VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATION TECHNOLOGY

Ústav informačních systémů

DEPARTMENT OF INFORMATION SYSTEMS

Aplikace pro eskalaci odchylek z výrobních linek

Application for Escalating Information from Production Lines

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Marek Skalník**

AUTHOR

VEDOUCÍ PRÁCE — **doc. RNDr. Jitka Kreslíková, CSc.**

SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Skalník Marek, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Aplikace pro eskalaci odchylek z výrobních linek**
Application for Escalating Information from Production Lines
Kategorie: Softwarové inženýrství

Zadání:

1. Seznamte se s aktuálními systémy pro vývoj aplikací pro mobilní zařízení. Zaměřte se na mobilní platformu iOS.
2. Seznamte se s technologiemi na získávání informací z výrobních linek technologií SAP ME a SAP MII ve firemním prostředí.
3. Analyzujte a specifikujte požadavky na mobilní aplikaci, která by v reálném čase poskytovala přehled KPI ukazatelů na jednotlivých linkách a současně by, v případě poklesu jakéhokoliv definovaného KPI, vytvářela upozornění na určitou událost na mobilním zařízení, aby vlastníci jednotlivých ukazatelů měli bezprostřední informaci o jakémkoliv nesouladu.
4. Na základě provedené analýzy a po dohodě s konzultantem firmy navrhněte architekturu systému pro operační systém iOS.
5. Zvolte vhodné vývojové prostředí a navržený systém implementujte.
6. Použitelnost vytvořené aplikace otestujte v reálném prostředí a demonstrujte na vhodně zvoleném vzorku dat vybraném po dohodě s vedoucí.
7. Zhodnoťte dosažené výsledky a navrhněte možné rozšíření projektu.

Literatura:

- KEUR, Christian a Aaron HILLEGASS. *IOS programming: the Big Nerd Ranch guide*. Sixth edition. Atlanta, GA: Big Nerd Ranch, 2016. ISBN 978-0134682334.
- PHILLIPS, Bill, Chris STEWART a Kristin MARSICANO. *Android programming: the Big Nerd Ranch guide*. 3rd edition. Atlanta, GA: Big Nerd Ranch, 2017. Big Nerd Ranch guides. ISBN 978-0-13-470605-4.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů zadání 1 až 4.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kreslíková Jitka, doc. RNDr., CSc.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 30. října 2018

Abstrakt

Tato práce se zabývá problematikou vývoje mobilních aplikací, aktuálními mobilními operačními systémy a zaměřuje se na programování v prostředí iOS. Dále popisuje hlavní funkci ERP systému SAP, systému SAP ME a jejich vzájemné propojení pomocí SAP MII. Také zmiňuje možnost komunikace se systémem SAP MII pomocí webových služeb. Nakonec se zde nachází analýza požadavků pro mobilní aplikaci její implementace a testování, která bude získávat a zobrazovat KPI ukazatele z výrobních linek, a v případě poklesu vytvářet notifikace pro zvolené osoby.

Abstract

The thesis deals with the development of mobile applications, current mobile operating systems and focuses on iOS programming. It also describes the main function of SAP ERP system, SAP ME system and their integration with SAP MII. It also mentions the way to communicate with SAP MII using web services. Finally, there is an analysis of mobile application requirements, implementation and testing. That application will collect and display KPI from work centers and, in the event of a decrease of any KPI, generate notifications for selected people.

Klíčová slova

iOS, Výrobní informační systém, MES, mobilní aplikace, SAP ME, vývoj

Keywords

iOS, Manufacturing Execution system, MES, mobile applications, SAP ME, development

Citace

SKALNÍK, Marek. *Aplikace pro eskalaci odchylek z výrobních linek*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Jitka Kreslíková, CSc.

Aplikace pro eskalaci odchylek z výrobních linek

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. RNDr. Jitky Kreslíkové, CSc. a s odbornou konzultací Ing. Lukáše Říhy. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Marek Skalník
22.května 2019

Poděkování

Rád bych na tomto místě poděkoval především vedoucí mé diplomové práce, doc. RNDr. Jitce Kreslíkové, Csc. za odborné vedení, cenné rady a vstřícný přístup. Dále bych rád poděkoval kolegovi Ing. Lukáši Říhovi za rady při konzultacích mé práce. Také bych rád poděkoval své rodině a přátelům za pomoc a podporu během tvorby této práce a Ing. Kateřině Sedláčkové za pomoc s jazykovou úpravou práce.

Obsah

Kapitola 1

Úvod

Firma Resideo vznikla koncem roku 2018 oddělením od společnosti Honeywell, což je americká technologická firma zabývající se mnoha odvětvími, jakou jsou letectví, dopravní systémy, automatizace domácnosti a další. Část firmy Honeywell Home, která se zabývá automatizací domácnosti, byla od Honeywellu oddělena pod názvem Resideo. Nově vzniklá firma Resideo výrobní závod Brno-Slatina poskytuje svým zákazníkům produkty v oblasti spalování, elektroniky, termoregulace a systémů s pitnou vodou. Výrobní závod je rozdělen do tří divizí dle jejich zaměření na termoregulace, mechanickou výrobu a elektronickou výrobu.

Ve firmě je zaváděn nový výrobní systém, který má pokrýt kompletní výrobní proces a stát se zde standardem. Systém je postaven na platformě SAP ME, což je výrobní informační systém, mezi jehož základní funkcionality patří kopírování výrobních příkazů do výrobních zařízení, hlídání správné sekvence výrobních kroků, měření výkonu výrobní linky nebo sběr dat z výrobního procesu, apod. Mezi standardní součásti tohoto systému patří také modul pro eskalační proces. Ten umožňuje informovat vybrané osoby o odchylce od ideálního stavu linky v oblastech dostupnosti linky, plánovaného výstupu produktů z linky nebo limitů kvality výsledných produktů. Tento modul ovšem umožňuje eskalovat pouze pomocí emailu. Takovýto druh komunikace není pro tento případ vhodný, neboť email se může ztratit mezi desítkami ostatních.

Z tohoto důvodu je vhodné vytvořit mobilní aplikaci v prostředí iOS, která by o takovýchto odchylkách z výrobních linek informovala odpovědného uživatele. Aplikace bude poskytovat přehled KPI ukazatelů na jednotlivých linkách. KPI (neboli *Key Performance Indicators*) jsou klíčové ukazatele, podle kterých lze hodnotit výkonnost zařízení, v tomto případě výrobních linek. Uživatelé se budou moci přihlásit k odběru informací z ním zvolených linek, a v případě poklesu jakéhokoli definovaného KPI bude aplikace vytvářet notifikaci na telefonu. Díky tomu budou mít uživatelé okamžitou informaci o jakémkoli nesouladu na výrobních linkách.

Tato práce se zabývá popisem technologie SAP ME a jejím použitím ve výrobním závodě. Je zde uvedena možnost a způsob programování v systému SAP ME, možnost zpracování dat uložených u jednotlivých výrobních linek, ukládání uživatelsky definovaných dat v tomto systému a také možnost, jak tato data zpřístupnit jiným aplikacím skrze webové služby. Dále se tato práce zabývá problematikou programování mobilních aplikací se zaměřením na mobilní operační systém iOS. Cílem práce je krom shrnutí výše zmíněné problematiky především specifikace, návrh a samotné vytvoření mobilní aplikace pro zobrazování KPI ukazatelů v prostředí iOS a následně i její testování.

Kapitola 2

Vývoj mobilních aplikací

Pro mobilní telefony lze v zásadě vytvářet nativní aplikace nebo webové stránky, přístupné skrz webový prohlížeč. Rozdílů mezi nativní aplikací a webovou stránkou je mnoho. Webové aplikace jsou multiplatformní, zatímco nativní mobilní aplikace jsou určeny vždy pro konkrétní operační systém. Podle statistiky z roku 2017 uživatelé strávili více než šest krát více času používáním mobilních aplikací než návštěvami webů[1].

Při vývoji mobilních aplikací je potřeba brát v potaz různě velké displeje a jejich různá rozlišení, hardwarové specifikace telefonů a také jejich platformu. Co se týká výběru operačního systému, měly v roce 2018 mobilní operační systémy Android nebo iOS více než 99% podíl[2]. Ve společnosti Resideo se jako firemní telefony nejvíce používají telefony iPhone, a proto aplikaci v této práci budu tvořit právě pro operační systém iOS.

Mobilní aplikace lze rozdělit na několik druhů: nativní aplikace, hybridní, univerzální, nebo progresivní webové aplikace.

2.1.1 Webové aplikace

Webové aplikace jsou psány standardizovanými technologiemi popsanými mezinárodním konsorciem W3C[3], převážně HTML, CSS a Javascript. Jsou přístupné z aplikačního serveru z internetu a o jejich zobrazení se stará internetový prohlížeč v mobilním telefonu.

Webové aplikace jsou nezávislé na platformě, což sebou nese výhodu jednodušší aktualizace a distribuce než u nativních aplikací. Na druhou stranu je zde menší možnost spolupráce s hardwarem mobilních telefonů. K webovým aplikacím lze přistupovat skrze mobilní i desktopové systémy, tudíž je potřeba brát v úvahu různé velikosti obrazovek, od malých telefonů

[1] Zdroj: https://www.comscore.com/Insights/Presentations-and-Whitepapers/2017/The-2017-US-Mobile-App-Report (citováno 15.prosince 2018)

[2] Zdroj: http://gs.statcounter.com/os-market-share/mobile/worldwide/ (citováno 15.prosince 2018)

[3] Webová adresa: https://www.w3.org/ (citováno 15.prosince 2018)

až po velké počítačové monitory, a různé druhy interakce s aplikací, jako je ovládání myší nebo pomocí dotykové obrazovky [2].

2.1.2 Nativní aplikace

Nativní aplikace jsou určené jen pro jednu konkrétní platformu a je potřeba je napsat v podporovaném prostředí. U iOS je nutno využít programovací jazyky Objective-C nebo Swift, na Androidu je to Java nebo jazyk Kotlin [2].

Aplikace v telefonu mohou uživateli zasílat notifikace. Dále lze různé aplikace v telefonu propojovat mezi sebou. Jejich výhoda je také úzká spolupráce s hardwarem telefonu a možnost využít například fotoaparát nebo GPS. Také je lze naprogramovat tak, aby byly spuštěné na pozadí, nebo aby šetřily baterii telefonu [2].

2.1.3 Hybridní aplikace

Hybridní aplikace představují kombinaci webových a mobilních aplikací. Jedná se o mobilní aplikaci, která slouží jako jakýsi obal obsahující prvky, které vykreslují web. Takovýto přístup je jednodušší pro vývoj pro více platforem a je lépe udržovatelný[4], nicméně není vhodný pro aplikace náročnější na hardwarové prostředky [2].

2.1.4 Progresivní webové aplikace

Progresivní webové aplikace jsou webové aplikace, které za určitých podmínek mohou získat výhody nativních aplikací. Po načtení s nimi lze pracovat off-line, poté se při dalším zapnutí rychleji spouštějí a mají možnost zasílat uživateli notifikace. Programují se stejně jako webové aplikace. Výhody nativní aplikace získá pouze v případě, že je spuštěna v prostředí, které to umožňuje. V aktuální době jsou progresivní webové aplikace podporované pouze operačním systémem Android [5].

2.1.5 Univerzální mobilní aplikace

Univerzální aplikace jsou aplikace, ve kterých je kód psaný pro všechny platformy stejně. Využívá se zde framework, který tohle umožňuje. Vývoj potom probíhá v prostředí pro daný framework a napsaný kód aplikace je překládán zvlášť pro všechny podporované platformy [2].

Tento postup v dnešní době zpřístupňuje například framework od společnosti Facebook React Native[5].

[4] Lehčí udržovatelnost v závislosti na verzích operačního systému

[5] https://facebook.github.io/react-native/ (citováno 15.prosince 2018)

2.2 Model-view-controller

Při programování nejen mobilních aplikací může být vhodné dodržovat softwarovou architekturu *Model-view-controller* (MVC). Ta rozděluje aplikaci do tří vrstev a odděluje tak data od řídící logiky a uživatelského rozhraní [3].

- Vrstva modelu udržuje data aplikace. Je to doménový model aplikace udržující vztahy reálného světa. Vrstva modelu nemá žádnou vazbu na uživatelské rozhraní. Většinou instance modelu udržují data reprezentující skutečné objekty v reálném světě, operace nad nimi a jejich omezující podmínky.
- Vrstva pohledu (view) popisuje uživatelské rozhraní a vše, co vidí a čím může interagovat uživatel. Tato vrstva by měla obsahovat pouze minimum logiky, jako jsou kontroly uživatelských vstupů.
- Řídící vrstva (controller) se stará o řízení celé aplikace. Zahrnuje vnitřní logiku aplikace, propojuje data se zobrazením a stará se, aby zobrazovaná data byla vždy aktuální.

Modifikace některé z vrstev by měla mít co nejmenší vliv na vrstvy ostatní. Použití této nebo i jiné architektury má zásadní vliv na čitelnost, srozumitelnost a udržovatelnost kódu. Také díky této architektuře lze aplikaci tvořit ze znovupoužitelných komponent, nebo rozdělit vývoj aplikace na více samostatných celků [3].

2.3 Vývoj aplikací pro Android

Android je mobilní operační systém od společnosti Google. Je postaven na Linuxovém jádře a je dostupný jako otevřený software (open source). Android je navrhnutý primárně pro mobilní telefony s dotykovým displejem [4].

Od roku 2011 se jedná o nejrozšířenější operační systém v mobilních telefonech a v roce 2018 byl tento systém nainstalován na 77 % telefonů[6]. V době psaní této práce je nejnovější verze Androidu 9.0 Pie, nicméně rozšiřování nových verzí systému je relativně pomalé. V říjnu 2018 mělo přibližně 50 % telefonů stále verzi Android 6.0 Marshmallow a starší[7], a z tohoto důvodu je vhodné nevytvářet Android aplikace pouze pro poslední verzi operačního systému, ale zachovávat kompatibilitu i s jeho staršími verzemi.

[6] Zdroj: http://gs.statcounter.com/os-market-share/mobile/worldwide/ (citováno 15.prosince 2018)

[7] Zdroj: https://developer.android.com/about/dashboards/ (citováno 15.prosince 2018)

2.3.1 Programování aplikací pro Android

Jednou z možností vývoje aplikací pro operační systém Android je využití nástroje Android Studio, který je vydán přímo společností Google. Též je potřeba mít v počítači nainstalovaný Java Development Kit kvůli podpoře Javy. Je nutné znát programovací jazyk Java nebo Kotlin [1].

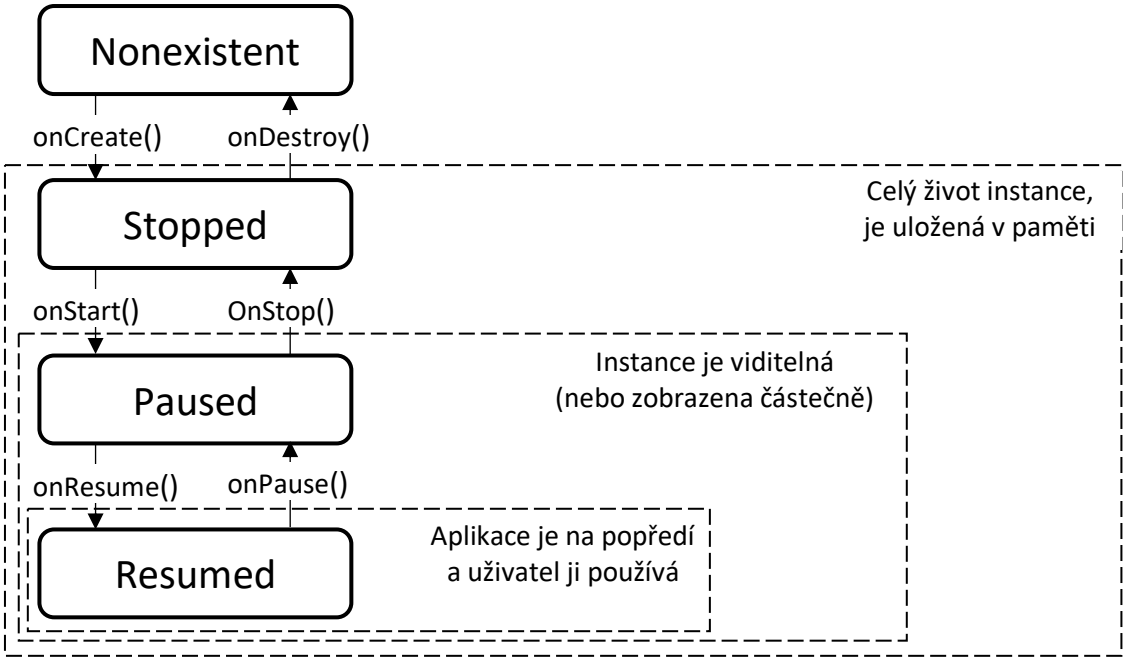
Programování uživatelského rozhraní probíhá pomocí zápisu do XML souboru, ve kterém se popisují prvky zobrazované na obrazovce. Chování uživatelského rozhraní a samotné aplikace se programuje v jazyce Java nebo Kotlin [1].

Kotlin je open source programovací jazyk vyvinutý v roce 2011 společností JetBrains. Tento jazyk lze používat v prostředí všude tam, kde je dostupná Java. Klade důraz na jednoduchost, rychlost překladu a bezpečnost, poskytuje systém zabraňující či omezující dereferenci *null* ukazatele za běhu aplikace. Tento jazyk se stal pro svoje výhody velmi populárním a v roce 2017 se Kotlin stal oficiálním jazykem pro vývoj pro operační systém Android [24].

Testování aplikací může probíhat buď v emulátoru, nebo přímo na připojeném mobilním telefonu. Při testování v emulátoru lze zvolit několik různých telefonů s rozdílnou úhlopříčkou displeje a jeho rozlišením. Dále se volí verze operačního systému Android na virtuálním telefonu. Při testování na skutečném telefonu lze vyzkoušet reálnou použitelnost aplikace na malé obrazovce, a také můžeme získat představu o hardwarové náročnosti naší aplikace [1].

2.3.2 Životní cyklus objektů

Objekty v jazyce Java v rámci aplikace mají svůj životní cyklus, během kterého se mohou nacházet ve čtyřech stavech: *nonexistent*, *stopped*, *paused* nebo *resumed*. Při každé změně stavu je volána patřičná metoda, viz Obrázek 1.



Obrázek 1: Stavy objektu v Java v aplikaci pro operační systém Android (inspirováno z [1])

Aplikace je ve stavu *resumed*, když se nachází na popředí a interaguje s ní uživatel. Do stavu *paused* se aplikace může dostat, pokud není označená na popředí. Například pokud telefon podporuje zobrazení více aplikací na obrazovce a uživatel aktuálně interaguje s jinou. Stav *stopped* znamená, že má aplikace uložená svoje data v paměti RAM, nicméně není zobrazená. Při vypnutí aplikace je volána metoda *onDestroy()*, aplikace přechází do stavu *nonexistent* a přestává existovat [1].

Při otočení telefonu se s telefonem zároveň i otáčí displej (tak aby byl stále vodorovný). Na to je při vývoji aplikací potřeba myslet. Při otočení displeje přestává celý objekt existovat (přechází do stavu *nonexistent*), a vytváří se znovu. To je z důvodu, že se mění mnoho nastavení, jako je rozlišení aktivní části displeje, klávesnice, vrchní lišta, atd. Proto je potřeba stav aplikace uložit a znovu načíst. Dále se u aplikací standardně využívají dva XML soubory s rozložením ovládacích prvků na obrazovce, zvlášť pro zobrazení displeje na výšku a zvlášť pro zobrazení na šířku [1].

2.4 Vývoj aplikací pro iOS

K vývoji aplikací pro operační systém iOS je potřeba IDE od společnosti Apple Xcode. To je dostupné pouze pro počítače Mac, tudíž je také potřeba vyvíjet pod macOS X. Pro programování iOS aplikace je používán programovací jazyk Swift, nicméně lze použít i starší Objective-C [3].

Vývojář, který chce vytvářet a distribuovat aplikace pro iOS, musí mít registrovaný vývojářský účet. Bez tohoto účtu je možné pouze aplikace vytvářet a testovat, nicméně takováto aplikace nainstalovaná bez vývojářského účtu na mobilní telefon přestává po sedmi dnech fungovat. Následná distribuce aplikací může probíhat několika způsoby. Aplikaci lze nahrát na App store. Jedná se o obchod s aplikacemi, který je přístupný pro celý svět, a odkud si je tedy může nainstalovat kdokoliv. Pro přidávání aplikací na App Store platí přísná pravidla, která je potřeba při publikování dodržet[8]. Další možností je Ad Hoc distribuce, kdy je aplikace instalována na telefon pomocí instalačního souboru, a může ji nainstalovat omezený počet uživatelů. Poslední, třetí možností je vývoj aplikace pro velké firmy. Zde je ovšem potřebný vývojářský účet pro firmy [6].

2.4.1 Swift

Swift je open source programovací jazyk vyvinutý a uvedený společností Apple v roce 2014 pro programování převážně v jejich systémech. Nahrazuje jazyk Objective-C, který sloužil pro programování mobilních aplikací v prostředí iOS dříve, a má se zbavovat některých jeho nevýhod. Z tohoto důvodu obsahuje jazyk Swift několik prvků, které zajišťují zpětnou kompatibilitu právě s jazykem Objective-C. Programovací jazyk Swift se stále vyvíjí a jeho nejnovější verze je aktuálně Swift 4 [3].

[8] https://developer.apple.com/app-store/review/guidelines/ (citováno 15.prosince 2018)

Při psaní kódu má být těžší udělat chybu, jazyk má být jednoduchý, intuitivní a velice rychlý. Jedna z jeho největších výhod je absence správy paměti, čehož je dosaženo mechanismem počítání referencí. Také není možné, aby v kódu nastala chyba typu dereference *null* ukazatele [3].

Proměnná nebo konstanta získává při inicializaci svoji počáteční hodnotu, a to buď explicitně, nebo automaticky. Proměnné v jazyce Swift jsou silně typované, a tak po jejich prvotní deklaraci nelze měnit jejich typ. Pro každý typ existuje *inicializátor*, který nastavuje počáteční hodnotu daného typu [3].

2.4.2 Uživatelské rozhraní

Storyboard je nejnovější metoda, jak tvořit uživatelské rozhraní v iOS. Jedná se o plátna, která plníme ovládacími prvky se statickými daty. Aplikace by měla být rozdělena na několik logických celků, každý zobrazen ve vlastním storyboard. Tento způsob návrhu cílí především na jednoduchost za cenu horší znovupoužitelnosti kódu. Výhodou je také nadhled nad celou aplikací, kdy je vidět, jak mezi sebou obrazovky vzájemně souvisí [3].

Další způsob je pomocí vlastního kódu. To je sice zdlouhavější, nicméně se zde získává výhoda znovupoužitelnosti kódu. Také jeho následné úpravy mohou být mnohem jednodušší, protože požadované vlastnosti mohou být upravovány z jednoho místa [7].

Při programování uživatelského rozhraní je potřeba brát v potaz různé velikosti obrazovek u koncových zařízení. Z tohoto důvodu přichází Swift s mechanismem *Autolayout*. To je způsob definice rozložení prvků tak, aby se automaticky přizpůsobovaly displeji. Jedná se o efektivní způsob návrhu uživatelského rozhraní pro různě velké obrazovky či jejich různou orientaci. Každé zobrazované komponentě je v tomto případě potřeba nastavit omezující podmínky pro její velikost a pozici. To se nastavuje relativně vůči ostatním prvkům nebo vůči kontejneru, ve kterém se prvek nachází [3].

Uživatelské rozhraní se skládá z pohledů. To je něco, co vidí uživatel a s čím může interagovat. Pohledy se do sebe mohou vzájemně zanořovat. Hlavní pohled je okno celé aplikace, všechny ostatní pohledy a prvky uživatelského rozhraní se nachází v něm. To tvoří stromovou hierarchii pohledů [3].

2.4.3 Kód aplikace

Při psaní kódu aplikace je potřeba vytvořit propojení mezi uživatelským rozhraním a kódem. To se dělá pomocí takzvaných spojení. Existují dva druhy spojení: *outlet* a *action*. *Outlet* slouží pro referenci objektu, aby bylo možné na něj někde v kódu odkázat, například na jeho textový obsah. *Action* je metoda, která je zavolána při interakci s objektem, například u tlačítka při jeho zmáčknutí [3].

Testování aplikace může probíhat v simulátoru nebo na připojeném telefonu. Při testování v simulátoru nabízí Xcode velké množství různých simulovaných iPhonů nebo iPadů. Simulovaný přístroj nabízí podobné internetové připojení a možnosti nastavení (např. přizpůsobení displeje apod.) jako u skutečného přístroje. Testování v simulátoru má i jisté nevýhody, jako je například nepodporování push notifikací [3].

2.4.4 Notifikace

Notifikace slouží k upozornění z aplikace, když zrovna není zapnutá. Existují dva druhy notifikací: lokální a vzdálené. Lokální notifikace vyvolá samotná aplikace na základě nějaké události. Například notifikace vyvolaná v konkrétní čas. Vzdálené (push) notifikace nejsou naplánovány aplikací. Jsou volány webovou službou, a zpravidla cílí na více zařízení současně [8].

Každá push notifikace je odeslána poskytovatelem aplikace skrze server *Apple Push Notification services* (APNs) až k cílovým zařízením. Ke zprovoznění push notifikací pro nějakou aplikaci je potřeba několik kroků:

- Je potřeba pro aplikaci vytvořit *App ID*. To slouží k jednoznačné identifikaci aplikace mezi všemi ostatními, které pro iOS existují a umožňuje serverům APNs ji najít.
- Dále je potřeba vytvořit SSL certifikát pro push notifikace. Ten umožňuje našemu notifikačnímu serveru kontaktovat APN server.
- Dále je potřeba aplikaci na konkrétním telefonu registrovat pro přijímání notifikací. Aplikace se uživatele zeptá, jestli chce přijímat notifikace, a pokud zvolí možnost ano, tak je registrována u APNs. Aplikace na konkrétním telefonu je na APNs registrovaná a jednoznačně identifikovatelná pomocí unikátního identifikátoru jménem *device token*.

Následné notifikování telefonu je možné pomocí kontaktování APNs serveru. Je třeba vytvořit zprávu s notifikací ve formátu JSON. Tuto zprávu společně s vytvořeným SSL certifikátem odeslat pomocí HTTP/2 požadavku na APN server. Cílový telefon, pro který je daná notifikace, určíme odesláním HTTP/2 požadavku na adresu, ve které použijeme identifikátor zařízení (*device token*) [8].

Kapitola 3

SAP ME

V návaznosti na rozvoj informačních technologií v posledních desítkách let vznikl v oblasti výroby systém *Manufacturing Execution System* (MES). Ten se stará o oblast sledování, zaznamenávání a o kontrolu výroby. Umožňuje tak větší flexibilitu a efektivitu výrobního procesu.

SAP Manufacturing Execution (SAP ME) je takovýto systém zaměřený na řízení výroby. Jedná se o flexibilnější, rozšiřitelné a programovatelné prostředí, které poskytuje kompletní pokrytí výrobního procesu na jednotlivých linkách. Umožňuje tak firmě využívající tento systém mít absolutní kontrolu nad výrobním procesem a nad aktuálním stavem výroby.

3.1 SAP

SAP je německá firma založená v roce 1972 v německém Walldorfu. Zkratka vznikla z názvu *Systeme, Anwendungen, Produkte in der Datenverarbeitung*, což v češtině znamená Systémy, aplikace, produkty v oblasti zpracování dat. Firmu založilo pět bývalých zaměstnanců IBM a velmi rychle se stala vůdčí společností v oblasti plánování podnikových zdrojů. Tato firma vydává stejnojmenný ERP systém SAP. První pobočka v České republice byla otevřena v roce 1992 v Praze. Další z poboček sídlí v Brně a jedná se o vývojové centrum, které je součástí celosvětové sítě laboratoří SAP [10].

3.1.1 ERP

Enterprise Resource Planning (ERP), česky plánování podnikových zdrojů, je druh systémů, které se v organizacích používají pro evidenci každodenních obchodních aktivit. Mezi ty patří například získávání objednávek od zákazníků, projektový management, zásobování, plánování výroby nebo účetnictví. Systémy se starají a zařizují evidenci a správu všech podnikových dat. [9]

Systémy ERP poskytují pro každou část organizace potřebné moduly, aplikace, které evidují všechna data a firemní procesy a dávají jim pevně definovanou datovou strukturu. Díky tomu se v organizaci eliminuje velké množství samostatných databází a vzniká centralizované úložiště zajišťující integritu aktuálních a úplných dat v rámci celé organizace [9].

3.1.2 SAP ERP

Jedná se o ERP systém stejnojmenné firmy, která vydala software určený pro finanční účetnictví s názvem *SAP R/1*. Následně se tento systém rozšířil o správu materiálu [10].

Koncem 70. let začala firma vyvíjet systém *SAP R/2*, který se skládal z několika modulů. Skládal se z těch, které nabízel předešlý systém a k tomu nově z modulů pro firemní procesy vývoje, prodeje a distribuce. Systém byl určený pro mainframe, tedy pro sálové počítače, a firmám pomohl eliminovat nákladné papírování. Každý zákazník si následně mohl vybrat pouze moduly, které pro svoje podnikání potřebuje. Postupem času se systém rozšiřoval o další moduly, například pro plánování a kontrolu produkce. Díky velkému růstu se firma začala postupně rozšiřovat i za hranice Německa [10].

Roku 1992 firma vydala systém *SAP R/3*. Ten funguje na konceptu klient-server a je určen i pro středně velké podniky. Postupně se rozšiřuje na tehdy nejrozšířenější operační systém Microsoft Windows. Roku 2004 firma vydává aplikační a integrační platformu *SAP NetWeaver*, což je technologická základna pro SAP aplikace a dokáže spolupracovat s již existující infrastrukturou podniku [10].

Kvůli stále většímu počtu zpracovávaných dat vzniká roku 2011 systém *SAP HANA*, který nyní systému SAP dává možnost využívat in-memory databázi. Ta k ukládání dat používá operační paměť, nikoli pevné disky, a dosahuje tak výrazně vyšších rychlostí [10].

V dnešní době je systém SAP určen pro střední a velké podniky, ale nabízí i řešení pro podniky malé. Obsahuje několik modulů pro podnikové procesy nejrůznějšího druhu, jako je například účetnictví, logistika, výroba nebo personalistika. V systému SAP se manipuluje s daty pomocí vestavěných nebo definovaných transakcí. Je postaven na aplikačním serveru *SAP NetWeaver*, který umožňuje integraci všech ostatních SAP řešení. Systém SAP je rozšiřitelný, upravitelný a programovatelný vlastním jazykem ABAP. Je postavený na třívrstvé architektuře (prezentační, aplikační a databázová vrstva) a podporuje několik druhů databází. *SAP S/4 HANA* je verze, která podporuje in-memory databázový server *SAP HANA* [10].

3.1.3 SAP NetWeaver

SAP NetWeaver je technologickým základem pro mnoho ostatních SAP aplikací. Jedná se o prostředí (run-time environment), na kterém ostatní SAP aplikace běží. Obsahuje několik aplikací, jako jsou vývojové nástroje, aplikační server, prostředky pro integraci ostatních aplikací a také zajišťuje bezpečnost. SAP funguje na principu servisně orientované architektury a skládá se tedy z několika na sobě nezávislých komponent propojených skrze *NetWeaver* [11].

SAP NetWeaver Application Server je jedna ze součástí *SAP NetWeaver*. Je to Java aplikační server programovatelný v Javě nebo v proprietárním jazyku ABAP (*Advanced Business Application Programming*) a zpřístupňuje všechny ostatní SAP aplikace. Umožňuje vyvíjet a provozovat firemní aplikace [11].

3.1.4 SAP PP

SAP Production Planning (SAP PP) je modul určený pro plánování v průmyslu a výrobě. Sleduje proces výroby, spojení mezi materiálem a hotovým výrobkem, plánování materiálu, vykonávání výrobních postupů a pohyby výrobků. Součástí tohoto modulu je několik hlavních druhů dat, která jsou pro konkrétní společnost statická, závisí na druhu jejich výroby a většinu času se nemění [12].

- *Bill of Material* (BOM), neboli kusovník je seznam materiálu a jeho množství, které je potřeba na sestavení výsledného produktu. Z tohoto lze získat přehled o potřebném materiálu a ceně produktů.
- *Material Master* obsahuje informace ohledně veškerého materiálu, který společnost kupuje, se kterým pracuje a který produkuje. Materiál se rozděluje do dvou skupin podle jeho typu:
  - surový materiál
  - hotové zboží

  Tyto informace ukazují, co je na skladu, fyzické lokace materiálu, nebo slouží pro plánování objednávek.
- *Work Center* jsou linky, na kterých se provádí samotná výroba.
- *Routing* je postup výroby a obsahuje seznam operací na strojích (Work Center). Je zde uvedena i doba jednotlivých operací a slouží též k jejich plánování.
- *Production Version* je kombinace materiálu (BOM) a operací (Routing) provedených na nějakém výrobku.

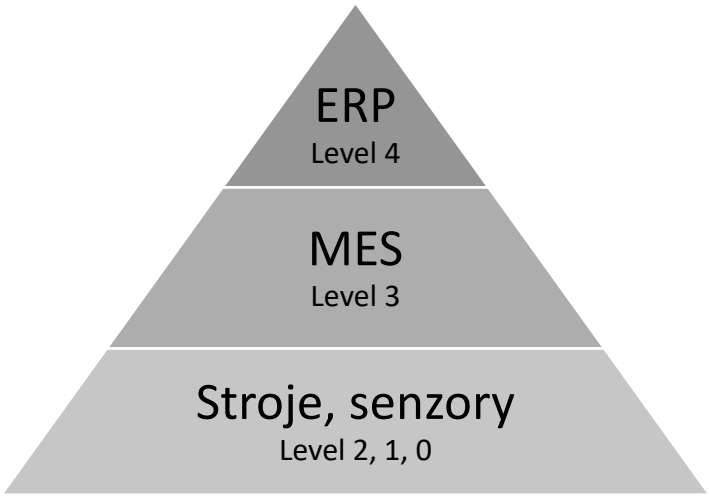
Dále SAP PP obsahuje *production order*, což je objednávka. Ta definuje, který materiál má být vyprodukován. Udává dílnu, kde se má výroba uskutečnit, požadované množství a čas kdy má být objednávka hotova [12].

Tento modul slouží k plánování výroby v závislosti na plánu plnění objednávek, zajištění všeho potřebného materiálu a případně jeho obstarání, pokud nějaký chybí, převedení objednávek na výrobní úkony, plánování výrobních kapacit a rozvržení času produkce. To vše se snahou co nejvíce omezit slabá místa výroby, takzvané „bottleneck“, což jsou stupně výrobní linky s nejmenší kapacitou, tedy místa, na kterých prakticky závisí celá doba výroby. Sledování průběhu výroby, odečítání a přeměnu materiálu, zaznamenávání stavu výrobních úkonů, objednávek a jejich dokončování [12].

3.2 MES

Pro firmu zabývající se výrobou je důležité sledovat a řídit kvalitu a efektivitu produkce. Systémy pro řízení a plánování podnikových zdrojů (ERP) neposkytují bližší pohled na proces samotné výroby ani informace o aktuálním stavu ve výrobní hale. Proto vznikl systém, který to umožní.

Manufacturing Execution Systém (MES) je výrobní informační systém, který je zaměřen na automatizaci výroby a sleduje proměnu materiálu skrze výrobní linky až k hotovým produktům. Poskytuje informace o aktuálním stavu ve výrobě, které mohou pomoci se zlepšením výrobního procesu a s plánováním další úkolů [13].



Obrázek 2: Pyramida ISA-95 popisující vztah mezi ERP systémy, MES systémy a výrobními linkami (inspirováno z [13])

Systém MES poskytuje bližší pohled na výrobu oproti systémům ERP. V roce 2000 vznikl standard *ANSI/ISA-95* který definuje hierarchii mezi těmito systémy (viz Obrázek 2). Jako nejvyšší patro pyramidy na „level 4“ situuje ERP systém, pod něj na „level 3“ systém MES a dolů monitorování výroby, výrobních procesů a aktuálního výrobního procesu [13].

3.2.1 KPI

Key Performance Indicators (KPI) jsou klíčové ukazatele pro hodnocení výkonnosti. Slouží pro konkrétní kvantifikovatelné posouzení množství práce vykonané systémem díky přesným číselným hodnotám. Poskytují tak pohled na proces dle pevně stanovených kritérií. Zobrazují informace o aktuálním stavu a slouží i pro sledování dlouhodobějšího trendu. Hodnoty KPI lze díky jejich přesnému významu zpracovávat i strojově. Dále je lze používat k plánování procesů na základě jejich aktuálních hodnot [20].

Overall Equipment Effectiveness (OEE), neboli celková efektivita zařízení se používá pro hodnocení výkonu výrobních zařízení. Jedná se o jedinou hodnotu udávanou v procentech. Poskytuje kvantifikovanou informaci ke srovnání výkonu jednotlivých zařízení. Její hodnota se počítá ze tří dílčích ukazatelů: dostupnosti, výkonu a kvality [21].

- *Dostupnost* (anglicky *availability*) je ukazatel udávající poměr skutečného času výroby ku plánovanému času výroby.
- *Výkon* (anglicky *effectiveness*) ukazuje poměr množství vyrobených kusů vůči původnímu plánovanému množství.
- *Kvalita* (anglicky *quality*) je poměr správných, korektních výrobků ku všem vyrobeným kusům.

Pro výpočet hodnoty OEE je potřeba vynásobit tyto tři ukazatele a jeho výsledná hodnota je udávána v procentech. Tyto dílčí ukazatele jsou často používány, nicméně nemají žádnou pevnou definici. Jejich hodnota je též udávána v procentech.

$$OEE = Aviability * Effectiveness * Quality \ [\%] \qquad (1)$$

Strom KPI je hierarchické uspořádání KPI ukazatelů. V nejnižší vrstvě jsou KPI od jednotlivých částí, například od jedné výrobní linky. Každá další vrstva je vypočítaná z KPI ukazatelů vrstvy pod ní. Tímto mechanismem lze získat přehled o větších částech systému. Nakonec úplně nahoře bude vypočítané celkové KPI pro celý systém [20].

Pro hodnocení kvality ve výrobním prostředí se požívá ukazatel anglicky nazývaný *yield*. Ten se dále dělí na dva druhy. *Frist Pass Yield* (FPY) udává poměr funkčních kusů ku všem vyrobeným kusům. Tento ukazatel je měřen na nejspodnější vrstvě stromu, je počítán zvlášť pro každou část výrobní linky, skrze které projde výrobek. *Rolled Throughput Yield* (RTY) je ukazatel kvality, který udává pravděpodobnost, že vyráběný kus projde správně přes všechny body výroby. Lze jej spočítat vynásobením FPY všech uzlů ve stromu pod ním [22].

3.2.2 Funkce systému MES

Mezi základní a nejčastější funkcionality systému MES patří správa výrobních zdrojů a postupů, plánování a řízení výroby, sběr dat a výkonnostní analýzy [14].

- Správa výrobních postupů zahrnuje veškeré informace k tvorbě výsledného produktu a zachycuje jeho životní cyklus. Zahrnuje výrobní pravidla, použité materiály, výrobní zdroje a udává, jakým způsobem je tvořen výsledný produkt.
- Správa zdrojů sleduje všechny zdroje a kapacity, které vstupují do výrobního procesu, jako je materiál, stroje, nebo energie, a sleduje jejich množství, využitelnost a dostupnost.
- Plánování výroby patří mezi nejdůležitější části. Definuje rozvržení materiálu, postup práce a využití všech potřebných zdrojů, což na takovéto úrovni obvykle ERP systémy neumožňují. Lze tak plánovat optimální využití všech zdrojů a efektivně plnit produkční požadavky. Dále se definují přesné výrobní procesy

a postupy, přiděluje se práce jednotlivým zdrojům a definuje postup řešení případných problémů.

- Je důležitá sledovatelnost (anglicky *traceability*) všech stupňů, přes které projde vyráběný produkt. Lze tak sledovat proces výroby a aktuální stav všech produktů od počátečního materiálu až k výslednému produktu, díky čemuž víme přesně, co se s produktem dělo pro případné pozdější řešení potíží.
- Sběr dat o výrobě, o materiálu a logování všech provedených akcí. To je důležité pro řízení kvality a minimalizaci defektů ve výrobním procesu. Dále se zaznamenává dostupnost linek, jejich produkce a kvalita výsledných produktů. Poskytují se zde dílčí informace o produktivitě jednotlivých linek a dále o celkové efektivitě zařízení. Ukazatel OEE kvantifikuje celkovou efektivitu zařízení a skládá se ze tří hlavních částí.
    - Dostupnosti linek, tedy časem který výrobní linky pracovaly.
    - Výkonem linek, což udává poměr skutečného výstupu z linky vůči plánovanému výstupu.
    - Kvalita udávající poměr správných, funkčních výrobků.

  Výpočet těchto ukazatelů se může lišit v závislosti na jednotlivých případech.

Systémy MES mohou být úzce provázané s ERP systémy a díky tomu propojit výrobní halu s řízením a plánováním v celé společnosti [14].

3.3 SAP ME

SAP Manufacturing Execution (SAP ME) je systém MES, který umožňuje snadnou integraci s ERP systémem SAP a dává tak možnost přiblížení mezi plánováním a skutečným stavem ve výrobní hale. Díky tomu může firma rychleji reagovat na aktuální stav výroby a regulovat plánování a požadavky [15].

SAP ME je systém zaznamenávající všechna hlavní data o výrobě, tedy data potřebná pro její řízení a plánování, a také zaznamenává informace o všech právě rozpracovaných položkách (WIP – *Work in Process*) [15].

V roce 2008 firma SAP koupila společnost *Visiprise* nabízející svoje MES řešení, z čehož následně vznikl produkt *SAP Manufacturing Execution*. Postupem času se SAP ME více přiblížil ERP systému SAP a byl také integrován s jeho aplikačním serverem *SAP NetWeaver* [15].

Nyní SAP ME zprostředkovává v reálném čase spojení ERP systému SAP se shop floor[9]. Umožňuje získávání a synchronizaci hlavních dat z ERP systému do vlastní databáze. Dále přidává možnost rozšíření informací z ERP systému o jednotlivé dílčí úkony na výrobních

[9] Spojení „shop floor“ se ve výrobním prostředí používá pro označení místa, kde pracují lidé těsně spjatí s výrobou, tedy například pro výrobní haly.

linkách. Umožňuje sledovat právě vyráběné položky. Zaručuje sledovatelnost všech produktů, tedy zaznamenání všeho, co se s danou položkou doposud dělo, sběr dat, případně umožňuje logovat chyby. Je zde i možnost testovat výrobky, hlídat, jestli nejsou některé defektní, a pokud ano, tak pro ně provádět předem definované úkony. Umožňuje připojení přímo ke strojům ve výrobní hale a jejich senzorům skrze *SAP Manufacturing Integration and Intelligence* (SAP MII) a *SAP Plant Connectivity* (PCo). Nakonec lze i zobrazovat mnoho hlášení pro analýzu aktuálního stavu na shop floor a též sbírat data pro pozdější statistické analýzy [15].

3.3.1 SAP MII

Sap Manufacturing Integration and Intelligence je aplikace starající se o synchronizaci dat mezi SAP ERP a provozními aplikacemi jako je MES. Též poskytuje analytická data a postupy pro zlepšení efektivity práce [15].

Tento systém se stará o integraci SAP ERP s jinými aplikacemi pomocí webových služeb. Sbírá a agreguje data z několika zdrojů a zpřístupňuje je pro ostatní aplikace. Dále poskytuje informace a analýzy výkonu výrobních operací a umožňuje vizualizovat KPI ukazatele [15].

3.3.2 Integrace SAP ME a SAP ERP pomocí SAP MII

Aplikaci MES potřebuje pro svoji funkci mnohá data z ERP systému SAP, jako je materiál, BOMs, pracovní stroje, routing, ale také data o objednávkách nebo aktuálním stavu ve skladu. Stejně tak potřebuje MES zasílat informace o vykonaných objednávkách, změnách materiálu a chybách přístrojů do ERP systému kvůli zachycení aktuálního stavu výroby. Objednávky *Product Order* pocházející z ERP systému se zde mění na objednávky *Shop Order* [15].

SAP ME běží na aplikačním serveru *SAP NetWeaver* a poskytuje propojení v reálném čase s ERP systémem SAP skrze SAP MII. To se stará o vzájemnou komunikaci mezi nimi. SAP MII tak obsahuje všechna data potřebná pro jejich vzájemnou komunikaci a mapování jednotlivých dat mezi těmito systémy [15].

SAP ME poskytuje produkční technický dashboard (POD – *Production Operation Dashboard*), který slouží pro grafické znázornění relevantních informací u jednotlivých výrobních linek. Dále poskytuje možnost sledovat životní cyklus produktů (WIP – *Work in Process*) pro zajištění výsledné očekávané kvality [15].

3.3.3 Architektura SAP ME

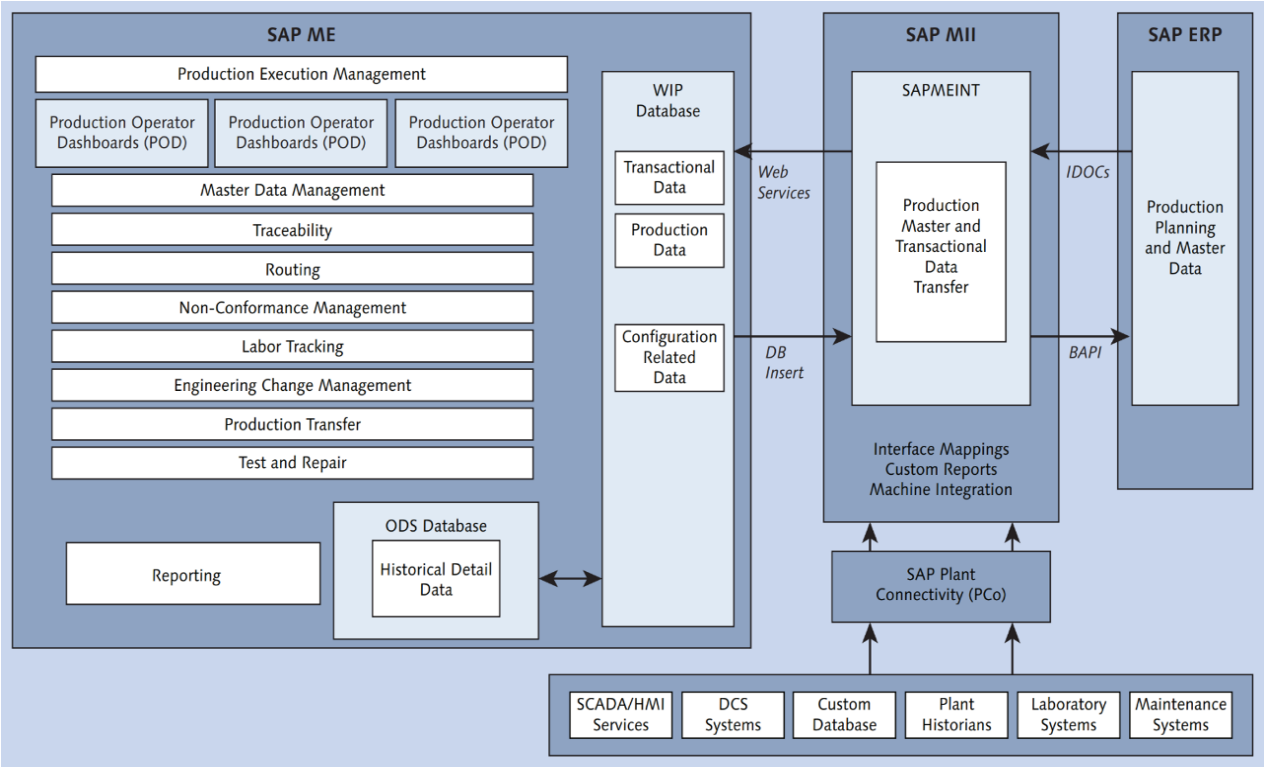
Pohled na celkové řešení SAP ME a jeho architekturu je zobrazen na Obrázek 3.

SAP ME a SAP MII běží na aplikačním serveru *SAP NetWeaver*. SAP MII zde figuruje jako propojení mezi SAP ME a ERP systémem SAP a stará se o přenášení hlavních dat určitým definovaným způsobem [15].

SAP ME používá svoji databázi WIP, kde ukládá všechna potřebná data, jako jsou hlavní data z ERP systému, transakční data a data související s nastavením SAP ME. Ukládají se zde tedy všechna data, co jsou vytvořena v systému SAP ME nebo přijata z ERP systému [15].

SAP ME také používá databázi *Operational Data Store* (ODS), která je určena pro ukládání historických dat. Systémem SAP ME prochází velké množství dat, která systém všechna ukládá.

Starší informace ovšem nejsou potřeba pro vykonávání úkonů na shop floor. Též by velké množství dat ve WIP databázi mohlo vést ke zpomalení celého systému, což je nežádoucí. Proto se historická data přemisťují do ODS databáze. Později je možno jejich využití pro analytické účely [15].



Obrázek 3: Pohled na řešení SAP ME a jeho součásti (převzato z [15])

SAP MII obsahuje databázi SAPMEINT. V ní jsou uloženy informace ohledně toho, jakým způsobem má SAP ERP komunikovat se systémem SAP ME. Standardně se tato komunikace převádí pomocí XSLT transformací [15].

Většina z funkcí, které poskytuje SAP ME, je přístupná skrze veřejné Java rozhraní PAPI a webové služby SOAP, které mohou být vykonány skrze komponentu SAP MII. SAP ME poskytuje SDK založené na Javě, díky kterému může být jeho funkcionalita volně rozšířena. SAP MII se také používá pro návrh vlastních reportů z dat uložených u SAP ME, které samotný SAP ME neposkytuje [15].

3.3.4 SAP MDO

V mnoha případech může nastat situace, kdy potřebujeme ukládat nějaká vlastní relevantní data v prostředí SAP ME. Mohou se například získávat data z několika různých zdrojů, dále může být požadováno je různě agregovat a vykonávat nad nimi výpočty. Pro takovéto případy je výhodnější ukládat data lokálně než je pro každý výpočet získávat z externích zdrojů.

Manufacturing Data Object (MDO) je technologie, která slouží pro ukládání perzistentních dat na platformě SAP MII. Takováto data jsou získávána z různých zdrojů a ukládána do MDO

databáze. MDO používá pro komunikaci s databází OpenSQL. OpenSQL je zde využíváno kvůli podpoře všech databází, které mohou fungovat se systémem SAP [16].

SAP MII umožňuje vytvářet MDO objekty a načítat do nich data z různých zdrojů, buď přímo, nebo nad nimi před načtením provádět nějaké operace. Tato data lze potom získávat ze zdrojů v určitých intervalech, nebo podle definovaných spouštěčů. MDO lze také používat pro ukládání vlastních persistentních dat v rámci systému SAP MII [16].

3.3.5 SAP MII Transakce

Prostředí SAP MII umožňuje vytvářet transakce. Ty slouží k přístupu k datům z různých zdrojů a vykonávání procesů. Transakce jsou spouštěny synchronně nebo asynchronně. Lze je vytvářet například pro získávání dat, provádění složitých výpočtů nebo vytváření reportů nad daty [17].

Transakce může mít vstupní a výstupní parametry. Vytváří se pomocí různých vzájemně propojených bloků. Jednotlivé bloky mohou načítat data například pomocí SQL příkazů, provádět nad nimi výpočty, v závislosti na nich odesílat emaily nebo ukládat logovací záznamy. Výsledná data může transakce buď někam uložit, nebo vrátit například v XML formátu [17].

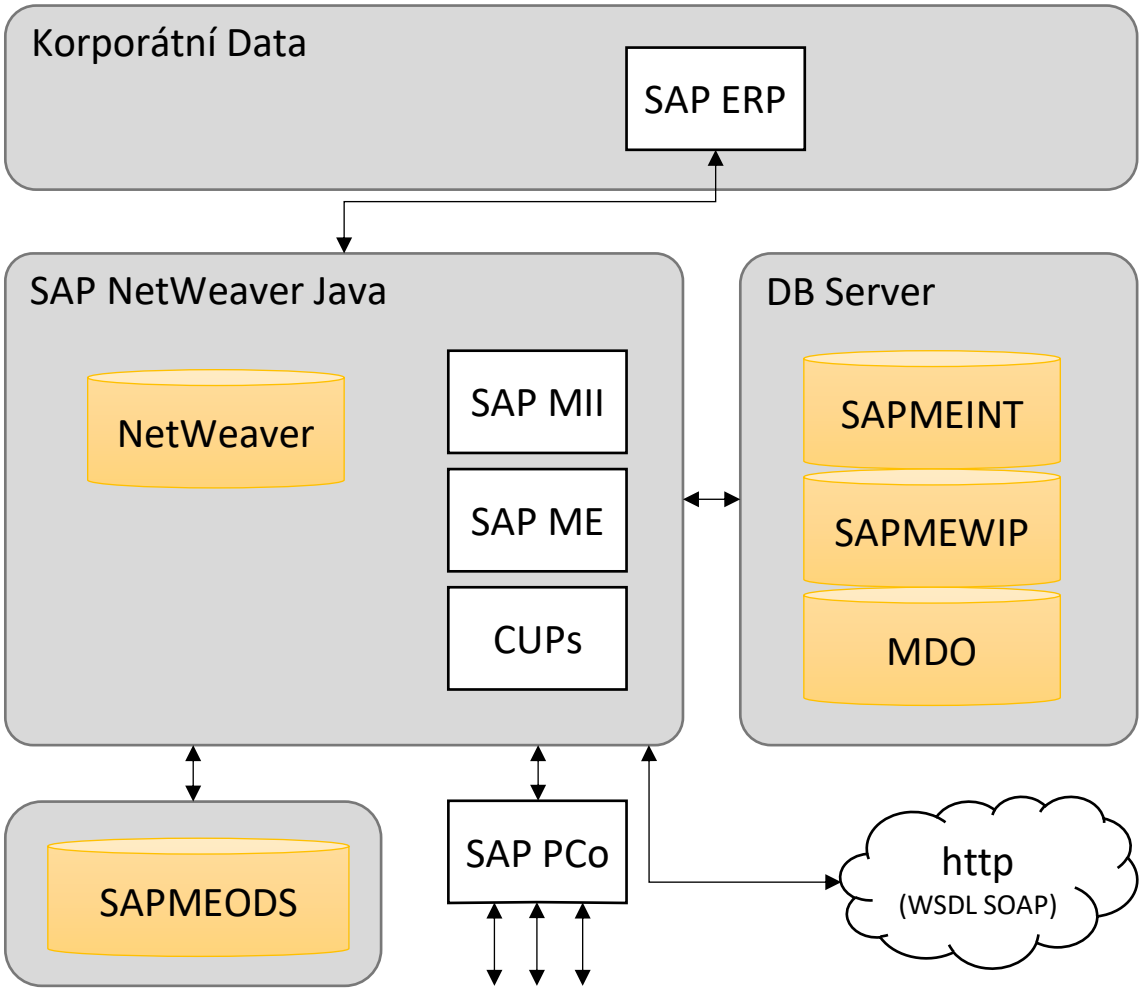
Vytvořené transakce lze spouštět buď přímo z prostředí SAP MII nebo z externích aplikací za pomocí webové služby *Simple Object Access Protocol* (SOAP) skrze HTTP. Vstupy a výstupy pro transakci se definují pomocí *Web Service Definition Language* (WSDL), což je jazyk vycházející z formátu XML pro popis funkcí a jejich parametrů a výstupu [17].

3.4 SAP ME v Resideo

Ve firmě Resideo je zaváděn nový výrobní systém SAP ME, který již funguje na několika linkách. Hlavní data čerpá z ERP systému SAP. SAP ME se skládá z několika serverů, kde nejdůležitější je databázový server a *SAP NetWeawer Java aplikační server*. Architekturu SAP ME ve firmě Resideo znázorňuji na Obrázek 4.

Na databázovém serveru se nachází integrační databáze SAPMEINT, která obsahuje důležité informace pro nastavení a fungování SAP MII. Dále se zde nachází databáze SAPMEWIP. Ta obsahuje veškerá data ohledně aktuálního dění na shop floor jako jsou informace o právě rozpracovaných položkách, či veškeré informace z jednotlivých výrobních linek. Nakonec se zde nachází i databáze MDO, která obsahuje všechny uživatelem definované tabulky a data.

Na aplikačním serveru se nachází webová aplikace pro přístup k SAP MII. Ta slouží pro spravování veškerého nastavení. Dále se zde nachází samotná aplikace SAP MII zařizující přenášení dat mezi systémy, SAP ME starající se o chod na shop floor a dále *Common Unix Printing System* (CUPs) starající se o tisk dokumentů. Tento aplikační server komunikuje s ERP systémem SAP, dále se všemi výrobními linkami skrze SAP PCo a nakonec s databázovými servery.



Obrázek 4: Architektura SAP ME ve firmě Resideo (autor: Marek Skalník)

Z důvodu, aby se příliš často nepřistupovalo k produkční databázi SAPMEWIP, jsou data z ní v pravidelných intervalech kopírována do MDO databáze. K těmto datům může mít uživatel přístup například skrze vytvořené transakce v SAP MII systému. K SAP MII lze přistupovat buď skrze NetWeaver webovou aplikaci, nebo přes webové služby WSDL SOAP.

Kapitola 4

Webové služby

Webové služby vznikly kvůli potřebě vzájemné komunikace strojů v počítačové síti. Umožňují tak dvěma strojům vzájemnou komunikaci skrze internet a využívají k tomu soubory nejčastěji ve formátech jako je XML nebo JSON, což jsou pro stroje snadno zpracovatelné formáty [19]. Existují dva hlavní typy webových služeb, což jsou REST a SOAP.

4.1 REST

Representational State Transfer (REST) je způsob, jak může žadatel (klient) přistupovat a manipulovat s informacemi na serveru skrze HTTP. Tento způsob je jednoduchý a nejedná se o formální standard. Představil je Ray Fielding ve své disertační práci v roce 2000 [19].

Jedná se o datově orientovaný druh komunikace, je tedy určen pro manipulaci (CRUD – *Create, Read, Update, Delete*) s daty (webové zdroje[10]). Ta jsou na webovém serveru adresována pomocí URI. Lze adresovat a pracovat s celými kolekcemi nebo jen s jejími prvky. Neexistuje žádný standardizovaný formát zpráv, které si mezi sebou mohou předávat klient se serverem. Proto mohou mít různé formáty, jako je XML, HTML, JSON, a další… K manipulaci s těmito daty jsou využívány metody, které standardně nabízí protokol HTTP [19]:

- GET – pro získání dat ze serveru určených podle URI
- PUT – pro přepsání (nebo případné vytvoření) aktuálních dat na serveru
- POST – pro vytvoření nového prvku na serveru
- DELETE – slouží pro smazání adresovaného prvku

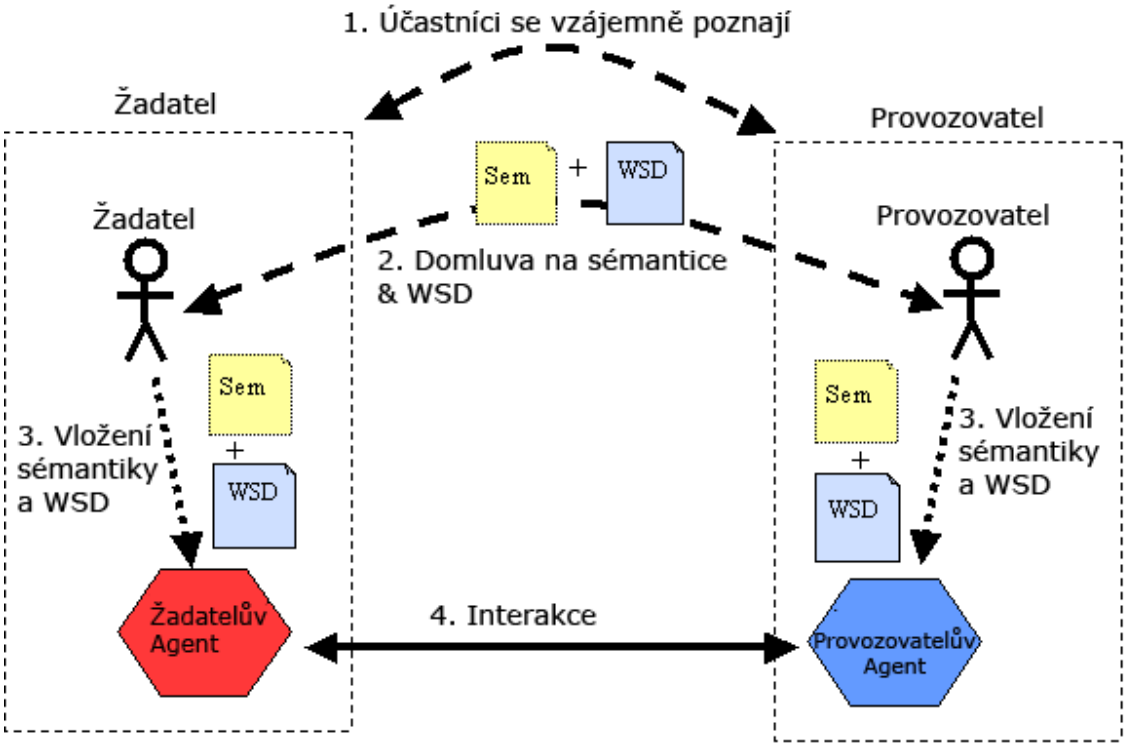
[10] Jedná se o digitální, fyzické či abstraktní položky jednoznačně identifikovatelné pomocí Uniform Resouce Identifier (URI)

4.2 SOAP

Simple Object Access Protocol (SOAP) je komunikační protokol určený pro výměnu strukturovaných informací. Zde popíši webové služby SOAP tak, jak je popisuje W3C.

SOAP formalizuje použití XML pro přenos údajů mezi dvěma stroji, definuje rámcový model pro rozšiřitelnost, verzi protokolu, způsob přenosu chybových informací a způsob zasílání zpráv přes HTTP [18].

SOAP není vázán na jeden jediný protokol. Umožňuje komunikaci skrze velké množství protokolů, nejčastěji však přes HTTP (*Hypertext Transfer Protocol*), nebo přes SMTP (*Simle Mail Transfer Protocol*). Tělo zprávy SOAP obsahuje jakékoli XML. Dále díky tomu, jak je navržen, zaručuje nezávislost vůči všem platformám [18].



Obrázek 5: Příklad komunikace pomocí webové služby SOAP (inspirováno [18])

Obrázek 5 ukazuje příklad volání služeb pomocí SOAP. Nejdřív uživatel potřebuje najít poskytovatele služby (viz dále UDDI). Dále si oba potřebují vzájemně vyměnit sémantiku služby a syntaxi zápisu XML, který bude mezi nimi odesílán (viz dále WSDL). Služba musí přijímat jimi domluvený formát zpráv. Nakonec může uživatel službu využívat domluveným způsobem [18].

4.2.1 WSDL

Web Services Description Language (WSDL) je jazyk pro popis síťové služby vycházející z formátu XML. Popisuje rozhraní pro dané služby, tedy sadu zpráv SOAP a způsob, jakým se zprávy vyměňují. Díky tomu lze získat informace, jaký formát má mít zpráva zasílaná na server, a jaký formát bude mít přijímaná odpověď [18].

4.2.2 UDDI

Universal Description, Discovery, and Integration (UDDI) je registr založený na formátu XML pro popis webových služeb. Obsahuje informace o poskytovatelích služeb, jejich nabízených službách a potřebná metadata. To je zapsané pomocí WSDL [18].

Kapitola 5

Specifikace aplikace

Ve výrobním závodě Brno Resideo se nachází několik výrobních linek, které jsou připojeny na informační systém SAP ME. Tento systém je schopen shromažďovat a spravovat aktuální data týkající se jejich chodu, výkonu a celkového fungování. V této kapitole se zabývám specifikací aplikace získávající tato data ze serveru SAP ME skrze webové služby, která je dále bude zobrazovat.

5.1 Analýza požadavků

Z dat, které dokáže informační systém SAP ME shromažďovat, lze vyčíst mnoho problémů, které mohou nastat na výrobních linkách. V případě, že nějaký problém nastane, je potřeba co nejrychleji informovat člověka, který má tento problém řešit. Standardně SAP ME neposkytuje způsob, jak o takovýchto problémech rychle informovat nějakou osobu, takže případný problém je potřeba řešit například skrze emaily. Ty nejsou vhodné, neboť se email může ztratit v desítkách ostatních.

Ve firmě Resideo je používán jako firemní telefon iPhone. Telefony mají zaměstnanci většinu času u sebe a z toho důvodu je vhodné je použít pro rychlé informování o případných problémech na výrobních linkách. Současně může být užitečné, když uživatel uvidí důležité informace o aktuálním dění na *shop floor* z aplikace dostupné na jeho telefonu.

Cílem je tedy vytvořit aplikaci na mobilní telefon s operačním systémem iOS, která bude zobrazovat informace dostupné z výrobních linek. V případě poklesu některého měřeného parametru pod definovanou mez bude uživateli zaslána notifikace.

5.1.1 Hierarchie výrobních linek

Firma Resideo Brno je výrobní závod, ve kterém jsou tři divize, v každé divizi několik výrobních center, které se dále dělí až po jednotlivé pracovní stanice. Na tento případ lze tedy nahlížet jako na hierarchickou strukturu strom. Konkrétní hierarchie tak, jak je uložena v systému SAP ME, včetně používaných anglických názvů je následující:

- *Work Center Division* – Označuje celou divizi, jako je například elektro nebo mechanika.
- *Work Center Linegroup* – Jedná se o skupinu výrobních linek. V našem případě jsou maximálně dvě pro celou divizi.
- *Work Center Line* – V překladu výrobní linka, která se dále dělí na několik buněk.
- *Work Center Cell* – Další rozdělení v rámci jedné výrobní linky. Zde už dochází například k výrobě nebo balení konkrétních vyrobených kusů. Může se jednat například o část výrobní a část analytickou.
- *Work Center Operation* – Operace jsou části buňky výrobní linky, na kterých se provádí jednotlivé operace spojené s výrobou.
- *Resource* – *Resource* je v systému SAP ME ještě elementárnější označení pro operace na výrobních linkách. V našem případě je zde v mnoha případech mapování 1:1.

Mobilní aplikace by měla být proto schopna tuto hierarchii zobrazit, a dále k patřičným objektům přiřadit měřená a vypočítaná data.

5.1.2 Měřené hodnoty na výrobních linkách

Pro získání základního přehledu o stavu na výrobních linkách lze použít ukazatele KPI (*Key Performance Indicators*). Ukazatele, které si definujeme, budou vycházet z výpočtů pro OEE, tedy kvalitu, dostupnost a efektivitu.

Protože jsou výrobní linky uspořádané ve stromové struktuře, lze pro ně obdobným způsobem počítat i KPI ukazatele. To znamená počítat jejich hodnoty na nejspodnější vrstvě tohoto stromu, v uzlech, a z nich vypočítat hodnoty pro vyšší úrovně stromu až ke kořeni, tedy pro celou divizi. Výpočet těchto ukazatelů ovšem v reálném provozu není jednoznačný, nebo nemusí mít dostatečnou vypovídající hodnotu, a proto bude docházet k situaci, kdy některý z ukazatelů nebude pro konkrétní uzel počítán.

Na nejnižší úrovni (*Resource*) bude zobrazován pouze ukazatel FPY (*First Pass Yield*). Na vyšší úrovni u jednotlivých buněk (*Work Center Cell*) může být počítán RTY (*Rolled throughput yield*) (popsán v kapitole 3.2.1), a dále zde bude zobrazen KPI ukazatel výkonu pro danou buňku.

Aplikace bude schopná zobrazit nejaktuálnější stav KPI na daném výrobním objektu a dále historii hodnot pro každou celou hodinu, aby uživatel viděl aktuální vývoj tohoto KPI na výrobní lince. Zobrazovaná historie bude stačit pouze po dobu osmi hodin, starší by neměla praktický smysl.

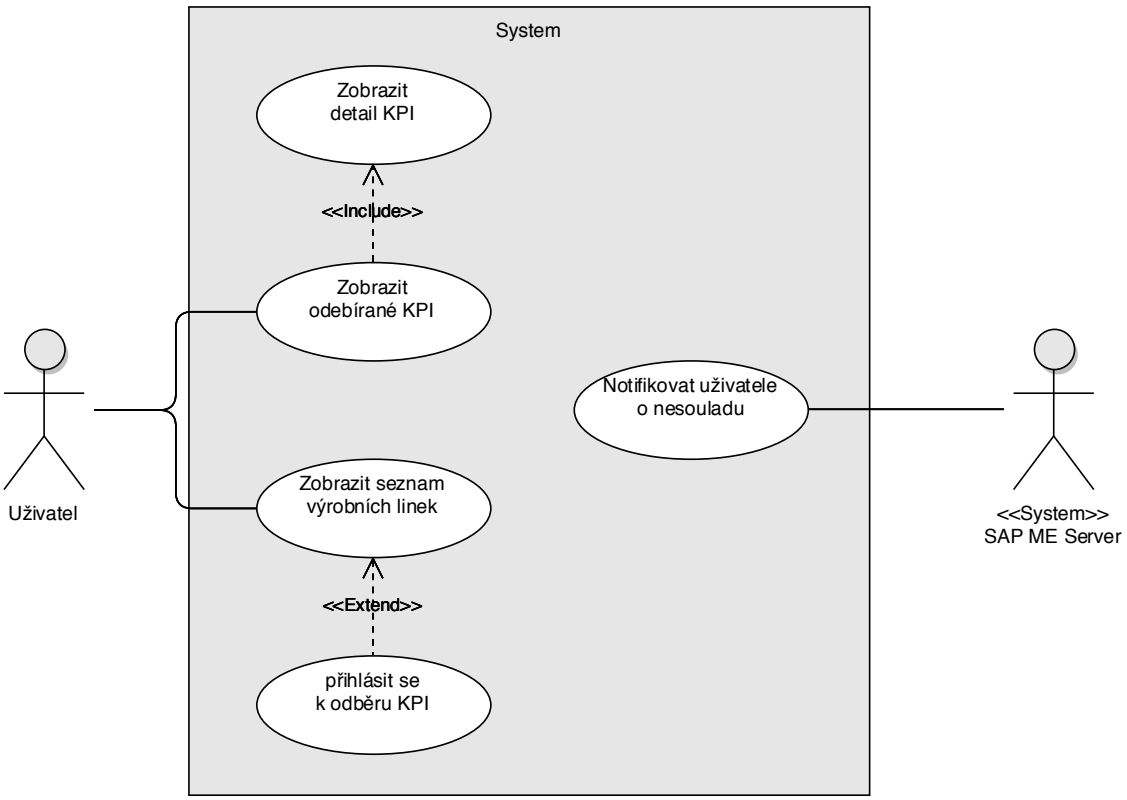
5.1.3 Systém upozorňování na nesoulady

Měřením KPI dostaneme ukazatel, který nám udává stav na výrobních linkách. Pokud se objeví nějaký problém, například s kvalitou výroby, je třeba upozornit pracovníky, odpovědné za tuto část výroby.

Proto každý ukazatel KPI pro konkrétní objekt, kromě zrovna naměřené hodnoty, bude mít definovanou hodnotu cílovou, kterou chceme dosáhnout, a dále hodnotu, při které uživatel dostane upozornění na problém.

K tomu, aby uživatelé získávali pouze upozornění o výrobních linkách, které je zajímají, zde bude systém odběrů. Každý uživatel si bude moci přihlásit odběr ke konkrétnímu objektu (např. výrobní lince) a jeho KPI. V případě, že uživatelem odebírané KPI na výrobní lince nedosáhne určité hodnoty, dostane tento uživatel upozornění na svůj mobilní telefon.

5.2 Návrh mobilní aplikace



Obrázek 6: Diagram užití mobilní aplikace (autor: Marek Skalník)

Výstupem tedy bude mobilní aplikace, která bude zobrazovat výrobní linky a ostatní objekty a na nich dostupná KPI. Dále bude umožňovat přihlásit se k odběru zvolených KPI na konkrétních linkách a bude zobrazovat hodnoty všech odebíraných KPI na linkách. Při poklesu některého z nich pod určitou mez bude uživatel upozorněn prostřednictvím notifikace. Všechny hodnoty bude mobilní aplikace načítat ze serveru SAP MII skrze webové služby. K přístupu na server SAP

MII je potřeba uživatelský účet, takže při prvním spuštění mobilní aplikace se uživatel bude muset přihlásit svými uživatelskými údaji, které budou dále posílány na server.

Aplikace by měla být uživatelsky přívětivá a také by měla podporovat českou a anglickou lokalizaci, tedy zobrazovat nápisy jak česky, tak anglicky, pro snazší zprovoznění aplikace i pro jiné výrobní závody než v České Republice.

5.2.1 Zobrazení stromu výrobních linek

Uživatel bude mít možnost zobrazit seznam všech zařízení, na kterých je počítána nějaká hodnota KPI. V systému SAP ME je stromová hierarchie výrobních linek (popsaná v kapitole 5.1.1 Hierarchie výrobních linek). Vhledem k četnosti konkrétních uzlů v tomto stromu bude stačit, aby mobilní aplikace byla schopna zobrazit pouze některé její úrovně. Tyto úrovně, které bude aplikace zobrazovat, budou:

- *Work Center Linegroup*
- *Work Center Line*
- *Work Center Cell*
- *Resource*

Toto rozlišení bude stačit pro jednoznačnou a zároveň snadnou orientaci v aplikaci. Kvůli tomu, že se v praxi běžně používají pro jednotlivé úrovně anglické názvy, bude i mobilní aplikace zobrazovat tyto názvy výjimečně pouze anglicky.

Měřené a počítané hodnoty KPI budou pouze na vrstvách *Work Center Cell* a *Resource*. Na ostatních vrstvách nebudou počítány. Dále zde bude u těchto položek možnost odebírat počítané KPI. Na úrovni *Resource* bude mít každá položka možnost odebírat kvalitu, a o úroveň výš na *Work Center Cell* bude možnost u položek odebírat výkon. V případě, že už dané KPI je odebíráno, bude zde místo možnosti „odebírat“, naopak „zrušit odběr“.

U každého KPI bude k dispozici jeho aktuální vypočítaná hodnota a taky jeho osmihodinová historie. Ta tu bude převážně kvůli tomu, aby uživatel viděl trend, jak se hodnota KPI na daném objektu vyvíjí v čase.

Aplikace bude umožňovat zobrazení všech odebíraných KPI. Zde to bude jednoduchý seznam s názvem objektu, typem měřeného KPI a jeho aktuální naměřenou hodnotou. Pro zobrazení detailních hodnot osmihodinové historie půjde u každé položky zobrazit její detailní přehled. Dále bude u každého KPI definována pevná hodnota, a pokud dojde k jejímu překročení, uživatel dostane notifikaci.

5.2.2 Zasílání notifikací

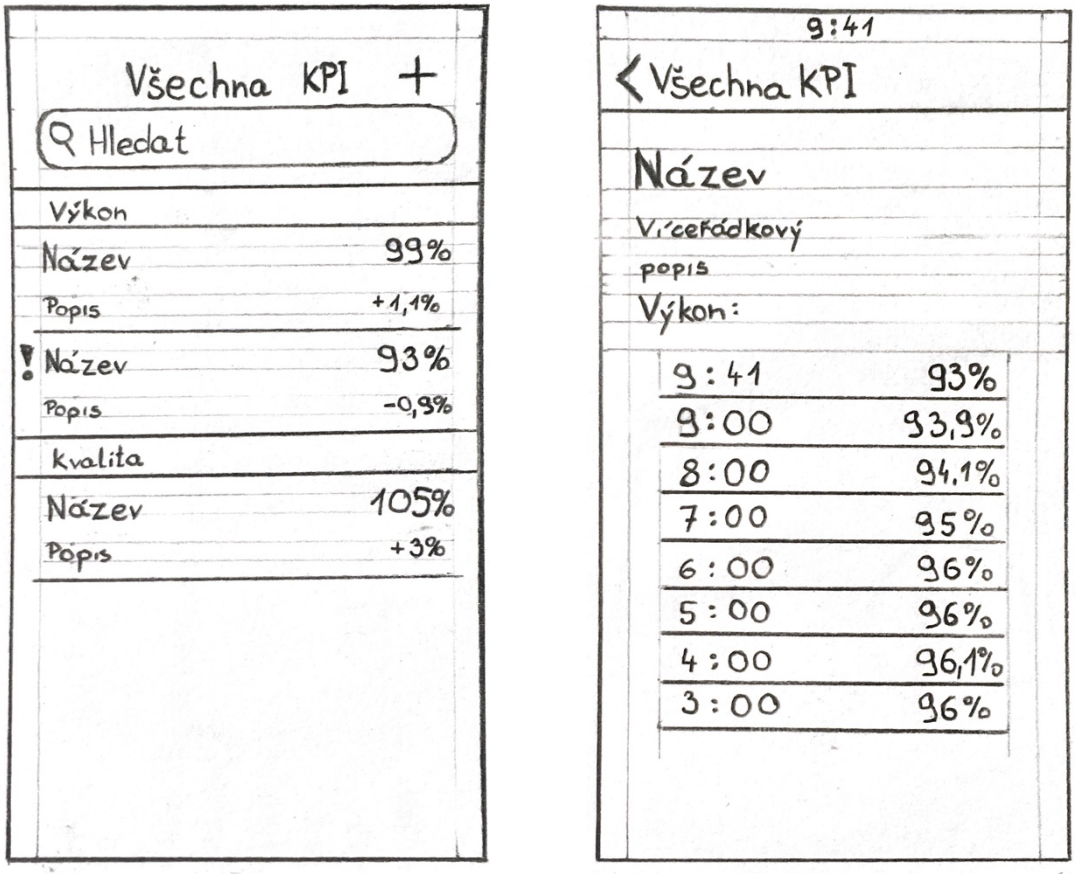
Aplikace by měla zasílat notifikace uživatelům v případě, že dojde k překročení jakékoli hodnoty odebíraného KPI. Toho by se docílilo tak, že by se na SAP MII serveru v pravidelných intervalech počítaly nové hodnoty KPI a v případě překročení meze by byla zaslána notifikace na patřičné uživatele skrze *Apple Push Notification service*.

Zde je ovšem problém, že server SAP MII ve firmě Resideo podléhá přísným bezpečnostním nařízením, a není zde povoleno komunikovat ze serveru ven do internetu (výjimku

tvoří zavolání transakce skrze HTTP SOAP a získání odpovědi). Z toho důvodu bude aplikace podporovat pouze lokální notifikace. To znamená, že si aplikace bude sama v určitých intervalech načítat data ze serveru a kontrolovat, jestli nedošlo k problému. Aplikace tak sama poběží na telefonu na pozadí.

5.3 Návrh uživatelského rozhraní

Aplikace bude obsahovat několik obrazovek ve stylu typickém pro operační systém iOS. Většinou se bude jednat o tabulky zobrazující nějaké informace. Na úvodní obrazovce uživatel uvidí seznam jím odebíraných výrobních linek a objektů. Každý výrobní objekt má svoje jméno a nějaký popis. Tyto objekty budou seskupené podle druhu KPI, které je na nich počítáno. Úvodní obrazovka bude též nabízet možnost vyhledávání v seznamu odebíraných objektů podle názvu a případně popisu. V pravém horním rohu bude možnost editace odebíraných linek.



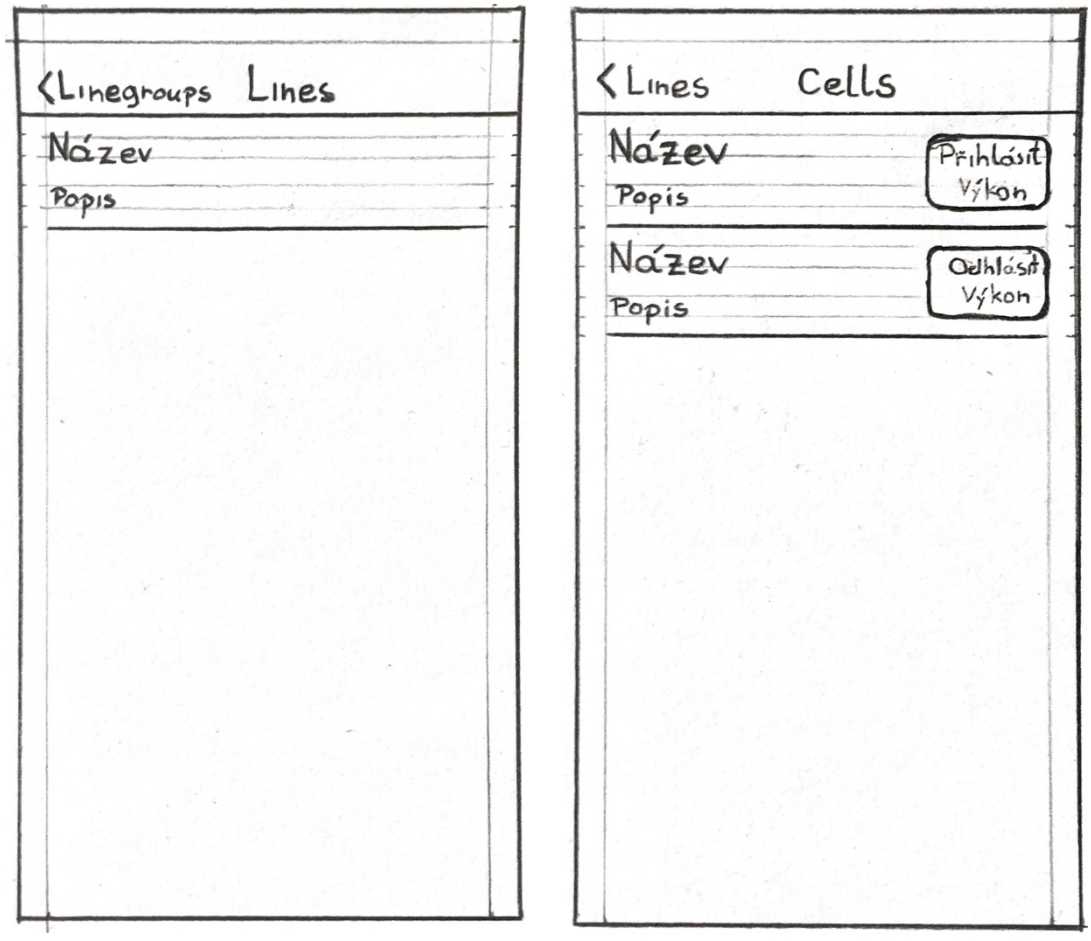
Obrázek 7: Návrh úvodní obrazovky uživatelského rozhraní iOS aplikace. Na levé straně seznam odebíraných objektů a na pravé straně je zobrazený detail pro zvolenou položku se seznamem hodnot KPI naměřených za posledních 8 hodin. (autor: Marek Skalník)

Aby si mohl každý uživatel definovat prioritu s jakou mu budou chodit notifikace, bude zde možnost změnit pořadí řádků, ve kterém se mu objekty zobrazují. Objekt na prvním řádku má větší prioritu než objekty na řádcích pod ním. Taky zde bude možnost jednotlivé řádky mazat,

což u daného objektu zruší odběr pro tohoto uživatele. Uživatel bude moci obnovit tabulku pomocí gesta tažením prstem směrem dolů, aby se načetla aktuální data ze serveru.

Ke každému zobrazenému objektu bude napsán jeho název, popis, poslední naměřená hodnota KPI a změna za poslední hodinu (viz Obrázek 7, levá strana). Hodnoty KPI pro kvalitu budou zobrazeny v procentech a hodnoty KPI pro výkon budou v celých číslech. Barva čísla bude buď zelená, nebo červená v závislosti na tom, jestli je daná hodnota nad nebo pod požadovanou mezí. Stejný barevný systém bude mít změna (v případě kvality procentuální) za poslední hodinu, v závislosti na tom, jestli bude kladná či záporná. V případě, že aktuální hodnota spočítaného KPI za poslední hodinu bude menší než je hodnota, při které má uživatel dostat upozornění, objeví se před názvem objektu vykřičník (viz Obrázek 7, levá část na řádku 2). Uživatel bude moci kliknout na každý řádek, a zobrazit detailní informace.

Na stránce s detailními informacemi bude opět název a popis odebíraného objektu. Dále zde bude tabulka s hodnotami KPI pro aktuální čas (hodnota KPI za posledních 60 minut), a sedm dalších, předešlých hodin (jak je ukázáno na Obrázek 7, na pravé straně).



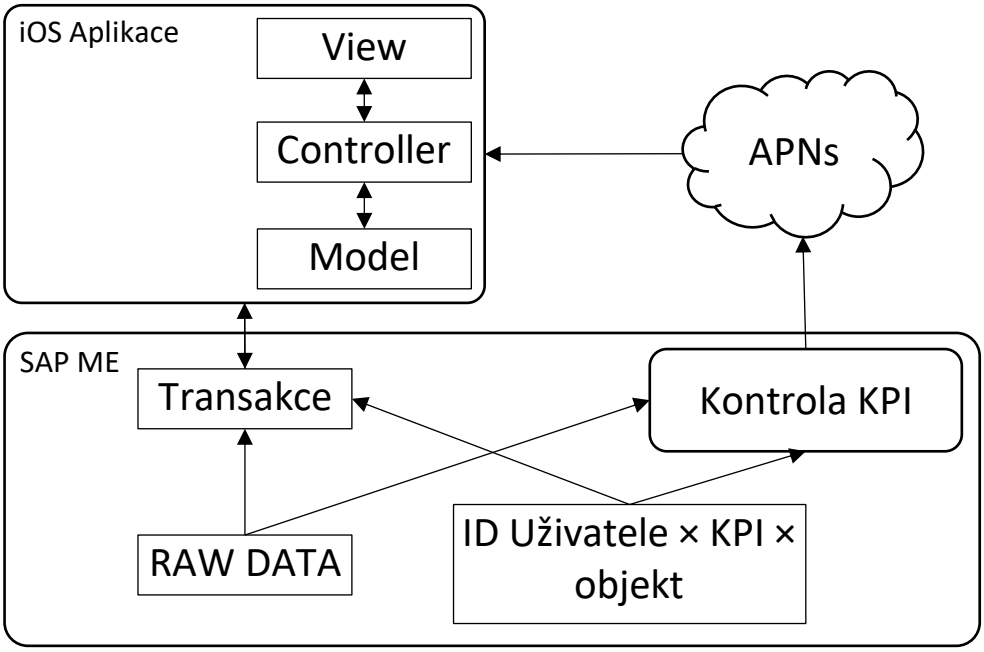
Obrázek 8: Návrh uživatelského rozhraní iOS aplikace pro zobrazení stromu výrobních linek včetně tlačítek pro přihlášení/odhlášení odběru KPI, pokud je to u daného řádku možné. (autor: Marek Skalník)

Pokud bude chtít uživatel přidat nebo odebrat odebírané KPI, zobrazí se mu strom všech výrobních linek. Ten se bude zobrazovat po jednotlivých úrovních. Na Obrázek 8 je na levé straně vidět seznam výrobních linek (*lines*) spadajících pod jednu konkrétní skupinu výrobních linek (*linegroup*), a na pravé straně je seznam výrobních buněk (*cells*) spadajících pod jednu výrobní linku (*line*).

V případě, že u objektu bude možnost odebírat nějaké KPI, bude zde tlačítko přihlásit nebo odhlásit odběr pro dané KPI. To znamená, že na úrovni výrobních buněk (*cells*) budou tlačítka s možností odebírat výkon (viz Obrázek 8, pravá strana), a na úrovni *resource* budou na řádcích tlačítka s možností odebírat kvalitu. V případě zmáčknutí tohoto tlačítku dojde k přidání tohoto objektu na úvodní obrazovku aplikace. Tlačítka pro přihlášení a odhlášení odběru se budou barevně odlišovat, aby byla rozpoznatelná na první pohled.

5.4 Návrh architektury

Existuje několik výrobních linek, které ukládají veškerá svoje data na SAP ME server. Odtud se v pravidelných intervalech kopírují data do uživatelsky definovaných tabulek. Také se při tom počítají různá KPI pro konkrétní linky. Tyto tabulky jsou stále na SAP ME serverech. Mobilní aplikace tato data bude číst a zobrazovat.



Obrázek 9: Grafické znázornění návrhu iOS aplikace a jejího vztahu se serverem SAP ME (autor: Marek Skalník)

Dále je potřeba na straně serveru sledovat vypočítávané hodnoty KPI a v případě poklesu pod určitou hranici odeslat notifikaci na telefony. V aplikaci na telefonu se bude moci uživatel přihlásit k odběru kteréhokoli počítaného KPI pro konkrétní objekt. Na straně serveru se tato informace uloží, tedy uživatel a jeho zvolená KPI, která odebírá. Na serveru dále poběží aplikace,

hlídající pokles kteréhokoli KPI a zasílající notifikace prostřednictvím APN Serveru. To je zobrazeno na Obrázek 9, blokem „Kontrola KPI“. Kvůli bezpečnostním pravidlům serveru SAP ME ve firmě Resideo znemožňujícím komunikaci ze serveru ven do internetu, zde v této diplomové práci tato část nebude implementována. Pro zasílání notifikací na mobilní telefony je totiž potřeba odesílat zprávy ze severu SAP ME na servery Applu, což aktuálně není možné.

5.4.1 Server SAP ME

Na serveru SAP ME budou data o jednotlivých linkách, jejich ukazatelích KPI, které poskytují, a pro každé z nich staticky definovaná požadovaná hodnota a jeho nejnižší povolená hranice. Dále zde budou aktuální hodnoty jednotlivých KPI. Také zde bude tabulka udávající, jaký uživatel odebírá jaká KPI (viz Obrázek 9). Pro komunikaci s mobilní aplikací zde budou sloužit transakce, které lze volat skrze webové služby a které mohou vracet data v XML formátu.

V případě, že na serveru SAP ME dostaneme výjimku umožňující komunikaci se servery Applu, poběží zde aplikace, která bude kontrolovat aktuální hodnoty KPI, a v případě překročení hranice kteréhokoli z nich odešle zprávu na APN server a ten notifikuje vybrané uživatele.

5.4.2 Aplikace

Samotná aplikace bude napsána v jazyce Swift. Grafické uživatelské rozhraní bude vytvořeno pomocí komponent storyboard. Data o zobrazených linkách na úvodní obrazovce budou načtena ze serveru a jejich pořadí, které může uživatel měnit bude uloženo lokálně na telefonu.

Vrstva aplikace model (viz Obrázek 9) se bude starat o načítání a ukládání veškerých potřebných dat uložených v telefonu a také dat z databáze SAP ME. Na serveru SAP ME budou definovány transakce vracející všechna potřebná data z tabulek na serveru. K nim bude aplikace přistupovat pomocí webových služeb, které SAP ME nabízí.

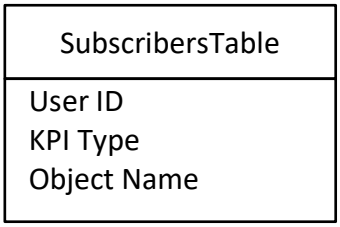
Vrstva controller (viz Obrázek 9) bude spojovat načítaná data a posílat je do uživatelského rozhraní.

5.5 Návrh serverové části mobilní aplikace

Všechna data, která budou v aplikaci zobrazená, budou získávána ze serveru SAP ME. Jedná se o definice výpočtu KPI, určení jejich maximálních požadovaných hodnot a jejich samotný výpočet. Taky zde bude stromová struktura všech linek, které dokáže aplikace zobrazit. Zde lze k datům přistupovat vzdáleně skrze transakce a webovou službu SOAP. Tyto transakce budou získávat data z již existujících datových zdrojů a z tabulky, kde budou uloženi uživatelé a k nim objekty a KPI, které odebírají.

5.5.1 Perzistentní data

Jediná data v rámci aplikace, která budou muset být uložena na SAP ME serveru, jsou informace o tom, který uživatel odebírá který objekt. Ostatní data týkající se hodnot KPI zde již jsou. Proto zde vytvořím tabulku odběratelů. Bude se jednat o MDO objekt, obsahující jméno uživatele a objekt, který odebírá (viz Obrázek 10). Pokud se v této tabulce bude nacházet kombinace uživatele a objektu, znamená to, že daný uživatel tento objekt odebírá.



Obrázek 10: Struktura tabulky odběratelů (autor: Marek Skalník)

V prostředí SAP ME je MDO objekt na pozadí běžná SQL tabulka. Ta ovšem má svůj název i názvy sloupců určené systémově a pro uživatele nečitelně. Proto se k přístupu k datům z MDO objektu používají MDO příkazy (což je obdobné SQL příkazům). Ty se po spuštění přeloží na SQL příkazy s přeloženými názvy pro tabulky a sloupečky na systémové názvy.

Budu zde tedy muset vytvořit MDO příkazy pro vyhledání záznamu v tabulce, smazání záznamu a přidání nového záznamu.

5.5.2 Transakce

Pro přístup mobilní aplikace k datům zde budou tři transakce. Jedna zobrazující strom výrobních objektů, druhá zobrazující odběry pro daného uživatele a poslední pro přidání a mazání odběrů.

Pro práci s tabulkou odběratelů tu bude transakce, která bude sloužit k vytváření a mazání záznamů. Vstupem se jí předá informace o záznamu a o tom, jestli jej má systém smazat nebo vložit. Transakce na serveru potom zavolá patřičný MDO příkaz v závislosti na operaci zvolené vstupním jejím parametrem.

Na serveru SAP MII se již počítají KPI pro výkon a kvalitu, a to sice každou hodinu. Proto budu využívat tato data. Vezmu již vypočítané hodnoty pro sedm posledních hodin a k tomu dopočítám hodnotu pro aktuální čas, počítáno za posledních 60 minut. V transakci potom budou vypočítány tyto hodnoty včetně názvu, popisu, cílové hodnoty a hodnoty, při které dojde k upozornění. Tohle vše pro všechny KPI, které odebírá uživatel, který transakci zavolá.

Poslední transakce bude sloužit k zobrazení stromu výrobních linek. Tato transakce dostane na vstup jméno nadřazeného objektu (kořene pod-stromu) a vrátí všechny objekty, které jsou ve stromu přímo pod tímto kořenem. Transakce vrátí kolekci všech těchto objektů, obsahující jejich název, popis a informaci o tom, zdali ji již uživatel odebírá, či nikoli. Zde tedy bude výstup taky závislý na uživateli, který transakci volá.

Kapitola 6

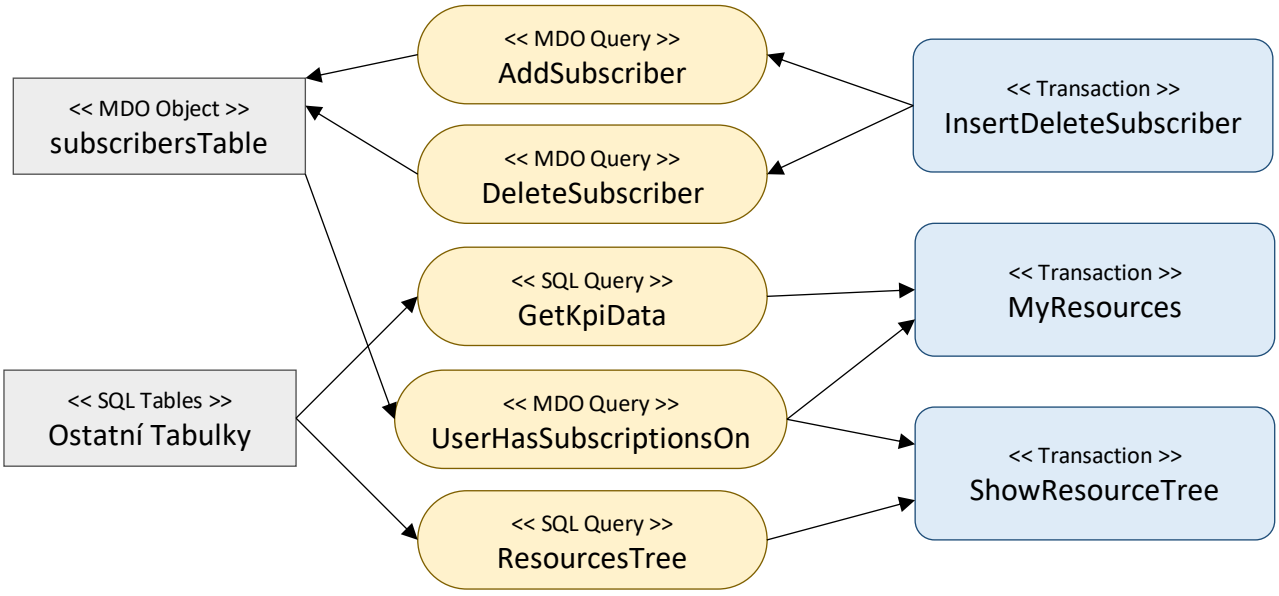
Implementace

V této kapitole nejdříve popíši serverovou část mobilní aplikace, kde si připravím všechna potřebná data, a až potom samotnou mobilní aplikaci, která data načte a zobrazí.

6.1 Implementace na straně serveru SAP ME

Na straně SAP ME se programuje pomocí Java aplikace „*SAP MII Workbench*“. Jedná se o nástroj, který programátorovi umožňuje vytvářet, nastavovat a spouštět transakce, MDO (*Manufacturing data objects*), různé databázové dotazy, atd.

Databáze, které jsou v systému SAP MII, jsou relační SQL databáze. Proto se pro práci s nimi využívá běžných SQL příkazů. Nicméně struktura těchto databází je již pevně daná. V případě, že potřebujeme ukládat a pracovat s vlastními daty, tak k tomu potřebujeme MDO objekty a pro přístup k jejím datům vytvářet MDO dotazy.

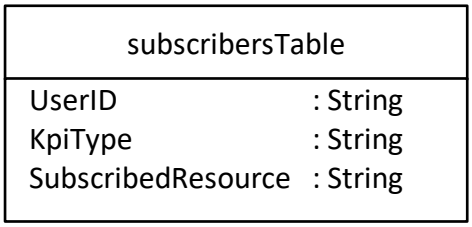


Obrázek 11: Diagram toku dat mezi objekty v systému SAP MII (autor: Marek Skalník)

V aplikaci na straně serveru tedy vytvořím několik databázových dotazů, kterými získám data a následně tato data zpracuji v transakcích a pošlu ke klientovi do mobilní aplikace. V diagramu na Obrázek 11 zobrazuji vytvořené objekty (jako jsou MDO, databázové dotazy a transakce) a toky dat mezi nimi. Blokem „Ostatní tabulky“ (v Obrázek 11) máme na mysli tabulky, které se již v systému SAP MII nacházely, a ze kterých pouze získáváme data. Pro jednoduchost je nerozepisuji podrobně.

6.1.1 Perzistentní data

Na straně serveru je potřeba uložit informace o tom, který uživatel odebírá kterou výrobní linku. Na ukládání perzistentních dat v prostředí SAP MII budou využity MDO. Na Obrázek 12 je zobrazen tento objekt jako tabulka se třemi sloupečky. Uživatel odebírá dané KPI na dané lince v případě, že se v tomto objektu bude nacházet daná kombinace uživatele, výrobního objektu a typu KPI.



Obrázek 12: Diagram zobrazující tabulku uchovávající data o odběratelích (autor: Marek Skalník)

- **UserID** – Je označení uživatele. Každý uživatel, který má přístup do systému SAP MII má svoje přihlašovací jméno. Tohle uživatelské jméno bude použito na jednoznačné označení uživatele.
- **SubscribedResource** – Jedná se o označení výrobního objektu (výrobní buňky, *Resource*), který uživatel odebírá. Každý takový objekt má v SAPu svůj jednoznačný identifikátor. Tento identifikátor je typu textový řetězec.
- **KpiType** – Textové označení typu KPI na daném objektu. Aby bylo přesně rozeznatelné, které KPI je na daném objektu měřeno, bude zde jeho unikátní slovní označení. Existují tři typy KPI, které budou měřeny:
    - „yield“ – Označení pro KPI typu kvalita
    - „perform“ – označení pro typ výkonu
    - „avail“ – označení pro typ dostupnosti

Každý identifikátor v SAPu (viz Obrázek 13) ve vestavěných tabulkách je textového typu a skládá se ze tří částí. První část je název tabulky, zakončen písmeny „BO“[11], dále následuje za dvojtečkou číslo označující areál (číselné označení budovy), ke kterému se daný objekt váže, a za čárkou následuje název daného objektu. Díky tomu je zajištěno, že pole *SubscribedResource* (z Obrázek 12) bude vždy odkazovat na unikátní objekt.

```
ResourceBO:5500,PX42 RS020
```

Obrázek 13: Ukázka identifikátoru v systému SAP pro objekt typu „resource“ v budově číslo 5500 s označením „PX42 RS020“

6.1.2 Databázové dotazy

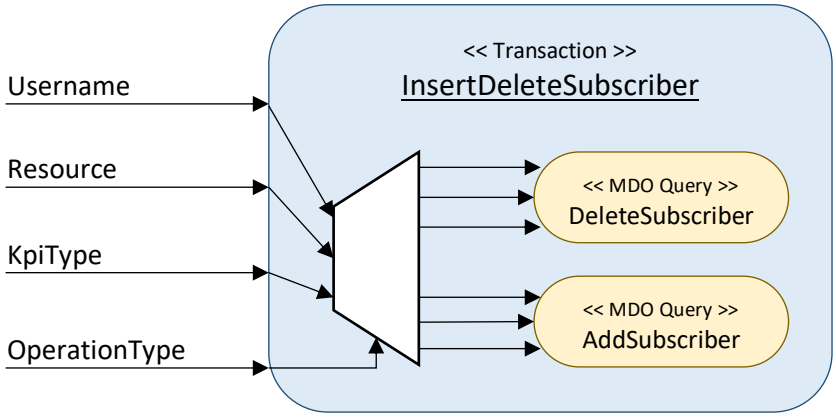
K tomu, aby bylo možné v systému SAP MII přistupovat k datům, je potřeba vytvořit objekty reprezentující databázové dotazy. Tyto dotazy jsou potom staticky uložené v systému, a mohou být volány například v transakcích. K tomu, aby dotazy pracovaly s různými daty, u nich slouží možnost nastavení vstupních parametrů, kterými lze potom data filtrovat. Dotazy mohou sloužit jak pro ukládání dat, tak i pro jejich vytváření a mazání. Vstupem do databázového dotazu může být několik skalárních hodnot, a pokud má dotaz výstup, tak je to tabulka s několika sloupečky reprezentována XML datovou strukturou.

Databázové dotazy v rámci této aplikace jsou zobrazeny na Obrázek 11 žlutě. Jsou to tři MDO dotazy pro práci s vytvořeným MDO objektem *subscribersTable*, jeden pro zobrazení dat, druhý pro přidání řádku a třetí pro smazání řádku. Dále jsou zde dva SQL příkazy, první sloužící pro získání dat reprezentujících strom výrobních objektů a druhý pro zobrazení všech dostupných KPI hodnot na všech výrobních objektech.

[11] BO je zkratka pro Bussiness Object.

6.1.3 Transakce – vložení / smazání záznamu odběratele

Transakce jsou funkční bloky v systému SAP MII, které mohou mít nějaký vstup a nějaký výstup. Tyto transakce lze volat z internetu a lze pomocí nich získávat a manipulovat s daty na SAP MII serveru.



Obrázek 14: Diagram transakce v systému SAP MII, která přijímá 4 parametry a na základě jednoho z nich (OperationType) buď přidá, nebo smaže záznam z MDO. (autor: Marek Skalník)

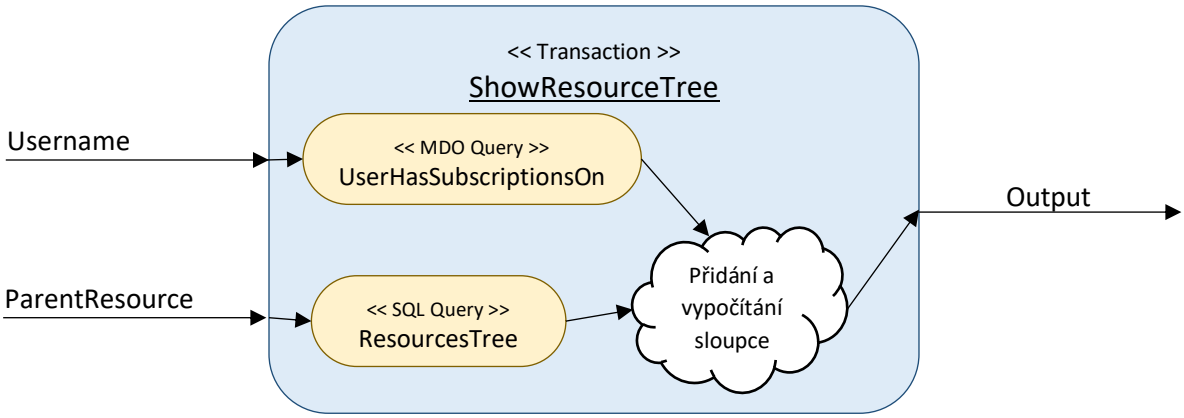
Na Obrázek 14 je zobrazena transakce sloužící pro vložení nebo smazání záznamu z MDO objektu *subscribersTable*. Tato transakce přijímá na vstupu 4 parametry. Parametry *Username*, *Resource* a *KpiType* slouží pro jednoznačné identifikování řádku tabulky *subscribersTable*. Další parametr *OperationType* slouží pro výběr operace a může nabývat dvou hodnot typu textový řetězec:

- „insert“ – pokud je hodnota parametru *OperationType insert*, zavolá se databázový dotaz pro vložení záznamu do tabulky odběratelů.
- „delete“ – pokud je jeho hodnota *delete*, zavolá se databázový dotaz pro smazání záznamu z tabulky odběratelů.

Tato transakce nemá žádný výstup, pouze aktualizuje tabulku odběratelů.

6.1.4 Transakce – Zobrazení stromu výrobních zařízení

Další transakce slouží pro načtení stromu výrobních linek. Vstupem transakce je vždy kořenový uzel, a výstupem bude seznam linek bezprostředně pod kořenovým uzlem. Kořenový uzel na nejvyšší úrovní má pevně dané jméno „ROOT“. Tato transakce je zobrazena na Obrázek 15. Dalším vstupním parametrem je i jméno uživatele, aby bylo možné ke každému výstupnímu výrobnímu zařízení dopočítat sloupec říkající, jestli již uživatel dané zařízení odebírá.



Obrázek 15: Transakce sloužící pro zobrazení výrobních zařízení, bezprostředně podřízených pod uzlem určeným vstupním parametrem ParentResource (autor: Marek Skalník)

Výstupem databázových dotazů uvnitř transakce jsou XML datové struktury. Transakce dále nabízí několik XML funkcí, které je dokáží dále zpracovávat. Transakce *ShowResourceTree* načte tabulku všech objektů a na nich dostupných KPI náležících bezprostředně pod uzel označený vstupním parametrem *ParentResource*. Dále transakce načte uživatelem odebírané KPI. Tyto dvě tabulky sloučí a vypočítá sloupec, jehož hodnota určí, jestli již uživatel dané KPI na daném objektu odebírá.

Na Obrázek 16 je příklad konkrétního výstupu z transakce *ShowResourceTree*. Výstupem je XML, obsahující několik řádků – záznamů. Každý tento záznam představuje jeden výrobní objekt s těmito údaji:

- **PARENTRESOURCE –** Je identifikátor nadřazeného výrobního objektu pro aktuální záznam. Identifikátor obsahuje za čárkou název tohoto objektu.
- **RESOURCE –** Je identifikátor aktuálního výrobního objektu na řádku.
- **DESCRIPTION –** Je popis výrobního objektu.
- **KPITYPE –** Jedná se o textový řetězec označující typ KPI, který je měřený na daném objektu (viz kapitola 6.1.1 Perzistentní data).
- **ISSUBSCRIBED** – Je celé číslo s hodnotou 1 nebo 0, kde 1 označuje, že uživatel již odebírá dané KPI na daném zařízení a 0 znamená, že jej neodebírá.

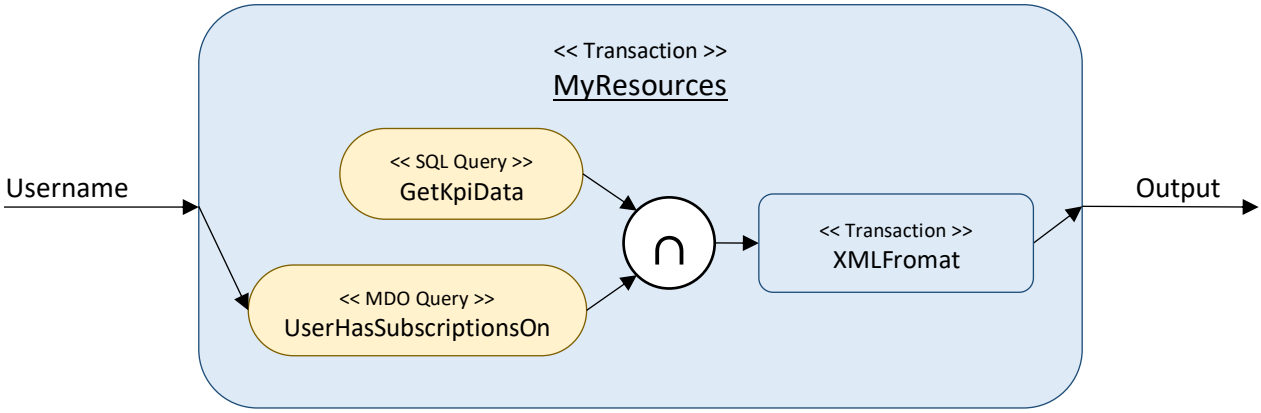
```
<Rowsets>
    <Rowset>
        <Row>
            <PARENTRESOURCE>WorkCenterBO:5500,2276</PARENTRESOURCE>
            <RESOURCE>ResourceBO:5500,2276_RS023</RESOURCE>
            <DESCRIPTION>PX42-OP023 Packing</DESCRIPTION>
            <KPITYPE>yield</KPITYPE>
            <ISSUBSCRIBED>0</ISSUBSCRIBED>
        </Row>
        <Row>
            <PARENTRESOURCE>WorkCenterBO:5500,2276</PARENTRESOURCE>
            <RESOURCE>ResourceBO:5500,PX42 RS011</RESOURCE>
            <DESCRIPTION>PX42-OP011 Ploining</DESCRIPTION>
            <KPITYPE>yield</KPITYPE>
            <ISSUBSCRIBED>1</ISSUBSCRIBED>
        </Row>
    </Rowset>
</Rowsets>
```

Obrázek 16: Příklad XML výstupu z transakce o dvou záznamech. Každý záznam obsahuje jméno nadřazeného uzlu, identifikátor záznamu, jeho popis, typ KPI, a označení, jestli již uživatel tento záznam odebírá. (autor: Marek Skalník)

6.1.5 Transakce – zobrazení aktuálních uživatelem odebíraných KPI

Poslední transakce, znázorněná na Obrázek 17, vrátí seznam uživatelem odebíraných výrobních objektů. Transakce zavolá databázový dotaz *GetKpiData*, který mu vrátí jeden řádek pro každou hodnotu KPI v určitou hodinu a na určitém výrobním objektu. Dále transakce načte uživatelem odebírané výrobní objekty a jejich průnikem získá aktuální vypočítané hodnoty KPI pro objekty, které uživatel odebírá.

Tato data obsahují mnoho redundance. Každý řádek obsahuje jak informace pro daný výrobní objekt, tak informace pro aktuální vypočítanou hodnotu KPI. To je 8 řádků pro každý výrobní objekt. Proto je zde pomocná transakce, která sloučí redundantní data a upraví výstupní formát XML.



Obrázek 17: Transakce sloužící pro vypsání všech objektů, které odebírá uživatel určený vstupním parametrem Username. Každý tento vrácený objekt má u sebe dále 8 hodnot KPI pro každou předešlou hodinu. (autor: Marek Skalník)

Příklad XML výstupu transakce na Obrázek 18 ukazuje výstup s jedním výrobním objektem, a pro jednoduchost jsou u něj dvě vypočítané hodnoty KPI. Výstup obsahuje následující hodnoty:

- **RESOURCE –** Jedná se o identifikátor výrobního objektu.
- **DESCRIPTION –** Jedná se o popis výrobního objektu.
- **KPITYPE –** Je textový řetězec udávající typ KPI měřený na daném objektu.
- **ALERTVALUE –** Je desetinné číslo. Pokud bude mít poslední naměřená hodnota KPI hodnotu menší než je hodnota ALERTVALUE, tak uživatel dostane upozornění.
- **TARGETVALUE –** Opět se jedná o desetinné číslo. Označuje cílovou hodnotu, kterou má dané KPI dosáhnout.
- **KPIVALUES –** Jedná se o podřazenou tabulku obsahující samotné hodnoty vypočítaného KPI. Ta dále obsahuje další dva sloupečky:
    - **DATETIME –** Čas, pro kterou hodinu tato hodnota KPI platí. Obsahuje celé datum a čas, oddělené písmenem „T“.
    - **VALUE –** Samotná hodnota vypočítaného KPI, desetinné číslo.

```
<Rowsets>
    <Rowset>
        <Rows>
            <Row>
                <RESOURCE>ResourceBO:5500,PX42 RS022</RESOURCE>
                <DESCRIPTION>PX42-OP022 Cap screw</DESCRIPTION>
                <KPITYPE>yield</KPITYPE>
                <ALERTVALUE>0.88</ALERTVALUE>
                <TARGETVALUE>0.95</TARGETVALUE>
                <KPIVALUES>
                    <KPIVALUE>
                        <DATETIME>2019-03-25T08:00:00</DATETIME>
                        <VALUE>0.8</VALUE>
                    </KPIVALUE>
                    <KPIVALUE>
                        <DATETIME>2019-03-25T09:00:00</DATETIME>
                        <VALUE>0.86</VALUE>
                    </KPIVALUE>
                </KPIVALUES>
            </Row>
        </Rows>
    </Rowset>
</Rowsets>
```

Obrázek 18: Příklad XML výstupu transakce MyResources. Na příkladu je výstup s jedním výrobním objektem, a dvěma posledními vypočítanými hodnotami KPI pro typ kvalitu. (autor: Marek Skalník)

6.2 Implementace mobilní aplikace

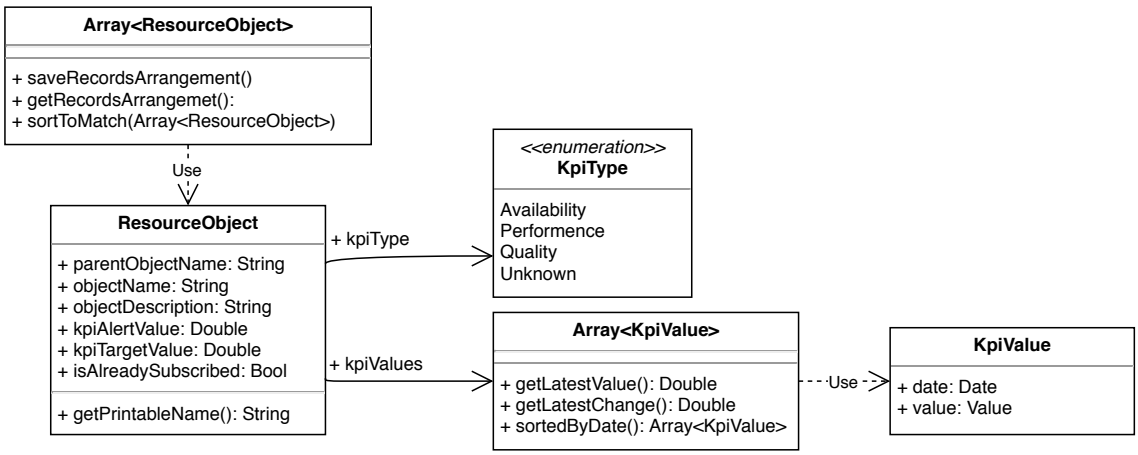
Mobilní aplikace je implementována architekturou MVC (model, view, controller). Skládá se tedy ze tří částí. Datová část jsou třídy mající na starost data aplikace. Uživatelské rozhraní je návrh

obrazovek mobilní aplikace, vytvořený pomocí systému storyboards a tyto dvě části jsou vzájemně propojené kontrolérem, který je řídí a předává mezi nimi data.

6.2.1 Vrstva modelu mobilní aplikace

Vrstva modelu mobilní aplikace se stará o zacházení s daty aplikace a patří sem objektový model dat. Dále je v této vrstvě logika zajišťující načítání dat ze serveru SAP MII, pomocné funkce pro běžné úkony používané různě po aplikaci, perzistence dat a v neposlední řadě i bezpečný způsob uložení uživatelského jména a hesla.

Pro uchovávání hesel v aplikacích na zařízeních se systémem iOS slouží klíčenka. Jedná se o databázi pro ukládání citlivých údajů a hesel. Pro zacházení a používání klíčenky slouží API od společnosti Apple, které taky používám. Uživatel se musí při prvním přihlášení do aplikace přihlásit svými uživatelskými údaji k serveru SAP MII ve firmě Resideo. Při vyplnění uživatelských údajů jsou jméno a heslo uloženy pomocí klíčenky tak, aby se k heslu mohla dostat příště jen samotná aplikace. Uživatelské údaje jsou potom potřeba na to, aby se vyplnily do každého SOAP požadavku směřujícího na SEP MII server.



Obrázek 19: Diagram tříd popisující uložení dat v mobilní aplikaci. Jako první ukazuje pole objektů typu ResourceObject, což slouží pro uložení výrobních linek, dále pole KpiValues sloužící pro ukládání hodnot KPI pro jednotlivé hodiny a nakonec i výčtový typ určující typ KPI na dané výrobní lince. (autor: Marek Skalník)

Na Obrázek 19 je zobrazen diagram tříd, popisující strukturu uložení dat v aplikaci. Jako první ukazuje pole *Array<ResourceObject>* pro uložení výrobních objektů. Objekty této třídy jsou načítány webovou službou a jejich pořadí může být libovolné. Proto jsou zde tři metody, jedna sloužící pro uložení aktuálního seřazení výrobních objektů v poli do paměti telefonu, druhá metoda slouží pro nahrání tohoto pořadí a poslední metoda slouží pro seřazení aktuální skupiny výrobních objektů tak, aby odpovídala seřazení jiného pole. Použití tedy bude takové, načtu uložené pořadí z paměti telefonu a aplikuji toto pořadí na výrobní objekty získané ze serveru, které mohou být v libovolném pořadí.

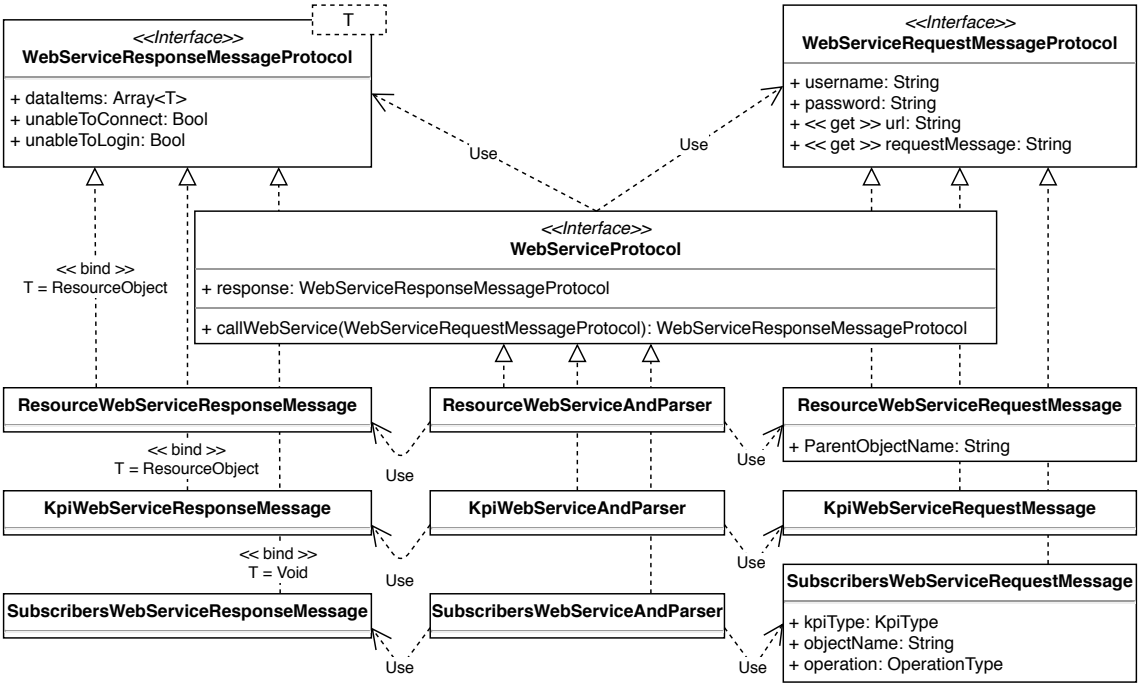
Dále je na Obrázek 19 zobrazen výrobní objekt *ResourceObject*. Jeden výrobní objekt obsahuje informace o typu KPI, který je na něm měřen určený výčtovým typem, seznam hodinových hodnot vypočítaného KPI a následující parametry:

- **parentObjectName –** Jde o identifikátor ze systému SAP nadřazeného výrobního objektu.
- **objectName –** Identifikátor aktuálního výrobního objektu. Tento identifikátor slouží současně i jako název. Protože identifikátory v systému SAP obsahují mnoho textu určujícího, k jaké tabulce se objekt váže (viz Obrázek 13), tak zde je metoda *getPrintableName()*, která vypíše pouze samotný název.
- **objectDescription –** Jedná se o popis výrobního objektu.
- **kpiAlertValue –** Hodnota, pod kterou když se dostane poslední vypočítané KPI, tak by měl uživatel dostat upozornění.
- **kpiTargetValue –** Cílová hodnota KPI, které u objektu chceme dosáhnout.
- **IsAlreadySubscribed –** Udává, jestli již uživatel odebírá tento výrobní objekt s tímto KPI.

Objekt třídy *ResourceObject* nebude muset vždy obsahovat všechny svoje parametry. Například v případě uložení a načtení pořadí objektů stačí, aby měly tyto objekty pouze parametry *objectName* a *kpiType*. Data těchto objektů budou získávána ze serveru skrze webové služby. Tyto webové služby, získají všechny data v textové podobě. Z toho důvodu bude mít tato třída inicializační funkci přijímající všechny její parametry v datovém typu text, což je v programovacím jazyce Swift typ *String*.

Samotné hodnoty KPI pro určitý čas jsou uloženy v poli *Array<KpiValues>*. Zde se nachází metody, pro vyhledání nejaktuálnější hodnoty KPI z pole, metoda pro získání rozdílu posledních dvou hodnot KPI a dále metoda pro seřazení těchto hodnot KPI podle času. Objekty této třídy budou také získávány z webové služby a půjdou proto inicializovat funkcí přijímající textové řetězce a budou obsahovat datum s časem a samotnou hodnotu.

Na Obrázek 20 je diagram tříd ukazující třídy v aplikaci starající se o načítání a zpracování dat ze serveru SAP MII. Třídy zpracovávají XML data přijaté ze serveru a vytváří z nich objekty. Jako hlavní jsou zde ukázaná tři rozhraní. Rozhraní pro požadavek na webový server obsahuje nastavitelné parametry, ze kterých se následně vytvoří webová adresa a požadavek odesílaný na server. Následně odvozená třída od tohoto rozhraní slouží jako vstup pro obecnou metodu *callWebService*. Ta odešle daný SOAP požadavek na danou URL adresu určenou vstupní zprávou. Následná přijatá data formátu XML zpracuje odvozená třída od tohoto protokolu, vytvoří objekty a uloží je do výstupního parametru typu odvozeného od protokolu *WebServiceResponseMessageProtocol*. V případě, že se při pokusu kontaktovat SAP MII server vyskytne chyba, odrazí se to ve výstupu. Jsou zde proto dva samo popisné parametry *unableToConnect* a *unableToLogin* logického typu *Bool*. Od těchto protokolů jsou zde dále odvozené tři třídy pro každou transakci na serveru zvlášť.



Obrázek 20: Diagram tříd znázorňující hierarchii tříd v aplikaci, které mají na starost načítání dat ze serveru SAP MII. (autor: Marek Skalník)

6.2.2 Upozornění na problém

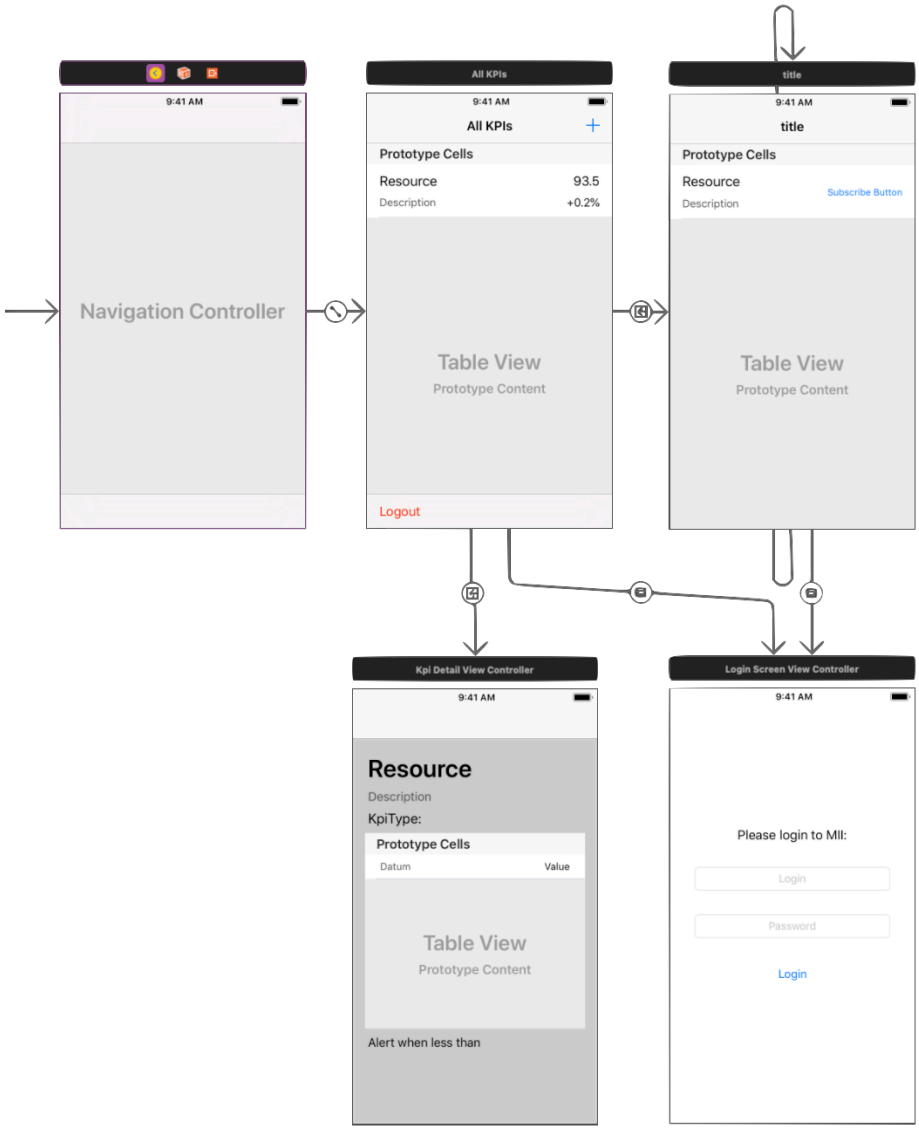
V aplikaci je implementovaný systém notifikací. Z důvodu bezpečnostní politiky na serveru SAP MII v Resideu jsem nemohl využít služeb push notifikací od společnosti Apple. Nicméně k tomu, aby uživatel získával alespoň občasné notifikace využívám služby načítání na pozadí.

Načítání na pozadí (*background fetch*) je funkce systému iOS, která v určitých intervalech probouzí mobilní aplikaci a provádí krátké úseky kódu. Tento systém probouzení má ovšem plně pod kontrolou operační systém mobilního telefonu, a tak nelze zajistit přesné intervaly mezi probouzením aplikace. Když se tak ale stane, aplikace si načte data ze serveru a na mobilním telefonu zkontroluje, jestli jsou všechny odebírané hodnoty KPI v patřičných mezích. Pokud se vyskytne problém, odešle se uživateli notifikace. Tato KPI jsou kontrolována v pořadí, v jakém má na úvodní obrazovce uživatel uložené výrobní objekty. V případě, že je nalezen problém a odeslána notifikace se s kontrolou končí a další notifikace se neodesílají.

6.3 Uživatelské rozhraní

Uživatelské rozhraní se v prostředí iOS skládá z komponent typu storyboard. Jedná se o pohledy, které jsou různě propojené, a mohou se do sebe i zanořovat. Každý pohled náleží jedné obrazovce v mobilní aplikaci. V každém pohledu je dále několik komponent uživatelského rozhraní, jako jsou tlačítka a nápisy. Všechny komponenty v rámci jednoho pohledu mají nastavené omezující podmínky na svoji velikost a pozici pro zajištění podpory různě velkých obrazovek a různé orientace displeje.

Na Obrázek 21 jsou zobrazené obrazovky mobilní aplikace. Jako první se při spuštění zobrazí *Navigation Controller* (na Obrázek 21 úplně vlevo). To je pohled, který zajišťuje horní lištu s ovládacími prvky, a do kterého jsou zanořeny všechny ostatní pohledy. V něm je přímo zanořený pohled zobrazující tabulku všech odebíraných výrobních objektů a jejich hodnot KPI. To je první, co uživatel uvidí při spuštění aplikace. Dole pod ním je zobrazení detailu při výběru řádku tabulky. Zde se nachází i menší tabulka se seznamem KPI hodnot za uplynulých 8 hodin. Úplně vpravo je do sebe zanořený pohled zobrazující jednotlivé vrstvy stromu výrobních objektů. A nakonec vpravo dole obrazovka vyžadující přihlášení.



Obrázek 21: Storyboards použité v aplikaci. Lze zde vidět všechny obrazovky, které jsou v aplikaci dostupné. Autor: (autor: Marek Skalník)

Kapitola 7

Testování

Tato kapitola popisuje obecně testování softwaru a způsob provedeného testování při vývoji aplikace. Je zde popsán způsob provedeného testování serverové části v prostředí SAP MII. Dále je popsána tvorba automatizovaných testů pro mobilní aplikaci a také její manuální testování.

7.1 Testování softwaru

Testování softwaru je součástí jeho vývoje. Jedná se o empirický proces zkoumání kvality softwaru. Zkoumá se, jakým způsobem testovaný software odpovídá jeho prvotním požadavkům a reaguje korektně na všechny vstupy v přijatelný čas. Je to iterativní proces se snahou zachytit všechny tyto případy. Hlavní podstatou je zkoumání, jestli daná situace v programu nastane, nebo nenastane [23].

Samotné testování potom může být manuální, kdy tester musí vše vyzkoušet ručně, nebo automatizované, kdy spuštění sady testů vypíše, jestli se testovaný software choval při testech korektně, nebo jestli a kde nastala chyba [23].

Systém lze testovat několika způsoby, buď s ohledem na jeho implementaci (*white-box testing*), nebo jako celek bez ohledu, jak se chová uvnitř (*black-box testing*). Dále lze testovat celý systém, nebo jen jeho dílčí části. Podle toho lze testování rozdělit na několik úrovní [23]:

- Unit testing – Jedná se o většinou automatizované testování určitých částí kódu jako jsou funkce nebo třídy. Tyto testy by měly být nezávislé na zbytku ostatního kódu.
- Integrační testování – Je proces testování spojení několika komponent. Testy mají zjistit, jestli spolu části kódu komunikují korektně. Jedná se o část testování, která následuje po *Unit testech*.
- Akceptační testování – Je testování celkového systému, jestli odpovídá prvotním požadavkům od zákazníka. Obsahuje jak zkoumání funkčnosti, tak i dobu odezvy a použitelnost.

7.2 Testování v prostředí SAP MII

Vývojové prostředí SAP MII neobsahuje žádné nástroje pro automatizované testování, a proto je třeba vše testovat ručně. Jedná se o transakce a databázové příkazy, které pracují s daty, a to často s těmi z aktuálního stavu výroby. Proto může být těžké rozeznat, že jsou získávaná data korektní.

Transakce na serverové části se skládají z menších bloků, často z databázových příkazů, takže lze tyto části spouštět a testovat jednotlivě. Proto všechny databázové dotazy byly testovány tímto způsobem. Samotné transakce obsahují možnost jejich ladění. Transakci lze spustit v režimu ladění a sledovat krok za krokem její průběh a lokální vlastnosti a proměnné. Největší problém činilo testování dotazů, které obsahovaly aktuální čas. Z důvodu, že server SAP MII firmy Resideo není ve středoevropském časovém pásmu, musejí zde být konverze časů, které nešly testovat v aplikaci SAP MII Workbench[12]. Tyto transakce tedy byly testované jejich přímým voláním ze serveru.

Další věc, na kterou musel být kladen důraz při testování, byla doba vykonání transakcí. Databázové dotazy, ze kterých čerpají data, mají jako svůj zdroj velké produkční databáze. Proto musely být tyto dotazy psány a testovány s ohledem na výkonnost. Muselo zde dojít k co největšímu omezení typových konverzí mezi parametry v dotazech.

7.3 Testování mobilní aplikace

Mobilní aplikace byla testována částečně automaticky a z části manuálně. Část modelu aplikace je pokryta unit testy, jejichž tvorbu umožňuje vývojové prostředí Xcode. Uživatelské rozhraní a funkce *controlleru* byly testovány převážně manuálně.

7.3.1 Unit testy

Unit testy, které byly v aplikaci použity testují korektní chování tříd modelu. Tyto třídy jsou zobrazeny na Obrázek 19. Jedná se o automatizované testy, které zkouší různé kombinace vstupů pro metody těchto tříd a kontrolují jejich korektní chování.

Dále zde testuji funkce na ukládání mazání a čtení uživatelských jmen a hesel z klíčenky. U těchto testů na začátku přečtu aktuální přihlašovací údaje z klíčenky, potom provedu samotné testování a na konci do klíčenky opět uložím původní údaje. Při selhání těchto testů tedy pravděpodobně nebudou fungovat ani funkce pro práci s klíčenkou a nedojde tak ke znovuzapsání uživatelských údajů do klíčenky, což způsobí smazání uživatelských údajů uživatele.

Dále pomocí unit testů testuji třídy pro získávání dat ze serveru. O které třídy se jedná, je zobrazeno na Obrázek 20. Tyto třídy dělají dvě věci: načítají data ze serveru ve formátu XML a dále je zpracovávají a získávají z nich hodnoty. Proto jedna sada testů kontroluje komunikaci těchto tříd se serverem a druhá zpracování dat. Testování komunikace probíhá odesláním požadavků na server a dále zkoumáním, jestli server vrátil nějakou odpověď. Jsou to testy se

[12] Vývojové prostředí pro SAP MII.

špatnými přihlašovacími údaji a s korektními údaji. Při testování s korektními přihlašovacími údaji musí být tyto údaje před spuštěním testu uloženy v klíčence mobilního zařízení. Ty se odešlou na server a čeká se typická odpověď. Zde se nijak nekontrolují konkrétní přijatá data. Při testování zpracování dat aplikace nijak nekomunikuje se SAP MII serverem, ale obdrží falešnou odpověď s předem připravenými XML soubory. Testované třídy tyto XML data zpracují a vytvoří z nich odpovídající datové struktury. Ty jsou potom v unit testech kontrolovány, jestli odpovídají datům ze statických XML.

7.3.2 Testování uživatelského rozhraní

Pro otestování korektního chování uživatelského rozhraní mé aplikace jsem chtěl využít možnosti tvorby UI testů v prostředí Xcode a kontrolovaného spouštění metod v třídách *controlleru*. Chtěl jsem vytvořit UI testy, které by kontrolovaly jednotlivé komponenty obrazovky z hlediska jejich viditelnosti, responzivity a korektní funkcionality. Bohužel se mi dané třídy *controlleru* nepodařily zakomponovat do mého řešení tak, aby při testech nezískávaly data ze serveru SAP MII. Díky tomu jsou všechna data na obrazovce závislá na aktuálním stavu dat na serveru, a tak jsem nedokázal pokrýt tyto případy pomocí automatizovaných testů.

Při manuálním testování lze mít telefon připojený k počítači a ve vývojovém prostředí Xcode sledovat stav prostředků telefonu, jako je paměť a vytížení procesoru. Dále lze sledovat kontrolní výstupy aplikace a sledovat volané funkce a metody. Proto jsem aplikaci testoval tímto způsobem a snažil se, aby bylo v rámci tohoto testování vykonáno co největší množství řádků v kódu.

Kapitola 8

Závěr

Ve firmě Resideo Brno jsou různé výrobní linky komunikující s výrobním informačním systémem SAP MII. Těchto výrobních linek je velké množství a obsluhují je různí pracovníci. V případě, že na výrobní lince vznikne problém, neexistuje způsob, jak rychle kontaktovat osoby, které tento problém může zajímat. Zároveň tyto osoby, odpovědné například za plánování výroby, nijak nevidí aktuální stav na výrobních linkách. Z dat, která tyto výrobní linky odesílají na server SAP MII lze vypočítat mnoho ukazatelů, které dostatečně vypovídají o aktuálním dění.

V této diplomové práci jsem popsal aktuální problém ve firmě Resideo s informováním patřičné osoby o problému na výrobních linkách. Popisuji zde funkci systému SAP ME pro monitorování a řízení výrobních zařízení a její použití ve firmě. Jeho možnost získat přehled o aktuálním stavu výroby a případných nedostatků. Dále zde popisuji možnost rychlé notifikace uživatele prostřednictvím mobilního telefonu. Rozebírám aktuální stav mobilních telefonů na trhu a jejich možnosti programování. Nakonec popisuji návrh, implementaci a testování mobilní aplikace a transakcí na serveru SAP MII, skrze které bude aplikace získávat svá data.

Mobilní aplikace umožňuje zobrazení KPI ukazatelů, jejich průběhu v čase a při příliš nízkých hodnotách zasílá notifikace na mobilní telefon. Je zde použit systém lokálních notifikací, takže aplikace sama si v určitých intervalech stahuje aktuální data ze serveru a kontroluje, jestli nedošlo k problému. To pomůže lidem, jako jsou plánovači výroby, vidět ihned aktuální stav na výrobních linkách a rychle na něj reagovat.

Možné pokračování a rozšíření této práce může spočívat hlavně v implementaci push-notifikací, aby aplikace nemusela běžet na telefonu na pozadí a současně aby bylo informování o problému okamžité. K tomu je ovšem potřeba získat bezpečnostní povolení komunikovat ze serveru SAP MII na servery společnosti Apple. Dále lze rozšířit možné KPI ukazatele, které bude aplikace zobrazovat, a přidat možnost si více přizpůsobovat notifikace, například nastavením hodnoty, na kterou chce být uživatel upozorněn. Aktuálně je tato hodnota pevně daná.

Literatura

[1] PHILLIPS, Bill, Chris STEWART a Kristin MARSICANO. *Android Programming: The Big Nerd Ranch Guide*. 3rd edition. Atlanta (Georgie): Big Nerd Ranch, 2017. ISBN 978-0134706054.

[2] *Vzhůru do (responzivního) webdesignu*. Praha: Martin Michálek, 2018. ISBN 978-80-88253-00-6.

[3] KEUR, Christian a Aaron HILLEGASS. *IOS Programming: The Big Nerd Ranch Guide*. 6th edition. Atlanta (Georgie): Big Nerd Ranch, 2017. ISBN 978-0134682334.

[4] Android 9 Pie. *Android* [online]. Paříž: Astrium, 2014 [cit. 2018-12-29]. Dostupné z: https://www.android.com/versions/pie-9-0/

[5] Progressive Web Apps. *Google Developers* [online]. San Francisco: Google, 2017 [cit. 2018-12-29]. Dostupné z: https://developers.google.com/web/progressive-web-apps/

[6] Apple Developer Program. *Apple Developer* [online]. California: Apple, 2018 [cit. 2018-12-29]. Dostupné z: https://developer.apple.com/programs/

[7] Swift. *Apple Developer* [online]. California: Apple, 2018 [cit. 2018-12-29]. Dostupné z: https://developer.apple.com/swift/

[8] Notifications. *Apple Developer* [online]. California: Apple, 2018 [cit. 2018-12-29]. Dostupné z: https://developer.apple.com/notifications/

[9] Co je ERP. *Oracle Česká Republika* [online]. Redwood Shores (California): Oracle Corporation, 2018 [cit. 2018-12-29]. Dostupné z: https://www.oracle.com/cz/applications/erp/what-is-erp.html

[10] SAP History. *SAP Software Solutions* [online]. Palo Alto (California): SAP, 2018, 2017 [cit. 2018-12-29]. Dostupné z: https://www.sap.com/corporate/en/company/history.html

[11] HEJTMÁNEK, Jakub. Architektury a SAP NetWeaver. *Systémová integrace 3/2004*. 2004, **11**(3), 13. ISSN 1210-9479.

[12] SAP PP (Production Planning) Training Tutorial. *Meet Guru99* [online]. Ahmedabad (India): Guru99 Tech, 2016 [cit. 2018-12-29]. Dostupné z: https://www.guru99.com/sap-pp-tutorials.html

[13] CUPEK, Rafal, Adam ZIEBINSKI, Lukasz HUCZALA a Huseyin ERDOGAN. Agent-based manufacturing execution systems for short-series production scheduling. In: *Computers in Industry* [online]. Volume 82. Amsterdam (Nizozemsko): Elsevier, 2016, 2016, s. 13 [cit. 2018-12-29]. ISSN 0166-3615. Dostupné z: https://www.sciencedirect.com/journal/computers-in-industry

[14] Manufacturing Execution System. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2018 [cit. 2018-12-29]. Dostupné z: https://de.wikipedia.org/wiki/Manufacturing_Execution_System#cite_note-18

[15] CHANDAN, Jash a Saha DIPANKAR. *Implementing SAP Manufacturing Execution*. Boston: Rheinwerk Publishing, 2015. ISBN 978-1-4932-1239-2.

[16] MII Manufacturing Data Objects (MDO) Guide. In: *SAP Software Solutions* [online]. Palo Alto (California): SAP, 2018, 2012 [cit. 2018-12-29]. Dostupné z: https://archive.sap.com/documents/docs/DOC-36230

[17] HUSAIN, Musarrat. Step by Step Guides for Creating Transactions in xMII. In: *SAP Software Solutions* [online]. Palo Alto (California): SAP, 2018, 2007 [cit. 2018-12-29]. Dostupné z: https://archive.sap.com/documents/docs/DOC-1749

[18] Web Services Architecture. *World Wide Web Consortium* [online]. Cambridge (Massachusetts): W3C, 2004 [cit. 2018-12-29]. Dostupné z: https://www.w3.org/TR/ws-arch/

[19] RODRIGUEZ, Alex. RESTful Web services. *IBM Developer* [online]. Armonk (New York): IBM's Corporate Privacy Office, 2018, 2008 - 2015 [cit. 2018-12-29]. Dostupné z: https://developer.ibm.com/articles/ws-restful/

[20] NOLAN, Dennis P. a Eric T. ANDERSON. OE/SHE Key Performance Indicators (KPIs). *Applied Operational Excellence for the Oil, Gas, and Process Industries*. Waltham (Massachusetts): Elsevier, 2015, s. 147-163. ISBN 978-0-12-802788-2.

[21] BRANDL, Dennis L. a Douglas BRAND. KPI Exchanges in Smart Manufacturing using KPI-ML. IFAC-PapersOnLine. 2018, 2018(33), 31-35. DOI: https://doi.org/10.1016/j.ifacol.2018.08.230. ISSN 2405-8963.

[22] Calculating the Rolled Throughput Yield. WEBBER, Larry a Michael WALLACE. *Quality Control for Dummies*. Hoboken (New Jersey): John Wiley, 2011, s. 300. ISBN 978-0-470-06909-7.

[23] J. MYERS, Glenford, Corey SANDLER a Tom BADGETT. *The art of software testing*. 3rd ed. Hoboken, New Jersey.: John Wiley, 2012. ISBN 978-1-118-13313-2.

[24] Kotlin (programming language). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2019-04-10]. Dostupné z: https://en.wikipedia.org/wiki/Kotlin_(programming_language)

Příloha A

Seznam použitých zkratek

APNs	Apple Push Notification service
BOM	Bill of material (kusovník)
ERP	Enterprise resource planning (plánování podnikových zdrojů)
FPY	First Pass Yield (ukazatel kvality)
HTTP	Hypertext Transfer Protocol
KPI	Key performance indicator (klíčový ukazatel výkonnosti)
MDO	Manufacturing Data Object
MES	Manufacturing Execution System (Výrobní informační systém)
MVC	Model-View-Controller
OEE	Overall Equipment Effectiveness (celková efektivita výroby)
REST	Representational State Transfer (webová služba)
RTY	Roll Throughput Yield (ukazatel kvality)
SAP	Systeme, Anwendungen und Produkte in der Datenverarbeitung (německá firma)
SAP ME	SAP Manufacturing Execution
SAP MII	SAP Manufacturing Intelligence and Integration
SAP PP	SAP Product Planning
SOAP	Simple Object Access Protocol (webová služba)
SQL	Structured Query Language (databázový dotazovací jazyk)
WIP	Work in Process
WSDL	Web Services Description Language (jazyk pro popis webových služeb)
XML	Extensible Markup Language (jazyk pro snadné zpracování počítačem)

Příloha B

Uživatelská příručka

Instalace

Instalace aplikace se provádí na dvou místech – na serveru SAP MII a na mobilním telefonu u klienta.

Na serveru je potřeba importovat transakce. Protože transakce spolupracují s ostatními objekty, které se již na serveru SAP MII v Resideu nachází, je tato instalace možná pouze na firemním serveru. Zdrojové soubory k importu transakcí, objektů a databázových dotazů jsou ve složce „Z_HON_iOS“.

Instalace na mobilní telefony bude probíhat skrze ad-hoc distribuci aplikací na telefony s operačním systémem iOS. Uživatel musí mít certifikát vývojáře od společnosti Apple, aby takovouto distribuci mohl použít. Dále je zde možnost aplikaci nainstalovat skrze vývojové prostředí Xcode, které při přeložení aplikace generuje dočasný sedmidenní certifikát pro aplikaci, a lze ji tak mít na svém telefonu nainstalovanou do vypršení certifikátu.

Použití aplikace

Při zapnutí aplikace je třeba se přihlásit uživatelskými údaji k serveru SAP MII v Resideu. Dále se na úvodní obrazovce zobrazuje seznam odebíraných výrobních linek a jejich nejaktuálnější hodnoty KPI. Při kliknutí na některý záznam se zobrazí detailní informace s tabulkou posledních naměřených časů.

Tlačítkem vlevo nahoře „Upravit“ lze upravit tento seznam, některé záznamy smazat (což zruší odběr), změnit jejich pořadí (což změní prioritu pro oznámení), nebo se z aplikace odhlásit.

V pravém horním rohu úvodní obrazovky je tlačítko označené symbolem plus. Touto možností se uživatel dostane na hierarchický seznam výrobních linek s možností přihlásit či zrušit odběr k danému KPI u těch linek, u kterých to je možné.

Je možné, že některé odebírané výrobní linky nejsou zobrazeny na úvodní obrazovce. K takové situaci dojde v případě, že na dané výrobní lince nebyla za posledních 8 hodin prováděna žádná činnost.

Příloha C

Obsah přiloženého CD

`./MyKPIApp/`	Složka s iOS aplikací
`./MyKPIApp.xcodeproj`	Projekt Xcode
`./MyKPIApp/`	Zdrojové soubory
`./MyKPIAppTests/`	Zdrojové soubory k testům
`./Z_HON_iOS/`	Složka projektu ze SAP MII
`./KPIApp/`	Složka s transakcemi a databázovými příkazy
`./MIIOBJ/`	Složka s MDO Objektem