

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

7. května 2018

Poděkování

Poděkování

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. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 7. 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

Abstrakt CZ

Klíčová slova SAP, Fiori

Abstract

Abstrakt EN

Keywords SAP, Fiori

Obsah

Úvod	1
1 Cíl práce	3
1.1 Vývojová část	3
1.2 Rešeršní část	3
1.3 Co není cílem práce	3
2 SAP	5
2.1 Společnost SAP	5
2.2 SAP R3	5
2.3 SAP Plant Maintenance (PM)	8
2.4 SAP BSP	12
2.5 SAP FIORI	14
3 Analýza a návrh aplikace	21
3.1 Základní popis aplikace	21
3.2 Uživatelské role	21
3.3 Model požadavků	22
3.4 Model případů užití (Use Case Model)	27
4 Návrh uživatelského rozhraní	31
4.1 Task Groups	33
4.2 Task Graph	33
4.3 Lo-Fi prototyp	34
4.4 Porovnání prototypovacích nástrojů	43
5 Implementace	46
5.1 SAP ERP	47
5.2 SAP GW	47
5.3 Apache Tomcat	50

5.4	PM SAPUI5 Aplikace	54
5.5	LOGIN SAPUI5 Aplikace	72
5.6	Porovnání vývojových prostředí	73
5.7	Testování	77
5.8	Doporučení pro vývoj	78
Závěr		79
Literatura		81
A Seznam použitých zkratk		83
B Obsah přiloženého CD		85

Seznam obrázků

2.1	Moduly SAP R3	7
2.2	Proces diagram PM	12
2.3	Struktura BSP aplikace	13
2.4	Struktura BSP aplikace	15
2.5	Struktura BSP aplikace	16
2.6	Struktura BSP aplikace	16
2.7	Struktura BSP aplikace	18
2.8	Struktura BSP aplikace	20
3.1	Diagram případu užití pro správu poruch	28
3.2	Diagram případu užití pro správu poruch	29
3.3	Diagram případu užití pro správu poruch	30
4.1	Diagram případu užití pro správu poruch	33
4.2	Diagram případu užití pro správu poruch	34
4.3	Diagram případu užití pro správu poruch	34
4.4	Diagram případu užití pro správu poruch	35
4.5	Diagram případu užití pro správu poruch	35
4.6	Diagram případu užití pro správu poruch	36
4.7	Diagram případu užití pro správu poruch	37
4.8	Diagram případu užití pro správu poruch	38
4.9	Diagram případu užití pro správu poruch	39
4.10	Diagram případu užití pro správu poruch	39
4.11	Diagram případu užití pro správu poruch	40
4.12	Diagram případu užití pro správu poruch	41
5.1	Diagram případu užití pro správu poruch	46
5.2	Seznam funkčních modulů v ERP systému	47
5.3	Seznam funkčních modulů v ERP systému	48
5.4	Ilustrační databázové schéma pro uživatelská data portálového uživatele	49

5.5	Struktura paketu realizující autentizaci a autorizaci	52
5.6	Struktura paketu realizujícího middleware vrstvu mezi PM SA- PUI5 aplikací a SAP GW	53
5.7	Struktura PM aplikace s hlavní prezentační logikou	55
5.8	Struktura jednotlivých view v Eclipse projektu	59
5.9	Struktura controllerů v Eclipse projektu	59
5.10	Úvodní stránka PM SAPUI5 aplikace	64
5.11	Stránka pro založení požadavku na údržbu	65
5.12	Stránka pro operátora údržby	67
5.13	Stránka pro údržbáře	69
5.14	Stránka pro správu uživatelů	71
5.15	Stránka pro správu uživatelů - mobilní verze	71
5.16	Dialog pro vydání náhradního dílu	72
5.17	Stránka pro přihlášení uživatele	73
5.18	Podoba Eclipse	74
5.19	Podoba SAP Web IDE	75
5.20	Nápověda při výběru komponenty v SAP Web IDE	75
5.21	Nápověda při definování atributů elementu v SAP Web IDE	76
5.22	Nápověda při definování atributů elementu v SAP Web IDE	78
5.23	Nápověda při definování atributů elementu v SAP Web IDE	78

Seznam tabulek

5.1	Tabulka shrnující vlastnosti vývojových prostředí pro SAPUI5 . . .	76
-----	--	----

Úvod

Tato práce se zabývá ...

Cíl práce

Cílem této práce je vytvoření webové SAP Fiori aplikace nad SAPovským modulem údržby ve frameworku SAPUI5. Pomocí této aplikace bude umožněno realizovat servisní zakázky i preventivní údržbu strojů a to včetně jejich vybavení.

1.1 Vývojová část

Cílem praktické části je navržení uživatelského rozhraní aplikace s ohledem na způsob zacházení s modulem údržby. Nadále pak implementace samotné aplikace dle provedeného návrhu.

1.2 Rešeršní část

Jedním z cílů rešeršní části je porovnání prostředí podporujících vývoj ve frameworku SAPUI5.

1.3 Co není cílem práce

Cílem této práce není implementace ani návrh funkčnosti uvnitř EPRového systému. Tato práce začíná na úrovni komunikačních rozhraní jednotlivých funkčních modulů realizujících požadované operace.

SAP

Kapitola obecně popisuje podnikový informační systém SAP. V jednotlivých podkapitolách jsou pak popsány informace o historii firmy a architektonické struktuře systému. Dále jsou zde popsány i jednotlivé technické komponenty, které jsou použity pro realizaci požadované aplikace.

2.1 Společnost SAP

Společnost SAP je v současné době jedním z největších poskytovatelů podnikových aplikací a jednou z největších 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 anglických slov „Systems - Applications - Products in data processing“. Zaměřují se na vývoj a provoz softwarových produktů podporujících podnikové procesy. Zejména pak tedy řízení podniku, systémy pro správu vztahu se zákazníky a v současné době pak především vývoj technologií pro webové aplikace a cloud computing [1].

2.2 SAP R3

První verze systému SAP R/1, tvořená pouze finančním účetnictvím, byla vydána již v roce 1973. Následující verze SAP R/2, vydaná o šest let později, se dá již označit za první funkční ERP systém (Enterprise resources planning). Ovšem nevýhodou tohoto systému byla vysoká technická náročnost na klienta, která v tu dobu přinášela nutnost využívání sálových počítačů. Verzi SAP R/3 z roku 1992 však byla kompletně změněna architektura SAPu. Změnou architektury na klient-server opadly nároky na klienta a odpadla tak tehdejší nutnost využívání sálových počítačů a zároveň se začaly využívat relační databáze. Hlavní výhoda této architektury spočívala především však pak v kompatibilitě s různými platformami a operačními systémy Microsoft

2. SAP

Windows nebo Unix. Tím se společnost dostala na špici poskytovatelů ERP systémů a na té se do dnes drží.

Architektura systému SAP R/3 Spolu se změnou architektury s příchodem verze SAP R/3 na model klient-server, došlo k rozčlenění do tří vrstev:

- **Databázová vrstva** je tvořena vlastními databázovými servery, které slouží pro ukládání dat. Jelikož SAP je multiplatformní systém, vývojáře nemusí zajímat, na jaké platformě (UNIX, ORACLE, SUN, MICROSOFT nebo jiné) databázová vrstva běží, protože na aplikační vrstvě bude přístup vypadat vždy stejně.

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

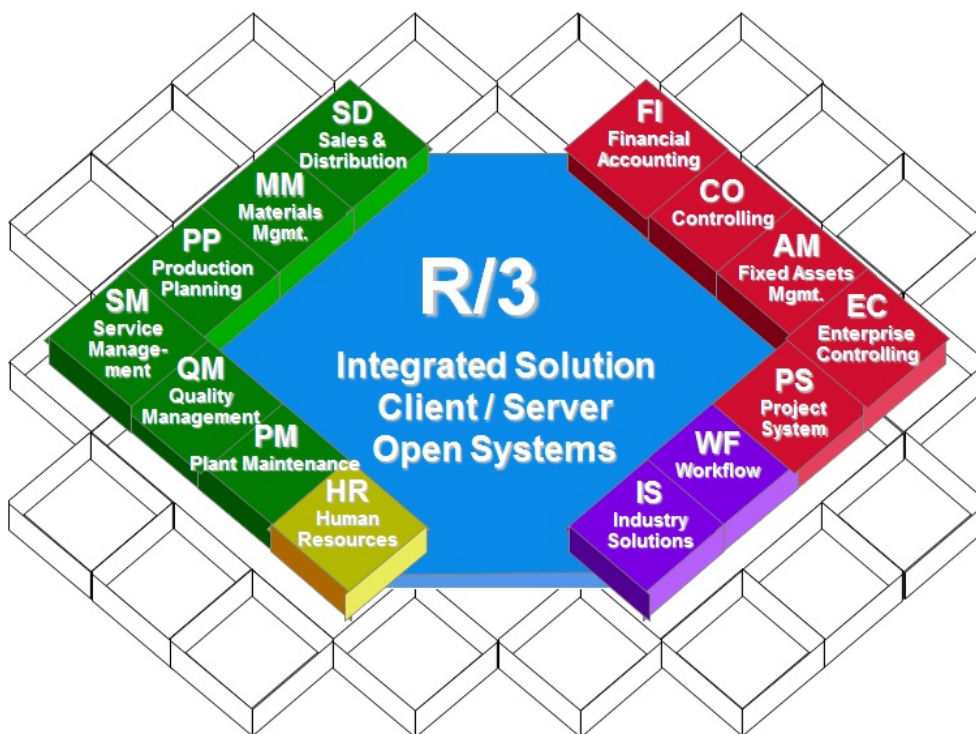
- **Aplikační vrstva** slouží především jako prostředí pro vykonávání programové logiky na straně aplikačního serveru. Centrálně se na něm zpracovávají data, které se načítají a ukládají z databázové vrstvy. Jednotlivé programy (funkce) se v tomto prostředí píšou zpravidla v SAPovském programovacím jazyce ABAP, někdy však i pomocí Javy.

Příkladem budiž uživatelský požadavek o vyčtení zakázek na výrobní stroj pro daného uživatele. Aplikační vrstva nejdříve musí načíst z databázové vrstvy potřebná data o uživateli, strojích v továrně i zakázkách a ty poté následně dle požadavku zpracovat. Vyloučí tak například stroje, ke kterým uživatel nemá oprávnění nebo zakázky nerelevantní k danému času a strojům. Tím získá požadovanou informaci, se kterou je poté následně nějakým způsobem naloženo, například přenosem do vyšší, prezentační vrstvy.

- **Prezenční vrstva** slouží pro komunikaci mezi uživatelem a počítačem. 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 přístupem přes webový prohlížeč. Typickým příkladem jsou SAPovské aplikace Fiori vytvořené ve frameworku SAPUI5, kterým je věnována samostatná sekce.

2.2.1 Moduly SAP R3

Systém SAP R/3 je vnitřně rozdělen do několika různých modulů. Každý z nich pak řeší konkrétní problematiku firmy.



Obrázek 2.1: Moduly SAP R3

fghfghgddh

- **Financial Accounting (FI)** označuje finanční účetnictví a je jedním z nejdůležitějších modulů SAP ERP. Používá se k uložení finančních dat organizace a pomáhá analyzovat finanční podmínky společnosti na trhu.
- **Controlling (CO)** podporuje koordinaci, monitorování a optimalizaci všech procesů v organizaci. Zahrnuje správu a konfiguraci základních dat, které pokrývají náklady a výnosy, interní objednávky a další nákladové prvky a funkční oblasti. Jeho hlavním účelem je plánování. Umožňuje určit odchylky srovnáním skutečných dat s údaji plánu a tím umožňuje řídit obchodní toky v organizaci.
- **Asset Management (AM)** slouží k optimální správě fyzického majetku organizace. Zahrnuje takové funkcionality jako jsou návrh, konstrukce, provoz, údržba a výměna zařízení. Spravuje majetek v jednotlivých odděleních (obchodních jednotkách).
- **Project system (PS)** je nástroj pro správu dlouhodobých projektů. Umožňuje uživatelům plánovat finanční prostředky i zdroje a kontrolovat jednotlivé části projektu tak, aby bylo zaručeno včasné dodání pokud možno v rámci rozpočtu.

2. SAP

- **Workflow (WF)** umožňuje navrhovat a realizovat obchodní procesy v rámci aplikačních systémů SAP. Zajišťuje aby se práce dostala v požadovaný čas do rukou správným lidem. Jeho cílem je usnadnění automatizace podnikových procesů.
- **Industry Solutions (IS)** poskytuje specifická řešení pro desítky industriálních odvětví jako například pro automobilový, chemický či energetický průmysl.
- **Human Resources (HR)** umožňuje organizaci strukturálně a efektivně zpracovávat informace údaje týkající se zaměstnanců k potřebám obchodním požadavkům.
- **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 činností.
- **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 výrovním kapacitou spolu s vytvořením plánů k dokončení komponentů a produktů.
- **Quality Management (QM)** je modul úzce provázaný s moduly MM, PP či PM a nedílnou součástí logistického řízení. Používá se k prováděnější kvalitativních funkcí jako je plánování jakosti, zajištění a kontroly kvality ve výrobním a spotřebním procesu.
- **Sales and Distribution (SD)** se používá pro ukládání údajů o zákaznících a produktech organizace. Pomáhá řídit fakturaci, prodej a přepravu produktů či služeb organizace. Řídí vztah se zákazníky od počáteční nabídky až po prodejní zakázku a fakturaci produktu.

2.3 SAP Plant Maintenance (PM)

Modul SAP Plant Maintenance je komplexní řešení, které poskytuje nástroje pro kompletní údržbu v rámci organizace. Veškeré prováděné činnosti jsou vzájemně propojeny s návaznými moduly (Production Planning, Materials Management a Sales and Distribution) v rámci podnikových procesů. Modul umožňuje provádět komplexní plánování, realizovat denní činnosti údržby nebo zaznamenávat případné vzniklé problémy. Díky provázanosti na ostatní moduly taktéž dovede sledovat a plánovat materiálové aktivity případně zaznamenávat i určovat dané náklady vzniklé údržbou.

K realizování zmíněných aktivit je modul rozdělen do následujících podmodulů:

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

2.3.1 Technické objekty

Jelikož je v organizaci zapotřebí správně a efektivně spravovat jednotlivé aktivity v rámci procesů modulu Plant Maintenance, je struktura údržby rozdělena na technické objekty. Ty slouží k definování jednotlivých typů strojů, kterými organizace disponuje. Za použití charakteristiky technických objektů lze pak zadefinovat jiný technický objekt, což umožňuje hierarchicky definovat strukturu organizace. Výčtem zmíněných vlastností technických objektů získáme následující výhody.

- Doba potřebná pro správu jednotlivých technických objektů je snížena.
- Zpracování údržby je zjednodušeno.
- Doba strávená při zadávání dat během zpracování údržby je značně snížena.
- Konkrétnější, důkladnější a rychlejší vyhodnocení údajů o údržbě.

Technická správa objektů se skládá z následujících činností:

- **Inspekce** - měřit a sledovat aktuální stav technického objektu
- **Preventivní údržba** - předvídat potřebu oprav a udržovat optimální stav technického objektu
- **Oprava** - měření a obnovení technického objektu
- **Další činnosti související s údržbou**

Zpracování údržby pomáhá řídit skutečné údržbářské práce prováděné v údržbě. Proces se skládá ze tří oblastí:

- **Upozornění na údržbu** - oznamte poruchu nebo popište technickou podmínku objektu
- **Objednávka údržby** - provést podrobný plán údržby a sledovat průběh práce a uhradit náklady na údržbu
- **Historie údržby** - uložení důležitých údajů údržby pro vykazování a vyhodnocení

2.3.2 Preventivní údržba

Preventivní údržba je dlouhodobý proces, jehož cílem je zajistit vysokou použitelnost zařízení a funkčních míst a minimalizovat prostoje způsobené opravami. Tato funkce podporuje údržbu založenou na výkonu, pokud jsou měřicí body nebo čítače používány pro řízení technických podmínek objektu. Součástí preventivní údržby lze použít k:

- Uložit seznam úkolů, které mají být provedeny
- Upřesněte rozsah inspekčních prací, preventivní údržbu a plánování činností
- Zadejte opakovanou frekvenci údržby
- Upřesněte přiřazení kontrolních činností a preventivní údržbu na základě nákladů
- Vyhodnotit náklady na budoucí preventivní údržbu a inspekční práci

Preventivní údržba v organizaci se používá k zabránění selhání systému a rozpadu výroby. Pomocí preventivní údržby můžete ve vaší organizaci dosáhnout různých výhod. Preventivní údržba se používá k provádění inspekcí, preventivní údržby a oprav. Plány údržby slouží k definování dat a rozsahu úkolů preventivní a inspekční údržby, které lze naplánovat pro technické objekty.

Seznam úkolů v Preventivní údržbě je definován jako sled činností, které jsou prováděny v rámci preventivní údržby v organizaci. Jsou používány k provádění opakovaných úkolů v rámci preventivní údržby a k jejich efektivnímu provedení.

Pomocí seznamů úkolů můžete snížit úsilí standardizací pracovní postup. Všechny aktualizace se provádějí na jednom konkrétním místě v seznamu úkolů údržby a všechny položky údržby a údržby v systému obdrží aktualizovaný stav pracovních postupů. Pomocí seznamů úkolů pomáhá při snižování úsilí potřebného pro vytvoření objednávek údržby a položek údržby, jak můžete vrátit do seznamu úkolů, abyste viděli pracovní postup. Klíčové funkce seznamů úkolů v SAP Plant Maintenance jsou následující plánovaná a probíhající údržba podrobněji popsány v následujících odstavcích.

Plánovaná údržba Všechny plánované činnosti, jako je kontrola, údržba a opravy, jsou součástí plánované údržby. V údržbě rostlin definujete časové intervaly, kdy je třeba pracovní kroky provést a pracovní sekvence, ve kterých musí být provedeny. Seznamy úkolů jsou při plánování plánování údržby přiřazeny plánu údržby.

Probíhající údržba Seznam úkolů pro průběžnou údržbu obsahuje pracovní postupy založené na aktuální kontrole. Všechna kontrola, která se provádí bez pravidelného rozvrhu, je předmětem trvalé údržby.

2.3.3 Zpracování údržby

Zpracování údržby se skládá z několika úrovní, které nemusí být nutně plně realizovány.

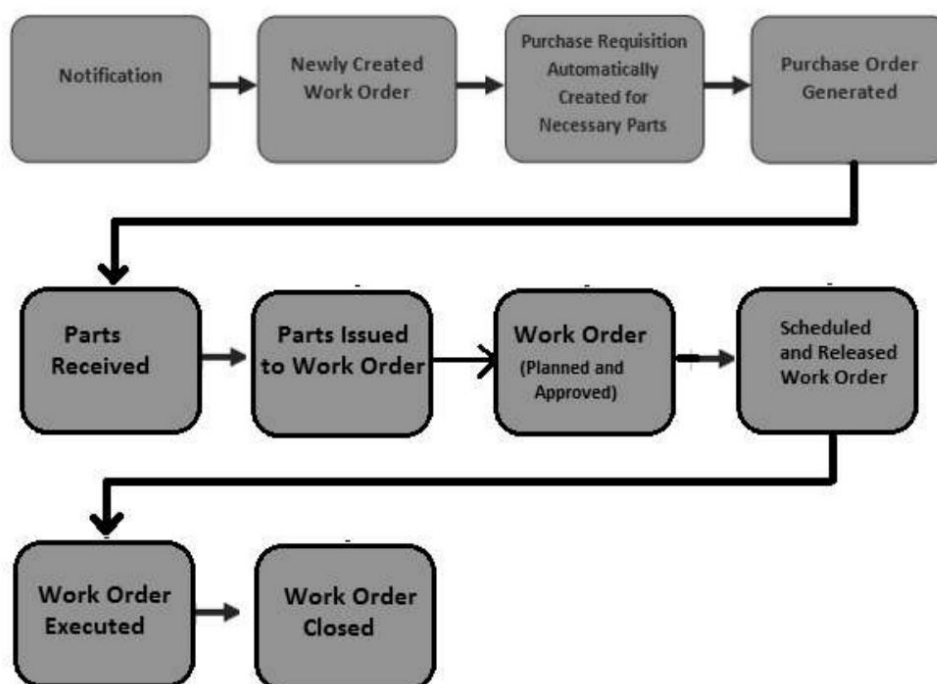
Proto je možné zpracovat opravu v mnoha fázích plánování, jako je předběžná kalkulace, plánování práce, materiálové zabezpečení, plánování zdrojů a povolení. Je však také možné okamžitě reagovat na škody způsobené událostmi, které způsobí vypnutí výroby, a v co nejkratší možné době předložit požadované objednávky a prodejní doklady s minimálními údaji.

Zpracování údržby lze rozdělit na následující tři oblasti:

- **Popis stavu objektu** - Nejdůležitějším prvkem v této oblasti je oznámení o údržbě. Používá se k popisu stavu technického objektu nebo hlášení poruchy na technickém objektu a požadavek na opravu poškození.
- **Provádění úkolů údržby** - Nejdůležitějším prvkem v této oblasti je objednávka údržby. Používá se k detailnímu plánování provádění údržbářských činností, sledování průběhu práce a vypořádání nákladů na údržbu.
- **Dokončení úkolů údržby** - Nejdůležitějším prvkem v této oblasti je historie údržby. Používá se k dlouhodobému uložení nejdůležitějších údajů o údržbě. Tyto údaje lze kdykoli vyžádat k vyhodnocení.

Tyto prvky umožňují zpracovat všechny úkoly, které je třeba provést v údržbě zařízení, stejně jako operace, které nepatří přímo do údržby zařízení, jako jsou investice, restrukturalizace, úpravy a podobně.

2. SAP



Obrázek 2.2: Proces diagram PM

Procesní diagram PM

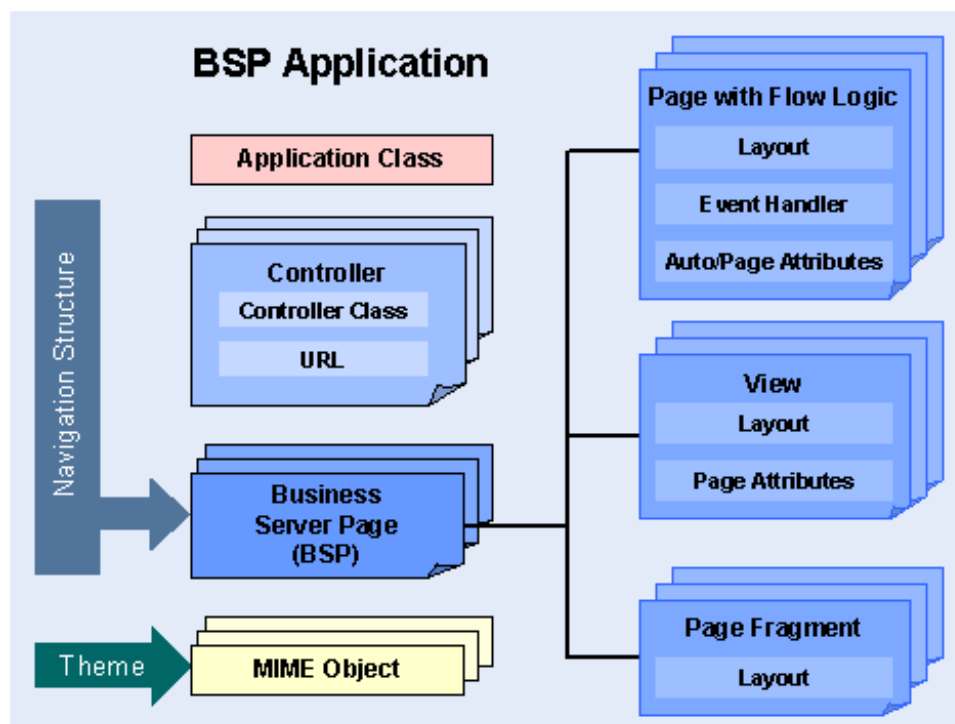
2.4 SAP BSP

Tato sekce se zaměřuje na front-endovou technologii SAP BSP. Jedná se o jednu z technologií použitých v cílové architektuře řešení mobilní aplikace a proto je zde stručně popsána její struktura.

SAP Web Application Server SAP WAS je produktový produkt aplikačního serveru a nová generace produktu Basis. Poskytuje všechny funkce, které Basis udělal, a pak mnohem víc. Přemýšlejte o tom jako o nadpřirozené základně. Přemýšlejte o tom jako o obalení základny s obrovskými možnostmi webových aplikací, mezi které patří i schopnost spouštět aplikace Java / J2EE vedle aplikací ABAP. A chci říci, že přidání Java / J2EE k tomuto aplikačnímu serveru v žádném případě neohrožuje podporu pro ABAP. Všechny vaše investice do řešení ABAP jsou dobře chráněny. Rozdíl spočívá v tom, že oba vývojové a běhové prostředí ABAP a Java / J2EE se nacházejí na jedné platformě, na jedné společné infrastruktuře. Tato sjednocená oblast systému ABAP / Java minimalizuje úsilí učitele o výuku a náklady na správu.

BSP Business Server Page (BSP) je kompletní funkční aplikace, stejně tak jako klasická transakce SAP. Rozhraním pro přístup však není software SAPGUI, spíše však libovolný webový prohlížeč. Díky využití protokolů HTTP nebo HTTPS je protokol používán pro přístup k aplikaci po celé síti, což umožňuje používání standardních produktů, jako jsou firewally a proxy servery.

Programovací program Stránky Business Server je podobný technologii serverových stránek. Zaměřením programovacího modelu BSP jsou body, které zajišťují optimální strukturu v rozhraní a obchodní logiku.



Obrázek 2.3: Struktura BSP aplikace

Struktura BSP aplikace

Každá BSP stránka využívá stejného principu zpracování dat. Obecně lze na BSP stránce definovat šest základních událostí, na které se zpracovávají data.

- **OnCreate:**
- **OnRequest:**
- **OnInitialization:**

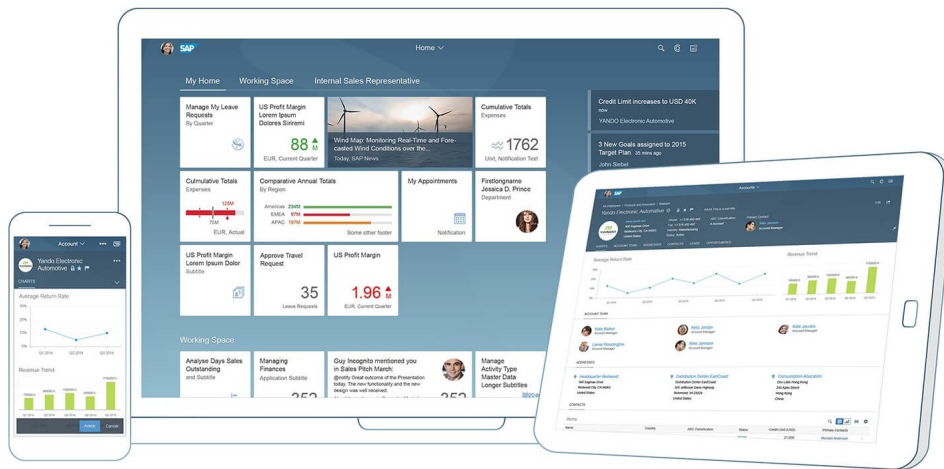
2. SAP

- **OnManipulation:**
- **OnInputOutputProcessing:**
- **OnDestroy:**

2.5 SAP FIORI

Stejně jako každý jiný tradiční počítačový software byly také přístupné ERP systémy prostřednictvím grafického uživatelského rozhraní na stacionárních místech PC nebo notebooků. Vzhledem k rychlému vývoji mobilních zařízení očekává velké množství uživatelů zlepšení a příležitosti pro mobilní použití.

V minulosti, mnoho zákazníků SAP vyjádřili svou nespokojenost ohledně staromódní vzhled a dojem z obrazovek SAP, stejně jako nedostatek exkluzivního přístupu přes desktop GUI pro většinu operací (jako schvalování objednávky, vytvoření prodejní objednávky, samostatně výdělečně servisní úkoly, vyhledávání informací.) Zpětná vazba byla oceněna a SAP podnikla kroky ke zlepšení použitelnosti a dostupnosti. (Bince 2015, 365) Dne 15. května 2013 představila společnost SAP platformu SAP Mobile Platform 3.0, otevřenou platformu, která byla k dispozici vývojářům softwaru. Společně se zavedením platformy zahájila společnost SAP svůj nový mobilní produkt s názvem Fiori. (SAP Fiori 2013, citováno 9. listopadu 2016.) Tento nový produkt je založen na pěti zásadách návrhu (obrázek 4). Prostřednictvím tohoto nového mobilního řešení uživatelsky orientované klienti nyní měl přístup k novým řešením, kde sbírka aplikací bylo možné použít na různých zařízeních, jako jsou stolní počítače, chytré telefony a tablety. V prvním vydání Fiori bylo zařazeno 25 aplikací, které slouží klientům v jejich nejčastějších obchodních funkcích. (SAP Fiori 2013, citováno 9. listopadu 2016.)

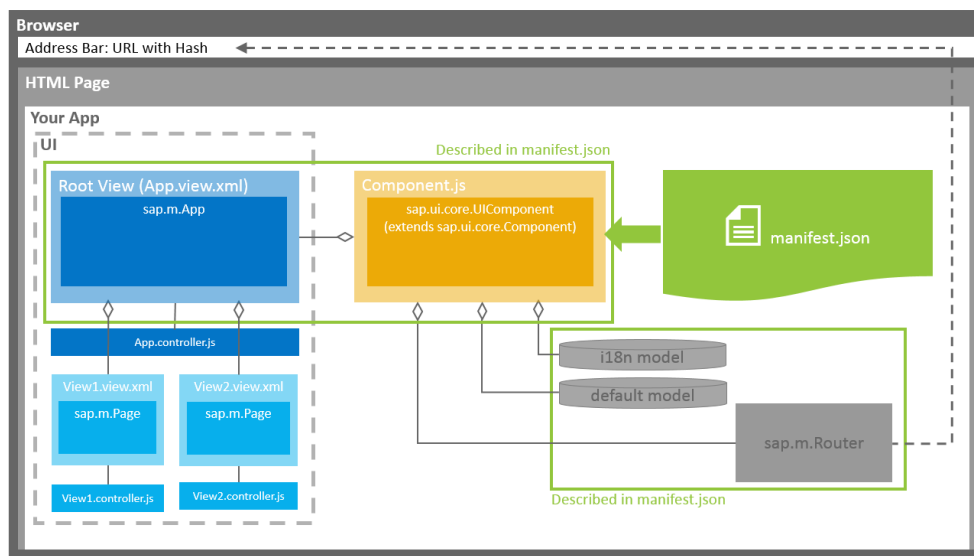


Obrázek 2.4: Struktura BSP aplikace

Struktura BSP aplikace

- Založené na rolích: - Uživatelsky orientované aplikace závislé na odpovědnosti uživatele - uživatel může mít více rolí a spouštět různé úkoly v několika doménách
- citlivý: - založené na formátu HTML5; pracuje bez problémů na různých zařízeních a velikostech obrazovky - automaticky upravuje rozložení aplikací na dostupné obrazovce - podporuje různé režimy interakce, jako klávesnice, myši a dotykové vstupy
- Jednoduché: - Jednoduché uživatelské rozhraní podporuje rychlé a snadné dokončení úkolů - má přístup 1: 1: 3: jeden uživatel, jeden případ, tři obrazovky (stolní, tabletové, mobilní)
- koherentní: - uživatel může mít mnoho aplikací, které mají stejný design a použitelnost - Snadno se naučíte nové aplikace poté, co se učíte používat jednu aplikaci Fiori
- Okamžitá hodnota: - stejný návrhový vzorec v aplikacích snižuje čas a náklady na školení nových uživatelů

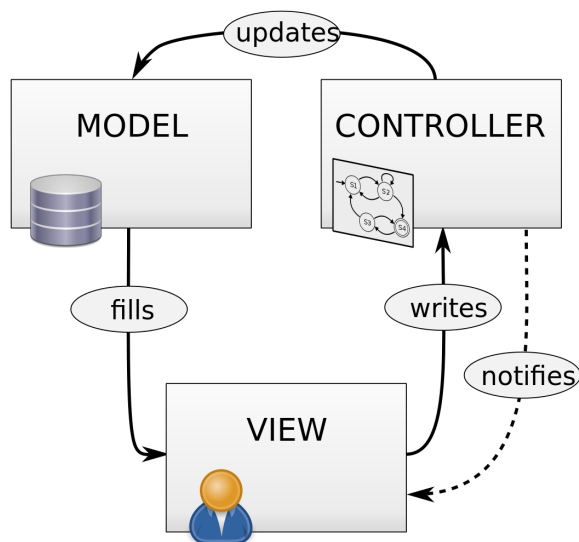
2. SAP



Obrázek 2.5: Struktura BSP aplikace

<https://sapui5.hana.ondemand.com/#/topic/28b59ca857044a7890a22aec8cf1fee9.html>

2.5.1 MVC



Obrázek 2.6: Struktura BSP aplikace

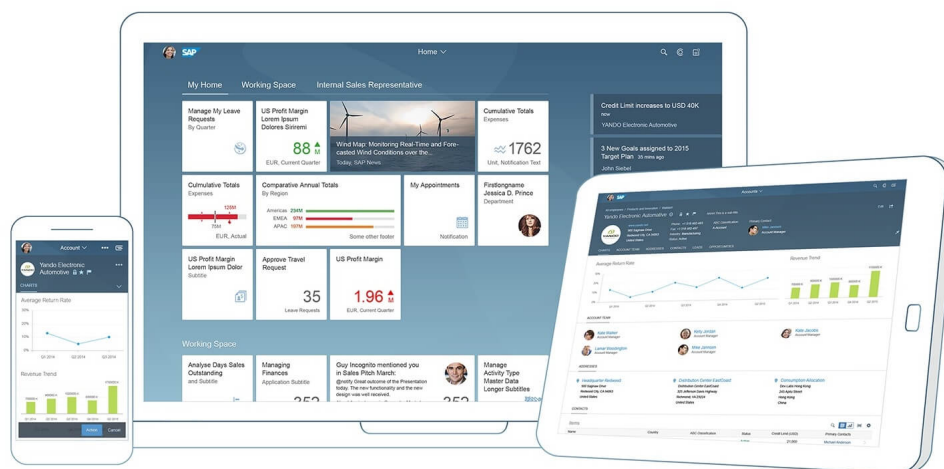
Struktura BSP aplikace

Model reprezentuje správu vlastních dat, nad nimiž aplikace pracuje: to může být třeba obrázek, textový dokument, databáze uložená na serveru SQL, ... Je dobré si uvědomit rozdíl mezi daty samotnými a jejich správcem (tedy právě tím modelem, o němž zde hovoříme). Třeba takový konkrétní obrázek jsou data; skupina tříd, umožňující obrázky načítat ze souborů a opět do nich ukládat, zjišťovat a měnit jejich atributy, převádět jejich formáty a pracovat s jejich obsahem – to je správce. V běžící aplikaci pak samozřejmě bude základním objektem vrstvy modelu nějaká vhodná instance, jež bude reprezentovat vlastní obrázek a nabízet všechny odpovídající služby.

View zahrnuje všechny objekty (grafického) uživatelského rozhraní a jejich služby. Uživatel aplikace s ní komunikuje výhradně prostřednictvím těchto objektů; právě ony mu prezentují data z modelu ve vhodné formě, a naopak pouze jejich prostřednictvím uživatel dává aplikaci příkazy, jež určují její další činnost. View má s modelem společnou tu nejdůležitější věc: jeho objekty nejsou závislé na konkrétní aplikační logice. Grafické uživatelské rozhraní aplikace tak můžeme podle potřeby (a podle požadavků zákazníků) snadno kdykoliv měnit, aniž by bylo zapotřebí přitom nějak zasahovat do aplikační logiky (nebo do-konce do modelu).

Controller mezi datovým modelem a objekty grafického uživatelského rozhraní, jež tvoří vzhled, stojí vrstva controller; právě na její úrovni je implementována funkční logika aplikace, takové věci jako „stiskne-li uživatel tohle tlačítko, provede se támhleto akce, a v tomto textovém poli se zobrazí výsledek“. (Čada, 2009)

2. SAP



Obrázek 2.7: Struktura BSP aplikace

Struktura BSP aplikace

2.5.2 SAP ODATA

Protokol OData umožňuje vytváření datových služeb založených na webovém protokolu REST (representational state transfer), který umožňuje uživatelům provádět CRUDQ operace nad zdroji identifikovanými pomocí Uniform Resource Identifier (URI) a definovanými v datovém modelu použitím jednoduchých HTTP zpráv.

Protokol původně vyvinul Microsoft, verze 1.0, 2.0 a 3.0 jsou uvolněny pod Microsoft Open Specification Promise. Aktuálně nejnovější verze 4.0 byla schválena jako standard prostřednictvím OASIS OData Technical Committee, jejímiž členy jsou BlackBerry, IBM, Microsoft, SAP a další. Následující informace ohledně OData protokolu se budou vztahovat k verzi 2.0, která je momentálně v SAPu navzdory zavádění verze 4.0, používána nejčastěji.

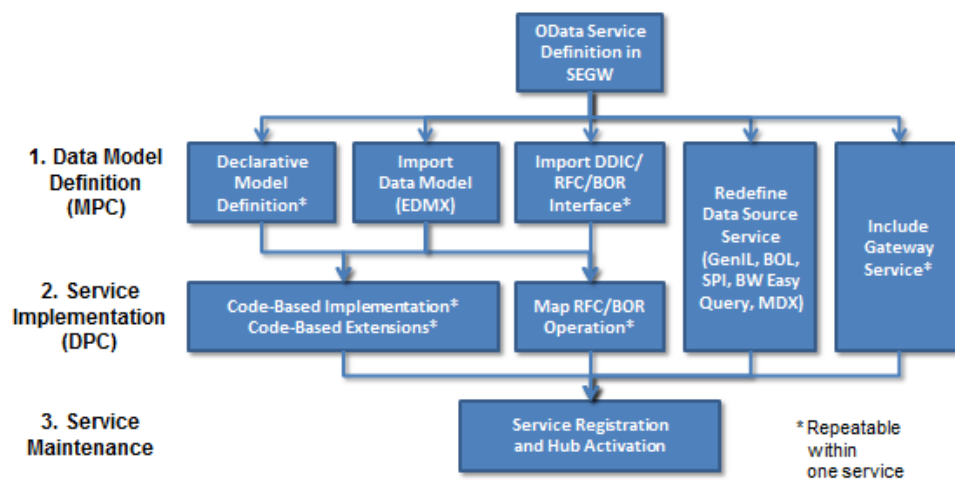
Data přenášená prostřednictvím OData protokolu mohou využívat různé datové formáty běžně používané ve webových technologiích, například Extensible Markup Language (XML), JavaScript Object Notation (JSON) nebo Atom Publishing Protocol (AtomPub) a další. Data jsou při přenosu zabalena do protokolu HTTP,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

Jádrum OData jsou feeds, které jsou kolekcemi Collections složené ze záznamů Entries. Každý Entry je identifikovaný klíčem a reprezentuje strukturovaný záznam, který má seznam vlastností Properties, ty mohou být komplexního, nebo primitivního typu. Entries mohou být součástí hierarchie typů a mohou mít související entries a feeds odkazované prostřednictvím Links.

Metadata OData služba poskytuje metadata dokumenty. Aby mohli uživatelé oData služby prozkoumat, co všechno služba nabízí, aniž by museli zkoumat implementaci služ-by na backendu. Jednodušší Service Document se nachází v root URI služby, obsahuje seznam všech feedů, takže je uživatelé mohou prozkoumat a zjistit jejich adresy. Přidáním segmentu \$metadata do root URI služby získáme Service Metadata Document, který popisuje celý datový model, jinými slovy strukturu a propojení všech zdrojů. Jak uvádí (Odata, 2013), Service Metadata Document popisuje svá data pomocí termínů EDM použitím XML jazyka pro popis modelů nazvaných Conceptual Schema Definition Language (CSDL). Tento CSDL dokument je pak zabalen použitím formátu EDMX. (Odata, 2015)

Entity Data Model (EDM) Hlavním konceptem v EDM jsou entity a asociace. Entity jsou instance Entity Type (například Inventura, Závod, Budova a tak dále), které jsou strukturovanými zá-znamy s klíčem Entity Key a složenými z pojmenovaných a typovaných vlastností. Entity Key je složen z podmnožiny vlastností v Entity Type. Entity Key (například InventuraID, nebo BudovaID) je zásadní koncept pro unikátní identifikaci instancí Entity Type a umožnění Entity Type instancím fungovat ve vztazích. Entity jsou seskupovány v Entity Sets (například Budovy je množina instancí Entity Type Budova). Asociace definují vztah mezi dvěma nebo více Entity Type (například Budova patří Závodu). Instance asociací jsou seskupovány v Association Sets. Navigation Properties jsou speciální vlastnosti v Entity Type, které jsou vázány na konkrétní asociaci a mohou být použity k odkazování na asociaci entity. Položením předchozích definic do OData termínů, feedy vystavované OData službou jsou reprezentovány pomocí Entity Set, nebo Navigation Property na Entity Type, které identifikuje kolekci entit. Například Entity Set identifikovaný pomocí URI <http://services.odata.org/OData/OData.svc/Products>, nebo kolekce entit identifikovaná pomocí Products Navigation Property v [http://services.odata.org/OData/OData.svc/Categories\(1\)/Products](http://services.odata.org/OData/OData.svc/Categories(1)/Products) identifikují feed složený z Entry vystavovaný OData službou. (Odata, 2015)

2. SAP



Obrázek 2.8: Struktura BSP aplikace

Struktura BSP aplikace

Analýza a návrh aplikace

Tato kapitola se věnuje analýze a návrhu požadované webové aplikace pracující nad SAP modulem Plant Maintenance pro vykonávání údržby v halách. Jednotlivé podkapitoly se pak věnují zpracování požadavků kladených na výslednou aplikaci, omezení aplikace a definování uživatelských rolí, které budou v aplikaci použity. Na základě stanovených požadavků je posléze proveden návrh uživatelského rozhraní.

3.1 Základní popis aplikace

Webová aplikace bude svým uživatelům umožňovat vykonávat potřebné úkony pro správný chod výrobních hal. To zahrnuje důležité operace jako je například hlášení požadavků na údržbu nebo poruch pro zadané vybavení, evidování práce na nahlášených poruchách apod. Podrobný soupis funkčních požadavků je popsán v následující kapitole.

3.2 Uživatelské role

Uživatelské role popsané této sekci jsou vytvořeny na základě požadavků v kapitole níže. Ovšem pro lepší čitelnost jsou popsány zde, ještě před důvody jejich vzniku.

3.2.1 Údržbář

Uživatel, který primárně řeší plánované údržby z preventivních důvodů (například periodicky nastavené v modulu PM), povinnosti vyřešit vznesený požadavek nebo nahlášené poruchy.

3.2.2 Operátor výroby

Uživatel, jehož primárním účelem je obsluha strojů na jeho pracovišti ve výrobním procesu. S aplikací přijde do styku pouze v případě, že bude chtít nahlásit poruchu na daném vybavení. Jednotlivá pracoviště jsou vybavena stolními počítači na kterých jsou spuštěny aplikace pro provoz. Z těchto aplikací bude umožněn odkázání se do budoucí webové aplikace pro nahlášení poruchy na příslušném pracovišti.

3.2.3 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. To bude zpravidla provádět z přenosného zařízení, které má neustále u sebe. Tím může být například chytrý telefon nebo tablet.

3.2.4 Správce - administrátor

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

3.3 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.

3.3.1 Funkční požadavky

Jsou takové požadavky, které musí být ve výsledné aplikaci implementovány, aby byla splněna požadovaná funkcionalita. Požadavky jsou rozděleny do 8 sekcí označených jako F1 až F7. Jedná se především o funkcionality spojené s nahlásováním poruch nebo požadavků na údržbu spolu s úkony prováděných nad již vzniklými hlášeními.

3.3.1.1 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, hala nebo závod), bude výběrová struktura příslušně omezena. Provedení výběru by mělo být 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 požaduje požadovanou údržbu provést. 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 práce, která je vykazována pomocí zpětného hlášení daným pracovištěm na zakázku PM.
- **Popis poruchy:** Jedná se o stručný popis hlášení, jasně přibližující daný problém (například došlý materiál nebo opotřebení).

Parametry, které uživatel nebude svévolně volit jsou následující.

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

Typickým uživatelem využívající tento funkční požadavek bude mistr, vedoucí směny a údržbáři.

3.3.1.2 F2: Nahlášení poruchy

Jelikož se technicky na úrovni modulu PM jedná o stejný záznam jako při zakládání požadavku na údržbu, je výčet potřebných parametrů totožný. Nicméně jelikož se hlášení poruchy reálně očekává od jiného typu uživatelů, je výběr následujících parametrů trochu odlišný. Typicky bude poruchu nahlašovat pracovník ve výrobě například na konkrétní lince.

- **Vybavení:** Jelikož se očekává mnohem menší počet vybavení, které bude umožněno uživateli vybrat, nebude se provádět výběr z hierarchické struktury technických míst a vybavení, ale bude k dispozici jenom takové, které spadá pod určité pracoviště. Hierarchické uspořádání však zůstane zachováno. Provedení výběru by mělo být 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.

3.3.1.3 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, kdy poruchu nahlásil a status 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, technicky realizována 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 omylné založení.
- **Vyřešení poruchy:** Poté, co se poruše začal někdo věnovat (zahájil na ní práci - došlo k založení zakázky PM) a svou práci ukončil a nikdo jiný už na ní nepracuje, je možné poruchu ukončit. Poté bude údržbář vyzván k odborné specifikaci poruchy. Dostane za úkol specifikovat část objektu, poškození a příčinu. V takovém případě dojde k ukončení celého procesu a dané hlášení již v aplikaci 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 změnit. Materiál, množství a možnost poté uživatel zadá ručně. Potvrzením zadaných parametrů dojde k vyskladnění požadovaného materiálu.
- **Správa akcí:** Hlášení mají jasný výčet operací, které při práci s poruchou může provést. Jedná se například o informaci objednání náhradního dílu nebo 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 v SAPu 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.

3.3.1.4 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 a status daného hlášení) pro správné zacházení s nimi. Uživateli budou umožněny stejné operace jako při správě poruch 3.3.1.3, kromě změn v následujícím seznamu. Vyřešení poruchy ze správy poruch se požadavků na údržbu netýká.

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

3.3.1.5 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 3.3.1.4, kromě změn v následujícím seznamu.

3.3.1.6 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.

3.3.1.7 F7: Administrace uživatele

Bude umožněna obecná správa účtů uživatelů, tedy základní operace jako přidání nebo odebrání účtu, nastavení jména uživatele a osobního čísla odpovídajícímu osobnímu číslu (identifikátor 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.

3.3.2 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 požadavků. Navržené požadavky pro výslednou aplikaci jsou rozděleny do 5 sekcí označených jako N1 až N4.

3.3.2.1 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 SAPu, požaduje se zachování stejného UI frameworku SAPUI5.

3.3.2.2 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.

3.3.2.3 N3: Spolehlivost a spravovovanost

Aplikace bude umožňovat logování činnosti uživatelů v systému z důvodu lepší identifikaci chyb. A to minimálně z počátku provozu aplikace. Taktéž bude umožněno i v případě vyššího zatížení musejí výt veškeré transakční kroky prováděné uživatelem uskutečněny.

3.3.2.4 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.

3.4 Model případů užití (Use Case Model)

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

3.4.1 Seznam účastníků

Níže zmínění účastníci odpovídají standardnímu pracovnímu modelu stanovenému v organizaci.

- **Operátor výroby:** Vychází z navržené uživatelské role operátor výroby 3.2.2. Na přiděleném pracovišti provádí přidělenou výrobní činnost.
- **Údržbář:** Odpovídá navržené roli údržbář 3.2.1, zpravidla bude disponovat i rolí pro zakládání požadavků na údržbu 3.2.3.
- **Správce / Administrátor:** Vychází z navržené uživatelské správce / administrátor 3.2.4.

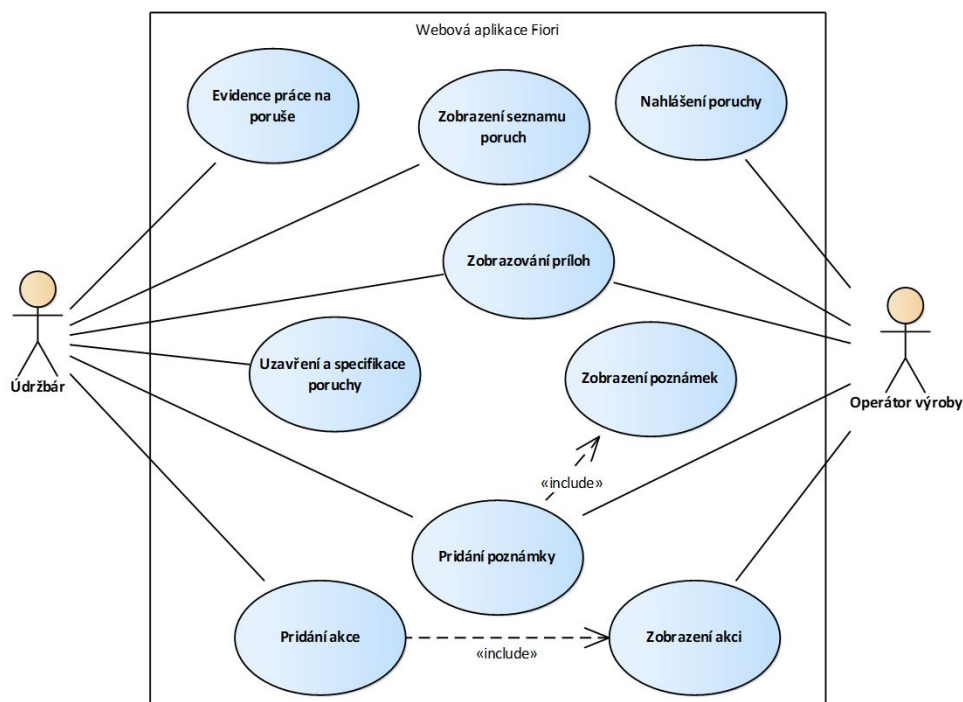
Nicméně nic administrátorovi nebrání tomu role různě kombinovat, nemají mezi sebou výlučný vztah. Tudíž administrátor může mít klidně i role údržbáře a operátora výroby, čímž je schopen provádět jejich příslušné operace.

3.4.2 Diagram případů užití

Souží 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.

3.4.2.1 UC1: Správa poruch

Následující případ užití týkající se správy poruch zahrnuje funkční požadavky pro hlášení poruch 3.3.1.2 a jejich následnou správu 3.3.1.3.

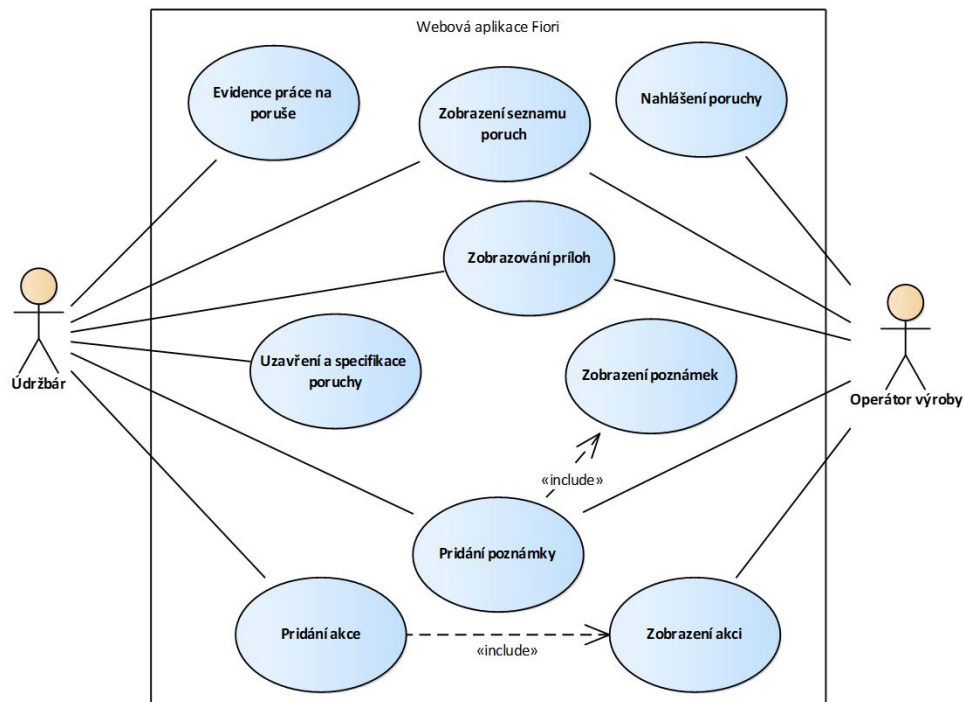


Obrázek 3.1: Diagram případu užití pro správu poruch

3.4.2.2 UC2: Správa požadavků na údržbu

Následující případ užití týkající se správy poruch zahrnuje funkční požadavky pro hlášení poruch 3.3.1.2 a jejich následnou správu 3.3.1.3.

3.4. Model případů užití (Use Case Model)

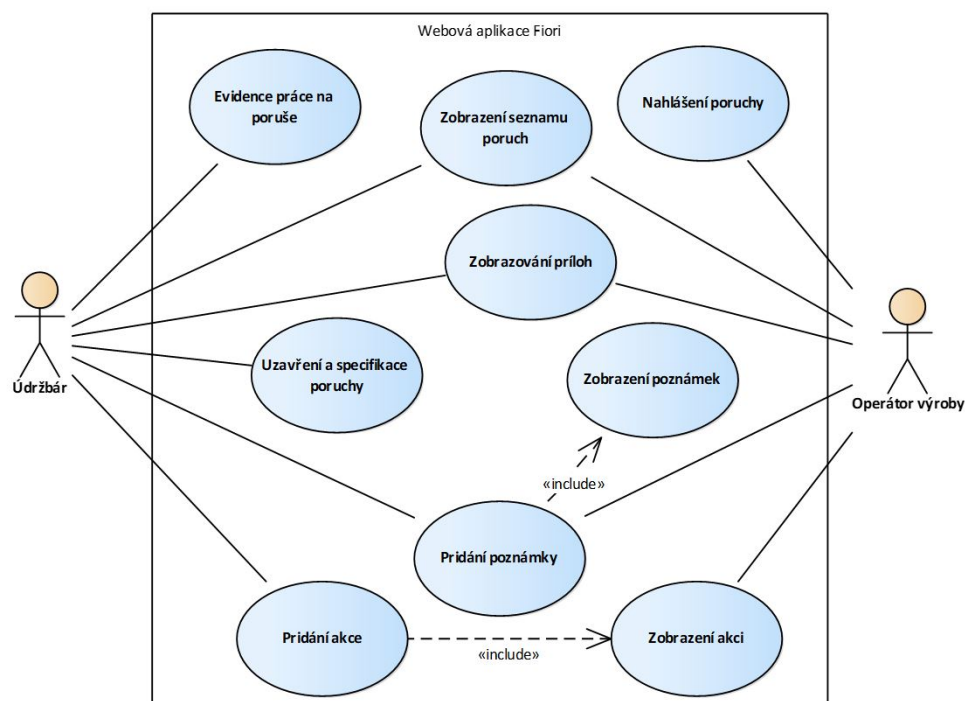


Obrázek 3.2: Diagram případu užití pro správu poruch

3.4.2.3 UC3: Správa prevencí

Následující případ užití týkající se správy poruch zahrnuje funkční požadavky pro hlášení poruch 3.3.1.2 a jejich následnou správu 3.3.1.3.

3. ANALÝZA A NÁVRH APLIKACE



Obrázek 3.3: Diagram případu užití pro správu poruch

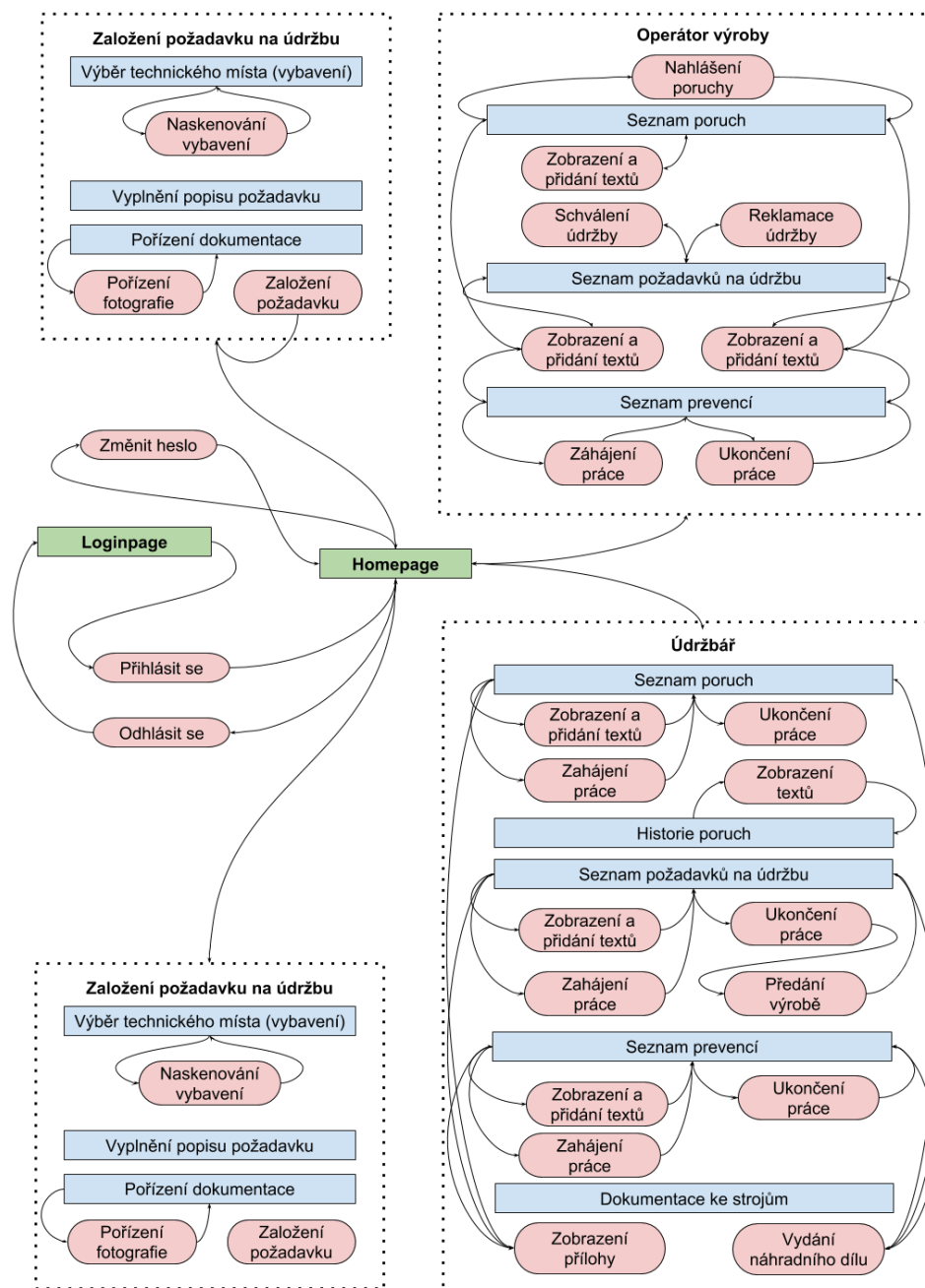
Návrh uživatelského rozhraní

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.

4. NÁVRH UŽIVATELSKÉHO ROZHRANÍ

4.1 Task Groups

4.2 Task Graph



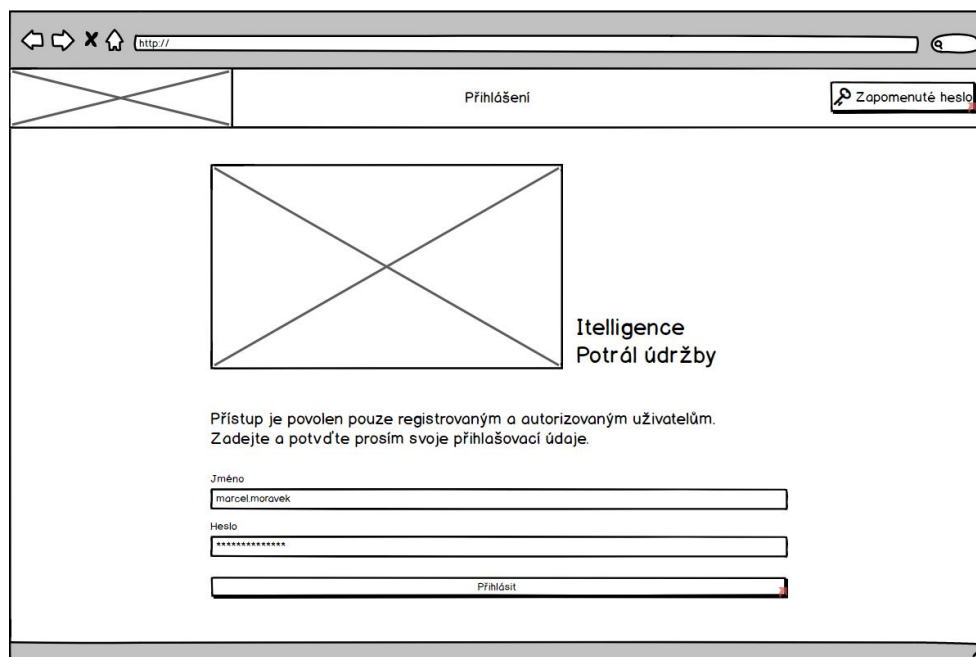
Obrázek 4.1: Diagram případu užití pro správu poruch

4. NÁVRH UŽIVATELSKÉHO ROZHRANÍ

4.3 Lo-Fi prototyp

4.3.1 Balsamiq

4.3.1.1 Údržbář

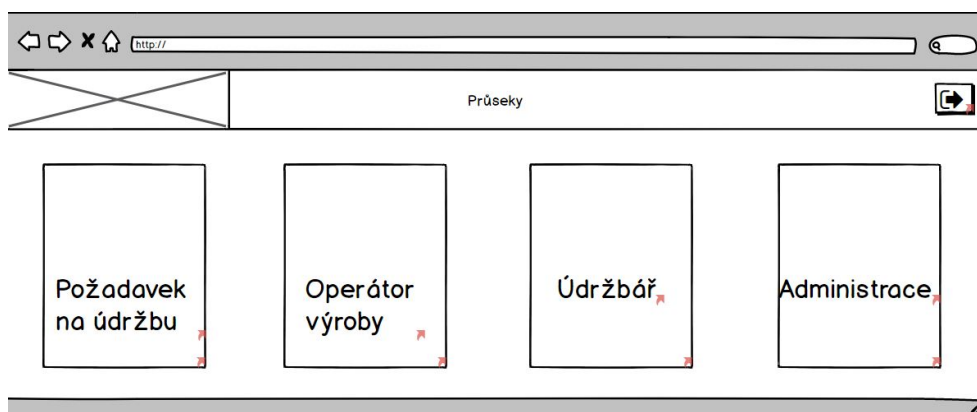


Obrázek 4.2: Diagram případu užití pro správu poruch



Obrázek 4.3: Diagram případu užití pro správu poruch

4.3. Lo-Fi prototyp



Obrázek 4.4: Diagram případu užití pro správu poruch

The diagram shows a web application interface for managing faults. The interface consists of a header bar with a title 'Seznam hlášení' and a search icon. Below the header, there are two tabs: 'Údržba' and 'Prevence'. The 'Údržba' tab is selected, showing a table of faults. The table has columns for 'Popis hlášení', 'Technické místo', 'Vyboření', 'Začátek poruchy', 'Stav', 'Založil', 'Operace', 'Práce', 'Akce', and 'Přílohy'. The table contains 9 rows of data.

Popis hlášení	Technické místo	Vyboření	Začátek poruchy	Stav	Založil	Operace	Práce	Akce	Přílohy
Neodtéká voda	Kanál KZ 4		15.2018 14:28	Rozpracováno	Morávek		Začít		Příloha
Neodtéká voda	Kanál KZ 4		15.2018 14:28	Rozpracováno	Morávek	Zrušit	Ukončit		Příloha
Neodtéká voda	Kanál KZ 4		15.2018 14:28	Rozpracováno	Horák	Zrušit	Začít	Zobrazit akce	Příloha
Neodtéká voda	Kanál KZ 4		15.2018 14:28	Rozpracováno	Morávek	Zrušit	Začít		Příloha
Neodtéká voda	Kanál KZ 4		15.2018 14:28	Nahlášeno	Novák	Zrušit	Začít		Příloha
Neodtéká voda	Kanál KZ 4		15.2018 14:28	Rozpracováno	Morávek		Začít	Zobrazit akce	Příloha
Neodtéká voda	Kanál KZ 4		15.2018 14:28	Rozpracováno	Morávek		Začít		Příloha
Neodtéká voda	Kanál KZ 4		15.2018 14:28	Rozpracováno	Morávek		Začít	Zobrazit akce	Příloha
Neodtéká voda	Kanál KZ 4		15.2018 14:28	Nahlášeno	Morávek	Zrušit	Ukončit		Příloha

Obrázek 4.5: Diagram případu užití pro správu poruch

4. NÁVRH UŽIVATELSKÉHO ROZHRAŇÍ

The screenshot shows a web application for managing faults. The browser window has a URL bar with 'http://'. The main content area is titled 'Založení poruchy' (Create fault). It contains several input fields and dropdown menus for creating a new fault record.

Poruchy (Faults) Table:

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

Založení poruchy Form:

Vybavení:

Priorita:

Plánovací skupina:

Pracoviště:

Popis požadavku:

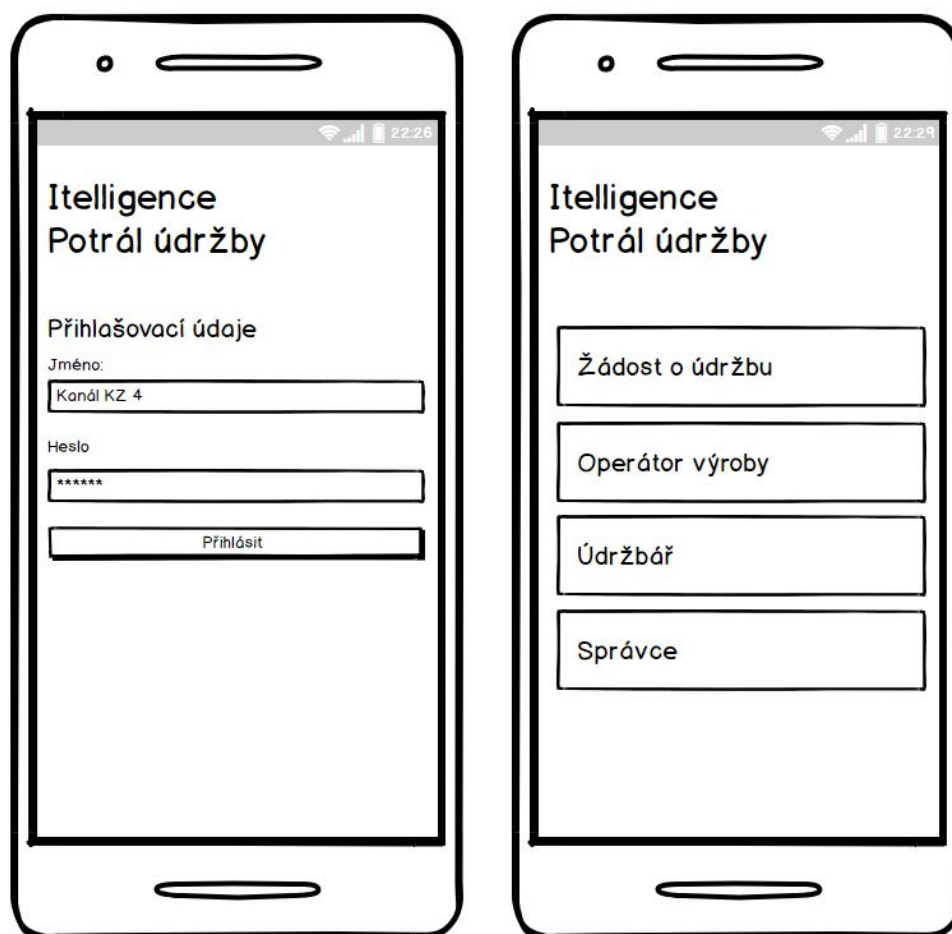
Příloha:

Údržba / Prevence Table:

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

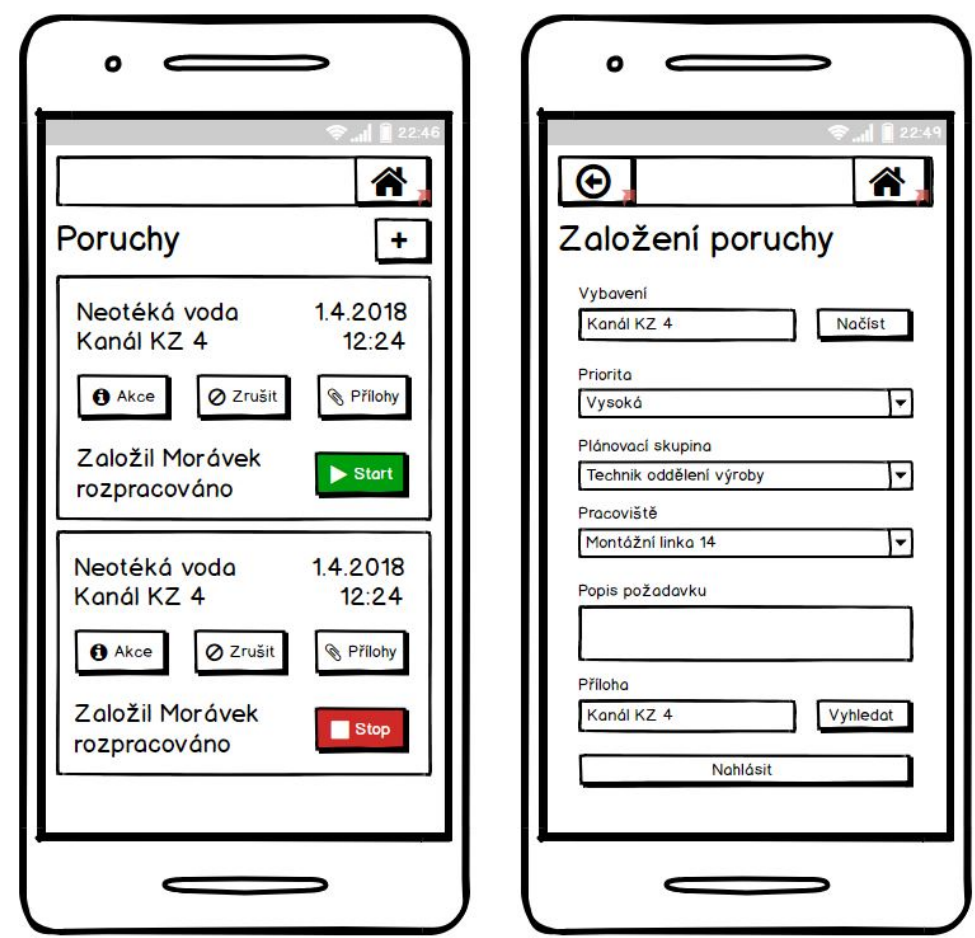
Obrázek 4.6: Diagram případu užití pro správu poruch

Desktopová verze



Obrázek 4.7: Diagram případu užití pro správu poruch

4. NÁVRH UŽIVATELSKÉHO ROZHRAŇÍ



Obrázek 4.8: Diagram případu užití pro správu poruch

4.3.2 Built


Seznam hlášení 

 Údržba  Prevence  Přidat

Poruchy									
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	 Start	 Akce	
Neodtéká voda	Kanál KZ 4	Text	1.5.2018 13:45	Rozpracováno	Novák	Text	 Stop	 Akce	

Obrázek 4.9: Diagram případu užití pro správu poruch

Založení poruchy

Vybavení:	<input type="text"/>
Priorita:	<input type="text"/>
Plánovací skupina:	<input type="text"/>
Pracoviště:	<input type="text"/>
Popis požadavku:	<input type="text"/>
Příloha:	<input type="text"/>
	 Vyhledat
<input type="button" value="Nahlásit"/> <input type="button" value="Zrušit"/>	

Obrázek 4.10: Diagram případu užití pro správu poruch

Desktopová verze

4. NÁVRH UŽIVATELSKÉHO ROZHRANÍ

Poruchy

+ Přidat

Popis poruchy

Neodtéká voda

Technické místo: Kanál KZ 4

Vybavení: Text


Začátek poruchy: 1.5.2018 14:28

Stav: Nahlášeo


Založil: Morávek

Operace: Text


Práce:

 Start

Akce:

 Akce

Přílohy:



Obrázek 4.11: Diagram případu užití pro správu poruch

Založení poruchy

Vybavení:

Priorita:

Plánovací skupina:

Pracoviště:

Popis požadavku:

Příloha:

 Vyhledat

Nahlásit

Zrušit

Obrázek 4.12: Diagram případu užití pro správu poruch

Mobilní verze

4.3.3 Heuristická analýza

Při návrhu uživatelského rozhraní je dobré držet se deseti následujících pravidel z Nielsenovi heuristické analýzy. V této kapitole je čerpáno ze zdrojů [8] a [9]. Nielsenova heuristická analýza je jednou ze základních metod pro testování uživatelského rozhraní. Jedná se o seznam pravidel, které by mělo uživatelské rozhraní 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.

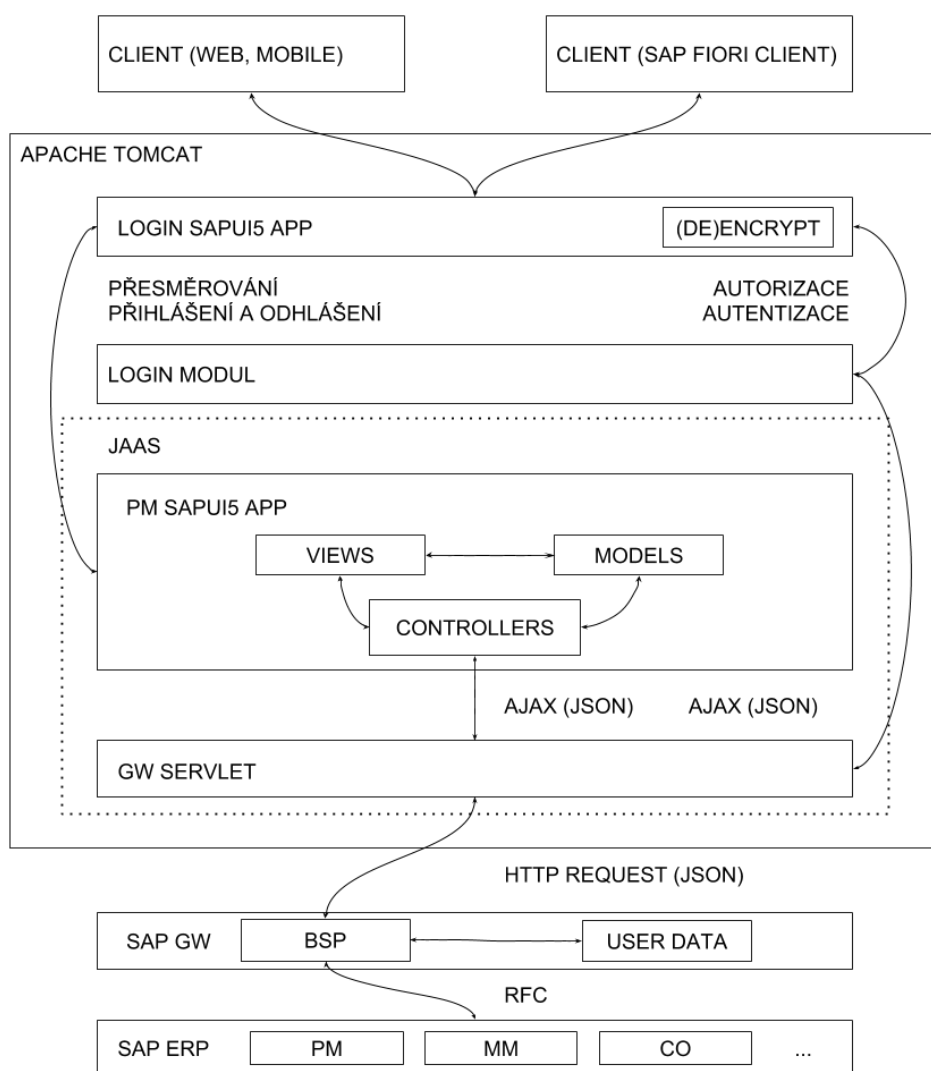
1. **Viditelnost stavu systému** - Uživatel by měl být vždy systémem vhodně informován (v rozumném čase) o tom co se zrovna děje. Systém by tak měl reagovat na uživatelský vstup, nebo v případě, že se například provádí nějaký časově náročnější výpočet nebo stahování dat, tak zobrazit progress bar.
2. **Propojení systému a reálného světa** - Systém by měl na uživatele mluvit jazykem uživatele se slovy, frázemi a koncepty, které jsou uživateli známé, zná je z reálného světa. Neměly by se využívat pojmy, které jsou například specifické pouze pro daný systém.
3. **Uživatelská kontrola a svoboda** - Uživatelé se často učí nové funkce systému pomocí chyb, které provedou. Když uživatelé udělají chybu, musí mít možnost provedenou akci vrátit zpět a vrátit tak systém do předchozího stavu. V případě, že se provádí nenávratná akce, je třeba na to uživatele řádně upozornit.
4. **Standardizace a konzistence** - Uživatel nesmí být zmaten různými termíny v různých situacích, přestože mají dané termíny stejný význam. Systém by měl vždy dodržovat standardy, které jsou na dané platformě. Proto je například potřeba dodržovat standardní chybové hlášky, správné umístění komponent apod. Ideální je používat standardní platformové komponenty.
5. **Prevence chyb** - Lepší než dobré chybové hlášky je návrh, který zabráňuje samotnému výskytu chyb. Buď je možné podmínky náchylné k chybám co nejvíce eliminovat, nebo na chyby uživatele upozornit ještě dříve než například potvrdí formulář.
6. **Rozpoznání namísto vzpomínání** - Je třeba minimalizovat zatěžování paměti uživatele tím, že uživatel vždy vidí potřebné informace a akce, které může provést. Uživatel si tak například nemusí pamatovat informace z jedné části formuláře na další.

7. **Flexibilní a efektivní použití** - Systém by měl v závislosti na jeho možnostech a typu umožňovat dvě verze ovládání. Pro méně zkušené uživatele a pro zkušené uživatele. Verze pro méně zkušené uživatele by měla obsahovat pouze základní funkce a možnosti nastavení tak, aby „nezkušený“ uživatel nebyl zbytečně zatěžován funkcionalitami, které stejně nepotřebuje. Naopak pro zkušené uživatele by se měli zobrazit všechny funkcionality, včetně těch složitějších. Zkušenější uživatel by měl mít také případně možnost si potřebné funkcionality přizpůsobit pomocí maker. Pro oba typy uživatelů je také dobré umožnit využívat klávesové zkratky
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 apod. Hlášky by měly přesně popisovat co je za problém a doporučit uživateli jak pokračovat dál. Ideální je, když uživatel nemá možnost dojít do stavu, kdy je potřeba chybové hlášení.
10. **Nápověda a návody** - Přestože je lepší, když je systém použitelný bez jakékoliv nápovědy, nápovědu by měl systém obsahovat. Veškeré informace by v systému měly být snadno vyhledatelné a obsahem nápovědy by měly být názorné příklady.

4.4 Porovnání prototypovacích nástrojů

KAPITOLA **5**

Implementace



Obrázek 5.1: Diagram případu užití pro správu poruch

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. Architektura je rozdělena do tří bloků a tomu odpovídají i tři následující podkapitoly.

5.1 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 věnující se teoretickému základu řečeno, základem veškerých potřebných procesů 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 vyvolání z jiných systémů. Ke vzdálenému volání slouží v rámci SAP systému 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 5.2 níže.

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 5.2: Seznam funkčních modulů v ERP systému

5.2 SAP GW

Jak již bylo zmíněno v kapitole věnující se tomuto produktu více teoreticky, primárním účelem serveru je komunikace s okolním světem. Standardní cesta pro Fiori aplikace je používáním servisů ODATA. Ne všichni zákazníci ovšem mají nakoupené licence pro provoz Fiori touto cestou a právě proto byla navržena architektura založená na technologii BSP, běžně dostupné v produktech GW, ale i ERP. Na obrázku 5.3 je seznam 36 BSP stránek, které pro aplikaci vznikly. V porovnání s počtem funkčních modulů na straně ERP je evidentní,

5. IMPLEMENTACE

že nejsou vytvořeny v poměru 1:1. Je to především z toho důvodu, že jsou v této úrovni uloženy uživatelská data portálových uživatelů.

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 5.3: Seznam funkčních modulů v ERP systému

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 6 níže.

Algorithm 1 Vytvoření instance třídy PortList

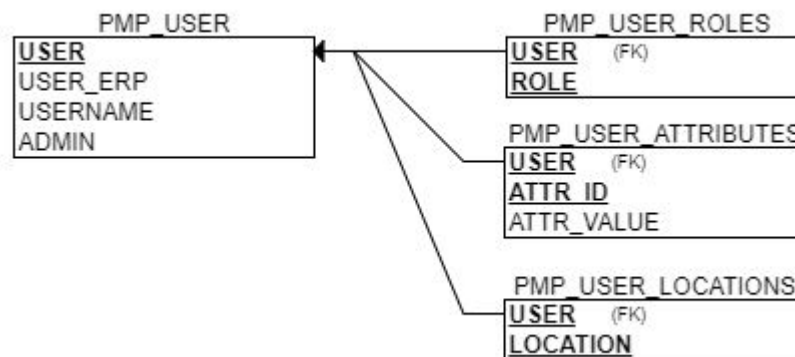
```

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 5.4 je k vidění ilustrační databázové schéma.



Obrázek 5.4: Ilustrační databázové schéma pro uživatelská data portálového uživatele

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 schematu 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.

5.3 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 tímž protokolom na ně odpovídat. Apache Tomcat zde slouží jako zásobník servletů starající se o jejich spouštění, běh, ukončování a podobně.

5.3.1 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).

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 5.4.1 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 aplikac. 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.

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.

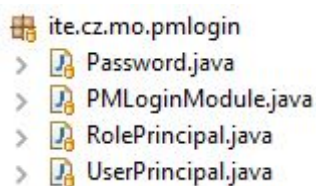
Algorithm 4 Definice typu autentizace FORM

```
<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 zadefinovány 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 5.5 zobrazené níže.



Obrázek 5.5: 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 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 níže.

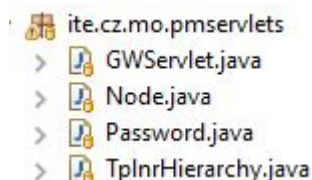
Algorithm 5 Konfigurační soubor JAAS

```
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.

5.3.2 GW Servlet

Na obrázku 5.6 je k vidění struktura paketu realizujícího mezivrstvu pro komunikace s SAP GW.



Obrázek 5.6: Struktura paketu realizujícího middleware vrstvu mezi PM SAPUI5 aplikací a SAP GW

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`.

Algorithm 6 Vytvoření instance třídy PortList

```
CALL FUNCTION 'ZITL_MOB_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.

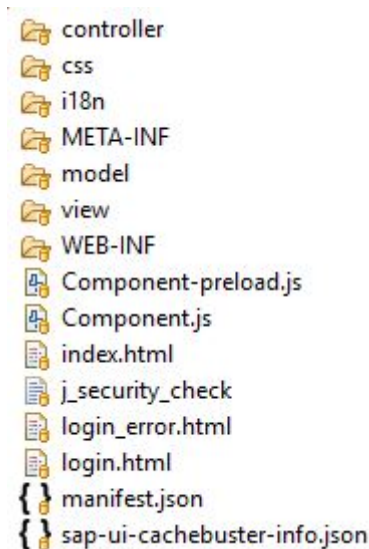
- **Node:** Třída reprezentující prvek hierarchie technických míst a vybavení z modulu SAP PM. Jelikož se v interně technické místo a vybavení datově liší, je vytvořena tato třída, která nesrovnalosti mezi těmito objekty eliminuje. Datově rozdílné identifikátory jsou nahrazeny jednotným id a ostatní lišící se parametry jsou sloučeny do atributů třídy odpovídajícím významu původního datového objektu. To posléze umožňuje v prezentační vrstvě lehce pracovat v
- **TplnrHierarchy:**
- **Password:**

5.4 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 zadané veškeré pohledy (stránky), se kterými se uživatel bude moci setkat.

5.4.1 Struktura aplikace

Zde je zobrazena a popsána celková struktura aplikace z vývojářského pohledu. Na obrázku 5.7 je k vidění hlavní strom adresářů a souborů v použitém Eclipse projektu.



Obrázek 5.7: 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 zařízení uživatele a podobně. V mém případě obsahuje tři následující soubory.
 - **models.js:** Slouží pouze k zdefinování modelu s informací o používaném zařízení. Poskytuje tak napříč zbytku 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 7 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 zadaná view. Tedy 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 zadefinování zdroje 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 datum, 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-timeovém 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

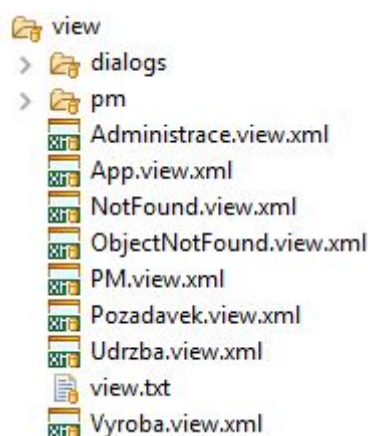
pro správný chod aplikace v rámci prostředí používaném uživatelem. V rámci hierarchie HTML je vytvořen element body, do kterého je vložena celá instance vytvořené aplikace odvozené od staženého jádra. Jedná se zpravidla o jediný kus čistého HTML při tvorbě aplikace ve frameworku SAPUI5.

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 funkcionality bohužel není implicitně ve frameworku zanesena, nicméně existují mechanismy, které obdobné chování umožňují. Celý proces spočívá v zadefinování ResourceServletu, který porovnává datum poslední modifikace souboru s tím, který má uživatel uložen v mezipaměti. Tento Servlet musí být zadefinován 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.

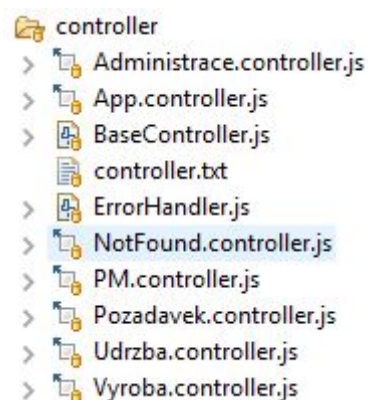
- **login.html**: V rámci JAAS logiky pro autorizaci a autentifikaci je definována implicitní HTML stránka, která je vyvolána v případě, že uživatel nemá doposud vytvořenou vůči aplikaci session. Z důvodu zachování konzistentního vzhledu aplikací je i pro přihlášení vytvořena SAPUI5 aplikace. Umístit 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.

5.4.2 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 5.8 a 5.9 níže.



Obrázek 5.8: Struktura jednotlivých view v Eclipse projektu



Obrázek 5.9: 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.

5.4.2.1 App

Není ani tak stránkou jako spíš základním view aplikace. V podstatě pouze všechny ostatní view obaluje a svůj obsah dynamicky střídá na základě uživatelských 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 8.

Algorithm 8 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 9 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 9 XML definice view App

```
BaseController.extend("sap.ui.mo.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 9 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 10 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 10 jsou posléze krátce popsány.

Algorithm 10 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 11 níže.

Algorithm 11 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 přerazení takového modelu k view pak může dojít k zobrazení požadované hodnoty. Jelikož svazování dat v modelu s view patří k velmi častým operacím a vyžaduje 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.

5.4.2.2 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 12 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 13 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 zdefinováno rozložení Grid kontejneru, které má implicitně responzivní chování. Vývojář tedy nemusí nijak extra řešit počet zobrazených agregovaných komponent, framework to zvládá sám.

Algorithm 14 XML definice zápatí úvodní stránky

```

</Page>

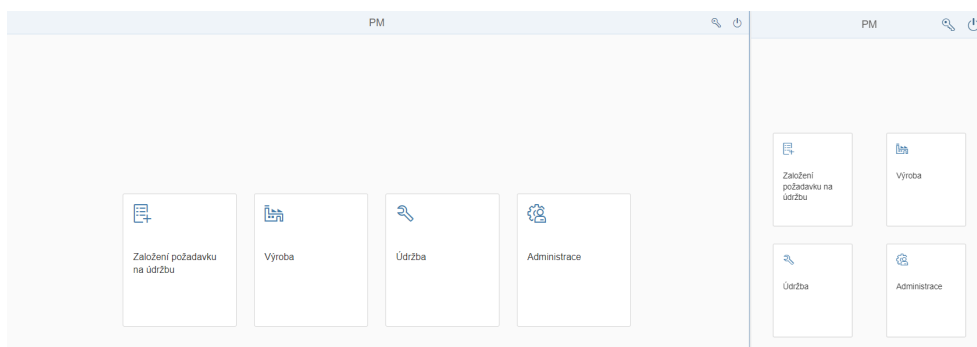
```

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 přidělení požadovaných dlaždic uživateli v závislosti na přidělených rolích. To spočívá v načtení uživatelských rolí a přidání dlaždic do Grid kontejneru zdefinovaného v ukázce kódu 13 výše.

Stránka Výsledná podoba stránky je zobrazena na obrázku 5.10 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 5.10: Úvodní stránka PM SAPUI5 aplikace

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

5.4.2.3 Pozadavek - Založení požadavku na údržbu

Stránka je navržena tak, aby odpovídala funkčnímu požadavku 3.3.1.1. 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 15 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 7 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 10 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 5.11 níže.

Obrázek 5.11: Stránka pro založení požadavku na údržbu

5.4.2.4 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 5.4.1` 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 odchyťování 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 16 Přiřazení posluchače ve formě funkce k hodnotě modelu

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

Jedná se o výtažek kódu v inicializační funkci `onInit`. Spočívá v načtení device modelu 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 17 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 5.12 níže.

Popis hlášení	Odp. prac.	Technické místo	Vyřazení	Začátek poruchy	Status zakázky	Založil	Operace	Akce	Přilož
17:34 Porucha / Moravek	PM_DUMMY	MOO-EXP Expedice		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		2018-02-27 09:15:32	UDR	MORMAITE			
porucha kurka mechanická	TUH_MECH	MOO-HKS-03-STR Strojní zařízení	RAPIDA 142 - DW1	2018-03-19 14:06:25	UDR	KURMAITE			
test		MOO-HKS-04-STR Strojní zařízení	RAPIDA 106 SRH	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					
Hýřk	TUE	MOO-HKS Výroba HKS		2018-03-12 09:55:19	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 5.12: Stránka pro operátora údržby

5.4.2.5 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 5.4.2.4, 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 5.4.1.

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

Controller Kromě standardních činností, jako je odchyťování 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.

Stránka Stránka je řešena trochu odlišně než tomu je u operátory výroby 5.4.2.4. 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 5.12 níže.

The screenshot shows two views of the SAPUI5 application. The left view displays a table of open faults, and the right view shows a detailed view of a specific fault.

Technické místo	Vybavení	Popis poruchy	Číslo zakázky	Odp. prac.	Začátek poruchy	Založil	Start Stop	Operace	Akce	Přiložky
MOO-HKS-03-STR Strojní zařízení	RAPIDA 142 H	porucha 16.2	0000700000 61	TUH_ELE	2018-02-16 10:56:05	KURMAITE				
MOO-HKS Výroba HKS		Chyba na HKS	0000700000 77	TUV_MECH	2018-02-27 09:15:32	MORMAITE				
MOO-HKS Výroba HKS		Rozbitý sklo	0000700000 82	TUV_MECH	2018-03-04 15:59:13	MORMAITE				
MOO-HKS Výroba HKS	RAPIDA 142 - DW1	Rozbitý sklo	0000700000 81	TUV_MECH	2018-03-04 15:59:48	MORMAITE				
MOO-HKS Výroba HKS	RAPIDA 106 SRH		0000700001 08	TUE	2018-03-08 16:26:13	MORMAITE				
MOO-HKS Výroba HKS		Hýřk	0000700001 01	TUE	2018-03-12 09:56:19	MORMAITE				
MOO-HKS Výroba HKS			0000700001 12	TUE	2018-03-13 10:55:15	MORMAITE				

The right view shows a detailed view of a fault. It includes the following information:

- porucha 16.2**
- Tech. místo:** MOO-HKS-03-STR Strojní zařízení
- Vybavení:** RAPIDA 142 H
- Založil:** KURMAITE TUV_ELE
- Start Stop:** [Start] [Stop]
- Operace:** [Operace]
- Akce a přílohy:** [Akce a přílohy]
- Začátek poruchy:** 2018-02-16 10:56:05
- Číslo zakázky:** 000070000061
- Chyba na HKS**
- Tech. místo:** MOO-HKS Výroba HKS
- Vybavení:** RAPIDA 106 SRH
- Založil:** MORMAITE TUV_MECH
- Start Stop:** [Start] [Stop]
- Operace:** [Operace]
- Akce a přílohy:** [Akce a přílohy]
- Začátek poruchy:** 2018-02-27 09:15:32
- Číslo zakázky:** 000070000077

Obrázek 5.13: Stránka pro údržbáře

5.4.2.6 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 18 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 18

```
<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 5.4.2.2.

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 desktopzařízení je vidět na obrázku 5.14 níže.

Seznam uživatelů	Administrace
ADMINITE >	Základní údaje
BALMAMOO >	Osobní číslo: 10100716 Upravit os. číslo
BHB >	Jméno uživatele: Marcel Morávek II. Upravit uživ. jméno
DVOTOMOO >	Pracoviště Role <u>Atributy</u>
KUBDAMOO >	Seznam atributů + -
KURMAITE >	<input type="checkbox"/> Klíč Hodnota
MORMAITE >	<input type="checkbox"/> CONF_LIMIT 0
MUHOTMOO >	<input type="checkbox"/> GEWRK TUE
NEADMIN >	<input type="checkbox"/> GEWRK_ELE TUH_ELE
R142HMOO >	<input type="checkbox"/> GEWRK_MECH TUV_MECH
R142_MOO >	
R74_MOO >	
RADIVMOO >	

Obrázek 5.14: Stránka pro správu uživatelů

Stránka pro mobilní verzi je z důvodu rozdělení tentokrát zobrazena samostatně na obrázku 5.15 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 >	Základní údaje
BALMAMOO >	Osobní číslo: 10100716 Upravit os. číslo
BHB >	Jméno uživatele: Marcel Morávek II. Upravit uživ. jméno
DVOTOMOO >	Pracoviště Role <u>Atributy</u>
KUBDAMOO >	Seznam atributů + -
KURMAITE >	<input type="checkbox"/> Klíč Hodnota
MORMAITE >	<input type="checkbox"/> CONF_LIMIT 0
MUHOTMOO >	<input type="checkbox"/> GEWRK TUE
NEADMIN >	<input type="checkbox"/> GEWRK_ELE TUH_ELE
R142HMOO >	
R142_MOO >	
R74_MOO >	
+ Zakočit 🔍 Změna hesla - Odebrat	

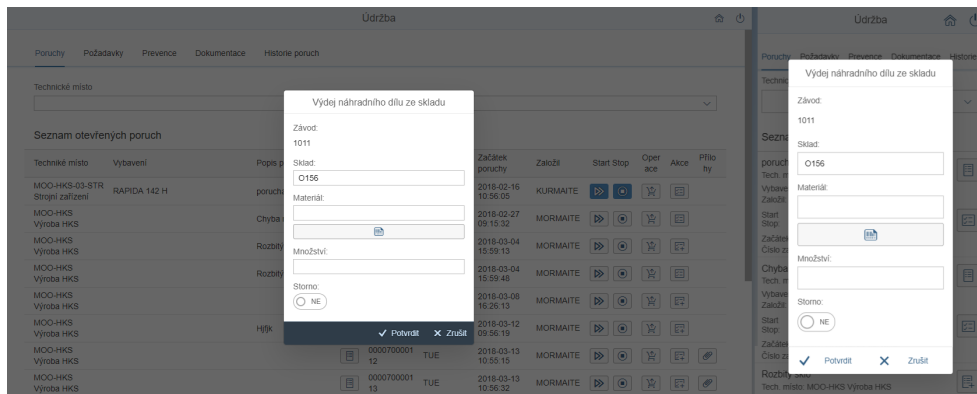
Obrázek 5.15: Stránka pro správu uživatelů - mobilní verze

5.4.2.7 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

5. IMPLEMENTACE

dialogy trochu komplikovanější. Zpravidla se jedná o dialogy vyžadující od uživatele vyplnění nějakých dat potřebných pro provedení daného úkonu. Na obrázku 5.16 níže je například k vidění dialog pro vydání náhradního dílu ze skladu jak pro desktop tak mobilní zařízení.



Obrázek 5.16: Dialog pro vydání náhradního dílu

5.5 LOGIN SAPUI5 Aplikace

5.5.1 Struktura aplikace

Struktura aplikace je víceméně totožná s aplikací PM popsané v kapitole 5.4.1 výše. Jediné, v čem se aplikace zásadně liší je její zabezpečení popsané v souboru web.xml. V kapitole 5.3.1 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.

5.5.2 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 15.

Controller Úkolem je především komunikace s PMLoginModulem pro zajištění autentizace uživatele. Vypořádává 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

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

The image shows two identical login forms side-by-side. Each form has a header bar labeled 'Testovací provoz.'. Below this is a section titled 'Přihlašovací údaje'. Inside this section, there are two input fields: one for 'Jméno:' (Username) and one for 'Heslo:' (Password). Below the password field is a button labeled 'Přihlásit' (Login).

Obrázek 5.17: Stránka pro přihlášení uživatele

Stránka

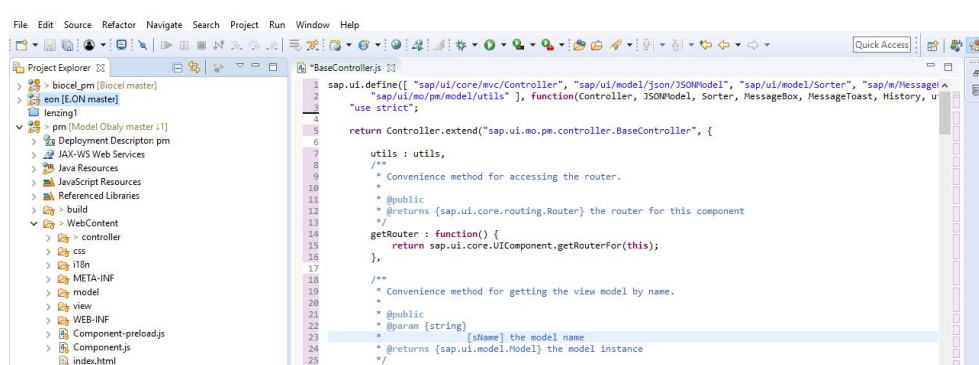
5.6 Porovnání vývojových prostředí

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.

5.6.1 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 knihovnamí 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 5.18 níže.

5. IMPLEMENTACE



Obrázek 5.18: Podoba Eclipse

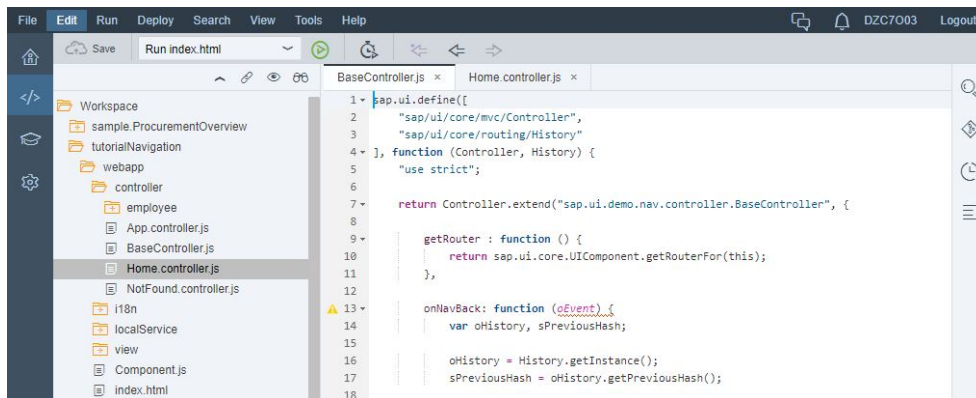
Prostředí Vzhled a pořadí oken se dá v Eclipse libovolně měnit, nicméně v základní verzi je v levé části seznam otevřených projektů, vedle něhož je prostor pro editaci jednotlivých souborů projektu. Ve spodní části je konzole a servery dostupné pro online testování. Je možné si zde nastavit Tomcat server, 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.

5.6.2 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.

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.

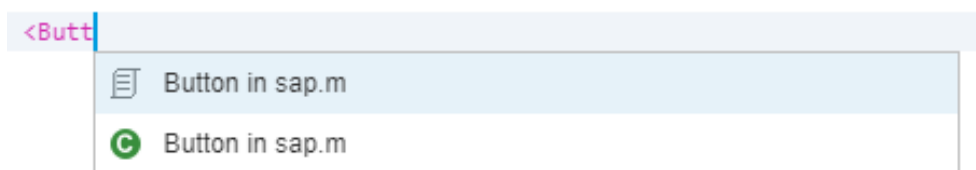
5.6. Porovnání vývojových prostředí



Obrázek 5.19: 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.

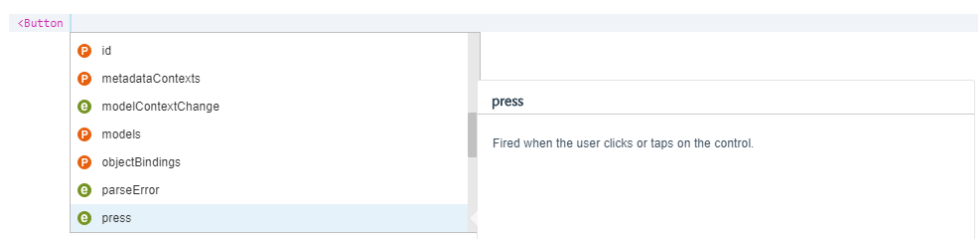
Pro ukázkou napovídání poslouží obrázky 5.20 a 5.21 zobrazené níže. První z nich ukazuje možnost definice tlačítka dvěma způsoby ještě před tím, než je dopsána plný název komponenty Button. První zobrazená varianta vytvoří v XML strukturu obsahující všechny atributy komponenty včetně všech možných agregací. Druhá varianta doplní pouze název komponenty.



Obrázek 5.20: 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 5.21 níže.

5. IMPLEMENTACE



Obrázek 5.21: Nápověda při definování atributů elementu v SAP Web IDE

5.6.3 Shrnutí

K porovnání obou přístupů poslouží tabulka 5.1. Obsahuje podstatné vlastnosti zmíněné napříč sekcemi 5.6.1 a 5.6.2.

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	-	+

Tabulka 5.1: 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 cloudovém řešení mnohem lepší.

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.

5.7 Testování

5.7.1 Middleware

Pro ukázkou testovací na úrovni middleware jsem zvolil test třídy `GWServlet` představující prostředníka v komunikaci z aplikace PM na SAP GW. Vnásledující ukázce kódu 19 je k vidění část rozhraní testovací třídy.

Algorithm 19 Rozhraní testovací třídy pro Unit testování

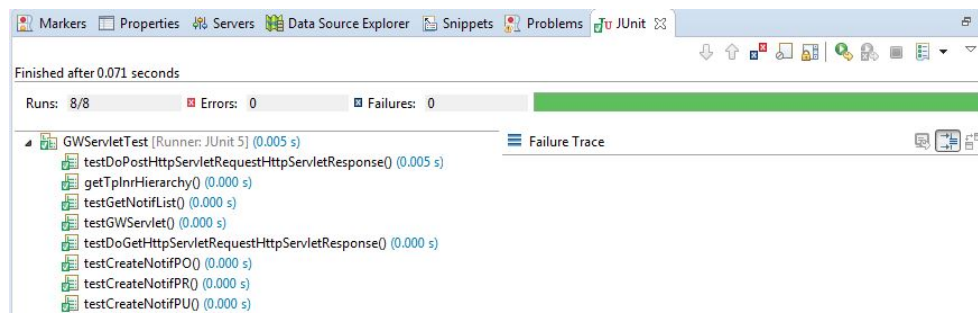
```
public class GWServletTest {  
    static GWServlet instance;  
    public GWServletTest();  
  
    public static void setUpClass();  
    public static void tearDownClass();  
    public void setUp();  
    public void tearDown();  
  
    public void testGetNotifList() throws Exception;  
    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.

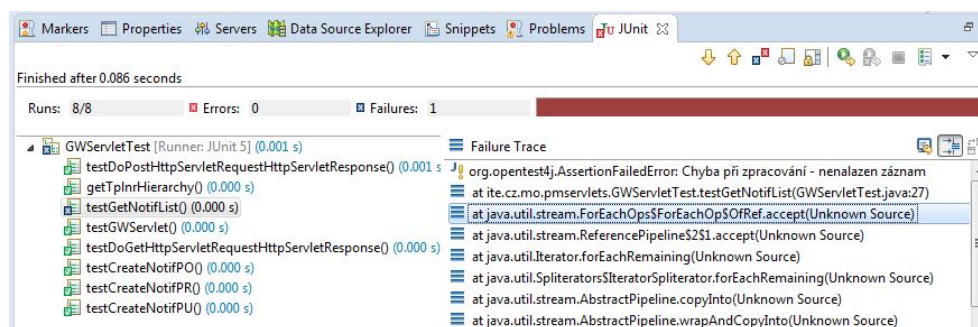
Metody `setUp` a `tearDown` zajišťují nezávislost testů, jsou totiž volány mezi jednotlivými testy a jde čas pro resetování inicializování instance nebo jiných aktivit nutných pro správnost testů.

Na následujících dvou obrázcích 5.22 a 5.23 jsou ukázky výstupu provedených jednotkových testů v prostředí Eclipse.

5. IMPLEMENTACE



Obrázek 5.22: Návod při definování atributů elementu v SAP Web IDE



Obrázek 5.23: Návod při definování atributů elementu v SAP Web IDE

5.7.2 Frontend

5.8 Doporučení pro vývoj

Závěr

Literatura

- [1] SAP: *SAP Company History [online]*. [cit. 2016-24-24]. Dostupné z: <https://www.sap.com/corporate/en/company/history.html>

Seznam použitých zkratk

GUI Graphical user interface

XML Extensible markup language

Obsah přiloženého CD

	readme.txt.....	stručný popis obsahu CD
	exe	adresář se spustitelnou formou implementace
	src	
	impl.....	zdrojové kódy implementace
	thesis	zdrojová forma práce ve formátu L ^A T _E X
	text	text práce
	thesis.pdf	text práce ve formátu PDF
	thesis.ps	text práce ve formátu PS