TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Napojení systému SAP na mobilní zařízení

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Jan Procházka**
Vedoucí práce: RNDr. Klára Císařová, Ph.D.

Liberec 2018

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Connect SAP to mobile devices

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jan Procházka**

Osobní číslo: **M14000067**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Napojení systému SAP na mobilní zařízení**

Zadávající katedra: **Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se se systémem SAP a jeho programovacím jazykem ABAP.
2. Analyzujte "místa" systému SAP vhodné pro export funkcí do mobilních zařízení a prostudujte možnosti vzájemného napojení SAP systému a mobilního zařízení.
3. Vytipované funkce naprogramujte jako mobilní aplikaci nad OS Android včetně uživatelského rozhraní. Aplikace bude napojená na systém SAP a umožní pro vybrané funkce SAP alternativní práci "v terénu".
4. Vybrané řešení pro spojení SAPu a mobilních zařízení zdůvodněte a funkčnost ověřte

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **HASEMAN, William D. a Ross. HIGHTOWER. Mobile development for SAP. Bonn: Galileo Press, 2013. ISBN 978-1-59229-647-7.**

[2] **GOEBELS, Christiane a Denise NEPRAUNIG. SAPUI5: the comprehensive guide. Bonn: Rheinwerk Publishing, 2016. ISBN 978-1-4932-1321-4.**

[3] **BAVARAJU, Anil. SAP Fiori implementation and development. Boston: Rheinwerk Publishing, 2016. ISBN 978-1-4932-1249-1.**

Vedoucí bakalářské práce: **RNDr. Klára Císařová, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2017**

Termín odevzdání bakalářské práce: **14. května 2018**

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

L.S.

Technická univerzita v Liberci - 14 -

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2017

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 11. 5. 2018

Podpis: Bucháčka

Poděkování

Rád bych chtěl touto cestou poděkovat v první řadě své rodině za neustálou morální a finanční podporu po celou dobu mého studia. Dále bych rád poděkoval své přítelkyni za pevné nervy a velkou dávku tolerance, své vedoucí paní RNDr., Kláře Císařové PhD. za odbornou pomoc a značné množství úsilí a energie vložené do vedení mé práce. V neposlední řadě bych rád poděkoval Ing. Lukáši Sýkorovi za možnost pracovat na bakalářské práci v rámci jeho firmy, kde jsem tak získal neocenitelné zkušenosti z praxe do dalšího studia i zaměstnání.

Abstrakt

Mezi systémy ERP v České republice dominuje systém SAP. Používají ho významné firmy jako například Škoda auto, a.s., Johnson Controls, k.s. a Magna Automotive, s.r.o., firmy navázané na automobilový průmysl. Využití SAP výrazně zasahuje i jiné odvětví, např. energetiku ČEZ, a.s., RWE, a.s. telekomunikace T-Mobile a mnoho dalších. Rozsáhlé nasazení systému SAP v ČR a rozvoj mobilních zařízení pro profesionální systémy vedlo k zadání této práce. Cílem bakalářské práce bylo prozkoumat možnosti připojení systému SAP na mobilní zařízení s operačním systémem Android. Součástí práce je vedle teoretického popisu také realizace těchto spojení na předem připravený SAP systém. Pro demonstrační účely budou vyvinuty aplikace s předpokladem jejich následného skutečného využití. Velký důraz byl kladen na jednoduchost a intuitivnost ovládání mobilního klienta pro uživatele.

Klíčová slova:

Vývoj aplikací, SAP, OS Android, webová služba, ABAP, Javascript

Abstract

ERP systems in the Czech Republic dominate the SAP system. It's used by leading companies such as Škoda car, a.s., Johnson Controls, k.s. and Magna Automotive, s.r.o., an automotive firm. The use of SAP significantly affects other industries, such as ČEZ, a.s., RWE, a.s. T-Mobile telecommunication and many more. The extensive deployment of the SAP system in the Czech Republic and the development of mobile devices for professional systems led to the assignment of this work. The goal of this bachelor thesis was to explore possibilities of connecting the SAP system to mobile devices with Android operating system. Part of the thesis is, besides the theoretical description, the realization of these connections to the pre-prepared SAP system. For demonstration purposes, applications will be developed with the assumption of their subsequent actual use. Great emphasis was put on the simplicity and intuitiveness of mobile client control for users.

Keywords:

Application development, SAP, OS Android, Web service, ABAP, Javascript

Obsah

Seznam zkratek 11

Úvod 13

1 Seznámení se systémem SAP (R/3) 14

1.1 ERP systémy 14

1.1.1 Výhody a nevýhody 14

1.1.2 Současná situace 16

1.2 Společnost SAP 16

1.2.1 Historie společnosti SAP 16

1.2.2 SAP ČR 17

1.2.3 Produkty 17

2 Programovací jazyk ABAP 19

2.1 Objekty SAP Repository 19

2.2 Struktura programu 20

2.3 Práce s databází 21

3 Využitelnost mobilních zařízení 24

3.1 Výhody a nevýhody používání mobilních zařízení 26

3.2 Využitelnost v prostředí SAPu 27

4 OS Android 28

4.1 Historie OS Android 28

4.2 Vývoj aplikací 29

4.3 Hlavní objekty 30

5 Možnosti napojení mobilního zařízení s OS Android na systém SAP 32

5.1 Použité technologie 33

5.1.1 SOAP 33

5.1.2 REST 33

5.1.3 OData 33

5.1.4 SAPUI5 34

5.1.5 MVC 34
5.1.6 SAP Fiori 34
6 Zadání mobilních aplikací 36
6.1.1 Timesheet 36
6.1.2 Skladiště 36
6.1.3 Elektroměr 36
7 Realizace pomocí webové služby 37
7.1 Timesheet 37
7.1.1 Datový model 37
7.1.2 Funkční Moduly SAP 39
7.1.3 Webová služba 41
7.2 Skladiště 43
7.3 Elektroměry 44
7.4 Mobilní aplikace 44
7.4.1 Menu 44
7.4.2 Timesheet 44
7.4.3 Skladiště 48
7.4.4 Elektroměr 49
8 Realizace pomocí SAP Fiori 50
8.1 Entity a OData služby 50
8.1.1 Timesheet 50
8.1.2 Skladiště 51
8.1.3 Elektroměry 52
8.2 Struktura a vývoj aplikace 52
8.2.1 Timesheet 53
8.2.2 Skladiště 54
8.2.3 Elektroměry 54
8.3 Zobrazení aplikace v klientu FIORI 54

9 Porovnání obou metod 57
9.1 Stabilita, intuitivnost ovládání a použitelnost 57
9.2 Vývoj 58
Závěr 60
Seznam použitých zdrojů 61
Seznam obrázků 64
Seznam zdrojových kódů 64
Seznam grafů 64
A Zdrojové kódy pro ABAP moduly 65
A.1 ZBPH_WRITE_TIME 65
A.2 ZBPH_HASH_PASSWD 66
A.3 ZBPH_LASTFIVEROWS 67
A.4 ZBPH_READTABLE 68
A.5 ZBPH_ERITE_TIME 69
B Zdrojové kódy pro ABAP REST 70
B.1 GET_ENTITYSET 70
B.2 CREATE 71
C Obsah přiloženého CD 72

Seznam zkratek

ABAP	programovací jazyk pro aplikace společnosti SAP - „Advanced Business Application Programming“
CSS	Cascading Style Sheets
ERP	systém pro plánování podnikových zdrojů – „Enterprise Resource Planning“
GUI	grafické uživatelské rozhraní – „Graphic User Interface“
HELIOS	jméno ERP systému
HTML	Hypertext Markup Language
http	Hypertext Transfer Protocol
ID	označení pro identifikátor
IDE	Integrated Development Environment
IT	Information Technology
JSON	způsob zápisu dat - „JavaScript Object Notation“
K2	jméno ERP systému a stejnojmenné firmy
LPD_CUST	transakce v SAPu
MVC	Model-View-Controller
OOP	objektově orientované programování
OS	operační systém – „Operate System“
PFK	cizí klíč – „Primary Foreign Key“
PHP	Hypertext Preprocessor, původně „Personal Home Page“
PK	primární klíč – „Primary Key“
REST	Representational State Transfer
SAP	název společnosti, často se používá i jako označení jejich ERP systému
SDK	Software Development Kit
SEGW	transakce v SAPu
SICF	transakce v SAPu
SMS	Short message service

SOAP	protokol pro výměnu XML zpráv – „Simple Object Access Protocol“
SQL	Structured Query Language
SRNO	transakce v SAPu
UI	User Interface
URI	Uniform Resource Identifier
URL	„Uniform Resource Locator“
VPN	virtuální privátní síť - „Virtual Private Network“
W3C	„World Wide Web Consorcium“
WSDL	„Web Services Description Language“
XML	obecný značkovací jazyk - „eXtensible Markup Language“

Úvod

V minulosti bylo velmi náročné zpracovávat velké množství dat v rámci jedné firmy, kdy bylo zapotřebí více specializovaných programů a skupin odborných zaměstnanců. Z důvodů zajištění všech potřebných nástrojů a integrace znalostí do jednoho systému byly vyvinuty tzv. ERP systémy. Nástupem těchto systémů došlo k radikální změně přístupu firem k jejich organizaci, zlepšení jejich produktivity spolu se snížením nákladů. Mimo změny přístupu a myšlení je velmi důležitý i technologický pokrok v IT, který poskytuje ERP systémům neustále rostoucí výpočetní výkon a paměťové kapacity.

Vývojem těchto systémů se nyní zabývá velké množství firem po celém světě. Příčinou je lukrativnost tohoto oboru. Nejúspěšnější firmou, která se v tomto oboru stala světovou špičkou, je německá společnost SAP. Pro udržení konkurenceschopnosti je ale zapotřebí, aby firma využívala nové technologie a postupy. V dnešní době, kdy jsou notebooky, chytré telefony a tablety nedílnou součástí běžného života, hledají firmy možnosti, jak tato mobilní zařízení nejlépe využít. Jedná se však o neustále se vyvíjející oblast jak z pohledu koncových zařízení, tak z pohledu standardů, pomocí kterých lze s nimi komunikovat. Z tohoto důvodu je zapotřebí nalézat stále optimálnější cesty propojení.

Jako cíl si proto tato práce dává prozkoumat aktuální stav možností napojení mobilních zařízení na podnikový systém SAP. Následně pak provézt jejich implementaci se závěrečným vyhodnocením.

Z hlediska rozsahu práce byla mobilní zařízení omezena pouze na ta, která běží pod operačním systémem Android. Napojení na další platformy (Apple iOS / Microsoft Windows) by mohlo být vhodným tématem pro následnou diplomovou práci.

1 Seznámení se systémem SAP (R/3)

1.1 ERP systémy

ERP je systém, kterým firma řídí a integruje většinu podnikových oblastí či procesů. ERP je anglickou zkratkou pro **Enterprise Resource Planning.** Tento výraz se dá přeložit jako plánování firemních (podnikových) zdrojů. Podnikovými oblastmi jsou například plánování, zásob, nákup, prodej, marketing, finance, personalistika, výroba atd. To, že jsou tyto oblasti zahrnuty v jednom systému, přináší značné množství výhod. [2, 8, 14]

Při výběru ERP systému jsou dvě možnosti. První je zvolit kvalitní, funkční, a především komplexní hotové řešení. Je nutno si uvědomit, že kvalitní a mnoha zákazníky prověřený systém, umožňuje zefektivnit a standardizovat firemní procesy. [2, 8, 13, 14]

Druhou možností je nechat si rozšířit stávající systém nebo vytvořit úplně nový systém od začátku přesně podle firemních požadavků.

1.1.1 Výhody a nevýhody

Jaká možnost je lepší nelze jednoduše definovat, protože vždy záleží na konkrétním případě a výchozích podmínkách. Stejně tak hledisko ceny a délky implementace se může lišit případ od případu. Proto by před začátkem volby jakéhokoliv ERP systému měla proběhnout důkladná vnitropodniková analýza. Nicméně základní výhody/nevýhody lze obecně shrnout do následujících bodů. [2, 3, 8, 13, 14, 15]

Komplexnost

Největší výhodou je komplexnost systému. Po správném nastavení systému již firma obvykle nepotřebuje žádný další software pro svou správu a udržování hladkého chodu firmy. Takovýto systém tedy dokáže pokrýt požadavky všech oddělení od nákupu, přes výrobu až po prodej. Z tohoto důvodu není firma nucena používat různá softwarová řešení pro každé oddělení a zabývat se předáváním dat, mezi jednotlivými softwary a realizovat jejich programové propojení. [2, 3, 8, 14]

Standardizace

Standardizace má zásadní vliv na zlepšování procesů. ERP systém přináší standardy, které se v té či oné oblasti již osvědčily. Daní za standardizaci však může být ztráta některých specifických firemních procesů. [2, 3, 8, 14]

Automatizace

Tyto systémy zefektivňují a automatizují práci s obrovským množstvím dat. Automatizace procesů zajišťuje jejich správný a jednotný chod, jejich logickou návaznost a správnost. Důsledkem toho dochází ke snížení počtu chyb, zejména díky omezení lidského faktoru. Systém toho docílí tak, že uživateli je zobrazována sekvence formulářů a tabulek s precizní kontrolou vstupních dat. Uživatel tak není schopen omylem změnit chod procesu. [2, 3, 8, 14]

Cena

Vzhledem k vysoké komplexnosti je systém náročný na počáteční implementaci a obvykle s sebou nese i nutnost dalších dodatečných zásahů a úprav. To logicky vede ke zvýšení počátečních finančních nákladů a nákladů spojených s následnou údržbou systému. Cena se obvykle pohybuje v řádu desítek tisíc až milionů korun podle zvoleného řešení, implementace a následné customizace. [2, 3, 8, 14]

Díky komplexnosti systému už firma nepotřebuje platit za žádný další software. Tímto dojde ke snížení nákladů. Díky zvýšení efektivity práce dojde ke zvýšení hodnoty práce zaměstnanců i hodnoty firmy. Cena za zavedení systému se obvykle rychle vrátí, proto jsou tyto systémy dnes vyhledávány všemi druhy firem – od velkých korporací až po malé firmy. [2, 3, 8, 14]

Využitelnost

Z popsaných výhod a nevýhod plyne, že jsou tyto systémy obecně vhodné převážně pro velké či střední firmy. Nicméně na trhu jsou dnes řešení i pro malé firmy, která nejsou tolik komplexní. Zavedení ERP systému je tedy dnes možno do téměř každé společnosti. [2, 3, 8, 14]

Závěrem můžeme říci, že výhody převažují nad nedostatky těchto systémů. Kdyby tomu tak nebylo, nedostaly by se tyto systémy do pozice, v jaké jsou dnes. Téměř každá větší firma se dnes bez kvalitního ERP systému již neobejde [2, 3, 8, 14].

1.1.2 Současná situace

Nejúspěšnější firmou, která se zabývá vývojem ERP systému je společnost SAP. V současné době její produkty dominují ve všech oblastech, kde se ERP systémy používají. Dalšími předními výrobci těchto systémů na světovém trhu jsou zejména Oracle a Microsoft. [2, 3, 8, 11, 14, 15].

Na českém trhu je přes 100 firem, které vyvíjejí ERP systémy. Ty jsou však cíleny především na český trh, a proto nepatří mezi světovou špičku. Mezi nejvýznamnější systémy patří HELIOS a K2 [2, 3, 8, 11, 12, 14].

HELIOS

ERP systém Helios patří společnosti Asseco Solutions. Ta se zabývá vývojem ERP systémů déle než dvacet let. V současné době nabízí na trhu tyto produkty:

- Red – "komplexní zpracování podnikových agend malých a středních firem a podnikatelů bez ohledu na obor, pomocník i pro účetní kanceláře".
- Fenix – řešení pro veřejnou správu.
- Easy – přednastavený ERP systém.

Dále je v nabídce společnosti ke všem systémům i mobilní platforma [16].

K2

Jedná se o ERP systém, který je postupně vyvíjen od roku 1991 společností K2 atmitec. Na rozdíl od ostatních systémů není založen na modulech, jedná se o jedno ucelené řešení. Nová verze systému obohacená o nové funkcionality vychází každý rok. Díky tomu dokáže držet krok se současnými trendy, přestože jej nejde rozšiřovat pomocí modulů [3].

1.2 Společnost SAP

1.2.1 Historie společnosti SAP

Firma SAP patří na trhu mezi největší firmy zabývající se ERP systémy. Je největším dodavatelem ERP systémů a je to třetí největší softwarová firma

na světě (k roku 2017). Tato německá firma byla založena v roce 1972 pěti inženýry IBM (Dietmar Hopp, Hans-Werner Hector, Hasso Plattner, Klaus Tschira a Claus Wellenreuther). Jejich cílem bylo vyvinout podnikový informační systém, který by obsahoval co nejvíce firemních funkcí. Z počátku vyvíjeli aplikace pro správu financí a skladů. V roce 1973 získali prvního zákazníka, a to chemickou společnost ICI [4].

V roce 1977 se společnost přesouvá do města Walldorf v Německu, kde má sídlo dodnes. V 90. letech produkty společnosti SAP používalo více než 70 % velkých německých firem. Největší úspěch zaznamenala tato firma nasazením produktu R/3 v roce 1992. Díky tomuto produktu se systémy firmy SAP dostaly do středních a menších firem. Dnes po více než 45 letech má firma přes 345 000 zákazníků a 500 000 zaměstnanců [4].

1.2.2 SAP ČR

SAP ČR je dceřiná společnost firmy SAP, která působí v České republice od roku 1992. Díky dlouholetým znalostem lokálního trhu se jí podařilo dostat produkty SAP mezi zdejší firmy. Firmy, které u nás tyto produkty používají patří zejména do automobilového průmyslu. Tyto firmy jsou například Škoda auto, a.s., Johnson Controls, k.s. a Magna Automotive, s.r.o. SAP produkty se v naší republice nepoužívají pouze v automobilovém průmyslu. Jejich využití nalézáme i v energetice (ČEZ, a.s., RWE, a.s.), telekomunikaci (T-Mobile, a.s.), službách (Česká pošta, s. p.), potravinářství (Agrofert Holding, a.s.), a dalších (Preciosa, a.s., Okay, a.s., AVAST Software, s.r.o. nebo Karlovarské minerální vody, a.s.) [4].

SAP ČR nabízí mimo svých produktů také konzultační a implementační služby. Pražská centrála mimo jiné např. zajišťuje podporu a konzultační služby pro Evropu a Afriku [4].

1.2.3 Produkty

Nejznámějším produktem je ERP systém pojmenovaný Bussines Suite. Jedná se již o dříve zmíněný produkt R/3. Nejnovější verze byla na trhu uvedena pod označením S/4 HANA. Dále pak SAP nabízí upravenou verzi tohoto systému (SAP Business One), který je určen pro střední a malé firmy. Všechny tyto

produkty jsou však velice často nepřesně označovány pouze pod slovem SAP [4].

Dále firma SAP nabízí přes 300 dalších produktů a služeb. Některé z těchto položek souvisí s Bussines Suite, jiné jsou na ní naopak naprosto nezávislé. Nejvýznamnější službou je SAP Cloud platform, nástroj pro šíření a sdílení firemních aplikací. Dále lze za významný považovat i analytický nástroj Bussines Intelligence. Ten slouží pro analýzu chování zákazníků, odhadů metrik či analyzování a následné odstranění odhadů při nastavování firemních procesů [4].

2 Programovací jazyk ABAP

ABAP je programovací jazyk, který vznikl pro potřeby programování business logiky v produktech společnosti SAP. První verze vznikala v 70. letech 20. století z jazyka assembler. Dalších 20 let se tento jazyk vyvíjel až do podoby jakou má dnes. V roce 2000 byl například rozšířen o objektové programování. Dnes jsou tímto jazykem programovány veškeré aplikace SAP Bussiness Suite (kromě jádra, které je napsáno v jazyce C). Od roku 2003 je navíc možné v některých SAP produktech používat i jazyk JAVA, nicméně dominantním jazykem je stále jazyk ABAP [5].

2.1 Objekty SAP Repository

Pro vývoj aplikací v prostředí SAP je potřeba znát jeho základní stavební prvky. Ty se uchovávají v SAP Repository a v SAP terminologii se těmto prvkům říká objekty (pozn. pozor neplést s objekty v OOP). Těmito základními objekty jsou:

- **Paket**
  - Každému objektu musí být přiřazen nějaký paket. Členění objektů do odpovídajících paketů přináší strukturovanost a tím lepší organizaci objektů.
- **Doména**
  - Definuje datový typ a možný rozsah hodnot. Není je možné využívat přímo, ale pouze přes datový prvek.
- **Datový prvek**
  - Konkrétní využití domény. Může se použít jak v definici transparentní tabulky, tak přímo v programech. Obsahuje navíc popisky k polím na dynpru a nápovědu.
- **Struktura**
  - Spojení logicky souvisejících polí do jednoho celku. Využívá se jak v programech, tak při definici transparentní tabulky. Struktura nemá žádný protějšek v databázi.
- **Transparentní tabulka**
  - Definice tabulky sloužící jako předpis pro databázovou tabulku. Více viz kapitola 2.3

- **Program**
    - Existuje několik typů programů. Nejčastějším programem je program typu report skládající se z výběrové obrazovky a výpisu dat. Více viz kapitola 2.2
- **Funkční skupina a moduly**
    - Je speciální blok ABAP programu sloužící jako kontejner pro funkční moduly. Funkční moduly jsou modulární jednotky s rozhraním, které mohou být vyvolány z jakéhokoliv programu ABAP. Ve funkčních modulech se zapouzdřuje logika programu. Slouží pro opakované použití a lepší strukturovanost kódu.
- **Třída a metody**
    - Stejné jako u funkční skupiny a modulů. Navíc ale přináší možnost vytvoření více instancí.
- **Dynpro**
    - Dynpro je dynamický program skládající se z obrazovky a obslužné logiky. Slouží k definici uživatelského rozhraní. Je zde možné použít všechny dostupné zobrazovací prvky jako textová/vstupní pole, tlačítka, check-boxy, radio-buttony, tabulky apod.
- **Transakce**

Transakce je výchozím bodem pro spuštění programů. Zadává se v okně SAP do příkazového řádku nebo se přiřazuje do menu [5].

    - .

2.2 Struktura programu

Základní programy ABAP obsahují dvě části – deklarační a funkční část. V deklarační části se deklarují veškerá data, tabulky, pole atd. Dále se zde deklarují i výběrové obrazovky se všemi parametry. Proměnné smějí být pojmenovány pouze podle specifických pravidel. První znak musí být písmeno a může být dlouhý maximálně 30 znaků. Bloky zpracování (funkční část) obsahují jednotlivé algoritmy zpracování dat. Funkční část se obvykle skládá z více bloků zpracování. Standardně tyto bloky začínají a končí klíčovými slovy. Pořadí těchto událostí je pevně dané. Program postupně vyhledává jednotlivé bloky v pořadí:

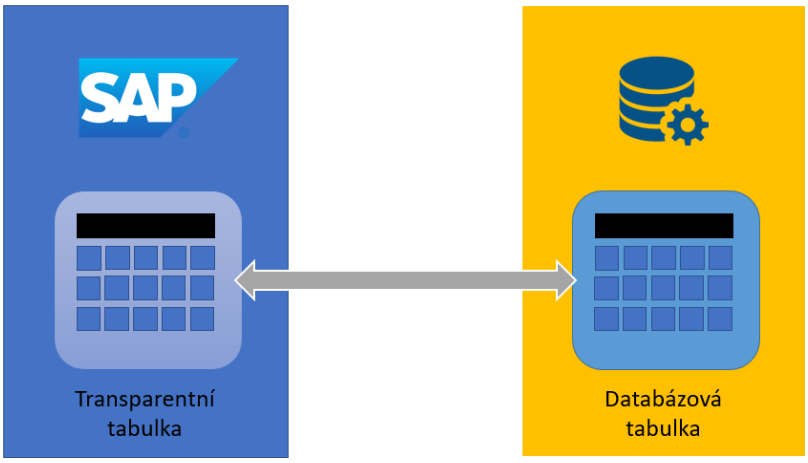
1. LOAD-OF-PROGRAM – provede se okamžitě po spuštění programu, nemá žádná omezení v obsahu
2. INITIALIZATION – v této události se provádí deklarace dat pro výběrové obrazovky nebo tuto obrazovku upravují
3. AT-SELECTION-SCREEN – slouží zejména pro složitější kontroly vstupu. Například po potvrzení výběrové obrazovky provede kontrolu, zda uživatel zadal platné hodnoty a případně vyvolá chybné hlášení a donutí uživatele zadat hodnoty znovu
4. START-OF-SELECTION – pokud jsou všechna data „v pořádku“, systém se dostane k události START-OF-SELECTION, kde už by měl být výpis reportu, který se uživateli na základě zadaných dat zobrazí [5]

```
REPORT <nazev_programu>.
PARAMETERS: <parametr> TYPE <datovy_typ>. "definice vstupniho parametru na vy
ber.obrazovce
DATA: <promenna> TYPE <datovy_typ> VALUE <hodnota>,
            <vysledek> TYPE <datovy_typ>.    "definice proměnnych
START-OF-SELECTION.   "startovaci blok
<vysledek> = <promenna> && <parametr>.  "Priklad spojeneni retezcu
WRITE: <vysledek>.   "vystup vysledku na obrazovku
```

Zdrojový kód 1 - ukázka programu v jazyce ABAP

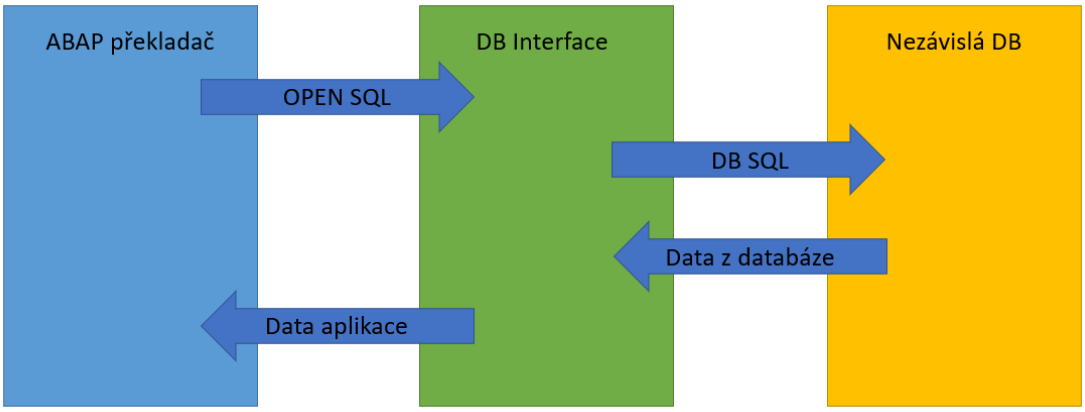
2.3 Práce s databází

V systému SAP se data ukládají v relačních databázových tabulkách. K přístupu k nim se používá strukturovaný dotazovací jazyk. Databázové tabulky se v prostředí SAP udržují v datovém slovníku (ABAP Dictionary) jako tzv. transparentní tabulky. Po aktivaci takovéto tabulky se daná tabulka založí pod stejným názvem a se stejnou strukturou i v použité databázi. [5]



Obrázek 1 - transparetní a databázová tabulka

Aby programy ABAP byly nezávislé na používaném databázovém systému, vytvořila firma SAP sadu zvláštních příkazů SQL, které se nazývají jako Open SQL. Open SQL obsahuje podmnožinu standardních SQL příkazů plus některá rozšíření, která jsou specifická pro SAP. Použití Open SQL tak umožnuje přístup k libovolným databázím bez ohledu na jejich výrobce. Databázové rozhraní totiž překládá příkazy Open SQL do příkazů SQL, které jsou specifické pro použitou databázi [5].



Obrázek 2 - komunikace mezi ABAP programem a databází

Klíčová slova jazyka vycházejí ze standardu SQL a na první pohled jsou Open SQL výrazy velice podobné klasickému SQL [5].

Tabulka 1 - přehled nejdůležitějších databázových příkazů

Klíčové slovo	Popis
SELECT	Čtení dat z databázových tabulek
INSERT	Přidávání nových řádků do db tabulek
UPDATE	Změny řádků v db tabulkách
MODIFY	Přidávání nebo změna řádků
DELETE	Výmaz řádků z db tabulky
COMMIT WORK	Potvrzení změn
ROLLBACK WORK	Zrušení změn

Ukázka databázové příkazu pro čtení dat:

```
SELECT <data> FROM <zdroj>
                  INTO <cíl>
                  WHERE <podmínka>
                  GROUP BY <fpole>
                  ORDER BY <pole>.
```

V krajních případech je možné použít i nativní SQL příkazy specifické pro použitou databázi. V tomto případě musí být takovýto příkaz zapouzdřen mezi příkazy EXEC SQL a END EXEC viz následující kód [5].

```
EXEC SQL.
<Nativní SQL příkaz>
END EXEC.
```

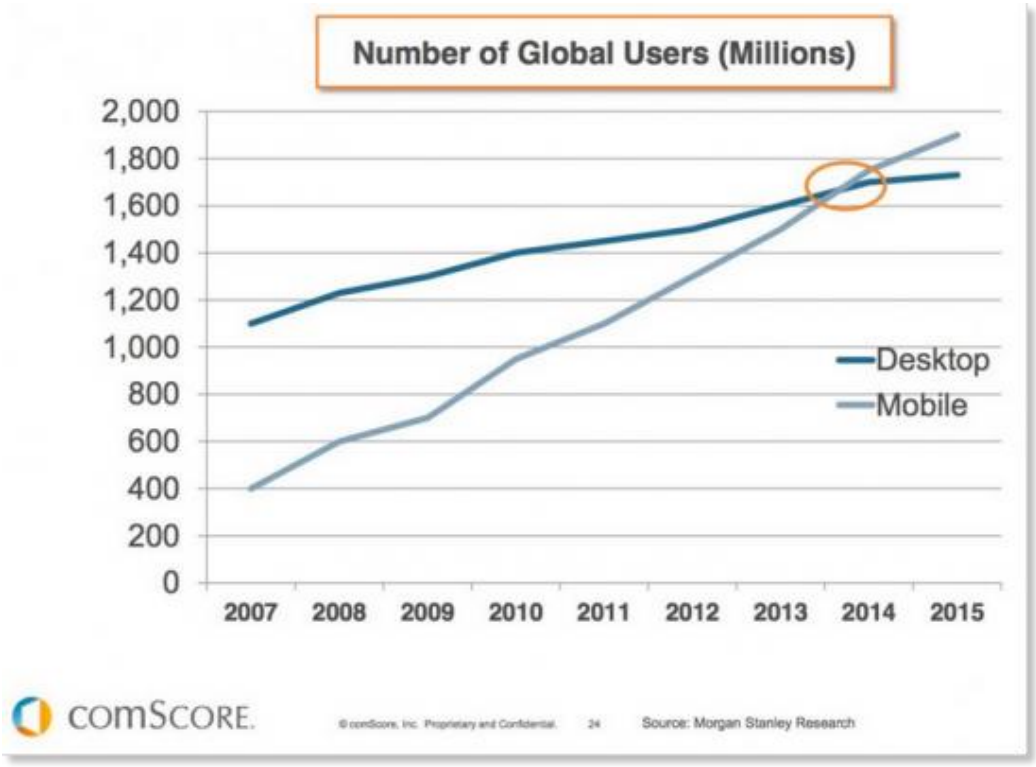
3 Využitelnost mobilních zařízení

V dnešní době je používání mobilních zařízení neodmyslitelnou součástí téměř všech zaměstnání. Důvodem je především jejich kompaktnost a s tím související snadná přenosnost. Nejdříve začal stolní počítače pozvolna nahrazovat laptop. Tento poměr se stále zvyšoval až do roku 2010, kdy se ustálil. Došlo k tomu proto, že v tomtéž roce byl zaznamenán významnější prodej tabletů. Prodej tabletů se postupně zvyšoval, až v roce 2013 prodeje tabletů trvale přesáhly prodej laptopů, jak je vidět na grafu 1. Přesto byly stolní počítače i nadále více používány [17, 18, 19].

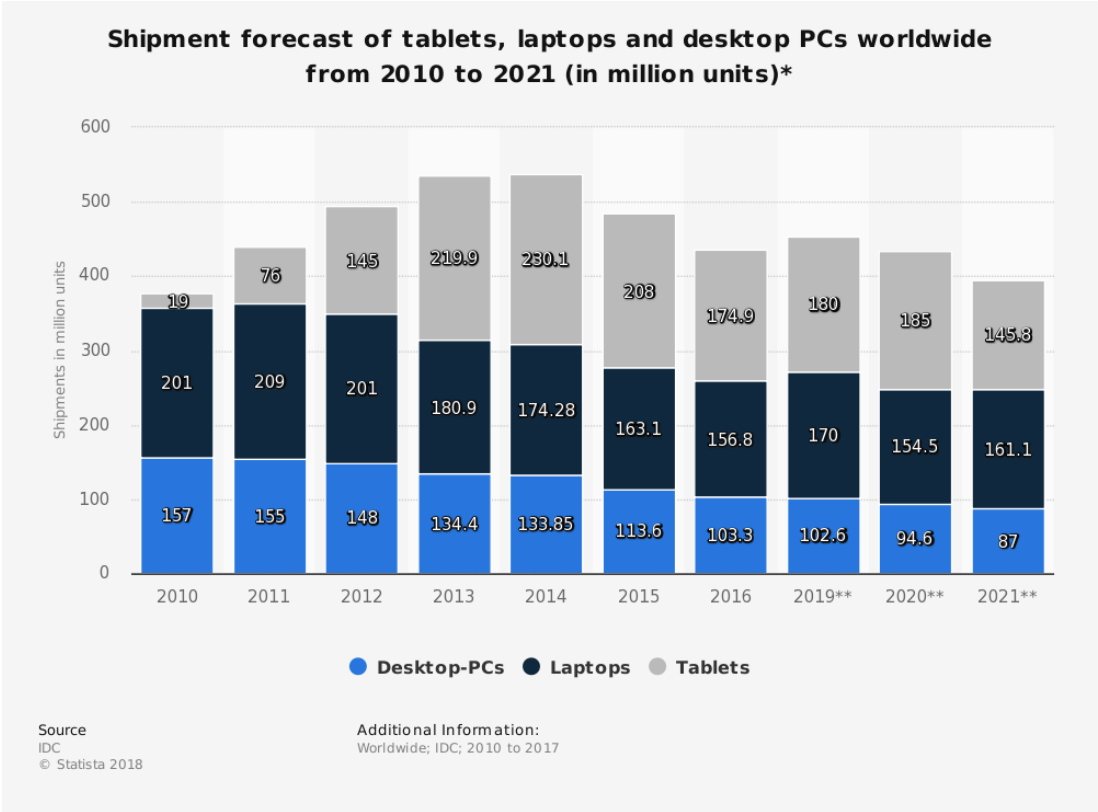
V roce 2014 došlo ke změně, když se mobilní zařízení začaly používat více než stolní počítače. Tuto skutečnost zachycuje graf 2. Na něm vidíme zvětšující se poměr uživatelů stolních počítačů a uživatelů mobilních zařízení [17, 18, 19].

Velký vliv na to měla poslední z kategorií mobilních zařízení – chytré telefony (smartphony). Od roku 2014 se každý rok prodá přes jednu miliardu kusů. V roce 2017 to bylo již přes 1,5 miliardy kusů a toto číslo stále roste. Tento trend můžeme sledovat na grafu 3 [17, 18, 19].

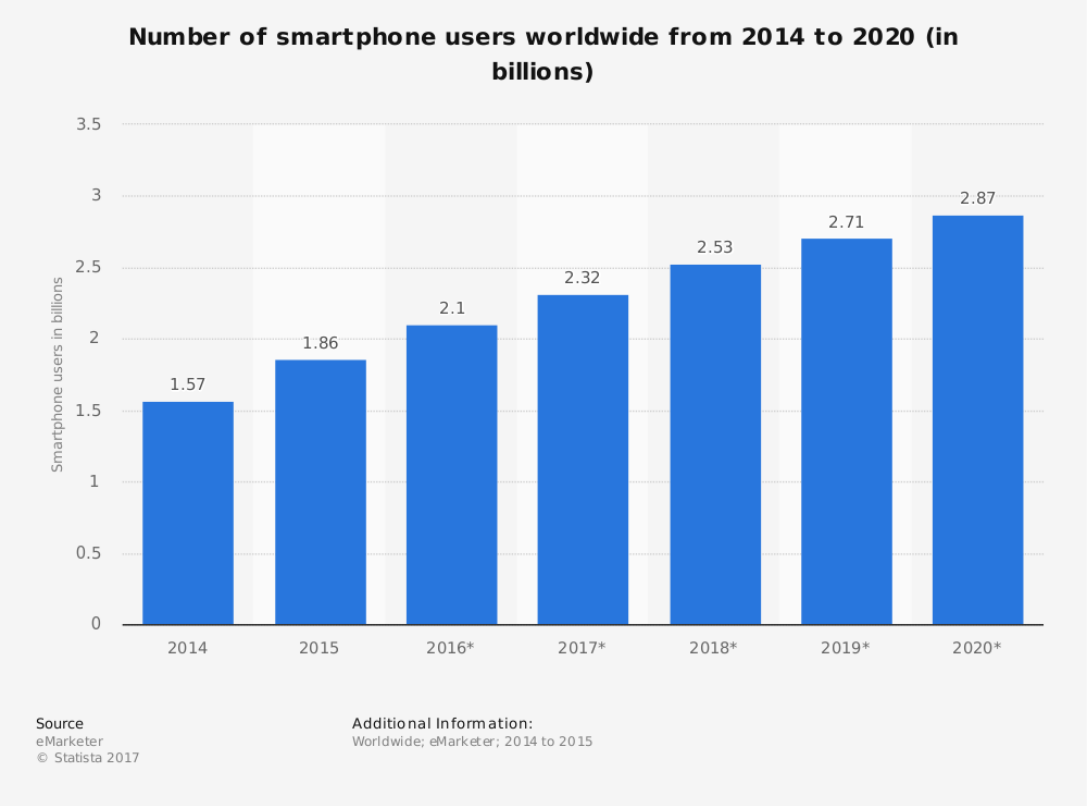
Se vzrůstajícím počtem mobilních zařízení se zároveň snižuje jejich cena. Díky tomu se mobilní zařízení stávají dostupnějšími jak pro firmy, tak pro jednotlivce. Není divu, že se firmy snaží napojit tato mobilní zařízení do svých podnikových procesů, resp. systémů [17, 18, 19].



Graf 1- Počet uživatelů Desktop X Mobile [19]



Graf 2 - Prodeje mobilních zařízení a stolních počítačů [17]



Graf 3 - Počet uživatelů chytrých telefonů [18]

3.1 Výhody a nevýhody používání mobilních zařízení

Napojení mobilních zařízení do systému nabízí mnoho výhod, ale zároveň přináší bezpečnostní rizika. Rizikem je vniknutí do systému neautorizovanou osobou. Docílit toho lze využitím chyby v aplikaci, či odcizením přihlašovacích údajů. Na druhou stranu toto riziko hrozí i na stolním počítači, ale v daleko menší míře.

Hlavní výhodou práce s mobilním zařízením je, jak již koneckonců vyplývá z názvu, jeho mobilita. Zaměstnanci mohou zadávat data do systému v reálném čase. Mohou rozhodovat o urgentních záležitostech ihned, bez ohledu na to, kde se právě nacházejí [4, 20].

3.2 Využitelnost v prostředí SAPu

Mobilní zařízení lze v prostředí SAP napojit na téměř jakoukoliv oblast či proces. Níže je uveden výčet potencionálních využití, rozdělený přes jednotlivé oblasti SAPu, tak jak jsem je vytipoval při studiu SAP.

- **CRM**: zadávání informací o zákaznících a obchodní partnerech přímo v terénu
- **Obchod**: Řízení faktur a pohledávek
- **Nákup:** Zadávání požadavků na objednávku
- **Výroba a plánování:** Zadávání pracovních příkazů, sledování výroby
- **Servis a služby:** Plánování služeb a údržby, zadávání poruch přímo v terénu, automatický odečet plynu a elektřiny
- **Řízení zásob:** Sledování stavu zásob a ceníků
- **Účetnictví:** Schvalování faktur pomoc workflow
- **Bankovnictví:** Zpracování platebních příkazů
- **Výkaznictví:** Výkazy práce, knihy jízd
- **Personalistika:** Evidence docházky

4 OS Android

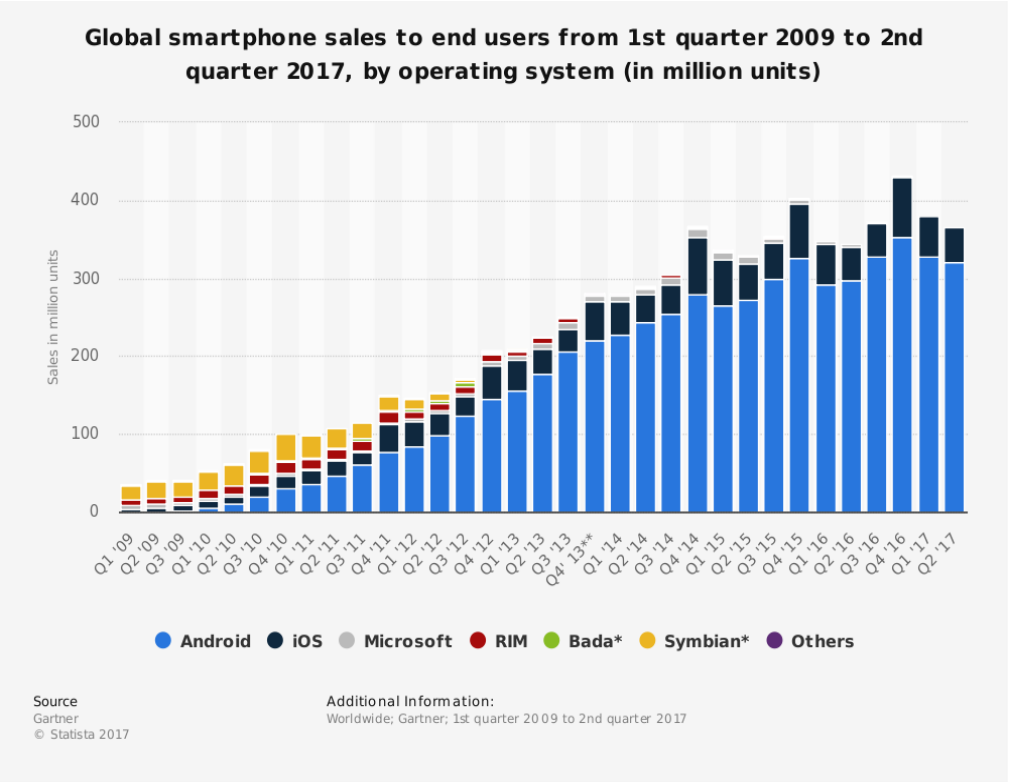
Jednou z nejzákladnějších věcí v mobilním zařízení je bezpochybně operační systém. Pro tuto chvíli již nebudeme uvažovat notebooky, kde hraje dominantní roli stále OS Windows, ale zaměříme se pouze na tablety a chytré telefony, V současné době ovládají trh zařízení s operačními systémy Android a iOS. Za zmínku stojí i třetí operační systém, a to upravená verze operačního systému Windows, nazvaná Windows Phone. S tímto operačním systémem se dnes vyrábí již pouze jeden model, a navíc společnost Microsoft, oznámila ukončení vývoje a podpory tohoto OS. Dá se tedy očekávat jeho brzké ukončení užívání i mezi jeho uživateli. [20]

4.1 Historie OS Android

V roce 2003 se objevil zajímavý startup s názvem Android Inc. Společnost Google v roce 2005 tento startup koupila a vytvořila z něj vlastní dceřinou společnost. Vzápětí (2007) vydal Google vlastní platformu založenou na linuxovém jádře.

V roce 2007 bylo také založeno konsorcium Open Handset Alliance (v čele se společností Google). Konsorcium tvořili zejména společnosti, které vyráběly mobilní telefony, čipy a aplikace. Jejich snahou bylo vyvinout otevřený standard pro mobilní zařízení. Výsledkem se stal ještě téhož roku první produkt – Android. Ihned poté vydali SDK 1.0 pro vývojáře pod licencí open source.

První mobilní zařízení s tímto operačním systémem bylo uvedeno na trh v roce 2008 ve Spojených státech amerických. V roce 2009 se vyrábělo již přes dvacet modelů s OS Android. Vysoká oblíbenost a dostupnost tohoto operačního systému způsobila, že v druhé polovině roce 2011 již měl 50procentní podíl mezi operačními systémy a od roku 2012 je ve více než v 80 procentech mobilních zařízení. Tato skutečnost je zobrazena na grafu 4[21].



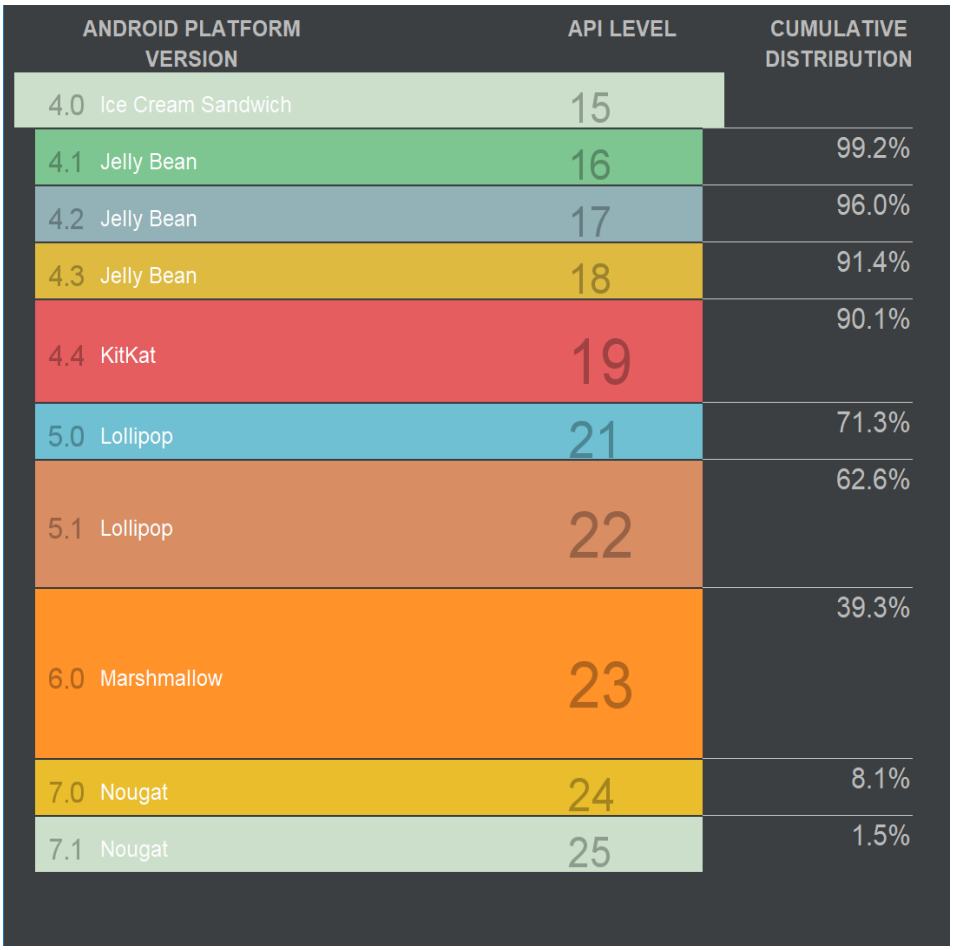
Graf 4 - podíl operačních systémů v chytrých telefonech

4.2 Vývoj aplikací

Vysoká popularita tohoto OS je zapříčiněna rozsáhlou nabídkou dostupných aplikací a podporou jejich vývoje. Podpora vývoje v tomto případě znamená, že Google dává k dispozici zdarma veškeré nástroje, tj. vývojové prostředí (IDE) i knihovny (SDK).

Tyto knihovny jsou označovány jako SDK – Software Development Kit. S každou novou verzí Android vychází i nová verze SDK. Mimo to vychází i nové verze SDK s novými funkcionalitami nebo bezpečnostními funkcemi bez nové verze OS. Z toho plyne, že se verze SDK a OS se nemusí shodovat. Současná verze OS je 8.1, kdežto verze SDK je 27.

U každé aplikace se musí vývojář rozhodnout pro minimální verzi SDK, kterou bude podporovat. Všechny novější verze jsou samozřejmě zpětně kompatibilní, takže např. při vývoji aplikace na verzi Android 4.3 bude aplikace fungovat i na novějších verzích. Pro správnou volbu minimální verze SDK dává Google k dispozici statistiky využití OS – čili i SDK. Google tyto statistiky aktualizuje přibližně jednou za měsíc, a proto mohou vývojáři rychle reagovat na vývoj situace na trhu. Google tyto informace získává sledováním verzí zařízení



Graf 5 - aktuální stav používanosti verzí Androidu [zdroj: IDE Android Studio]

připojených na jejich oficiální obchod Google Play. Tyto statistiky jsou zároveň integrovány i do jejich IDE, jak je vidět na grafu 5.

Vývojové prostředí (IDE) oficiálně vytvořené k vývoji aplikací pro OS Android, se jmenuje Android Studio. Nabízí pohodlný vývoj a testování těchto aplikací. Mimo všech verzí knihoven má v sobě integrované i virtuální zařízení, na kterém je možno aplikaci testovat. Pokud však nechceme aplikaci testovat na virtuálním zařízení, nabízí Android možnost testovat aplikaci i na reálných zařízeních připojených do počítače přes USB [22, 23].

4.3 Hlavní objekty

Nejdůležitějším objektem je **Activita**. Activita reprezentuje obrazovku. Obsahuje informace o uživatelském rozhraní a zprostředkovává jeho interakce s uživatelem. Při vytvoření aktivity dochází k alokování paměti pro uživatelské rozhraní rozloženého do layoutu aktivity. Aby nedocházelo ke značným ztrátám těchto prostředků, je zde tzv. Activity Manager. Ten spravuje spuštěné aktivity.

Přidává je do zásobníku a teprve při zaplnění zásobníku uvolňuje použité prostředky. Tím je docíleno toho, že nedochází ke zbytečnému zatěžování zařízení při opětovném spouštění stejných Activit (např. tlačítko zpět místo aby vytvořilo novou activitu, otevře již spuštěnou, pokud se nachází v zásobníku).

Service komponenta umožňuje spouštět kód na pozadí aplikace. To je dobré využít u výpočetně nebo časově náročných úkolech jako je komunikace se serverem.

Další objektem je **Content** provider. Content provider poskytuje rozhraní pro komunikaci mezi aplikacemi, nebo v rámci jedné aplikace mezi Activitami. Data jsou ukládána do souboru, SQLLite databáze nebo na web.

Poslední ze základních objektu je **Broadcoast receiver.** Broadcoast receiver odposlouchává komunikaci telefonu. Ve chvíli, kdy odposlechne nějakou událost, je schopen na ni vykonat určitou reakci. Příkladem toho může být přijetí SMS nebo také pokles stavu baterie [23].

5 Možnosti napojení mobilního zařízení s OS Android na systém SAP

Základem praktické části této bakalářské práce je realizace napojení systému SAP na mobilní zařízení s operačním systémem Android. Nejprve bylo nutné provést průzkum, jakými způsoby je možné se na systém SAP napojit. Na základě těchto výsledků poté zvolit dvě možnosti, podrobit je analýze a zrealizovat je pomocí jednoduchých aplikací.

Vlastní průzkum se opíral o následující zdroje:

- Internetové vyhledávání ve specializovaných SAP fórech pomocí klíčových slov (Fiori, OData, SOAP, Android, webservice)
- Oficiální SAP literaturu z edice SAP PRESS ([32])
- Rady a podněty od zkušenějších kolegů

Tímto způsobem se mi podařilo shromáždit informace, které se staly výchozím bodem pro další postup. Následně došlo k vyhodnocení nalezených informací z hlediska relevantnosti, aktuálnosti a četnosti. Konečné výsledky jsou zaznamenány v následující tabulce.

Tabulka 2 - realizované možnosti napojení na SAP

Stručný popis	**Realizace aplikací**
Napojení pomocí klasické webové služby	ANO
Napojení s využitím SAP FIORI	ANO
Využití SAP Mobile Platform	NE
Speciální konektory třetích stran	NE

Z nalezených možností byly vybrány jako nejvhodnější první dvě. Napojení pomocí klasické webové služby z hlediska univerzálnosti a napojení s využitím SAP FIORI, které se nyní SAP snaží podporovat jako aktuální standard. Obě tyto

možnosti nevyžadují žádné dodatečné licence, jsou proto realizovatelné v aktuálním firemním prostředí bez dodatečných nákladů. Využití SAP Mobile Platform bylo vyloučeno z hlediska potřeby dalších licencí. Stejně tak speciální konektory třetích stran jsou placené produkty.

5.1 Použité technologie

Před vlastní realizací bylo potřeba se nejprve seznámit s celou řadou nových pojmů a technologií. Nejpodstatnější z nich jsou stručně popsány v této kapitole.

5.1.1 SOAP

SOAP je typ webové služby, která charakterizuje své chování pomocí akcí. Každý požadavek, který přijme, obsahuje definice akce a služba vykoná svůj kód. Tato služba je dostupná přes koncový bod na doméně serveru. Komunikace s koncovým bodem služby probíhá pomocí protokolu http. Součásti zasílaného požadavku je objekt Envelope, který obsahuje informaci o tom, jakou akci má provézt a vstupní data [24].

5.1.2 REST

Je typ architektury webové služby, která využívá metody protokolu http k určení požadavku. Zkratka vychází z anglického „**RE**presentational **S**tate **T**ransfer“. Oproti SOAP, kde součástí požadavku je i akce, služba určuje cílovou akci podle URI adresy. Součástí URI adresy je metoda http a data. Podle metody, která je v URI obsažena, vykoná konkrétní akci [28].

Dostupné metody jsou:

- POST
- GET
- UPDATE
- DELETE

5.1.3 OData

Je protokol od společnosti Microsoft, který umožňuje jednoduše manipulovat s rozhraním REST. Protokol umožňuje klientu sestavovat a editovat URI a připojovat. Protokol tedy používá ke komunikaci http a jeho metody [27, 29, 30].

5.1.4 SAPUI5

SAP UI5 je javascriptová knihovna udržovaná společností SAP, která je pro její zákazníky zdarma. Dále existuje ještě její ‘open-source‘ verze pod názvem Open UI5. Obě verze mají shodné jádro a liší se pouze rozsahem (SAP UI5 > Open UI5). Pomocí těchto knihoven je možné jednoduše vytvářet uživatelská rozhraní, která běží na jakýchkoliv zařízení s podporou HTML5.

SAP UI5 (resp. Open UI5) využívá knihovnu jQuery a ostatní webové standardy (CSS, XML, JSON, OData). Knihovna obsahuje celou řadu UI prvků od tlačítek až po složité ovládací prvky [25, 31].

5.1.5 MVC

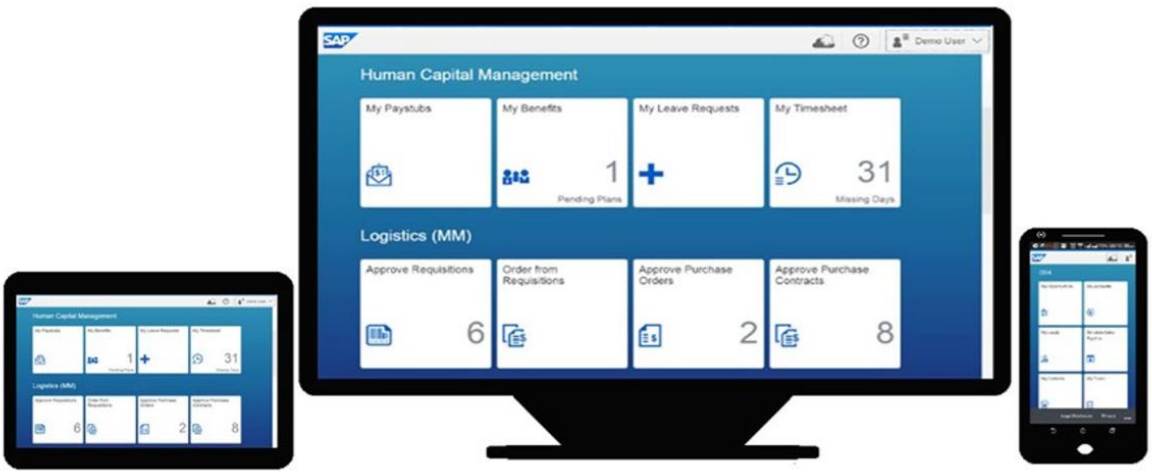
MVC (Model-view-controller) je softwarová architektura, která rozděluje strukturu aplikace do třech komponent, které jsou vzájemně nezávislé.

- Datový model (Model)
- Uživatelské rozhraní (View)
- Řídící logika (Controller)

Takovýto přístup zpřehledňuje vývoj, rozšiřitelnost a testování aplikací [33].

5.1.6 SAP Fiori

SAP Fiori vzniklo jako reakce na požadavky zákazníků SAP, kterým se klasické uživatelské prostředí (SAP GUI) již zdálo zastaralé a zejména že je dostupné pouze z desktopových zařízení. Na základě těchto podnětů SAP vyvinul sadu základních aplikací, které nahrazují klasické SAP transakce a jsou dostupné ze všech typů zařízení. Z uživatelského hlediska jsou tyto aplikace mnohem intuitivnější na ovládání a působí mnohem moderněji. Původně SAP nabízel Fiori pod dodatečnou licencí, ale nakonec od roku 2014 uvolnil Fiori zdarma pro všechny své zákazníky [26].



Obrázek 3 - Fiori na všech typech zařízení

Inspirací pro grafickou podobu Fiori aplikací se stal tzv. Flat design, který se již úspěšně využíval u produktů společností jako Google, Apple nebo Microsoft. Společným prvkem těchto uživatelských rozhraní je orientace na univerzálnost, jednoduchost, rychlost a tím minimální HW náročnost.

Z pohledu technologií je Fiori postaveno na:

- MVC architektuře
- HTML5 (s nadstavbou SAP UI5)
- OData službách

Fiori je nezávislé na použité platformě (Windows, Android, iOS). Každá Fiori aplikace by měla fungovat nejen ve webových prohlížečích, ale i pod konkrétním OS jako nativní aplikace. Z pohledu této práce je nejdůležitější kompatibilita Fiori s OS Android. Fiori podporuje OS Android od verze 4.1, a to jak přes integrovaný prohlížeč, tak přes nativní aplikaci Fiori Client. Ta je volně stáhnutelná přes Google Play Store [25, 26, 27].

6 Zadání mobilních aplikací

6.1.1 Timesheet

Úkolem aplikace bude zapisovat do SAP systému odpracované hodiny zaměstnance přihlášeného do systému z mobilního zařízení. Aplikace si při prvním spuštění vyžádá přihlašovací údaje do SAPu, ověří přihlašovací údaje a pokud budou validní, spustí uživateli formulář na vyplnění odpracovaných hodin. Uživatel vyplní do formuláře počet odpracovaných hodin, činnost, místo, projekt a případnou poznámku. Aplikace poté zapíše tyto informace do databáze SAP systému.

6.1.2 Skladiště

Úkolem této testovací aplikace bude zobrazovat uživateli na požádání aktuální stav skladů pro konkrétní materiál. Uživatel si vybere jeden z dostupných materiálů, a pro něj mu bude následně zobrazen seznam skladů, kde se materiál nachází. Seznam dostupných materiálu bude reprezentován drop-down seznamem. Ten bude samozřejmě také automaticky naplňován.

6.1.3 Elektroměr

Tato aplikace bude nejjednodušší z testovacích aplikací. Bude umožňovat uživateli zadávat stav elektroměru přímo v terénu. Aplikace bude vyžadovat po uživateli, aby vyplnil číslo a stav elektroměru. Aplikace bude pouze kontrolovat, aby hodnota stavu elektroměru obsahovala pouze číselné znaky s desetinnou čárkou.

7 Realizace pomocí webové služby

Dle zadání práce musí výsledné aplikace běžet na operačním systému Android. Tento operační systém byl zvolen, protože na trhu zastupuje přes 80 % mobilních zařízení (viz graf 5). K implementaci se používá programovací jazyk Java, rozšířený o další knihovny pro manipulaci s objekty v OS Android.

Nejvhodnějším nástrojem pro vývoj je Android studio. Studio má stejného výrobce jako OS Android, proto je považováno jako oficiální vývojový nástroj. Použití Studia tak zajišťuje naprostou kompatibilitu při vývoji. Samotné Android studio však k vývoji nestačí. Zapotřebí je ještě zařízení, na kterém budeme aplikaci testovat. Použít můžeme virtuální nebo fyzické zařízení s OS Android. Jeden z nejlepších nástrojů pro simulaci zařízení je Genymotion. Genymotion je ovšem pro komerční využití placený. Pro osobní účely se dá použít verze zdarma, která ovšem neobsahuje všechny možné funkce, jako je například simulace polohy.

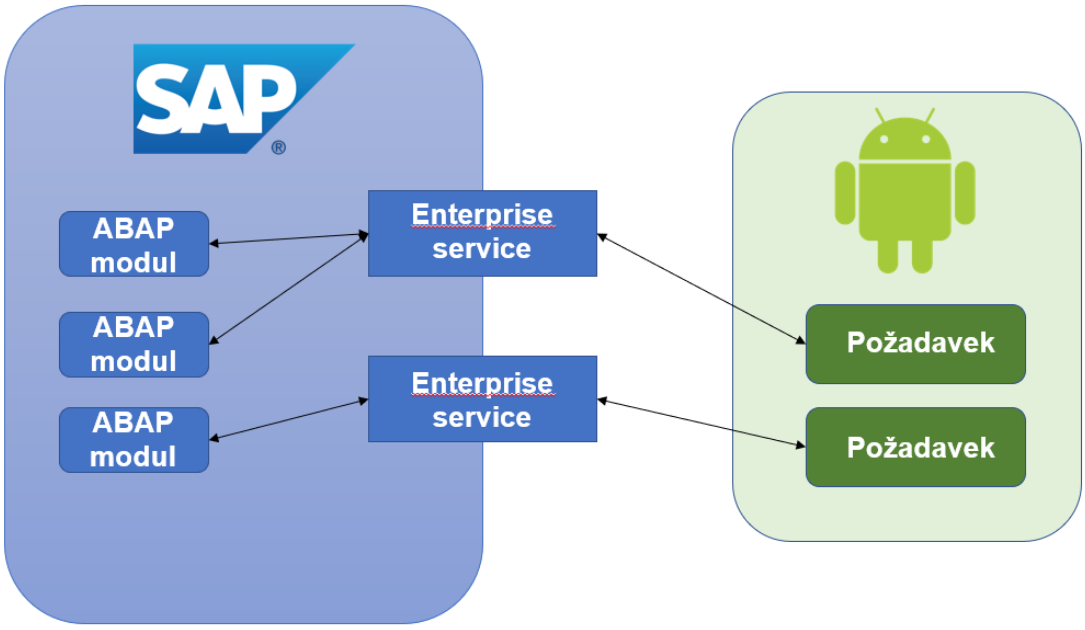
V tomto případě byla aplikace testována na fyzickém zařízení. Aplikace byla testována na tabletu od společnosti Alcatel s verzí systému Android 6.0 a na mobilním zařízení od společnosti Xiaomi s verzí systému Android 5.1.1. Spustitelná a správně fungující aplikace by měla ovšem být již od verze systému Android 4.3.

7.1 Timesheet

7.1.1 Datový model

Podle obecného schématu v grafu 6, byl vývoj aplikace rozdělen na 3 části – aplikace v mobilním zařízením, datový a funkční model v SAPu a webovou službu, která zprostředkovává jejich komunikaci.

Aby mohla být vytvořena logika na straně SAPu, musel být k dispozici datový model Veškeré objekty v SAPu je možné vytvářet v nástroji ABAP Development Workbench. Aby bylo možné odlišit standardní objekty od zákaznických, musejí všechny nově vytvořené objekty začínat prefixem ‘Z’. Nejprve tak byly vytvořeny databázové tabulky ZBPH_TIME_DATA, ZBPH_USER, ZBPH_PLACES,



Graf 6 - komunikace mezi SAP serverem a mobilním zařízení s OS Android

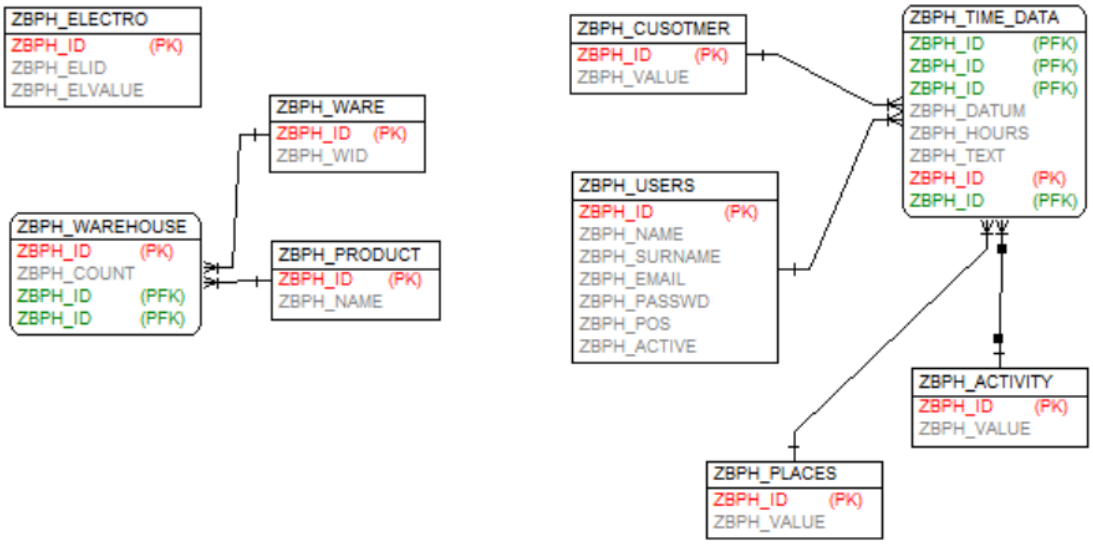
ZBPH_CUSTOMER a ZBPH_ACTIVITY, ve kterých budou uložena veškerá data aplikace Timesheet.

V následující tabulce 3 je přehled jejich polí a datový typ.

Jak je vidět, tabulky ZBPH_PLACES, ZBPH_ACTIVITY a ZBPH_CUSTOMERS mají stejnou strukturu, protože se jedná a jednoduché číselníky. Na obrázku 4 jsou pak vidět jejich vzájemné relační vztahy.

Tabulka 3 - jména a datové typy položek tabulek pro aplikaci Timesheet

Jméno	Datový typ	Jméno	Datový typ	Jméno	Datový typ
ZBPH_USERS		ZBPH_TIME_DATA		ZBPH_PLACES, ZBPH_ACTIVITY, ZBPH_CUSOTMERS	
ZBPH_ID	INT4	ZBPH_ID	INT4	ZBPH_ID	CHAR03
ZBPH_NAME	CHAR0032	ZBPH_PLACES	INT4	ZBPH_VALUE	STRING128
ZBPH_SURNAME	CHAR0032	ZBPH_CUSTOMER	INT4		
ZBPH_EMAIL	CHAR0032	ZBPH_ACTIVITY	INT4		
ZBPH_PASSWD	CHAR0064	ZBPH_TIME	DATS		
ZBPH_POS	CHAR0032	ZBPH_TEXT	STRING		
ZBPH_ACTIVE	INT3	ZBPH_HOURS	CHAR		
		ZBPH_USER	CHAR		



Obrázek 4 - relační vztahy tabulek v databázi

7.1.2 Funkční Moduly SAP

Tato aplikace provádí 4 druhy požadavků. Každý požadavek má vlastní funkční modul, který ho obsluhuje:

- Ověření uživatele – ZBPH_HASHPASSWD
- Načtení počátečních dat – ZBPH_READTABLE
- Odeslání formuláře – ZBPH_WRITETIME
- Vypsání posledních záznamů z databáze – ZBPH_FIVEROWS

ZBPH_PASSWD

Tento modul má bezpečnostní funkci, a to ověřit platnost přihlašovacích údajů. Na výběr byly dvě možnosti přihlašování. První byla využít přihlašovacích údajů do systému SAP a druhá byla vytvořit si vlastní databázi uživatelů. Byla vybrána druhá možnost, aby mohli vykazovat strávený čas i lidé, kteří nemají přístup do SAPu.

Přihlašovací údaje se skládají ze dvou informací – email a heslo. Email byl zvolen, protože se jedná unikátní řetězec znaků. Heslo je pak v databázi reprezentováno hash řetězcem kvůli bezpečnosti. Modul tedy zahashuje přijaté heslo a porovná oba řetězce. Výsledek pak vrací pouze shodu/neshodu. Tento kód je vidět v příloze A.

ZBPH_READTABLE

Tento modul vrací obsah jakékoliv tabulky jako tabulku znaků. Tento dynamický přístup je sice komplikovanější, ale zase není potřeba podobnou logiku definovat ve více modulech. Základním vstupem je název databázové tabulky, ze které se budou číst data. Z názvu tabulky se dynamicky vytvoří potřebné proměnné pro práci s tabulkou a jejím řádkem. Načtený obsah tabulky se poté prochází řádek po řádku a transformuje do výstupní tabulky znaků.

Tabulka 4 - seznam atributů modulu ZBPH_READTABLE

Jméno	Typ	Význam
I_TABNAME	Vstupní	Jméno tabulky.
I_HEADER	Vstupní	Tento flag signalizuje, jestli má být součástí výstupní tabulky hlavička.
I_DELIMITER	Vstupní	Definujeme znak/řetězec, kterým jsou odděleny sloupce v řádku.
I_WHERE	Vstupní	Doplňující WHERE podmínka.
STAB	Výstupní	Tabulka, která obsahuje obsah tabulky naformátovaný podle vstupních parametrů.

Tuto transformaci lze ovlivnit pomocí dalších vstupních parametrů, které jsou vidět v tabulce 4. Pro tuto práci je nejdůležitější možnost zvolit si oddělovač sloupců a možnost zvolit si, jestli má být součástí tabulky i hlavička.

Díky tomu byl modul schopen vracet obsah všech číselníků. Samozřejmě se dá tento modul využít i k vracení obsahu jiných tabulek, čehož bylo využito i v druhé aplikaci Skladiště.

ZBPH_WRITE_TIME

Tento modul ukládá záznamy do databáze pomocí klasického příkazu INSERT. Jelikož databázové rozhraní SAP neumí automaticky generovat ID, musí se tato logika definovat již v modulu. Pro tyto účely byl využit SAP číselník – interval

hodnot a funkce, která vrací první nepoužité číslo z tohoto objektu. Tento zásobník bylo nutno předem definovat. To bylo provedeno v transakci SRNO a byl tak vytvořen objekt s názvem ZBPH_ID.

ZBPH_FIVELINES

Jedná se o jednoduchou funkci, která vrátí posledních 5 řádků z tabulky ZBPH_TIME_DATA. Výstup této funkce je opět tabulka Stringů.

Kompletní zdrojové kódy těchto modulů jsou v příloze A.

7.1.3 Webová služba

Webová služba se v SAPu označuje jako Enterprise service. K jejímu vytvoření byl potřeba modul schopný remote. V takovémto modulu pak musejí být všechny vstupní a výstupní proměnné předávány hodnotou. To lze nastavit jednoduše v nastavení modulu.

Ve chvíli, kdy byly takto upraveny všechny výše popsané moduly, mohly být vytvářeny jednotlivé služby. Při vytváření lze nastavit různé parametry. V tomto případě byl důležitý především název služby, umístění a bezpečnost. Umístění bylo nastaveno do samostatného packetu ZBPH. Služby byly pojmenovány podle modulů ZBPH_WRITETIME, ZBPH_FIVELINES, ZBPH_LOGIN a ZBPH_TS_READ_BLE.

Tabulka 5 - přiřazení webových služeb k jejich modulům

Koncový bod	Jméno služby	Jméno modulu
/lastfiverows	ZBPH_WRITETIME	ZBPH_WRITE_TIME
/lastfiverows	ZBPH_FIVELINES	ZBPH_FIVELINES
/zbph_login	ZBPH_LOGIN	ZBPH_PASSWD
/readtable	ZBPH_TS_READ_TABLE	ZBPH_READTABLE

Poslední věc, kterou bylo nutno nastavit, byla bezpečnost. V tomto případě se jednalo o možnosti zabezpečení komunikace. Pro tento příklad byla použita možnost bez zabezpečení. Tato možnost ovšem vyžaduje alespoň minimální zabezpečení formou uživatelského jména a hesla.

V tomto kroku byly tedy vytvořeny webové služby z tabulky 5. V této tabulce je vidět jméno služby a modul, ze kterého byla vytvořena.

Nově vytvořené služby nejsou stále přístupné z internetu. Dále pro každou webovou službu byl vytvořen její koncový bod. To lze provést pouze v transakci **SOAMANAGER** v sekci *Confiure webservice*.

V průběhu zakládání bylo vypnuto ověřování. Služba nicméně nemůže pracovat v naprosté anonymitě – musí mít nastavenou nějakou identitu. Proto byly v tomto kroku vyplněny přihlašovací údaje. Služba tyto údaje následně používá pro komunikaci se SAPem. V dalším kroku byla nastavena URL, na které byla služba k dispozici a funkční modul, který používala. Po vytvoření koncového bodu se pro něj vytvořil soubor wsdl. Ten popisuje chování služby. Jeho ukázka je vidět na obrázku 5. Po vytvoření všech koncových bodů bylo vytvořeno funkční spojení. To lze otestovat i z prohlížeče. K tomu byl použit Google chrome. Pro něj

Obrázek 5 - ukázka wsdl

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
    <Body>
        <ZBPH_WRITE_TIME xmlns="urn:sap-com:document:sap:rfc:functions">
            <ACTIVITY xmlns="">[int]</ACTIVITY>
            <CUSTOMER xmlns="">[int]</CUSTOMER>
            <DATE xmlns="">[string]</DATE>
            <HOURS xmlns="">[string]</HOURS>
            <MESSAGE xmlns="">[string]</MESSAGE>
            <PLACES xmlns="">[int]</PLACES>
            <USER xmlns="">[string]</USER>
        </ZBPH_WRITE_TIME>
    </Body>
</Envelope>
```

Obrázek 7 - ukázka požadavku SOAP

```
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
    <soap-env:Header/>
    <soap-env:Body>
        <n0:ZBPH_LASTFIVEROWSResponse xmlns:n0="urn:sap-com:document:sap:rfc:functions">
            <STAB>
                <item>15211120180322111111213</item>
                <item>7411120180419Xyd5TEST</item>
                <item>73151201804186tmc012</item>
                <item>7286320180421Last test5tmc012</item>
                <item>71111201804185tmc012</item>
            </STAB>
        </n0:ZBPH_LASTFIVEROWSResponse>
    </soap-env:Body>
</soap-env:Envelope>
```

Obrázek 6 - ukázka odpovědi SOAP

bylo ovšem nutno použít rozšíření, které umožňuje komunikaci přes wsdl, Wizdler. Ukázka takového požadavku je na obrázku 7 a ukázka odpovědi je na obrázku 6. Celkový stav vytvořených objektů pro aplikaci Timesheet v SAPu zachycuje tabulka 5.

7.2 Skladiště

Nejprve byly vytvořeny databázové tabulky ZBPH_WARE, ZBPH_WAREHOUSE a ZBPH_PRODUCT. Jejich struktura je vidět v tabulce 6 a vzájemné relace jsou vidět na obrázku 4. Tabulky ZBPH_WARE a ZBPH_PRODUCT jsou číselníky. ZBPH_WARE ukládá informace o skladech a tabulka ZBPH_PRODUCT ukládá informace o produktech. Tabulka ZBPH_WAREHOUSE obsahuje kombinace skladů a produktů a přiřazuje jim počet dostupných kusů.

Tabulka 6 - vlastnosti tabulek pro Skladiště

Jméno	Datový typ	Jméno	Datový typ
ZBPH_WAREHOUSE		ZBPH_WARE, ZBPH_PRODUCT	
ZBPH_PRODUCT_ID	INT4	ZBPH_ID	INT4
ZBPH_WARE_NID	INT4	ZBPH_NAME	CHAR0032
ZBPH_COUNT	INT4		

Pro tuto aplikaci byl použit modul ZBPH_READTABLE, který byl vytvořen již dříve pro aplikaci Timesheet. Díky univerzálnosti mohl být tento modul opětovně využit i v tomto programu. Jak jsem zmiňoval výše, tak se webová služba vytváří ze SAP modulu. Z toho vyplývá že mohla být opětovně využita i dříve vytvořená webová služba s koncovým bodem /readtable, protože nově vytvořená služba by byla naprosto identická.

7.3 Elektroměry

Pro aplikaci Elektroměry byla vytvořena databázová tabulka ZBPH_ELECTRO. Tato tabulka obsahuje pouze záznamy skládající se z čísla elektroměru a jeho hodnoty. Poté byl vytvořen modul ZBPH_ELECTRO, který ukládá přijaté vstupy do databáze. Opět se jednalo o příkaz INSERT. ID bylo nutné generovat pomocí ZBPH_ID – číselníku. Pro zjednodušení v rámci bakalářské práce byl použit stejný číselník jako v aplikaci Timesheet, jelož rozsah hodnot je pro obě apliakce zcela dostatečný.

Z tohoto modulu byla vytvořena webová služba se stejným nastavením, jako bylo využito v aplikaci Timesheet. Pak pro ní byl v transakci SAOPMANAGER vytvořen koncový bod - /elmetr.

7.4 Mobilní aplikace

7.4.1 Menu

Nejprve byla vytvořena *Activita* pro menu. Tato *Activita* byla nastavena jako launcher, což znamená, že se zapne vždy po novém startu aplikace. Obsahuje pouze tlačítka, kterými se spouští jednotlivé podaplikace.

7.4.2 Timesheet

Timesheet je tvořen dvěma aktivitami. První *Activita* je přihlašovací, která je spouštěna přímo z menu. Zde uživatel vyplní email a heslo. Mobilní aplikace pak vygeneruje požadavek.

Tento požadavek se skládá z http hlavičky a těla. Hlavička má v sobě mimo jiné i informace nutné k napojení na správný koncový bod. Tj. adresu, na které se nachází, a název akce. Sestavená hlavička je ve zdrojovém kódu 2.

Zdrojový kód 2 - požadavek pro webovou službu SOAP

```
<v:Envelope
    xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:d="http://www.w3.org/2001/XMLSchema"
    xmlns:c="http://www.w3.org/2003/05/soap-encoding"
    xmlns:v="http://www.w3.org/2003/05/soap-envelope">
    <v:Header />
    <v:Body>
        <n0:ZBPH_WRITE_TIME id="o0" c:root="1"
            xmlns:n0="urn:sap-com:document:sap:rfc:functions">
            <ACTIVITY>1</ACTIVITY>
            <CUSTOMER>1</CUSTOMER>
            <DATE>2018-05-11</DATE>
            <HOURS>3</HOURS>
            <PLACES>1</PLACES>
            <MESSAGE></MESSAGE>
            <USER>test@t-mc66.cz</USER>
        </n0:ZBPH_WRITE_TIME>
    </v:Body>
</v:Envelope>
```

Poslední věc, která v požadavku chybí, jsou data. Ta jsou serializována do objektu *Envelop*. V tomto případě se jednalo pouze o email a heslo.

Požadavek je proveden a aplikace čeká na odpověď. Pokud by byl tento kód vykonáván v hlavním vláknu, došlo by k zamrznutí aplikace nebo jejímu pádu. Android ovšem tyto části kódu umí detekovat a takový kód nelze zkompilovat. Toto chování lze v krajních případech potlačit, ale je to nežádoucí.

Bylo tedy vytvořeno samostatné vlákno pro komunikaci, které je spouštěno na pozadí. Toto vlákno vykoná požadavek a vyčkává na odpověď. Mezitím hlavní vlákno signalizuje uživateli v podobě *progressBaru*, že aplikace čeká na výsledek ověření. Problém nastal až ve chvíli, kdy vlákno dostalo odpověď. Vlákna mají

standardně oddělen prostor, ve kterém běží a tím pádem spolu nesdílejí proměnné atd.

Aby bylo možno detekovat ukončení vlákna, musela by být neustále prováděna jeho kontrola v hlavním vláknu. To by ovšem vedlo ke stejné situaci, jako kdyby v něm byl vykonáván http požadavek. Hlavní vlákno by bylo zahlceno zkoušením vedlejšího. Proto musela být nalezena jiná cesta, jak synchronizovat stavy vláken.

K tomu byl použit objekt *Handler*. Tento objekt odposlouchává komunikaci v prostoru k tomu určenému. Ten se vytváří při jeho inicializaci. Inicializaci byla provedena v hlavním vláknu. V těle *Handleru* byl definován kód, který má být proveden po ukončení vedlejšího vlákna. Odposlechnout lze v tomto případě dvě

Zdrojový kód 3 - posílání zpráv do prostoru Handleru

```
Message message = mHandler.obtainMessage(1);
message.sendToTarget();
```

situace – konec vlákna s úspěšným ověřením a konec vlákna s neúspěšným ověřením. Pro každou situaci vyvolá specifickou situaci a to:

Úspěch - spustí druhou *activitu* Timesheet. Pomocí *Content provideru* mu předá email uživatele

Neúspěch – vymaže formulář a informuje uživatele o neúspěchu pomocí Toastu.

V druhém vláknu po přijetí odpovědi byla před ukončením vlákna zaslána do prostoru objektu *Handler* adekvátní zpráva – přepínač (viz zdrojový kód 3).

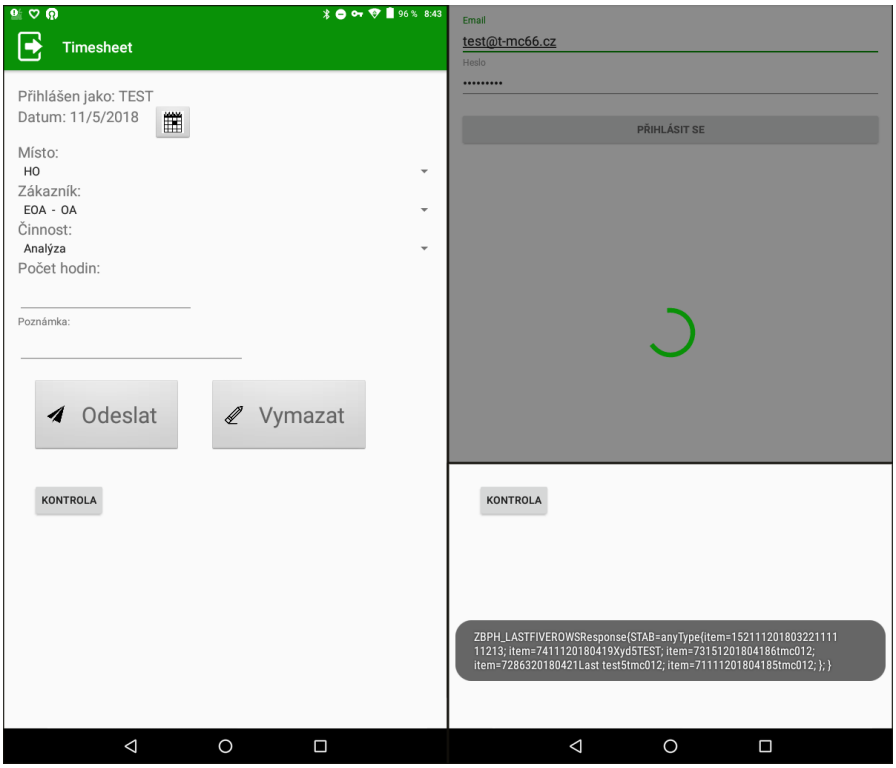
Druhá *Activita* Timesheet obsahuje formulář pro vyplnění údajů o odpracovaném čase. Obsahuje 3 komponenty *Spinner*. Ty jsou plněny ze SAPu pomocí služby s koncovým bodem /readtable. Volání této služby bylo opět provedeno ve vlastním vláknu. K synchronizaci byl opět použit *Handler.* Spinnery (drop-down box) jsou určeny k vybrání správné hodnoty z číselníků míst práce, druhů činnosti a zákazníků.

Další komponenty jsou *DatePicker* pro vybrání data. Automaticky je předdefinováno aktuální datum. Dále input pro počet hodin. Možnosti zadávání

byly omezeny pouze na číselné vstupy a znaky s tím spojené. Kvůli tomu, že uživatel má možnost kromě desetinné čárky zadat i jiné znaky, je prováděna kontrola, jestli je načtená hodnota typu *double*. Poslední komponenta je *TextArea*, která slouží k vložení poznámky. Na závěr byla ve formuláři vytvořena tři tlačítka – odeslat, vymazat a kontrola.

- Tlačítko Vymazat – toto tlačítko vymaže obsah všech editovatelných polí (počet hodin, poznámka) a nastaví výchozí hodnoty pro *DatePicker* a *Spinnery*.
- Tlačítko Odeslat – toto tlačítko odešle formulář na server. Postup je obdobný stejný jako u tlačítka přihlásit v předešlé aktivitě. Tzn. že je vytvořena hlavička požadavku. Poté je vytvořen objekt *Envelop*. Do něj jsou serializovány hodnoty z formuláře. Celý takto vytvořený požadavek je zaslán ve vedlejším vláknu na webovou službu s koncovým bodem /writetime. Hlavní vlákno opět mezitím signalizuje uživateli čekání na odpověď. Pro komunikaci mezi vlákny a synchronizaci stavů byl opět použit objekt *Handler*. Pomocí něj je signalizován uživateli úspěch či neúspěch, jak bylo popsáno výše.
- Tlačítko kontrola – toto tlačítko slouží pro kontrolní vypsání posledních pěti řádků tabulky ZBPH_TIME_DATA, aby v rámci testování mohl uživatel ověřit funkčnost aplikace. Aplikace tedy sestaví http požadavek a odešle ho v samostatném vláknu na server. Hlavní vlákno signalizuje uživateli čekání na odpověď. Po přijetí odpovědi druhé vlákno pomocí *Handleru* předá informace hlavnímu vláknu, které je vykreslí pomocí Toastu testeru.

Grafické znázornění varianty formuláře je znázorněno na obrázku 8 vlevo, stav čekání na odpověď je znázorněn napravo nahoře a výsledek kontroly je znázorněn napravo dole.



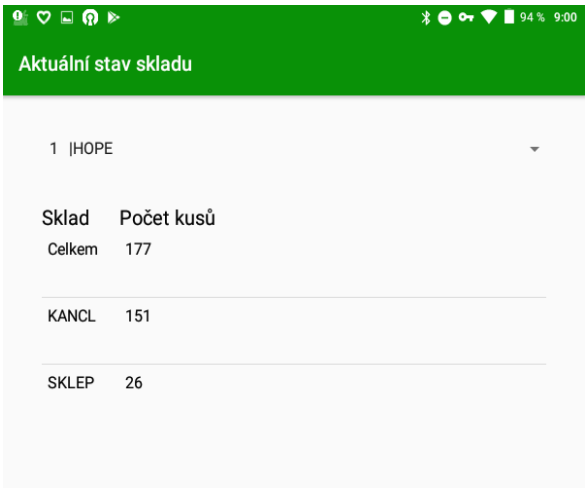
Obrázek 8 - grafické znázornění aplikace – v levé části Timesheet, vpravo nahoře načítání a vpravo dole kontrolní výstup z tabulky

7.4.3 Skladiště

Aplikace se skládá pouze z jedné aktivity. Tato aktivita načte data z databáze a umožní uživateli jejich filtraci. Nejprve tedy bylo nutné při inicializaci provést http požadavek na webovou službu s koncovým bodem /readtable na tabulku ZBPH_WARE. Požadavek byl opět proveden v samostatném vláknu. Po obdržení odpovědi bylo nutné data zpracovat do použitelné podoby. Data jsou zaslána v podobě tabulky Stringů a nejsou tak vhodná pro další přímé použití. Musel tedy být vytvořen speciální parser, který tuto tabulku Stringů převedl na objekty. Pomocí vzniklého *ArrayListu* byl naplněn *List* (vizuální prvek Android).

Pro filtrování relevantních hodnot byl umístěn do horní části okna Spinner (obrázek 9), který byl při inicializaci aplikace naplněn pomocí webové služby s koncovým bodem /readtable hodnotami z tabulky ZBPH_PRODUCT.

Při výběru libovolné položky ze seznamu filtr následně skryje všechny záznamy, které neobsahují vybraný produkt. V *Listu* poté zůstanou pouze sklady s aktuálním množstvím kusů. Do prvního řádku Listu byl umístěn sumační řádek, který zobrazuje celkový součet kusů daného produktu napříč všemi sklady. Toto rozložení znázorňuje obrázek 9.



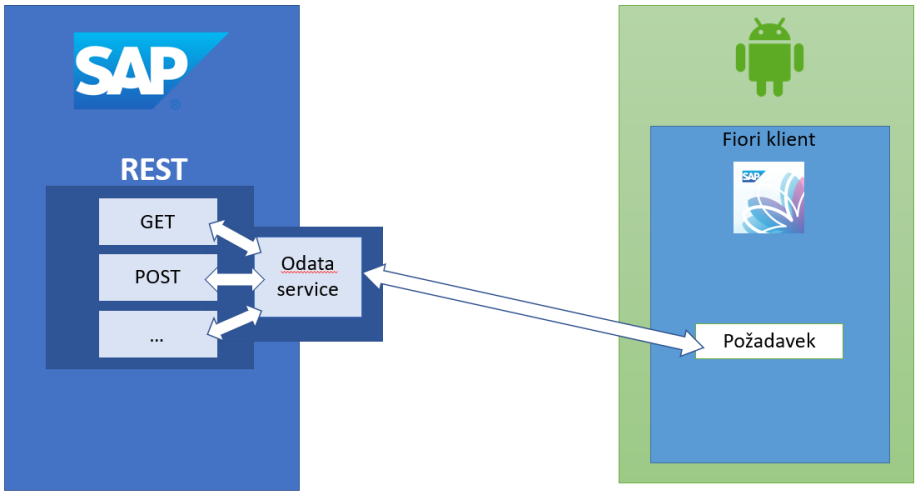
Obrázek 9 - grafické rozložení aplikace Skladiště

7.4.4 Elektroměr

Aplikace se skládá pouze z jedné Activity. Tato Activita obsahuje pouze jeden formulář a odesílací tlačítko. Při odesílání dat aplikace opět sestaví požadavek pro webovou službu s koncovým bodem /elmetr. Odesílání a vyčkávání na odpověď je prováděno v samostatném vláknu. Pomocí objektu *Handler* je detekován konec druhého vlákna, jak je popsáno v aplikaci Timesheet.

8 Realizace pomocí SAP Fiori

Vlastní realizace se skládá z vytvoření potřebných objektů a služeb a vývojem jednotlivých uživatelských rozhraní. Obecný popis aplikace je na obrázku 10. Datový model pro všechny aplikace zůstává stejný jako u předchozí realizace.



Obrázek 10 - obecný nákres logiky aplikace

8.1 Entity a OData služby

8.1.1 Timesheet

Vytváření objektů a jejich nastavování bylo provedeno v transakci **SEGW**. Na rozdíl od webové služby je nutné, aby každá OData služba byla vázána na nějaký objekt z *ABAP Dictionary*. Pro tyto účely slouží objekty označované jako Entity. V tomto případě byla navázána tabulka ZBPH_TIMESHEET na nově založenou stejnojmennou Entitu. Vznikl tedy stejnojmenný objekt Entity. Následně byly vytvořeny její vlastnosti (Properties). Ty odpovídají položkám v tabulce. Těmto vlastnostem byly povoleny možnosti Readable a Updatable. Dostupné možnosti zachycuje Tabulka 7. Aplikaci tím bylo uděleno právo využívat pouze metody Read a Update.

Další objekt, který byl vytvořen je EntitySet. Tento objekt reprezentuje objekt Entity s jeho vlastnostmi a umožňuje mu využívat další vlastnosti. Povoleny byly vlastnosti Creatable, Updatable, Deletable a Addressable. Kompletní přehled vlastností je opět v tabulce 7.

Tabulka 7 - přehled vlastností Entity a EntitySet

Entity	**EntitySet**
Readable	Creatable
Sortable	Updatable
Nullable	Deletable
Filterable	Pageable
Updatable	Addressable
	Searchable
	Subscribable
	Require filter

Ve chvíli, kdy byly vytvořeny všechny potřebné objekty, byly vygenerovány skripty. Tyto skripty jsou pojmenovány *Runtime artifacts* a definují chování služby. V tomto případě bylo nutné ještě upravit požadované metody (read, update atd.). Zdrojový kód je v příloze B.

Aby bylo možno naplnit comboboxy reálnými hodnotami, musel být proveden dotaz na relevantní číselník. Bohužel, vzhledem k tomu, že každá služba musí být vázána na jednu tabulku, bylo nutno vytvořit službu pro každý číselník.

Tyto služby jsou všechny stejné, liší se pouze hlavním objektem. V práci je tedy popsána pouze jedna služba.

Vytváření hlavního objektu a jeho nastavení bylo provedeno podle stejného postupu, jako v úloze Timesheet. Zdrojový kód přepsaných metod služby je v příloze B.

8.1.2 Skladiště

Obdobně jako v příkladu Timesheet byla vytvořena i služba pro Skladiště. Entita, na kterou se v tomto případě služba váže, je ZBPH_WAREHOUSE. Zásadní rozdíl v nastavení Entity je, že ve vlastnostech byla nastavena možnost pouze Readable a ve vlastnostech EntitySetu byla nastavena pouze možnost Filterable. Zdrojový kód se neliší od kódu číselníku.

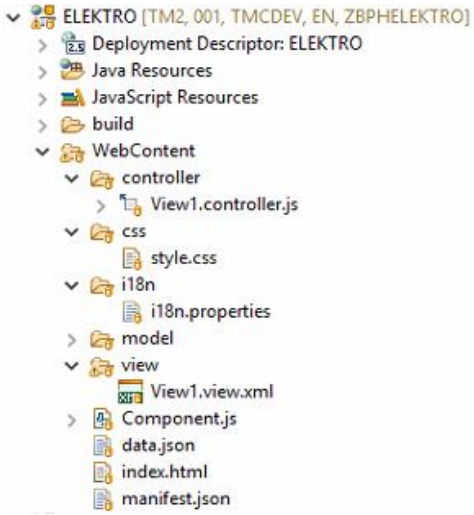
8.1.3 Elektroměry

Zadání této aplikace bylo nejjednodušší, a proto má aplikace nejméně povolených vlastností. Nejprve byla vytvořena služba podle stejného postupu jako dříve. Tato služba se váže na tabulku ZBPH_ELECTRO. V nastavení Entity se nastaví vlastnost Readable. V nastavení EntitySetu se nepovolí žádná operace. Zdrojový kód operace je v příloze B.

Tímto byly vytvořeny OData webové služby. Posledním krokem byla jejich aktivace. Ta se provedla přes možnost *Register* v každé službě v sekci *Maintain service.* Dostupnost koncového bodu byla ověřena v transakci **/n/iWfnd/gw_client.** Dostupný koncový bod musí vracet http hlavičku „200 OK“.

8.2 Struktura a vývoj aplikace

Aplikace pro Fiori mají striktně danou strukturu. Tuto strukturu je možno nechat vygenerovat na Webide. Na Webide byl vytvořen prázdný projekt. Jeho struktura pak byla importována do projektu na lokálním PC. S tím byl vázán drobný problém. Bylo nutno ověřit jméno komponenty aplikace. V ABAP Repository nesmí být dvě aplikace se stejnými jmény komponenty. Komponenta v tomto případě funguje zároveň jako identifikátor.

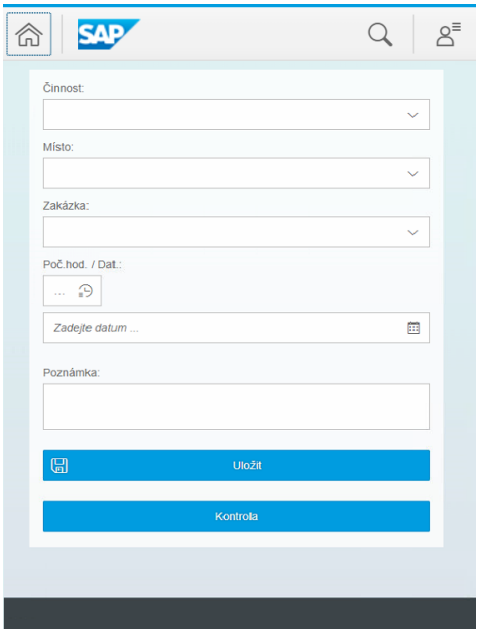


Obrázek 11 - struktura aplikace

Struktura aplikace je vidět na obrázku 11. Ze struktury je patrné, že se jedná o MVC model architektury. Ve složce model se nacházejí pouze testovací data, jinak je reprezentován serverem SAP.

8.2.1 Timesheet

Vzhledem k tomu, že ověření uživatele je prováděno automaticky při vstupu do klienta, nebylo nutné ho implementovat znovu. Aplikaci nebylo potřeba rozdělovat a skládá se tedy pouze z jednoho Controlleru a View, viz obrázek 11. Pomocí komponent knihovny SAPUI5 bylo vytvořeno GUI pro tuto aplikaci. GUI je patrné z obrázku 12. Následně bylo potřeba naplnit formulář daty ze SAPu. Konkrétně se jednalo o číselníky s údaji o místě, činnosti a zákazníka, pro kterého byla práce vykonávána. Tyto číselníky byly naplněny pomocí dříve vytvořených OData služeb ZBPH_PLACE, ZBPH_ACTIVITY a ZBH_CUSTOMER. Pomocí metody GET byly získány hodnoty pro číselníky.



Obrázek 12 - GUI pro Fiori

Poslední věc byla zjistit identitu přihlášeného uživatele. Uživatel se přihlásil do aplikace Fiori a nemáme pro něj tedy přihlašovací údaje ve formě emailu, jako tomu bylo v aplikace Android. Musel být tedy získán jeho uživatelský identifikátor. Ten byl získán pomocí funkce getId(). Tato funkce se nachází ve třidě *Shell* a její volání znázorňuje Zdrojový kód 4.

Zdrojový kód 4 - zjištění identity uživatele

```
var oUser = sap.ui2.shell.getUser();
var userId = oUser.getId();
```

Součástí formuláře je tlačítko odeslat a vymazat. Pro odeslání formuláře byla použita metoda POST. Pro vymazání formuláře nebylo potřeba vytvářet žádnou REST metodu, protože se jednalo o lokální funkci, tj. uvedení formuláře do výchozího stavu.

8.2.2 Skladiště

Samozřejmě i zde se jednalo o jednoúrovňovou aplikaci. Aplikace pomocí metody GET OData služby ZBPH_WAREHOUSE načte při inicializaci data. Tato data jsou pak pouze při změně výběru hodnoty v Combo-boxu tříděna.

8.2.3 Elektroměry

Jednoduchá jednoúrovňová aplikace. Pro odesílání používá metodu POST z OData služby ZBPH_ELECTRO. Komponenty formuláře tvoří opět objekty SAPUI5.

8.3 Zobrazení aplikace v klientu FIORI

Aby aplikace mohla být spouštěna z klienta Fiori je nutno, aby program byl na stejné doméně, jako je systém SAP. Z toho plyne, že nelze provádět *Cross-domain* požadavky. Z tohoto důvodu bylo nutno nahrát aplikace na server SAP do ABAP Repository. Nejjednodušší cesta je využít nástrojů vývojového prostředí. Pro vývoj používáme aplikace prostředí Eclipse. Pro toto prostředí společnost SAP distribuuje rozšíření, které obsahuje veškeré knihovny a mimo to i nástroj pro nahrání aplikací do ABAP Repository. Pro úspěšné nahrání musel být ve stejné síti jako cílový server a uživatel musel mít vývojářské oprávnění pro cílový server. Po nahrání již byla aplikace dostupná ze SAPu. Otestování aplikací proběhlo z transakce ***SICF***.

Další krok bylo vytvořit launcher, který bude sloužit ke zobrazování aplikací. Ten byl vytvořen v transakci ***LPD_CUST***. Byl zde tedy vytvořen objekt s rolí ZBPH a instancí BP. (Role a instance jsou volitelné, slouží pro identifikaci mezi ostatními launchery.)

Po vytvoření do něj byly přidány všechny tři aplikace z ABAP Repository. Aplikace je typu URL, což znamená, že se k ní přistupuje přes URL. Konkrétně se musela nastavit absolutní cesta. Aby bylo možno korektně aplikace zobrazit, bylo nutné do parametru *Additional Information* vyplnit „SAPUI5.Component=<<jméno komponenty>>“. <<jméno komponenty>> je namespace aplikace, který se dá nalézt např. v souboru Component.js, který je v každé aplikaci. Nastavení aplikací zobrazuje tabulka 8.

Dále bylo nutné vytvořit *Semantic object*. Ten slouží pro zobrazování aplikací v launcheru. Pro každou aplikaci musel být vytvořen vlastní sémantický objekt. Při pokusech s jedním sémantickým objektem pro všechny aplikace bylo zjištěno, že následně dochází k chybám při volání aplikace. Sémantický objekt nevěděl, kterou aplikaci volat.

V tomto kroku bylo připraveno vše na straně SAPu a byl nastaven samotný launcher. V launcheru bylo potřeba vytvořit katalog. Při jeho vytváření je nutno nastavit jeho typ, jméno a identifikátor. Typ byl nastaven na standart, jméno na ZBPH a ID také na ZBPH.

Následně byl vytvořen mapping pro každou aplikaci. Při vytváření mappingu je důležité nastavit *Semantic Object*, *Action*, *Application Type*, *Title*, *URL*, *Component* a *Device Types*.

- *Semantic Object* – aplikaci byl přiřazen sémantický objekt, pomocí kterého bude zobrazována.
- *Action* – výběr akce, kterou bude sémantický objekt vykonávat. Na výběr je *display* a *displayFactSheets*. Pro naše účely byla vybrána akce *display*.
- *Application Type* bylo nutné nastavit na SAPUI5 Fiori App
- *Title* – název, pod kterým se bude aplikace zobrazovat v launcheru
- *URL* – je stejná URL jako byla zadána v LPD
- *Component* – jméno komponenty
- *Device Types* – dovoluje definovat, pro která zařízení bude aplikace dostupná (Desktop, Tablet, Mobile)

Tabulka 8 - nastavení parametrů mappingu jednotlivých aplikací

	Timesheet	Skladiště	Elektroměry
Semantic Object	ZBPH_TIME	ZBPH_WARE	ZBPH_ELE
URL	/sap/bc/ui5_ui5/sap/zbph_timeshee	/sap/bc/ui5_ui5/sap/zbphwarehouse	/sap/bc/ui5_ui5/sap/zbphelektro
Title	Timesheet	Warehosue	Electrometr
Component	Timesheet	Warehouse	Electro

Nastavení vlastností, které jsou odlišné pro jednotlivé aplikace, je zobrazeno v tabulce 8.

Po vytvoření mapování pro každou aplikaci byly vytvořeny dlaždice. V dlaždici byl zadán pouze název (může se lišit od hodnoty, byla zadána do Title v mappingu), ikona k zobrazení a v neposlední řadě sémantický objekt, který má použít. Ikony jsou klasické ikony systému SAP, nelze bohužel nahrát vlastní ikonu.

Následně byla vytvořena skupina. Nastavení skupiny bylo obdobné jako katalogu – vyplňuje se pouze jméno a identifikátor. Do skupiny byly přidány dlaždice z vytvořeného katalogu. Skupina slouží k uspořádávání aplikací mezi uživateli. Tj. aplikace je v jednom katalogu, ale může být ve více skupinách.

V tento moment byl Fiori launcher nastaven a aplikace jsou dostupné z Fiori klienta. Poslední problém, který je nutno vyřešit je dostupnost pro uživatele. Aplikace byly sice dostupné v klientu, ale žádný uživatel je neviděl, protože k tomu neměl práva. Jako poslední věcí tedy bylo vytvoření SAP role. To bylo provedeno v transakci **PFCG**. Vytvořena byla role ZBPH. Roli byl umožněn přístup k vytvořenému katalogu a vytvořené funkci. Dále byla role přiřazena uživatelům, kteří testovali aplikace.

9 Porovnání obou metod

Základní věcí, kterou bylo potřeba definovat, bylo z jakých hledisek bude aplikace srovnávána. Těchto hledisek může být celá řada, nicméně jako nejdůležitější byla vybraná stabilita, intuitivnost ovládání, použitelnost a doba vývoje. Aplikace byly proto hodnoceny podle těchto aspektů.

9.1 Stabilita, intuitivnost ovládání a použitelnost

Použitelnost testoval vzorek uživatelů, kteří neměli uživatelské a programátorské znalosti SAPu. Testování proběhlo formou vyplnění anonymního dotazníku. Dotazník otázkami prověřoval stabilitu aplikací a úroveň jejich využitelnosti v podobě, jaké jsou nyní. Zároveň s tím byla hodnocena i intuitivnost ovládání.

Testeři vybírali vždy u každé otázky ze slovních odpovědí. Odpovědi byly následně převedeny na číselné hodnoty. Pozitivní opověď odpovídala hodnotě jedna a každé jinéí odpovědi byla snížena známka o jeden stupeň. Otázky se dvěma možnostmi mohly mít známky 1-2, otázky se třemi odpověďmi mohly mít známky 1-3, atd. … Výsledné hodnocení v podobě aritmetického průměru lze nalézt v tabulce 9.

Tabulka 9 - hodnocení aplikací

		Android	**Fiori**
Stabilita		1	1
Intuitivnost ovládání		1	1,2
Použitelnost	Timesheet	1	1,9
	Skladiště	1,1	1
	Elektroměr	2	2

Jak je vidět, uživatelům aplikace nikdy nespadly. Stabilitu mají výsledné aplikace stejnou na obou platformách. Ovšem u intuitivnosti ovládání již dochází k odchylce. Paradoxně uživatelské rozhraní navržené od SAPu, které je součástí standardu Fiori, má o něco málo horší hodnocení. To je způsobeno tím, že prvky použité pro aplikace nelze 100 procentně přizpůsobovat požadavkům firmy. Na druhou stranu u aplikací pro Android lze jednoduše docílit jakýkoli požadavků.

Použitelnost aplikací byla hodnocena pro každou platformu a aplikaci zvlášť. Timesheet – primární a nejrozsáhlejší aplikace, byla lépe hodnocena ve verzi pro Android. V aplikaci Fiori byla aplikace limitována klientem Fiori. Největším problémem se ukázala v tomto případě nemožnost ověření uživatele bez přístupových údajů do SAPu. Samozřejmě by to šlo vyřešit vlastní tabulkou uživatelů, nicméně uživatel by se musel přihlásit dvakrát a pokaždé s jinými přihlašovacími údaji. Proto byla tato cesta opuštěna a zvolena aktuální implementace.

Aplikace Skladiště byla pozitivně hodnocena v obou platformách. Lepší hodnocení získala implementace pro Fiori. Aplikace Elektroměry byla hodnocena nadprůměrně na obou platformách. Tato aplikace byla vyhodnocena jako skvělá možnost pro terénního pracovníka, jak jednoduše zadávat aktuální data do SAPu.

Z těchto výsledků vyplývá to, že komplexní ovládací prvky jsou vhodné pro složitější aplikace. Naopak pro jednodušší aplikace je vhodnější implementace pomocí webové služby.

9.2 Vývoj

Posledním aspektem hodnocení je vývoj. Vývoj aplikace pro Android byl jednoznačně jednoduší a pohodlnější. Android disponuje velmi propracovanou dokumentací a velmi širokou aktivní programátorskou komunitou. Je tedy velmi pravděpodobné, že vzniklé problémy již někdo řešil. Lze snadněji dohledat řešení a urychlit tak celkový vývoj.

Aplikace pro klienta Fiori se zatím programují hůře. Slovo zatím je velmi důležité. Přestože SAPUI5 není již horká novinka, není zatím mnoho dokumentace a řešených problémů. Velikost programátorské komunity také není příliš velká.

Do budoucna má tato platforma velký potenciál, avšak zatím je podle mého názoru stále lepší, zvolit možnosti vlastního klienta a aplikací přímo pro Android.

Dalším důvodem je, že přestože knihovny jsou z části openSource a lze je upravovat, SAP Fiori klient je již hotová a uzavřená aplikace. Úpravy aplikace na úrovni klienta tak nejsou ve Fiori možné.

Závěr

Vzhledem k tomu, že se mobilní zařízení stávají každodenní součásti lidského života, projevuje se tento trend i v pracovní sféře. Tato zařízení se stávají nástrojem, který firmy používají každý den. Proto byl v rámci bakalářské práce proveden průzkum, jak tato zařízení napojit i na velké systémy jako je SAP. Jeho výsledek jsem uvedl v kapitole 5.

Z tohoto průzkumu bylo zjištěno, že v současné době je k dispozici několik možností, jak toho docílit. Z těchto možností pak byly vybrány dvě nejpoužívanější, a to webová služba s klientem pro Android a OData služba s klientem Fiori.

Aby bylo možno provést kvalitní vyhodnocení, která z těchto metod je vhodnější, byly tyto dvě metody implementovány. Jako testovací vzorek posloužily tři aplikace, které byly implementovány s oběma možnostmi. Na těchto aplikacích pak bylo provedeno uživatelské testování, které je popsané v kapitole 9. Dále byly metody porovnány z pohledu náročnosti jejich implementace.

Z těchto výsledků celkově plyne, že aplikace implementovány pomocí SOAP a vlastního klienta, jsou celkově lépe hodnoceny. Proto byly vyhodnoceny jako lepší řešení minimálně do té doby, než dojde u SAP Fiori s UI5 ke zlepšení kvality dokumentace, zvětšení odborné komunity a tím ke zjednodušení vývoje.

Jako pokračování této práce by bylo možné rozšířit mobilní zařízení i na jiné zařízení než s operačním systémem Android – např. s iOS. Následně by bylo vhodné implementovat testovací aplikace i na tyto OS a provést jejich vyhodnocení. Dále by mohlo dojít k integraci VPN přímo do klienta, aby byl zmenšen počet nutných aplikací pro připojení se do SAPu.

Seznam použitých zdrojů

[1] *Consuming soap web services from android. Nascenia [online]. 2016. Dostupné z: 0http://www.nascenia.com/consuming-soap-web-services-from-android/*

[2] *Co je ERP systém? [online]. [cit. 2018-02-07]. Dostupné z: http://bluedynamic.cz/co-je-erp-enterprise-resource-planning/*

[3] *K2 atmitec [online]. [cit. 2018-02-07]. Dostupné z: https://www.k2.cz/*

[4] *SAP [online]. [cit. 2018-02-07]. Dostupné z: https://www.sap.com/cz/index.html*

[5] *SAP ABAP [online]. 2016. Tutorialspoint [cit. 2018-02-07]. Dostupné z: https://www.tutorialspoint.com/sap_abap/sap_abap_tutorial.pdf*

[6] *KÜHNHAUSER, Karl-Heinz. ABAP: výukový kurz. Brno: Computer Press, 2009. ISBN 978-802-5121-177.*

[7] *MAASSEN, André. SAP R/3: kompletní průvodce. Brno: Computer Press, 2007. Informační systémy. ISBN 978-80-251-1750-7.*

[8] *What is ERP? [online]. [cit. 2018-02-07]. Dostupné z: http://www.netsuite.com/portal/resource/articles/erp/what-is-erp.shtml*

[9] *XML services [online]. [cit. 2018-02-07]. Dostupné z: https://www.w3schools.com/xml/xml_services.asp*

[10] *Mobile Marketing Statistics compilation [online]. [cit. 2018-02-07]. Dostupné z: https://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/*

[11] *Přehled informačních systémů [online]. [cit. 2018-02-07]. Dostupné z: https://www.systemonline.cz/prehled-informacnich-systemu/erp-systemy/*

[12] *Aktuální trendy vývoje českého ERP trhu (2. vydání) [online]. [cit. 2018-02-07]. Dostupné z: http://www.cvis.cz/hlavni.php?stranka=novinky/clanek.php&id=764*

[13] *Priority pro výběr ERP systému [online]. [cit. 2018-02-07]. Dostupné z: http://cfoworld.cz/analyzy/priority-pro-vyber-erp-systemu-4288*

[14] *Enterprise resource planing [online]. [cit. 2018-02-11]. Dostupné z: http://www.tech-faq.com/erp.html*

[15] *Microsoft vs. SAP vs. Oracle 2017 [online]. [cit. 2018-02-11]. Dostupné z: https://ax-dynamics.com/resources/microsoft-vs-sap-vs-oracle-2017*

[16] *Produkty společnosti HELIOS. HELIOS [online]. [cit. 2018-03-21]. Dostupné z: https://www.helios.eu/produkty/*

[17] *Global shipments forecast for tablets, laptobs and mobile [online]. [cit. 2018-03-24]. Dostupné z: https://www.statista.com/statistics/272595/global-shipments-forecast-for-tablets-laptops-and-desktop-pcs/*

[18] *Number of smartphone users worldwide [online]. [cit. 2018-03-24]. Dostupné z: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/*

[19] *ComScore [online]. [cit. 2018-04-01]. Dostupné z: https://www.comscore.com/*

[20] *Android, oficiální stránky [online]. [cit. 2018-04-03]. Dostupné z: https://www.android.com/*

[21] *The history of Android OS: its name, origin and more [online]. [cit. 2018-04-04]. Dostupné z: https://www.androidauthority.com/history-android-os-name-789433/*

[22] *Android – Developer Guide [online]. [cit. 2018-04-04]. Dostupné z: https://developer.android.com/guide/*

[23] *LACKO, Ľuboslav. Mistrovství – Android. Přeložil Martin HERODEK. Brno: Computer Press, 2017. Mistrovství. ISBN 978-80-251-4875-4.*

[24] *SOAP [online]. [cit. 2018-04-05]. Dostupné z: https://www.w3schools.com/xml/xml_soap.asp*

[25] *OpenUI5 [online]. [cit. 2018-04-05]. Dostupné z: https://openui5.org/*

[26] *SAP Fiori [online]. [cit. 2018-04-05]. Dostupné z: https://www.sap.com/products/fiori.html*

[27] *ANUBHAV, Pandey. OData – Everything that you need to know (Part 1) [online]. 2016 [cit. 2018-05-14]. Dostupné z: https://blogs.sap.com/2016/02/08/odata-everything-that-you-need-to-know-part-1/*

[28] *ANUBHAV, Pandey. OData – Everything that you need to know (Part 2) [online]. 2016 [cit. 2018-05-14]. Dostupné z: https://blogs.sap.com/2016/02/08/odata-everything-that-you-need-to-know-part-2/*

[29] *ANUBHAV, Pandey. OData – Everything that you need to know (Part 3) [online]. 2016 [cit. 2018-05-14]. Dostupné z: https://blogs.sap.com/2016/02/08/odata-everything-that-you-need-to-know-part-3/*

[30] *ANUBHAV, Pandey. OData – Everything that you need to know (Part 4) [online]. 2016 [cit. 2018-05-14]. Dostupné z: https://blogs.sap.com/2016/02/08/odata-everything-that-you-need-to-know-part-4/*

[31] *SAPUI5 [online]. [cit. 2018-04-16]. Dostupné z: https://sapui5.hana.ondemand.com/#/topic*

[32] *HASEMAN, William D. a Ross. HIGHTOWER. Mobile development for SAP. Bonn: Galileo Press, 2013. ISBN 9781592296477.*

[33] *ČÁPKA, David. 1. díl - Popis MVC architektury [online]. 2014 [cit. 2018-05-14]. Dostupné z: https://www.itnetwork.cz/php/mvc/objektovy-mvc-redakcni-system-v-php-popis-architektury*

Seznam obrázků

Obrázek 1 - transparetní a databázová tabulka 22
Obrázek 2 - komunikace mezi ABAP programem a databází 22
Obrázek 3 - Fiori na všech typech zařízení 35
Obrázek 4 - relační vztahy tabulek v databázi 39
Obrázek 5 - ukázka wsdl 42
Obrázek 6 - ukázka odpovědi SOAP 43
Obrázek 7 - ukázka požadavku SOAP 43
Obrázek 8 - grafické znázornění aplikace – v levé části Timesheet, vpravo nahoře načítání a vpravo dole kontrolní výstup z tabulky 48
Obrázek 9 - grafické rozložení aplikace Skladiště 49
Obrázek 10 - obecný nákres logiky aplikace 50
Obrázek 11 - struktura aplikace 52
Obrázek 12 - GUI pro Fiori 53

Seznam zdrojových kódů

Zdrojový kód 1 - ukázka programu v jazyce ABAP 21
Zdrojový kód 2 - požadavek pro webovou službu SOAP 45
Zdrojový kód 3 - posílání zpráv do prostoru Handleru 46
Zdrojový kód 4 - zjištění identity uživatele 53

Seznam grafů

Graf 2 - Prodeje mobilních zařízení a stolních počítačů 25
Graf 1- Počet uživatelů Desktop X Mobile 25
Graf 3 - Počet uživatelů chytrých telefonů 26
Graf 4 - podíl operačních systémů v chytrých telefonech 29
Graf 5 - aktuální stav používanosti verzí Androidu 30
Graf 6 - komunikace mezi SAP serverem a mobilním zařízení s OS Android .. 38

A Zdrojové kódy pro ABAP moduly

A.1 ZBPH_WRITE_TIME

```
FUNCTION ZBPH_WRITE_ELECTRO.
*"----------------------------------------------------------------------
*"*"Lokální rozhraní:
*"  IMPORTING
*"     VALUE(ELID) TYPE  CHAR0032
*"     VALUE(EVALUE) TYPE  CHAR0032
*"----------------------------------------------------------------------

DATA row type ZBPH_ELECTRO.

    row-ZBPH_ELID = ELID.
    row-ZBPH_EVALUE = EVALUE.

 CALL FUNCTION 'NUMBER_GET_NEXT'
   EXPORTING
     nr_range_nr                   = '01'
     object                        = 'ZBPH_ID'
*    QUANTITY                      = '1'
*    SUBOBJECT                     = ' '
*    TOYEAR                        = '0000'
*    IGNORE_BUFFER                 = ' '
   IMPORTING
     NUMBER                        = row-ZBPH_ID
*    QUANTITY                      =
*    RETURNCODE                    =
*  EXCEPTIONS
*    INTERVAL_NOT_FOUND            = 1
*    NUMBER_RANGE_NOT_INTERN       = 2
*    OBJECT_NOT_FOUND              = 3
*    QUANTITY_IS_0                 = 4
*    QUANTITY_IS_NOT_1             = 5
*    INTERVAL_OVERFLOW             = 6
*    BUFFER_OVERFLOW               = 7
*    OTHERS                        = 8
           .
 IF sy-subrc <> 0.
* Implement suitable error handling here
 ENDIF.

INSERT ZBPH_ELECTRO FROM row.



ENDFUNCTION.
```

A.2 ZBPH_HASH_PASSWD

```
FUNCTION ZBPH_HASH_PASSWD.
*"----------------------------------------------------------------------
*"*"Lokální rozhraní:
*"  IMPORTING
*"     VALUE(PASSWD) TYPE  STRING OPTIONAL
*"     VALUE(LOGIN) TYPE  STRING OPTIONAL
*"  EXPORTING
*"     VALUE(VALID) TYPE  FLAG
*"----------------------------------------------------------------------
DATA: hash TYPE string.
DATA: algo TYPE string.
DATA: kont type CHAR0064.

DATA: kont2 type CHAR0064.
DATA: hashoriginal TYPE CHAR0064.


TRY.
    algo = 'SHA256'.

    cl_abap_message_digest=>calculate_hash_for_char( EXPORTING
                                                       if_algorithm = algo
                                                       if_data = PASSWD
                                                     IMPORTING
                                                       ef_hashstring = hash )
.

  CATCH cx_root INTO DATA(e_text).
    WRITE: / e_text->get_text( ).
ENDTRY.
kont2 = hash.
kont = LOGIN.
SELECT SINGLE ZBPH_PASSWD FROM ZBPH_USERS INTO hashoriginal WHERE ZBPH_EMAIL
= kont .



if hashoriginal = kont2.
  valid = 'X'.
else.
  valid = ' '.
endif.



ENDFUNCTION.
```

A.3 ZBPH_LASTFIVEROWS

```
FUNCTION ZBPH_LASTFIVEROWS .
*"----------------------------------------------------------------------
*"*"Lokální rozhraní:
*"  EXPORTING
*"     VALUE(STAB) TYPE  STRINGTAB
*"----------------------------------------------------------------------
 DATA: table TYPE standard table of ZBPH_TIME_DATA,
       struct  like line of table,

       stringstruct type string.


  SELECT COUNT( * ) FROM ZBPH_TIME_DATA.

  IF sy-dbcnt > 0 .

    SELECT * from ZBPH_TIME_DATA into table table.

      SORT table by ZBPH_ID descending.

    do 5 times.
      read table table into struct index sy-index .
      if sy-subrc = 0.
        "abz mohli byt meyerz musi se dealt pres cntacenate jiank neumi
          stringstruct = struct-zbph_id && struct-ZBPH_PLACES && struct-
ZBPH_CUSTOMER && struct-ZBPH_ACTIVITY && struct-zbph_TIME && struct-
ZBPH_TEXT && struct-ZBPH_HOURS && struct-ZBPH_USER.
          append stringstruct to STAB.
      endif.

    enddo.
  endif.


ENDFUNCTION.
```

A.4 ZBPH_READTABLE

```
FUNCTION zbph_readtable.
*"----------------------------------------------------------------------
*"*"Lokální rozhraní:
*"  IMPORTING
*"     VALUE(I_TABNAME) TYPE  DD02L-TABNAME
*"     VALUE(I_HEADER) TYPE  FLAG DEFAULT SPACE
*"     VALUE(I_DELIMITER) TYPE  CHAR01 DEFAULT ';'
*"     VALUE(I_WHERE) TYPE  STRING OPTIONAL
*"  EXPORTING
*"     VALUE(STAB) TYPE  STRINGTAB
*"----------------------------------------------------------------------
  DATA: table TYPE REF TO data,
          cislo(3) TYPE c,
          stred TYPE flag VALUE '',
          start TYPE flag VALUE '',
          kont TYPE string,
          pom TYPE string.
  DATA strucdescr TYPE REF TO cl_abap_structdescr.
  FIELD-SYMBOLS <itab> TYPE ANY TABLE.
  FIELD-SYMBOLS <value> TYPE any.

  SELECT COUNT( * ) FROM dd02l WHERE tabname = i_tabname .

  IF sy-dbcnt > 0 .
    CREATE DATA table TYPE STANDARD TABLE OF (i_tabname).

    ASSIGN table->* TO <itab>.

    SELECT * FROM (i_tabname) INTO TABLE <itab>.

    strucdescr ?= cl_abap_typedescr=>describe_by_name( i_tabname ).

    DATA(componenttab) = strucdescr->get_components( ).
    IF NOT i_header IS INITIAL .
      LOOP AT componenttab INTO DATA(head).
        IF stred = 'X'.
          CONCATENATE pom i_delimiter INTO pom .

        ENDIF.
        stred = 'X'.
        CONCATENATE pom head-name INTO pom.
      ENDLOOP.
      stred = ''.
      APPEND pom TO stab.
    ENDIF.
    CLEAR pom.

    LOOP AT <itab> ASSIGNING FIELD-SYMBOL(<fs_dyn>).
      LOOP  AT componenttab ASSIGNING FIELD-SYMBOL(<fs_comp>).
        ASSIGN COMPONENT <fs_comp>-name OF STRUCTURE <fs_dyn> TO <value>.
        IF <value> IS ASSIGNED.
          IF stred = 'X'.
            CONCATENATE pom i_delimiter INTO pom .
          ENDIF.
          stred = 'X'.
          kont = <value>.
          CONCATENATE pom kont  INTO pom.
        ENDIF.
      ENDLOOP  .
      stred = ''.
      APPEND pom TO stab.
      CLEAR pom.
    ENDLOOP.
  ELSE.
    APPEND 'Tabulka neexistuje' TO stab.
  ENDIF.
ENDFUNCTION.
```

A.5 ZBPH_ERITE_TIME

```
FUNCTION ZBPH_WRITE_TIME.
*"----------------------------------------------------------------------
*"*"Lokální rozhraní:
*"  IMPORTING
*"     VALUE(CUSTOMER) TYPE  INT4
*"     VALUE(ACTIVITY) TYPE  INT4
*"     VALUE(PLACES) TYPE  INT4
*"     VALUE(HOURS) TYPE  /1FB/MD_________M8300GL
*"     VALUE(MESSAGE) TYPE  /BA1/F4_DTE_BADI_IMPL_LONGTEXT
*"     VALUE(USER) TYPE  CHAR0032
*"     VALUE(DATE) TYPE  DATUM
*"----------------------------------------------------------------------

DATA row type ZBPH_TIME_DATA.

    row-ZBPH_PLACES = places.
    row-ZBPH_CUSTOMER = customer.
    row-ZBPH_ACTIVITY = activity.
    row-ZBPH_TIME = date.
    row-ZBPH_TEXT = message.
    row-ZBPH_USER = user.
    row-ZBPH_HOURS = hours.

 CALL FUNCTION 'NUMBER_GET_NEXT'
   EXPORTING
     nr_range_nr                   = '01'
     object                        = 'ZBPH_ID'
*    QUANTITY                      = '1'
*    SUBOBJECT                     = ' '
*    TOYEAR                        = '0000'
*    IGNORE_BUFFER                 = ' '
   IMPORTING
     NUMBER                        = row-ZBPH_ID
*    QUANTITY                      =
*    RETURNCODE                    =
*  EXCEPTIONS
*    INTERVAL_NOT_FOUND            = 1
*    NUMBER_RANGE_NOT_INTERN       = 2
*    OBJECT_NOT_FOUND              = 3
*    QUANTITY_IS_0                 = 4
*    QUANTITY_IS_NOT_1             = 5
*    INTERVAL_OVERFLOW             = 6
*    BUFFER_OVERFLOW               = 7
*    OTHERS                        = 8
           .
 IF sy-subrc <> 0.
* Implement suitable error handling here
 ENDIF.

INSERT ZBPH_TIME_DATA FROM row.

ENDFUNCTION.
```

B Zdrojové kódy pro ABAP REST

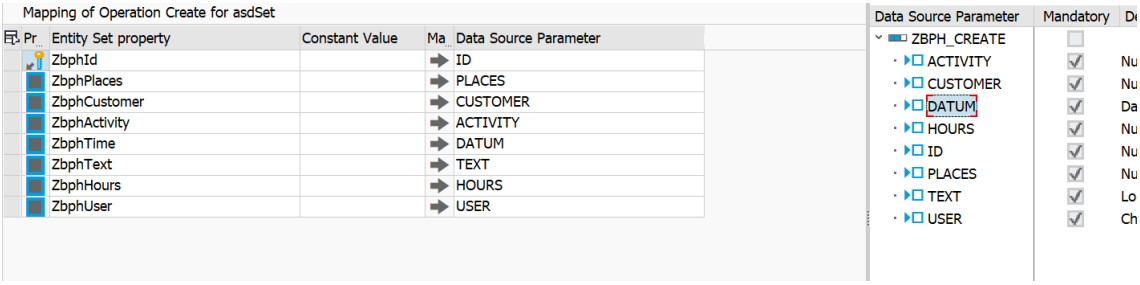
B.1 GET_ENTITYSET

Po generování *Runtime artifacts* je nutno přepsat metody. V našem případě to byla pouze metoda GET_ENTITY_SET. Ve všech příkladech je obsah této funkce stejný, liší se pouze jménem tabulky, se kterou pracuje.

```
* <SIGNATURE>---------------------------------------------------------------
----------------------+
* | Instance Protected Method ZCL_ZBPH_PRODUCT_DPC->PRODUCTSSET_GET_ENTITYSET
* +-------------------------------------------------------------------------
----------------------+
* | [--->] IV_ENTITY_NAME                 TYPE        STRING
* | [--->] IV_ENTITY_SET_NAME             TYPE        STRING
* | [--->] IV_SOURCE_NAME                 TYPE        STRING
* | [---
>] IT_FILTER_SELECT_OPTIONS       TYPE        /IWBEP/T_MGW_SELECT_OPTION
* | [--->] IS_PAGING                      TYPE        /IWBEP/S_MGW_PAGING
* | [---
>] IT_KEY_TAB                     TYPE        /IWBEP/T_MGW_NAME_VALUE_PAIR
* | [---
>] IT_NAVIGATION_PATH             TYPE        /IWBEP/T_MGW_NAVIGATION_PATH
* | [---
>] IT_ORDER                       TYPE        /IWBEP/T_MGW_SORTING_ORDER
* | [--->] IV_FILTER_STRING               TYPE        STRING
* | [--->] IV_SEARCH_STRING               TYPE        STRING
* | [---
>] IO_TECH_REQUEST_CONTEXT        TYPE REF TO /IWBEP/IF_MGW_REQ_ENTITYSET(opt
ional)
* | [<---
] ET_ENTITYSET                    TYPE        ZCL_ZBPH_PRODUCT_MPC=>TT_PRODUCT
S
* | [<---
] ES_RESPONSE_CONTEXT             TYPE        /IWBEP/IF_MGW_APPL_SRV_RUNTIME=>
TY_S_MGW_RESPONSE_CONTEXT
* | [!CX!] /IWBEP/CX_MGW_BUSI_EXCEPTION
* | [!CX!] /IWBEP/CX_MGW_TECH_EXCEPTION
* +-------------------------------------------------------------------------
------------</SIGNATURE>
  method PRODUCTSSET_GET_ENTITYSET.
    select * from zbph_products
      into corresponding fields of table @et_entityset.
  endmethod.
```

B.2 CREATE

Pro tuto operaci byl zvolen postup automatického nastavení. Skript byl tedy nastaven podle následujícího screenshot.



Automatické generování objektů vygeneruje volání zadané funkce a předá parametry.

C Obsah přiloženého CD

Přiložené CD obsahuje:

- bakalářskou práci ve formátu PDF
- složku Webová služba, která obsahuje celý projekt pro Android
- složku OData, která obsahuje zdrojové soubory pro aplikace na Fiori