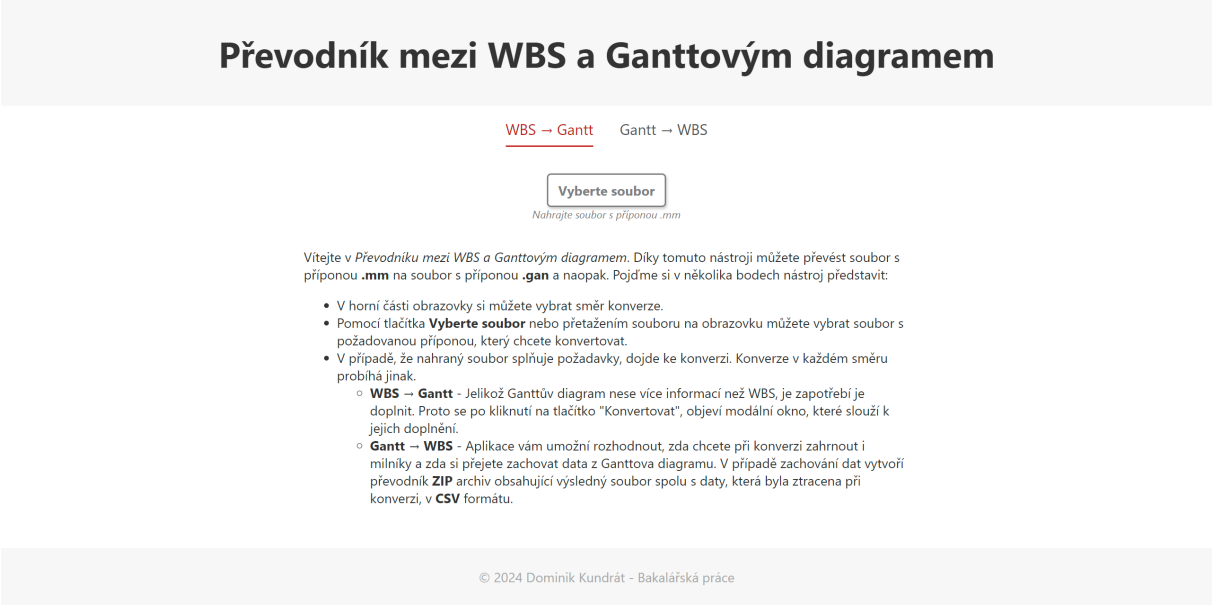
Struktura v základní komponentě, zobrazené na ukázce 9.11, umožňuje správu celkového vzhledu a chování aplikace. Zajišťuje konzistenci mezi různými stránkami nebo částmi aplikace.

9.3.6 Uživatelské rozhraní

Jak bylo řečeno v úvodu této sekce (9.3), nebylo uživatelské rozhraní hlavním cílem této práce. Proto byl zvolen jednoduchý grafický design se zaměřením na poskytnutí všech funkcionalit aplikace konvertoru, která poskytuje možnost konverze mezi WBS a Ganttovým diagramem. Na obrázku 9.6 je zobrazena úvodní stránka aplikace. V horní části obrazovky si uživatel může vybrat směr konverze. Pod výběrem směru se nachází tlačítko pro nahrání souboru ke konverzi. Na stránce se nachází i stručný popis aplikace.

Celé uživatelské rozhraní je ukázáno v příloze B. Na první pohled je vidět, že oproti grafickému návrhu (viz příloha A) došlo k mnoha změnám, které vyplynuly, jak z průběhu implementace, tak z uživatelského testování (viz sekce 11.2).

Nejvíce zajímavá část UI z uživatelského i implementačního hlediska je rozhodně modální okno na doplnění informací do Ganttova diagramu (viz obrázek B.4 v příloze B) při převodu z WBS do Ganttova diagramu. Doplňování informací k jednotlivým úkolům je doprovázeno jejich automatickou aktualizací. V modálním okně jsou ošetřeny všechny špatné vstupy a při úspěchu (popř. neúspěchu) se v levém dolním rohu zobrazuje modální okénko s informační zprávou.



Obrázek 9.6: Podoba úvodní stránky aplikace konvertoru.

Kapitola 10

Průběh konverze

V této kapitole si nejprve ukážeme, detailně popíšeme a porovnáme formáty souborů vybraných nástrojů používaných ke konverzi. Následně si podrobně popíšeme průběh konverze v obou směrech. Ukážeme si, které části souborů jsou mapovány a které ne, a zdůvodníme si, proč tomu tak je. Zaměříme se hlavně na teoretickou část převodu, protože část technická byla již z většiny popsána v minulé kapitole (viz podsekce 9.2.4, 9.2.5 a 9.2.6). Na logice této kapitoly je postavený celý model zmíněný v podsekci 9.2.4 a ukázaný v diagramu tříd (viz obrázek 7.5).

10.1 Formáty využívaných souborů

Pro tvorbu aplikace konvertoru bylo nutné důkladně zanalyzovat oba formáty souborů, jejichž nástroje byly vybrány v 5. kapitole a které se v ní využívají. I přes to, že oba formáty se skládají z jiných prvků, sdílí spolu stejný značkovací jazyk – XML.

Shoda v použití XML jako značkovacího jazyka mezi oběma formáty souborů značně usnadnila vývoj aplikace tím, že snížila potřebu použití různých technologií, zlepšila interoperabilitu[1] a integrační možnosti, a umožnila efektivnější využití existujících nástrojů a knihoven.

10.1.1 .mm soubor

Formát souboru `.mm` je produkován nástrojem na tvorbu myšlenkových map FreeMind[2] a je založený na XML. Na zkrácené ukázce 10.1 jsou uvedeny základní stavební kameny tohoto souboru, které budou využity při konverzi:

- `<map>` – Kořenový prvek diagramu s atributem `version`.
- `<node>` – Element, který představuje uzel (resp. úkol) mapy s atributy `ID`, `TEXT`, `POSITION`, `STYLE`, `LINK`, `CREATED` a `MODIFIED`.

Je vidět, že mapa obsahuje uzly v hierarchické struktuře.

Na uvedené ukázce vidíme problémovou nekonzistenci, které jsou zmíněny v sekci s diagramem tříd (viz sekce 7.5) – atribut `version` je malými písmy, ale všechny ostatní jsou velkými; hodnota atributu `ID` je datové typu `String`.

```
<map version="1.0.1">
    <node ID="ID_1" TEXT="Welcome To WiseMapping" POSITION="right">
        <node ID="ID_2" POSITION="right" STYLE="fork" TEXT="Introduction"/>
        <node ID="ID_1041815192" POSITION="right" TEXT="Basics"/>
    </node>
</map>
```

Ukázka kódu 10.1: Příklad `.mm` souboru.

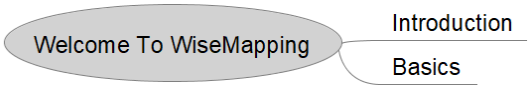
[1] Schopnost systémů vzájemně si poskytovat služby a efektivně spolupracovat.
[2] `https://freemind.sourceforge.io/wiki/index.php/Main_Page`

Mezi další prvky, které soubor může obsahovat a jsou mapovány pomocí Jackson XML (viz podsekce 9.2.4), patří[3]:

- `<richcontent>` – Prvek, který obsahuje text ve formátu `XHTML`.
- `<arrowlink>` – Element, který spojuje dva uzly šipkou.
- `<icon>` – Element, který přidává k uzlu ikonu.
- `<font>` – Element, který specifikuje vlastnosti písma v elementu.
- `<edge>` – Element, který specifikuje vlastnosti hrany.

.mm soubor zobrazený nástrojem FreeMind

Na obrázku 10.1 je ukázán soubor obsahující WBS z ukázky 10.1 v nástroji FreeMind[2].



Obrázek 10.1: WBS vytvořený `.mm` souborem.

10.1.2 .gan soubor

Formát souboru `.gan` je tvořený nástrojem GanttProject[4], který je určený pro tvorbu a správu Ganttových diagramů. Tento formát je založen na XML a obsahuje kompletní informace o projektu, včetně úkolů, jejich závislostí a dalších detailů. Následující popis a zkrácená ukázka 10.2 poskytují přehled o struktuře a klíčových prvcích tohoto formátu[5]:

- `<project>` – Kořenový prvek, který definuje projekt. Atribut `name` udává název projektu.
- `<tasks>` – Sekce, která obsahuje všechny úkoly projektu. Atribut `empty-milestones` určuje, zda jsou v projektu prázdné milníky.
- `<task>` – Prvek reprezentující jednotlivé úkoly – základní stavební kámeny Ganttova diagramu. Atributy jako `id`, `name`, `start`, `duration` a další specifikují vlastnosti úkolu.
- `<depend>` – Prvek, který definuje logickou závislost mezi úkoly. Atributy `id`, `type`, `difference` a `hardness` popisují charakteristiky závislosti.

Na ukázce 10.2 je vidět, že projekt obsahuje úkoly v hierarchické struktuře stejně jako v předchozí ukázce 10.1.

```
<project name="Welcome To WiseMapping" ...>
    <tasks empty-milestones="true">
        <task id="1" name="Welcome To WiseMapping" start="2024-05-22" duration="8" ...>
            <task id="2" name="Introduction" start="2024-05-22" duration="3" ...>
                <depend id="3" type="2" difference="0" hardness="Strong"/>
            </task>
            <task id="3" name="Basics" start="2024-05-27" duration="5" .../>
        </task>
    </tasks>
</project>
```

Ukázka kódu 10.2: Příklad `.gan` souboru.

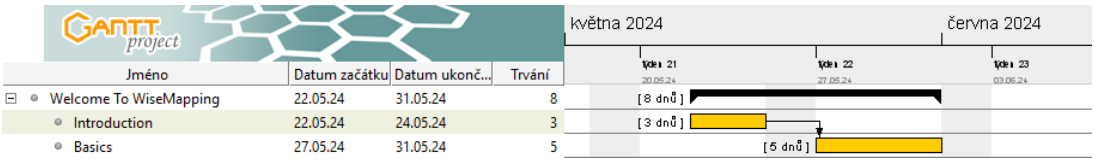
[3] U následujících prvků nejsou uvedeny jejich atributy pro přehlednost.
[4] `https://www.ganttproject.biz`
[5] U následujících prvků jsou uvedeny pouze hlavní atributy.

Mezi další prvky, které soubor může obsahovat a jsou mapovány pomocí Jackson XML (viz podsekce 9.2.4), patří[6]:

- `<description>` – Element pro popis projektu.
- `<view>` – Sekce pro nastavení zobrazení atributů Ganttova diagramu. Obsahuje nastavující prvky.
- `<calendars>` – Sekce pro kalendáře. Obsahuje prvek `<day-types>`, který v sobě obsahuje další prvky pro nastavení projektu.
- `<taskproperties>` – Sekce, která předepisuje vlastnosti úkolů. Obsahuje jednotlivé `<taskproperty>` prvky.
- `<resources>` – Sekce pro definici zdrojů.
- `<allocations>` – Sekce pro přidělení zdrojů k úkolům.
- `<vacations>` – Sekce pro definici dovolených v projektu.
- `<roles>` – Sekce pro definici rolí v projektu.

.gan soubor zobrazený nástrojem GanttProject

Na obrázku 10.2 je ukázán soubor s Ganttovým diagramem z ukázky 10.2 nástrojem GanttProject[4].



Obrázek 10.2: Ganttův diagram vytvořený `.gan` souborem.

10.1.3 Vzájemné porovnání

Z obou ukázek je na první pohled vidět podoba mezi formáty – úkoly a jejich hierarchické struktura. Fakt, že soubor obsahující Ganttův diagram má v sobě více informací, dává smysl vzhledem k přidanému časovému faktoru v diagramu oproti WBS.

10.2 Průběh konverze

Konverze v aplikaci probíhá dvěma směry – z WBS do Ganttova diagramu a naopak. V následujících podsekcích je detailně popsán průběh konverze v obou směrech. Na začátku každé konverze probíhá kontrola správnosti přípony nahraného souboru.

10.2.1 Konverze z WBS do Ganttova diagramu

Konverze z WBS do Ganttova diagramu je z obou směrů ten vývojově složitější a to hlavně kvůli potřebě součinnosti od uživatele, který konverzi provádí. Je zapotřebí doplnit informace, které ve WBS chybí – časové ohraničení jednotlivých úkolů (resp. celého projektu) a informaci o tom, jestli je úkol milníkem. Pro přehlednost je doplňování logických vazeb při procesu konverze vynecháno. Uživatel si jednotlivé vazby do Ganttova diagramu může poté doplnit v nástroji GanttProject[4], který je pro tuto činnost uzpůsobený.

[6] U následujících prvků nejsou uvedeny jejich atributy pro přehlednost.

Převod v tomto směru probíhá, oproti směru opačném, ve více krocích. Prvním je konverze `.mm` souboru do Dto reprezentujícího Ganttův diagram. Při tomto kroku dochází k nahraní souboru a jeho následné kontrole. Rekurzivně se prochází celý WBS a zjišťuje se, zdali u nějakého uzlu nebyl použit `richcontent` s typem `NODE`. Při nalezení takového uzlu dochází k vyhození výjimky. Toto ošetření bylo zavedeno hlavně z důvodu, že v takovém uzlu se může nacházet jakýkoli text v `XHTML` formátu – to znamená, že může obsahovat např. tabulku. Parsování takového uzlu by tedy bylo velmi obtížné a je rozumnější takový soubor vrátit uživateli k přepracování. V tomto kroku se z WBS přenáší hierarchická struktura úkolů a jejich názvy. Všechny ostatní potřebné informace se nastavují pomocí konstant – např. u všech úkolů je počáteční datum nastaveno na datum, kdy proběhla konverze a délka trvání na jeden den. Barva jednotlivých úkolů se odvíjí od toho, jestli jsou koncovými[7] (žlutá) nebo nekoncovými[8] (černá). `ID` úkolů se generuje od jedné místo jejich přenesení z nahraného souboru, protože nástroj FreeMind[2] při tvorbě myšlenkové mapy přiřazuje jednotlivým uzlům `ID` ve formě devítimístného čísla. Diagramu se ještě nastaví základní parametry pro správné zobrazování v nástroji GanttProject[4] a je vrácen.

Druhým krokem je zobrazení modálního okna obsahujícího zmíněný Dto Ganttova diagramu. V tomto okně může uživatel měnit parametry jednotlivých úkolů podle pravidel stanovených ve funkčních požadavcích (viz podsekce 7.3, konkrétně FRQ05, 06 a 07). Při aktualizaci úkolu také dojde k aktualizaci počátečních a koncových dat a délky trvání rodičovských úkolů. Délka trvání se počítá ze začátku a konce úkolu, při výpočtu se nezapočítávají víkendy. Když uživatel přemění úkol na milník, změní se i jeho barva na modrou. Když je úprava diagramu hotová, přechází uživatel ke třetímu kroku.

Ve třetím kroku může uživatel diagram v modálním okně převést do *CSV* souboru nebo ho konvertovat do finálního `.gan` souboru. V tomto kroku se může rozhodnout, zdali chce při převodu zachovat úkoly v zobrazeném pořadí nebo úkoly seřadit podle počátečního data. Uživatel může druhý krok přeskočit a diagram rovnou konvertovat.

Finální produkt konverze je soubor s příponou `.gan`, který se dá bezproblémově otevřít v desktopovém programu GanttProject[4].

10.2.2 Konverze z Ganttova diagramu do WBS

Tento směr konverze je ve srovnání s druhým směrem jednodušší, jelikož nevyžaduje přidávání jakýchkoli dalších informací. Po nahrání `.gan` souboru má uživatel možnost vybrat si způsob konverze. Může zvolit mezi konverzí se ztrátou nebo bez ztráty důležitých dat a také mezi konverzí, která zahrnuje nebo nezahrnuje milníky.

Při převodu se zachovává hierarchická struktura a názvy jednotlivých úkolů. Další atributy a prvky, např. pozice, se nastavují jako konstanty. Milníky se přenášejí pouze při výběru varianty převodu s milníky. Nedává smysl přenášet logické závislosti mezi úkoly, neboť pro výslednou WBS nemají přínos a mohly by výsledný diagram spíše znepřehlednit.

Aby se mohl Ganttův diagram správně převést do WBS, musí obsahovat **právě jeden** zastřešující prvek, který pak bude hlavním prvkem ve WBS. Proto probíhá při konverzi kontrola počtu úkolů v první úrovni Ganttova diagramu a pokud jejich počet překročí jeden, je vyhozena výjimka.

Výstup konverze záleží na volbě mezi ztrátovým a bezeztrátovým převodem. Při ztrátovém převodu je vrácen jeden soubor s příponou `.mm` obsahující WBS ve formě myšlenkové mapy. V druhém případě dojde ke stažení *ZIP* archivu se zmíněným `.mm` souborem společně s *CSV* souborem, který obsahuje data ztracená při konverzi. Patří k nim: datum začátku, konce, délka trvání, logické závislosti, barva a milník (pravda/nepravda). Uživatel si výsledný soubor poté může bezproblémově otevřít v desktopovém nástroji FreeMind[2].

[7] Úkol, který pod sebou nemá žádné další úkoly.

[8] Úkol, který pod sebou má další úkoly.

Kapitola 11

Testování aplikace

Následující kapitola se věnuje testování finální aplikace. Sekce 11.1 se věnuje systémovému testování a sekce 11.2 testování uživatelskému.

11.1 Systémové testy

V rámci systémových testů byla vytvořena řada End-to-end (E2E) testů pro ověření správnosti jednotlivých API endpointů. Testy slouží k zachování konzistence aplikace při jakékoli změně.

E2E je metoda testování, která vyhodnocuje celý průběh aplikace od začátku do konce. Zajišťuje, že všechny komponenty pracují tak, jak je očekáváno, a že softwarová aplikace funguje správně v reálných scénářích. Při E2E testování je software testován z perspektivy koncového uživatele, simulující reálný uživatelský scénář, včetně uživatelského rozhraní, backendových služeb, databází a komunikace v síti. Cílem E2E testování je ověřit celkové chování aplikace, včetně její funkčnosti, spolehlivosti, výkonu a bezpečnosti. [50]

Ve všech testech v aplikaci se začíná načtením vstupního souboru a končí porovnáním výsledného statusu a odpovědi, opět pomocí souboru. V ukázce 11.1 je zobrazeno, jak takový test vypadá. Vstupní a očekávaný soubor se načítají z testových zdrojů (`test/resources`). Test je možné uskutečnit díky třídě `MockMvc`, která je nakonfigurovaná ve třídě `BaseAT` a využívá jí každá testovací třída. Nacházejí se v ní všechna potřebná nastavení pro testování.

V každém testu se volá jeden endpoint, v ukázce 11.1 je to konkrétně `/wbs/upload`. Třída `MockMvc` poté při volání nasimuluje celý průběh aplikace.

```
1  class WBSToGanttTest extends BaseAT {
2      @Value("classpath:files/wbs_upload/success/test.mm")
3      private Resource wbsFile;
4      @Value("classpath:files/wbs_upload/success/expected_gantt.json")
5      private Resource expectedGantt;
6
7      private static final LocalDate DATE = LocalDate.of(2024, 4, 29);
8
9      @Test
10     void testConvertWBSToGantt() throws Exception {
11         MockMultipartFile file = new MockMultipartFile(...); //file load
12         try (...) { // settings for LocalDate mock
13             mockedLocalDate.when(LocalDate::now).thenReturn(DATE);
14             mockMvc.perform(MockMvcRequestBuilders.multipart("/wbs/upload")
15                             .file(file)
16                             .contentType(...))
17                     .andExpect(...) //status check
18                     .andExpect(...); //result check
19         }
20     }
21 }
```

Ukázka kódu 11.1: Příklad E2E testu.

11.2 Uživatelské akceptační testy

Systémové testy nikdy nedokáží plně otestovat a předpovědět chování uživatele v aplikaci, proto k nim je skvělým doplňkem testování uživatelské. Díky akceptačním testům se pokryjí další možné průchody aplikací a obdrží se cenná zpětná vazba na design výsledného produktu a na to, jak uživatelsky intuitivní je. Během testování se můžeme také setkat s novými požadavky koncových uživatelů, které lze začlenit do dodatečného vývoje.

Hlavním cílem tohoto uživatelského testování bylo ověření funkčnosti konvertoru. Vzhled je ke konvertoru pouze doplňkový.

11.2.1 Průběh testování

Uživatelského testování se zúčastnili celkem 4 participanti. Byli napřímo osloveni a pozváni na osobní testování, na kterém postupovali podle pokynů vypsaných ve dvou uživatelských scénářích (viz příloha C). Tyto scénáře byly vytvořeny za účelem otestování všech procesů v aplikaci. Konkrétně dvou hlavních – převodu souboru obsahujícího WBS do souboru s Gantovým diagramem. Na první jmenovaný proces byl kladen větší důraz, protože při konverzi dochází k doplňování informací uživatelem, a tudíž je větší pravděpodobnost, že v průběhu nastane neočekávaná chyba.

Při testování využívali participanti předem připravený počítač, na kterém běžela aplikace a byly v něm nachystány testovací soubory, pomocí kterých mohli testování uskutečnit.

V průběhu testování byly participantům postupně kladeny otázky z dotazníku vytvořeného pro účely tohoto testování (viz příloha D). V první části dotazníku se zjišťovaly předešlé zkušenosti uživatele. V druhé byl pak pomocí otázek zhodnocen průběh testování. Odpovědi jednotlivých participantů jsou rovněž zahrnuty v již zmíněném dotazníku.

11.2.2 Výsledky testování

Z odpovědí (viz příloha D) participantů testování byly identifikovány tyto připomínky:

1. Je zapotřebí vylepšit nápovědy v aplikaci.
2. Je zapotřebí přidat nápovědu k jednotlivým možnostem převodu ve směru Gantt → WBS.
3. Hierarchické znázornění není srozumitelné a působí jako chyba.
4. Chybně se aktualizují úkoly v modálním okně.
5. Je zapotřebí vylepšit ovladatelnost programu pomocí klávesnice.
6. Není srozumitelné, podle kterého data se úkoly při konverzi řadí.
7. Chybí ukázka seřazení úkolů ještě před převodem.

Velmi pozitivním faktorem bylo zjištění, že veškeré připomínky a nalezené chyby byly způsobeny pouze FE částí aplikace. U aplikace, která měla za hlavní cíl vytvořit kvalitní a funkční BE, jsou výsledky uživatelského akceptačního testování zcela přijatelné a rozhodně se dají považovat za úspěšné. Navíc byly všechny vypsané připomínky až na poslední již zakomponovány do finální aplikace. Poslední připomínka nebyla do aplikace integrována z důvodu náročnosti implementace a nedostatku času.

Proces uživatelského testování by byl potřeba minimálně ještě jednou zopakovat, aby se dalo zhodnotit, zdali byly připomínky správně pochopeny a opraveny.

Z uživatelských testů vyplývá, že projektové kancelář a v ní obsažený konvertor splňuje požadavky stanovené na začátku této práce a nabízí efektivní plánování a řízení projektů pomocí konverze dat mezi vybranými nástroji.

Kapitola 12

Závěr

Tato bakalářská práce zkoumala a implementovala řešení projektové kanceláře využívající volně dostupné nástroje pro plánování a řízení menších projektů za použití vodopádové metody, Work Breakdown Structure (WBS) a Ganttova diagramu. Cílem bylo integrovat tyto nástroje do jednotného systému, který by usnadnil a zefektivnil projektové plánování a řízení.

Nejprve byly analyzovány různé volně dostupné nástroje a jejich formáty souborů. Na základě této analýzy byly vybrány dva nástroje: GanttProject pro tvorbu a ukládání Ganttova diagramu a FreeMind pro tvorbu a ukládání WBS. Následně bylo navrženo a implementováno řešení, které umožňuje konverzi dat mezi těmito dvěma nástroji, což zajišťuje jejich vzájemnou kompatibilitu a integritu dat.

V průběhu práce bylo nutné čelit několika výzvám, zejména při hledání vhodného nástroje pro konverzi dat mezi WBS a Ganttovým diagramem. Bylo zjištěno, že neexistuje žádný volně dostupný nástroj, který by plně vyhovoval požadavkům, a proto byl navržen a implementován vlastní konvertor na zelené louce. Tento konvertor byl navržen tak, aby byl bezstavový, což znamená, že konverze dat probíhá v reálném čase bez nutnosti ukládání dat do mezipaměti, což zajišťuje vyšší škálovatelnost a nižší využití zdrojů.

Implementované řešení bylo úspěšně systémově a uživatelsky otestováno. Výsledky ukázaly, že se díky integraci zmíněných nástrojů do jednoho celku a vytvoření konvertoru podařilo realizovat řešení, které zefektivní plánování a řízení menších projektů.

Tato práce ukázala, že i s využitím volně dostupných nástrojů lze vytvořit efektivní systém pro podporu projektového plánování a řízení. Výsledky této práce mohou sloužit jako základ pro další vývoj a rozšíření funkcionalit, například implementaci dalších nástrojů nebo metod pro řízení projektů.

12.1 Budoucí vývoj aplikace

Aplikaci lze v budoucnu nasadit do produkce a rozšířit o další funkcionality, které mohou ještě více pomoci projektovým manažerům malých projektů při jejich plánování a řízení.

12.1.1 Nasazení aplikace do produkce

Aplikace není momentálně nasazena do produkce. Nasazení aplikace do produkce znamená, že bude moci být využita projektovými manažery na skutečných projektech, což výrazně zvýší její užitečnost a přínos. Tento krok zahrnuje nejen samotné technické nasazení aplikace, ale také přípravu prostředí a vytvoření uživatelské dokumentace pro budoucí uživatele.

12.1.2 Automatizace převodu ve směru z WBS do Ganttova diagramu

Projektová kancelář není ve směru konverze z WBS do Ganttova diagramu uzpůsobena na vkládání dodatečných dat, která by obsahovala např. počáteční a koncová data jednotlivých úkolů. Tato varianta by mohla převod v tomto směru ještě více usnadnit a zrychlit.

12.1.3 Modální okno pro úpravu Ganttova diagramu

Modální okno na doplňování informací do Ganttova diagramu ve směru konverze z WBS do Ganttova diagramu by mohlo být více interaktivní. Mohlo by uzpůsobovat změnu pořadí jednotlivých úkolů nebo uživatelsky přívětivě umožnit doplnění logických vazeb mezi úkoly.

12.1.4 Seřazení úkolů v modálním okně pro úpravu Ganttova diagramu

V modálním okně by se mohly úkoly seřazovat a ihned ukazovat podle počátečního data pomocí dedikovaného tlačítka, aby byl lépe předvídatelný Ganttův diagram na výstupu.

Literatura

[1] Kerzner, H. (2022). *Project Management Case Studies* [e-kniha]. 6. aktualiz. vyd. John Wiley & Sons. 816 s. Dostupné z: https://app.knovel.com/kn/resources/kpPMCSE027/toc

[2] Larson, R. (2004). *The critical steps to managing small projects.* In: PMI®Global Congress 2004–EMEA. Praha, Česká republika [online]. Newtown Square, PA: Project Management Institute. [cit. 23.05.2024] Dostupné z: https://www.pmi.org/learning/library/unique-challenges-managing-small-project-8439

[3] Project Management Institute, Inc. (PMI). (2013). *A Guide to the Project Management Body of Knowledge (PMBOK®Guide)* [e-kniha]. 5. aktualiz. vyd. Project Management Institute, Inc. (PMI). 589 s. Dostupné z: https://www.academia.edu/43812564/PMBOK_Guide_Fifth_Edition

[4] Hoory, L & Bottorff, C.. (2022). *Agile Vs. Waterfall: Which Project Management Methodology Is Best For You?.* In: Forbes Advisor – Business [online]. 10.08.2022 [cit. 17.05.2024]. Dostupné z: https://www.forbes.com/advisor/business/agile-vs-waterfall-methodology

[5] Devi, T. & Reddy, V. (2012). *Work Breakdown Structure of the Project* [e-kniha]. 2. aktualiz. vyd. International Journal of Engineering Research and Applications (IJERA). Dostupné z: https://www.yumpu.com/en/document/view/8124745/work-breakdown-structure-of-the-project-international-journal-of-

[6] Freeman, J. (rok neuveden). *What is Work Breakdown Structure (WBS) Diagram?.* In: Edrawsoft [online]. [cit. 19.05.2024]. Dostupné z: https://www.edrawsoft.com/what-is-work-breakdown-structure-diagram.html

[7] Autor neuveden (2016). *WBS (Work Breakdown Structure).* In: ManagementMania [online]. 03.08.2016 [cit. 05.11.2023]. Dostupné z: https://managementmania.com/cs/work-breakdown-structure

[8] Project Management Institute, Inc. (PMI). (2019). *Practice Standard for Work Breakdown Structures (3rd Edition)* [e-kniha]. Project Management Institute, Inc. (PMI). 100 s. Dostupné z: https://app.knovel.com/hotlink/toc/id:kpPSWBSE11/practice-standard-work/practice-standard-work

[9] Cadle, J. & Yeates, D. (2008). *Project management for information systems* [e-kniha]. 5. aktualiz. vyd. Pearson Education Limited. 465 s. Dostupné z: https://www.greatertzaneen.gov.za/documents/news/Project%20management%20for%20information%20system%205th.pdf

[10] Drbohlavová, T. (2023). *Myšlenková mapa v podnikání.* In: OrangeAcademy – Články [online]. 07.07.2023 [cit. 17.05.2024]. Dostupné z: https://orangeacademy.cz/clanky/myslenkova-mapa-2

[11] Autor neuveden (2015). *Ganttův diagram (Gantt Chart)*. In: ManagementMania [online]. 30.07.2015 [cit. 05.11.2023]. Dostupné z: https://managementmania.com/cs/ganttuv-diagram

[12] Kiran, D. R. (2019). *Production Planning and Control - A Comprehensive Approach* [e-kniha]. Elsevier. 582 s. Dostupné z: https://app.knovel.com/hotlink/toc/id:kpPPCACA02/production-planning-control/production-planning-control

[13] Ramos, D. (2021). *Convert Work Breakdown Structures to Gantt Charts for Project Success*. In: SmartSheet – Content Center [online]. 08.12.2021 [cit. 20.05.2024]. Dostupné z: https://www.smartsheet.com/content/wbs-gantt

[14] Autor neuveden (2022). *Task management vs project management: How do they work together?*. In: MindManager – Blog [online]. 21.12.2022 [cit. 19.05.2024]. Dostupné z: https://www.pmconsulting.cz/2021/08/kanban-stihla-metoda-pro-spravu-a-zlepsovani-prace

[15] Mariam, A. (2023). *What is Data Conversion: Techniques, Tools, and Best Practices*. In: Astera – Blog [online]. 25.08.2023 [cit. 06.05.2024]. Dostupné z: https://www.astera.com/type/blog/data-conversion

[16] Autor neuveden (2023). *Stateful vs stateless*. In: Red Hat – Topics [online]. 21.12.2023 [cit. 06.05.2024]. Dostupné z: https://www.redhat.com/en/topics/cloud-native-apps/stateful-vs-stateless

[17] Autor neuveden (rok neuveden). *WBS – klíčový nástroj pro úspěch projektu*. In: PM Consulting [online]. [cit. 26.12.2023]. Dostupné z: https://www.pmconsulting.cz/pm-wiki/wbs/?fbclid=IwAR0TbodO5oT5UEz9JGUMQHf_dmBKUfiTv3XhkyO2YVaJ-_pb7Itky-luSKU

[18] Mau, D. (2020). *Základní průvodce Ganttovými diagramy*. In: Microsoft – Obchodní přehledy a nápady [online]. [cit. 26.12.2023]. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-365/business-insights-ideas/resources/gantt-chart-guide

[19] Autor neuveden (2016) *PMBOK (Project Management Body of Knowledge)*. In: ManagementMania [online]. 24.06.2016 [cit. 11.12.2023]. Dostupné z: https://managementmania.com/cs/project-management-body-of-knowledge

[20] Autor neuveden (rok neuveden). *Ganttův diagram a závislosti úkolů*. In: Caflou – Vzdělávací centrum [online]. [cit. 26.12.2023]. Dostupné z: https://www.caflou.cz/ganttuv-diagram-a-zavislosti-ukolu

[21] Doležal, J. (2021). *KANBAN – štíhlá metoda pro správu a zlepšování práce*. In: PM Consulting – Blog [online]. 30.08.2021 [cit. 19.05.2024]. Dostupné z: https://blog.mindmanager.com/task-management-vs-project-management

[22] Kukhnavets, P. (2022). *Gantt Chart vs. WBS: Choosing From Two Reliable Ways to Plan Your Project*. In: GanttPRO – Project Management Blog [online]. 25.10.2022 [cit. 05.11.2023]. Dostupné z: https://blog.ganttpro.com/en/gantt-chart-vs-work-breakdown-structure-wbs

[23] Bento, A. (2021). *Tutorial: Convert you notes into a WBS and Gantt charts with PlantUML*. In: Idea Shortcut [online]. 03.11.2021 [cit. 05.11.2023]. Dostupné z: https://ideashortcut.com/tutorial-convert-you-notes-into-a-wbs-and-gantt-charts-with-plantuml

[24] Dayani, M., Gelbard, R. (2015). *Automatic Conversion of Software Specification into a Gantt-chart subject to Organization's Constraints* [online]. Procedia Computer Science.

Článek byl prezentován na ProjMAN 2015, Vilamoura, Portugalsko. [cit. 04.12.2023]. Dostupné z: `https://www.sciencedirect.com/science/article/pii/S1877050915026009`

[25] Royzen, M. & Wei, J. (2022) *Chatová konverzace s Phind (GPT-4)* [chat log]. [cit. 03.12.2023]. Dostupné z: `https://www.phind.com/search?cache=pafvk6sbgsq8a95t19ne0m1t&source=sidebar`

[26] Royzen, M. & Wei, J. (2022) *Chatová konverzace s Phind (GPT-4)* [chat log]. [cit. 03.12.2023]. Dostupné z: `https://www.phind.com/search?cache=io3gtrqgrs83y9c66m0yinzx&source=sidebar`.

[27] Autor neuveden (2023). *Frontend vs Backend*. In: Geeks for Geeks [online]. 18.04.2023 [cit. 06.05.2024]. Dostupné z: `https://www.geeksforgeeks.org/frontend-vs-backend`

[28] Autor neuveden (rok neuveden). *What can PHP do?*. In: PHP – Documentation [online]. [cit. 06.05.2024]. Dostupné z: `https://www.php.net/manual/en/intro-whatcando.php`

[29] Autor neuveden (rok neuveden). *What is Java?*. In: IBM – Topics [online]. [cit. 06.05.2024]. Dostupné z: `https://www.ibm.com/topics/java`

[30] Autor neuveden (rok neuveden). *What is Python? Executive Summary*. In: Python – Essays [online]. [cit. 06.05.2024]. Dostupné z: `https://www.python.org/doc/essays/blurb/`

[31] Autor neuveden (2023). *A Guide to XML in Java*. In: Baeldung [online]. 28.09.2023 [cit. 21.05.2024]. Dostupné z: `https://www.baeldung.com/java-xml`

[32] Autor neuveden (rok neuveden). *Reading XML Data into a DOM*. In: Oracle – Java Documentation [online]. [cit. 21.05.2024]. Dostupné z: `https://docs.oracle.com/javase%2Ftutorial%2F/jaxp/dom/readingXML.html`

[33] Autor neuveden (rok neuveden). *Java DOM Parser – Overview*. In: TutorialsPoint – Java XML Tutorial [online]. [cit. 06.05.2024]. Dostupné z: `https://www.tutorialspoint.com/java_xml/java_dom_parser.htm`

[34] Autor neuveden (rok neuveden). *JAXB*. In: Java EE – Migrated Projects [online]. [cit. 06.05.2024]. Dostupné z: `https://javaee.github.io/jaxb-v2`

[35] Autor neuveden (rok neuveden). *Jackson XML*. In: FasterXML – GitHub [online]. [cit. 06.05.2024]. Dostupné z: `https://github.com/FasterXML/jackson-dataformat-xml`

[36] Autor neuveden (2024). *Performance of Java Mapping Frameworks*. In: Baeldung [online]. 11.05.2023 [cit. 21.05.2024]. Dostupné z: `https://www.baeldung.com/java-performance-mapping-frameworks`

[37] Autor neuveden (rok neuveden). *MapStruct*. In: MapStruct [online]. [cit. 08.05.2024]. Dostupné z: `https://mapstruct.org`

[38] Autor neuveden (rok neuveden). *JMapper*. In: JMapper – GitHub Wiki [online]. [cit. 08.05.2024]. Dostupné z: `https://github.com/jmapper-framework/jmapper-core/wiki`

[39] Autor neuveden (rok neuveden). *ModelMapper*. In: ModelMapper [online]. [cit. 08.05.2024]. Dostupné z: `https://modelmapper.org`

[40] Autor neuveden (2024). *React Tutorial*. In: Geeks for Geeks [online]. 16.04.2024 [cit. 08.05.2024]. Dostupné z: `https://www.geeksforgeeks.org/react-tutorial`

[41] Vijay, P. (2023). *What is Svelte?*. In: Svelte – Introduction [online]. 25.05.2023 [cit. 08.05.2024]. Dostupné z: `https://svelte.dev/tutorial/basics`

[42] Autor neuveden (2023). *What is Angular?*. In: Angular – Getting started [online]. 15.08.2023 [cit. 08.05.2024]. Dostupné z: `https://angular.io/guide/what-is-angular`

[43] Harris, Ch. (rok neuveden). *Microservices vs. monolithic architecture*. In: Atlassian – Microservices architecture [online]. [cit. 08.05.2024]. Dostupné z: `https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith`

[44] Autor neuveden (2024). *MVC Design Pattern*. In: Geeks for Geeks [online]. 19.02.2024 [cit. 09.05.2024]. Dostupné z: `https://www.geeksforgeeks.org/mvc-design-pattern`

[45] Bierman, G. (2024). *JEP 395: Records*. In: OpenJDK [online]. 03.02.2024 [cit. 09.05.2024]. Dostupné z: `https://openjdk.org/jeps/395`

[46] Autor neuveden (rok neuveden). *Project Lombok*. In: Project Lombok [online]. [cit. 09.05.2024]. Dostupné z: `https://projectlombok.org`

[47] Autor neuveden (rok neuveden). *OpenAPI Specification*. In: Swagger – Specification [online]. [cit. 11.05.2024]. Dostupné z: `https://swagger.io/specification/v2`

[48] Wang, R. (2024). *Svelte components*. In: Svelte – Docs [online]. 08.04.2024 [cit. 14.05.2024]. Dostupné z: `https://svelte.dev/docs/svelte-components`

[49] Farley, J. (2023). *Basic markup*. In: Svelte – Docs [online]. 16.09.2023 [cit. 11.05.2024]. Dostupné z: `https://svelte.dev/docs/basic-markup`

[50] Bose, S. (2023). *What is End To End Testing?*. In: BrowserStack – Guide [online]. [cit. 12.05.2024]. Dostupné z: `https://www.browserstack.com/guide/end-to-end-testing`

Příloha A

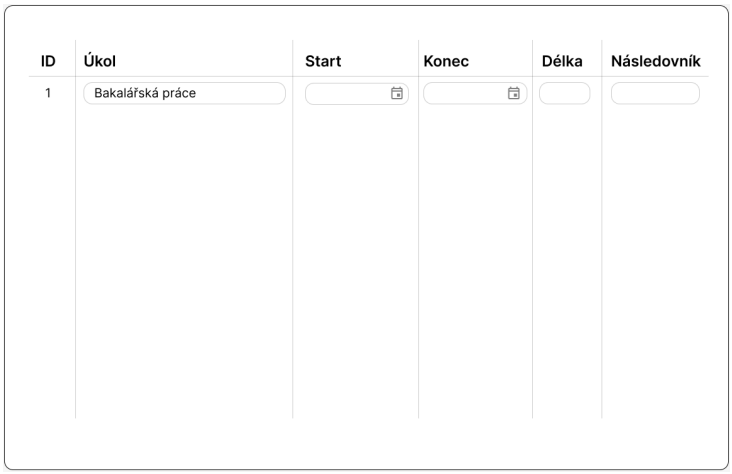
Grafický návrh aplikace

Pro návrh aplikace byl použit nástroj Figma[1].

Obrázek A.1: Úvodní obrazovka aplikace.

Obrázek A.2: Obrazovka po nahrání souboru.

[1] `https://www.figma.com`

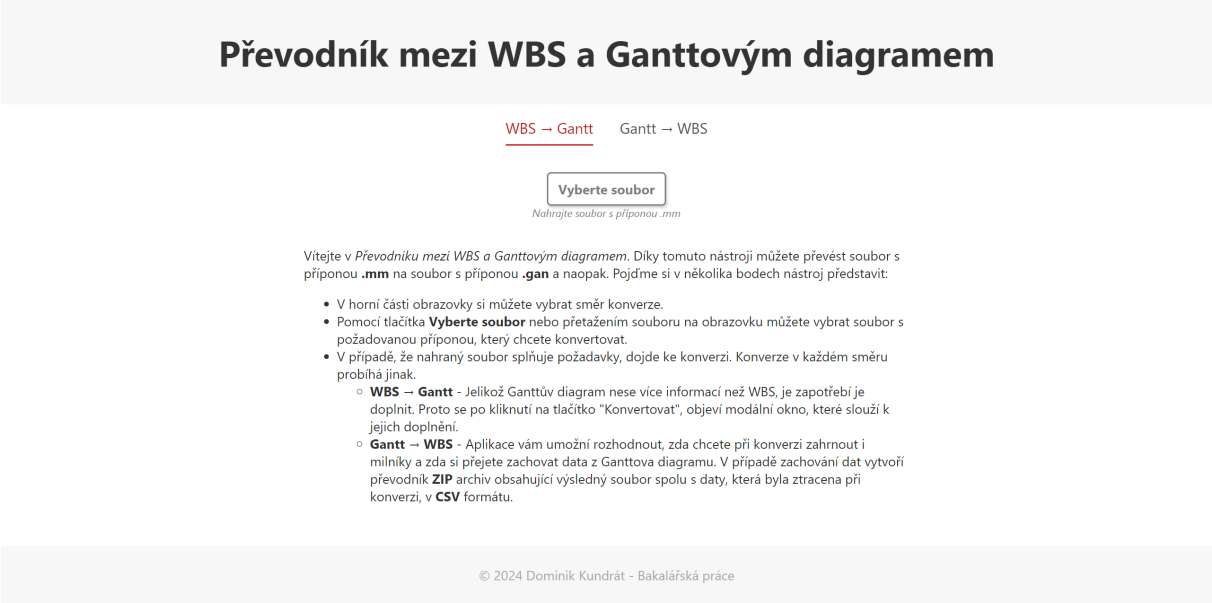


Obrázek A.3: Modální okno, pro úpravu aktivit při konverzi z WBS do Ganttova diagramu.

Obrázek A.4: Obrazovka na stažení konvertovaného souboru.

Příloha B

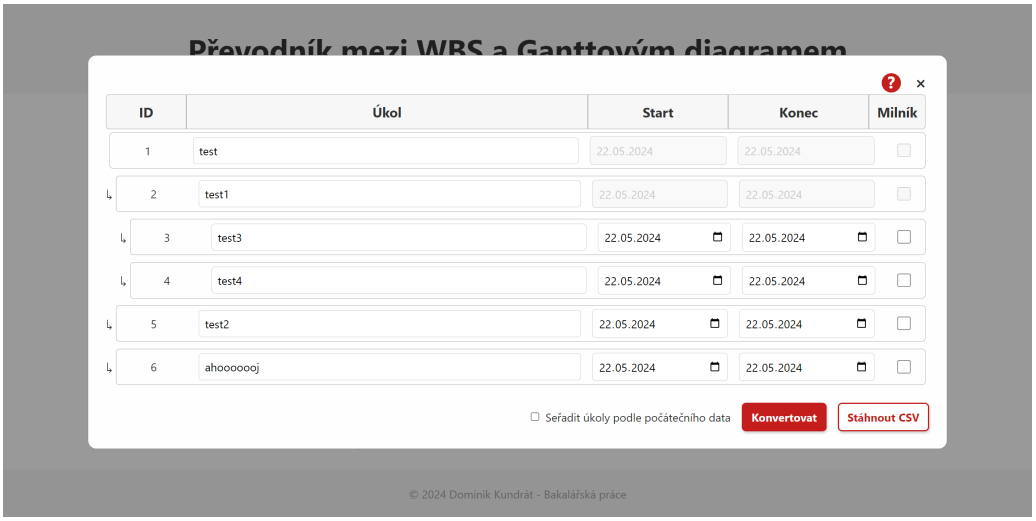
Uživatelské rozhraní konvertoru



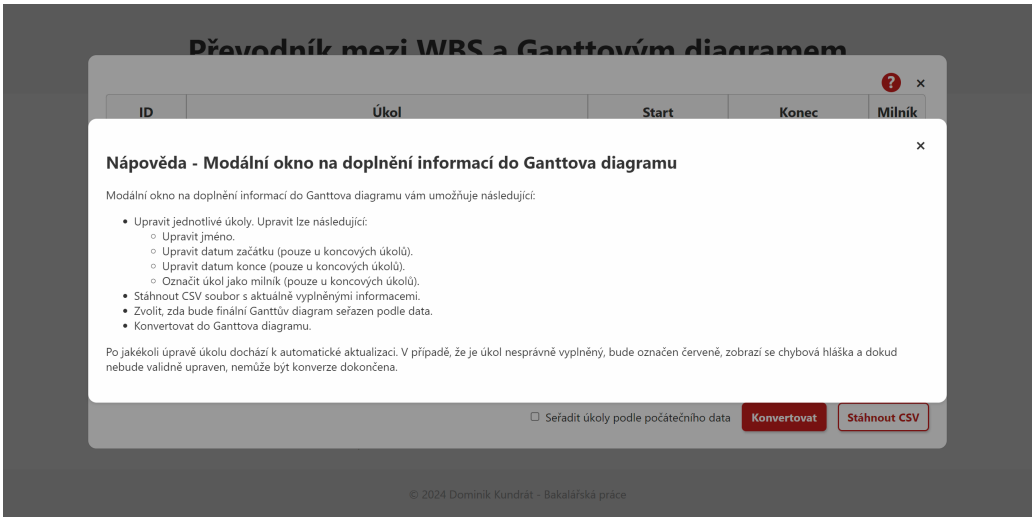
Obrázek B.1: Úvodní obrazovka aplikace.

Obrázek B.2: Obrazovka aplikace po nahrání `.mm` souboru.

Obrázek B.3: Obrazovka aplikace po nahrání `.gan` souboru.



Obrázek B.4: Modální okno pro úpravu jednotlivých úkolů v Ganttově diagramu.



Obrázek B.5: Nápověda v modálním okně.

Příloha C

Testovací scénáře

C.1 Testovací scénář převodu z Ganttova diagramu do WBS

Parametr	**Obsah**
ID testu	GANTT_WBS_01
Název testu	Konverze z Ganttova diagramu na WBS
Shrnutí testu	Soubor s příponou `.gan` bude úspěšně konvertován na soubor s příponou `.mm`.
Popis testu	Uživatel nahraje testovací soubor s příponou `.gan` obsahující Ganttův diagram. Po nahrání souboru si uživatel zvolí variantu: • Bez převodu milníků, bez zachování dat. • S převodem milníků, bez zachování dat. • Bez převodu milníků, se zachováním dat. • S převodem milníků, se zachováním dat. Nakonec uživatel pomocí tlačítka „Konvertovat“ převede diagram do `.mm` souboru.
Vstupní podmínky	Žádné.
Testovací data	Soubor s příponou `.gan` (`gantt.gan`)
Očekávaný výsledek	Objeví se hláška „Soubor byl úspěšně konvertován do WBS“ a dojde ke stažení v případě bez zachování dat (1. a 2.), souboru s příponou `.mm` (`wbs.mm`) nebo, v opačném případě (3. a 4.), *ZIP* archivu obsahující *CSV* soubor s daty z Ganttova diagramu (`tasks.csv`) a soubor s příponou `.mm` (`wbs.mm`). Výsledný `.mm` soubor lze pak bezproblémově otevřít pomocí nástroje *FreeMind*.

Tabulka C.1: Testovací scénář převodu z Ganttova diagramu do WBS.

C.2 Testovací scénář převodu z WBS do Ganttova diagramu

Parametr	**Obsah**
ID testu	WBS_GANTT_01
Název testu	Konverze z WBS na Ganttův diagram
Shrnutí testu	Soubor s příponou `.mm` bude úspěšně konvertován na soubor s příponou `.gan`.
Popis testu	Uživatel nahraje testovací soubor s příponou `.mm` obsahující WBS ve formě myšlenkové mapy. Po nahrání souboru uživatel klikne na tlačítko „Konvertovat“. Následně se zobrazí dialogové okno s úkoly, které jsou hierarchicky uspořádány a seřazeny podle struktury testovacího souboru. Uživatel vyzkouší následující povolené: • Upravit jméno tak, aby nebylo prázdné. • Upravit datum začátku tak, aby bylo před datem konce. • Upravit datum konce tak, aby bylo po datu začátku. • Označit úkol jako milník. a nepovolené změny: • Upravit jméno tak, aby bylo prázdné. • Upravit datum začátku tak, aby bylo po datu konce. • Upravit datum konce tak, aby bylo před datem začátku. Uživatel si pomocí tlačítka „Stáhnout CSV“ stáhne soubor ve formátu CSV obsahující vyplněná data z modálního okna. Nakonec si uživatel zvolí, že chce úkoly seřadit podle data a kliknutím na tlačítko „Konvertovat“ převede diagram do `.gan` souboru.
Vstupní podmínky	Žádné.
Testovací data	Soubor s příponou `.mm` (`wbs.mm`)
Očekávaný výsledek	Objeví se hláška „Soubor byl úspěšně konvertován do Ganttova diagramu“ a dojde ke stažení souboru s příponou `.gan` (`gantt.gan`), který má správně seřazené úkoly podle data a lze ho bezproblémově otevřít v nástroji *GanttProject*.

Tabulka C.2: Testovací scénář převodu z WBS do Ganttova diagramu.

Příloha D

Testovací dotazník a souhrn výsledků

1. **Působil/a jste někdy jako projektový vedoucí?**
   - Ano – 1x
   - Ne – 3x
2. **Účastnil/a jste se někdy plánování projektu?**
   - Ano – 3x
   - Ne – 1x
3. **Setkal/a jste se někdy s pojmy WBS a Ganttův diagram?**
   - Ano – 3x
   - Ne – 1x
4. **Pracoval/a jste někdy s těmito diagramy?**
   - Ano – 3x
   - Ne – 1x
5. **Jak jste vnímal/a proces převodu z WBS na Ganttův diagram?**
   (1 – velmi intuitivní, 6 – velmi neintuitivní)
   - 1
   - 2 – 3x
   - 3
   - 4 – 1x
   - 5
6. **Jak jste vnímal/a proces převodu z Ganttova diagramu na WBS?**
   (1 – velmi intuitivní, 6 – velmi neintuitivní)
   - 1
   - 2 – 2x
   - 3
   - 4 – 2x
   - 5

7. **Jaké překážky jste při používání tohoto převodníku zažil/a?**
   - Neintuitivní odsazení, nenalezl jsem nápovědu v modálním okně.
   - Aplikace není dobře přizpůsobena na procházení pomocí klávesnice. Nenalezl jsem nápovědu.
   - Nenalezl jsem nápovědu.
   - Nápovědy. Aktualizace úkolů nefunguje, jak bych očekával (neukládají se správně vložené hodnoty).
8. **Jaké funkce nebo vlastnosti byste v převodníku vylepšil/a?**
   - Přidal bych nápovědu k možnostem ve směru Gantt → WBS.
   - Vylepšil bych ovladatelnost programu pomocí klávesnice.
   - Dal bych nápovědy na lépe viditelné místo (např. tlačítko na nápovědu v modálním okně bych dal ke křížku).
   - Vylepšil bych celkově nápovědy. Dodal bych podle jakého data se úkoly řadí ve směru WBS → Gantt.
9. **Jaké funkce nebo vlastnosti Vám při užívání převodníku chyběly?**
   - Přidal bych nápovědu k možnostem ve směru Gantt → WBS.
   - Přidal bych ukázku úkolů po seřazení ještě před převodem.
   - Přidal bych nápovědu k jednotlivým možnostem ve směru Gantt → WBS. Přidal bych ještě návaznosti v modálním okně.
   - Přidal bych nějaké představení stránky a nápovědu k jednotlivým možnostem ve směru Gantt → WBS.
10. **Použili byste kombinaci nástrojů (FreeMind, GanttProject), které aplikace využívá, nebo byste se raději přiklonili k jinému nástroji/postupu? Pokud pro Vás platí druhý případ, proč?**
    - Proč ne. S FreeMindem jsem se ale nikdy nesetkal.
    - Proč ne. Neznám jiné možnosti.
    - Pro plánování projektu bych použil pouze GanttProject, ve WBS nevidím příliš smysl.
    - Nevím.
11. **Máte nějaké další komentáře, které byste rád/a sdíleli?**
    - Celkově je aplikace pěkná, ale chce to vylepšit nápovědu.
    - Aplikace má několik mušek, nejvíce se mi líbila drag'n'drop funkce. Celkově je aplikace dobře ošetřená proti špatným vstupům.
    - Kdybych udělal WBS a chtěl bych ho převést do Ganttova diagramu, tak mi přijde aplikace jako příjemné ulehčení. Opačný směr konverze mi nedává smysl.
    - Ne.

Příloha E

Struktura a obsah přiložených souborů

- wbs-gantt-converter.zip
  - .mvn → Konfigurace specifické pro Maven wrapper, který umožňuje spouštění Maven projektů bez nutnosti mít Maven nainstalovaný na systému
  - app → Frontendová část aplikace
    - public → Veřejně dostupné statické soubory, jako jsou obrázky, HTML soubory apod.
    - scripts arrow → Skripty potřebné pro běh aplikace, jako jsou build nebo deployment skripty
    - src → Hlavní zdrojové kódy frontendu aplikace
    - package.json → Konfigurační soubor pro Node.js a npm, který obsahuje metadata o projektu, závislosti a skripty pro běh a build aplikace
    - package-lock.json → Zámkový soubor pro npm, který přesně specifikuje verze závislostí, aby bylo zajištěno, že instalace bude konzistentní napříč různými prostředími
    - rollup.config.js → Konfigurační soubor pro Rollup, modulární bundler JavaScriptu, který definuje, jak budou moduly zkompilovány
  - files → Složka obsahující zmíněné přílohy v práci, například diagram tříd nebo textová podoba souboru
  - src → Zdrojové kódy backendové části aplikace
    - main → Hlavní zdrojové kódy backendu aplikace
    - test → Složka obsahující testovací kódy a testovací sady, které ověřují správnost a funkčnost aplikace
  - BP-Kundrát.pdf → Plný text bakalářské práce, který obsahuje veškeré textové a grafické materiály související s projektem
  - latex.zip → Archiv obsahující zdrojový kód psané podoby práce v LaTeXu, včetně všech potřebných souborů pro kompilaci a úpravy dokumentu
  - mvnw → Unixový skript pro Maven wrapper, umožňuje spuštění Maven projektů na Unixových systémech bez nutnosti mít Maven nainstalovaný
  - mvnw.cmd → Windows skript pro Maven wrapper, umožňuje spuštění Maven projektů na Windows systémech bez nutnosti mít Maven nainstalovaný
  - pom.xml → Primární konfigurační soubor pro Maven, který obsahuje informace o projektu, závislostech, pluginech a dalších konfiguracích potřebných pro build procesu
  - README.md → Dokumentace projektu, která poskytuje přehled o účelu, instalaci, použití a dalších důležitých informacích souvisejících s projektem