VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

SYSTÉM PRO SPRÁVU KONTROL KVALITY ZAŘÍZENÍ

SYSTEM FOR QUALITY CONTROL MANAGEMENT OF DEVICES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. LUKÁŠ ČERNÝ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JIŘÍ HYNEK**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Černý Lukáš, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Systém pro správu kontrol kvality zařízení**
System for Quality Control Management of Devices
Kategorie: Uživatelská rozhraní
Zadání:

1. Seznamte se s principy tvorby informačních systémů s důrazem na uživatelská rozhraní, responzivní zobrazení a User experience.
2. Analyzujte existující formáty pro ukládání dokumentů známých textových procesorů (MS Word, Libre Office). Proveďte průzkum existujících knihoven pro práci s těmito formáty.
3. Analyzujte požadavky firmy SIGMA GROUP kladené na systém pro správu kontrol kvality zařízení. Zaměřte se na potřebu generování, ukládání a verzování dokumentů ve vybraných formátech analyzovaných v bodě 2. Pro dané potřeby navrhněte vhodný model reprezentující atributy kontrol kvality zařízení.
4. Navrhněte informační systém pro správu kontrol kvality zařízení zohledňující požadavky z bodu 3.
5. Navržený informační systém implementujte.
6. Otestujte funkcionalitu a použitelnost výsledné aplikace. Testování proveďte ve spolupráci s firmou SIGMA GROUP. Vyhodnoťte přínosy a nedostatky informačního systému.

Literatura:

- Johnson, J.: *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Burlington: Morgan Kaufmann Publishers/Elsevier, 2010, ISBN 978-0-12-375030-3.
- Řezáč, J.: *Web ostrý jako břitva: návrh fungujícího webu pro webdesignery a zadavatele projektů.* Vydání druhé. Brno: House of Řezáč, 2016. ISBN 978-80-270-0644-1.
- Microsoft: *Office developer documentation.* Online: https://msdn.microsoft.com/en-us/library/office/
- OASIS: *Standards.* Online: https://www.oasis-open.org/standards

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 4.

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

Vedoucí práce: **Hynek Jiří, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 23. října 2018

Abstrakt

Tato diplomová práce popisuje jak v teoretické, tak praktické části proces návrhu a realizace interaktivního počítačového softwaru pro správu kontrol kvality zařízení zadaného společností SIGMA GROUP a.s., resp. její divizí Energo, jejíž předmětem činností je provádění údržby a oprav čerpacích zařízení. Teoretická část vysvětluje pojem user experience, jeho metody používané při získávání, specifikaci požadavků a metody pro návrh rozhraní. Další část je zaměřena na popis formátů, které používají běžné textové editory. V rámci teoretické části je popsán samotný návrh interaktivní aplikace. V praktické části je uveden popis stěžejních částí systému a závěry z testování systému.

Abstract

This diploma thesis describes both the theoretical and practical part of the process of design and implementation of interactive computer software for the management of the quality control of the devices, assigned by SIGMA GROUP a.s., more precisely its Energo division, which is engaged in the maintenance and repair of pumping equipment. The theoretical part explains the concept of user experience, its acquisition methods, specification of requirements and interface design methods. The next part is focused on the description of the formats used by common text editors. Within the theoretical part is described the design of the interactive application itself. In the practical part there is a description of the main parts of the system and the conclusions from the system testing.

Klíčová slova

UX, Wireframe, Prototyp, analýza požadavků, UML, UI, návrh UI, kontrola kvality zařízeni, MVC, REST, React, textové formáty, DOC, DOCX, ODT, web, JS, JavaScript

Keywords

UX, Wireframe, Prototype, requirements analysis, UML, UI, draft UI, quality control of device, MVC, REST, React, text formats, DOC, DOCX, ODT, web, JS, JavaScript

Citace

ČERNÝ, Lukáš. *Systém pro správu kontrol kvality zařízení.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jiří Hynek

Systém pro správu kontrol kvality zařízení

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jiřího Hynka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Černý
15. května 2019

Poděkování

V první řadě bych zde rád poděkoval vedoucímu práce Ing. Jiřímu Hynkovi, za jeho cenné rady a připomínky, když práce směřovala špatným směrem. A také za pomoc od zaměstnanců firmy SIGMA GROUP a. s., se kterými jsem konzultoval návrhy aplikace.

Obsah

Kapitola 1

Úvod

Toto téma práce zadala firma SIGMA GROUP a. s., která je nejvýznamnějším výrobcem čerpací techniky v České republice, resp. její divize Energo, pro kterou je základním předmětem podnikání zabezpečování komplexní servisní činnosti čerpacích zařízení provozovaných v jaderné a klasické energetice, hutním průmyslu a petrochemii.

V uvedených odvětvích jsou na čerpací zařízení kladeny vysoké požadavky v oblasti dodržování stanovených provozních parametrů. S tím souvisí vysoká kvalita realizovaných servisních činností spojená s dokladováním kvality jednotlivých částí včetně celkového stavu zařízení v rámci zabezpečení technické, provozní a jaderné bezpečnosti provozovaných technologických systémů. Po celou dobu životnosti čerpacích zařízení je prováděn rozsáhlý soubor plánovaných periodických kontrol a měření, včetně zpracování, vyhodnocení a archivace příslušné dokumentace, doplněné kontrolními a vyhodnocovacími procesy neplánovaných poruchových stavů čerpacích zařízení. Více o tomto procesu je možné se dozvědět v kapitole 2.

Pro usnadnění plnění všech úkolů v oblasti zabezpečení procesu kvality v servisní činnosti má sloužit právě navržený „Systém pro správu kontrol kvality zařízení“. Navržený systém má pracovníkům v oblasti kontroly kvality hlavně usnadnit práci s přípravou záznamů a protokolů, jejich vyhodnocení a následnou archivaci jednotlivých záznamů a protokolů kvality. Pracovníkům, kteří zabezpečují péči o provozované čerpací zařízení, systém umožní na základě analýzy životnosti jednotlivých částí zpracovat podklady pro sestavování plánů v oblasti nákupu potřebných náhradních dílů, případně kompletních nových zařízení. Celkový návrh a následná specifikace potřebných metod k dosažení stanoveného cíle vychází z podrobné analýzy potřeb a požadavků pracovníků kontroly kvality. Získané informace byly zpracovány do formální podoby, na základě které bylo navrženo a otestováno grafické uživatelské rozhraní systému.

Kapitola 3 popisuje známé metody pro analýzu, návrh a testování uživatelských rozhraní. V kapitole 4 jsou tyto metody využity pro analýzu požadavků. Je zde analyzován program Mocev, který firma doposud využívala, ale už potřebám zaměstnanců nevyhovuje. Jelikož má systém pracovat s dokumenty, které používají známe textové editory jako je Microsoft Word nebo LibreOffice Writer, kapitola 5 analyzuje formáty *doc*, *docx* a *odt*, které jsou používané uvedenými editory. V kapitole 6 je popsán návrh grafického rozhraní a také serverové části, kde se vychází ze specifikace požadavků. V kapitole 7 je popsána implementace jeho hlavních částí. V kapitole 8 je popsán průběh testování ve spolupráci se zaměstnanci firmy. Jsou zde uvedena možná vylepšení a také vyhodnoceny přínosy a nedostatky aplikace.

Kapitola 2

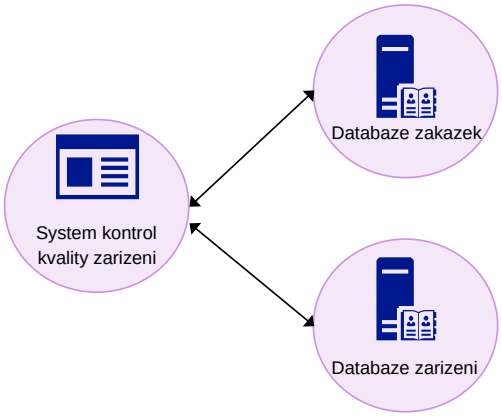
Správa kontrol kvality zařízení

V procesu realizace komplexní servisní činnosti zaujímá systém pro správu kontrol kvality zařízení (dále jen *Systém*) prioritní postavení s cílem zabezpečení stanovených technických parametrů provozovaných zařízení a jejich částí, které zásadně ovlivňují bezpečnost a spolehlivost technologických provozních systémů a v konečném důsledku mají dopad na celkovou životnost zařízení.

Systém je určen pro evidenci a vyhodnocení stanovených kontrolních operací v rámci plánované (preventivní) i neplánované servisní činnosti s následným průběžným vyhodnocováním kontrolních parametrů, sledování jejich trendů a analýzy s cílem předvídání poruch zařízení a jejich předcházení formou opravy nebo výměny kontrolovaných částí zařízení.

2.1 Popis ukládaných dat

Systém umožňuje základní evidenci stanoveného souboru servisovaného zařízení (dále jen *Zařízení*) se zobrazením vybraných informací prostřednictvím karet zařízení a přehled realizovaných servisních činností (dále jen *Zakázky*). Současně *Systém* umožňuje pomocí filtrů vyhledávání záznamových karet *Zařízení* nebo souboru *Zakázek* vyhovujících požadovaným kritériím (otevřené nebo ukončené *Zakázky*). Na obrázku 2.1 lze vidět schéma systému a spolupracujících databází.



Obrázek 2.1: Schéma systému a databáze, se kterými systém pracuje.

Databáze *Zařízení* zobrazuje vybrané údaje z karet *Zařízení* a celkový rozsah databáze koresponduje s počtem servisovaného zařízení, u kterých je komplexní servisní činnost realizována na základě smluvních vztahů s jednotlivými zákazníky. Údaje o zařízeních jsou následující:

- druh,
- výrobce,
- typ,
- velikost,
- pracoviště,
- projektové označení,
- výrobní číslo,
- vyhláška,
- zakázka / obchodní případ.

Databáze *Zakázek* zobrazuje vybrané údaje realizovaných *Zakázek* v základním rozlišení ukončených nebo v daném okamžiku realizovaných *Zakázek*. *Zakázky* mohou být *otevřené* nebo *ukončené*. Jednotlivé zakázky jsou samostatně označovány generovaným číslem dle stanovené metodiky číslování dokumentů integrovaného systému kvality divize Energo a vztahují se ke konkrétnímu zvolenému *Zařízení* ze základní databáze *Zařízení*.

Každé servisované *Zařízení* je specifické svými provozními parametry, účelem použití a konstrukčním uspořádáním, technickou úrovní jednotlivých částí Z těchto základních údajů vychází požadovaná úroveň kvality a rozsah kontrolovaných parametrů pro bezpečný provoz a dodržení projektované životnosti *Zařízení*.

2.2 Typy kontrol

Dosažení celkové požadované úrovně kvality *Zařízení* a jeho částí je podmíněno rozsahem stanovených kontrolních operací, a především úrovní prováděných záznamů za dodržení podmínek platné legislativy v procesu kontrolní činnosti kvality *Zařízení*. Základní rozdělení kontrol:

- **Servisní kontroly**: Kontroly prováděné v průběhu realizované *Zakázky* v rozsahu údržby nebo kontroly *Zařízení* s cílem kontroly stanovených konstrukčních parametrů a jejich hodnot. Kontroly jsou prováděny na stanovených částech *Zařízení* dle schválených metodik technických kontrol a v rozsahu stanoveného a schváleného technologického postupu. Podmínky, výsledky a celkové vyhodnocení jsou zpracovány ve formě záznamů a systémově ukládány pro následné vyhodnocení a analýzu způsobilosti pro následný provoz nebo přijetí nápravných opatření v rámci zjištěných neshod, případně hodnocení trendu zhoršujících se kontrolovaných parametrů z hlediska dosažení projektové životnosti v závislosti na plánované době provozu *Zařízení*.

- **Provozní kontroly**: Kontroly prováděné na *Zařízení* s cílem ověření správné funkce a technických parametrů a jejich limitních hodnot v předepsaných provozních režimech daného zařízení. Kontroly jsou prováděny na provozovaném *Zařízení* dle schválených metodik technických kontrol a v souladu s provozní dokumentací příslušného *Zařízení*. Podmínky, výsledky a celkové vyhodnocení jsou zpracovány ve formě záznamů (viz kapitola 2.2.1) a systémově ukládány pro následné vyhodnocení a analýzu způsobilosti pro následný provoz nebo přijetí nápravných opatření v rámci zjištěných neshod, případně hodnocení trendu zhoršujících se kontrolovaných parametrů z hlediska dosažení projektové životnosti v závislosti na plánované době provozu *Zařízení*.

2.2.1 Typy záznamů

Základní rozdělení záznamů kontrol:

- **Zápis**: Písemný záznam o provedení předepsané kontroly formou odepsání provedení dané operace v příslušném dokumentu bez uvedení předepsaných kritérií a bez uvedení zjištěných údajů. Formulář není vystavován ze *Systému* a není číslován, je vystaven z odděleného integrovaného systému kvality divize Energo a po vyhodnocení je ve formátu PDF ukládán do *Systému* v databázi příloh. Zápis provádí pracovník pracovní skupiny provádějící servisní činnosti.

- **Záznam**: Písemný záznam o provedení předepsané kontroly s uvedením předepsaných kritérií a s vyhodnocením zjištěných údajů. Formulář není vystavován ze *Systému* a není číslován, je vystaven z odděleného integrovaného systému kvality divize Energo a po vyhodnocení je ve formátu PDF ukládán do *Systému* v databázi příloh. Zápis provádí pracovník pracovní skupiny provádějící servisní činnosti.

- **Protokol**: Písemný záznam o provedení předepsané kontroly s uvedením předepsaných kritérií, měřidel (odkazem na dokumentaci, metodiku) a vyhodnocení zjištěných údajů. Jednotlivé formuláře jsou vystaveny ze *Systému* a jsou samostatně označovány generovaným číslem dle stanovené metodiky číslování dokumentů integrovaného systému kvality divize Energo. Protokol vystavuje a elektronicky potvrzuje oprávněný zástupce divize Energo, funkčně nezávislý na výrobních (servisních) útvarech. Protokol je v elektronické formě uložen v *Systému* pro možnost následného užití.

Všechny protokoly vystavované ze *Systému* jsou vytvořeny jako dokumenty aplikace MS Word, popřípadě LibreOffice Writer a využívají schválených šablon. Tyto dokumenty lze následně editovat dle vlastních požadavků. V *Systému* je uložen pouze odkaz na takto vytvořený dokument (protokol).

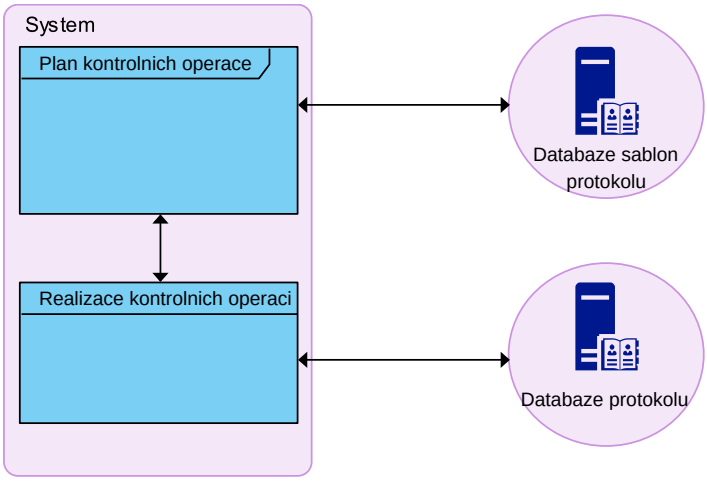
2.3 Proces servisní činnosti

V procesu servisní činnosti (realizace údržby a oprav *Zařízení*) se průběh jednotlivých servisních úkonů řídí stanoveným a schváleným technologickým postupem, ve kterém jsou chronologicky uvedeny předepsané pracovní operace včetně příslušných kontrolních operací, u kterých jsou uvedena kritéria kontrol s odkazem na vystavení odpovídající úrovně kvality záznamu.

V *Systému* je pro každé jednotlivé *Zařízení* stanoven plán kontrolních operací (servisní kontroly), který je sestaven v maximálním rozsahu odpovídajícímu plánované realizaci generální opravy *Zařízení* a v souladu s příslušným technologickým postupem tj. kom-

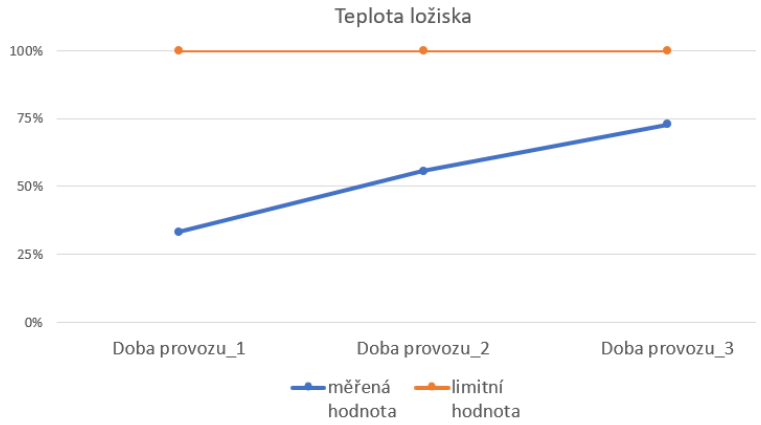
pletní demontáž *Zařízení*, kontrola jednotlivých částí, oprava nebo výměna neshodných částí a zpětná montáž *Zařízení*. Součástí plánu kontrol jsou i kontrolní operace v rámci provozních kontrol, u kterých je celkový rozsah stanoven technickou dokumentací výrobce *Zařízení* a provozní dokumentací provozovatele *Zařízení*. Výběr jednotlivých kontrolních operací je prováděn z číselníku šablon kontrolních operací.

V rámci konkrétní realizované *Zakázky* je v přípravné fázi stanovený plán kontrolních operací potvrzen v plném rozsahu nebo v souladu se specifikací smluvního zadání *Zakázky* je rozsah plánu editován. V průběhu následné realizace *Zakázky* jsou u jednotlivých potvrzených kontrolních operací vytvářeny záznamy formou editace připravených šablon protokolů. Po vyhodnocení kontrolní operace a elektronického schválení příslušného protokolu je v aplikaci *Systému* vytvořen odkaz na takto vytvořený a odsouhlasený dokument. Tento proces je znázorněn na obrázku 2.2.



Obrázek 2.2: Schéma systému a databáze, se kterými se pracuje.

Systém je doplněn o aplikaci tabulkově nebo graficky zpracovaných dat kontrolních operací vytipovaných částí *Zařízení*, u kterých je výrobcem *Zařízení* stanovena projektová životnost, nebo u kterých je z hlediska legislativních požadavků stanovena vysoká úroveň bezpečného provozu *Zařízení*.



Obrázek 2.3: Ukázka graficky zpracovaných dat.

Cílem aplikace je monitorování stanovených dat kontrolních operací ve stanoveném časovém období s vyhodnocením trendu opotřebení částí zařízení nebo stanovení provozní doby do dosažení stanovených limitních hodnot technických parametrů částí zařízení. Ukázku grafického zpracování monitorovaných dat lze vidět na obrázku 2.3.

Navržený *Systém* a jeho aplikace v procesu servisní komplexní činnosti zajistí plnění stanovené úrovně kvality celého souboru požadavků kladených na správnou funkci certifikovaného integrovaného systému kvality divize Energo v rámci plnění stanoveného předmětu podnikání.

2.4 Integrace systému

Systém svým uceleným souborem funkcí plně zajišťuje své nedílné postavení v celém integrovaném informačním systému divize Energo a to především v oblasti:

- On-line přístup k záznamům kvality jednotlivých servisních středisek včetně vedení společnosti.
- Zajištění přístupových práv uživatelů v souladu se stanoveným systémem pravomocí a odpovědností.
- Plnění stanovených procesů v oblasti systému kontrol kvality *Zařízení* v rámci integrovaného systému kvality společnosti.
- Archivace záznamů kvality v souladu s platnou legislativou, a to zejména v oblasti servisní činnosti v jaderné energetice, v prostředí s nebezpečím výbuchu, tlakových zařízení a elektrotechnice.

Kapitola 3

User experience

User experience (dále jen UX) je pojem, který při vývoji uživatelského rozhraní hraje velmi důležitou roli. UX klade důraz na návrh a tvorbu produktu, který je užitečný, použitelný, splňuje požadavky uživatelů a soustředí se pozitivní zkušenost (*experience*) uživatelů. Pojem UX lze chápat různými způsoby, a proto se čtenáři mohou setkat s různými definicemi. Podle normy ISO 9241-210 [12] z roku 2010 je UX definován jako:

> Person's perceptions and responses resulting from the use and/or anticipated use of a product, system or service.

Pro srovnání, například [2] chápe UX jako všechny aspekty toho, jak lidé používají výrobek, jak dobře chápou funkcionalitu produktu, jaký mají dojem z používání, jak dobře slouží výrobek svým účelům a jak dobře vše zapadá do celého kontextu, ve kterém je produkt používán. Podle [9] UX zahrnuje všechny aspekty interakce koncového uživatele s firmou, jejími službami a produkty.

Z výše uvedených definic a z názvu vyplývá, že uživatel hraje v této etapě hlavní roli. Hlavním cílem UX je získat od uživatelů potřebné informace a navrhnout grafické rozhraní, které bude splňovat požadavky, usnadní uživatelům práci a bude jim vyhovovat. Sekce 3.1 popisuje metody, které lze použít pro získání informací od uživatelů. Sekce 3.2 popisuje metody pro zpracování a analýzu získaných informací a stanovení si požadavků na aplikaci. Na základě zpracovaných informací a specifikovaných požadavků je možné provést návrh rozhraní. K tomu mohou být využity tzv. drátěné modely (*wireframes*) a prototypy rozhraní popsané v sekci 3.3. Poslední etapou, popsanou v sekci 3.4, je testování návrhu uživatelem.

3.1 Získání informací

Cílem této fáze je zjistit co nejvíce informací od uživatelů, kteří budou s aplikací pracovat. Jaké požadavky mají uživatelé na aplikaci a co od ní očekávají. Tyto informace můžeme získat různými metodami, které lze rozdělit do dvou skupin [15], kde každá se hodí pro jiné účely:

- kvantitativní,
- kvalitativní.

Kvantitativní metody slouží pro zmapování většího počtu uživatelů. Typickým představitelem je dotazník. Za pomocí metod tohoto typu se získávají informace například o tom,

jaký je finanční příjem rodiny, jaké služby uživatelé využívají (například ICQ, Facebook) nebo jaké zařízení mají uživatelé doma. Získaná data jsou objektivní, lze je vyjádřit číslem, měřit nebo také zpracovávat počítačem.

Kvalitativní metody slouží pro získání detailnějších a přesnějších informací. Nedají se použít hromadně jako metody kvantitativní na více uživatelů, ale tyto metody jsou obvykle náročnější na čas. V následujících podkapitolách si popíšeme dva zástupce kvalitativního výzkumu, a to rozhovor a přímé pozorování. Z kvalitativních metod jsou získaná data subjektivní, nelze je vždy vyjádřit číslem, ale slovy.

3.1.1 Interview

Rozhovor [15, 24] (z angličtiny *Interview*) s uživateli je jedním z klíčových nástrojů pro ověření prvních nápadů a prototypů. Rozhovory s uživateli jsou strukturované konverzace se současnými nebo potenciálními uživateli stránek. Mohou být prováděny po telefonu, osobně na neutrálním místě (např. V konferenční místnosti) nebo v ideálním případě v prostředí, ve kterém uživatel pravděpodobně stránky používá.

Cílem rozhovoru je získání podrobnějších informací, zjištění preferencí a postojů účastníků. Návrhář uživatelského rozhraní ví, co se v současné době používá a jaké jsou trendy, ale pravděpodobně není cílová skupina, takže nápady návrháře by mohly být nevyhovující. Tato metoda bude použita v kapitole 4 pro získání informací od zaměstnanců firmy.

3.1.2 Přímé pozorování

Tato metoda patří k základním technikám sběru dat [18]. Při přímém pozorování jde o bezprostřední pozorování procesů, činností podle definovaného plánu, bez dotazů a jakéhokoliv ovlivňování pozorovaného subjektu. Pozorovaným subjektem může být například chování osoby či chování skupiny v nějaké situaci. Přímé pozorování je omezeno možnostmi vizuálního, sluchového, časového a prostorového záběru reality pozorovatelem.

Během rozhovoru se může stát, že si účastník na nějaký detail nevzpomene. Tento problém lze vyřešit pomocí přímého pozorování. Při něm si můžeme sami všimnout detailu, který by uživatel opomenul nebo jej nepovažoval za důležitý.

Použitím této metody je možné zjistit, jak se uživatel chová ve svém přirozeném prostředí, co dělá, a hlavně jak využívá náš produkt. V kapitole 4 bude metoda použita při analýze programu Mocev.

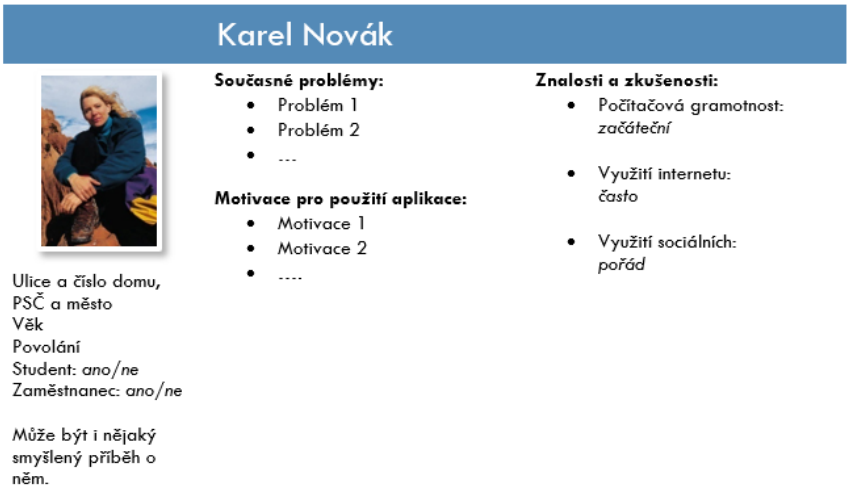
3.2 Formální specifikace požadavků

Poté, co jsme od zákazníka získali požadavky na systém a od uživatelů jsme zjistili kromě jiného, co očekávají od systému, je vhodné tyto informace zpracovat do podoby, se kterou se nám bude lépe pracovat. Informace získané z dotazníků a rozhovorů jsou neformální a subjektivní. Je vhodné začít tvorbou tzv. *persony*, protože jejím vytvořením vznikne pár abstraktních osob, kde každá reprezentuje skupinu tázaných lidí. Po takovém zpracování můžeme už jednoduše sepsat požadavky na aplikaci. Pro sepsání požadavků lze použít diagramy patřící do UML[1] (zkratka z anglického *Unified Markup Language*). O zmíněných technikách se můžeme dozvědět v následujících podsekcích.

[1] Více informací lze nalézt na https://www.techopedia.com/definition/3243/unified-modeling-language-uml. Vhodným diagramem je tzv. *diagram případů užití*, který definuje interakce mezi rolemi uživatelů, které se systémem mohou pracovat a aplikací. Pokud je nějaká interakce složitější, můžeme ji popsat pomocí diagramu aktivit. U diagramy patří do .

3.2.1 Persony

Persony [20, 10] někdy bývají též nazývané uživatelskými profily. Jsou to smyšlené postavy, které reprezentují potřeby celé řady skutečných uživatelů. Obrázek 3.1 reprezentuje příklad persony. Persona obsahuje smyšlené jméno, fotku a údaje pro lepší ilustrativnost. Persony se používají v rámci UX, ale i v marketingu. Odlišují se hlavně v tom, co zobrazují. Naše persony zobrazují popis interakce, motivaci k použití systému, zatímco marketingové persony popisují, co uživatelé chtějí zobrazit. V kapitole 4 bude metoda použita pro zmenšení počtu uživatelů a vytvoření jednotného pohledu na to, jací uživatelé budou systém používat.



Obrázek 3.1: Ukázka smyšlené persony

Správný počet person není nikde stanovený. Jejich počet záleží na tom, jak jsou moc uživatelé odlišní. Většinou bývá zvykem mít až 2 až 6 person, ale může jich být i více. Je to z jednoduchého důvodu. Naše paměť je schopná pracovat s omezeným množstvím lidí (person). Pokud je v projektu více než šest osob, není snadné od sebe osoby odlišit a také je obtížné si pamatovat detaily o každé osobě.

3.2.2 Diagram případů užití

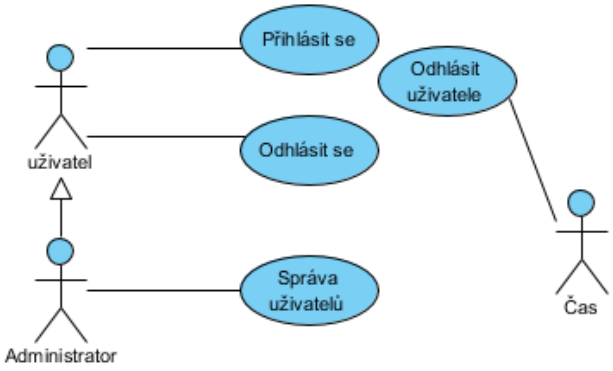
Diagram případů užití [17] (z angličtiny *Use case diagram*) je jedním z nástrojů softwarového inženýrství pro specifikaci požadavků na systém. Diagram popisuje, co má systém dělat, tj. funkční požadavky systému. Nespecifikuje však, co systém neudělá.

Použitím tohoto nástroje na začátku vývoje systému, tedy při specifikaci požadavků, ušetří návrhářům UI a programátorům čas i peníze, protože při tvorbě tohoto diagramu lze odhalit nějaké nesrovnalosti, na základě nich upravit návrh, popřípadě udělat malé změny v implementaci. V opačném případě by bylo už náročnější provedení odpovídajících změn v systému. V kapitole 4 bude metoda použita pro formální specifikaci požadavků, které vychází ze získaných informací od zaměstnanců firmy.

Pomocí diagramu případů užití můžeme:

- **Definovat role**: Mohou komunikovat se systémem a v diagramu jsou zobrazeny jako postavy (aktéři) s názvem.

- **Definovat dědičnost mezi rolemi**: V diagramu je dědičnost znázorněna jako šipka mezi aktéry. Šipka směřuje od rodiče k potomku.
- **Definovat případ užití**: Je to interakce, kterou systém umožňuje. V diagramu je znázorněn jako elipsa s názvem interakce. Pokud chceme přiřadit interakci konkrétní roli, propojíme je pouze úsečkou.
- **Zahrnout v jedné interakci i jinou interakci**: Pokud máme případ užití, který provádí operaci, kterou chceme zdůraznit, můžeme vytvořit nový případ užití a propojit je úsečkou, kterou nazveme *«include»*. Klíčové slovo *include* znamená, že se vždy případ užití provede.



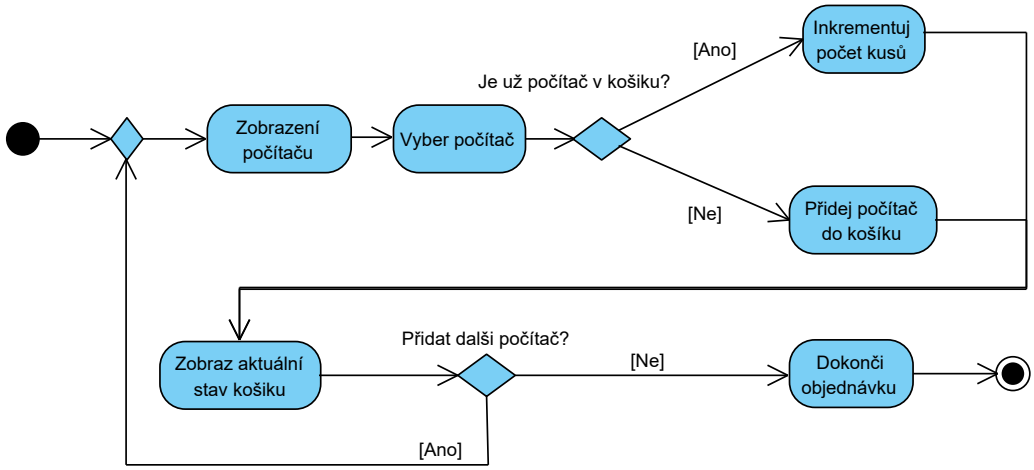
Obrázek 3.2: Ukázka Use case diagramu

3.2.3 Diagram aktivit

Diagram aktivit [17] (z angličtiny *Activity diagram*) je jedním z nástrojů softwarového inženýrství pro popis procesů, jak v systému dosáhneme našeho cíle, tj. jaké akce, kroky musíme provést pro dosažení našeho cíle. Tento diagram je využitý v kapitole 4 pro popis složitějších procesů. Diagramy aktivit se můžou skládat z následujících prvků:

- **Počátek procesu**: v diagramu se označuje jako černý bod
- **Ukončení procesu**: v diagramu se označuje jako černý kruh s černým bodem uprostřed
- **Akce**: je znázorněna jako obdélník se zaoblenými rohy, který obsahuje název prováděné akce
- **Tok**: znázorňuje se šipkou ve směru provádění akcí
- **Selekce**: kosočtverec, který vybírá podle podmínky, která je u něj uvedena, následující tok
- **Spojení**: v diagramu se zobrazuje jako kosočtverec, který spojuje dva a více toků do jednoho

Výše popsané prvky můžeme vidět na ukázce 3.3, která popisuje proces objednání počítače.



Obrázek 3.3: Ukázka diagramu aktivit

3.3 Návrh uživatelského rozhraní

Na základě dříve uvedených etap víme, co se od systému požaduje a jaké by měl mít systém funkce, a tak se můžeme přesunout k návrhu, jak by mělo rozhraní vypadat. Návrh je další etapou, která specifikuje rozložení elementů na obrazovce a přibližný grafický návrh UI.

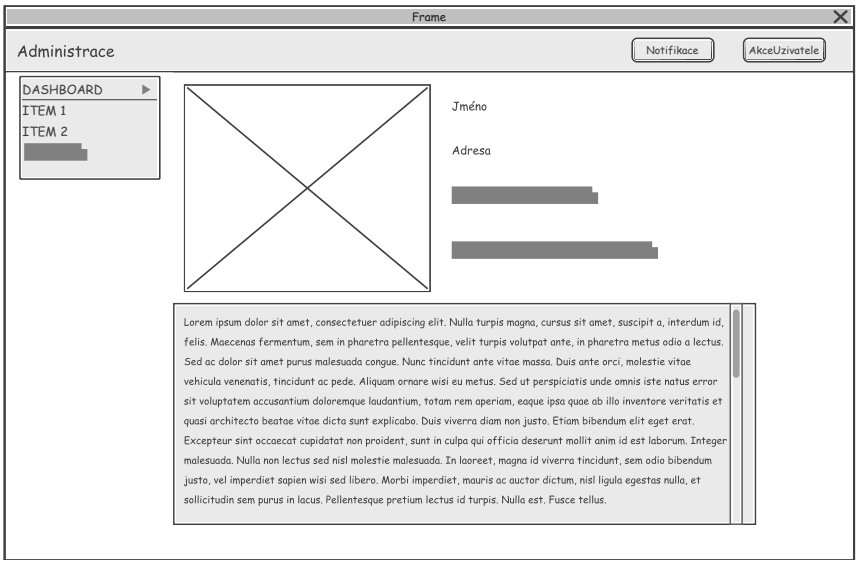
V prvním kroku začneme tvorbou drátěných modelů. Můžeme vytvořit více verzí modelů, aby si uživatel při testování měl z čeho vybírat. Po výběru vhodného drátěného modelu se můžeme přesunout k tvorbě prototypů, které jsou opět testovány uživateli. Kdybychom vynechali drátěné modely a začali rovnou prototypem, hrozilo by vytvoření nevyhovujícího prototypu, který by bylo nutné předělat. Z tohoto důvodu by návrh UI zabral více času, protože upravit prototyp je složitější, než upravit drátěné modely a poté vytvořit prototyp. Drátěné modely jsou jednoduché statické modely, které se dají rychle vytvořit, nezáleží u nich, jak rozhraní vypadá, ale jak jsou rozmístěny funkční prvky na stránce.

3.3.1 Drátěný model

Drátěný model (anglicky *wireframe*) [5] je diagram založený na tom, že umožňuje jednoduše definovat rozložení elementů na obrazovce, a to bez jakýchkoliv aspektů návrhu. Jak můžeme vidět na ukázce 3.4, jedná se o zjednodušený pohled na obrazovku zbavený grafiky. Snaží se zaměřit pozornost na to, co obrazovka dělá a ne jak vypadá. Nemusí se vytvářet vždy celkový pohled, ale je možné se zaměřit na konkrétní prvek. Například lze vytvořit model, který bude zobrazovat strukturu nákupního košíku. Tento model bude použit v kapitole 6 při návrhu nového rozhraní.

Na obrázku 3.4 můžeme vidět ukázku drátěného modelu, ve kterém se nezobrazují barvy, maximálně stupně šedi, nepoužívají se skutečné obrázky ani texty, ale pouze schématicky se naznačí, že na daném místě má být obrázek, texty se používají náhodné (např. generované pomocí Lorem Impsum[2]) a položky v horním a postranním menu jsou smyšlené, alespoň některé. Podstatná je zde informace, že menu bude na levé straně a hlavní navigační panel bude nahoře. Často se může stát, že po okrajích jsou vidět poznámky k různým elementům na modelu, které popisují, co by měl daný element dělat.

[2] Je to generátor pseudonáhodných textů. Dostupný na: http://www.lorem-ipsum.cz



Obrázek 3.4: Ukázka drátěného modelu

3.3.2 Prototyp

Prototyp [10, 4] je další krok v návrhu uživatelského rozhraní, který vychází z drátěného modelu. Hlavní věcí, ve které se odlišují od drátěných modelů, je možnost nějaké interakce, tj. proklikávání mezi pohledy. To představuje velkou výhodou, protože při testování se uživatel dokáže lépe „vcítit“ do rozhraní. Tvorbou prototypů získáme při testování zpětnou vazbu, aniž bychom strávili hodiny implementací rozhraní. Dalším rozdílem oproti drátěnému modelu je, že se prototyp už podobá výslednému uživatelskému rozhraní. Také respektuje rozložení prvků, které specifikují drátěné modely.

Jak už bylo řečeno, tak v prototypu bude fungovat proklikávání mezi různými pohledy. Například při kliknutí na položku z menu se zobrazí odpovídající pohled. Může fungovat i jiná interakce s uživatelem. Tato metoda bude použita v kapitole 6, kde se pro vytvoření prototypu použijí jazyky HTML[3] a CSS3[4].

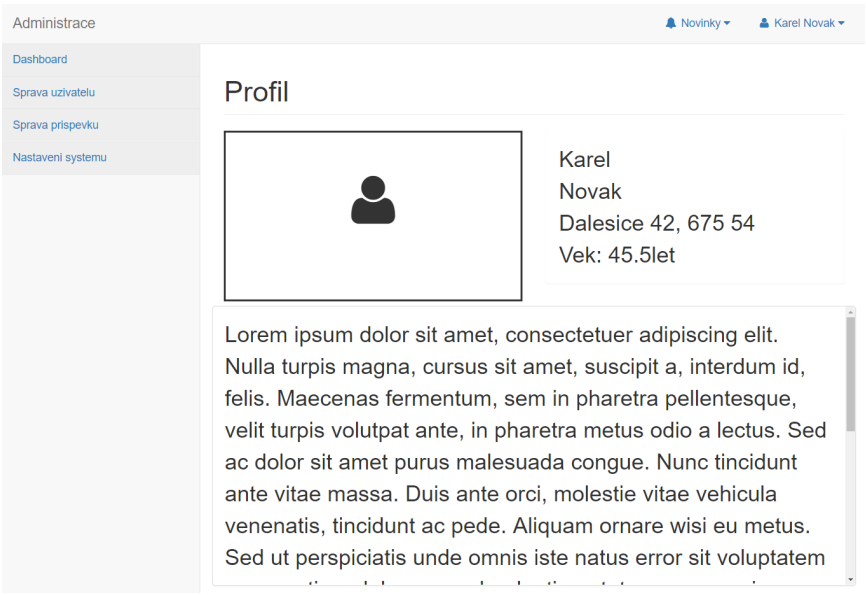
Obrázek 3.5 vznikl z drátěného modelu 3.4 a můžeme v něm vidět, že postranní menu obsahuje reálné položky. To stejné platí i pro hlavní navigační menu. Z grafické stránky to vypadá lépe. Jelikož chceme vytvářet prototypy, které se co nejvíce přibližují finálnímu návrhu, tak už v tuto chvíli musíme vybírat takové prvky rozhraní, které uživateli pomohou dosáhnout svého úkolu a zvolit uspořádání prvků, které bude snadno srozumitelné a snadno použitelné. Je plno věcí, na které musíme při návrhu myslet, například volit vhodná slova nebo poskytovat zpětnou vazbu. Více se dozvíme v sekci 3.3.2.

Pravidla pro návrh rozhraní

V této podkapitole si uvedeme pravidla, kterými je vhodné se řídit při návrhu rozhraní. Můžeme se inspirovat pravidly, které sepsal Ben Schneiderman [23]. Při návrhu bychom se měli držet nějakých pravidel. Níže uvedená pravidla nemusíme striktně dodržovat, ale můžeme si je upravit podle druhu projektu.

[3] Je to zkratka „Hypertext Markup Language“. Více informací na: https://techterms.com/definition/html

[4] Je to zkratka „Cascading Style Sheet“. Více informací na: https://techterms.com/definition/css



Obrázek 3.5: Ukázka prototypu

- **Usilujeme o konzistenci**: Toto pravidlo bývá nejčastěji porušováno. Existuje mnoho forem konzistence:
  - v podobných situacích by se měl systém chovat podobně (konzistentně)
  - stejná terminologie ve výzvách a nabídkách
  - barvy, rozložení, styl písma atd. by měly být v rámci celé aplikace stejné
- **Rozpoznat potřeby různých uživatelů**: Návrhář by si měl ujasnit, kteří uživatelé budou aplikaci používat a na základě toho přidat funkce. Například pokud to budou začátečníci, tak by neměly chybět vysvětlivky a nápovědy, ale pokud to budou třeba experti, tak může přidat nějaké klávesové zkratky, které urychlí práci se systémem.
- **Poskytujeme zpětnou vazbu**: Pro každou akci uživatele by měla být zpětná vazba systému. Uživatel by měl být informován o výsledcích operací, které provádí v systému, zda uspěly či neuspěly. Zpětná vazba je důležitá, ale pokud systém bude informovat v podstatě o všem (například: že položky dané kategorie se zobrazily), tak to začne být pro uživatele otravné.
- **Předcházení chybám**: Pokud je to možné, tak navrhnout systém tak, aby předcházel chybám, které uživatel může vyvolat. Systém musí uživatelům umožnit vrátit se zpět a opravit chybu. Například při vyplňování rodného čísla ve formuláři tak uživatele můžeme informovat o chybně vyplněné hodnotě.
- **Možnost vrátit se zpět**: Pokud je to možné, tak by měl systém umožnit vrátit se zpět. Například vytváříme objednávku, všechno vyplníme a dokončíme objednávku, následně se zobrazí náhled objednávky, kde uvidíme, že máme chybnou adresu, tak díky možnosti se vrátit zpět, můžeme chybu napravit.
- **Uživatel inicializuje a systém reaguje**: Uživatel musí mít pocit, že on řídí systém a rozhraní reaguje na jeho akce. Rozhraní musí být předvídatelné. Pokud bude rozhraní reagovat nepředvídatelně, tak uživatel nebude mít dobrý pocit z rozhraní.

- **Nepřetěžování lidské paměti**: Uživatel nesmí být nucen si rozhraní pamatovat. Rozhraní by mělo být přehledné. Lidská krátkodobá paměť si dokáže uchovat v paměti 7 ± 2 věcí (informací), z toho vyplývá, že rozhraní by mělo být také jednoduché.

3.4 Evaluace použitelnosti

Testovaní návrhu a celého systému [25] z hlediska UX znamená, že musíme zjistit, kromě jiného, jak se uživatel cítí v našem systému a také musíme zjistit, zda návrh splňuje očekávání uživatelů. Hlavní zaměření metod pro hodnocení návrhu je pomoct při výběru nejlepšího návrhu, zjištění toho, aby byl vývoj na správné cestě nebo zda konečný produkt splňuje původní cíle a požadavky.

Testování návrhu můžeme provádět kdykoliv. Je vhodné nejdříve vytvořit několik typů drátěných modelů s různým rozložením elementů, ty nechat vyhodnotit a na základě výběru se vrátit k návrhu a vytvořit interaktivní model, a ten poté nechat opět vyhodnotit.

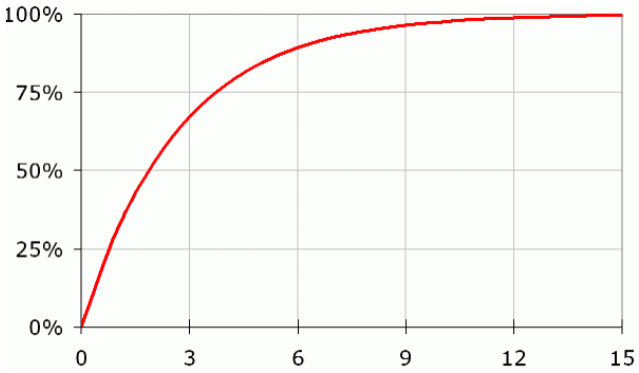
Pro výběr nejvhodnějšího drátěného modelu můžeme s každým uživatelem modely projít, ohodnotit je například číslem 1-5, kde 1 nejvíce vyhovuje. Poté spočítáme průměrné ohodnocení pro každé modely a vybereme takový, který bude mít nejlepší tj. nejnižší průměrné ohodnoceni.

Když máme vybraný model, můžeme přejít k tvorbě prototypu, který opět necháme otestovat uživateli, abychom zjistili, zda návrh vyhovuje jejím požadavkům a splňuje to co od systému požadovali. Pro otestování máme více metod, které můžeme použít. Tyto metody lze podle [13] rozdělit do následujících kategorií:

- **uživatelské testování** (z anglického *user testing*): Hodnotitel pozoruje, jaké interakce uživatel se systémem provádí. Patří sem například metoda uživatelské testování, která bude popsána v sekci 3.4.1.
- **inspekční metody** (z anglického *inspection methods*): Hodnotitel hledá chyby v rozhraní a k tomu používá sadu kriterii nebo heuristik. Patří sem například:
  - **kontrola funkce**: hodnotí se funkce produktu,
  - **kontrola standardů**: kontroluje se dodržování norem.
- **dotazovací metody** (z anglického *inquiry methods*): Uživatelé poskytují zpětnou vazbu prostřednictvím rozhovorů, průzkumů apod. Lze sem zařadit například:
  - **dotazník**: poskytuje odpovědi na konkrétní otázky,
  - **pozorování**: sleduje se uživatel při používání systému v jeho přirozeném prostředí.
- **analytické metody** (z anglického *analytical methods*): Hodnotitel využívá uživatelé a modely rozhraní pro generování předpovědí. Například sem patří:
  - **analýza návrhu**: posuzuje složitost návrhu,
  - **analýza kognitivních úloh**: předpovídá problémy s použitelností.
- **simulační metody** (z anglického *simulation methods*): Hodnotitel využívá modely uživatelského rozhraní k napodobování interakce se systémem. Patří sem například:
  - **Model Petriho sítě**: napodobují interakci uživatele s daty,

– **Generické algoritmy**: napodobují interakci nových uživatelů.

Podle [19] pro testování stačí 5 uživatelů, kteří při testech dokážou odhalit téměř 85% chyb. Lze to vyčíst z obrázku 3.6. První testovací uživatel dokáže odhalit nejvíce chyb. S dalším testovacím uživatelem se odhalí další chyby, ale už méně než předchozí testovací uživatel.



Obrázek 3.6: Graf závislosti počtu odhalených chyb na počtu testovacích uživatelů. Převzato z [19].

Pro otestování prototypu a výsledné aplikace jsem vybral následující metody.

3.4.1 Uživatelské testování

Podle [1] je testování použitelnosti (z angličtiny *Usability testing*) sbírka technik používaných k měření charakteristik uživatelské interakce s produktem. Cílem metody je zhodnotit použitelnost produktu. Pro aplikování metody potřebujeme funkční produkt nebo alespoň prototyp. Zaměřujeme se v ní na měření toho, jak uživatelé mohou dokončit konkrétní úkoly a jaké problémy při tom nastanou. Metoda bude použita v kapitole 6 při testování prototypu a v kapitole 8 při testování výsledné aplikace.

Metoda probíhá tak, že účastník pod dohledem návrháře provádí specifické úlohy. Mohou to být například úlohy specifikované v diagramu užití. Při testování může návrhář měřit:

- čas, potřebný na spravení úlohy
- počet chyb
- počet kliknutí

Podle [1] lze pomocí testování použitelnosti otestovat:

- **Pojmenování**: Jsou vhodná označení sekcí a názvy tlačítek v rámci kontextu programu?
- **Organizace**: Jsou zobrazené informace seskupeny do smysluplných kategorií? Zobrazují se informace, které uživatelé potřebují a hledají?
- **Prvotní použití a objevitelnost**: Jsou obyčejné položky snadno nalezitelné pro nové uživatele? Jsou pokyny jasné? Jsou pokyny potřebné?
- **Efektivitu**: Dokáží uživatelé efektivně dokončit konkrétní úkoly? Dělají chyby? Kde dělají chyby? Jak často dělají chyby?

Kapitola 4

Analýza požadavků

V této kapitole jsou aplikovány metody uvedené v kapitolách 3.1 a 3.2 a je zde provedena analýza systému Mocev, který firma doposud používala. Cílem této práce je navrhnutí a implementace informačního systému pro správu kontrol kvality zařízení.

Nejdříve bylo potřebné získat informace od zaměstnanců firmy, kteří navržený systém budou používat nebo používali program Mocev. S každým zaměstnancem jsem provedl rozhovor. Pro získání více informací o programu Mocev jsem dostal kontakt na zaměstnance, který mi přiblížil práci v něm. V tomto případě jsem použil metodu přímého pozorování.

Získané informace jsem zpracoval do person, požadavky na systém jsem znázornil pomocí diagramu případů užití a informace získané pozorováním a analýzou programu jsem zpracoval pomocí diagramu aktivit. V následujících podkapitolách jsem popsal průběh rozhovorů, přímého pozorování a získané výsledky.

4.1 Interview

Na začátku rozhovorů jsem se nejdříve snažil zjistit, co zaměstnanec dělá ve firmě a poté, jak si představuje nový systém. Rozhovory probíhaly v jejich přirozeném prostředí, tedy v jejich kancelářích. Pokud jsem prováděl rozhovor se zaměstnancem pracujícím na pozici kontrolor kvality, tak se rozhovoru účastnil i jeho nadřízený. Před začátkem rozhovoru jsem zaměstnance upozornil, že rozhovor je nahráván, abych nezdržoval psaním poznámek.

- Jaká je náplň Vaší práce?
- Na co bude nový systém využíván?
- Co by měl umět nový systém jiného než současný?

Výše uvedené otázky se týkají spíše nového systému. Pokud jsem prováděl rozhovor se zaměstnancem, který již pracoval v programu Mocev, tak jsem se ještě ptal na další doplňující otázky:

- Co za údaje, kromě naměřených dat, jsou v protokolu?
- Jakého druhu jsou měřené údaje?
- Kolik údajů se v protokolu měří?
- Jaké jsou nedostatky programu Mocev?

- Jaké funkce programu nejvíce používáte?
- Co program neumožňuje a Vy byste to potřeboval?

Provedl jsem celkem 7 rozhovorů. Z toho 3 rozhovory byly provedeny se zaměstnanci na vedoucích pozicích a ostatní rozhovory byly s pracovníky na pozici kontrolor kvality. Výsledky rozhovorů jsem zpracoval pomocí person v sekci 4.4 a také pomocí diagramů případu užití v sekci 4.5.

4.2 Přímé pozorování

Tuto metodu jsem použil pro získání informací o programu Mocev, jaké má funkce a jak se s ním pracuje. V následujících bodech popíšu, co jsem se potřeboval o programu dozvědět.

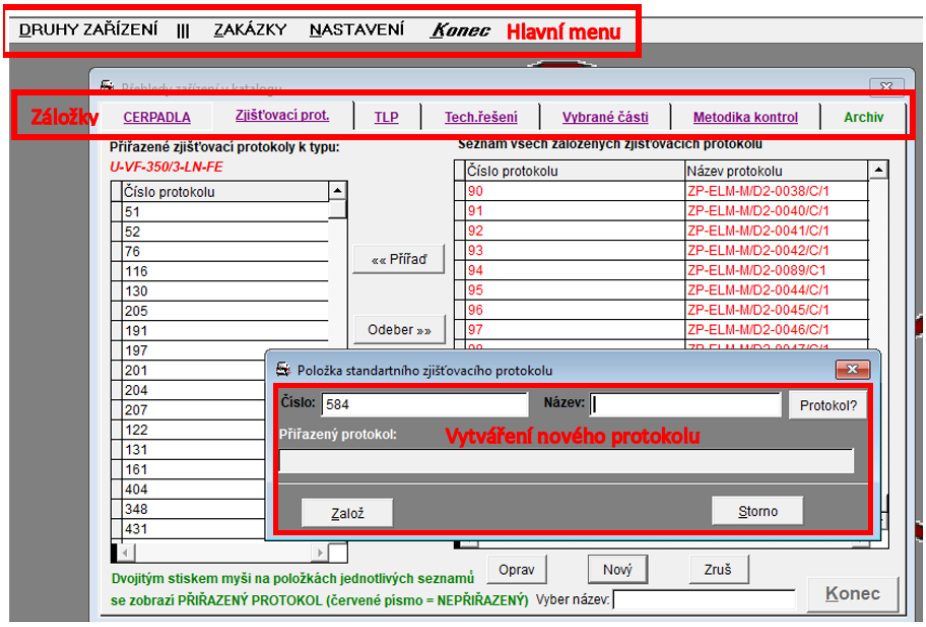
- **Jak se vyhledává konkrétní zařízení?** V hlavním menu se vybere druh zařízení a zobrazí se seznam všech zařízení vybraného typu. V seznamu lze vyhledávat podle jeho označení.
- **Jak se vytvoří nový protokol** a **jak se přiřadí ke konkrétnímu zařízení?** V hlavním menu se vybere druh zařízení, ze seznamu zařízení se vybere konkrétní zařízení. Poté se změní záložka na *Zjišťovací protokoly*, kde se zobrazí dva seznamy. První (levý) seznam obsahuje protokoly přiřazené k vybranému typu zařízení a druhý (pravý) seznam obsahuje všechny založené seznamy. Když bychom chtěli vytvořit nový protokol, vybereme položku nový a vyplníme potřebné údaje. Pokud bychom chtěli přiřadit protokol k vybranému typu zařízení, tak stačí v pravém seznamu vybrat požadovaný protokol a kliknout na přiřadit. Více informací na obrázku 4.1.
- **Jaký je účel položky *Zakázky* v hlavním menu?** Při vybrání této možnosti se zobrazí seznam otevřených obchodních zakázek, kde předmětem zakázky je konkrétní zařízení.

Jak už bylo řečeno, tak pro dotazy a pro získání bližších informací o programu jsem dostal kontakt na pracovníka, který mi ukázal program Mocev a zodpověděl mi výše uvedené otázky. Hned na začátku jsem si všiml, že chvíli trvá, než se program spustí, protože je umístěný na vzdáleném síťovém disku, ke kterému se připojuje přes VPN.[1] Během pozorování už nenastaly komplikace. Účastník v programu dělá už delší dobu a ukázka mých dotazů mu nedělala problém. Výsledky jsem zpracoval v sekci 4.5 do diagramů případů užití a v sekci 4.1 do person.

4.3 Program Mocev

V této kapitole přiblížím program Mocev. Je to software vytvořený pro počítače s operačním systémem Windows XP a novější. Není třeba ho instalovat, stačí ho pouze nakopírovat do počítače a nainstalovat potřebné knihovny. Největší problém je nalezení správné verze knihovny Visual FoxPro, která je nezbytná pro spuštění programu.

[1] Je to technologie pro vzdálený zabezpečený přístup. Více informací lze nalézt na: https://searchnetworking.techtarget.com/definition/virtual-private-network.



Obrázek 4.1: Přidávání nového protokolu v programu Mocev

Program Mocev pracuje s následujícími typy dat:

- protokol
- zařízení
- zakázka

Protokol je textový dokument ve formátu, který podporují pouze textové editory jako je Microsoft Word nebo LibreOffice Writer. Jedná se o dokumenty ve formátu `doc`, `docx` a `odt`. Protokol se v programu vyskytuje ve dvou verzích a to jako šablona protokolu nebo vyplněný protokol.

Zařízení se dělí do několika kategorií podle druhu, typu, velikosti a pracoviště, kde je umístěno. Zařízení konkrétního typu, velikosti a na konkrétním pracovišti má jednoznačný identifikátor, kterému se říká *projektové označení*. Ke každému typu zařízení se přiřazují šablony protokolů.

Obchodní případ je zakázka, jejíž předmětem je zařízení identifikované pomocí již zmíněného projektového označení. V programu je uložen seznam několika druhů protokolů a zařízení. V rámci každé zakázky se vytváří protokoly podle šablon, které jsou připojené k zařízení.

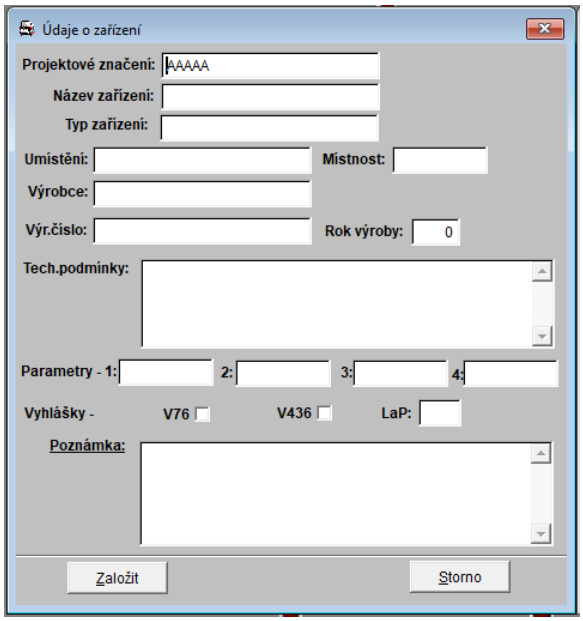
V následujících podsekcích popíšu funkce programu Mocev, které chtějí zaměstnanci i v novém systému a na závěr uvedu získané informace.

4.3.1 Správa zařízení

Po výběru typu zařízení se zobrazí nové okno, ve kterém je tabulka obsahující všechny založené zařízení. Tabulku lze seřadit nebo je možné vyhledávat ve sloupci *Projektové označení*. Řádky tabulky, tedy zařízení, lze mazat, editovat a vytvářet. Parametry, které je možné vyplnit při editaci nebo vytvoření nového zařízení, můžeme vyčíst z formuláře 4.2. Ve formuláři se nerozlišují povinné a volitelné parametry, tak proto jsem musel různě vyplnit

formulář, abych zjistil, jaké parametry jsou povinné a které ne. Zjistil jsem, že povinné parametry jsou:

- název zařízení,
- typ zařízení,
- umístění,
- výrobce,
- projektové označení.



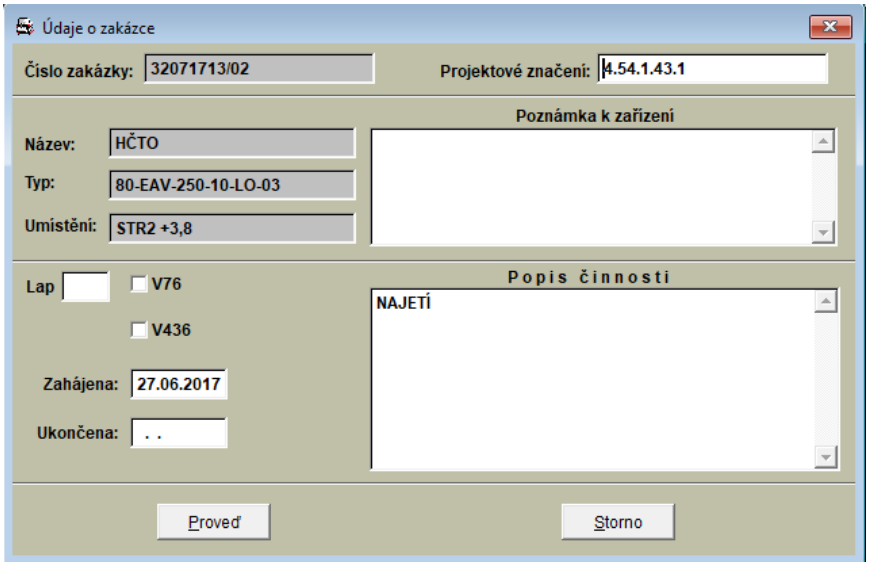
Obrázek 4.2: Formulář aplikace Mocev, který se používá při editaci a vytváření zařízení.

4.3.2 Správa zakázek

Při vybrání možnosti *Zakázky* v hlavním menu se zobrazí nové okno, ve kterém je tabulka obsahující všechny otevřené zakázky. S tabulkou lze provádět obdobné operace jako s tabulkou obsahující zařízení, tj. vyhledávání a řazení. Řádky tabulky, tedy zakázky, lze opět jako u *Zařízení* mazat, editovat a vytvářet. Pokud chceme vytvořit novou zakázku, tak nejdříve musíme vybrat druh zařízení a poté vytvořit novou zakázku. Parametry, které je možné vyplnit, můžeme vidět na obrázku 4.3. Opět bylo nutné zjistit, které parametry jsou povinné a volitelné. Mezi povinné parametry patří pouze *Číslo zakázky* a *Projektové označení*. Parametry (*Název*, *Typ*, *Umístění*) jsou převzaté na základě projektového označení. Ostatní parametry jsou volitelné.

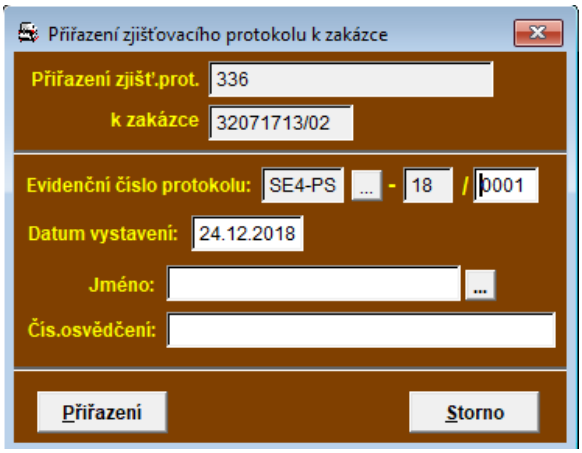
4.3.3 Přiřazení protokolu k zakázce

Při vybrání možnosti *Zakázky* v hlavním menu se zobrazí nové okno, ve kterém je tabulka obsahující všechny otevřené zakázky. Když zvolíme libovolnou zakázku a poté přejdeme



Obrázek 4.3: Parametry zakázky

na záložku *Zjišťovací protokoly*, tak se zobrazí dvě tabulky. Levá tabulka obsahuje hotové protokoly v rámci vybrané zakázky a pravá tabulka obsahuje protokoly, které jsou přiřazené k zařízení, které je předmětem vybrané zakázky. Když chceme přiřadit k zakázce protokol, tak ho vybereme na pravé straně a zvolíme přiřadit. Objeví se okno 4.4, ve kterém se vybírá volí autor, typ protokolu a jsou zde zobrazeny informace pro vložení do protokolu.



Obrázek 4.4: Přiřazení protokolu k zakázce

4.3.4 Archiv

Po výběru typu zařízení se zobrazí nové okno, ve kterém je tabulka obsahující zařízení. Pro zobrazení archivu se musí vybrat konkrétní zařízení, od kterého chceme zobrazit archiv. Poté můžeme změnit záložku na *Archiv* a zobrazí se seznam všech uzavřených zakázek, které byly prováděny na vybraném zařízení. Pokud vybereme možnost *Přepni do ukončených zakázek*, tak se zobrazí nové okno se všemi ukončenými zakázkami. Vybráním uzavřené zakázky

a přepnutím na záložku *Zjišťovací protokoly* tak můžeme procházet vytvořené protokoly v rámci zakázky.

4.3.5 Vyhodnocení programu Mocev

Program Mocev je starý přibližně 10 let. V té době nejspíše splňoval požadavky všech zaměstnanců, ale od té doby se změnilo několik věcí, zejména:

- Firma přijala **další zaměstnance** na nová pracoviště, kde potřebují také využívat tento program.
- Změnila se **síťová infrastruktura** lokální sítě. Lokální síťové disky na pracovištích, které sloužily zaměstnancům pro zálohování, se zrušily a pro zálohování se používají disky dostupné přes VPN na vzdáleném serveru.

Program nebyl navržen, aby posílal přes internet pouze některá data, a proto se práce v něm velmi zpomalila, protože se musí připojovat na vzdálený disk. Jakmile s programem pracovalo více zaměstnanců na stejném středisku současně, tak se práce ještě více zpomalí.

Zjištěné výhody a nevýhody:

- **Výhody:** funguje správa zařízení, zakázek a protokolů.
- **Nevýhody:**
  - některé funkce se nepoužívají,
  - chybí správa uživatelů,
  - nelze definovat plán kontrol,
  - zastaralé grafické rozhraní,
  - nelze přidat nové pracoviště,
  - lze filtrovat zařízení pouze pomocí druhu zařízení.

Uvedené výhody a nevýhody jsou zakomponovány v diagramu případu užití v kapitole 4.5. V programu jsou uloženy šablony protokolů a vypracované protokoly. Protokolem se rozumí textový dokument, který lze editovat ve známých textových editorech jako jsou MS Word a LibreOffice Writer. Analýzu nejpoužívanějších protokolů jsem provedl v kapitole 4.8.

4.4 Persony

Při analýze získaných informací z rozhovorů jsem zjistil, že uživatele lze roztřídit do dvou skupin. První skupinou jsou nadřízení pracovníci, kteří s programem Mocev nedělali a chtějí vidět přehledy naměřených dat. Druhou skupinou jsou pracovníci pracující na pozici *kontrolor kvality*, kteří pracovali s programem Mocev a potřebují hlavně pracovat s protokoly. Na obrázcích 4.5 a 4.6 jsou znázorněny vzniklé persony. Použil jsem kreslené postavy, aby nevznikl problém s GDPR.[2]

[2] Více informací lze nalézt na https://www.gdpr.cz/gdpr/

Břetislav Novák

Věk: *40 až 60 let*
Pozice: *Vedoucí pracovník*

Současné problémy:

- Musí procházet vytvořené protokoly, aby zjistil, v jakém stavu jsou zařízení

Motivace pro použití nové aplikace:

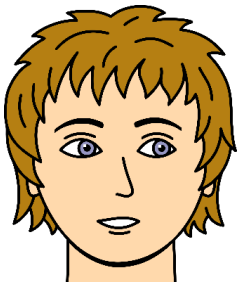
- Nastavovat a kontrolovat protokoly, které je třeba vypracovat
- Mít přehled nad stavy jednotlivých zařízení

Znalosti a zkušenosti:

- Počítačová gramotnost: *začátečník*
- Využití internetu: *často*
- Kancelářské nástroje: *základní úroveň*

Obrázek 4.5: Persona odpovídající skupině vedoucích pracovníků

Tereza Veselá



Věk: 25 *až 40 let*
Pozice: kontrolor kvality

Současné problémy:

- Neumožňuje práci více uživatelů zároveň
- Zastaralé rozhraní

Motivace pro použití nové aplikace:

- Uživatelsky přívětivější rozhraní
- Rychlejší práce se soubory

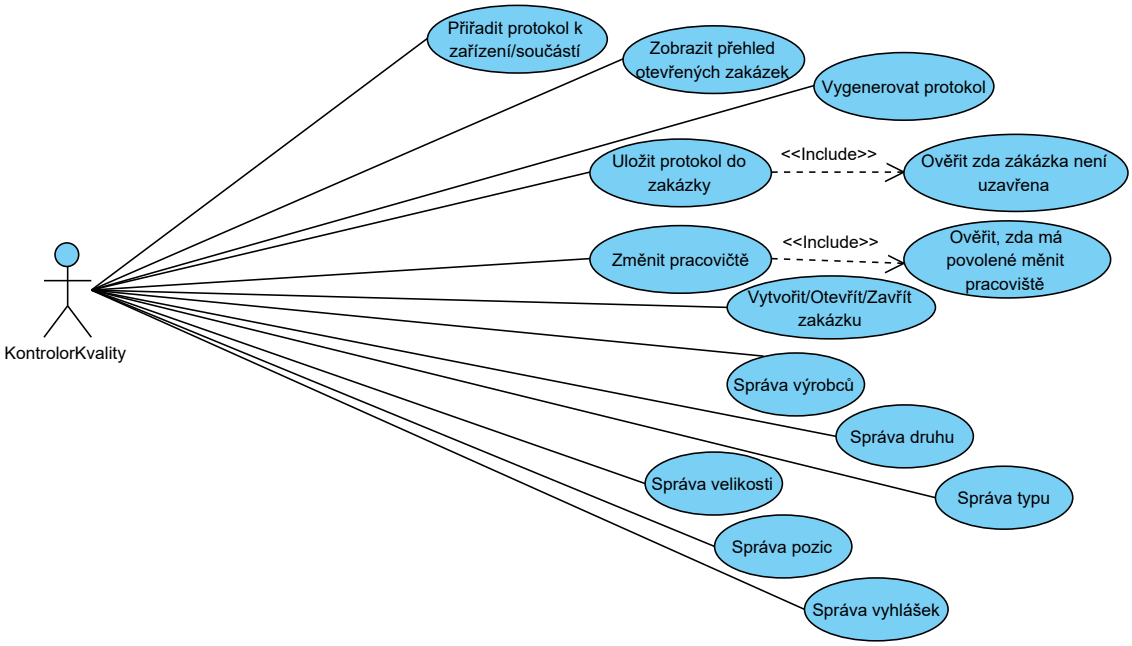
Znalosti a zkušenosti:

- Počítačová gramotnost: *mírně pokročilá*
- Využití internetu: *často*
- Kancelářské nástroje: *mírně pokročilá*

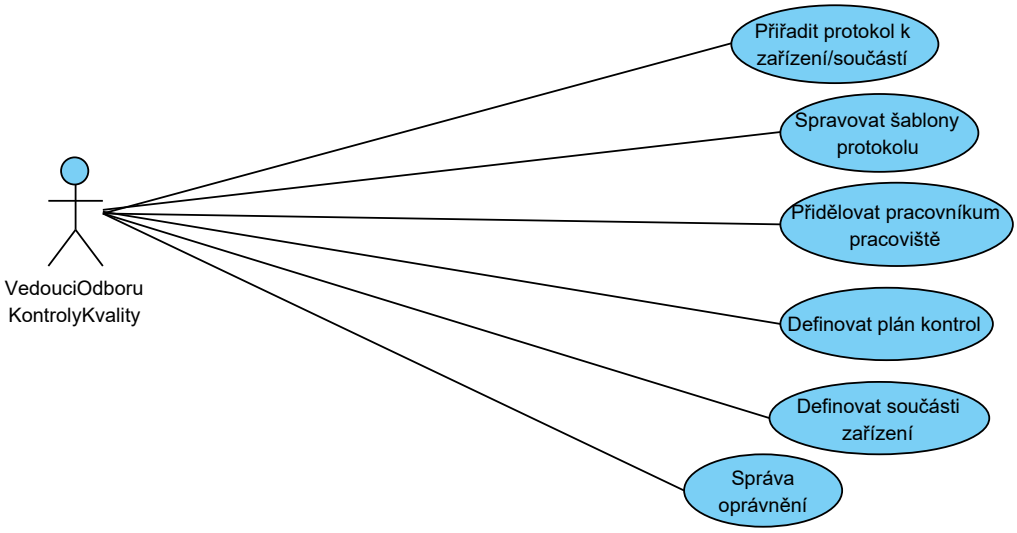
Obrázek 4.6: Persona odpovídající skupině pracovníků pracujících na pozici kontrolor kvality

4.5 Use case diagram

Z analýzy získaných informací od pracovníků pracujících na pozici *kontrolor kvality* a také na základě analýzy programu Mocev jsem mohl vytvořit diagram případů užití, které jsou znázorněny na obrázcích 4.7 a 4.8.



Obrázek 4.7: Diagram případu užití pro zaměstnance pracující na pozici kontrolor kvality



Obrázek 4.8: Diagram případu užití pro vedoucí pracovníky

Každý uživatel si může libovolně prohlížet protokoly, z toho vyplývá, že je může vyhledat a zobrazit si stav zařízení. Jelikož je požadováno, aby vedoucí pracovník (v diagramu *VedouciOdboruKontrolyKvality*) mohl nastavovat oprávnění pro kontrolory kvality, tak je nutné identifikovat uživatele. Z toho důvodu se musí každý uživatel v systému autentizovat.

Dále je třeba přidat dalšího aktéra, *Administrátora*, který bude mít oprávnění vytvářet nové uživatele. Výsledný diagram užití lze vidět v příloze A.

Oproti interakcím, které umožňuje program Mocev, jsem přidal správu pracovišť, možnost definovat plán kontrol a hlavně každý uživatel má svoji roli a od toho se odvíjí i jeho možná interakce se systémem. Program Mocev neobsahoval správu uživatelů, takže kdo měl přístup k programu, mohl v něm dělat jakékoliv úpravy.

4.6 Konceptuální návrh datového modelu

Firma si vede informace o zařízeních, která opravuje, o protokolech, které vypracovává a o zakázkách, které dostává od zadavatelů. Během zakázky jsou prováděny na konkrétním zařízení kontroly, o kterých se většinou vystavují protokoly. Protokoly, které se na daném zařízení vypracovávají, se určí podle toho, jaká šablona protokolu je přiřazena k typu zařízení. Protokoly se vystavují k různým druhům kontrol a podléhají určitým vyhláškám. Vypracované protokoly se přiřazují k zakázce, v rámci které se vytvářejí. Během zakázky se nemusí vypracovávat všechny protokoly, které jsou přiřazeny k typu. Po provedení všech požadovaných kontrol lze vytvořit plán kontrol a zkoušek, který obsahuje nejen kontroly, u kterých se vystavují protokoly, ale i kontroly, u kterých se protokol nevystavuje. Tyto kontroly musí probíhat v určitém pořadí. Do vypracovávaných dokumentů se uvádí zaměstnanec, který protokol vypracoval, datum vytvoření, název a typ protokolu.

Každé zařízení má určitý typ, výrobce a druh. Každý výrobce má vlastní označení typů. Nemůže být jeden typ od více výrobců. Zařízení se mohou vyskytovat v různých velikostech. Pokud je zařízení někde umístěno, na nějakém pracovišti, tak dostane přiřazené unikátní číslo, které se většinou skládá ze čtyřech alphanumerických skupin spojené tečkou. Zařízení se může myslet jedna součást nebo také zařízení složené z více částí.

4.7 Diagramy aktivit

V této části uvedu diagramy aktivit pro popis procesu mazaní (obrázek 4.10) a pro popis procesu přiřazení protokolu k zakázce (obrázek 4.9). Tyto protokoly popisují, jak se v programu Mocev provádí konkrétní operace a také je zde popsána možná optimalizace procesu.

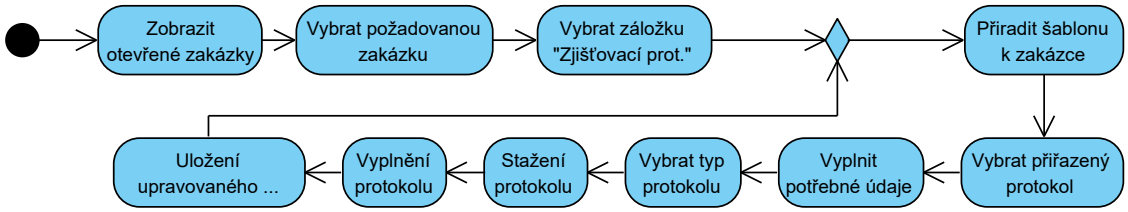
Proces přiřazení protokolu znázorněný na obrázku 4.9 lze v několika krocích optimalizovat:

1. **Vybrat záložku „Zjišťovací protokol“** lze odstranit, protože při zobrazení detailu zakázky se zobrazí seznam protokolů, které jsou už vyplněné a které nejsou.

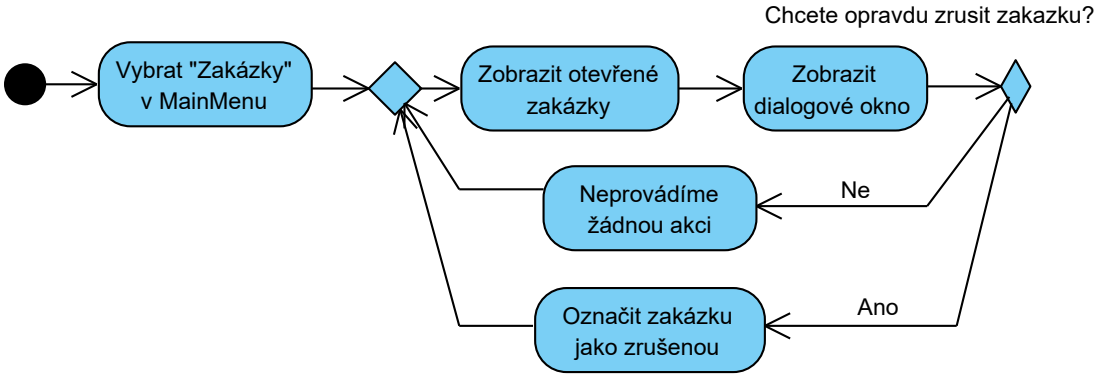
2. **Přiřadit šablonu k zakázce** lze odstranit, protože při vytváření nové zakázky se nadefinuje plán kontrol, který se musí provést v rámci zakázky a na základě toho se připojí i požadované šablony.

4.8 Analýza struktury protokolu

Každý protokol je reprezentován tabulkou popř. tabulkami. V záhlaví tabulky, popř. v první tabulce, jsou informace o zařízením, ke kterému se protokol vystavuje a o zakázce, v rámci které se protokol vytváří. Druhá tabulka nebo následující část obsahuje data, které se



Obrázek 4.9: Popis procesu, přiřazení protokolu k zakázce. Nejdříve se protokol musí správně vygenerovat a poté se mohou do protokolu vložit získaná data.



Obrázek 4.10: Popis proces „odstranění“ zakázky. Zakázka se ze systému nesmaže, ale pouze se označí že byla zrušená.

liší na základě typu protokolu. Poslední část nebo poslední tabulka obsahuje informace o vypracování (např. kdo a kdy protokol vypracoval, schválil). Ukázku celého protokolu lze vidět v příloze C.

Protokoly, u kterých se budou využívat naměřená data k monitorování stavu zařízení, obsahují obdobnou část tabulky jako na obrázku 4.11. Do této tabulky se doplňují naměřené údaje a rozmezí hodnot, mezi kterými se naměřené hodnoty mohou vyskytovat. Na základě naměřených a mezních hodnot se vyhodnocuje stav zařízení. Pokud jsou hodnoty v požadovaném rozsahu, tak se zařízení vyhodnocuje jako funkční, v opačném případě se zařízení musí vyměnit.

Protokoly jsou uloženy v textových dokumentech, které lze editovat ve známých textových editorech jako je MS Word nebo LibreOffice Writer. Z toho důvodu je třeba provést analýzu textových formátů, se kterými tyto editory pracují. Jelikož jsou dokumenty tvořeny pouze tabulkou a textem, tak je třeba se zaměřit hlavně na práci s tabulkami a textem. Více o používaných formátech lze nalézt v kapitole 5.

Vzájemné ustavení přírub		Poloha odchylky	Kriteria	Sací příruba			Výtlačná příruba		
				Měření	Δ, Ψ	Hod. *)	Měření	Δ, Ψ	Hod. *)
Rovnoběžnost Δ (mm)	$Y_{1\ (mm)}$	12	0,5mm						
	$Y_{2\ (mm)}$	6							
	$Y_{1\ (mm)}$	9	0,5mm						
	$Y_{2\ (mm)}$	3							
Přesazení Ψ (mm)	$X_{1\ (mm)}$	12	0,5mm						
	$X_{2\ (mm)}$	6							
	$X_{1\ (mm)}$	9	0,5mm						
	$X_{2\ (mm)}$	3							

Obrázek 4.11: Část protokolu, do které se doplňují naměřené hodnoty. Levá část obsahuje mezní rozměry, mezi kterými se naměřené hodnoty mohou pohybovat. Pravá část je prázdná a doplňují se do ní naměřené hodnoty.

Kapitola 5

Formáty textových dokumentů

Na základě požadavků specifikovaných v kapitole 4.8 je potřeba pracovat s textovými soubory, které používají známé textové editory jako je od Microsoft Word nebo od LibreOffice Writer. Tyto soubory jsou ve formátech `doc`, `docx` nebo `odt`.

V této kapitole popíšu výše zmíněné formáty. Hlavně se zaměřím na to, jak se v jednotlivých formátech pracuje s textem a tabulkami. Nejdříve popíšu formát DOC a poté DOCX, který je nástupcem formátu DOC a na konec popíšu formát ODT.

Formát DOC vytvořila společnost Microdoft Corporation (dále jen MS) jako výchozí formát pro jejich aplikaci Word 97. Tento formát byl používán do verze MS Word 2007 a od verze MS Word 2007 se stal výchozím formátem DOCX, který je výchozím formátem až doposud. Tyto formáty jsou podporovány i v jiných textových editorech například Writer od LibreOffice. Formát ODT je otevřený souborový formát pro textové editory, který je popsán standardem ODF. Je podporován výše uvedenými editory.

5.1 DOC

Formát DOC nebo doc je zkratka vzniklá z anglického slova „document“. Je to přípona názvu souboru pro dokumenty pro zpracování textu, které mohou obsahovat text, obrázky, tabulky, vlastní XML[1] a informace o rozvržení stránky.

Dokument [7] v tomto formátu je binární soubor složený z úložišť a dalších binárních toků. Umožňuje přenos a sdílení informací mezi aplikacemi vložením souboru nebo části souboru do složeného dokumentu.

5.1.1 Struktura dokumentu

Hlavní tok souboru ve formátu DOC začíná informačním blokem FIB (je zkratka z angličtiny *File Information Block*), který specifikuje lokace všech dat v souboru. Lokace jsou definovány jako dvojice čísel, kde první znamená lokaci a druhé velikost. Lokace má prefix „fc“ a velikost má prefix „lcb“.

Pro lepší pochopení, uvedu všechna jména proměnných stejně jako v dokumentaci a pokud nebude uvedeno jinak, tak jsou to jména atributů ve strktuře *FIB*.

Jak už bylo řečeno, soubory ve formátu DOC jsou složené soubory a mohou se skládat z následujících toků a úložišť:

[1] XML je nástroj pro ukládání a transport dat. Více informací na: https://www.w3schools.com/xml/xml_whatis.asp

- **WordDocument Stream**: Je to povinná součást dokumentu. Musí na začátku, tj. na indexu 0, obsahovat strukturu FIB. Velikost nesmí být větší než 2^{31} bajtů.
- **Table Stream**: Je to povinná součást dokumentu. Tato hodnota je boolovská, která rozlišuje, který datový tok se použije. Pokud není soubor šifrován, tak nemá tento tok definovanou strukturu. V opačném případě obsahuje data, na které je odkazováno z FIB nebo z jiných částí souborů. Velikost nesmí být větší než 2^{31} bajtů.
- **Data Stream**: Nemá předdefinovanou strukturu. Obsahuje data, na která se odkazuje z jiných části souboru. Pokud nejsou odkazy do této části, tak nemusí být tato část v souboru přítomna. Velikost nesmí být větší než 2^{31} bajtů.
- **ObjectPool Storage**: Je to úložiště pro OLE objekty[2]. Toto úložiště nemusí být přítomné v dokumentu, pokud dokument neobsahuje vložené OLE objekty.
- **Custom XML Data Storage**: Je to volitelné úložiště, které se musí jmenovat „MsoDataStore“. Toto úložiště určuje jak ukládat fragmenty XML [6].
- **Summary Information Stream**: Je to volitelná část, která musí být pojmenována „
005SummaryInformation“, kde „
005“ je znak s hexadecimální hodnotou 0x0005. V této sekci se definují jednoduché OLE vlastnosti [6].
- **Document Summary Information Stream**: Je to volitelný tok, který se musí jmenovat „
005DocumentSummaryInformationStream“, kde „
005“ je znak s hexadecimální hodnotou 0x0005.
- **Encryption Strem**: Je to volitelný tok. Musí se jmenovat „encryption“. Tento tok nesmí být přítomen pokud jsou splněný následující podmínky:
  - Dokument je chráněn heslem pomocí šifrování CryptoAPI RC4
  - „EncryptionHeader.Flags“ je nastaven na hodnotu „fDocProps“
- **Macros Storage**: Je volitelné úložiště obsahující makra definovaná v souboru.
- **XML Signature Storage**: Volitelné úložiště pro digitální podpisy. Musí se jmenovat „_xmlsignatures“.
- **The Information Rights Management Data Space storage**: Je to volitelné úložiště, které se musí jmenovat „006DataSpaces“, kde „006“ je hexadecimální hodnotou 0x0006. Popisuje [6] několik definic datového prostoru používaných k vynucení práv, které byly aplikovány na dokument.

[2]OLE objekt je objekt podporující připojení a vkládání objektů (z angličtiny Object Linking and Embedding). OLE je technologie pro přenos sdílení informací mezi aplikacemi vložením souboru nebo části souboru do složeného dokumentu.

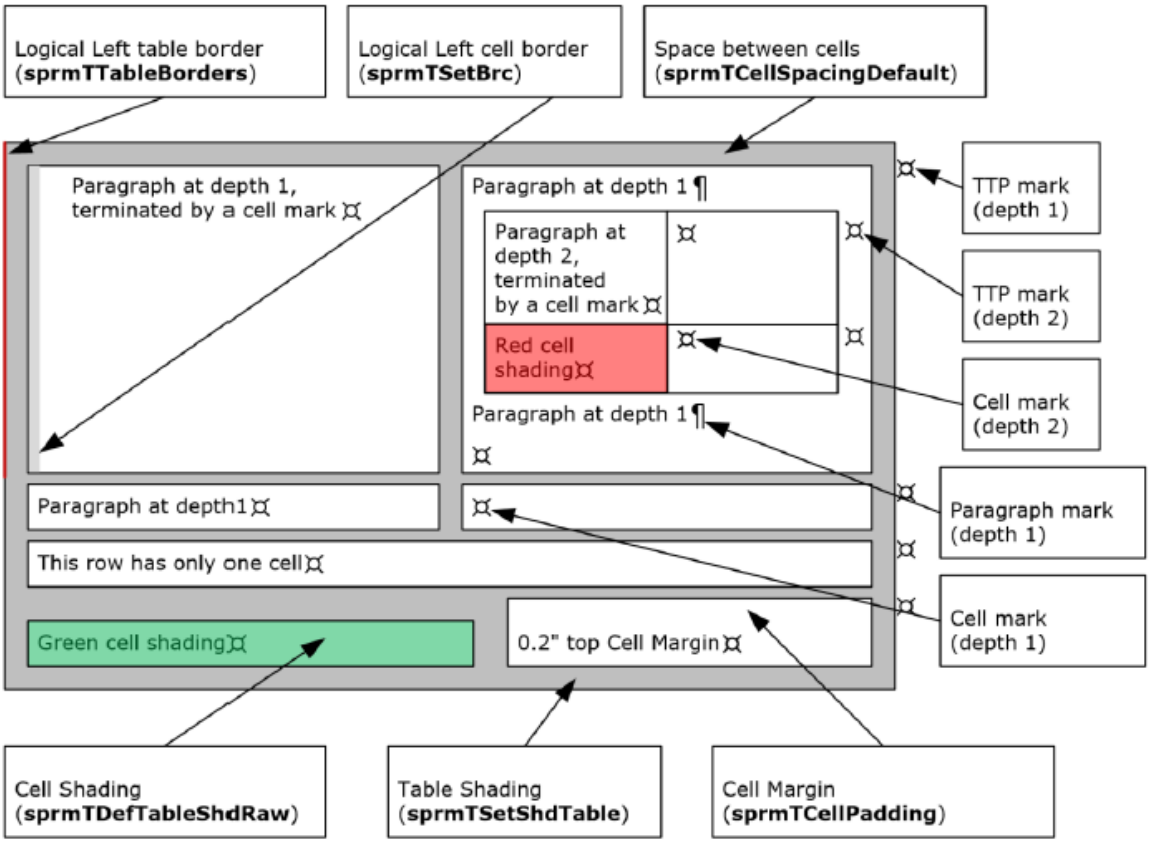
5.1.2 Pozice znaku

Pozice znaku nebo také CP (z anglického *character position*) je 32-bitové bez-znaménkové číslo. Pod znakem si můžeme představit znak textu dokumentu, kotvu pro objekt, jako jsou poznámky pod čarou, textová pole, značky odstavců a buněk tabulky.

Velikost jednotlivých znaků v souboru se liší. Umístění a velikost každého znaku v souboru lze vypočítat pomocí algoritmu uvedeného v sekci 5.1.4.

5.1.3 Práce s tabulkou

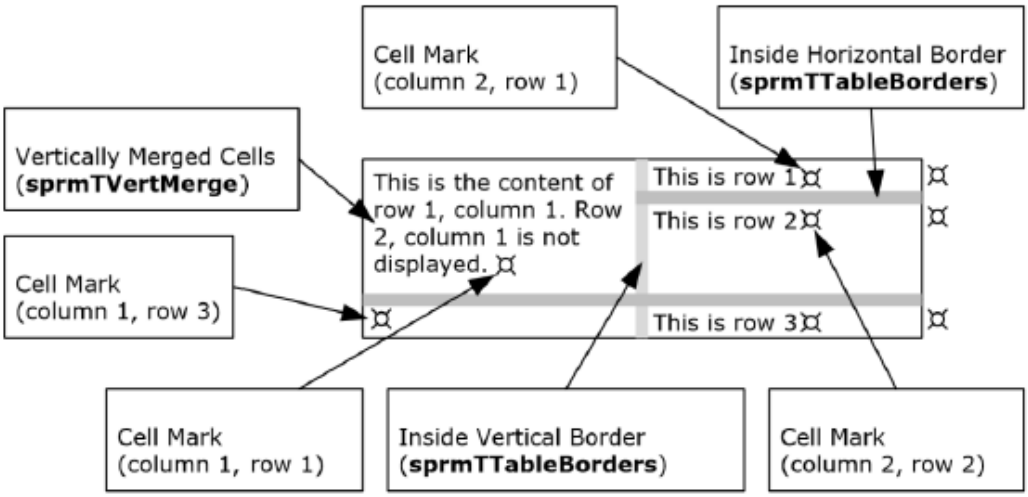
Tabulka [7] se skládá z jednoho či více odstavců stejné hloubky a také může obsahovat jednu či více vnořených tabulek, jejichž hloubka je o jednu větší než tabulka obsahující buňku. Odstavce, které nejsou v tabulce mají hloubku 0. Poslední odstavec v tabulce je ukončen znakem buňky. Pokud je hloubka tabulky rovna 1, tak značka buňky má hodnotu *0x0007*. Pokud je větší než 1, tak značka buňky má hodnotu *0x000D*. Na obrázku 5.1 lze vidět ukázku zanoření tabulek a také jsou tam uvedeny některé vlastnosti, které můžeme ovlivnit u tabulek.



Obrázek 5.1: Schéma zobrazující tabulku a do ní vnořenou tabulku. Jsou zde znázorněny značky konce řádků a další vlastnosti, které můžeme nastavit u tabulek. Převzato z [7].

Řádky tabulky obsahují 1 až 63 buněk, z nichž každá má stejnou hloubku a poté následuje značka konce řádku, která je také ve stejné hloubce jako tabulka. Buňky tabulky můžeme slučovat vertikálně nebo horizontálně. Vertikální slučování probíhá tak, že se vy-

tvoří jedna buňka, která patří do více řádků. Obdobně to probíhá i při horizontálním slučováním. Ukázku vertikálního slučování můžeme vidět na obrázku 5.2.



Obrázek 5.2: Schéma zobrazující tabulku s vertikálně sloučenými buňkami. Převzato z [7].

Tabulku lze zpracovat podle následujícího algoritmu, který se používá pro nalezení hranice nejvnitřejší buňky, ve které se nachází pozice CP nebo pro určení, zda pozice je či není v buňce tabulky.

```
var depth = ...; // hloubka tabulky
if (depth == 0) throw "Odstavec neni v~bunce tabulky";
if (znak na pozici CP neni znakem konce radku) throw ...;
if (depth == 1) throw "CP neni v~bunce tabulky";

function lastChar(CP, depth) {
    var cpLast = ...; // pozice posledniho znaku v~odstavci obsahujici CP
    var props = ...; // vlastnosti odstavce obahujici cpLast
    if (props.depth < depth)
        throw "Hloubka tabulky v~props je mensi nez ziskana hloubka podle CP"
    else if (props.depth == depth) {
        var char = ...; // ziskame znak na pozici cpLast
        if (char je znacka bunky) return cpLast
    }
    return lastChar(cpLast + 1, depth);
}
function firstChar(CP, depth) {
    var cpFirst = ...; // pozice prvniho znaku v~odstavci obsahujici CP
    if (cpFirst == 0) return cpFirst;
    else if (cpFirst <= 0) throw "Zaporna pozice znaku"
    var cpPrev = cpFirst - 1;
    var props = ...; // vlastnosti odstavce obsahujici cpPrev
    var depthPrev = props.depth; // hloubka tabulky predchazejiciho znaku
    if (depthPrev < depth) return cpFirst;
    else if (depthPrev == depth) {
        var char ...; // ziskame znak na pozici cpPrev
        if (char je znacka bunky nebo konce radku) return cpFirst;
    }
    return firstChar(cpPrev, depth);
}
```

Hloubka tabulky na řádku 1 se získá z vlastností získaných algoritmem pro nalezení vlastností odstavce. Na řádcích 2-4 je ověření, zda znak na pozici CP je v buňce a v hloubce *depth*. Pozici posledního znaku na řádku 7 nebo prvního znaku odstavce na řádku 18 získáme pomocí algoritmu pro nalezení hranic odstavce. Pro získání vlastností odstavce na řádcích 8 a 22 použijeme algoritmus pro nalezení vlastností odstavce. Funkce na řádcích 6-17 slouží pro získání pozice posledního znaku. Funkce na řádcích 17-25 slouží pro nalezení pozice prvního znaku buňky. Zmíněné algoritmy lze nalézt v dokumentaci [7].

5.1.4 Algoritmus pro získání textu

Algoritmus [7] určuje, jak najít text na konkrétní pozici CP.

```
1  var offset = FIB.FibRgFcLcb97.fcClx;
2  var clx = file[offset]; // file je zpracovavany soubor
3  clx.Pcdt.PlcPcd.aCp.forEach(function(elem, index) {
4      if(elem je posledni v~aCp) throw "Neplatny znak na pozici CP" ;
5      else if (index <= CP and x neni nejvetsi)
6          continue;
7      var fCompressed = PlcPcd.aPcd[index].fc;
8      if (fCompressed.fCompressed == 0)
9          return {
10             size: 16b Unicode,
11             offset: fCompressed.fc + 2(cp - elem)
12         }
13     else return {
14         size: 8b ASCII,
15         offset: (elem.fc.fc/2) + (cp - elem)
16     }
17 });
```

Na 3. řádku procházíme kolekci *aCp*. Na řádcích 9-12 a 13-16 jsou hodnoty které algoritmus vrací. Výsledkem algoritmu je velikost znaku a pozice (offset), na kterém se znak v souboru nachází. Pokud element x není největším možným elementem v kolekci *aCp*, ale menší nebo roven CP, tak algoritmus pokračuje dalším elementem kolekce.

5.2 DOCX

Formát DOCX je výchozí formát pro Word 2007 a novější. Je to zip archiv XML souborů a je popsán standardem Office Open XML. Tento standard specifikuje souborové formáty pro ukládání dokumentů kancelářských balíků (například textové, tabulkové dokumenty). Formát vytvořila společnost Microsoft. Poté byl standardizován sdružením ECMA[3] [11].

Soubory popsané standardem Office Open XML jsou reprezentovány jako série souvisejících částí, které jsou uloženy v kontejneru nazvaném balík [11]. Balíkem se rozumí ZIP archiv. V následujících podkapitolách popíšeme hlavní strukturu textových dokumentů a poté se zaměříme na, to jak je uložený text a tabulky.

5.2.1 Struktura dokumentu

Jak už bylo řečeno, soubor popsaný standardem Office Open XML je série souvisejících částí. Některé části se odvíjí od toho, jakého je soubor formátu nebo jinak řečeno o jaký

[3] Je to mezinárodní organizace pro normalizaci informačních a komunikačních systémů. Více na http://www.ecma-international.org/memento/Ecmabylaws.htm

typ balíku se jedná. Některé části mají tyto formáty společné [11]. Jsou to části *DrawingML*, *Math*, *Bibliography* a *File Properties*.

File Properites je část, kde jsou popsány vlastnosti souboru, bez ohledu na to, o jaký typ formátu jde. Jedná se o vlastnosti:

- Jméno autora
- Datum vytvoření
- Název
- Popis

Ostatní vlastnosti jsou specifické pro konkrétní balík. Například pro balík WordprocessingML[4], který je pro textové typy dokumentů, jsou to následující vlastnosti:

- Počet znaků
- Počet slov
- Počet odstavců
- Počet stránek

Tyto vlastnosti si můžeme specifikovat sami. Například můžeme přidat vlastnost *Jméno klienta*, pro kterého byl dokument vytvořen, *Datum a čas* kdy došlo k nějaké události. Každá vlastní vlastnost má hodnotu a datový typ hodnoty.

Standard Office Open XML popisuje několik druhů balíku. Jedná se o následující balíky:

- WordprocessingML - pro textové editory
- PresentaionML - pro prezentace
- SpreadsheetML - pro tabulkové editory

V následujících podkapitolách se budeme věnovat pouze balíku pro textové editory, tedy WordprossingML. Tento balík má definovanou strukturu souborů, kterou můžeme vidět na obrázku v příloze B.

Většina dat se nachází v hlavním dokumentu (soubor */word/document.xml*). Kořenovým elementem je element s názvem *document* a ten obsahuje element s názvem *body*. Element *body* může obsahovat různé typy elementů, ale my se zaměříme na to, jak jsou tam uloženy texty a jak tabulky, ale o tem se dozvíme v sekcích 5.2.3 a 5.2.4.

5.2.2 Odkazování

Balíky popsané standardem Office Open XML jsou série souvisejících částí. Tyto části jsou mezi sebou provázané pomocí odkazů [11]. Můžeme se odkazovat na vnitřní soubory soubory nebo externí. Odkazování můžeme vidět na obrázku 5.3.

Vtahy mohou být implicitní nebo explicitní. Rozdíl mezi nimi je takový, že *ID* v implicitním vztahu odkazuje na prvek se stejným *ID* v cílové části, ale při explicitním odkazování *ID* odkazuje na vztah se stejným *ID*.

[4]Znaky „ML“ znamenají, že se jedná o značkovací jazyk (z angličtiny *Markup Language*).

```
<Relationships ...>
    <Relationship Id="rId5" Type=".../footnotes"
        Target="footnotes.xml"/>
    <Relationship Id="rId7" Type="../hyperlink"
        Target="http: //www.ecma-international.org/" TargetMode="External"/>
</Relationships>
```

Obrázek 5.3: Ukázka odkazů (vztahů). Jsou vidět dvě ukázky, kde první se odkazuje na vnitřní soubor „footnotes.xml“ a druhá se odkazuje na internetovou stránku.

5.2.3 Text

Text bývá uložený v odstavcích, které reprezentuje element *p*. Veškeré formátování odstavce je uloženo v elementu *pPr*, který obsahuje element *rPr*. Odstavec také obsahuje elementy *r*. Tento element obsahuje element *t*, který obsahuje text. Pokud text je jinak formátován než odstavec, tak element *r* obsahuje také element *rPr*, který obsahuje formátování textu, ale obsahuje pouze ty vlastnosti, ve kterých se liší od stylů odstavce. Více můžeme vidět na ukázce 5.4.

```
<w:p>
    <w:pPr>
        <w:rPr>
            <w:i/>
            <w:u w:val="single"/>
        </w:rPr>
    </w:pPr>
    <w:r>
        <w:t xml:space="preserve">Ahoj </w:t>
    </w:r>
    <w:r>
        <w:rPr>
            <w:b/>
        </w:rPr>
        <w:t>svete!!!</w:t>
    </w:r>
</w:p>
```

Obrázek 5.4: Ukázka XML pro vytvoření jednoduchého odstavce s textem „Ahoj **svete!!!**“.

5.2.4 Tabulka

Tabulka je sada odstavců uspořádaných v řádcích a sloupcích. Tabulky v balíku WordprocessML jsou definovány elementem *tbl* [11]. Element *tlb* může obsahovat například:

- elementy *tr* reprezentující řádky
- element *tlbGrid*, který obsahuje nastavení pro řádky a sloupce (například jejich rozměry)
- element *tlbPr*, který obsahuje styly

Na obrázku 5.5 můžeme vidět ukázku XML, které vytvoří prázdnou tabulku s dvěma stejně širokými sloupci.

```
<w:tbl>
    <w:tblPr>
        <w:tblstyle w:val="Mkatabulky"/>
        <w:tblw w:w="0" w:type="auto"/>
        <w:tblLook w:val="04A0" w:firstRow="1" w:lastRow="6" w:firstColumn="1"
            w:lastColumn="0" w:noHBand="0" w:noVBand="1"/>
    </w:tblPr>
    <w:tblGrid>
        <w:gridCol w:w="4531"/>
        <w:gridCol w:w="4531"/>
    </w:tblGrid>
    <w:tr w:rsidR="00483C75" w:rsidTr="00483C75">
        <w:tc>
            <w:tcPr><w:tcW w:w="4531" w:type="dxa"/></w:tcPr>
            <w:p w:rsidR="00483C75" w:rsidRDefault="00483C75"></w:p>
        </w:tc>
        <w:tc>
            <w:tcPr><w:tcW w:w="4531" w:type="dxa"/></w:tcPr>
            <w:p w:rsidR="00483C75" w:rsidRDefault="00483C75"/>
        </w:tc>
    </w:tr>
</w:tbl>
```

Obrázek 5.5: Ukázka XML pro vytvoření prázdné tabulky se stejné širokými sloupci.

5.3 ODT

Formát ODT je otevřený souborový formát určený pro ukládání a výměnu dokumentů vytvořených kancelářskými textovými editory. Je popsán standardem ODF, (zkratka z anglického *Open Document Format*, celým názvem *Open Document Format for Office Application*) [21]. ODF je otevřený souborový formát založený na XML souborech. Tento standart Standard ODF byl vytvořen společností OASIS (zkratka z anglického *Organization for the Advacement of Structured Information Standards*). ODF popisuje formáty pro tabulky, grafy, prezentace a textové dokumenty [21].

V následujících podkapitolách popíšeme strukturu dokumentu a jak se pracuje s tabulkami a texty.

5.3.1 Požadavky na strukturu textového dokumentu

Pokud je soubor ve formátu ODT, tak musí splňovat požadavky, které platí pro všechny dokumenty popsané standardem ODF a také musí splňovat požadavky na textový dokument [21]. Nejprve si popíšeme obecné požadavky:

- Balík musí obsahovat nejméně jeden ze souborů *content.xml* a *styles.xml*. Může obsahovat volitelné soubory *settings.xml* a *meta.xml*. Sturčný popis obsahů souborů:
  - content.xml – obsahuje text dokumentu
  - styles.xml – obsahuje styly používané v dokumentu
  - meta.xml – obsahuje data jako například počet znaků v souboru
  - settings.xml – nastavení dokumentu jako například ochrana formulářů
- Všechny soubory typu *xml* musí být ve formátu validního XML 1.0[5].

[5] Více informací o této verzi jsou dostupné na https://www.w3.org/TR/xml/

- Kořenové elementy souboru jsou pojmenovány podle jména souboru, ve kterém se nachází. Například soubor *styles.xml* musí obsahovat element s názvem *office:document-styles*.
- Pokud kořenový element je *math:math*, tak XML soubor musí být validní s ohledem na schéma Math 2.0[6].
- Pokud dokument bude pouze jeden XML soubor, pak kořenový element se musí jmenovat *office:document*.

To byly požadavky na všechny soubory popsané standardem ODF bez ohledu na to, jakého je soubor formátu. Nyní si uvedeme požadavky na textové soubory popsané standardem ODF:

- Aby soubor byl validní textový dokument ve formátu ODT, tak hodnota atributu *office:mimetype* v elementu *office:document* musí být *application/vnd.oasis.opendocument.text*.
- Element *office:body* musí obsahovat element *office.text*

5.3.2 Text

Text je reprezentován formou odstavců. Mezi prvky odstavce patří nadpis, který je reprezentován elementem *text:h* a tělo odstavce, které je reprezentováno elementem *text:p*.

Element *text:h* představuje záhlaví dokumentu. Nadpisy definují strukturu dokumentu například kapitoly nebo sekce začínají nadpisem a rozšiřují se úrovněmi. Element může mít například následují atributy: *xml:id*, *text:style-name* nebo *text:is-list-header*. Může mít vnořené elementy jako například *text:hidden-text*, *text:chapter* nebo *draw:line*.

Element *text:p* představuje odstavce, které reprezentují základní text v dokumentu. Tento element může obsahovat prostý text nebo může obsahovat další vnořené elementy jako například *text-hidden-paragraph*, *text:bookmark-ref* nebo *draw:control*. Na obrázku 5.6 můžeme vidět ukázku, která je XML reprezentací textu „Ahoj **svete!!!**“.

5.3.3 Tabulka

Reprezentace tabulek je založena konceptu mřížky, která je tvořena řádky a sloupci. Při výběru řádku, se vyberou všechny buňky na řádku. Pokud vybereme sloupec, tak se označí všechny buňky se stejnou pozici uvnitř řádků. Tabulky mohou být i vnořené uvnitř buňky. Tabulka je tvořena následujícími elementy:

- **table:table**: jméno kořenového elementu tabulky
- **table:table-row**: jméno elementu, který reprezentuje řádek tabulky
- **table:table-cell**: jméno elementu, který reprezentuje buňku tabulky
- **table:covered-table-cell**: jméno elementu, který reprezentuje buňku, která je rozprostřena přes více řádku nebo sloupců
- **table:table-header-rows**: jméno elementu, který reprezentuje řádek hlavičky

Na obrázku 5.7 můžeme vidět ukázku jednoduché tabulky, která obsahuje výše uvedené elementy.

[6] Je to značkovací jazyk pro popis matematické notace a zachycení její struktury i obsahu. Více informací lze najít na https://www.w3.org/TR/2003/REC-MathML2-20031021/

```
<office:font-face-decls>
    <style:font-face style:name="Calibri Light" svg:font-family="Calibri Light".../>
</office:font-face-decls>
<office:automatic-styles>
    <style:style style:name="P1" style:parent-style-name="Normalni"
    style:family="paragraph">
        <style:paragraph-properties fo:break-before="page"/>
    </style:style>
    <style:style style:name="T2" style:parent-style-name="Standardnipismoodstavce"
            style: family="text">
        <style:text-properties fo:font-weight="bold" style: font-weight-asian="bold"/>
    </style:style>
</office:automatic-styles>
<office:body>
    <office:text text:use-soft-page-breaks="true">
        <text:p text:style-name="P1">
            Ahoj <text:s/><text:span text: style-name="T2">svete!!!</text:span>
        </text:p>
    </office:text>
</office:body>
```

Obrázek 5.6: Ukázka XML, která reprezentuje text „Ahoj **svete!!!**“. V elementu *office:font-face-decls* jsou definovány písma v rámci dokumentu. V elementu *office:automatic-styles* jsou uloženy vlastnosti formátování, které jsou považovány za vlastnosti objektu, ke kterému jsou vlastnosti přiřazeny. V elementu *office:body* obsahuje tělo dokumentu. Element *text:s* reprezentuje mezeru (ASCII 0x20). Element *text:span* se používá pro aplikování stylu na část textu. Obsahem tohoto elementu je text, který používá tento styl textu.

```
<table:table table:style-name="Table1">
    <table:table-columns>
        <table:table-column table:style-name="TableColumn2"/>
        <table:table-column table:style-name="TableColumn3"/>
        <table:table-column table:style-name="TableColumn4"/>
    </table:table-columns>
    <table:table-row table:style-name="TableRowS">
        <table: table-cell table:style-name="TableCell6" table:number-columns-spanned="2">
            <text:p text:style-name="P7"/>
        </table:table-cell>
        <table:covered-table-cell/>
        <table:table-cell table:style-name="TableCell8" table:number-rows-spanned="2">
            <text:p text:style-name="P9"/>
        </table:table-cell>
    </table:table-row>
    <table:table-row table:style-name="TableRow10">
        <table:table-cell table:style-name="TableCell19">
            <text:p text:style-name="P10"/>
        </table:table-cell>
        <table:table-cell table:style-name="TableCell11">
            <text:p text:style-name="P12"/>
        </table:table-cell>
        <table:covered-table-cell><text:p text:style-name="P13"/></table:covered-table-cell>
    </table:table -row>
</table:table>
```

Obrázek 5.7: Ukázka XML, která vytvoří tabulku se třemi sloupci a dvěma řádky. Jedna buňka je spojena vertikálně a druhá horizontálně.

5.4 Knihovny pro práci s formáty

Pro práci s výše uvedenými formáty existuje několik knihoven, které umožňují číst nebo vytvářet soubory v daném formátu. Většinou tyto knihovny nepodporují všechny formáty nebo pouze částečně. V následujících podkapitolách si uvedeme knihovny pro práci s výše uvedenými formáty.

5.4.1 PHPOffice/PHPWord

Je to knihovna napsaná v čistém PHP, která poskytuje funkce pro čtení a zapisování do textových dokumentů [22]. V současné době je vydána verze **0.15.0**, která částečně podporuje následující formáty:

- Office Open XML (docx)
- Open Document Format (odt)
- Rich Text Format (rtf)[7]
- HTML
- PDF

Knihovna poskytuje několik tříd, které umožňují pracovat s výše uvedenými formáty. Knihovnu lze jednoduše rozšířit, stačí pouze vytvořit novou třídu, která implementuje rozhraní a bude dědit od abstraktní třídy. Pro čtení souboru se jedná o rozhraní *ReaderInterface* a abstraktní třídu *AbstractReader*. Pro zapisování do souboru se jedná o rozhraní *WriterInterface* a abstraktní třídu *AbstractWriter*. Knihovna pro svojí funkčnost potřebuje: Na obrázku 5.8 můžeme vidět jak se pomocí této knihovny vytvoří dokument s obsahem „Ahoj **svete!!!**“.

```
$phpWord = new \PhpOffice\PhpWord\PhpwWord();
$section = $phpWord->addSection();
$section->addText("Ahoj ");
$section->addText("svete!!!", ['bold' => true]);
$objwriter = \PhpOffice\Phpword\IOFactory::createWriter($phpWord, "word2007");
$objwriter->save('result.docx');
```

Obrázek 5.8: Ukázka PHP kódu, která za pomocí knihovny PHPWord vytvoří dokument ve formátu DOCX, který bude obsahovat text „Ahoj **svete!!!**“

5.4.2 dolanmiu/docx

Knihovna napsaná v jazyce JavaScript[8]. Knihovna nepotřebuje speciální balíčky pouze internetový prohlížeč. Podporuje pouze formát docx. Na obrázku 5.9 můžeme vidět, jak se vytvoří dokument ve formátu docx s obsahem „Ahoj **svete!!!**“.

[7] Více informací o tomto formátu lze nalézt na https://www.microsoft.com/en-us/download/details.aspx?id=7105.

[8] Více informací o této knihovně lze nalézt na https://docx.js.org/

```
import * as fs from "fs";
import { Document, Packer } from "../build";

const doc = new Document({});
const para = doc.createParagraph();
para.createTextRun("Ahoj ");
para.createTextRun("svete!!!").bold();

const packer = new Packer();
packer. toBuffer(doc).then((buffer) => {
    fs.writeFileSync("My Document.doc", buffer);
});
```

Obrázek 5.9: Ukázka JS kódu, která za pomocí knihovny dolanmiu/docx vytvoří dokument ve formátu DOCX, který bude obsahovat text „Ahoj **svete!!!**“

5.4.3 Apache POI

Knihovna napsaná v jazyce Java pro čtení a zápis do souborů ve formátu pro MS Office [3]. Takže podporuje všechny formáty popsané standardem OOXML a binární formáty, které byly nahrazeny OOXML formáty. Nepodporuje formáty standardu ODF. Pro svoji funkčnost potřebuje nainstalovanou podporu pro jazyk Java SDK 1.7+[9]. Pro práci s formátem DOC použijeme komponentu *HWPF* a pro práci s DOCX komponentu *XWPF*, které jsou obsaženy v této knihovně. Na obrázku 5.10 můžeme vidět, jak se vytvoří dokument ve formátu docx s obsahem „Ahoj **svete!!!**“.

```
XWPFDocument document = new XWPFDocument();
FileOutputStream out = new FileOutputStream(new File("result.docx"));

XWPFParagraph paragraph = document.createParagraph();
XWPFRun run = paragraph.createRun();
run.setText("Ahoj ");
XWPFRun runBold = paragraph.createRun();
runBold.setText("svete!!!");
runBold.setBold(true) ;

document .write (out);
out.close();
```

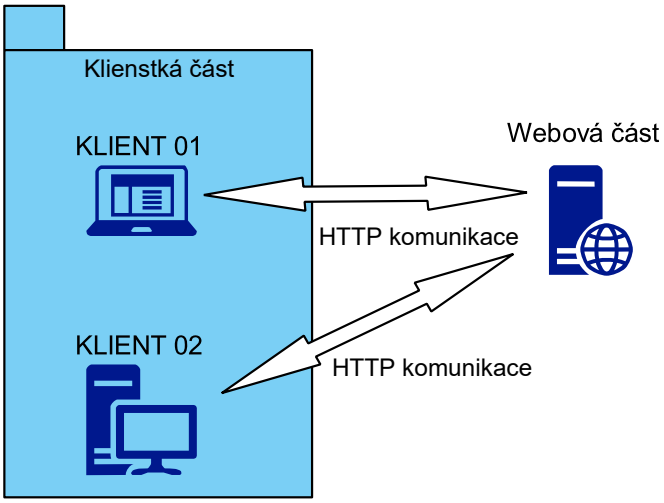
Obrázek 5.10: Ukázka kódu v jazyce JavaScript, která za pomocí knihovny Apache POI a komponenty XWPF vytvoří dokument ve formátu DOCX, který bude obsahovat text „Ahoj **svete!!!**“

[9] Více informací jak tuto knihovnu nainstalovat lze nalézt na https://www.java.com

Kapitola 6

Návrh systému

Návrh systému pro správu kontrol kvality zařízení vychází z principů uvedených v kapitole 3.3 a z požadavků specifikovaných v kapitole 4. Z hlediska architektury je systém rozdělen do dvou částí, a to klientské a serverové. Schéma architektury a komunikace mezi těmito částmi je znázorněna na obrázku 6.1.



Obrázek 6.1: Schéma komunikace mezi klientem a serverem.

6.1 Návrh klientské části

V následujících podkapitolách popíšu vybrané drátěné modely a výsledný prototyp grafického rozhraní. Tyto metody byly popsány v kapitole 3.3.

6.1.1 Wireframe

Na začátku návrhu jsem vytvořil 5 různých drátěných modelů, které se odlišovaly hlavně v rozložení elementů. Návrhy jsem ukázal zúčastněným zaměstnancům a vysvětlil jim, co které elementy znamenají a co reprezentují. Zaměstnanci si poté vybrali nejvíce vyhovující model. Vybraný model prošel ještě dvěma modifikacemi, které se týkaly:

- provázanosti mezi různými pohledy,

- zobrazení stavu zařízen, kdy byl přidán postranní sloupec pro výběr typu prováděné kontroly.

Popisovaný model má finální rozložení elementů, pouze v malých detailech se může lišit od prototypu.

Na obrázku D.1 je znázorněna základní obrazovka, která se objeví ihned po přihlášení. Tato obrazovka jako jediná zobrazuje i horní nabídku a postranní menu. Ostatní pohledy by zobrazovaly vždy stejné postranní menu a horní nabídku. Z tohoto pohledu se lze dostat na obrazovku zobrazujících detail otevřené zakázky (obrázek D.2), ale pokud je zakázka uzavřena, tak se zobrazí pohled znázorněný na obrázku D.3. Z každého pohledu lze přejít na obrazovku zobrazující detailní informace o zařízení, které je předmětem zakázky. Pohled na detail zařízení je znázorněn na obrázku D.4.

6.1.2 Prototyp

V této kapitole popíšu některé pohledy prototypu, který vznikl na základě drátového modelu uvedeného v sekci 6.1.1. Ve vytvořeném prototypu se na základě zpětné vazby při testování provedly změny. Změny se týkaly:

- změny používané terminologie,
- změny v zobrazování odpovídajících dat,
- změny v chování plánu kontrol.

Tohoto testování se zúčastnil pouze jeden zaměstnanec, který mi byl přidělen.

Při návrhu jsem použil komponenty z knihovny Material Design[1], která má už předdefinované styly pro vykreslované prvky a také definuje určitá pravidla, jak jednotlivé styly používat, aby vytvořené grafické rozhraní bylo například responzivní [14], tj. na různých zařízeních se zobrazovalo stejně.

Pohled znázorněný na obrázku E.3 je vytvořený podle modelu na obrázku D.1. Z tohoto pohledu se lze dostat na detail zařízení nebo si lze zobrazit otevřenou zakázku (obrázek E.5) nebo uzavřenou zakázku. Pohled na detail zařízení se nachází v příloze E.1. Sekce, kde se definuje plán kontrol na daném zařízení, je znázorněna na obrázku E.4. Na obrázku E.6 je zobrazena druhá sekce z pohledu na uzavřenou zakázku. Tato sekce zobrazuje kontroly, které byly provedeny v rámci zakázky. Celkový pohled na uzavřenou zakázku je znázorněný na obrázku E.2.

6.2 Návrh serverové části

V serverové části se nachází databázový a aplikační server. Data jsou uložena v relační databázi. Tato část poskytuje pro komunikaci REST API.

6.2.1 Model dokumentu

Pro generování dokumentů bylo třeba navrhnout model, který bude reprezentovat strukturu dokumentu a podle které lze dokument generovat a zpracovávat. Dokument se může skládat ze tří částí:

[1] Knihovna vyvinutá společností Google pro psaní aplikací s jednotným designem. Více o této knihovně lze nalézt na https://material.io/design/

- záhlaví,
- tělo dokumentu,
- zápatí.

Každá z těchto částí může obsahovat elementy stejného nebo různého typu, které se mohou libovolně opakovat. Na základě analýzy struktury protokolu v sekci 4.8 musí reprezentující model umožnit generovat elementy typu:

- tabulka,
- text,
- nadpis.

Ke každému výše uvedenému typu elementu a také k celému dokumentu lze přiřadit styl.

Tabulka

Element tabulka povoluje do sebe vkládat pouze řádky a do nich buňky. Buňky je možné spojovat vertikálně nebo horizontálně a nastavovat jim styly (např.: šířku, vycentrování vnořeného elementu, barvu pozadí). Každá buňka může v sobě obsahovat elementy typu *text* nebo další *tabulku*.

Text

Tento element je prostý text, který může mít nastavené různé styly, a to zda text bude podtržený, tučný nebo jakou bude mít velikost. Do textu je možné vložit hodnoty na konkrétní pozice a také získat změněné hodnoty.

Nadpis

Nadpis je prostý text, který má několik úrovní. Úrovně se od sebe odlišují pouze svým stylem.

6.2.2 Datový model

Datový model vychází z požadavku sepsaných v kapitole 4.6. Z požadavků jsem vytvořil ER diagram, který je ve 3. normální formě[2]. Uvedený ER diagram umožňuje přiřadit protokol přímo konkrétní pozici zařízení nebo jeho části. Návrh předpokládá, že plán kontrol se může měnit plán kontrol a zkoušek. Nyní popíšu netriviální databázové entity.

Entita Position

Entita představuje konkretní zařízení tj. zařízení určitého *typu*, *velikosti* a *umístění*. Zařízení má *název* a lze jednoznačně určit kombinací *sériového čísla* a *projektového označení*.

[2] Více o databázích a normálních formách lze nalézt na [8].

Entita User

Uživatel má následující vlastnosti: *jméno*, *příjmení*, *email*, *heslo* a *stav*. Uživatel se autentizuje pomocí svého *emailu* a *hesla*. Vlastnost *heslo* obsahuje hash hesla, který je generován jednosměrným hašovacím algoritmem. Atribut *stav* jestli uživatelský profil je aktivní či nikoliv.

Entita History

Entita představuje *zakázku*, která má vlastní *datum vytvoření*, *datum ukončení*, *předmět zakázky*, *číslo zakázky*, *číslo objednávky* a je možné k zakázce přidat i poznámku.

Entita ProtocolTemplate

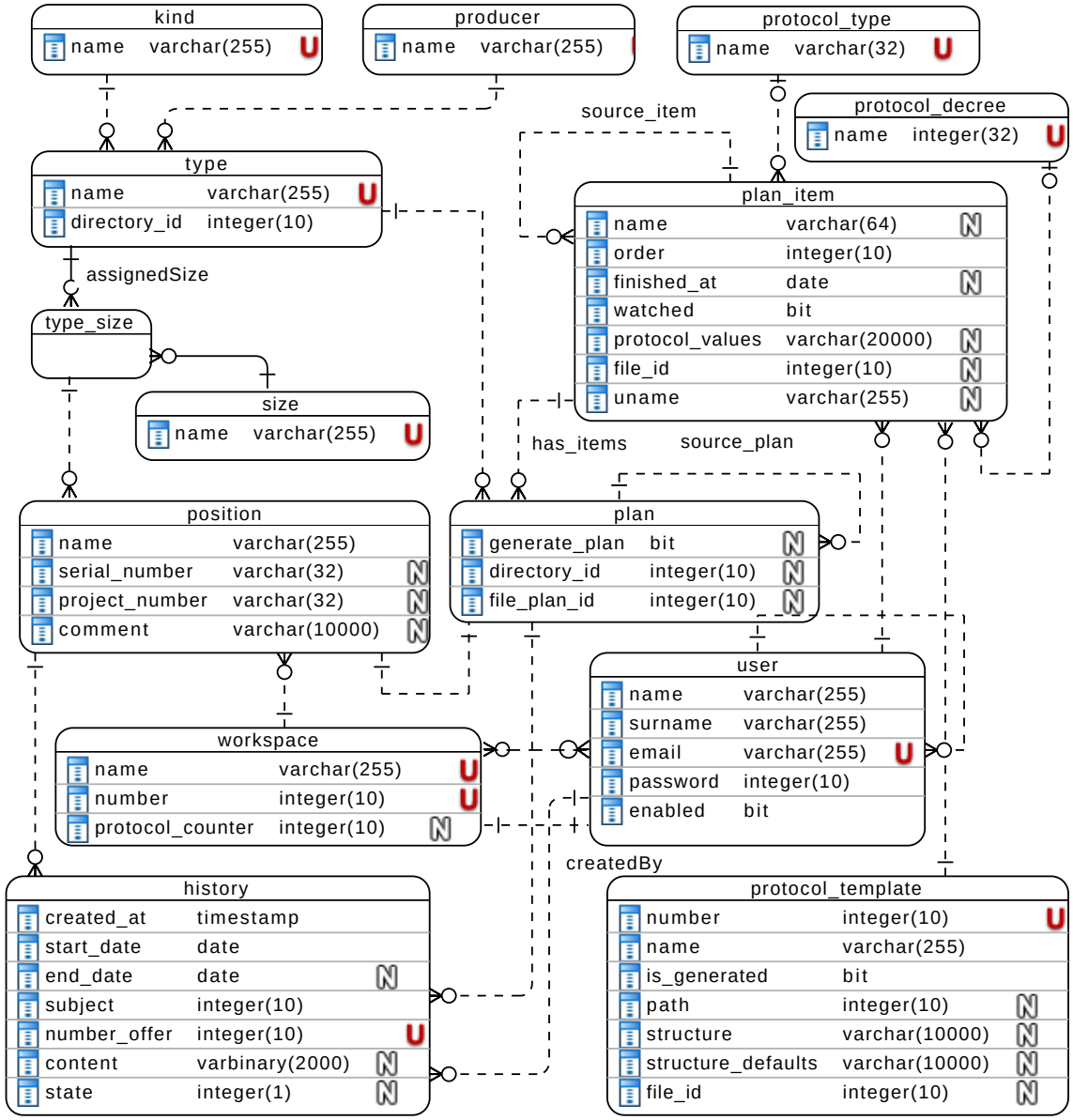
Šablona protokolu má své konkrétní *číslo* a *název*. Protokol se může generovat nebo může být založený na zdrojovém souboru. Pokud jde o druhý případ, tak protokol neumožňuje získávání dat z něj pro pozdější výpočty. Pokud se jedná o první případ, tak je možné hodnoty vyplněné v protokolu sledovat. Hodnoty ze sledovaných protokolů lze použit pro odpovídající výpočty a operace.

Entita PlanItem

Entita reprezentuje kontrolu, která se provádí na zařízení v rámci zakázky. U kontroly lze nastavovat, které *vyhlášky* kontrola splňuje, jakého je *typu*, v jakém *pořadí* se kontroly provádí, *kdy* byla kontrola splněna. Pokud jsou hodnoty z protokolu sledovány, tak jsou zde uložena i *data* získaná při zpracování protokolu. Data z protokolu jsou uložena jako JSON řetězec.

Entita Plan

Reprezentuje dva typy seznamů kontrol. První typ seznamu byl vytvořen na základě plánu, který se vytvořil na začátku zakázky. Vypracovaný plán je po odevzdání přiřazen k této entitě. Druhý typ seznamu je tvořen kontrolami, které se provádí během zakázky. Oproti prvnímu případu není u druhého generován plán kontrol.

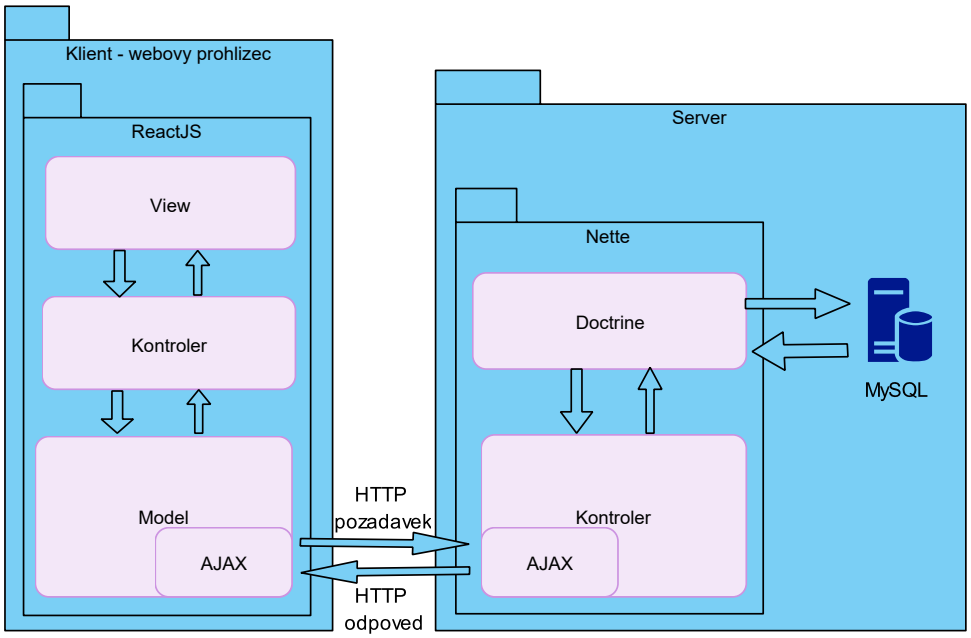


Obrázek 6.2: Výsledný ER Diagram

Kapitola 7

Implementace systému

V této části je podrobněji popsána implementace hlavních částí aplikace a to jak v klientské, tak i serverové části. Tato implementace vychází z návrhu v kapitole 6. Popis je zaměřen na práci s dokumenty ve formátech popsaných v kapitole 5, generování plánu kontrol a zobrazování historie stavů konkrétního zařízení. Schéma architektury a komunikace mezi těmito částmi a logickými celky je znázorněna na obrázku 7.1.



Obrázek 7.1: Schéma komunikace mezi klientem a serverem. Jsou zde také znázorněny logické části aplikace.

7.1 Použité nástroje, principy a technologie

Pří návrhu bylo třeba si stanovit základní nástroje, které budou využity při vývoji aplikace. Technologie byly vybrány po analýze požadavků a problematiky. Vývoj probíhal ve vývo-

jových prostředích PhpStorm[1] pro vývoj serverové části a WebStorm[2] pro vývoj serverové části. Tyto nástroje vytvořila společnost JetBrains.

Pro správu verzí jsem použil verzovací systém git. Využil jsem ho pro revizi změn a také to byla forma zálohování projektu na vzdáleném serveru. Pro vzdálené ukládání repozitáře jsem použil server GitLab[3].

7.1.1 Docker

Docker[4] je software, který poskytuje jednotné rozhraní pro izolaci procesů do kontejnerů. Kontejner obsahuje pouze požadované aplikace, ale neobsahuje virtualizovaný operační systém. Umožňuje zabalit aplikace a všechny jejich závislosti do kontejneru.

Docker jsem použil pro namodelování prostředí, jaké je na ostrém serveru. Poté jsem mohl vyvíjet aplikaci kdekoliv bez ohledu na nainstalovaných službách. Vytvořil jsem si dva kontejnery (jeden pro MySQL server a druhý pro webový server), které jsem mezi sebou propojil a získal tak prostředí odpovídající ostrému serveru.

7.1.2 REST

REST (zkratka z anglického *Representation State Transfer*) je architektura rozhraní, která umožňuje přistupovat k datům na určitém místě pomocí standardních metod HTTP. REST je orientován datově [16]. Webové služby definují způsob komunikace a REST určuje, jak se přistupuje k datům.

7.2 Implementace klientské části

Pro tvorbu uživatelského rozhraní jsem vybral možnost generování HTML kódu u klienta, tedy v prohlížeči. To mi umožnilo vytvořit rozhraní poskytující lepší interakci s uživatelem. K tomu jsem použil knihovnu ReactJS[5], která slouží pro popis toho, jak data získaná ze serveru budou u klienta reprezentována. Tato knihovna také umožňuje udržovat kontext, ve kterém uživatel aplikaci používá. Tohoto jsem využil při validaci formulářů a při přechodech mezi jednotlivými pohledy. Klientská část je tvořena z několika komponent, které lze používat na více místech, řada kontrolérů a k nim příslušných pohledů. Kontroléry komunikují s aplikačním rozhraním serveru přes služby, které zprostředkovávají AJAX volání. Komponenty mají na starosti logické celky, například obsluhu formuláře nebo navigaci. Každý kontrolér, pohled, komponenta mají jednotné rozhraní, které deklaruje metodu `render()`, která umožní jejich vykreslení.

Klientská část komunikuje se serverem pomocí asynchronních zpráv. Klient vygeneruje požadavek a pošle ho, ale nečeká na odpověď. Při získání odpovědi ze serveru se odpověď zpracuje a vyvolá se událost, na kterou reagují pouze ty části aplikace, které se získanými daty pracují nebo chtějí pracovat. Mezi hlavní části aplikace patři:

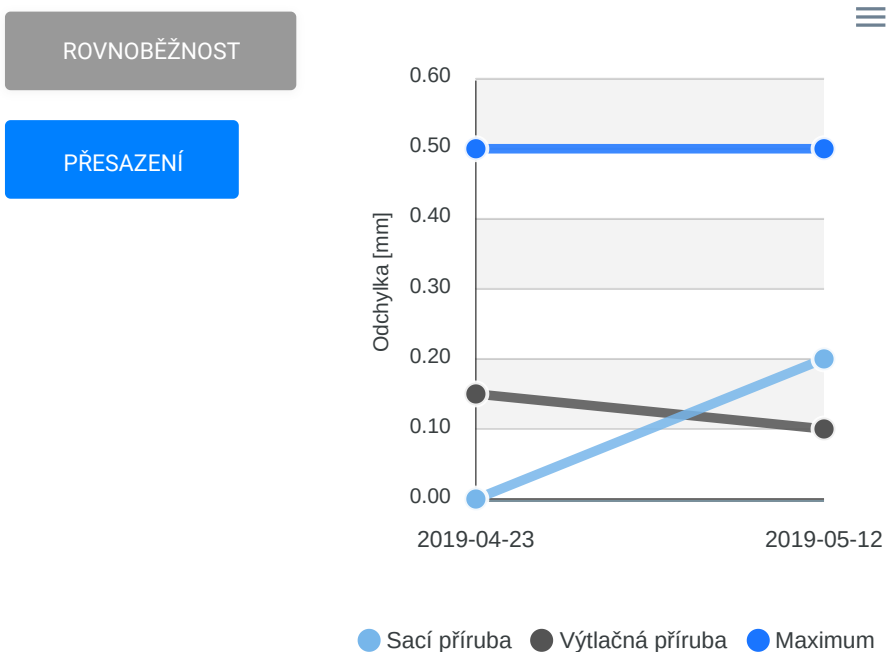
- vykreslování stavu zařízení,
- plán kontrol.

[1] Více informací o nástroji lze nalézt na https://www.jetbrains.com/phpstorm/.
[2] Více informací o nástroji lze nalézt na https://www.jetbrains.com/webstorm/.
[3] Více informací lze nalézt na https://about.gitlab.com
[4] Více informací dostupných na https://www.docker.com.
[5] Knihovna napsaná v jazyce JavaScript. Více se lze dozvědět na https://reactjs.org/

7.2.1 Zobrazení stavu zařízení

Stav *Zařízení*[6] se určuje na základě kontrol, které se na něm provádí během zakázek. V rámci některých kontrol se může provádět i více typů kontrol.

Na obrázku 7.2 je znázorněn graf, který byl vytvořen ze dvou zakázek na základě kontroly, ve které se kontroluje rovnoběžnost a přesazení. Pro zobrazování údajů o stavu zařízení byl zvolen spojnicový graf v závislosti vypočítaných hodnot na čase (čas provedení kontroly). Pokud se na *Zařízení* sleduje více dat, tj. provádí se něm více protokolů různého typu, tak se v levé nabídce objevují odpovídající možnosti. Pravá část je vykreslována pomocí knihovny *react-apexcharts*[7], která vygeneruje SVG (zkratka z anglického *Scalable Vector Graphics*) obrázek.



Obrázek 7.2: Ukázka stavu zařízení, který se vygeneruje na základě dvou zakázek, kde se provedly kontroly rovnoběžnosti a přesazení u výtlačné a sací příruby.

7.2.2 Plán kontrol

Plán kontrol je spravován dvěma hlavními komponentami:

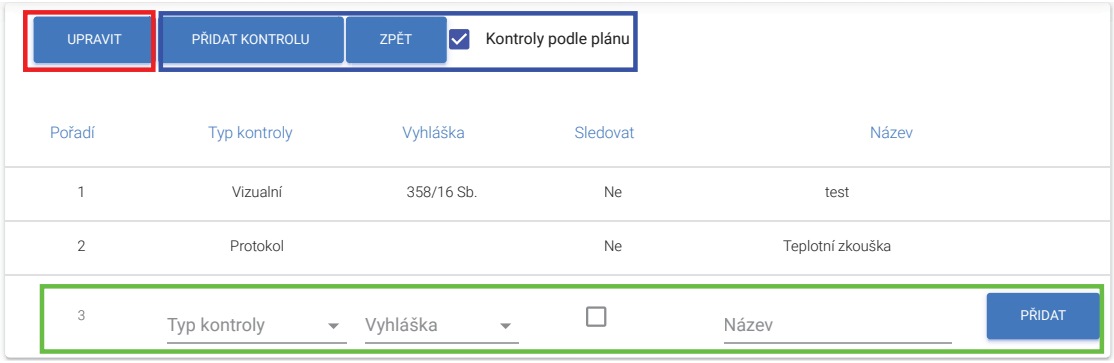
- **OrderPlan**: S touto komponentou uživatel pracuje, pokud zakázka není ve stavu otevřená. Poskytuje funkce:
  - odevzdání a stažení vypracovaného dokumentu,
  - stažení prázdného (vygenerovaného) dokumentu,
  - vygenerování, stažení a odevzdání plánu kontrol, pokud kontroly byly prováděny podle plánu kontrol, a to pomocí tlačítek, které se zobrazí pod seznamem provedených kontrol, ale pouze tehdy pokud všechny kontroly byly provedeny,

[6] Zařízení konkrétního typu, na určité místě (pozici).
[7] Více informací o této knihovně lze nalézt na https://apexcharts.com/.

  - přepnutí plánu kontrol do režimu úprav (pokud je to možné) a to kliknutím na tlačítko *Upravit* zobrazeném v červeném rámečku na obrázku 7.3.

- **OrderPlanEdit**: Zobrazuje stejná data jako komponenta *OrderPlanEdit*, ale poskytuje jiné funkce:
  - přidat novou kontrolu,
  - uložit nebo zrušit provedené změny.

Obě komponenty jsou založeny na komponentě *React.Component*. Bez tohoto dědění by komponenty nešly vůbec vykreslit. Komponenty se mezi sebou odlišují svými funkcemi, ale vypadají téměř stejně. Rozdíly lze ukázat na obrázku 7.3. Komponenta *OrderPlan* může zobrazit pouze obsah modrého rámečku a komponenta *OrderPlanEdit* zobrazuje pouze obsah modrého rámečku a může i obsah zeleného rámečku.



Obrázek 7.3: Implementovaný plán kontrol. Obrázek zobrazuje vykreslenou komponentu *OrderPlanEdit*, ale tlačítko v červeném rámečku pochází z komponenty *OrderPlan*. To je z toho důvodu, aby bylo možné ukázat rozdíl mezi těmito dvěma komponentami. První komponenta může zobrazit obsahy modrého a zeleného rámečku a druhá komponenta pouze obsah červeného rámečku.

7.3 Implementace serverové části

Server požívá technologie označované jako LAMP (**L**inux, **A**pache, **M**ySQL, **p**hp). Serverová část aplikace je postavena na frameworku Nette[8], který poskytuje řadu komponent pro jazyk PHP. Zvolil jsem tento framework, protože aplikace, do které se integruje tento systém, je postavena na stejném frameworku. Pro práci s databází jsem použil knihovnu Doctrine[9]. Data jsou uložena v databázi MySQL. Tato část poskytuje pro komunikaci REST API[10], které framework přímo nepodporuje, proto bylo třeba implementovat podporu pro toto rozhraní.

[8] Více informací o tomto frameworku lze nalézt na https://nette.org
[9] Více informací lze nalézt na https://www.doctrine-project.org
[10] AJAX je asynchorní komunikace mezi klientem a serverem. Více informací se lze dozvědět na https://www.w3schools.com/js/js_ajax_intro.asp

7.3.1 Anotace pro REST API

Pro jednoduchou tvorbu REST API jsem vytvořil mechanismus, který při prvním spuštění aplikace projde všechny kontroléry v architektuře MVC a pokud najde anotaci *@ApiRoute*, tak ji zpracuje a uloží do souboru (do cache).

Anotace musí vždy obsahovat cestu (část URL), kterou je identifikován konkrétní zdroj, na který se můžeme dotazovat. Vytvořená anotace se může objevit na dvou místech:

- **u kontroléru**: Jedná se o třídu, která obsahuje metody. Pokud má metoda jeden ze speciálních názvů, které jsou uvedeny níže, jedná se o tzv. „speciální metody“. U těchto speciálních metod není potřeba uvádět znovu anotaci. Mechanismus metodu zavolá automaticky podle typu požadavku od klienta. Speciální metody a jejich odpovídající požadavky jsou:
  - `actionCreate`: POST,
  - `actionRead`: GET,
  - `actionUpdate`: PUT,
  - `actionDelete`: DELETE,
  - `actionOptions`: OPTIONS.

  Pokud libovolná speciální metoda není v kontroléru definována, tak tento typ požadavku není podporován.

- **u metody**: Tato možnost se používá, když chceme v jednom kontroléru zpracovávat požadavky s různými cestami, ale požadavky spolu nějak souvisí. Název metody musí odpovídat regulárnímu výrazu *action[0-9a-zA-Z_\-]+*, aby framework s metodou správně pracoval. Například: máme kontrolér *User*, který spravuje následující operace s uživateli a jim odpovídající cesty:
  - registrace uživatele: „/sign-in“,
  - přihlášení uživatele: „/sign-out“.

  K tomu stačí vytvořit dvě metody. K metodě zpracovávající požadavek pro přihlášení uživatele přidat anotaci *@ApiRoute("/sign-in", method="POST")* a k druhé *@ApiRoute("/sign-out", method="POST")*.

Na ukázce kódu 7.1 jsou zobrazeny výše popsané možnosti použití.

```
/**
 * @ApiRoute("/path")
 */
class PathResource {
    ...
    /**
     * @ApiROute("/path/example", method="GET")
     */
    public function actionList() {
        ...
    }
}
```

Výpis 7.1: Ukazka použití anotace *@ApiRoute*

7.3.2 API serveru

Pro přístup a k práci s daty jsem vytvořil na serveru rozhraní (REST API). Data mezi klientskou části serverem lze přenášet pouze ve formátu JSON. Rozhraní umožňuje operace (typu) CRUD s databázovými entitami zobrazené na obrázku 6.2. Celé rozhraní je popsané v příloze F. CRUD operace jsou definovány následovně:

- **C**reate: vytvoření nové entity
- **R**ead: pokud je součástí požadavku parametr ID, tak se získá pouze entita s tímto ID, v opačném případě se vrátí seznam všech entit
- **U**pdate: upraví se entita s ID
- **D**elete: odstraní se entita podle předaného ID

Při vývoji klientské části se objevila komplikace a to, že docházelo k blokování asynchronních požadavků z klientské části na server. Požadavky byly zablokovány bezpečnostním mechanismem CORS[11] (zkratka z anglického *Cross-origin Resource Sharing*), který je podporován dnešními prohlížeči. Tento mechanismus ve výchozím nastavením povoluje spouštění skriptů pocházející pouze ze stejného zdroje. To byl při vývoji problém, protože klientská část ve vývojovém režimu běží na portu 3000 a serverová část na portu 80, a již toto je mechanismem považováno za různé zdroje. Z toho důvodu bylo potřeba na serveru implementovat i podporu pro HTTP metodu OPTIONS[12], kterou mechanismus používá pro zjištění nastavení sdílení zdrojů mezi různými doménami. Natavení se posílá klientovi v odpovědi na požadavek typu OPTIONS ve formě hlaviček. Bylo třeba nastavit následující hlavičky:

- **Access-Control-Allow-Origin**: pro povolení domény *http://localhost:3000*
- **Access-Control-Allow-Methods**: pro povolení metod *GET, POST, PUT, DELETE, HEAD*
- **Access-Control-Allow-Credentials**: nastavením na *true* je umožněno posílat v požadavku i cookies a HTTP autentizační data
- **Access-Control-Allow-Headers**: seznam povolených hlaviček
- **Access-Control-Max-Age**: nastavení doby, po kterou je odpověď mít uloženou v cache

7.3.3 Model dokumentu

Podle návrhu v kapitole 6.2.1 bylo třeba vytvořit model, kterým lze reprezentovat textové dokumenty. Model, který byl implementován jako asociativní pole, se ukládá v třídě *ProtocolTemplate* do atributu *template*, kde se při vložení převede na text a při získání je převeden zpět na asociativní pole. Převod modelu na text umožňuje funkce *json_encode* a funkce *json_decode* je zpětně převeden text na model. Zmíněné funkce jsou vestavěné funkce

[11] Více informací na https://medium.com/@electra_chong/what-is-cors-what-is-it-used-for-308cafa4df1a.

[12] Více informací dostupných na https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS.

jazyka php. Třída *ProtocolTemplate* je mapována pomocí mechanismu ORM (zkratka z anglického *Object-relation mapping*) na databázovou tabulku *protocol_template*.

Kořenové asociativní pole má následující možné klíče a jejich hodnoty:

- **header**: obsah hlavičky,
- **body**: tělo dokumentu,
- **footer**: obsah patičky dokumentu,
- **orientation**: nastavení orientace dokumentu,
- **styles**: globální styly pro dokument.

V následujících odstavcích budu pro větší přehlednost budu místo prvek v asociativním poli pod klíčem *XY* psát jen prvek *XY*. Pokud není uvedeno jinak, všechny rozměry jsou v jednotkách *twip*[13]. Prvky *header*, *body* a *footer* mohou obsahovat pole prvků nebo pouze jeden prvek. Každý typ prvku má jedinečný identifikátor tj. klíč v asociativním poli, který určuje, jak se s daným prvkem bude pracovat. Implementovaný model podporuje následující typy prvků:

- **table**: Prvek reprezentující tabulku v dokumentu. Je to asociativní pole, které muže obsahovat prvek *style*, který obsahuje název stylu definovaný v prvku *styles* v kořenovém asociativním poli. Povinný datový prvek *data* obsahuje pole řádků tabulky. Každý řádek obsahuje ve svém prvku *data* pole buněk a může obsahovat i prvek *height* pro nastavení výšky řádku. Buňka řádku může obsahovat i prvky:
  - **width**: Je to volitelný prvek pro nastavení šířky buňky. Pokud prvek neexistuje, buňky v řádku mají stejnou šířku. Pokud má hodnotu „null“, prvek *content* v buňce je ignorován a buňka slouží pro vertikální spojování buněk.
  - **content**: Je to povinný prvek, pokud hodnota prvku *width* není rovna *null*. Může obsahovat pole prvků nebo pouze jeden prvek.
  - **style**: Prvek pro nastavení stylů buňky. Obsahuje asociativního pole, kde klíč je název nastavovaného stylu. Jsou podporované styly popsané v dokumentaci PHPOffice/PHPWord[14].
- **decreeText**: Speciální prvek, který jako jediný ze všech prvků se připojuje k databázi, aby zjistil, jaké jsou používané vyhlášky v protokolech.
- **decreeValue**: Speciální prvek, který souvisí s předchozím prvkem. Mění své chování na základě toho k jaké vyhlášce se vykresluje. Pokud se vykresluje k vyhlášce, pod kterou se generovaný dokument vypracovává, zaškrtne se hodnota *ano* v opačném případě hodnota *ne*.
- **text**: Prvek reprezentující prostý text v dokumentu, který muže obsahovat přímo text k zobrazení nebo asociativní pole s těmito prvky:
  - **alignment**: volitelný prvek, který nastavuje zarovnání textu
  - **style**: volitelný prvek, který definuje styl textu

[13] Twip je 1/20 palce.
[14] Odkaz na dokumentaci stylů https://phpword.readthedocs.io/en/latest/styles.html

  - **data**: povinný prvek, který obsahuje text k zobrazení

  Text k zobrazení může obsahovat speciální místa, která slouží pro vložení nebo získání hodnot. Místa mají tvar *%[a-zA-Z˙-0-9]+%*. Text mezi znaky % je identifikátor, podle kterého se vyhledávají hodnoty pro vložení hodnot nebo jsou tímto identifikátorem označeny hodnoty, které se do dokumentu doplnily.

- **blockText**: Prvek obsahující pole textů, které jsou převáděny na prvky typu *text*.
- **title**: Prvek reprezentující text. V dokumentu je text zobrazen jako nadpis úrovně 1, na který je použit styl definovaný v prvku *styles* v kořenovém asociativním poli pod hodnotou „1“.
- **title2**: Prvek je podobný prvku *title* s jediným rozdílem, a to že je na něj aplikován styl pod hodnotou „2“ v prvku *styles* v kořenovém asociativním poli.

V prvku *styles* jsou definované styly pro používané tabulky v dokumentu a úrovně nadpisů, na které se odkazuje v dokumentu pomocí unikátního identifikátoru. Styly definované v tomto prvku a také v prvku *style* jsou popsány v již zmíněné dokumentaci PHPOffice/-PHPWord, která se používá pro generování dokumentů ve formátu *docx*.

Pomocné třídy

Pro generování dokumentů se používá knihovna PHPOffice/PHPWord, ale bylo potřeba implementovat mezivrstvu, která by sloužila pro obousměrný převod mezi navrženým modelem dokumentu a touto knihovnou. Mezivrstva je tvořena následujícími třídami:

- **AbstractElement**: Základní třída, od které dědí další elementy, obsahuje metody pro vytvoření třídy *Table* a *TableRow*. Dále obsahuje funkce pro vkládání hodnot do textových šablon a také pro zpětné získání vyplněných hodnot.
- **Table**: Generuje a zpracovává obsah tabulky.
- **TableRow**: Generuje a zpracovává řádek tabulky.
- **ProtocolGenerator**: Generuje dokument podle předaného modelu dokumentu.
- **ProtocolReader**: Má na starost zpracování dokumentu podle předaného modelu a získává vyplněné hodnoty.
- **ProtocolFunctions**: Obsahuje pomocné funkce pro zpracování získaných hodnot z protokolu.

7.3.4 Generování plánu kontrol

Plán kontrol je dokument generující se na konci zakázek, u kterých se kontroly provádí podle plánu, jako výstupní dokument ze zakázky.

Pro každou kontrolu je vygenerován řádek zobrazený na obrázku 7.4, do kterého se vloží:

1. pořadí kontroly,
2. název kontroly,

3. rozsah kontroly (P = protokol, V = vizuální),
4. jméno zaměstnance, který kontrolu provedl,
5. název protokolu (pokud se jedná o protokol),
6. datum provedení kontroly.

2	Teplotní zkouška	P		Petr Novák					SE-01_24	23-04-2019

Obrázek 7.4: Ukázka vygenerovaného řádku v plánu kontrol

7.3.5 Generování dokumentů v různých formátech

Knihovna PHPOffice/PHPWord, jak už bylo zmíněno v kapitole 5.4.1, částečně podporuje několik formátů souboru. Nejvíce je podporován formát *docx*[15]. Z toho důvodu bylo generování souborů v jiných formátech rozděleno do dvou kroků:

1. Pomocí knihovny PHPOffice/PHPWord byl vygenerován soubor ve formátu *docx*.
2. Pomocí programu Libre Office je dokument převeden do požadovaného formátu. Tento program poskytuje rozhraní umožnující kromě jiného převod mezi různými formáty. Zavoláním příkazu:
   `soffice headless convert-to FORMAT outdir DIR SOURCE`
   kde:
   - **headless**: spuštění aplikace bez uživatelského rozhraní
   - **convert-to FORMAT**: pro převod vstupního souboru do nového souboru typu *FORMAT* (přípona nového souboru)
   - **outdir DIR**: složka pro uložení výstupního souboru
   - **SOURCE**: vstupní soubor

Při zpracování se nahraný soubor převede opět do formátu *docx* pomocí programu Libre Office a poté se zpracuje pomocí knihovny PHPOffice/PHPWord. Aby knihovna dokázala nahraný soubor správně zpracovat, je nutné dodržet následující omezení:

- Formát *doc* lze editovat pouze v programu Microsoft Word 2003 nebo ve starší verzi.
- Formát *docx* lze editovat pouze v programu Microsoft Word 2007 nebo v novější verzi.
- Formát *odt* lze editovat pouze v programu Writer od LibreOffice.

Při nedodržení těchto omezení dochází při editaci souboru ke změnám zdrojové struktury souboru, kvůli kterým knihovna není schopna rozpoznat strukturu souboru.

[15] Více informací o podpoře formátů na https://phpword.readthedocs.io/en/latest/intro.html#file-formats.

Kapitola 8

Testování

Pro ověření správné funkčnosti aplikace je vhodné aplikaci před jejím použitím otestovat. Abych co nejlépe ověřil správnou funkčnost aplikace, provedl jsem následující druhy testování:

- testování pomocných tříd,
- kompatibilita mezi prohlížeči,
- uživatelské testování.

8.1 Testování pomocných tříd

Pomocí tohoto typu testování jsem ověřil správnou funkčnost vytvořených pomocných tříd popsaných v kapitole 7.3.3. Pro testování jsem použil framework PHPUnit[1] pro tvorbu automatizovaných testů a také rozšíření pro jazyk php ImageMagick[2].

Většina testů pro pomocné třídy pracuje na principu vygenerování dokumentu, následného převedení na obrázek a porovnávání obrázků. Důvodem je, že kdyby se porovnávaly pouze dokumenty, testy by selhaly, protože by se dokumenty lišily v datech vytvoření.

Vytvořil jsem několik testů testující elementy, které lze pomocnými třídami generovat. Další testy jsem vytvořil pro vložení dat do dokumentu a opětovného přečtení. Testoval jsem pouze dokumenty ve formátu *docx*, protože při generování jiných typů se vygeneruje nejdříve dokument typu *docx* a poté se převede pomocí programu LibreOffice do požadovaného formátu. Vytvořené testy ověřují funkčnost:

- nastavení orientace stránek dokumentu,
- vložení elementů do hlavičky, těla nebo patičky dokumentu,
- generování textu s různými styly,
- generování tabulky s různými styly,
- nastavení výšky řádku, vertikální a horizontální spojení buněk
- nastavení stylů buňky.

[1] Více informací lze nalézt na: https://phpunit.de/.
[2] Více informací lze nalézt na: https://www.php.net/manual/en/intro.imagick.php.

8.2 Kompatibilita mezi prohlížeči

Aplikace se může v různých zařízeních a prohlížečích zobrazovat odlišně. V některých typech prohlížečů nemusí být podporované všechny používané prvky a funkce. Toto testování může být automatizované, ale vytvořením takovýchto testů může být velmi komplikované, protože by musely porovnávat, jak se daný prvek zobrazuje. Testy lze provádět i manuálně, ale to může být časově náročné. V rámci těchto testů je důležité specifikovat, na jakých platformách a v jakých typech prohlížečů se bude aplikace používat.

Aplikaci jsem testoval v operačních systémech (dále jen OS) Windows 7 a Windows 10 a v internetových prohlížečích (dále jen IntP) Internet Explorer (ve verzích 9 až 11) a Google Chrome (ve verzích 70.0 až 73.0). Vybral jsem takové OS, protože se jiné systémy ve firmě nepoužívají a ostatní IntP nejsou povoleny.

Testování probíhalo manuálně a za pomocí programu TeamViewer[3]. Pomocí programu TeamViewer jsem byl připojený na dva firemní počítače. Jeden počítač byl s OS Windows 7 a druhý počítač byl s OS Windows 10. Na obou počítačích byl nainstalovaný IntP Internet Explorer 11 a Google Chrome 73.0. Ostatní verze jsem testoval pomocí virtualizace OS, kde jsem nainstaloval požadované verze IntP.

8.3 Uživatelské testování

Pro zjištění, zda produkt funguje a ulehčuje zaměstnancům firmy práci, jsem použil metodu popsanou v kapitole 3.4.1. Zaměstnancům jsem na začátku testu zadal úkol, který měli vypracovat v implementované aplikaci a také v aplikaci Mocev. Při testování jsem měřil čas, který zaměstnanci potřebovali ke splněné požadované úlohy.

Aplikaci jsem testoval po nasazení pouze s jedním zaměstnancem a po 14 dnech jsem testoval aplikaci znovu, ale už se všemi zaměstnanci, kteří aplikaci budou používat. Testoval jsem aplikaci ve dvou etapách, protože jsem chtěl zjistit, zda se naměřené hodnoty u zaměstnance testovaného v první etapě, nezmění. Mezi těmito měřeními měl aplikaci zpřístupněnou. Jeho naměřené časy při druhém měření byly přibližně o 7s lepší než časy při prvním měření. Časy nejsou o tolik rozdílné, protože zaměstnanec se mohl zlepšit pouze při interakci se systémem, ale doba zpracování stejného protokolu stále zůstane stejná.

V druhé etapě testování probíhalo se 7 zaměstnanci. V každé etapě jsem zadal tyto úkoly, ale s jinými testovacími daty:

1. **Vytvoření zakázky**: Zaměstnanec musel vytvořit novou zakázku a vyplnit data podle údajů, které jsem zaměstnanci před testem předal.

2. **Přidání protokolu do seznamu kontrol**: Zaměstnanec měl přiřadit konkrétní protokol do seznamu kontrol u konkrétní zakázky.

3. **Vyplnění plánu kontrol a odevzdání**: Zaměstnanec si měl u konkrétní zakázky vygenerovat plán protokol a vyplnit v něm údaje, které jsem zaměstnanci před testem předal. Plán kontrol nesměl obsahovat vizuální kontroly, protože program Mocev tento typ kontrol nepodporuje, a proto by naměřené časy nešly srovnávat.

Na začátku každého úkolu se uživatel nacházel na obrazovce E.3 nebo v případě aplikace Mocev na obrazovce, která se zobrazí při spuštění aplikace. Testování probíhalo v jejich přirozeném prostředí (v kanceláři).

[3]Více o tomto programu lze nalézt na: https://www.teamviewer.com/cs/.

V tabulce 8.1 jsou průměrné naměřené hodnoty v druhé etapě. Podle naměřených dat lze říct, že implementovaná aplikace ušetřila čas. Na třetím úkolu je nejvíce vidět ušetření času, protože se v tomto úkolu pracuje s dokumentem, který uživatel musí vyplnit. V posledním úkolu při práci s programem Mocev jsem měřil pouze čas potřebný pro vyplnění dokumentu do stejného stavu, jaký vygeneruje implementovaná aplikace.

	Naměřené časy		**Vyhodnocení**
	Implementovaná aplikace	**Mocev**	
1. Vytvoření zakázky	56s	75s	**-25%**
2. Přidání protokolu...	16s	41s	**-61%**
3. Vyplnění plánu kontrol...	13s	239s	**-95%**

Tabulka 8.1: Naměřené časy v implementovaném systému a v aplikaci Mocev. Poslední sloupec obsahuje porovnání naměřených hodnot.

Poslední sloupec tabulky 8.1 obsahuje porovnání naměřených časů mezi implementovanou aplikací a programem Mocev vyjádřený v procentech. Pokud je hodnota záporná (kladná), tak to znamená, že čas se zlepšil (zhoršil) v porovnání s časem naměřeným při použití programu Mocev. Vypočítal jsem to podle následujícího vzorce:

$$\frac{t_{app} - t_{mocev}}{t_{mocev}} * 100\%$$

Kde:

- t_{app}: naměřená hodnota v implementované aplikaci,
- t_{mocev}: naměřená hodnota v programu Mocev.

Kapitola 9

Závěr

V rámci práce jsem navrhl a implementoval systém umožňující správu kontrol kvality zařízení. Při návrhu jsem vycházel ze zpracovaných informací získaných z rozhovorů se zaměstnanci, kteří systém budou používat a z přímého pozorování zaměstnanců při práci s programem Mocev, který tato aplikace nahradí. Uživatelské rozhraní jsem začal tvořit od drátěného modelu a po jeho schválení jsem přešel k tvorbě prototypu. Nakonec po otestování prototypu jsem celý systém implementoval.

V rámci práce jsem musel často komunikovat s firmou. Vedení firmy mi pro tyto účely přidělilo pracovníka. Získané informace o kontrolách, které se používají pro zajištění kvality zařízení, jsem zpracoval v úvodní kapitole. S přiděleným zaměstnancem jsem testoval každý větší pokrok nebo změnu v návrhu a implementaci.

Také jsem se seznámil s některými metodami pro získání informací od zákazníků, zpracování získaných informací, na základě nich navrhnout grafické rozhraní a s metodami pro otestování rozhraní, zda splňuje požadavky uživatelů. Tyto metody jsem použil při analýze požadavků a také při návrhu grafického rozhraní.

Na základě získaných požadavků jsem potřeboval zjistit, jak se pracuje s textem a tabulkami v textových souborech. Provedl jsem analýzu formátů DOC, DOCX a ODT, které používají známé textové editory.

Systém v současné době již plně nahradil doposud používaný program Mocev a je zaintegrován do informačního systému divize Energo. Aplikace splňuje stanovené požadavky, ale již v současné době mne napadá jak systém vylepšit. Mezi možná vylepšení patří změna rozlišení před uploadem obrázku a přidání podpory pro PDF dokumenty, které by mohly v budoucnu nahradit používané textové formáty (DOC, DOCX a ODT).

Literatura

[1] Alan Cooper, D. C. C. N., Robert Reimann: *About Face: The Essentials of Interaction Design*. John Wiley & Sons, Inc., 2014, ISBN 978-1-118-76657-6.

[2] Alben, L.: Defining the criteria for effective interaction design. *interactions*, ročník 3, č. 3, 1996: s. 11–15.

[3] Apache: Apache POI - the Java API for Microsoft Documents. Technická zpráva, The Apache Software Foundation, 2018.

[4] Ben Coleman, D. G.: *Designing UX: Prototyping: Because Modern Design is Never Static*. SitePoint, 2017, ISBN 978-0-9953827-1-8.

[5] Brown, D. M.: *Communicating Design: Developing Web Site Documentation for Design and Planning (2nd Edition) (Voices That Matter)*. Addison Wesley, 2011, ISBN 978-0-321-71246-2.

[6] Corporation, M.: Office Common Data Types and Objects Structures. Technická zpráva, Microsoft Corporation, 2018.

[7] Corporation, M.: Word (.doc) Binary File Format. Technická zpráva, Microsoft Corporation, 2018.

[8] Date, C. J.: *Database design and relational theory : normal forms and all that jazz*. O'Reilly, 2012, ISBN 978-1-449-32801-6.

[9] Don Norman, J. N.: The Definition of User Experience (UX). *Nielsen Norman Group*, 2017.
URL https://www.nngroup.com/articles/definition-user-experience/

[10] Garrett, J. J.: *The Elements of User Experience: User-Centered Design for the Web and Beyond, 2nd Edition (Voices That Matter)*. New Riders, 2011, ISBN 978-0-321-68368-7.

[11] International, E.: Standard ECMA-376 Office Open XML File Formats. Technická zpráva, ECMA, 2016.

[12] ISO: Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems. Technická Zpráva 9241-210:2010, ISO, 03 2010.

[13] Ivory, M. Y.; Hearst, M. A.: The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys (CSUR)*, ročník 33, č. 4, 2001: s. 470–516.

[14] Jehl, S.: *Responsible Responsive Design*. Jeffrey Zeldman, 2014, ISBN 978-1-9375571-6-4.

[15] Lang, J.; Howell, E.: *Researching UX: User Research*. SitePoint, 2017, ISBN 978-0-9953826-3-3.

[16] Masse, M.: *REST API Design Rulebook*. O'Reilly, 2012, ISBN 978-1-449-31050-9.

[17] Miles, R.; Hamilton, K.: *Learning UML 2.0*. O'Reilly, 2006, ISBN 978-0-596-00982-3.

[18] Moule, J.: *Killer UX Design*. SitePoint, 2012, ISBN 978-0-9872478-0-3.

[19] Nielsen, J.: Why You Only Need to Test with 5 Users. *Nielsen Norman Group*, 2000. URL https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[20] Nielsen, L.: *Personas - User Focused Design*. Springer-Verlag London, 2013, ISBN 978-1-4471-4084-9.

[21] OASIS: Open Document Format for Office Applications (OpenDocument). Technická zpráva, OASIS Standard, 2011.

[22] PHPOffice: Welcome to PHPWord's documentation. Technická zpráva, [online], 2017.

[23] Shneiderman, B.; Plaisant, C.: *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison Wesley, 2005, ISBN 0-321-19786-0.

[24] Unger, R.; Chandler, C.: *A Project Guide to UX Design*. New Riders, 2009, ISBN 978-0-321-60737-9.

[25] Vermeeren, A. P.; Law, E. L.-C.; Roto, V.; aj.: User experience evaluation methods: current state and development needs. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, ACM, 2010, s. 521–530.

Příloha A

Diagram případů užití

Obrázek A.1: Výsledný diagram případu užití

Příloha B

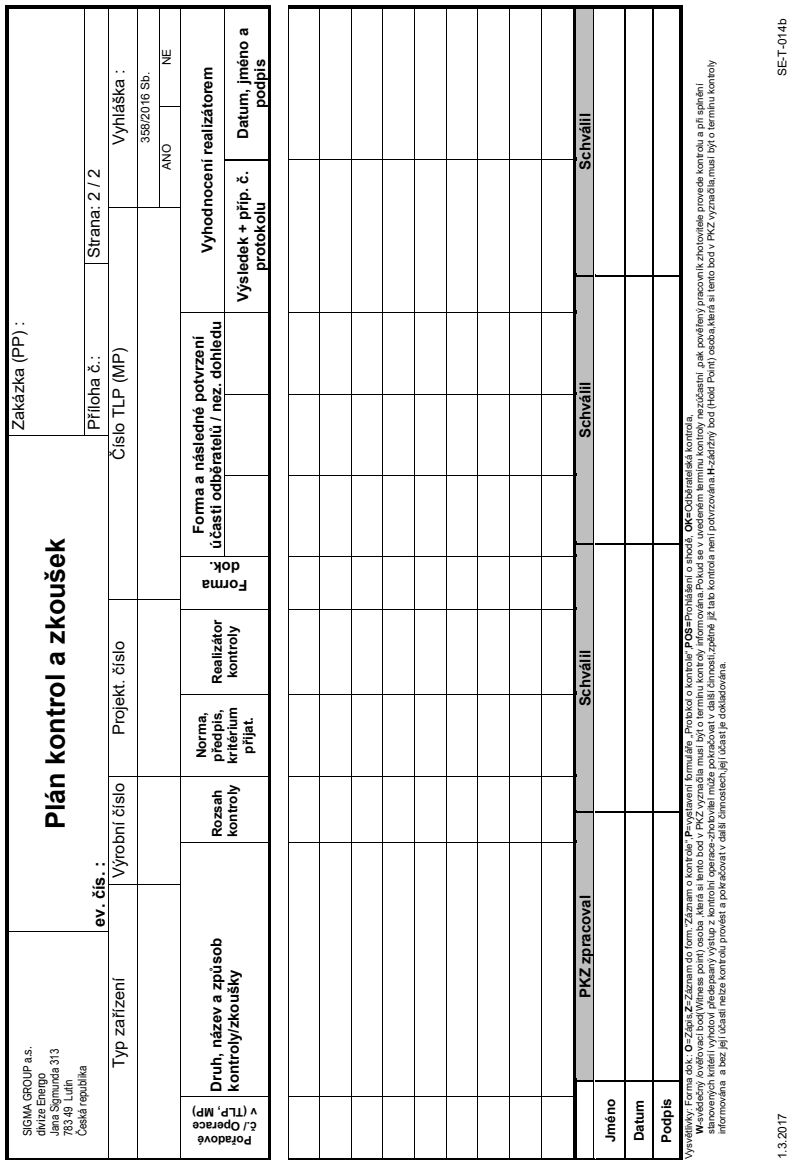
DOCX - adresářová struktura

/[Content_Types].xml	Typ obsahu
/_rels/.rels	Odkazy na balíčky
/docProps/app.xml	Aplikací definované vlastnosti
/docProps/core.xml	Vlastnosti jádra
/word/document.xml	Hlavní dokument
/word/_rels/document.xml.rels	Odkazy na části
/word/comments.xml	Komentáře
/word/endnotes.xml	Poznámky
/word/fontTable.xml	Písma
/word/footer1.xml	Část pro patu dokumentu
…	
/word/footnotes.xml	Poznámky pod čarou
/word/header1.xml	Část pro hlavičky
…	
/word/numbering.xml	Definice číslování
/word/settings.xml	Nastavení dokumentu
/word/styles.xml	Definice stylů
/word/theme/theme1.xml	Téma

Obrázek B.1: Struktura souborů balíku WordprocessingML.[11]

Příloha C

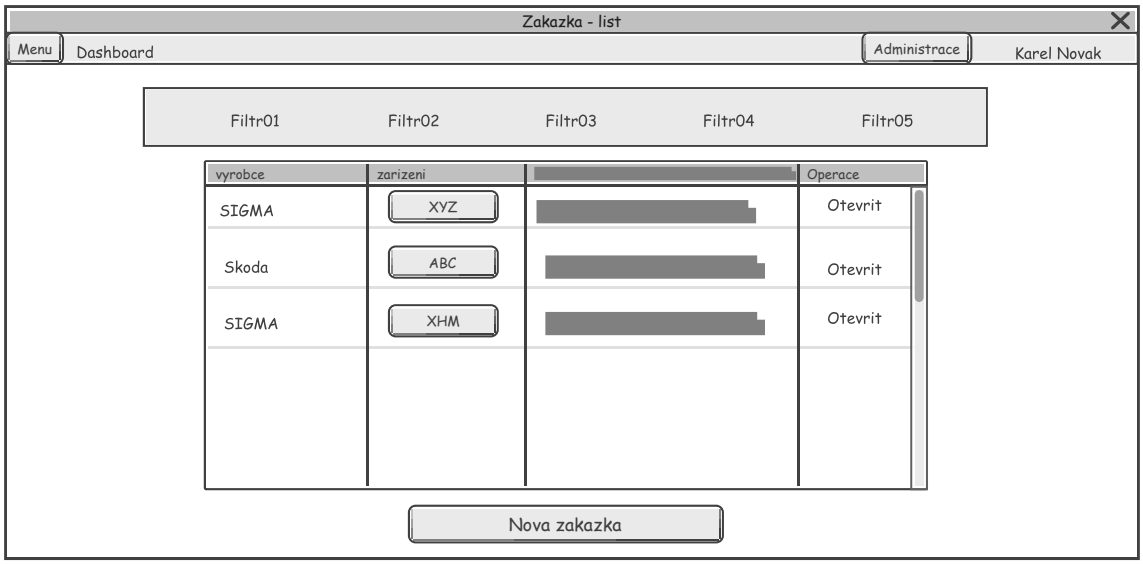
Ukázka protokolu



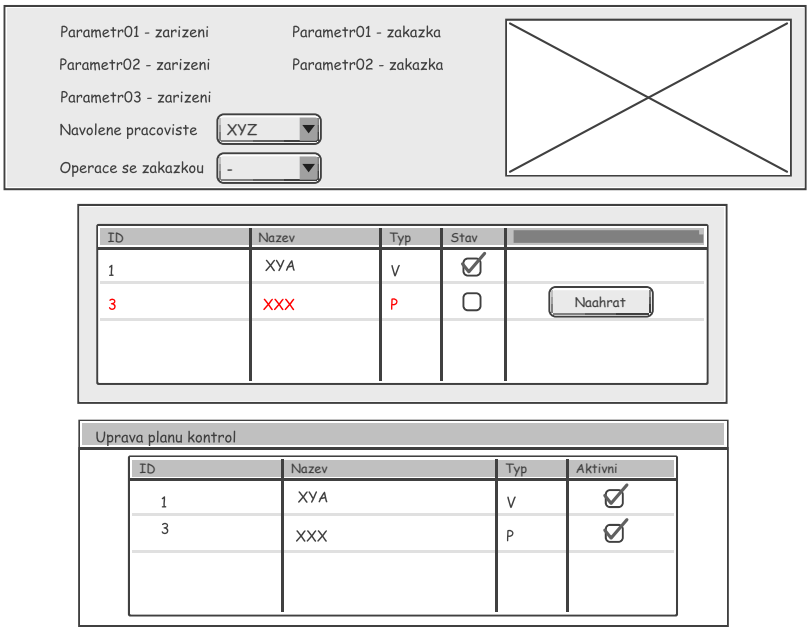
Obrázek C.1: Ukázka protokolu „Plán kontrol a zkoušek“

Příloha D

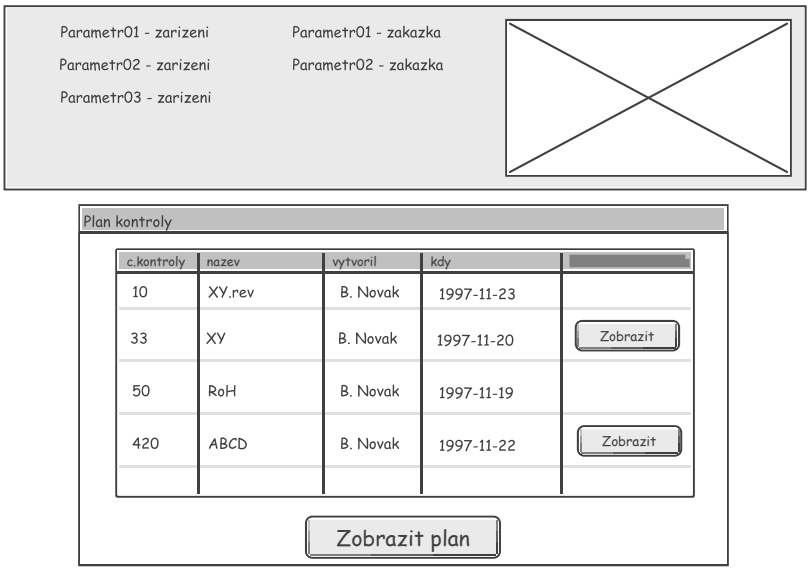
Drátěné modely



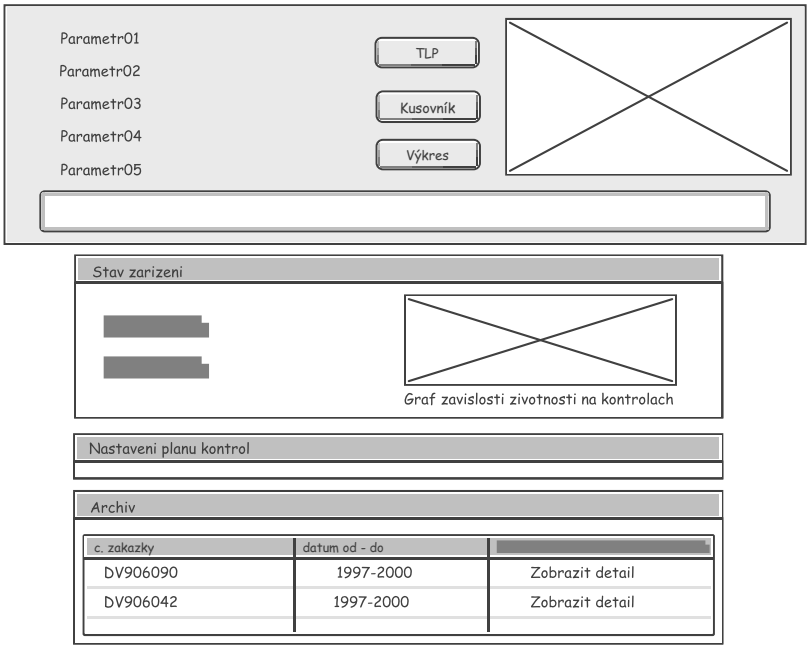
Obrázek D.1: Pohled na přehled zakázek. V horní části je hlavní menu, které obsahuje tlačítko pro zobrazení vysouvacího postranního menu, odkaz na zobrazení úvodního pohledu (přehled zakázek), odkaz pro přejití do administrace aplikace a jméno přihlášeného uživatele. V těle dokumentu je zobrazen filtr, který odfiltruje zakázky podle různých parametrů (stav zakázky, druh zařízení, na kterém je zakázka prováděna, ...). Odfiltrované zakázky jsou zobrazeny v tabulce. Novou zakázku lze vytvořit pomocí tlačítka *Nova zakazka*. Zobrazit si konkrétní zařízení lze kliknutím na požadované zařízení



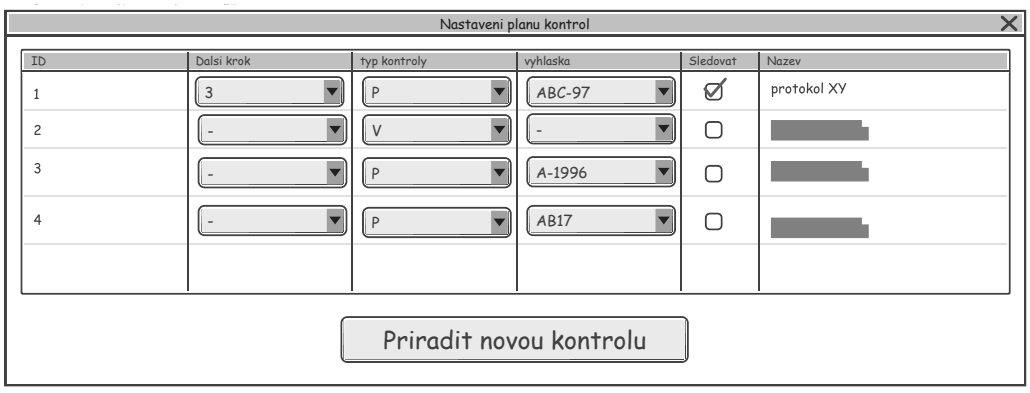
Obrázek D.2: Pohled na otevřenou zakázku (bez hlavního menu). Na modelu je několik sekcí. První sekce zobrazuje informace o zařízení, obrázek zařízení a informace o zakázce. V této sekci, si zaměstnanec může navolit pracoviště a provést se zakázkou příslušní operace (např.:zrušit, uzavřít). V druhé sekci je seznam kontrol, které se musí v zakázce provést. Zaškrtávací pole reprezentuje stav kontroly, zda byla kontrola provedena či nikoliv. Červeně jsou zobrazeny nevypracované protokoly. V poslední sekci se generuje přednastavený plán kontrol, který lze ještě upravovat.



Obrázek D.3: Pohled na uzavřenou zakázku (bez hlavního menu). Na modelu jsou zobrazeny dvě sekce. První sekce zobrazuje informace o zařízení, obrázek zařízení a informace o zakázce. V druhé sekci jsou zobrazeny kontroly, které se provedly v rámci zobrazené zakázky. Je možnost i zobrazit plán dané zakázky.



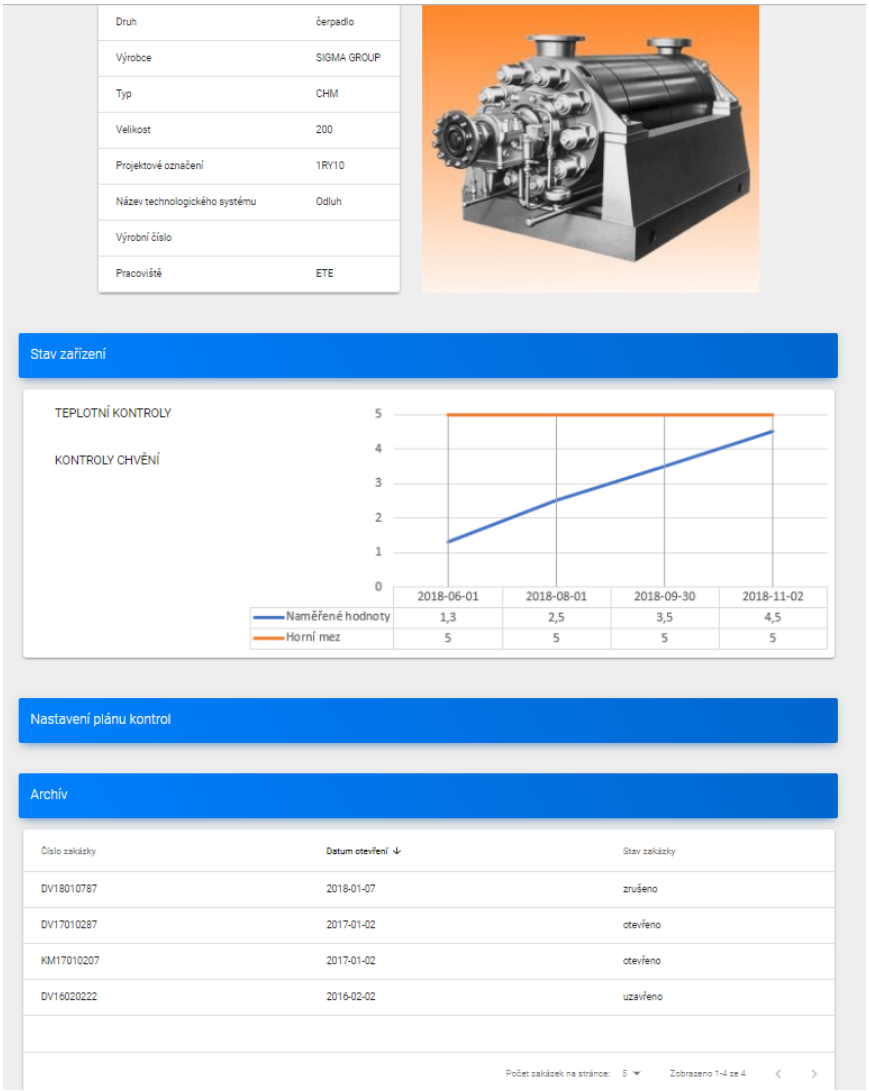
Obrázek D.4: Pohled na zařízení (bez hlavního menu). Na modelu je několik sekcí. První sekce zobrazuje informace o zařízení a jeho obrázek. Je možné zobrazit *Kusovník* a další dokumenty přiřazené k zařízení. V druhé sekci je seznam sledovaných parametrů a k nim odpovídají graf závislosti životnosti na provedených kontrolách. V třetí sekci se k zařízení přiřazují protokoly, které se máji na daném zařízení vypracovávat. V poslední sekci je nastavení plánu kontrol, který lze vidět na obrázku D.5. Poslední sekce zobrazuje zakázky, které se na zobrazeném zařízení povádělý nebo provádějí.



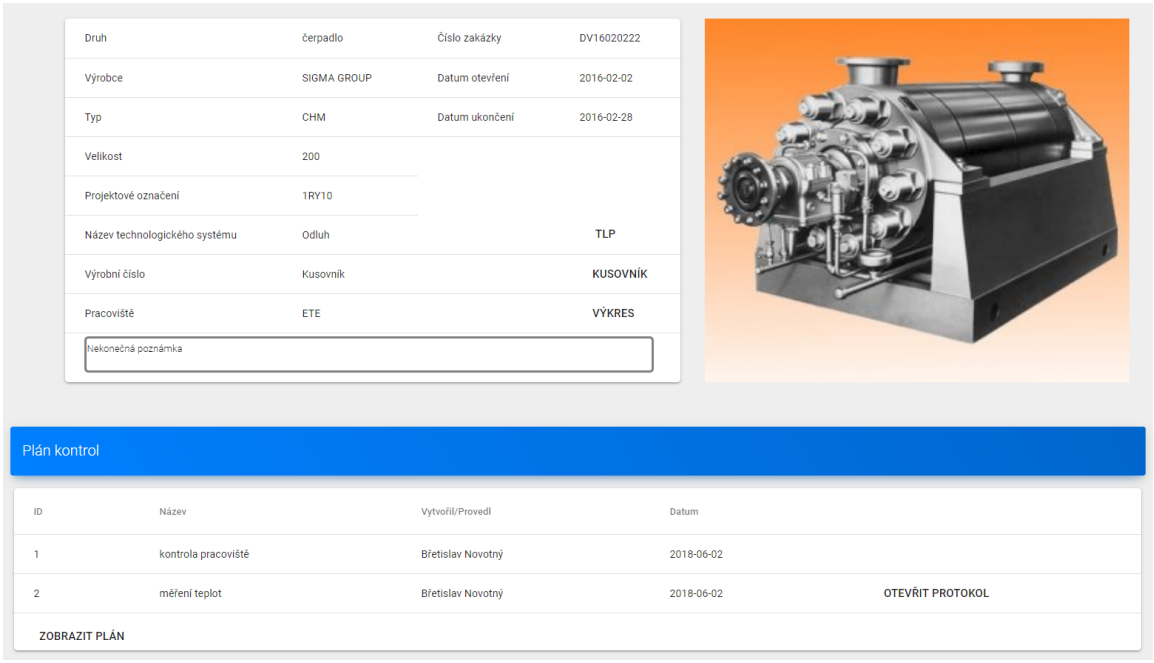
Obrázek D.5: Pohled na komponentu pro nastavení plánu kontrol. Na modelu je tabulka, která reprezentuje plán kontrol. Novou kontrolu lze přidat pomoci tlačítka *Přiřadit novou kontrolu* a následně se vygeneruje nový řádek tabulky, který se vyplní.

Příloha E

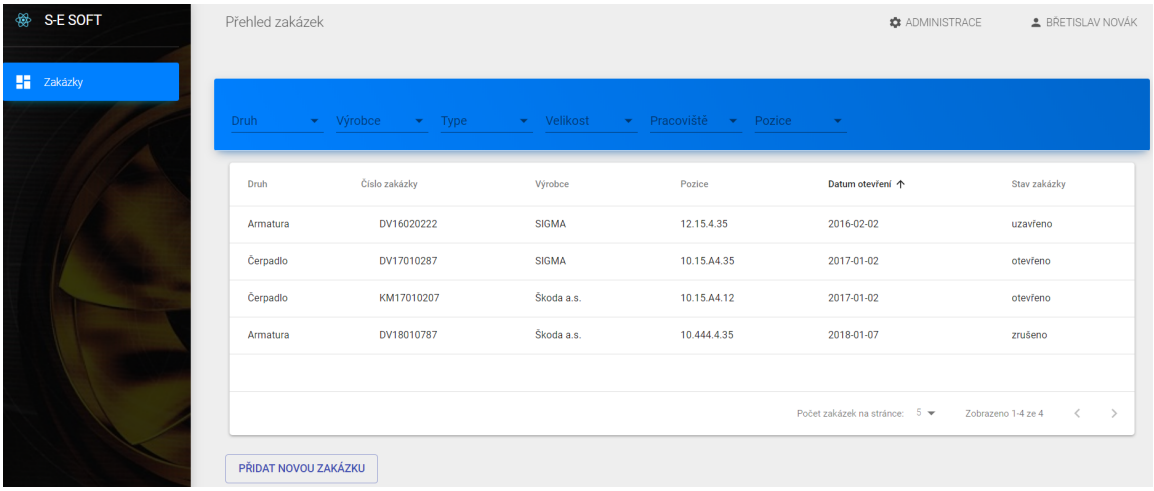
Prototypy



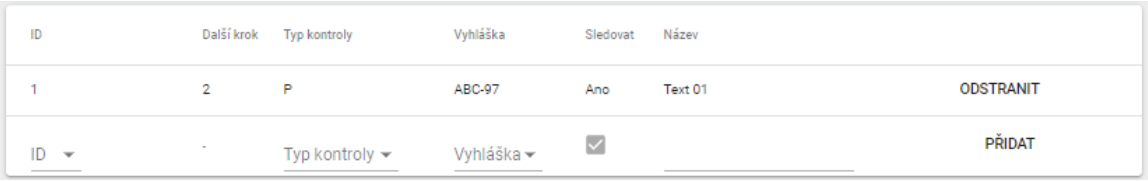
Obrázek E.1: Pohled na celou stránku zobrazující kartu zařízení



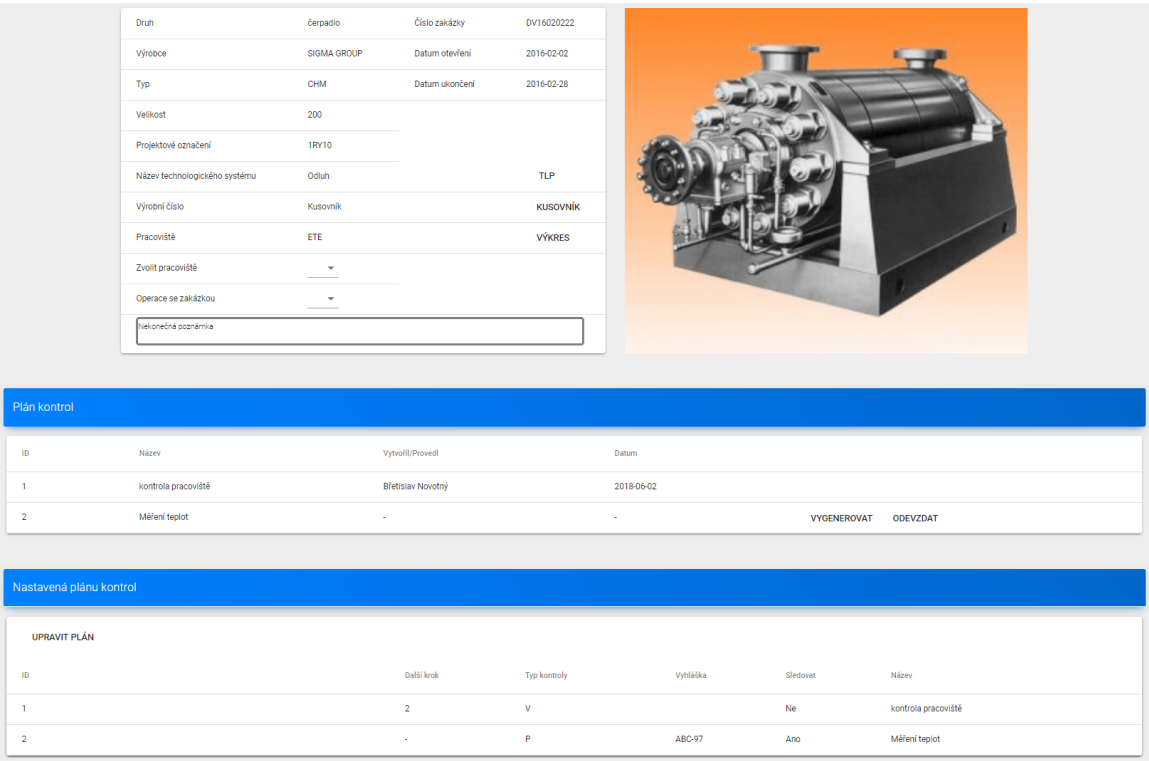
Obrázek E.2: Pohled na celou stránku zobrazující informace o uzavřené zakázce



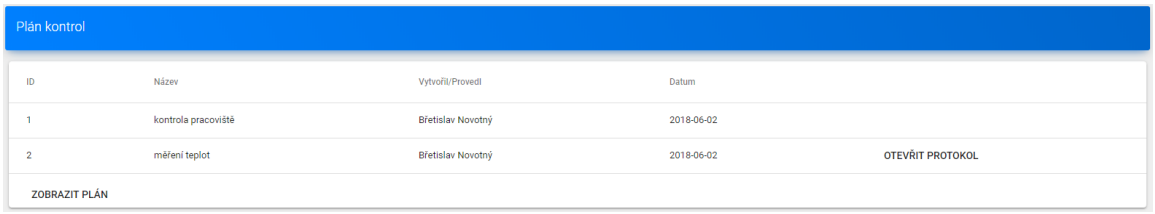
Obrázek E.3: Pohled na přehled zakázek. Pohled je založený na modelu znázorněný na obrázku D.1. Pro přechod na detail zařízení, stačí kliknout na konkrétní zařízení ve sloupci *Pozice*. Pro zobrazení zakázky, stačí kliknout na číslo požadované zakázky. Tabulku lze filtrovat, řadit a listovat mezi výsledky. Postranní menu obsahuje v horní části jméno programu, kde při kliknutí na něj se dostaneme na přehled zakázek. Níže je tělo menu.



Obrázek E.4: Sekce pro tvorbu plánu kontrol u zařízení. Karta zařízení, do které patří tato sekce je zobrazena na obrázku E.1. Nová kontrola se přidá tak, že se vyplní poslední řádek. Pokud se zvolí *typ kontroly* hodnota „P“ (P jako protokol), tak se zobrazí nabídka protokolů, které jsou uloženy v databázi šablon protokolů a na základě výběru se šablona přiřadí ke kontrole. Po kliknutí na tlačítko *Přidat*, se přiřadí nová kontrola na poslední místo v seznamu. Při kliknutí na *Odstranit*, se odpovídající kontrola odstraní. Pořadí přidaných kontrol lze změnit operace „Drag and Drop“.



Obrázek E.5: Pohled na otevřenou zakázku (bez hlavního menu). První sekce zobrazuje informace o zařízení, obrázek zařízení a informace o zakázce. Pohled na uzavřenou zakázku má první sekci téměř stejnou, jen chybí možnost změnit pracoviště a nelze provádět operace se zakázkou. Pohled na detail zařízení obsahuje také podobnou sekci, ale bez informací o zakázce. V druhé sekci je seznam kontrol, které se musí v zakázce provést. Pokud je třeba ke kontrole odevzdat protokol, tak v posledním sloupci je tlačítko pro stažení vygenerovaného protokolu a odevzdání protokolu. Pokud se jedná o kontrolu bez protokolu, tak se zobrazí tlačítko *Zkontrolováno*. Pokud je kontrola hotová, tak se nezobrazí žádné tlačítko nebo tlačítko pro stažení protokolu. V poslední sekci se nastavuje plán kontrol. Pořadí kontrol lze měnit pomocí operace „Drag and Drop“, a to přetažením řádku v tabulce. Nové kontroly lze přidat vybráním kontroly v plánu kontrol pro zařízení.



Obrázek E.6: Druhá sekce pohledu na uzavřenou zakázku. Zobrazuje kdy a kdo kontrolu provedl. Pokud ke kontrole byl přiřazen protokol, tak se v posledním sloupci zobrazení tlačítko pro stažení protokolu. Pod tabulkou, je tlačítko pro zobrazení plánu zakázky. Celkový pohled je zobrazen na obrázku E.2.

Příloha F

Rest API serveru

/api/kind	CRUD operace s entitou *Kind*
/api/producer	CRUD operace s entitou *Producer*
/api/type	CRUD operace s entitou *Type*
/api/type/assignType?id=ID	Operace C pro přiřazení velikosti k typu ID
/api/type/assignType	Operace RD pro práci s přiřazenými velikostmi k typu
/api/size	CRUD operace s entitou *Size*
/spi/workspace	Operace CRUD s entitou *Workspace*
/spi/workspace/user	Pouze operace CD pro práci s přiřazenými uživateli k entitě *Workspace*
/api/position	CRUD operace s entitou *Position*
/api/position/state?id=ID	Operace R pro získání stavu zařízení ID
/api/history	CRUD operace s entitou *History*
/api/history/plan	Operace R pro získání seznamu kontrol v zakázce
/api/history/plan/finish ?id=ID&itemId=ITEM	Operace U pro dokončení kontroly ITEM u zakázky ID
/api/history/plan/generate ?id=ID&itemId=ITEM	Operace R pro vygenerování protokolu ITEM v rámci zakázky ID
/api/history/plan/plan-generate	Operace R pro vygenerování plánu kontrol
/api/history/plan/upload	Operace C pro odevzdání vyplněného protokolu nebo plánu kontrol
/api/protocol	Pouze operace R pro získání všech dostupných šablon protokolů (entita *ProtocolTemplate*)
/api/decree	Pouze operace R pro získání všech dostupných entit *Decree*
/api/controlType	Pouze operace R pro získání všech dostupných entity *ControlType*

Tabulka F.1: Rozhraní implementované na serveru (Operace C=Create, R=Read, U=Update a D=Delete

Příloha G

Obsah CD

Přiložené CD obsahuje:

- **DIP.pdf**: text této práce ve formátu `pdf`,
- **README.md**: návod na instalaci aplikace,
- **src/**: adresář obsahují zdrojové soubory aplikace,
- **src-router/**: adresář obsahující rozšíření pro framework Nette o anotaci `@ApiRoute`,
- **src-tex/**: adresář obsahující zdrojové LaTeX soubory této práce.