

Sem vložte zadání Vaší práce.





**FAKULTA  
INFORMAČNÍCH  
TECHNologiÍ  
ČVUT V PRAZE**

Diplomová práce

## **Vývoj FIORI aplikace nad SAP PM modulem pro realizaci servisních zakázek a preventivní údržby**

*Bc. Marcel Morávek*

Katedra softwarového inženýrství  
Vedoucí práce: Ing. Martin Šindlář

9. května 2018



---

## Poděkování

Velké díky patří Ing. Martinovi Šindlářovi za jeho čas a výpomoc při návrhu architektury a vedení této práce. Dále bych chtěl poděkovat Ing. Martinovi Kurkovi za konzultace ohledně modulu SAP PM.



---

## Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 9. května 2018

.....

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2018 Marcel Morávek. Všechna práva vyhrazena.

*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.*

## **Odkaz na tuto práci**

Morávek, Marcel. *Vývoj FIORI aplikace nad SAP PM modulem pro realizaci servisních zakázek a preventivní údržby*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.



---

## Abstrakt

Tato práce se zabývá tvorbou webové aplikace nad modulem údržby SAP Plant Maintenance. Popisuje komponenty SAP potřebné pro vývoj požadované aplikace, analýzu požadavků a následnou implementaci.

**Klíčová slova** SAP, Model údržby, Fiori, SAP Gateway, BSP, SAPUI5, uživatelské rozhraní

---

## Abstract

This thesis deals with the creation of web application over SAP Plant Maintenance module. Describes the SAP components needed to develop required application, requirements analysis and implementation itself.

**Keywords** SAP, Module Plant Maintenance, Fiori, SAP Gateway, BSP, SAPUI5, user interface



---

# Obsah

<b>Úvod</b>	<b>1</b>
Cíl práce . . . . .	1
Co není cílem práce . . . . .	2
Struktura práce . . . . .	2
<b>1 SAP</b>	<b>3</b>
1.1 Společnost SAP . . . . .	3
1.2 SAP ERP . . . . .	3
1.3 SAP Plant Maintenance (PM) . . . . .	7
1.4 SAP BSP . . . . .	10
1.5 SAP FIORI . . . . .	11
<b>2 Analýza</b>	<b>17</b>
2.1 Základní popis aplikace . . . . .	17
2.2 Uživatelské role . . . . .	17
2.3 Model požadavků . . . . .	18
2.4 Model případů užití (Use Case Model) . . . . .	24
2.5 Závěry analýzy . . . . .	27
<b>3 Návrh uživatelského rozhraní</b>	<b>29</b>
3.1 Task Groups . . . . .	29
3.2 Task Graph . . . . .	31
3.3 Lo-Fi prototyp . . . . .	33
3.4 Heuristická analýza . . . . .	40
<b>4 Implementace</b>	<b>43</b>
4.1 Architektura . . . . .	43
4.2 SAP ERP . . . . .	44
4.3 SAP GW . . . . .	45
4.4 Apache Tomcat . . . . .	48

4.5	PM SAPUI5 Aplikace . . . . .	52
4.6	LOGIN SAPUI5 Aplikace . . . . .	69
4.7	Testování . . . . .	70
4.8	Porovnání vývojových prostředí . . . . .	73
4.9	Doporučení pro vývoj . . . . .	77
<b>Závěr</b>		<b>79</b>
	Budoucí vývoj . . . . .	79
	Dosažení cílů . . . . .	80
<b>Literatura</b>		<b>81</b>
<b>A Seznam použitých zkratk</b>		<b>83</b>
<b>B Obsah přiloženého CD</b>		<b>85</b>

---

## Seznam obrázků

1.1	Moduly SAP R3 [1]	5
1.2	Proces diagram PM [2]	9
1.3	Struktura BSP aplikace [3]	10
1.4	Design Fiori aplikací [4]	12
1.5	Struktura BSP aplikace [5]	13
1.6	Návrhový vzor MVC [6]	14
1.7	Ukázka JSON Objektu	16
2.1	Diagram případu užití pro správu poruch	25
2.2	Diagram případu užití pro správu požadavků na údržbu	26
2.3	Diagram případu užití pro správu prevencí	27
3.1	Task Graph pro aplikaci	32
3.2	Návrh přihlašovací obrazovky	34
3.3	Ukázka designu SAP Fiori Launchpad [4]	34
3.4	Návrh domovské obrazovky	35
3.5	Návrh seznamu poruch pro operátora výroby	35
3.6	Návrh založení poruchy pro operátora výroby	36
3.7	Návrh přihlašovací a domovské obrazovky	36
3.8	Návrh obrazovky pro seznam poruch a jejich založení	37
3.9	Návrh obrazovky pro seznam poruch	38
3.10	Návrh obrazovky pro založení poruchy	38
3.11	Návrh obrazovky pro seznam poruch a jejich založení	39
4.1	Architektura navrženého systému	44
4.2	Seznam funkčních modulů v ERP systému	45
4.3	Seznam funkčních modulů v SAP GW	46
4.4	Ilustrační databázové schéma portálových uživatelů	47
4.5	Struktura balíku realizující autentizaci a autorizaci	50
4.6	Struktura balíku realizujícího middleware vrstvu	51
4.7	Struktura PM aplikace s hlavní prezentační logikou	52

4.8	Struktura jednotlivých view v Eclipse projektu . . . . .	56
4.9	Struktura controllerů v Eclipse projektu . . . . .	57
4.10	Úvodní stránka PM SAPUI5 aplikace . . . . .	61
4.11	Stránka pro založení požadavku na údržbu . . . . .	63
4.12	Stránka pro operátora údržby . . . . .	65
4.13	Stránka pro údržbáře . . . . .	66
4.14	Stránka pro správu uživatelů . . . . .	68
4.15	Stránka pro správu uživatelů - mobilní verze . . . . .	68
4.16	Dialog pro vydání náhradního dílu . . . . .	69
4.17	Stránka pro přihlášení uživatele . . . . .	70
4.18	Úspěšné JUnit testy . . . . .	71
4.19	Neúspěšné JUnit testy . . . . .	72
4.20	Úspěšný OPA5 test . . . . .	73
4.21	Podoba Eclipse . . . . .	74
4.22	Podoba SAP Web IDE . . . . .	75
4.23	Nápověda při výběru komponenty v SAP Web IDE . . . . .	75
4.24	Nápověda při definování atributů elementu v SAP Web IDE . . . .	76
4.25	Diagram agilního vývoje [7] . . . . .	78
4.26	Diagram prototypovacího vývoje [7] . . . . .	78

---

## Seznam tabulek

3.1	Tabulka s hodnocením prototypovacích nástrojů . . . . .	39
4.1	Tabulka shrnující vlastnosti vývojových prostředí pro SAPUI5 . .	76





---

# Úvod

Společnost SAP je v současné době jedním z největších poskytovatelů podnikových aplikací na celém světě. Zaměřuje se na vývoj a provoz softwarových produktů podporujících podnikové procesy a v nedávné době pak především na vývoj technologií pro webové aplikace a cloud computing. Jádrem celého systému je SAP ERP, pomocí něhož dochází k řízení a integrování většiny oblastí činností organizace vlastníci tento systém. Ten se skládá z modulů, které pomáhají řídit společnost v jednotlivých odvětvích, jako například finance, marketing nebo řízení lidských zdrojů. Jedním z nich je i modul údržby SAP Plant Maintenance poskytující nástroje pro údržbu strojů a vybavení dané organizace.

Jedním z dalších produktů je portfolio webových aplikací, které jsou obchodně označovány jako SAP Fiori. Pro jejich funkci musí mít zákazník zakoupen produkt SAP NetWeaver Gateway. Ovšem ne všichni ho mají a přesto by chtěli Fiori aplikace používat.

Společnost itelligence je jedním z předních mezinárodních poskytovatelů řešení SAP a zároveň zadavatelem této práce. Jejím hlavním úkolem je navržení architektury, která umožní komunikaci se systémem SAP ERP i bez produktu SAP NetWeaver Gateway. Konkrétně je pak požadována aplikace nad modulem Plant Maintenance pro vykonávání preventivní údržby a řešení poruch ve výrobním procesu.

## Cíl práce

Cílem práce je představení potřebných komponent systému SAP pro realizaci Fiori aplikací. Analyzovat požadavky kladené na výslednou aplikaci nad modulem Plant Maintenance a v závislosti na nich navrhnout uživatelské rozhraní. Následně důkladně implementovat a zdokumentovat jednotlivé kroky vývoje včetně navržené architektury. Součástí budou i doporučení pro budoucí projekty týkajících se frameworku SAPUI5.

### Co není cílem práce

Cílem této práce není implementace jednotlivých procesů na úrovni modulu údržby. Rozsah této práce končí voláním jednotlivých funkčních modulů, které tyto procesy realizují.

### Struktura práce

V první kapitole *SAP* je nejdříve popsána historie společnosti a její business model. Následně jsou popsány jednotlivé komponenty SAP potřebné pro vývoj Fiori aplikace zahrnující webové technologie i moduly SAP ERP.

Druhá kapitola *Analýza* popisuje funkční a nefunkční požadavky kladené na aplikaci. Následně definuje nejdůležitější případy užití, na základě kterých dojde k rozčlenění aplikace do nezávislých komponent.

Třetí kapitola *Návrh uživatelského rozhraní* zahrnuje rozdělení jednotlivých uživatelských operací do skupin dle navržených komponent aplikace. Následně je zde proveden návrh některých obrazovek ve dvou prototypovacích prostředích. Ty jsou posléze porovnány.

Čtvrtá kapitola *Implementace* je nejobsáhlejší, zahrnuje popis vývoje na všech vrstvách architektury a řeší komunikaci mezi nimi. Je zde popsána struktura aplikace ve frameworku SAPUI5 a porovnány dvě vývojová prostředí umožňující její vývoj. Jsou zde popsány i možnosti testování v rámci komunikační mezivrstvy realizované pomocí Java servletu a samotného UI frameworku SAPUI5.

# SAP

Kapitola obecně popisuje podnikový informační systém SAP a trochu podrobněji se věnuje částem systému týkajících se požadované aplikace. V jednotlivých podkapitolách jsou nejdříve popsány informace o historii firmy a architektonické struktuře systému. Dále jsou zde popsány i jednotlivé technické komponenty, které jsou použity pro realizaci požadované aplikace. To se týká především frameworku SAPUI5 a předcházejících webových technologií, které jsou dodnes jeho součástí.

## 1.1 Společnost SAP

Společnost SAP je v současné době jedním z největších poskytovatelů podnikových aplikací a jednou z nejpřednějších softwarových společností na celém světě. Pod zkratkou SAP se schovávají počáteční písmena německých slov „Systeme, Anwendungen, Produkte in der Datenverarbeitung“. Anglicky si lze zkratku přeložit pomocí slov „Systems - Applications - Products in data processing“. Zaměřují se na vývoj a provoz softwarových produktů podporujících podnikové procesy. Zejména se pak jedná o řízení podniku, systémy pro správu vztahu se zákazníky a v současné době pak především vývoj technologií pro webové aplikace a cloud computing [8].

## 1.2 SAP ERP

První systém, vydaný společností již v roce 1973, měl obchodní označení SAP R/1 a byl tvořen pouze finančním účetnictvím. Následující verze SAP R/2 byla vydána o šest let později a dá se již označovat za první funkční ERP systém (Enterprise Resources Planning). Tedy informační systém určený pro plánování podnikových zdrojů, pomocí kterých dochází k řízení a integrování většiny oblastí činností organizace jako jsou zásoby, nákup, prodej, marketing, finance nebo personalistika a podobně [9]. Ovšem nevýhodou takového sys-

## 1. SAP

---

tému v této době byla vysoká technická náročnost na klienta, která s sebou přinášela nutnost využívání sálových počítačů. S verzí SAP R/3 z roku 1992 byla však kompletně změněna architektura systému. Změnou architektury na klient-server opadly vysoké nároky na klienta a s tím i nutnost využívání sálových počítačů. Další změnou bylo i zavedení relačních databází. Hlavní výhoda této architektury však spočívala v kompatibilitě s různými platformami a operačními systémy Microsoft Windows nebo Unix. Tímto systém se společnost dostala na špici poskytovatelů ERP systémů a toto místo si drží dodnes [8].

**Architektura systému SAP R/3** Spolu se změnou architektury, spojenou s příchodem verze SAP R/3 na model klient-server, došlo k rozčlenění systému do tří vrstev. Tento model bývá označován jako **SAP Web Application Server** (dále již jen SAP Web AS). Jednotlivé vrstvy jsou popsány v následujícím architektonicky odspodu seřazeném výčtu. Informace v tomto bloku se zakládají na článku [10] popisující jádro SAP.

- **Databázová vrstva:** Je tvořena databázovými servery, které slouží pro ukládání dat. Ty mohou být nainstalovány na odděleném serveru a tvořit tak samostatnou architektonickou vrstvu. I přes to, že je SAP multiplatformní systém, vývojáře nemusí zajímat, na jaké platformě (Oracle, Microsoft SQL, nebo jiné) databázová vrstva běží, protože na aplikační vrstvě bude přístup vypadat vždy stejně.

Příkladem mohou být uložená data týkající se vybavení tovární haly, která jsou roztroušená po jednotlivých databázových tabulkách.

- **Aplikační vrstva:** Slouží především jako prostředí pro vykonávání programové logiky na straně aplikačního serveru. Zpracovávají se na něm data, které se načítají a ukládají z databázové vrstvy. Jednotlivé programy (funkce) se na této vrstvě zpravidla píšou v programovacím jazyce ABAP (Advanced Business Application Programming) vyvinutém společností SAP.

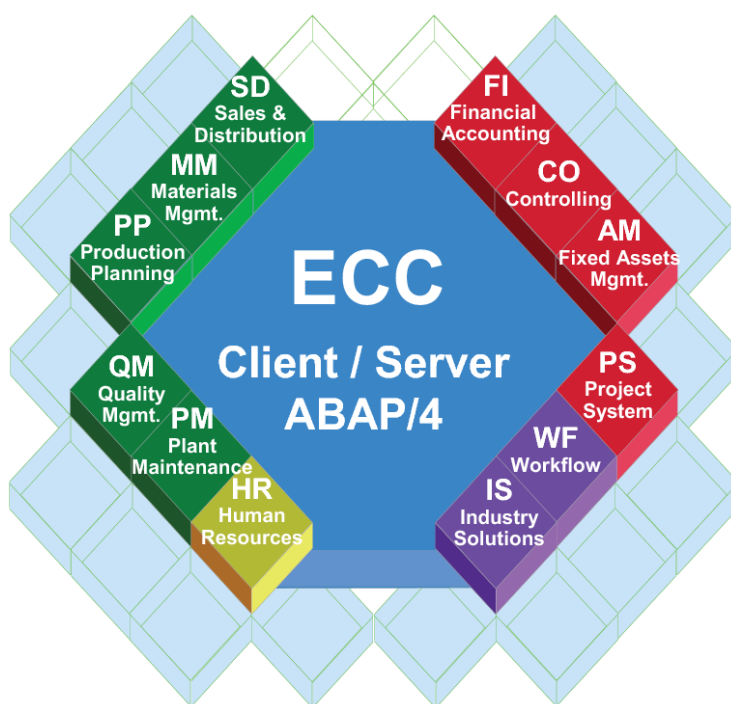
Příkladem budiž požadavek o vyčtení zakázek na výrobní stroj pro daného uživatele. Aplikační vrstva nejdříve musí načíst z databázové vrstvy potřebná data o uživateli, strojích v tovární hale a zakázkách. Ty poté musí dle specifikace požadavku zpracovat. Vyloučí tak například stroje, ke kterým uživatel nemá oprávnění nebo nerelevantní zakázky k danému stroji a času. Tím získá požadovanou informaci, která je například následně přenesena do prezentační vrstvy.

- **Prezenční vrstva:** Má za úkol předávat informace uživateli. Vlastní komunikace probíhá na prezentačním serveru, tedy klientské části. Její nedílnou součástí je rozhraní SAP GUI, které se stará o komunikaci mezi prezentačním a aplikačním serverem. To je ovšem v poslední době nahrazováno modernějším přístupem přes webový prohlížeč. Typickým

příkladem jsou aplikace SAP Fiori vytvořené ve frameworku SAPUI5, kterým je dále věnována samostatná kapitola 1.5.

### 1.2.1 Moduly SAP R3

Systém SAP R/3 je vnitřně rozdělen do dvanácti různých modulů a každý z nich řeší konkrétní problematiku organizace. Jednotlivé moduly jsou mezi sebou více či méně provázány. Úroveň provázanosti jednotlivých modulů je pak barevně znázorněna na obrázku 1.1. Výčet a krátká charakteristika modulů následuje ihned po něm.



Obrázek 1.1: Moduly SAP R3 [1]

- **Financial Accounting (FI):** Označuje finanční účetnictví a je jedním z nejdůležitějších modulů SAP ERP. Používá se k ukládání finančních dat organizace a pomáhá k analyzování jejího finančního postavení na trhu.
- **Controlling (CO):** Jeho primárním účelem je plánování a následná kontrola vytvořených plánů uvnitř organizace. Podporuje koordinaci, monitorování a optimalizaci všech procesů. Zahrnuje správu a konfiguraci základních dat, které pokrývají náklady a výnosy, interní objednávky a další nákladové prvky.

## 1. SAP

---

- **Asset Management (AM):** Slouží k optimální správě fyzického majetku organizace. Zahrnuje takové funkcionality jako jsou návrh, konstrukce, provoz nebo údržba a výměna zařízení.
- **Project system (PS):** Je nástroj pro správu dlouhodobých projektů. Umožňuje plánovat finanční prostředky i zdroje a kontrolovat jednotlivé části projektu tak, aby bylo zaručeno včasné dodání, pokud možno v rámci rozpočtu.
- **Workflow (WF):** Umožňuje navrhovat a realizovat obchodní procesy v rámci aplikačních systémů SAP. Zajišťuje, aby se práce v požadovaný čas dostala do rukou správným lidem. Jeho cílem je usnadnění automatizace podnikových procesů.
- **Industry Solutions (IS):** Poskytuje specifická řešení pro desítky industriálních odvětví jako například automobilový, chemický či energetický průmysl.
- **Human Resources (HR):** Umožňuje organizaci strukturálně a efektivně zpracovávat informace týkající se zaměstnanců. Poskytuje přehled o lidských kapacitách a nástroje k jejich strategickému rozvoji.
- **Plant Maintenance (PM):** Poskytuje nástroj pro provádění veškerých potřebných činností týkajících se údržby organizace a jejích součástí. Umožňuje plánovat údržbu i s ohledem na materiálovou potřebu, zaznamenávat a vyrovnávat náklady spojené s činnostmi.
- **Materials Management (MM):** Se zabývá řízením materiálů a skladových zásob. Kontroluje, aby nedocházelo k nedostatkům zboží a nevznikaly tak mezery v řetězci dodavatelského procesu.
- **Production Planning (PP):** Sleduje a zaznamenává toky ve výrobním procesu. Má za úkol sladění poptávky s výrobní kapacitou spolu s vytvořením plánů k dokončení jednotlivých komponent a produktů.
- **Quality Management (QM):** Je modul úzce provázaný s moduly MM, PP či PM a nedílnou součástí logistického řízení. Používá se k provádění kvalitativních funkcí jako je například vyhodnocení výkonnosti a spolehlivosti procesů, počtu reklamací od zákazníků atd. Důslednou kontrolou produktů je dosaženo vyšší spokojenosti zákazníka.
- **Sales and Distribution (SD):** Se používá pro ukládání údajů o zákaznících a produktech organizace. Pomáhá řídit fakturaci, prodej a distribuci produktů či služeb organizace. Řídí vztah se zákazníky od počáteční nabídky až po prodejní zakázku a fakturaci produktu.

Kombinací všech těchto modulů potom vzniká velmi komplexní a modulární systém, který se dá ještě rozvíjet a upravovat dle zákaznických požadavků. Výsledkem je pak podnikový informační systém „ušitý na míru“.

## 1.3 SAP Plant Maintenance (PM)

Modulem, který je pro tuto práci stěžejním, je modul SAP Plant Maintenance. Jedná se o komplexní řešení, které poskytuje nástroje pro údržbu strojů a vybavení v rámci celé organizace. Veškeré aktivity jsou vzájemně propojeny s návaznými moduly (Production Planning, Materials Management a Sales and Distribution) v rámci podnikových procesů. Modul umožňuje provádět plánování, realizovat denní činnosti údržby nebo zaznamenávat vzniklé problémy a poruchy. Díky provázanosti s ostatními moduly taktéž dovede sledovat a plánovat materiálové aktivity, případně určovat náklady vzniklé údržbou. K realizování všech těchto aktivit je modul rozdělen do následujících podmodulů. Hlavním zdrojem informací pro tuto část je dokument [2] popisující většinu základních procesů v modulu PM.

- Správa technických objektů a vybavení
- Plánování úkolů údržby
- Řízení notifikací v rámci nastavených procesů

### 1.3.1 Technické objekty

Pro správné a efektivní provádění jednotlivých aktivit v rámci procesů modulu PM je struktura údržby rozdělena na technické objekty. Ty slouží k definování jednotlivých typů strojů, kterými organizace disponuje. S použitím charakteristiky technických objektů lze pak nadefinovat jiný technický objekt, což umožňuje hierarchické definování struktury organizace. To s sebou přináší následující výhody.

- Doba potřebná pro správu jednotlivých technických objektů je snížena.
- Zpracování údržby je zjednodušeno.
- Konkrétnější, důkladnější a rychlejší vyhodnocení údajů o údržbě.

Klíčová funkcionality systému se dá shrnout do následujících bodů. Jednotlivým bodům se poté věnují podrobnější podkapitoly.

- **Inspekce:** Spočívá v měření a sledování aktuálního stavu technického objektu.
- **Preventivní údržba:** Slouží k zaručení vysoké spolehlivosti výrobního procesu. Zahrnuje plánování údržby a časování aktivit pro jednotlivé technické objekty.

## 1. SAP

---

- **Oprava poruchy:** Zahrnuje veškerá opatření, která jsou nutná k obnovení ideálního stavu technických objektů. Cílem je maximální eliminace prostojů ve výrobě a snížení nákladů na výrobu.

### 1.3.2 Inspekce a preventivní údržba

Preventivní údržba je dlouhodobý a plánovaný proces, jehož cílem je zajištění vysoké použitelnosti zařízení a minimalizování prostojů způsobovaných opravami. Pomocí preventivního chování lze potom v organizaci dosáhnout řady výhod. Dosažením maximální využitelnosti strojů lze snížit jejich počet blíže k minimu a tím tak dosáhnout menších nákladů na pořizování technologií a tím i snížení potřebných kapacit lidských zdrojů. Vybavení ve špatném stavu má negativní dopady na morálku zaměstnanců a jejich výkonnost. I to je jeden z dalších důvodů, proč je dobré mít nastavené správné preventivní procesy. Pro stanovení takového procesu lze určit řadu specifikujících parametrů. Příklady takových parametrů jsou nastíněny v následujícím seznamu.

- Seznam úkolů, které mají být v rámci prevence provedeny
- Rozsah inspekčních a preventivních prací
- Frekvence kontroly
- Limit nákladů spojených s preventivní údržbou

Seznam úkolů pro preventivní údržbu je definován jako sled činností, které jsou prováděny zodpovědnými zaměstnanci. Pomocí pevně daného seznamu úkolů lze standardizovat pracovní postupy a tím tak dosáhnout vyšší efektivity. V rámci systému je potom v závislosti na aktuálním čase prevence dělena na plánovanou a přetrvávající údržbu.

**Plánovaná údržba** Všechny plánované aktivity, jakou jsou údržba, prevence, inspekce a opravy spadají pod plánovanou údržbu. V modulu PM jim mohou být přiřazeny časové intervaly vykonávání údržby. Čas může být přidělen až na úrovni jednotlivých úkonů. Příkladem tak může být úkon pro dotažení šroubů na výrobní lince XYZ každé druhé pondělí ve 14 hodin.

**Přetrvávající údržba** Seznam úkonů přetrvávající údržby je založen na základě aktuálního stavu inspekce. Všechny takové, které jsou provedeny mimo regulérní rozvrh, spadají pod přetrvávající údržbu.

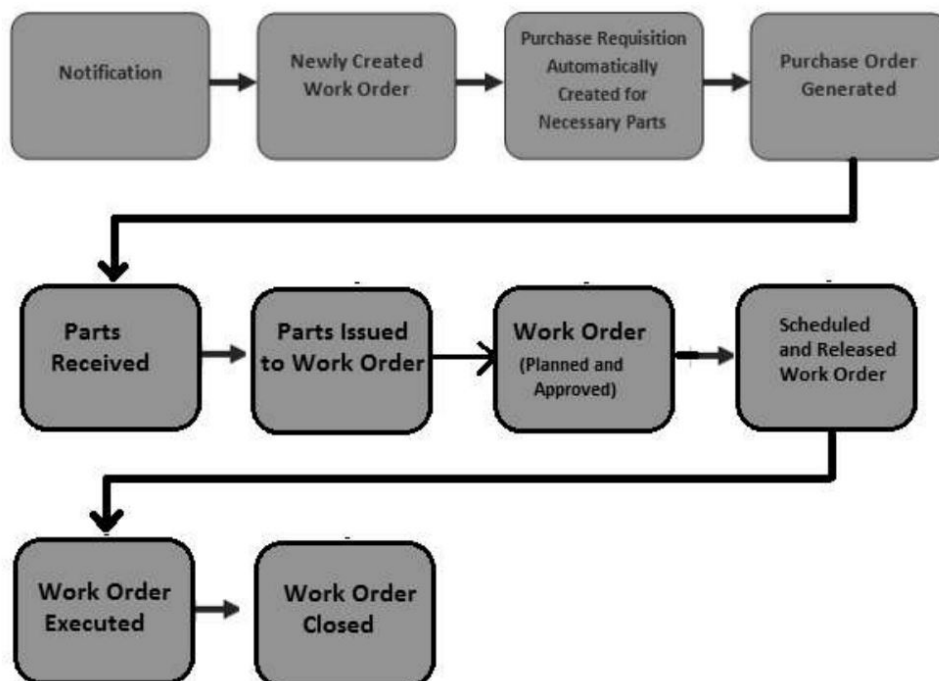
**Údržba z technického hlediska** Zpracování údržby se z technického pohledu skládá z několika fází, jejichž pořadí nemusí být striktně dodrženo. Jelikož se údržba týká plánování, předběžné kalkulace, zabezpečení dostatečného množství materiálu a případných povolení, může v případě nutného provedení



údržby dojit k přeskočení některých kroků. Mohlo by totiž například dojit k nežádoucímu čekání na schválení kalkulace i v případě nutnosti opravy. Zpracování údržby lze rozdělit na následující tři oblasti:

- **Popis stavu objektu:** Slouží k podání oznámení o údržbě, používá se k popisu stavu technického objektu nebo k nahlášení poruchy spolu s následným požadavkem na opravu.
- **Provádění úkolů údržby:** Spočívá v objednání údržby. Používá se k detailnímu plánování provádění údržbářských činností, sledování průběhu práce a vypořádání nákladů na údržbu.
- **Dokončení úkolů údržby:** Slouží zejména k zaznamenávání historie údržby. Používá se k dlouhodobému uložení nejdůležitějších údajů o údržbě, které lze následně z různých důvodů analyzovat.

Na obrázku 1.2 je zobrazen životní **cyklus požadované údržby, případně nahlášené poruchy**.



Obrázek 1.2: Proces diagram PM [2]

V prvním kroku musí dojit k podání požadavku na údržbu. Poté technicky dojde k založení objednávky se všemi náležitostmi. Tu posléze může realizovat některý z údržbářů, který má možnost daný problém řešit. Počátkem jeho

## 1. SAP

---

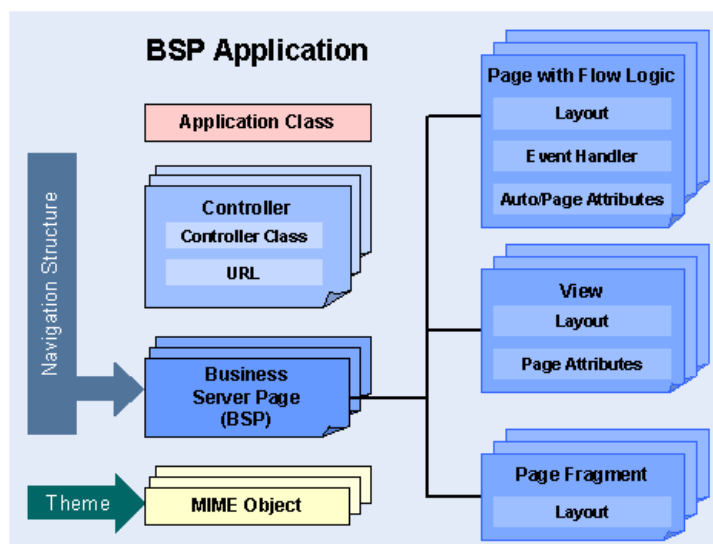
práci dojde k založení servisní zakázky, pro kterou díky propojení na ostatní moduly probíhá kalkulace nákladů. Jelikož k vyřešení problému může údržbář potřebovat různý materiál (například šrouby a matice), je v procesu zobrazeno vytvoření objednávky na vydání materiálu. Po případném vyzvednutí materiálu a dokončení údržby dojde k ukončení servisní zakázky a celého procesu údržby. Závěrem je celý proces zaznamenán a finančně vyhodnocen.

Tím je popsán základní proces nad modulem PM potřebný k pochopení a následnému návrhu aplikace, která bude tento, mimo jiné, proces realizovat.

### 1.4 SAP BSP

Tato sekce se zaměřuje na frontend technologii SAP BSP. Jedná se o jednu z technologií použitých v cílové architektuře řešení mobilní aplikace, a proto je zde stručně popsána její struktura pro pochopení její role v celkové architektuře.

**BSP (Business Server Page)** Je jednou z variant, jak SAP programy a transakce zpřístupnit i mimo desktop aplikaci SAPGUI běžně používanou pro přístup do systému. Představuje funkční HTML aplikaci, u které lze dosáhnout takového chování jako u klasické SAP transakce. Tím je umožněno získat přístup do systém skrze libovolný webový prohlížeč. Na následujícím obrázku 1.3 je k vidění struktura BSP aplikace.



Obrázek 1.3: Struktura BSP aplikace [3]

Každá taková stránka se skládá z layoutu, do kterého je možné vložit standardní HTML komponenty (head, body, input a podobně). Kromě toho

je možné do takové stránky vložit i kód programovacího jazyka ABAP. Tím je vytvořen silný nástroj umožňující dynamické tvoření hierarchie stránky v závislosti na získaných datech. Jelikož se jedná o HTML stránku, tak je možné v každé BSP stránce zpracovávat vstupní data. Ty jsou získány buď pomocí metody HTTP(s) GET (informace se nese v uživateli zadané URL) nebo pomocí HTTP(s) POST (informace zakódovány v těle HTTP requestu).

Každá taková stránka má několik událostí. V následujícím výčtu jsou popsány relevantní pro vývoj požadované aplikace [3].

- **OnCreate:** Událost je vyvolána v případě prvního načtení v rámci session. Může posloužit k vytvoření potřebných tříd nebo načtení neměnných dat.
- **OnRequest:** Nejdůležitější událost z hlediska vývoje požadované aplikace. Je vyvolána při každém requestu vůči BSP aplikaci. V tento moment lze vyčíst zasílaná data a v závislosti na nich provést adekvátní operace. Příkladem může být přijetí requestu s požadavkem na detail uživatele MORMAITE. Pomocí kódu v programovacím jazyce ABAP lze z databázových tabulek vyčíst požadované informace a uživateli je nějakým způsobem zobrazit zpátky.
- **OnDestroy:** Slouží především pro uvolnění použitých prostředků. To sice není nezbytně nutné, systém to po čase provede sám. Nicméně se vůči systému jedná o nenáročnost slušnost.

## 1.5 SAP FIORI

Systém SAP byl po dvě desetiletí přístupný zejména prostřednictvím desktop aplikace SAPGUI. Jedná se tedy o aplikaci navrženou v 90. letech 20. století a tomu odpovídá i její vzhled. Nejedná se zrovna o User-Friendly rozhraní, a to si SAP management samozřejmě uvědomuje. Kromě toho se nejedná zrovna ani o nejdostupnější řešení. Proto v roce 2013 společnost SAP přišla platformou SAP Mobile Platform 3.0 startující novou éru přístupu k systému SAP. Mobilní platforma dostala obchodní název Fiori. Bylo založena na základě následujících pěti pravidlech [4].

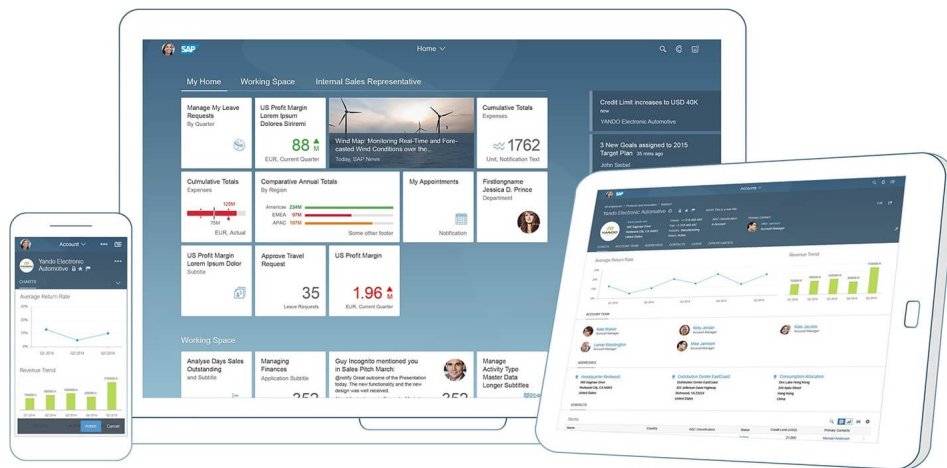
- **Založené na rolích:** Má se jednat o uživatelsky orientované aplikace závislé na rolích uživatele. Uživatel může mít více rolí a provádět tak zcela rozdílné úkoly v rámci různých aplikací.
- **Responzivní:** Budou založené na HTML5. Musí bez problému pracovat na různých zařízeních bez ohledu na velikost obrazovky. V případě změny rozlišení musí aplikace adekvátně reagovat a přizpůsobit svůj vzhled. A nakonec musí podporovat různé režimy interakce jako jsou klávesnice, myši nebo dotykové vstupy.

## 1. SAP

- **Jednoduché:** Jednotlivé obrazovky musí být jednoduché a zobrazovat relevantní informace. Práce s aplikacemi musí být intuitivní a neměla by vyžadovat návod k zacházení.
- **Koherentní:** Napříč aplikacemi musí být zachován stejný design a princip použití. Snadno se pak uživatel sžije s novými aplikacemi poté, co už s některou Fiori aplikací pracoval.
- **Líbivé:** Aplikace se musí uživatelům líbit, motivuje je to při práci a tím snižuje náklady společnosti.

Zavedením této technologie má dojít k odstranění problémů s nespokojenými zákazníky ohledně staromódního vzhledu a komplikované práce. Společnost SAP od té doby vytvořila stovky Fiori aplikací, které si mohou zákazníci zakoupit. Jedná se například o aplikace pro vyřizování objednávek, správu uživatelů a mnoho dalších.

Na obrázku 1.4 je k vidění aktuální design prezentovaný společností SAP.

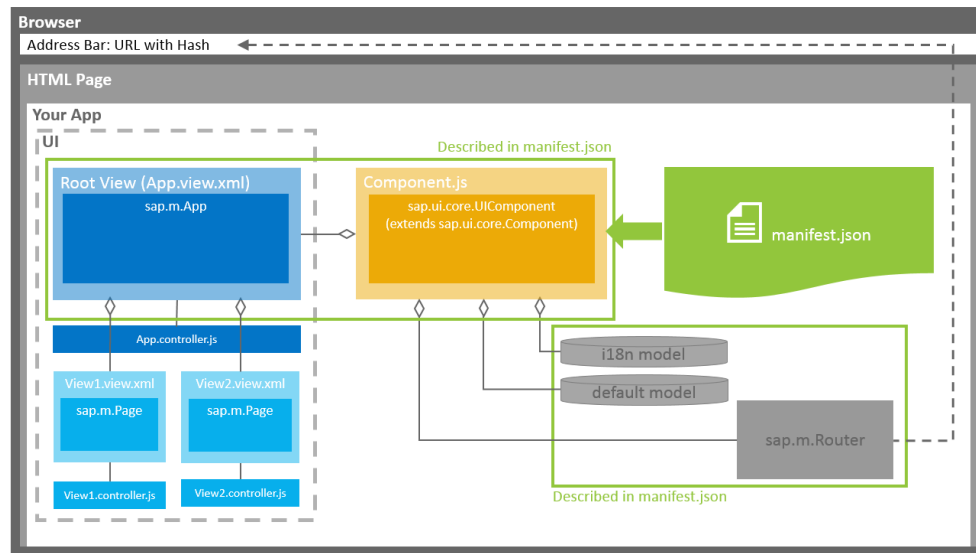


Obrázek 1.4: Design Fiori aplikací [4]

### 1.5.1 Technický pohled

Frameworkem pro tvorbu Fiori aplikací je SAPUI5. Jeho základní struktura je zobrazena na obrázku 1.5 níže. Jedná se o sadu vývojářských nástrojů, které slouží k vytváření aplikací s bohatým uživatelským rozhraním pro moderní business webové aplikace. Jeho základem je JavaScript knihovna Core, která poskytuje zejména prostředky pro podporu MVC 1.5.2 konceptu. Dále nabízí

možnosti provazování dat s UI prvky z různých datových zdrojů (JSON, XML, OData 1.5.3) a efektivní práci s jednotlivými HTML prvky [11].



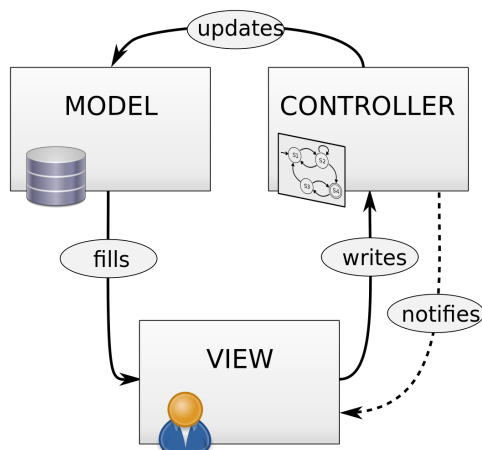
Obrázek 1.5: Struktura BSP aplikace [5]

Struktura frameworku spočívá především v konceptu **MVC**. Při zavádění aplikace dojde k vytvoření hlavní **komponenty** (Component.js), která obaluje jednotlivá **view** nadefinovaná v rámci aplikace. Pod každým view si lze představit libovolnou stránku. Každá taková stránka má přiřazen právě jeden **controller**, který reaguje a obstarává uživatelskou interakci. Klikne-li uživatel na tlačítko vymazat záznam, je to právě controller, kdo odchytl uživatelskou akci a musí na ni adekvátně zareagovat. V takovém případě by nejspíš zobrazil dialog pro potvrzení nebo rovnou poslal na backend request pro vykonání dané operace. Podrobněji se jednotlivým souborům použitým ve frameworku SAPUI5 věnuje implementační část v kapitole 4.5.1.

**UI komponenty** Framework má k dispozici více než 200 komponent, které umožňují skládání výsledné podoby uživatelského rozhraní. Takovou komponentou může být jednoduché zobrazení textu, ale i složitý layout rozdělující stránku na několik nezávislých částí. Obsahuje spoustu typických layoutů potřebných při téměř každém vývoji aplikace. Tím může být například vstupní formulář nebo zobrazení seznamu prvků. Všechny komponenty jsou popsány v dokumentaci [11].

### 1.5.2 MVC

Základem celého frameworku je návrhový vzor MVC (Model-View-Controller). Rozděluje aplikaci na tři nezávislé komponenty. Jedná se o datový mode, uživatelské rozhraní a řídicí logiku. Symbolické znázornění funkčnosti vzoru je k vidění na obrázku 1.6 zobrazeném níže.



Obrázek 1.6: Návrhový vzor MVC [6]

**Model** Reprezentuje správu dat, nad nimiž aplikace pracuje. Takovými daty může být seznam uživatelů, detail objednávky nebo obrázek v binární podobě. Jelikož se z v architektonickém modelu server-klient jedná o prezentační vrstvu, je vhodné, aby data měly co možná nejvíce přijatelnější podobu pro prezentování uživateli.

**View** Zahrnuje všechny grafické objekty uživatelského rozhraní. Jedná se o komponenty, s kterými uživatel přímo pracuje. Pomocí nich mu jsou prezentována požadovaná data v prezentovatelné formě. A naopak pouze jejich prostřednictvím uživatel dává aplikaci příkazy k provedení požadované činnosti. Výhodou jednotlivých komponent je nezávislost na modelu a controlleru. Grafické komponenty lze tak v průběhu aplikace měnit.

**Controller** Dělá prostředníka datovému modelu a uživatelským rozhraním. Má na starost obstarávání uživatelských akcí vůči rozhraní. Klikne-li uživatel na tlačítko, je právě na něm, aby provedl adekvátní akci očekávanou od uživatele. Upravuje data uložená v modelu, čímž může měnit uživateli prezentovaná data. A taktéž může zasahovat do view jako takového. V případě, že je zapotřebí některé elementy schovat nebo naopak zobrazit, je to právě controller, kdo může takovou akci provést [6].

### 1.5.3 SAP ODATA

Protokol OData umožňuje vytváření datových služeb založených na webovém protokolu REST (Representational State Transfer), který umožňuje uživatelům provádět CRUD (Create, Read, Update, Delete) operace nad zdroji identifikovanými pomocí URI (Uniform Resource Identifier) a definovanými v datovém modelu použitím HTTP requestů.

Data přenášená prostřednictvím OData protokolu mohou využívat různé datové formáty běžně používané ve webových technologiích. Příkladem může být XML (Extensible Markup Language), JSON (JavaScript Object Notation) nebo další. Data jsou při přenosu zabalena do protokolu HTTP, případně do jeho zabezpečené verze HTTPS.

Jádrem OData jsou feeds, které jsou kolekcemi složených ze záznamů Entries. Každý Entry prvek je identifikovaný klíčem a reprezentuje strukturovaný záznam, který má seznam vlastností Properties. Ty mohou být komplexního nebo primitivního typu [6].

**EDM (Entity Data Model)** Hlavním konceptem v EDM jsou entity a asociace. Entita je strukturovaný záznam představující libovolný datový objekt. Tím může být například uživatel nebo objednávka. Každá taková entita má svůj klíč, kterému se říká klíč entity. Pomocí tohoto klíče lze asociovat entity mezi sebou. Bude-li objednávka přiřazena k nějakému uživateli, bude mít klíč uživatele u sebe dostupný. Mapováním klíče uživatele na objednávky obsahující jeho klíč dostaneme asociaci uživateli objednávky. Takto lze hierarchicky libovolně navazovat asociace na jiné a tím tak vytvářet požadované datové struktury připravené k exportu.

**JSON (JavaScript Object Notation)** Jedná se o datový formát velmi často využívaný ve frameworku SAPUI5 a je i jednou z podob, ve které lze prezentovat OData. Jedná se o nezávislou datovou reprezentaci, jejíž základní prvek je objekt, který může být:

- Reálnou hodnotou (číslo, text, boolean hodnota true/false nebo null)
- Struktura objektů
- Pole objektů

Jedná se tak o velmi striktní, ale jednoduchou definici objektu. Postupným skládáním jednotlivých objektů do sebe lze vytvořit neomezenou hierarchickou strukturu. Na obrázku 1.7 níže je zobrazena ukázková struktura JSON objektu.

## 1. SAP

---

```
{
  "data": [
    {
      "klic": "K1",
      "klicPopis": "Odpovědné pracoviště",
      "hodnota": "STA"
    },
    {
      "klic": "K2",
      "klicPopis": "Plánovací skupina",
      "hodnota": "TUVL"
    }
  ]
}
```

Obrázek 1.7: Ukázka JSON Objektu

JSON je textový, na jazyce zcela nezávislý formát, využívající však konvence dobře známé programátorům jazyků rodiny C (C, C++, C#, Java, JavaScript, Perl, Python a dalších). Díky tomu je JSON pro výměnu dat opravdu ideálním jazykem [12].



---

# Analýza

Tato kapitola se věnuje analýze a návrhu požadované webové aplikace pracující nad modulem SAP Plant Maintenance pro vykonávání údržby. Jednotlivé podkapitoly se pak zabývají zpracováním požadavků kladených na výslednou funkcionalitu. Dále jsou zde nadefinována omezení aplikace a uživatelské role, které budou v aplikaci použity. Na základě stanovených požadavků je posléze proveden návrh uživatelského rozhraní. Postupy aplikované v této kapitole vychází z přednášek a doporučení popsanych v přednáškách předmětu Softwarového inženýrství [13].

## 2.1 Základní popis aplikace

Webová aplikace bude svým uživatelům umožňovat vykonávat potřebné úkony pro bezproblémový chod výrobních hal. To zahrnuje důležité operace jako je například hlášení požadavků na údržbu nebo případných poruch. Takovou poruchu lze nahlásit na konkrétní vybavení nacházející se na pracovišti. Musí zaměstnancům zodpovědným za údržbu umožňovat evidování práce na nahlášených poruchách. Podrobný soupis funkčních požadavků je popsán v následující kapitole.

## 2.2 Uživatelské role

Uživatelské role popsané v této sekci jsou vytvořeny na základě požadavků v kapitole 2.3. Ovšem pro jejich lepší čitelnost jsou popsány již zde.

### 2.2.1 Údržbář

Uživatel, který primárně řeší plánované údržby z preventivních důvodů. Tím může být například periodicky v modulu PM nastavená kontrola výrobní linky. Jeho povinností je řešení nahlášených požadavků na údržbu nebo poruch.

### 2.2.2 Operátor výroby

Uživatel, jehož primárním účelem je obsluha strojů na jemu přiděleném pracovišti ve výrobním procesu. S aplikací přijde do styku pouze v případě, když bude chtít nahlásit poruchu vybavení v jeho pracovní sekci. Jednotlivá pracoviště jsou vybavena stolními počítači, na kterých jsou spuštěny aplikace potřebné pro provoz. Z těchto aplikací bude umožněna navigace do budoucí webové aplikace pro nahlášení případného problému.

### 2.2.3 Uživatel s možností založení požadavku na údržbu

Uživatel, jehož zodpovědností je bezproblémový chod strojů. Osoba by se dala charakterizovat jako revizní technik, který má na starost obcházení všech pracovišť a kontrolu jednotlivých výrobních linek. V případě, že shledá za vhodné provést na nějakém stroji údržbu, založí adekvátní požadavek. Tento úkon se bude zpravidla provádět z přenosného zařízení, které má pracovník neustále u sebe. Tím může být například chytrý telefon nebo tablet.

### 2.2.4 Správce - administrátor

Uživatel zodpovědný za správu ostatních uživatelských účtů. Pomocí rolí bude definovat možnosti jednotlivých uživatelů. Jelikož role nejsou uživatelsky výlučné, budou mít dvě fyzické osoby představující údržbáře možnost provádět jiný okruh úkonů. Oba dva budou moci například vykonávat údržbářské činnosti, ale jenom jeden z nich bude moci zakládat požadavky na údržbu.

## 2.3 Model požadavků

V této sekci jsou uvedeny veškeré požadavky kladené na výslednou aplikaci, které byly probírány se zadavatelem. Požadavky představují minimální kritéria potřebná pro samotný návrh uživatelského rozhraní. Veškeré požadavky byly probírány se zadavatelem, většina z nich byla jasně stanovena v rámci rámcového zadání, některé však byly lehce v rámci konzultací během vývoje aplikace. Cíle při vytváření kteréhokoli modelu požadavků jsou vypsány v následujícím výčtu.

- **Vymezení hranic aplikace:** Slouží ke stanovení oblastí, pro kterou bude aplikace sloužit. Příkladem takové hranice nad modulem PM může být práce pouze pro zakládání požadavků na údržbu a poruch. Od takové aplikace se potom nebude očekávat správa technického vybavení.
- **Odhad pracnosti:** Za pomoci stanovené funkcionality a ostatních požadavků je mnohem jednodušší stanovit odhad celkové pracnosti. nad seznamem jednotlivých funkcionalit se provede jednodušší odhad a rezerva potřebná pro vývoj.

- **Vyjasnění zadání:** Ne vždy bohužel zákazník dobře ví, co vlastně od aplikace očekává. Proto je dobré si s ním dané funkcionality prodiskutovat a následně pevně stanovit.
- **Zachycení omezení:** Slouží k předejití fatálních problémů při vývoji. Je zapotřebí zabránit problémům, které se například až v době nahrávání aplikace do produktivního prostředí, protože je zde pro server použita jiná verze operačního systému nebo jsou zde problémy s oprávněním a přístupem do internetu.

Jednou z pomůcek pro dobré definování cílů aplikace je technika **SMART**. Ta popisuje vlastnosti cílů pomocí následujícího výčtu [14].

- **S (Specific):** Musí být definován přesně. Čím přesnější cíl bude, tím snadněji bude docházet k jeho implementaci a předejde se tak možným nedorozuměním. Obecně platí, že to, co je jasné jednomu člověku nemusí být pochopeno na opačné straně stejně.
- **M (Measurable):** Aby bylo možné dosažení cílů posoudit, měli by být dobře měřitelné.
- **A (Accepted):** K akceptaci musí dojít odpovědnou osobou. Dokud nebude mít tým potvrzený cíl, tak si jednotliví členové vždy najdou něco „zajímavějšího“ na práci.
- **R (Realistic):** Měli by být uskutečnitelné v reálném čase. Pevně stanovený termín je sice nezbytný, nicméně by se nemělo jednat o nesplnitelný termín. Ten by mohl pracovní morálku týmu jenom zhoršit.
- **T (Timed):** Časové omezení je nezbytně nutné. Pokud nebude jasné stanoven termín dodání bude mít tendenci jednotlivé úkony odkládat.

### 2.3.1 Funkční požadavky

Jsou takové požadavky, které musí být ve výsledné aplikaci implementovány tak, aby byla splněna očekávaná funkcionality. Požadavky jsou rozděleny do sedmi sekcí označených jako F1 až F7. Jedná se především o funkcionality spojené s nahlašováním poruch nebo požadavků na údržbu společně s úkony prováděných nad již vzniklými hlášeními.

#### 2.3.1.1 F1: Založení požadavku na údržbu

V případě zjištění potřeby údržby daného vybavení bude uživateli s oprávněním tuto činnost provádět k dispozici založení hlášení v modulu PM s následující editovatelnými parametry.

- **Vybavení:** Výběr bude umožněn pomocí hierarchické struktury představující podobu závodu. V případě, že bude uživateli přednastaveno výchozí technické místo (například pracovní linka), bude výběrová struktura příslušně omezena. Provedení výběru by mělo být v mobilním zařízení umožněno i pomocí naskenování QR kódu.
- **Příloha:** Ke každému požadavku bude umožněno přiložené jedné přílohy s tím, že prozatím bude omezeno pouze na fotografie (omezený výčet typů souborů). Do budoucna se počítá s rozšířením na vyšší počet příloh.
- **Priorita:** Stanovuje termín, do kterého se očekává provedení požadované údržby nebo opravy. Řešeno pomocí tří úrovní důležitosti, podle kterých se očekává zpracování požadavku v horizontu dne, týdne nebo měsíce.
- **Plánovací skupin:** Spočívá ve výběru subjektu zodpovídajícího za údržbu. Fyzicky se jedná o skupinu lidí spravující vymezený okruh údržby (například elektromechanici, mechanici nebo revizní technici). V případě, že bude uživateli přednastavena výchozí plánovací skupina, dojde automaticky k jejímu předvyplnění.
- **Pracoviště:** Reprezentuje skupinu údržbářů, kteří jsou podřízeni příslušnému mistru údržby. Z důvodu propojení s modulem CO dochází s návazností na pracoviště k ocenění prováděné práce, která je vykazována pomocí zpětného hlášení daným pracovištěm na zakázku PM.
- **Popis poruchy:** Jedná se o stručný popis hlášení, který jasně přibližuje daný problém. Tím může být například došlý materiál nebo opotřebení šroubu.

Parametry, které uživatel nebude svévolně volit jsou následující.

- **Typ hlášení:** Vychází z prováděné operace, je stanoven konstantou určující typ hlášení požadavku na údržbu.
- **Závod:** Je definován přihlášeným uživatelem. Každý uživatel bude mít nějaký závod přidělený, jedná se o potřebný údaj při zakládání hlášení.

Typickými uživateli využívající tento funkční požadavek budou mistři, vedoucí směn a údržbáři.

### 2.3.1.2 F2: Nahlášení poruchy

Jelikož se technicky na úrovni modulu PM jedná o stejný záznam jako při zakládání požadavku na údržbu (rozdílná konstanta typu hlášení), je seznam

potřebných parametrů stejný. Nicméně jelikož se hlášení poruchy reálně očekává od jiného typu uživatelů, je zadání hodnot u následujících parametrů trochu odlišné. Typicky totiž bude poruchu nahlašovat pracovník ve výrobě, který má přidělené své pracovní místo a spolu s tím omezené možnosti zadávání.

- **Vybavení:** Jelikož se očekává mnohem menší počet vybavení, které bude umožněno uživateli vybrat, nebude se provádět výběr z hierarchické struktury technických míst, ale bude k dispozici jenom takové vybavení, které spadá pod určité pracoviště. Hierarchické uspořádání však zůstane zachováno. Provedení výběru by mělo být v mobilním zařízení umožněno i pomocí naskenování QR kódu.
- **Plánovací skupin:** Uživatel bude mít na výběr hlášení poruch dvojího typu. Bude na něm, jestli si vybere poruchu s mechanickou nebo elektrotechnickou příčinou. V závislosti na tom bude plánovací skupina přednastavena z uživatelského nastavení.

Needitovatelné parametry budou stejně jako při zakládání požadavku na údržbu přednastaveny z uživatelského nastavení.

#### 2.3.1.3 F3: Správa poruch

Uživateli musí být k dispozici seznam poruch obsahující potřebné informace (popis hlášení, technické místo spolu s vybavením, pracoviště a informace o tom kdo a kdy poruchu nahlásil včetně statusu daného hlášení) pro správné zacházení s nimi. Uživateli budou nad jednotlivými poruchami umožněny následující operace. Některé z nich jsou k dispozici v závislosti na stavu (statusu) hlášení.

- **Evidence práce na poruše:** Údržbář (člověk s oprávněním provádět opravy) může na konkrétní poruše zahájit práci. To je technicky realizováno založením PM zakázky, u které díky integraci na modul CO dochází k evidenci nákladů. Takový uživatel může poté práci na svojí zakázce ukončit.
- **Zrušení hlášení:** V případě založení poruchy (status odpovídající stavu právě založeno) je umožněno hlášení zrušit. To například pro nevhodné nebo nechtěné založení.
- **Vyřešení poruchy:** Poté, co se poruše začal někdo věnovat (zahájil na ní práci a tím došlo k založení zakázky PM), svou práci následně ukončil a nikdo jiný už na ní nepracuje, je možné poruchu ukončit. Poté bude údržbář vyzván k odborné specifikaci poruchy. Dostane za úkol specifikovat postiženou část objektu, typ poškození a příčinu. V takovém případě dojde k ukončení celého procesu a dané hlášení již v seznamu poruch nebude dostupné.

- **Výdej náhradního dílu ze skladu:** Pro provedení této činnosti se uživateli přednastaví z uživatelského nastavení závod a sklad s tím, že sklad bude moci případně změnit. Materiál, množství a možnost navrácení materiálu poté uživatel zadá ručně. Potvrzením zadaných parametrů dojde k vyskladnění požadovaného materiálu ze skladu.
- **Správa akcí:** Hlášení mají jasný výčet operací, které při práci s poruchou může údržbář provést. Jedná se například o informaci objednání náhradního dílu nebo požadovaného servisu. K takové akci pak může dotyčný uživatel dodat vlastní poznámku. Tyto akce mohou být zakládány, ale i zpětně prohlíženy.
- **Zobrazení textů:** Ke každému hlášení je pomocí dlouhých textů (speciální technický objekt pro ukládání řetězců neomezené délky) v SAP umožněno přidávat poznámky. V rámci hlášení bude všem dostupná historie těchto poznámek, přidávání bude umožněno v závislosti na typu uživatele.
- **Zobrazení přílohy:** V důsledku možnosti přidávat přílohu při zakládání poruchy je i v případě práce s poruchou umožněno si danou přílohu zobrazit a eventuálně stáhnout na lokální disk.

### 2.3.1.4 F4: Správa požadavků na údržbu

Uživateli musí být k dispozici seznam požadavků obsahující potřebné informace (popis požadavku, technické místo spolu s vybavením, pracoviště, mezní zahájení spolu s ukončením a informace o tom, kdo požadavek založil včetně statusu daného hlášení) pro správné zacházení s nimi. Uživatelé budou umožněny stejné operace jako při správě poruch 2.3.1.3, kromě změn v následujícím seznamu. Vyřešení poruchy ze správy poruch se požadavků na údržbu netýká.

- **Provedení údržby:** Údržbář může na konkrétním požadavku zahájit práci. Ta je technicky realizována založením PM zakázky, u které díky integraci na modul CO dochází k evidenci nákladů. Po ukončení prací na daném požadavku se bude moci údržbář rozhodnout, zdali je údržba provedena dostatečně a může tak dojít k předání hlášení operátorům výroby pro schválení.
- **Akceptace údržby:** Operátor výroby může rozhodnout o dostatečném provedení údržby daného stroje. V takovém případě dojde k ukončení celého procesu a dané hlášení již v seznamu požadavků na údržbu nebude dostupné.
- **Reklamování údržby:** Operátor výroby může rozhodnout o nedostatečném provedení údržby daného stroje. V takovém případě dojde k navrácení hlášení údržbářům tak, aby mohli na dané údržbě znovu pracovat.

#### 2.3.1.5 F5: Správa prevencí

Uživatelé musí být k dispozici seznam prevencí obsahující potřebné informace (popis prevence, technické místo spolu s vybavením, pracoviště, mezní ukončením a informace o statusu daného hlášení) pro správné zacházení s nimi. Uživatelé budou umožněny stejné operace jako při správě požadavků na údržbu 2.3.1.4, kromě toho, že provedení údržby může provést jak údržbář, tak operátor výroby.

#### 2.3.1.6 F6: Zobrazení dokumentace ke stroji (vybavení)

V závislosti na vybraném technickém místě nebo vybavení dojde k zobrazení seznamu přiložené dokumentace. Ta je uložena na sdíleném úložišti v interní síti společnosti a bude tedy dostupná pouze v případě použití aplikace uvnitř dané sítě. Výběr technického místa nebo vybavení bude umožněn z hierarchické struktury.

#### 2.3.1.7 F7: Administrace uživatele

Bude umožněna obecná správa účtu uživatelů, tedy základní operace jako přidání nebo odebrání účtu, nastavení jména uživatele a osobního čísla odpovídajícímu identifikátoru zaměstnance v ERP. Administrátorský účet bude moci měnit nastavení ostatních účtů, bude přiřazovat uživatelské role (oprávnění k zacházení s jednotlivými funkcionalitami) a parametry charakterizující daného uživatele. Pod tím se schovává nastavení závodu, plánovací skupiny, předdefinovaného technického místa a dalších atributů ulehčujících uživateli práci s aplikací (například přednastavené hodnoty pro výběr pracovišť, plánovacích skupin při zakládání požadavků na údržbu nebo hlášení poruch). V případě ztráty uživatelského hesla, bude z administrátorského účtu umožněno inicializování hesla daného uživatele.

### 2.3.2 Nefunkční požadavky

Jsou takové požadavky, které nejsou přímo nutné pro splnění požadované funkcionality, nicméně vhodné pro správný chod aplikace. Jedná se například o specifikaci očekávání od designu, zabezpečení nebo dostupnosti systému a dalších pasivních vlastností. Navržené požadavky pro výslednou aplikaci jsou rozděleny do čtyř sekcí označených jako N1 až N4.

#### 2.3.2.1 N1: Grafické uživatelské rozhraní

Uživatelské rozhraní bude dostupné v českém jazyce. Nicméně pro budoucí plánované využití i v zahraničních závodech se očekává snadné rozšíření do ostatních jazyků jako je například angličtina nebo němčina. Jelikož se nejedná o první organizační aplikaci pracující nad nějakým z modulů SAP, požaduje se zachování stejného UI frameworku SAPUI5.

### 2.3.2.2 N2: Dostupnost

Aplikace musí být viditelná v internetu, musí být tedy dostupná z veřejné internetové adresy. Aplikace musí být ovšem plně funkční i ve vnitřní síti, která nemá přístup do internetu. Veškeré zdroje aplikace musí být tedy uloženy na interním serveru poskytujícího run-time prostředí pro webovou aplikaci.

### 2.3.2.3 N3: Spolehlivost a spravovanost

Aplikace bude umožňovat logování činnosti uživatelů z důvodu lepší identifikaci chyb. A to minimálně z počátku provozu aplikace. Taktéž bude i v případě vyššího zatížení systému zajištěno požadováno dokončení transakčních kroků prováděných uživatelem.

### 2.3.2.4 N4: Bezpečnost

Z bezpečnostních důvodů musí v aplikaci docházet k autentizaci a autorizaci každého uživatele. Uživatel bude v aplikaci smět dělat pouze ty úkony, které mu administrátor povolí. Komunikace napříč komponentami musí být šifrována.

## 2.4 Model případů užití (Use Case Model)

Jedná se o detailní specifikaci funkčních požadavků. Typicky slouží pro tvorbu uživatelské příručky, jako jsou podklady k tvorbě akceptačních testů, zpřesnění odhadů pracnosti nebo zadání pro programátora. Zahrnuje informace o tom, kdo bude se systémem pracovat a jaké funkcionality kdo bude využívat. K tomu slouží vydefinovaný seznam účastníků a diagramy případů užití.

### 2.4.1 Seznam účastníků

Níže zmínění účastníci odpovídají standardnímu pracovnímu modelu stanovenému v organizaci a navrženým uživatelským rolím v kapitole 2.2.

- **Údržbář:** Odpovídá navržené roli údržbář 2.2.1, zpravidla bude disponovat i rolí pro zakládání požadavků na údržbu. Je zodpovědný za eliminaci nahlášených poruch a provádění nahlášené údržby jednotlivého vybavení.
- **Operátor výroby:** Vychází z navržené uživatelské role operátor výroby 2.2.2. Na přiděleném pracovišti provádí svoji přidělenou výrobní činnost a v aplikaci mu bude umožněno především hlásit vzniklé poruchy a zakládat požadavky na údržbu vybavení.
- **Správce / Administrátor:** Vychází z navržené uživatelské správce / administrátor 2.2.4. Zodpovídá za správné nastavení uživatelských účtů.



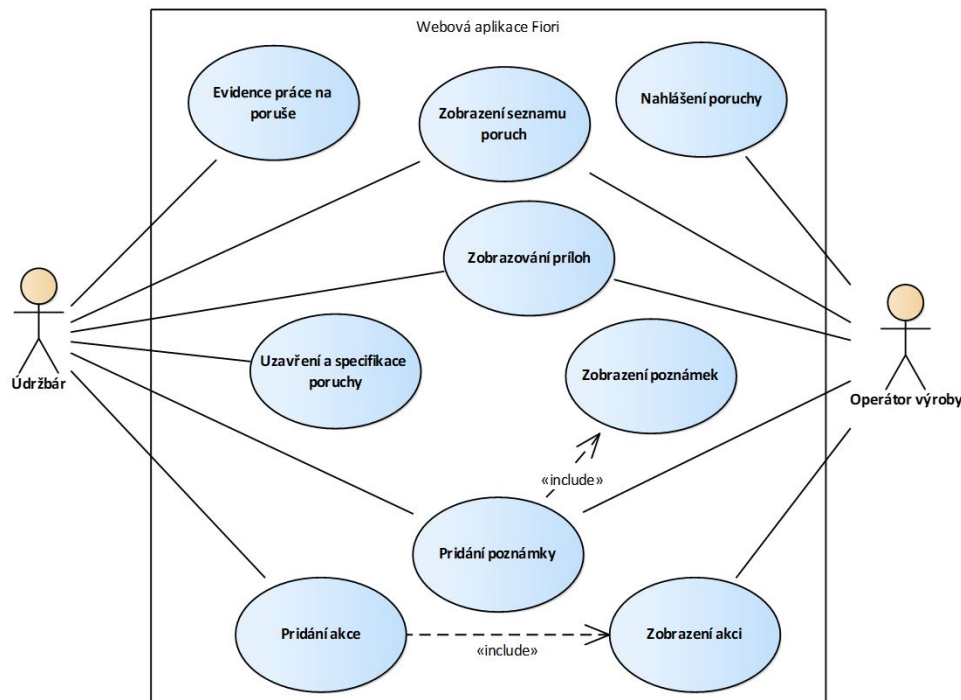
Nicméně nic administrátorovi nebrání tomu role různě kombinovat, nemají mezi sebou totiž výlučný vztah. Administrátor může mít klidně i role údržbáře a operátora výroby, čímž je schopen provádět jim přiřazené operace.

### 2.4.2 Diagram případů užití

Slouží pro detailní specifikaci požadavků na systém s tím, že graficky zobrazuje účastníky a jejich příslušná oprávnění. V následujících podkapitolách jsou vytvořeny diagramy pro nejdůležitější procesy očekávané d výsledné aplikace.

#### 2.4.2.1 UC1: Správa poruch

Následující případ užití týkající se správy poruch zahrnuje funkční požadavky pro hlášení poruch 2.3.1.2 a jejich následnou správu 2.3.1.3.



Obrázek 2.1: Diagram případu užití pro správu poruch

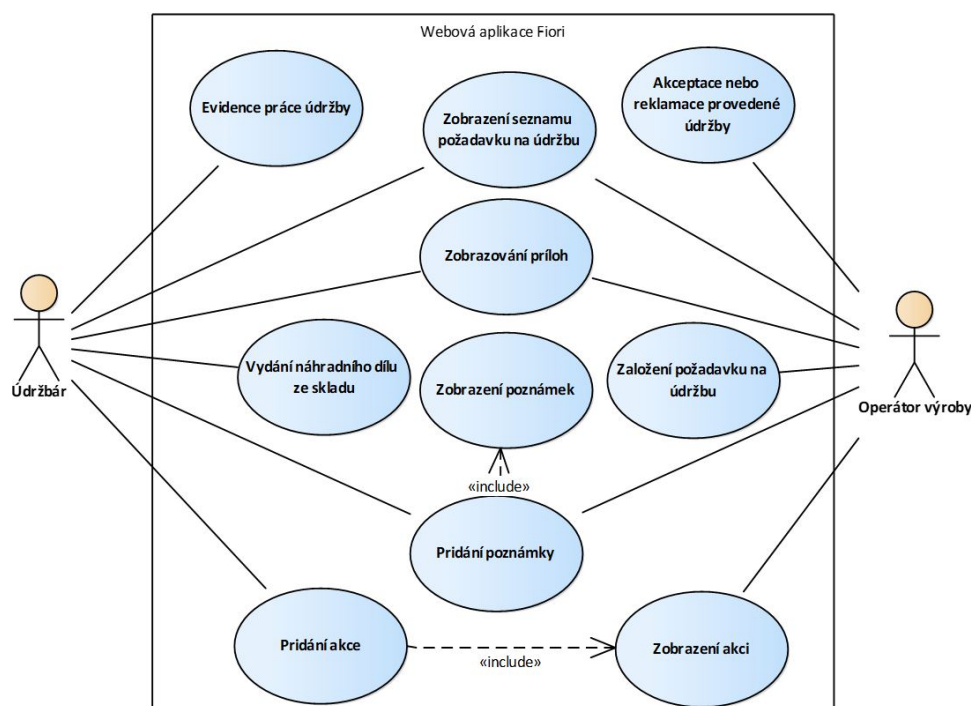
V diagramu je vidět, jak jsou jednotlivé funkcionality mezi údržbáři a operátory výroby provázány. Příkladem může být nutnost zobrazení příslušného seznamu poruch. Dále je zde například vidět, že oba typy uživatelů mohou k hlášením přidávat texty a zobrazovat nad nimi provedené akce. Ovšem pouze údržbář k takovým akcím může přidat novou.

## 2. ANALÝZA

Zásadním rozdílem je ovšem možnost poruchu založit a vyřešit. Nahlášení poruchy spadá do kompetence operátora výroby a vyřešení společně s následnou specifikací poruchy do kompetence údržbáře.

### 2.4.2.2 UC2: Správa požadavků na údržbu

Následující případ užití se týká správy požadavků na údržbu. Zahrnuje funkční požadavky pro zakládání požadavků 2.3.1.1 a jejich následnou správu 2.3.1.4.

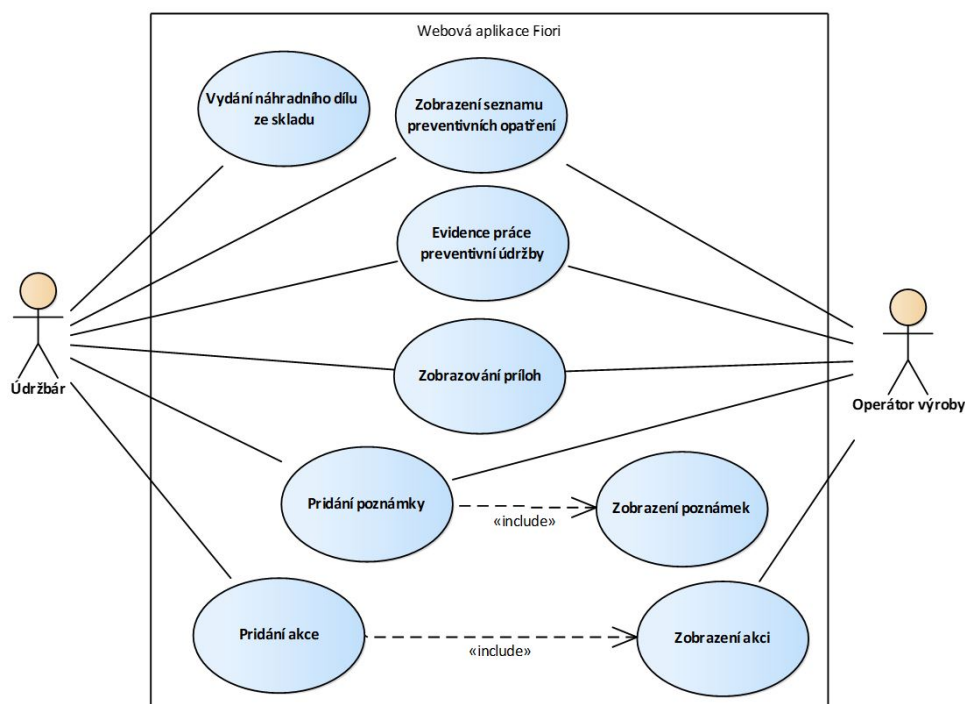


Obrázek 2.2: Diagram případu užití pro správu požadavků na údržbu

V diagramu je vidět přiřazení jednotlivých funkcionalit uživatelům. Příkladem může být přidělení možnosti založení požadavku na údržbu pouze operátorovi výroby. Ten má taktéž přidělenou možnost akceptace nebo reklamace práce provedené údržbářem. Dále je zde například vidět, že oba typy uživatelů mohou k hlášením přidávat texty a zobrazovat nad nimi provedené akce. Ovšem pouze údržbář k takovým akcím může přidat novou.

### 2.4.2.3 UC3: Správa prevencí

Následující případ užití se týká správy prevencí a příslušného funkčního požadavku 2.3.1.5.



Obrázek 2.3: Diagram případu užití pro správu prevencí

Na rozdíl od předchozích dvou případů užití 2.4.2.1 a 2.4.2.2 může preventivní údržbu provést jak údržbář, tak i operátor výroby. To s sebou přináší i možnost zadávání akcí pro operátora výroby. Nicméně provést výdej potřebného materiálu ze skladu může stále pouze údržbář.

## 2.5 Závěry analýzy

Hlavním výstupem celé analýzy je rozdělení aplikace do čtyř komponent odpovídajících uživatelským rolím. Realizování jedné společné komponenty, která by byla sdílena všemi uživateli by s sebou přinášela velké množství komplikací odpočívajících v ošetřování rolí a podmínek, kde se která operace může provést. Rozdělením sice vznikne potřeba realizovat stejné funkcionality na více místech, ale řešení pro tento problém je popsáno v kapitole 4 týkající se implementace.



## Návrh uživatelského rozhraní

Tato kapitola se věnuje průběhu vytváření návrhu uživatelského rozhraní výsledné aplikace. Na základě funkčních požadavků vzešlých z prvních schůzek se zadavatelem a modelu případu užití jsou jednotlivé vzniklé úkony sdruženy v task group. Na základě task group je vytvořen task graph, který v grafické podobě přiřazuje funkcionality k navrženým obrazovkám. Na základě toho je zde vytvořen Lo-Fi prototyp. K tomu byly použity dva nástroje Built.me a Balsamiq Mockups, které jsou následně porovnány. Na základě výsledků z testování Lo-Fi prototypu s uživateli jsou odpovídající změny zaneseny do výsledné aplikace. Postup použitý pro návrh uživatelského rozhraní a metody ověření jeho správnosti vycházejí z přednášek předmětu Návrh uživatelské rozhraní [15]

V první fázi je doporučováno pospat business očekávání a případy užití. Ty jsou ale již zahrnuty v rámci kapitoly 2 věnující se analýze. Proto je zde rovnou k přistoupení dalšímu kroku a to Task Groups.

### 3.1 Task Groups

Jsou založeny na **seznamu operací (Task List)**, které lze v rámci aplikace provést. Prvek takového seznamu může být přidání atributu uživatele nebo schválení provedené údržby. Granularita a důležitost jednotlivých položek pak může být, a zpravidla bývá, proměnná. Základním pravidlem pro tvorbu takového seznamu je rčení „Don't think too much, just list the tasks“, tedy příliš nepitvat, zdali je úkon dostatečně adekvátní nebo naopak zbytečný, ale prostě ho přidat do seznamu. Vypisovat pouhý seznam úkonů možných provést ve výsledné aplikaci je zbytečné. Jednak protože takových operací je mnoho a hlavně proto, že by se výpis duplikoval v následujícím kroku. A tím je rozřazení jednotlivých úkonů do skupin, v rámci kterých je lze provádět. A přesně takový seznam je v rámci této sekce vytvořen. Skupiny odpovídají čtyřem komponentám vzešlých ze závěrů analýzy v kapitole 2.5. Seznam úkonů je z důvodu šetření vertikálního místa co nejvíce agregován do celkových procesů.

#### 3.1.1 Založení požadavku na údržbu

- Popsání požadavku (nastavení priority, plánovací skupiny a dalších potřebných atributů)
- Pořízení dokumentace (vyfocení poškozeného vybavení) a přiložení přílohy
- Výběr vybavení (možnost naskenování QR kódu)
- Založení požadavku

#### 3.1.2 Operátor výroby

- Zobrazení seznamu hlášení (včetně všech akcí, které lze nad nimi provést). Je nutné poskytnout zobrazení hlášení všech již zmíněných druhů (poruchy, údržba a prevence).
- Nahlášení poruch a požadavků na údržbu, které zahrnují následující body.
  - Popsání požadavku (nastavení priority, plánovací skupiny a dalších potřebných atributů).
  - Pořízení dokumentace (vyfocení poškozeného vybavení) a přiložení přílohy.
  - Výběr vybavení (možnost naskenování QR kódu).
- Schválení nebo reklamace provedené údržby.
- Zobrazování akcí, textů a příloh k danému hlášení.
- Zahájení a ukončení práce na preventivní výrobě (spolu s tím i zadávat prováděné akce nad hlášením)

#### 3.1.3 Údržbář

- Zobrazení seznamu hlášení (včetně všech akcí, které lze nad nimi provést). Je nutné poskytnout zobrazení hlášení všech již zmíněných druhů (poruchy, údržba a prevence).
- Zobrazování akcí, textů a příloh k danému hlášení.
- Zahájení a ukončení práce všech typech hlášení. To s sebou obnáší následující body.
  - Možnost zadávat akce k hlášení.
  - Adekvátní ukončení. To v případě údržby spočívá v předání hlášení operátorovi výroby ke schválení, u prevence přímé ukončení poruchy ukončení s vyplněním specifikace.

### 3.1.4 Administrátor

- Zobrazení seznamu uživatelů.
- Přidání nového uživatele. To s sebou obnáší následující body.
  - Zadání uživatelského jména a hesla.
  - Určení typu uživatele.
  - Přiřazení osobního čísla v SAP ERP nově vznikajícímu portálovému uživateli.
- Zobrazení detailu uživatele, které je rozděleno do následujících úkonů.
  - Úprava jména.
  - Změna osobního čísla.
  - Změna hesla.
  - Odebrání uživatele.
  - Přidání (odebrání) uživatelských atributů.
  - Přidání (odebrání) uživatelských rolí.
  - Přidání (odebrání) zodpovědných pracovišť.

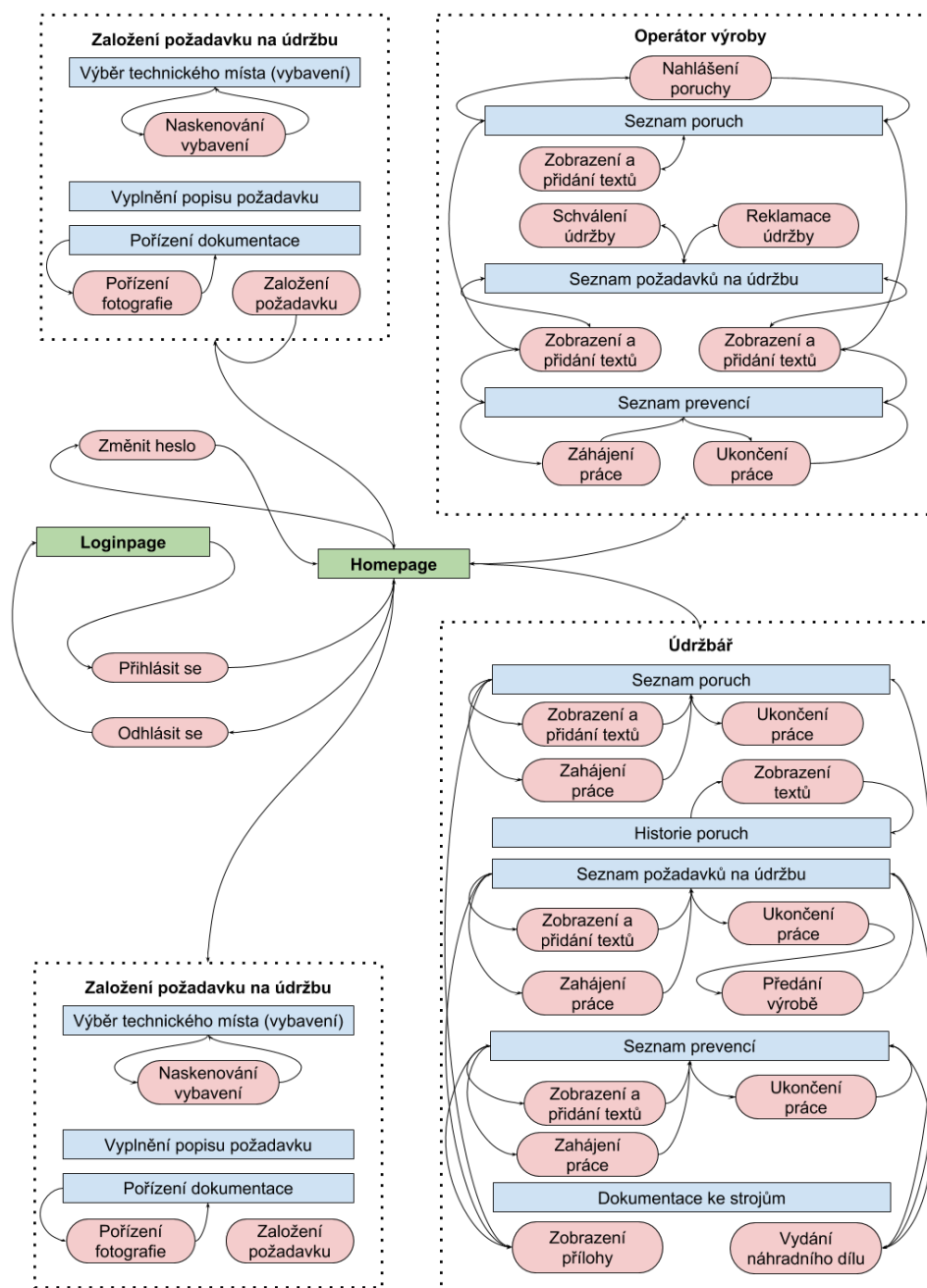
### 3.1.5 Nezatříděné

- Změna hesla.
- Přihlášení a odhlášení.

## 3.2 Task Graph

Jedná se o grafický výstup, jehož vstupem je do skupin rozdělený seznam úkonů. Jako vstup posloužil seznam vytvořený v kapitole 3.1. Nezbytnou součástí výstupního grafu je znázornění možné interakce a pohybu uživatele mezi jednotlivými úkony. Výstupní graf je vidět na obrázku 3.1 zobrazeném níže. Graf má vyznačené dva startovní body, stránky Loginpage a Homepage. Z těch lze v závislosti na přiřazených rolích pokračovat do jednotlivých komponent zahrnujících sadů svých funkcionalit.

### 3. NÁVRH UŽIVATELSKÉHO ROZHRANÍ



Obrázek 3.1: Task Graph pro aplikaci



### 3.3 Lo-Fi prototyp

Lo-Fi (Low-Fidelity) prototypy slouží pro snadné a hlavně rychlé znázornění pochopení konceptu a možné podoby budoucí aplikace. Demonstruje sled požadovaných procesů a informačních struktur. Díky tomu lze získat levnou zpětnou vazbu pro zlepšení aplikace. Obecně jsou charakterizovány nízkou technologickou implementací a ne zřídka k tvorbě bývá použit i obyčejný papír a tužka [16].

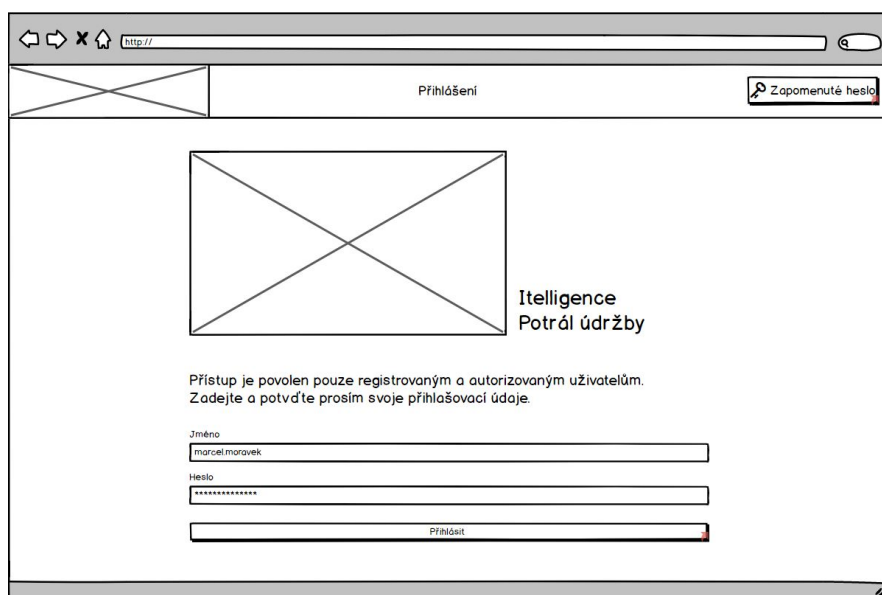
Pro návrh prototypu byly použity dva k tomu určené nástroje. Prvním z nich je univerzální nástroj pro tvorbu prototypových aplikací Balsamiq Mockups [17] druhým je nástroj Built [18] určený k prototypování přímo ve frameworku SAPUI5. Návrh rozhraní probíhá pro dva typy zařízení. Jednak pro desktop aplikace běžně disponujícím rozlišením 1200 pixelů a více a poté i pro přenosná zařízení jako jsou chytré telefony nebo tablety. V obou případech se jedná o **WYSIWYG editory** (What You See Is What You Get), které umožňují tvořit cílovou podobu stránky bez nutnosti cokoliv programovat. Většinou pracují na principu táhni a pusť. Tudíž uživatel pracující s tímto editor si vybírá ze základny použitelných UI komponent, které potom skládá do sebe a tvoří tím výslednou stránku.

#### 3.3.1 Balsamiq

Jedná se o prototypovací nástroj disponující širokou škálou univerzálních kreslicích nástrojů a připravených elementů, které se v moderních aplikacích vyskytují. Takovým elementem může být například obecná struktura webové stránky nebo vzhled mobilního telefonu. Tím lze vytvořit kostru navrhovaného prototypu, do kterého lze poté sázet konkrétní UI prvky jako tlačítka, texty, obrázky, tabulky a podobně. Na každý element lze navázat událost, pomocí které lze mezi jednotlivými stránkami navigovat. Výstupem je soubor ve formátu PDF, ve kterém se lze mezi jednotlivými obrazovky pohybovat jako by byly reálnou webovou stránkou nebo aplikací [17].

**Desktopová verze** První navržená stránka se týká přihlášení uživatele. Návrh přihlašovací stránky je vidět na obrázku 3.2. Byl pro ni použit layout představující webový prohlížeč. Do něj byl zasazen formulář pro vyplnění uživatelského jména a hesla. Z důvodu testování nástroje byly do stránky doplněny i doprovodné texty a náznak úvodního obrázku.

### 3. NÁVRH UŽIVATELSKÉHO ROZHRAŇÍ



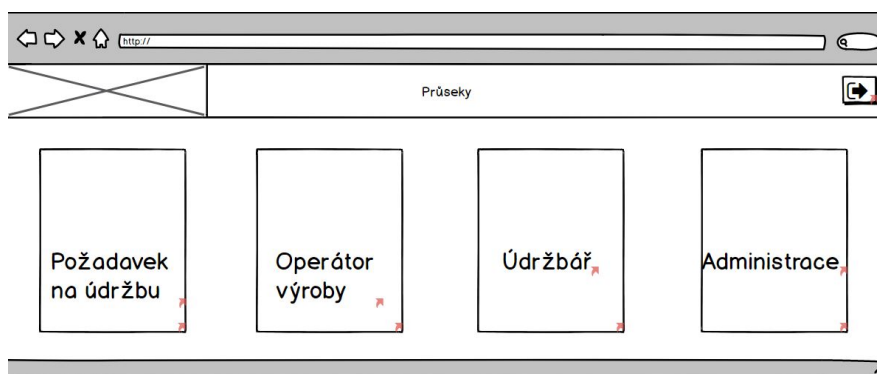
Obrázek 3.2: Návrh přihlašovací obrazovky

Pro návrh domovské stránky je čerpáno z konceptu SAP Fiori Launchpad, který slouží jako rozcestník do aplikací ve standardním licencovaném řešení SAP. Ten umožňuje uživatelům jednotlivé aplikace seskupovat a upravovat si designové téma. Ukázkou Launchpadu je obrázek 3.3.



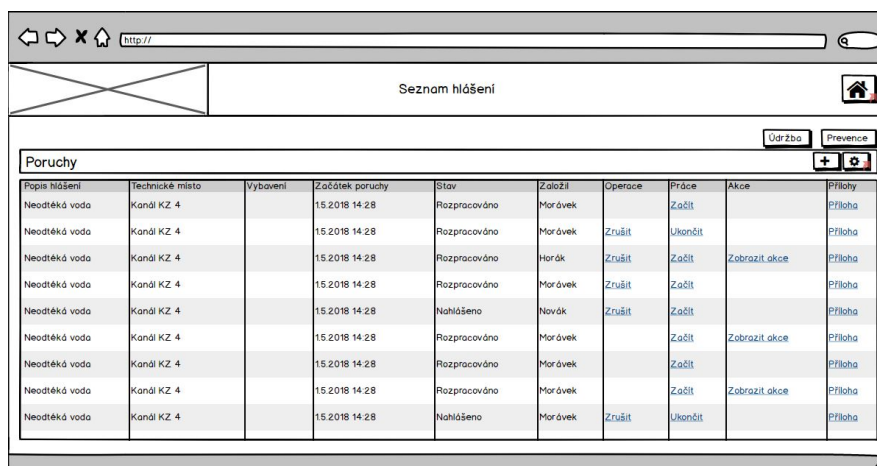
Obrázek 3.3: Ukázka designu SAP Fiori Launchpad [4]

Nic takové ovšem není ve vyvíjené aplikaci plánováno. Nicméně pro důvod zachování konceptu a možného budoucího zprovoznění SAP Fiori Launchpad je přistoupeno ke stejnému dlaždicovému návrhu. Návrh je k vidění na obrázku 3.4. Jednotlivé dlaždice umožňují navigaci do cílených aplikací navržených závěrem analýzy v kapitole 2.5.



Obrázek 3.4: Návrh domovské obrazovky

Obrázek 3.5 představuje navrženou stránku pro operátora výroby. Skládá se z již použitého layoutu pro webové stránky a především pak tabulky zobrazující seznam poruch. V tabulce jsou zobrazované požadované sloupce a v její pravé horní části jsou tlačítka pro přepínání druhů hlášení a spolu s tlačítkem pro založení nové poruchy.



Obrázek 3.5: Návrh seznamu poruch pro operátora výroby

Po stisknutí tlačítka pro založení nové poruchy dojde k otevření dialogu, tak jak je znázorněno na obrázku 3.6. Zobrazený dialog se zobrazí do popředí aplikace s tím, že zbytek (seznam hlášení a zbytek komponent) bude neaktivní. Formulář pro založení poruchy se skládá z několika polí, pro které jsou využity příhodné vstupní elementy. To znamená, že například pro stanovení priority je použit select box umožňující výběr z pevně dané množiny hodnot. Pro výběr souboru je naopak připraveno standardní tlačítko vyhledat, které by mělo otevřít průzkumníka pro vyhledání souboru v závislosti na uživateli používané platformě.

### 3. NÁVRH UŽIVATELSKÉHO ROZHRANÍ

The screenshot shows a web browser window with a URL bar. The main content area is divided into three sections. On the left, a table titled 'Poruchy' lists existing faults. In the center, a form titled 'Založení poruchy' allows users to report a new fault. On the right, a table titled 'Práce' shows a list of tasks and their status.

Popis hlášení	Technické m...
Neodtéká voda	Kandl KZ 4
Neodtéká voda	Kandl KZ 4
Neodtéká voda	Kandl KZ 4
Neodtéká voda	Kandl KZ 4
Neodtéká voda	Kandl KZ 4
Neodtéká voda	Kandl KZ 4
Neodtéká voda	Kandl KZ 4
Neodtéká voda	Kandl KZ 4
Neodtéká voda	Kandl KZ 4
Neodtéká voda	Kandl KZ 4

**Založení poruchy**

Vybavení:

Priorita:

Plánovací skupina:

Pracoviště:

Popis požadavku:

Příloha:

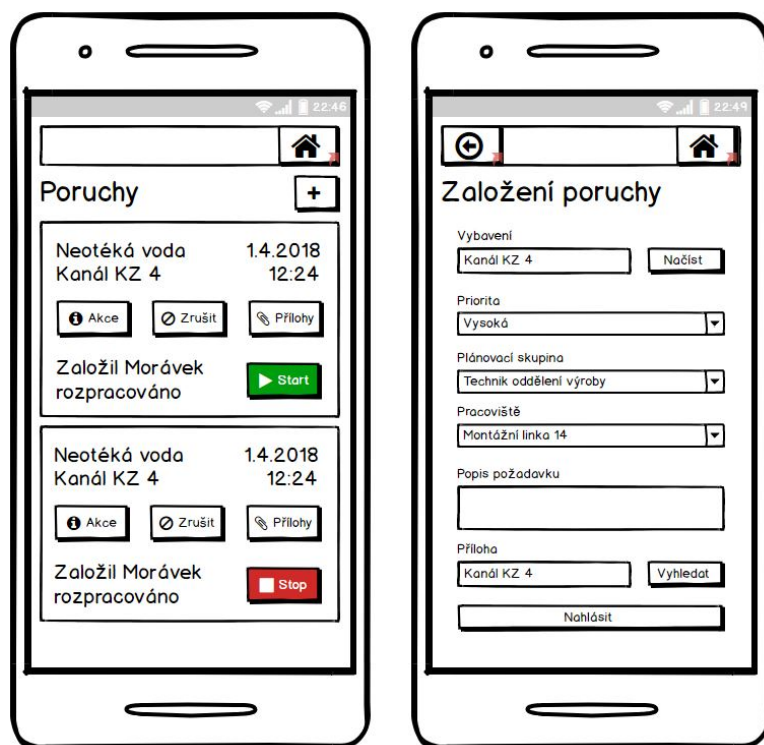
Práce	Akce	Přílohy
Začít		Příloha
Ukončit		Příloha
Začít	Zobrazit akci	Příloha
Začít		Příloha
Začít	Zobrazit akci	Příloha
Začít	Zobrazit akci	Příloha
Začít	Zobrazit akci	Příloha
Ukončit		Příloha

Obrázek 3.6: Návrh založení poruchy pro operátora výroby

**Mobilní verze** Mobilní verze obsahuje návrh stejných obrazovek jako desktopová verze popsaná výše. Návrhy jsou zobrazeny v obrázcích 3.7 a 3.8.

The image shows two mobile phone screens. The left screen displays a login form with the title 'Itelligence Potrál údržby'. It has fields for 'Jméno' (username) and 'Heslo' (password), both containing the text 'Kandl KZ 4'. Below these fields is a 'Přihlásit' button. The right screen displays the home page with the same title. It features four large buttons: 'Žádost o údržbu', 'Operátor výroby', 'Údržbář', and 'Správce'.

Obrázek 3.7: Návrh přihlašovací a domovské obrazovky



Obrázek 3.8: Návrh obrazovky pro seznam poruch a jejich založení

### 3.3.2 Built

Jedná se o oficiální produkt společnosti SAP, který umožňuje tvorbu prototypových aplikací ve frameworku SAPUI5. Tvorba v tomto nástroji si vyžaduje alespoň základní znalosti o jednotlivých komponentách frameworku a strukturování navrhovaných stránek. Pro výběr prvků se totiž využívá omezený výběr UI komponent poskytovaných přímo frameworkem SAPUI5. V základním výběru jsou především prvky z responzivní knihovny „sap.m“ [18].

**Desktopová verze** První navržená stránka je pro operátora výroby. Zobrazuje mu tabulku se seznam poruch. každá porucha má ve sloupcích vypsane požadované hodnoty, případně tlačítka pro provádění požadovaných akcí. Na obrázku 3.9 je k vidění například tlačítko pro započetí práce na poruše nebo pro zobrazené přiložené přílohy.

### 3. NÁVRH UŽIVATELSKÉHO ROZHRAŇÍ



Seznam hlášení

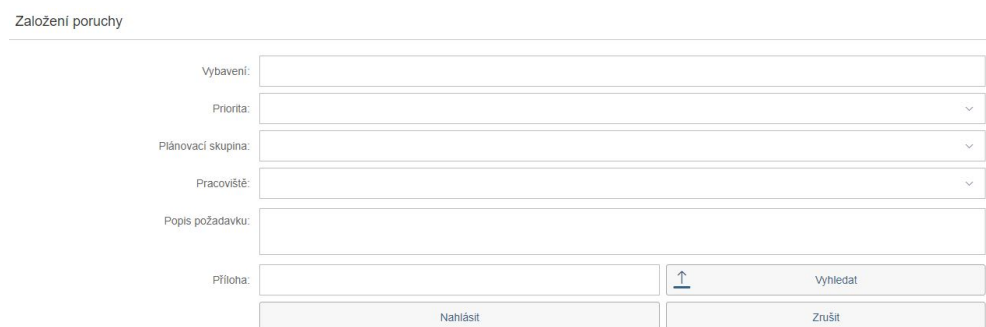
Údržba Prevence

Poruchy + Přidat

Popis poruchy	Technické místo	Vybavení	Začátek poruchy	Stav	Založil	Operace	Práce	Akce	Přílohy
Neodtéká voda	Kanál KZ 4	Text	1.5.2018 14:28	Nahlášeo	Morávek	Text			
Neodtéká voda	Kanál KZ 4	Text	1.5.2018 13:45	Rozpracováno	Novák	Text			

Obrázek 3.9: Návrh obrazovky pro seznam poruch

Prvním technickou překážkou tohoto nástroje je vytváření dialogů. To totiž není možné. A proto pro stlačení tlačítka na založení reakce musí dojít k zavolání zcela nové stránky. Stránka s jednotlivými elementy pro popsání poruchy a následného založení je k vidění na obrázku 3.10.



Založení poruchy

Vybavení:

Priorita:

Plánovací skupina:

Pracoviště:

Popis požadavku:

Příloha:

Vyhledat

Nahlásit Zrušit

Obrázek 3.10: Návrh obrazovky pro založení poruchy

**Mobilní verze** Druhou technickou překážkou je tvorba návrhu pro mobilní zařízení. Knihovna „sap.m“ je sice responzivní, ale ne vždy nabízí takové možnosti, které by od ní mohly být očekávány. Na obrázku 3.11 je k vidění podoba tabulky se seznamem poruch. Místo horizontálního skládání sloupců vedle sebe je správně použito skládání vertikální. Nicméně při zobrazení pěti hlášení by uživatel nedělal nic jiného, než zdlouhavě hledal požadovanou informaci v zdánlivě nekonečném seznamu.

Obrázek 3.11: Návrh obrazovky pro seznam poruch a jejich založení

### 3.3.3 Porovnání prototypovacích nástrojů

Na porovnání použitých nástrojů je k dispozici skromná tabulka hodnocení (jako ve škole) od autora práce.

Prostředí	Balsamiq	Built
Výběr komponent	1	3
Rychlost prototypování	1	3
Možnost využití při Hi-Fi	3	2
Dostupnost (cena)	2	3

Tabulka 3.1: Tabulka s hodnocením prototypovacích nástrojů

Pro tvorbu Lo-Fi prototypu se zdáti být Balsamiq účinnějším nástrojem. Je mnohem rychlejší, přehlednější a při troše úsilí a znalosti frameworku SAPUI5 lze pomocí něho dosáhnout velmi dobrých výsledků v napodobení Fiori

aplikace. Nástroj Built sice navíc umožňuje následné vyexportování aplikace, nicméně je technicky náročnější a výstup není pro budoucí vývoj nejvhodnější.

## 3.4 Heuristická analýza

Při návrhu uživatelského rozhraní je dobré držet se deseti následujících pravidel z Nielsenovi heuristické analýzy. Nielsenova heuristická analýza je jednou ze základních metod pro testování uživatelského rozhraní. Jedná se o seznam pravidel, které by uživatelské rozhraní mělo splňovat. Jakob Nielsen a Rolf Molich v roce 1990 vytvořili heuristiku pro heuristické vyhodnocení a poté v roce 1994 Jakob Nielsen revidoval tuto heuristiku na množinu pravidel [19]

1. **Viditelnost stavu systému:** Uživatel by měl být vždy vhodně informován o tom co se zrovna děje. Systém by měl vždy dát uživateli vědět, co se právě odehrává. Nesmí zůstat zamrzlý a nereagovat na uživatelské vstupy.
2. **Propojení systému a reálného světa:** Komunikace systému s uživatelem by se měla odehrávat uživatelsky příjemným způsobem (rozumitelný jazyk bez odborných termínů). Měl by zachovávat konvence reálného světa, ikony by měli znázorňovat akci, která se pod nimi ukrývá a podobně.
3. **Uživatelská kontrola a svoboda:** Uživatelé při práci se systémem dělají chyby a potřebují proto únikový východ pro návrat do předchozího stavu. Měla by tak být vždy k dispozici funkce zpět (návrat do předchozího stavu) nebo alespoň varování o nevratné akci. Uživatel tím pádem více experimentuje a rychleji se učí aplikaci ovládat.
4. **Standardizace a konzistence:** Uživatelé by neměli být nuceni přemýšlet, jestli různé termíny znamenají to stejné, proto se doporučuje dodržovat obecné zásady. Pokud je to možné, je vhodné používat platformové (frameworkové) komponenty pro zachování jednotného designu.
5. **Prevence chyb:** Vyvarovat se chybovým hlášením bezpečným designem, který bude preventivně působit proti problémům. Uživateli je dobré například sdělit špatně vyplněný formulář ještě před tím, než se ho pokusí potvrdit a odeslat.
6. **Rozpoznání namísto vzpomínání:** Uživatel by neměl být nucen vzpomínat si na provádění operací v systému, instrukce by měly být v systému vždy viditelně umístěny. Musí tak být vždy zobrazeny relevantní údaje pro uživatelskou práci.



7. **Flexibilní a efektivní použití:** Umožnění zrychlení práce se systémem pro pokročilé uživatele. Mělo by tak být umožněno pokročilým uživatelům zpřístupnit pokročilý režim, zobrazující například méně informací nebo s méně kroky v procesu.
8. **Estetický a minimalistický:** Uživatel by měl mít co nejméně možností kam může kliknout, protože každá další možnost soutěží o pozornost uživatele. Čím méně možností uživatel má, tím rychleji je schopen pokračovat. Na obrazovce by také měly být zobrazeny pouze informace, které uživatel v dané situaci opravdu potřebuje.
9. **Pomoc uživatelů pochopit, poznat a vzpamatovat se z chyb:** Chybové hlášky by měly být v přirozeném jazyce a neměly by například obsahovat žádné chybové kódy a podobně. Nejlepší je ovšem nedojít do stavu, kdy je chybového hlášení třeba.
10. **Nápověda a návody:** Všechny informace se musí dát lehce vyhledat, nápověda by měla obsahovat postupy. Spíše než popisy by se měla zabývat příklady.

**Jedná se o pomůcky poskytovanou testerům UI, nicméně se jedná o velmi užitečné zásady už pro samotný návrh.**



---

# Implementace

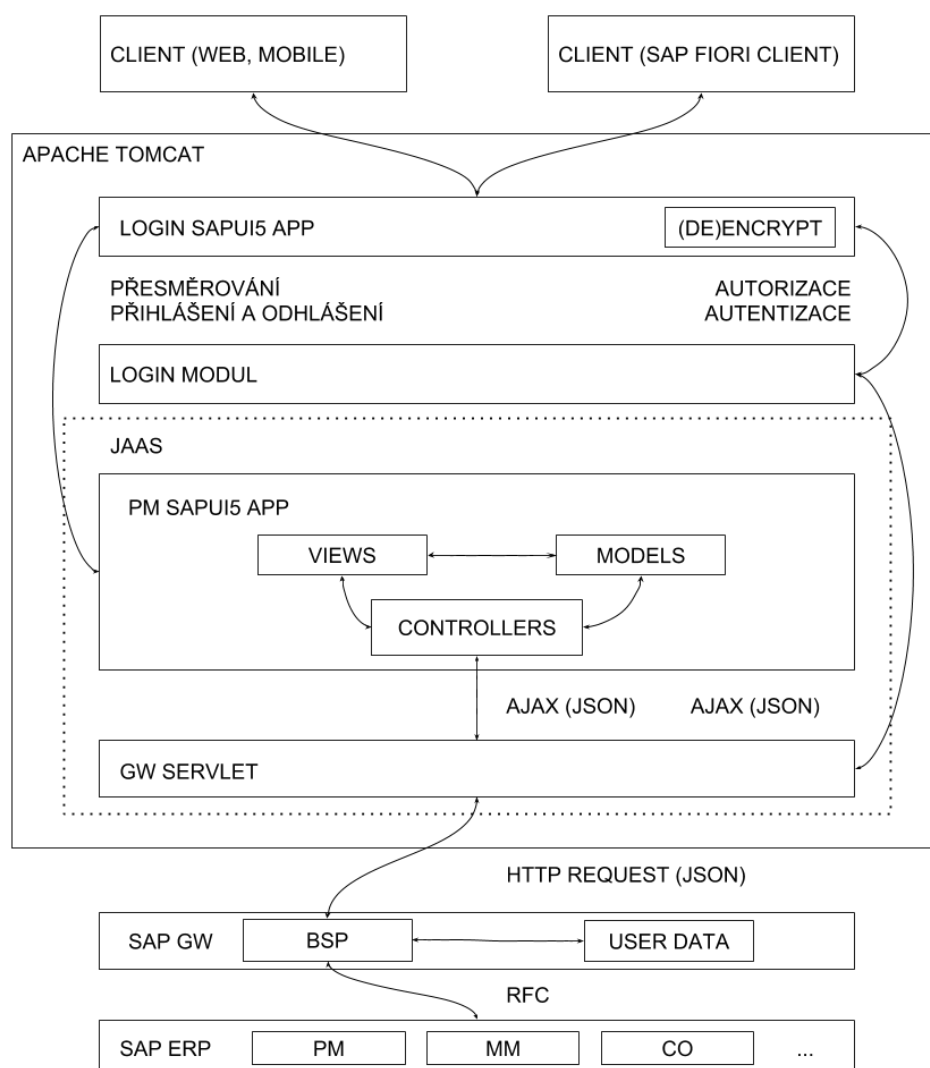
Tato kapitola se věnuje implementaci jednotlivých částí navržené architektury, která je ihned v první podkapitole zobrazena a popsána. Podrobnější popis jednotlivých komponent následují ihned poté.

## 4.1 Architektura

Architektura celého systému je zobrazena na obrázku 4.1, je rozdělena do tří bloků a tomu odpovídají i tři následující podkapitoly.

Standardním řešením je Fiori aplikace nahraná v SAP GW, která je přístupná přes SAP Fiori Launchpad. Pokud však zákazník nedisponuje dostatečnými licencemi a zakoupenými moduly, není tato cesta možná. Proto je zde navrženo řešení, které Fiori aplikaci spouští v prostředí Apache Tomcat a komunikaci s prostředím SAP řeší pomocí Java servletů a SAP BSP.

Toto řešení ovšem při požadavcích jako autentizace a autorizace uživatele přináší vývoj navíc. Ten však nemusí a zpravidla nepřesahuje náklady na moduly a licence SAP. Naopak přináší řadu výhod. V prostředí Apache Tomcat lze uchovávat a zpracovávat přijatá data a vytvořit si tak komunikační mezivrstvu mezi Fiori aplikací a SAP ERP. Tím lze ulehčit potenciálně zatíženým serverům a zkrátit uživateli webové aplikace odezvu v komunikaci.



Obrázek 4.1: Architektura navrženého systému

## 4.2 SAP ERP

Touto částí architektury se tato práce přímo nezabývá, nicméně je zde pro lepší představu o celkovém systému krátce popsána. Jak již bylo v kapitole 1.2 věnující se teoretickému základu řečeno, místem veškerých potřebných procesů pro realizaci požadované aplikace je SAP ERP. Konkrétně potom moduly PM, MM a CO. Každý z těchto modulů má desítky BAPI (Business Application Programming Interface) funkcí představujících rozhraní pro základní operace nad daným modulem. Správné sekvence a data volaných BAPI funkcí jsou

potom obaleny do funkčních modulů, které umožňují vzdálené volání z jiných systémů. K tomu slouží v rámci SAP funkcionalita RFC (Remote Function Call), které umožňuje přenášet data napříč jednotlivými systémy. Takovýchto funkcí vzniklo v ERP systému více než 20, výčet modulů je k vidění na obrázku 4.2.

Funkční moduly	
• ZITL_MOB_PM_CHANGE_NOTIF	Změna - Hlášení - změna katalogu, akcí, dlouhého textu
• ZITL_MOB_PM_CHANGE_ORDER	Změna - Zakázka - uživatelského statusu
• ZITL_MOB_PM_CLOSE_NOTIF_ORDER	Změna - Hlášení + Zakázka - Technické uzavření
• ZITL_MOB_PM_CREATE_CONF	Založení - Zpětné hlášení
• ZITL_MOB_PM_CREATE_DOCUM	Založení - Dokumentu k hlášení
• ZITL_MOB_PM_CREATE_MATER	Založení - Materialového dokladu pohyb 261-262
• ZITL_MOB_PM_CREATE_NOTIF	Založení - Hlášení nebo poruchy
• ZITL_MOB_PM_CREATE_ORDER	Založení - Zakázky a uvolnění zakázky a zápis startu do tab. ZPM_CONF
• ZITL_MOB_PM_DELETE_NOTIF	Výmaz - hlášení - změn statusu
• ZITL_MOB_PM_GET_ARBPL	Číselník pracovišť pro daný závod
• ZITL_MOB_PM_GET_CATAL	Zobrazení - Katalogy seznam
• ZITL_MOB_PM_GET_CONF_TIME	Zobrazení - Kontrola času pro zpětné hlášení
• ZITL_MOB_PM_GET_DOCUM	Zobrazení - Dokument k hlášení
• ZITL_MOB_PM_GET_DOCUM_SHOW	Zobrazení - Připojené DMS dokumentu k danému TM či VYB
• ZITL_MOB_PM_GET_NOTIF	Zobrazení - Hlášení - poruchy - seznam
• ZITL_MOB_PM_GET_NOTIF_ACT	Zobrazení - Hlášení - akce
• ZITL_MOB_PM_GET_NOTIF_TXT	Zobrazení - Hlášení - dlouhý text
• ZITL_MOB_PM_GET_PLAN	Číselník načtení plánovacích skupin pro daný závod
• ZITL_MOB_PM_GET_PRIOR	Číselník načtení priorit pro druh hlášení
• ZITL_MOB_PM_GET_TPLNR	Číselník - Seznam technických hlášení a vybavení nadřazené - podřízené
• ZITL_MOB_PM_RELEASE_ORDER	Změna - Hlášení + Zakázka - Anulace technického uzavření

Obrázek 4.2: Seznam funkčních modulů v ERP systému

## 4.3 SAP GW

Primárním účelem serveru SAP GW (Gateway) je komunikace s okolním světem. Je také standardním prostředím pro běh Fiori aplikace a pro servis pomocí protokolu OData popsanych v kapitole 1.5.3. Ne všichni zákazníci ovšem mají nakoupené licence pro provoz Fiori touto cestou, a právě proto byla navržena tato architektura založená na technologii BSP, která je běžně dostupná v produktech GW, ale i ERP. Na obrázku 4.3 je seznam 36 BSP stránek, které pro aplikaci vznikly. V porovnání s počtem funkčních modulů na straně ERP je evidentní, že nejsou vytvořeny v poměru 1:1. Je to především z toho důvodu, že jsou v této úrovni uloženy uživatelská data portálových uživatelů.

#### 4. IMPLEMENTACE

Strany s logikou běhu	
▶ zitl_mob_pm_change_notif.json	Změna - Hlášení - změna katalogu, akcí, dlouhého textu
▶ zitl_mob_pm_change_order.json	Změna zakázky
▶ zitl_mob_pm_change_user_pwd.json	Změna hesla portálového uživatele
▶ zitl_mob_pm_check_user.json	Kontrola portálového uživatele
▶ zitl_mob_pm_close_notif_order.json	Technické uzavření hlášení a zakázky
▶ zitl_mob_pm_create_conf.json	Založení zpětného hlášení
▶ zitl_mob_pm_create_docum.json	Založení - Dokumentu k hlášení
▶ zitl_mob_pm_create_mater.json	Založení - Materialového dokladu pohyb 261-262
▶ zitl_mob_pm_create_notif.json	Založení hlášení - poruchy
▶ zitl_mob_pm_create_order.json	Založení zakázky a uvolnění zakázky
▶ zitl_mob_pm_create_user.json	Založení portálového uživatele
▶ zitl_mob_pm_delete_notif.json	Výmaz - hlášení - změn statusu
▶ zitl_mob_pm_delete_user.json	Odstranění portálového uživatele
▶ zitl_mob_pm_edit_user.json	Změna portálového uživatele
▶ zitl_mob_pm_get_arbpl.json	Číselník pracovišť pro daný závod
▶ zitl_mob_pm_get_catal.json	Zobrazení - Katalogy seznam
▶ zitl_mob_pm_get_docum.json	Zobrazení - Dokumentu k hlášení
▶ zitl_mob_pm_get_doc_list.json	Zobrazení - Seznam DMS dokumentů
▶ zitl_mob_pm_get_notif.json	Načtení hlášení - poruchy
▶ zitl_mob_pm_get_notif_act.json	Zobrazení - Hlášení - akce
▶ zitl_mob_pm_get_notif_txt.json	Zobrazení - Hlášení - dlouhý text
▶ zitl_mob_pm_get_plan.json	Načtení číselníku priorit pro druh hlášení PU
▶ zitl_mob_pm_get_prior.json	Načtení číselníku plánovacích skupin pro daný závod
▶ zitl_mob_pm_get_ro.json	Role
▶ zitl_mob_pm_get_tplnr.json	Číselník - seznam technických hlášení
▶ zitl_mob_pm_get_tplnr_4qr.json	Načtení technického místa pro QR
▶ zitl_mob_pm_get_user.json	Uživatel
▶ zitl_mob_pm_get_user_list.json	Seznam uživatelů
▶ zitl_mob_pm_get_user_tplnr.json	Přednastavené pracoviště pro uživatele
▶ zitl_mob_pm_release_order.json	Změna - Zakázka - anulace technického uzavření
▶ zitl_mob_pm_remove_user_at.json	Odstranění atributu pro portálového uživatele
▶ zitl_mob_pm_remove_user_lo.json	Odstranění technického místa pro portálového uživatele
▶ zitl_mob_pm_remove_user_ro.json	Odstranění rolí pro portálového uživatele
▶ zitl_mob_pm_set_user_at.json	Založení atributu pro portálového uživatele
▶ zitl_mob_pm_set_user_lo.json	Založení technického místa pro portálového uživatele
▶ zitl_mob_pm_set_user_ro.json	Založení role pro portálového uživatele

Obrázek 4.3: Seznam funkčních modulů v SAP GW

Všechny výše zobrazené BSP stránky jsou implementovány na stejném principu. Využívají události OnRequest, ve které dojde ke zpracování vstupních dat a vygenerování adekvátních dat na výstup. V první fázi jsou vytvořeny lokální proměnné potřebné pro správné zpracování dat. Následně je deserializován vstupní objekt typu JSON do odpovídajících struktur programovacího jazyka ABAP. Poté je pomocí RFC zavolán příslušný funkční modul. Stejně tak jako v ukázce algoritmu 1.

**Algorithm 1** RFC volání funkčního modulu

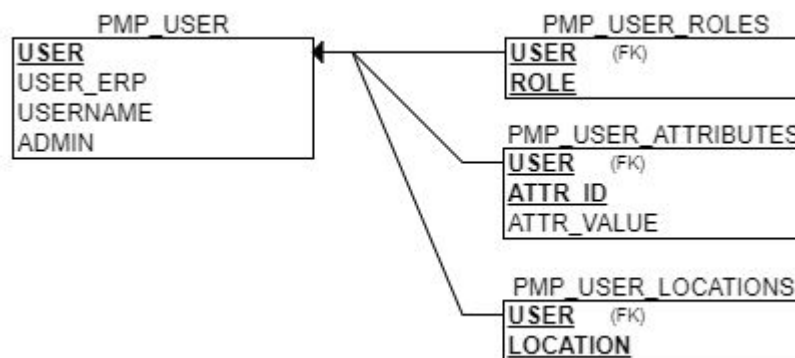
```

CALL FUNCTION 'ZITL_PM_CREATE_NOTIF' DESTINATION lv_dest
EXPORTING
    is_notif_get = zitl_input_json-qmart
    it_tplnr     = zitl_input_json-tplnr
IMPORTING
    et_notif     = lt_notif
    ev_error     = lv_error.

```

Kód zobrazuje volání funkčního modulu ZITL\_MOB\_PM\_GET\_NOTIF pro načtení seznamu hlášení. Parametrem DESTINATION je řízeno směrování volání. Hodnotou musí být nastavené spojení se vzdáleným serverem zabezpečeného pomocí metody BASIC. Parametry EXPORTING a IMPORTING určují vstupní a výstupní proměnné z funkčního modulu. Vstupními parametry je tak řízen například typ hlášení, který určuje, zdali se jedná o poruchu, prevenci nebo požadovanou údržbu. Obsahuje však i další parametry určující datumový rozsah a podobně.

Načtená data jsou zpětně serializována pro výstup. Obsahem celé BSP HTML stránky je tedy objekt typu JSON. Jak již ale bylo zmíněno, na úrovni GW se nacházejí i uživatelská data představující portálové uživatele. Na následujícím obrázku 4.4 je k vidění ilustrační databázové schéma.



Obrázek 4.4: Ilustrační databázové schéma portálových uživatelů

Databázové schéma neodpovídá úplně přesně skutečnosti. Některá pole a vazby tabulek jsou úmyslně skryta, a to buď protože nejsou pro celý koncept aplikace relevantní nebo z důvodu zvýšení bezpečnosti. Ze schématu vychází vazby 1 ku N pro všechny kombinace kmenové tabulky PMP\_USER a pomocných tabulek PMP\_USER\_ROLES, PMP\_USER\_ATTRIBUTES a PMP\_USER\_LOCATIONS. Trochu podrobněji jsou jednotlivé tabulky popsány v následujícím výčtu.

- **PMP\_USER:** Tabulka obsahuje záznamy jednotlivých portálových uživatelů. Jedná se tedy především o přihlašovací údaje a jednotlivé informace přímo související s uživatelem. Jako je například jméno, pod kterým uživatel ve webové aplikaci vystupuje, osobní číslo v ERP, informace o tom, zdali se jedná o správce nebo také platnosti uživatele a data s časy posledních změn spolu s jejich autory.
- **PMP\_USER\_ROLES:** Tabulka s cizím klíčem uživatel. Primární klíč je identifikátor role, která je uložena ve standartních SAP tabulkách, které jsou zpracovatelné pomocí standardních transakcí.
- **PMP\_USER\_ATTRIBUTES:** Tabulka s cizím klíčem uživatele. Primární klíč je identifikátor atributu. Výčet atributů není nikterak stanoven a zodpovědnost za správné vyplnění je přenesena na správce účtů, který má k dispozici seznam užitečných atributů pro webovou aplikaci. Součástí tabulky je i pole pro hodnotu takového atributu. S vytvořením aplikační logiky kontrolující správnost atributů z pevně stanoveného výčtu je počítáno v budoucím rozšíření aplikace.
- **PMP\_USER\_LOCATIONS:** Tabulka s cizím klíčem uživatele. Primární klíč je identifikátor zodpovědného pracoviště. Ten není prozatím nikterak kontrolován vůči hierarchickému uspořádání technických míst v modulu SAP PM. S vytvořením aplikační logiky kontrolující správnost pracovišť ze stanovené hierarchie je počítáno v budoucím rozšíření aplikace.

### 4.4 Apache Tomcat

Apache Tomcat je známý open-source webový server a servletový kontejner. Jedná se o oficiální referenční implementaci technologií Java Servlet a Java Server Pages (JSP). Na serveru mohou běžet uživatelské servlety (programy napsané v Javě), které umí zpracovávat požadavky zasílané pomocí HTTP protokolu a tímtéž protokolem na ně odpovídat. Apache Tomcat zde slouží jako zásobník servletů starající se o jejich spouštění, běh, ukončování a podobně [20].

#### 4.4.1 Login Modul

Login modul využívá systému **JAAS** (Java Authentication and Authorization Service), který slouží pro autentizaci a autorizaci uživatele. Jak již z názvu vyplývá, jedná se o bezpečnostní systém založený na technologii Java. Vyskytuje se od verze 1.4 v J2EE (Java 2 Enterprise Edition). Veškeré informace týkající se JAAS technologie vycházejí z dokumentace [21]

Deklarativní zabezpečení J2EE chrání webové aplikace podle aktuálního vzorce uživateli URL. Cesta k souborům může být popsána absolutním i



relativním způsobem. Jeli například požadováno povolení přístupu k aplikaci uživatelům s rolí USERS, je zapotřebí v definici web.xml 4.5.1 zavést následující definici.

---

**Algorithm 2** Definice zabezpečení aplikace pro roli USERS

---

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>AllPublic</web-resource-name>
    <url-pattern>/</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>USERS</role-name>
  </auth-constraint>
</security-constraint>
```

Tato definice chrání webovou aplikaci od kořenového adresáře J2EE, jak je naznačeno vzorem „\“. Pokud by mělo být zabezpečení aplikováno jen na část aplikace, je třeba uvést za „\“ jméno adekvátního podadresáře.

---

OVĚŘOVÁNÍ V DEKLARATIVNÍ OCHRANĚ je vynuceno, právě tehdy když si uživatel vyžádá chráněnou oblast webové aplikace. Pokud nebyl dříve ověřen, zobrazí se přihlašovací dialog, aby se uživatel mohl identifikovat. Běžně používané způsoby ověření jsou FORM a BASIC.

**BASIC** Je základním typem autentizace. Využívá standardního dialogu prohlížeče pro vložení uživatelského jména a hesla. Tento dialog nelze nikterak modifikovat, proto se jeho vzhled liší pouze na typu aktuálně používaného prohlížeče. Uživatelská pověření pro autentizovanou oblast jsou uložena v rámci session prohlížeče. Uživatelská data (jméno a heslo) jsou potom v zakódované formě posílána s každým HTTP requestem [22].

---

**Algorithm 3** Definice typu autentizace BASIC

---

```
<login-config>
  <auth-method>BASIC</auth-method>
</login-config>
```

---

**FORM** Je přizpůsobivější variantou autentizace. Umožňuje vývojáři specifikovat vlastní dialog pro přihlášení. Jediné omezení spočívá v pojmenování vstupní hodnoty uživatelského jména na j\_username a hesla na j\_password. Přihlašovací akce pro autentizaci potom musí mít hodnotu j\_security\_check v rámci J2EE kontejneru. Uživatel posléze zůstává ověřen přes session server [22].

---

**Algorithm 4** Definice typu autentizace FORM

---

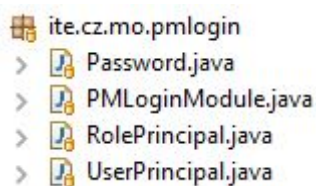
```
<login-config>
  <auth-method> FORM</auth-method>
  <form-login-config>
    <form-login-page>/login.html</form-login-page>
    <form-error-page>/login_error.html</form-error-page>
  </form-login-config>
</login-config>
```

Navíc od autentizace typu BASIC jsou zde nadefinovány dvě html stránky. Element `<form-login-page>` stanovuje stránku, která je uživateli zobrazena při prvotním pokusu uživatele o načtení aplikace, stránka stanovená elementem `<form-error-page>` se zobrazí v případě, že uživatel zadá špatnou kombinace jména a hesla nebo se v průběhu autentizace vyskytne jiná chyba.

---

Oba přístupy autentizace jsou zranitelná vůči útokům skrze odposlouchávání. Proto je silně doporučeno zpřístupňovat aplikaci skrze HTTPS protokol.

Kompletní proces autentizace a autorizace je implementován pomocí tříd napsaných v programovacím jazyce Java. Výčet tříd odpovídá struktuře 4.5 zobrazené níže.



Obrázek 4.5: Struktura balíku realizující autentizaci a autorizaci

Nejdůležitější třídou je **PMLoginModule**, která spočívá v přepisu čtyř procesních metod. Inicializační metodou je **initialize**, která nastavení instančním proměnným výchozí hodnoty. Prvním z nich je základní objekt **CallbackHandler**, sloužící k předávání hodnot mezi modulem a uživatelským prohlížečem. Druhým je objekt **Subject** udržující informace o uživateli. Takovou hodnotou je například identifikátor session. Metodou realizující autentizaci je metoda **Login**, která pomocí objektu **CallbackHandler** získá uživatelem vyplněné hodnoty jména a hesla, jejíž ověření se právě v rámci této metody musí provést. Pro kontrolu očekávaného slouží statická třída **Password**, která obsahuje metody pro generování hesel i ověření jejich shodnosti. Výstupem je potom boolean hodnota **true** nebo **false**. O autorizaci se stará metoda **Commit**, jejíž úkolem je přidělení příslušných rolí uživateli. K tomu slouží třídy **UserPrincipal** a **RolePrincipal** reprezentující jednotlivé role a uživatele. Poslední metodou k doplnění celého procesu je **Logout**, která má za úkol odstranit uživateli role a následně celý objekt **Subject**, který ho reprezentuje.

**Nasazení** Celý projekt je zapotřebí zabalit do archivu typu JAR, sloužící pro distribuci programů a knihoven. Následně ho vložit do run-time prostředí Tomcat serveru. Pro uložení takové knihovny je určen adresář `libs`.

K tomu, aby server vůbec věděl, že má pro aplikace vyžadující autentizaci uživatele použít vytvořenou JAR knihovnu, je zapotřebí definovat JAAS konfigurační soubor obsahující cestu k dané knihovně. Ukázka takové definice je v kódu 5.

---

#### Algorithm 5 Konfigurační soubor JAAS

---

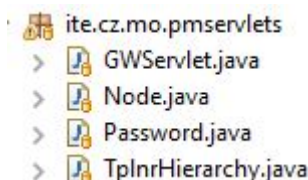
```
pmloginmodule {
    ite.cz.mo.pmlogin.PMLoginModule required debug=true;
};
```

---

Cestu souboru je pak zapotřebí definovat v Java parametrech serveru. K určení konfiguračního souboru slouží parametr `Djava.security.auth.login.config`. Hodnotou k tomuto parametru je absolutní cesta požadovaného souboru.

#### 4.4.2 GW Servlet

Na obrázku 4.6 je k vidění struktura paketu realizujícího mezivrstvu pro komunikace s SAP GW.



Obrázek 4.6: Struktura paketu realizujícího middleware vrstvu

Jedná se o čtyři třídy napsané v programovacím jazyce Java. Podrobněji jsou jednotlivé třídy popsány v seznamu níže.

- **GWServlet:** Je Java třída implementující interface `javax.servlet.Servlet` sloužící pro zpracování `Http` požadavků. Základem tohoto servletu je přepsaná metoda `doGet`, která obsahuje parametry `HttpServletRequest` a `HttpServletResponse` umožňují jednak načtení přijatých dat společně s dalšími parametry, které s sebou požadavek nese a potom také adekvátní data pomocí odpovědi vrátit. Metoda `doGet` je volána z další přepsané metody `doPost`, která je vyvolána v případě `HTTP` requestu typu `POST`.
- **Node:** Třída reprezentující prvek hierarchie technických míst a vybavení z modulu SAP PM. Jelikož se v interně technické místo a vybavení datově liší, je vytvořena tato třída, která nesrovnalosti mezi těmito objekty

eliminuje. Datově rozdílné identifikátory jsou nahrazeny jednotným id a ostatní lišící se parametry jsou sloučeny do atributů třídy odpovídajícím významu původního datového objektu. To posléze umožňuje v prezentační vrstvě lehce pracovat v

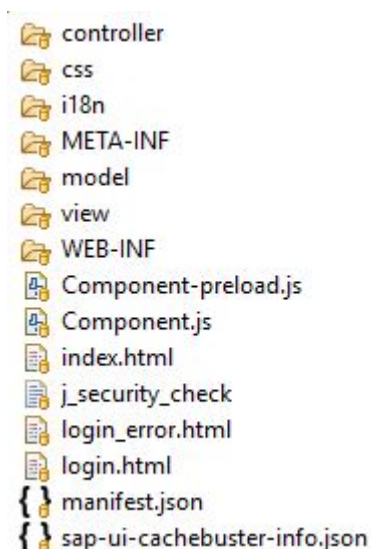
- **TplnrHierarchy**: Třída sloužící pro hierarchické skládání. Jejím hlavním atributem je množina (Set) objektů TplnrHierarchy. Tím, že lze uložit stejné prvky do sebe lze vytvořit požadovaná stromová struktura. Při pozdějším načítání struktury od požadovaného místa stačí v logaritmickém čase najít požadovaný uzel a vzít všechny jeho následníky.
- **Password**: Statická třída obsahující statické třídy pro generování hash z hesel a opačně k i jejich kontrolování.

### 4.5 PM SAPUI5 Aplikace

V následujících podkapitolách je popsána implementační struktura projektu pro PM SAPUI5 Aplikaci. Jelikož se jedná o stěžejní část celkové architektury, obsahuje veškerou logiku očekávanou od prezentační vrstvy. Jsou zde nadefinovány veškeré pohledy (stránky), se kterými se uživatel bude moci setkat.

#### 4.5.1 Struktura aplikace

Zde je zobrazena a popsána celková struktura aplikace z vývojářského pohledu. Na obrázku 4.7 je k vidění hlavní strom adresářů a souborů v použitém Eclipse projektu.



Obrázek 4.7: Struktura PM aplikace s hlavní prezentační logikou

Jednotlivé položky jsou potom popsány v následujícím seznamu. Konceptně je popis strukturován tak, že nejdříve jsou napsány obecné vlastnosti očekávané od daného adresáře nebo souboru a poté jsou dopsány případné poznámky z implementace.

- **WEB-INF:** Adresář obsahuje externí knihovny potřebné pro správnou funkčnost aplikace. Jelikož základní datovou reprezentací používanou v této aplikaci je JSON a programovací jazyk Java nativně práci s tímto formátem nepodporuje, je v tomto adresáři přiložena externí knihovna **java-json.jar**, která práci s tímto typem objektů umožňuje. Dále je zde uložen soubor **web.xml** popisující nasazení webové aplikace včetně jejich webových služeb. Slouží k deklaraci servletů a filtrů používaných danou webovou službou. V případě této aplikace se jedná například o GWServlet.
- **view:** Složka obsahuje zadanou view (UI rozvržení elementů) ve značkovacím jazyce XML. Hlavním úkolem těchto souborů je hierarchické rozvržení použitých komponent jednotlivých stránek. Pomocí různých layoutů tak umožňuje jasně definovat, že dané view reprezentuje dialog, jehož obsahem je formulář o pěti prvcích, z nichž každý používá jinou vstupní uživatelskou komponentu. Podrobněji se jednotlivým souborům věnuje podkapitola views.
- **controller:** Adresář obsahující controllery napsané v programovacím jazyce JavaScript. Slouží především pro obsluhu uživatelské interakce s aplikací. Klikne-li například uživatel na tlačítko přidat uživatele, právě zde se nachází část kódu, která požadovanou funkcionalitu provede. Dojde například k přípravě dat pro dialog obsahující potřebná pole pro založení nového uživatele. Jednotlivým souborům se podrobněji věnuje kapitola controllery níže.
- **css:** Obsahuje soubory pro úpravu kaskádových stylů. Jelikož aplikace využívá z drtivé většiny pouze předdefinované styly frameworku SAPUI5, je zde pouze jeden soubor style.css obsahující pár úprav oproti standardu.
- **i18n:** Pro internacionalizaci se používá numeronymum i18n a v tomto případě adresář obsahuje soubory s dvojicemi hodnot klíč - hodnota, které slouží pro zobrazení v textu požadovaného jazyku. K rozlišení jazyků se používá postfix v názvu souborů. Pro český jazyk je to například název i18n\_cs.properties, určený dle koncovky \_cs. Seznam koncovek odpovídá **standardu ISO 639-2**.
- **model:** Obsahuje pomocné JavaScriptové soubory s funkcemi, které se opakovaně používají napříč celou aplikací. Jedná se například o formátovací funkce, stanovení modelů obsahující informace o používaném za-

řízení uživatele a podobně. V mém případě obsahuje tři následující soubory.

- **models.js**: Slouží pouze k nadefinování modelu s informací o používaném zařízení. Poskytuje tak napříč zbytkem aplikace například aktuální šířku displeje, používaný operační systém nebo typ prohlížeče. Na základě toho poté dochází ve zbytku aplikace k používání komponent příslušných dané velikosti displeje, nedochází tak k zobrazování tabulky na mobilním zařízení, ale k adekvátně upravenému listu záznamů.
- **formatter.js**: Obsahuje funkce pro úpravu zobrazované informace získaných z backendu. SAPí interní formát data 2018-05-04, lze tak pomocí takových funkcí převést na datum v požadovaném tvaru a naopak.
- **utils.js**: Disponuje především funkcemi pro určení, zdali se má daná komponenta zobrazovat. Například tlačítko pro schválení údržby u operátora výroby musí být zobrazené pouze v případě, že na něm údržbáři dokončili práci. Na základě statusu hlášení poté funkce vrací boolean hodnoty true nebo false pro zobrazení daného tlačítka. Dále jsou zde funkce pro přeposílání dat na GWServlet využívající AJAX, který umožňuje asynchronní komunikaci pro výměnu dat s backendem.

---

**Algorithm 6** Ukázka AJAX volání

---

```
query : function(data , success , error) {  
    $.ajax({  
        type : 'POST',  
        url : url ,  
        cache : false ,  
        async : true ,  
        data : data ,  
        dataTye : "json" ,  
        success : success ,  
        error : error  
    });  
},
```

Tato funkce se používá u každého volání z aplikace na GWServlet pro následné zpracování na backendu. Na vstupu jsou tři parametry. Data obsahuje serializovaný JSON objekt obsahující potřebná data. Parametry success a error jsou ukazateli na funkce, které se mají vykonat v případě (ne)úspěšného volání AJAXu.

---

- **Component.js**: Jeden ze stavebních kamenů celé aplikace. Představuje objekt obalující všechna nadefinovaná view. Tudíž jakékoliv informace

uložené v modelu jsou dostupné napříč celou aplikací. Právě zde se uplatní modely nesoucí znalosti o použitém zařízení klienta nebo sloužící pro překlady textů.

- **index.html:** Jedná se o soubor, který je implicitně volán v případě, že uživatel ve svém webovém prohlížeči zadá adresu, pod kterou se požadovaná aplikace skrývá. Nacházejí se zde základní informace nutné pro správné spuštění aplikace. Nejdůležitější z nich je nadefinování zdrojů frameworku SAPUI5. To spočívá v odkazu na obsáhlý soubor `sap-ui-core.js` (dále již jen jádro), které představuje kostru nutnou pro běh aplikace. Použití slova `core` - jádro v názvu pochopitelně není náhoda. Jádro si v průběhu používání aplikace dotahuje další frameworkové knihovny jako jsou například komponenty pro uživatelské vstupní pole typu `date`, kdy je uživateli poskytnuto příjemné kalendářní rozhraní pro výběr požadovaného data. Tyto knihovny jsou stahovány až v případě, že si je uživatelská interakce vyžaduje. Dochází tedy k tak zvanému *lazy loadingu*. Stanovit přístup k jádru se dá pomocí dvou základních metod. První z nich je mapování na veřejně dostupný soubor pod adresou <https://sapui5.hana.ondemand.com/resources/sap-ui-core.js>. Druhým způsobem je mít veškeré knihovny dostupné v *run-time*ovém prostředí aplikace. Zároveň se také jedná o možnost použitou v této implementaci. Interní bezpečnostní politika totiž nedovoluje volání mimo vnitřní síť. Dále se zde nacházejí odkazy na externí knihovny nutné pro správný chod aplikace v rámci prostředí používaném uživatelem. V rámci hierarchie HTML je vytvořen element `body`, do kterého je vložena celá instance vytvořené aplikace odvozené od staženého jádra. Jedná se zpravidla o jediný kus čistého HTML při tvorbě aplikace ve frameworku SAPUI5.

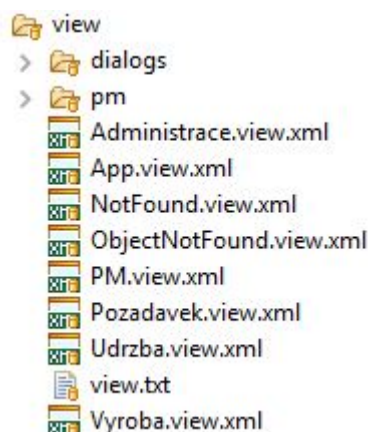
**Aplikační Cache** Implicitně jsou veškeré zdroje a knihovny používané ve frameworku ukládány do mezipaměti prohlížeče, aby byla uživateli zkrácena doba načítání a nedocházelo k opakovanému stahování potřebných souborů. To s sebou ovšem přináší problém při vydávání nové verze aplikace. V případě, že by došlo ke změně některého ze souborů a libovolný uživatel měl uloženy staré soubory v mezipaměti, pravděpodobně by se stala aplikace nefunkční. Tento problém je oficiálně řešen na straně SAP Gateway, kde dochází k porovnání jednotlivých souborů před samotným stažením. Tato funkcionalita bohužel není implicitně ve frameworku zanesena, nicméně existují mechanismy, které obdobné chování umožňují. Celý proces spočívá v nadefinování `ResourceServletu`, který porovnává datum poslední modifikace souboru s tím, který má uživatel uložen v mezipaměti. Tento Servlet musí být zdefinován v souboru `web.xml` a aplikace musí dostat informaci o tom, u kterých souborů má k takové kontrole docházet. Proto je v tomto HTML souboru odkaz na

soubor **sap-ui-cachebuster-info.json**, obsahující JSON pole, s dvěma prvky. Jedním z nich je relativní cesta k požadovanému souboru a druhým je datum poslední modifikaci. Pro správný chod aplikace je tak nutné při každém exportování dbát na aktualizaci potřebných záznamů.

- **login.html**: V rámci JAAS logiky pro autorizaci a autentifikaci je definována implicitní HTML stránka, která je vyvolána v případě, že uživatel nemá doposud vytvořenou vůči aplikaci session. Z důvodu zachování konzistentního vzhledu aplikací je i pro přihlášení vytvořena SAPUI5 aplikace. Umístit ji zde v rámci jednoho projektu by způsobilo značnou nepřehlednost a taktéž by mohlo způsobit neočekávané problémy při vývoji. Z toho důvodu má HTML stránka login.html velmi jednoduchou funkčnost. A tou je přesměrování uživatele do přihlašovací SAPUI5 aplikace.
- **manifest.json**: Manifest slouží především k rychlému definování možného směrování v rámci aplikace. Dochází zde k mapování uživateli aktuální url na jednotlivá view. Lze tak například přiřadit k postfixu pm/#/vyroba view „Vyroba“ a tím tak uživateli zobrazit očekávané informace. Dále se zde dají zadefinovat modely přiřazené komponentě nebo zdroje kaskádových stylů.

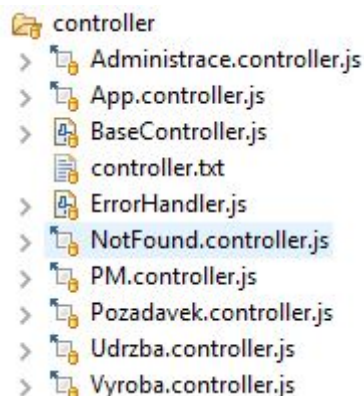
#### 4.5.2 Jednotlivé stránky aplikace

V této kapitole jsou popsány a zobrazeny nejdůležitější stránky aplikace. Koncept popisování jednotlivých stránek je velmi podobný. Nejdříve je obecně popsáno, co vlastně daná stránka dělá, poté následují dvě trochu více technické sekce View a Controller a následně popis stránky z uživatelského pohledu společně s ukázkou. Popsány jsou hlavní view a controllery z obrázků 4.8 a 4.9 níže.



Obrázek 4.8: Struktura jednotlivých view v Eclipse projektu





Obrázek 4.9: Struktura controllerů v Eclipse projektu

Jak je z obrázků patrné, view mají zpravidla stejnojmenný protějšek v controllerech. To je z toho důvodu, že každé view má přiřazen jeden controller a z důvodu přehlednosti jsou proto pojmenovány stejně.

#### 4.5.2.1 App

Není ani tak stránkou jako spíš základním view aplikace. V podstatě pouze všechny ostatní view obaluje a svůj obsah dynamicky střídá na základě uživatelových kroků. Je nastaveno jako výchozí pro všechny možné kombinace URL, které je uživatel v rámci aplikace schopen vytvořit.

**View** View je zadefinováno způsobem zobrazeném v následujícím kódu 7.

---

#### Algorithm 7 XML definice view App

---

```
<mvc:View controllerName="sap.ui.mo.pm.controller.App"
           xmlns="sap.m" xmlns:mvc="sap.ui.core.mvc" >
  <App id="app" />
</mvc:View>
```

---

Definice neříká nic jiného, než že v případě zobrazení tohoto view má být vnořena standardní SAPUI5 komponentu App, která slouží jako základ aplikace. V rámci této komponenty jsou následně agregována view popsána dále. Jak je v kódu vidět, je zde parametr „controllerName“, který přiřazuje k view controller.

**Controller** Stejně jako v ostatních případech je z důvodu přehlednosti totožně pojmenován. Jedinou jeho funkcí je v první fázi spouštění aplikace nastavit dále neměnné atributy. Jak je vidět v ukázce kódu 8 níže, dochází

zde k nastavení výchozího jazyka aplikace a kaskádových stylů v závislosti na typu použitého zařízení. Jiné styly jsou tak použity pro zařízení s rozlišením odpovídající tabletům, mobilům, nebo desktopům. V potaz se bere i dotykový displej, vyžadující si například větší tlačítka, než by tomu bylo v případě použití obyčejného monitoru.

---

**Algorithm 8** XML definice view App

---

```
BaseController.extend("sap.ui.pm.controller.App",{
  onInit : function() {
    sap.ui.getCore().getConfiguration().setLanguage("cs");
    var component = this.getOwnerComponent();
    var css = component.getContentDensityClass();
    this.getView().addStyleClass(css);
  }
});
```

---

Za zmínku ovšem stojí i první řádek zobrazeného kódu 8 výše. Výraz „`BaseController.extend("sap.ui.mo.pm.controller.App"`“ říká, že nedochází k rozšíření standardního controlleru frameworku, ale v tomto případě k aplikaci přiloženému controlleru pojmenovaného „`BaseController`“.

**BaseController** Od tohoto controlleru odvozují všechny ostatní implementované. To, protože se nemalá část funkcí dá využít na více místech v aplikaci a nemusí tak docházet k duplikování kódu, které by přinášelo riziko snížení konzistence a zvýšení pracnosti v případě budoucích změn nebo opravování chyb. Tento controller je již odvozen od standardního frameworkového controlleru „`sap.ui.core.mvc.Controller`“. Následující ukázka kódu 9 zobrazuje tři takové společné funkce. Celý výčet je pochopitelně mnohem delší, ale tyto byly vybrány, protože jsou nejčastěji používané a poslouží tak k dobré demonstraci snížené pracnosti a náročnosti na údržbu aplikace z pohledu vývojáře. Funkce zobrazené v ukázce kódu 9 jsou posléze krátce popsány.

---

**Algorithm 9** XML definice view App

---

```

Controller.extend("sap.ui.pm.controller.BaseController",{

    createDialog : function(that, id, dialog, fragment) {
        if (!dialog) {
            var frag = sap.ui.xmlfragment(id, fragment, that);
            that.getView().addDependent(frag);
            jQuery.sap.syncStyleClass("sapUiSizeCompact",
                                    that.getView(), res);

            return frag;
        }
        return dialog;
    },

    setModel : function(oModel, sName) {
        return this.getView().setModel(oModel, sName);
    },

    getI18NText : function(that, id) {
        var oModel = that.getView().getModel("i18n");
        var oBundle = oModel.getResourceBundle();
        return oBundle.getText(id);
    },
});

```

---

- **createDialog**: Aby mohly být v rámci controlleru vytvářeny jednotlivé dialogy (až desítky na jeden controller) téměř bezpracně, byla vytvořena následující prototypová funkce vytvářející v požadovaném controlleru objekt reprezentující dialog. K vytvoření takového dialogu poté stačí jeden příkaz jako v ukázce 10 níže.

---

**Algorithm 10** Ukázka kódu pro vytvoření objektu dialogu

---

```

that.oEqnrSTD = that.createDialog(that, that.oEqnrSTD,
    "../view.dialogs.EqnrSelectTreeDialog");

```

---

- **setModel**: Slouží k provázání modelu (objekt obsahující data) s požadovaným view. Každá UI komponenta frameworku umožňuje svázání s modelem a jeho konkrétním prvkem. V případě přeražení takového modelu k view pak může dojít k zobrazení požadované hodnoty. Jelikož svazování dat v modelu s view patří k velmi častým operacím a vyža-

duje volání více funkcí, je obaleno do této metody, které stačí předat jako parametr požadovaný model a jeho jméno.

- **getI18NText:** Velmi často v controlleru dojde k situaci, že je potřeba uživateli sdělit nějakou informaci. Může se jednat například o dialogové okno nebo jenom probliknutí textu informujícího o provedené nějaké akce. Aby v controlleru nebyly ošetřovány situace aktuálního jazyka a podobně, je vytvořena tato funkce, která načte hodnotu prvku v aktuálním používaném jazyce z příslušného internacionalizačního souboru.

#### 4.5.2.2 PM - Úvodní stránka

Reprezentuje úvodní obrazovku, jejíž obsah je přímo závislý na rolích, které má uživatel k dispozici. Zde bude uživateli kromě otevření aplikací umožněno změnit si heslo.

**View** Pro demonstraci struktury jednotlivých view je v této, jakožto první, ukázce obsahující větší počet komponent zobrazena téměř celá XML struktura rozdělena do tří bloků. V implementaci

---

**Algorithm 11** XML definice hlavičky úvodní stránky

---

```
<Page>
  <customHeader>
    <Bar design="Header">
      <contentLeft />
      <contentMiddle>
        <Title text="{i18n>pmPageTitle}" />
      </contentMiddle>
      <contentRight>
        <Button icon="//key" press="onEditPassword" />
        <Button icon="//log" press="onLogout" />
      </contentRight>
    </Bar>
  </customHeader>
```

Ve všech aplikacích se jedná o lehkou modifikaci této struktury. Zpravidla dochází ke změnám pouze u svazování nadpisu a použitých tlačítkách v horní liště. Jak lze vykoukat, hlavičková lišta je rozdělena do tří částí, které lehce umožňují zarovnání použitých komponent.

---

**Algorithm 12** XML definice obsahu úvodní stránky

```

<VBox width="100%" justifyContent="Center" >
  <l:Grid id="gridContainer" defaultSpan="L3_M6_S6" />
</VBox>

```

Jak je v ukázce vidět, jedná se o velmi jednoduchý obsah stránky. V podstatě je pouze zadefinováno rozložení Grid kontejneru, které má implicitně responzivní chování. Vývojář tedy nemusí nijak extra řešit počet zobrazených agregovaných komponent, framework to zvládá sám.

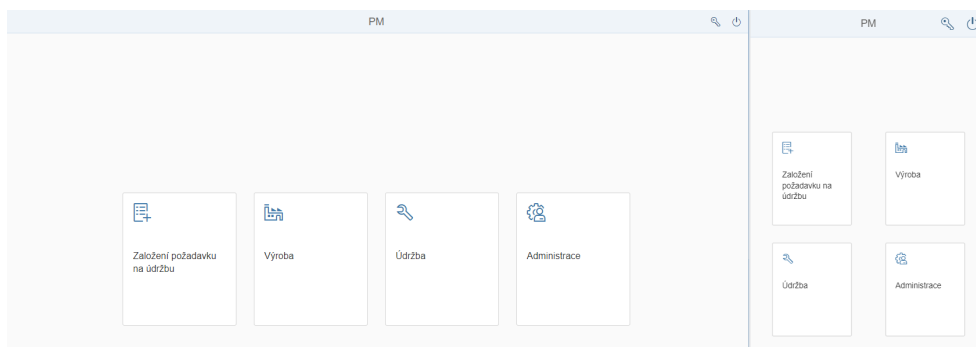
**Algorithm 13** XML definice zápatí úvodní stránky

```
</Page>
```

V tomto případě se jedná pouze o ukončení stránky. U ostatních aplikací mohou být zobrazovány například různá zápatí obsahující tlačítka s funkcemi a podobně.

**Controller** Má v rámci této stránky primární úkol k přidělené požadovaných dlaždic uživatel v závislosti na přidělených rolích. To spočívá v načtení uživatelských rolí a přidání dlaždic do Grid kontejneru zadefinovaného v ukázce kódu 12 výše.

**Stránka** Výsledná podoba stránky je zobrazena na obrázku 4.10 níže. Jedná se o velmi minimalistické provedení, jelikož se neočekává, že by zde uživatel trávil větší množství času. Obrázek je rozdělen na dvě části. V první je podoba stránky zobrazené v běžném desktop rozlišení (více jak 1200 pixelů na šířku). V druhé části je zobrazena podoba v mobilním zařízení odpovídajícímu dnešnímu standardnímu chytrému telefonu s obrazovkou velkou přibližně 5 palců (šířka alespoň 350 pixelů na šířku).



Obrázek 4.10: Úvodní stránka PM SAPUI5 aplikace

Podoba stránky odpovídá případu, kdy má uživatel přiřazené všechny dosavadní role.

#### 4.5.2.3 Pozadavek - Založení požadavku na údržbu

Stránka je navržena tak, aby odpovídala funkčnímu požadavku 2.3.1.1. Je zde proto vytvořen formulář umožňující zadat veškeré potřebné údaje ke specifikování požadavku.

**View** Jelikož se jedná v podstatě pouze o formulář určený k vyplnění od uživatele, je obsahem view především komponenta SimpleForm, pomocí které lze snadno vytvořit cílený formulář.

---

**Algorithm 14** XML definice obsahu úvodní stránky

---

```
<f:SimpleForm>
  <Label text="{i18n>pozadavekTplnrLabel}" />
  <Input value="{hlaseni>/tplnr}" showValueHelp="true"
        valueHelpRequest="onTplnrMatchCodeRequest" />
  <ndc:BarcodeScannerButton scan="handleScan" />
  <Label text="{i18n>pozadavekVybaveniLabel}" />
  <Text text="{hlaseni>/eqktx}" />
</f:SimpleForm>
```

Komponenta SimpleForm vytváří formulář na základě komponenty Label, která od sebe jednotlivé části separuje. Všechny elementy od Labelu až k dalšímu tvoří jeden celek a jsou výsledně rendrovány v jedné skupině.

---

**Controller** Úkolem controlleru na této stránce je především předvyplnění uživatelských atributů do vstupního formuláře a jeho následné odeslání na backend. To spočívá v načtení uživatelských dat v případě navštívení stránky s následným naplnění modelu příslušnými daty. Obsahuje také funkce pomáhající uživateli vybrat data. Takovým příkladem může být načtení hierarchie technických míst. K tomu je zapotřebí poslat požadavek pro data. K tomu poslouží funkce query implementující AJAX request 6 v JavaScriptové knihovně utils. Předáním lokálních funkcí pro úspěšné a neúspěšné volání poté může dojít zpracování přijatých dat nebo chyb. Pomocí funkce setModel popsané v BaseControlleru 9 a nebo přístupem ke konkrétnímu prvku modelu (funkce setProperty) lze potom nastavit požadovaná data do svázaného formuláře.

**Stránka** Výsledná stránka je reprezentována formulářem o sedmi prvcích. Podle typu zadávaných hodnot v poli je pak přizpůsobena komponenta ulehčující uživateli vyplnění. Pro technické místo je možné si nechat zobrazit dialog s hierarchií technických míst a vybavení si z něho vybrat. U elementů s pevně daným výběrem a zároveň striktně omezeným počtem je pak vybrána komponenta SelectList a podobně. Podoba stránky pro desktop a mobilní zařízení je vidět na obrázku 4.11 níže.

Obrázek 4.11: Stránka pro založení požadavku na údržbu

#### 4.5.2.4 Vyroba - Operátor výroby

Stránka je navržena tak, aby odpovídala funkčním požadavkům spadající pod roli Operátora výroby. Musí zde tedy být dostupný seznam jednotlivých hlášení (poruchy, prevence a údržby) pro jeho přidělené pracoviště. V rámci jednotlivých hlášení je zapotřebí mít k dispozici relevantní operace, které s nimi operátor může provést.

**View** Z důvodu responzibility se jedná o první stránku, kde je zapotřebí již v rámci view řešit rozlišení používaného zařízení z důvodu obsáhlé tabulky, která se na mobilních zařízeních nebude vhodně zobrazovat. V horní části obrazovky jsou navrženy dva filtry. První z nich je na typ zobrazovaného hlášení a druhým je filtr na technická místa. Filtr přes druh hlášení nejen, že filtruje data, ale i mění zobrazované informace. Požadovaná zobrazovaná data k jednotlivým typům hlášením nejsou totiž identická. Tato část je pro desktopová i mobilní zařízení stejná. Dále se však struktura stránky liší.

**View (desktop)** Pod filtry je navržena tabulka se všemi možnými sloupci, které se mohou v rámci stránky zobrazit. V rámci jednotlivých sloupců jsou pak přiděleny agregace na UI komponenty vyhovujícím požadavkům. Jedná se především o texty a tlačítka, která jsou zobrazována v závislosti na stavu (statusu) hlášení. O logiku se stará knihovna utils 4.5.1 implementující funkce rozhodující o informaci zobrazit nebo nezobrazit.

**View (mobile)** Pod filtry je navržen komplikovanější rozložení skládající se z desítek komponent různých layoutů uspořádaných do hierarchické struktury. Bylo zapotřebí změnit celou strukturu oproti tabulkovému zobrazení. Každé hlášení tak horizontálně zabírá více místa. Namísto maximálně dvou řádků v rámci jednoho hlášení se tak nyní vyskytuje až sedm řádků informací.

**Controller** Kromě standardních činností, jako je odchyťávání uživateli interakci s patřičným zpracováním, je zde zapotřebí dynamicky řešit obsah hlavního view. Toho je dosaženo za pomoci modelu zařízení a **návrhového vzoru pozorovatel**, který řeší informování požadovaných objektů o změně stavu jiného objektu. Pozorovaným objektem je v tomto případě model zařízení (konkrétně jeho část řešící aktuální rozlišení). Pozorovatelem je objekt (funkce) controlleru. Jelikož je tabulka společná pro všechny druhy hlášení, řeší controller zobrazení jednotlivých sloupců.

---

**Algorithm 15** Přiřazení posluchače ve formě funkce k hodnotě modelu

---

```
var dev = this.getOwnerComponent().getModel("device");
dev.getProperty("/resize").attachHandler(this.onResize);
```

Jedná se o výtazek kódu v inicializační funkci onInit. Spočívá v načtení modelu zařízení a přiřazení posluchače v podobě funkce onResize, která se provede vždy při změně rozlišení. To znamená zahrnuje i případ otočení displeje na mobilních zařízeních.

---

---

**Algorithm 16** Implementace funkce onResize

---

```
onResize : function(window) {
    var layout = that.getView().byId("notifLayout");
    layout.removeAllContent();
    if (window.width > 1024) {
        layout.addContent(that.notifTable);
    } else {
        layout.addContent(that.notifList);
    }
}
```

Jako hraniční hodnota pro zobrazení tabulky nebo listu byla vybrána hodnota 1200 pixelů. V případě změny dojde k odebrání fragmentu z layoutu a přiřazení adekvátního obsahu.

---

**Stránka** Cílem návrhu této stránky bylo maximální možné eliminování tlačítek a textů, které uživatel nepotřebuje znát. Tudíž všechny texty i tlačítka se zobrazují v případě, že mají nějaký význam. U textů to jsou pro uživatele potřebné informace k vykonávání jeho práce. V případě tlačítek umožňujících akce nad daným hlášením je tento problém řešen zobrazením jenom těch tlačítek, které lze nad daným hlášením v danou chvíli provést. Zrušit hlášení tak lze pouze do doby, než s ním někdo začne pracovat. Zobrazovat texty k hlášení jdou pouze tehdy, když už nějaký text k němu existuje a podobně. Podoba stránky pro desktop a mobilní zařízení je vidět na obrázku 4.12 níže.



Popis hlášení	Odp. prac.	Technické místo	Vybavení	Začátek poruchy	Status zakázky	Založí	Operace	Akce	Přiložky
17.34 Porucha / Moravěk	PM_DUMMY	MOO-EXP Expedice	RAPIDA 142 H	2018-02-04 17:34:30	UDR	DVOTOMOO			
porucha 16.2	TUH_ELE	MOO-HKS-03-STR Strojní zařízení	RAPIDA 142 H	2018-02-16 10:56:05	UDR	KURMAITE			
Chyba na HKS	TUV_MECH	MOO-HKS Výroba HKS	RAPIDA 142 - DW1	2018-02-27 09:15:32	UDR	MORMAITE			
porucha kurka mechanika	TUH_MECH	MOO-HKS-03-STR Strojní zařízení	RAPIDA 106 SRH	2018-03-19 14:06:25	UDR	KURMAITE			
test		MOO-HKS-04-STR Strojní zařízení	RAPIDA 142	2018-04-03 13:34:09					
test pro Tomase	TUH_MECH	MOO-HKS-03-STR Strojní zařízení	RAPIDA 142	2018-04-23 09:37:48					
Hýk	TUE	MOO-HKS Výroba HKS		2018-03-12 09:56:19	UDR	MORMAITE			
	TUE	MOO-HKS Výroba HKS		2018-03-13 10:55:15	UDR	MORMAITE			
	TUE	MOO-HKS Výroba HKS		2018-03-13 10:56:32	UDR	MORMAITE			

Obrázek 4.12: Stránka pro operátora údržby

#### 4.5.2.5 Udržba - Údržbář

Stránka je navržena tak, aby odpovídala funkčním požadavkům spadající pod roli Údržbáře. Musí zde tedy být dostupný seznam jednotlivých hlášení (poruchy, prevence a údržby) připravených k provedení servisního úkonu. V rámci jednotlivých hlášení je zapotřebí mít k dispozici relevantní operace, které s nimi údržbář může provést.

**View** Jelikož je seznam jednotlivých druhů hlášení řešen pomocí tabulky, stejně jako v případě operátora výroby 4.5.2.4, je opět přistoupeno k oddělení jednotlivých view na fragmenty, které se v závislosti na šířce e používaného zařízení přidávají a odebírají. Oproti stránce pro operátory výroby je zde ale požadovaných funkcionalit trochu více, a proto je rozdělení typů hlášení řešeno jinak. Místo přepínacího filtru mezi poruchami, prevencemi a údržbami vzniklo pět záložek. Tři z nich kopírují druhy hlášení, zbylé dvě slouží pro dohledání dokumentace k vybavení a zobrazení historie poruch.

**View (desktop)** Všech pět záložek je řešeno pomocí tabulek s různými sloupci. Požadavek na minimální počet zobrazených elementů zůstává. A i zde jsou proto texty a tlačítka zobrazována na základě knihovny utils 4.5.1.

**View (mobile)** Všech pět záložek je realizováno pomocí různých layoutů uspořádaných do hierarchické struktury komponent.

**Controller** Kromě standardních činností, jako je odchyťávání uživateli interakci s patřičným zpracováním, je zde zapotřebí dynamicky řešit obsah hlavního view. Toho je dosaženo za pomoci modelu zařízení a **návrhového vzoru pozorovatel**, který řeší informování požadovaných objektů o změně

## 4. IMPLEMENTACE

stavu jiného objektu. Pozorovaným objektem je v tomto případě model zařízení (konkrétně jeho část řešící aktuální rozlišení). Pozorovatelem je objekt (funkce) controlleru. Jelikož je tabulka společná pro všechny druhy hlášení, řeší controller zobrazení jednotlivých sloupců.

**Stránka** Stránka je řešena trochu odlišně než tomu je u operátory výroby 4.5.2.4. Místo filtru nad druhem hlášení jsou navrženy jednotlivé záložky s tím, že každá má přidělenou svoji adekvátní tabulku. Podoba stránky pro desktop a mobilní zařízení je vidět na obrázku 4.12 níže.

The screenshot shows a web application interface for maintenance (Údržba). It features a top navigation bar with tabs: Poruchy, Požadavky, Prevence, Dokumentace, and Historie poruch. Below the navigation bar, there's a section for 'Technické místo' with a dropdown menu. The main content area is titled 'Seznam otevřených poruch' and displays a table of open faults. The table has columns: Technické místo, Vybavení, Popis poruchy, Číslo zakázky, Odp. prac., Začátek poruchy, Založí, Start Stop, Operace, Akce, and Přílohy. The table lists several faults, including 'porucha 16.2' and 'Chyba na HKS'. To the right of the table, there's a detailed view of a specific fault, showing its description, technical location, equipment, and a list of actions (Start, Stop, Operace, Akce a priority).

Technické místo	Vybavení	Popis poruchy	Číslo zakázky	Odp. prac.	Začátek poruchy	Založí	Start Stop	Operace	Akce	Přílohy
MOO-HKS-03-STR	RAPIDA 142 H	porucha 16.2	0000700000	TUH_ELE	2018-02-16 10:56:05	KURMAITE	Start Stop	Operace	Akce a priority	
MOO-HKS	Výroba HKS	Chyba na HKS	0000700000	TUV_MECH	2018-02-27 09:15:32	MORMAITE	Start Stop	Operace	Akce a priority	
MOO-HKS	Výroba HKS	Rozbitý sklo	0000700000	TUV_MECH	2018-03-04 15:59:15	MORMAITE	Start Stop	Operace	Akce a priority	
MOO-HKS	Výroba HKS	Rozbitý sklo	0000700000	TUV_MECH	2018-03-04 15:59:48	MORMAITE	Start Stop	Operace	Akce a priority	
MOO-HKS	Výroba HKS	RAPIDA 142 - DW1	0000700001	TUE	2018-03-08 16:26:13	MORMAITE	Start Stop	Operace	Akce a priority	
MOO-HKS	Výroba HKS	RAPIDA 106 SRH	0000700001	TUE	2018-03-12 09:56:19	MORMAITE	Start Stop	Operace	Akce a priority	
MOO-HKS	Výroba HKS	Hřtk	0000700001	TUE	2018-03-13 10:55:15	MORMAITE	Start Stop	Operace	Akce a priority	
MOO-HKS	Výroba HKS		0000700001	TUE	2018-03-13 10:55:15	MORMAITE	Start Stop	Operace	Akce a priority	

Obrázek 4.13: Stránka pro údržbáře

### 4.5.2.6 Administrace

Stránka je navržena tak, aby odpovídala funkčním požadavkům spadající pod roli Administrátora. Musí zde být k dispozici seznam všech uživatelů a umožněno provádět nad jednotlivými uživateli požadované operace jako je editace atributů, rolí nebo jména a hesla uživatele.

**View** Administrace používá koncepčně trochu jiný typ přístup než předchozí stránky. Je zde použita komponenta SplitApp pomyslně rozdělující stránku na master a detail část. Ačkoli to není nutné, obecně se v master části očekává libovolný seznam prvků, jejichž výběrem se v detail části zobrazí podrobné informace příslušného prvku. V tomto případě se tak jedná o seznam uživatelů pro master část a jeho podrobné informace v detail části. Pro seznam je použita standardní listová komponenta obsahující pouze název uživatelského účtu. V detailní části je potom v hlavičce osobní číslo a jméno uživatele a pod tím záložky pro pracoviště, role a atributy. Struktura této stránky je k vidění v ukázce kódu 17 níže. Rozdílné chování je znát i na mobilních zařízeních. Master a Detail stránky jsou v tomto případě rozděleny a chovají se jakoby samostatné stránky, ačkoliv je za nimi schován jeden controller a sdílí všechny data tak, jako by jednou stránkou byly.

**Algorithm 17**


---

```

<SplitApp initialDetail="detailDefault "
    initialMaster="master">
  <masterPages>
    <Page id="master " >
      <customHeader />
      <content />
      <footer />
    </Page>
  </masterPages>
  <detailPages>
    <Page id="detailDefault">
      <customHeader />
      <content />
      <footer />
    </Page>
    <MessagePage id="detailNotFound " />
  </detailPages>
</SplitApp>

```

---

Ukázka má nastínit základní strukturu SplitApp komponenty. Základním stavebním kamenem jsou agregace masterPages a detailPages. Těm lze přiřadit libovolný počet komponent typu Page, jejichž struktura byla už dříve zmíněna v kapitole 4.5.2.2.

---

**Controller** Oproti předchozím controllerům zde musí být implementováno chování pro správný chod SplitApp aplikace. To obnáší vyčítání parametrů z URL z důvodu zjištění, jestli je nějaký konkrétní uživatel již vybrán. Příkladem může být cesta „/pm/#/administrace/MORMAITE)“, ze které lze poznat vybraný uživatelský účet MORMAITE a na základě toho z backendu získat příslušná data. Ale to se netýká pouze načítání dat. Je zapotřebí pro případ mobilních zařízení rozlišit, zdali má být zobrazena master nebo detail stránka. Pro to však lze nastavit jednoduché pravidlo. V případě, že je účet znám (například URL „/pm/#/administrace/MORMAITE)“), dojde k zobrazení detailu, jinak master částí se seznamem uživatelů k výběru (například URL „/pm/#/administrace/“).

**Stránka** Pro stránku byl zvolen rozdělený layout, který zřetelně odděluje seznam uživatelů od jejich detailu. V zápatí seznamu je tlačítko pro založení nového uživatele realizovaného pomocí formuláře zobrazeného v dialogovém okně. V detailní části jsou pak zobrazeny všechny požadované informace. Lze tak tedy například editovat osobní číslo a jméno uživatele. V záložkách jsou potom tabulky jednotlivých pracovišť, rolí a atributů, které může správce editovat. Podoba stránky pro desktop zařízení je vidět na obrázku 4.14 níže.

## 4. IMPLEMENTACE

Seznam uživatelů	Administrace										
ADMINITE >	<b>Základní údaje</b>  Osobní číslo: 10100716 <span>Upravit os. číslo</span> Jméno uživatele: Marcel Morávek II. <span>Upravit užv. jméno</span>  Pracoviště Role <u>Atributy</u>  <b>Seznam atributů</b> + - <table border="1"><thead><tr><th><input type="checkbox"/> klíč</th><th>Hodnota</th></tr></thead><tbody><tr><td><input type="checkbox"/> CONF_LIMIT</td><td>0</td></tr><tr><td><input type="checkbox"/> GEWRK</td><td>TUE</td></tr><tr><td><input type="checkbox"/> GEWRK_ELE</td><td>TUH_ELE</td></tr><tr><td><input type="checkbox"/> GEWRK_MECH</td><td>TUV_MECH</td></tr></tbody></table>	<input type="checkbox"/> klíč	Hodnota	<input type="checkbox"/> CONF_LIMIT	0	<input type="checkbox"/> GEWRK	TUE	<input type="checkbox"/> GEWRK_ELE	TUH_ELE	<input type="checkbox"/> GEWRK_MECH	TUV_MECH
<input type="checkbox"/> klíč		Hodnota									
<input type="checkbox"/> CONF_LIMIT		0									
<input type="checkbox"/> GEWRK		TUE									
<input type="checkbox"/> GEWRK_ELE		TUH_ELE									
<input type="checkbox"/> GEWRK_MECH		TUV_MECH									
BALMAMOO >											
BHB >											
DVOTOMOO >											
KONPAMOO >											
KUBDAMOO >											
KURMAITE >											
MORMAITE >											
MUHOTMOO >											
NEADMIN >											
R142HMOO >											
R142_MOO >											
R74_MOO >											
RADIVMOO >											

Obrázek 4.14: Stránka pro správu uživatelů

Stránka pro mobilní verzi je z důvodu rozdělení tentokrát zobrazena samostatně na obrázku 4.15 níže. Z důvodu zpětné navigace z detailu do seznamu přibila v hlavičce šipka zpět pro vykonání tohoto úkonu. Jinak je vzhled více-méně totožný s verzí pro desktop zařízení.

Seznam uživatelů	Administrace								
ADMINITE >	<b>Základní údaje</b>  Osobní číslo: 10100716 <span>Upravit os. číslo</span> Jméno uživatele: Marcel Morávek II. <span>Upravit užv. jméno</span>  Pracoviště Role <u>Atributy</u>  <b>Seznam atributů</b> + - <table border="1"><thead><tr><th><input type="checkbox"/> klíč</th><th>Hodnota</th></tr></thead><tbody><tr><td><input type="checkbox"/> CONF_LIMIT</td><td>0</td></tr><tr><td><input type="checkbox"/> GEWRK</td><td>TUE</td></tr><tr><td><input type="checkbox"/> GEWRK_ELE</td><td>TUH_ELE</td></tr></tbody></table>	<input type="checkbox"/> klíč	Hodnota	<input type="checkbox"/> CONF_LIMIT	0	<input type="checkbox"/> GEWRK	TUE	<input type="checkbox"/> GEWRK_ELE	TUH_ELE
<input type="checkbox"/> klíč		Hodnota							
<input type="checkbox"/> CONF_LIMIT		0							
<input type="checkbox"/> GEWRK		TUE							
<input type="checkbox"/> GEWRK_ELE		TUH_ELE							
BALMAMOO >									
BHB >									
DVOTOMOO >									
KONPAMOO >									
KUBDAMOO >									
KURMAITE >									
MORMAITE >									
MUHOTMOO >									
NEADMIN >									
R142HMOO >									
R142_MOO >									
R74_MOO >									

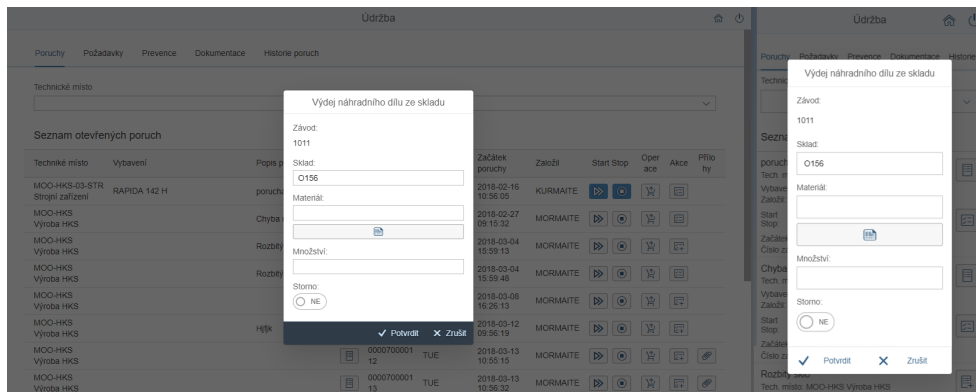
+ Založit Změna hesla - Odebrat

Obrázek 4.15: Stránka pro správu uživatelů - mobilní verze

### 4.5.2.7 Pomocné dialogy

V rámci aplikace vznikly přibližně tři desítky různých dialogů sloužící pro různé účely. Ty nejjednodušší z nich mají například jednoduchý účel vyžádání potvrzení úkonu od uživatele před provedením. Údržbář je například při ukončení práce dotázán, zdali chce svoji předat operátorovi výroby ke schválení. Tím dojde k omezení chyb vzniklých z důvodů neopatrnosti. Vznikly však i

dialogy trochu komplikovanější. Zpravidla se jedná o dialogy vyžadující od uživatele vyplnění nějakých dat potřebných pro provedení daného úkonu. Na obrázku 4.16 níže je například k vidění dialog pro vydání náhradního dílu ze skladu jak pro desktop tak mobilní zařízení.



Obrázek 4.16: Dialog pro vydání náhradního dílu

## 4.6 LOGIN SAPUI5 Aplikace

### 4.6.1 Struktura aplikace

Struktura aplikace je víceméně totožná s aplikací PM popsané v kapitole 4.5.1 výše. Jediné, v čem se aplikace zásadně liší je její zabezpečení popsané v souboru web.xml. V kapitole 4.4.1 věnující se autentizačnímu a autorizačnímu mechanismu JAAS popsána nutná definice zobrazená v kódu 2. Ta je použita v aplikaci PM, ale aplikace PM ji už postrádá. Z toho důvodu nepotřebuje i další soubory týkající se zabezpečení. A to HTML stránky login.html a error.html popsané v kapitole 4.

### 4.6.2 Login

Úkolem Login stránky je autentizace uživatele. K tomu jednoduše poslouží jednoduchý vstupní formulář pro jméno a heslo.

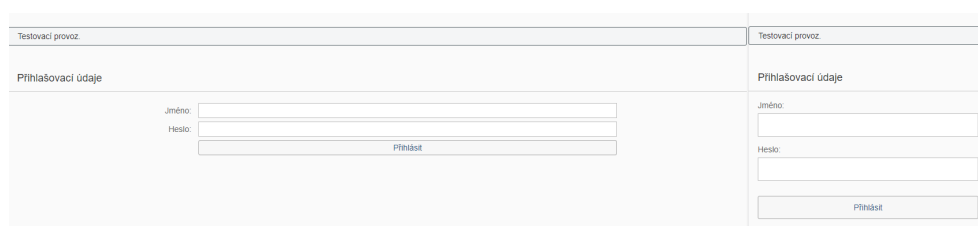
**View** Pro ten byl vybrán layout SimpleForm popsaný v ukázce 14.

**Controller** Úkolem je především komunikace s PMLoginModulem pro zajištění autentizace uživatele. Vyporádává se také s tím, že v případě pokusu uživatele přistoupit na konkrétní stránku nebo soubor zabezpečený pomocí JAAS, nedochází k implicitnímu přesměrování na domovskou stránku, ale přímo na tu cílenou. K tomu využívá dočasné lokální uložení prohlížeče, do

## 4. IMPLEMENTACE

---

kterého si při prvním pokusu o načtení uloží aktuální URL a tu se po autentizaci pokusí otevřít již dostupné aplikaci PM.



Testovací provoz	
Přihlašovací údaje	
Jméno:	<input type="text"/>
Heslo:	<input type="password"/>
<input type="button" value="Přihlásit"/>	

Testovací provoz	
Přihlašovací údaje	
Jméno	<input type="text"/>
Heslo:	<input type="password"/>
<input type="button" value="Přihlásit"/>	

Obrázek 4.17: Stránka pro přihlášení uživatele

### Stránka

## 4.7 Testování

V této sekci jsou popsány testy použité při vývoji a testování dvou vrstev navržené architektury. Jedná se o vrstvu realizující prostředníka v komunikaci PM aplikace a SAP GW dále označované jako middleware a poté samotnou UI vrstvu na úrovni Fiori aplikace.

### 4.7.1 Middleware

Pro ukázkou testovací na úrovni middleware jsem zvolil test třídy GWServlet představující prostředníka v komunikaci z aplikace PM na SAP GW. V následující ukázce kódu 18 je k vidění část rozhraní testovací třídy.

**Algorithm 18** Rozhraní testovací třídy pro Unit testování

---

```

public class GWServletTest {
    static GWServlet instance;
    public GWServletTest();

    public static void setUpClass();
    public static void tearDownClass();
    public void setUp();
    public void tearDown();

    public void testGetNotifList() throws Exception;
    public void testCreateNotifPU() throws Exception;
    public void testGetTplnrHierarchy() throws Exception;
    ...
}

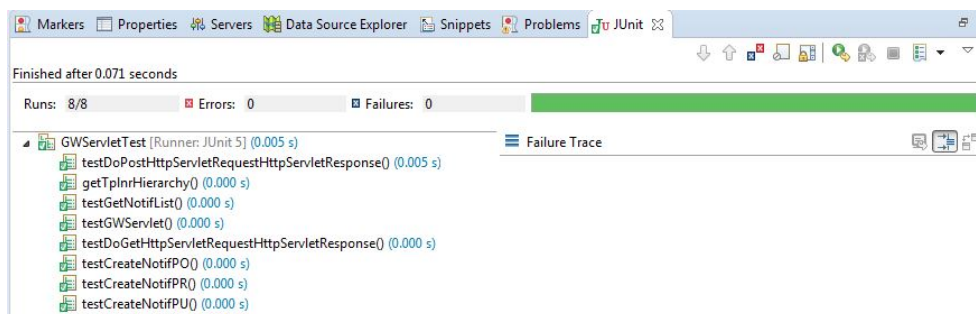
```

---

Při spuštění testu dojde k vytvoření instance třídy GWServlet z metody setUpClass, která je vyvolána před provedením prvního testu. Tím dojde na servletu k vyčtení hodnot potřebných pro HTTP komunikaci se SAP GW.

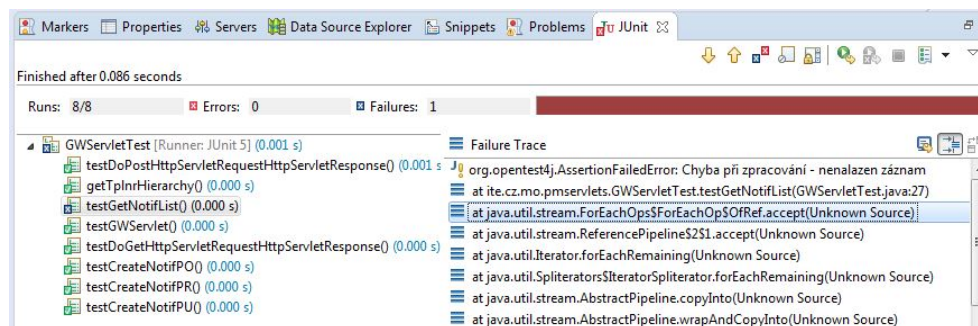
Metody setUp a tearDown zajišťují nezávislost testů, jsou totiž volány mezi jednotlivými testy a jde čas pro resetování inicializování instance nebo jiných aktivit nutných pro správnost testů.

Na následujících dvou obrázcích 4.18 a 4.19 jsou ukázky výstupu provedených jednotkových testů v prostředí Eclipse [23].



Obrázek 4.18: Úspěšné JUnit testy

## 4. IMPLEMENTACE



Obrázek 4.19: Neúspěšné JUnit testy

### 4.7.2 UI vrstva

K testování má framework SAPUI5 vlastní technologii OPA5 (One Page Acceptance Tests), která umožňuje testovat UI aplikace. Poskytuje asynchronní přístup k prvkům SAPUI5. Síla technologie spočívá pro testování uživatelské interakce, integraci, navigaci a svazování komponent s datovými modely. Technologie je založena na programovacím jazyku JavaScript, což je vzhledem k použití tohoto jazyka i aplikačního frameworku celkem očekávané a vhodné. V ukázce kódu 19 níže je k nahlédnutí struktura základního testu. V podstatě se jedná o HTML stránku, která si pomocí JavaScriptu natáhne potřebná view aplikace testované aplikace a za pomoci komponenty OPA5 na nich provádí testy.

---

**Algorithm 19** Ukázka testovacího skriptu v hlavičce testovací stránky OPA5

---

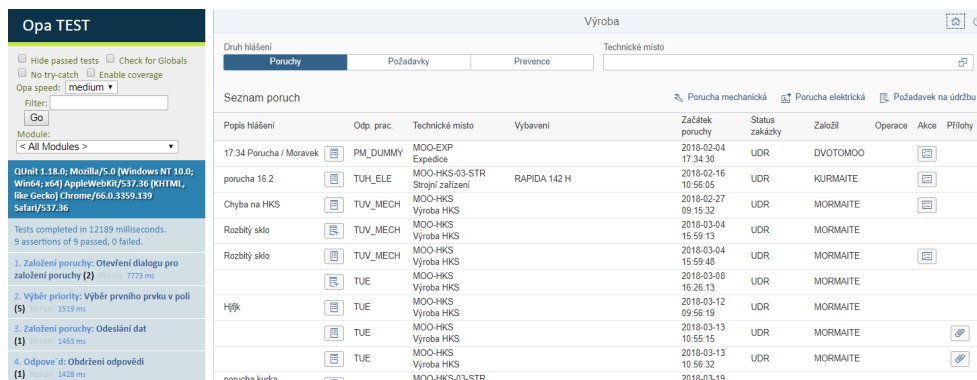
```
<script>
sap.ui.require(["../Administrace", ...], function () {
    QUnit.module("Zobrazení detailu");
    opaTest("ToDetail", function(Given, When, Then) {
        Given.iStartMyApp();
        Given.onTheIntro.iPressOnGoToDetail();
        When.onTheList.iPressOnGoToDetail();
        Then.onPage.iShouldSeeTheDetailPage().
        and.iTeardownMyAppFrame();
    });
});
QUnit.start();
</script>
```

Nastíhne strukturu testu pro otestování navigace ze seznamu uživatelů do jeho detailu.

---



Skládáním jednoduchých testů naznačených v kódu ?? lze vytvořit komplexní skládající se z několika kroků. Takovým testem pak může být otevření dialogového okna pro založení poruchy. Vybrání prvku a pokusu o vytvoření daného hlášení. Výsledkem takové testu je pak znázorněn na obrázku 4.20.



Obrázek 4.20: Úspěšný OPA5 test

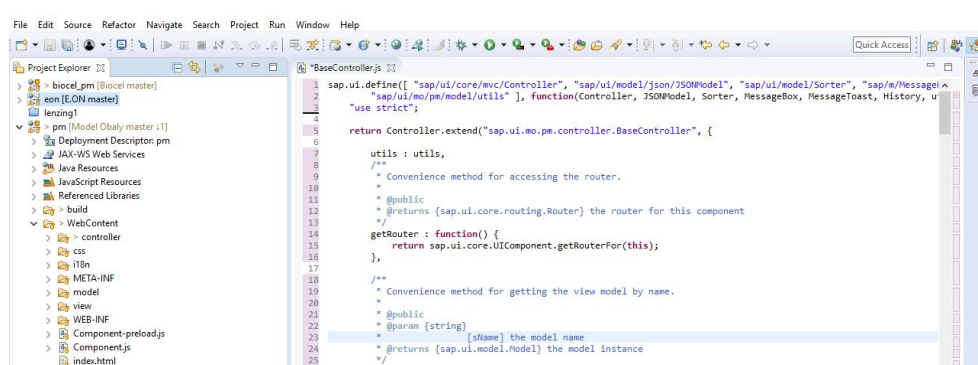
## 4.8 Porovnání vývojových prostředí

V této sekci jsou popsána dvě vývojová prostředí umožňující vývoj aplikací ve frameworku SAPUI5. Prvním z nich je open source vývojové prostředí Eclipse a tím druhým je cloudové prostředí SAP Web IDE.

### 4.8.1 Eclipse s pluginem pro SAPUI5

Eclipse umožňuje vyvíjet plnohodnotné aplikace ve frameworku SAPUI5. Jelikož se v zásadě jedná o standardní dynamický web projekt akorát s připojenými knihovnamy SAPUI5, vývoj tak probíhá dle očekávání. Umožňuje exportovat v mnoha variantách. Pro běh aplikace v Tomcat Apache je zapotřebí projekt vyexportovat do web archivu WAR. Ten poté stačí nahrát do kořenového adresáře web serveru, kde je následně rozbalen do potřebných souborů sloužících k přístupu do aplikace z webového prohlížeče. Podoba vývojové prostředí je viditelná na obrázku 4.21 níže.

## 4. IMPLEMENTACE



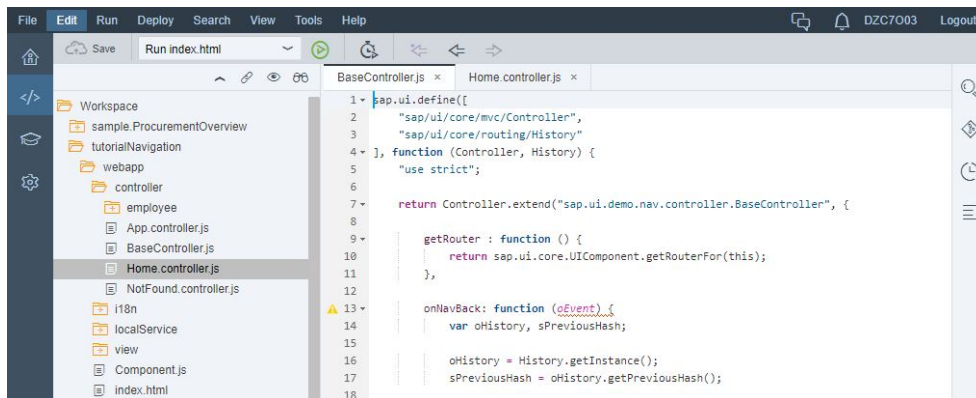
Obrázek 4.21: Podoba Eclipse

**Prostředí** Vzhled a pořadí oken se dá v Eclipse libovolně měnit, nicméně v základní verzi je v levé části seznam otevřených projektů, vedle něhož je prostor pro editaci jednotlivých souborů projektu. Ve spodní části je konzole a servery dostupné pro online testování. Je možné si zde nastavit Tomcat server, který má zpracovávat a spouštět projekt. Každá úprava v kódu pak může být okamžitě zkontrolována v rámci vývojového prostředí, což je při vývoji velmi užitečné.

### 4.8.2 SAP Web IDE

Reprezentuje modernější přístup k vývoji webových aplikací. Jedná se o cloudové vývojové prostředí, tudíž sdílet identické prostředí může více vývojářů současně. Kompletní vývoj je uložen ve sdíleném úložišti, které může být sdíleno do verzovacího prostředí Gitu. Nabízí širokou škálu různých dynamických funkcionalit, které pomáhají urychlit vývoj aplikace. Takovou funkcionalitou je především tvorba aplikací z předem připravených šablon. Dále se jedná o možnost rozšíření zakoupených aplikací od společnosti SAP. Zakoupená aplikace pro přehled objednávek tak může být například rozšířena o zákaznické pole u položky objednávky [24].

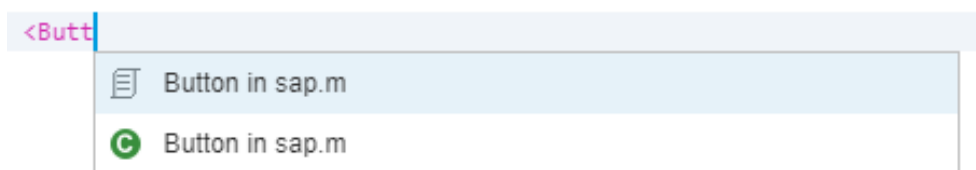
Určité omezení nastává v případě, že aplikace není cílena pro standardní nasazení očekávané od společnosti SAP. Z vývojového prostředí se očekává nahrání aplikace pouze na SAP GW. A to je znát i při vývoji aplikace z datového hlediska. V případě, že by bylo zapotřebí použít nějakou komunikační mezivrstvu (Servlet), je zapotřebí si takovou vrstvu vybudovat mimo SAP Web IDE. Standardní cestou vývoje se očekávají statické modely vytvořené v rámci aplikace nebo komunikace pomocí OData se SAP GW.



Obrázek 4.22: Podoba SAP Web IDE

**Prostředí** Je z designového pohledu velmi elegantní, neobsahuje žádné rušivé elementy. V levé části je seznam otevřených projektů a ve zbylé části místo pro vlastní vývoj, tedy editování jednotlivých souborů. Po stranách jsou umístěny dobře viditelná tlačítka pro verzovací nástroj Git. Při psaní kódu je k dispozici nápověda ve formě doplňování názvu včetně funkcí a jejich parametrů. To však nefunguje úplně stoprocentně v rámci JavaScriptových souborů. V případě složitějších konstrukcí a volání funkcí nadřazeného objektu se nepředávají dobře návratové objekty a jejich funkce. Proto u nich nedochází k výběru volaných metod ze všech možných. Dědění z nadřazených objektů však velmi dobře pracuje u XML definic UI.

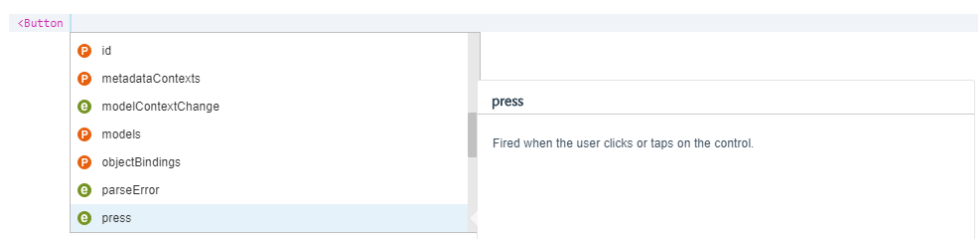
Pro ukázkou napovídání poslouží obrázky 4.23 a 4.24 zobrazené níže. První z nich ukazuje možnost definice tlačítka dvěma způsoby ještě před tím, než je dopsána plný název komponenty Button. První zobrazená varianta vytvoří v XML strukturu obsahující všechny atributy komponenty včetně všech možných agregací. Druhá varianta doplní pouze název komponenty.



Obrázek 4.23: Nápověda při výběru komponenty v SAP Web IDE

U již vybrané komponenty lze však využívat nápovědu jednotlivých atributů a agregací. K výčtu jednotlivých variant je dokonce k dispozici jeho popis a vlastnosti. Ukázka je viditelná na obrázku 4.24 níže.

## 4. IMPLEMENTACE



Obrázek 4.24: Nápověďa při definování atributů elementu v SAP Web IDE

### 4.8.3 Shrnutí

K porovnání obou přístupů poslouží tabulka 4.1. Obsahuje podstatné vlastnosti zmíněné napříč sekcemi 4.8.1 a 4.8.2.

Prostředí	Eclipse	SAP Web IDE
Podpora SAPUI5	+	+
Šablony pro tvorbu aplikací	-	+
Integrace na GitHub	+	+
Export do WAR	+	-
Online testování	+	±
Připojený servlet	+	-
Nápověda pro atributy elementů	-	+
Testování komunikace s GW (EPR)	+	±

Tabulka 4.1: Tabulka shrnující vlastnosti vývojových prostředí pro SAPUI5

Z tabulky plusově o trochu lépe vychází varianta Eclipse. To především proto, že obecně slouží pro vývoj v jazyce Java, který se velmi často používá pro integrační účely. Tudíž tvorba pomocných servletů nebo export do Web archivu zde není problém. Kde ovšem začíná tato varianta ztrácet na síle oproti SAP Web IDE je podpora pro vývoj ve frameworku SAPUI5. Množství nápověd při skládání komponent a vypsání controlleru k nim je v cloudu řešeno mnohem lépe. Z vývojářského pohledu ovšem postrádá možnost účinnějšího testování komunikace se SAP GW oproti realizaci s middleware vrstvou implementovanou pomocí Java servletů.

**Doporučení** V závislosti na cíleném run-time prostředí, ve kterém aplikace bude nahrána, je zapotřebí vybírat vhodné vývojové prostředí. Pokud se jedná o standardní SAP GW řešení pomocí protokolu OData, pak je jednoznačně

správnou volbou SAP Web IDE. V opačném případě se nejspíše očekává ohýbání standardní cesty, které s sebou přináší potřebu řešení komunikace pomocí více integrací a v tom případě se variantě zahrnující Eclipse víceméně nedá vyhnout. Víceméně vyhnout z toho důvodu, že je možné přistoupit k hybridnímu řešení. Je totiž možné UI část zahrnující XML view a JavaScriptové controllery vyvíjet v SAP Web IDE a v požadovaný moment vývoje daný projekt exportovat a nahrát do projektu v Eclipse. To lze provést buď manuální cestou exportu a importu nebo skrze verzovací systém Git. Tím lze dosáhnout uložení kódu v jednom uložišti a na konkrétních částech projektu pracovat v příslušných prostředích. UI část tak lze editovat z cloudu a část s Java vývojem a exportovacími parametry v Eclipse.

## 4.9 Doporučení pro vývoj

V této závěrečné jsou shrnuty poznatky z analýzy, návrhu a implementace do několika bloků s krátkými poznámkami, na které by se mělo při vývoji aplikace brát zřetel.

### 4.9.1 Seznámení se zadáním

V případě podnětu od zákazníků a partnerů nebo vlastní iniciativy je dobré uvědomit si celkový koncept očekávaný od aplikace a v závislosti na tom pokračovat následujícím bodem.

### 4.9.2 Kontaktáž potenciálního zákazníka - Lo-Fi prototyp

V této fázi je vhodné využít nástroje určené k rychlému prototypování a neztrácet čas vývoj prototypové aplikace, u které většinou nikdo nemá představu o finální podobě, v cílovém frameworku. Časová náročnost tvorby v prototypovacím nástroji Balsamiq popsaném v kapitole 3.3 je doslova zanedbatelná vůči potřebnému času pro vývoj prototypové aplikace ve frameworku SAPUI5.

### 4.9.3 Komunikace se vyvíjí slibně - Hi-Fi prototyp

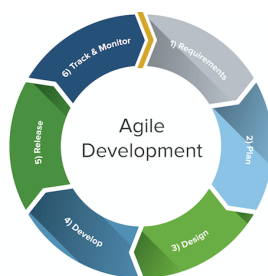
Je žádoucí se do této fáze vyhnout programování na straně controllerů a nebo případně začít v této fázi od znova. Špatným návrhem a rozmístěním funkcí poté vzniká ve finálové aplikaci mnoho redundantních částí kódů, které se špatně spravují a rozvíjí. V této části už by měla být co možná nejvíce eliminována možnost zásadních změn z pohledu jednotlivých komponent aplikace.

### 4.9.4 Implementace

V případě zelené od vedení (zodpovědného managementu) a započatí implementace je nutné si určit způsob, kterým bude aplikace vyvíjena. V násled-

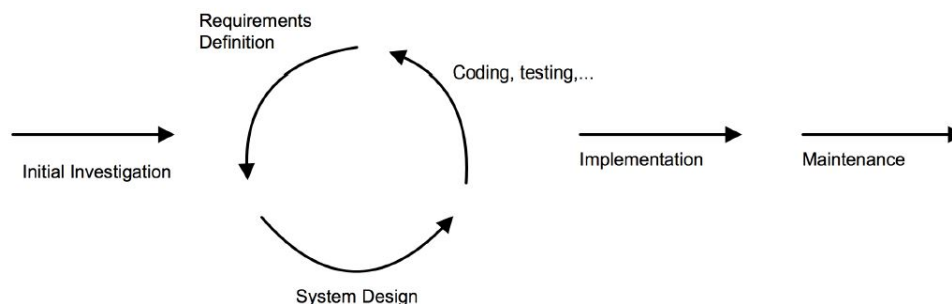
dujících odstavcích jsou popsány dvě metody, které by mohly být pro vývoj vhodné. Teorie k těmto metodikám je čerpána z dokumentu [7].

**Agilní vývoj** Agilní vývoj zahrnuje skupiny iteračních a inkrementálních metodik vývoje software. Pomocí této kombinace lze rychle reagovat na případné změny v požadavcích od zákazníka. Ten totiž pravidelně dostává aktuální podobu aplikace k testování v rámci sprintů (opakujících se vývojových cyklů). Diagram vývoje je na obrázku 4.25.



Obrázek 4.25: Diagram agilního vývoje [7]

**Prototyping** Je jedním z iterativních přístupů k vývoji. Dochází u něj k vytváření prototypů představujících jednotlivé komponenty výsledné aplikace. V případě, že se od očekávají izolovatelné funkcionality, může to být velmi vhodnou alternativou k agilnímu přístupu. Diagram vývoje je na obrázku 4.26.



Obrázek 4.26: Diagram prototypovacího vývoje [7]

**Verzování** Ačkoliv byly aplikace doposud vyvíjeny vždy pouze jedním vývojářem, je nezbytné kód revidovat pomocí verzovacího systému. Zabrání se tak možným ztrátám kódu již implementovaných aplikací a usnadní možnost obnovení zahozené funkcionality. Další výhodou využití verzovacího systému (v itelligence nově zavedený Git) je možnost issue trackingu.

---

# Závěr

Ve své práci jsem představil společnost SAP a její softwarový produkt SAP Enterprise Resource Planning. Stručně popsal význam jeho jednotlivých modulů a poté se podrobněji věnoval modulu údržby SAP Plant Maintenance. Následně jsem popsal produkt SAP Fiori a webovou technologii BSP použitou při implementaci komunikace mezi webovou aplikací a systémem SAP Gateway.

Na základě konzultací s PM konzultantem jsem sepsal funkční požadavky kladené na výslednou aplikaci Fiori a znázornil k nim nejdůležitější případy užití. Na základě těchto požadavků jsem vytvořil návrh uživatelského rozhraní a soustředil se u něho na dodržení správných postupů návrhu. K tvorbě prototypů jsem použil dva nástroje, které jsem následně porovnal. Popsal jsem heuristickou analýzu, jakožto nástroj na určení kvality uživatelského rozhraní z důvodu vhodného použití při testování prototypu zákazníkem.

Poté jsem na základě získaných znalostí o komponentách systému SAP, funkčních požadavcích kladených na webovou aplikaci a návrhu uživatelského rozhraní implementoval celý systém. V kapitole věnující se implementaci jsem popsal architekturu jako celek a poté podrobněji zdokumentoval její vrstvy. Nejpodrobněji je popsána část týkající se frameworku SAPUI5, kde jsou detailně popsány jeho jednotlivé stavební prvky. Poté jsou porovnány dvě vývojová prostředí uzpůsobená pro tento framework a na závěr jsem přidal ukázky možného testování uživatelského rozhraní a komunikační vrstvy tvořené Java servlety. Práce je uzavřena stručnou sadou doporučení pro budoucí vývoj aplikací v tomto frameworku.

## Budoucí vývoj

Věřím, že na základě získaných znalostí a následného zdokumentování vývoje budu moci svým kolegům ulehčit práci při vytváření podobných aplikací. Struktura aplikace je i v případě vývoje nad zcela jiným modulem lehce znovupoužitelná a může tak razantně urychlit vývoj.

### Dosažení cílů

S výsledkem své práce jsem spokojený. Podařilo se mi dosáhnout všech stanovených cílů, byť u některých částí méně, než jsem si na začátku představoval. Nicméně implementační část, kterou považuji za stěžejní v rámci této práce, byla již v menší modifikaci použita v ostrém provozu, a to mě velmi těší.



---

## Literatura

- [1] SAP R/3 informační systém [online]. [cit. 2018-04-07]. Dostupné z: <http://www.itica.cz/sap-r3-informacni-system/>
- [2] *SAP PM* [online]. [cit. 2018-04-08]. Dostupné z: [https://www.tutorialspoint.com/sap\\_pm/sap\\_pm\\_tutorial.pdf](https://www.tutorialspoint.com/sap_pm/sap_pm_tutorial.pdf)
- [3] SAP Business Server Pages [online]. [cit. 2018-04-09]. Dostupné z: [https://help.sap.com/SAPHELP\\_NWPI71/helpdata/en/e9/bb153aab4a0c0ee10000000a114084/frameset.htm](https://help.sap.com/SAPHELP_NWPI71/helpdata/en/e9/bb153aab4a0c0ee10000000a114084/frameset.htm)
- [4] SAP Fiori [online]. [cit. 2018-04-14]. Dostupné z: <https://experience.sap.com/fiori-design/>
- [5] SAP Fiori - The Basic Files of Your App [online]. *tutorialspoint*, [cit. 2018-04-08]. Dostupné z: <https://sapui5.hana.ondemand.com/#/topic/28b59ca857044a7890a22aec8cf1fee9.html>
- [6] Tvorba moderních webových aplikací pomocí SAPUI5 a oDATA služeb [online]. [cit. 2018-04-08]. Dostupné z: [https://theses.cz/id/0u6njb/zaverecna\\_prace.pdf](https://theses.cz/id/0u6njb/zaverecna_prace.pdf)
- [7] What are the Benefits of Prototyping? [online]. [cit. 2018-04-18]. Dostupné z: <https://nectarpd.com/what-are-the-benefits-of-prototyping/>
- [8] *SAP Company History* [online]. [cit. 2018-04-07]. Dostupné z: <https://www.sap.com/corporate/en/company/history.html>
- [9] Nasazením ERP systému práce nekončí, ale začíná [online]. *ERPForum*, [cit. 2018-04-07]. Dostupné z: <https://www.erpforum.cz/erp-systemy/nasazenim-erp-systemu-prace-nekonci-ale-zacina.html>

- [10] SAP Modules - SAP FI, CO, SD and more. [cit. 2018-04-16]. Dostupné z: <https://www.simplilearn.com/sap-modules-sap-fi-sap-co-sap-sd-sap-hcm-and-more-rar111-article>
- [11] SAP Fiori Documentation [online]. [cit. 2018-05-26]. Dostupné z: <https://sapui5.hana.ondemand.com/#/topic>
- [12] Úvod do JSON [online]. [cit. 2018-04-08]. Dostupné z: <https://www.json.org/json-cz.html>
- [13] Softwarové inženýrství I. [online]. *ČVUT FIT*, [cit. 2018-04-08]. Dostupné z: <https://edux.fit.cvut.cz/archive/B171/BI-SI1/lectures/start>
- [14] Projektové a změnové řízení [online]. *ČVUT FIT*, [cit. 2018-04-11]. Dostupné z: [https://edux.fit.cvut.cz/archive/B171/MI-PCM.16/\\_media/lectures/mi-pcm\\_prednaska\\_1.pdf](https://edux.fit.cvut.cz/archive/B171/MI-PCM.16/_media/lectures/mi-pcm_prednaska_1.pdf)
- [15] Návrh uživatelského rozhraní [online]. *ČVUT FIT*, [cit. 2018-04-14]. Dostupné z: <https://edux.fit.cvut.cz/archive/B171/MI-NUR/lectures/start>
- [16] Low-fi prototyping: What, Why and How? [online]. [cit. 2018-04-16]. Dostupné z: <https://mobgen.com/low-fi-prototyping/>
- [17] Balsamiq Mockups [online]. [cit. 2018-04-14]. Dostupné z: <https://balsamiq.com/>
- [18] Built.me [online]. [cit. 2018-04-20]. Dostupné z: <http://built.me>
- [19] Mobilní aplikace na hlášení závad ve městě [online]. [cit. 2018-04-12]. Dostupné z: <https://dspace.cvut.cz/bitstream/handle/10467/65128/F8-DP-2016-Soukup-Karel-thesis.pdf?sequence=1&isAllowed=y>
- [20] Apache Tomcat [online]. *Apache Tomcat*, [cit. 2018-04-24]. Dostupné z: <http://tomcat.apache.org/>
- [21] J2EE authentication and authorization with JAAS [online]. *ORACLE*, [cit. 2018-04-17]. Dostupné z: <http://www.oracle.com/technetwork/topics/index-089689.html>
- [22] BASIC and FORM Authentication [online]. [cit. 2018-04-16]. Dostupné z: <http://java.boot.by/wcd-guide/ch05s03.html>
- [23] Eclipse [online]. *Eclipse*, [cit. 2018-04-08]. Dostupné z: <https://www.eclipse.org/>
- [24] SAP Web IDE [online]. *SAP*, [cit. 2018-04-11]. Dostupné z: <https://www.sap.com/developer/topics/sap-webide.html>

## Seznam použitých zkratek

**SAP** Systems - Applications - Products in data processing

**ERP** User Interface

**ABAP** Advanced Business Application Programming

**GW** Gateway

**BSP** Business Server Page

**JAAS** Java Authentication and Authorization Service (Java autentizační a autorizační servis)

**UI** User Interface (uživatelské rozhraní)

**MVC** Model - View - Controller (Model - Uživatelské rozhraní - Řídící logika)

**XML** Extensible markup language

**JSON** JavaScript Object Notation



## Obsah přiloženého CD

thesis.....	textová část diplomové práce spolu se zadáním
└ DP_Morávek_Marcel_2018.pdf .....	diplomová práce ve formátu PDF
└ assignment.pdf .....	zadání diplomové práce ve formátu PDF
src	
└ fiori .....	zdrojové soubory SAPUI5 aplikací
└ └ pm .....	hlavní SAPUI5 aplikace
└ └ pmlogin .....	login SAPUI5 aplikace
└ └ pmloginmodule .....	JAAS login modul
└ latex .....	zdrojové soubory textové části diplomové práce
└ └ images .....	použité obrázky