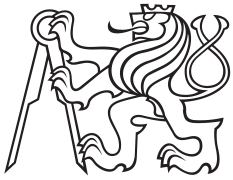
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Webová aplikace pro plánovaní údržby letadel

Yevgeniy Ulchenkov

Školitel: Mgr. Miroslav Blaško, Ph.D
Obor: Softwarové inženýrství a technologie
Studijní program: Enterprise systémy
Květen 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ulchenkov** Jméno: **Yevgeniy** Osobní číslo: **510855**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

Specializace: **Enterprise systémy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Webová aplikace pro plánovaní údržby letadel

Název bakalářské práce anglicky:

Web application for aircraft maintenance planning

Pokyny pro vypracování:

Cílem práce je vyvinout webovou aplikaci pro plánování údržby letadel v rámci hangáru. Aplikace umožní řízenou tvorbu plánu údržby, kde vstupem jsou historické data z plánování, nebo doménová znalost vyjádřená pomocí ontologie sémantického webu [2]. Aplikace umožní manuální i poloautomatickou tvorbu událostí, náhled na historii a validaci plánů. Projekt bude realizován ve spolupráci s odborníky z domény údržby, kteří na základě sběru dat z CSAT ověří vyvinutou aplikaci na reálných datech z provozu. Dalším cílem bude popsat jak tuto aplikaci začlenit do existující infrastruktury vytvořené v rámci projektu [1].
Instrukce:
1) seznámte se s technologiemi sémantického webu, které reprezentují (OWL, RDF, JSON-LD) a dotazy (SPARQL) doménové znalosti
2) prozkoumejte existující nástroje a knihovny pro vizuální konstrukci plánů
3) analyzujte požadavky na webovou aplikaci
4) navrhnete a implementujte webovovou aplikaci
5) otestujte implementovaný prototyp a validujte ho pomocí doménového odborníka

Seznam doporučené literatury:

1) Improving effectiveness of aircraft maintenance planning and execution, project no. CK01000204 in TAČR Doprava 2020+, Online at https://starfos.tacr.cz/en/projekty/CK01000204
2) Guizzardi, Giancarlo. "Ontological foundations for structural conceptual models." (2005).
3) Pluhař, Vít. "Web components for aircraft maintenance planning." (2022). Online at https://dspace.cvut.cz/handle/10467/100928.
4) React - A JavaScript library for building user interfaces (https://reactjs.org/)

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: **15.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

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

Nejdříve bych rád poděkoval vedoucímu mé bakalářské práce, Mgr. Miroslavu Blaškovi, Ph.D. Děkuji za jeho vstřícné vedení, trpělivost a spolupráci během celého procesu psaní této práce.

Dále bych chtěl poděkovat všem respondentům, kteří se zúčastnili evaluace a přispěli svými názory k obohacení praktické části práce.

Nesmírně děkuji také své rodině a přátelům, kteří mě při vytváření této práce podpořili a bez jejichž pomoci by nebylo možné práci dokončit.

Prohlášení

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

V Praze, 24. května 2024

Abstrakt

Tato bakalářská práce se zaměřuje na vývoj webové aplikace pro plánování údržby letadel v hangárovém prostředí. Hlavním cílem je poskytnout nástroj, který podporuje tvorbu plánů údržby a zároveň nabízí přístup k historickým plánovacím datům. Další část projektu se věnuje integraci vyvinuté aplikace do existující infrastruktury jako součást širšího projektu. Práce je vyvíjena ve spolupráci s odborníky z oblasti údržby CSAT.

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

Školitel: Mgr. Miroslav Blaško, Ph.D Praha, Resslova 307/9

Abstract

This bachelor's thesis focuses on developing a web application for aircraft maintenance planning within a hangar. The primary goal is to provide a tool that supports the creation of maintenance plans while also offering access to historical planning data. Another part of the project addresses the integration of the developed application into existing infrastructure as part of a broader project. The work is conducted in collaboration with maintenance experts from the CSAT domain.

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

Title translation: Web application for aircraft maintenance planning

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Plánování údržby letadel představuje rozsáhlý úkol. Historicky bylo plánování časově náročné [9] a náchylné k chybám [21], často realizované pomocí tabulkových procesorů. S rostoucím objemem úkolů spojených s údržbou letadel se vynořuje potřeba inovativního řešení[1]

1.1 Motivace

V současné době odborníci v oblasti údržby letadel v CSAT[2] nemají k dispozici žádný software pro plánování údržby letadel v rámci hangáru. Stávající postupy, včetně používání tabulkového procesoru Excel, nejsou optimální, a to jak z hlediska přehlednosti plánu, tak z hlediska rizika chyb.

1.2 Cíl

Cílem práce je vytvořit webovou aplikaci pro plánování údržby letadel v rámci hangáru, která umožní systematické a řízené vytváření plánu údržby. Aplikace bude schopna efektivně zpracovávat vstupní data z historického plánování a využívat doménovou znalost vyjádřenou prostřednictvím ontologie sémantického webu.

V projektu[3] byla vytvořena webová komponenta[4] pro přehled plnění jednotlivých úkolů v rámci plánů údržby konkrétních letadel a je třeba zvážit, zda je možné přepoužít některou její část v implementaci webové aplikace.

Klíčovým prvkem je spolupráce s odborníky z oblasti údržby, kteří na základě sběru dat z CSAT ověří funkcionalitu a účinnost vyvinuté aplikace na reálných datech z provozu.

[1] Aircraft Maintenance Tracking Software SAM. Více informace je na `https://asasoftware.aero/sam/`

[2] Czech Airlines Technics. Více informace je na `https://www.csatechnics.com/`

[3] České vysoké učení technické v Praze, fakulta dopravní. Zvýšení efektivity plánování a provádění údržby dopravních letadel (1.3.2020 - 28.2.2023). Více informace je na `https://starfos.tacr.cz/en/projekty/CK01000204`

[4] Pluhař, Vít. "Web components for aircraft maintenance planning."(2022). Více informace je na `https://dspace.cvut.cz/handle/10467/100928`

Kapitola 2
Úvod do problematiky

Tato kapitola je zaměřena na představení nejpodstatnějších aspektů zkoumaného tématu, které poskytují podrobný základ pro porozumění studie. Kapitola poskytuje ucelený přehled o významu hangárové údržby letadel, přičemž se zaměřuje na její klíčové [1] procedury. Navíc, kapitola nastiňuje základní informace o sémantickém webu, jeho klíčových[2] principech a technologiích.

2.1 Hangárová údržba letadel

Hangárová údržba letadel je klíčovou součástí procesu udržování leteckého průmyslu, zajišťující bezpečnost a efektivitu leteckého provozu. Tento termín se vztahuje na široké spektrum aktivit a procedur, které se provádějí na letadlech, když nejsou v aktivním provozu, a zahrnují jak pravidelnou, tak speciální údržbu. Hangárová údržba se obvykle provádí v uzavřeném, kontrolovaném prostředí, jako jsou hangáry. Do hangárové údržby patří:

- **Periodické kontroly**. Toto zahrnuje rutinní inspekce letadel, známé jako checky[3], které se provádějí v různých intervalech. Tyto kontroly zahrnují vizuální prohlídky, testování systémů, diagnostiku a opravy.
- **Opravy a výměny komponent**. Zahrnuje výměnu nebo opravu zastaralých nebo poškozených komponent, jako jsou motory, elektronické systémy, hydraulické součásti a strukturální prvky letadla.
- **Modernizace**. Tato část údržby se zabývá instalací nových technologií nebo upgradem existujících.
- **Speciální inspekce**. Tato kategorie zahrnuje speciální inspekce, které mohou být vyžadovány regulačními orgány nebo výrobcem letadel a modifikace pro zajištění souladu s nejnovějšími bezpečnostními normami nebo provozními požadavky.

[1] Aircraft maintenance hangar checklist. Získáno z `https://www.tronair.com/resources/aircraft-maintenance-hangar-checklist/`

[2] Marja-Riitta Koivunen and Eric Miller. W3C Semantic Web Activity. Získáno z `https://www.w3.org/2001/12/semweb-fin/w3csw`

[3] National Aviation Academy. Types of Aviation Maintenance Checks. Získáno z `https://www.naa.edu/types-of-aviation-maintenance-checks/`

Příklady hangárové údržby letadel:

- D-Check (Heavy Maintenance Check): tento typ kontroly je nejintenzivnější a provádí se každých 5-10 let. Zahrnuje kompletní demontáž letadla, podrobnou kontrolu všech částí, opravy nebo výměny poškozených dílů.
- Oprava poškození od krupobití: oprava vnější vrstvy letadla, která byla poškozená krupobitím.

2.2 Semantický web

V současné době se rozlišují tři fáze webu: Web 1.0, Web 2.0 a Web 3.0 [24].

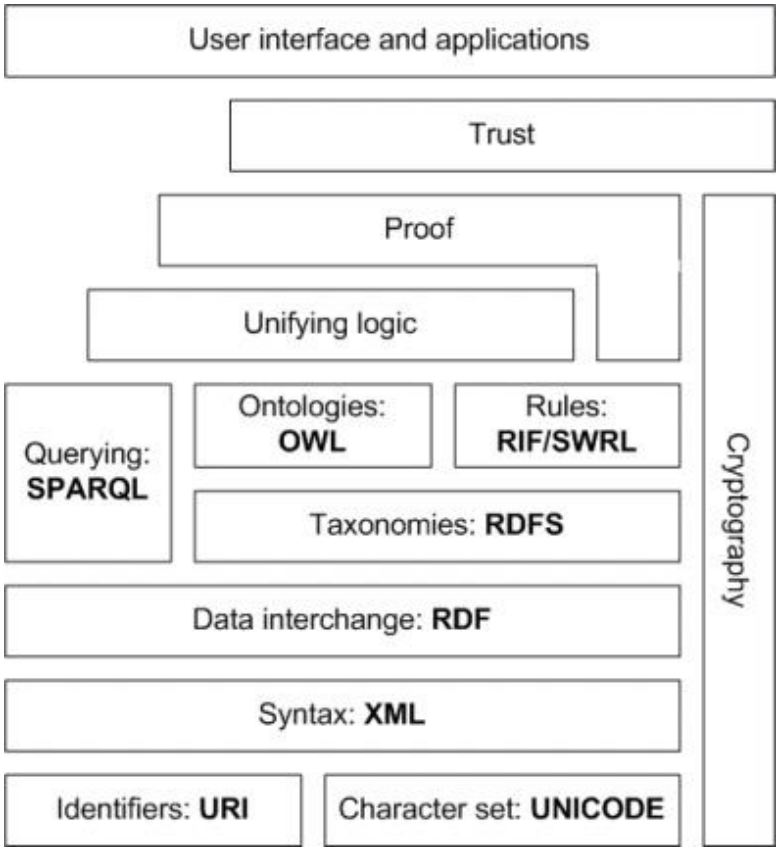
Původní Web 1.0 představuje nejstarší fázi webu a je typicky spojován s 90. lety 20. století a počátkem 21. století. Často je označován jako "web pouze pro čtení" (Read Only Web) a umožňoval jenom prohlížet statické stránky načtené ze serveru, obsah kterých byl publikován převážně jednotlivci nebo organizacemi, které vlastnily této webové stránky [24].

Web 2.0 představuje současnou fázi webu a je rozšířením Web 1.0. Často označován jako "web pro čtení a zápis" (Read-Write Web) a umožňuje uživateli přispívat obsahem a komunikovat s ostatními uživateli [12]. Jinými slovy, výrazným posunem, který Web 2.0 přinesl, byla zvýšená interaktivita v rámci online prostředí. Přestože tvorba obsahu je decentralizovanější, data jsou shromažďována a spravována centrálně velkými společnostmi, které provozují platformy a služby Webu 2.0.

Web 3.0 představuje další fázi rozvoje webu. Často označován jako "web pro čtení, zápis a vlastnictví" (Read-Write-Execute Web) a je rozšířením Web 2.0. Je postaven na základních konceptech decentralizace, otevřenosti a větší uživatelské užitečnosti [24]. V kontextu Web 3.0:

- Decentralizace znamená, že data a informace nejsou centrálně kontrolovány jednotlivými společnostmi nebo autoritami, místo toho jsou distribuovány napříč různými uzly v síti.
- Otevřenost odkazuje na principy svobodného přístupu a výměny informací. To zahrnuje použití otevřených standardů a otevřeného zdrojového kódu.
- Uživatelská užitečnost se zaměřuje na poskytování bohatších a personalizovanějších zkušeností pro uživatele (inteligentní vyhledávání, personalizované služby).

Sémantický web, často označován jako Web 3.0 [24], představuje web, který je o přechodu z webu primárně navrženého pro čtení lidmi na ten, který mohou číst stroje [1]. Sémantický web usiluje o vytvoření prostředí, kde data nejsou jen propojená, ale také strukturovaná a označená tak, aby byla srozumitelná pro stroje. Tento přístup je realizován prostřednictvím technologií (Semantic Web Stack, také známý jako Semantic Web Layer Cake [1], Obrázek 2.1, které umožňují explicitní popis významů (semantiky) dat a vztahů mezi nimi.



Obrázek 2.1: Semantic Web Stack.

2.3 Ontologie v sémantickém webu

Ontologie v sémantickém webu je formální popis znalostí jako soubor konceptů a vztahů v určité doméně. Ontologie definuje, jak jsou různé entity spojeny, jaký význam mají. Hlavním účelem ontologie je formálně strukturovat informace a poskytnout jasný model pro popis doménových znalostí. Poskytnutý model může být použít k vytvoření grafu znalostí – kolekce entit, kde typy a vztahy mezi nimi jsou vyjádřeny uzly a hranami mezi těmito uzly [15].

2.4 Jazyky pro ontologie a dotazování ontologií

2.4.1 RDF

Resource Description Framework (RDF) je konceptuální model, popisující jak reprezentovat informace v decentralizovaném světě. Je to základní technologie sémantického webu, která umožňuje strojům využívat strukturované informace (znalosti) distribuované v uzlech webu [23]. Slovník RDF poskytuje základy pro konstrukci sémantického webu a umožnění vytváření RDF trojic [1]. RDF trojice jsou způsobem reprezentace a propojování informací jako trojdílných

definic nebo faktů strukturovaných v formátu subjekt-predikát-objekt [1]. Existuje mnoho formátů vhodných pro reprezentaci dat v RDF. Nejčastěji používané a standardizované formáty jsou: N-Triples[4], Turtle[5], JSON-LD[6] a RDF/XML[7].

2.4.2 RDFS

RDF Schema (RDFS) je sémantickým rozšířením RDF, poskytujícím mechanismy pro popis skupin souvisejících zdrojů a vztahů mezi těmito zdroji pomocí systému tříd a vlastností, který je podobný typovým systémům objektově orientovaných programovacích jazyků, jako je Java [17].

2.4.3 OWL

Web Ontology Language (OWL, známé také jako OWL 1) je jazyk navržený pro vytváření komplexních ontologií pro lepší strojovou interpretaci webového obsahu než je podporováno RDF a RDF Schema (RDFS), a to poskytnutím dalšího slovníku spolu s formální sémantikou [25].
V reakci na narůstající požadavky z praxe byla vyvinuta nová verze tohoto jazyka, označovaná jako OWL 2. Tato revize přináší řadu nových funkcí, které rozšiřují výrazové možnosti původní specifikace OWL 1. Mezi klíčové rozdíly patří zvýšená expresivní síla pro vlastnosti, rozšířená podpora pro datové typy, zjednodušené možnosti meta-modelování, rozšířené možnosti anotace a implementace klíčů [26].

2.4.4 SPARQL

SPARQL Protocol and RDF Query Language (SPARQL) je standardizovaný dotazovací jazyk pro RDF, stejně jako SQL je standardizovaný dotazovací jazyk pro relační databáze. Dotaz SPARQL se skládá ze sady trojic, kde subjekt, predikát a/nebo objekt mohou sestávat z proměnných. Cílem je spojit RDF trojice v dotazu SPARQL s existujícími trojicemi RDF a najít je [20].

Pro čtení dat z databáze jazyk SPARQL definuje čtyři různé druhy dotazování pro různé účely.

- SELECT Dotaz: Tento typ dotazu vrací výsledky z SPARQL endpointu ve formě tabulky, která obsahuje pouze data odpovídající dotazu.
- CONSTRUCT Dotaz: Výsledky tohoto dotazu z SPARQL endpointu obsahují data vyhovující dotazu, která jsou převedena do formátu RDF.

[4] W3C. RDF 1.2 N-Triples. Více informace je na `https://www.w3.org/TR/rdf12-n-triples/`
[5] W3C. RDF 1.2 Turtle. Více informace je na `https://www.w3.org/TR/rdf12-turtle/`
[6] W3C. JSON-LD 1.1. Více informace je na `https://www.w3.org/TR/json-ld/`
[7] W3C. RDF 1.1 XML Více informace je na `https://www.w3.org/TR/rdf-syntax-grammar/`

- ASK Dotaz: Tento dotaz poskytuje výsledek jako boolean hodnotu (true/false, neboli pravda/nepravda), která indikuje, zda dotaz odpovídá požadovaným kritériím.
- DESCRIBE Dotaz: Vrací RDF formát, který poskytuje popis dat odpovídajících dotazu. Specifické trojice zahrnuté ve výsledku závisí na konkrétním SPARQL endpointu, což znamená, že dotaz vrací popis dat, nikoli přímo data samotná.

2.5 Formáty reprezentace ontologií

2.5.1 TTL

Terse RDF Triple Language (TTL, vyslovováno jako "turtle") představuje formát souboru používaný k vyjádření dat RDF. Formát TTL je standardem W3C[8] a je popisován jako "všestranný jazyk pro reprezentaci informací na webu"[27].

Následující příklad 2.1 [23] demonstruje reprezentaci jednoduchého objektu pomocí formátu TTL.

Příklad 2.1: TTL příklad.

```
@prefix : <http://www.example.org/>.
:richard :hasSister :rebecca
{ ?a :hasFather ?b . ?b :hasSister ?c . } => { ?a :
   ↪ hasAunt ?c } .
```

2.5.2 JSON-LD

JSON for Linked Data (JSON-LD), je metoda pro strukturování dat způsobem, který je snadno čitelný a srozumitelný pro stroje. Toho dosahuje poskytováním více kontextu prostřednictvím propojování dat s jejich významem pomocí mapování na definované významy v ontologii a jejich efektivního formátování. [18].

Následující příklad[9] 2.2 reprezentuje entitu Person pomocí JSON-LD:

Příklad 2.2: JSON-LD příklad.

```
{
  "@context": "https://json-ld.org/contexts/person.
     ↪ jsonld",
  "@id": "http://dbpedia.org/resource/John_Lennon",
  "name": "John Lennon",
  "born": "1940-10-09"
}
```

[8]W3C. Více informace je na https://www.w3.org/
[9]JSON for Linking Data. Získáno z https://json-ld.org/

2.6 Persistence dat (reprezentace v DBMS)

2.6.1 RDF Triple Stores

Triple Stores jsou systémy pro správu databází (DBMS) pro data modelovaná pomocí RDF. Na rozdíl od systémů pro správu relačních databází (RDBMS), které ukládají data do relací (nebo tabulek) a jsou dotazovány pomocí SQL, triplestores ukládají RDF trojice a jsou dotazovány pomocí SPARQL [19]. Klíčovou vlastností Triple Stores je schopnost provádět inferenci – odvození nových znalostí na základě aktuálních znalostí [16]. Triple Stores lze rozdělit do dvou kategorií [19]:

- Nativní Triple Stores, které jsou implementovány od nuly a využívají datový model RDF. Příkladem je AllegroGraph [10].
- Triple Stores podporované RDBMS jsou vytvořeny přidáním specifické vrstvy RDF k existujícímu RDBMS. Příkladem je Virtuoso[11].

[10] AllegroGraph. Více informace je na `https://allegrograph.com/products/allegrograph/`

[11] Virtuoso. Více informace je na `https://virtuoso.openlinksw.com/`

Kapitola 3

Analýza

V této kapitole jsou zahrnuty následující body:

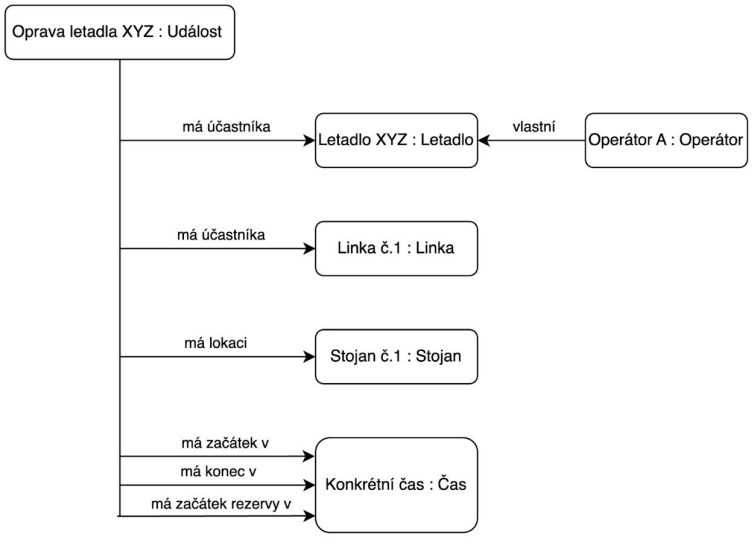
- Vysvětlení základních pojmů, které jsou nezbytné pro srozumitelnost a koherenci celého textu.
- Datová struktura vstupních dat, který popisuje, jak jsou vstupní data existující aplikace[1] organizována.
- Požadavky na vytvoření webové aplikace.
- Scénáře případů užití aplikace.

3.1 Základní pojmy

- Stojan (bay) – je specifická lokace v hangáru, která slouží pro umístění letadla.
- Linka – je skupina mechaniků.
- Událost – je entita, která je charakterizována tím, že se odehrává v určité prostorově-časové oblasti. V tomto kontextu je prostorově-časovou oblastí myšleno, že událost se děje v konkrétním času a trvá konkrétní dobu, během které letadlo je umístěno na určitém stojanu. Ke konkrétní události je přiřazená konkrétní linka. Událost může být typu:
  1. Plánovaná – událost, která se ještě neuskutečnila.
  2. Probíhající – událost, která se začala a momentálně neskončila.
  3. Vykonaná – událost, která již skončila.

  Událost se skládá ze dvou částí: jádra události a rezervy události.
- Jádro události – představuje hlavní časový úsek události, který začíná s jejím počátkem a pokračuje až do bodu, kde začíná její rezerva.

[1] Pluhař, Vít. "Web components for aircraft maintenance planning."(2022). Více informace je na `https://dspace.cvut.cz/handle/10467/100928`



Obrázek 3.1: UML diagram tříd pro entitu Událost s instancemi.

- Rezerva události – označuje doplňující časový úsek události, který navazuje na konec jádra a trvá až do konce celé události. Tento úsek je určen pro řešení potenciálních neplánovaných rozšíření nebo pro reakci na nepředvídatelné změny.
- Zdroj – v kontextu aplikace je zvolený účastník plánu (resp. relevantních událostí tohoto plánu). Příklady zdrojů:
  1. oblast letadla, kde se událost vykonává.
  2. hangar, kde se událost vykonává.
  3. linka, ktera se účastní události.
- Rozvrh – plán množin všech typů událostí, který je rozvržen do konkrétních zdrojů.
- Operátor/Klient – je společnost vlastnící letadlo, které se udržuje v hangáru.
- Work package (WP) – je plán údržby konkrétního letadla. Představuje určitý typ události (viz sekci 3.2. Požadavky, požadavek č. 2).

3.2 Požadavky

Prvotní požadavky na systém byly definovány po úvodní konzultaci s doménovým expertem. Tato fáze byla klíčová pro identifikaci a definici klíčových funkcionalit požadovaných pro úspěšnou implementaci systému.
Požadavky na aplikaci jsou rozděleny do dvou kategorií – funkční a nefunkční. Následně jsou funkční požadavky dále strukturovány do dvou podkategorií – požadavky spojené s zobrazením a ovládáním prvků. Každá kategorie

je následně prioritizována pomocí metodiky MoSCoW [4], která rozděluje požadavky do 4 kategorií:

1. Must-haves (M) - požadavky, které musejí být splněny. Jejich implementace je nezbytná v první fázi projektu.
2. Should-haves (S) - požadavky, které měly by být splněny. Jejich implementace může být odložena na pozdější fáze.
3. Could-haves (C) - požadavky, které mohou být splněny. Jejich implementace může být provedena, pokud je k dispozici dostatek času a zdrojů.
4. Wont-haves (W) - požadavky, které mohou být vyřazeny. Jejich implementace nebude provedena v aktuálním projektu, mohou být odloženy na budoucí fáze.

3.2.1 Funkční požadavky

Požadavky na zobrazení

- **FR 1. Zvýraznění rezervy události (M)**. Systém zvýrazní vizuálně rezervu události pro snadnou identifikaci.
- **FR 2. Zobrazení různých typů událostí (S)**. Systém odliší vizuálně mezi různými typy událostí, jako jsou:
  - Work package.
  - Denní událost.
  - Událost na konkrétní datum.
- **FR 3. Zobrazení detailů události (M)**. Systém umožní zobrazení detailů pro vybranou událost.
- **FR 4. Zvýraznění události (M)**. Systém zvýrazní událost na základě uživatelské akce pro lepší orientaci.
- **FR 5. Zobrazení rozvrhu (M)**. Systém umožní zobrazení rozvrhu hangáru na časové ose vzhledem ke každé události s následujícími informacemi:
  1. číslo/název stojanu
  2. počáteční datum
  3. koncové datum
  4. název WP
- **FR 6. Označení aktuálního času (M)**. Systém označí aktuální čas na časové ose pro snadnou orientaci.
- **FR 7. Označení svátků a víkednů(C)**. Systém zvýrazní vizuálně svátky na časové ose pro výraznější identifikaci.

- **FR 8. Přepínání zobrazení (S)**. Systém umožní přepínání zobrazení podle určitých zdrojů. Např. zobrazení dle stojanu nebo dle operátora.
- **FR 9. Zobrazení různých úrovní plánu (C)**. Systém zobrazí různé úrovně plánu v hierarchické struktuře.

Požadavky na ovládání

- **FR 10. Manuální plánování událostí různého typu (M)**. Systém umožní ruční plánování různých typů událostí, jako jsou:
  - work package
  - denní událost
  - událost na konkrétní datum
- **FR 11. Přidávání rezervy události (M)**. Systém umožní přidávání rezervy k události.
- **FR 12. Přidávání poznámek k události (M)**. Systém umožní přidávání poznámek k jednotlivým událostem. Příklady poznámek na událost:

  Příklady takových poznámek:

  - **Poznámky na work package**
    1. začátek posunout kvůli zákazníkovi
    2. revize posunuta kvůli začátku betonování
  - **Poznámka na denní událost**
    1. betonování stojánku
  - **Poznámky na konkrétní datum**
    1. začátek udržbové sezony
    2. uzavřená příjezdová dráha k hangáru
- **FR 13. Změna rozsahu časové osy rozvrhu (M)**. Systém umožní:
  - zvětšení
  - zmenšení

  časové osy rozvrhu pro zobrazení jeho konkrétních částí.
- **FR 14. Navigace v časové ose plánu (M)**. Systém umožní posunutí časové osy rozvrhu.
- **FR 15. Přiřazení linky (M)**. Systém umožní přiřazení linky k jednotlivým událostem.
- **FR 16. Historie plánování (S)**. Systém umožní náhled do historie plánování.

- **FR 17. Vrácení a odstranění úprav (S)**. Systém umožní funkci pro odstranění poslední změny (undo) a pro opětovné provedení nedávno odstraněné změny (redo) při tvorbě rozvrhu.
- **FR 18. Simulace plánování (S)**. Systém umožní simulaci při plánování, včetně přidávání, přesouvání a mazání časových os udalostí.
- **FR 19. Automatické plánování (M)**. Systém umožní automatické plánování podle doménových pravidel, které bude možné zadávat v čase deploymentu aplikace. (např. Airbus (A32x) se bude plánovat jenom na stojan 2/stojan 3)
- **FR 20. Validace plánu (M)**. Systém umožní validaci plánu na základě doménových pravidel.
- **FR 21. Vyhledávání události (M)**. Systém umožní vyhledávání událostí podle:
  - názvu události
  - klienta
  - letadla
- **FR 22. Single–Sign On autorizace a autentizace (S)**. Systém provádí autorizaci a autentizaci pomocí protokolu OpenID Connect. provádí autorizaci a autentizaci pomocí protokolu OpenID Connect.
- **FR 23. Přidávání stojanu (C)**. Systém umožní přidávání stojanu (např. v souvislosti s rozšířením hangáru).

3.2.2 Nefunkční požadavky

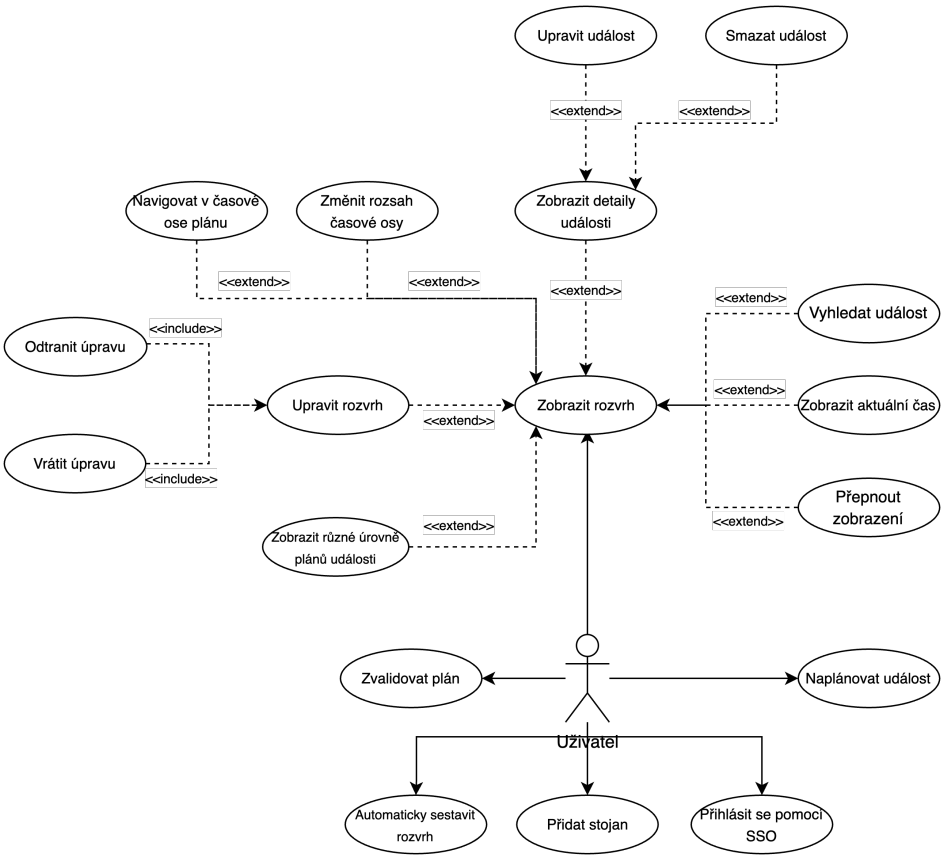
- **NFR 1. Systém je přístupný pomocí moderních internetových desktopových webových prohlížečů**:
  1. Google Chrome verze 96.0.4664 nebo vyšší (M)
  2. Firefox 91.0 verze nebo vyšší (M)
  3. Safari verze 14.0 nebo vyšší (S)
- **NFR 2. Uživatelské rozhraní systému je v angličtině (M)**.
- **NFR 3. Automatizované nasazení změn na server (S)**. Všechny změny, které jsou nahrány do repozitáře, se automaticky nahrají na produkční server.
- **NFR 4. Systém je vyvinut pomocí moderních a široce používaných technologií (M)**.
  1. Frontend: knihovna React v. 18
  2. Backend: Java v. 17

- **NFR 5. Vizuální konzistence s existující aplikací (M)**. Vyvíjená aplikace musí vizuálně korespondovat s designem a uspořádáním existující aplikace. Tento požadavek zahrnuje následující specifické aspekty:

  1. **Barevné schéma:** Barevné schéma vyvíjené aplikace musí přesně odpovídat barevnému schématu použitému v existující aplikaci. To zahrnuje primární, sekundární a akcentové barvy.
  2. **Poloha hlavního komponentu pro plánování:** Hlavní komponent pro plánování v nové aplikaci musí být umístěn na stejném místě v uživatelském rozhraní jako v existující aplikaci.
  3. **Vizuální prvky a komponenty:** Všechny hlavní vizuální prvky (např. tlačítka, formulářová pole, ikony) musí být ve vyvíjené aplikaci stylizovány a rozmístěny tak, aby vizuálně odpovídaly těm v existující aplikaci.

3.3 Případy užití

V diagramu případů užití, zobrazeném na obrázku 3.2, jsou reprezentovány interakce aktérů s navrhovanou plánovací komponentou.

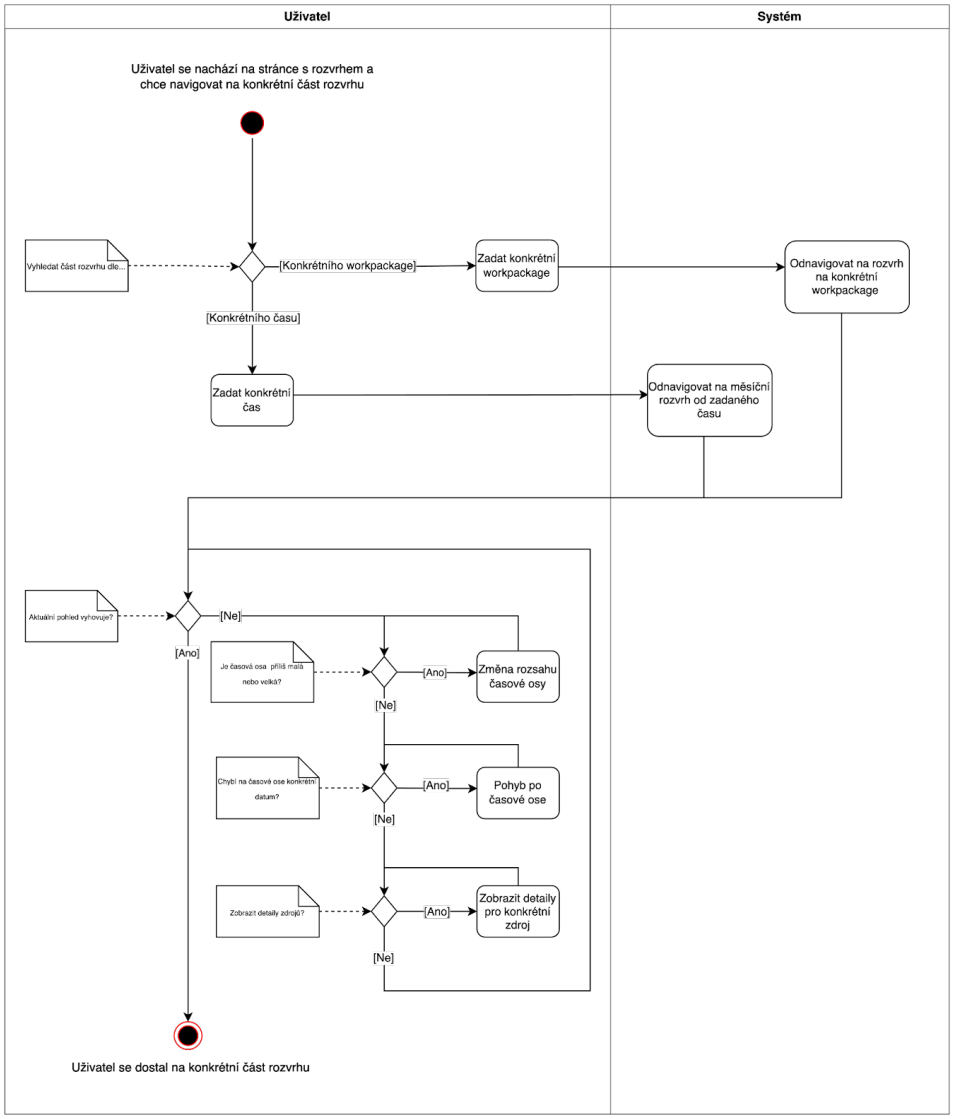


Obrázek 3.2: UML diagram případů užití aplikace.

3.4 Scénáře případů užití

3.4.1 Zobrazení rozvrhu od konkrétního času nebo zobrazení rozvrhu na konkrétní work package.

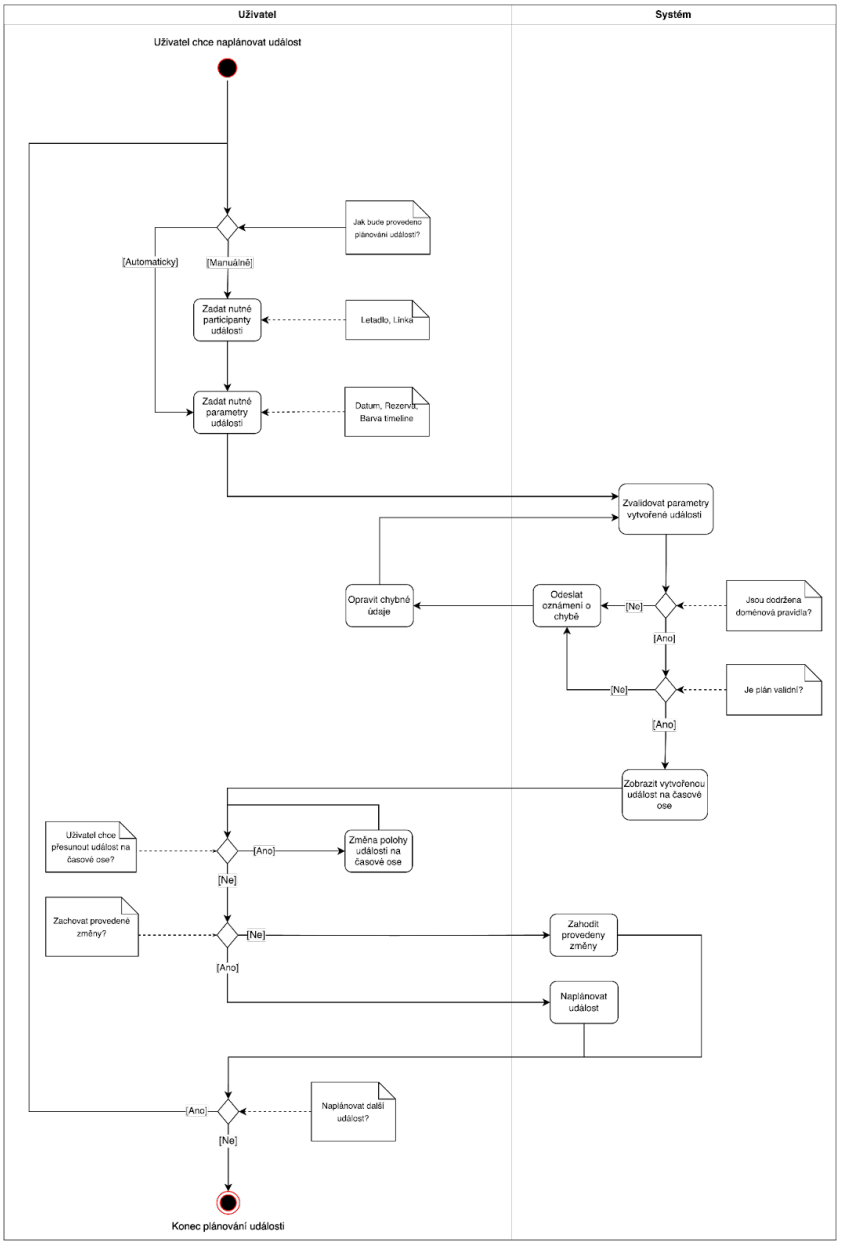
Cílem je navigovat uživatele na požadovanou část rozvrhu.



Obrázek 3.3: UML diagram: Scénář zobrazení rozvrhu od konkrétního času nebo zobrazení rozvrhu na konkrétní work package.

3.4.2 Plánování sady událostí.

Cílem je naplánovat sadu událostí bud' automaticky nebo manuálně podle parametrů.



Obrázek 3.4: UML diagram: Scénář plánování sady událostí.

3.5 Analýza existujících řešení

Vývoj aplikace pro plánování údržby letadel představuje rozsáhlý a komplexní úkol. Vzhledem k obtížnosti tvorby takovéto aplikace od základů je nezbytné provést důkladný průzkum stávající aplikace s cílem zjistit, zda lze některé prvky znovu využít. Analýza existující aplikace bude zahrnovat následující klíčové aspekty:

- **Prozkoumání datových struktur:** Bude provedeno zkoumání dat, s kterými stávající aplikace pracuje. Cílem je posoudit možnosti jejich aplikace a případného převzetí pro potřeby vyvíjené aplikace.
- **Analýza plánovacích knihoven:** Součástí průzkumu bude také hodnocení knihoven pro plánování. Knihovny budou podrobně porovnány s cílem identifikovat nejvhodnější řešení pro implementaci v navrhované aplikaci.

3.5.1 Komponenta pro zobrazení plánu údržby letadel

V této kapitole je analyzována datová struktura vstupních dat existující webové komponenty a na základě této analýzy je zvažováno, zda je možné použít tato vstupní data v implementaci vyvíjené aplikace.

Identifikace typů vstupních dat

Existující webová komponenta, která realizuje zobrazení plánu údržby, čerpá vstupní data ve formátu JSON z backendového systému[2] navrženého pro účely plánování údržby letadel. Tato data mají hierarchickou, vícevrstvou strukturu.

Struktura dat podrobněji

Zjednodušený diagram tříd na obrázku 3.5 ukazuje, jak jsou hlavní prvky strukturovány a vzájemně propojeny. Zde je podrobnější popis každé třídy a jejich funkcí v rámci plánu:

- Kořenovým elementem je RevisionPlan, který reprezentuje samotný plán pro revizi.
- PhasePlan seskupuje TaskPlan do určité fáze v plánu, např. fáze testování a kontrola.
- RestrictionPlan reprezentuje restrikce v plánu, např. kdy bude vypnutí elektřiny.
- GeneralTaskPlan reprezentuje lokaci, v které se vykonávají úkoly, např. kokpit.

[2] KBSS at FEE CTU, Improving effectiveness of aircraft maintenance planning and execution. Více informace je na `https://kbss.felk.cvut.cz/projects/doprava2020-csat/`

- TaskPlan reprezentuje určitý úkol:
  - Část taskType představuje typ úkolu.
  - Část taskStepPlans představuje kroky, provedeny v konkrétním úkolu.
  - Část MaintenanceGroup reprezentuje linku, která je zodpovědná za provedení samotných kroku v úkolu.
  - SessionPlan reprezentuje práci vykonanou konkrétnymi lidmi.

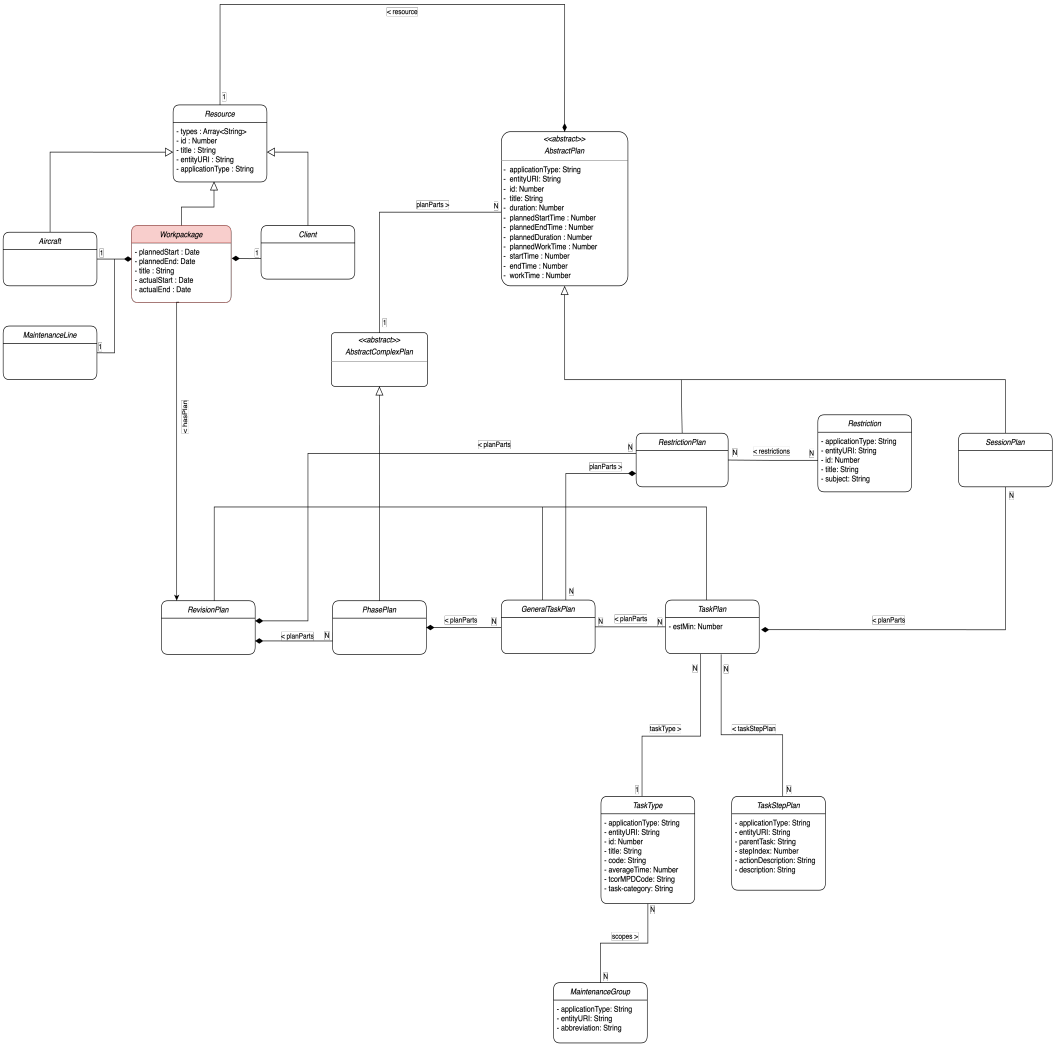
Důležitost třídy Workpackage

V rámci hierarchie datového modelu, prezentovaného na obrázku 3.5, je klíčovým prvkem třída **Workpackage**, která je na diagramu zvýrazněna červenou barvou. Třída *Workpackage* poskytuje strukturu pro organizaci údržbových úkolů. Tato třída představuje základní jednotku, na úrovni které se bude realizovat plánování v rámci vyvíjené aplikace.

Zhodnocení datové struktury a její dopady

Data ve formátu, jak jsou prezentována, výborně vyhovují požadavkům pro budování aplikace, nicméně, bude nutné rozšířit existující strukturu o některé atributy, které jsou v současnosti absentující, jako například možnost přidání poznámek k jednotlivým událostem.

Navíc, využití této stávající struktury dat umožní hladší integraci s již existující aplikací, což je výhodné z hlediska snížení nákladů na vývoj a zkrácení času potřebného pro implementaci. Díky tomu bude možné efektivně splnit požadavek FR10 (viz. Požadavky na zobrazení) na hierarchické zobrazení plánu.



Obrázek 3.5: Diagram tříd datové struktury vstupních dat.

3.5.2 Analýza plánovacích knihoven

Po úvodní identifikaci datových struktur je nezbytné se zaměřit na zkoumání existujících knihoven pro plánování událostí, včetně té, která je již využívána v existující aplikaci, a porovnat je pomocí definovaných kritérií.

Kritéria pro porovnání knihoven. Pro porovnání plánovacích knihoven pro vývoj aplikace je důležité zvážit několik klíčových aspektů, které ovlivní jak funkčnost aplikace, tak její budoucí udržitelnost a rozšiřitelnost. Následující kritéria byla stanovena na základě kapitoly 3.2. Požadavky a potřeb projektu

- **C1.** Plná kompatibilita s React frameworkem.
- **C2.** Schopnost knihovny podporovat vytváření, čtení, aktualizaci a mazání událostí v rámci aplikace.
- **C3.** Podpora interaktivní manipulace s událostmi:
  1. **C3a.** Drag and drop funkce pro snadné plánování a přesun událostí na časové ose.
  2. **C3b.** Zvětšení/zmenšení událostí pomocí tahání pro přesné nastavení doby trvání.
- **C4.** Možnosti, které knihovna nabízí pro navigaci a manipulaci s časovou osou:
  1. **C4a.** Pohyb po časové ose.
  2. **C4b.** Zvětšení/zmenšení rozsahu časové osy.
- **C5.** Možnosti zobrazení prvků:
  - **C5a.** Schopnost modifikovace způsobu zobrazení jednotlivých prvků.
  - **C5b.** Hierarchické zobrazení plánu událostí.
  - **C5c.** Hierarchické zobrazení skupin událostí.
- **C6.** Možnost snadného rozšíření existujících funkcí.
- **C7.** Kompletní dokumentace funkcí a možností.
- **C8.** Licenční podmínky a možnost modifikace zdrojového kódu.

Metodika vyhledávání. Při hledání vhodných plánovacích knihoven využity platformy GitHub[3] a Google[4]. Na těchto platformách provedeno systematické vyhledávání pomocí klíčových slov relevantních pro potřeby, jako jsou:

- react planner.
- react calendar.

[3] GitHub. Více informace je na `https://github.com/`
[4] Google. Více informace na `https://www.google.com/`

- react planning tool.
- react scheduler.
- kombinace z výše uvedených.

Zkoumané knihovny. Po provedení vyhledávání bylo identifikováno několik knihoven, které potenciálně vyhovují požadavkům aplikace pro plánování údržby letadel. Celkem bylo nalezeno 5 knihoven, které splňují kritéria vyhledávání a nabízejí rozličné funkce vhodné pro implementaci plánovacího systému:

- react-calendar-timeline[5].
- react-big-calendar[6].
- react-big-scheduler[7].
- DayPilot[8].
- Planby[9].

Analýza se začne knihovnou, která je již využívána v existující aplikaci. Tato knihovna byla vybrána k dalšímu zkoumání z důvodů její předchozí integrace a osvědčení v praxi.

react-calendar-timeline

Moderní a responzivní komponenta časové osy pro React.

Vyhodnocení shody s kritérii.

- **C1:** react-calendar-timeline je speciálně navržena pro React a je plně kompatibilní s tímto frameworkem, což je podpořeno širokou komunitou uživatelů a vývojářů.
- **C2:** Knihovna nemá vestavěné funkce pro vytváření, mazání a aktualizaci událostí. Tyto interakce s událostmi musí být implementovány uživatelem s využitím externích ovládacích prvků.
- **C3:**
  1. **C3a:** Podporuje drag and drop událostí na časové ose.
  2. **C3a:** Podporuje interaktivní zvětšení a zmenšení událostí.

[5]react-calendar-timeline. Více informace je na `https://github.com/namespace-ee/react-calendar-timeline`

[6]react-big-calendar. Více informace je na `https://github.com/jquense/react-big-calendar`

[7]react-big-scheduler. Více informace je na `https://github.com/StephenChou1017/react-big-scheduler`

[8]DayPilot. Více informace je na `https://www.daypilot.org/`

[9]Planby. Více informace je na `https://planby.app/`

- **C4:**
  1. **C4a:** Umožňuje dynamický pohyb a posouvání časové osy.
  2. **C4b:** Podporuje dynamický zoom in a zoom out.
- **C5:**
  1. **C5a:** Uživatelé mohou přizpůsobit vzhled událostí a časové osy dle svých potřeb.
  2. **C5b:** Knihovna podporuje strukturované zobrazení událostí.
  3. **C5c:** Umožňuje hierachicky zobrazit skupiny událostí.
- **C6:** Knihovna je modulárně navržena, což umožňuje snadné rozšíření a přidání nových funkcí, jak vyžaduje aplikace.
- **C7:** Má dobře dokumentované API a uživatelskou příručku, která pokrývá všechny základní a pokročilé funkce.
- **C8:** Knihovna je distribuována pod licencí MIT, což umožňuje její volné používání, modifikaci a distribuci, včetně komerčního využití [4].

Dodatečné výhody.

- **Bohatá sbírka příkladů**: Knihovna nabízí rozsáhlou kolekci praktických příkladů kódu, které usnadňují pochopení jejího použití a integraci.
- **Plná funkcionalita v bezplatné verzi**: Uživatelé mají přístup ke všem funkcím knihovny bez nutnosti platit za prémiové verze.

Rozšíření knihovny. K dispozici je nadstavba[10] speciálně vytvořená pro tuto knihovnu, která disponuje funkcemi nezbytnými pro existující aplikaci. Tato nadstavba dokáže zjednodušit integraci vyvíjené aplikace s již existující a významně přispívá ke zlepšení její funkcionality a adaptace na specifické požadavky.

Dodatečné nevýhody.

- **Neaktivní vývoj**: Poslední commit v repozitáři byl proveden před dvěma lety, což může naznačovat nedostatek pravidelných aktualizací a potenciální zastarání technologie.

react-big-calendar

Plánovač a komponenta pro plánování zdrojů speciálně vytvořená pro React a optimalizovaná pro moderní prohlížeče.

[10] react-maintenance-planner. Více informace je na `https://github.com/kbss-cvut/react-maintenance-planner`

Vyhodnocení shody s kritérii.

- **C1:** react-big-calendar je výhradně určená pro React a využívá jeho ekosystém a principy.
- **C2:** Knihovna poskytuje funkce, které umožňují vybrat časový slot pro vytváření událostí. Nicméně funkce pro aktualizaci a mazání událostí nejsou dostupné.
- **C3:**
  1. **C3a.** Knihovna obsahuje drag and drop funkce pro manipulace s událostmi.
  2. **C3b.** Možnost interaktivně měnit dobu trvání událostí není.
- **C4:**
  1. **C4a:** Knihovna zobrazuje události ve vertikální poloze, což znamená, že pohyb po časové ose je směrem nahoru a dolů. Toto řešení může být potenciálně nevýhodné v případě, že je v kalendáři mnoho událostí, což může ztížit přehlednost a orientaci.
  2. **C4b:** Možnost zvětšení a zmenšení časové osy je přítomna, ale není dynamická. Uživatelé mohou vybírat pouze mezi předdefinovanými časovými intervaly, jako jsou den, týden a měsíc.
- **C5:**
  1. **C5a:** react-big-calendar nabízí možnost pro přizpůsobení vzhledu událostí.
  2. **C5b:** Knihovna nepodporuje zobrazení událostí v hierarchické struktuře.
  3. **C5c:** Nepodporuje vlastní zobrazení skupin událostí, pouze výběr z předdefinovaných možností. Mimo jiné, skupinami událostí mohou být pouze dny, což omezuje flexibilitu při organizaci a zobrazení událostí.
- **C6:** Má mnoho vystavěných API pro rozšíření funkcí.
- **C7:** Dokumentace knihovny je detailní a dobře pokrývá všechny aspekty funkcí a možností.
- **C8:** Knihovna je distribuována pod licencí MIT, což umožňuje volné použití a modifikaci.

Dodatečné výhody.

- **Sbírka interaktivních příkladů:** Knihovna poskytuje velkou kolekci praktických příkladů kódu.
- **Plná funkcionalita v bezplatné verzi:** Uživatelé mají přístup ke všem funkcím knihovny bez nutnosti platit za prémiové verze.

- **Overlap funkce:** Knihovna podporuje overlap funkce, které jsou užitečné pro zobrazení různých typů událostí na jednom kalendáři.
- **Velká komunita vývojářů:** Knihovna má širokou podporu v komunitě vývojářů.

Dodatečné nevýhody.

- **Směr zobrazení:** Jak již bylo zmíněno v hodnocení podle kritéria **C5c**, knihovna zobrazuje události směrem nahoru a dolů, což je zásadní nevýhoda při velkém počtu událostí. Tento vertikální směr zobrazení může způsobit problémy s přehledností a efektivitou při procházení rovrhu událostí.

react-big-scheduler

Plánovací komponenta pro plánování a správu zdrojů postavená pro React, určená pro moderní prohlížeče

Vyhodnocení shody s kritérii.

- **C1:** react-big-scheduler je plně kompatibilní s React frameworkem.
- **C2:** Knihovna poskytuje funkce pro výběr časového slotu při vytváření událostí. Přestože přímé možnosti pro mazání a aktualizaci událostí nejsou standardně integrovány, knihovna disponuje rozhraním API, které umožňuje vývojářům implementovat tyto operace podle specifických potřeb.
- **C3:**
  1. **C3a:** Knihovna je vybavena funkcemi pro drag and drop pro přesun událostí.
  2. **C3b:** Funkce umožňující interaktivní úpravu délky trvání událostí je součástí knihovny.
- **C4:**
  1. **C4a:** Knihovna poskytuje možnost dynamického pohybu a posouvání časové osy, které je realizováno prostřednictvím speciálně navrženého ovládacího prvku.
  2. **C4b:** Možnost zvětšení a zmenšení časové osy je dostupná, avšak omezená na výběr z přednastavených časových intervalů, jako jsou např. den, týden a měsíc.
- **C5:**
  1. **C5a:** Knihovna umožňuje přizpůsobení vzhledu jednotlivých událostí, včetně barev, stylů a dalších vizuálních prvků.
  2. **C5b:** Nepodporuje zobrazení událostí ve strukturovaném formátu.

3. **C5c:** Knihovna nabízí možnost hierarchického uspořádání skupin událostí.

- **C6:** Architektura knihovny zahrnuje robustní API, který umožňuje rozšíření a adaptaci funkcionalit.
- **C7:** Dokumentace knihovny je detailní a dobře pokrývá všechny aspekty funkcí a možností.
- **C8:** Knihovna je distribuována pod licencí MIT, což umožňuje volné použití a modifikaci.

Dodatečné výhody.

- **Sbírka interaktivních příkladů:** Knihovna nabízí mnoho interaktivních ukázkových kódů, které demonstrují její praktické využití.
- **Plná funkcionalita v bezplatné verzi:** Uživatelé mají přístup ke všem funkcím knihovny bez nutnosti platit za prémiové verze.
- **Funkce přizpůsobení zobrazení:** Knihovna umožňuje modifikaci zobrazení kalendáře podle různých parametrů, jako je např. zdroj událostí.
- **Estetický a přehledný layout:** Knihovna již od začátku poskytuje vizuálně atraktivní a čistý design.
- **Přítomnost zajímavých funkcí:** Knihovna disponuje řadou atraktivních funkcí, jako je built-in vyskakovací okno pro detailnější informace o událostech, validace pro detekci překryvu událostí, které zlepšují interaktivitu a uživatelskou zkušenost.

Dodatečné nevýhody.

- **Malá komunita vývojářů:** Knihovna nemá velkou komunitu vývojářů, což může omezovat dostupnost podpory.
- **Zastarání:** Poslední commit v repozitáři byl proveden před pěti lety, což naznačuje, že knihovna nemusí být pravidelně aktualizována a může být zastaralá vzhledem k nejnovějším technologickým trendům.

DayPilot

DayPilot je rozsáhlý software pro plánování a kalendářní komponenty určené pro aplikace založené na webu. Tento nástroj poskytuje bohaté možnosti pro implementaci plánovačů, kalendářů, Ganttových diagramů a dalších souvisejících funkcí do webových aplikací.

Vyhodnocení shody s kritérii.

- **C1:** DayPilot nabízí plnou kompatibilitu s React frameworkem prostřednictvím DayPilot React komponent.
- **C2:** DayPilot podporuje všechny CRUD operace (vytváření, čtení, aktualizaci a mazání) pro události. Tato funkcionalita je podporována přes dobře definované API.
- **C3:**
  1. **C3a:** DayPilot nabízí plnou podporu pro drag and drop funkce.
  2. **C3b:** DayPilot umožňuje uživatelům změnit délku trvání událostí pomocí tahání hran událostí.
- **C4:**
  1. **C4a:** Umožňuje dynamický pohyb po časové ose pomocí ovládacího prvku.
  2. **C4b:** Podporuje zvětšení a zmenšení rozsahu časové osy.
- **C5:**
  1. **C5a:** Poskytuje rozsáhlé možnosti pro přizpůsobení vzhledu událostí a dalších prvků.
  2. **C5b:** Knihovna nabízí hierarchické zobrazení událostí.
  3. **C5c:** Knihovna nabízí hierarchické zobrazení skupin událostí.
- **C6:** DayPilot nabízí API, které umožňuje snadné rozšíření a přizpůsobení existujících funkcí.
- **C7:** Má vynikající dokumentaci, která obsahuje detailní popisy funkcí, ukázky kódu a návody, což výrazně usnadňuje implementaci a použití knihovny.
- **C8:** DayPilot je komerční produkt s dostupnými licencemi pro různé potřeby. Licenční podmínky umožňují komerční použití, avšak modifikace zdrojového kódu je omezena.

Dodatečné výhody.

- **Široká škála funkcí:** DayPilot nabízí rozsáhlou sadu funkcí, které pokrývají různé aspekty plánování a kalendářního managementu. Mezi nejzajímavější patří: validace plánů, třídění, filtrování, specifická logika pro manipulaci s událostmi v různých skupinách, různé druhy optimalizace výkonu a další.
- **Komerční produkt s podporou:** DayPilot je komerční produkt, což znamená, že uživatelé mohou očekávat profesionální podporu a pravidelné aktualizace. Licenční podmínky zajišťují, že produkt je udržován a vyvíjen s ohledem na potřeby zákazníků.

- **Kompletní příklady aplikací:** DayPilot poskytuje kompletní příklady aplikací, které demonstrují, jak knihovna funguje v reálných scénářích.
- **Komplexní funkcionalita pro podnikové aplikace:** Mnoho funkcionalit, které jsou běžně požadovány v podnikových aplikacích, je již v DayPilot implementováno. To zahrnuje správu zdrojů, různé typy událostí a jejich zobrazování, a pokročilé možnosti přizpůsobení.

Dodatečné nevýhody.

- **Omezená komunita:** Jako komerční produkt nemusí mít DayPilot tak rozsáhlou komunitu uživatelů a vývojářů jako některé open-source alternativy.
- **Komplexita:** I když je dokumentace podrobná, množství dostupných funkcí a možností může být pro nové uživatele zpočátku ohromující. Vyžaduje to určitou dobu na seznámení a pochopení všech nabízených možností.

Planby

Planby je moderní knihovna pro plánování a správu událostí v aplikacích postavených na Reactu.

Vyhodnocení shody s kritérii.

- **C1:** Knihovna je navržena specificky pro použití s Reactem.
- **C2:** Vystavené API knihovny nezahrnuje explicitní funkce pro vytváření, mazání a aktualizaci událostí; tyto funkce je nutné implementovat dodatečně.
- **C3:**
  1. **C3a:** Planby podporuje funkci drag and drop událostí na časové ose.
  2. **C3b:** Knihovna umožňuje uživatelům interaktivně upravovat délku trvání událostí.
- **C4:**
  1. **C4a:** Umožňuje uživatelům dynamicky se pohybovat po časové ose pomocí ovládacího prvku.
  2. **C4b:** Knihovna nabízí pouze přednastavené možnosti pro změnu rozsahu časové osy, chybí flexibilní nastavení.
- **C5:**
  1. **C5a:** Nabízí možnosti pro úpravu vizuální prezentace jednotlivých prvků.

2. **C5b:** Neumožňuje hierarchické zobrazení plánu událostí.
3. **C5c:** Umožňuje hierarchické zobrazení skupin událostí.

- **C6:** Planby má omezené veřejně dostupné API, což omezuje její flexibilitu pro specifické implementace.
- **C7:** I přesto, že dokumentace poskytuje přehled funkcí, postrádá hlubší popis a detailní příklady kódů, což může komplikovat její použití pro složitější aplikace.
- **C8:** Planby má specifickou licenci, která umožňuje užití knihovny pouze pro její zamýšlený účel. Licenční smlouva je nevýhradní a nepřenosná, bez možnosti sublicencování.

Dodatečné výhody.

- **Moderní design:** Planby nabízí moderně navržené uživatelské rozhraní, které je vizuálně přitažlivé.
- **Aktivní vývoj:** Knihovna je pravidelně aktualizována a rozšiřována o nové funkce.

Dodatečné nevýhody.

- **Omezené API:** Planby neobsahuje široce rozvinuté API, což může omezovat možnosti pro rozšíření a přizpůsobení funkcionalit podle specifických požadavků projektu. Kromě toho, některé části API nejsou poskytovány zdarma.
- **Optimalizace pro složité plánovací úlohy:** Planby je více orientována na základní plánovací úkoly, jako je organizace událostí nebo zobrazování programů, a není ideální pro složité plánovací úlohy.
- **Malá komunita:** Omezená uživatelská a vývojářská komunita může vést k menší podpoře a méně dostupným zdrojům pro troubleshooting a učení, což může komplikovat implementaci a řešení problémů.

3.5.3 Shrnutí analýzy plánovacích knihoven

Po důkladném prozkoumání a srovnání pěti potenciálních knihoven, které by mohly být využity pro vývoj aplikace, byly identifikovány klíčové výhody a nevýhody každé z nich. *Tabulka 3.1* poskytuje vizuální přehled shody každé knihovny s definovanými kritérii, kde:

- Zelená barva značí, že knihovna plně splňuje dané kritérium.
- Žlutá barva indikuje, že knihovna splňuje kritérium částečně.
- Červená barva signalizuje, že knihovna kritérium nesplňuje.

- Sloupec **V-NV** zobrazuje, zda převládají dodatečné výhody nad nevýhodami a zda jsou tyto faktory dostatečně zásadní, aby ovlivnily rozhodovací proces při výběru knihovny pro projekt.
  - Zelená barva značí, že výhody převažují nad nevýhodami a jejich dopad na vývoj aplikace je minimální.
  - Červená barva indikuje opačnou situaci.

Tabulka 3.1: Srovnání knihoven podle kritérií.

Kritéria / Knihovna	C1	C2	C3	C4	C5	C6	C7	C8	V-NV
React-calendar-timeline									
React-big-calendar									
React-big-scheduler									
DayPilot									
Planby									

Vyhodnocení plánovacích knihoven

react-big-calendar se ukázala jako adaptabilní knihovna, nicméně má omezení v pokročilých manipulačních funkcích s událostmi a její vizuální styl nesplňuje požadavky projektu.

Planby je vhodná pro méně komplexní plánovací úkoly. Avšak její API je omezené a nedostatečné pro pokročilé požadavky tohoto projektu.

DayPilot, **react-big-scheduler** a **react-calendar-timeline** nabízejí širší rozsah funkcionalit, které mohou lépe vyhovovat komplexním plánovacím požadavkům. Přesto mají tyto knihovny určitá omezení co se týče licenčních podmínek a možností úprav.

Výběr plánovací knihovny pro vyvíjenou aplikaci

Pro finální výběr byla zvolena knihovna 3.5.2. **react-calendar-timeline**, zejména její **rozšíření**, které již bylo úspěšně integrováno do existující aplikace. Tento faktor hraje zásadní roli, neboť předchozí osvědčení v praxi a zkušenosti s touto knihovnou znamenají, že integrace nových funkcí bude pravděpodobně hladší a méně náročná na zdroje. Nezbytné funkce, které knihovna původně neobsahuje, budou doimplementovány pro splnění specifických požadavků projektu.

Kapitola 4
Návrh aplikace

Tato část se soustředí na podrobný popis klíčových cílů návrhu a struktury aplikace.

4.1 Cíle navrhu

Pro úspěšný vývoj webové aplikace je nezbytné navrhnout dvě základní komponenty, které společně zajišťují plnou funkcionalitu. Jedná se o komponentu uživatelského rozhraní, známou jako frontend, a komponentu serverové strany plánovácího systému, znamou jako backend.

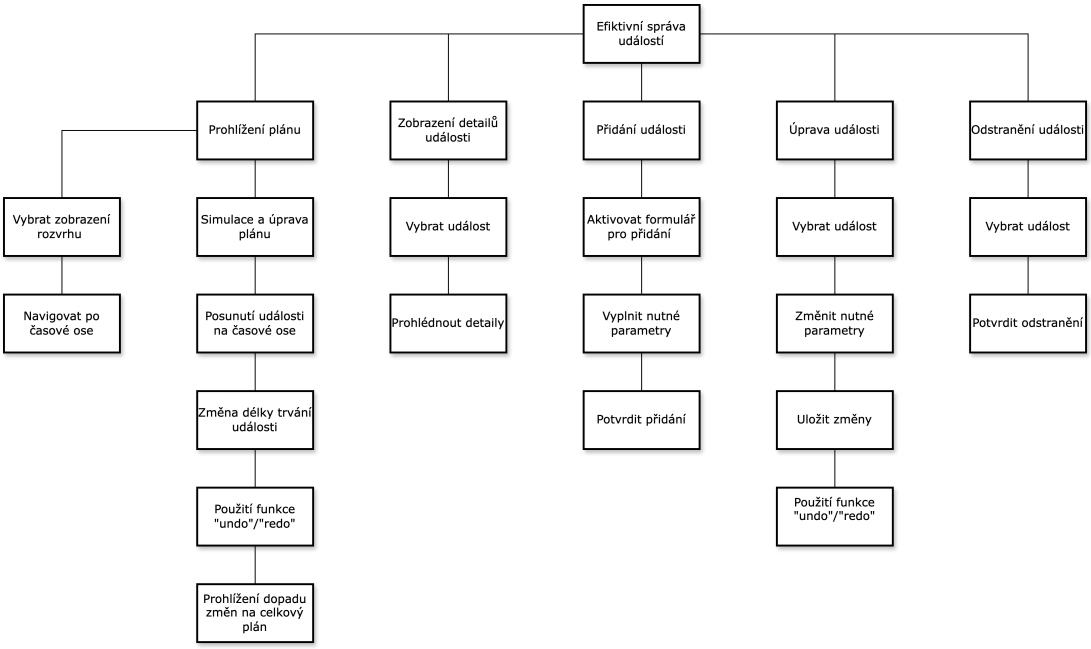
4.2 Návrh uživatelského rozhraní

4.2.1 Prototypování uživatelského rozhraní

Prototypování je klíčovým krokem v procesu návrhu aplikace, jehož cílem je vytvořit vizuální a interaktivní reprezentaci systému, která slouží jako základ pro další vývoj [2]. Prvním krokem v tomto procesu bylo specifikovat jednotlivé úkoly a efektivně je rozdělit.

Hierarchická analýza úkolů

Hierarchická analýza úkolů (HTA - Hierarchical task analysis) je metodika používaná k systematickému rozložení úkolů na menší, snadno spravovatelné části [22]. V kontextu vyvíjené aplikace HTA poskytuje detailní přehled hlavních úkolů a jejich podúkolů, které uživatelé vykonávají při správě a plánování událostí. HTA diagram na obrázku 4.1 zobrazuje hlavní úkoly a podúkoly, vizualizuje posloupnost kroků a interakce potřebné k dokončení různých funkcí, jako je např. přidávání, úprava událostí.



Obrázek 4.1: HTA diagram hlavních úkolů a podúkolů ve vyvíjené aplikaci.

Vizualizace uživatelského rozhraní

Na základě specifikovaných úkolů a jejich hierarchie byly vytvořeny drátěné modely (wireframes) v nástroji Figma[1]. Tyto drátěné modely slouží jako vizuální reprezentace a základ pro další vývoj aplikace.

Popis obecného layoutu a uživatelského rozhraní. Drátěné modely zobrazují čistý a přehledný layout, který zdůrazňuje klíčové informace. Klíčovým prvkem drátěných modelů je interaktivní tabulka, která umožňuje plánování na jednotlivých stojanech v hangáru. Tato tabulka je navržena tak, aby poskytovala komplexní přehled o událostech. Využívají se zde jasné vizuální prvky, jako jsou výrazné barvy pro různé události a snadno čitelné fonty, které zlepšují orientaci uživatele.

Interakce. Drátěné modely efektivně vizualizují následující funkce a možnosti:

- **Pohyblivost tabulky:** Tabulka plánu v drátěném modelu umožňuje horizontální posun vlevo a vpravo, což uživatelům umožňuje přizpůsobit zobrazení plánu podle svých potřeb.
- **Změna zobrazení tabulky plánu:** Drátěný model podporuje možnost změny zobrazení tabulky buď podle jednotlivých stojanů nebo podle operátorů.

[1] Figma. Více informace je na `https://www.figma.com/`

- **Vyhledávání podle WP:** Drátěný model naznačuje, že v aplikaci bude možné provádět vyhledávání podle názvu work package.
- **Přidávání nových událostí:** Drátěný model ukazuje možnost přidání nové události.
- **Přesun událostí mezi stojany:** Drátěný model demonstruje, jak mohou být události snadno a rychle přesunuty mezi různými stojany.

Vizuální ukázka drátěného modelu. Na obrázku v příloze C. Ukázka drátěného modelu je prezentován drátěný model, který reprezentuje klíčový element v uživatelském rozhraní aplikace — tabulku rozvrhu.

Vyhodnocení drátěných modelů

Drátěné modely, jako klíčový nástroj v rané fázi návrhu aplikace, byly představeny doménovému expertovi s cílem získat zpětnou vazbu a přesněji specifikovat funkce, které by vyvíjená aplikace měla obsahovat.

Specifikace potřeb. Během prezentace drátěných modelů doménovému expertovi byly podrobně probrány klíčové aspekty uživatelského rozhraní a interakce. Doménový expert poskytl cenné připomínky k funkčnosti aplikace, které byly zaměřeny především na následující oblasti:

- **Integrace pokročilého vyhledávání:** Bylo doporučeno začlenit pokročilé vyhledávací funkce, které umožní uživatelům vyhledávat nejen podle názvu události, ale také podle registrace letadla a operátora.
- **Zobrazení prodloužení plánu údržby:** Expert navrhl implementovat možnosti pro dynamické zobrazení prodloužení plánovaných údržeb.

Na základě těchto připomínek byly aktualizovány požadavky v kapitole 3.2. Požadavky na návrh a funkcionalitu aplikace.

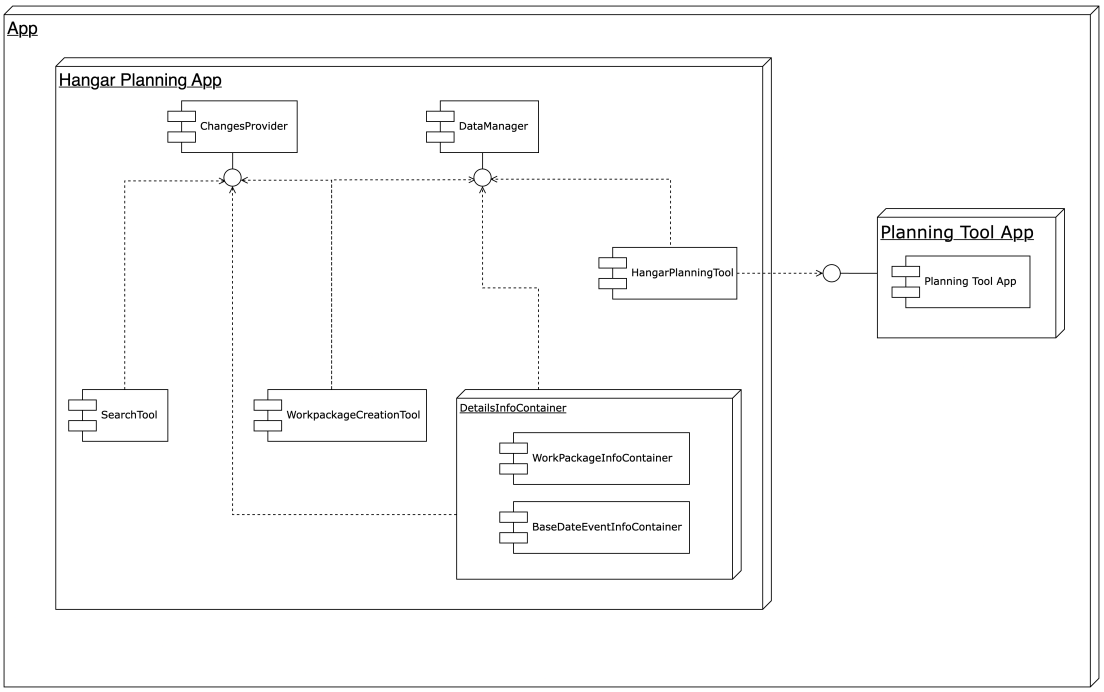
4.2.2 Moduly komponenty uživatelského rozhraní

Po schválení drátěných modelů a jejich revizi na základě cenné zpětné vazby od doménového experta se proces přesunul k další fázi — detailnímu návrhu uživatelského rozhraní. V této etapě byla vytvořena konkrétní struktura komponent, které se budou v aplikaci používat. Přesná specifikace komponent byla motivována potřebou zachovat koherentní a intuitivní rozhraní, které reflektuje všechny požadované funkce identifikované během evaluace drátěných modelů.
Následující obrázek 4.2 ilustruje, jak jsou jednotlivé moduly komponenty uživatelského rozhraní propojeny.

Popis modulů

Zde je uveden podrobnější popis jednotlivých modulů:

- **Hangar Planning App**
  - **ChangesProvider:** Tento modul je zásadní pro zachycení a provedení změn na globální úrovni v aplikaci. Zajišťuje aktualizace stavů a dat v reálném čase.
  - **DataManager:** Správa datových dotazů a zachycení chyb. Modul zpracovává všechny operace související s daty, včetně načítání, aktualizací a error handlingu.
  - **SearchTool:** Komponenta pro vyhledávání událostí v systému.
  - **WorkPackageCreatinTool:** Odpovědný za vytváření work packages.



Obrázek 4.2: UML Diagram komponent pro uživatelského rozhrání.

- **RightSideInfoContainer:** Zobrazuje detailní informace o jednotlivých událostech.
- **HangarPlanningTool:** Hlavní komponenta, která zobrazuje celkový rozvrh údržby.

- **PlanningToolApp:** Komponenta vybrané knihovny (viz. Výběr plánovací knihovny pro vyvíjenou aplikaci)

4.3 Návrh serverové části aplikace

Tato sekce je věnována návrhu serverové strany aplikace, kde je kladen důraz na doménový model a architekturu.

4.3.1 Doménový model

Pro návrh objektově-ontologického mapování je nezbytné představit doménový model aplikace. Tento proces byl zahájen zkoumáním stávající ontologie „CSAT-maintenance“. Jak již bylo zmíněno v sekci 3.5.1. Identifikace typů vstupních dat, klíčovou entitou v modelu je *Work package*. Tato entita představuje skupinu úkolů spojených s prováděním údržbového plánu.
Aby bylo možné zajistit úplnou funkčnost systému pro plánování událostí v hangáru, je nezbytné rozšířit stávající doménový model o nové atributy a třídy.

Rozšíření doménového modelu

Rozšíření jsou navrženy tak, aby odpovídaly specifickým požadavkům systému. Tato rozšíření zahrnuje:

1. **Přídání nových vlastností**
   - **background-color** - Tato vlastnost umožňuje uchovávat barvu, která bude reprezentovat událost v uživatelském rozhraní.
   - **reserve-time** - Atribut sloužící jako rezerva pro událost.
   - **marks** - Poznámky k jednotlivým událostem.
   - **bay** - Stojan, na kterém se událost vykonává.
2. **Definice znalosti o stojanech**
   Aby bylo možné přiřadit jednotlivé události k specifickým stojanům, je nezbytné definovat a implementovat znalostní strukturu, která bude zahrnovat informace o různých typech stojanů v hangáru a jejich specifikacích.
3. **Definice třídy pro denní události**
   Dále je zapotřebí model rozšířit o denní události, pro splnění **požadavku** na plánování různých typů události.

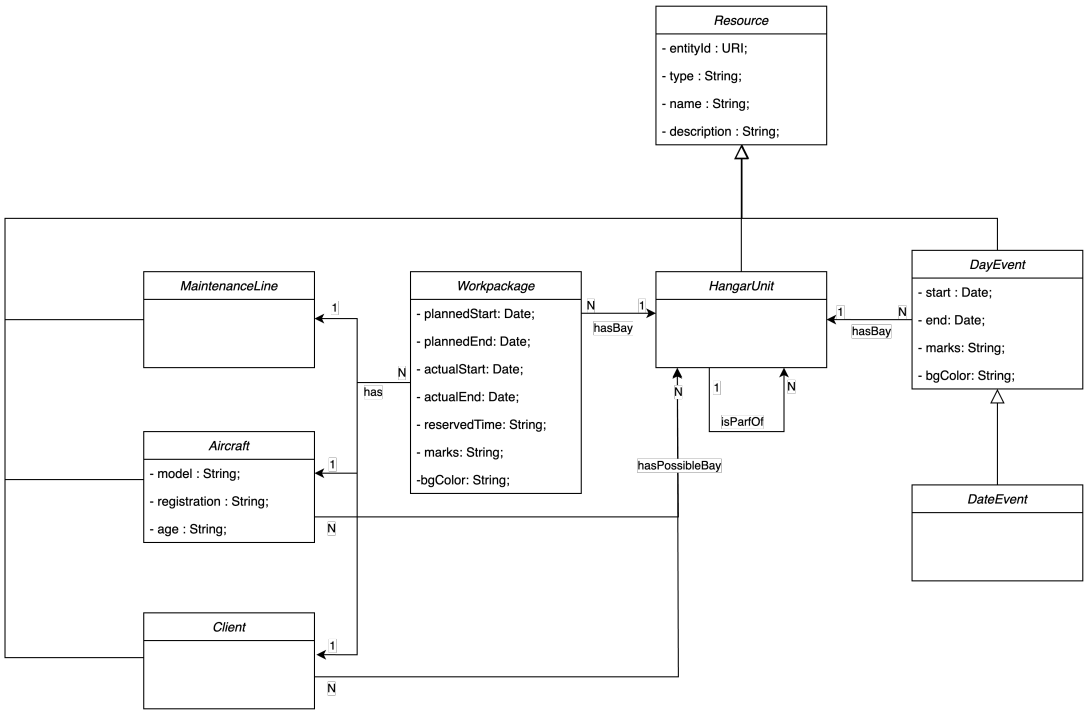
4.3.2 Datová struktura plánovacího systému

Po rozšíření doménového modelu je nyní možné navrhnout datovou strukturu pro aplikaci, která definuje, jak budou data reprezentována a manipulována na úrovni aplikace. Datová struktura aplikace je vyjádřena pomocí diagramu tříd na obrázku 4.3.

Slovní popis diagramu tříd datové struktury

Níže je uveden detailnější popis tříd, které jsou zodpovědné za různé funkce v rámci aplikace:

- Třídy *Workpackage*, *Aircraft*, *Client* a *MaintenanceLine* reflektují již existující model, který byl podrobně popsán v Identifikace typů vstupních dat. Tyto třídy představují základní komponenty systému a jsou nezbytné pro definování a správu klíčových entit.
- Třída *DayEvent* reprezentuje denní událost, která je vyjádřena daty začátku, konce a příslušným stojanem.
- Třída *DateEvent* představuje událost vázanou na konkrétní datum.



Obrázek 4.3: UML diagram tříd datové struktury.

4.3.3 Architektonický styl serverové strany

Vzhledem k tomu, že serverová strana aplikace není příliš rozsáhlá, bylo rozhodnuto implementovat architekturu v monolitickém stylu. Tento přístup umožňuje jednodušší vývoj a správu aplikace vzhledem k její menší velikosti a méně komplexním požadavkům [11].
Pro zajištění čistého kódu a efektivního oddělení zodpovědností bude serverová strana aplikace využívat vícevrstevnou architekturu, konkrétně rozdělení na *Vrstvu prezentace*, *Vrstvu servisu* a *Vrstvu persistence* [8]. Tato struktura umožňuje:

- **Vrstva prezentace (Presentation layer)** - Tato vrstva je zodpovědná za interakci s uživateli aplikace, zpracování uživatelských vstupů a zobrazení odpovídajících výstupů. Slouží jako most mezi uživatelem a logikou aplikace, poskytuje uživatelské rozhraní.
- **Vrstva servisu (Service layer)** - Srdce logiky aplikace, vrstva servisu zpracovává byznysovou logiku a pravidla. Spravuje transakce, koordinuje operace mezi uživatelským rozhraním a databázovou vrstvou, a poskytuje potřebné služby pro zpracování dat.
- **Vrstva persistence (Persistence layer)** - Zodpovídá za trvalé ukládání dat aplikace. Umožňuje vytváření, čtení, aktualizaci a mazání dat v databázi. Tato vrstva abstrahuje technické aspekty přístupu k datům a poskytuje jednoduché rozhraní pro ostatní vrstvy aplikace.

Zajištění komunikace s uživatelským rozhraním

Zajištění efektivní komunikace mezi uživatelským rozhraním (UI) a serverovou stranou je zásadním prvkem v architektonickém designu aplikace. K tomuto účelu je implementován architektonický styl REST, který umožňuje UI komunikovat s serverem prostřednictvím HTTP metod. Komunikace a data budou přenášena skrze RESTful API, které využívá následující klíčové principy REST [5]:

- **Uniformní rozhraní**: REST API vyžaduje, aby všechny požadavky na stejný zdroj vypadaly stejně, nezávisle na původu požadavku.
- **Oddělení zodpovědností**: V REST architektuře musí být aplikace klienta a serveru úplně nezávislé.
- **Bezstavovost**: REST API je bezstavové, což znamená, že každý požadavek musí obsahovat všechny informace nutné pro jeho zpracování.
- **Cacheovatelnost**: Zdroje by měly být, kde je to možné, cacheovatelné na straně klienta nebo serveru.
- **Vrstvená systémová architektura**: V REST API by volání a odpovědi měla procházet různými vrstvami.
- **Kód na vyžádání (volitelné)**: REST API obvykle odesílá statické zdroje, ale může také zasílat spustitelný kód, který se spouští na vyžádání.

4.4 Integrace do existující infrastruktury

Existující systém[2] každodenně shromažďuje historická data, která jsou ukládána do CSV souborů a následně transformována do kontextů[3] v databázi. Tato data jsou využívána pro generování různých statistik a dashboardů.

Pro zajištění plné funkčnosti vyvíjené aplikace pro hangárové plánování je klíčové vytvořit nový, samostatný kontext v databázi. Tento krok umožní efektivní správu a manipulaci s daty specifickými pro hangárové plánování, aniž by došlo k zásahům do existujících datových struktur. Tímto způsobem lze zajistit, že nový systém bude integrován do stávající infrastruktury bez negativního ovlivnění funkcionality a stability existujících systémových komponent.

Vzhledem k tomu, že existující systém průběžně aktualizuje data, je potřeba definovat rozhraní umožňující aktualizaci dat v nově vytvořeném kontextu pro hangárové plánování.

[2] KBSS at FEE CTU, Improving effectiveness of aircraft maintenance planning and execution. Více informace je na `https://kbss.felk.cvut.cz/projects/doprava2020-csat/`

[3] Ontotext. GrapgDB. Storage. Více informace je na `https://graphdb.ontotext.com/documentation/10.6/storage.html#storage`

Celá existující infrastruktura je kontejnerizovaná, což zjednodušuje integraci nových komponent. Pro zahrnutí vyvíjené aplikace do této struktury bude v následující sekci 4.5 navržena kontejnerizace vyvíjené aplikace. Tyto nové kontejnery bude nutné integrovat do stávajícího systému kontejnerů, aby bylo zajištěno bezproblémové začlenění do infrastruktury.

4.5 Nasazení

4.5.1 Obecný přehled

Zajištění hladké integrace a efektivního fungování aplikace vyžaduje nasazení v různých prostředích, jako jsou testovací a produkční. To umožňuje oddělit testování s testovacími daty od finálního nasazení v produkčním prostředí.

4.5.2 Nasazení pro testování

Během vývoje aplikace a implementace nových funkcí je klíčové provádět průběžné nasazení (Continuous Deployment, CD) těchto funkcí. Tento proces umožňuje, aby nové funkce mohly být rychle prezentovány a otestovány [7]. Testovací nasazení zahrnuje implementaci uživatelského rozhraní spolu s testovacími daty, což umožňuje efektivní validaci nových funkcionalit ve skutečných uživatelských scénářích.

4.5.3 Produkční nasazení

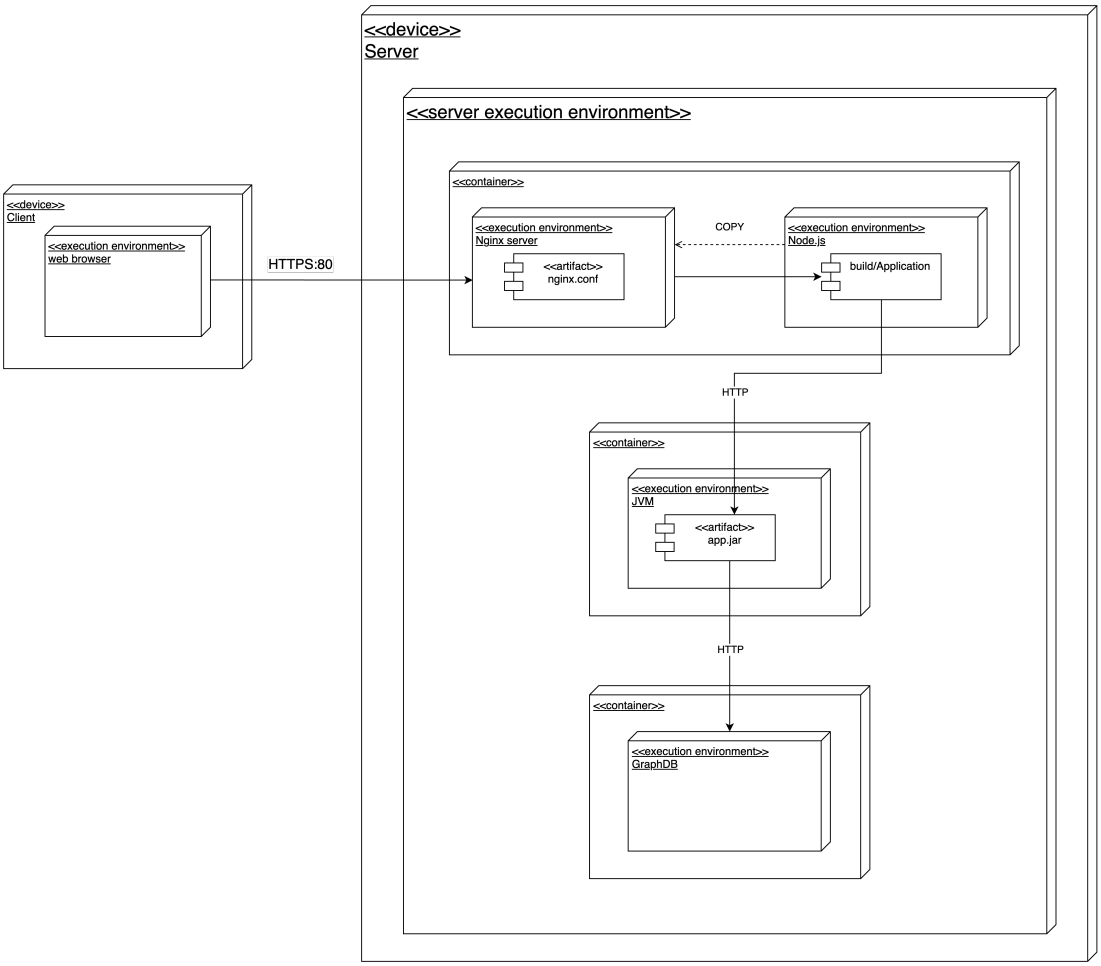
Po úspěšném otestování a ověření funkcí v testovacím prostředí dochází k plnohodnotnému nasazení systému na produkční server. Toto nasazení zahrnuje finální verzi aplikace, která byla pečlivě testována a je připravena pro použití v reálném prostředí.

Kontejnerizace

Kontejnerizace umožňuje balení aplikace a jejích závislostí do izolovaných kontejnerů. Tento přístup zjednodušuje nasazení a zvyšuje přenositelnost mezi různými vývojovými a produkčními prostředími. Kontejnery jsou lehké, rychle se spouštějí a poskytují konzistentní prostředí pro aplikaci, což minimalizuje ”funguje na mé mašině”problémy [6]. Integrace kontejnerizované aplikace do existujícího systému je obvykle jednodušší díky standardizaci, kterou kontejnery poskytují. Nezávislost kontejnerů na hostitelském operačním systému znamená, že mohou být snadno nasazeny na jakémkoliv serveru podporujícím kontejnerovou platformu.

Přehled diagramu nasazení

Níže je uveden Obrázek č. 4.4. UML diagram nasazení do produkcního prostředí, který zobrazuje rozložení systemových komponent, jejich vzájemné propojení a interakce s dalšími systémovými komponentami, jako je databáze.



Obrázek 4.4: UML diagram nasazení do produkcního prostředí.

Kapitola 5
Implementace

Tato kapitola podrobně rozebírá proces vývoje a zpracování požadavků na webovou aplikaci. Detailně jsou zde popsány použité technologie, softwarové knihovny a kroky vedoucí k nasazení aplikace. Specifikem tohoto vývojového procesu je jeho opírání se o technologie sémantického webu, které jsou podrobněji rozebrány v sekci 2. Úvod do problematiky.

5.1 Technologie

V této sekci je podrobně popsán výběr a využití technologií pro vývoj, sestavení, nasazení a správu verzí vyvíjené aplikace.

5.1.1 Frontend

Kromě základních technologií, jako jsou HTML[1] a CSS[2], které tvoří základ každé webové stránky, budou využívány níže uvedené technologie.

Hlavní vývojové frameworky

React. React je open-source JavaScript knihovna určená pro vývoj uživatelských rozhraní, zejména pro jednostránkové aplikace. Byl vyvinut společností Facebook a je široce používán pro tvorbu interaktivních webových a mobilních aplikací. React umožňuje vývojářům vytvářet velké webové aplikace, které mohou dynamicky měnit zobrazený obsah bez nutnosti načítání celé stránky [18].

Odůvodnění výběru. Výběr Reactu jako základní technologie pro frontend byl motivován několika klíčovými faktory:

1. Základním důvodem je požadavek na použití Reactu v projektu (viz. 1 Nefunkční požadavky)
2. Knihovna pro plánování, která byla vybrána pro integraci do aplikace (viz. 3.5.3 Výběr plánovací knihovny pro vyvíjenou aplikaci, je rovněž postavena na Reactu.

[1] HTML. W3C. Více informace je na `https://www.w3.org/html/`
[2] CSS. W3C. Více informace je na `https://www.w3.org/Style/CSS/Overview.en.html`

Komponenty a knihovny

Moment.js. Moment.js[3] je JavaScriptová knihovna, která umožňuje snadné parsování, validaci, manipulaci a formátování dat. Umožňuje převádět textové řetězce na objekty datumů a času, provádět operace s daty jako přičítání, odčítání či porovnávání a nabízí rozsáhlé možnosti pro formátování dat.

Využití v aplikaci. Moment.js je použit k manipulaci a zobrazení dat a časů v aplikaci, zajišťuje konzistenci formátování a manipulace s daty napříč celou aplikací.

React Date Picker. React Datepicker[4] je komponenta pro React, která umožňuje uživatelům snadný výběr data přes interaktivní kalendářní rozhraní. Tato knihovna nabízí širokou škálu konfigurací a možností přizpůsobení, včetně nastavení formátu data, výběru časového rozsahu a lokalizace.

Využití v aplikaci. React Datepicker je klíčová komponenta pro výběr data pro události, poskytuje uživatelům intuitivní rozhraní pro zadávání datumů bez chyb.

React Select. React Select[5] je flexibilní a konfigurovatelná komponenta pro React, která umožňuje uživatelům snadno vybírat jednu nebo více položek z rozbalovacího seznamu. Tato knihovna poskytuje širokou škálu možností přizpůsobení, včetně multi-select možností, asynchronního načítání, vyhledávání a mnoha dalších.

Využití v aplikaci. React Select je využíván k definování a výběru specifikovaných parametrů událostí, zajišťuje flexibilitu a širokou možnost konfigurace.

5.1.2 Backend

Hlavní programovací platforma

Java. Java je programovací jazyk a výpočetní platforma, která byla poprvé vydána společností Sun Microsystems v roce 1995. Dnes Java je klíčovou součástí mnoha digitálních služeb a aplikací, poskytující spolehlivou platformu pro jejich vývoj a provoz [17].

Odůvodnění výběru. Výběr Java jako základní platformy je určen požadavkem (viz. 2 Nefunkční požadavky)

Frameworky a knihovny

Maven. Maven[6] je nástroj pro automatizaci a správu procesů sestavování v projektech založených na Java. Jeho hlavními cíli je standardizovat způsob

[3] Moment.js. Více informace je na `https://momentjs.com/`
[4] React Datepicker. Více informace je na `https://reactdatepicker.com/`
[5] React Select. Více informace je na `https://react-select.com/home`
[6] Maven. Více informace je na `https://maven.apache.org/`

sestavování projektů, jasně definovat obsah projektů, usnadnit publikaci informací o projektech a umožnit sdílení JAR souborů mezi více projekty.

Odůvodnění výběru. Hlavním důvodem pro výběr je předchozí zkušenost s tímto nástrojem, což umožňuje efektivnější a rychlejší vývoj.

Využití v aplikaci. Maven je využíván pro správu závislostí a sestavení projektu.

Spring Boot. Spring Boot je vývojový framework, který je součástí širšího ekosystému Spring[7] a slouží k usnadnění procesu vývoje a nasazení aplikací postavených na platformě Java. Byl představený organizací Pivotal Software (nyní součást VMware) a má za cíl zredukovat množství konfigurační práce spojené s nastavováním aplikací založených na Spring frameworku.

Odůvodnění výběru. Hlavním důvodem pro výběr je předchozí zkušenost.

Využití v aplikaci. Spring Boot je nutný pro server-side zpracování dotazů.

JOPA. JOPA (Java OWL Persistence API) představuje framework pro persistenci OWL v Javě, jehož hlavním cílem je poskytnout efektivní a formalizované objektově-ontologické mapování. Tento systém umožňuje realizaci transakcí, přístup k různým repozitářům a generování Linked Data [10].

Využití v aplikaci. JOPA je používána pro správu ontologií v aplikaci.

Kontejnerizace

Docker. Docker[8] je otevřená platforma určená pro vývoj, distribuci a spouštění aplikací v izolovaných kontejnerech, což umožňuje rychlejší dodání softwaru a oddělení aplikací od infrastruktury. Kontejnery Dockeru jsou lehké a obsahují vše potřebné pro běh aplikace, což zajišťuje jejich nezávislost na hostitelském systému.

Odůvodnění výběru. Stávající infrastruktura projektu již využívá Docker pro kontejnerizaci a správu aplikací. Použití Dockeru pro nový modul zajišťuje konzistenci a kompatibilitu s celkovým systémem.

5.1.3 Kontinuální nasazení

Pro automatizaci vývoje a nasazení webových aplikací je využívána kombinace nástrojů GitHub a Netlify, které společně zajišťují efektivní proces kontinuálního nasazení.

[7] Spring framework. Více informace je na `https://spring.io/`
[8] Docker. Více informace je na `https://docker.com/`

GitHub. GitHub[9] je platforma pro hostování kódu, která slouží pro správu verzí a spolupráci. Umožňuje pracovat společně na projektech z jakéhokoli místa. GitHub je základním nástrojem pro verzování a správu zdrojového kódu aplikace.

Netlify. Netlify[10] je platforma pro webhosting a automatizaci vývoje webových aplikací. Poskytuje nástroje a služby pro rychlý a efektivní vývoj moderních webových projektů, od statických stránek až po složité aplikace. Netlify zahrnuje automatizované procesy pro nasazení, optimalizaci a škálování aplikací v prostředí navrženém pro vysoký výkon a bezpečnost.

5.2 Vývoj backendových služeb

Tato sekce se zabývá klíčovými aspekty vývoje backendových služeb pro vyvíjenou plánovací komponentu. Podrobně jsou zde popsány procesy, které jsou nezbytné pro efektivní fungování serverové části aplikace, od základů datového modelu až po implementaci komunikačních rozhraní.

5.2.1 Základní nastavení

V úvodní fázi vývoje bylo klíčové provést inicializaci projektu, která byla realizována prostřednictvím nástroje Spring Initializr [11]. Tento počáteční krok zahrnoval konfiguraci základních parametrů a integraci nezbytných závislostí. Součástí přípravy bylo také vytvoření struktury projektu odpovídající normám třívrstvé architektury, což zásadně přispívá k systematickému rozdělení aplikace na prezentační, aplikační a datovou vrstvu. Taková struktura je fundamentální pro efektivní vývoj v dalších fázích vývoje backendové části systému.

5.2.2 Rozšíření domenového modelu

V této podkapitole je popsáno rozšíření datového modelu, jak bylo navrženo v kapitole 4.3.1. Rozšíření doménového modelu. Tento krok zahrnuje přidání nových tříd a atributů do doménového modelu.
Proces rozšíření zahrnuje definici nových entit a jejich vztahů v rámci existující databáze. Je nezbytné zajistit, aby všechny nové entity byly správně integrovány do systému, včetně vazeb a omezení, které zajišťují konzistenci a integritu dat.
Datový model byl rozšířen s využitím jazyka Turtle (TTL, viz 2.5.1. TTL). Následují příklady konkrétních rozšíření modelu spolu s popisem jejich účelu pomoci komentáře (rdfs:comment). Pro lepší čitelnost jsou použity následující prefixy namísto plných URI:

[9] GitHub. Více informace je na `https://github.com/`
[10] Netlify. Více informace je na `https://www.netlify.com/`
[11] Spring Initializr. Podrobnější informace naleznete na `https://start.spring.io/`

Příklad 5.1: Použité prefixy

```
@prefix owl: http://www.w3.org/2002/07/owl# .
@prefix rdfs: http://www.w3.org/2000/01/rdf-schema# .
@prefix cm: http://onto.fel.cvut.cz/ontologies/csat-
   ↪ maintenance/ .
@prefix hmp: http://onto.fel.cvut.cz/ontologies/hangar-
   ↪ maintenance-planning/ .
```

Příklad 5.2: Definice tříd

```
hmp:hangar-unit rdf:type owl:Class ;
    rdfs:comment "Reprezentuje zdroj spojený se stojany
       ↪  v hangáru."@cs .

hmp:one-day-event rdf:type owl:Class ;
    rdfs:comment "Reprezentuje denní událost"@cs .
```

Příklad 5.3: Definice vlastností

```
hmp:background-color rdf:type owl:DatatypeProperty ;
    rdfs:range xsd:string ;
    rdfs:domain cm:workpackage ;
    rdfs:domain hmp:one-day-event ;
    rdfs:comment "Reprezentuje barvu pozadí konkrétní
       ↪ události."@cs .

hmp:reserved-time rdf:type owl:DatatypeProperty ;
    rdfs:range xsd:int ;
    rdfs:domain cm:workpackage ;
    rdfs:comment "Reprezentuje rezervovanou dobu v
       ↪ dnech pro work package."@cs .

hmp:marks rdf:type owl:DatatypeProperty ;
    rdfs:range xsd:string ;
    rdfs:domain hmp:one-day-event ;
    rdfs:domain cm:workpackage ;
    rdfs:comment "Reprezentuje poznámky spojené se
       ↪ zdrojem."@cs .

hmp:has-group rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf <http://onto.fel.cvut.cz/
       ↪ ontologies/hangar-maintenance-planning/is-
       ↪ part-of-bay-group> ;
    owl:inverseOf hmp:is-part-of-bay-group .
    rdfs:comment "Specifikuje skupinu, do které daný
       ↪ objekt spadá."@cs
```

Příklad 5.4: Definice vztahů

```
hmp:has-bay rdf:type owl:ObjectProperty ;
    rdfs:range hmp:hangar-unit ;
    rdfs:domain cm:workpackage ;
    rdfs:domain hmp:one-day-event ;
    rdfs:comment "Specifikuje hangár spojený s
       ↪ konkrétní událostí."@cs .

 hmp:is-part-of-group rdf:type owl:ObjectProperty ;
    rdfs:range hmp:hangar-unit ;
    rdfs:domain hmp:hangar-unit ;
    rdfs:comment "Vlastnost pro hierarchickou strukturu
       ↪  hangárových jednotek."@cs .

hmp:has-possible-bay rdf:type owl:ObjectProperty ;
  rdfs:range hmp:hangar-unit ;
  rdfs:domain hmp:client  ;
  rdfs:domain cm:aircraft ;
  rdfs:comment "Specifikuje stojan pro konkrétní  zdroj
     ↪ "@cs .
```

5.2.3 Implementace objektově-ontologického mapování

Po rozšíření doménového modelu bylo možné přistoupit k vytvoření datového modelu a jeho mapování. Jak bylo specifikováno v 5.1 Technologie, pro objektově-ontologické mapování byl vybrán framework JOPA, který umožňuje definovat třídy a vlastnosti pomocí anotací. Níže je uveden příklad 5.5 entity **Workpackage**, který demonstruje, jak lze v rámci JOPA frameworku aplikovat anotace pro mapování ontologických tříd a vlastností na Java třídy. Podobným způsobem byla navržena celá datová struktura, která byla představena v sekci 4.3.2. Datová struktura plánovacího systému.

Příklad 5.5: Definice entity Workpackage v JOPA.

```
@OWLClass(iri = Vocabulary.s_c_workpackage)
public class Workpackage implements Serializable {
    @OWLObjectProperty(iri = Vocabulary.s_p_is_repair_of)
    protected Aircraft aircraft;
    @OWLObjectProperty(iri = Vocabulary.s_p_has_client)
    protected Client client;
    @OWLObjectProperty(iri = Vocabulary.s_p_has_responsible_maintenance_line)
    protected MaintenanceLine maintenanceLine;
    @OWLObjectProperty(iri = Vocabulary.s_p_has_bay)
    protected HangarUnit bay;
    @OWLDataProperty(iri = Vocabulary.s_p_workpackage_start_time)
    protected LocalDateTime startTime;
    @OWLDataProperty(iri = Vocabulary.s_p_workpackage_end_time)
    protected LocalDateTime endTime;
    @OWLDataProperty(iri = Vocabulary.s_p_workpackage_scheduled_start_time)
    protected LocalDateTime plannedStartTime;
    @OWLDataProperty(iri = Vocabulary.s_p_workpackage_scheduled_end_time)
    protected LocalDateTime plannedEndTime;
    @OWLDataProperty(iri = Vocabulary.s_p_reserved_time)
    protected String reservedTime;
    @OWLDataProperty(iri = Vocabulary.s_p_marks)
    protected String marks;
    @OWLDataProperty(iri = Vocabulary.s_p_background_color)
    protected String bgColor;
    @Transient
    protected URI graph;
}
```

5.2.4 Konstrukce REST API

Následně po definici celého datového modelu a implementaci objektově-ontologického mapování byla rozpracována persistenční vrstva využívající návrhový vzor DAO [12].(Data Access Object). Tato vrstva zahrnuje všechny nezbytné operace pro práci s daty, tedy CRUD (Create, Read, Update, Delete) operace, které zajišťují manipulaci s entitami v databázi.

Byznysová logika byla implementována v rámci servisních tříd, které abstrahují logiku manipulace s daty od jejich reprezentace a ukládání. Tyto servisní třídy byly následně využity pro definici koncových bodů (endpoints), které zpřístupňují funkce aplikace jako webové služby.

Tabulka 5.1 poskytuje kompletní přehled REST API endpointů, které aplikace poskytuje. Každý endpoint je specifikován odpovídající HTTP metodou, URL adresou, typem operace, kterou provádí, a krátkým popisem jeho funkce. Všechny endpointy nepracují přímo s entitami naimplementovanými v sekci 5.2.3. Implementace objektově-ontologického mapování. Pro posílání a přijímání dat k zpracování se používá pattern DTO[13] (Data Transfer Object), který zajišťuje, že citlivé informace nebo údaje, které nejsou nutné k zobrazení, jsou skryty.

Tabulka 5.1: Přehled REST API endpointů

Metoda	Endpoint	Popis
GET	/api/events?includeAll=true/false	Vrátí seznam událostí. Parametr includeAll určuje, zda je nutné vrátit události všech typů nebo jen work packages.
POST	/api/workpackage	Přidá nový work package do systému.
DELETE	/api/workpackage	Smaže work package ze systému.
PATCH	/api/day-event	Vytváří a aktualizuje denní event.
DELETE	/api/day-event	Smaže denní event.
GET	/api/units	Vrátí seznam hangárových jednotek (stojanů).
GET	/api/clients	Vrátí seznam klientů.
GET	/api/aircrafts	Vrátí seznam letadel.
GET	/api/maintenanceLine	Vrátí seznam údržbových linek.

5.3 Vývoj frontendové části aplikace

Implementace frontendové části aplikace je klíčovou součástí vývoje, která přímo ovlivňuje uživatelskou zkušenost. Tato sekce podrobně popisuje postup implementace, modifikace vybraných knihoven a integrace navržených modulů.

[12]DAO Pattern. Podrobnější informace naleznete na `https://www.oracle.com/java/technologies/data-access-object.html`

[13]DTO Pattern. Podrobnější informace naleznete na `https://www.okta.com/identity-101/dto/`

5.3.1 Základní nastavení

Implementace klíčových funkcí projektu začíná základním nastavením, které zahrnuje vytvoření projektu pro plánovací komponentu a integraci nadstavby knihovny 3.5.2. react-calendar-timeline.
Začátkem procesu bylo vytvoření forku [3] původní nadstavby knihovny. Tímto krokem byl zajištěn základ pro další modifikace a zároveň možnost sdílení výsledků práce.
Vzhledem k předpokládané nutnosti provádět úpravy přímo v zdrojovém kódu knihovny, bylo rozhodnuto importovat knihovnu přímo do projektu namísto jejího přidání jako závislosti.

5.3.2 Modifikace vybrané knihovny

Pro účely projektu bylo nutné modifikovat nadstavbu vybrané knihovny, což vyžadovalo několik klíčových kroků k zajištění její plné funkčnosti a integrace do vyvíjené aplikace.

Expozice API. Původní knihovna poskytovala API, jak je popsáno v sekci 3.5.2. react-calendar-timeline, ale její nadstavba toto API ve výchozím stavu nenabízela. Prvním krokem modifikace bylo proto vystavení tohoto API, aby bylo možné knihovnu efektivněji integrovat a využívat její plný potenciál ve vyvíjené aplikaci.

Úpravy a refaktoring kódu. Během začlenění knihovny do projektu byla rozpoznána nutnost několika důležitých úprav a doplnění funkcionalit, aby knihovna vyhovovala specifickým potřebám vyvíjené aplikace. Kromě toho, byl proveden i refaktoring kódu. Tento proces zahrnoval extrakci klíčových metod z obsáhlejších bloků kódu, což zlepšilo jejich přehlednost a umožnilo jejich efektivnější znovupoužití.

Publikace knihovny. Po dokončení všech úprav byla upravená verze knihovny publikována na npm[14] jako nezávislý balíček, což usnadnilo její další použití jako závislosti v rámci projektu a potenciálně i v dalších projektech.

5.3.3 Implementace navržených modulů

Tato sekce se zaměřuje na popis implementace modulů uživatelského rozhrání, které byly navrženy ve fázi 4.2.2. Moduly komponenty uživatelského rozhraní. Implementace těchto modulů byla provedena s přímým dodržením specifikací, které byly stanoveny během návrhové fáze. Důraz byl kladen na zajištění, že každý modul splňuje své specifické funkční požadavky a integruje se harmonicky do celkového systému.

Každý modul byl implementován s cílem maximálně odpovídat funkcím,

[14] npm. Více informace je na `https://www.npmjs.com/`

které byly požadovány pro splnění konkrétních potřeb uživatelů. Proces implementace zahrnoval:

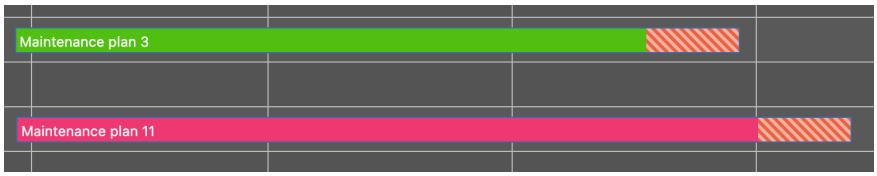
- **Precizní kódování:** Vývoj kódu, který přímo reflektuje funkčnost popsanou v 4.2.2. Moduly komponenty uživatelského rozhraní.
- **Integrace s dalšími systémovými komponentami:** Zajištění, že moduly efektivně komunikují s ostatními částmi systému pro ucelené řešení.

5.4 Implementace požadavků

Tato část se zaměřuje na detailní popis procesu implementace jednotlivých požadavků, které byly stanoveny pro systém. Cílem je poskytnout přehled o tom, jak byly specifické požadavky transformovány do pracovních řešení.

5.4.1 FR 1. Zvýraznění rezervy události

Tento požadavek je splněn prostřednictvím API poskytovaného vybranou knihovnou (viz 3.5.3. Výběr plánovací knihovny pro vyvíjenou aplikaci), které umožňuje definovat vlastní funkci pro úpravu zobrazení jednotlivých událostí. Pro zvýraznění rezervy byla implementována speciální funkce, která pomocí CSS pravidel aplikuje na rezervu událostí určené barvy. Toto zvýraznění je navrženo tak, aby se dynamicky přizpůsobovalo změnám velikosti časové osy, čímž je zajištěno, že vizuální reprezentace rezervy vždy korektně odpovídá aktuálnímu měřítku zobrazení. Níže na obrázku 5.1. Příklad zobrazení rezervy události work package je uveden příklad rezervy události (rezerva je zvýrazněna zaškrtnutými pruhy):

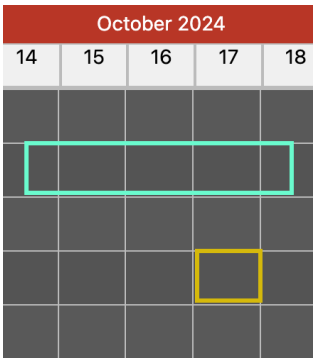


Obrázek 5.1: Příklad zobrazení rezervy události work package.

5.4.2 FR 2. Zobrazení různých typů událostí

Splnění tohoto požadavku je zajištěno využitím stejného rozhraní API, které bylo využito pro 5.4.1. FR 1. Zvýraznění rezervy události. API umožňuje dynamickou modifikaci vizuálního zobrazení událostí na časové ose. Pro každý typ události je definována specifická vizuální reprezentace, což umožňuje uživatelům rychle rozlišit mezi různými typy událostí na základě jejich vizuálního stylu.
Jak bylo znázorněno na obrázku 5.1. Příklad zobrazení rezervy události work package, jednotlivé work packages jsou zobrazeny výrazně odlišným vizuálním

stylem, který usnadňuje jejich rozpoznání a orientaci v plánu.
Níže na obrázku 5.7. Příklady plánování různých typů událostí jsou uvedeny příklady vizuálního zobrazení pro ostatní typy událostí, které aplikace podporuje.



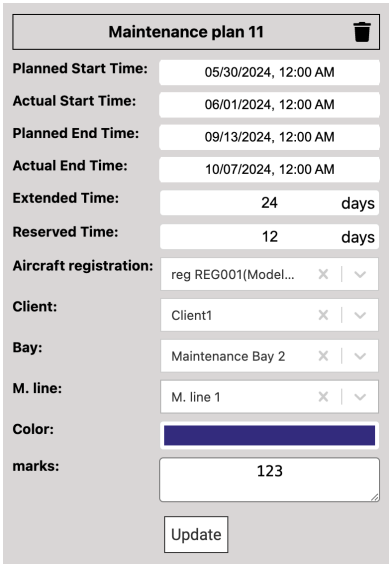
(a) : Zobrazení vícedenní události.

(b) : Zobrazení události na konkrétní datum

Obrázek 5.2: Příklady zobrazení denní události a události na konkrétní datum.

5.4.3 FR 3. Zobrazení detailů události

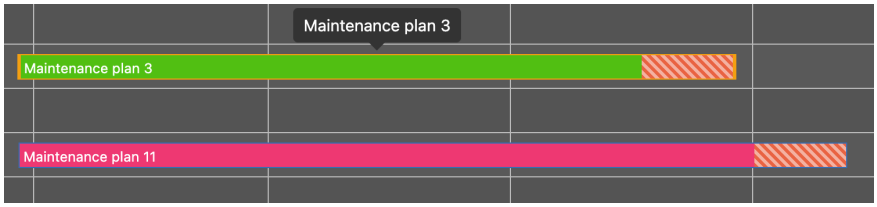
Detaily jednotlivých událostí jsou zobrazovány pomocí bočního panelu, který se dynamicky přizpůsobuje podle typu události. Tento panel poskytuje veškeré relevantní informace týkající se vybrané události. Na níže uvedeném obrázku je demonstrace bočního panelu s detailními informacemi pro work package s názvem ”Maintenance Plan 11”.



Obrázek 5.3: Příklad zobrazení detailů události work package.

5.4.4 FR 4. Zvýraznění události

Implementace tohoto požadavku využívá stávající funkcionality plánovací komponenty, která umožňuje dynamické zvýraznění události vybrané uživatelem. Specifická interakce, jako je dvojitý klik myší, aktivuje vizuální zvýraznění události a dochází změně barvy pozadí vybrané události na barvu definovanou atributem **selectedBgColor**.
Na obrázku 5.4. Příklad zvýraznění události je demonstrativní příklad, kde událost s názvem "Maintenance Plan 3"je zvýrazněna po dvojitém kliknutí uživatelem.



Obrázek 5.4: Příklad zvýraznění události.

5.4.5 FR 5. Zobrazení rozvrhu

Implementace tohoto požadavku je zajištěna využitím jádra použité plánovací knihovny, která slouží jako hlavní komponenta pro zobrazení událostí. Plánovací komponenta využívá dvě základní struktury dat pro správu a zobrazení událostí:

- **groups** - Pole pro třídy událostí. Tyto objekty jsou charakterizovány jedinečnými identifikátory (ID) a slouží k kategorizaci událostí do logických celků.
- **items** - Pole pro události. Každá událost je specifikována datem začátku, konce a identifikátorem skupiny, ke které náleží.

Každý objekt z těchto polí je převeden do odpovídající formy, která je požadována plánovací komponentou, přičemž nepovinné atributy jsou doplněny na standardní hodnoty, což zajišťuje konzistenci datových struktur v rámci celé aplikace.
Příklad struktury dat 5.6, který je uveden níže, demonstruje zařazení jednotlivých plánů údržby do specifických skupin. Plán údržby označený jako "Maintenance Plan 1"je přiřazen do skupiny 1, která je definována jako "Bay 1". Obdobně, plán údržby "Maintenance Plan 2"je přiřazen do skupiny 2. Toto zařazení ilustruje vazbu mezi plány údržby a odpovídajícími hangárovými jednotkami, kde se údržba realizuje.

Příklad 5.6: Ukázka JSON struktury dat plánovací komponenty.

```
{
  "groups": [
    {
      "id": 1,
      "name": "Bay 1"
    },
    {
      "id": 2,
      "name": "Bay 2"
    }
  ],
  "items": [
    {
      "id": 1,
      "group": 1,
      "name": "Maintenance Plan 1",
      "start": "2024-02-20",
      "end": "2024-03-20"
    },
    {
      "id": 2,
      "group": 2,
      "name": "Maintenance Plan 2",
      "start": "2024-03-20",
      "end": "2024-04-20"
    }
  ]
}
```

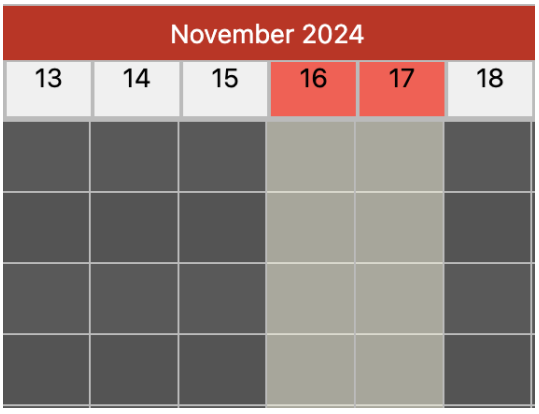
5.4.6 FR 6. Označení aktuálního času

Pro splnění tohoto požadavku byla využita specifická komponenta, která v rozhraní aplikace dynamicky označuje aktuální čas. Tato komponenta je implementována jako časová linie, která prochází vertikálně přes všechny stojany.
Na obrázku 5.5. Označení aktuálního času je ilustrováno, jak se v aplikaci zobrazuje aktuální čas pomocí modré linie.

Obrázek 5.5: Označení aktuálního času.

5.4.7 FR 7. Označení svátků a víkendů

Požadavek byl realizován prostřednictvím API poskytnutého plánovací komponentou, které umožňuje modifikaci vizuálního stylu jednotlivých časových intervalů v tabulce rozvrhu. Víkendy jsou v kalendáři zobrazeny výraznější barvou, zatímco běžné dny ponechávají standardní vzhled.
Na přiloženém obrázku 5.6. Označení svátků a víkendů je demonstrace této funkcionality, kde jsou data víkendů vyznačena červenou barvou a dny na jednotlivéch stojanech světlejší barvou.



Obrázek 5.6: Označení svátků a víkendů.

5.4.8 FR 8. Přepínání zobrazení a FR 21. Vyhledávání události

Kromě základní vyhledávací funkce, která umožňuje uživatelům efektivně nalézat konkrétní události podle jména, klienta nebo registrace letadla, systém také podporuje dynamické přepínání mezi různými zobrazeními plánů. Uživatelské interakce jsou zprostředkovány prostřednictvím intuitivních ovládacích prvků, jako jsou rozbalovací seznamy pro výběr letadla a klienta, a textového pole pro zadávání jména události.

5.4.9 FR 13. Změna rozsahu časové osy rozvrhu a FR 14. Navigace v časové ose plánu

Implementace požadavků byla již zabudována do vybrané plánovací knihovny, jak bylo specifikováno během výběru plánovací knihovny pro projekt (viz sekce 3.5.2. Analýza plánovacích knihoven).

5.4.10 FR 16. Historie plánování

Tento požadavek je realizován tak, že již vykonané work packages jsou zobrazeny jako pouze pro čtení. Tímto způsobem je zajištěno, že historie plánování zůstává nezměněná a uživatelé mají možnost zhlédnout dříve provedené plánovací aktivity bez možnosti jejich následné editace.

5.4.11 FR 17. Vrácení a odstranění úprav

Tento požadavek je realizován prostřednictvím **nadstavby** vybrané knihovny, která již obsahuje implementované funkce pro vrácení a odstranění provedených změn v plánu. Uživatelé mohou pomocí jednoduchého uživatelského rozhraní provádět akce jako ”undo”(vrácení posledních akcí) a ”redo”(znovu provedení vrácených akcí).

5.4.12 FR 18. Simulace plánování

Implementace tohoto požadavku je úzce spojena s realizací funkcionality popsané v sekci 5.4.11. Vybraná knihovna nabízí rozhraní pro interaktivní manipulaci s událostmi v plánu, jako jsou přetažení (drag), změna velikosti (resize) a další akce. Díky integraci funkcí pro vrácení (undo) a znovu provedení (redo) akcí, je možné v rámci této komponenty simulovat plánovací procesy bez trvalých změn.

5.4.13 FR 19. Automatické plánování a FR 20. Validace plánu

Implementace těchto požadavků je úzce provázána s rozšířením doménového modelu prostřednictvím vztahu **hmp:has-possible-bay**, který umožňuje

specifikovat možné stojany pro konkrétní zdroje (viz 4.3.1. Rozšíření doménového modelu).
Validace plánu událostí začíná definováním pravidel v TTL souboru, která jsou poté importována do databázového serveru. Tato pravidla určují vhodné stojany pro jednotlivé typy zdrojů. Příklad pravidla v TTL:

Příklad 5.7: Příklad doménového pravidla.

```
?aircraft hmp:has-possible-bay hmp:bay1;
```

Alternativně lze pravidla zadávat přímo do databáze. Následuje příklad pravidla, které umožňuje inferovat [14] možné stojany pro konkrétní model letadla:

Příklad 5.8: Příklad pravidla pro inferenci možných stojanů.

```
Id: infer-bay-from-aircraft-model
?aircraft a cm:aircraft
?aircraft cm:model "73H"
-----------------------------------------------------
?aircraft cm:has-possible-bay cm:bay-1, cm:bay-2
```

Aplikace těchto pravidel během plánování umožňuje systému provádět kontrolu a automaticky doplňovat vhodné hodnoty při plánování události. Při nesprávném výběru stojanu systém upozorní uživatele prostřednictvím chybové hlášky.

5.4.14 FR 10. Manuální plánování událostí různého typu:

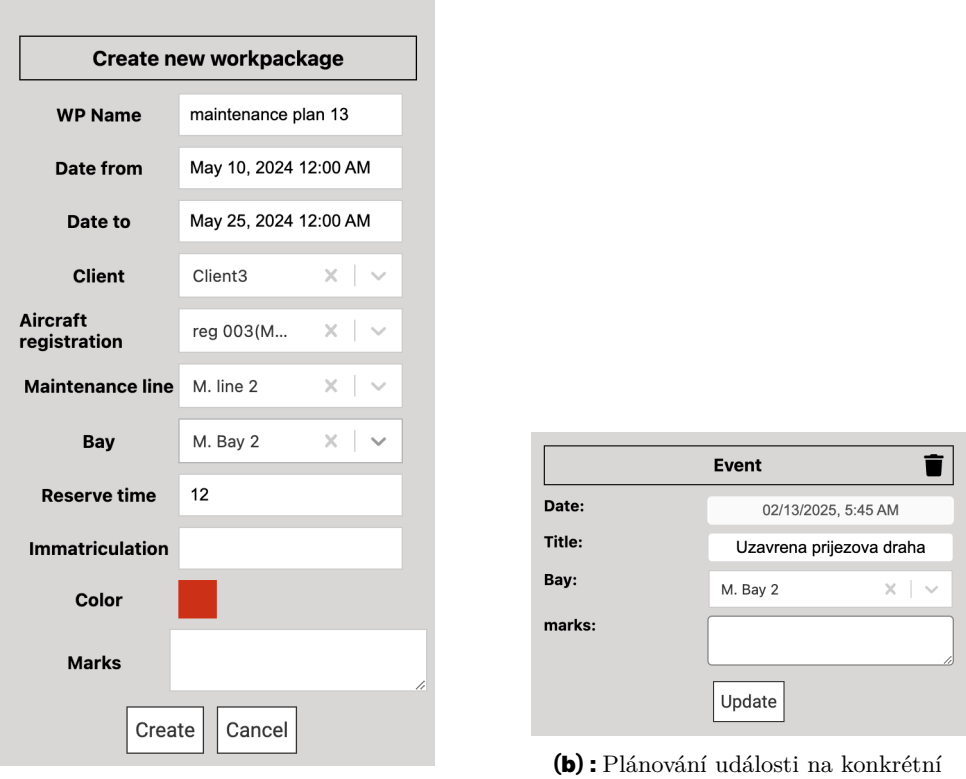
Komponenta nabízí uživatelské rozhraní pro ruční vytvoření různých typů událostí. Požadavky FR 11, FR 12, FR 15 jsou již zahrnuty v implementaci daného požadavku.

Plánování události work package

Implementace je realizována prostřednictvím modálního okna, které obsahuje formulář s různými polemi potřebnými k vyplnění, jako jsou datum a čas začátku a konce, identifikace zdrojů, a další specifické parametry.

Plánování denní události a události na konkrétní datum

Plánování denních událostí a událostí na specifická data je realizováno pomocí bočního panelu. Tento panel slouží nejen k zadávání nových událostí, ale také k jejich úpravám.



(a) : Plánování události work package.

(b) : Plánování události na konkrétní datum.

Obrázek 5.7: Příklady plánování různých typů událostí.

5.4.15 NFR 3. Automatizované nasazení změn na server

Automatizované nasazení změn je zajištěno prostřednictvím integrace hostingu Netlify[15] který je propojen s GitHub repozitářem určeným pro vyvíjenou komponentu. Netlify je nakonfigurováno tak, aby automaticky nasazovalo změny ve `main` a vývojových větvích repozitáře, přičemž větev `main` slouží jako produkční verze. Tento přístup umožňuje udržovat zobrazení produkční verze komponenty a zároveň poskytuje přehled o vyvíjených verzích.

Navíc soubor `README.md` v GitHub repozitáři obsahuje odznak (badge), který zobrazuje stav nasazení nejnovější verze komponenty. Tento odznak může nabývat různých stavů, které indikují aktuální status nasazení[16].

[15] Netlify. Nasazenou verzi vývojové větve naleznete na `https://dev-main-component-test--csat-hangar-maintenance-planner.netlify.app/`

[16] Netlify Badges. Více informace je na `https://docs.netlify.com/monitor-sites/status-badges/`

5.5 Implementace kontejnerizace systémových komponent

Kontejnerizace byla implementována pro každou komponentu systému zvlášť.

5.5.1 Kontejnerizace frontendové části

Pro frontendovou část aplikace byl vytvořen samostatný Docker kontejner. Tento kontejner obsahuje všechny potřebné závislosti a knihovny specifické pro klientskou část aplikace. Dockerfile pro frontend je specificky navržen tak, aby optimalizoval sestavení a nasazení statických souborů.

5.5.2 Kontejnerizace backendové části

Backend aplikace je rovněž izolována do vlastního Docker kontejneru. Tento kontejner zahrnuje veškeré serverové komponenty a služby.

5.5.3 Proměnné prostředí

Pro správu konfigurace v obou kontejnerech se používají proměnné prostředí, které umožňují dynamické nastavení parametrů bez nutnosti zásahu do kódu aplikace. Skript pro nastavení těchto proměnných je navržen tak, aby automaticky detekoval a aplikoval příslušné konfigurační hodnoty v závislosti na prostředí, ve kterém je kontejner spuštěn.

5.6 Ukázka finálního rozhraní

Prezentovaná ukázka finálního rozhraní aplikace (viz Příloha D. Ukázka finálního rozhraní) demonstruje výsledný vzhled a funkčnost.

5.7 Vyhodnocení implementace požadavků

Tato sekce poskytuje zhodnocení míry splnění funkcionalit definovaných v kapitole s popisem požadavků. Na následujícím obrázku 5.8 je prezentováno grafické vyhodnocení, kde:

- Zelená barva indikuje plné splnění požadavků, což značí, že webová aplikace danou funkcionalitu plně podporuje.
- Žlutá barva signalizuje částečné splnění požadavků, což znamená, že aplikace funkcionality podporuje, avšak s určitými omezeními nebo nedostatky.
- Červená barva ukazuje na funkcionality, které nejsou v současné verzi aplikace implementovány a jejich realizace je plánována pro další fáze vývoje.

Obrázek 5.8: UML diagram případů užití s grafickým vyhodnocením splnění požadavků.

Kapitola 6

Evaluace

Kapitola poskytuje podrobný pohled na testování a vyhodnocení systému jak z pohledu jeho funkčnosti, tak z pohledu uživatelské zkušenosti. Klíčové aspekty této fáze zahrnují uživatelské testování, ověření správnosti a účinnosti systému na základě reálných dat z provozu. Cílem je ověřit, zda systém splňuje stanovené požadavky a očekávání, a identifikovat případné oblasti pro další zlepšení.

6.1 Uživatelské testování

Uživatelské testování je zásadní pro ověření, že systém je intuitivní, efektivní a přátelský k jeho uživatelům. Během této fáze jsou různí uživatelé pozváni k interakci se systémem, aby vykonali řadu předem definovaných úkolů. Zpětná vazba uživatelů je poté analyzována, aby se zjistilo, jak dobře systém splňuje jejich potřeby a jaké funkce vyžadují úpravy nebo vylepšení. [13]

6.1.1 Návrh testovacích scénářů

Testovací scénáře byly pečlivě navrženy tak, aby co nejúčinněji pokryly široké spektrum potenciálních uživatelských interakcí s aplikací. Seznam níže uvádí vybrané scénáře spolu s jejich cíli:

1. **Vyhledání specifické události:** ověřit funkčnost a efektivitu vyhledávací funkce.
2. **Vytvoření události work package:** ověřit, že systém umožňuje správné přidání nové události do plánu.
3. **Úprava události work package:** ověřit možnost modifikace detailů existující události a reflektovat tyto změny v rozvrhu.
4. **Testování workflow s denní událostí:** ověřit, že systém správně zpracovává workflow denních událostí, umožňuje jejich úpravy a tyto změny jsou správně reflektovány v rozvrhu.
5. **Testování workflow s denní událostí na specifickém datu:** ověřit, že systém správně zpracovává workflow denních událostí přiřazených

k určitému datu, umožňuje jejich úpravy a tyto změny jsou správně reflektovány v rozvrhu.

6. **Simulace změn plánování:** otestovat schopnost systému simulovat různé plánovací scénáře bez trvalého ovlivnění aktuálního rozvrhu.

6.1.2 Kontrolní otázky po testování

1. Poznámky k prvnímu scénáři.
2. Poznámky k druhému scénáři.
3. Poznámky k třetímu scénáři.
4. Poznámky k čtverému scénáři.
5. Poznámky k pátému scénáři.
6. Poznámky nebo návrhy k vylepšení systému.

Detailní a podrobnější kontrolní otázky těchto scénářů jsou uvedeny v příloze B. Testovací scénáře.

6.1.3 Výsledky testovacích scénářů

Tato sekce poskytuje přehled výsledků testování, které bylo provedeno doménovými experty.

Tester 1

Testovací scénář 1 – Vyhledání specifické události.

1. U druhého bodu jsem si nebyla jista, kde mám vyhledávat.
2. Do pole “Title” bych přidala např. obrázek lupy, aby bylo jasnější, kam má uživatel psát pro vyhledávání.

Testovací scénář 2 – Vytvoření události work package.

1. Při vyplňování mi nebylo jasné, zda to je součástí vybrané doby Od - Do, nebo je to ještě doba navíc.
2. Při vyplňování “Reserve time” není jasné v jakých je to jednotkách, jestli v hodinách nebo ve dnech.
3. Při výběru “Date from” je ve všech měsících zvýrazněna “23”.

Testovací scénář 3 – Úprava události work package.

1. Po zmáčknutí “Undo” se mi nezměnily informace v pravém panelu (např. v panelu je Bay 5 a v kalendáři Bay 3, jiná barva).
2. Z popisu scénáře mi nebylo úplně jasné, kdy použít “Update” a kdy “Save”.

Testovací scénář 4 – Testování workflow s denní událostí.

1. Mění se pozice v kalendáři, ale informace v panelu ne.
2. Pokud vybírám buňku v kalendáři pro denní událost a kliknu na ní, tak bych jí možná nějak vizuálně odlišila (že je aktuálně vybraná).

Testovací scénář 5 – Testování workflow s denní událostí na specifickém datu.

1. Nic mě k tomu nenapadá, vše mi fungovalo.

Testovací scénář 6 – Simulace změn plánování.

1. Po ruční úpravě události se po uložení nemění vlastnosti události v panelu (kde je v panelu jiné datum, než v kalendáři)

Otázky k položení po testování.

1. Je to intuitivní až na pár věcí, které jsem psala výše
2. Některé změny se podle mého názoru nepropisují do panelu, kde jsou informace o událostech (viz komentáře u scénářů)
3. Celkem rychle, je to intuitivní, po tom co se s tím člověk trochu seznámí
4. Pro urychlení prvotní fáze učení se se systémem, by bylo dobré přidat k některým pojmům vysvětlivky, aby si uživatel mohl přečíst přímo v aplikaci, co daný pojem znamená a co tam má uživatel zadat

Tester 2

Testovací scénář 1 – Vyhledání specifické události.

1. Scénář nebyl testován - nenašla jsem, kde se nachází event Maintenance plan.

Testovací scénář 2 – Vytvoření události work package.

1. Pozor, událost lze ručně přemístit, ale informace ve WP se nezmění.

Testovací scénář 3 – Úprava události work package.

1. Proběhlo v pořádku.

Testovací scénář 4 – Testování workflow s denní událostí.

1. Scénář nebyl testován - nenašla jsem, kde se nachází denní událost.

Testovací scénář 5 – Testování workflow s denní událostí na specifickém datu.

1. Scénář nebyl testován - nenašla jsem, kde se nachází denní událost.

Testovací scénář 6 – Simulace změn plánování.

1. Při tažení konce a začátku události se informace v panelu nezmění.

Otázky k položení po testování.

1. Pro mě to nebylo dostatečně uživatelsky přívětivé, nedokázala jsem udělat 3 scénáře.

2. Nedošlo k žádné změně v datumu, když jsem ručně měnila konec událostí. Při posouvání událostí v plánu se datum v plánu nezměnilo.

3. Nad testováním jsem strávila 15 min. 3 scénáře jsem vynechala.

Tester 3

Testovací scénář 1 – Vyhledání specifické události.

1. Chvíli mi trvalo než jsem pochopil, že vyhledávání proběhlo. Aplikace nemá žádnou hlášku nebo potvrzení a pokud se zrovna nekoukám do plánku, nevšimnu si že filtrace proběhla. Taky jsem nebyl schopen ověřit, zda filtrace dle registrace letadla je správně, jelikož v plánku nevidím, které události jsou vázány na jaké letadlo.

2. Vadilo mi, že událost na bay 4 v 12/2025 nemá název - nebyl jsem schopen ověřit její filtraci.

3. Třeba podbarvit zeleně vyhledávací pole, když filtrace (vyhledávání) proběhne. Nebo vypsat někde hlášku o tom.

Testovací scénář 2 – Vytvoření události work package.

1. Ne zcela. Chvíli mi trvalo než jsem pochopil, že vyhledávání proběhlo. Aplikace nemá žádnou hlášku nebo potvrzení a pokud se zrovna nekoukám do plánku, nevšimnu si že filtrace proběhla. Taky jsem nebyl schopen ověřit, zda filtrace dle registrace letadla je správně, jelikož v plánku nevidím, které události jsou vázány na jaké letadlo.

2. Vadilo mi, že událost na bay 4 v 12/2025 nemá název - nebyl jsem schopen ověřit její filtraci.

3. Třeba podbarvit zeleně vyhledávací pole, když filtrace (vyhledávání) proběhne. Nebo vypsat někde hlášku o tom.

Testovací scénář 3 – Úprava události work package.

1. Proběhlo v pořádku.

Testovací scénář 4 – Testování workflow s denní událostí.

1. Hodnoty se s použitím “undo” a “redo” nemění. V kalendáři ano, ale v jejim prehledu vpravo ne.

2. Ano, i když se opakoval problém z předchozího scénáře - událost zůstane zobrazena v bočním panelu/přehledu.

3. Zmínil výše v bodech výše. Navíc mi vadilo že nemohu změnit datum konce události do momentu, dokud neaktualizuji událost z pohledu datumu počátku. Uživatelsky přívětivější by bylo umožnit takovou změnu v okamžiku, kdy v poli počátku události již je zvolena jiná hodnota, ne nutně čekat ještě na update.

4. Vyřešit zmíněné problémy výše.

Testovací scénář 5 – Testování workflow s denní událostí na specifickém datu.

1. Opakoval se problém z předchozího scénáře - událost zůstane zobrazena v bočním panelu/přehledu.

Testovací scénář 6 – Simulace změn plánování.

1. Znovu zde není propojení bočního přehledu události s kalendářem. Editace v kalendáři nemá dopad na detaily události v okně vpravo, což je matoucí.

Otázky k položení po testování.

1. Obecně v pořádku. Bez nápovědy bych ale některé funkce nebyl schopen najít.

2. Ano. Popsal jsem podrobně ve scénářích výše.

3. S nápovědou během okamžiku.

4. Nemám nic více, co jsem již nepopsal výše.

6.1.4 Vyhodnocení výsledků testovacích scénářů

V této sekci jsou analyzovány a shrnuty výsledky uživatelského testování. Testování bylo zaměřeno na ověření funkčnosti, intuitivnosti a uživatelské přívětivosti systému. Zpětná vazba od testerů byla získána prostřednictvím pečlivě navržených testovacích scénářů, které pokrývaly různé aspekty aplikace.

Celkové hodnocení intuitivnosti uživatelského rozhraní

Testy odhalily, že ačkoliv je rozhraní obecně vnímáno jako intuitivní, existují určité prvky, které by mohly být zlepšeny pro zvýšení uživatelského komfortu. Například, někteří uživatelé měli problémy s identifikací toho, jak a kde provádět specifické akce jako je vyhledávání nebo filtrování událostí. Tyto problémy byly částečně způsobeny nedostatečnými vizuálními indikátory a feedbackem po provedení určitých akcí.

Technické problémy a nejasnosti

Během testování byly identifikovány některé technické nedostatky, zejména v případech, kdy systém nezobrazil očekávané výsledky nebo kdy nedošlo k aktualizaci informací v reálném čase. Výsledky naznačují, že synchronizace mezi různými částmi aplikace může být problematická, zejména v situacích, kde uživatelé očekávají okamžitou odezvu nebo aktualizaci dat.

6.1.5 Závěr testování

Na základě provedených uživatelských testů lze konstatovat, že aplikace je obecně intuitivní, přestože někteří uživatelé narazili na určité problémy s intuitivností některých funkcí a navigací. Hlavní oblasti pro zlepšení zahrnují:

- Zřetelnější indikace filtrů.
- Vylepšení zpětné vazby při vyhledávání událostí, například přidáním vizuálních indikátorů nebo potvrzovacích zpráv.
- Zajištění, že změny provedené v kalendáři se správně promítají do detailního zobrazení událostí.
- Přidání vysvětlujících návodů nebo nápovědy pro klíčové funkce, aby uživatelé mohli rychleji pochopit, jak s aplikací pracovat.

Kapitola 7

Závěr

Hlavním cílem práce bylo vyvinout webovou aplikaci pro efektivní plánování a správu údržby v hangárovém prostředí. Pro splnění tohoto cíle byla provedena práce od počátečního prototypování až po návrh a implementaci. Po důkladné analýze dostupných knihoven, byla zvolena nadstavba knihovny react-calendar-timeline, která odpovídala většině stanovených kritérií. Na základě této volby následoval návrh a vývoj modulární webové aplikace, která je rozdělena na dvě hlavní komponenty, každá z nich kontejnerizovaná pro snadné nasazení do cílového prostředí.

Uživatelské rozhraní aplikace bylo podrobeno testovacímu procesu doménovými experty, což umožnilo získat detailní zpětnou vazbu na její funkčnost a interakci s uživateli. Toto testování odhalilo několik oblastí, které vyžadují další vylepšení, zejména co se týče uživatelské přívětivosti a intuitivnosti rozhraní.

Přestože je aplikace ve své základní verzi již připravena k integraci a umí plánovat a spravovat událostí, přechod do ostrého provozu vyžaduje řešení několika klíčových úkolů, které zajišťují plnou funkčnost v produkčním prostředí. Jednou z hlavních výzev je zajištění aktualizace dat, jak je diskutováno v sekci 4.4. Tento problém byl identifikován v pozdější fázi vývoje, a proto nebyl zcela adresován.

V následujících fázích vývoje je prioritou doimplementace dosud nesplněných požadavků a potenciálně rozšíření funkcionality aplikace o nové vlastnosti. Za klíčové jsou považovány požadavky označené jako ”must háve”, které jsou nezbytné pro plnou funkčnost aplikace v ostrém provozu. Dále je zásadním krokem pro dosažení vyšší uživatelské spokojenosti detailní zpracování zpětné vazby získané během testování s doménovými experty. Tato zpětná vazba bude sloužit jako klíčový vstup pro optimalizaci uživatelského rozhraní s cílem zvýšit intuitivnost a ergonomii aplikace.

Závěrem lze konstatovat, že dosažené výsledky této bakalářské práce představují významný krok směrem k efektivnímu plánování a správě údržby v hangárovém prostředí. Aplikace byla navržena a implementována s ohledem

na potřeby koncových uživatelů a možnosti budoucího rozšíření a integrace do existující infrastruktury. I když byly identifikovány některé oblasti, které vyžadují další práci, základní struktura a funkcionalita aplikace poskytují solidní základ pro další vývoj a optimalizaci.

Příloha A

Literatura

[1] Stuart D. Practical Ontologies for Information Professionals: Ontologies and the Semantic Web (Červen 2018). [Citováno: 12.1.2024].

[2] DeepSea Developments. Why is prototyping important? <https://www.linkedin.com/pulse/why-prototyping-important-deepseadev-owbce/> (Listopad 2023). [Citováno: 14.5.2024].

[3] GitHub. Fork a repository. <https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/fork-a-repo/>. [Citováno: 14.5.2024].

[4] HotPMO. MoSCoW or Kano Models – how do you prioritize? <https://www.hotpmo.com/management-models/moscow-kano-prioritize/>. [Citováno: 12.5.2024].

[5] IBM. What is a REST API? <https://www.ibm.com/topics/rest-apis>. [Citováno: 14.5.2024].

[6] IBM. What is containerization? <https://www.ibm.com/topics/containerization>. [Citováno: 14.5.2024].

[7] IBM. What is continuous deployment? <https://www.ibm.com/topics/continuous-deployment>. [Citováno: 14.5.2024].

[8] IBM. What is three-tier architecture? <https://www.ibm.com/topics/three-tier-architecture>. [Citováno: 14.5.2024].

[9] Adam Kehoe. Scheduling problems 101. <https://medium.com/opex-analytics/opex-101-scheduling-problems-f77f49e636a8> (Duben 2019). [Citováno: 12.1.2024].

[10] Petr Křemen Martin Ledvinka, Bogdan Kostov. JOPA: Efficient Ontology-based Information System Design. <https://2016.eswc-conferences.org/sites/default/files/papers/Accepted%20Posters%20and%20Demos/ESWC2016_DEMO_JOPA.pdf>. [Citováno: 12.5.2024].

[11] Mehmet Ozkaya. Medium. When to use monolithic architecture. <https://medium.com/design-microservices-architecture-with-patterns/when-to-use-monolithic-architecture-57c0653e245e> (Únor 2023). [Citováno: 14.5.2024].

[12] Umesha Naik and D. Shivalingaiah. Comparative study of web 1.0, web 2.0 and web 3.0. <https://www.researchgate.net/publication/264845599_Comparative_Study_of_Web_10_Web_20_and_Web_30> (Březen 2009). [Citováno: 12.1.2024].

[13] Omniconvert. What is user testing? <https://www.omniconvert.com/what-is/user-testing/> (Květen 2024). [Citováno: 20.5.2024].

[14] Ontotext. Reasoning. <https://graphdb.ontotext.com/documentation/10.1/reasoning.html#entailment-rules>. [Citováno: 22.5.2024].

[15] Ontotext. What are ontologies? <https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies/>. [Citováno: 13.1.2024].

[16] Ontotext. What is inference? <https://www.ontotext.com/knowledgehub/fundamentals/what-is-inference/>. [Citováno: 20.5.2024].

[17] Oracle. Java platform. <https://www.java.com/en/>. [Citováno: 13.1.2024].

[18] Seobility. JSON-LD. <https://www.seobility.net/en/wiki/JSON-LD>. [Citováno: 13.1.2024].

[19] Juan Sequeda. Introduction to: Triplestores. <https://www.dataversity.net/introduction-to-triplestores> (Leden 2013). [Citováno: 12.1.2024].

[20] Juan Sequeda. SPARQL. <https://www.dataversity.net/introduction-to-sparql> (Září 2011). [Citováno: 13.1.2024].

[21] Kerstin Sieckmann. Production planning with excel - is your spreadsheet error free? <https://blog.netronic.com/production-planning-with-excel-is-your-spreadsheet-error-free> (Duben 2018). [Citováno: 12.1.2024].

[22] RNDr. Petr Skřehot. Hierarchical Task Analysis. <https://vubp.cz/soubory/prevence-zavaznych-havarii/metodiky/hierarchical-task-analysis_web.pdf>. [Citováno: 14.5.2024].

[23] Joshua Tauberer. What is RDF. XML. com (Červenec 2006). [Citováno: 12.1.2024].

[24] THE INVESTOPEDIA TEAM. Web 3.0 explained, plus the history of web 1.0 and 2.0. <https://www.investopedia.com/web-20-web-30-5208698> (Říjen 2023). [Citováno: 12.1.2024].

[25] W3C. OWL. <https://www.w3.org/TR/owl-features/> (Listopad 2009). [Citováno: 13.1.2024].

[26] W3C. OWL 2. <https://www.w3.org/TR/owl2-new-features/>. [Citováno: 12.5.2024].

[27] W3C. Turtle - Terse RDF Triple Language. <https://www.w3.org/TeamSubmission/turtle/> (Březen 2011). [Citováno: 4.2.2024].

Příloha B

Testovací scénáře

V této příloze jsou prezentovány detailní testovací scénáře navržené pro ověření funkčnosti a uživatelské přívětivosti systému. Pro každý scénář je uveden podrobný postup, který má uživatelé následovat, a sada kontrolních otázek sloužících k evaluaci zkušenosti uživatele a funkčních aspektů systému.

B.1 Vyhledání specifické události

Cíl scénaře

Ověřit funkčnost a efektivitu vyhledávací funkce.

Průchod

1. Navigujte na stránku[1] kalendáře.
2. Vyhledejte události s názvem **“Maintenance plan”**.
   - Sledujte výsledky: nalezení všech událostí s podobným názvem.
3. Specifikujte vyhledávání přidáním registraci letadla **“reg REG003(ModelX3)”**.
   - Sledujte výsledky: systém zobrazí všechny události s danou registrací letadla a názvem ”Maintenance plan”.
4. Specifikujte vyhledávání přidáním klienta “Client **2**”.
   - Sledujte výsledky: prázdný výsledek vyhledávání.
5. Specifikujte vyhledávání přidáním klienta **“Client 3”**.
   - Sledujte výsledky: výsledky odpovídají specifikovanému klientu, registraci letadla a názvu události.
6. Doplňte vyhledání podle názvu do “Maintenance plan 2”.

[1] Netlify. Nasazenou verzi vývojové větve naleznete na `https://dev-main-component-test--csat-hangar-maintenance-planner.netlify.app/`

- Sledujte výsledky: systém reflektuje specifičtější vyhledávání.

7. Vymažte všechna vyplněná pole vyhledávání.
   - Sledujte výsledek: systém správně resetuje vyhledávací kritéria a zobrazuje všechny události.

Kontrolní otázky

1. Byly výsledky vyhledávání relevantní a přesné vzhledem k zadaným kritériím?
2. Byl proces vyhledávání intuitivní a snadno pochopitelný?
3. Jaké problémy jste během testování vyhledávací funkce zaznamenali?
4. Máte nějaké návrhy na zlepšení vyhledávacího nástroje v systému?

B.2 Vytvoření události work package

Cíl scénaře

Ověřit, že systém umožňuje správné přidání nové události do plánu.

Průchod

1. Vytvořte nový Work package.
2. Vyplňte:
   - Zadejte název události, například **"Work package 123"**.
   - Vyberte datum začátku a konce události.
   - Vyberte datum konce události jako **2024-05-25**.
   - Vyberte datum začátku události jako **2024-05-02**.
     - Sledujte výsledek: automaticky se datum konce posune do data začátku události.
3. Vyberte datum konce události jako **2024-06-25**.
4. Přiřaďte událost k stojanu **M.Bay 4**.
5. Přiřaďte událost k klientu **Client2**.
6. Přiřaďte událost k letounu **reg 001(Model X1)**.
   - Sledujte výsledek: zobrazí se hláška, že se nelze naplánovat událost na vybraný stojan.
7. Přiřaďte událost k stojanu **M.Bay 2**.

8. Vyberte **zodpovědnou linku**.
9. Přidejte **rezervu**.
10. Vyberte **barvu** pro obarvení události.
11. Přidejte **specifické poznámky** související s událostí.
12. Uložte změny.
    - Sledujte výsledky: Ověřte, že nová událost se zobrazuje v kalendáři nebo plánovacím systému a zkontrolujte správnost zobrazených informací.

Kontrolní otázky

1. Zobrazuje se nová událost v kalendáři po jejím přidání?
2. Jsou všechny zadané informace o události správně zobrazeny v kalendáři?
3. Je událost zvýrazněna vybranou barvou?
4. Funguje rezerva správně?
5. Jaké problémy jste během testování vytvoření události Work package zaznamenali?
6. Máte nějaké návrhy na zlepšení procesu vytváření událostí v systému?

B.3 Úprava události work package

Cíl scénaře

Ověřit možnost modifikace detailů existující události a reflektovat tyto změny v rozvrhu.

Průchod

1. Najděte událost **“Maintenance plan 11”** v rozvrhu.
2. Vstupte do editačního módu události.
3. Změňte parametry (např. změna času, přiřazení nového stojanu).
4. Uložte změny.
    - Sledujte výsledek: provedené změny viditelné v rozvrhu.
5. Vraťte předchozí stav události pomocí tlačítka **“Undo”**.
    - Sledujte výsledek: systém vrátí událost do předchozího stavu, což bude viditelné v detailních informacích.

6. Uložte změny.
7. Proveďte opětovné změny stisknutím tlačítka **“Redo”**.
   - Sledujte výsledek: systém vrátí provedené změny, což bude viditelné v detailních informacích.
8. Změňte parametry (např. změna času, přiřazení nového stojanu).
9. Uložte změny.
   - Sledujte výsledek: provedené změny viditelné v rozvrhu.

Nápověda

Editační mód události: vyberte událost dvojitým kliknutím.

Kontrolní otázky

1. Zobrazuje se upravená událost v kalendáři správně po provedení změn?
2. Jsou všechny provedené změny správně zobrazeny v detailech události?
3. Funguje tlačítko “Undo”/”Redo” správně?
   - Jsou všechny vrácené změny správně zobrazeny v detailech události?
   - Jsou všechny opětovné změny správně zobrazeny v detailech události?
4. Jaké problémy jste zaznamenali během úpravy a vrácení předchozího stavu události?
5. Máte nějaké návrhy na zlepšení procesu úpravy událostí v systému?

B.4 Testování workflow s denní událostí

Cíl scénaře

Ověřit, že systém správně zpracovává workflow denních událostí, umožňuje jejich úpravy a tyto změny jsou správně reflektovány v rozvrhu.

Průchod

1. Vytvořte denní událost.
   - Zadejte název události: **”Denní údržba”**.
   - Přidejte specifické poznámky související s událostí.
   - Uložte změny.
   - Sledujte výsledek: nová událost se zobrazuje v kalendáři.

2. Vyhledejte nově vytvořenou událost:
   - Použijte vyhledávací nástroj a zadejte název **”Denní údržba”**.
   - Sledujte výsledek: ověřte, že vyhledávač nalezne správnou událost.
3. Upravte detaily události:
   - Změňte datum konce události na 3 dny dopředu.
   - Změňte poznámku.
   - Změňte barvu.
   - Uložte změny a přibližte zobrazení na danou událost.
     - Sledujte výsledek: změny jsou reflektovány v kalendáři.
4. Vraťte změny pomocí tlačítka **“Undo”**.
   - Sledujte výsledek: Ověřte, že systém vrátí událost do původního stavu.
5. Opětovně proveďte změny pomocí tlačítka **“Redo”**.
   - Sledujte výsledek: systém znovu aplikuje provedené změny.
6. Smažte vytvořenou denní událost.
   - Sledujte výsledek: událost byla úspěšně smazána z kalendáře.

Nápověda

- Vytvoření denní událost: vyberte buňku dvojitým kliknutím v kalendáři, která odpovídá křížení vybraného stojanu (maintenance bay) a zvoleného data.
- Posun po časové ose: držte stisknuté levé tlačítko myši a táhněte na rozvrhu vpravo nebo vlevo pro posun v čase.

Kontrolní otázky

1. Je nově vytvořená denní událost správně zobrazena v kalendáři?
2. Funguje vyhledávací nástroj správně při vyhledávání denní události?
3. Jsou provedené změny události správně zobrazeny v kalendáři?
4. Funguje tlačítko “Undo”/“Redo” správně při vracení/aplikaci provedených změn?
5. Funkce pro smazání události funguje správně?
6. Jaké problémy jste zaznamenali během testování workflow s denní událostí?
7. Máte nějaké návrhy na zlepšení procesu vytváření, úpravy a mazání denních událostí v systému?

B.5 Testování workflow s denní událostí na specifickém datu

Cíl scénaře

Ověřit, že systém správně zpracovává workflow denních událostí přiřazených k určitému datu, umožňuje jejich úpravy a tyto změny jsou správně reflektovány v rozvrhu.

Průchod

1. Vytvořte novou denní událost na specifickém datu:
   - Zadejte název události: **"Příjezdová dráha mimo provoz"**.
   - Přidejte specifické poznámky související s událostí.
   - Uložte změny.
     - Sledujte výsledek: nová událost se zobrazuje v kalendáři.
2. Upravte detaily události:
   - Změňte datum události na 3 dny dopředu.
   - Změňte poznámku.
   - Uložte změny a přibližte zobrazení na danou událost.
   - Sledujte výsledek: změny jsou reflektovány v kalendáři.
3. Vraťte změny pomocí tlačítka **"Undo"**:
   - Sledujte výsledek: ověřte, že systém vrátí událost do původního stavu.
4. Opětovně proveďte změny pomocí tlačítka **"Redo"**:
   - Sledujte výsledek: systém znovu aplikuje provedené změny.
5. Smažte vytvořenou denní událost:
   - Sledujte výsledek: událost byla úspěšně smazána z kalendáře.

Nápověda

- Vytvoření události na specifickém datu: vyberte datum v kalendáři pravým kliknutím na příslušný den.
- Přiblíženi zobrazení na danou událost: držte stisknuté klávesy cmd (na Mac) nebo ctrl (na Windows) a použijte kolečko myši pro přiblíženi v kalendáři.

Kontrolní otázky

Stejné jako jsou v B.4. Testování workflow s denní událostí

B.6 Simulace změn plánování

Cíl scénáře

Ověřit, zda systém správně zaznamenává, vrací a opětovně aplikuje změny v plánu, a to bez chyb.

Průchod

1. Proveďte pět různých změn v plánu, jako jsou:
   - Přesun události Work Package na jiný čas nebo datum.
   - Zvětšení doby trvání události tažením konce události.
   - Zmenšení doby trvání události tažením začátku události.
2. Uložte změny pomocí tlačítka **"Save"**.
   - Sledujte výsledek: změny by měly být úspěšně zaznamenány.
3. Použijte tlačítko pro vrácení předchozího stavu a vraťte stav o tři kroky zpět.
   - Sledujte výsledek: systém by měl správně vrátit stav.
4. Použijte tlačítko pro opětovné provedení změn a vraťte zpět tři kroky vpřed.
   - Sledujte výsledek: systém by měl správně vrátit stav.
5. Vraťte systém do původního stavu.
   - Sledujte výsledek: systém by měl správně vrátit stav.

Kontrolní otázky

1. Jak efektivně systém zvládl vrácení a obnovení změn? Bylo to bez chyb?
2. Máte nějaké návrhy na zlepšení simulačního režimu?

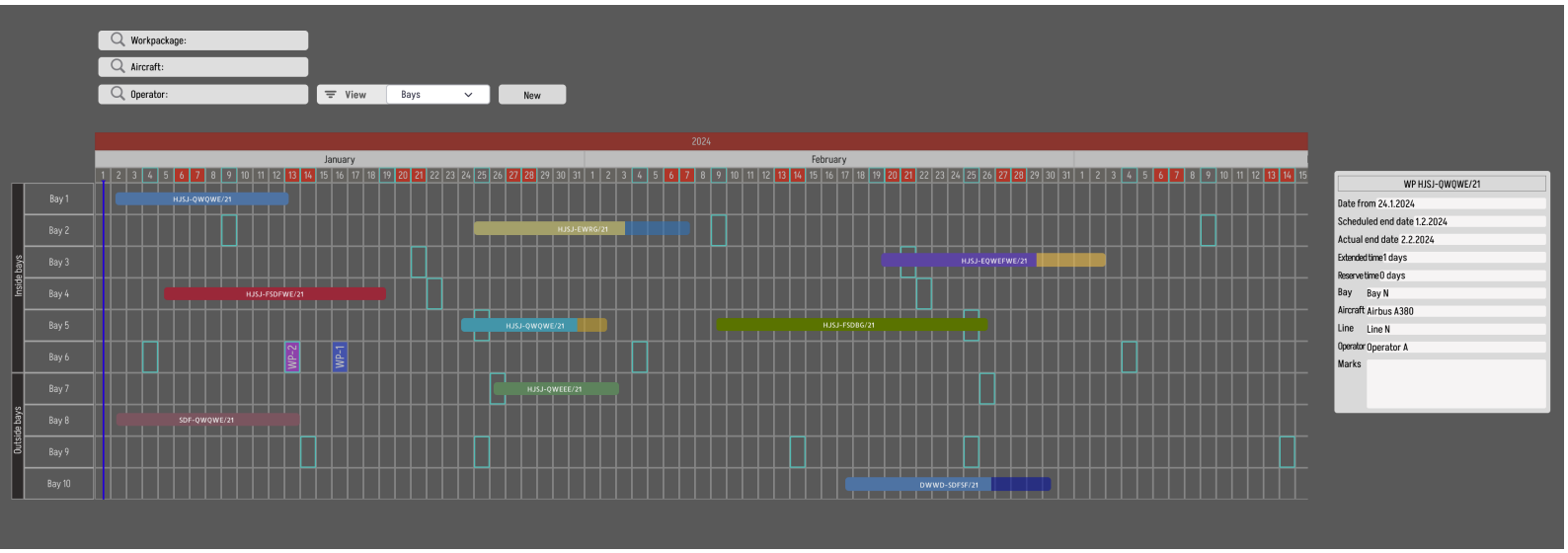
B.7 Otázky k položení po testování

1. Jak hodnotíte intuitivnost uživatelského rozhraní?
2. Setkali jste se během testování s nějakými technickými problémy nebo nejasnostmi?

3. Jak rychle jste byli schopni se naučit pracovat s funkcemi?
4. Existuje něco, co by podle Vás vyžadovalo další zlepšení?

Příloha C

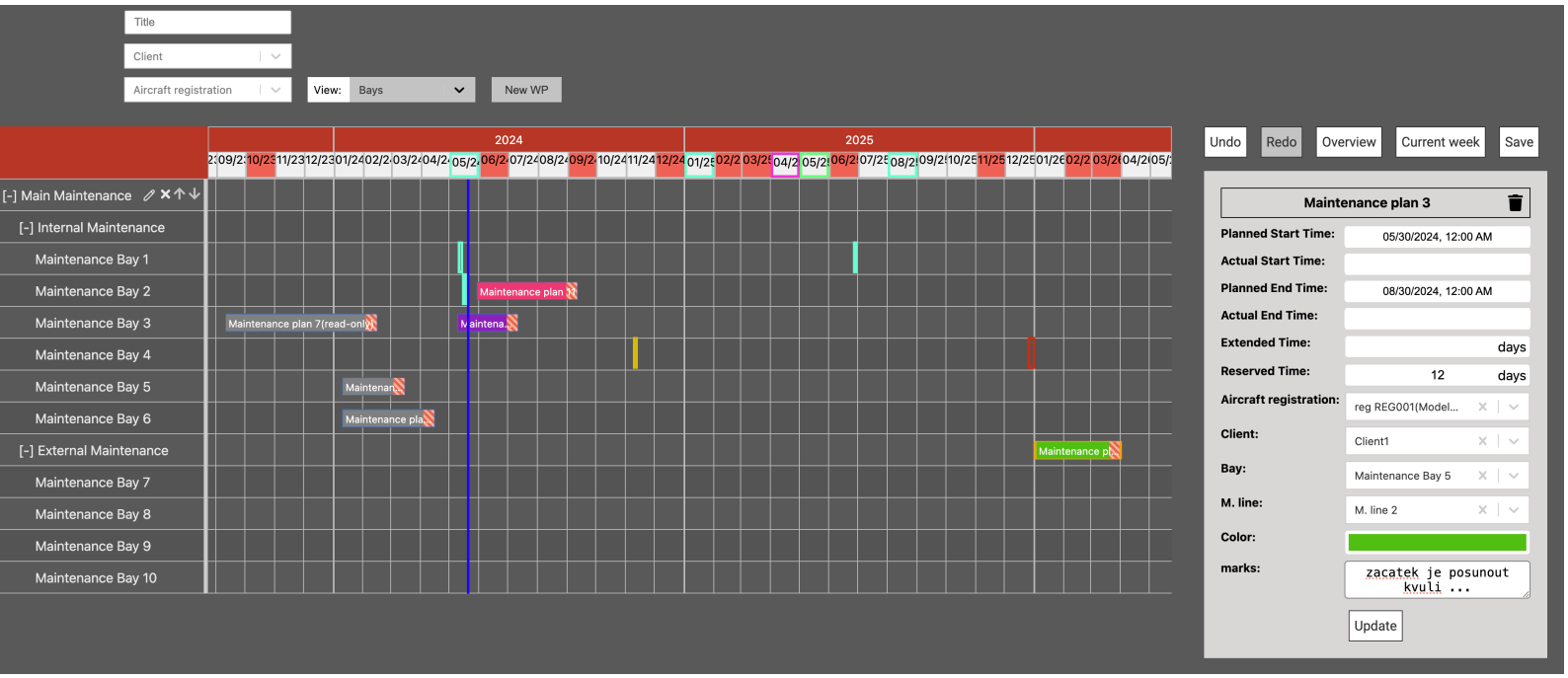
Ukázka drátěného modelu



Obrázek C.1: Drátěný model tabulky rozvrhu.

Příloha D

Ukázka finálního rozhraní



Obrázek D.1: Finální rozhraní aplikace.

Příloha E

Obsah přiložených souborů

- **YU-thesis** - text bakalářské práce.
- **csat-hangar-maintenance-planner-wireframes** - wireframy.
- **csat-hangar-maintenance-planner** - frontendová část aplikace.
- **csat-hangar-maintenance-planning-system** - backendová část aplikace.