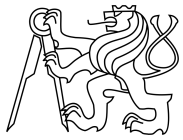
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Výpůjční systém s podporou inventarizace
Student:	Bc. Jan Mráček
Vedoucí:	Ing. Petr Pulc
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2018/19

Pokyny pro vypracování

Navrhněte, implementujte a uživatelsky otestujte systém pro výpůjčky a inventarizaci vybavení v testovací laboratoři společnosti Siemens s.r.o.

Tento systém bude sloužit jako podpora každodenní činnosti uživatelů testovací laboratoře. Proto musí být uživatelsky přívětivý, dostatečně výkonný, robustní a spolehlivý.

1. Proveďte rešerši současných možností systémů výpůjčky a inventarizace.
2. Analyzujte požadavky a potřeby uživatelů testovací laboratoře a navrhněte koncept systému.
3. Vyberte vhodné nástroje pro systém výpůjčky a inventarizace (mobilní zařízení, QR kódy, RFID ...).
4. Navrhněte architekturu systému jak pro stranu klienta, tak pro stranu serveru.
5. Implementujte tento systém jako prototyp řešení.
6. Proveďte vhodné testování prototypu a analyzujte zpětnou vazbu od uživatelů.

Seznam odborné literatury

MOHAMED Y. JABER. Inventory management non-classical views. Boca Raton: CRC Press, 2009.
ISBN 978-1-4200-7998-2.
MÜLLER, Max. Essentials of inventory management. 2nd ed. New York: AMACOM, 2011.
ISBN 0-8144-1655-1.
ISO/IEC 18004: Information technology -- Automatic identification and data capture techniques -- QR Code 2005 bar code symbology specification. 2006.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 21. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Výpůjční systém s podporou inventarizace

Bc. Jan Mráček

Vedoucí práce: Ing. Petr Pulc

9. května 2017

Poděkování

Rád bych na tomto místě poděkoval Ing. Janu Vernerovi ze společnosti Siemens za možnost pracovat na tomto pro mě obohacujícím projektu a za spoustu cenných rad, kterých se mi průběžně dostávalo. Dále bych také rád poděkoval Ing. Petru Pulcovi za bezproblémovou spolupráci při vedení této diplomové práce. V neposlední řadě bych také rád ocenil lidi z mého blízkého okolí, kteří pro mě měli spoustu pochopení a slov podpory ve chvílích, kdy jsem se naplno věnoval této práci.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 9. května 2017 .....

Odkaz na tuto práci

Mráček, Jan. *Výpůjční systém s podporou inventarizace.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Výpůjční systém s podporou inventarizace je webová aplikace pro podporu procesů souvisejících se správou zařízení, zvláště pak zařízení v laboratořích. Tato diplomová práce se zabývá nahrazením několika původních neefektivních procesů modernějším systémem, který by splňoval nejnovější požadavky, které jsou na něj od uživatelů kladeny. Kromě výpůjčního systému a inventarizace aplikace také podporuje efektivní řízení elektrorevizí. Dále jsou také zanalyzovány a navrženy další funkce určené pro budoucí verze.

Klíčová slova Webová aplikace, Vaadin, Siemens, Výpůjční systém, Inventarizace, Elektrorevize, QR kód, Čárový kód

Abstract

Lending Management System With Inventorying Support is a web application for endorsing of processes related to device management, especially for devices in various laboratories. This master thesis covers replacing of several original inefficient processes by modern system which would comply with up-to-date demands placed by users. This application also supports electronic revision management. Additional functions intended for future versions are being analysed and designed.

Keywords Web application, Vaadin, Siemens, Lending management system, Invetorying, Electronics revision, QR code, Barcode

Obsah

Seznam obrázků

Úvod

V dnešní době si už u mnohých činností málokdo dokáže představit, jaké to bylo v dobách, kdy nám ještě internet s jejich řešením nebyl nápomocen. Ať už se jedná o vyhledávání jízdních řádů, instant messaging nebo třeba kontrolu stavu účtu v zapadlé vesnické knihovně, máme díky moderním technologiím možnost získávat nejrůznější informace a nad to s nimi i následně dále pracovat. Moderní technologie a jejich propojení skrze datové sítě zkrátka vytváří základ našich každodenních životů a usnadňuje nám spoustu činností, nad kterými již nemusíme zdlouhavě přemýšlet, ale spíše na nich stavět a využít je v náš prospěch - ať už ke zvyšování naší produktivity nebo pouze k usnadnění každodenních rutinních činností.

Ne ve všech oblastech je ovšem situace tak příznivá. Existuje veliká spousta pracovních úkonů, které jsou v současnosti stále vykonávány postaru, nemoderně a neefektivně. V některých případech může chybět vůle, někdy naproti tomu může být omezujícím faktorem to, že lidé vykonávají spoustu věcí v zajetých kolejích a třeba ani nemají přílišnou motivaci cokoliv měnit. Existují také případy, kdy se v průběhu času rozrůstá komplexita dané problémové domény, a to, co krásně fungovalo pár let zpátky s několika málo zaměstnanci, může být zcela nevyhovující pro větší a dynamičtější pracovní týmy v současnosti.

I toto je mimo jiné případ, který úzce souvisí se společností Siemens, kde se postupně rozrůstaly vývojové laboratoře. Procesy, které mohly v prvopočátku fungovat na kamarádské bázi mezi několika málo lidmi, kteří se mezi sebou znali, začaly být z ničeho nic nedostačující. I proto se začalo uvažovat o vzniku systému, který by podpořil řešení několika nejpalčivějších problémů, se kterými se zaměstnanci dennodenně potkávají. V ideálním případě by jim tento systém měl poskytnout všechny informace v přehledné a konzistentní formě na jednom místě a také by je měl podporovat v jejich každodenních činnostech. A o jednom z těchto systémů, který by mohl pomoci mimo jiné se zvýšením produktivity, je i tato práce.

Kapitola 1

Cíl práce

Aplikace pro výpůjční systém s podporou inventarizace by měla být dostupná všem zaměstnancům a pracovníkům laboratoří společnosti Siemens napříč různými odděleními a měla by podporovat a zjednodušovat jak jejich každodenní činnosti, tak úkoly, které se vykonávají jednou za delší časový úsek. Přínosem by mělo být sjednocení vedení záznamů o výpůjčkách do jednoho systému, dostupnost všech potřebných a relevantních informací na jednom místě a následně také využití modernějšího přístupu při každoročním provádění inventarizace. To vše by mělo mimo zjednodušení těchto činností vyústit ve vznik technického základu pro nový systém, nad kterým v budoucnu nebude problém doplňovat další funkcionalitu a nadále zlepšovat a zkvalitňovat podporu vykonávání procesů souvisejících s touto problematikou.

Následujícím seznamem dílčích úkolů se tato diplomová práce zabývá, ať už jejich analýzou, popisem nebo návrhem, a v řadě případů i implementací jejich řešení:

Rešerše současných možností systému výpůjčky a inventarizace

Problematika řízení výpůjček není žádnou unikátní záležitostí. Naopak již dnes existuje množství systémů, které výpůjčky v nejrůznějších oblastech podporují - stačí se například vydat do téměř kterékoliv knihovny. Existuje proto předpoklad, že již dnes lze nalézt a využívat systémy, které by se daly využít buďto přímo nebo v nich alespoň nalézt inspiraci pro tvorbu nového systému na míru požadovaným potřebám. Tato část tedy představí v současnosti využívané aplikace, jejich výhody a nevýhody, a také jejich stručný popis. Dále by bylo vhodné také ve zkratce nastínit, jak by tyto aplikace mohly být nápomocny pro potřeby této práce.

Analýza požadavků a potřeb uživatelů testovací laboratoře

Aby bylo možné vytvořit jednotnou aplikaci, která by byla využívána napříč různými odděleními, je v první řadě nutné, aby ji chtěli používat samotní uživatelé. K tomu je zapotřebí zanalyzovat jejich potřeby a přání, které by se následně mohly přetavit ve funkční a intuitivní systém, který nabídne průřezově co nejvíce prvků a funkcionalit, které uživatelé ocení. Tato část se tedy pokusí zachytit pokud možno co nejpřesněji očekávání a tužby samotných uživatelů, aby na tomto základu mohly pevně stát další části týkající se návrhu a implementace samotného systému.

Zde se dá jít ještě dál a zamyslet se i nad různými drobnostmi, které mohou uživatelům ulehčit práci s celým systémem. Dost často se uživatelé soustředí především na funkce, které chtějí mít podporované, ale oblast možností, jak jich dosahovat, nemusí být maximálně efektivně zanalyzovaná. Proto může být v některých případech přínosné, když se v rámci fáze návrhu po zvolení konkrétní technologie vrátíme zpět k analýze a upřesní se, jaké z možností poskytnutých zvolenou technologií je vhodné vybrat k maximálnímu možnému uspokojení uživatelských požadavků.

Výběr vhodného nástroje pro systém výpůjčky a inventarizace

Výběr vhodného nástroje je pro účely této práce také velmi důležitý. Ten bude určovat přibližný směr, kterým se bude aplikace ubírat. Dále také bude předurčovat, jak složité může v budoucnu být doplňovat novou funkcionalitu a celkově vytvářet nové verze. Dnes totiž rozhodně neplatí, že vznikají systémy, které se bez dalších úprav mohou využívat dlouhá léta. Je potřeba jít s technickým pokrokem, a aby využívání aplikace bylo stále lákavé a přínosné, je vhodné jak doplňovat novou funkcionalitu, tak upravovat, optimalizovat a inovovat tu současnou.

Další oblastí, na kterou je potřeba klást důraz, je vyvážení mezi přínosem zvolených metod a technologií a jejich budoucí využitelností a podporou. Pokud by byl například zvolen framework, u kterého je již dnes jasné, že nebude dále vyvíjen, podporován a spravován komunitou uživatelů, mělo by to zcela určitě představovat další z důležitých aspektů v rámci průběhu rozhodování o výběru technologií, na kterých projekt bude dalších několik let stát. S tím souvisí nejen budoucí, ale i aktuální stav. Systém, který je špatně zdokumentovaný, případně trpí absencí dostatečné podpory komunity uživatelů, kteří s ním pracují, je mnohem složitější na implementaci a následnou správu.

Návrh architektury systému

Návrh architektury systému úzce souvisí s výběrem vhodného nástroje z předchozí části. Ten totiž tuto architekturu do značné míry určuje. Samotná architektura musí samozřejmě reagovat také na požadavky sesbírané v rámci analýzy a vhodně je zaintegrovat do připravovaného celku. I během této fáze

je vhodné dále komunikovat se zadavatelem a průběžně odstraňovat nově objevené nejasnosti v zadání. Ačkoliv cílem analýzy je vyjasnit maximální množství informací a požadavků hned na začátku, málokdy se stane, že je zadání tak exaktní, že není potřeba dále upřesňovat.

Mimo to je třeba věnovat pozornost nejen současným požadavkům uživatelů na vyvíjený systém, ale zamyslet se i nad požadavky, které mohou přijít ať už v bližší, nebo i vzdálenější budoucnosti. Je potřeba podchytit takové věci, jako budoucí rozšiřitelnost, či znovupoužitelnost kódu, ale také například zda vybraná technologie dostačuje budoucímu očekávanému zaměření zadavatele. Opomenout by se nemělo ani budoucích zapojení požadavků správců systému a dalších lidí zainteresovaných do jeho rozvoje.

Implementace systému jako prototypu řešení

Implementace systému by měla být završením všech předchozích částí a jejich transformace do hmatatelného výsledku. Měl by být kladen důraz na dodržení všech postupů softwarového inženýrství. Jednat se může například o znovupoužitelnost jednou vytvořeného kódu, psaní komentářů pro usnadnění budoucího vstupu do projektu dalším lidem nebo například průběžné testování popsané v kapitole 5.2.

Výstup z této části rozhodne o výsledcích a přínosech této práce a o jejím úspěchu mezi samotnými uživateli. Aby k tomu vůbec mohlo dojít, je nutné mít předchozí části zpracované co možná nejpřesněji, protože jen na pevných základech z analýzy a následného návrhu lze vytvořit skutečně kvalitní, stabilní a používaný systém, který splní většinu na něj kladených nároků.

Vzhledem k obrovskému množství požadavků na systém není cílem této práce návrh a implementace všech požadovaných součástí. Tato práce spíše nastíní nový technický směr umožňující další rozvoj, pokusí se převést většinu staré funkcionality do nové a přidat několik nových technických řešení, která by měla sloužit jako ukázka reálných možností. Rozhodně není ambicí od základů zcela nahradit původní systém a zároveň nabídnout zapracované veškeré zanalyzované požadavky.

Testování prototypu a analýza zpětné vazby od uživatelů

Na závěr každé realizace nového softwarového řešení je potřeba systém otestovat a zanalyzovat zpětnou vazbu od uživatelů. Zde se kruh uzavírá a napojuje se zpět k prvnímu kroku týkající se analýzy. Zde se totiž střetávají předpokládaná očekávání a reálné výsledky. Na jejich základně můžou být některé připomínky (typicky závažného charakteru) rovnou zaneseny do systému.

Vhodnější ovšem je označit, co se podařilo a díky tomu splňuje uživatelská očekávání a na co by mělo být pamatováno při další iteraci vývojového cyklu i s volitelným pořadím dalších kroků. To je sice již nad rámec testování, ale může to výrazně pomoci při dalším pokračování vývoje.

Kapitola 2

Analýza

V rámci analýzy je vhodné zaměřit se na detailnější průzkum současného řešení systému výpůjček a inventarizace. Mezi zajímavé výstupy patří například to, kdo se systémem pracuje, proč a v čem současný stav nevyhovuje, nebo jaká jsou očekávání budoucího vývoje. Dále je potřeba zjistit, jestli už existují nějaké dostupné nástroje, které by šly využít v návrhové fázi nového systému. Je potřeba taktéž hovořit s lidmi, kteří mají předpoklad s nově vzniklým systémem co nejčastěji pracovat a následně zanalyzovat jejich pohled na uchopení řešeného problému. Nakonec to budou totiž ti, kdo rozhodne o úspěchu, nebo neúspěchu nově vytvořené aplikace.

2.1 Současný stav

Společnost Siemens patří zcela bez pochyby do kategorie velkých a zavedených firem. Ty mají oproti malým, případně nedávno vzniklým společnostem několik nevýhod:

1. Díky své velikosti vzniká potřeba vytvářet rozvětvenou organizační strukturu. Za tohoto stavu pak dochází k situacím, kdy zaměstnanci z různých oddělení sdílejí jak společné prostory, tak i společná elektronická zařízení a nástroje. To vyžaduje kvalitní komunikaci mezi jednotlivými odděleními, aby nedocházelo k informačním šumům a nedorozuměním, která jsou při takto členěné organizační struktuře ne zcela výjimečná.

2. Z minulosti si s sebou nesou již zajeté procesy, případně celé systémy, které není úplně jednoduché měnit. Důvodem může být ať už pracovní přetížení zaměstnanců a nedostatek kapacit pro změnu stavu vlastními silami, těžko zdůvodnitelné ekonomické hodnocení investic do zlepšení stavu, nebo třeba jen liknavost a nedůslednost, či neprozíravost některých vedoucích pracovníků.

Z analýzy vyplynulo, že pro společnost Siemens je v současné době největší problém to, že si každé oddělení vede evidenci o výpůjčkách zařízení odlišným způsobem. Některá oddělení využívají papírové evidence, jiná používají pro evidenci výpůjček aplikaci *MS Excel*[1], zatímco další oddělení využívají před několika lety vytvořený, ale dále nerozvíjený projekt založený na systému Xataface. To způsobuje jednak některé datové nekonzistence a značnou míru nepřehlednosti. Zmínit je potřeba také snížení pracovní efektivity, protože udržování a využívání několika systémů pro záznamy je mnohem časově náročnější než udržování jednoho systému.

2.1.1 Použité systémy

2.1.1.1 Xataface

Xataface je open source framework založený na jazyku PHP pro vývoj webových aplikací umožňující vývojářům flexibilní přizpůsobování chování a funkcí aplikací, které jsou na něm založeny[1]. Jedná se o nadstavbu nad databází *MySQL*[2] umožňující nad ní vytvořit front-end. Tento framework podporuje generování formulářů, listů nebo položek menu a umožňuje uživatelům interagovat s databází bez nutnosti znát syntaxi jazyka *SQL*[3]. Dále také obsahuje podporu jednoduchých šablon a pluginů[2].

Hlavní výhoda tohoto frameworku, totiž že je určen pro uživatele neovládající databázový dotazovací jazyk SQL, je zároveň jeho největší slabinou. Aby bylo dosaženo jednoduchosti i pro netechnické uživatele, muselo být zvoleno spoustu kompromisů, které systém velmi omezují, co se týká přidávání a úprava nových funkcí. Jak již bylo řečeno, jedná se především o front-end nad databází a tedy chybí podpora pro téměř všechny potřebné funkčnosti mimo práce s databází. Samozřejmě je možné vytvořit je v jazyce PHP, nicméně technická a časová náročnost takového řešení je značně vysoká oproti volbě jiných technologií.

V minulosti tento framework jistě splnil svůj účel, protože mezi požadavky nebylo nic o dalších prvcích a funkcích. Ovšem dnes, s postupným zvyšováním komplexnosti řešené problémové domény, začaly přicházet nové a nové požadavky na potřebu a následné zařazení těchto nových prvků a funkcí do systému. To byl hlavní důvod, proč bylo rozhodnuto posoudit, zda-li framework Xataface bude umožňovat další budoucí rozvoj nebo zda-li bude zvoleno některé z dalších možných technických řešení.

2.1.1.2 MySQL

Jak již bylo zmíněno v předchozí kapitole 2.1.1.1, je framework Xataface postaven nad databází MySQL. Jedná se o populární open-source systém řízení

[1] MS Excel - Aplikace z kancelářského balíku Microsoft Office
[2] MySQL - Systém řízení báze dat vytvořený společností MySQL AB
[3] SQL - Structured Query Language

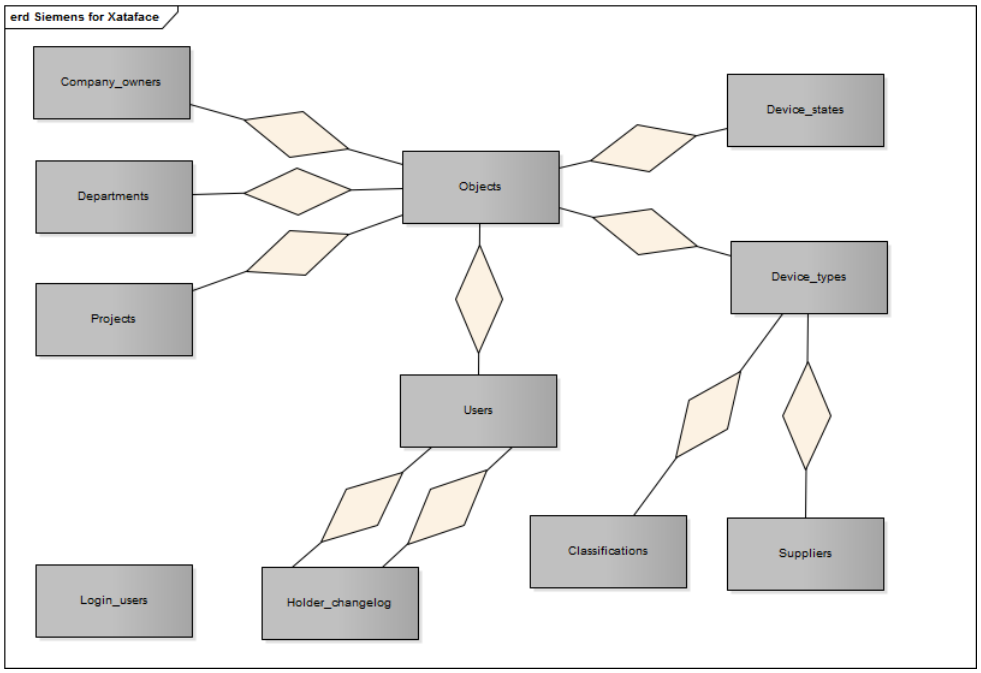
báze dat v kategorii SQL databází. V současnosti je spravována a distribuována společností Oracle. Jedná se o spolehlivý, rychlý a škálovatelný systém, který je velmi jednoduchý na použití[3].

2.1.2 Rozbor aktuálně používaného systému

Pro získání obecného přehledu nad celým systémem složeného z databáze MySQL a front-endového frameworku Xataface se následující podkapitoly budou zabývat detailnějším rozborem jeho jednotlivých částí. V těch se jednak pokusíme sesbírat, ale také stručně vysvětlit všechny důležité existující části používané v současném řešení.

2.1.2.1 Databázové tabulky

- **classifications**
  (rozlišení sledované položky na hardware a software)
- **company_owners**
  (seznam všech vnitřních společností, které mohou vlastnit danou věc)
- **departments**
  (seznam oddělení vlastnících nějaké věci)
- **device_states**
  (stav dané věci)
- **device_types**
  (seznam typů zařízení)
- **holder_changelog**
  (přehled všech změn ve vlastnictví)
- **info**
  (nevyužívaná tabulka)
- **login_users**
  (seznam přihlašovacích údajů uživatelů a přístupových práv k aplikaci)
- **objects**
  (seznam všech zařízení)
- **projects**
  (seznam všech zainteresovaných firemních projektů)
- **suppliers**
  (seznam všech dodavatelů daných zařízení)
- **users**
  (seznam uživatelů a jejich mnoha atributů)



Obrázek 2.1: Diagram struktury propojení databázových tabulek v databázi MySQL využívaných aplikací Xataface

Obrázek 2.1 ilustruje propojení jednotlivých databázových tabulek. Jak je patrné, jejich propojení je celkem jednoduché, což může představovat výhodu pro budoucí rozšiřitelnost. Základem je databázová tabulka Objects reprezentující konkrétní zařízení.

Na levé straně je znázorněno propojení s tabulkami organizační struktury - nejprve jaké firmě spadající pod mateřskou firmu dané zařízení náleží, následované informací příslušnosti ke konkrétnímu oddělení a pro zjemnění granularity je obsažen i údaj o příslušnosti ke konkrétnímu projektu. Oproti tomu na pravé straně jsou naznačeny vazby na stav zařízení (informace o tom, jestli je zařízení dostupné, zapůjčené nebo rozbité) a na příslušnost ke konkrétnímu typu zařízení. Tabulka *Objects* totiž představuje samostatné exempláře, kterých může být od jednoho typu více a které mají i své vlastní atributy. U typu zařízení se dále drží vazby jednak na databázovou tabulku o klasifikaci dané položky (rozlišení na hardware a software) a také o dodavatelích.

Ve spodní části se nachází tabulka *Holder_changelog* obsahující původního a nového uživatele dané položky a datum, kdy ke změně došlo. Poslední zatím nepopsaná tabulka *Login_users* se nachází v levém spodním rohu a obsahuje uživatelská jména, hash hesla pro jednotlivé uživatele a druh oprávnění pro přístup k aplikaci, kde je možné vybírat z nabídky neobsahující žádná oprávnění nebo přístup pouze pro čtení a dále pak nejsilnější oprávnění, umožňující veškeré operace, které systém podporuje.

Mimo to obsahuje databáze ještě několik dalších servisních tabulek pro framework Xataface, ale jejich rozbor není pro tuto práci relevantní. Jednak je velice těžko dohledatelné, k čemu jednotlivé tabulky slouží, a jednak tím, že neobsahují žádná relevantní data, by takové informace nepřinesly nic nového pro řešení tématu této práce.

2.1.2.2 Pohledy na data ve front-endu Xataface

Jak již z předchozího textu vyplývá, je aplikace Xataface (pojmenována podle využitého frameworku) především pohledem na data v databázi. Proto mezi samotnou databází a touto aplikací nejsou žádné výrazné rozdíly. Jediná odlišnost se dá nalézt v zobrazení těch databázových tabulek, které obsahují cizí klíče do dalších tabulek. Zde Xataface provádí jejich sloučení a zobrazí až kompletní data relevantní pro uživatele.

2.1.2.3 Práce s aplikací pouze s využitím čárových kódů

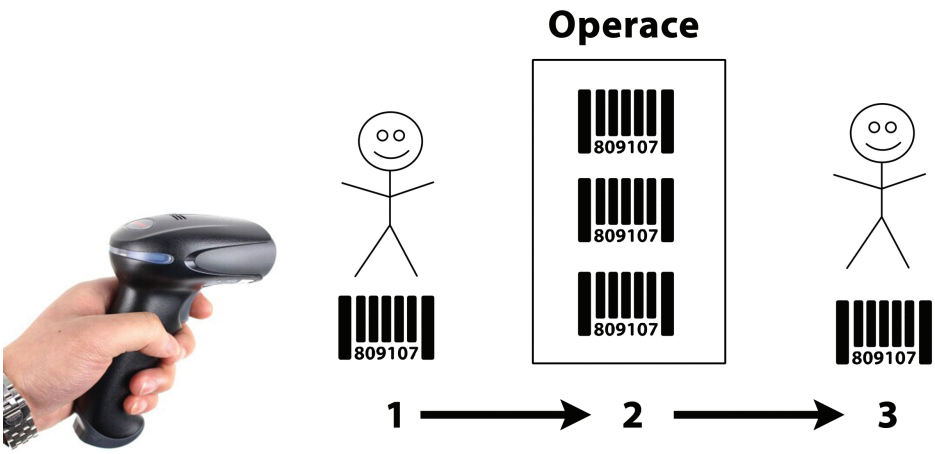
Tím, že Xataface je pouze pohledem na data v databázi, je dán předpoklad k tomu, aby bylo možné vedle sebe provozovat více systémů navázaných právě na tuto jednu databázi. Tato situace z různých historických důvodů nastala právě také u tohoto systému.

V současnosti je možné vidět v provozu ne zcela obvyklý, ale přesto funkční systém, který je znázorněn na obrázku 2.2. Jeho zvláštnost spočívá v tom, že pro jeho ovládání není vůbec potřeba žádné zařízení s obrazovkou, ale všechny běžné operace se vykonávají pouze čtečkou čárových kódů. Předpokladem je, že existují čárové kódy jak pro všechny podporované operace, tak pro všechny uživatele, kteří s tímto systémem pracují.

V laboratořích, kde je tento systém využit, se na jednom specifikovaném místě nachází čtečka čárových kódů. Pracovní postup je takový, že v kroku jedna uživatel načte svůj čárový kód, kterým se vůči systému identifikuje. V druhém kroku je potřeba zvolit nějakou konkrétní operaci. Aby to bylo možné, je vždy na místě se čtečkou čárových kódů vytisknutý jejich seznam. Z toho je možné vybrat jednu operaci a načíst k ní příslušný čárový kód. Tím se odešle vybraná operace. V některých případech se po operaci volí další uživatel. Jedná se kupříkladu o situaci, kde se půjčovaná věc předává z jednoho zaměstnance na druhého. Třetí krok tedy již není povinný, ale některé operace ho vyžadují.

Z analýzy vyplynulo, že uživatelé s tímto systémem nejsou spokojeni hlavně kvůli chybějící zpětné vazbě (lze těžko ověřit, zda se opravdu všechno povedlo správně) a také kvůli absenci přehledu o stavu celého systému. Tato podkapitola proto neslouží jako vyčerpávající a absolutně přesný popis práce s tímto systémem, ani jako jeho uživatelský manuál. Její přínos by měl být spíše v ukázce toho, jak rozličné možnosti pro výpůjční systém byly v součas-

ném stavu využívány a z důvodu budoucího předpokládaného nahrazení nově vyvíjenou aplikací nejsou hlubší detaily analyzovány.



Obrázek 2.2: Diagram znázorňující ovládání současného systému výhradně čárovými kódy[4].

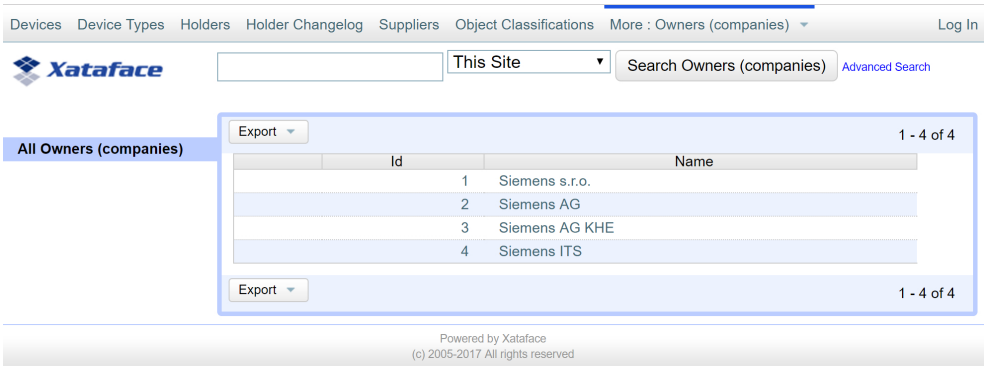
2.1.2.4 Využití excelových tabulek

Ve stručnosti je také nutné zmínit poslední z možností, která se pro výpůjční systém využívá. V tomto případě se veškeré záznamy vedou v jednom *excelovém dokumentu*[4]. Jedná se o velice málo flexibilní a zastaralý systém se spoustou problémů, který si přímo říká o nahrazením něčím modernějším.

2.1.2.5 Výhody současného systému

Mezi největší výhody současného systému patří jeho technická jednoduchost. Pro jakýkoliv záznam není potřeba znát ovládání rozsáhlé aplikace, ale uživatelé si vystačí pouze s editováním řádků jednotlivých tabulek. Nemusí to ovšem dělat přes jazyk SQL, ale vše stačí naklikat vzhledem k využití frontendu Xataface. Systém má také velmi dobrou odezvu, protože se jedná o několik málo tříd napsaných v jazyce PHP. Stejně tak v databázi se jedná maximálně o spojení 3 tabulek s řádově desítkami až stovkami řádků. Zohlední-li se počet uživatelů, kteří s aplikací mohou pracovat, a předpoklad, že každý z uživatelů využije aplikaci maximálně párkrát do týdne, tak si s tím poradí jakýkoli průměrný webový server.

[4]excelový dokument = dokument aplikace Excel z kancelářského balíku Microsoft Office



Obrázek 2.3: Znázornění současné aplikace založené na frameworku Xataface znázorňující, jakým firemním jednotkám mohou patřit jednotlivá zařízení.

2.1.2.6 Nevýhody současného systému

Technická jednoduchost jakožto hlavní výhoda tvoří ovšem zároveň také největší nevýhodu. Jednoduchosti je dosaženo hlavně využitím frameworku Xataface, který je sice velmi dobrým pomocníkem při práci s databázovými daty bez požadavku na nějaké větší funkce a transformace dat, ale pokud potřebujeme vytvořit o něco komplexnější aplikaci, začne zde působit jako úzké hrdlo. Že si uživatelé přejí, aby aplikace byla komplexnější a více toho uměla, vychází z rozboru požadavků uvedených v kapitole 2.2. Jak aplikace implementovaná ve frameworku Xataface vypadá, znázorňuje obrázek 2.5.

Jedná se totiž o velice úzce zaměřený framework, který je samozřejmě možné využít jako základ pro vytvoření nějaké nadstavby. Jak ale z analýzy dále vyplynulo, existují další možnosti, například některé moderní frameworky, které podporují celou škálu dalších operací a ovládacích prvků, které by celý proces značně urychlily. Bylo by proto zbytečné ztrácet čas objevováním již objeveného a bude lepší zbývající kapacitu využít na další vylepšení celé aplikace, kterou uživatelé ocení více.

Další vcelku velikou nevýhodu představuje skutečnost, že některá oddělení současný systém ani nevyužívají a místo toho provizorně pracují s excelovými tabulkami zmíněnými v kapitole 2.1.2.4. U těch samozřejmě pracují jen s daty jejich oddělení, což jednotlivé části dělá téměř úplně nekompatibilními. Také se úplně ztrácí možnost podívat se na data přes celou firmu, což by bylo jednak zajímavé, ale hlavně by to mohlo přinést dosud netušená data a souvislosti mezi nimi.

2.2 Specifikace požadavků

Během analýzy požadavků byla oslovena všechna oddělení, pro která by v případě zájmu mohl být nový systém určený. Jedná se o ta oddělení, která by ho

využila ať už jako drobné vylepšení současného stavu nebo náhradu současného systému. Dokonce by systém mohl v budoucnu sloužit i pro ta oddělení, která by díky podpoře nových funkcí teprve tento systém začala prvně využívat.

Protože se sešlo opravdu velké množství různých nápadů a požadavků na funkce, jejichž návrh a implementace by byly pro účely této diplomové práce příliš rozsáhlé, bylo nutné sáhnout k jejich rozdělení do několika kategorií podle priorit a podle toho, o jak kritické funkce se jedná a jakou váhu jim přikládají jednotlivá oddělení.

2.2.1 Funkční požadavky s vysokou prioritou (F1)

F1.1 Zařízení - základní požadované vlastnosti:

- Povinné unikátní ID
- Jméno vlastníka
- Datum zařazení do aplikace
- Srozumitelné jméno
- Sériové číslo
- Referenční číslo objednávky
- Vlastník (organizace)
- Číslo objednávky
- Jméno výrobce
- Název dodavatele

F1.2 Odstranění zařízení:

- Každé zařízení může být odstraněno.
- Při odstranění zařízení je potřeba zachovat historii (odstranění formou flagu).

F1.3 Výpůjčky - základní funkcionalita:

- Každé zařízení musí mít možnost být půjčeno konkrétním uživatelem.
- Zařízení může být předáno z jednoho uživatele na dalšího bez nutnosti zařízení vracet.
- U každého zařízení je informace o jeho stavu (dostupnosti).
- Musí být uchovány historie výpůjček pro všechna zařízení.

F1.4 Podpora pro QR kódy:

- Každé zařízení obsahuje povinný QR kód.
- U každého zařízení musí být možnost ověřit dostupnost přidělovaného QR kódu.
- Každé zařízení může dostat nový QR kód (změna v průběhu).
- Systém musí být schopen přečíst a zaznamenat existující čárové i QR kódy.
- Čárové a QR kódy mohou být zadávány také ručně.

F1.5 Základní třízení:

- Je možné filtrovat zobrazovaná data.
- Je podporováno klonování a hromadné vkládání položek.

F1.6 Revize elektrických zařízení:

- Každé zařízení může mít údaj o datu poslední provedené revize.
- Zažízení mohou být kategorizována podle revizní periody.

F1.7 User Management:

- Je podporováno více přístupových rolí uživatelů do aplikace.
- Databáze uživatelů pochází z SCD[5].
- Uživatelé jsou třízeni na základě hierarchie z SCD.
- Je podporováno jednoduché přihlašování, například pomocí Siemens Card.
- V případě zapomenutí nebo diskreditace může být nové heslo posláno emailem.

2.2.2 Funkční požadavky s nízkou prioritou (F2)

F2.1 Zařízení - pokročilé vlastnosti:

- Každé zařízení by mělo mít nějaké umístění.
- Každé zařízení by mělo obsahovat fakturační cenu.
- Každé zařízení by mělo evidovat zůstatkovou cenu (odpisy).
- Každé zařízení by mělo obsahovat údaj, které oddělení ho nabylo.

[5]SCD - Siemens Corporate Directory

F2.2 Výpůjčky - pokročilá funkcionalita:

- Musí být možné omezit maximální výpůjční dobu.
- Každému uživateli může být nastaveno oprávnění na umožnění půjčování si věcí.
- Výpůjčku některých speciálních zařízení musí nejprve někdo schválit.

F2.3 Kalibrace:

- Zařízení může obsahovat údaj o datu poslední kalibrace.

F2.4 Propojení s aplikací Majetek:

- Systém musí být propojený s aplikací Majetek.
- Každé zařízení musí obsahovat ID do aplikace Majetek.

F2.5 Vlastní pohledy na data na míru:

- Každý projekt může mít svůj vlastní pohled na data (rozdílné požadavky na záznamy).
- U všech pohledů je možné je upravovat.
- Data pro oddělení Metrology musí mít vlastní pohled.

F2.6 Kompatibilita, Import, Export:

- Je podporován import a export data jednotlivých uživatelů.
- Databázové sloupečky z bývalého systému Xataface (zpětná kompatibilita) musí zůstat zachovány.
- Musí být umožněn kompletní import dat ze současného systému Xataface.

F2.7 Práce se záznamy:

- Systém umožňuje zobrazit data dle vlastních nadefinovaných záznamů.
- Systém umožňuje kategorizace zařízení do kategorií.
- Systém podporuje fulltextové vyhledávání.

F2.8 Hierarchie zařízení:

- Aplikace umožňuje tvorbu hierarchie zařízení (rodič x potomek).

F2.9 Podpora reportů:

- Existuje funkcionalita pro generování reportů specifikovaných uživateli.
- Existuje funkcionalita pro generování grafů o využívanosti zařízení.

F2.10 Notifikace:

- Je podporována funkce pro sledování konkrétního zařízení (notifikace při změně každého stavu).
- Systém umožňuje automatické zasílání emailů o provedených akcích.
- Majitel je upozorněn na končící termín elektro revize a kalibrace.
- Je umožněna eskalace notifikací (upozornění nadřízeného).

F2.11 Hodnocení:

- Jsou podporována hodnocení spolehlivosti uživatelů.

F2.12 Přílohy:

- Každé zařízení může obsahovat fotografie.
- Každé zařízení musí mít možnost uložit k němu manuál.
- Každé zařízení podporuje možnost přiložit k němu další přílohy.

F2.13 Ostatní:

- Aplikace podporuje Intellisence.
- Zařízením lze přiřadit další doplňující data ze SAPu (např. školení).

2.2.3 Nefunkční požadavky

N1 V novém systému využít stejnou databázi, kterou využívá současný systém Xataface. Je možné doplnit nové databázové tabulky, ale není možné měnit ty existující z důvodů zpětné kompatibility.

N2 Umožnit využívání aplikace z mobilních zařízení napříč platformami.

N3 Umožnit využívání aplikace z webového rozhraní.

N4 Umožnit přístup k systému také z veřejně přístupných zařízení.

2.2.4 Zhodnocení požadavků kladených na systém

Celá analýza mezi uživateli byla pojata formou blízkou brainstormingu a i proto se sešlo tolik systémových požadavků. Cílem nebylo pouze definovat nejprioritnější požadavky, která by následně tato práce vyřešila, ale shromáždit co nejvíce všech požadavků, i třeba těch méně důležitých, aby na ně v budoucnu byla již aplikace připravena a další změny pro uspokojení ne tak palčivých přání uživatelů nevyžadovaly naprosto žádný, nebo pouze nezbytně nutný zásah do již existujícího kódu.

Nakonec byly vybrány 3 oblasti, které by v novém systému doplnily již existující funkcionalitu a které tedy budou ještě hlouběji analyzovány:

- Načítání čárových kódů a QR kódů skrz kameru
- Doplnění podpory pro inventarizaci a elektrické revize
- Napojení aplikace na SCD (obdoba Active Directory)

2.3 QR kódy

Všechna zařízení, která náleží ať už přímo laboratořím Siemens, ale i ta další, která budou společně spravována novou aplikací, na sobě mají ve většině případů štítek s QR kódem, v některých pouze štítek s čárovým kódem. Zařízení s čárovými kódy byla využita už v rámci některých oddělení jako součást systému spravující výpůjčky pouze na základně čárových kódů, který byl popsán v kapitole 2.1.2.3.

Obrázek 2.4: Příklad použitého čárového kódu k identifikaci zařízení.

Jelikož ale většina nových zařízení na sobě již štítek s QR kódem má, bylo rozhodnuto, že právě z něj se stane primární identifikátor pro novou aplikaci. Nicméně stále musí být také zaručena podpora i pro původní čárové kódy. Důvodem pro nasazení QR kódů je především umožnění jejich načítání z počítače pomocí webkamery, případně mobilním telefonem pomocí fotoaparátu. To usnadní spoustu činností, které je třeba s daným zařízením vykonávat. Především se jedná o půjčování a vracení věcí, ale zvláště o provádění inventur všech dotčených zařízení. Zde již nebude nutné ručně zaznamenávat každou jednotlivou položku, ale pouze načíst kód mobilním telefonem.

Obrázek 2.5: Příklad použitého QR kódu k identifikaci zařízení.

2.4 Podpora inventarizace a revizí elektronických zařízení

Inventarizace je proces probíhající jednou za rok, který má za úkol zaevidovat všechna zařízení, aby bylo možné mít přehled o případných ztrátách. Při inventarizaci není potřeba pracovat s mnoha údaji, stačí pouze kód zařízení (ať už čárový, nebo QR kód), název a typ zařízení (pro případnou kontrolu, jestli kód někdo nepřelepil na jiné zařízení) a flag informující o tom, zda dané zařízení již prošlo inventurou. Po dokončení inventury by mělo být možné výsledky exportovat jako **.csv*[6] soubor. Na začátku dalšího kola inventur o rok později je potřeba umožnit vynulovat informace o provedené inventarizaci, aby se nepletly aktuální záznamy s těmi z předchozích let.

2.5 Napojení aplikace na SCD

V dnešní době má již téměř každá firma nějakou formou zpracovanou a zpřístupněnou elektronickou evidenci zaměstatnců, respektive uživatelů systému. Ne jinak je tomu v případě společnosti Siemens, která využívá Siemens Corporate Directory (SCD), obdobu Active Directory ve Windows. Jedná se vlastně o implementaci adresářových služeb *LDAP*[7], což je jednoduchý a dobře navržený protokol schopný nejen klást poměrně složité dotazy, ale i vkládat, modifikovat a mazat záznamy[5].

Před nově vytvářenou aplikací tedy stojí výzva napojit ji na ten tento existující systém. V současné aplikaci založené na frameworku Xataface není situace úplně přehledná. Vůbec se zde nevyužívá záznamů z SCD, místo toho existuje databázová tabulka *users* pro evidenci vlastníků i aktuálních držitelů spravovaných zařízení.

[6] *.csv = comma-separated values (souborový formát pro výměnu tabulkových dat)
[7] LDAP = Lightweight Directory Access Protocol

Zároveň existuje také databázová tabulka *login_users*, která s předešlými vůbec nesouvisí. Funguje zde velmi jednoduché přiřazení přístupových rolí do 3 kategorií:

1. **NO ACCESS** - neumožňuje prohlížet ani editovat žádná data
2. **READ ONLY** - umožňuje prohlížení dat bez možnosti jejich editace
3. **ADMIN** - umožňuje vykonávat všechny podporované operace

Vzhledem k absenci provázání s ostatními tabulkami a zvláště pak se systémem SCD zde vůbec nemůže být řeč o dalších požadavcích, které jsou na systém nově kladeny. Zde je možné zmínit například využití hierarchie organizační struktury k tomu, aby vedoucí pracovníci měli přehled o aktivitách svých podřízených v tomto systému a zároveň mohli dostávat například notifikace o změnách nebo provedených akcích.

Kapitola 3

Návrh

Velmi důležitým bodem, který ovlivní celé další směřování tohoto projektu, je výběr použitých technologií pro jeho implementaci. Jde o podobně důležité rozhodnutí, jako jestli při stavbě nového domu zvolíme zděnou stavbu, či dřevostavbu. Leccos se dá sice ladit za běhu, ale základní rysy a směr budou již pevně určeny a pokud přestanou z nějakých důvodů vyhovovat, budou značně obtížně nahrazovány. Proto je třeba této volbě věnovat patřičný čas a značnou pozornost. Při výběru je nutné zohlednit některé z existujících omezení, ať už na straně technických či uživatelských požadavků, či zkušeností s podobnými projekty z minulosti.

3.1 Výběr frameworku

Mluvíme-li v této kapitole o výběru technologií, patří výběr vhodného frameworku jednoznačně k těm nejdůležitějším. Podívejme se proto nejprve na definici slova framework tak, jak ji nabízí server programujte.com[6]. Ten říká, že framework je softwarová struktura, která slouží jako podpora při programování, vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovnu API, návrhové vzory nebo doporučené postupy při vývoji.

3.1.1 Kritéria pro výběr frameworku

Jak ale správný framework vybrat? Jeden z možných přístupů je popsán na webových stránkách softwarové společnosti Skoumal[7]. Podle té hrají nejdůležitější roli tato kritéria:

1. **KOMUNITA** - Užitečný nástroj udělá z frameworku až silná komunita uživatelů kolem něj. Bez ní je totiž řešení každého sebemenšího problému velmi komplikované. Se silnou komunitou bude mnohem snazší

najít řešení na různé problémy, které se můžou a pravděpodobně i budou v průběhu vývoje vyskytovat.

2. **DOKUMENTACE** - Kvalitní dokumentace je jako návod k obsluze nějakého složitějšího zařízení. I bez něj zařízení funguje, jen si každý musí metodou pokusu a omylu objevit žádané funkce. To je jednak velmi časově náročné a pak také spousta zajímavých metod či nástrojů nemusí být vůbec objevena. Pro rychlý a bezstarostný vývoj je kvalitní dokumentace nutnou podmínkou.

3. **PŘÍVĚTIVOST FRAMEWORKU** - Jedná se sice o subjektivní kritérium a člověk si časem zvykne na ledacos, ale pokud framework splňuje všechna očekávání, a ještě je k tomu také přívětivý, je to velmi příjemný bonus, který je také potřeba brát v potaz.

4. **ZPĚTNÁ KOMPATIBILITA** - Je dobré prozkoumat jednotlivé starší verze daného frameworku. Pokud se mezi sebou často a výrazně liší a navíc nejsou zpětně kompatibilní, existuje značný předpoklad, že při budoucím upgradu systému na vyšší verzi frameworku bude docházet k značným a časově náročným problémům.

5. **REFERENCE** - Zde platí asi jediné - nebát se zeptat se lidí, kteří s danými technologiemi, v našem případě frameworky, již pracovali. Nemá smysl po dlouhých měsících programování podruhé objevit kolo, když stejnou cestou před námi již prošlo mnoho lidí, mnohdy i značně zkušenějších. Stačí naslouchat.

V dnešní době se dle mého názoru vůbec nevyplatí pro netriviální projekty uvažovat o řešení napsané čistě v nějakém jazyce bez využití čehokoliv dalšího. Takový přístup může mít svůj význam u specializovaných, případně značně nestandardních projektů, ale pokud se jedná o klasickou aplikaci s webovým uživatelským rozhraním postaveným nad databází, není moc o čem přemýšlet. Výhody využití frameworku zde budou jasně převažovat nad negativy.

Je nutné si uvědomit, že sice jeho komplexita může celý systém zpomalovat, na druhou stranu může nabídnout velké množství funkcí, které ho celý zlepší a zrychlí, ať už jde o kešování, nebo například o vnitřní optimalizaci nejčastěji používaných funkcí.

3.1.2 Výběr programovacího jazyka

Další kritérium, které je třeba brát v potaz, je omezení výběru frameworku lidskými kapacitami. První možností tedy je nejprve vybrat vhodný programovací jazyk, potažmo nějakou jeho frameworkovou nadstavbu a teprve následně shánět programátory a další členy týmu, kteří mají s danými technologiemi již bohaté zkušenosti. Druhou možností je postup opačný, tedy že už na začátku

jsou definováni lidé, kteří na projektu budou pracovat, jejich znalosti a zkušenosti, ke kterým je potřeba vhodně nakombinovat adekvátní programovací jazyk, případně framework.

Lze si těžko představit, že člověk, který v současnosti dělal jednoduché webové stránky v HTML[8] a CSS[9], se přes noc stane odborníkem na programování například v jazyce C#[10]. To je myšlenka sice hezká, ale zároveň ne úplně reálná.

Toto úzce souvisí i s tímto projektem. Budu-li mluvit za sebe, je mnohem lepší pracovat s jazykem, ve kterém se už orientuji a kde už vím, jak překonat některé z počátečních problémů, se kterými se potkává každý začátečník, než se na zelené louce učit sice možná o malinko lepší, ale úplně neznámý jazyk. To je o to důležitější ve chvíli, kdy je na projekt vyhrazeno pouze omezené množství času, což je i kvůli počátečním organizačním problémům přesně případ této práce. Nakonec by se tedy i díky časovým úsporám mělo povést zrealizovat bohatší a funkčně rozsáhlejší, ale zároveň i kvalitativně obdobnou aplikaci.

Volba tedy padla na programovací jazyk Java, konkrétně jeho verzi určenou mimo jiné i pro webové aplikace Jave EE[11]. Ta je postavena na klasické verzi Javy SE[12]. Díky mým zkušenostech v obou těchto verzích existuje vcelku reálný předpoklad, že se projekt nebude zdržovat u základů programovacího jazyka a raději bude možné zbývající čas využít pro vyladění a doplňování celé aplikace o další zajímavé prvky.

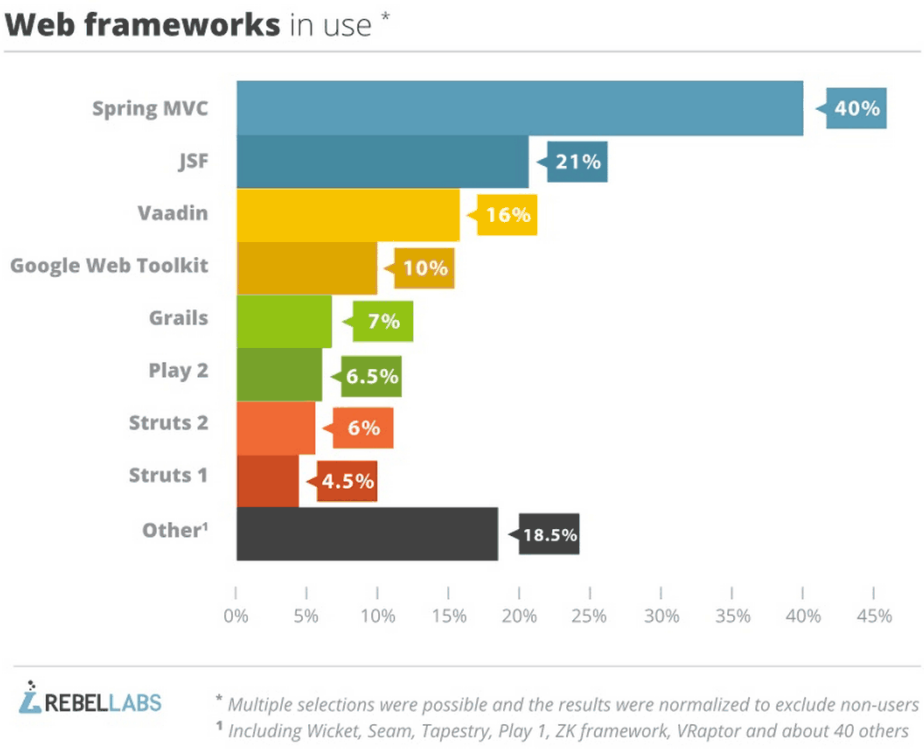
3.1.3 Výběr frameworku pro tento projekt

Při výběru je vhodné začít shromážděním všech možností, aby se bylo možné v celé problematice správně zorientovat. Na obrázku 3.1 lze proto najít porovnání oblíbenosti všech nejčastěji využívaných frameworků dostupných pro jazyk Java. Na medailové pozice dosáhly frameworky Spring MVC, JSF[13] a Vaadin. V různých internetových článcích se dá najít mnoho zkušeností uživatelů s každým z nich, ale není nad osobní zkušenost, proto jsem se všechny frameworky z prvních 3 míst rozhodl vyzkoušet.

Všechny mezi sebou mají mnohé rozdíly, které jsou shrnuty ve velmi stručné podobě níže:

- **Spring MVC** - Přednosti tohoto frameworku jsou v silné modularitě rozšiřující čitelnost kódu a flexibilní využívaní Dependency Injection. Jedná se o velice mocný a oblíbený nástroj, který je ovšem hůře pochopitelný pro nováčky, kteří se Springem nikdy nepracovali.

[8] HTML - HyperText Markup Language
[9] CSS - Cascading Style Sheets (Kaskádové styly)
[10] C# - objektově orientovaný programovací jazyk vyvinutý firmou Microsoft
[11] Java EE - Java Enterprise Edition
[12] Java SE - Java Standard Edition
[13] JSF - JavaServer Faces



Obrázek 3.1: Graf znázorňující popularitu jednotlivých frameworků pro jazyk Java na přelomu let 2015 - 2016[8]

- **Java Server Faces** - Hlavní výhodou JSF je skutečnost, že je přímo součástí specifikace Java EE a proto je vhodný pro uživatele využívající IDE[14]. Nevýhodou je, že se jedná o velice rozsáhlý technický koncept, který je zároveň těžko uchopitelný bez značných předchozích zkušeností s javovým programováním pro web.

- **Vaadin** - Mezi výhody patří především to, že se programuje pouze serverová strana celé aplikace. Framework obsahuje velmi bohatou podporu pro různé grafické prvky, layouty, listenery a mnoho dalšího. Podporuje také mnoho kvalitních a dobře zdokumentovaných pluginů, ať už placených, či neplacených. K nevýhodám patří, že aplikace neumí defaultně pracovat v offline módu, nicméně tuto podporu je možné dodělat.

Tvorba pokusných aplikací s těmito frameworky byla velmi obohacující. U JSF ani několik dnů nezajistilo výraznější posun v jeho pochopení. Následovalo tedy otestování frameworku Spring, který je nejpoužívanější ze všech. Zde byla situace již optimističtější, nicméně množství složitostí také poměrně

[14]IDE - Integrated Development Environment = vývojové prostředí)

odrazovalo. Poslední na řadě byl Vaadin, který velmi příjemně překvapil jednoduchostí, se kterou se dají vytvořit celkem mocné výsledky. Je také vcelku dobře zdokumentovaný, a hlavně několikrát do měsíce vychází jeho aktualizace přinášející stále nové funkce. Pro účely aplikace v tomto projektu byl tedy vyhodnocen jako nejlepší, mimo jiné i z důvodu podpory pro různé součásti, které bude náš systém vyžadovat.

3.2 Návrh rozvržení vývoje do modulů

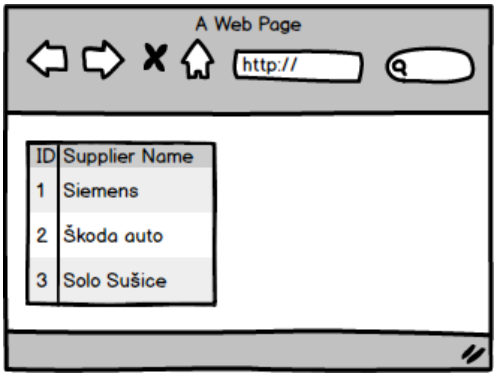
Několik prvních modulů pouze přenáší funkcionalitu z původního systému do nového, proto bude stačit pouze jejich stručný popis. Pokud modul přidává funkcionalitu novou, bude popsán detailněji.

3.2.1 Modul 1: Pohled na dodavatele

Současný stav: Neexistující systém.

Nový Use Case: Systém zobrazí uživateli na úvodní stránce tabulku s dodavateli. Jedná se o pouhé zobrazení jednoduché tabulky z databáze.

Popis podporovaných uživatelských akcí: Uživateli se tabulka zobrazí automaticky načtením úvodní stránky aplikace.



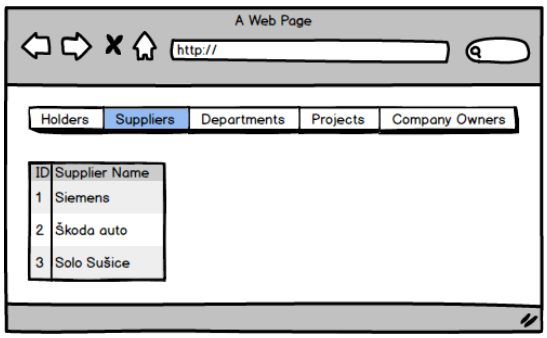
Obrázek 3.2: Návrh uživatelského rozhraní pro modul 1 (zobrazení jednoduché tabulky)

3.2.2 Modul 2: Menu a pohled na více jednoduchých tabulek

Současný stav: Pohled na 1 jednoduchou tabulku.

Nový Use Case: Aplikace obsahuje několik pohledů na různé samostatné tabulky. Mezi jednotlivými pohledy se přepíná na horizontálním menu.

Popis podporovaných uživatelských akcí: Uživatel může změnit zobrazený pohled na jednu konkrétní databázovou tabulku.



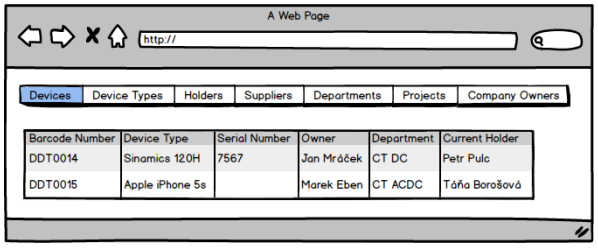
Obrázek 3.3: Návrh uživatelského rozhraní pro modul 2 (menu s více pohledy)

3.2.3 Modul 3: Složitější tabulky, které je třeba joinovat

Současný stav: Pohled na několik jednoduchých tabulek.

Nový Use Case: Aplikace umí zobrazit pohled vytvořený spojením více databázových tabulek.

Popis podporovaných uživatelských akcí: Uživatel může změnit zobrazený pohled na konkrétní databázovou tabulku nebo výsledek spojení několika tabulek dohromady.



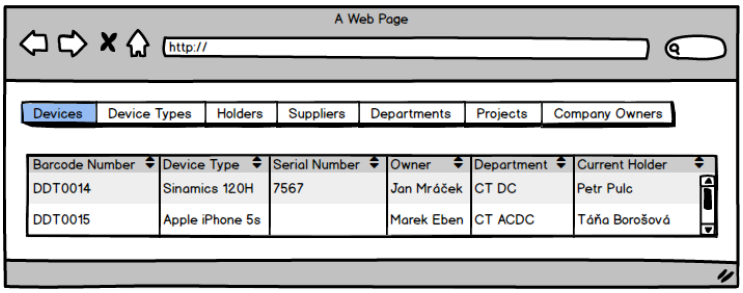
Obrázek 3.4: Návrh uživatelského rozhraní pro modul 3 (pohled na spojované tabulky)

3.2.4 Modul 4: Řazení záznamů podle jejich hodnoty

Současný stav: Pohled vytvořený spojením více tabulek.

Nový Use Case: Aplikace umí řadit záznamy od nejmenšího po největší a od největšího po nejmenší podle hodnot v jednotlivých sloupcích.

Popis podporovaných uživatelských akcí: Uživatel může řadit záznamy podle hodnot v jednotlivých sloupcích. Prvním kliknutím na záhlaví sloupce se záznamy seřadí vzestupně, při druhém kliknutí sestupně.



Obrázek 3.5: Návrh uživatelského rozhraní pro modul 4 (řazení záznamů)

3.2.5 Modul 5: Vytváření, úprava a mazání záznamů

Současný stav: Pohled na data, která je možné vzestupně, nebo sestupně řadit.

Nový Use Case: Aplikace umí vytvářet nové záznamy a editovat a odstraňovat ty stávající.

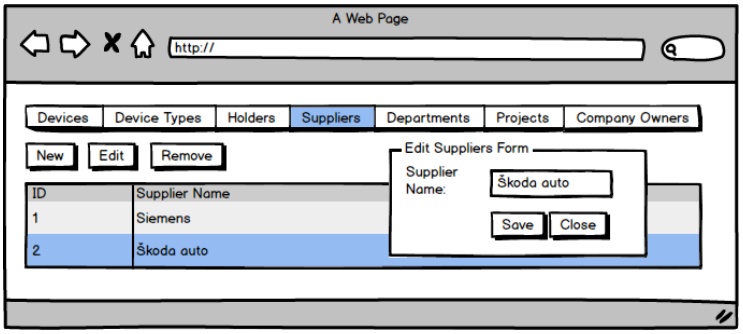
Popis podporovaných uživatelských akcí: Uživatel může stisknutím tlačítka „New“ vytvořit novou položku, stisknutím tlačítka „Edit“ upravit vybraný záznam nebo stisknout tlačítka „Remove“ vybraný záznam odstranit. Pro vytváření a úpravu se zobrazí připravený formulář. Pro smazání je potřeba potvrdit OK / Cancel dialog.

Uživatelské rozhraní: Návrh vzhledu znázorňuje Obrázek 3.6.

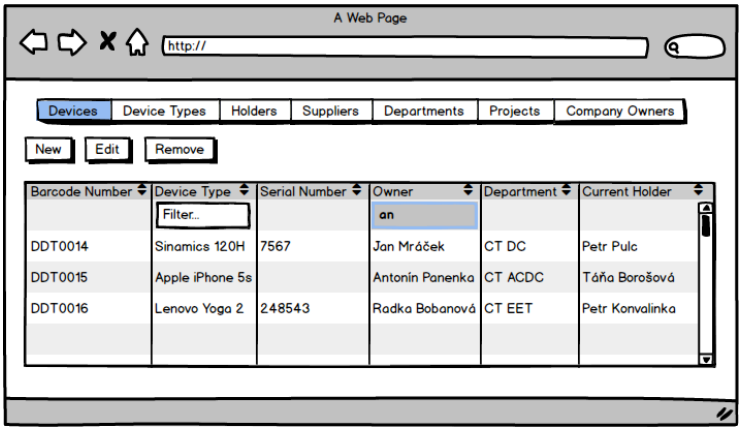
3.2.6 Modul 6: Filtrování záznamů

Současný stav: Zobrazená data lze pouze vzestupně či sestupně řadit, nelze v nich vyhledávat.

Nový Use Case: Aplikace umí vyfiltrovat záznamy obsahující daný podřetězec pro konkrétní sloupec. Je možné kombinovat i více podřetězců přes více sloupců. Pro vyfiltrované záznamy je dále možné používat řazení definované ve vlastním modulu v kapitole 3.2.5.



Obrázek 3.6: Návrh uživatelského rozhraní pro modul 5 (vytváření, úprava a mazání záznamů)



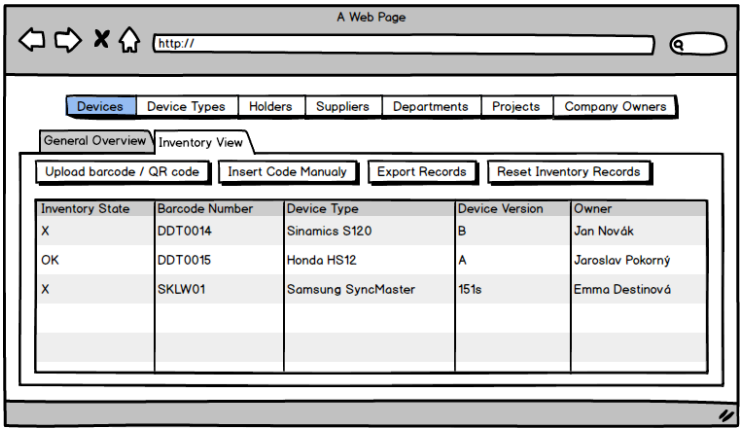
Obrázek 3.7: Návrh uživatelského rozhraní pro modul 6 (filtrování záznamů)

Popis podporovaných uživatelských akcí: Uživatel může pod hlavičkou každého definovaného sloupce zadat filtrační kritérium. Podle toho se vyfiltrují jen ty řádky, které v příslušném sloupci dané kritérium splňují. Kritéria je možné mezi sebou i kombinovat. Při jejich odstranění se zpátky zobrazí opět všechny záznamy.

3.2.7 Modul 7: Zařízení (Devices) - Pohled na inventury

Současný stav: Existuje pouze obecný pohled na zařízení (General Overview). Neexistuje možnost výběru z více možných pohledů.

Nový Use Case: Aplikace umí zobrazit více pohledů na zařízení. Mezi jednotlivými pohledy se vybírá v podokně nabídky Devices kliknutím na příslušný název karty reprezentující požadovaný pohled. Aplikace získá nový pohled Inventory View (pohled na inventury). Zde je v prvním sloupci označeno,



Obrázek 3.8: Návrh uživatelského rozhraní pro modul 7 (pohled na inventury)

zda-li již byla daná položka zainventarizována. To lze udělat buďto načtením kódu kamerou zařízení nebo ručním zadáním. Záznamy je možné vyexportovat. Po skončení inventury je možné záznamy zresetovat a připravit je tak na další inventarizační iteraci.

Popis podporovaných uživatelských akcí:

- Stisknutím tlačítka „Upload barcode / QR code“ získá uživatel možnost nahrát obrázek nového QR kódu. Ten může buďto vložit z filesystému nebo ho získat z obrazového zařízení (většinou webkamera u počítačů, případně fotoaparát v případě mobilních zařízení). Systém z něj získá jeho hodnotu a příslušný řádek označí jako inventarizovaný.

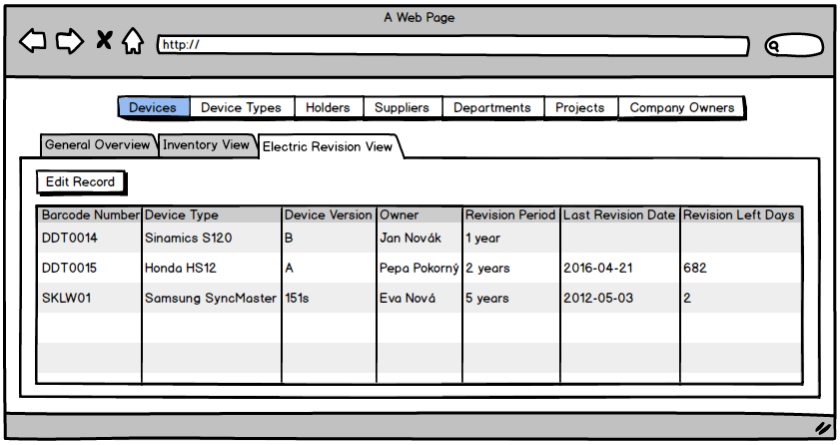
- Stisknutím tlačítka „Insert Code Manually“ se zobrazí formulář s možností zadat daný kód ručně.

- Stisknutím tlačítka „Export Records“ lze spustit export aktuálních dat z tohoto pohledu do souboru *.csv.

- Stisknutím tlačítka „Reset Inventory Records“ zresetovat všechny záznamy - to znamená nastavit v prvním sloupci všem zařízením atribut říkající, že nebyly inventarizovány. Po stisknutí tlačítka je ještě potřeba potvrdit, případně odmítnout potvrzovací dialog.

3.2.8 Modul 8: Zařízení (Devices) - Pohled na elektrorevize

Současný stav: Existuje pouze obecný pohled na zařízení (General Overview) a pohled Inventury zařízení (Inventrory View).



Obrázek 3.9: Návrh uživatelského rozhraní pro modul 8 (pohled na elektrorevize)

Nový Use Case: Aplikace umí zobrazit další pohled na zařízení, totiž pohled na elektrorevize. U jednotlivých záznamů lze nastavit datum poslední revize a revizní periodu. V posledním sloupci aplikace z těchto dvou údajů vypočítá počet dnů zbývající do další elektrorevize.

Popis podporovaných uživatelských akcí:

- Stisknutím tlačítka „Edit record“ je možné editovat revizní periodu a datum poslední elektrorevize u zvoleného zařízení. Výchozí hodnota není definována.

3.2.9 Modul 9: Podpora autentizace

Současný stav: Aplikace je přístupná všem uživatelům bez ohledu na jejich přihlašovací údaje a oprávnění.

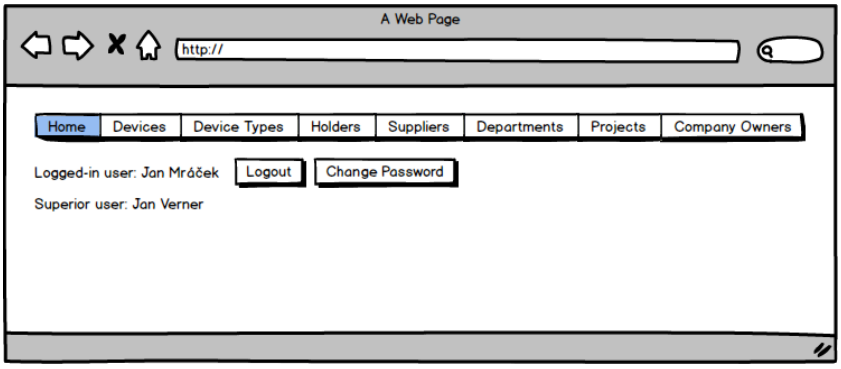
Nový Use Case: Aplikace je nově uživatelům dostupná pouze po přihlášení.

Popis podporovaných uživatelských akcí:

- Zadání uživatelského jména a hesla a přihlášení kliknutím na příslušné tlačítko.

3.2.10 Modul 10: Podpora importu uživatelů a definice jejich oprávnění

Současný stav: Nelze přidávat nové uživatele, ani měnit jejich oprávnění.



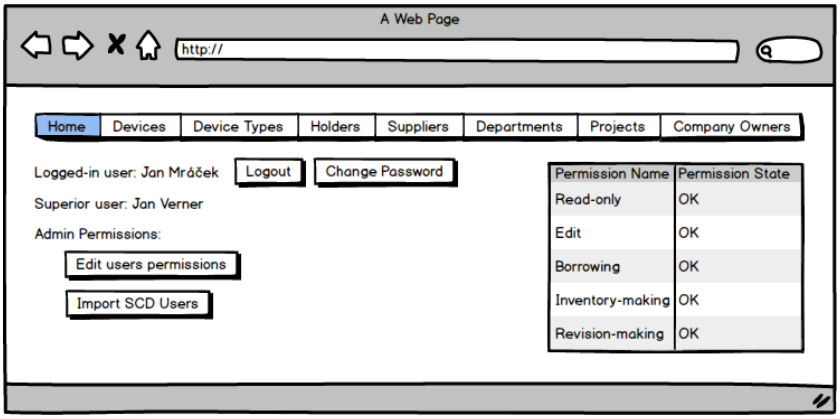
Obrázek 3.10: Návrh uživatelského rozhraní pro modul 9 (podpora autentizace)

Nový Use Case: Nově je možné importovat uživatele spolu se základními atributy ze Siemens Corporate Directory. Import přijímá *.csv soubor a informuje uživatele notifikací o počtu nově přidaných uživatelů. Pokud již uživatel se stejným identifikátorem v systému je, tak se v importovaném souboru ignoruje. Všem uživatelům je nově také možné nastavit práva pro tyto oblasti:

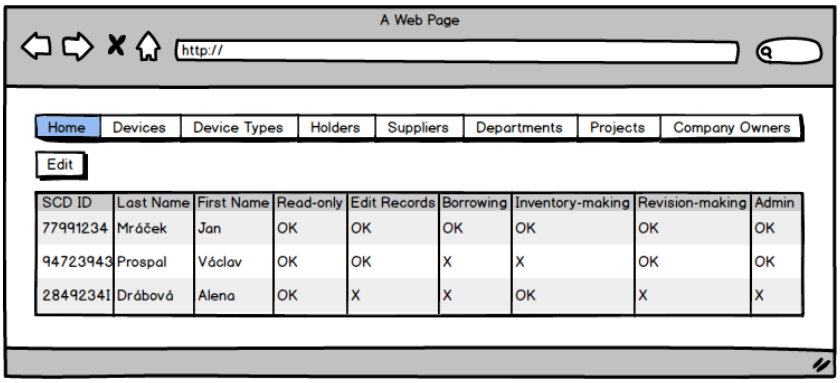
- Read-only (právo na prohlížení existujících záznamů)
- Edit (možnost upravovat existující záznamy)
- Borrowing (právo na půjčování zařízení)
- Inventory-making (oprávnění pro změny záznamů inventury)
- Revision-making (oprávnění pro změny záznamů revizí)
- Admin (oprávnění pro nastavování práv ostatním uživatelům)

Popis podporovaných uživatelských akcí:

- Pro uživatele s oprávněním admin (viz předchozí seznam) stisknutím tlačítka „Edit users' permissions“ přejít do pohledu pro nastavování oprávnění všech uživatelů, jak je znázorněno na obrázku 3.12. Zde je možné vybrat jednoho konkrétního uživatele a tomu přidat nebo odebrat některá z oprávnění.
- Pro uživatele s oprávněním admin (viz předchozí seznam) stisknutím tlačítka „Import SCD Users“ umožnit vybrat z filesystému *.cvs soubor pro import předem vyexportovaných uživatelských záznamů z SCD.
- Podle nastavených oprávnění bránit ve využívání funkcí definovaných v předchozích modulech.



Obrázek 3.11: Návrh uživatelského rozhraní pro modul 9 (podpora importu uživatelů a nastavování jejich oprávnění)



Obrázek 3.12: Vzhled pohledu pro nastavování práv všem uživatelům

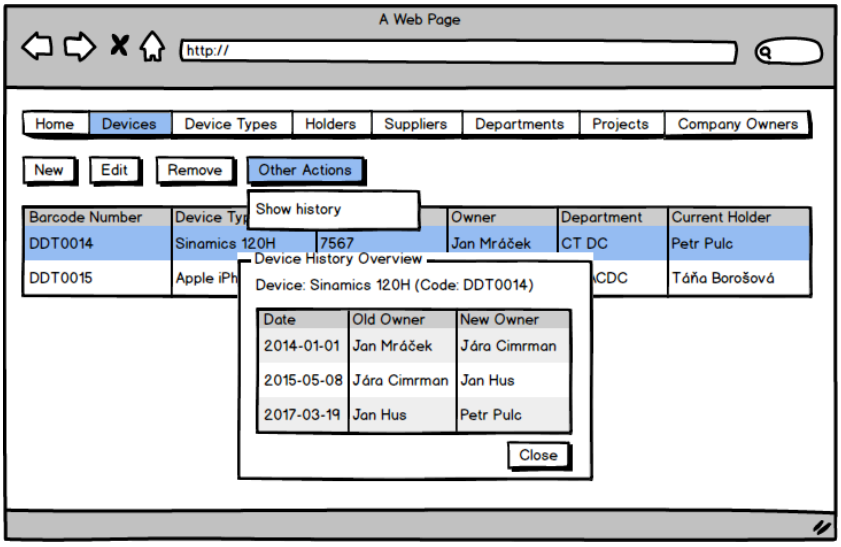
3.3 Návrh budoucího pokračování vývoje

Celá předchozí kapitola 3.2 se zabývá návrhem těch nejdůležítějších modulů, které budou v rámci této práce také implementovány. V rámci analýzy bylo ovšem zjištěno značné množství požadavků přesahující rozsah této práce. Ty byly v kapitole 2.2.1 rozděleny do dvou skupin podle toho, o jak prioritní požadavky se jedná. Mnoho z nich již bylo vyřešeno návrhem v rámci předchozí kapitoly. Návrh ostatních modulů s vysokou prioritou bude představen zde.

Protože se jedná spíše o výhled do budoucna, nebude zde zacházeno tolik do podrobností. Hlavním cílem je především nastínit, jak by se na celém projektu dalo efektivně pokračovat a jak bez dlouhého hloubání systém začít rozšiřovat. I v této části budeme pokračovat v dosavadním využívání intuitivního rozdělení jednotlivých kroků do modulů.

3.3.1 Modul 11: Zobrazení historie výpůjček

K nedostatkům aktuálně vyvíjené aplikace patří absence podpory zobrazování historie výpůjček. Jedná se o jedinou funkcionalitu původního systému Xataface, která nedostala tak vysokou prioritu, aby byla implementována v hlavní části této práce. Přeci jenom tolik neovlivňuje práci se samotnou aplikací, ale spíše poskytuje určitou přidanou hodnotu nad rámec základních požadavků. Jedná se o zajímavý údaj, ale jeho absence v prvních fázích projektu určitě nebude nějakým výrazným způsobem systém blokovat. Jak by mohla nakonec aplikace s tímto modulem vypadat ukazuje obrázek 3.13.

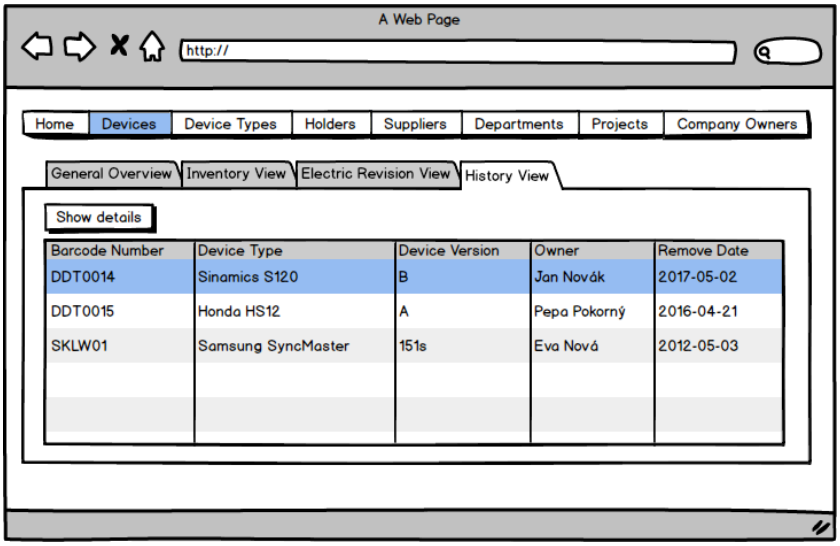


Obrázek 3.13: Pohled na aplikaci při zobrazení historie výpůjček (modul 11)

3.3.2 Modul 12: Vytvoření pohledu na odstraněná zařízení

Podle jednoho z požadavků by se měla uchovávat historie všech zařízení pro udržení komplexní historie v systému. Aby bylo možné následující požadavek naplnit, bude nutné mírně pozměnit logiku aplikace. Již nyní obsahuje každé zařízení informaci o jeho stavu. Na výběr je jedna z následujících možností:

- FUNKČNÍ
- ZAPŮJČEN
- VRÁCEN
- ROZBIT
- VYŘAZEN



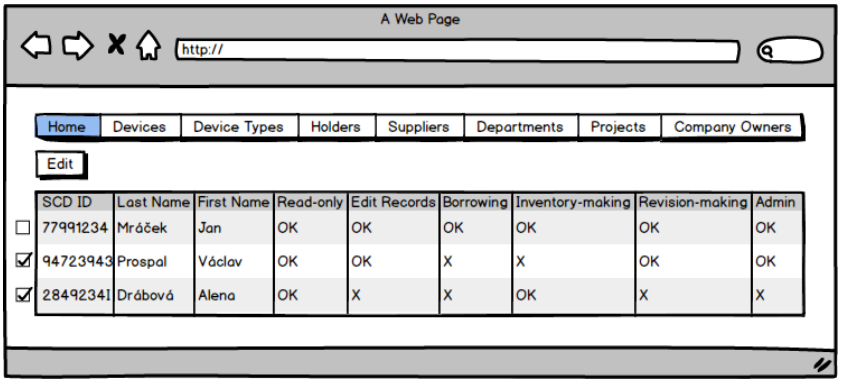
Obrázek 3.14: Vyobrazuje aplikaci při pohledu na historii odstraněných zařízení (modul 12)

Poslední z těchto stavů by šel využít jako indikátor odstranění daného zařízení. To jinými slovy znamená, že by se tento stav automaticky nastavil po stisku tlačítka „Remove“. Následně by se na všechny pohledy na zařízení nastavil filtr, který by zobrazoval jen ty, které by byly v jiném stavu, než v tomto. Dále je zapotřebí připravit nový pohled, který by naopak zobrazoval jen zařízení s tímto stavem a případně časové razítko (datum), k jakému datu k přepnutí do tohoto stavu došlo.

Vzhledem k dalším budoucím požadavkům, aby zařízení obsahovala přílohy, by tento přístup umožnil pěkné uchovávání dalších materiálů k nahlédnutí i po jeho vymazání. Tato funkcionalita je znázorněna na obrázku 3.14. Její zapracování se může mírně změnit ještě v závislosti na požadovaných právech pro toto zobrazení. Pokud by bylo přístupné pro každého, jedná se o jinou situaci, než pokud by bylo přístupné pouze pro administrátory celého systému.

3.3.3 Modul 13: Hromadné operace

V tomto modulu jde především o usnadnění hromadných změn. Velmi využívaný bude například administrátory systému pro hromadné nastavení práv nebo například pro nastavení provedené revizi více zařízením najednou, což ušetří značné množství času. Pro podporu tohoto požadavku bude přidán do všech pohledů nultý sloupec, který bude pro každý řádek obsahovat pouze neoznačený checkbox. Jeho výběrem se daný řádek připojí do výběru více řádků a po kliknutí na nějaké tlačítko se bude daná akce provádět pro všechny položky najednou.



Obrázek 3.15: Zde je zachycena aplikace s více vybranými řádky připravenými pro vykonání hromadné operace (modul 13)

Na obrázku 3.15 je například znázorněno hromadné nastavení práv pro 2 uživatele zároveň. Těch samozřejmě ale může být libovolné množství. Pokud se při editaci jedné položky vyplnil formulář podle aktuálního nastavení atributů u daného řádku, bude v případě hromadných změn formulář vyplněn výchozími hodnotami nebo bude prázdný, pokud žádné takové hodnoty nejsou definovány. Po uložení se zadané atributy aplikují na všechny položky najednou.

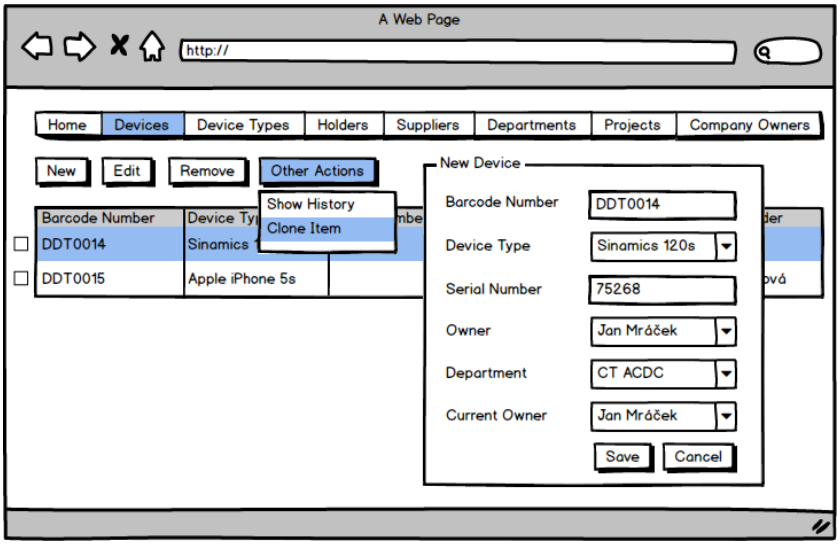
3.3.4 Modul 14: Klonování položek

Dost často se stává, že je potřeba do systému vložit položku velmi podobnou nějaké jiné, která již v systému existuje. Než ztrácet čas opisováním téměř všech hodnot, je lepší navrhnout funkcionalitu, která právě na takovéto případy bude myslet. Smyslem tohoto modulu je přidat do systému akci, která si zapamatuje právě označenou položku (řádek), otevře formulářové okno pro vkládání nového záznamu a ten předvyplní hodnotami z již existující označené položky. Ta bude samozřejmě muset mít minimálně unikátní identifikátor.

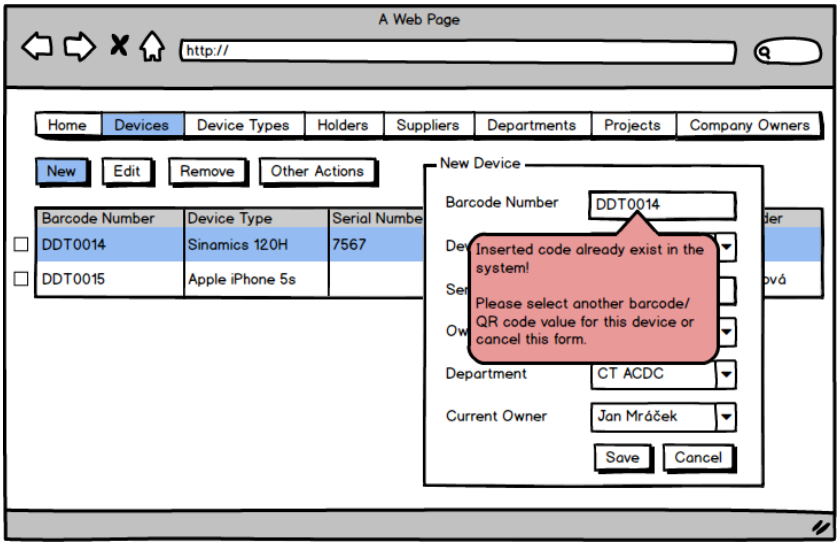
Celá situace je zachycena na ilustraci 3.16. V případě klonování existujícího zařízení je nezbytné minimálně změnit hodnotu čárového kódu, protože ten musí být pro každý záznam unikátní.

3.3.5 Modul 15: Ověření dostupnosti QR kódu při jeho přidělování či změně

I poslední modul řešící požadavek s vysokou prioritu přidává novou funkčnost, ale tentokrát spíše nepřímo. Kontroly na možnost přidělení nového QR kódu musí probíhat především interně. Bude potřeba prohledat existující záznamy, zda-li již zadávaný kód neobsahují, aby nezačalo docházet k nekonzistencím. Pokud k takovému případu dojde, měla by se zobrazit notifikace informující



Obrázek 3.16: Ilustrace zachycující návrh klonování existujícího zařízení s automaticky předvyplněným formulářem (modul 14).



Obrázek 3.17: Notifikace zobrazená kvůli zadání hodnoty čárového kódu, která již v systému existuje (modul 15).

uživatele o příčině problému a navrhnout řešení tak, jak je představeno na obrázku 3.17.

3.4 Návrh databáze

Jedním z omezení, které vyplývá přímo z analýzy, je požadavek na dočasné souběžné fungování nového a starého systému, potažmo využití jedné databáze pro oba systémy. V původním projektu byla využita databáze MySQL představená v kapitolách 2.1.1.2 a 2.1.2.1. Ta na základě požadavků musí zůstat zachována. Stejně tak musí být zachována i struktura současných tabulek a jedinou povolenou změnou je vkládání tabulek nových.

Toto omezení je sice potřeba respektovat, ale je podstatné zmínit, že se nejedná o zrovna optimální a efektivní řešení. Místo toho, aby se vždy z tabulky vyčetl požadovaný řádek s údaji, bude nutné pro čtení každé položky nejprve udělat join (spojení) více tabulek, aby následně bylo možné požadované hodnoty přenést do systému. To není úplně časově levná operace, která vždy o něco zvedne reakční dobu systému. Nejlepším řešením by bylo navrhnout novou databázi zcela na míru potřebám nové aplikace, ale v tomto projektu zvolíme z důvodu zachování kompatibility požadovaný, i když méně efektivní přístup. Naštěstí se nejedná o žádný velký a komplexní problém, takže by popisované nevýhody neměly nikterak výrazně ovlivňovat práci s novou aplikací.

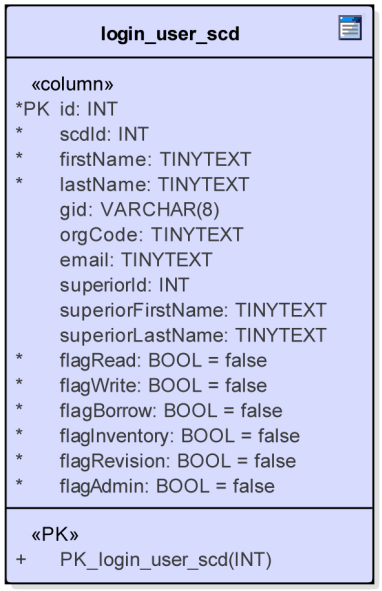
3.4.1 Nová tabulka pro přihlašování uživatelů

Jeden z požadavků na nový systém vzešlý z předchozí analýzy přímo zmiňuje nutnost napojit novou aplikaci na SCD. To ovšem znemožňuje využít současného způsobu autentizace vůči systému, kde se každý uživatel přihlašuje nějakým uživatelským jménem vytvořeným bez pevných pravidel a je tedy nutné vytvořit novou databázovou tabulku. Ta bude zároveň řešit i rozsáhlejší možnosti v nastavení práv a propojení každého uživatele na svého vedoucího pracovníka.

Návrh nové databázové tabulky pro přihlašování uživatelů je znázorněn na obrázku 3.18. Každý uživatel má v rámci SCD svoje vlastní ID, ale protože není úplně dobře znám způsob přidělování jednotlivých ID uživatelům, ale hlavně protože je třeba připravit systém na případnou situaci, kdy by měli se systémem krátkodobě pracovat i lidé bez vlastního Siemens Corporate Directory ID, bylo nad rámec toho vytvořeno ještě jedno umělé ID, aby o jedinečnosti tohoto identifikátoru nemohl být žádný spor.

Dále ze spousty navrhnutých atributů nelze nezmínit atributy *superiorId*, *superiorFirstName* a také *superiorLastName*. Jedná se o SCD ID a jméno a příjmení nadřízeného. Kvůli těmto údajům nesplňuje tato tabulka podmínky proto, aby mohla být alespoň ve *2. normální formě*[15]. Atributy se jménem

[15]2NF (2. normální forma) - doporučení pro návrh datové struktury databáze



Obrázek 3.18: Struktura nové databázové tabulky *login_user_scd*

a příjmením nadřízeného jsou totiž závislé na ID nadřízeného, nikoliv na ID konkrétního zaměstnance. Tento postup byl nutný z důvodu, že se v průběhu psaní této práce nepodařilo získat přímý přístup k SCD a bylo tak navrhnuto přenášet dočasně údaje exportem a následným importem podmnožiny uživatelů do této aplikace. Z té se ale ztratí některé údaje a může tak dojít k případům, kde je známa vazba na nadřízeného, ale kvůli nekompletním datům nebude možné zjistit jeho další údaje.

Po budoucím úspěšném napojení bude možné sloupce *superiorFirstName* a *superiorLastName* opět odstranit a tím tuto databázovou tabulku dostat do *3. normální formy*[16]. Teoreticky bude možné smazat i samotný sloupec *superiorId* a data získávat přímo z SCD, ale to už záleží na technickém provedení onoho napojení.

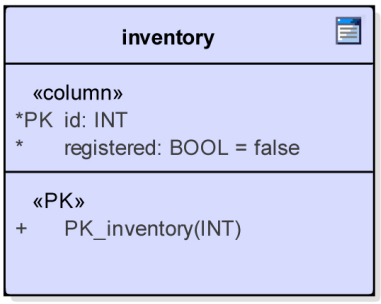
Na závěr je také potřeba zmínit, že tato databázová tabulka nemá v současné době žádné další vazby na jiné tabulky. V budoucích verzích je možné vyčlenit jména a příjmení do zvláštní tabulky organizované podle ID, jakožto klíče.

3.4.2 Nová tabulka pro inventury

Druhá nová tabulka, která doplní současnou databázi, ponese jméno *inventory* a je znázorněna na obrázku 3.19. Jedná se o velice jednoduchou tabulku, která by za normálních okolností ani nevznikla a místo ní by stačilo přidat jeden

[16] 3NF (3. normální forma) - doporučení pro návrh datové struktury databáze

nový sloupec do rodičovské tabulky. Jak již ovšem bylo zmíněno v předchozím textu, úpravy starých tabulek nejsou možné.

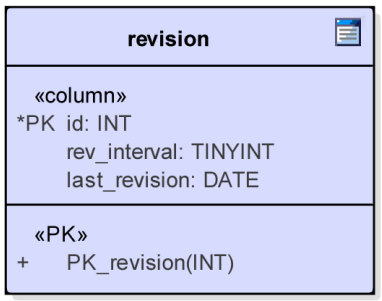


Obrázek 3.19: Struktura nové databázové tabulky *inventory*

Tato tabulka obsahuje sloupeček s primárním klíčem *id*, který je stejný, jako id v databázové tabulce *objects*. Přes tyto atributy jsou následně tabulky také spojovány. Druhý sloupec obsahuje už jenom booleovskou hodnotu říkající, jestli položka s daným ID již prošla inventurou. Její výchozí hodnota je vždy **„false“** a po zanesení daného zažízení nebo dané věci do systému změní hodnotu na **„true“**. Po skončení inventury mohou být opět všechny záznamy vynulovány, což představuje nastavení všech položek na hodnotu **„false“**. Pokud by v budoucnu vyvstal požadavek na evidenci historie inventur, dala by se tato tabulka rozšířit ještě o sloupeček s časovým razítkem.

3.4.3 Nová tabulka pro revize elektrozařízení

Tato tabulka bude velice podobná té z předchozí kapitoly 3.4.2. Úplně stejným způsobem je řešeno ID jako její primární klíč, který mají všechny řádky stejný, jako v tabulce *objects*. Za normálních okolností by ani nevznikla, ale byla by nahrazena 2 dodatečnými sloupečky v tabulce se všemi zařízeními. Stejně jako v předchozím případě lze přidat sloupeček s časovým razítkem pro případnou evidenci historie.



Obrázek 3.20: Struktura nové databázové tabulky *db_inventory*

Její vzhled je patrný z obrázku 3.20. Kromě již zmíněného ID identifikátoru dále obsahuje sloupce rev_interval a last_revision. První z nich slouží pro uložení revizivního intervalu, který je podle české legistlativy 1 - 5 roků[9]. Hodnota nula představuje, že daný atribut není definovaný a aplikace ho bude ignorovat. V posledním sloupci je uvedeno datum poslední revize. Z těchto 2 sloupců bude možné následně v aplikaci dopočítávat počet zbývajících dnů do propadnutí revize a případně posílat notifikace na zainteresované emailové adresy. To by ale ještě vyžadovalo rozšíření databáze tak, aby u každého zařízení bylo možné nastavit, komu se mají posílat notifikace, v jakém časovém předstihu a s jakou intenzitou.

3.5 Shrnutí návrhové části

V této kapitole byla nejprve v úvodní části popsána kritéria pro výběr vhodného frameworku. Po shrnutí bodů, které je dobré při výběru sledovat, byl jako nejvhodnější zvolen framework Vaadin. Následovala návrhová část celého systému. Ta byla zpracována pro iterační programování po jednotlivých modulech od úplného základu až po specifické funkce. Návrh se pro potřeby této práce omezil pouze na ty požadavky, které byly ohodnoceny vysokou prioritou. Byla popsána funkcionalita všech systémů a byl přiložen návrh možného vzhledu uživatelského rozhraní po jejich implementaci. Závěr této kapitoly patřil návrhům nutných úprav databáze, jakožto uložiště pro data z aplikace.

Kapitola 4

Realizace

Realizace této aplikace sledovala moduly navržené v kapitole 3.2, a to jak jejich obsahem, tak jejich pořadím. Cílem této kapitoly není vysvětlovat každý napsaný řádek kódu, ale především vypíchnout jak zajímavosti, které bylo v průběhu práce potřeba řešit, tak představit použitý framework z pohledu implementace. Zmínit můžeme kupříkladu způsob, kterým se implementovaly nejběžnější prvky vyvíjené aplikace, ale také kde čekala jaká úskalí a jak byla překonána.

4.1 Příprava před zahájením implementace

Následující kroky bylo nutné vykonat ještě před zahájením samotné implementace:

1. **Instalace JVM**[17]
   Každý program napsaný v jazyce Java vyžaduje ke svému běhu JVM. Po zkompilování vytvořeného kódu vzniká mezikód, který je v Java světě nazýván jako „Java bytecode“. Ten slouží jako vstup do JVM, který ho transformuje do strojového jazyka a následně ho vykoná.[10]

2. **Instalace IDE**[18]
   Pro vývoj větších projektů je záhodno použít některé z vývojových prostředí. Pro jazyk Java se nejčastěji používá buďto NetBeans IDE, nebo Eclipse IDE. Zde jsme zvolili druhou variantu, protože pro ni existuje několik rozšiřujících pluginů pro tvorbu webových aplikací založených na zvoleném frameworku Vaadin.

3. **Instalace databáze**
   Z analýzy vyplývá potřeba využívat již existující databázi MySQL. V té je uložena většina business dat, se kterými aplikace pracuje.

[17] JVM - Java Virtual Machine
[18] IDE - Integrated Development Environment (vývojové prostředí)

4. **Instalace webového serveru**
   Protože pro implementaci byl zvolen jazyk Java, respektive jeho verze pro webové aplikace Java Enterprise Edition, bylo potřeba zvolit takový webový aplikační server, který bude umět spouštět aplikace napsané v tomto jazyce. Volba zde padla na **Apache TomEE**, jehož použití je doporučeno přímo komunitou uživatelů, která s Vaadinem dennodenně pracuje.[11]

5. **Instalace systému na správu databáze**
   Pro správu databází a databázových tabulek nešla využít původní aplikace PhpMyAdmin, protože aplikační server TomEE PHP projekty spouštět neumí. Proto byla zvolena aplikace DBeaver, která je volně k dispozici a nabízí vše potřebné k základní správě databáze.

4.2 Zprovoznění frameworku Vaadin

4.2.1 Apache Maven

Nejjednodušší pro zprovoznění frameworku Vaadin je využít nástroje Apache Maven. Jedná se o nástroj pro správu a automatizaci buildů aplikací, kde hlavním podporovaným jazykem je Java. V jeho konfiguračním souboru je pro každý projekt možné nadefinovat jeho závislosti na externích knihovnách. Ty jsou vyhledávány v uložištích, která jsou také definována v konfiguračním souboru.

Z uložišť je možné využít globální uložiště nástroje Maven, se kterým je možné si ve většině případů vystačit. V případě potřeby je možné nadefinovat více uložišť, ať už firemních, nebo soukromých. Pro zprovoznění aplikace tedy bude potřeba těchto knihoven:

- com.vaadin.vaadin-server (jádro frameworku)
- com.vaadin.vaadin-push (stále spojení webové aplikace se serverem[12])
- com.vaadin.vaadin-client-compiled (základní vestavěné widgety)
- com.vaadin.vaadin-themes (podpora předdefinovaných stylů)
- javax.servlet.javax.servlet-api

4.2.2 Hibernate

Pro usnadnění práce s databází z aplikace slouží framework Hibernate. Ten umožňuje takzvané objektově-relační mapování (ORM). Jedná se o jednu z mnoha implementací specifikace JPA[19]. Hibernate je nápomocen v úloze

[19] JPA - Java Persistence API

zachovávat data perzistentní. Termín „perzistentní“ je možné chápat jako podmínku pro dlouhodobé zachování dat, například při vypnutí systému při přerušení elektrické energie. Tento framework také umožňuje anotacemi řídit souběžný přístup k datům, nicméně tato funkcionalita nebyla vyhodnocena pro prototyp jako stěžejní, a proto není dále rozpracována.

Jeho úloha je v transformování dat z databáze na objekty a opačně. Do databáze se pak dotazuje jazykem Hibernate Query Language (HQL), který je velmi podobný jako klasický databázový dotazovací jazyk Structured Query Language (SQL). Následně jsou získaná data přetrasformovaná na objekty, které jsou posléze vráceny aplikaci k dalšímu zpracování.

Aby bylo možné framework Hibernate využívat v aplikaci, je nutné přidat jeho knihovnu do projektu. K tomu se dá opět využít systém Apache Maven, který byl představen v kapitole 4.2.1. Tam se vloží jednak knihovna *org.hibernate.hibernate-core* jako hlavní část frameworku, ke které se připojí knihovna *mysql.mysql-connector-java* obsahující logiku nutnou pro připojení k využité databázi MySQL.

4.2.3 Sestavení komponent

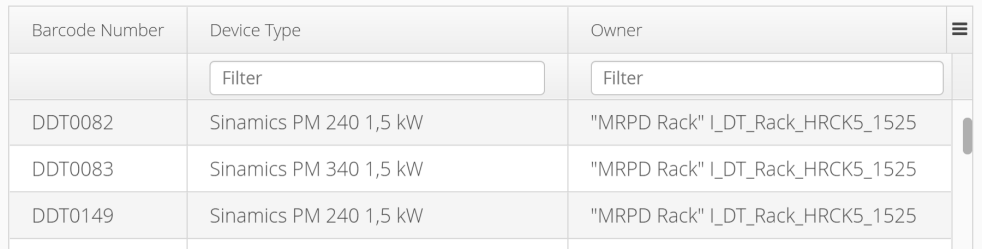
Před začátkem vývoje je tedy nainstalované vývojové prostředí Eclipse. V něm se vytvoří propojení na nainstalovaný webový aplikační server, ve kterém bude běžet výsledná aplikace. V dalším kroku se vytvoří nový projekt typu Maven, kde se v konfiguraci nastaví všechny potřebné knihovny popsané v předešlých podkapitolách. Na závěr je potřeba vytvořit také konfigurační soubor k frameworku Hibernate. Jeho zkrácené nastavení je vidět na ukázce níže:

```
<hibernate-configuration>
  <session-factory>

    <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost/siemens</property>
    <property name="connection.username">dbUserName</property>
    <property name="connection.password">dbPassword</property>
    <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.characterEncoding">utf8</property>

    <mapping class="cz.siemens.inventory.model.Device" />
    <mapping class="cz.siemens.inventory.model.User" />

  </session-factory>
</hibernate-configuration>
```

4.3 Vaadin - hlavní prvky

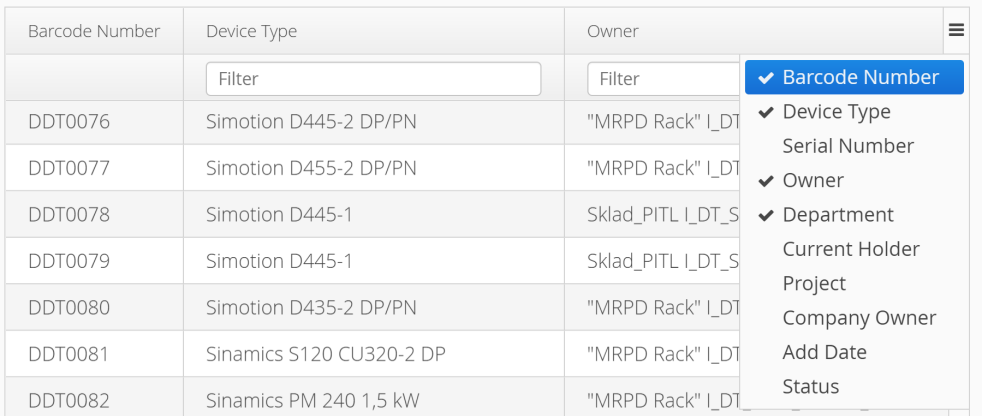
4.3.1 Vaadin: Grid (tabulka)

Tato aplikace má sloužit především na zobrazování a organizaci dat z databáze. Hlavní prvek zde tedy představuje komponenta zastupující tabulku - ty se totiž nachází v nějaké formě na každé stránce (respektive pohledu). Takový prvek framework nabízí a nese jméno Vaadin Grid. Jeho vzhled je znázorněn na obrázku 4.1.



Obrázek 4.1: Prvek tabulky Vaadin Grid

Mezi jeho výhody patří již vestavěná podpora řazení ve všech sloupcích a možnost jednoduše doplňovat komponentám vlastní funkcionalitu. Tak je například implementováno filtrování. Poté, co uživatel do textového pole pod hlavičkou sloupce vypíše nějakou hodnotu, automaticky se zobrazí jen ty záznamy, které obsahují v daném sloupci řetězec zadaný uživatelem. Tato filtrační kritéria lze kombinovat napříč různými sloupci a vykonávat tak velmi komplexní dotazy. Dalo by se dokonce říci, že tento způsob filtrování záznamů je téměř úplně schopen nahradit i nejrůznější vyhledávací funkce.

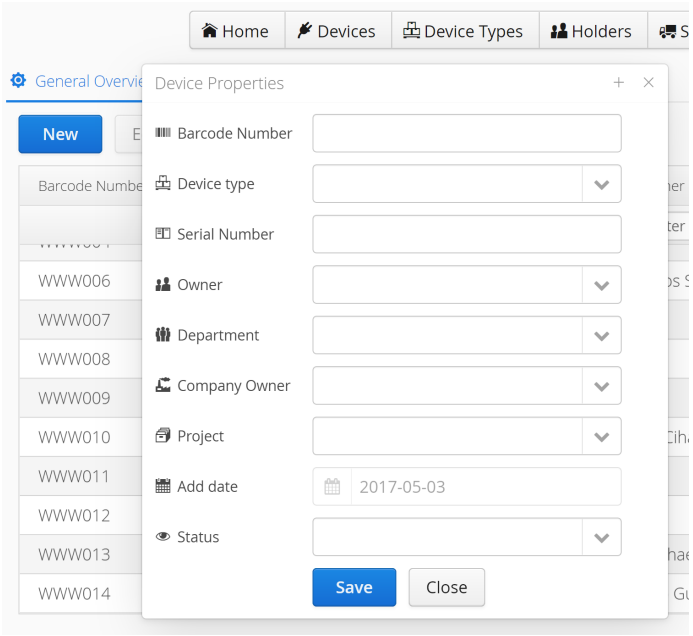


Obrázek 4.2: Ukázka výběru sloupců, které se následně zobrazí v tabulce zobrazené uživateli.

Další již vestavěnou funkcí je podpora zobrazování jen některých sloupců. K výběru slouží ikonka tří tlustých čárek nad sebou úplně vpravo nahoře. Po kliknutí na ní je možné vybrat sloupce, které uživatele zajímají. Tento přístup je vhodný z toho důvodu, že uživatelé nejsou přehlceni informacemi, které pro ně nejsou relevantní. To také zvyšuje přehlednost celého systému. Obrázek 4.2 demonstruje výběr několika sloupců pro zobrazení z jejich široké nabídky. Jak nabídku, tak to, které sloupce budou ve výchozím stavu zobrazeny a které ne, lze samozřejmě definovat přímo v aplikaci.

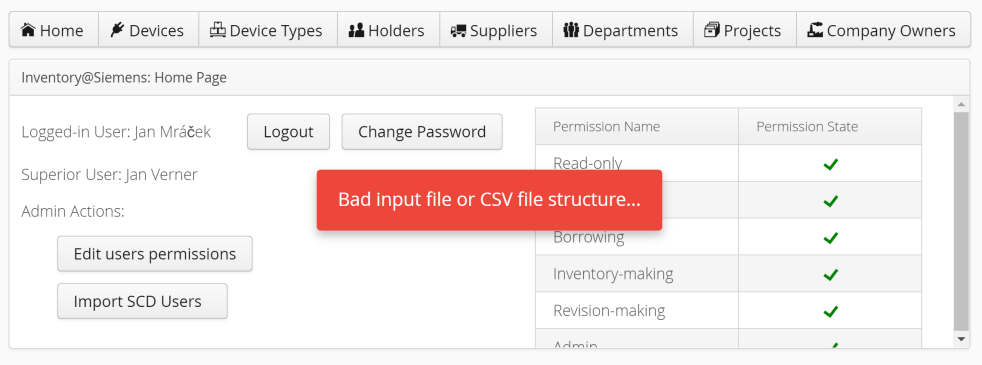
4.3.2 Vaadin: Formulář

Hned po zobrazení dat z databáze bude také potřeba tato data nějakým způsobem modifikovat. K tomu slouží Vaadin Form a klasické prvky, které lze na formulářích najít (tlačítko, zaškrtávací tlačítko, textové pole, popisek atd.). Klasický vzhled formulářového okna je představen na obrázku 4.3.



Obrázek 4.3: Ukázka formuláře vytvořeného ve frameworku Vaadin.

Výhoda tohoto formuláře je hlavně v jednoduchém způsobu, jak namapovat objekt získaný pomocí frameworku Hibernate z databáze na jeho jednotlivé prvky. Celé kouzlo spočívá v tom, že namapování probíhá nastavením dvou funkcí k danému prvku. První funkce čte hodnotu daného atributu z příslušného objektu (tzv. *getter*) a druhá funkce nastavuje novou hodnotu atributu příslušného objektu po stisknutí tlačítka „save“ (tzv. *setter*). Zároveň je možné jednotlivé prvky doplnit o vhodné ikonky z kolekce Vaadin Icons, což zvyšuje



Obrázek 4.4: Ukázka notifikace typu *Error message*

přehlednost v jednotlivých prvcích a mírně přidává aplikaci na vizuální přitažlivosti.

4.3.3 Vaadin: Notifikace

Podpora notifikací ze strany Vaadinu je velmi přímočará. Notifikace jsou přitom velmi užitečným nástrojem, jak předat uživateli okamžitou zpětnou vazbu o nějaké události. Způsob, jak v aplikaci zobrazit notifikaci je pouhý jeden řádek kódu. O zbytek se už není potřeba starat, vše je součást frameworku. Na obrázku 4.4 je vykreslena jedna z možných notifikací. Zdrojový kód, který takovou notifikaci vytvoří, může vypadat například následovně:

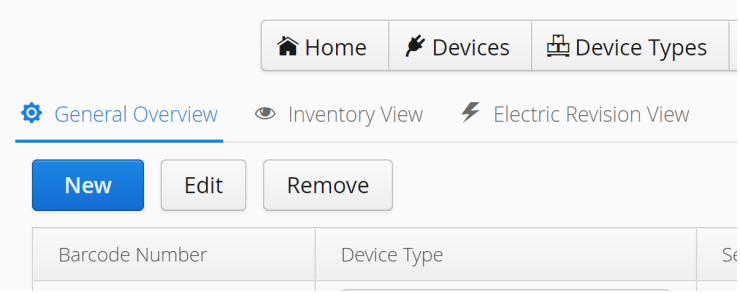
```
Notification.show("Bad input file or CSV file structure", Type.ERROR_MESSAGE);
```

První parametr této funkce určuje text, který bude v notifikaci zobrazen, druhý parametr určuje typ notifikace. Ten určuje nejen styl notifikace, ale také jak dlouho bude daná notifikace zobrazena.

4.3.4 Vaadin: Další prvky

Vaadin samozřejmě obsahuje desítky různých prvků. Některé z nich jsou zcela jednoduché a vesměs i známé (například tlačítko) a nemá cenu je jednotlivě popisovat. Stejně tak nemá cenu popisovat ani množství různých prvků, které do této aplikace nebyly použity. Proto pouze souhrnně uveďme ty využité, které jsou ještě něčím zajímavé.

Vaadin Layouts Za zmínku stojí určitě způsob rozmisťování prvků. Ten základní tvoří horizontální a vertikální layout. Prvky přidávané do horizontálního layoutu jsou skládány vedle sebe, ty přidávané do vertikálního layoutu pod sebe. Zároveň je možné tyto layouty vnořovat navzájem do sebe do libovolné hloubky. V obou je také možné nastavovat zarovnání prvků, okraje



Obrázek 4.5: Ukázka vzhledu navigačních karet (Vaadin Tabs)

kolem jednotlivých elementů, barvy a mnoho dalšího. Vaadin zde čerpá z předdefinovaného *CSS souboru*[20], který lze dále libovolně upravovat, pokud si nevystačíme s výchozími možnostmi.

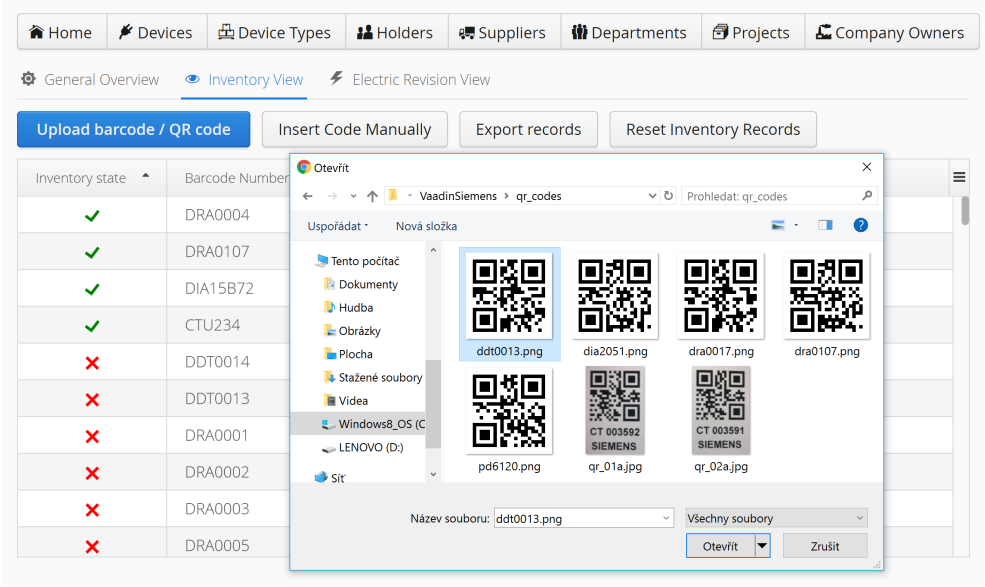
Pokud si při nějakém pohledu nevystačíme s tímto rozvržením, je další možností využít takzvaný *Grid Layout*. Ten je obdobou imaginární tabulky. Určí se u něj počet řádků a sloupců a do následně vzniklých pomyslných buněk lze vkládat další prvky nabízené frameworkem, včetně již zmíněných horizontálních a vertikálních layoutů.

Vaadin Tabs Posledním důležitým využitým předdefinovaným elementem toho frameworku jsou Vaadin Tabs, tedy obdoba karet. Ty umožňují vytvářet podtypy informací zařazené pod svým rodičem. V tomto systému lze například nalézt rozlišení několika pohledů na zařízení. Konkrétně se jedná o obecný pohled *General Overview*, dále pak o *Inventory View* a také o *Electric Revision View*. Kliknutím na danou kartu se zobrazí pohled týkající se pouze části vyjádřené názvem příslušné karty tak, jak to známe z jiných, ať už webových či desktopových aplikací. Znázornění toho prvku patrné na obrázku 4.5

4.4 Implementace nových funkcí

Tato část se zaměří na průběh implementace nových funkcí. Jedná se především o modul z návrhu popisující inventury v kapitole 3.2.7 a modul z návrhu popisující elektrorevize v kapitole 3.2.8. Pak také půjde o následné moduly popisující jednak autentizace uživatelů a také jejich import ze Siemens Corporate Directory a nastavování jejich práv, což je funkcionalita popsaná v kapitolách 3.2.9 a 3.2.10.

[20]CSS - Cascading Style Sheets (kaskádové styly)



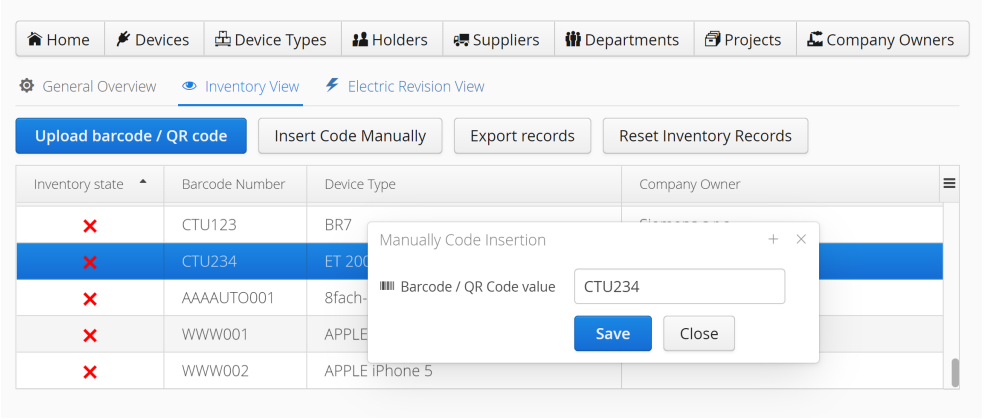
Obrázek 4.6: Inventura: Ukázka načítání QR kódu z filesystému počítače

4.4.1 Implementace pohledu na inventury

Inventury tvoří klíčovou část právě vznikající aplikace. Z chybějících funkcí ve starém systému byla největší poptávka pravděpodobně právě po této. Důvod tkví v tom, že právě tato funkce může značně ulehčit práci všem vlastníkům zařízení při každoročně konaných inventurách. Lze totiž využít již existující QR kódy, případně čárové kódy a inventury částečně zautomatizovat. Na obrázku 4.6 je ilustrováno načítání čárového kódu.

Většinou se bude jednat o načítání z mobilního telefonu, ale není problém například udělat více fotek kódů a ty teprve později takto jednotlivě vkládat do systému přímo z paměťové karty, případně souborového systému. Při načítání z mobilního telefonu se po stisknutí tlačítka *Upload barcode / QR code* otevře nejprve fotogalerie, ale z té lze hned vybrat možnost pořízení nového obrázku přední či zadní kamerou. Tato pořízená fotografie se odešle na server, který z ní vyčte uloženou hodnotu. Pokud se tento proces podaří, zobrazí se uživateli notifikaci o úspěšném zaznamenání nového kódu. V opačném případě bude uživatel taktéž notifikací informován o tom, že se načítání kódu nepodařilo a bude také připojen důvod, proč tomu tak bylo.

Rozpoznávání jak čárových, tak QR kódů je možné díky knihovnám *com.google.zxing.core* a *com.google.zxing.javase*. Jejich import zajišťuje stejně jako u ostatních knihoven Apache Maven, do jehož konfigurace se odkaz na tyto dvě knihovny vložil. Tato knihovna načte obrázek, vyrobí z něj binární bitovou mapu a z té se zároveň spolu s nastavením citlivosti na chyby pokusí vyčíst obsaženou hodnotu.

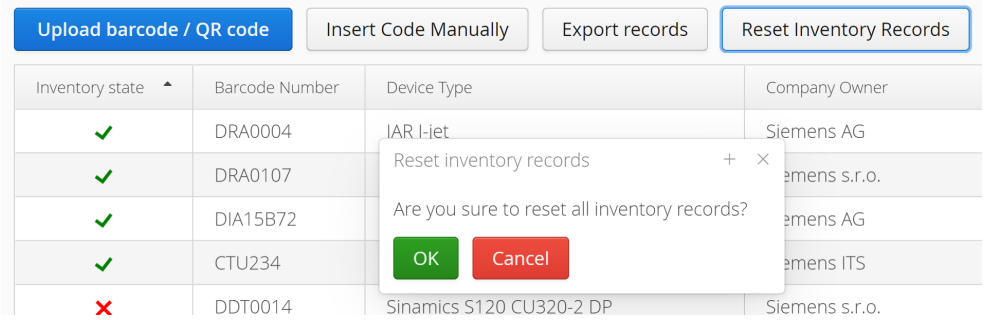


Obrázek 4.7: Inventura: Ukázka manuálního načítání QR kódu

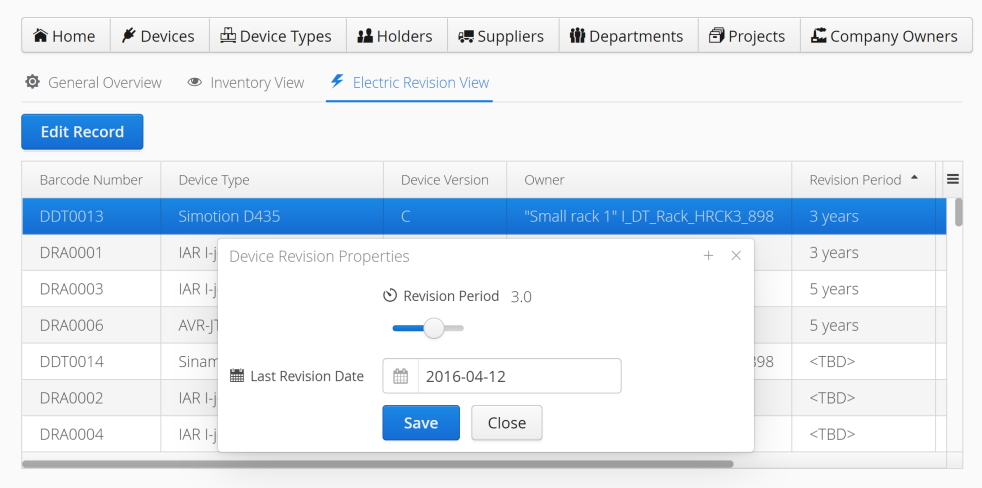
Pro případy, kdy bude kód například nečitelný, nebo ho nebude z jiných důvodů možné načíst nějakým optickým zařízením, existuje zde možnost hodnotu kódu zadat ručně do připraveného formuláře, který se spustí stisknutím tlačítka *Insert Code Manually*, jak je ukázáno na obrázku 4.7.

Po skončení inventury bude možné všechny záznamy nechat vyexportovat, nicméně tato funkcionalita zatím není implementována. Již nyní je ale pro ni připraveno tlačítko *Export records*, která v současné verzi pouze zobrazí notifikaci s informací, že daná funkcionalita zatím není implementována.

Poslední možná akce se skrývá pod tlačítkem *Reset Inventory Records*. Ta slouží k odstranění výsledků inventury tak, že všechna zařízení označí jako zatím neinventarizovaná. Tím může začít další kolo inventur. Ještě před definitivním resetováním záznamů musí uživatel potvrdit dialog znázorněný na obrázku 4.8.



Obrázek 4.8: Inventura: Ukázka resetu všech záznamů



Obrázek 4.9: Elektrorevize: Ukázka formuláře pro nastavení nových parametrů k jednomu konkrétnímu zařízení.

4.4.2 Implementace pohledu na elektrorevize

Mezi klíčové funkcionality patří také část věnující se elektrorevizím. Důvod je stejný, jako u předchozí části věnující se inventurám. Nejedná se jen o nějaké drobné vylepšení, ale o vytvoření cenné přidané hodnoty. Podle plánu vývoje již tato verze obsahuje podporu, ovšem jen částečnou. Systém bude umět evidovat jak revizní periody, tak datum poslední revize. Výrazného vylepšení se může dosáhnout napojením systému na email a odesíláním automatických notifikací o končící platnosti aktuální revize. Nicméně to patří již mimo rozsah této práce.

V základu je využit podobný pohled na zařízení, jako u obecného pohledu na zařízení. Jsou zde ale tři nové sloupce:

1. Revision Period - určuje jak často je potřeba eletrorevize dělat
2. Last Revision Date - datum poslední vykonané inventury
3. Revision Left Days - počet dnů do vypršení aktuální revize

Celkový pohled na elektrorevize v této aplikaci je znázorněn na obrázku 4.9. Za zmínku také stojí, že například pro výpočet zbývajících dnů bylo využito zjednodušení díky vylepšené podpoře pro aritmetiku s časovými údaji v nové verzi programovacího jazyka Java, konkrétně nejnovější verzi Java Standard Edition 8, viz ukázka níže:

```
long daysBetween =
Duration.between(currentDate.atStartOfDay(), newDate.atStartOfDay()).toDays();
```

4.4.3 Implementace propojení se Siemens Corporate Directory

Z analýzy vyplynula nutnost nahradit současnou správu uživatelů více sofistikovaným přístupem a minimálně propojit tuto aplikaci se Siemens Corporate Directory, kde jsou již všechny možné informace o uživatelích dostupné. Jak toho nejlépe docílit je popsáno v kapitolách 3.2.9 a 3.2.10 věnujících se návrhu.

Z důvodů technických obtíží spojených se získáním přístupu do SCD bylo nakonec rozhodnuto o náhradním řešení. Pro získání uživatelů se využívá importovaného souboru, který tato aplikace načte a zpracuje. Využito je formátu CSV[21], který předpokládá vstupní přesně definovanou strukturu. Aby bylo možné tento soubor úspěšně naimportovat a zpracovat, musí obsahovat přesně deset sloupců. Konkrétně se jedná o:

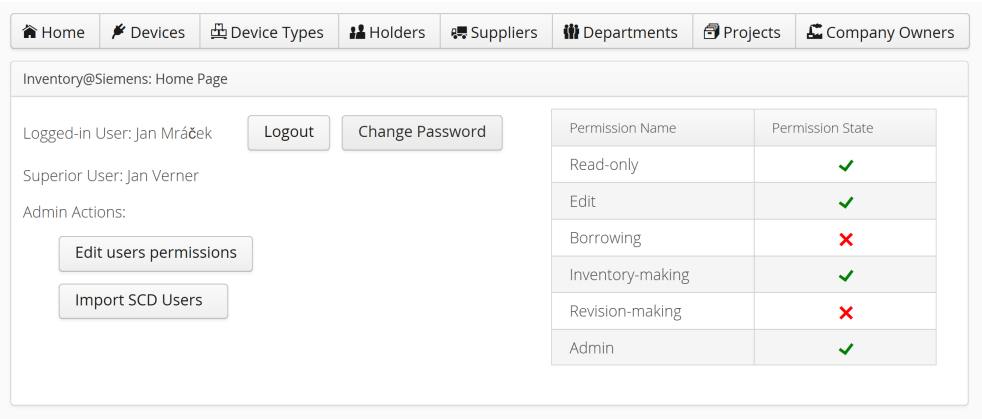
1. SCD ID
2. Jméno
3. Příjmení
4. GID
5. Lokace
6. Oddělení
7. Email
8. SCD ID nadřízeného
9. Jméno nadřízeného
10. Příjmení nadřízeného

Po úspěšném importu a ověření jeho struktury je z každého řádku vyčteno SCD ID jako jednoznačný identifikátor uživatelů a porovnán se záznamy v databázi. Pokud daný uživatel zatím v databázi chybí, je vytvořen podle údajů v dalších sloupcích. Situace, kdy se nějaké údaje uživatele změní (například dostane nového nadřízeného), nebo je uživatel z SCD úplně odstraněn, zatím není ošetřena. V dalších verzích této aplikace, které již prototypem nebudou, bude třeba na tuto situaci pamatovat a implementaci dokončit. Více informací o nutných dodělávkách je shrnuto v kapitole 4.5. Uživatelům se v každém případě zobrazí notifikace informující o tom, jak import dopadl. Při úspěšném importu obsahují informace o tom, kolik nových uživatelů bylo do aplikace importováno, v případě chyby čtení souboru budou uživatelé taktéž informováni.

[21] CSV - Comma-Separated Values

4.4.4 Implementace řízení přístupových práv

Po přihlášení se zobrazí úvodní stránka znázorněna na obrázku 4.10. Zde je na pravé části stránky tabulka s přehledem všech práv právě přihlášeného uživatele. Tato práva se uloží do aktuální relace, aby nebyl uživatel nucen vyplňovat přihlašovací údaje při každé změně pohledu do jiné části aplikace.



Obrázek 4.10: Nastavování práv: úvodní stránka

Také zde lze nalézt tlačítko pro odhlášení, které oprávnění z aktuální relace odmaže a uživatele přesune na přihlašovací stránku. Příklad práce s relacemi ilustruje následující část zdrojového kódu:

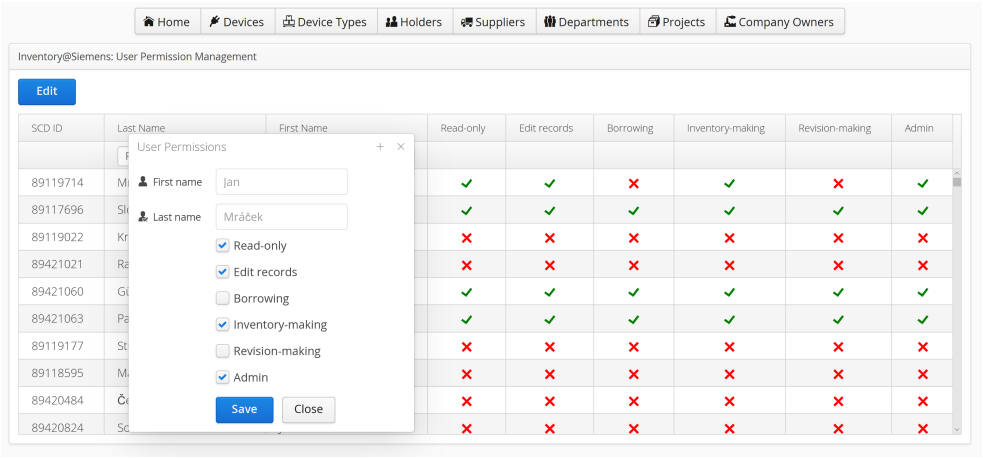
```
VaadinSession.getCurrent().setAttribute("loggedUser", MyUI.AUTH.getLoggedUser());
```

V tomto případě se do VaadinSession, frameworkové implementace pro podporu relací, uloží objekt přihlášeného uživatele, aby byly z dalších pohledů dostupné i ostatní údaje, například emailová adresa, na kterou by mohly být zasílány emailové notifikace.

Pokud někdo disponuje administrátorskými právy, má možnost kliknutím na tlačítko *Edit Users' Permissions* zobrazit pohled pro editaci práv všech uživatelů. Na tom je možné zvolit požadovaného uživatele, případně si ho nejprve vyfiltrovat a stisknutím tlačítka Edit otevřít formulář, kde je možné nastavit jednotlivá oprávnění. Implementace tohoto nastavování je stejná, jako v případě všech ostatních prvků popsaných v předchozích kapitolách. Situace právě probíhajícího nastavování parametrů je znázorněna na obrázku 4.11.

4.5 Otevřené problémy

V závěru implementační části by bylo ještě potřeba shrnout, jaké problémy zůstaly otevřené. Tím se nemyslí, jaké další funkce by šly do systému dále přidávat - tomu se ostatně věnuje již kapitola 3.3, ve které jsou popsány další



Obrázek 4.11: Nastavování práv: detail nastavování

navrhnuté moduly, které nebyly implementovány. Jde o nedostatky v již implementovaných modulech, které byly zanedbány, protože nebyly nikterak podstatné pro cíl této práce - vytvoření prototypu - ale v plně funkčním systému by se určitě vyskytovat neměly. Následující výčet lze tedy brát spíše jako návod pro budoucí plynulé navázání na vývoj této aplikace.

- Zprovoznit možnost změnit heslo kliknutím na tlačítko *Change password* na úvodní stránce, které v současné verzi zobrazí notifikaci o chybějící podpoře pro tuto akci.
- Doimplementovat předpřipravenou funkcionalitu pro export záznamů z inventury před jejich resetováním.
- Pokud se nepodaří rovnou napojit aplikaci přímo na SCD a bude se dále jednou za čas využívat pouze importu dat, je potřeba upravit způsob provádění aktualizací. Současná verze umí pouze přidávat uživatele nové. Ddodělat je potřeba aktualizace údajů u stávajících a taky mazání již neexistujících uživatelů.
- Upravit aplikaci tak, aby při aktualizaci stránky stisknutím klávesy F5, nebo využitím ekvivalentní funkce prohlížeče, nebyla ztracena relace zapříčiňující nutnost nového přihlášení uživatelů

Kapitola 5

Testování

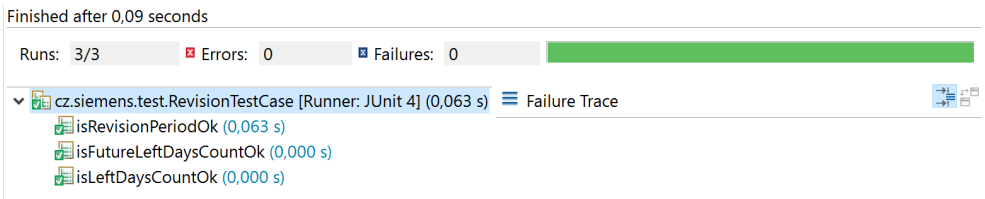
5.1 Testování v průběhu vývoje

Již v průběhu implementace existovala silná potřeba provádět jednoduché testy ověřující základní funkcionalitu. Čím později se totiž na případné chyby přijde, tím složitější a časově náročnější bývá jejich odstraňování. Vzhledem k tomu by bylo chybou považovat čas vynaložený na testování za ztracený, proto je vhodné nasměrovat ho k vytváření a vykonávání smysluplných testů.

Jedním z oblíbených frameworků pro psaní unit testů v Javě je JUnit. Ten využívá anotací k tomu, aby oddělil jednotlivé testy od přípravy objektů a dalších prvků jak bezprostředně před zahájením testů, tak i těsně po nich.

Tento druh testů má především dát programátorům zpětnou vazbu o jejich kódu. Většinou se jedná o automatické testy pro určitou funkčnost programu, které se dají spouštět mnohokrát po sobě. Ty by měly být pokud možno co nejjednodušejší, aby byly snadno pochopitelné a v žádném případě by se neměly navzájem ovlivňovat.[13]

Na obrázku 5.1 jsou vidět výsledky test casu, který se zabývá správným výpočtem zbývající doby do uplynutí platné revize. Jak je vidět, v tomto případě dopadly všechny testy dobře.



Obrázek 5.1: Znázornění kladného výsledku jednoho test casu vytvořeného pomocí frameworku JUnit.

Následující úryvek kódu přibližuje velmi jednoduchý příklad použití frameworku JUnit. Metoda s anotací *Before* se zavolá před vykonáním každého testu z důvodu zaručení jejich nezávislosti. Testem se zde myslí vykonání jedné metody uvozené anotací *Test*, která testuje jednu konkrétní funkci. Důležité je také mít na paměti, že se testy v rámci jednoho test casu provádějí v náhodném pořadí.

```
Device device;
@Before
public void setUp() {
  ApplianceRevision revision = new ApplianceRevision();
  revision.setInterval((byte)2);
  device = new Device();
  device.setLastRevision(revision);
}

@Test
  public void isLeftDaysCountOk() {
  String leftDays = device.getLeftDaysCount();
  assertEquals("730", leftDays);
}
```

5.2 Testování hotového prototypu

Po dokončení prototypu představeného v předchozích kapitolách byla aplikace předvedena uživatelům, kteří by ji měli v budoucnu používat. Pro ukázku byl vytvořen obraz virtuálního počítače, který byl mezi uživateli distribuován. Ten byl nakonec tak veliký, že ho nebylo možné připojit na datové médium k této diplomové práci. Případný zájemce na něm však najde vše potřebné, aby byl schopen si systém nainstalovat i na vlastním počítači. Postup pro to je popsán v příloze této diplomové práce.

Výstup z uživatelského testování lze rozdělit do dvou částí. V první z nich jsou zmíněny další úpravy aplikace - většinou se jedná o menší potíže, které proklouzly skrz síto analýzy a návrhu, takže je cílový systém nepodchycuje. V druhé části bylo nalezeno několik chyb, které bude třeba v některé z budoucích verzí opravit.

Seznam objevených nedostatků a doporučení:

- Zařízení se na rozdíl od ostatních položek nesmí odstranit smazáním, ale pouze nastavením stavu smazáno a nadále musí zůstat součástí systému.
- Mělo by být podporováno hromadné nahrávání QR kódů (v současném stavu je lze přidávat pouze po jednom).

- Uživatelé by měli při všech pohledech na zařízení vidět a mít možnost upravovat pouze vlastní, nikoliv všechna zařízení (nadřízení by mohli mít možnost vidět volitelně také zařízení jejich podřízených).
- Po vzoru *Suppliers* (dodavatelé) založit novou kategorii *Manufacturers* (výrobci) a v pohledu *Device Types* nahradit jejich editaci vyplňováním textu jejich výběrem jedné z předem definovaných položek.
- Mimo přímého nastavování práv mít také možnost definovat role, které by pak byly přidělovány uživatelům místo jednotlivých oprávnění.
- Podpora pro hromadné operace (možnost označit více položek najednou a provést s nimi jednu operaci).
- Doplnit zobrazení historie výpůjček jak na straně zařízení, tak na straně uživatelů.
- Zařízení získávat z aplikace Majetek (nesouvisí s touto prací) a pravidelně je aktualizovat; zároveň zachovat možnost přidávat další zařízení neevidovaná v aplikaci Majetek.
- Pro držitele (*Holders*) přidat flag určující, jedná-li se o importovaného, nebo ručně přidaného uživatele; při importu kontrolovat, aby nedocházelo k duplicitám.

Seznam objevených chyb (bugů):

- Po stisknutí tlačítka *Cancel* na potvrzovacím dialogu při nastavování administrátorských práv jsou tato práva neočekávaně nastavena.
- Font aplikace nepodporuje některé znaky české abecedy (ěčř).
- Při vkládání nového zařízení umožnit editaci data vložení místo automatického nastavení aktuálního data.
- Při nastavování vlastníka zařízení (*Owner*) chybí kontrola existence zadaného údaje; když nesouhlasí, vloží poslední ze seznamu.

Závěr

V této práci se podařilo vytvořit prototyp nové webové aplikace, která by měla nahradit dosavadní způsob vedení záznamů o výpůjčkách, respektive pohledu na jejich aktuální stav. Dále se také podařilo vytvořit několik nových funkcí pro usnadnění dalších činností, které se se zařízeními dají vykonávat, ať už se jedná o inventury, nebo o elektrorevize. Díky zachování staré databáze je možné po nějakou přechodnou dobu používat oba systémy, jak tento, tak původní Xataface, najednou. Na závěr byla také vylepšena podpora autentizace tak, aby bylo možné uživatelům nastavovat více práv, například těch pro nově přidané funkce inventury a revize.

Splněné zadání

Rešerše současných možností systému výpůjčky a inventarizace

Během analýzy bylo zjištěno, že požadavky kladené na tento systém jsou natolik specifické, že nebude možné využít žádné univerzální aplikace. Nejlepší schůdnou cestou se ukázalo být zvolení vhodného frameworku, jehož předpřipravené prvky dokáží následný vývoj ulehčit a urychlit. Dále pak byly prozkoumány způsoby dosavadního vedení záznamů, kde byl především popsán způsob různých interakcí mezi uživateli a systémem.

Analýza požadavků a potřeb uživatelů testovací laboratoře

Byly zanalyzovány požadavky uživatelů, vyhodnoceny podle priorit a vybrány ty, které je třeba zachovat ze současného systému, plus ty z těch nových, které mají vysokou prioritu. Dále byly uvedeny i ostatní požadavky, které ale již nebyly vyhodnoceny jako nezbytné, a tudíž jsou sice zanalyzovány, ale nejsou do detailu navrhnuty, ani implementovány v rámci prototypu. Při komunikaci s lidmi pracujícími se současnými systémy byly zjišťovány největší neduhy sou-

časného stavu, aby mohla být následně představena nová vylepšení. Nad rámec toho byly sesbírány také další požadavky formou velice podobnou brainstormingu (co by bylo pěkné, kdyby někdy v budoucno bylo...).

Výběr vhodného nástroje pro systém výpůjčky a inventarizace

Jako nejvhodnější nástroj pro systém výpůjčky a inventarizace byl vybrán framework Vaadin usnadňující tvorbu webových aplikací. Dále pak byla využita databáze MySQL, podle požadavku na její zachování. Pro usnadnění komunikace mezi Vaadinem a databází byl ještě využit framework Hibernate umožňující objektově-relační mapování. Framework Vaadin byl zvolen hlavně pro výborný poměr mezi jeho složitostí a přidanou hodnotou pro tento projekt. Byl vybrán po otestování použitelnosti spolu s dalšími jeho konkurenty a následně byly představeny jejich výhody a nevýhody. Dalším důvodem jeho volby byla jak vynikající dokumentace, tak podpora komunity uživatelů.

Návrh architektury systému

Byly navrhnuty všechny části, které vyšly z analýzy jako vysoce prioritní. Pro vývoj byla zvolena forma využívání modulů. Šlo ale spíše o vývojové iterace, než že by se v systému opravdu objevovaly skutečné moduly. V rámci každého modulu byl popsán nejprve současný stav, následovaný popisem nového Use Case a také popisem nových uživatelských akcí. Každý modul byl také opatřen wireframem, ilustrací naznačující budoucí možnou podobu systému po implementaci daného modulu. Dále byly také již navrženy některé méně prioritní moduly, které už jsou ale pouze slovně popsané, nicméně pro přehlednost také obsahují ilustrační obrázky jejich budoucího možného vzhledu. V závěru byly také představeny nutné změny v databázi, aby bylo možné navrženou funkcionalitu skutečně realizovat.

Implementace systému jako prototypu rešení

Byl implementován prototyp podle návrhu vzešlého z analýzy. Při implementaci bylo využito různých metod softwarového inženýrství a správných zásad pro programování. Implementace byla realizována v iteracích věrně kopírujících navržené moduly. Byl tedy stanoven směr, kterým by se aplikace mohla dále ubírat a byly pro to připraveny znovupoužitelné komponenty, které stačí v nových verzích pouze podědit a drobně modifikovat.

Testování prototypu a analýza zpětné vazby od uživatelů

Výsledná aplikace byla otestována uživatelským testováním. Zpětná vazba od uživatelů byla shromážděna a sepsána v kapitole věnované právě testování.

Výhled do budoucna

Při pokračování prací na tomto systému bude nejprve potřeba zohlednit jednak připomínky vzešlé z testování, které odhalují některé drobné nedostatky aplikace v současném stavu a stejně tak také implementaci věcí zanedbaných v tomto prototypu a popsaných na závěr kapitoly věnující se implementaci.

Dalším krokem by měla být snaha pochopit systém a jeho koncept, k čemuž lze využít jak komentáře ve zdrojovém kódu, tak i bohatou dokumentaci samotného frameworku. Tím, že je spousta prvků již implementována, bude jejich pochopení mnohem jednodušší, protože lze porovnat již existující kód s tím, co vykonává.

Při implementaci složitějších funkcí by také bylo dobré zdokonalit způsoby testování, protože s komplexitou aplikací roste také jejich náchylnost k různým druhům chyb.

Na úplný závěr lze snad vyjádřit naději, že směr, kterým se aplikace vydala, byl zvolen správně. Bylo by dobré, kdyby na tuto práci bylo v budoucnu jakkoliv navázáno a aplikace byla dotažena do stavu reálného nasazení.

Literatura

[1] *Xataface - představení* [online]. [cit. 2017-04-28]. Dostupné z: http://cs.softoware.net/apps/download-xataface-for-linux.html

[2] *About Xataface* [online]. [cit. 2017-04-28]. Dostupné z: http://xataface.com/wiki/about

[3] *What is MySQL* [online]. [cit. 2017-04-29]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

[4] *Honeywell Xenon1900GHD laser 2D barcode scanner supermarket express bar code reader usb interface* [online]. [cit. 2017-04-29]. Dostupné z: http://www.aliexpress.com/item-img/Honeywell-Xenon1900GHD-laser-2D-barcode-scanner-supermarket-express-bar-code-reader/32340332051.html

[5] Zapletal, L.: *Lehký úvod do LDAP* [online]. [cit. 2017-04-30]. Dostupné z: https://www.root.cz/clanky/lehky-uvod-do-ldap/

[6] Škrášek, J.: *PHP frameworky* [online]. [cit. 2017-05-01]. Dostupné z: http://programujte.com/clanek/2008022000-php-frameworky

[7] Skoumal, V.: *Jak vybrat framework?* [online]. [cit. 2017-05-01]. Dostupné z: https://www.skoumal.net/cs/jak-vybrat-framework/

[8] Skoumal, V.: *Top 4 Java Web Frameworks Revealed: Real Life Usage Data of Spring MVC, Vaadin, GWT and JSF* [online]. [cit. 2017-05-01]. Dostupné z: https://zeroturnaround.com/rebellabs/top-4-java-web-frameworks-revealed-real-life-usage-data-of-spring-mvc-vaadin-gwt-and-jsf/

[9] Horáček, F.: *UN2: Lhůty revizí a ČSN 33 1500* [online]. [cit. 2017-05-01]. Dostupné z: http://elektrika.cz/data/clanky/un2-lhuty-revizi-a-csn-33-1500

[10] *ENCYCLOPEDIA: Definition of: Java Virtual Machine* [online]. [cit. 2017-05-03]. Dostupné z: http://www.pcmag.com/encyclopedia/term/45578/java-virtual-machine

[11] *Developing Vaadin JEE6 applications on Eclipse and Apache TomEE* [online]. [cit. 2017-05-03]. Dostupné z: https://devsoap.com/developing-vaadin-jee6-applications-on-eclipse-and-apache-tomee/

[12] *Vaadin Documentation: Vaadin Push* [online]. [cit. 2017-05-03]. Dostupné z: https://vaadin.com/docs/-/part/framework/advanced/advanced-push.html

[13] Kripner, M.: *Unit testy v Javě a JUnit* [online]. [cit. 2017-05-08]. Dostupné z: https://www.itnetwork.cz/java/pokrocile/java-unit-testy-v-junit

Příloha **A**

Uživatelský manuál

A.1 Průvodce nasazením

Všechny potřebné instalační soubory jsou dostupné na CD, které je přiloženo k této diplomové práci, v tomto umístění:

`src\install\`

A.1.1 Instalace Java SE 8

- Standardně nainstalovat Javu SE 8 kliknutím na *jdk-8u121-windows-x64* z instalační složky.
- Přidat systémovou proměnnou reprezentující hodnotu cesty ke kořeni instalační složky Javy (např. C:\Program Files\Java\jdk1.8.0_131).
- Přidat cestu k podsložce *bin* do systémové proměnné *PATH* (např. %JAVA_HOME%\bin).

A.1.2 Instalace databáze MySQL

- Standardně nainstalovat MySQL kliknutím na *mysql-installer-community-5.7.18.1.msi* z instalační složky.

A.1.3 Instalace aplikačního serveru Apache TomEE

- Rozbalit balíček *apache-tomee-1.7.4-webprofile.zip* do nějakého adresáře, například *C:\App*.
- Přidat systémovou proměnnou reprezentující hodnotu cesty ke kořeni instalační složky TomEE (C:\App\apache-tomee-1.7.4-webprofile).
- Přidat cestu k podsložce *bin* do systémové proměnné *PATH* (např. %CATALINA_HOME%\bin).

- Z instalační složky vložit knihovnu Hibernate (*hibernate-jpa-2.1-api-1.0.0.Final*) přímo do aplikačního serveru do tohoto adresáře:

  `C:\App\apache-tomee-1.7.4-webprofile\lib\`

A.1.4 Instalace databázového manageru DBeaver

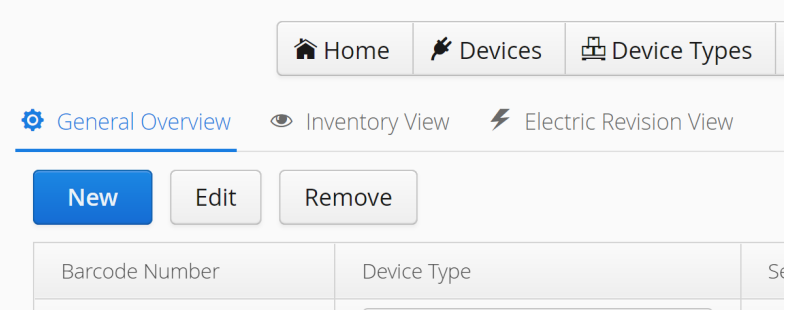
- Standardně nainstalovat MySQL kliknutím na *dbeaver-ce-4.0.5-x86_64-setup.exe*.

A.1.5 Import databázové struktury a dat

- Spustit databázový manažer DBeaver
- Database → New connection → MySQL → Next
- Database: siemens
  User Name: dbadmin
  Password: <zde zvolit nějaké heslo>
- Next → Next
- Connection name: DB Siemens
  Connection Type: Production
- Finish
- Database Navigator → DB Siemens → Databases → kliknout pravým tlačítkem myši na „siemens“ → Tools → Restore Databse
- Vybrat soubor pro import (*siemens.sql*) → Start

A.1.6 Nasazení aplikace VaadinSiemens

- Zkopírovat soubor *VaadinSiemens.war* do podsložky *webapps* kořenového adresáře aplikačního serveru (např. C:\App\apache-tomee-1.7.4-webprofile\webapps). Zde při spuštění aplikačního serveru již proběhne automatické nasazení aplikace.
- Po nasazení vznikne ve složce *webapps* adresář *VaadinSiemens*. V té zeditovat soubor WEB-INF\classes\hibernate.cfg.xml tak, že se upraví klíč *connection.password* na hodnotu nastavenou při vytváření databáze. Pokud uživatelské jméno, jméno databáze a číslo portu bylo použito v doporučení tohoto manuálu, není třeba nastavovat a mohou se nechat výchozí hodnoty.



Obrázek A.1: Aplikace - tlačítka pro změnu záznamů

A.2 Uživatelská příručka

A.2.1 Spuštění aplikace

- Pokud neběží aplikační server, je potřeba ho zapnout spuštěním tohoto souboru:

  ```
  C:\App\apache-tomee-1.7.4-webprofile\bin\startup.bat
  ```

- Aplikace se spouští na adrese `http://localhost:8080/VaadinSiemens`
- Pro příhlášení je potřeba zadat email a jako výchozí heslo slouží GID řetězec

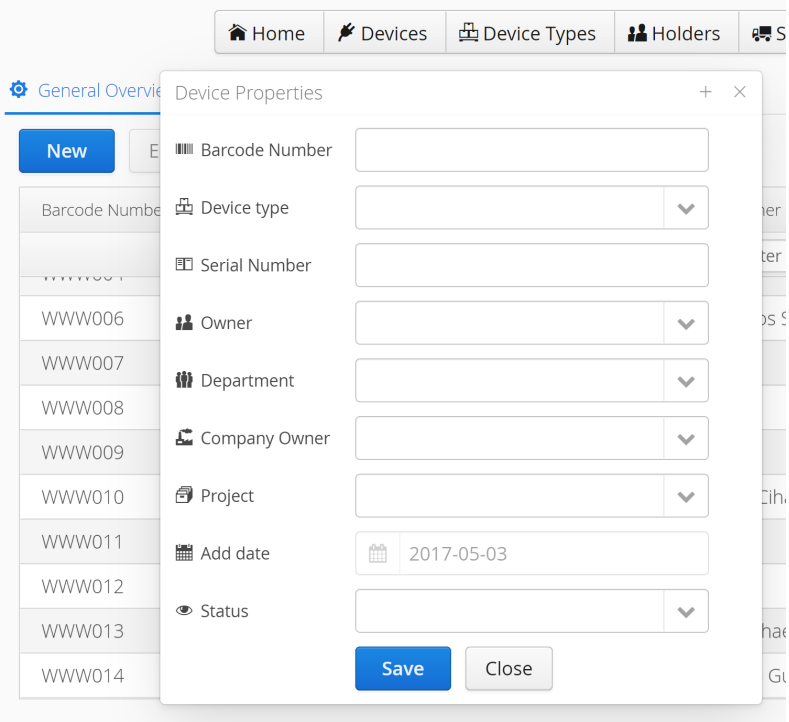
A.2.2 Změny záznamů

Pro změny záznamů je potřeba mít práva pro zápis. Tato práva umožní zobrazení tlačítek New, Edit a Remove, která slouží pro podporu základních operací, které se se záznamy dají dělat. Ukázka editace je znázorněna na obrázcích A.1 a také A.2.

Tyto úpravy se týkají téměř všech pohledů vyjma úvodní stránky a specializovaných pohledů na zařízení. Pro editaci zařízení se dá využít pouze obecný pohled *General View*. Specializované pohledy na zařízení umí editovat pouze vlastnosti týkající se daného pohledu.

A.2.3 Správa uživatelů a oprávnění

Pro přidání nových uživatelů je nutné, aby tito uživatelé byli součástí SCD a také byl k dispozici aktuální export dat, který by šel do naší aplikace importovat. To lze udělat přechodem na hlavní stránku, kliknutím na tlačítko *Import SCD Users* a výběrem příslušného souboru *.csv. Toto tlačítko ale vidí pouze uživatelé s oprávněním *Admin*.

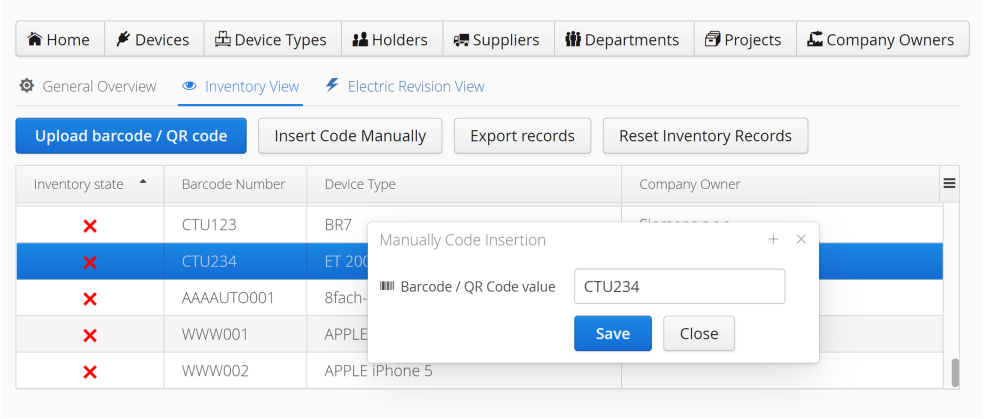


Obrázek A.2: Aplikace - formulář pro změnu záznamů

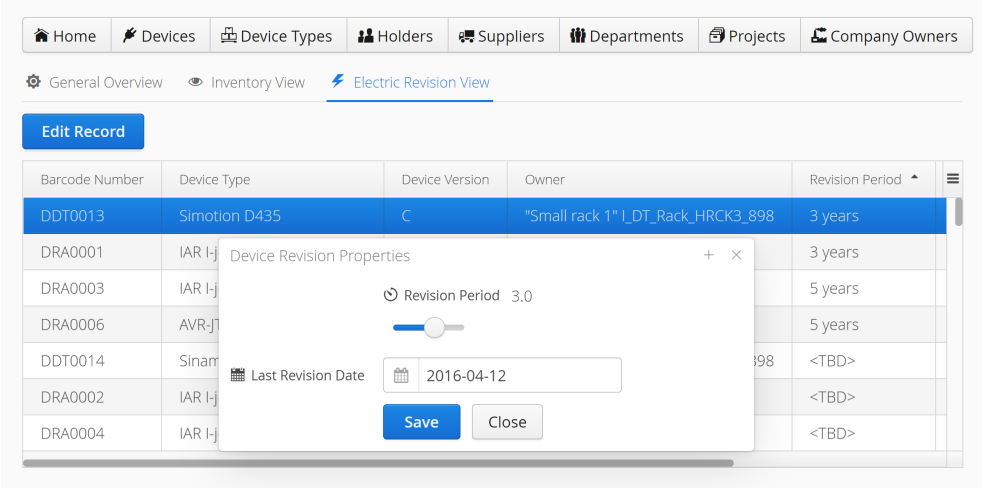
Pro následné úpravy je potřeba kliknout na tlačítko *Edit users' permissions*, kde je po označení požadovaného uživatele možné buď dvojklikem na příslušný řádek nebo označením tlačítka Edit měnit práva všem uživatelům. I toto tlačítko je viditelné pouze pro uživatele s oprávněním *Admin* jako v předchozím případě. Při nastavování administrátorského oprávnění je třeba potrvdit, zda spolu s ním cheme nastavit všechna oprávnění, a nebo pouze to administrátorské umožňující pouze pracovat s uživateli.

A.2.4 Inventarizace

Pro inventarizaci je nejprve nutné přejít do pohledu *Devices*, odkud dále do *Inventory View*. Zde je možné stisknutím tlačítka *Upload barcode / QR code* buďto nahrát QR kód z filesystému nebo ho při přístupu z mobilu rovnou vyfotit. Dále je také možné stisknutím tlačítka *Manually code insertion* zadat kód ručně, jak je znázorněno na obrázku A.3. Po správném zadání kódu bez ohledu na zvolenou metodu se zobrazí zpráva o (ne)úspěchu daného načtení. V případě, že byl kód načten správně, změní se v příslušném řádku symbol červeného křížku na symbol zeleného zátržítka. Po skončení inventury je možné stisknutím tlačítka *Reset inventory records* všechny záznamy zresetovat, to znamená nastavit, že žádná ze zařízení zatím nebyla vybrána.



Obrázek A.3: Aplikace - formulář pro ruční zadání kódu



Obrázek A.4: Aplikace - formulář pro úpravy revizních parametrů

A.2.5 Elektrorevize

Protože elektrorevize jsou také specializovaným pohledem na zařízení, je pro jejich zobrazení také nutné nejdříve přejít do pohledu *Devices* a následně se zanořit do pohledu *Electric Revision View*. Pokud má aktuálně přihlášený uživatel oprávnění pro provádění elektrorevizí, může upravovat jak revizní interval, tak především datum poslední revize buď stisknutím tlačítka *Edit*, nebo dvojklikem nad danou položkou. Každá změna jednoho z těchto údajů vyvolá přepočítání posledního sloupce ukazujícího počet dnů zbývajících do vypršení aktuálně platné revize. Tento pohled spolu s úpravou jednoho ze záznamů jsou znázorněny na obrázku A.4.

A.2.6 Výpůjčky

Pro změnu aktuálního vlastníka nějaké věci je potřeba přejít do pohledu *Devices*, označit danou položku, kliknout na tlačítko *Edit* a změnit hodnotu u vlastnosti *Owner* nebo *Current Holder*. Konkrétní hodnota se vybírá ze seznamu, který se dá upravovat v příslušném pohledu. Při psaní do políčka pro výběr položky se nabízené možnosti přefiltrují tak, aby obsahovaly zadaný podřetězec. Touto změnou je převod realizován a není potřeba dalších kroků. Pro tuto operaci je potřeba mít přidělené oprávnění pro zápis.

Příloha B

Seznam použitých zkratek

MS Microsoft

PHP Hypertext Preprocessor

SQL Structured Query Language

SCD Siemens Corporate Directory

HTML HyperText Markup Language

CSS Cascading Style Sheets

C# Objektově orientovaný programovací jazyk vyvinutý firmou Microsoft

IDE Integrated Development Environment

JVM Java Virtual Machine

ORM Object-reational mapping

HQL Hibernate Query Language

CSV Comma-Separated Values

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `install` ..... soubory potřebné k instalaci aplikace
  - `implementation` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text`
  - `DP_Mracek_Jan_2017.pdf` ..... text práce ve formátu PDF
  - `DP_Zadani.pdf` ..... zadání práce ve formátu PDF