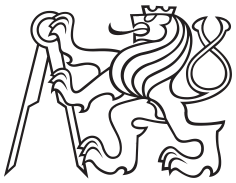
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Webové komponenty pro plánovaní údržby letadel

Vít Pluhař

Školitel: Mgr. Miroslav Blaško, Ph.D.
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pluhař** Jméno: **Vít** Osobní číslo: **487609**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Webové komponenty pro plánovaní údržby letadel

Název bakalářské práce anglicky:

Web components for aircraft maintenance planning

Pokyny pro vypracování:

Cílem práce je vyvinout webové komponenty v React frameworku [1] pro plánování údržby letadel. Komponenty by měli umožnit řízenou tvorbu plánu údržby, kde vstupem jsou historické data z plánování a doménová znalost vyjádřená pomocí ontologie sémantického webu [2,3]. Projekt bude realizován ve spolupráci s odborníky z domény údržby, kteří na základě sběru dat z CSAT ověří vyvinuté komponenty na reálných datech z provozu.
Instrukce:
1) seznámte se s technologiemi sémantického webu pro reprezentaci doménové znalosti (OWL, RDF, JSON-LD)
2) prozkoumejte existující nástroje a knihovny pro vizuální konstrukci plánů
3) analyzujte požadavky na webové komponenty
4) navrhnete a implementujte webové komponenty
5) implementovaný prototyp otestujte na vybraných scénářích pomocí alespoň tří doménových expertů

Seznam doporučené literatury:

[1] React - A JavaScript library for building user interfaces (https://reactjs.org/)
[2] Guizzardi, Giancarlo. "Ontological foundations for structural conceptual models." (2005).
[3] Lanthaler, Markus, and Christian Gütl. "On using JSON-LD to create evolvable RESTful services." Proceedings of the Third International Workshop on RESTful Design. ACM, 2012

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Mgr. Miroslav Blaško, Ph.D. skupina znalostních softwarových systémů

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

Datum zadání bakalářské práce: **14.02.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **19.02.2024**

Mgr. Miroslav Blaško, 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

Poděkování

Rád bych poděkoval především Mgr. Miroslavovi Blaškovi, Ph.D. za odborné vedení projektu, za veškerou nápomoc a cenné rady při tvorbě bakalářské práce, za vstřícnost a ochotu při pravidelných schůzkách.

Dále bych rád poděkoval všem, kteří mi byli nápomocni při tvorbě této práce.

V neposlední řadě bych rád poděkoval celé své rodině a všem přátelům, kteří mě podporovali,a to jak při tvorbě bakalářské práce, tak během průběhu celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

V Praze, dne 20. května 2022

Abstrakt

Předmětem této bakalářské práce je vytvořit ve frameworku React webové komponenty pro plánování údržby letadel. Práce je realizována ve spolupráci s odborníky z domény údržby. Cílem této práce je vytvořit komponenty podporující plánování údržby na různých úrovních detailu a umožnit zobrazení vykázané práce. Výsledkem této práce je prototyp otestovaný pomocí čtyř doménových expertů.

Klíčová slova: komponenta pro plánování údržby letadel, komponenta pro plánování, plánování údržby, údržba letadel, react komponenta pro plánování údržby

Školitel: Mgr. Miroslav Blaško, Ph.D. Praha 2, Karlovo náměstí 13, E-116

Abstract

The subject of this bachelor thesis is to create web components for aircraft maintenance planning in the React framework. The work is carried out in collaboration with experts from the maintenance domain. The goal of this thesis is to create components that support maintenance planning at different levels of detail and support to display performed work. The result of this work is a prototype tested by four domain experts.

Keywords: aircraft maintenance planning component, planning component, maintenance planning, aircraft maintenance, react maintenance planner

Title translation: Web components for aircraft maintenance planning

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Plánování údržby letadel je velmi rozsáhlá záležitost. Podle historických dat získaných ze sběru dat z CSAT plány údržby dosahují až tisíce různých úkolů. Každý úkol se vztahuje k nějaké části letadla, na které je při plnění daného úkolu vykonávána údržba. Pro vykonání různých skupin úkolů je potřeba, aby stav letadla splňoval různá kritéria. Mezi taková kritéria může patřit, jestli je letadlo pod elektrickým proudem, stojí na zemi nebo je na zvedáku. Některé úkoly na sebe mohou přímo navazovat a nelze na navazujícím úkolu začít pracovat, dokud není předchozí úkol hotový. Zároveň je při plánování potřeba počítat s omezenými lidskými zdroji[1].

Proto je potřeba plánovat tak, aby byly úkoly prováděny ve chvíli, kdy stav letadla splňuje kritéria pro vypracování úkolu. V plánu je také potřeba zohlednit, aby úkoly, jejichž dokončení je nutné pro plnění dalších úkolů, byly provedeny s dostatečnou časovou rezervou před zahájením navazujících úkolů.

1.1 Motivace

V současné situaci nemají odborníci z domény údržby CSAT dostupný žádný konkrétní software pro plánování údržby letadel. Tvorba plánu aktuálně probíhá pomocí tabulkového procesoru excel. Plán na první pohled působí nepřehledně a existuje zvýšené riziko přehlédnutí nedostatků plánu. Navíc kvůli množství úkolů je možné plánovat jenom na vyšší úrovni abstrakce (tedy pro větší skupiny úkolu) a je těžké zpětně zakomponovat aktuální plnění plánu a na základe této informace plán pozměnit.

V projektu byla vytvořena i aplikace pro přehled plnění plánů údržby[1] a je tedy vhodné vytvořit komponenty pro plánování údržby letadel, které se v dané aplikace mohou použít, navržené tak, aby co možná nejvíce usnadnilo plánování a eliminovalo riziko výskytu nedostatků plánů vytvořených lidským faktorem.

[1] https://dspace.cvut.cz/handle/10467/99084

1.2 Cíl

Cílem je vytvořit komponenty podporující plánování údržby, aby bylo možné pomoci těchto komponent plánovat údržbu letadel na různých úrovních detailu a v různé době. Tedy ne jenom na začátku před spuštěním revize letadla, ale i během revize. Komponenty by měly být schopny zobrazit aktuálně plnění plánu a na základě této informace umožnit plán změnit. Komponenty musí být dostatečně výkonné pro práci s velkou množinou vstupních dat odpovídající datům údržbové organizace CSAT.

Kapitola 2
Uvedení do problematiky

2.1 Údržba letadel

Údržba je součástí provozu letadla. Zajišťuje bezpečnost, spolehlivost a letovou způsobilost. Každé letadlo se musí podrobit údržbě, za kterou je zodpovědné letecké oddělení údržby letadel. Údržba se provádí podle požadavků výrobce letadla a společnosti vlastnící letadlo[1].

Z provozní důvodů je údržba rozdělena na údržbu na letadle a údržbu mimo letadlo. Údržba na letadle se dále dělí na údržbu linky a hangárovou údržbu letadel. Údržba mimo letadlo obsahuje různé druhy údržbářských dílen provádějících údržbu vyjmutých částí a dílů z letadla[1].

Údržba linky zahrnuje pravidelné krátké prohlídky mezi přílety a odlety z letiště. Ačkoliv údržba linky nevyžaduje velké investice ani pracovní síly, její četnost představuje významnou část nákladů na údržbu letadla[2].

Hangárová údržba letadel se provádí na letadlech mimo provoz, nehledě na to, jestli letecká společnost disponuje hangárem pro tuto činnost. Patří sem jakákoliv větší údržba nebo modifikace letadel. Do hangárové údržby patří:

1. Plán letové kontroly
2. Modifikace draku letadla (trup, nosné plochy, ocasní plochy) nebo systémů podle servisních bulletinů, příkazů k zachování letové způsobilosti nebo technických příkazů
3. Směrnice pro kampaně flotily
4. Demontáž a montáž leteckých motorů
5. Úpravy interiéru letadla
6. Zvláštní inspekce vyžadované FAA (např. antikorozní program)

Každá návštěva hangáru může zahrnovat různé kombinace uvedených aktivit za účelem dosažení cílů údržby a minimalizace doby, kdy je letadlo mimo provoz[1].

2.2 UML notace

Tato sekce vysvětluje význam některých použitých UML.

Abstraktní třída - umožňuje definovat společnou datovou strukturu, rozhraní a společné chování pro všechny odvozené třídy. Vztah se vyznačuje plnou čárou s vyplněnou šipkou na konci, který je připojen k abstraktní třídě[5].
Asociační vztah - strukturální vztah mezi dvěma třídami popisující důvody a pravidla, jimiž se tento vztah řídí a vyznačuje se plnou čárou[6].
Kompozice - vztah představující celek a jeho části. Životnost těchto částí závisí na životnosti celku. Vztah se vyznačuje jako plná čára s vyplněným kosočtvercem na konci asociace, který je připojen k celku[7].
Agregace - speciální typ asociace zobrazující soustavu. Objekty jsou sestaveny dohromady, aby vytvořili složitější objekt. Agregace popisuje skupiny objektů, způsob interakce s nimi a chrání integritu sestavy objektů definováním jediného řídícího bodu nazývaného agregát, který představuje sestavu. Jednotlivé části sestavy mohou patřit do více sestav a zároveň mohou existovat nezávisle na agregátu. Agregační asociace se zobrazuje jako plná čára s nevyplněným kosočtvercem na konci asociace, který je spojený s agregátem[9].
Reflexivní asociace - podtyp asociačního vztahu, ve kterém mohou být instance stejné třídy ve vzájemném vztahu[10].

Kapitola 3

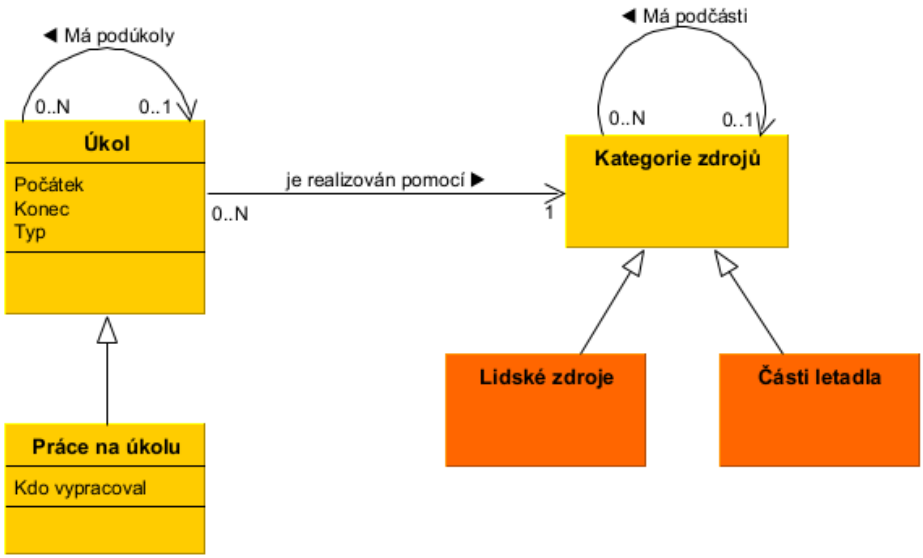
Analýza a návrh aplikace

Tato kapitola popisuje konceptuální model, datovou strukturu vstupních dat, požadavky pro vytvoření plánovací komponenty, scénáře a případy použití, rešerši knihoven, které by mohly být vhodné jako základ pro samotnou tvorbu plánovací komponenty, porovnání nejvhodnějších z vybraných knihoven a na závěr datovou strukturu plánovací komponenty.

Při provedení analýzy jsem měl k dispozici nahlédnutí na doménovou ontologii plánování, která byla popsána technologiemi RDF[1] a OWL[2]. Tato doménová ontologie představuje konceptuální model části domény plánování, která se používala v rámci projektu Zvýšení efektivity plánování a provádění údržby služeb dopravních letadel[3]. Konkrétní data byla reprezentována pomocí jazyka JSON-LD[4], což je extenze jazyka JSON pro technologii RDF, která kromě reprezentace dat tato data propojuje s jejich významem tím, že jednotlivé části schématu JSON jsou namapované na významy definované v ontologii.

[1] https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
[2] https://www.w3.org/TR/2004/REC-owl-features-20040210/
[3] https://starfos.tacr.cz/cs/project/CK01000204?query_code=2jiiaackhjiq
[4] https://www.w3.org/TR/2020/REC-json-ld11-20200716/

3.1 Konceptuální model



Obrázek 3.1: UML diagram tříd konceptuálního modelu

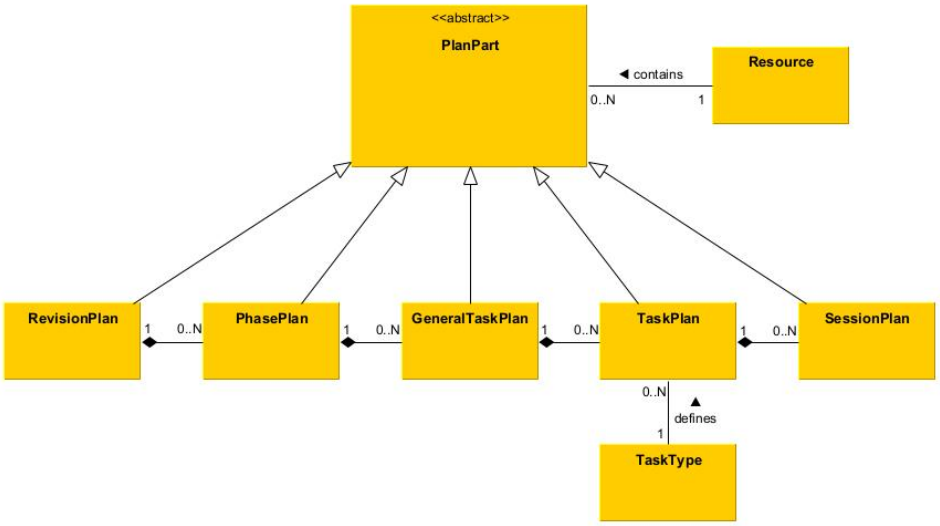
Konceptuální model na obrázku 3.1 obsahuje úkoly a kategorie zdrojů obsažené v plánovací komponentě. Kategorie zdrojů reprezentují jednotlivé části letadla vymezující omezený prostor, ve kterém mohou lidské zdroje provádět údržbu (letadlo, kokpit, trup), nebo lidské zdroje provádějící údržbu.

Lidské zdroje a části letadla jsou obsaženy v konceptuálním modelu pouze pro pochopení, co představují kategorie zdrojů. Z pohledu plánovací komponenty jsou chápány všechny kategorie stejným způsobem, a proto jsou v diagramu odlišeny jinou barvou.

Do kategorií spadají úkoly, které jsou vykonávány daným zdrojem. Úkol může nabývat speciálního typu *Práce na úkolu* představující vykonávanou práci lidskými zdroji. Některé úkoly mohou být detailněji popsány dílčími úkoly. Stejně tak kategorie mohou být blíže specifikovány dílčími kategoriemi.

3.2 Datová struktura vstupních dat

Datová struktura vstupních dat popisuje získaná vstupní data od doménových odborníků na základě sběru historických dat z domény údržby letectví. Díky těmto získaným datům je možné vizualizovat strukturu vstupních dat pomocí diagramu tříd.



Obrázek 3.2: UML diagram tříd datové struktury vstupních dat

Diagram tříd datové struktury plánu na obrázku 3.2 poukazuje na složení plánu z pěti různých částí vycházejících z abstraktní třídy *PlanPart*. Jednotlivé části detailněji specifikují část plánu, ke které je připojena kompozičním vztahem. Část *TaskPlan* navíc obsahuje informace, o jaký typ úkolu se jedná. Část *SessionPlan* představuje vykázanou práci.

Každá část plánu je přiřazena do nějaké kategorie zdrojů a zároveň do jedné kategorie může být přiřazeno více částí najednou.

3.3 Požadavky

Jak vychází ze zadání, hlavním požadavkem je, aby plánovací komponenta byla vyvinuta ve frameworku React. Následně jsou popsány jednotlivé požadavky, jaké by měla komponenta umět, včetně označení jejich prioritizace vyjádřené technikou MoSCoW, která rozděluje požadavky do 4 kategorií[20].

Must have (M)

Plánovací komponenta musí umět tyto požadavky.

Should have (S)

Plánovací komponenta by měla umět tyto požadavky.

Could have (C)

Plánovací komponenta by mohla umět tyto požadavky. Jsou žádoucí, ale méně důležité.

Won't have this time (W)

Bylo by dobré, kdyby tyto požadavky plánovací komponenta uměla, ale nejsou nutné pro správnou funkčnost komponenty. Zaměříme se na ně až v případě dostatku času.

3.3.1 Vykreslení úkolu

FR1 Zobrazit úkol (M)

Plánovací komponenta zobrazí veškeré přijaté úkoly na časové ose dle následujících kritérií

- Počáteční datum
- Konečné datum
- Kategorie

FR2 Závislost úkolu (M)

Jednotlivé úkoly mohou být závislé na některém z dalších úkolů. Je tedy potřeba zobrazit jejich závislost i v plánovací komponentě.

FR3 Zobrazení různých typů úkolů (S)

Každý úkol zobrazený v plánovací komponentě může být nějakého libovolného typu.
Příkladové typy úkolů:

- task_card
- scheduled_wo
- maintenance_wo

Jednotlivé úkoly bude možné odlišit podle jejich typu.

FR4 Logování práce na úkolu (C)

Doba trvání úkolu určuje dobu, ve které je potřeba úkol zpracovat, nikoliv však skutečný čas potřebný na jeho vypracování. Proto je zajímavá informace, jakou reálnou dobu trvalo lidským zdrojům vykonání úkolu.

FR4.1 Zobrazení logu
V plánovací komponentě bude zobrazeno

- Kdy
- Kdo
- Jakou dobu

na určitém úkolu pracoval.

FR4.2 Editace logu
Jednotlivé logy je možné upravit nebo smazat.

FR5 Progres úkolu (C)

Tento požadavek je závislý na požadavku **RQ4**.
Každý úkol může mít odhad doby potřebné pro jeho vykonání. Následně lze na základě logů zobrazit progres zpracování úkolu.

FR6 Hierarchická struktura

FR6.1 Zobrazení v hierarchické struktuře (S)
Každý úkol může mít úkoly jemu přiřazené a tuto strukturu je třeba hierarchicky zobrazit v plánovací komponentě.

FR6.2 Zvýraznění v hierarchické struktuře (C)
Selekcí úkolu se zároveň zvýrazní i jeho přiřazené úkoly, aby bylo v plánovací komponentě zřetelné, jaké úkoly k danému úkolu patří.

FR7 Zvýraznění skupiny úkolů (S)

Možnost zvýraznit určité skupiny úkolů dle specifických požadavků.
Například:

- Jack ON
- Jack OFF
- Konflikt

FR8 Vyznačení aktuálního času (S)

Na časové se bude pro přehlednost vyznačen aktuální čas.

FR9 Zobrazení detailu úkolu (S)

Při selekci úkolu by bylo vhodné uživateli zobrazit detailní informace o zvoleném úkolu.

3.3.2 Ovládání

FR10 Zvýraznění úkolů (S)

Zvýraznění úkolů na základě uživatelské akce.
Například:

- Selekce úkolu
- Změna doby trvání úkolu
- Pohyb úkolu po časové ose
- Vyhledávání
- Zobrazení konfliktu

FR11 Manuální úprava úkolu

FR11.1 Úprava jednoho úkolu (M)
Každý úkol bude možné upravit:

- Pohyb po časové ose
- Změnit dobu trvání
- Přesunout do jiné kategorie
- Vytvořit závislost
- Zrušit závislost

FR11.2 Práce s více úkoly najednou (S)
Bude možné označit více úkolů najednou a následně je společně upravit.

FR11.3 Uzamčení úkolu (C)
Možnost uzamčení úkolu, aby jej nikdo nemohl upravit, a následně přidat poznámku, proč je úkol uzamčený.

FR12 Zaměření na vybranou skupinu úkolů (C)

Zaměření na skupinu jednoho nebo více vybraných úkolů a jejich zvýraznění. Například při rozkliknutí konfliktu.

FR13 Přepínání zobrazení (C)

Přepínání mezi zobrazením plánu podle kategorií, zdrojů nebo jiných parametrů.

FR14 Export (C)

Možnost exportovat plán.

FR14.1 Excel
Export plánu do excelu.

FR14.2 SVG
Export plánu do SVG.

FR15 Podpora pro sestavení plánu pomocí externí služby z historických dat (S)

Úkoly se v plánu automaticky sestaví podle historických dat.

FR16 Plánování zdrojů (S)

Možnost zahrnutí zdrojů do plánů.
Například mechaniky.

FR17 Vyhledávání úkolů (W)

Plánovací komponenta bude obsahovat vyhledávací pole, které umožní uživateli vyhledat zobrazené úkoly a následně je zaměřit na časové ose.

FR18 Ovládání pohybu po časové ose (W)

Podpora pro jednoduché ovládání po časové ose v rámci časového období, ve kterém jsou zobrazeny úkoly.

FR19 Rozšiřitelnost panelu kategorií (C)

Podpora pro upravení šířky postranního panelu, ve kterém jsou zobrazeny kategorie.

FR20 Změna stavu komponenty

FR20.1 Krok zpět (C)
Možnost navrátit zpět provedené akce v plánovací komponentě.

FR20.2 Krok vpřed (C)
Možnost obnovit navrácené akce viz **RQ22.1**.

3.3.3 Omezení

FR21 Omezení úkolu

FR21.1 Dle typu úkolu (M)
Některé úkoly vyžadují specifické nastavení. Příkladem jsou úkoly, které proběhly v minulosti, u kterých je nežádoucí měnit dobu jejich trvání nebo s nimi pohybovat, nebo právě probíhající, u kterých je žádoucí pouze změna doby trvání vpřed v čase.

FR21.1.1 Omezení změny doby trvání (M)
Nelze změnit dobu trvání úkolu, a to buď od počátku, nebo od konce.

FR21.1.2 Omezení minimální / maximální doby trvání (M)
Úkol může mít určenou minimální nebo maximální dobu trvání. Následně není možné úkolu nastavit kratší nebo delší dobu trvání, než má úkol povolené.

FR21.1.3 Zamezení pohybu (M)
Úkol má pevně stanovenou kategorii. Nelze měnit kategorii, do které patří.

FR21.1.4 Pevně daná doba trvání (S)
Úkol má pevně danou dobu trvání. Nelze ji zmenšit ani zvětšit.

FR21.2 Dle uživatelské role (C)
Uživatelé mohou mít omezená práva k úpravám v plánovací komponentě. Například uživatel bude smět pouze:

- Zobrazit plánovací komponentu
- Upravovat jen určitý typ úkolů
- Pohybovat úkoly v čase
- Změnit dobu trvání úkolu

FR22 Porušení omezení (S)

Zobrazení porušení omezení, jejich zvýraznění a návrh na vyřešení. Tato porušení omezení a návrhy na vyřešení vypočítává externí služba.

FR23 Skóre kvality plánu (W)

Závislé na požadavku **RQ24**.
Komponenta na základě výpočtu porušení omezení externí službou určí skóre kvality plánu.

FR24 Prioritní úkoly (S)

Některé úkoly mohou být označeny jako prioritní (měly by být naplánovány přednostně).

3.3.4 Časové vymezení

FR25 Milníky (M)

Na časové ose bude možné

- Zobrazovat
- Přidávat
- Upravovat
- Odebírat

milníky.

FR26 Deadline kategorie (S)

Každá kategorie může mít vlastní deadline pro dokončení. Deadline je třeba možné vyznačit na časové ose.

FR27 Fáze projektu (C)

Podpora pro vymezení fází v projektu.

3.3.5 Nefunkční požadavky

NFR1 Podpora pro mobilní zařízení (S)

Komponenta bude správně fungovat i na mobilních zařízeních.

NFR2 Výkonnost pro zobrazení velké množiny úkolů (M)

Komponenta musí být dostatečně výkonná i pro zobrazení reálných dat CSAT.

NFR3 Automatizované nasazení změn na server (S)

Všechny změny nahrané do repositáře se automaticky nahrají na produkční server. A to změny ve všech branchích repositáře.

NFR4 Build znovu použitelné knihovny

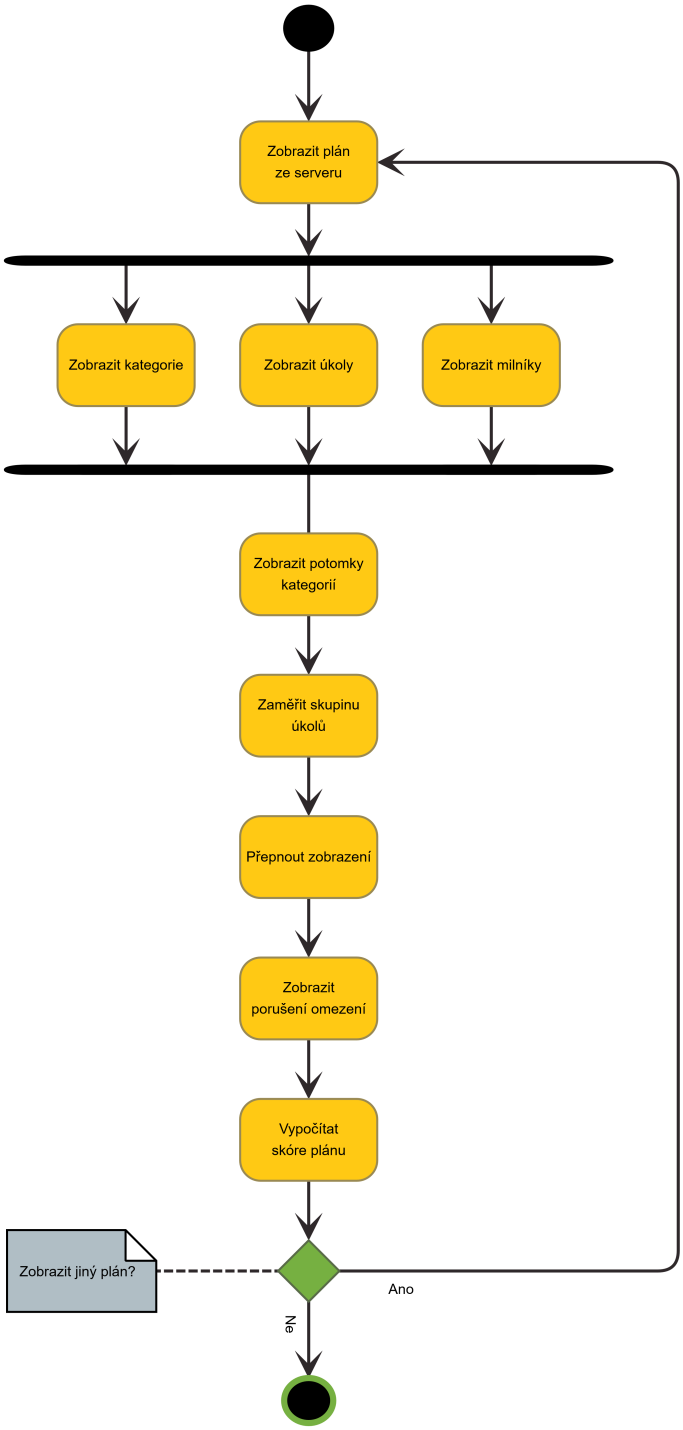
Možnost buildu plánovací komponenty jako znovu použitelnou knihovnu react komponent.

NFR5 Framework React

Plánovací komponenta bude vyvinuta ve frameworku React

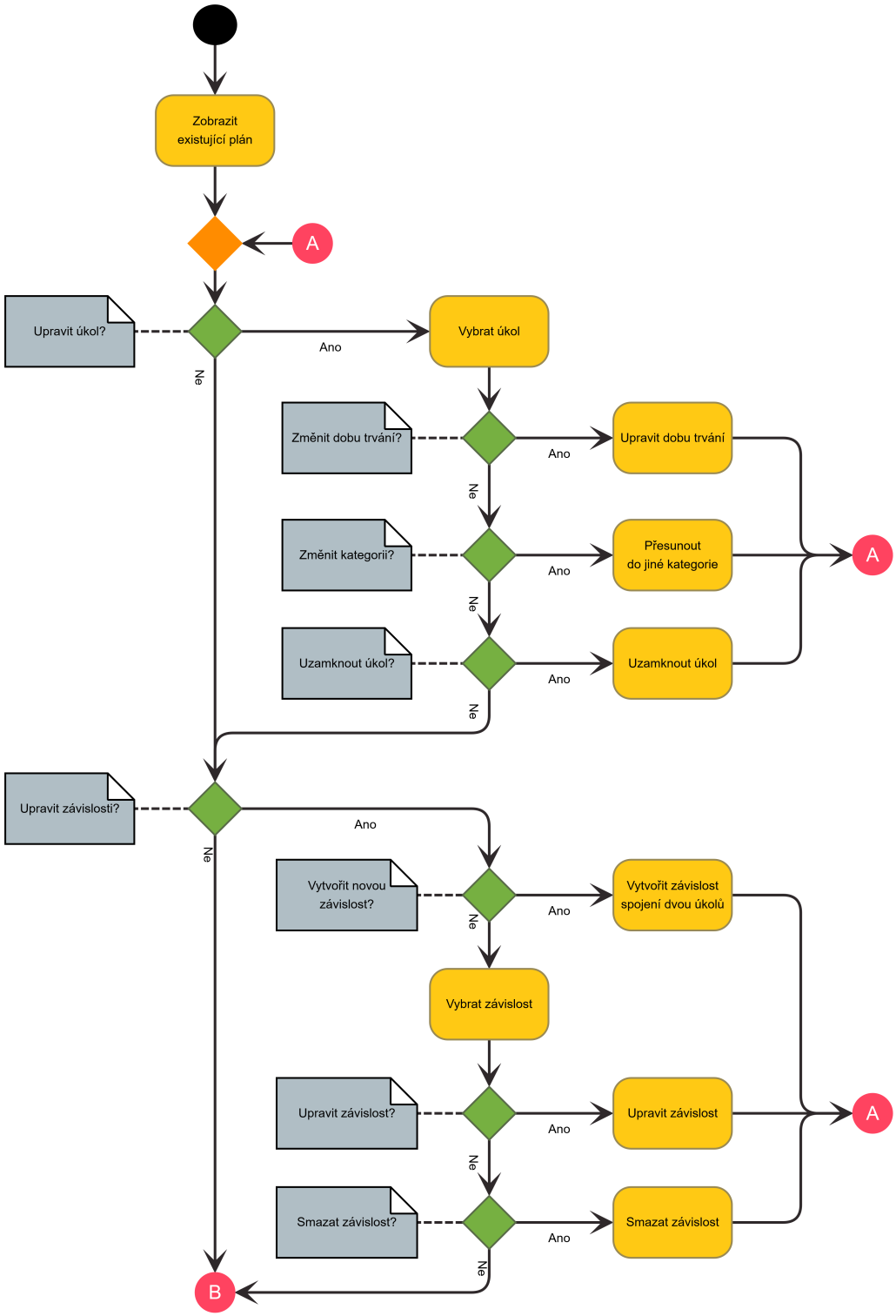
3.4 Scénáře případů užití

Zobrazení a zorientování se v plánu ze serveru

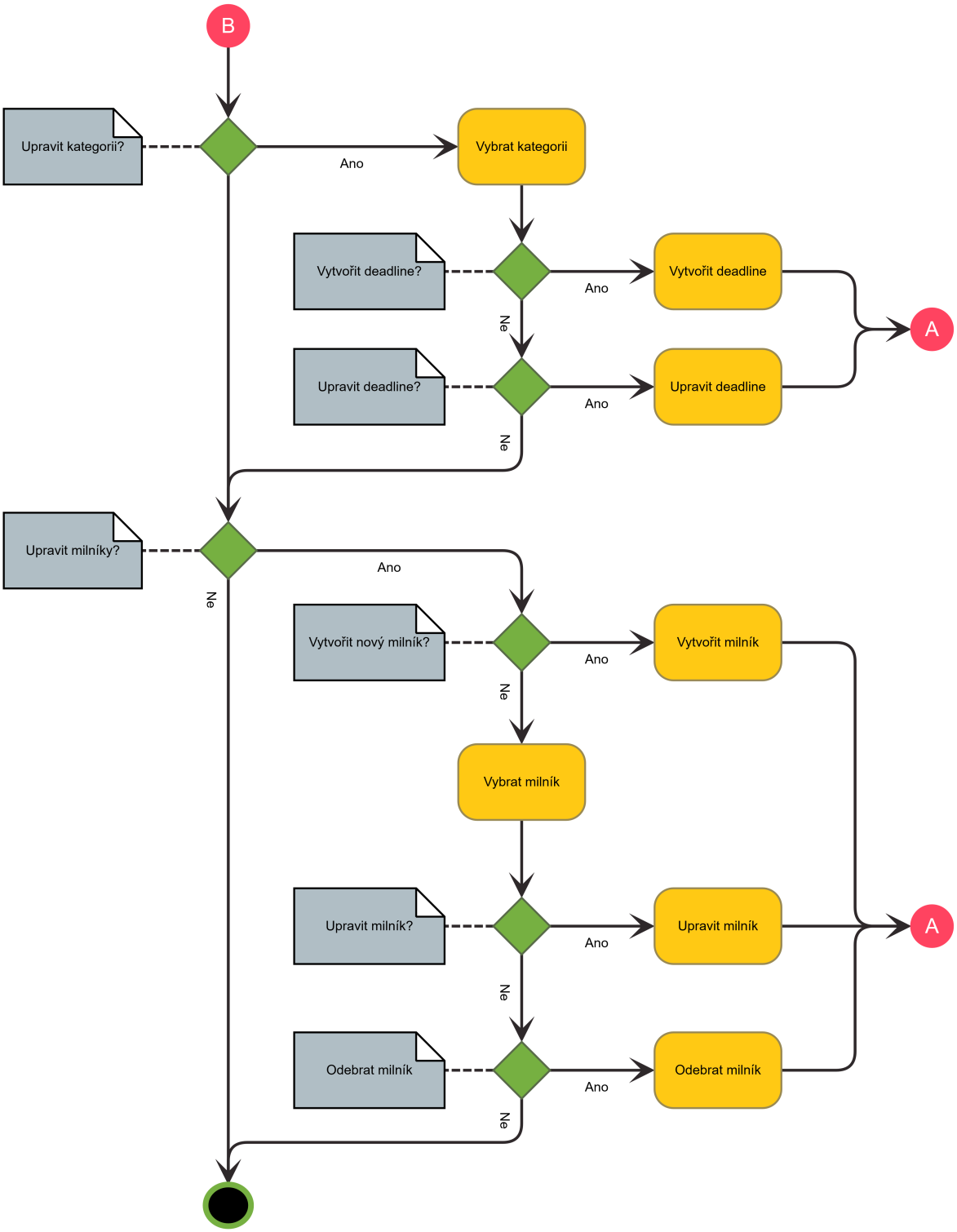


Obrázek 3.3: UML diagram aktivit zobrazení a orientace v plánu

Optimalizace existujícího plánu

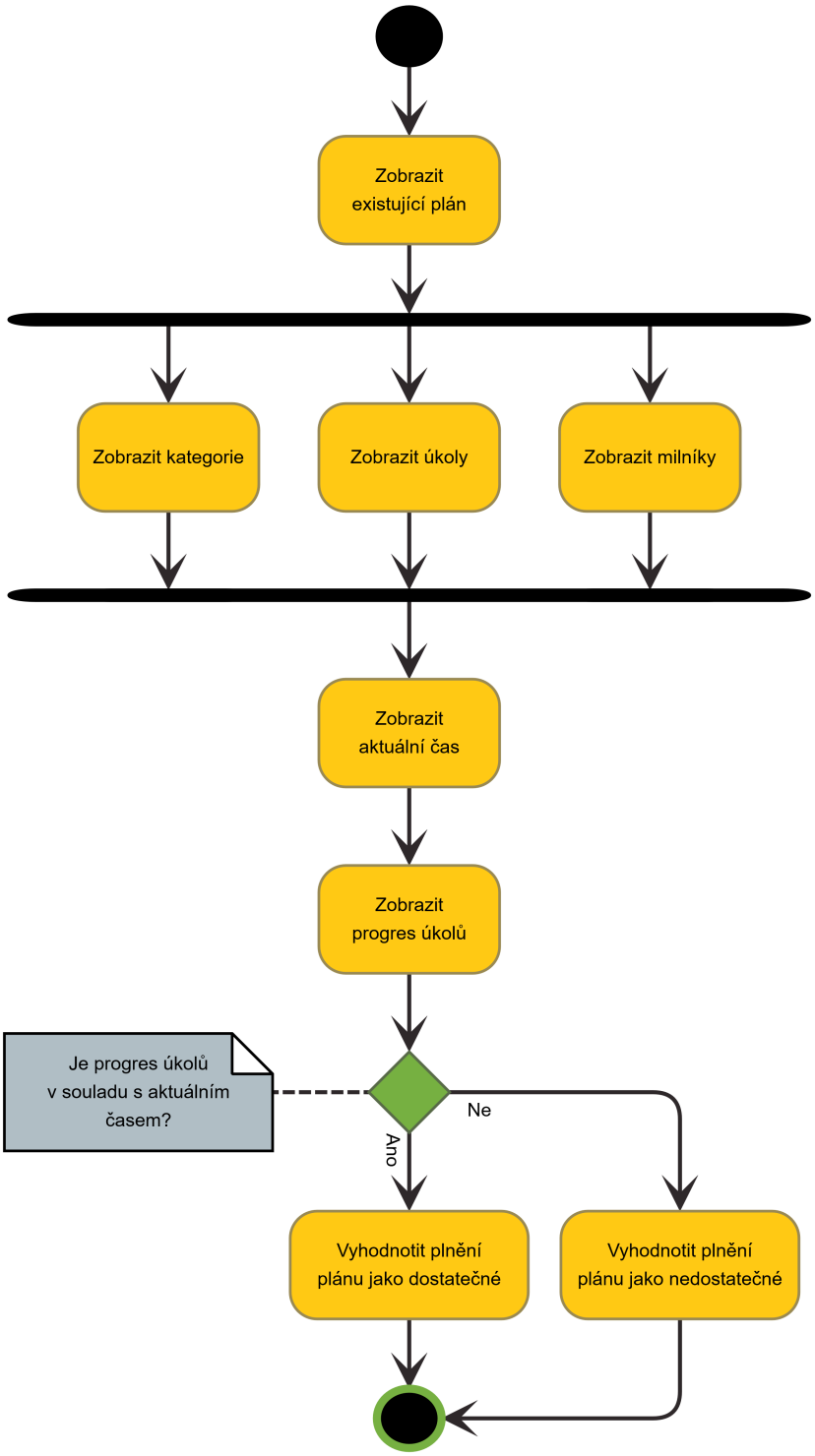


Obrázek 3.4: UML diagram aktivit optimalizace existujícího plánu - část 1 / 2



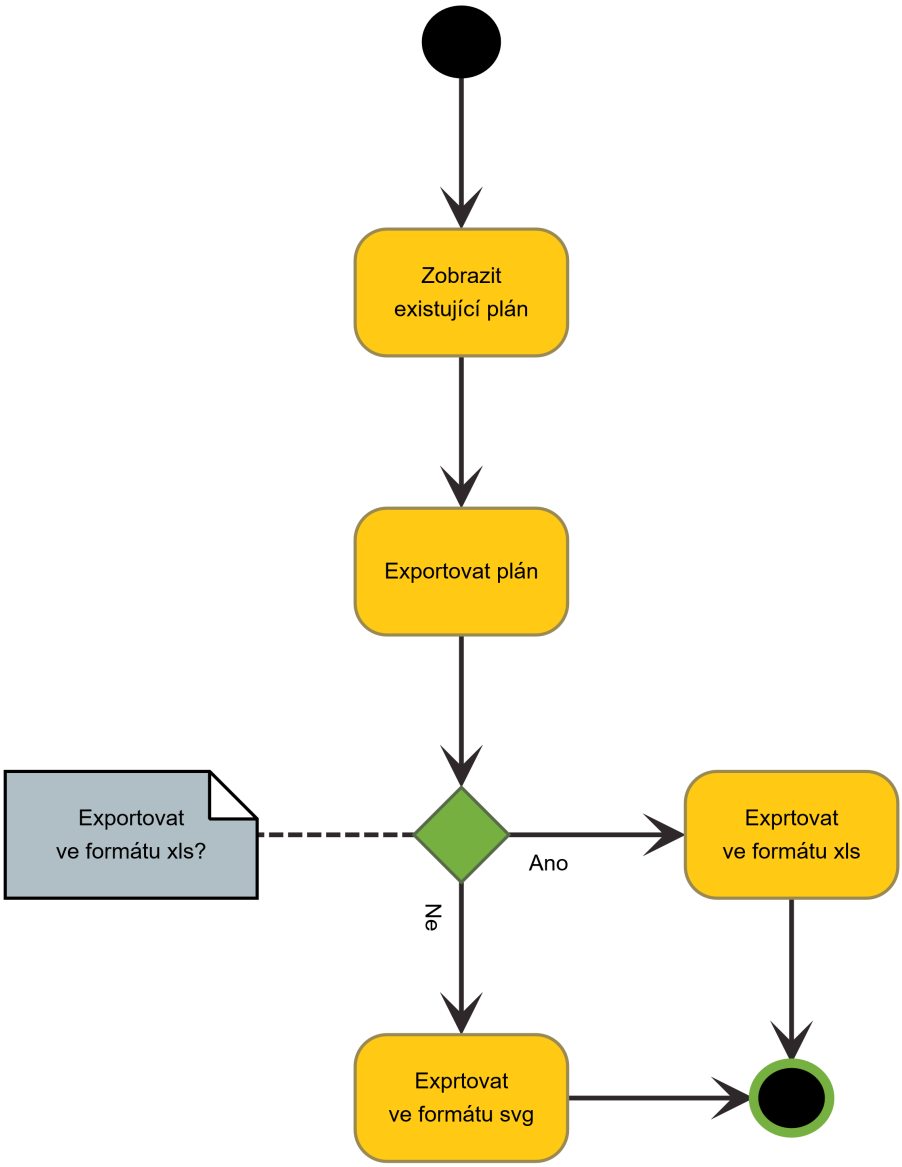
Obrázek 3.5: UML diagram aktivit optimalizace existujícího plánu - část 2 / 2

Zhodnocení plnění aktuálního plánu



Obrázek 3.6: UML diagram aktivit zhodnocení plnění aktuálního plánu

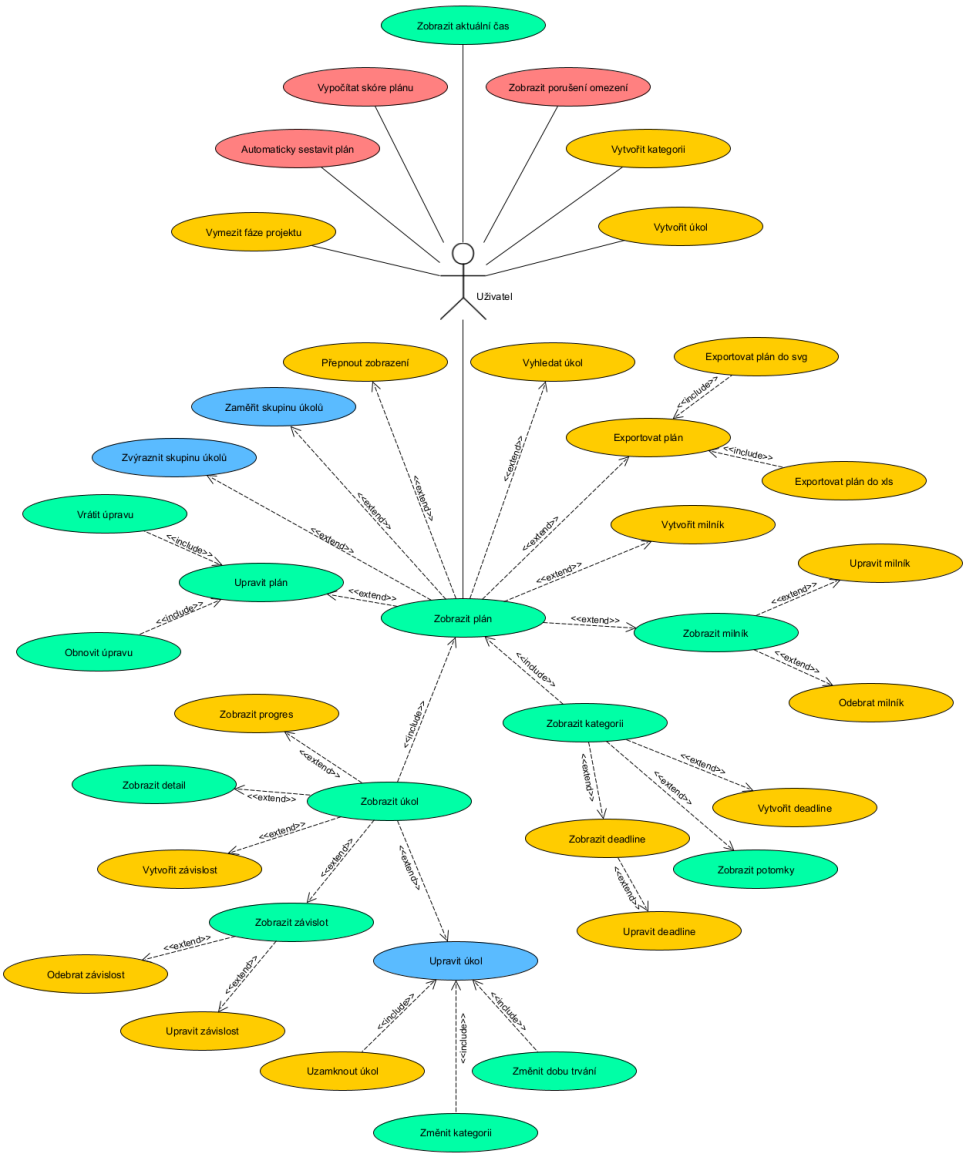
Export plánu



Obrázek 3.7: UML diagram aktivit exportu plánu

3.5 Případy užití

Případy užití pojednávají o interakci jednotlivých aktérů s navrhovanou plánovací komponentou. Jejich vztahy vůči případům užití jsou zobrazeny v diagramu užití na obrázku 3.8.



Obrázek 3.8: UML diagram případů užití plánovací komponenty

Případy užití na diagramu 3.8 jsou barevně vyznačeny následujícím způsobem:

- zelená - plánovací komponenta umí tento případ užití
- modrá - plánovací komponenta částečně umí tento případ užití
- oranžová - plánovací komponenta neumí tento případ užití
- červená - případ užití řeší aplikace implementující plánovací komponentu

3.6 Knihovny pro plánování

Vytvoření komponenty pro plánování je velmi rozsáhlé téma a bylo by velmi složité vytvořit celou plánovací komponentu od úplného začátku. Proto jsme se na začátku celého projektu rozhodli prozkoumat již existující reactové knihovny, umožňující nějaký způsob plánování, a následně vybrat nejvhodnější řešení, na základě kterého by se tato komponenta dala vyvíjet.

3.6.1 Kritéria pro knihovny

Pro zkvalitnění rešerše jsme si nejdříve definovali kritéria a funkčnosti, které jsou nutné, aby knihovna splňovala a uměla.
Veškerá tato kritéria a funkčnosti vycházejí z požadavků na komponentu.

- Kompatibilita knihovny s frameworkem React.
- Sestavení úkolů na časové ose
- Přiřazení úkolů do kategorií
- Základní úpravy úkolu
- Pohyb po časové ose
- Jednotlivé kategorie mohou obsahovat více úkolů
- Licence umožňuje komerční použití a zásah do zdrojového kódu

Dále jsme si definovali funkčnosti, které by byly velkým přínosem, pokud by je knihovna uměla.

- Hierarchické zobrazení kategorií
- Podpora pro milníky
- Závislosti úkolů
- Vytvářet nové úkoly
- Úprava více úkolů najednou
- Podpora zobrazení pro mobilní zařízení

3.6.2 Rešerše knihoven

Na začátku rešerše jsme určili, za pomocí jakých klíčových slov vyhledávat knihovny, které by bylo možné použít. Mezi klíčová slova se řadí:

- React planner
- React planning tool
- React maintenance planner
- React timeline
- React gantt chart
- React scheduler
- React calendar

Samotné vyhledávání knihoven jsem začal prohledáním stránky npmjs[5] za pomocí klíčových slov, kde jsem prověřil sto možných knihoven.

Dále jsem prošel padesát repositářů na GitHubu[6] vyhledaných pomocí klíčových slov, kdy se některé z repositářů kryly s nalezenými knihovnami na stránce npmjs.

Jako poslední zdroj pro vyhledávání možných knihoven jsem využil internetový vyhledávač Google[7], díky jehož výsledkům jsem prohledal dalších deset knihoven, které nejsou publikované na npmjs a nemají repositář na GitHubu.

Na závěr rešerše jsem vybral několik knihoven na základě podpory stanovených kritérií, které jsme si před rešerší definovali, s uvážením jejich důležitosti a osobního dojmu pro detailní prozkoumání a ověření jejich vhodnosti jako základ plánovací komponenty.

[5] https://www.npmjs.com/
[6] https://github.com/
[7] https://www.google.com/

3.6.3 Vybrané knihovny

Gantt-schedule-timeline-calendar

Gantt-schedule-timeline-calendar[8] je velmi rozsáhlá knihovna, vycházející z ganttova diagramu[9], umožňující zobrazení úkolů na časové ose v hierarchické struktuře a spoustu dalších funkčností.
Knihovna podporuje:

- Zobrazení úkolů v hierarchické struktuře
- Zobrazení kategorií
- Úprava více úkolů najednou
- Závislosti úkolů
- Podpora pro zobrazení na mobilních zařízeních
- Rozsáhlou přizpůsobitelnou konfiguraci

Licence
NEURONET Free Licence
Úpravy softwaru ze strany držitele jsou povoleny pouze změnou konfigurace popsané v dokumentaci softwaru, pokud nejsou v rozporu s licencí.[3] Pro komerční použití je licence placená.

Rozšíření
Jak vyplývá z licence, knihovnu je možné upravovat pouze konfiguracemi popsanými v dokumentaci. Zasahování do kódu knihovny je zakázané.

Závěr
Ačkoliv se tato knihovna ze všech prozkoumaných knihoven jeví jako nejlepší základ pro vývoj plánovací komponenty, není ji možné z licenčních důvodů použít.

[8] https://github.com/neuronetio/gantt-schedule-timeline-calendar
[9] https://www.gantt.com/

Gantt-task-react

Gantt-task-react[10] je knihovna vykreslující ganttův diagram umožňující pohybovat s úkoly a tvořit mezi nimi závislosti.
Knihovna podporuje:

- Prosté zobrazení úkolů
- Zobrazení kategorií
- Vykreslení progresu úkolu
- Úprava jednotlivých úkolů

Velkým nedostatkem knihovny je absence přiřazení více úkolů do jedné kategorie.

Licence
MIT
Jednoduchá, tolerantní licence s podmínkou zachování autorských práv a licenčních oznámení. Licencovaná díla, modifikace a větší díla mohou být distribuována za různých podmínek a bez zdrojového kódu.[4] Je možné použití pro komerční účely.

Rozšiřitelnost
Knihovna je psána v TypeScriptu[11], což by mohlo vést k obtížím při nutnosti zásahu do zdrojového kódu knihovny při zpracovávání požadavků na plánovací komponentu.
V základu knihovna vyvolává čtyři události, pomocí kterých je možné chování knihovny přizpůsobit

- onDateChange
- onTaskDelete
- onProgressChange
- onDoubleClick

Vyhodnocení
Knihovna je vhodná pro prosté a přehledné zobrazení na sebe navazujících událostí, nikoliv však jako základ pro vývoj plánovací komponenty.

[10] https://github.com/MaTeMaTuK/gantt-task-react
[11] https://www.typescriptlang.org/

React-big-scheduler

React-big-scheduler[12] je velmi rozsáhlá knihovna pro plánování úkolů připomínající diář.
Knihovna podporuje:

- Zobrazení úkolů v hierarchické struktuře
- Zobrazení kategorií
- Podpora pro zobrazení na mobilních zařízeních
- Rozsáhlé API[13] pro přizpůsobení a ovládání knihovny
- Podporuje mnoho možností zobrazení (například zobrazení podle zdrojů)

Velkou výhodou této knihovny je možnost vytváření nových úkolů. Naopak nevýhodou je vzhled vhodnější například spíše pro diář než plánovací komponentu.

Licence
MIT
Jednoduchá, tolerantní licence s podmínkou zachování autorských práv a licenčních oznámení. Licencovaná díla, modifikace a větší díla mohou být distribuována za různých podmínek a bez zdrojového kódu.[4] Je možné použití pro komerční účely.

Rozšiřitelnost
Knihovnu lze jednoduše modifikovat za pomoci opravdu rozsáhlého API. Zdrojový kód je psaný v JavaScriptu[14], tudíž by nemělo být problém v případě potřeby upravit zdrojový kód knihovny.

Vyhodnocení
Ačkoliv je knihovna vhodná spíše pro použití tvorby diáře, je možné tuto knihovnu využít jako základ pro vývoj plánovací komponenty.

[12] https://github.com/StephenChou1017/react-big-scheduler
[13] https://www.mulesoft.com/resources/api/what-is-an-api
[14] https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

React Google Charts - gantt

React Google Charts - gantt[15] je knihovna umožňující zobrazení různých variant diagramů. Mezi ně se řadí například ganttův diagram nebo timeline. Knihovna podporuje pouze zobrazení zobrazení kategorií, úkolů a závislostí mezi úkoly. Výhodou knihovny je podpora pro zobrazení progresu úkolu.

Licence
MIT
Jednoduchá, tolerantní licence s podmínkou zachování autorských práv a licenčních oznámení. Licencovaná díla, modifikace a větší díla mohou být distribuována za různých podmínek a bez zdrojového kódu.[4] Je možné použití pro komerční účely.

Rozšiřitelnost
Knihovna je psána v TypeScriptu, což by mohlo vést k obtížím při nutnosti zásahu do zdrojového kódu knihovny při zpracovávání požadavků na plánovací komponentu.

Vyhodnocení
Knihovna je vhodná pro prosté a přehledné zobrazení na sebe navazujících událostí, nikoliv však jako základ pro vývoj plánovací komponenty.

[15] https://github.com/rakannimer/react-google-charts

React Calendar Timeline

React Calendar Timeline[16] je velmi rozsáhlá knihovna sloužící pro zobrazení úkolů na časové ose.
Knihovna podporuje:

- Zobrazení úkolů v hierarchické struktuře (avšak pouze ve dvou úrovních)
- Zobrazení kategorií
- Úprava jednotlivých úkolů
- Částečná podpora pro zobrazení na mobilních zařízeních
- Zobrazení milníků
- Rozsáhlé API pro přizpůsobení a ovládání knihovny

Velkou výhodou knihovny je zobrazení jako časové osy a možnost přiblížení na specifický čas.

Licence
MIT
Jednoduchá, tolerantní licence s podmínkou zachování autorských práv a licenčních oznámení. Licencovaná díla, modifikace a větší díla mohou být distribuována za různých podmínek a bez zdrojového kódu.[4] Je možné použití pro komerční účely.

Rozšiřitelnost
Knihovnu lze jednoduše modifikovat za pomoci opravdu rozsáhlého API. Zdrojový kód je psaný v JavaScriptu, tudíž by nemělo být problém v případě potřeby upravit zdrojový kód knihovny.

Vyhodnocení
Knihovna splňuje veškerá nutná a většinu přínosných kritérií a funkčností, které jsme si definovali. Zároveň se knihovna vykresluje jako časová osa, což je pro plánovací komponentu vhodné. Přestože se knihovna nevykresluje tak hezky jako některé další knihovny, jeví se jako velmi vhodný základ pro vývoj plánovací komponenty.

3.6.4 Porovnání vybraných knihoven

Z vybraných knihoven jsme vyhodnotili knihovnu **gantt-schedule-timeline-calendar**3.6.3 jako jednoznačně nejvhodnější. Bohužel je pro vývoj této plánovací komponenty z licenčních důvodů nepoužitelná.

Proto jsme se rozhodli pro porovnání knihoven **React Calendar Timeline**3.6.3 a **React-big-scheduler**3.6.3, které lze na základě dat získaných z rešerše považovat za nejvhodnější licenčně dostupné knihovny coby základ pro vývoj plánovací komponenty.

[16] https://github.com/namespace-ee/react-calendar-timeline

React Calendar Timeline x React-big-scheduler

Obě knihovny disponují licencí MIT, lze je tedy za předpokladu dodržení licenčních podmínek využít jako základ pro vývoj plánovací komponenty.

Porovnání na základě požadavků plánovací komponenty
V následující tabulce 3.1 jsou na základně požadavků porovnány tyto dvě knihovny, kde je názorně vyznačeno, jaké požadavky která knihovna

- Splňuje
- Částečně splňuje (knihovna tento požadavek splňuje, ale ne v jeho plném rozsahu)
- Nesplňuje

Na základě tabulky3.1 pro porovnání požadavků, co knihovny umožňují, je zjevné, že obě knihovny mají srovnatelné zastoupení v dostupných funkčnostech. Knihovna **React Calendar Timeline** 3.6.3 však požadavkům vyhovuje nepatrně více.

	React Calendar Timeline	React-big-scheduler
FR1 Zobrazit úkol (M)	ANO	ANO
FR2 Závislost úkolu (M)	NE	NE
FR3 Zobrazení různých typů úkolů (S)	NE	NE
FR4 Logování práce na úkolu (C)	NE	NE
FR5 Progres tasku (C)	NE	NE
FR6 Podpora pro mobilní zařízení (M)	ANO	ANO
FR7 Zvýraznění skupiny úkolů (S)	NE	NE
FR8 Vyznačení aktuálního času (S)	ANO	NE
FR9 Zobrazení detailu úkolu (S)	NE	ČÁSTEČNĚ
FR10 Zvýraznění úkolů (S)	NE	NE
FR11 Manuální úprava úkolu	ČÁSTEČNĚ	ČÁSTEČNĚ
FR12 Zaměření na vybranou skupinu úkolů (C)	ANO	NE
FR13 Přepínání zobrazení (C)	NE	ANO
FR14 Export (C)	NE	NE
FR15 Podpora pro sestavení plánu pomocíexterní služby z historických dat	NE	NE
FR16 Plánování zdrojů (S)	ANO	ANO
FR17 Vyhledávání úkolů (W)	NE	NE
RQ19 Ovládání pohybu po časové ose (W)	ANO	ANO
FR18 Rozšiřitelnost panelu kategorií (C)	NE	NE
FR19 Změna stavu komponenty	NE	NE
FR20.1 Krok zpět (C)	NE	NE
FR20.2 Krok vpřed (C)	NE	NE
FR21 Omezení úkolu	ČÁSTEČNĚ	ČÁSTEČNĚ
FR22 Porušení omezení (S)	NE	NE
FR23 Skóre kvality plánu (W)	NE	NE
FR24 Prioritní úkoly (S)	NE	NE
FR25 Milníky (M)	ANO	NE
FR26 Deadline kategorie (S)	NE	NE
FR27 Fáze projektu (C)	NE	NE

Tabulka 3.1: Porovnání knihoven na základě požadavků

Slovní porovnání obecné funkčnosti
Obě knihovny jsou srovnatelné a rozlišují se od sebe především vzhledově a způsobem použití. Disponují srovnatelně rozsáhlým API pro zadání vstupních dat a následného ovládání celé komponenty. Všechny akce provedené v komponentách vyvolávají události, pomocí nichž lze jednoduše modifikovat chování komponenty.

Ačkoliv se knihovna **gantt-schedule-timeline-calendar**3.6.3 jeví vzhledově přehlednější a nabízí rozsáhlejší možnosti zobrazení, je však vhodnější spíše jako diář než plánovací komponenta, kdy přesné trvání události je zřetelné až po přepnutí na daný den.

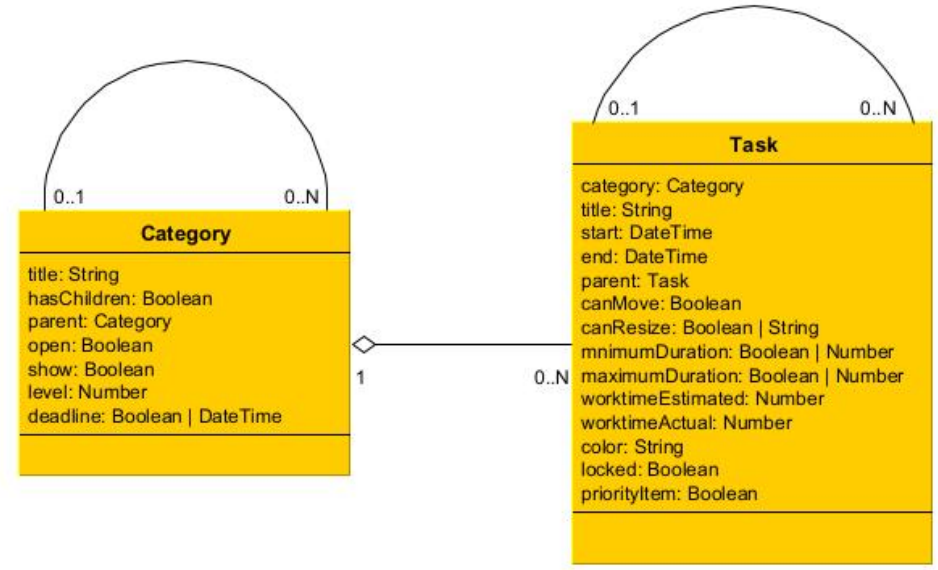
Knihovna **React Calendar Timeline**3.6.3 je vhodnější pro přesné plánování událostí v čase. Nejenže v základu nabízí nepatrně větší pokrytí požadavků, ale zároveň působí příjemnějším pocitem při ovládání. Přesné zobrazení trvání události (od kdy do kdy) a možnost přiblížení a oddálení napomáhá plynulosti a přesnosti práce při plánování.

Vyhodnocení
Obě porovnávané knihovny lze využít jako základ pro vývoj požadované plánovací komponenty a doimplementování zadefinovaných požadavků, kterými knihovny zatím nedisponují.

Na základě jejich porovnání se však knihovna **React Calendar Timeline**3.6.3 jeví jako vhodnější pro použití.

3.7 Datová struktura plánovací komponenty

Datová struktura plánovací komponenty řeší podobu dat, se kterými bude plánovací komponenty pracovat.



Obrázek 3.9: UML diagram datové struktury plánovací komponenty

3.7.1 Popis tříd datové struktury

Datová struktura plánovací komponenty je navržena včetně proměnných a jejich datových typů. Tyto proměnné vycházejí ze struktury diagramu3.9 a požadavků plánovací komponenty obsažených v kapitole3.3.

Třída Category

Třída Category představuje kategorie plánovací komponenty. Každá kategorie může odkazovat na nadřazenou kategorii. Společně tak tvoří hierarchickou strukturu plánu.

Title
Titulek kategorie, pod kterým se kategorie bude zobrazovat.

HasChildren
Obsahuje informaci, zda kategorie obsahuje potomky.

Parent
Odkazuje na nadrženou kategorii.

Open
Určuje, zda se mají zobrazovat potomci dané kategorie.

Show
Určuje, zda se má daná kategorie zobrazit.

Level
Stupeň zanoření v hierarchické struktuře kategorií.

Třída Task

Třída Task představuje úkoly obsažené v plánovací komponentě. Každý úkol patří do jedné kategorie a může odkazovat na nadřazený úkol.

Category
Obsahuje odkaz na kategorii, jejíž soustavy je daný úkol součástí.

Title
Titulek úkolu.

Start a end
Společně tvoří délku trvání, ve které by měl být daný úkol zpracovaný.

- Start - počáteční datum úkolu
- End - konečné datum úkolu

Parent
Odkazuje na nadřazený úkol.

CanMove
Omezuje možnost pohybu úkolu po časové ose a mezi kategoriemi.

CanResize
Omezuje možnost změny doby trvání.

MinimumDuration
Nastavuje minimální dobu trvání, pod kterou nelze daný úkol zkrátit.

MaximumDuration
Nastavuje maximální dobu trvání, nad kterou nelze daný úkol prodloužit.

WorktimeEstimation
Časový odhad doby potřebné pro vypracování úkolu.

WorktimeActual
Skutečný doba ztrávená vypracováním úkolu.

Color
Umožňuje změnu barvy úkolu.

Locked
Určuje, zda je daný úkol uzamčený.

PriorityItem
Určuje, zda by měl být úkol odbaven prioritně.

Kapitola 4
Implementace

Kapitola implementace popisuje proces vývoje a zpracování požadavků plánovací komponenty, použité technologie, použité knihovny a samotné nasazení aplikace. Veškeré odkazované ukázky naleznete v příloze B.

4.1 Techonlogie

V této sekci jsou popsány jednotlivé technologie použité pro vývoj, dokumentaci a buildování (sestavení) plánovací komponenty.

4.1.1 React

React je JavaScriptová knihovna umožňující snadno vytvářet interaktivní uživatelská rozhraní. Knihovna funguje na základě komponent, které mají vlastní stav a díky němu bude React efektivně aktualizovat dané komponenty, když se změní data. Díky tomu je možné vytvářet jednostránkové aplikace a při změně stavu překreslovat aktualizované části aplikace bez nutnosti znovu načtení celé stránky[11].

4.1.2 CSS

Kaskádové styly jsou jazyk, který se používá k popisu prezentace dokumentu napsaného v jazyce HTML nebo XML. CSS popisuje, jak mají být prvky zobrazeny na displejích, při tisku nebo jiných médiích[12].

4.1.3 Storybook

Storybook je open-source nástroj pro vytváření osamostatněných komponent uživatelského rozhraní a stránek za účelem usnadnění vývoje uživatelského rozhraní, testování a dokumentace[13].

4.1.4 Microbundle

Microbundle umožňuje vytvořit malý a optimalizovaný build knihovny[14]. Knihovnu microbundle jsme vybrali za účelem čitelnosti zdrojového kódu

(source map) knihovny v aplikaci, která bude knihovnu implementovat.

4.1.5 Netlify

Netlify je platforma pro hosting webových aplikací. Integrace libovolného verzovacího systému git, který Netlify podporuje, umožňuje automatizaci nasazení nových změn aplikace. Pro dosažení co nejvyšší rychlosti načtení webové aplikace nasazené na Netlify dostanou návštěvníci předem načtenou verzi z geograficky nejbližšího serveru[17].

4.1.6 GitHub

GitHub je služba k hostování úložišť git poskytující webové grafické rozhraní. Jedná se o největší komunitu programátorů na světě. GitHub pomáhá členům týmu spolupracovat na projektu z libovolného místa, zároveň usnadňuje spolupráci a umožňuje revidovat změny verzí vytvořených v dřívějším časovém okamžiku[19].

4.2 Použité knihovny

Celá plánovací komponenta vychází z knihovny React Calendar Timeline3.6.3, kterou jsme na základě analýzy knihoven vhodných pro plánování3.6 zvolili jako nejvhodnější.

Pro některé z požadavků již existují hotová řešení, vhodná, pro jejich splnění.

4.2.1 React-xarrows

Knihovna react-xarrows umožňuje tvořit propojení mezi jednotlivými prvky aplikace. Díky poskytnutému API knihovny lze jednotlivá propojení přizpůsobit dle potřeb uživatele[15].

Knihovna disponuje licencí MIT[4].

4.2.2 Moment

Moment je JavaScriptová knihovna sloužící pro pasování, neboli převod z textové podoby na nějaký specifický typ - v tomto případě na datum, validaci, manipulaci a formátování dat[16].

Knihovna disponuje licencí MIT[4].

4.3 Implementace požadavků

Sekce implementace požadavků popisuje proces a způsob, jakým byly jednotlivé požadavky implementovány. Z důvodu náročnosti projektu nebylo možné

v rámci této práce zpracovat veškeré požadavky. Proto jsou zde tedy popsány pouze požadavky, které byly implementovány.

4.3.1 Základní nastavení

Na základním nastavení spočívá implementace všech požadavků. Nejdříve bylo potřeba vytvořit projekt pro plánovací komponentu a následně do ni implementovat knihovnu React Calendar Timeline, ze které vychází celá komponenta. Implementaci knihovny do projektu lze provést dvěma způsoby.

- Nastavení závislosti na knihovnu
- Stažení zdrojového kódu a vložení do projektu

Z důvodu pravděpodobné potřeby zasáhnout přímo do kódu knihovny jsem vybral druhou možnost. Repositář knihovny jsem naklonoval do projektu, která se následně importuje do plánovací komponenty.

Na vstupu plánovací komponenta přijímá dva parametry.

- Kategorie
- Úkoly

Oba tyto parametry musí tvořit pole 0 až N objektů. Pokud u některých prvků polí přijatých na vstupu, ať už kategorií nebo úkolů, chybí některé z nepovinných parametrů, dosadí se do nich výchozí hodnoty. Z přijatých úkolů se vypočítá datum, od kterého se má časová osa knihovny zobrazovat a společně s kategoriemi a úkoly se nastaví do stavu komponenty tak, aby bylo možné s daty dále pracovat a aktivně je aktualizovat.

Ve funkci pro vykreslení komponenty se ze stavu společně se základním nastavením pro vzhled předávají kategorie a úkoly jako parametry do importované knihovny.

4.3.2 FR1 Zobrazit úkol

Požadavek zobrazit úkol určuje, že úkol má mít počáteční datum, konečné datum a kategorii, do které patří. Proto každý z přijatých úkolů musí obsahovat parametry:

- group - identifikátor kategorie, do které úkol patří
- start - počáteční datum
- end - konečné datum

Na základě těchto parametrů se každý úkol zobrazí ve správný čas na správném místě.

K vykreslení úkolů je použitá vlastní funkce sloužící k plné přizpůsobitelnosti vzhledu a chování jednotlivých úkolů.

4.3.3 FR2 Závislost úkolu

Pro zobrazení závislosti mezi úkoly jsem využil knihovnu react-xarrows 4.2.1. Závislost úkolu se určuje podle závislého úkolu, který odkazuje na jemu nadřazený úkol. Pro přidání závislosti stačí závislému úkolu nastavit parametr *parent* s hodnotou identifikátoru nadřazeného prvku.

Obrázek 4.1: Závislost úkolů

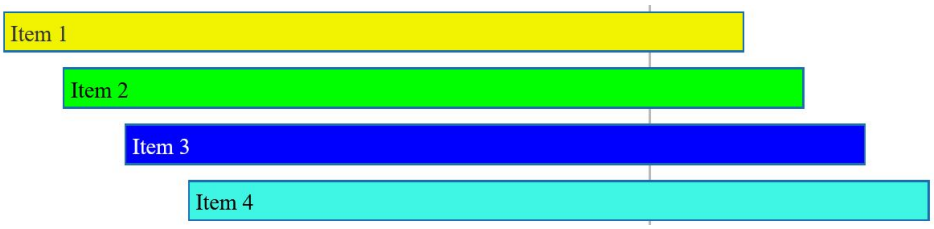
Na obrázku 4.1 je názorná ukázka, jak plánovací komponenta za pomoci knihovny react-xarrows vykreslí závislost dvou úkolů na jim nadřazený úkol.

4.3.4 FR3 Zobrazení různých typů úkolů

K odlišení různých typů je možné jednotlivým úkolům přidat dva parametry:

- color - barva textu titulku
- bgColor - barva pozadí úkolu

Tyto parametry na sobě nejsou nijak závislé a není nutné používat oba dva najednou.



Obrázek 4.2: Zobrazení různých typů úkolů

Na obrázku 4.2 jsou názorně zobrazené čtyři odlišené úkoly.

Toto řešení odlišení nemusí být konečné a v případě požadavku na jiný druh odlišení se může celé řešení změnit.

4.3.5 FR4 Logování práce na úkolu

Logy představují vypracované úkoly lidskými zdroji a chovají se jako úkoly.

FR4.1 Zobrazení logu

Zobrazení logů je vyjádřeno kategoriemi páté vrstvy hierarchické struktury představující lidské zdroje, kdo danou práci vykonal. Kdy a jakou dobu byla práce na úkolu vykonávána se zobrazí jako úkol v příslušné kategorii.

FR4.2 Editace logu

U jednotlivých logů lze upravit, kdo je prováděl, přesunutím do jiné kategorie představující lidský zdroj. Čas od kdy do kdy byla práce na úkolu vykonána lze měnit stejným způsobem jako doba trvání úkolů.

4.3.6 FR6 Hierarchická struktura

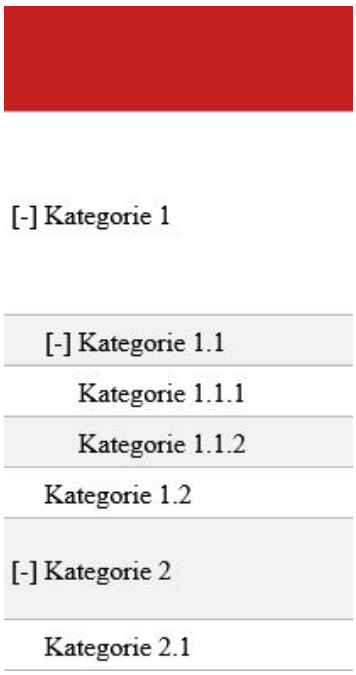
Implementace se dělí do dvou kroků - zobrazení a zvýraznění. Aby bylo možné pracovat s kategoriemi jako s hierarchickou strukturou, musí mezi nimi existovat závislost. Stejně jako u závislosti úkolů, jednotlivé kategorie pomocí parametru *parent* odkazují na jim nadřazenou kategorii. Na každou kategorii může odkazovat více kategorií najednou.

FR6.1 Zobrazení v hierarchické struktuře

Přijaté kategorie jsou seřazeny tak, aby na sebe navazovaly podle jejich závislosti (ukázka B.1.1). Seřazené kategorie je následně potřeba správně zobrazit v hierarchické struktuře. Z analýzy vstupních dat vychází, že plán má pět úrovní kategorií. Protože knihovna podporuje pouze dvě úrovně zobrazení hierarchické struktury, bylo potřeba doimplementovat zobrazení pro další tři. úrovně.

Kategorie, které se mají zobrazovat, přetvářím pomocí funkce *map* na nové pole prvků obsahujících styl zobrazení podle toho, jestli mají nějaké potomky a zda se mají potomci dané kategorie zobrazovat (ukázka B.1.2). Tato funkce vychází z původního dvouúrovňového zobrazení, ale navíc umožňuje libovolný počet úrovní zobrazení.

Kategoriím obsahujícím potomky zároveň nastavuji event listener (posluchač událostí) na kliknutí, aby bylo možné zobrazit nebo schovat potomky kategorie. Kliknutí na některou z těchto kategorií zpracuje funkce *toggleGroup*, která zobrazí nebo schová její potomky a aktualizuje stav komponenty (ukázka B.1.3).

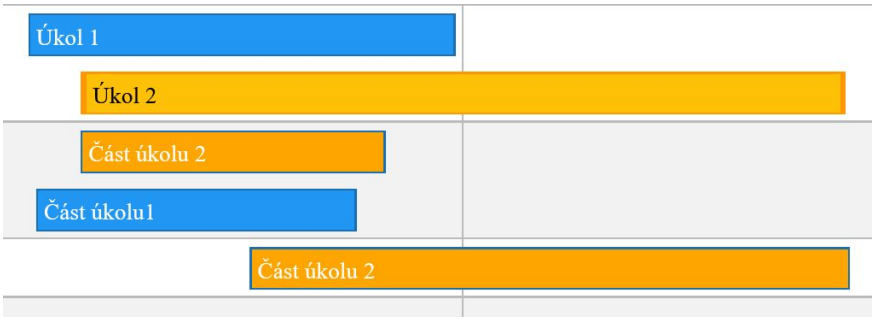


Obrázek 4.3: Hierarchické zobrazení kategorií

Obrázek 4.3 demonstruje názornou ukázku, jak vypadá rozbalená hierarchická struktura kategorií v plánovací komponentě.

FR6.2 Zvýraznění v hierarchické struktuře

Při selekci úkolu se provede rekurzivní algoritmus, který zvýrazní všechny potomky vybraného úkolu a po jeho dokončení se aktualizuje stav komponenty (ukázka B.1.4).
Barvu pozadí zvýrazněných potomků je možné upravit pomocí parametru *highlightBgColor*.



Obrázek 4.4: Zvýraznění úkolů v hierarchické struktuře

Obrázek 4.4 obsahuje ukázku, kde je vybraný Úkol 2 a společně s ním jsou zvýrazněny jeho části (potomci).

4.3.7 FR7 Zvýraznění skupiny úkolů

Pro tento požadavek je komponenta téměř připravena, samotným zvýrazněním určitých skupin již disponuje. Aby bylo možné požadavek plně dokončit, je potřeba do budoucna specifikovat, kdy a jaké skupiny se mají zvýraznit.

4.3.8 FR8 Vyznačení aktuálního času

K vyznačení aktuálního času jsem využil marker, kterým knihovna disponuje. Marker je nastavený, aby se každou vteřinu aktualizoval. Díky tomu je možné sledovat postup plánu v reálném čase.

Obrázek 4.5: Vyznačení aktuálního času

Z ukázky na obrázku 4.5 je zřetelné, že aktuální čas vyznačený v komponentě je pátek 13.05.2022 10:30.

4.3.9 FR9 Zobrazení detailu úkolu

Detail se zobrazí jako samostatná komponenta pod časovou osou na základě selekce úkolu. Tím předé do stavu komponenty informace o daném úkolu a kategorii, do které patří. Stav detailu se zároveň aktualizuje během manuální úpravy úkolu, tím poskytuje uživateli přesnou informaci o právě prováděné úpravě.

Implementace plánovací komponenty poskytuje možnost vložit vlastní komponentu pro zobrazení detailu úkolu pomocí parametru *popup*.

4.3.10 FR10 Zvýraznění úkolů

Úkoly jsou zvýrazněny při uživatelských akcích:

- Selekce úkolu
- Pohyb úkolu po časové ose

Jednotlivá zvýraznění je možné přizpůsobit zvlášť pro každý úkol přidáním parametrů:

- selectedColor - barva textu titulku vybraného úkolu
- selectedBgColor - barva pozadí vybraného úkolu
- draggingBgColor - barva pozadí úkolu při pohybu po časové ose

Komponenta je připravená pro zvýraznění dalších uživatelských akcí, které budou dále specifikovány nebo nově implementovány.

4.3.11 FR11 Manuální úprava úkolu

Manuální úprava úkolu vychází z funkcionality použité knihovny. Současně komponenta disponuje pouze úpravou jednoho úkolu a nelze mezi úkoly tvořit závislost.

4.3.12 FR12 Zaměření na vybranou skupinu úkolů

Komponenta již disponuje funkcí sloužící k zaměření určitých skupin úkolů. Zbývá pouze definovat nebo do implementovat akce a skupiny úkolů, které se mají při dané akci zaměřit.

4.3.13 FR15 Podpora pro sestavení plánu pomocí externí služby z historických dat

Ve výsledku se ukázalo, že tento požadavek nesouvisí přímo s plánovací komponentou, ale je součástí aplikace implementující tuto komponentu, která bude řešit propojení se serverovou částí aplikace.

Je možné, že do budoucna bude komponenta obsahovat prvek vyvolávající event pro sestavení plánu podle historických dat a předá ho aplikaci.

4.3.14 FR16 Plánování zdrojů

Do plánu je možné zahrnout libovolné zdroje. Komponenta však současně řeší pouze zobrazení dat a jejich následné úpravy, není tedy možné plánování zdrojů ve smyslu vytvářet nebo mazat zdroje.

4.3.15 FR19 Rozšiřitelnost panelu kategorií

Původně knihovna podporovala pouze možnost pevného nastavení šířky postranního panelu kategorií. Aby bylo možné tento panel rozšiřovat, respektive zužovat, bylo nutné zasáhnout přímo do kódu knihovny. Na pravou stranu panelu jsem přidal pruh sloužící jako úchop a parametr *handleSidebarResize* pro předání funkcí zpracovávající události vyvolané při změně šířky panelu. Samotné funkce zpracovávající události jsou řešeny v plánovací komponentě a na jejich základě se aktualizuje šířka postranního panelu ve stavu komponenty.

4.3.16 FR20 Změna stavu komponenty

Úpravy provedené v komponentě se ukládají do pole provedených změn. Kliknutí na tlačítko *undo* navrátí poslední provedenou úpravu, odebere ji z pole provedených změn a přidá ji do pole navrácených změn.

Kliknutí na tlačítko *redo* obnoví poslední navrácenou úpravu, odebere ji z pole navrácených změn a přidá ji do pole provedených změn.

Provedení libovolné nové úpravy odebere všechny akce z pole navrácených změn a nelze následně obnovit žádnou z navrácených změn.

4.3.17 FR21.1 Omezení úkolu - dle typu

FR21.1.1 Omezení změny doby trvání

Zde zahrnu i požadavek **FR21.1.4 Pevně daná doba trvání**, protože v implementaci spolu tyto dva požadavky přímo souvisí.

Možnosti změny doby trvání se nastavují parametrem *canResize*. Tento parametr může nabývat hodnot:

- "both"(výchozí hodnota) - umožní změnit dobu trvání od počátku i od konce
- "right"- umožní změnit dobu trvání pouze od konce
- "left"- umožní změnu doby trvání pouze od počátku
- false - úplně zamezí změnit dobu trvání - FR21.1.4

FR21.1.2 Omezení minimální / maximální doby trvání

K omezení minimální doby trvání slouží parametr *minimumDuration* a k omezení maximální doby trvání parametr *maximumDuration*. Oba parametry mají výchozí hodnotu *false* a netvoří žádné omezení.

Pro nastavení minimální nebo maximální doby trvání lze do příslušného parametru dosadit numerickou hodnotu rovnající se minimálnímu nebo maximálnímu počtu minut, jakého může úkol nabývat.

4.3.18 FR21.1.3 Zamezení pohybu

Zamezení pohybu určuje parametr *canMove* s výchozí hodnotou *false*. Pro zamezení pohybu mezi kategoriemi stačí do tohoto parametru dosadit hodnotu *true*, takový úkol je v komponentě vyznačen šrafováním.

4.3.19 FR25 Milníky

Milníky lze aktuálně pouze zobrazovat přidáním parametru *milestones* na vstupu plánovací komponenty. Tento parametr musí tvořit pole 0 až N objektů obsahujících parametry:

- date - datum, ve kterém se má milník zobrazit
- label - titulek milníku
- color - barva milníku

Parametr *date* je jako jediný povinný. Pokud nebudou předány parametry *label* a *color*, titulek bude prázdný a milník se zobrazí černou barvou.

4.3.20 NFR1 Podpora pro mobilní zařízení

Plánovací komponenta se dá aktuálně požívat na mobilních zařízeních, ale některé z funkcionalit nejsou plně optimalizované pro pohodlné používání.

4.3.21 NFR3 Automatizované nasazení změn na server

Automatizované nasazení změn funguje díky napojení vytvořeného hostingu Netlify[1] napojenému na GitHub repositář vytvořený pro plánovací komponentu. Hosting Netlify je nastavený, aby nasazoval změny ve všech branchích repositáře a jako produkční verzi použil branch *master*. Díky nasazení všech branchí je možné zachovat zobrazení produkční verze komponenty a zároveň zobrazit náhled na vývojové verze.

Příkaz pro build aplikace vytvoří statickou verzi storybooku s plánovací komponentou do adresáře *storybook-static*, který je nastavený pro publikování webové aplikace.

Při každém nasazení nové verze Netlify porovná novou verzi se stávající a určí, které soubory jsou upraveny a je třeba nahrát. Na veřejnou adresu webu se nedostanou žádné změny, dokud nejsou všechny změny nahrány. To znamená, že nasazení jsou atomická a webová aplikace se během nasazování nových změn nikdy nenachází v nekonzistentním stavu[18].

Obrázek 4.6: Stav nasazení změn

Soubor *README.md* v GitHub repositáři obsahuje odznak zobrazující stav nasazení nejnovější verze komponenty. Odznak může nabývat stavů zobrazených na obrázku 4.6.

[1] https://react-maintenance-planner.netlify.app

4.3.22 NFR4 Build znovu použitelné knihovny

Plánovací komponentu lze buildovat jako znovu použitelnou knihovnu spuštěním příkazu *build:lib*. Build je tvořen pomocí knihovny **microbundle** 4.1.4 s parametry zajišťujícími kompatibilitu knihovny se syntaxí JSX.

4.4 Implementace storybooku

Pomocí storybooku jsou popsány tři různé příběhy implementující plánovací komponentu lišící se obsaženými vstupními daty (ukázka B.2).

- Výchozí zobrazení
- Vložení vlastní komponenty pro zobrazení detailu úkolu
- Zobrazení milníků

Vstupní data jednotlivých zobrazení jsou popsaná v záložce *Docs*. Data lze upravovat, přidat či smazat. Při vkládání vlastních vstupů je potřeba dodržet datové typy definované ve sloupci *description*, v opačném případě nebude komponenta fungovat správně.

Kapitola 5
Evaluace

V této kapitole je popsané uživatelské testování plánovací komponenty na zvolených scénářích včetně jeho vyhodnocení5.1 a také zhodnocení použití knihovny v produkčním prostředí5.2.

5.1 Uživatelské testování

Tato sekce se věnuje uživatelskému testování plánovací komponenty. Uživatelské testování je proces, při kterém jsou funkce aplikace prováděny skutečnými uživateli, kteří provádějí konkrétní úkoly v reálných podmínkách. Účelem tohoto procesu je vyhodnocení použitelnosti dané aplikace a určit, zda je aplikace vhodná ke spuštění pro skutečné uživatele. Pro dosažení relevantních výsledků by testující osoby měly zapadat do cílové skupiny uživatelů, neměly být příliš usměrňovány a mělo by jim být umožněno přirozeně interagovat s danou aplikací, aby se zjistilo, zda je aplikace dostatečně intuitivní a pohodlná pro uživatele, kteří s ní nejsou příliš obeznámeni[8].

5.1.1 Scénáře testování

Testovací scénáře jsou navrženy tak, aby pokryly co největší množinu případů užití vhodných pro uživatelské testování. Jednotlivé scénáře jsou obsaženy v příloze C.

5.1.2 Doplňující otázky k testování

1. Vaše poznámky k prvnímu scénáři
2. Vaše poznámky k druhému scénáři
3. Vaše poznámky k třetímu scénáři
4. Bylo pro vás ovládání komponenty intuitivní a pohodlné?
5. Je podle vás komponenta přehledná?

6. Bylo pro vás náročné některý ze scénářů splnit?
7. Vyskytly se nějaké komplikace při plnění některého ze scénářů?
8. Máte nápad, jak tuto komponentu vylepšit?

5.1.3 Testující osoby

V této podkapitole je obsažena zpětná vazba o testujících osob.

Tester 1

1. Vše bylo jasné.
2. Chvíli mi trvalo kym jsem nasel jak ukol zkratit. Je to proto, ze pro zkraceni musi uzivatel nejprve vybrat dany ukol, to mi nepride vubec stastne (IMHO, nemelo by to tak byt).
3. Trochu priliš komplikovaně popsaný scénař, jinak to šlo.
4. Bylo těžke se trafit do konkretního času (trafil jsem napr. 07:29 misto 07:30) když jsem zkracoval dobu ukolu, čekal bych, že se čas podle rozlíšení plánu bude zarovnávat na hodiny, pulhodiny, 15 min a pod.
5. Ano, kromě toho že šrafovani delá text úkolu nečítelný (je zrejme potreba zvolit jine barvy).
6. Ne, krome trefeni konkretniho casu viz. vyse.
7. Ne.
8. Zvolil bych lepe barvy, aby byly jmena ukolu citelne.

Tester 2

1. Maybe it could be written somewhere explicitly “task: xxx”, but otherwise no problem to complete the task.
2. All clear and working.
3. Instructions are a bit confusing, we need to refer to previous steps and remember names of all tasks.
4. Yes, maybe it would be better to be able to enter a precise date and time with an input field rather than moving the task precisely.
5. Yes
6. Just because the instructions were a bit confusing, yes. But otherwise it was easy to accomplish once understood.
7. No.
8. As mentioned before, maybe be able to enter with input field date and time to move a task rather than moving it.

Tester 3

1. Chvíli mi trvalo, než jsem pochopil, jak komponenta funguje, ale poté bylo snadné scénář 1 splnit.

2. Scénář 2a bylo snadné provést. Scénář 2b je poměrně těžké provést precizně. Postrádám možnost manuálního vyplnění času. Uvítal bych možnost napsat konkrétní dobu trvání úkonu v hodinách a konkrétního času, kdy má začít. Scénář 2c je obdobný jako 2b. Je poměrně uživatelsky nepřívětivé trefit se pouhým posunováním úkonu přesně o půl hodiny dříve. Při přílišném přiblížení za účelem přesnějšího zadání času pro jednotlivé údržbové úkony uživatel ztrácí přehled o kontextu ostatních úkonů.

3. Na menším monitoru je trochu těžší sledovat při posouvání úkonu čas, ve kterém se úkon provede. Zejména pokud je údržbový úkon posouván "nahoru".

4. Chvíli mi trvalo se zorientovat se v tom, jak komponenta funguje, ale poté to bylo snadné a jasné. Nepohodlné bylo sledování času při práci na menším monitoru. Ocenil bych možnost přímého vkládání času, jak dlouho má daný úkon údržby trvat a kdy začne.

5. V zásadě ano.

6. Přesné nastavení času začátku a konce úkonů.

7. Obtížné nastavení přesného času.

8. Uvítal bych možnost přímo vkládat čas pro dobu trvání úkonu a znát údaj o celkovém počtu pracovních hodin, jak dlouho má úkon trvat.

Tester 4

1. Splněno bez potřebné nápovědy.

2. Bez nápovědy jsem intuitivně nepřišel na možnost změny doby tasku (absence myši, pouze touch pad). Zároveň se mi na první pokus podařilo item 4 deletovat a již jsem nebyl schopen akci vrátit, ani přes tlačítko undo. Pravděpodobně je to tím, že jsem provedl mezitím jinou akci a tlačítko nefunguje jako krok zpět? Nastavení doby tasku bylo pomocí touch padu velice komplikované.

3. Přesouvání tasků z jedné skupiny do jiné funguje správně, taktéž oceňuji funkce označení obsahu podkategorií a celkové označení všech položek. Vzhledem k počtu tasků v obsáhlých revizích letadel, nevím jak plán bude přehledný a funkční. To by bylo nutné testovat na větší sadě tasků.

4. Po pochopení systému bez problémů. Zároveň si ale umím představit pro delší užívání více uživatelsky přívětivé prostředí.

5. Možná je to uživatelským nastavením, ale v určitých momentech je obtížné přesouvat task a zároveň se posouvat po časové ose.

6. Jak je popsáno, než uživatel pochopí systém a zorientuje se, tedy nejdelší bylo pochopení a učení se u scenáře 1, následně bez problémů.

7. Komplikace způsobeny spíše podmínkami uživateli.

8. Jakožto uživatel si umím představit zobrazení většího množství detailu o tasku, manuální zadávání dat k taskům nebo například vytvoření si vlastního tasku v plánu revize.

5.1.4 Vyhodnocení

Nálezy, které vyplývají z odpovědí respondentů, bych klasifikoval jako nedostatky. Pouze v jednom případě se jedná o chybu systému.

Při plnění druhého scénáře se testerovi z neznámého důvodu podařilo smazat jeden úkol a akce nešla vrátit zpět. Je potřeba prozkoumat, jak k takové události došlo a vymyslet řešení.

Prvním a největším nedostatkem se jeví nastavení konkrétní času, protože komponenta plánuje na minutu přesně a při velkém přiblížení za účelem přesnějšího zadání času uživatel ztrácí přehled o kontextu ostatních úkoů. Jako řešení navrhuji upravovat dobu, o jakou se čas změní, podle úrovně přiblížení. Jak bylo navrženo, bylo by vhodné přidat možnost zadat přesný čas do nějakého vstupního pole. Konkrétně navrhuji implementovat nějakou komponentu nabízející volbu přesného data a času, od kdy do kdy má vybraný úkol trvat.

Druhým nedostatkem je možnost prodloužit nebo zkrátit dobu trvání až po vybrání úkolu. Řešení vyplývá z popisu nedostatku.

Třetím nedostatkem je horší orientace v čase při zobrazení na menším monitoru. Řešením je optimalizace responzivního[1] zobrazení.

Čtvrtým nedostatkem je komplikované nastavení času úkolu na touchpadu. Řešení je identické jako u prvního nedostatku.

Pátým nedostatkem je nedostatečné množství obsažené v detailu úkolu. Řešení je potřeba konzultovat s doménovými odborníky, co všechno je potřeba v detailu zobrazit.

Posledním nedostatkem je nečitelnost úkolů vyznačených šrafováním. Navrhnuté řešení je zvolení jiných barev, aby byla jména úkolů lépe čitelná. Tento nedostatek je potřeba probrat s doménovými odborníky a sestavit paletu

[1] https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/

dobře čitelných barev, které budou vyhovovat jejich požadavkům na různá zobrazení a zvýraznění.

Během vyhodnocení uživatelského testování jsem při návrhu řešení prvního nedostatku dospěl k závěru, že k možnosti vybrat, od kdy do kdy má úkol trvat, by bylo dobré zároveň přidat možnost výběru kategorie, do které je úkol zařazen.

5.1.5 Závěr

Během testování došlo k nálezu jedné fatální chyby, kdy se testerovi z neznámého důvodu podařilo omylem smazat jeden úkol, a šesti nedostatků, které nemají přímo na funkcionalitu komponenty vliv. Jedná se o nedostatky, které nejsou uživatelsky přívětivé při procesu plánování. Největším nedostatkem u všem respondentů bylo naplánování přesného času úkolu.

Plánovací komponenty je přehledná a její ovládání je intuitivní, ale je potřeba optimalizovat uživatelskou přívětivost.

5.2 Použití knihovny v produkčním prostředí

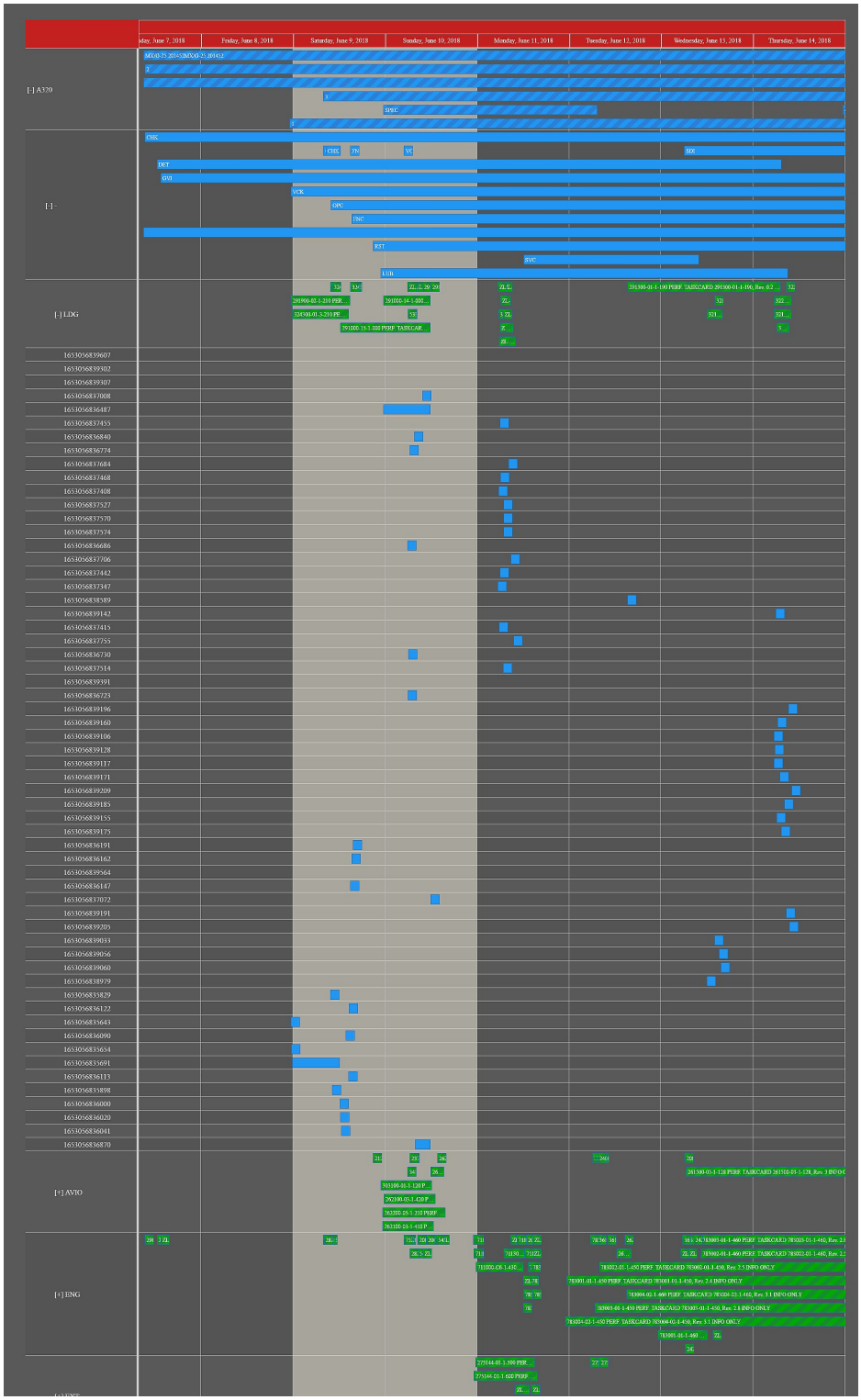
Od 19.5.2022 je knihovna použita v rámci projektu Zvýšení efektivity plánování a provádění údržby dopravních letadel[2], patřící do programu DOPRAVA 2020+ Technologické agentury ČR.

Pomocí knihovny byly úspěšně zobrazeny aktuální revize letadel CSAT. K dispozici je 76 různých plánů. Každý plán obsahuje průměrně 426 úkolů v pěti úrovních detailu včetně výkazů provedené práce, které jsou zobrazeny na páté úrovni. Největší plán obsahuje 1393 úkolů.

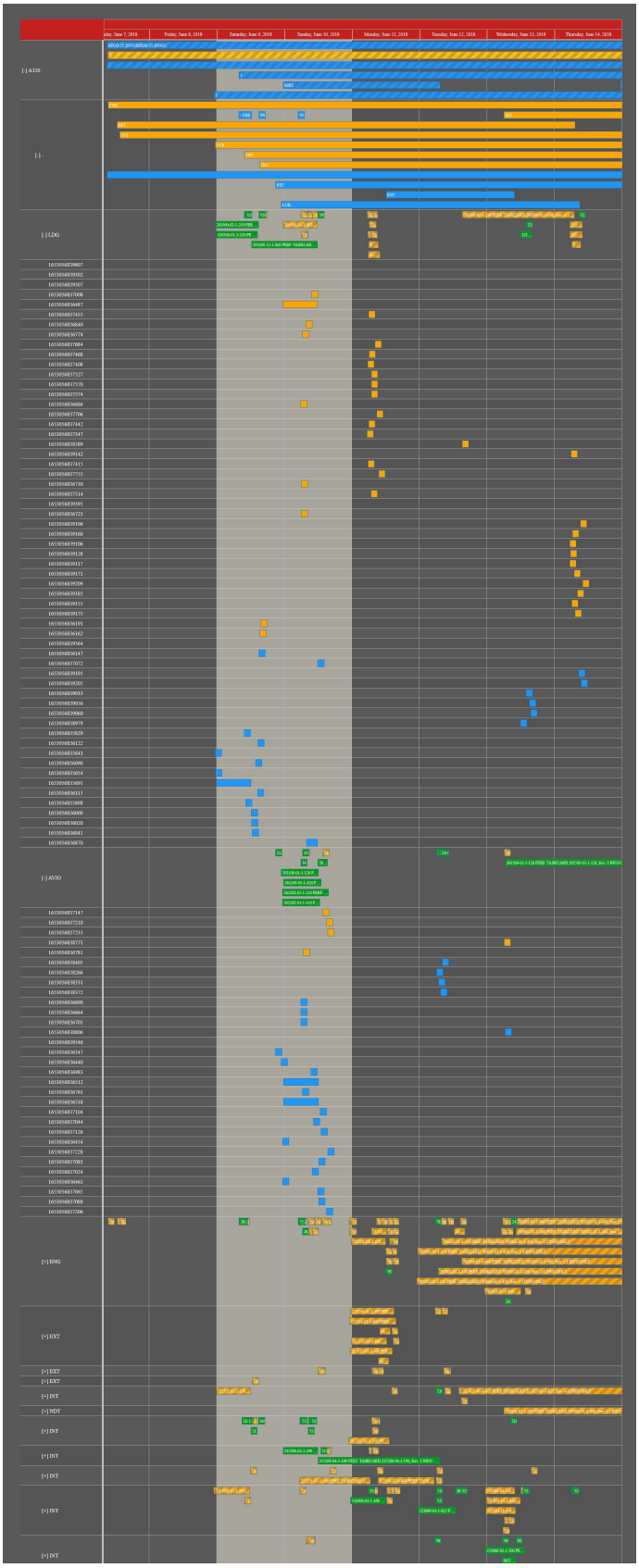
Knihovna byla úspěšně otestována na reálných datech a funguje rychle.

[2]https://starfos.tacr.cz/cs/project/CK01000204?query_code=2jiiaackhjiq

5.2.1 Ukázky použití



Obrázek 5.1: Použití knihovny v produkčním prostředí



Obrázek 5.2: Zvýraznění v hierarchické struktuře

Kapitola 6

Závěr

Výstupem práce je komponenta podporující plánování údržby letadel v libovolném počtu různých úrovní detailu s podporou zobrazení a úprav vykázané práce. Komponenta byla otestována čtyřmi doménovými odborníky a prošla testem velké množiny úkolů na vstupu. Komponenta není nutně specifická pro plánování údržby letadel a je možné ji použít i pro obecné plánování, avšak je vhodná zejména pro plánování údržby nebo plánů, ve kterých je vhodné rozdělit zdroje do hierarchických kategorií.

Pro distribuci byl využit standardní mechanizmus tvorby balíčku v JavaScriptu a jeho korektnost byla ověřena použitím v jiném projektu, kde se tento balíček používá jako závislá knihovna.

V rámci bakalářské práce se nám podařilo implementovat velkou část požadavků a odhalit několik nedostatků během uživatelského testování, například náročné nastavení konkrétního času nebo možnost změnit čas úkolu až po jeho vybrání. Zároveň byla komponenta použitá v produkční prostředí na reálných datech a prošla výkonnostním testem na reálných datech s velkou množinou úkolů.

Do stavu, kdy by byla komponenta schopna spuštění v ostrém provozu, je pořád daleko. Rád bych se i na dále podílel na tomto projektu a byl bych rád, kdyby se nám společnými silami podařilo dostat komponentu do stavu, ve kterém by ji mohli převzít doménoví odborníci a začít používat pro plánování skutečné údržby letadel.

Do budoucna je potřeba doimplementovat zbylé a případné nové požadavky, přizpůsobit a odladit grafické zobrazení a optimalizovat ovládání aplikace na základě chyb a nedostatků objevených při uživatelském testování. Například je potřeba naimplementovat vytváření, upravování a odebírání závislostí a milníků, tím se pokryjí veškeré aktuální "Must have"požadavky. Z objevených nedostatků při testování je rozhodně vhodné zaměřit se na uživatelsky přívětivější plánování konkrétního času.

Příloha A

Literatura

[1] Kinnison, Harry A. Aviation maintenance management. McGraw-Hill Education, 2013. [Citováno: 19.5.2022].

[2] Gupta, Payal, Massoud Bazargan, and R. N. McGrath. "Simulation model for aircraft line maintenance planning."Annual Reliability and Maintainability Symposium, 2003.. IEEE, 2003. [Citováno: 19.5.2022].

[3] NEURONET - Rafał Pośpiech. NEURONET Free / Trial License Terms. https://github.com/neuronetio/gantt-schedule-timeline-calendar/blob/master/LICENSE. [Citováno: 03.5.2022].

[4] Jared Palmer. MIT License. https://oss.ninja/mit/jaredpalmer/. [Citováno: 03.5.2022].

[5] WOOLF, Bobby. The abstract class pattern. Pattern Languages of Program Design, 1997, 4. https://www.javaspecialists.eu/courses/dpc/archive/AbstractClass-Woolf.pdf. [Citováno: 04.05.2022].

[6] IBM. Association relationships. https://www.ibm.com/docs/en/rsm/7.5.0?topic=ricd-association-relationships. [Citováno: 04.05.2022].

[7] IBM. Composition association relationships. https://www.ibm.com/docs/en/rsm/7.5.0?topic=diagrams-composition-association-relationships. [Citováno: 04.05.2022].

[8] What is user testing. https://www.omniconvert.com/what-is/user-testing/. [Citováno: 05.05.2022].

[9] IBM. Aggregation relationships. https://www.ibm.com/docs/en/rsm/7.5.0?topic=diagrams-aggregation-relationships. [Citováno: 08.05.2022]

[10] Alyssa Walker. UML Relationships Types: Association, Dependency, Generalization. https://www.guru99.com/uml-relationships-with-example.html. [Citováno: 08.05.2022].

[11] React. https://reactjs.org/. [Citováno: 15.05.2022].

[12] CSS: Cascading Style Sheets. https://developer.mozilla.org/en-US/docs/Web/CSS. [Citováno: 15.05.2022].

[13] Storybook. https://storybook.js.org/. [Citováno: 15.05.2022].

[14] Microbundle. https://www.npmjs.com/package/microbundle. [Citováno: 15.05.2022].

[15] react-xarrows. https://github.com/Eliav2/react-xarrows. [Citováno: 15.05.2022].

[16] Moment.js. https://github.com/moment/moment. [Citováno: 15.05.2022].

[17] Netlify. https://devopedia.org/netlify. [Citováno: 18.05.2022].

[18] Site deploys overview. https://docs.netlify.com/site-deploys/overview. [Citováno: 18.05.2022].

[19] What is GitHub And How To Use It?. https://www.simplilearn.com/tutorials/git-tutorial/what-is-github. [Citováno: 18.05.2022].

[20] MOSCOW PRIORITISATION. https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation. [Citováno: 18.05.2022].

Příloha B

Ukázky kódu

B.1 Zobrazení hierarchické struktury

B.1.1 Seřazení kategorií podle závislostí

```
groups = groups.sort((a, b) => a.level - b.level)
  .reduce((accumulator, currentValue) => {
    let item = accumulator.find(x => x.id === currentValue.parent)
    let index = accumulator.indexOf(item)
    const count = accumulator.filter(x => x.parent ===
  currentValue.parent).length
    index = index !== -1 ? index + count + 1 : accumulator.length
    accumulator.splice(index, 0, currentValue)
    return accumulator
  }, [])
```

B.1.2 Nastavení chování kategorií

```
const newGroups = groups.filter((g) => g.show).map((group) => {
    return Object.assign({}, group, {
      title: group.hasChildren ? (
        <div onClick={() => this.toggleGroup(group.id)}
             style={{
               cursor: 'pointer',
               paddingLeft: group.level * 20
             }}>
          {group.open ? '[-]' : '[+]'} {group.title}
        </div>
      ) : (
        <div style={{paddingLeft: group.level * 20}}>
          {group.title}
        </div>
      )
    })
  })
```

B.1.3 Ovládání kategorií

```
toggleGroup = (id) => {
  let {groups} = this.state

  const group = groups.find(g => g.id === id)
  group.open = !group.open

  groups.filter(g => g.parent === id).forEach((g) => {
    g.show = group.open

    if (!group.open) {
      g.open = false
      groups = this.closeChildren(groups, g.id)
    }
  })

  this.setState({
    groups: groups
  })
}

closeChildren = (groups, id) => {
  groups.filter(g => g.parent === id).forEach((g) => {
    g.show = false

    if (g.hasChildren) {
      g.open = false
      groups = this.closeChildren(groups, g.id)
    }
  })

  return groups
}
```

B.1.4 Zvýraznění kategorií

```
highlightChildren = (item) => {
  const items = this.state.items.filter(i => i.parent === item.id)
  for (const item of items) {
    item.highlight = true
    this.highlightChildren(item)
  }
}
```

B.2 Storybook

```
import React from 'react'
import PlanningTool from '../components/PlanningTool'
import moment from 'moment'

export default {
  component: PlanningTool,
  title: 'Planning tool',
```

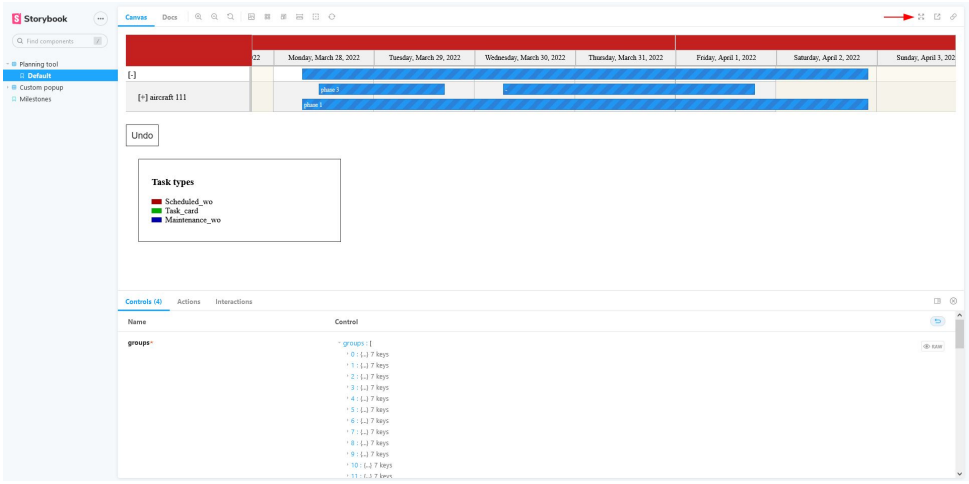
```
  parameters: {
    actions: {
      handles: ['click .rct-item'],
    },
  },
}

const groups = [
  {
    "id": 0,
    "title": "",
    "hasChildren": true,
    "parent": null,
    "open": true,
    "show": true,
    "level": 0
  }
]
const items = [
  {
    "id": 1,
    "group": 0,
    "title": "phase 3",
    "start": moment("2022-03-28T09:00:00.000Z"),
    "end": moment("2022-03-29T15:00:00.000Z"),
    "parent": 1,
    "className": "item",
    "bgColor": "#2196F3",
    "color": "#fff",
    "selectedBgColor": "#FFC107",
    "selectedColor": "#000",
    "draggingBgColor": "#f00",
    "highlightBgColor": "#FFA500",
    "highlight": false,
    "canMove": true,
    "canResize": "both",
    "minimumDuration": false
  },
]

const Template = (args) => <PlanningTool {...args} />

export const _default = Template.bind({})
_default.args = {
  groups: groups,
  items: items,
}
```

Příloha C

Testovací scénáře

C.1 Prerekvizita

1. Před každým scénářem si znovu načtěte plánovací komponentu[1].
2. Doporučení: po načtení plánovací komponenty si pomocí tlačítka "Go full screen[F]"přepněte komponentu do režimu zobrazení na celé obrazovce viz obrázek C.1.



Obrázek C.1: Přepnutí režimu zobrazení na celé obrazovce

Scénář 1

Zaměření

Tento scénář testuje schopnost uživatele zorientovat se v hierarchické struktuře kategorií. Cílem scénáře je zorientování se v hierarchické struktuře a správné vyhledání požadované kategorie.

[1] https://react-maintenance-planner.netlify.app/

Zadání

1. V levém panelu zobrazujícím kategorie v hierarchické struktuře najděte kategorii "AVIO", která je součástí podkategorie "Cockpit".
2. Zjistěte název prvního úkolu v této kategorii.

Nápověda

- Kategorie ve stromové struktuře lze sbalit, respektive rozbalit kliknutím na kategorii.
- Detail úkolu se nachází pod časovou osou.

Scénář 2

Zaměření

Tento scénář testuje interakce uživatele s různými omezeními úkolu. Cílem scénáře je zjistit, jaký úkol má jaké vlastnosti, vybrat správný úkol pro provedení jednotlivých částí scénáře a ověřit intuitivnost různých akcí s úkoly.

Zadání

1. Časovou osu posuňte do data 05.04.2022, kde naleznete čtyři různé úkoly (Item 1, Item 2, Item 3, Item 4).
2. Každý z úkolů má rozdílné vlastnosti od ostatních. Proveďte s každým z úkolů právě jednu z následující akcí a zapište, se kterým úkolem se vám podařilo akci splnit:
   a. Přesuňte úkol do jiné kategorie
   b. Zkraťte dobu trvání úkolu o jednu hodinu tak, aby se čas, ve který úkol začíná, nezměnil
   c. Prodlužte dobu trvání úkolu tak, aby úkol začínal o půl hodiny dříve

Nápověda

- Posun časové osy vpřed, respektive zpět, provedete zmáčknutím a následným podržením levého tlačítka myši a následným pohybem vlevo, respektive vpravo.
- Pro pohodlnější úpravu doby trvání můžete využít zoom pomocí podržení klávesy Control a kolečka myši. V případě touchpadu můžete využít gesto přiblížit, respektive oddálit.
- V případě provedení nechtěné akce můžete využít tlačítko "Undo"pro navrácení provedené akce.

Scénář 3

Zaměření

Tento scénář testuje, jak si uživatel dokáže poradit s několika po sobě jdoucími akcemi.

Zadání

1. Z kategorie "Interior Body"vyberte úkol "Functional Check".
2. Vybraný úkol přesuňte do kategorie "Fuselage"tak, aby začínal 03.04.2022 v 07:30.
3. Přesunutému úkolu z kroku č.2 prodlužte dobu trvání o dvě a půl hodiny.
4. Z kategorie "AVIO", která je podkategorií kategorie "Interior Body", přesuňte úkol "341300-07-1-210 AIR DATA"do kategorie "AVIO", která je podkategorií kategorie "Fuselage", tak, aby začátek tohoto úkolu korespondoval se začátkem přesunutého úkolu v kroku č.2.
5. Z kategorie "AVIO", která je podkategorií kategorie "Cockpit", přesuňte úkol do kategorie "AVIO", která je podkategorií kategorie "Fuselage", tak, aby začínal 03.04.2022 ve 20:00.
6. Úkolu přesunutému v kroku č.5 prodlužte dobu trvání tak, aby jeho konec korespondoval s koncem úkolu přesunutém v kroku č.2.
7. Výběrem úkolu "phase 1"v kategorii "aircraft 111"se zároveň zvýrazní veškeré úkoly, které jsou součástí tohoto úkolu.
8. Upravte tomuto úkolu dobu trvání tak, aby pokryla veškeré části tohoto úkolu.
9. Předchozí krok proveďte zároveň s úkolem "maintenance plan".

Nápověda

- Posun časové osy vpřed, respektive zpět, provedete zmáčknutím a následným podržením levého tlačítka myši a následným pohybem vlevo, respektive vpravo.
- Pro pohodlnější úpravu doby trvání můžete využít zoom pomocí podržení klávesy Control a kolečka myši. V případě touchpadu můžete využít gesto přiblížit, respektive oddálit.
- V případě provedení nechtěné akce můžete využít tlačítko "Undo"pro navrácení provedené akce.
- Aby byly úkoly, které jsou součástí vybraného úkolu, zvýrazněné, je potřeba, aby kategorie, ve které se nachází, byla zobrazená.