VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NÁVRH A IMPLEMENTACE NÁSTROJE PRO PLÁNOVÁNÍ PROJEKTŮ A ČASU S VYUŽITÍM PRINCIPŮ Z OBLASTI PLÁNOVÁNÍ ÚLOH REÁLNÉHO ČASU

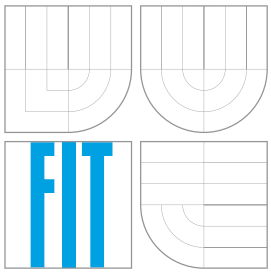
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADKA MOKRÁ

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NÁVRH A IMPLEMENTACE NÁSTROJE PRO PLÁNOVÁNÍ PROJEKTŮ A ČASU S VYUŽITÍM PRINCIPŮ Z OBLASTI PLÁNOVÁNÍ ÚLOH REÁLNÉHO ČASU

DESIGN AND IMPLEMENTATION OF PROJECT AND TIME MANAGEMENT TOOL BASED ON REAL-TIME TASK SCHEDULING PRINCIPLES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADKA MOKRÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JOSEF STRNADEL, Ph.D.

BRNO 2015

Abstrakt

Tato diplomová práce se zabývá návrhem aplikace pro plánování projektů a času s využitím mechanismů plánování úloh reálného času. V úvodu práce je popsáno řízení projektů a času společně s principy, které se k tomu využívají. V další kapitole jsou potom popsány principy a algoritmy plánování úloh reálného času. Následující kapitola je věnována návrhu aplikace pro plánování projektů a času s využitím mechanismů plánování úloh reálného času. V další kapitole je popsán návrhový vzor MVC. Implementace nástroje je popsána v předposlední kapitole a kapitola poslední je věnována příkladům použití vytvořeného nástroje.

Abstract

This master's thesis deals with design of application for scheduling projects and time using principles of real time scheduling. In the beginning this thesis describes project management and time management and their principles. In the next chapter are principles and algorithms of scheduling in real time system described. The next chapter deals with design of application for scheduling projects and time using principles of real time scheduling. MVC design pattern is described in following chapter.Implementation of tool is desribed in next-to-last chapter a the last chapter is devoted to use cases of final tool

Klíčová slova

projekt, řízení projektů, řízení času, úloha reálného času, plánování úloh reáného času, DASA, MVC, Django, D3.js, Bootstrap, Ganttův graf

Keywords

project, project management, time management, real time task, real time scheduling, DASA, MVC, Django, D3.js, Bootstrap, Gantt chart

Citace

Radka Mokrá: Návrh a implementace nástroje pro plánování projektů a času s využitím principů z oblasti plánování úloh reálného času, diplomová práce, Brno, FIT VUT v Brně, 2015

Návrh a implementace nástroje pro plánování projektů a času s využitím principů z oblasti plánování úloh reálného času

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením pana Ing. Josefa Strnadela, Ph.D.

.....
Radka Mokrá
26. května 2015

Poděkování

Zde bych chtěla poděkovat vedoucímu práce.

Obsah

Kapitola 1

Úvod

To nejdůležitější, co v životě máme a co nejsme schopni ovlivnit, je čas. Proto je důležité s ním dobře nakládat. Pro nakládání s čase nám mohou pomáhat plánovací nástroje, které pomáhají drahocenný čas efektivně zorganizovat.

Tato diplomová práce se proto zabývá návrhem aplikace pro plánování projektů a času.

V této diplomové práci byly využity materiály, které byly zpracovány v rámci semestrálního projektu. Jedná se především o kapitoly 2, 3 a 4 přičemž v kapitole druhé byli provedeny pouze nepatrné změny a bylo zde doplněno několik obrázků. Kapitola tři nezaznamenala také výraznější změny, pouze zde byli doplněny algoritmy. Kapitola čtvrtá byla rozšířena především o podrobnější návrh aplikace.

V první kapitole je popsán úvod do plánování projektů včetně metod, které se k tomuto používají. Na konci kapitoly jsou potom uvedeny existující nástroje používaté k plánování projektů.

Následující kapitola se zabývá plánováním úloh reálného času. V úvodu kapitoly je popsáno jaké má parametry úloha reálného času a jak lze dělit plánovací algoritmy. V kapitole jsou potom dále popsány algoritmy a mechanismy, které se využívají při plánování úloh reálného času.

V další kapitole, která se zabývá návrhem samotné aplikace je v úvodu porovnána úloha v plánování projektů a úloha reálného času. Následně jsou potom navrženy parametry úlohy ve výsledné aplikaci.

Dále je potom v kapitole navržena výsledná aplikace, která bude implementována jako webová aplikace. V kapitole jsou vybrány vhodné plánovací mechanismy pro implementovaný plánovač a je zde rovněž návrh databáze pro výslednou aplikaci a také vzhled uživatelského rozhraní.

Pátá kapitola je potom věnována návrhovému vzoru MVC, pomocí kterého je výsledná aplikace implementována. V kapitole jsou ještě popsány nástroje pomocí kterých je výsledná aplikace implementována.

Předposlední kapitola je již věnována samotné implementaci nástroje a je zde především popsán výsledný plánovací mechanismus, který je v aplikaci nakonec použit.

Poslední kapitola ze zabývá použitím výsledné aplikace a jsou zde prezentovány příklady použití aplikace.

V závěru je potom uvedeno shrnutí a především návrhy na zlepšení výsledné aplikace.

Kapitola 2

Řizení projektů

V této kapitole bude popsáno a vysvětleno co je to projekt a řízení projektů. Dále pak budou podrobněji vysvetleny a popsány jednotlivé části z řízení projektů společne s nástroji a metodami k tomu určenými. Kapitola vychází z přednášek k předmětu MPR [2].

Na konci kapitoly budou představeny softwarové nástroje, sloužící k podpoře řízení projektů.

2.1 Pojmy projekt a řízení projektu

Dříve, než se začneme zabývat řízením projektů, je nutné ujasnit pojem projekt jako takový. Ve většině definic je projekt definován jako ohraničené časové usíli, které je vynaložené na vytvoření nějakého unikátního výstupu.

- V [10] je projekt definován následovně: Projekt je jedinečný časově, nákladově a zdrojově omezený proces realizovaný za účelem vytvoření definovaných výstupů (naplnění projektových cílů) v požadované kvalitě a v souladu s platnými standardy a odsouhlasenými požadavky.
- Nebo dle [16] je projekt:
  - síť činností mající formální začátek a konec, přidělené zdroje a směřující k vytvoření určitého produktu. Má také stanoven rozpočet, v rámci kterého musí být stanovených cílů dosaženo. S vytvořením tohoto produktu je vždy spojeno určité riziko.
  - jedinečný proces sestávající z řady koordinovaných a řízených činností s daty zahájení a ukončení, prováděný pro dosažení cíle, který vyhovuje specifickým požadavkům, včetně omezení daných časem, náklady a zdroji.

Z definic projektu lze vyvodit následující atibuty projektu[7]:

- *Projekt má jedinečný účel.*
- *Projekt je dočasný.*
- *Projekt se vytváří postupně.*
- *Projekt vyžaduje zdroje z nejrůznějších oblastí.*
- *Projekt by měl mít hlavního zákazníka.*

- *Projekt je rizikový.*

Po té, co je definován projekt, lze řízení projektů definovat podle [10] následovně:

- Aplikace znalostí, dovedností, nástrojů a technik na činnosti v projektu tak, aby projekt splnil požadavky na něj kladené. Zahrnuje plánování, organizování, monitorování a předávání zpráv o všech aspektech projektu a motivaci všech zúčastněných dosáhnout cílů projektu.

2.2 Životní cyklus projektu

Jak již bylo v definici řízení projektu uvedeno, řízení projektu má několik jednotlivých fázi, které nám usnadňují celkové řízení projektu.

Tyto fáze jsou nazývány životní cyklus projektu, který slouží pro definování vstupů a výstupů do a z jednotlivých etap a tím následně umožnuje kontrolu projektu. Životní cyklus projektu má následující fáze [2]:

- Fáze Definice
- Fáze Plánování
- Fáze Organizování
- Fáze Kontrola
- Fáze Ukončení

Těchto 5 fází bude blíže popsáno v následujících pěti podkapitolách.

2.2.1 Definice

Fáze definice je první fází u každého projektu. V této fázi dochází k formování projektu a prověření, zda je vůbec reálné projekt uskutečnit.

Studie příležitosti

Studie příležitostí umožňuje nahlédnou na projekt a jeho cíle z reálného pohledu. Umožňuje nám zvolit takové cíle, které lze splnit s ohledem na omezení, jako jsou například náklady, čas nebo dostupnost zdrojů.

Ve studii příležitosti se uplatňují následující analýzy [6]:

- SLEPT analýza – využívá se k analýze okolí projektu. Zkoumáním pěti hledisek (sociálního, legislativního, ekonomického, politického a technologického) hodnotí faktory ovlivňující projekt.
- SWOT analýza – skupinová analýza umožňující identifikovat příležitosti, hrozby, silné a slabé stránky projektu.

Specifikace cíle

Předpokladem k úspěšnému projektu je nutností mít správně definované cíle. Ke správné definici projektu je vhodné použití techniky SMART. Technika SMART pomáhá správně formulovat cíl projektu.

Správně definovaný cíl by měl mít dle techniky SMART následující náležitosti:

- **S – specifický** – Musí být jasně specifikované čeho chce projekt dosáhnout.
- **M – měřitelný** – Stanovení kritérií, která urči dosažení cíle projektu včetně definování jak se budou daná kritéria měřit.
- **A – akceptovatelný** – Stanovený cíl musí být přijatelný pro všechny zainteresované strany.
- **R – realistický** – Cíl projektu musí být reálně dosažitelný.
- **T – termínovaný** – Projek má stanovený začátek a konec.

Studie proveditelnosti

Tato metoda identifikuje možná řešení projektu, snaží se stanovit tu nejoptimálnější variantu. Pro přeběžnou variantu řešení projektu potom určuje následující požadavky na projekt:

- specifikace cíle,
- náklady a zdroje,
- základní časový plán,
- očekávaný přínos projektu,
- významná rizika, která se mohou vyskytnou v průběhu projektu.

Během studie proveditelnosti se většinou uplatňuje analýza zainteresovaných stran a analýza kritických faktorů úspěšnosti projektu.

Výstupem fáze definice by mělo být rozhodnutí o kontraktu.

2.2.2 Plánování

Během fáze plánování se provádí především identifikace úkolů a kritických aktivit. Odhady délky trvání, ceny a potřebných zdrojů včetně personálních.

Hierarchická struktura prací – WBS [6, 10]

Je technika, která umožňuje rozklad cíle projektu na úrovně, které postupně definují rozklad výsledku projektu. Na nejvyšší úrovni rozkladu je výsledek, jenž je postupně rozkládán na dílčí výsledky, až na nejnižší úroveň jednotlivých pracovních balíků.

Seznam činností

Hierarchická struktura prací rozložila projekt na jednotlivé pracovní balíky. Tyto pracovní balíky se následně rozloží do seznamu činností. U každé činnosti v seznamu jsou potom uvedeny potřebné zdroje a potřebný čas na provedení daného úkolu.

Metody odhadování

Metody odhadování se používají při stanovování dély trvání jednotlivých úkolů. Odhady musí být prováděny s ohledem na pracnost daného úkolu a také na dostupnost zdroje přiřazeného k danému úkolu.

Pro odhadování se používají následující metody a nástroje: jednobodový odhad, metoda PERTH, expertní odhad, normativní odhad, porovnávací metoda a další. Některé z těchto metod budou popsány v další kapitole.

2.2.3 Organizování

Ve fázi organizování se především přiřazují jednotliví členové pracovního týmu k úkolům ze seznamu činností. Společne s úkoly se daným členům přiřazuje i zodpovědnost za tyto činnosti. Dále jsou zde ještě stanoveny kontrolní nástroje.

Matice zodpovědnosti (RACI matice) [6]

Je nástroj pro přiřazení a znázornění odpovědností osob k jednotlivým úlohám projektu.

Matice zodpovědnosti je tabulka, kde na levé straně (neboli v prvním sloupci) jsou uvedeny jednotlivé úlohy, ke kterým je nutné přiřadit zodpovědnosti. V horní části (nebo také v prvním řádku) tabulky se potom nacházejí jednotliví členové týmu. Do jednotlivých buněk tabulky se vyplňují odpovědnosti členů pomocí písmen ze zkratky RACI, jenž mají následující význam:

- **R** – responsible – přiřadí se členovi realizující daný úkol.
- **A** – accountable – představuje managerskou zopovědnost za výsledek.
- **C** – consulted – je přiřazeno členovi, který může poskytnout konzultaci k danému úkolu.
- **I** – informed – označení člena, který má být informován.

2.2.4 Realizace a kontrola

Po naplánování projektu je nutné začít realizovat projekt dle plánu. To obnáší provádění kontrol, zda se jednotlivé body v plánu plní tak, jak mají a v případě odchýlení od očekávaného plánu je nutné stanovit protiopatření. V této fázi se také kontroluje kvalita výstupu dle předem stanovených metrik.

Metoda procentuálního plnění

Procentuální ohodnocení stavu dané činnosti, balíku činností, případně stav prací člena týmu. Při určování stavu prací má pracovník či člen týmu dvě možnosti, jak stanovit rozpracovanost daného úkolu a to:

- pracovník provede odhad na základě vlastní zkušenosti nebo
- lze provést výpočet na základě vzorce (2.1), kde P_s představuje počet skutečně odpracovaných hodin na dném úkolu a P_d je odhad počtu hodin zbývajících pro dokončení úkolu.

$$PP = \frac{P_s}{P_s + P_d} \tag{2.1}$$

2.2.5 Ukončení

Poslední fází životního cyklu projektu je ukončení projektu. V této fázi se výsledek předává zákazníkovi a provádějí se analýzy ukončeného projektu, které mohou být zdrojem pro zlepšení projektů následujících. Fáze ukončení také obstarává případné udržování projektu pro zákazníka.

2.3 Řízení času

Již z definice projektu uvedené v předchozí kapitole vyplývá, že čas je jeden z klíčových parametrů projektu a proto na něm silně závisí úspěch celého projektu a je proto nutné ho vhodně řídit.

2.3.1 Časové plánování projektu

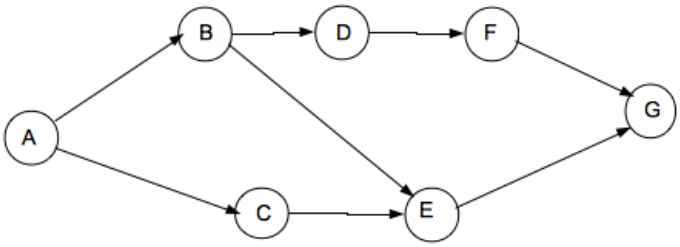
Je jednou z klíčových součástí plánování celého projektu. Probíhá současně s plánováním zdrojů.

Pro plánování se využívají naásledující metody a struktury:

Síťový graf [5]

Síťový graf je grafické znázornění závislotí mezi jednotlivými činnostmi. Síťový graf může být:

- Uzlově definovaný viz. obrázek 2.1 – činnosti jsou reprezentovány uzly a hrany vyjadřují závislosti.
- Hranově definovaný – činnosti jsou reprezentovány hranami a uzly vyjadřují stavy, kterých je dosaženo vstupními a výstupními činnostmi.



Obrázek 2.1: Ukázka uzlově definovaného síťového grafu. [5]

Při konstrukci grafu je nutné dodržovat následující pravidla:

- Konstrukce grafu probíhá zleva doprava, nelze vytvářet cykly.
- Graf má jeden začátek a jeden konec.

Technika PERT [5]

Technika PERT slouží k výpočtu nejpravděpodobnější doby trvání jednotlivých činností. Výpočet doby trvání je dle vzorce (2.2) založena na třech odhadech délky trvání a to na optimistickém t_o, pesimistickém t_p a normálním t_n.

$$T = \frac{t_o + 4.t_n + t_p}{6} \tag{2.2}$$

Metoda kritické cesty [2, 5]

Metoda pro časový rozbor projektu za využití síťového grafu hledá kritickou cestu. Kritická cesta je cesta grafem, která má nejdelší možnou dobu trvání a je složena z kritických činností. Pokud by se jakákoliv činnost na kritické cestě zpozdila, znamenalo by to celkové zpoždění projektu.

Při sestavování síťového diagramu je nutné stanovit pro každou činnost dobu jejího trvání. Výpočet kritické cesty se provádí pomocí výpočtu následujících časových položek:

- nejdříve možný začátek činnosti,
- nejdříve možný konec činnosti,
- doba trvání,
- nejpozději přípustný začátek činnosti,
- nejpozději přípustný konec činnosti,
- časová rezerva.

Nejdříve možné časové začátky a konce činností jsou odvozovány od začátku projektu a předcházejících činností.

Nejpozději přípustné časové konce a začátky jsou zpětně určovány pomocí následujících činností a času ukončení projektu.

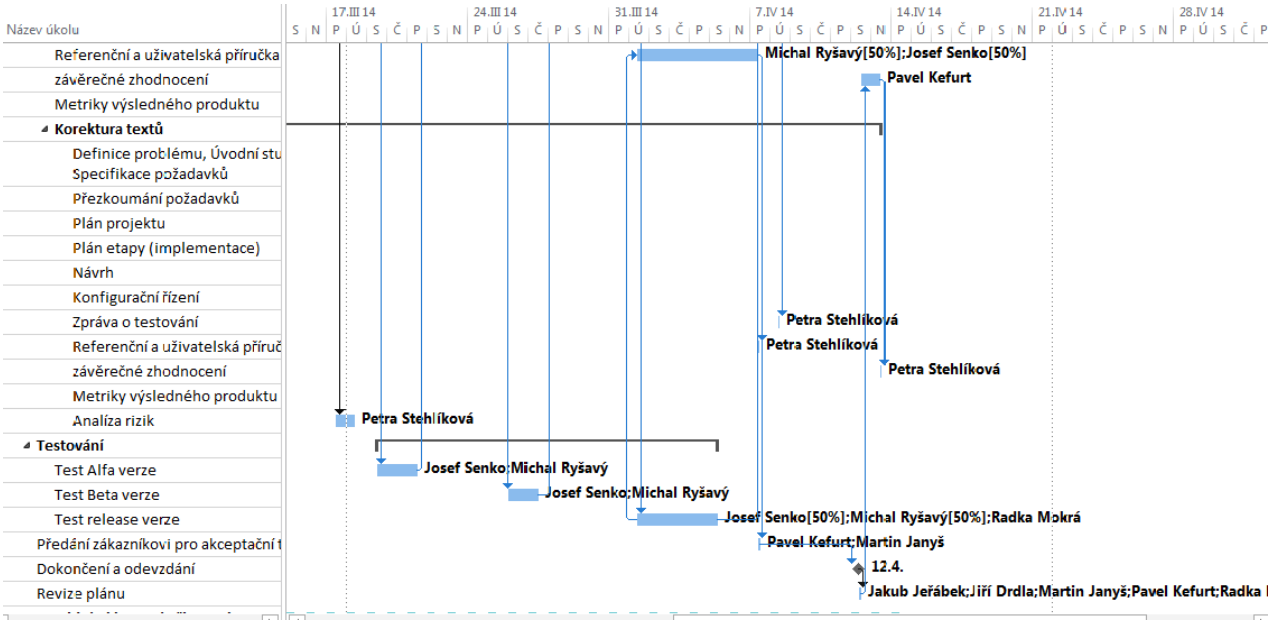
Ganttovy grafy

Pro lepší přehlednost se místo síťových grafů začaly využívat Ganttovy grafy. Ganttův graf je úsečkový diagram graficky znázorňující časový plán projektu a logické vazby mezi jednotlivými činnostmi v plánu.

Ganttův graf se skládá z časové osy, která je většinou uvedena v horní části grafu a ze seznamu úkolů uvedeným v levé části grafu. Uvnitř grafu jsou pomocí úseček zaznamenány doby začátků, konců a doba trvání jednotlivých činností. Ukázka Ganttova grafu je na obrázku 2.2.

2.4 Dostupné nástroje

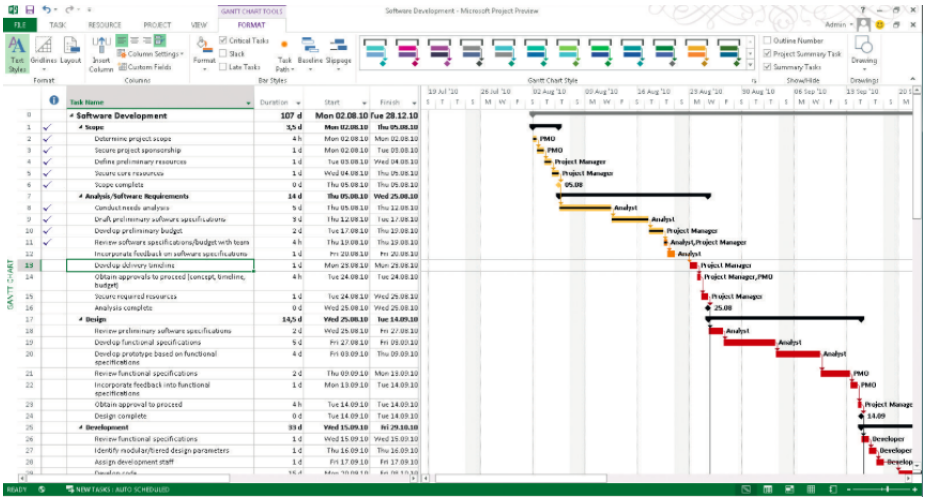
Nástrojů pro řízení projektu je dnes na trhu nepřeberné množství. Zde budou některé z nich posány.



Obrázek 2.2: Ganttův graf

Microsoft Project

Produkt od společnosti Microsoft. Poskytuje velké množství nástrojů a technik pro řízení projektů. Obsahuje většinu zmíněných nástrojů uvedených výše od hierachické struktury prací až po síťové a Ganttovy grafy, ale i nástroje pro výpočty nákladů a správu zdrojů. Je to jeden z nejpoužívanějších nástrojů pro řízení projektů, jeho cena se pohybuje okolo 15 tisíc za standardní verzi a 30 tisíc za verzi profesionální.

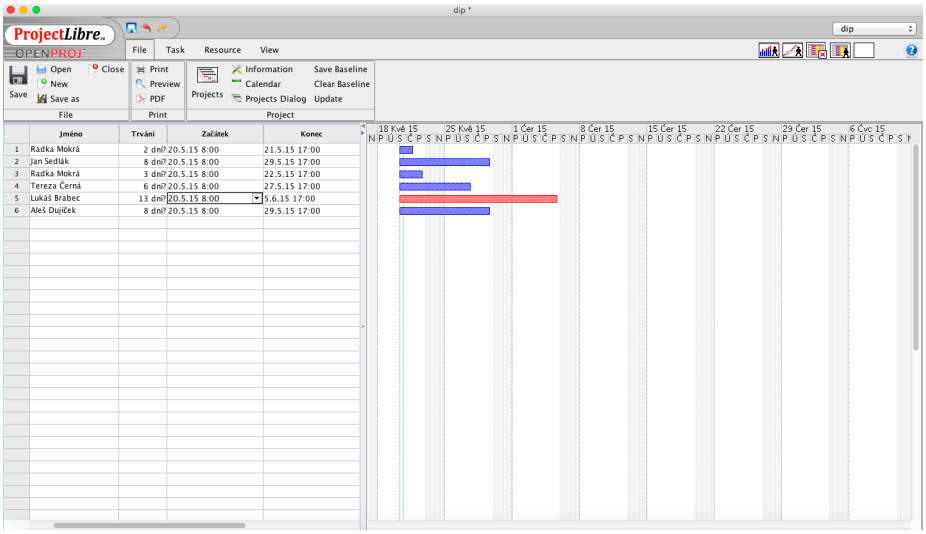


Obrázek 2.3: Ukázka aplikace Microsoft Project

Project Libre

Open source varianta k Microsoft project. Nástroj, určený k plánování projektů, obsahuje stejně jako komerční produkty metody a techniky jako techniku hierarchického rozkladu

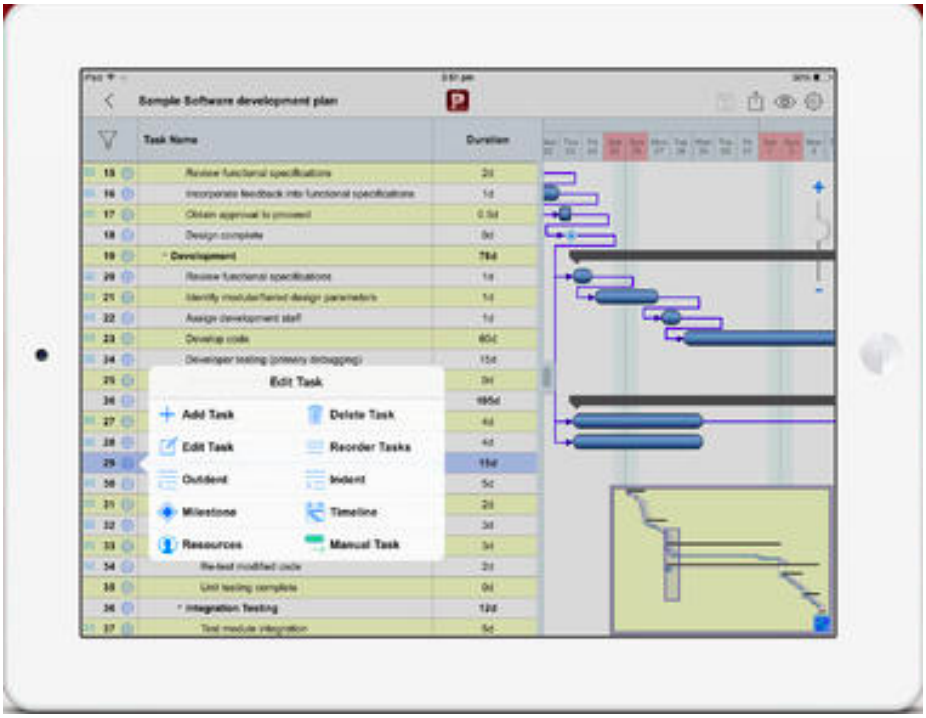
prací, Ganttovy grafy, síťové diagramy a další. Jeho výhdou je jeho dostupnost i na operační systémy jako Linux nebo OS X.



Obrázek 2.4: Ukázka aplikace Project Libre

Project Planning Pro – Project, Task & Resource Management

Aplikace pro řízení projektů pro zařízení s operačním systémem iOS. Aplikace určená k plánování projektů, která umožňuje výstupy exportovat do formátu Microsoft Project. Aplikace podporuje vytváření seznamů činností podle metody hierarchického rozkladu prací či plánování pomocí Ganttových grafů.



Obrázek 2.5: Ukázka aplikace Project Planning Pro

Kapitola 3

Plánování úloh reálného času

V této kapitole bude popsáno základní členění a plánování úloh a základní mechanismy. Především se zde zaměřím na plánování pomocí on-line prioritních plánovacích mechanismů. Tato kapitola čerpá především z předmětu ROS [3].

3.1 Popis úlohy a dělení plánovacích mechanismů [1]

Úloha reálného času je základním prvkem pro plánování. Úlohy reálného času mohou být periodické nebo aperiodické a mohou mít lehká (soft) nebo těžká (hard) časová omezení.

Popis úlohy

Úloha reálného času je charakterizována primárními a dynamickými parametry. Primární parametry úlohy:

- r, doba příchodu úlohy (r_0 je první příchod úlohy a $r_k = r_0 + k.T$).
- C, nejhorší doba běhu úlohy při plně dostupném CPU.
- D (*deadline*), relativní časová mez úlohy.
- T, perioda úlohy (pouze pro periodické úlohy).
- Jestliže má úloha hard časové omezení potom $d = r + D$ je její absolutní časová mez úlohy. V případě překročení této meze může dojít k selhání systému.

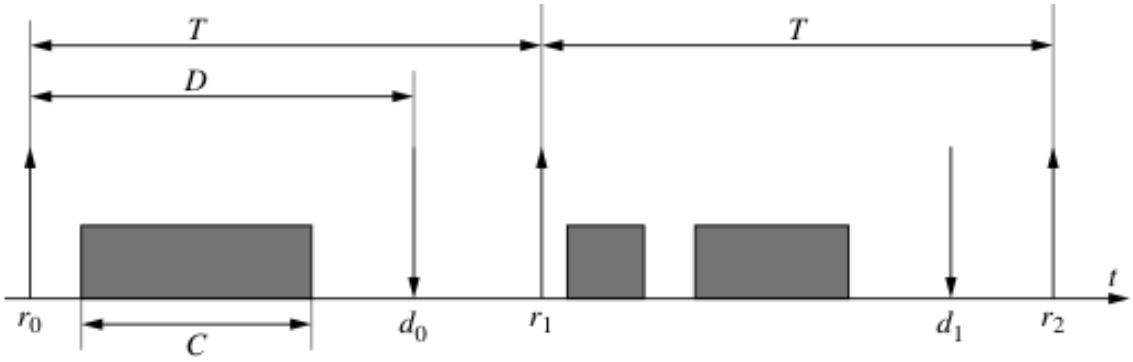
Paremetr T se vyskytuje pouze u periodických úloh. Úlohy jsou tedy popsány následovně:

- periodická úloha – popsána pomocí $\tau(r, C, D, T)$
- aperiodická úloha – popsána pomocí $\tau(r, C, D)$

Úloha je dobře formulovaná, jestliže platí následující: $0 < C \leq D \leq T$. Na obrázku 3.1 jsou znázorněny jednotlivé parametry úlohy.

Jak již bylo zmíněno výše, úloha reálného času má kromě primárních i následující dynamické parametry:

- s začátek provádění úlohy.



Obrázek 3.1: Model úlohy [1]

- e konec provádění úlohy.
- $D(t) = d - t$ zbytková relativní časová mez.
- $C(t)$ zbývající čas do konce úlohy.
- $L = D - C$ nejdelší možné čekání na zahájení úlohy.
- $L(t) = D(t) - C(t)$ nejdelší možné čekání od času t při plně dostupném CPU.
- $TR = e - r$ čas odezvy úlohy. Musí platit $C \leq TR \leq D$, aby nedošlo k časové chybě.
- $CH(t) = C(t)/D(t)$ zbytkové zatížení

Mimo výše uvedené parametry lze úlohám přiřadit ještě následující vlastnosti:

- **Preemptivnost nebo nepreemptivnost** – nepreemptivní úloha je taková úloha, která když obdrží procesorový čas a dostane se k výpočtu, tak už nemůže být přerušena. Naopak preemptivní úloha je úloha, která může být přerušena v průběhu výpočtu v zájmu úlohy s například vyšší prioritou.
- **Závislost úlohy** – jednotlivé úlohy mohou mít mezi sebou závislosti a to buď statické, nebo dynamické. Statická závislost mezi úlohami je známá již před během úloh, narozdíl od dynamických závislostí, které vznikají až v průběhu běhu úloh.
- **Důležitost (kritičnost)** – umožňuje vybrat úlohu před jinou úlohou, například v případě blížícího se uplynutí časové meze.
- **Externí priorita** – priorita, kterou může přidělit návrhář.

Dělení plánovacích mechanismů

Plánovací algoritmy je možné rozdělit následovně:

On-line nebo Off-line plánování

- **On-line plánování** probíhá až za běhu úloh a proto je schopné reagovat na úlohy, které do systému přichází průběžně. Nevýhodou tohoto typu plánování implementační náročnost.
- **Off-line plánování** se provádí ještě před spuštěním úloh. Jeho výhodou je snadná implementace a lze jej optimalizovat. Nevýhodou je nemožnost změn prametrů úloh.

Preemptivní nebo nepreemptivní plánování

- **Preemptivní plánování** umožňuje plánovači, aby odebral aktuální úloze CPU a přiřadil ho úloze s aktuálně vyšší prioritou nebo vyšší důležitostí. Preemptivní plánování je možné pouze u preemptivníh úloh.
- **Nepreemptivní plánování** narozdíl od preemptivního plánování neumožňuje plánovači přerušení běhu úlohy.

Nejlepší snaha a netolerování chyb v časování

- **Plánování s nejlepší snahou** se uplatňuje při plánování úloh s lehkými (soft) časovými omezeními, které umožní plně využít CPU a současně tolerovat jistou míru chyb v časování.
- Naopak pří plánování úloh s těžkými (hard) časovými omezeními je tolerance chyb v časování nepřípustná.

Centralizované nebo distribuované plánování

- **Centralizované plánování** je centralizované pokud probíhá na centralizované architektuře.
- **Distribuované plánování** probíhá na více uzlech a každý uzel vytváří lokální plán dle globálního plánu.

Naplánovaný plán je přípustný, jestliže jsou v něm dodrženy časové meze u všech úloh.

Plánovací mechanismy lze ještě dělit na základě přidělování priority úlohám na úlohy se statickou prioritou, kterým se priorita v průběhu nemění, nebo úlohy s dynamickou prioritou, jenž se priorita v průběhu mění.

3.2 On-line prioritní plánovací mechanismy

3.2.1 Plánování množin nezávislých periodických úloh se statickou prioritou

Algoritmy RM a DM patří k základním algoritmům od kterých se odvíjí další algoritmy a mechanismy.

Priorita přidělena na základě periody – Rate monotonic (RM) [9, 8]

RM plánovácí algoritmus je jedním z nejvíce využívaných. Je to jednoprocesorový preemptivní algoritmus využívající statické přidělování priorit. V RM plánování jsou všechny úlohy periodické a priorita jim je přiřazena na základě jejich periody. Podmínky pro RM plánování:

1. Periodické úlohy mají konstantní doby výpočtu a jsou připraveny pro výpočet na začátku každé periody.
2. $D = T$

3. Úlohy jsou na sobě nezávislé, neexistuje závislost mezi dvěma úlohami a úlohy se navzájem neblokují.

4. Přepínání kontextu a swapování má nulovou režii.

Priorita přidělena na základě relativní časové meze – Deadline monotonic (DM) [8]

DM algoritmus je velice podobný předchozímu RM algoritmu s tím rozdílem, že zatímco u RM byla úloha s největší prioritou ta s největší periodou, u DM je nejprioritnější úloha ta, která má nejnižší časovou mez (D). Podmínky DM plánování jsou stejné jako u RM plánování s tím rozdílem, že se v bodě tři použije podmíka $D < T$.

3.2.2 Plánování množin nezávislých periodických úloh s dynamickými prioritami

Zatímco u algoritmů RM a DM se priorita přidělovala staticky, u algoritmů EDF a LLF se priorita přiděluje dynamicky za běhu.

Úloha s nejkratší relativní mezí první – Earlines deadline first (EDF) [8]

Pro EDF platí stejné podmínky, jako pro algoritmus DM. EDF přiřazuje priority dynamicky za běhu na základě relativní časové meze, nejvyšší prioritu má ta úloha, která má neblíž relativní časovou mez.

Úloha s nejmenší dobou odezvy první – Least laxity first (LLF) [8]

Stejně jako algoritmus EDF je algoritmus LLF řízený dynamickým přidělováním priorit. V algoritmu LLF se největší priorita přiřadí úloze, která má v aktuálním čase nejmenší dobu odezvy (L). Jestliže mají dvě úlohy stejné L, budou se navzájem preemptivně přepínat mezi sebou.

3.2.3 Plánování množin nezávislých hybridních úloh

Předchozí plánovací algoritmy plánovaly pouze množiny periodických úloh. Proto zde budou uvedeny způsoby, jak lze plánovat i úlohy aperiodické.

Server na pozadí - The background server

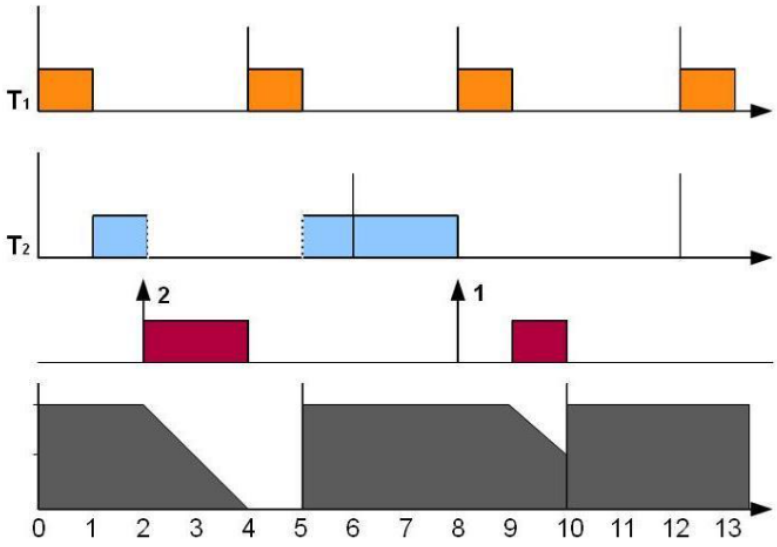
Je určen pro lehká (soft) časová omezení. Aperiodické úlohy plánuje v době, kdy v systému není žádná periodická úloha.

Vyzývací server – The polling server (PS)

U vyzývacího serveru jsou aperiodické úlohy obsluhováný pomocí periodické úlohy PS. Příchozí aperiodické úlohy jsou obsluhovány na začátku každé periody PS. Jestliže se na začátku periody nevyskytuje žádná aperiodická úloha, PS se pozastaví a umožní využít svoji kapacitu pro periodické úlohy. Úlohy, co přisly po začátku periody, jsou obsouženy při následující periodě.

Odkládací server – Deferrable server (DS) [8, 3]

DS je algoritmus zaměřený na rychlejší obsluhu aperiodických úloh. DS je speciální periodická úloha s kapacitou a periodou, která obsluhuje aperiodické úlohy, dokud ji stačí kapacita. Pokud na začátku periody nepříjde úloha, DS kapacitu nevrací, ale uchovává si ji pro případ příchodu aperiodické události. Pokud na konci periody zbývá nějaká kapacita, tak tato kapacita propadá a je obnovena v další periodě. Pokud chceme dosáhnout včasné odezvy aperiodických úloh, je vhodné dát DS nejvyšší prioritu. Na obrázku 3.2 je ukázka plánování pomocí DS.



Obrázek 3.2: DS plánování s úlohami $\tau_1(1,4)$, $\tau_2(2,6)$, $Server(2,5)$ kde $\tau(C_i,T_i)$ [8]

Server se změnou priority – The priority exchange server (PE) [8]

PE je založen na stejném principu, jako algoritmus DS - také má pro obsluhu aperiodických úloh periodickou událost s kapacitou, která se obnovuje s další periodou. Rozdíl mezi DS a PE je v tom, že při DS má po celou dobu nejvyšší prioritu, zatímco u PE se priorita mění v závislosti na existenci aperiodických úloh. Pokud jsou v systému aperiodické úlohy, PE má vysokou prioritu, jinak si prioritu včetně kapacity mění s méně prioritními periodickými úlohami.

Společné EDF plánování [3]

Toto plánování nevyužívá k plánováni periodickou úlohu jako předchozí řešení. U tohoto přístupu se stanoví u každé aperiodické úlohy fiktivní časová mez úlohy (fictive deadline) tak, aby měla co nejkratší odezvu. Fiktivní časová mez fd se nastaví na nejbližší čas t tak, že doba potřebná pro vykonání úlohy je stejná jako doba volnosti CPU v čase t.

3.2.4 Plánování množin úloh závislých na pořadí jejich provádění [3]

Úlohy závislé na pořadí jsou takové úlohy, které vyžadují, aby bylo respektováno pořadí úloh. Například v případě, že pro svůj běh vyžadují výsledky předchozí úlohy. To, že úloha

a musí předcházet úlohu b, lze zapsat následovně: $\tau_a \rightarrow \tau_b$, nebo graficky pomocí precedenčního grafu, který se podobá uzlově definovanému síťovému grafu z předchozí kapitoly na obrázku 2.1.

Plánování úloh závislých na pořadí pomocí algoritmu RM/DM

Plánování závislých úloh pomocí principu RM algoritmu, který byl popsán výše, se provádí pomocí úpravy hodnot jednotlivých úloh tak, aby byla respektována daná omezení.

Pokud tedy chceme, aby $\tau_a \rightarrow \tau_b$ potom musí platit:

- $r_b^* \geq max(r_b, r_a^*)$, kde r_a^*, r_b^* jsou upravované parametry r_a, r_b.
- $P_a \geq P_b$ je ve shodě s RM, kde P_a a P_b jsou priority úloh.

U plánování pomocí DM algoritmu se navíc musí ještě změnit parametr D následovně:

- $D_b^* \geq max(D_b, D_a^*)$, kde D_a^*, D_b^* jsou upravované parametry D_a, D_b.

Plánování úloh závislých na pořadí pomocí algoritmu EDF

V případě plánování závislých úloh na pořadí pomocí EDF je nutné upravit u jednotlivých závislých úloh pametr d a r. Při modifikaci těchto parametrů se začíná od těch úloh, které nemají žádné předchůdce nebo následovníky.

3.2.5 Plánování úloh závislých na sdílení kritických prostředků [3]

V případě plánování úloh závislých na sdílení prostředků je nutné implementovat mechanismus přístupu k tomuto prostředku. Takový přistup do kritické sekce daného prostředku lze zajistit například pomoci semaforů nebo monitorů.

Při špatné správě prostředků mohou nastat následující problémy, kterým je třeba se vyhnout:

- Blokování
- Inverze priorit
- Uváznutí

Protokoly přístupu k prostředkům [8]

Protokoly přístupu k prostředkům zamezují výše uvedeným problémům.

- **Protokol dědění priorit** – Priority inheritance protokol (PIP) Umožňuje dynamicky měnit priority úlohám, které jsou blokovány úlohami s vyšší prioritou. Priorita se nastaví na prioritu úlohy, která ji blokuje.
- **Protokol stropování priorit** – Priority ceiling protocol (PCP) Tento protokol zabraňuje uváznutí tím, že pro každý semafor u sdíleného prostředku existuje stropová priorita. Ta je získána od úlohy s nejvyšší prioritou čekající na vstup do kritické sekce. Úloha potom může vstoupit do kritické sekce pouze tehdy pokud má vyšší prioritu než je priorita stropová.

3.2.6 Plánování úloh při přetížení [3, 9]

U plánování úloh při přetížení je nutné rozhodnout, kterým úlohám se dá přednost před ostaními. K tomu se využívají algoritmy s nejlepší snahou. V případě, že nedojde k přetížení, jsou úlohy plánovány pomocí mechanismu, který je ekvivalentní algoritmu EDF.

V obou algoritmech jak LBESA tak i DASA se využívá hodnota hustoty užitku VU, která se vypočítá podle vztahu (3.1).

$$VU_i = \frac{V_i}{D_i(t)} \tag{3.1}$$

Uzamykající plánovací algoritmus nejlepší snahy – Lockes Best Effort Scheduling Algorithm (LBESA) [9]

Algoritmus LBESA seřadí úlohy vzestupně podle parametru D. Následně vkládá postupně seřazené úlohy do předběžného plánu a pokud vložením úlohy vznikne nenaplánovatelný plán, vybírá postupně úlohy z nejnižší hodnotou VU tak dlouho, dokud není množina úloh plánovatelná.

1. Vytvořit prázdný plán
2. Nechť t je čas výskytu plánovací události
3. Seraď fáze úloh dle jejich časových mezí
4. Pro každou fázi proveď (v pořadí rostoucích časových mezích)
a) Umísti fázi do plánu počínajícího v čase t
b) Dokud je plán nepřípustný a neprázdný tak vyřaď nejméně důležitou fázi plánu
5. K naplánování vyber úlohu s nejbližší mezí

Algorithm 1: Algoritmus LBESA [3]

Plánovací algoritmus zavislý na aktivitě – Depend Activity Scheduling Algorithm (DASA) [9]

Algoritmus DASA je velice podbný algoritmu LBESA s tím rozdílem, že na počátku seřadí posloupnost sestupně podle parametru VU. Následně je vkládá do přeběžného plánu a zjišťuje, jestli je plánovatelný. V případě, že plán není plánovatený, odebírá, stejne jako LBESA, úlohy z plánu s nejmenší hodnotou VU tak dlouho, dokud není množina úloh plánovatelná.

1. Vytvořit prázdný plán
2. Nechť t je čas výskytu plánovací události
3. Seraď fáze úloh dle jejich hustot důležitosti
4. Pro každou fázi proveď (v pořadí klesajících jejich hustot důležitosti)
a) Umísti fázi do plánu počínajícího v čase t
b) Test na přípustnost plánu v čase t. Pokud není plán přípustný, potom odeber fázi z plánu.
5. K naplánování vyber úlohu s nejbližší mezí

Algorithm 2: Algoritmus DASA [3]

Algoritmus DASA se obecně považuje za lepší, než algoritmus LBESA [9].

Kapitola 4

Návrh aplikace

V této kapitole bude navržena aplikace pro plánování projektů a osobního času s využitím principů z plánování úloh reálného času.

4.1 Úloha v projektu vs. úloha reálného času

V úvodu srovnám, jaký je rozdíl mezi úlohou/činností při plánování projektů a při plánování v systémech reálného času.

Jak již bylo uvedeno v druhé kapitole, činnost v rámci plánování projektů je dána následujícími parametry:

- nejdříve možný začátek činnosti,
- nejdříve možný konec činnosti,
- doba trvání,
- nejpozději přípustný začátek činnosti,
- nejpozději přípustný konec činnosti,
- časová rezerva.

Ze třetí kapitoly potom parametry pro úlohu reálného času:

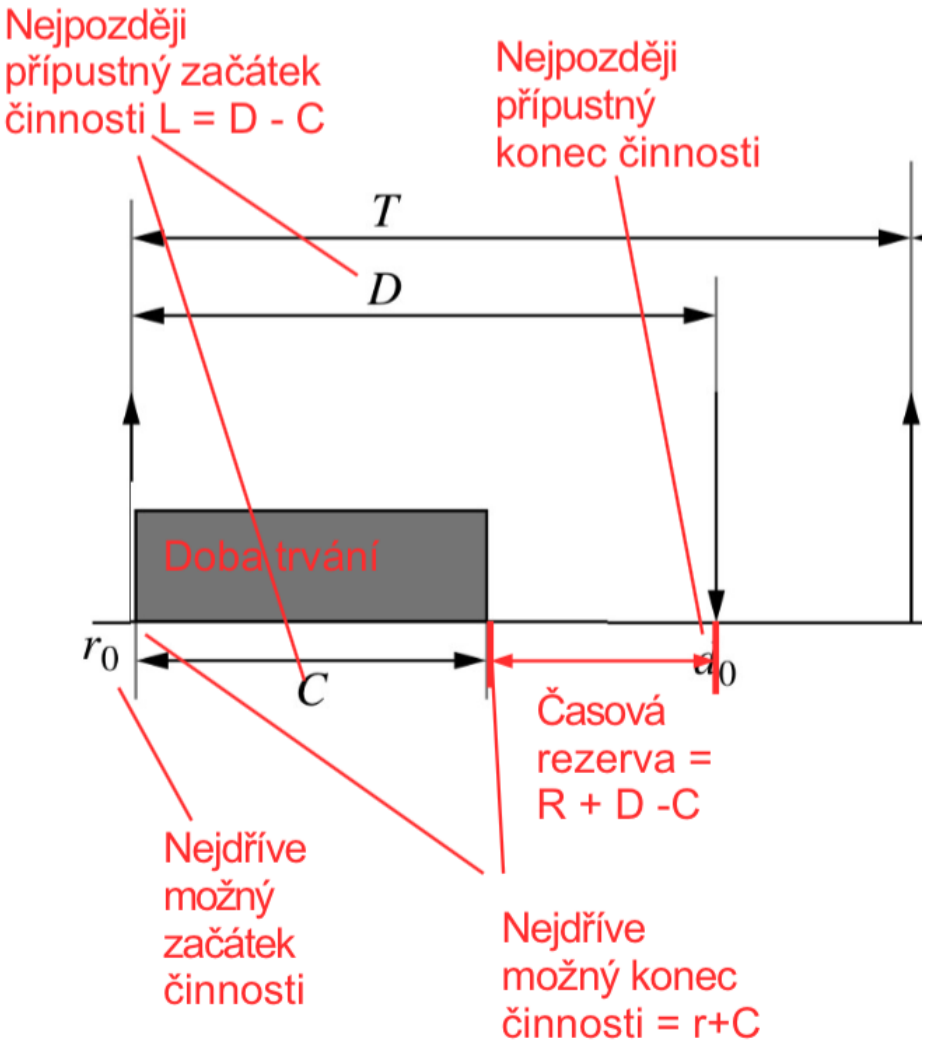
- r, doba příchodu úlohy (r_0 je první příchod úlohy a $r_k = r_0 + k.T$).
- C, nejhorší doba běhu úlohy při plně dostupném CPU.
- D, relativní časová mez úlohy.
- T, perioda úlohy (pouze pro periodické úlohy).
- Jestliže má úloha hard časové omezení potom $d = r + D$ je její absolutní časová mez úlohy. V případě překročení této meze může dojít k selhání systému.

Po srovnání lze mezi úlohami spatřit následující podobnosti:

Úloha reálného času	**Úloha/činnost v plánování projektů**
r	nejdříve možný začátek činnosti
C	doba trvání
$D(d)$	nejpozději přípustný konec činnosti

Zbývající parametry bude nutné převzít z dynamických parametrů úloh reálného času uvedených v předchozí kapitole, nebo je vypočítat pomocí dostupných parametrů následovně:

Úloha reálného času	**Úloha/činnost v plánování projektů**
$r + C$	nejdříve možný konec činnosti
$L = D - C$	nejpozději přípustný začátek činnosti
$r + D - c$	časová rezerva



Obrázek 4.1: Srovnání úlohy reálného času a úlohy v projektu.

4.1.1 Vhodné mechanismy pro řízení projektů a času

Plánování projektů lze dle úloh reálného času charakterizovat následovně:

- Vyskytují se zde periodické i aperiodické úlohy, ovšem aperiodické převažují.
- Úlohy jsou preemptivní.
- Využívá se on-line plánování.

- Jednotlivé úlohy na sobě mohou být závislé.
- V případě plánování projektů jsou přiřazeny zdroje.

Této charakteristice odpovídají následující plánovací mechanismy:

- Algoritmus pro přetížení systému DASA, upravený stejně, jako EDF pro precedenční závislosti.
- Přidělování zdrojů pomocí protokolu PCP.

Úloha v algoritmu DASA potřebuje následujíci parametry k plánování úloh:

- C – doba běhu.
- d – nejpozdější možné ukončení úlohy.
- V – odměna z provedení úlohy. Tato hodnota se bude v aplikaci odvozovat od zadané důležitosti úlohy a bude na ní lineárně závislá.

Aplikace bude využívat výše uvedených principů k plánování projektů nebo času. U plánování projektů se při plánováni bude nutné zaměřit i na použité zdroje a přístup k nim, u plánování času se toto nebude vyžadovat.

4.2 Návrh databáze

Jelikož se bude jednat o webovou aplikaci, je nutné řídit autentizaci uživatelů. Proto bude nutné ukládat informace o uživatelském účtu, k tomu bude sloužit třída **Uživatel**. Dále je nutné mít třídu **Projekt**, kvůli seskupování úloh a jejich plánovaní. Samostatnou třídu bude také mít již zmiňovaná **Úloha**. Poslední třídou potom bude třída **Zdroj**.

Celkově tedy navržená databáze bude mít následující čtyři třídy:

- Uživatel
- Projek
- Úloha
- Zdroj

Na obrázku 4.2 je návrh databáze společne s atributy a vztahy mezi jednotlivými třídami. Takto jsou popsány jednotlivé atributy, které třídy obsahují:

- **Uživatel** – třída uživatel slouží především pro evidenci uživatelů a proto má pouze jeden atribut. Tímto atributem je:
  - Jméno, které slouží k identifikaci uživatele.
- **Projekt** – třída projekt slouží pro sjednocení jednotlivých úloh. Třídá má následující atributy:
  - Název – pojmenování projektu,
  - Začátek – datum začátku projektu,
  - Konec – datum ukončení projektu.

- **Úloha** je potom nejdůležitější třídou a uchovává informace o jednotilivých úlohách v následujícící atributech:
  - Název – pojmenování dané úlohy,
  - Začátek – začátek úlohy,
  - Konec – nejpozdější termín dokončení úlohy,
  - Doba trvání – doba trvání úlohy,
  - Důležitost – neboli priorita, čím nižší hodnota tím je úloha více duležitá.
- **Zdroj** – je poslední třídou a slouží k určení dostupnosti případných zdrojů a má následující atributy:
  - Název – udává jméno nebo název konkrétního zdroje,
  - Dostupnost – informace o dostupnosti daného zdroje.

4.3 Diagram případů užití

Ve výsledné aplikaci se budou nacházet dvě role uživatelů a to Administrátor a Uživatel.

- **Administrátor** – má možnost spravovat systém přes administrátorské rozhraní a to mu umožňuje manipulaci s daty uživatele. Kromě toho může administrátor odebírat a přidávat nové uživatele.
- **Uživatel** – po přihlášní může přidávat, editovat odebírat projekty a do nich následně přidávat úlohy. Může také přidávat a odebírat jednotlivé zdroje. Po zadání všech úloh může být daný projekt uživatelem naplánován.

Na obrázku 4.3 je graficky znázorněn diagram případů užití.

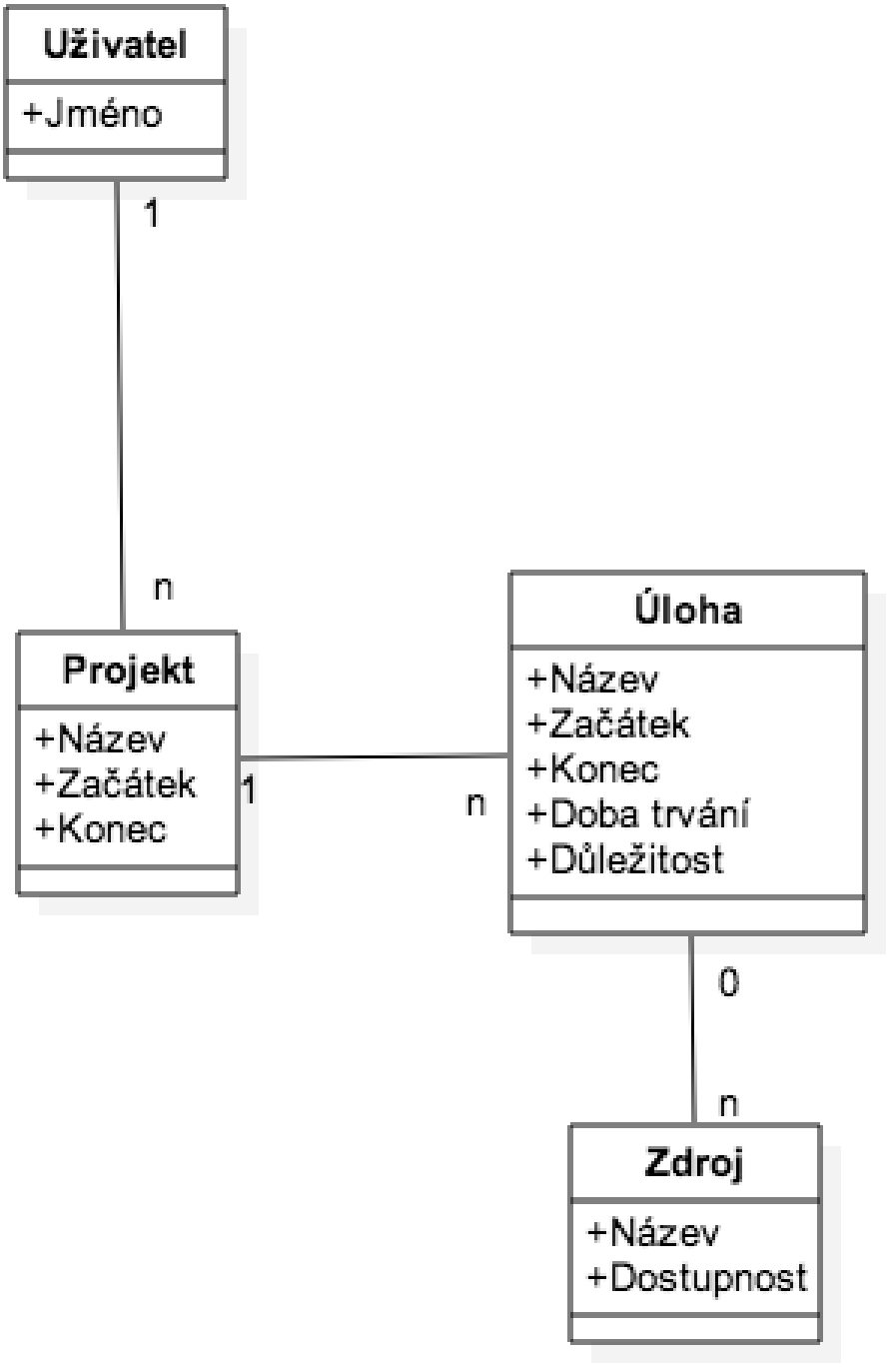
4.4 Návrh uživatelského rozhraní aplikace

Výsledná aplikace bude vyvíjena jako webová aplikace pomocí webového frameworku (`aplikačního rámce`) Django s pomocí webového frameworku pro grafické prvky Bootstrap a knihovou pro dynamiku a vizualizaci dat D3.js. Tyto technologie budou blíže popsáný v následující kapitole.

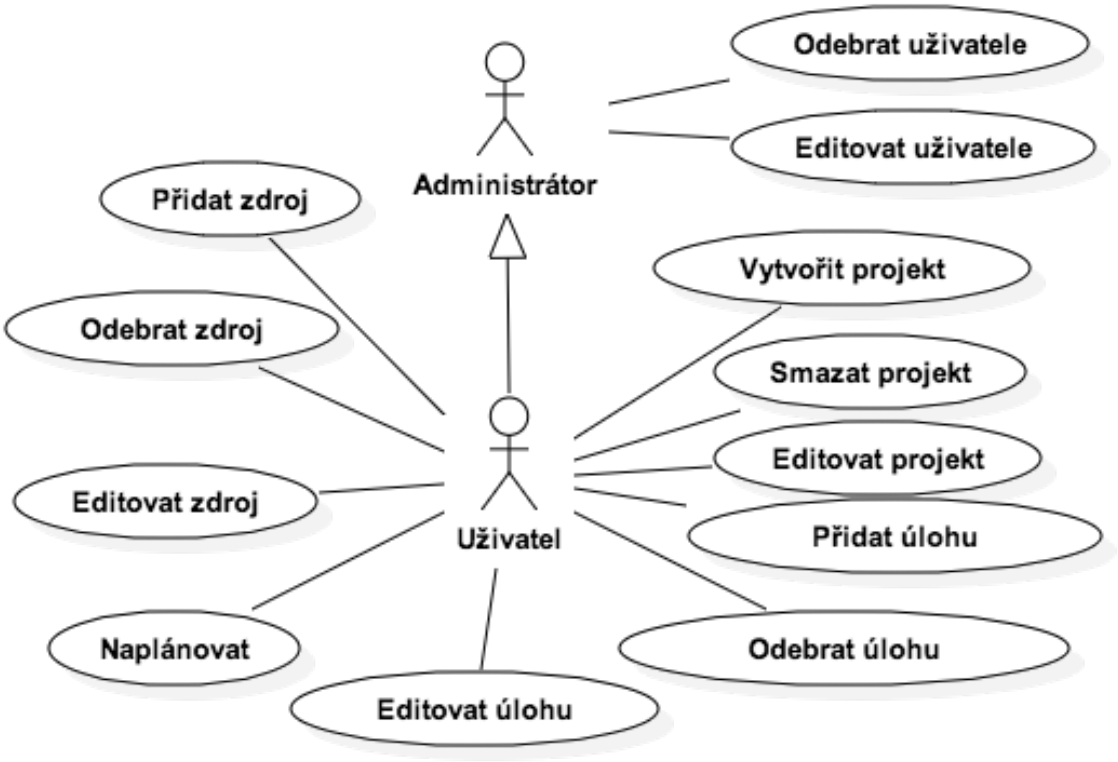
Po přihlášení bude mít uživatel možnost vybrat již stávající projekt nebo vytvořit projekt nový. U vytváření projektu bude nutné zadat jak název projektu, tak jeho začátek a konec.

Po otevření příslušného projektu se uživateli zobrazí seznam úloh a seznam zdrojů, ktery může editovat a rovněž do něj přidávat další úlohy nebo zdroje.

V případě, že jsou zadány všechny úlohy, je možné projekt naplánovat pomocí záložky „ganttův graf“.

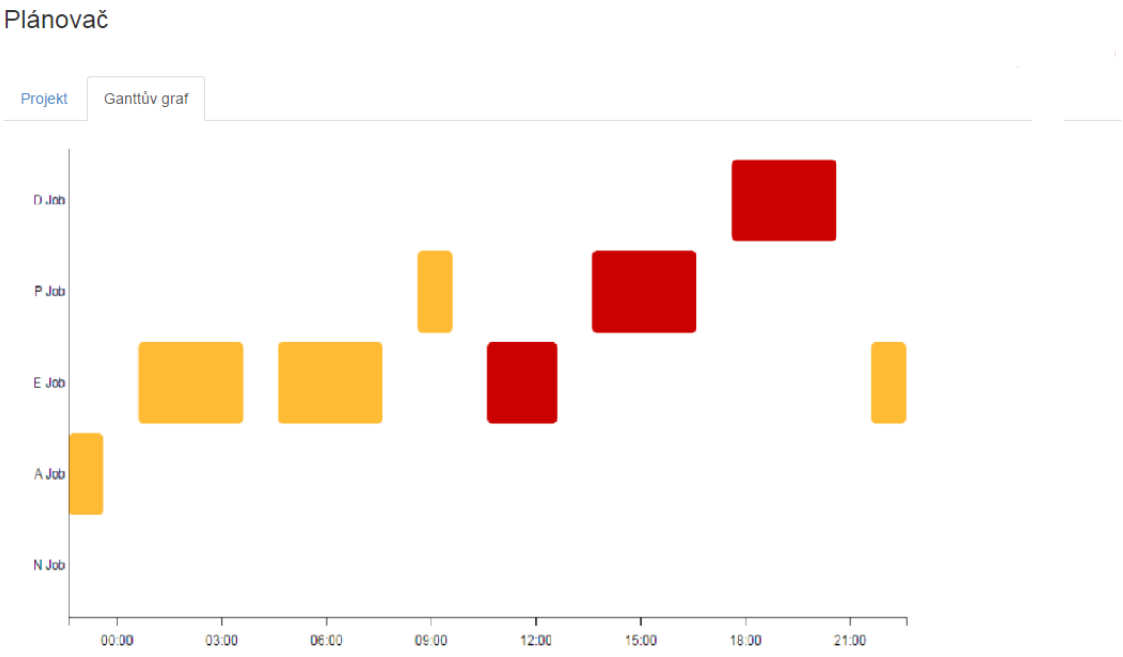


Obrázek 4.2: Návrh databáze aplikace



Obrázek 4.3: Ukázka diagramy případů užití

Obrázek 4.4: Návrh vzhledu aplikace



Obrázek 4.5: Návrh vzhledu aplikace

Kapitola 5

Implementační nástroje a technologie použité pro implementaci

V této kapitole budou popsány technologie a nástroje, které jsou využity pro implementaci webové aplikace pro plánování projektů a času. Jedná se o framework (*aplikační rámec*) Django a Bootstrap a knihovnu pro vizualizaci dat D3.js.

5.1 MVC [15, 4, 13]

MVC, neboli model-view-controller (model-pohled-řadič), je návrhový vzor nebo také architektonický vzor. MVC slouží především k oddělení logiky aplikace od jejich vstupů a výstupů. Konkrétně (jak již název napovídá) se MVC aplikace skládá ze tří následujících komponent, které mají mezi sebou vazby (obrázek 5.1):

- Model (Model)
- View (Pohled)
- Controller (Řadič)

5.1.1 Model

Model je část aplikace, která se stará o popis dat dané aplikace. Model popisuje způsob uložení dat v databázi a poskytuje metody jak k těmto datům přistupovat a dále jak tyto data vkládat do databáze, editovat a mazat.

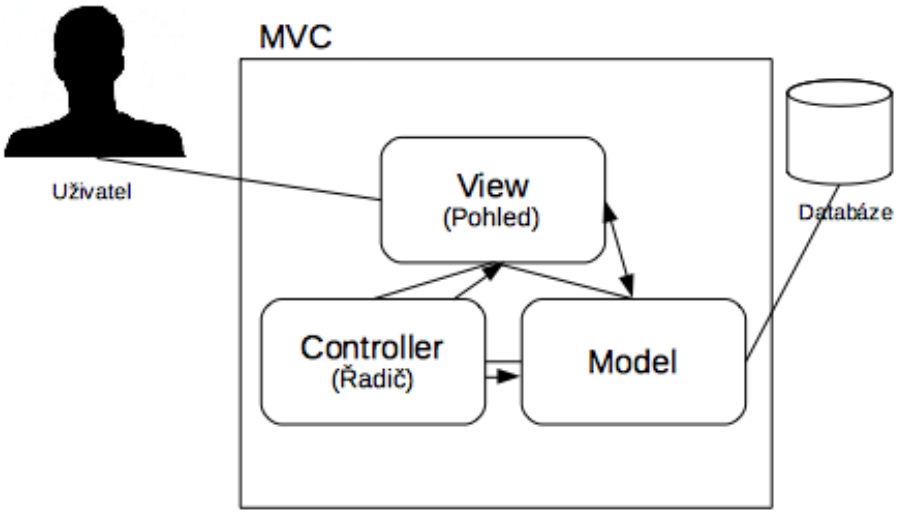
Model by měl být, stejně jako ostatní části architektury MVC, nezávislý na zbytku aplikace.

5.1.2 View

View, neboli pohled, je další částí MVC architektury. View je prezentační vrstva, která se stará o zobrazení výstupu uživateli. Zabývá se prezentací dat modelu různým uživatelům skrze tzv. *pohledy*.

Výsledná aplikace většinou obsahuje několik na sobě nezávislých view (pohledů).

Výstup view bývá nejčastěji prezentován, jako html stránka.



Obrázek 5.1: MVC, inspirováno z [4].

5.1.3 Controller

Poslední částí architektury MVC je controller neboli také řadič. Controller slouží jako propojovací prvek nebo spíše mezivrstva mezi modelem, view (pohledem) a uživatelem. Úlohou controlleru tedy je především přijímat požadavky od uživatele. Požadavky od uživatelů zpracuje a podle jejich významu je buď prostřednictvím pohledu zobrazí uživateli, nebo pomocí modelu například uloží získaná data do databáze.

Controller především zodpovídá za to, aby se k ostatním částem dostávala ta data, která mají. Například aby daná data uživateli zobrazil vybraný pohled nebo naopak, aby data zpracoval model a uložil je do databáze.

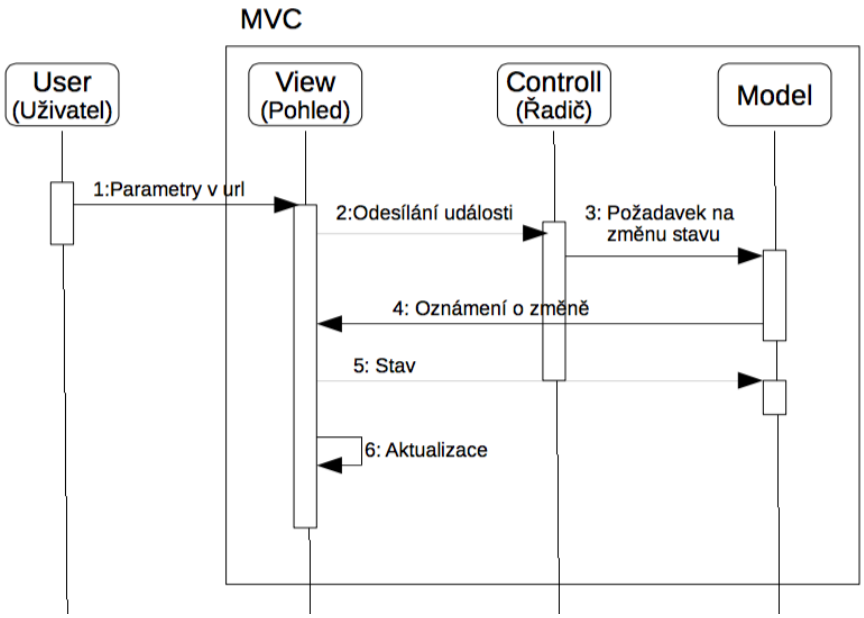
Jednotlivé části architektury MVC musí mezi sebou komunikovat. Příklad takovéto komunikace je na obrázku 5.2. Komunikace potom funguje následovně:

1. Uživatel zadá do prohlížeče adresu dané stránky společně s parametry pro pohled.
2. Pohled odešle údaje o akci provedené uživatelem.
3. Řadič zpracuje příchozí událost od pohledu a zasílá modelu žádost s příslušnou akcí.
4. Následně model zašle oznámení o změně stavu pohledu.
5. Pohled získá od modelu informace o aktuálním stavu.
6. Pohled tyto informace vykreslí pro uživatele.

5.1.4 Django a MVC

V této podkapitole bude popsáno, jak framework (*aplikační rámec*) Django využívá pro vývoj webové aplikace návrhový vzor MVC. Projekt v Djangu může být rozdělen například do těchto čtyř souborů:

- `models.py`



Obrázek 5.2: MVC komunikace, inspirováno z [15, 4].

- `views.py`
- `urls.py`
- `lastes_risk.html`

Soubory s příponou `.py` jsou zdrojové kódy v programovacím jazyce Python, zatímco soubor s příponou `.html` je HTML šablona. Následně budou jednotlivé části popsány spolu s jednoduchými příklady.

Prvním souborem je `models.py`. Tento soubor obsahuje popis tabulek databáze, které jsou reprezentovány třídou v Pythonu jenž se nazývá model. Pomocí této třídy je možné provádět změny v databázi jako jsou aktualizace, získání hodnot a mazání záznamů pomocí kódu v Pythonu místo příkazů v SQL.

V příkladu se uvedeno vytvoření jednoduchého modelu pro úlohu, které má dva atributy. Prvním atributem je název úlohy a druhým atributem je doba běhu úlohy.

```
# models.py (databáze)

from django.db import models

class Task(models.Model):
    name = models.CharField(max_length=50)
    value_time = models.IntegerField()
```

Dalším souborem je `views.py`. Tento soubor obsahuje logiku stránky - definice jednotlivých pohledů, jako je například funkce `biggest_time` v příkladu níže.

```
# views.py (logika)

from django.shortcuts import render
```

```
from models import Task

def biggest_time(request):
    task_list = Task.objects.order_by('-value_time')[:5]
    return render(request, 'biggest_time.html', {'task_list': task_list})
```

Třetím souborem je `urls.py`. V `urls.py` se přiřazují cesty URL funkcím z `views.py`. Například při zadání `http:www.planuj.czbiggest` se zavolá funkce `biggest_time`.

```
# urls.py (URL configurace)

from django.conf.urls.defaults import *
import views

urlpatterns = patterns('',
    (r'^biggest/$', views.biggest_time),
)
```

Posledním souborem je `biggest_time.html`. Jedná se o HTML šablonu (ne HTML soubor), která popisuje samotný design stránky.

```
# biggest_time.html (šablona)

<html><head><title>Tasks</title></head>
<body>
<h1>Tasks</h1>
<ul>
{% for task in task_list %}
<li>{{ task.name }}</li>
{% endfor %}
</ul>
</body></html>
```

5.2 Django[14]

Django je webový aplikační rámec pro vytváření webových aplikací. Je open source pod BSD licencí.

Django je souborem knihoven napsaných v Pythonu. Hlavní předností je jeho tzv. objektově–relační mapovač (ORM), který je zprostředkovatelem mezi datovým modelem a relační databází. Datové moduly jsou napsané v jazyce Python, z čehož vyplývá jeho výhoda, kterou je eliminace psaní příkazů databáze v jazycích, jako je SQL.

5.3 Bootstrap[11]

Bootstrap je HTML, CSS a Javascriptový aplikační rámec pro vývoj responsivních webových aplikací. Responsivní aplikace jsou takové aplikace, u kterých se jednotlivé grafické prvky dokáží přizpůsobit velikosti a rozlišení obrazovky.

Bootstrap poskytuje především grafické a typografické prvky.

5.4 D3.js [12]

D3.js (*Data-Driven Documents*) je knihovna napsaná v jazyce JavaScript. Tato knihovna slouží pro vytváření dynamických a interaktivních vizualizacích přímo v prohlížeči.

Pro vizualizaci dat využívá technologie jako jsou SVG, HTML5 a řadu příkazů CSS standardu.

Kapitola 6

Implementace

V této kapitole bude blíže popsána implementace nástroje. Především zde bude popsán plánovací mechanismus.

Jak již bylo zmíněno v předchozí kapitole, aplikace byla implementován pomocí následujících nástrojů: Django, Bootstrap, D3.js (obrázek 6.1).

Obrázek 6.1: Technologie použité pro implementaci.

6.1 Výsledná databáze

Po dokončení implementace aplikace bylo vygenerováno schéma databáze z `models.py`, které je na obrázku 6.2.

Ve vygenerovaném schématu chybí třída uživatel, která slouží pouze pro přístup do systému a není v jádru aplikace. Ve výsledné aplikaci je oproti návrhu několik atributů navíc. Nejvíc změn nastalo v třídě `uloha`, kde byl odstraněn atribut `zacatek`, jelikož se ukázal v plánovacícm algoritmu jako nepotřebný, dále potom přibyly následující atributy:

- `zavisi_na` – tento atribut bylo nutné přidat pro výjádření závislostí na předchozích úlohách.
- `zdroje` – atribut uchovájící informaci o potřebě konkrétních zdrojů.

U zbývajích tříd `Projekt` a `Zdroj` přibyl atribut `uloha_set`, který slouží pro uchování množin úloh u dané třídy.

6.2 Plánovací algoritmus

Použitý plánovací mechanizmus vychází z algoritmu DASA (*Depend Activity Scheduling Algorithm*), všechny plánované úlohy jsou preemptivní.

Plánovací algoritmus má následujícíj podobu:

```
1  def dasa(projekt):
2      ulohy = projekt.ulohy
3      zdroje = projekt.zdroje
4
5      volne_zdroje = zdroje
6      hotove_ulohy = []
7      naplanovane = []
8
9      cas = projekt.zacatek
10     while cas < projekt.konec and len(ulohy) > 0:
11         sort(ulohy, vc) # setridi ulohy sestupne podle hodnot vc
12         aktualni = [] # naplanovane ulohy pro aktualni hodinu
13
14         for uloha in ulohy:
15             if uloha.zbyva_do_deadline < uloha.zbyva_casu:
16                 continue
17             if not all uloha.zdroje in volne_zdroje:
18                 continue
19             if not all uloha.zavisle_ulohy in hotove_ulohy:
20                 continue
21             # uloha je planovatelna
22             remove all uloha.zdroje from volne_zdroje
23             aktualni.append(uloha)
24
25         for uloha in aktualni:
26             dec(uloha.zbyva_casu)
27             if uloha.zbyva_casu == 0:
28                 ulohy.remove(uloha)
29                 hotove_ulohy.append(uloha)
30
31         naplanovane.append(aktualni)
32
33         volne_zdroje = zdroje
34         cas += 1 hodina
35
36     return naplanovane
```

Na začátku alogritmu se vytvoří seznamy úloh a zdrojů do kterých se přiřadí patřičné úlohy a zdroje z projektu. Následně se vytvoří prázdný seznam pro hotové úlohy a prázdný seznam pro naplánované úlohy.

Plánování se provádí po hodinách od času počátku projektu, dokud je čas menší, než konečné datum projektu a zároveň není seznam úloh k naplánování prázdný. Úlohy se setřídí podle hodnoty *vc*. Tato hodnota se počítá na základě toho, zda daná úloha je, či není označená jako prioritní. Podle toho příznaku se určí hodnota *vc* jako *10/zbývající čas* pro úlohy s vysokou prioriotu a *1/zbývající čas* pro úlohy s nízkou prioritou.

V každém kroku se vytvoří seznam úloh, naplánovaných na aktuální hodinu. Pro každou úlohu se zkontroluje následující:

- Jestli zbývá dostatek času na to aby se daná úloha stihla dokončit do její deadline.
- Jestli jsou pro danou úlohu k dispozici všechny zdroje - každý zdroj, který daná úloha potřebuje, musí být v seznamu volných zdrojů.
- Jestli jsou všechny úlohy, na kterých je daná úloha závislá, již vyřízené - vyřízené úlohy jsou takové, které jsou v seznamu hotových úloh.

Pokud jsou výše uvedené podmínky splněny, potom je daná úloha plánovatelná na aktuální čas. Daná úloha se následně přidá do naplánovaných úloh.

Následně se pro každou úlohu v seznamu úloh, naplánovaných na aktuální čas, aktualizuje hodnota zbývajícího času a pokud je hodnota zbývajícího času rovna nule, je úloha umístěna přesunuta ze seznamu úloh k naplánování do seznamu hotových úloh.

Poté se všechny zdroje označí jako volné, hodnota času se inkrementuje o jednu hodinu.

6.3 Popis aplikace

V této části bude podrobněji popsána výsledná aplikace.

Přihlášení a registrace uživatele

Po zadání adresy do prohlížeče se uživateli zobrazí přihlašovací obrazovka (obrázek 6.3a), pomocí které se může buď přihlásit, nebo zaregistrovat (obrázek 6.3b).

Projekty

Po úspěšném přihlášení se uživateli vypíše seznam projektů (obrázek 6.4a). V seznamu projektů si uživatel může vybrat již exitující projekt, nebo vytvořit projekt úplně nový.

Při vytváření nového projektu se otevře dialog, do kterého uživatel zadá požadovaný název projektu a jeho začátek a konec (obrázek 6.4b).

Plánování

Po vytvoření nového projektu nebo výběru již vytvořeného se uživateli zobrazí hlavní obrazovka projektu a na ní seznamy úloh a zdrojů a možnosti pro naplánování, úpravu, smazání projektu nebo možnost exportu naplánovaných činností do kalendáře pomocí iCal (obrázek 6.5a).

Uživatel může přidávat zdroje pomocí dialogu do kterého zadá jméno a barvu zdroje v hexadecimálním kódu, kterou bude mít daný zdroj v Ganttově diagramu (obrázek 6.5b). Po přídání zdroje lze otevřít jeho detail a podívat se na informace o něm, případně ho upravit, nebo smazat. Pokud má daný zdroj přidělené nějaké úkoly, bude u zdroje zobrazen jeho vlastní plán s možností exportu plánu samotného zdroje do kalendáře přes iCal obrázek 6.5d.

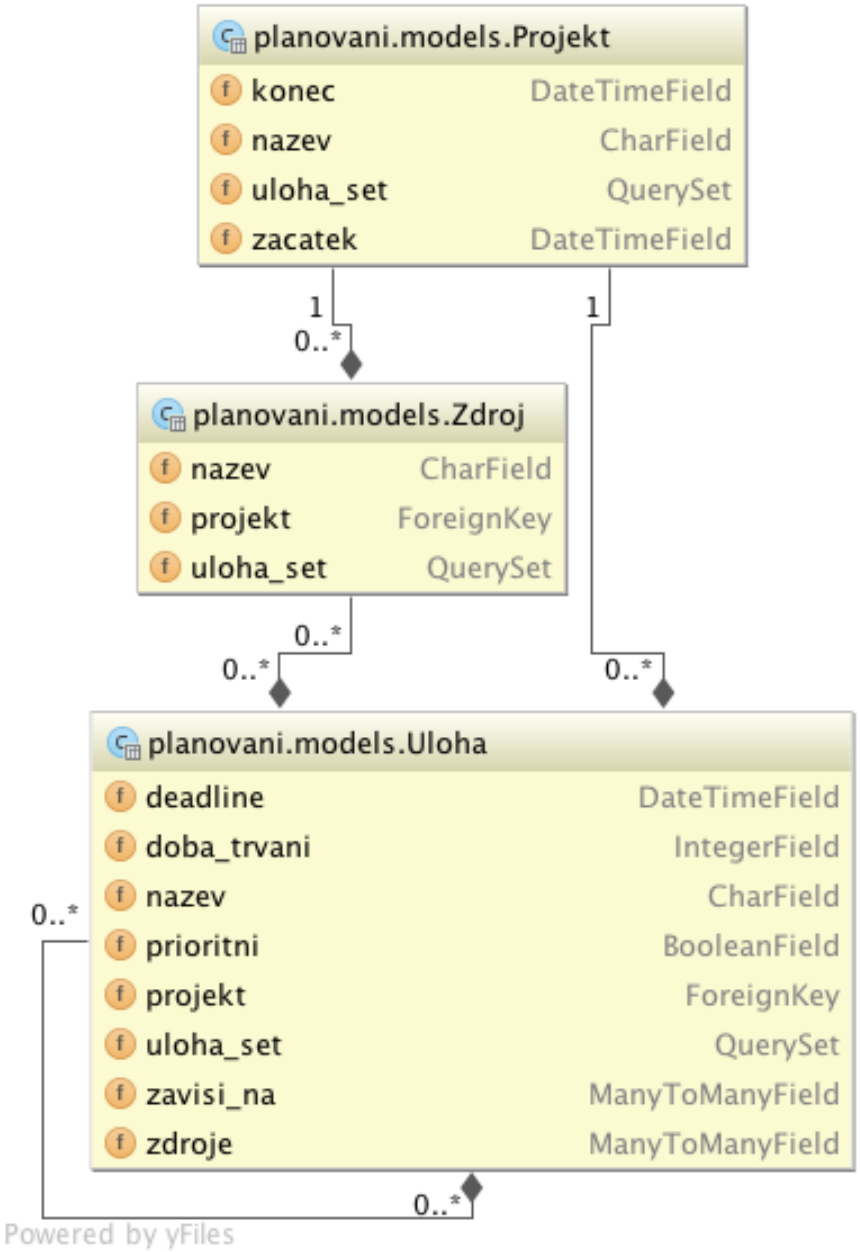
Uživatel může rovněž přidávat nové úlohy pomocí dialogu pro přidání úlohy. Pro vytvoření nové úlohy je nutné zadat její název, přidělit zdroje z nabídnutého seznamu, kterých může být i víc, vybrat datum do kterého musí být daný úkol hotový a vyplnit dobu trvání v hodinách. Volitelnými parametry potom jsou priorita úlohy a závislost na předešlých úlohách (obrázek 6.5c). V detailu úlohy potom nalezneme informace o úloze, které jsem zadali a navíc čas na který byla úloha naplánovaná (obrázek 6.5e).

Pokud zvolíme na hlavní obrazovce projektu tlačítko naplánovat, vykreslí se Ganttův graf s naplánovaným projektem (obrázek 6.6). Ganttův graf má na svislé ose vypsány postupně jednotlivé úlohy a na vodorovné má vypsánu časovou osu projektu. Jednotlivé úkoly jsou označeny barvou zdroje, která je vykonává (legenda zdrojů a jejich barev je v levém horním rohu). V případě že úlohu vykonává více zdrojů najednou, je úloha označená barvou prvního zdroje.

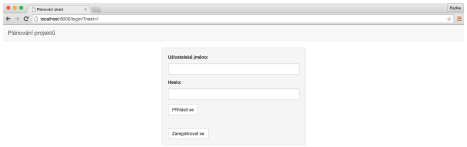
6.4 Vlastnosti nástroje

Vzniklý nástroj má následující vlastnosti:

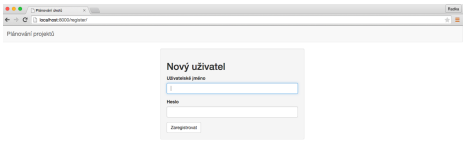
- Použitý plánovací alogitmus by se dal charakterizovat jako on-line plánovací mechanismu s preemptivním plánováním s nejlepší snahou, kde při nedodržení časových omezení se daná úloha nenaplánuje.
- Vzniklá aplikace je schopna plánovat projekty nebo osobní čas. Umožňuje přidávat k jednotlivým projektům zdroje a úlohy s patřičnými závislostmi a požadavky na zdroje. Výsledný plán je vykreslen pomocí Ganttova grafu a zároveň je i možné výsledný plán vyexportovat pomocí iCal.



Obrázek 6.2: Vygenerované schéma databáze.

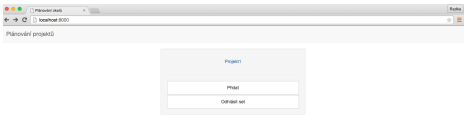


(a) Přihlašovací obrazovka

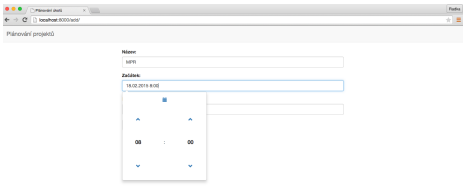


(b) Vytváření nového uživatele.

Obrázek 6.3

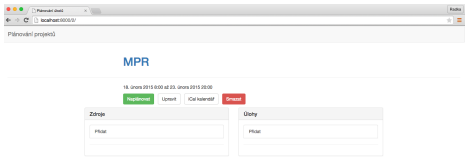


(a) Seznam projektů.

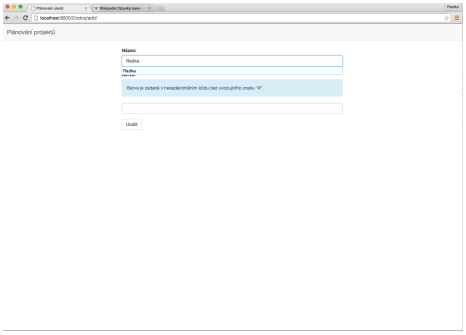


(b) Vytvoření nového projektu.

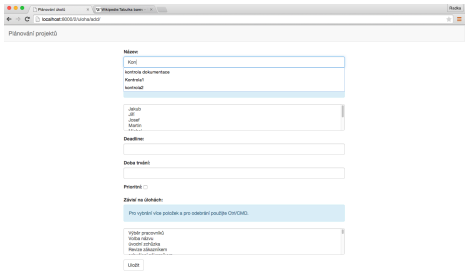
Obrázek 6.4



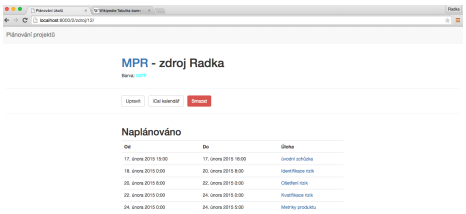
(a) Hlavní obrazovka projektu.



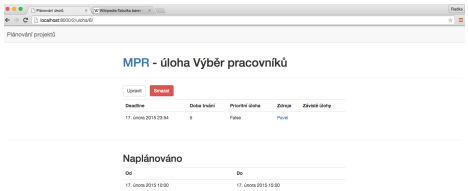
(b) Dialog pro přidání nového zdroje.



(c) Dialog pro přidání nové úlohy

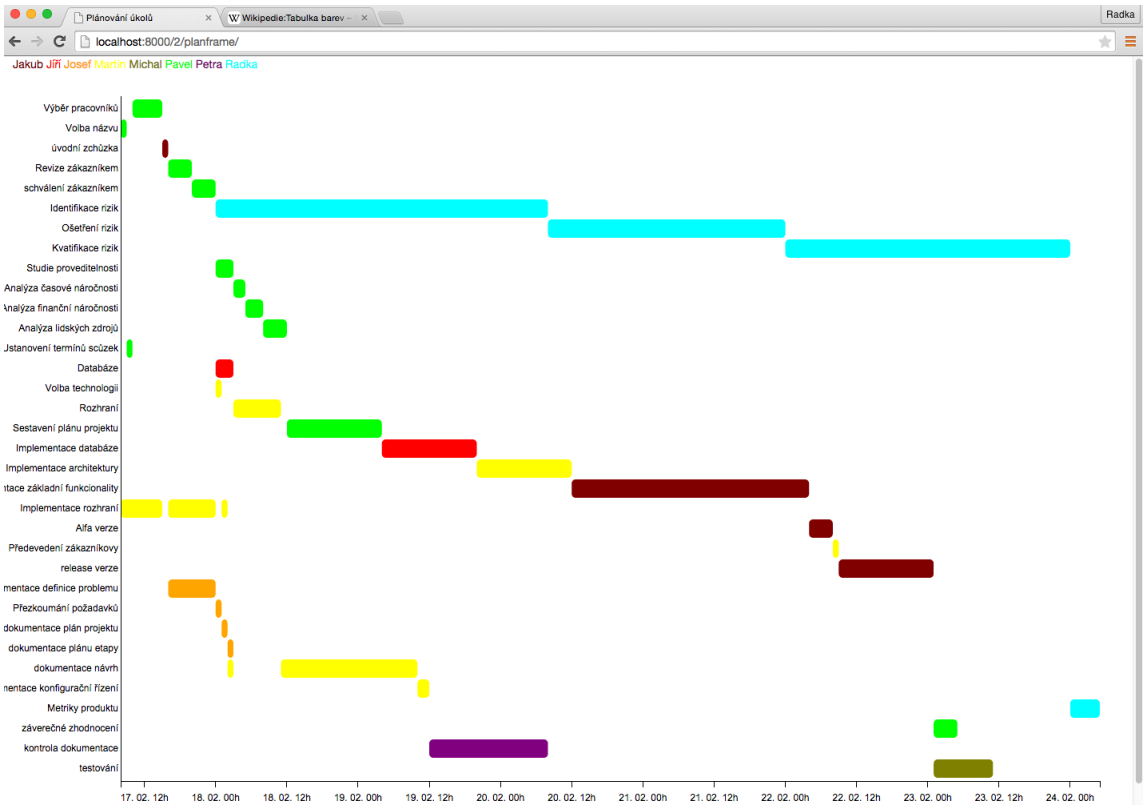


(d) Detail konkrétního zdroje.



(e) Vytvoření nového projektu.

Obrázek 6.5



Obrázek 6.6: Vykreslený Ganttův diagram

Kapitola 7

Příklady použití iplementovaného nástroje

V této kapitole bude demonstrována funkčnost výsledné aplikace. Aplikace byla testována a zkoušena v prohlížeči Google Chrome pod operačním systémem OS X Yosemite.

7.1 Plánování projektů

V prvním příkladu bude demonstrováno použití nástroje pro plánování projektu. Plán prezentovaného projektu byl částečne přebrán z projektu do předmětu MPR (Managemnet projektů).

Projekt byl plánován pro časové období od 17. února 2015 8:00 do 30. dubna 2015. V projektu byly použity následující zdroje:

- Jakub,
- Jiří,
- Josef,
- Martin,
- Michal,
- Pavel,
- Petra,
- Radka.

Plán byl generován pro následující výčet úkolů:

Název úlohy	Deadline	Doba trvání	Priorita	Zdroje	Závislost
1.Výběr pracovníků	17.2. 23:54	5	X	Pavel	X
2.Volba názvu	17.2. 23:56	1	X	Pavel	X
3.Úvodní schůzka	17.2. 23:57	1	X	Všichni	1,2
4.Revize zákazníkem	21.2. 23:58	4	X	Pavel,Petra	X
5.Schválení zákazníkem	22.2. 23:59	4	X	Pavel,Petra	4
6.Identifikace rizik	28.2. 00:08	56	X	Radka	5
7.Ošetření rizik	2.3. 0:00	40	X	Radka	6
8.Kvatifikace rizik	14.3. 00:10	48	X	Radka	7
9.Studie proveditelnosti	22.2. 00:11	3	X	Pavel	5
10.Časová náročnosti	23.2. 00:12	2	X	Pavel	5
11.Finanční náročnosti	23.2. 00:13	3	X	Pavel	5
12.Lidské zdroje	24.2. 00:14	4	X	Pavel	5
13.Termín schůzek	24.2. 00:16	1	X	Pavel	X
14.Databáze	24.2. 00:20	3	X	Jiří	5
15.Volba technologií	21.2. 00:20	1	X	Martin	5
16.Rozhraní	26.2. 00:21	8	X	Martin	14,15
17.Plán projektu	1.3. 00:16	2	X	Pavel	5
18.Implementace databáze	4.3. 00:27	16	X	Jiří	17
19.Implementace architektury	8.3. 00:28	16	X	Martin	18
20.Zákaldní funkcionalita	15.3. 00:29	40	X	Jiří,Jakub	19
21.Implementace rozhraní	13.3. 00:33	16	X	Martin	X
22.Alfa verze	15.3. 00:34	4	X	Jiří,Jakub,Martin	20,21
23.Ukázka zákazníkovi	15.3. 00:38	16	X	Martin,Pavel	22
24.Release verze	1.4. 00:41	16	X	Jiří,Jakub,Martin	23
25.Definice problému	19.2. 00:43	8	X	Josef, Michal	3
26.Přezkoumání požadavků	21.2. 00:45	1	X	Josef, Michal	25
27.Dokumentace plánu	1.3. 00:46	1	X	Josef, Michal	26
28.Dokumentace etapy	26.2. 00:48	1	X	Josef, Michal	27
29.Dokumentace návrh	7.3. 00:48	24	X	Martin	15
30.Konfigurační řízení	27.2. 00:52	2	X	Martin	X
31.Meriky produkty	31.3. 00:53	5	X	Radka	23
31.Závěrečné zhodnocení	13.4. 00:54	4	X	Pavel	24
32.Kontrola dokumentace	30.4. 00:55	20	X	Petra	25,27,28,29,30
33.Testování	30.4. 01:00	10	X	Michal	24

Výsledný plán je potom znázorněn pomocí Ganttova grafu na obrázku 6.6.

7.2 Plánování osobního času

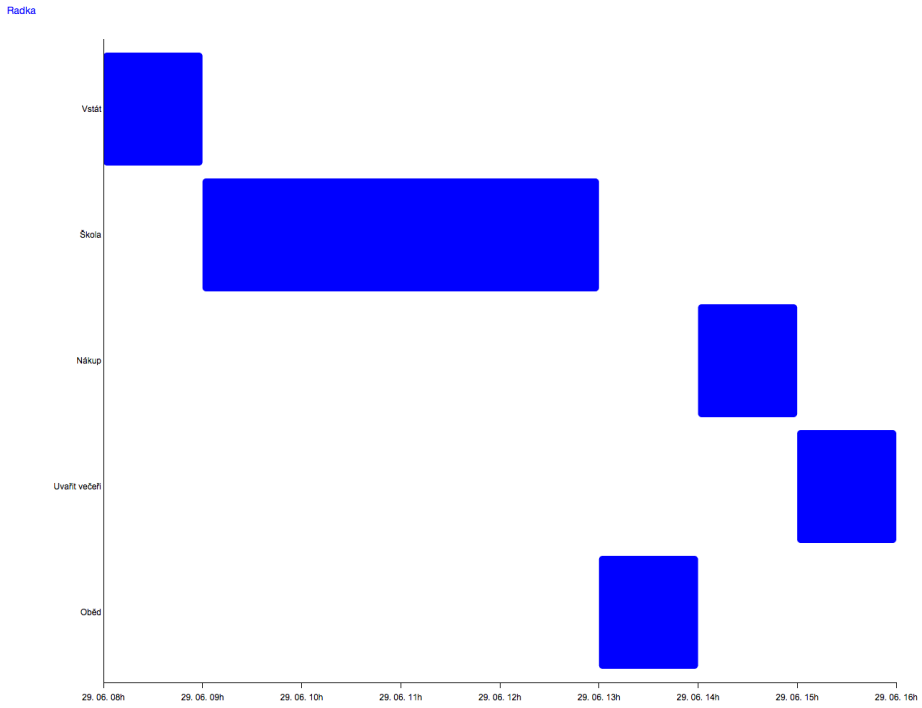
Plánování osobního času se v aplikaci uskuteční, pokud je k dispozici pouze jeden zdroj, pro který se plánuje.

Příklad na plán jednoho dne (29.6. 8:00 až 29.6. 20:00), který má následující úlohy:

Název úlohy	Deadline	Doba trvání	Priorita	Zdroje	Závislost
1.Vstát	29.6. 09:00	1	X	Radka	X
2.Škola	29.6. 15:00	4	ANO	Radka	1
3.Nákup	29.6. 17:00	1	X	Radka	1
4.Uvařit večeři	29.6. 19:00	1	X	Radka	3
5.Oběd	29.6. 15:00	1	ANO	Radka	1

Výsledný Ganttův graf je zobrazen na obrázku 7.1.

Výsledek lze exportovat do kalendáře pomocí iCal, ukázka je na obrázku 7.2.



Obrázek 7.1: Vykreslený Ganttův diagram pro osobní čas

7.3 Plánování rozvrhů

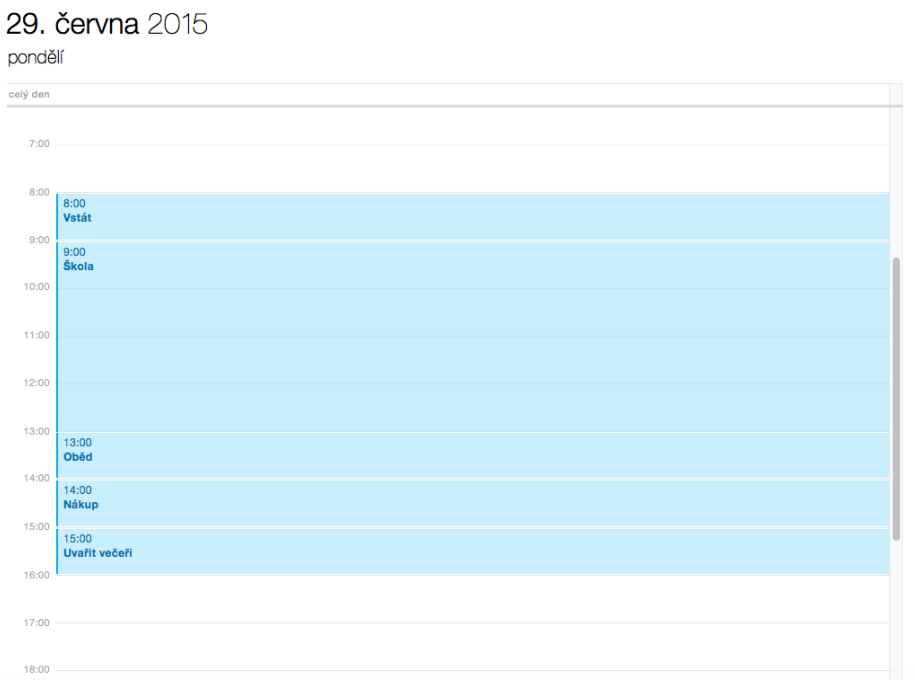
Posledním příkladem bude ukázka plánování rozvrhů. U plánování rozvrhů je k dispozici pět učitelů (zdroje: Ucitel 1-5), dále je nutné mít jako zdroj skupinu studentů (Studenti), která se účastní dané výuky.

Rozvrh se plánuje na jeden den a to konktrétně v rozmezí 2.5.2015 7:00 až 2.5.2015 19:00.

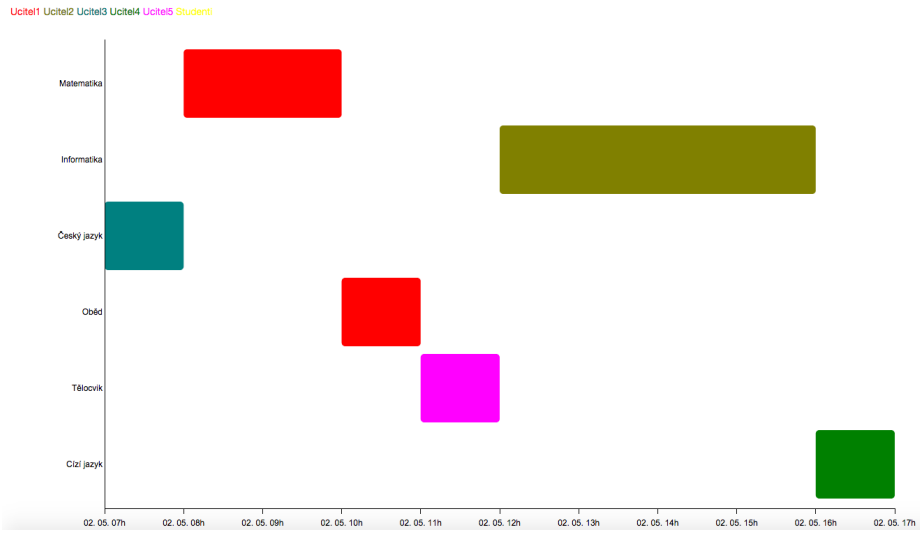
Úlohy k plánování jsou uvedeny v následující tabulce:

Název úlohy	Deadline	Doba trvání	Priorita	Zdroje	Závislost
1.Matematika	2.5. 13:00	2	X	Ucitel1, Studenti	X
2.Informatika	2.5. 19:00	4	ANO	Ucitel2, Studenti	5
3.Český jazyk	2.5. 09:00	1	X	Ucitel3, Studenti	X
4.Oběd	2.5. 13:00	1	ANO	Ucitel1-5, Studenti	1,3
5.Tělocvik	2.5. 16:00	1	X	Ucitel5, Studenti	1,4
6.Cizí jazyk	2.5. 19:00	1	X	Ucitel4, Studenti	3,4

Výsledný rozvrh je zobrazen pomocí Ganttova grafu na obrázku 7.3.



Obrázek 7.2: Export plánu do kalendáře pomocí iCal



Obrázek 7.3: Výsledný rozvrh

Kapitola 8

Závěr

V této diplomové práci bylo v druhé kapitole popsáno plánování projektů a řízení času společně s postupy, které jsou k tomu využívány. Byly zde uvedeny i existující nástroje pro plánování projektů včetně ukázky vzhledů jejich uživatelské rozhraní.

V třetí kapitole bylo popsáno plánování úloh reálného času. Byly zde uvedeny parametry úloh reálného času a dále pak algoritmy a mechanismy pro plánování takovýchto úloh.

Následující kapitola se zabývala srovnáním úlohy reálného času a úlohy v projektu, nazákladě čehož byly navrhnuty parametry úlohy používané ve výsledné aplikaci. V závěru této kapitoly byly navrženy mechanismy plánování vhodné pro aplikaci. Byl zde rovněž navržen vzhled aplikace a její databáze.

V páté kapitole byl popsán návrhový vzor MVC, pomocí kterého byla výsledná aplikace implementována. Dále zde byly blíže představeny nástroje, pomocí kterých je výsledná aplikace implementována.

Šestá kapitola je věnována již samotné implementaci aplikace a především popisu samotného plánovacího algoritmu. V závěru kapitoly pak byla představena a popsána výsledná aplikace včetně jejího uživatelského rozhraní.

Poslední kapitola této práce se věnuje příkladům použití vytvořeného nástroje v různých situacích (pro plánování projektů, plánování osobního času a plánování rozvrhů).

Cílem této práce bylo navrhnou a implementovat nástroj pro plánování projektů a času s využitím principů z oblasti plánování úloh reálného času. Na základě tohoto zadání byla navrhnuta aplikace využívající pro plánování algoritmus, který je založen na algoritmu DASA a je rozšířen o sdílení zdrojů a rovněž řeší závislosti mezi úlohami.

Možnosti dalšího vývoje by mohly být následující: V současné době nelze jednotlivé zdroje omezovat a proto by bylo vhodné do aplikace tuto možnost doiplmementovat. Další možností rozvoje aplikace, která nevylučuje první možnost, je udělat z aplikace nástroj pro plánování a sledování průběhu projektu jak v čase, tak i například pro tok finančních prostředků vzhledem k finančnímu plánu projektu, který by mohl být rovněž součástí nástroje. Dále by potom jistě bylo vhodné, aby aplikace podporovala více možností exportu. V současném stavu aplikace umožňuje export pouze pomocí iCal.

Literatura

[1] Cottet, F.; Delacroix, J.; Kaiser, C.; aj.: *Basic Concepts*. John Wiley & Sons, Ltd, 2003, ISBN 9780470856345, s. 1–21, doi:10.1002/0470856343.ch1.
URL <http://dx.doi.org/10.1002/0470856343.ch1>

[2] J. Kreslíková: Výukové texty k předmětu MRP. Dostupné v informačním systému /Soubory k předmětům/Management projektů/Studijní opory.

[3] J. Strnadel: Výukové texty k předmětu ROS. Dostupné v informačním systému /Soubory k předmětům/Real-time operační systémy.

[4] J. Zendulka: MVC u webových apikací. https://www.fit.vutbr.cz/study/courses/AIS/private/prednasky/7_WebMVC.pdf.

[5] J.Doležal; P.Máchal; B.Lacko; aj.: *Projektový management podle IPMA*. Grada Publishing, 2009, iSBN 978-80-247-2848-3.

[6] Ježková, Z.; Krejčí, H.; Lacko, B.; aj.: *Projektové řízení: Jak zvládnout projekty*. Akademické centrum studentských aktivit, iSBN 978-80-905297-1-7.

[7] K. Schwalbe: *Řizení projektů v IT*. Computer Press, 2010, iSBN 978-80-251-2882-4.

[8] Lindh, F.; Otnes, T.; Wennerström, J.: Scheduling Algorithms for Real-Time Systems. http://www.idt.mdh.se/kurser/ct3340/ht10/FinalPapers/2-Lindh_Otnes_Wennestrom.pdf.

[9] Mohammadi, A.; Akl, S. G.: Scheduling Algorithms for Real-Time Systems. http://research.cs.queensu.ca/home/akl/techreports/scheduling.pdf.

[10] Společnost pro projektové řízení Česká republika: Výkladový slovník pojmů. http://ipma.cz.

[11] WWW stránky: Bootstrap. http://www.getbootstrap.com.

[12] WWW stránky: D3 Data-Driven Documents. http://d3js.org.

[13] WWW stránky: The Django Book. http://www.djangobook.com.

[14] WWW stránky: Django projects. http://www.djangoproject.cz/.

[15] WWW stránky: IT network. http://www.itnetwork.cz/mvc-architektura-navrhovy-vzor.

[16] ČSN ISO 10006: Systémy managementu jakosti – Směrnice pro management jakosti projektů. 2004.

Příloha A

Obsah CD

Adresářová struktura přiloženého CD:

- doc – zdrojové kody textové zprávy
- aplikace – zdrojové kody aplikace
- readme