Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Webová API aplikace pro správu reklamačních dat z JSON serveru společnosti VSP DATA

Pavel Vydra

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE
Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
Pavel Vydra

Informatika

Název práce
Webová API aplikace pro správu reklamačních dat z JSON serveru společnosti VSP DATA

Název anglicky
Web API application for controlling warranty data from JSON server of company VSP DATA

Cíle práce

Bakalářská práce je tematicky zaměřena na problematiku a vytvoření aplikace pro přenos dat mezi klientem a společností, poskytující záruční služby VSP DATA. Hlavním cílem je zprovoznit a uživatelsky zpříjemnit zobrazování a správu reklamačních a servisních protokolů. Dílčí cíli jsou:

- analyzovat fungování komunikačního JSON serveru
- analyzovat způsob propojení a komunikace JSON serveru s php
- vytvořit možnost přidávání reklamačních případů
- vytvořit možnost aktualizace reklamačních protokolů
- vytvořit správu cenových nabídek a jejich schvalování
- vytvořit možnost zobrazení přesunu součástek
- vytvořit zobrazení stavu skladů součástek a jejich využití
- zprovoznit výpis chyb
- zabezpečit přenos pomocí bezpečnostního tokenu

Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze jednotlivých odborných zdrojů. Vlastní řešení je realizováno formou studia jednotlivých bodů a promítnutí nabytých znalostí do vývoje jednoduchého software pro danou činnost. Na základě syntézy teoretických poznatků a výsledků vlastního řešení budou formulovány závěry bakalářské práce.

Doporučený rozsah práce

35-40 stran

Klíčová slova

php, JSON, jQuery, HTML, css, API, transformace dat, správa reklamací, webová aplikace

---

Doporučené zdroje informací

Beginning JSON,Springer, Berlin, 2015. EAN: 9781484202036
Brooks, David R., Guide to HTML, Javascript and PHP, 2011. 428 s.ISBN: 9780 857 2944 87 Eric Meyer,Eric Meyer o CSS – Kompletní průvodce, 2007. 560 s. ISBN: 978-8086815-64-0 Jonathan Chaffer , Mistrovství v jQuery,2013. 384 s. EAN: 9788025141038
PHP knihovna, Client URL Library. Odkaz: http://php.net/manual/en/book.curl.php

---

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 21. 2. 2017

Ing. Martin Pelikán, Ph.D.
Vedoucí katedry

Elektronicky schváleno dne 21. 2. 2017

Ing.Martin Pelikán, Ph.D
Děkan

V Praze dne 23. 02. 2017

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Webová API aplikace pro správu reklamačních dat z JSON serveru společnosti VSP DATA" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 14.3.2016 _____

Poděkování

Rád bych touto cestou poděkoval panu doktoru inženýru Jiřímu Brožkovi za vedení mé práce a cenné rady při zpracovávání problematiky. Dále bych rád poděkoval společnosti elem6 s.r.o. za kooperaci při vývoji aplikace a za její testování.

Webová API aplikace pro správu reklamačních dat z JSON serveru společnosti VSP DATA

Souhrn

Tato práce se zabývá problematikou API, neboli propojení dvou různých systému a realizací řešení na daný problém. Konkrétně se jedná o server společnosti VSP DATA, která poskytuje smluvní záruční servis společnostem, prodávajícím elektroniku, a jeho komunikaci s uživatelem (společností). Řešení propojuje serverová data s uživateli za pomocí různých technologií. Využíváme hlavně PHP ve spojení s HTML a CSS za pomoci jQuery. Naše aplikace obsahuje většinu funkcí, které server podporuje. Vývoj probíhal podle technické dokumentace serveru. Tato aplikace byla vyvíjena jako testovací verze a později byla nasazena na testování do firemního provozu. Před zahájením vývoje byly analyzovány požadavky zákazníka, tedy firmy požadující danou aplikaci a byly vytvořeny návrhy uživatelského prostředí. Po schválení započal vývoj s průběžnými konzultacemi a testy na straně společnosti. Ukončení vývoje zatím nebylo provedeno. Aplikace se bude rozšiřovat o další funkce, jako je předpokládaný počet spotřebovaných součástek pro další období, průměrné reklamace jednotlivého zboží vzhledem k prodejům a další statistické funkce. V tuto chvíli aplikace umožňuje práci s daty reklamačních protokolů a další jež jsou zmíněny v práci.

Klíčová slova:

php, JSON, jQuery, HTML, CSS, API, transformace dat, správa reklamací, webová aplikace, přenos dat, uživatelské prostředí, analýza projektu

Web API application for controlling warranty data from JSON server of company VSP DATA

Summary

This theses is focused on problematics of API or as we can say connection between two different systems and realization of solution to that problem. Specifically it is about server of company VSP DATA which provides contractual warranty service for companies selling electronics and its communication with user (company which have the contract). This solution connects and translates the server data for user with use of different technologies. We mainly use PHP in connection witch HTML, CSS and jQuery. This application contains most functions that the server provides. The development was oriented by the server documentation. This application was developed as a test version and later was put to test process in company. Before the development started I have collected and analysed data from the customer, meaning company requiring the application, and user interface and functions behaviour were created. After approval the development started along with ongoing consultations and testing with the company. End of development has not been taken yet. Application will be extended with more functions as is expected part consumption in further periods, average reclamations of goods based on sales and reclamations and more statistic functions. At this point application allows the company to work with the data from protocols and more which are explained further in the thesis.

Keywords:
php, JSON, jQuery, HTML, CSS, API, data tranformation, reclamation manage, web application, data transfer, user interface, project analysis

Obsah

1 ÚVOD 13

2 CÍL PRÁCE, METODIKA A ÚKOLY 14

2.1 Cíl práce 14

2.2 Metodika 14

2.3 Úkoly 15

3 TEORETICKÁ VÝCHODISKA 17

3.1 Internet 17

3.1.1 WWW (world wide web) 17

3.1.2 DNS server 18

3.1.3 IP adresa 18

3.1.4 Připojení k internetu 18

3.1.4.1 Typy přípojek 18

3.1.4.2 Veřejná IP adresa 19

3.1.5 Server 19

3.1.5.1 Hardwarové typy serverů 19

3.1.5.2 Softwarové typy serverů 20

3.1.5.3 Webhosting server 20

3.1.5.4 JSON server 21

3.2 Webová aplikace 21

3.2.1 PHP 21

3.2.2 HTML 22

3.2.3 Formuláře 23

3.2.3.1 Funkčnost 23

3.2.3.2 Metody předávání dat ve formulářích 24

3.2.3.3 Rizika formulářů 25

3.2.4 CSS 26

3.2.5 jQuery 26

3.2.6 JSON 27

3.2.7 Cookies 28

3.3 Bezpečnost internetové komunikace 29

3.3.1 Bezpečnostní token 29

3.3.2 Sniffing 30

3.3.3 php injection 30

3.3.4 SQL injection 31

3.3.4.1 Příklad SQL injection v PostgreSQL 31
3.3.4.2 Příklad SQL injection jakýkoliv systém 31
3.3.5 Bezpečnost PHP formulářů 32
4 VLASTNÍ PRÁCE 33
4.1 Příprava projektu 33
4.1.1 Zpracování zadání 33
4.1.2 Termíny, způsob kontrol a testování 33
4.1.3 Způsob spolupráce 33
4.2 Struktura programu 34
4.2.1 Sumarizace použitých jazyků a technologií 34
4.2.2 Front-end 35
4.2.3 Back-end 36
4.3 Front-end 37
4.3.1 Přihlašování a odhlašování 37
4.3.1.1 Přihlášení 37
4.3.1.2 Odhlášení 38
4.3.2 Způsob zobrazování dat 39
4.3.3 Timeout 39
4.3.4 Fronta – historie hodnot 39
4.3.5 Způsob dotazování dat 40
4.4 Back-end 40
4.4.1 Formuláře 40
4.4.1.1 Funkce zajíždění 40
4.4.1.2 Paměť hodnot polí - ukládání 41
4.4.1.3 Paměť hodnot polí – načtení 41
4.4.1.4 Paměť hodnot polí – smazání 42
4.4.2 Dotazy na server a odpovědi ze serveru 42
4.4.3 Dekódování odpovědí 43
4.4.4 Historie (zásobník) 44
4.5 Zabezpečení aplikace 45
4.5.1 Htaccess 45
4.5.2 Bezpečnostní token 45
5 VÝSLEDKY A DISKUZE 46
5.1 Hodnocení uživatelů – UI 46
5.2 Hodnocení uživatelů – Přínos aplikace 46
5.3 Hodnocení uživatelů – Připomínky 46

5.4 Celkové zhodnocení zadavatele - Citace 46
6 ZÁVĚR 47
7 SEZNAM POUŽITÝCH ZDROJŮ 48
8 PŘÍLOHY 49
8.1 Příloha 1: Dokumentace k serveru VSP DATA 49
8.1.1 Introduction 49
8.1.1.1 Interface 50
8.1.1.2 Legend of doc 50
8.1.1.3 Compatibility 51
8.1.1.4 Test usage 51
8.1.2 Authorization 51
8.1.2.1 Acquire session token 51
8.1.2.2 Send requests using token 52
8.1.2.3 Close connection 52
8.1.2.4 Prolong token lifetime 52
8.1.2.5 Get user's privileges 52
8.1.3 Unit 53
8.1.3.1 List range of units 53
8.1.3.2 Get single unit 54
8.1.3.3 Create new case 58
8.1.3.4 Update case 60
8.1.3.5 Approve quotation 61
8.1.3.6 Release case 62
8.1.3.7 List quotation for approval 62
8.1.3.8 Decision price quotation 63
8.1.4 Transfer order 63
8.1.4.1 List range of transfer orders 63
8.1.4.2 Get single transfer order 64
8.1.4.3 Create new transfer order 65
8.1.4.4 Request body JSON properties: 65
8.1.4.5 Remove transfer order 66
8.1.5 Stock levels 66
8.1.5.1 List stock levels 66

8.1.6 Parts information 66
8.1.6.1 The status of parts in stock 66
8.1.6.2 The number of parts in a given status 67
8.1.6.3 Consumption of parts on case 67
8.1.6.4 Consumption of parts for a specified period 68
8.2 Příloha 2 – Prototyp UI aplikace 70
8.3 Příloha 3 – Aplikace 71

SEZNAM OBRÁZKŮ

Obrázek 1 19
Obrázek 2 19
Obrázek 3 23
Obrázek 4 30
Obrázek 5 35
Obrázek 6 - vývojový diagram 36
Obrázek 7 - proces přihlášení 37
Obrázek 8 - Proces odhlášení 38
Obrázek 9 - Zobrazení dat 39
Obrázek 10 - Po neúspěšném pokusu o přihlášení 70
Obrázek 11 - Úvodní obrazovka před příhlášením 70
Obrázek 12 – Po úspěšném pokusu o přihlášení 71

SEZNAM TABULEK

Tabulka 1 23
Tabulka 2 34

1 ÚVOD

Tento projekt vzešel z vlastní iniciativy a zájmu. Začal jsem se zajímat o PHP a HTML více a chtěl jsem se naučit další programovací jazyk. K tomuto projektu jsem se dostal úplnou náhodou, když jsem ve firmě dělal brigádu jako pomocný servisní technik. Díky tomuto projektu jsem nezískal znalost pouze PHP, ale také přehled o HTML5, CS3 a velice užitečném a zajímavém jQuery. Další zajímavosti jsem se také dozvěděl při konfiguraci samotného serveru.

Server je virtuální linuxová distribuce Ubuntu 14.01, na kterou jsem nainstaloval a nakonfiguroval služby jako je mysql, phpmyadmin, apache2, php-fastcgi, php, samba, postfix, squirellmail, ispconfig a další.

Cílem v tomto projektu je uspokojit všechny požadavky firmy včetně dokumentace a zaškolení pracovníků.

Toto téma je aktuální i dnešnímu dni, kdy se aplikace využívá a připravuje se verze 2. Většina zdrojů je aktuální a aktualizovaná v roce 2017. Technologie v tomto prostředí se pohybují velice rychle kupředu, a proto jsou internetové zdroje nejvhodnější.

Tento nástroj je důležitý pro provoz reklamačního oddělení, protože v nástroji od VSP DATA jsou pouze základní data. Nenajdeme tam například stavy součástek, předchozí reklamace a další důležité informace.

Tento projekt má, dle mého názoru, velký potenciál a je možnost aplikaci více do budoucna rozšířit.

V této práci budeme projednávat teoretická východiska, která byla k projektu zapotřebí a také zajímavé kousky programu, které jsem napsal. Práce skutečně pojednává o výtažku toho nejdůležitějšího. Podrobnější rozbor problematiky je námětem na diplomovou práci.

2 CÍL PRÁCE, METODIKA A ÚKOLY

2.1 Cíl práce

Cílem této práce je naprogramovat a zdokumentovat webovou aplikaci komunikující s JSON serverem. V praktické části je cílem naprogramovat funkční aplikaci podle požadavků ze zakázky za pomoci dokumentace poskytnuté provozovatelem serveru a dokumentace k jednotlivých programovacím jazykům. Program bude obsahovat všechny požadované funkce ze strany uživatele, tedy společnosti, která by aplikaci měla používat. V této společnosti také proběhne celý vývoj a testování.

V teoretické části bude vysvětlen vývoj aplikace po jednotlivých úsecích kódování. Budou v ní vysvětleny jednotlivé části kódu, ale také celkové fungování programu. Cílem je aby tato práce sloužila jako finální produkt hotový pro distribuci a užívání ve firmách jako balíček programu včetně podrobné technické dokumentace. V dokumentaci bude vysvětleno podrobně chování programu a účel jednotlivých funkcí.

2.2 Metodika

Tato práce je zpracovávána jako standartní vývoj aplikace. Proto jsem vybral, společně se zadávající společností, prototypový přístup řešení. Dalším důvodem bylo, že komplexnost tohoto problému je vyšší a zadání je nejednoznačné a komunikace v reálném čase se zadavatelem je obtížná. Posledním důvodem je, že nebylo ze začátku zcela jasné co řešit a pomocí čeho.

Tento problém je v praxi velmi častým například ve stavebnictví při modelování prototypů domů před jejich stavbou a postupným zaváděním změn do těchto prototypů. Dalším příkladem může být automobilový průmysl a vývoj nového modelu automobilu, kde se nejdříve určí, jak bude modelová řada vypadat a poté se udělá prototyp, na kterém se ladí nedostatky. Po „vychytání“ všech zjištěných nedostatků se přistupuje k finálnímu produktu. [1]

Při vývoji IS je rozsah a obtížnost problému srovnatelná jako u výše zmíněných případů. Hlavním kritériem je, aby se zákazníkovi předal systém, který není pomalý nebo příliš rozsáhlý. Dále je důležité, aby neměl příliš komplikované a nepřehledné uživatelské rozhraní, ale zároveň by měl splňovat všechny zadané požadavky.

1 BRUCKNER, Tomáš. Tvorba informačních systémů: principy, metodiky, architektury. 2012

Při použití prototypového vývoje se musíme vyhnout několika vážným problémům, jako jsou:

- Zadavatel projektu vyžaduje okamžité výsledky, ale přitom není ochoten financovat i ty prototypy, které neprojdou
- Zadavatel neumí správně zformulovat zadání
- Programátor podlehne sebeklamu, že problému dostatečně rozumí
- Programátor nebere v potaz zvyky a zkušenosti uživatelů, a systém pak plní požadavky, ale nevhodnou formou

Klasický postup podle MMDIS (Multidimensional Management and Development of Information Systems) je založen na sekvenčním pořadí fází. Těmi jsou:

- Úvodní studie
- Globální analýza a návrh
- Detailní analýza a návrh
- Implementace
- Zavedení

Fáze jsou součástí jednotlivých etap. Mezi fázemi se vrací zpět, avšak ne plánovaně, ale až po objevení závažného problému. Čím později je chyba odhalena, tím je projekt dražší. Celá řada chyb se objeví až po předání projektu a zaškolování jednotlivých uživatelů. V SW inženýrství se prototypem rozumí funkční, i když nekompletní model aplikace resp. Informačního systému [WHITTEN, 2005][2]

Mezi výhody použití prototypů spadá například to, že se oproti původnímu zadání přijde na funkce a požadavky, které zadavatele předtím nenapadly. Spolupráce je jednodušší, protože konzultace probíhají pravidelně na předem domluvených schůzkách. Při správném užití modelu může mít i nižší náklady než jiné metody. [3]

Mezi nevýhody patří například to, že pokud je prototyp navržen ad-hoc bez větších úvah o jeho funkčnosti, nemusí pak být výsledek přiměřený času a vynaloženým financím. Je vždy nutné před prototypem provést odhad nákladů a přínosů. Možným nástrojem odhadu je například pravděpodobnostní odhad hodnoty informace získané prototypem, který použil Boehm v [BOEHM, 1981]. Nebezpečím hladkého průběhu a změn v prototypu může přivodit zadavateli pocit, že vývoj je stejně rychlý. Tomuto klamu musíme předejít. [4]

2.3

2 BRUCKNER, Tomáš. Tvorba informačních systémů: principy, metodiky, architektury. 2012
3 BRUCKNER, Tomáš. Tvorba informačních systémů: principy, metodiky, architektury. 2012
4 BRUCKNER, Tomáš. Tvorba informačních systémů: principy, metodiky, architektury. 2012

Úkoly

Jako hlavní úkoly jsem si stanovil dodržení všech požadavků na funkčnost. Těmi podle technické dokumentace jsou:

- Authorization 8.1.2
- Unit 8.1.3
    - List range of units 8.1.3.1
    - Get single unit 8.1.3.2
    - Approve quotation 8.1.3.5
    - List quotation for approval 8.1.3.7
    - Decision price quotation 8.1.3.8
- Stock levels 8.1.5
    - List stock levels 8.1.5.1
- Parts information 8.1.6
    - The status of parts in stock 8.1.6.1
    - The number of parts in a given status 8.1.6.2
    - Consumption of parts on case 8.1.6.3
    - Consumption of parts for a specified period 8.1.6.4

Dalšími úkoly jsou:

- Analýza projektu a zadání
- Vypracování prototypů aplikace
- Ladění aplikace se zadavatelem
- Vytvoření dokumentace
- Otestování aplikace ve firemním prostředí

3 TEORETICKÁ VÝCHODISKA

3.1 Internet

Internet je síť několika milionů až miliard navzájem propojených serverů a počítačů za pomoci aktivních síťových prvků, jako jsou switche, routery, AP zařízení a jiné, ale také pasivních prvků, jako jsou například kabely (metalické, či optické), spojky, zásuvky a jiné.

Internet je postaven na takzvaných páteřních sítích, které si můžeme představit jako obrovské datové haly, plné optických kabelů a DNS serverů. [1]

3.1.1 WWW (world wide web)

Celosvětová síť world wide web je tvořen kolekcí veřejných webových stránek uložených na webových serverech, připojených k internetu a klientských zařízení, jako jsou mobilní telefony, počítače a jiná zařízení připojená k internetu. Slangově se WWW říká: „WEB“. [1]

Historie WWW sahá do osmdesátých let dvacátého století. Vývojář Tim Berners-Lee pomáhal stavět původní prototypy jádra webové technologie a zavedl název WWW. Velký rozvoj mělo prohlížení webu, takzvané „surfování“, v devadesátých letech dvacátého století. [1]

Webové technologie využívají značkovacích jazyků. Největším a nejužívanějším je HTML (Hypertext Markup Language), který byl původně zamýšlen pouze pro text, ale rozrostl se o možnost vkládání obrazů, stylů a dalších rozšíření.

Jako technologie (protokol) přenosu se využívají protokoly http a https, přičemž https je zabezpečená (šifrovaná) verze http. Tyto protokoly komunikují na portech 80 a 443.

Existují i jiné služby, které vznikly na internetu. Příkladem může být email, P2P (peer2peer) sítě známé jako torrent nebo síť DarkNet.[1]

3.1.2 DNS server

V překladu zkratka znamená „Domain Name Server“ a slouží pro překlad doménových jmen, jako je například google.com, na IP adresu. Pro fungování internetu jsou tyto servery nezbytné. Z páteří jsou rozvedeny další, menší sítě a z nich další až ke koncovým uživatelům.

3.1.3 IP adresa

IP adresa slouží pro identifikaci a komunikaci počítačů v sítích. Máme dva typy IP adres. IPv4 a IPv6.

IPv4 se zakládá na 32bitovém základu. Formát tohoto typu adresy je rozdělen na 4 bajty následovně: „xxx.xxx.xxx.xxx“, kde jednotlivé bajty (xxx) nabývají hodnot 0 až 255. Celkově je v tomto formátu dostupných 2^{32} adres. V posledních letech se přechází z IPv4 na IPv6, jelikož adresy IPv4 jsou již vyčerpány. Tato změna poznamenala z většiny jen veřejné IP adresy, zatímco se v lokálních sítích používají stále IPv4.

1 Internet. Businessdictionary.com 2017
2 WWW. Lifewire [online]. Praha: Lifewire.com, 2017

IPv6 se zakládá na 128bitovém základu. Formát je rozdělen na 8 bajtů následovně: „xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx“, kde jednotlivé bajty (xxxx) nabývají hexadecimálních hodnot 0000 až FFFF. Celkové je k dispozici 2^{128} adres.

3.1.4 Připojení k internetu

Důležité v řešení našeho problému je vědět, co znamená mít veřejně dostupný počítač nebo server. K tomu je třeba mít veřejnou IP adresu.

3.1.4.1 Typy přípojek

V případě, že si pořídíme internet u poskytovatele, dostaneme nabídky na různé typy připojení. Jsou tři hlavní typy přípojek, se kterými se můžeme setkat.

První z nich je připojení přes telefonní linku, které využívá například O2 a používá technologií ADSL, VDSL a podobných.

Druhé nejběžnější připojení je koaxiálním kabelem, které využívá například UPC. Technologie se v tomto případě CATV (linka pro kabelovou televizi).

Třetí možností je připojení bezdrátově, za pomoci aktivního síťového prvku. Tento typ je nejvíce využívaný na venkově, okrajích měst a na místech s nerozvinutou infrastrukturou na rozdíl od telefonních přípojek pro technologii ADSL, VDSL, které mají pokrytí mnohem hustší.

Nejobtížnější je pak připojení pomocí CATV, kde je zapotřebí hybridních opticko-koaxiálních sítí, které jsou pouze v centrech velkých měst.

3.1.4.2 Veřejná IP adresa

Ve chvíli pořízení služby máme na výběr mezi veřejnou a neveřejnou IP adresou.

Neveřejná adresa znamená, že váš router nebo přípojka není dosažitelná z jiného PC zapojeného do sítě internet. Adresa je dosažitelná pouze z lokální sítě poskytovatele a nelze spojit tuto adresu s doménou.

Veřejná adresa je přesný opak předešlého příkladu. Vaše vnitřní adresa je za pomoci poskytovatele zveřejněna a přesměrována tak, aby byla dosažitelná z jakéhokoliv počítače v síti internet. Na tuto adresu můžeme pak pomocí DNS záznamů přesměrovat námi zakoupenou doménu. Z toho vyplývá, že můžeme mít doma webový nebo jakýkoliv jiný server a ten můžeme zpřístupnit.

3.1.5 Server

Server je aktivní síťový prvek. Jedná se o zařízení, které je samostatný počítač se serverovým operačním systémem a poskytuje služby buď do lokální sítě, anebo do internetu. V lokální síti jde například o doménový server, souborový server a podobně. V případě služeb pro internet se užívá pro server síť s veřejnou IP adresou a poskytuje například webový hosting, VPS (Virtual Private Server) a jiné.

3.1.5.1 Hardwarové typy serverů

Servery mají dvě podoby. První podoba je rack[obr. 2]. Můžeme si ji představit jako šuplík, který zasouváme do velké skříně, mezi ostatní. Druhá podoba je tower[obr. 3]. Vypadá jako klasický stolní počítač, ale mnohem vetší. Servery mají standardně dvě patice na procesor, mnoho patic na operační paměti a podle modelu větší, či menší počet míst na disky.

Obrázek 1

Autor: Servaris – CC BY-SA 3.0, https://www.servaris.com/images/servers/e1400p-open-425.jpg

Obrázek 2

Autor: ASA Computers, Inc – CC BY-SA 3.0, http://www.asacomputers.com/images/D/2u%20rackmount%20server-front.jpg

Tento počítač může být uložen ve specializovaných prostorách (serverovnách) v podobě „rack“ počítače nebo „tower“ sestavy. Serverovny poskytují většinou stabilní a rychlé připojení k internetu, protože se nachází ve většině případů na, anebo v blízkosti páteřní sítě, kde se používá vysokorychlostního připojení optickými kabely.

3.1.5.2 Softwarové typy serverů

Každý server plní specifický účel, dle potřeby uživatele. Záleží na nainstalovaném operačním systému a softwaru. Servery mohou mít operační systém od společnosti Microsoft a to distribuci Server používané většinou ve firmách pro lokální účely. Alternativou je systém založený na jádře linux. Ty se používají pro hostování virtuálních systémů, hostingy a další služby.

3.1.5.2.1 Microsoft Server

Microsoft Server se dále rozděluje na dílčí verze. Konkrétním příkladem pak je Server 2008 R2, Server Standard / Enterprise / Datacenter 2012 a další. Rozdíly jsou ve funkcích, v podpoře hardwaru, kapacit operačních pamětí, disků a například maximálním počtem uživatelů. Ve většině případů se používá se službami Active Directory, Doménový server, DNS server, Souborový server, Aplikační server a další. Na tomto typu systému lze provozovat i hosting s podporou .NET ASP.

Prostředí je z pravidla grafické.

3.1.5.2.2 Linux server

V této sekci se dostáváme k mnoha distribucím. Příkladem jsou RedHat, Ubuntu server, SuSE enterprise server a další. Každá distribuce má své specifické výhody v podpoře různých služeb a programů a hlavně v rychlosti pro jednotlivé služby. Na těchto systémech většinou provozujeme služby pro hosting, sdílení souborů, cloudové služby, VPS a další, které jsou veřejné.

Prostředí bývá většinou voleno konzolové kvůli navýšení výkonu.

3.1.5.3 Webhosting server

Webový server je provozován jako aplikace na fyzickém serveru. V našem případě budeme řešit webový hosting. Ten je řešen principem více hostingů na jeden server. Tímto způsobem je sdílen výkon, respektive je rozdělen mezi jednotlivé hostingy. Příkladem máme server s konfigurací 96GB operační paměti, dvakrát procesor s šesti jádry, celkem tedy 12 jader (24 threadů) a 4TB diskového pole. Tyto prostředky poskytovatel rozdělí podle potřeby. Například rozdělíme na 96 hostingů. Každý dostane vyhrazenou paměť pro php, jedná-li se o linux server. Dále sdílený čas procesoru, operační paměť pro ostatní operace, a kapacitu na disku pro ukládání souborů.

Tento systém bude mít více síťových karet s připojením do sítě, pro pokrytí provozu a požadavků.

3.1.5.4 JSON server

JSON server je takový počítač, který běží obvykle na operačním systému LINUX a obsahuje službu, která komunikuje v JSON. Zmiňovaná služba je nainstalována na serveru, má přiřazený komunikační port a na vstupní požadavky vybírá záznamy z databázového serveru a ty pak následně vrátí jako odpověď na požadavek.

Server neobsahuje samostatně pouze službu pro JSON, ale obsahuje i jiné služby jako je například databázový server a webový server.

3.2 Webová aplikace

Webová aplikace je taková aplikace, která je uložena na webovém serveru, který je dostupný veřejně, anebo je dostupný pouze z interní sítě. Webová aplikace se skládá z uživatelského prostředí takzvaného „front-end“ a z programového pozadí neboli „back-end“.

Prostředí front-end je složeno z prvků HTML[3.2.2] (text, formuláře, tabulky), dále z kaskádových stylů CSS[3.2.4], skritptovacího jazyka „JavaScript“ nebo nástavby „jQuery“[3.2.5].

Programová část back-end může být naprogramována pomocí technologií jako jsou PHP, ASP .NET a jiných. V převážné většině se používá technologie PHP.

3.2.1 PHP

Php (Hypertext preprocesor) je nejužívanější open-source obecný skriptovací jazyk, který je speciálně vyvíjen pro užití a vývoj na webových stránkách a může být použit v HTML.[1] Příklad:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>

    <?php
      echo "Hi, I'm a PHP script!";
    ?>

  </body>
</html>
```

PHP. Php.net [online]. 2017

1 PHP. Php.net [online]. 2017

Díky této vlastnosti je PHP velice jednoduše schopné vkládat text do HTML kódu.

Php sekvence vždy začíná „<?php“ a končí „?>“. Php je od jiných skriptovacích jazyků jako je jQuery[3.2.5] odlišeno tím, že zpracování probíhá na straně serveru. Výhodu spatřujeme v tom, že oproti jQuery klient nevidí zdrojový kód, který na stránce běží. [1]

Oba jazyky mohou generovat dynamicky obsah stránky, ale pouze u php není vaši práci schopen nikdo zkopírovat. [1]

3.2.2 HTML

HTML je „Hyper text Markup Language“, což je možno volně přeložit jako: „odkazový značkovací jazyk“. Definuje strukturu webových stránek pomocí značek. Značky jsou elementy, neboli „stavební kameny“, stránek. Elementy jsou definovány pomocí takzvaných „tagů“. Příklady „tagů“:[2]

- Heading - <heading>
- Paragraph -
- Table - <table>

Prohlížeče tyto „tagy“ vyrenderují a použijí se na ně styly CSS[3.2.4], pokud

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

HTML. W3cschools.com [online]. 2017

existují.

Příklad HTML:

„Tagy“ v html jsou většinou párové. To znamená, že paragraf má začátek: „ “ a konec . Výjimkou je například tag img, kde je jeho syntaxe následující: „“.

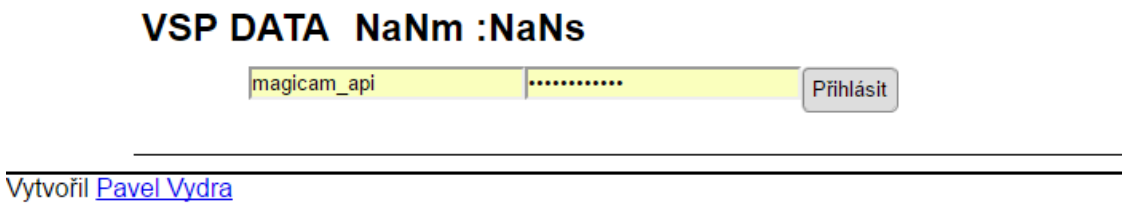
HTML umožňuje ve spojení php užití formulářů[3.2.3]. Ty slouží pro předávání vstupních dat do programové vrstvy webu.

Aktuální verze HTML je HTML5.

1 PHP. Php.net [online]. 2017
2 HTML. W3cschools.com [online]. 2017

3.2.3 Formuláře

Formulář v HTML[3.2.2] v kombinaci s PHP[3.2.1] je prezentován tagem <form action="skript, kam formulář odešle data" method="POST / GET"> a využívá se pro odesílání dat pro zpracování. Typickým příkladem je přihlašování, nahrávání souborů na server, stahování souborů ze serveru a jiné. Formulář[obr. 5] obsahuje pole pro potřebné údaje. V případě přihlašování těmito údaji jsou: přihlašovací jméno, heslo, zůstat přihlášen.



Obrázek 3
Autor: Pavel Vydra – CC BY-SA 3.0, Vlastní aplikace

3.2.3.1 Funkčnost

Data ve formuláři jsou uváděna tagem: „<input type="typ_vstupu" name="nazev_vstupu" value="hodnota_promenne"/>". Proměnná „typ_vstupu" může nabývat hodnot:

Tabulka 1

Proměnná	Hodnota
typ_vstupu	text – obyčejný text; password – pro hesla (vstupní text je nahrazen hvězdičkami); date – pro datum (umožní v prohlížeči nativní zobrazení výběrového kalendáře);checkbox – zaškrtávací pole
nazev_vstupu	Hodnota, kterou určí programátor. Přes tento název můžeme v back-end získat hodnotu (value) proměnné
hodnota_proměnné	Hodnota proměnné, která je volitelná. V případě text a password přenášíme většinou hodnotu pole a tudíž value nevyplňujeme. V případě zaškrtávacích polí můžeme hodnotu určit sami.

Ovládací prvek ve formuláři bývá z pravidla tlačítko uváděné tagem: „<button type="submit"/>Text tlačítka</button>". Tento prvek způsobí, že data budou odeslána na příslušný skript.

3.2.3.2 Metody předávání dat ve formulářích

Máme dva typy odesílání dat. Prvním je POST a druhým GET. Nyní se podíváme na hlavní rozdíly mezi těmito dvěma typy. V zásadě se jedná o bezpečnost přenosů.

3.2.3.2.1 POST

Tento způsob odesílání dat je mnohem bezpečnější, než GET. V tomto případě dochází k přenosu dat na pozadí. Co to znamená? Ukážeme si to na příkladu.

Máme stránku A, která obsahuje formulář s následující hlavičkou:

```
<form action="skriptA.php" method="POST">
    <div class="form-group">
      <p class="title">Přihlašovací jméno</p>
      <input name="login" type="text"/>
    </div>
    <div class="form-group">
      <p class="title">Heslo</p>
      <input name="pass" type="password"/>
    </div>
</form>
```

V tomto případě se otevře: Aktuální složka /skriptA.php
Proměnné získáme přes příkazy $název_proměnné = $_POST[„název_proměnné“];
URL adresa přitom neobsahuje dané parametry, které jsme odeslali.

3.2.3.2.2 GET

Tento způsob odesílání dat je mnohem méně bezpečný, než POST. V tomto případě dochází k přenosu dat za pomoci URL linku. Co to znamená? Ukážeme si to na příkladu.

Máme stránku A, která obsahuje formulář s následující hlavičkou:

```
<form action="skriptA.php" method="GET">
    <div class="form-group">
      <p class="title">Přihlašovací jméno</p>
      <input name="login" type="text"/>
    </div>
    <div class="form-group">
      <p class="title">Heslo</p>
      <input name="pass" type="password"/>
    </div>
</form>
```

V tomto případě se otevře:
aktuální složka/skriptA.php?login=hodnota_login&pass=hodnota_pass
Proměnné získáme přes příkazy $název_proměnné = $_GET[„název_proměnné“];
URL adresa přitom obsahuje dané parametry, které jsme odeslali. To není bezpečné pro přenos hesel v nezašifrovaném formátu a dalších citlivých dat.

3.2.3.3 Rizika formulářů

3.2.3.3.1 Textová pole

V případě formulářů s textovými poli odkazujících na skript s SQL dotazem musíme dávat pozor na vstupní parametry. Ověřovat je třeba například datové typy, délku, formát a další.

Pokud v poli login útočník zadá například:

```
"";
select * from configuration where 1;
```

Pokud tedy náš skript je

```
<?php
   $query  = "SELECT id, name size FROM users
         WHERE user = '$_POST[\"user\"]'";
   $result = odbc_exec($conn, $query);
?>
```

Tak se uživateli zobrazí všechny konfigurační údaje v tabulce configuration.

Ošetření provedeme tak, že uděláme test, aby v proměnné nebyly mezery a aby to byl string.

3.2.3.3.2 Soubory

V případě formulářů pro nahrávání souborů odkazujících na skript musíme dávat pozor na vstupní parametry.

Soubor nesmí být validní php kód. Ověříme pomocí eval(). Soubor by neměl obsahovat řetězec „base64“, který slouží ke kódování dat.

Nahrávání souborů je dobré omezit pouze na lokální síť. Z pomoci proměnné $_SERVER[„REMOTE_ADDR“], která obsahuje adresu IP, ze které probíhá připojení.

3.2.3.3.3 Využití metody GET

Při užití metody GET se přenáší parametry ve fyzické URL a jsou tedy čitelné pro kohokoliv, kdo poslouchá komunikaci v síti. Při přenášení údajů jako jsou hesla, musíme používat metodu POST.

V moderních systémech se používá pro komunikaci takzvaný „token“. Tento „token“ generuje systém jako zašifrovanou informaci, která obsahuje například ID připojení. Tento údaj se předává v URL a pomoci GET. To by mohl být problém, ale ověřování probíhá i na úrovni cookies, aby nemohlo dojít k ukradení relace.

1 PHP. Php.net [online]. 2017
2 HTML. W3cschools.com [online]. 2017

3.2.4 CSS

CSS je „Cascade Style Sheet“, což je v češtině přeloženo jako kaskádové styly. Je to jazyk, který definuje styl zobrazování HTML[3.2.2] elementů. Funguje na principu takzvaných „selektorů“, které vybírají prvky a přiřazují jim styl zobrazování. [1]

Příklad syntaxe:

```
body {
  background-color: lightblue;
}

h1 {
 color: white;
 text-align: center;
}

p {
 font-family: verdana;
 font-size: 20px;
}
```

CSS. W3cschools.com [online]. 2017

3.2.5 jQuery

jQuery je nástavba na JavaScript. Jedná se o knihovnu, optimalizující syntaxi JavaScriptu. Stará se o responzivitu a programovou funkčnost front-end prostředí webových stránek. Umožňuje manipulaci s daty, ovlivňování událostí na stránce, animace a také AJAX. [2]

AJAX je framework pro JavaScript, který umožňuje tvorbu interaktivních webových stránek.[2]

jQuery, podobně jako HTML[3.2.2], používá pro výběr elementů „selektory“. Selektory mají stejnou syntaxi jako v CSS[3.2.4].

Příklad jQuery syntaxe:

```
var hiddenBox = $( "#banner-message" );
$( "#button-container button" ).on( "click", function( event ) {
  hiddenBox.show();
});
```

1 CSS. W3cschools.com [online]. 2017
2 jQuery. JQuery.com [online]. 2017

JSON

JSON je „JavaScript object notation“. Používá se jako odlehčený formát pro výměnu dat. Výhodou je, že je jednoduše čitelný člověkem a zároveň jednoduše generovatelný strojem. JSON je řídí standardem Standard ECMA-262 3rd Edition - December 1999.

Jedná se o textový, na jazyce nezávislý, formát, využívající konvence dobře známé programátorům z rodiny jazyků C (C, C++, C#, Java, JavaScript, Perl, Python a další). Díky tomu je JSON univerzálním jazykem.

Je založen na dvou strukturách. Jednou je kolekce párů (název/hodnota), jako známe z polí v PHP. Druhou je seřazený seznam hodnot. Ve většině jazyků je prezentován jako pole, vektor nebo seznam.

Jedná se o univerzální strukturu, kterou podporují všechny moderní programovací jazyky.

Základem JSON je objekt, obsahující jednotlivé atributy, které mají hodnoty.

Ukázka jazyka JSON:

```
"roles": [
  {
    "name": "partnerAdmin",
    "privileges": [
      "case/create",
      "case/update",
      "case/confirm",
      "case/list"
    ],
    "domains": [
      "htc",
      "hp"
    ]
  },
  {
    "name": "partnerLogistic",
    "privileges": [
      "stock/view",
      "stock/order"
    ],
    "domains": [
      "htc"
    ]
  }
]
```

Příloha 1

3.2.7

1 JSON. JSON.org [online]. 2017

Cookies

Jako „Cookies“ jsou v kontextu webových technologií označovány informace uložené na straně klienta. Tyto informace jsou uloženy v podobě dat, nebo chceme-li hodnoty, přiřazené k určitému klíči, který je v rámci úložiště jedinečný. Pokud bychom se podívali na specifický datový formát uložení, pak se jedná o malé textové soubory. Každá uložená hodnota má svůj čas expirace. [1]

Název (identifikátor) úložiště je nastavitelný programátorem. Toto úložiště musí být stejné pro všechny subdomény a aplikace, které web obsahuje, pokud mají mít možnost navzájem ke cookies přistupovat. [1]

Cookies vznikly za účelem udržení dat i po zavření webové stránky. Proč? V případě, že navštěvujeme webový portál, pak dochází k potřebě udržení přihlášení, obsahu košíku, bezpečnostního tokenu, prohlédnutých produktů, stránek a jiných. Cookies tímto vyřešily problém jak udržet informace o klientovi pokud web opustí. [1]

Cookies se bez rozšiřujícího frameworku vytváří pomocí příkazu: „`document.cookie = "username=John Doe";`“. [1]

Cookies mají možnost nastavení expirace platnosti dat. To se provede následujícím příkazem: „`document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";`“. [1]

Pokud chceme u cookies nastavit i cestu uložení provedeme to následujícím příkazem: „`document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";`“. [1]

Čtení uložených dat probíhá příkazem: „`var x = document.cookie;`“, kde se nám vrátí string, který obsahuje všechny hodnoty. Tvar návratové hodnoty je následující: „`cookie1=hodnota; cookie2=hodnota; cookie3=hodnota;`“. Tento řetězec dekódujeme například do pole pomocí metody „parse_str“. [1]

Změny v hodnotách se provádí stejnými příkazy jako v případě vytváření nové. Můžeme tak měnit například hodnotu, nebo i expiraci. [1]

Mazání hodnoty probíhá stejnými příkazy, jako vytváření nové, jen je hodnota prázdná. [1]

V našem projektu užíváme jQuery frameworku pro práci s cookies. Název pluginu je: “jQuery Cookie Plugin v1.4.1“, který celou práci zjednodušuje pomocí metod. Více o použití ve vlastní práci.

1 JavaScript Cookies. W3schools.com/ [online]. USA 2017

3.3 Bezpečnost internetové komunikace

3.3.1 Bezpečnostní token

Bezpečnostní token v našem případě označuje zašifrovaný textový řetězec, který se používá pro komunikaci se serverem místo přihlašovacího jména a hesla. Tento systém pracuje na principu prvního ověření přihlašovacím jménem a heslem a pozdější komunikace pomocí tokenu. Token systém obdrží jako odpověď na přihlášení jménem a heslem, pokud je přihlášení úspěšné.

Tato metoda se používá z důvodu bezpečnosti přihlašovacích údajů. Zamezíme, nebo minimálně snížíme, možnost případným útočníkům, kteří poslouchají naši síť, aby jméno a heslo obdrželi.

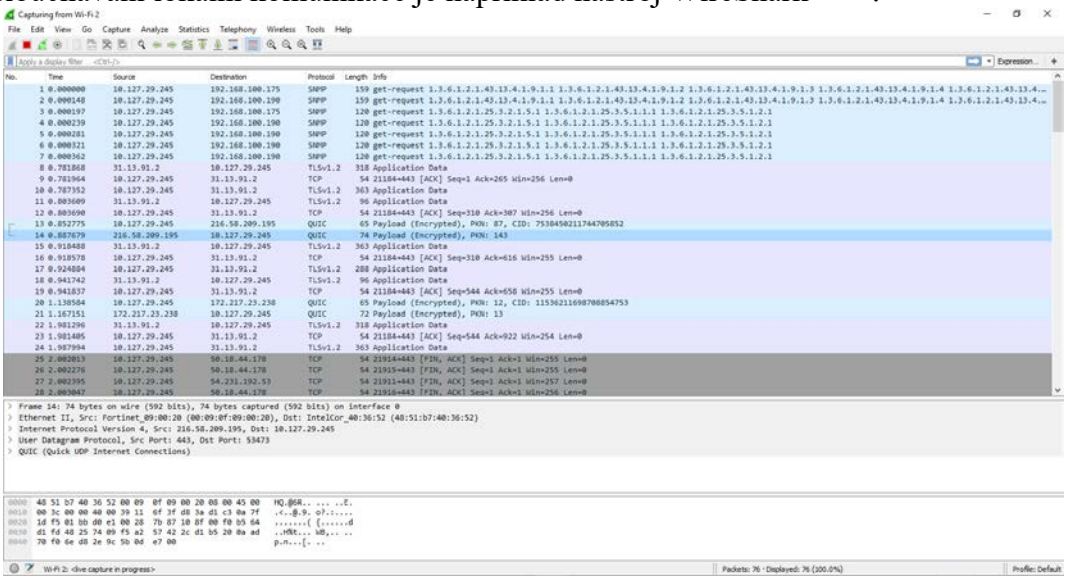
V případě, kdy bychom používali pouze přihlašovací jméno a heslo, zvyšujeme možnost úniku citlivých údajů. Při každém požadavku na komunikaci se serverem musí dojít k ověření identity Pokud toto ověření proběhne na základě pro člověka neidentifikovatelného textového řetězce máme mnohem větší šanci, že útočník neuspěje. V případě, že bychom posílali stále jméno a heslo může dojít k zachycení těchto údajů útočníkem. V kapitole sniffing [3.3.2] je tento způsob popsán.

Token má určenou svou platnost, která je uložena na serveru a podle jeho nastavení může docházet k obnovování času platnosti. Například když nám v internetovém bankovnictví běží časomíra na 10 minut. Udává to platnost současné relace, a pokud se zaměříme na hodnotu na časomíře, tak si všimneme, že při každém požadavku se časomíra obnoví na maximální hodnotu. V případě, že časomíra vyprší, může dojít k odhlášení nebo při pokusu o další požadavek nám systém nahlásí, že se musíme znovu přihlásit.

Token může mít například následující podobu: „d9da33b8135248f068b213c2aa5abbd4“.

3.3.2 Sniffing

Sniffing je způsob odposlouchávání cizí komunikace v lokální síti. Nástroj pro odposlouchávání lokální komunikace je například nástroj WireShark[obr. 4].



Obrázek 4

Autor: Pavel Vydra – CC BY-SA 3.0, Vlastní snímek obrazovky

Za pomoci této aplikace jsme schopni odposlouchávat veškerou komunikaci v síti. To znamená, že vidíme, odkud kam a jaký uživatel sítě sahá a jaké stránky otevírá.

V rukou zkušeného uživatele je možné toto provozovat nepozorovaně a získávat tak hesla a přihlašovací údaje k síťovým a webovým službám.

3.3.3 php injection

Jedná se o způsob vkládání souborů, přes špatně zabezpečené skripty php. Ve skriptu php nám chybí ověření, že soubor nepochází z cizí sítě nebo například nevyžaduje pro nahrání ověření z nadřazeného informačního systému. Dalším problémem může být, když útočník nahraje pomocí php injection škodlivý skript, který může číst konfigurační soubory, rozesílat spam, a nebo přehrát soubory index.php. Při přehrání index.php nebo html se jedná o útok „Deface“.

Nejvíce využívané metody pro tento typ útoku jsou include(), include_once(), require(), require_once(), které nahrají požadovaný soubor z parametru do kódu.

Dále jsou také zranitelná místa, kde se používá relativní cesta ../ (odkaz na nadřazený adresář). Útočník může při správném užití přistoupit téměř k jakémukoliv souboru, který je zapojen do php procesu.

Důležité je aby nahrávané soubory neprošly přes test eval(), kde se ověřuje správnost php skriptu. Pokud například soubor .png projde tímto testem jedná se o škodlivý soubor.

3.3.4 SQL injection

[2]Spousta webových vývojářů není srozuměna s tím, jak mohou být SQL dotazy zfalšovány a jak určit, zda je dotaz důvěryhodný. SQL dotazy mají možnost obejít řízení přístupů, také přeskočit standartní autentifikační a autorizační kontroly a občas mohou dotazy přistupovat k nadřazenému operačnímu systému na úrovni systémových příkazů.

Přímý SQL injection je technika kde útočník vytvoří nebo změní existující SQL dotaz k získání skrytých dat nebo k přepsání cenných dat nebo dokonce spustit příkazy na systémové úrovni, které mohou daný systém ohrozit. Tohoto se dá docílit tak, že se zkombinuje vstup uživatele se statickými parametry a tím se změní podstata funkčnosti SQL dotazu.

3.3.4.1 Příklad SQL injection v PostgreSQL

```
<?php
   $offset = $argv[0]; // beware, no input validation!
   $query  = "SELECT id, name FROM products ORDER BY name LIMIT 20
OFFSET $offset;";
   $result = pg_query($conn, $query);
?>
```

V tomto případě nemáme ošetřená data na vstupu z proměnné $argv. Tento script má provádět stránkování. To znamená, že v $argv[0] je očekáváno číslo. Tento parametr se přenáší v URL. Za pomoci urlencode() můžeme do proměnné vložit například:

```
„0;
insert                                                    into
pg_shadow(usename,usesysid,usesuper,usecatupd,passwd)
select 'crack', usesysid, 't','t','crack'
from pg_shadow where usename='postgres';“
```

Toto doplnění nám z hlavní databáze vybere údaje uživatele postgres a vytvoří uživatele crack s heslem crack, kde skupiny a práva uživatele jsou jako uživatele postgres.

3.3.4.2 Příklad SQL injection jakýkoliv systém

```
<?php
   $query  = "SELECT id, name, inserted, size FROM products
        WHERE size = '$size'";
   $result = odbc_exec($conn, $query);
?>
```

V tomto případě máme proměnnou $size, která udává velikost nějakého článku. Tato proměnná může být zaplněna hodnotou:

```
'
union select '1', concat(uname||'-'||passwd) as name, '1971-01-
01', '0' from usertable;
--
```

Tento doplněk nám zobrazí hesla všech uživatelů.

3.3.5 Bezpečnost PHP formulářů

Problematika je vysvětlena v kapitole Formuláře 3.2.3

4 VLASTNÍ PRÁCE

4.1 Příprava projektu

Projekt byl zahájen 1. 2. 2016 v Praze na základě poptávky firmy. Firma nese název elem6 s.r.o. (původně MagiCam HD SOLUTIONS s.r.o.) se sídlem Braškovská 15/308, 161 00 Praha 6, Česká republika, IČ: 24232335, DIČ:CZ24232335. Všechny konzultace probíhaly v provozovně firmy na adrese Papírenská 1, Praha 6.

4.1.1 Zpracování zadání

Na první schůzce, tedy 1. 2. 2016 byly domluveny všechny detaily o průběhu jednotlivých činností a cíle projektu. Do týdne jsem zpracoval zadání a podal první prototyp aplikace ve formě papírového návrhu a hrubého funkčního diagramu práce programu.

4.1.2 Termíny, způsob kontrol a testování

Pro tento projekt byl zvolen způsob vývoje metodou prototypů[2.2], protože byl nejvhodnější z důvodu časových a proto, že je možnost předávání ve fázích. Na projekt nebyl určen termín plánovaného dokončení.

Termíny schůzek byly stanoveny na každý měsíc k druhému dni v měsíci. Minimálně deset dní před tímto termínem bylo nutné odevzdat prototyp k testování a firma měla tedy vždy minimálně deset dní na připravení připomínek.

Na schůzkách byla vždy probrána příslušná problematika a určen další postup prací. To vše bylo vždy ústně schváleno a údaje byly také přenesené do projektového softwaru TeamWork. Do tohoto nástroje byly odevzdávány i jednotlivé prototypy aplikace se sumarizací jednotlivých provedených prací a celkového postupu.

Způsob testování byl určen při zadávání projektu. Tato procedura měla následující postup: otestování hotových funkcí -> otestování nových funkcí -> vytvoření a odeslání připomínek.

4.1.3 Způsob spolupráce

Spolupráce a tvorba programu probíhala zdarma včetně konzultací. Řešení projektu a připomínek probíhalo s oddělením IT a příslušnými budoucími uživateli aplikace. Uživatelům aplikace bylo přiděleno otestování programu jednou za dva měsíce, aby se odladilo i uživatelské prostředí a také, aby došlo k odstranění chyb, které IT odborník oproti standartnímu uživateli nevidí.

4.2 Struktura programu

Struktura programu byla založena na prvním předaném prototypu a byla postupně vyvíjena až do finální podoby, jakou má dnes. Obecně se skládá z uživatelského prostředí zvaného: „Front-end“ a aplikační vrstvy: „Back-end“. Program byl navrhován pro co nejjednodušší orientaci i pro člověka, který aplikaci nepsal. Programová vrstva má logickou strukturu.

4.2.1 Sumarizace použitých jazyků a technologií

Technologie využívané v aplikaci: viz Tabulka 2.

Front-end	Back-end
HTML [3.2.2]	PHP [3.2.1]
CSS [3.2.4]	jQuery [3.2.5]
jQuery [3.2.5]	JSON [3.2.6]
	Cookies [3.2.7]

Tabulka 2

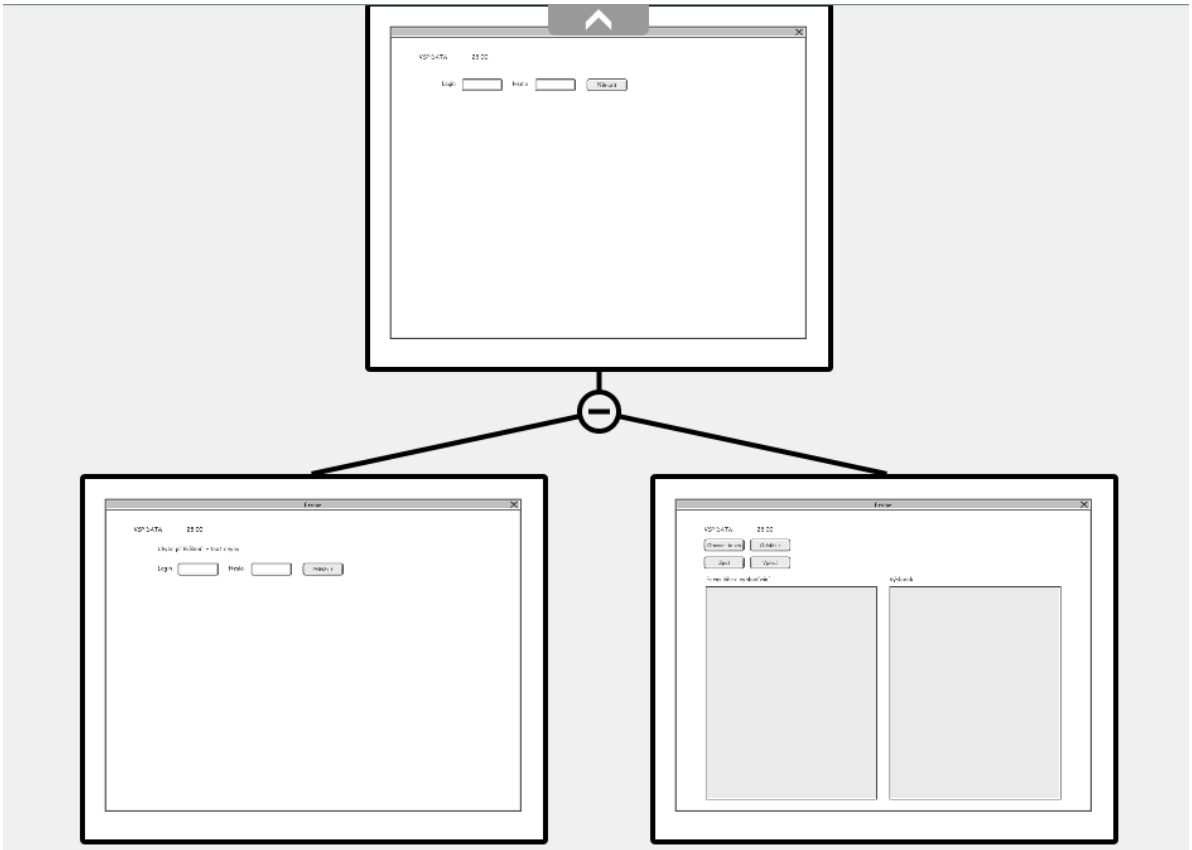
Technologie využívané na serveru: viz Tabulka 3.

Server	
Operační systém	Jádro: LINUX, Distribuce: Ubuntu 14.04
Služba webového serveru	Apache2
PHP	5.4 + Fast-cgi mod
Poskytovatel hostingu	WEDOS
Konfigurační prostředí virtuálního PC	UNIX konzole
Konfigurační prostředí hostingu	ISP Config 3

Veškerá konfigurace serveru proběhla mnou a byla úspěšně provedena. Vysvětlení postupů konfigurace a další podrobnosti jsou námětem na další práci. V této práci se tím zabývat nebudeme.

4.2.2 Front-end

Front-end se u naší aplikace skládá především z HTML[3.2.2], CSS[3.2.4] a prvků jQuery[3.2.5]. Design uživatelského prostředí byl navrhnut následovně:



Obrázek 5

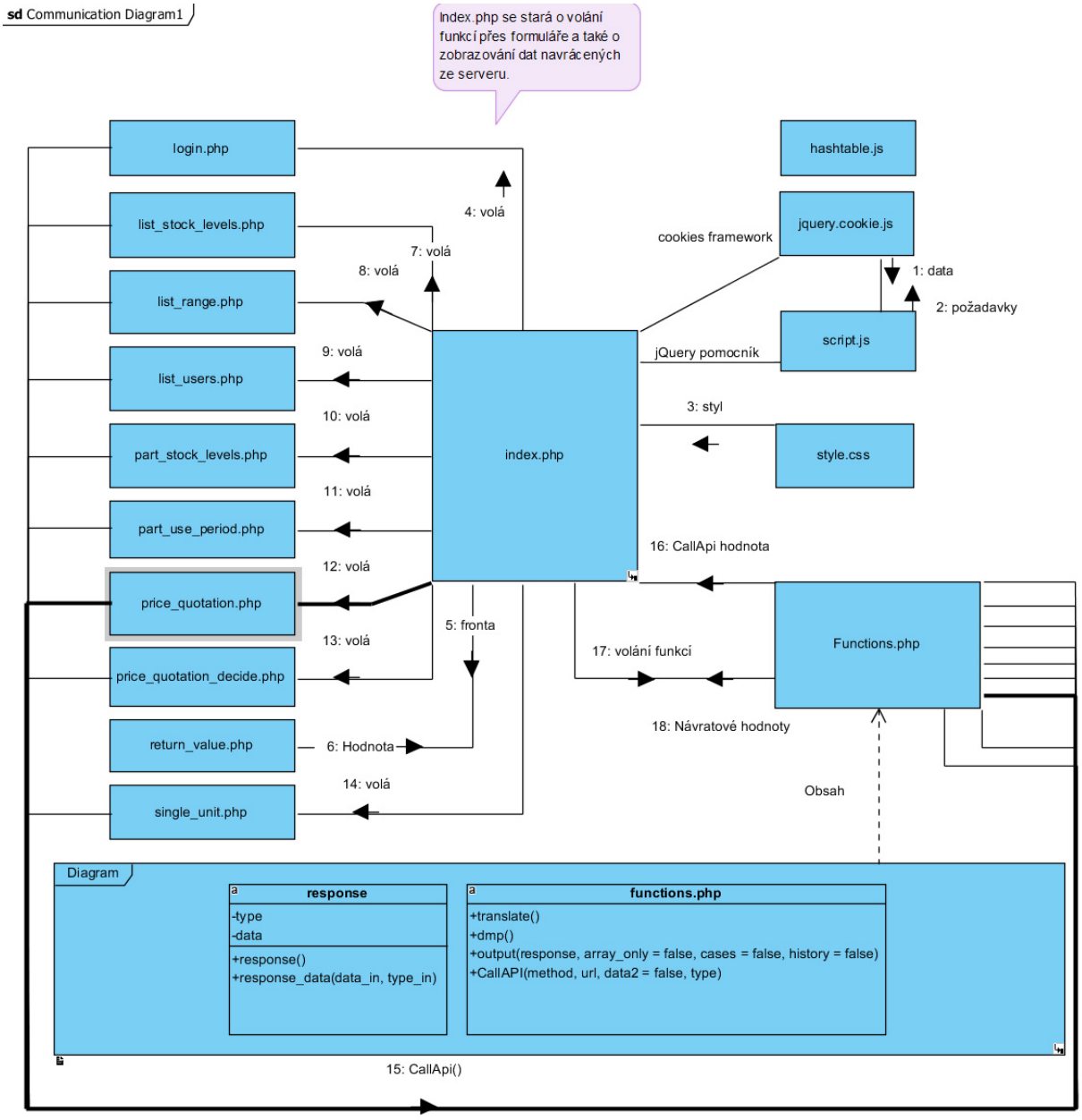
Detailnější náhled prototypu prostředí aplikace viz příloha 2[8.2]

Front-end funguje jako ovládací panel pro programovou část. Spouštíme přes něj dotazy a příkazy, které vrací nějaké hodnoty a ty zde zobrazujeme. V naší aplikaci tvoří front-end jediná stránka, která dynamicky mění svůj obsah. Na toto zobrazování nebylo použito žádných frameworků ani technologie AJAX.

Prostředí bylo navrženo tak, aby vyhovovalo funkčním požadavkům a hlavně, aby bylo jeho užívání příjemné pro uživatele.

4.2.3 Back-end

Programová část aplikace má několik souborů, které se starají o funkcionalitu. Síťový diagram struktury viz obr. 6.



Obrázek 6 - vývojový diagram

4.3 Front-end

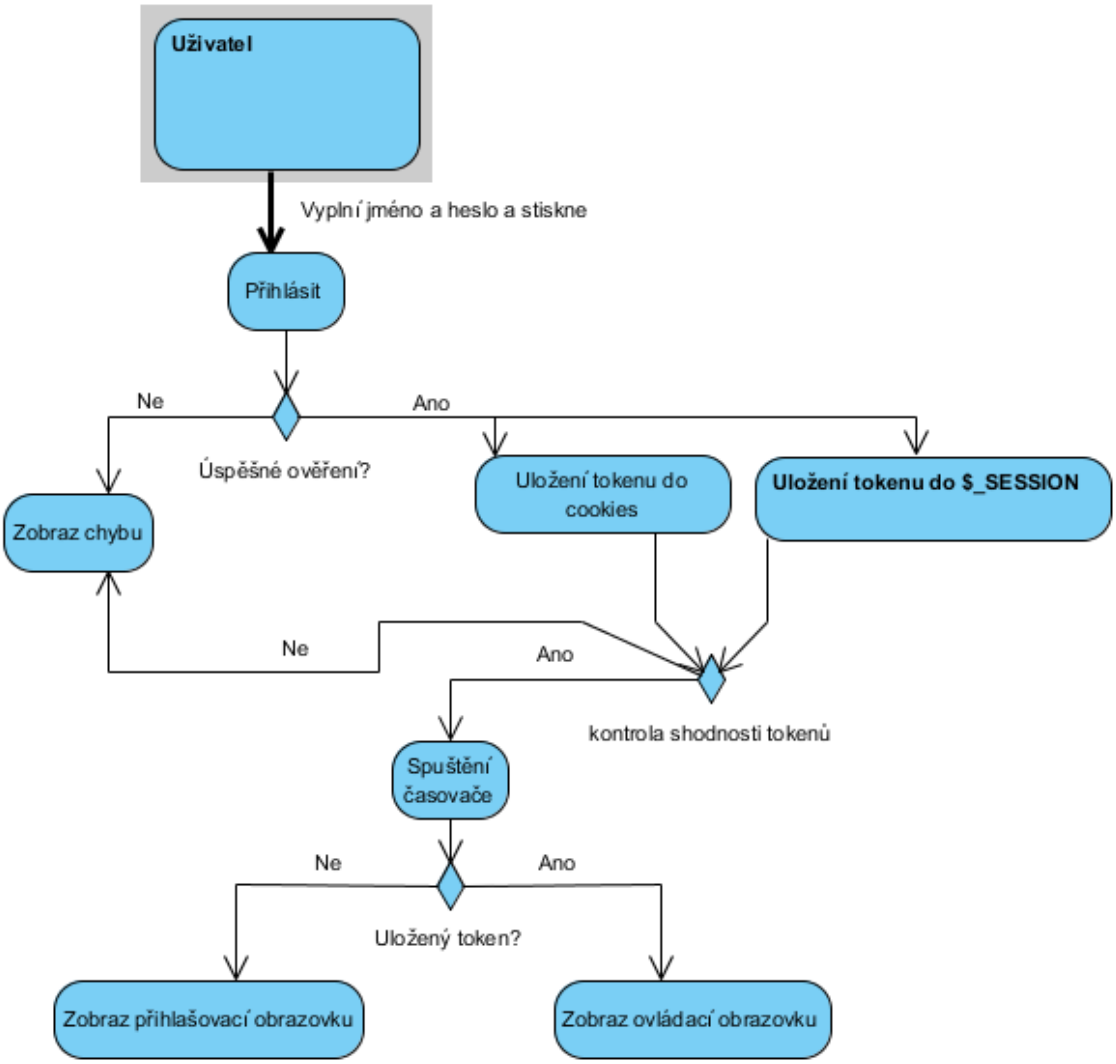
Jak již bylo řečeno, sekce front-end se zabývá zobrazováním výstupů uživateli a přijímáním příkazů od uživatele. V tomto bodě se podíváme podrobně na jednotlivé procedury.

4.3.1 Přihlašování a odhlašování

Přihlašování v naší aplikaci má dvě fáze. V první fázi dochází k odeslání přihlašovacího jména a hesla na server a vrací se nám jako odpověď bezpečnostní token, který je vygenerovaný serverem. V druhé fázi tento token uložíme a pro další požadavky používáme jako autorizaci právě ten.

4.3.1.1 Přihlášení

Proces v detailním pohledu probíhá následovně:

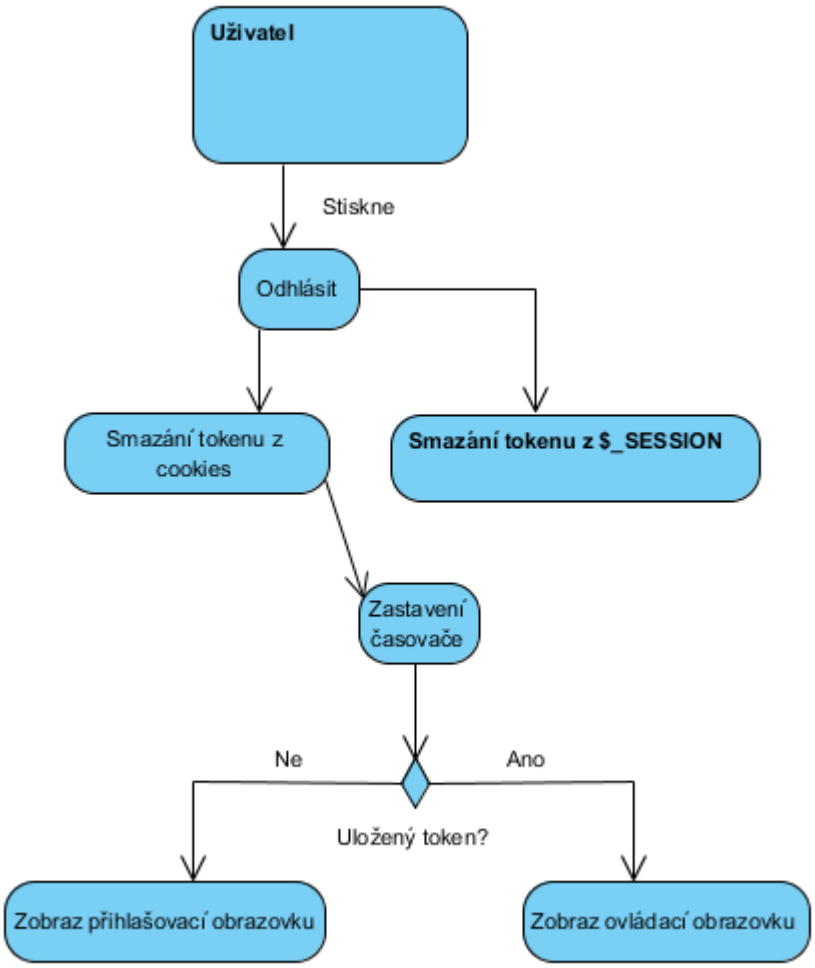


Obrázek 7 - proces přihlášení

4.3.1.2 Odhlášení

Proces odhlášení proběhne smazáním tokenu z cookies a z $_SESSION. Dále se nastaví cookie login na false.

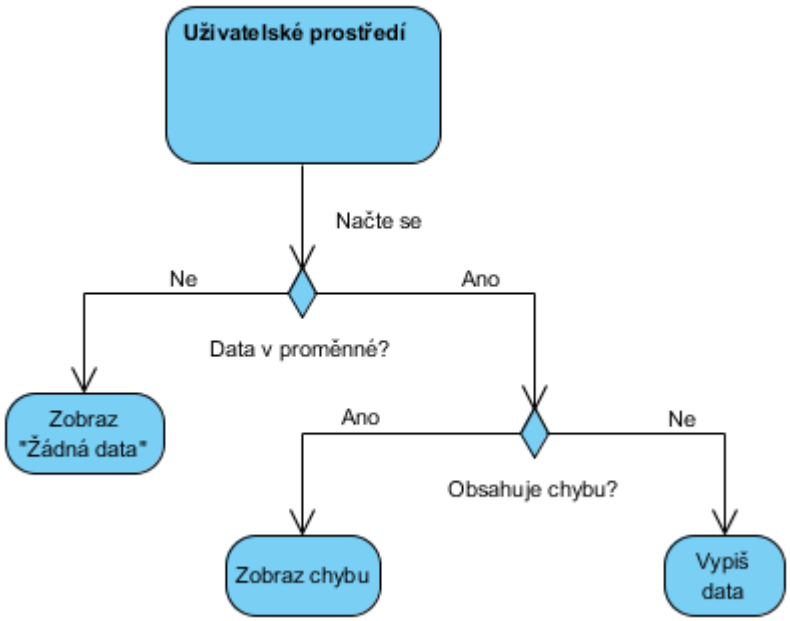
Odhlášení probíhá následovně:



Obrázek 8 - Proces odhlášení

4.3.2 Způsob zobrazování dat

Do uživatelského prostředí se data vypisují při každém načtení stránky. V proměnné $_SESSION["curl_response"] máme uloženou návratovou hodnotu z dotazu, která se vypíše zavoláním metody output. Proces zobrazování dat je následující:



Obrázek 9 - Zobrazení dat

4.3.3 Timeout

Funkčnost časového limitu platnosti tokenu je založena na informaci z dokumentace serveru[8.1.2.1.1]. Jedná se konkrétně o dvacet pět minut, po které je token platný. Při každé operaci se serverem dochází k obnovení tohoto časového limitu.

Po vypršení dochází ke spuštění procesu odhlášení[4.3.1.2].

4.3.4 Fronta – historie hodnot

V naší aplikaci byl problém, jak zobrazovat předchozí hodnoty získané ze serveru. Vzhledem k tomu, že existuje jediná funkční stránka, na kterou se dynamicky zobrazují data, není se kam vracet. Tento problém jsem vyřešil vytvořením vlastní historie v proměnné $_SESSION[„history“], kam se ukládají všechna data do pole za sebe V proměnné $_SESSION[„current_index“] máme uloženou aktuální pozici v zásobníku dat. Zásobník má omezenou velikost na 10.

Přepínaná probíhá pomocí tlačítek zpět a vpřed viz Obrázek 10[8.2].

4.3.5 Způsob dotazování dat

Po úspěšném přihlášení proběhne zobrazení dotazovacích formulářů. Tyto formuláře jsou zobrazené v levé části obrazovky. Pro každou funkci je speciální formulář, který volá specifickou funkci.

Volání probíhá podle diagramu viz Obrázek 7 [4.3.2].

4.4 Back-end

Programová část naší aplikace je rozložena podle diagramu viz Obrázek 7 [4.3.2].

V tomto bodě si popíšeme jaké funkce přesně program má a jak fungují vnitřní procedury. Struktura je spletitá a probereme zde nejzajímavější body programu.

4.4.1 Formuláře

Formuláře jsou hlavně částí front-end prostředí, ale v tomto případě jsem v jQuery doprogramoval paměť na nastavené hodnoty v textových polích, stejně tak jako nastavení zobrazení jednotlivých částí, které mají funkci zajíždění.

4.4.1.1 Funkce zajíždění

```
$("a.showMore").click(function(e){
e.preventDefault();
if($(this).parent().children(".hidden").css("display")=="none"){
    $(this).parent().children(".hidden").slideDown();
            $.cookie($(this).parent().attr("id"),true);
    }
    else{
            $(this).parent().children(".hidden").slideUp();
            $.removeCookie($(this).parent().attr("id"));
 }
```

V této funkci zabráníme přesměrování po kliknutí na odkaz s třídou „showMore“ a zjišťujeme, jestli je sekce zobrazená, či nikoli. Pokud zobrazená je, tak ji skryjeme jQuery animační metodou „slideUp“, která animuje zajetí prvku nahoru a následné zneviditelnění. V opačném případě metodou „slideDown“ se prvek znovu animací zobrazí vyjížděje směrem dolů.

4.4.1.2 Paměť hodnot polí - ukládání

Všechny hodnoty se na kliknutí odeslání formuláře uloží do cookies. Algoritmus je nastavený na ukládání všech hodnot, což znamená, že při odeslání formuláře „A“ se uloží změny i z formulářů „B“ a „C“.

Při kliknutí na odeslání formuláře dojde k cyklu, který použije název každého vstupu jako klíč pro cookies a jeho hodnotu jako hodnotu.

```
$("form.main button").click(function(){

$(this).parent().children("div").children("div").children("input").each(function()
{
        $.cookie($(this).attr("id"),$(this).val());
    });

$(this).parent().children("div").children("div").children("select").each(function(
){
        $.cookie($(this).attr("id"),$(this).val());
```

4.4.1.3 Paměť hodnot polí – načtení

Při načtení stránky dochází zároveň i k načtení hodnot z cookies pro všechny položky formulářů.

```
$("form.main input").each(function(){
    var cookie = $.cookie($(this).attr("id"));
    if(cookie != "")
        $(this).val(cookie);
  });
  $("form.main select").each(function(){
    var cookie = $.cookie($(this).attr("id"));
    if(cookie != "")
        $(this).val(cookie);
  });
```

4.4.1.4 Paměť hodnot polí – smazání

Při stisknutí tlačítka vymazat hodnoty u formuláře se hodnoty příslušného formuláře odstraní z cookies a stránka se znovu načte.

```
$(".clear_cookies button").click(function(e){
    e.preventDefault();

$(this).parent().parent().children(".main").children("div").children("div").childr
en("input").each(function(){
        $.removeCookie($(this).attr("id"));
    });

$(this).parent().parent().children(".main").children("div").children("div").childr
en("select").each(function(){
        $.removeCookie($(this).attr("id"));
    });
    location.reload();
});
```

4.4.2 Dotazy na server a odpovědi ze serveru

Komunikace se serverem probíhá pomocí knihoven pro php curl. Jedná se o technologii podobnou jQuery AJAX.

Tato knihovna odešle data v podobě parametrů v odkazu na požadovanou adresu a vrátí zpět zobrazená data, která příslušná stránka vypíše. V našem případě vypíše vzdálená aplikace data strukturovaná v JSON.

Existují 4 typy dotazů:

- POST – odeslání dat a získání návratové hodnoty
- GET – získání návratové hodnoty
- PUT – vložení dat na server a získání potvrzení nebo chyby
- DELETE – odstranění dat a získání potvrzení

Zde je zobrazeno základní volání pomocí curl a uložení návratové hodnoty do proměnné curl_response.

```
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
curl_setopt($ch, CURLOPT_USERPWD, "auth:$token");
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

$curl_response = curl_exec($ch);
curl_close($ch);
```

4.4.3 Dekódování odpovědí

Odpovědi mají vlastní třídu nazvanou „response“. Obsahuje typ odpovědi a konkrétní hodnotu. Strukturu má následují:

```
class response{
   var $type;
   var $data;
   function response() {
      $this->type = NULL;
      $this->data = NULL;
   }
   function response_data($typein,$datain){
      $this->$type=$typein;
      $this->$data=$datain;
   }
}
```

O jejich dekódování do zobrazitelné formy se stará metoda „output“, která se řídí podle typu odpovědi. Metoda „output“ je rekurzivní.

Metoda je nejdelší z celého projektu, proto je níže pouze její hlavička.

```
function output($response,$array_only = false,$cases = false,$history = false){
   if(!$array_only){
      switch($response->type){
```

4.4.4 Historie (zásobník)

Zásobník je tvořen hodnotami z předešlých dotazů a to maximálně 10 dotazů zpět. Pokud vyhledáváme po jedenácté tak se zásobník posune a první hodnotu ztrácíme. Pro posun v zásobníku slouží následující kód:

```
switch((int)$_GET["control"]){
   case 0:
      if($_SESSION["current_index"]-1 >= 0 )
         $_SESSION["current_index"]--;
      break;
   case 1:
      if($_SESSION["current_index"]+1 < sizeof($_SESSION["history"]) )
         $_SESSION["current_index"]++;
      break;
   case 2:
      $_SESSION["current_index"] = 0;
      break;
   case 3:
      $_SESSION["current_index"] = sizeof($_SESSION["history"])-1;
      break;
   case 4:
      $_SESSION["history"] = null;
      $_SESSION["current_index"] = null;
      break;
   case 5:
      break;
   case 6:
      break;
   case 7:
      break;
   default:
      break;

}
$_SESSION["curl_response"]                                    =
```

Dochází zde k přepínání proměnných pomocí historie uložené v $_SESSION a jejím přiřazení do aktuální zobrazované proměnné, která je taktéž uložena v $_SESSION.

4.5 Zabezpečení aplikace

Tato aplikace má dva typy zabezpečení.

4.5.1 Htaccess

Prvním je ověřování přistupující IP adresy, který je omezen na přístup pouze z intranetu. Toto ošetření je provedeno pomocí „.htaccess“. Vypadá následovně:

```
order deny,allow
deny from all
allow from 62.168.46.116
```

4.5.2 Bezpečnostní token

Další součástí bezpečnostní politiky je použití bezpečnostního tokenu. Postup a princip využití je podrobně popsán v kapitole „Bezpečnostní token“ [3.3.1].

5 VÝSLEDKY A DISKUZE

Uživatelům aplikace byl přidělen dotazník na vyplnění a zde jsou výsledky. Dotazník vyplnilo 32 respondentů z 32. Byl vyplňován písemně.

5.1 Hodnocení uživatelů – UI

90,6% (26) respondentů hodnotilo aplikaci jako uživatelsky příjemnou a jednoduchou. 100% z těchto respondentů bylo mladší 60-ti let.

9,4% (6) respondentů hodnotilo aplikaci jako uživatelsky náročnější, ale srozumitelnou. 100% z těchto respondentů bylo starších 60-ti let.

5.2 Hodnocení uživatelů – Přínos aplikace

100% (32) respondentů hodnotilo aplikaci jako přínosnou a užitečnou pro další užití.

96,875% (31) respondentů hodnotilo aplikaci jako potřebnou. 3,225% respondentů hodnotilo jako nepotřebnou.

5.3 Hodnocení uživatelů – Připomínky

- Vylepšit UI design – barvy + animace
- Export do XML / XSL
- Odeslání mailu zákazníkovi

5.4 Celkové zhodnocení zadavatele - Citace

Dle našeho pohledu jste splnil všechny předpokládané požadavky. Uživatelé jsou spokojení s aplikací. Do budoucna určitě uvažujeme o další spolupráci.

6 ZÁVĚR

Tento projekt lze na základě stanovených cílů společně s dosaženými cíli hodnotit jako úspěšný. Byl přínosem nejen pro firmu, ale také pro mě jako pro zpracovatele díky nabytým znalostem.

Uvažujeme o rozšiřování této verze o další funkce, jak již bylo zmíněno v cílech práce. Statistické moduly jsou aktuálně ve vývoji.

Dalším projektem je verze 2, která je restrukturalizována z pohledu back-end. V této práci uvažujeme o administračním prostředí pro vytvoření uživatelů a přidělení práv na jednotlivé operace. Dále se bude jednat o redesign front-end směrem k dnes hodně využívanému „material design“. Nová verze je zatím ve fázi příprav a plánování. Dále je na pořadí implementace tohoto nástroje do IS firmy.

Jak vyplývá z výsledků dotazníků, firma je s výsledkem spolupráce spokojena a to včetně zaměstnanců. Oproti zadaným funkcím bylo zapracováno 50% funkcí navíc. Například to je historie dotazů nebo paměť na hodnoty v polích.

Tento projekt byl dobrou výzvou a motivací pro získání nových zkušenosti z praxe a získání kontaktů ve firemní sféře. Dále mi pomohl rozšířit si obzory v projektovém plánování a profesionální spolupráci.

7 SEZNAM POUŽITÝCH ZDROJŮ

BRUCKNER, Tomáš. Tvorba informačních systémů: principy, metodiky, architektury. Praha: Grada, 2012. Management v informační společnosti. ISBN 9788024741536.

MMDIS. Management mania [online]. Praha: ManagementMania.com, 2016 [cit. 2017-02-25]. Dostupné z: https://managementmania.com/cs/mmdis

WWW. Lifewire [online]. Praha: Lifewire.com, 2017 [cit. 2017-02-25]. Dostupné z: https://www.lifewire.com/history-of-world-wide-web-816583

Internet. Businessdictionary.com [online]. Fairfax, VA, USA: businessdictionary.com, 2017 [cit. 2017-02-25]. Dostupné z: http://www.businessdictionary.com/definition/internet.html

PHP. Php.net [online]. USA: php.net, 2017 [cit. 2017-02-25]. Dostupné z: http://php.net/manual/en/

HTML. W3cschools.com [online]. USA: w3cschools.com, 2017 [cit. 2017-02-25]. Dostupné z: https://www.w3schools.com/html/

CSS. W3cschools.com [online]. USA: w3cschools.com, 2017 [cit. 2017-02-25]. Dostupné z: https://www.w3schools.com/css/

jQuery. JQuery.com [online]. USA: jquery.com/, 2017 [cit. 2017-02-25]. Dostupné z: https://jquery.com/

JSON. JSON.org [online]. USA: json.org, 2017 [cit. 2017-02-25]. Dostupné z: http://www.json.org

JavaScript Cookies. W3schools.com/ [online]. USA: w3schools, 2017 [cit. 2017-03-03]. Dostupné z: https://www.w3schools.com/js/js_cookies.asp

8 PŘÍLOHY

8.1 Příloha 1: Dokumentace k serveru VSP DATA

8.1.1 Introduction
8.1.1.1 Interface
8.1.1.2 Legend of doc
8.1.1.3 Compatibility
8.1.1.4 Test usage
8.1.2 Authorization
8.1.2.1 Acquire session token
8.1.2.2 Send requests using token
8.1.2.3 Close connection
8.1.2.4 Prolong token lifetime
8.1.2.5 Get user's privileges
8.1.3 Unit
8.1.3.1 List range of units
8.1.3.2 Get single unit
8.1.3.3 Create new case
8.1.3.4 Update case
8.1.3.5 Approve quotation
8.1.3.6 Release case
8.1.3.7 List quotation for approval
8.1.3.8 Decision price quotation
8.1.4 Transfer order
8.1.4.1 List range of transfer orders
8.1.4.2 Get single transfer order
8.1.4.3 Create new transfer order
8.1.4.4 Remove transfer order
8.1.5 Stock levels
8.1.5.1 List stock levels
8.1.6 Parts information
8.1.6.1 The status of parts in stock
8.1.6.2 The number of parts in a given status
8.1.6.3 Consumption of parts on case
8.1.6.4 Consumption of parts for a specified period

8.1.1 Introduction

VSP API is provided by HTTP REST interface.

API is available on domain http://api.vspdata.cz.

8.1.1.1 Interface

8.1.1.1.1 Response

Every response that contains body will send data in **JSON** format. This means that the *Content-type* header will contain *application/json*.

After success the server returns **200 OK** or **201 Created** HTTP status code. Otherwise some of **4xx or 5xx** HTTP codes are returned.

When the request was successful and there is no data to be returned according to the nature of the request, the success message will be available in **"message"** property.

When requesting a list due to certain conditions and no data met the conditions, the **204 No Content** response with no body is returned.

The body of unsuccessful response will contain error message in **"error"** property.

8.1.1.1.2 I/O data types

Base types **bool**, **int**, **string**.
float: Decimal number with a dot as a decimal mark.
date: Format "YYYY-MM-DD".
datetime: Format "YYYY-DD-MM HH:MM".
<type>[]: That means the array of inputs of the declared type. Eg. **string[]** means array of strings (**["first string", "second", "third"]**).
object: Object followed by its properties description in deeper level.

8.1.1.1.3 Ranges

When requesting a range of items based on date range, there will be two input parameters: **<dateSince>** and **<dateTo>**. Use the **date** or **datetime** format. When **<dateTo>** is only in **date** format, it will be considered as is at time *23:59:59*.

The result JSON will contain properties **"totalCount"**, **"pageLimit"** and **"pagesCount"**. If the **"pagesCount"** is bigger than 1, you may need include argument **<page>** into your request if you want to list older units, and/or include argument **<pageLimit>** to customize units per page limit (default is 30, maximal number allowed is 200).

8.1.1.2 Legend of doc

In whole API documentation there are certain marks in URL and headers examples that has special meaning. They are the following:

< and **>** encapsulates custom parameters, e.g.: **<unit_number>**
[and **]** encapsulates parts that are optional to be present, e.g.: **[since=2014-05-15]**

If the request contains body, there will be JSON properties description list in this doc. Additionally there will be JSON Schema file for download, that describes required structure of the JSON body. The description list is structured as follows:

(data_type) **"property_name":** [required] description

That is for example:

(string) **"partNumber":** Material number.

The [required] label means that property must be present in JSON request. Otherwise the property is optional and must not be necessarily present or it may have a value of *null*. If the object have no label [required] it is optional even though its properties does have the label.

8.1.1.3 Compatibility

The API keeps the backward compatibility, but request or response body can be extended with new properties in future. Be aware of this.

8.1.1.4 Test usage

For testing and development usage, append header **X-Connect-To: dev** and the request will be send to testing database that contains testing data. So that is the sandbox where you can try everything without worry.

8.1.2 Authorization

8.1.2.1 Acquire session token

Every request must be authorized by user's **login** and **password** that the user uses to sign in Business zone on VSP Data Services web site.

Because sending the real password with every request is dangerous, there is a section that lets you to acquire an **authorization token** to be used with consequent requests. Let make a request:

8.1.2.1.1 Request

URL: **/authorization**
METHOD: **POST**
Header: **Authorization: Basic <credentials>**

<credentials> is base64 encoded string in format **login:password**. In PHP with cURL it can be written as:

```
// fill $login and $password variables
curl_setopt( $ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC );
curl_setopt( $ch, CURLOPT_USERPWD, "$login:$password" );
```

For more information see RFC 2616 chapter 14.8.

Token lifetime is 25 minutes. If there is no request using this token received during last 25 minutes, token will expire. After that, new session must be opened.

Note that token is granted for and only for the requester IP address.

8.1.2.1.2 Example response

```
{
   "token": "75b4e474279f60aa81cbcb3609148eed1662173d",
   "dev": false
}
```

The **"dev"** property says whether request was sent to testing database according to `X-Connect-To` header (for your check).

8.1.2.2 Send requests using token

After you have got a fresh token, lets call custom requests and add following header into each one:

Header: `Authorization: Basic <credentials>`

Where `<credentials>` means base64 encoded string `auth:<token>`. As before, in PHP:

```
// $token contains freshly acquired token
curl_setopt( $ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC );
curl_setopt( $ch, CURLOPT_USERPWD, "auth:$token" );
```

8.1.2.3 Close connection

After your work has been done, it is recommended to close the connection. It will make your token useless anymore, so it cannot be used again.

8.1.2.3.1 Request

URL: `/authorization`
METHOD: `DELETE`

You must normally authorize as described in section 2.2.

8.1.2.4 Prolong token lifetime

If you need to use connection longer then 25 minutes but you know that there will be longer deaf interval, you may refresh token lifetime by something like "ping" or "heartbeat". Use following request:

8.1.2.4.1 Request:

URL: `/authorization`
METHOD: `PUT`

8.1.2.5 Get user's privileges

To get to know what privileges the authenticated user has, you can ask by following request:

8.1.2.5.1 Request

URL: `/authorization`
METHOD: `GET`

8.1.2.5.2 Example response


```
"roles": [
  {
    "name": "partnerAdmin",
    "privileges": [
      "case/create",
      "case/update",
      "case/confirm",
      "case/list"
    ],
    "domains": [
      "htc",
      "hp"
    ]
  },
  {
    "name": "partnerLogistic",
    "privileges": [
      "stock/view",
      "stock/order"
    ],
    "domains": [
      "htc"
    ]
  }
]
```


8.1.3 Unit

Unit means single repair case.

To get data about unit, you need to be permitted to list that unit. It is assigned by system administrator.

8.1.3.1 List range of units

8.1.3.1.1 Request

URL: `/unit?since=<dateSince>[&to=<dateTo>][&status=<status>][&updatedOnly][&search=<search>][&customer=<customer>][&page=<page>][&pageLimit=<pageLimit>][&createdByMe]`
METHOD: `GET`
example: `/unit?since=2014-08-14&status=RW&customer=ABC+Electro`

Units are ordered from newest.

You can limit the result to list only units with certain status (or statuses), by adding the `<status>` parameter into URL.

If you provide an extra parameter *updatedOnly*, you will get only units that has changed their status since your last request about the units. **Notice that this means, that next result for this request will differ from previous, because every request restarts "updated" flags on the units.**

To filter **case number**, **identifier** or **serial number** by some characters, use `<search>` parameter, to filter customer (means **name** in delivery address), use

parameter <customer>. Note that these parameters are accepted only with at minimum 4 characters length.

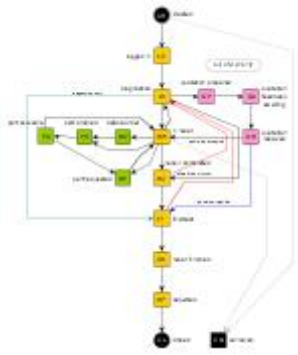
You may filter only units that was created by you by appending parameter *createdByMe*.

Server will return JSON object with property **"units"** that contains array of units that met the conditions. Each unit will contain the data of case **id**, **case number**, partner's company **identifier**, **serial number**, **IMEI number** and the current **status**. See example:

8.1.3.1.2 Example response


```
{
   "count": 2,
   "totalCount": 2,
   "currentPage": 1,
   "pagesCount": 1,
   "pageLimit": 30,
   "units": [
      {
         "id": 123789,
         "caseNumber": "HTC0123456",
         "vendor": "HTC",
         "identifier": "12345",
         "serialNumber": "ABC012345678",
         "imei": "123456789012345",
         "status": "RW"
      },
      {
         "id": 123456,
         "caseNumber": "HTC6543210",
         "vendor": "HTC",
         "identifier": "12346",
         "serialNumber": "DEF012345678",
         "imei": "543210987654321",
         "status": "PO"
      }
   ]
}
```


The **"status"** property will contain two characters long string (ore several ones separated with comma) expressing the position in workflow schema:



8.1.3.2 Get single unit

More detailed information about single unit can be obtained as follows:

8.1.3.2.1 Request

URL: /unit/<caseNumber>
URL: /unit/<id>
URL: /unit?serialNumber=<serialNumber>
URL: /unit?imei=<imeiNumber>
URL: /unit?identifier=<identifier>
METHOD: GET
example: /unit/HTC6543210

8.1.3.2.2 Example response


```
{
   "id": 123456,
   "caseNumber": "HTC6543210",
   "vendor": "HTC",
   "identifier": "123456",
   "serialNumber": "DEF012345678",
   "imei": "543210987654321",
   "serialNumberNew": null,
   "imei2": "356102050573520",
   "imeiNew": null,
   "imei2New": null,
   "status": "PO",
   "productNumber": "99ABC012-00",
   "productName": "Tablet ABC 1",
   "warranty": "IWV",
   "symptomCode": "F123",
   "symptomCodeDescription": null,
   "symptomDescription": "Customer's defect description...",
   "errorCodes": [
      {
         "code": "E456",
         "description": "Defective keyboard"
      }
   ],
   "fund": 111.5,
   "accessories": [
      "headphones",
      "battery"
   ],
   "actionProposal": "Customer requests repair.",
   "visualAppearance": "Appearance of the device upon receipt...",
   "diagnosticMessage": "Message about diagnosis of the device...",
   "engineersMessage": "Engineer's repair result message...",
   "result": "REP",
   "repairCode": "F456",
   "repairCodeDescription": null,
   "repairLevel": 2,
   "shipmentIn": {
      "number": "01234567890123",
      "courier": "DPD"
   },
   "shipmentOut": {
      "number": null,
      "courier": null,
      "price": null
   },
   "purchaseDate": "2014-08-15",
   "claimDate": "2014-10-01",
   "shipmentInDate": "2014-10-01",
   "registeredDate": "2014-10-01",
   "receivedDate": "2014-10-03",
   "inProgressDate": "2014-10-04",
   "quotationDate": null,
   "quotationAcceptedDate": null,
   "finishedDate": null,
   "departedDate": null,
   "closedDate": null,
   "delivery": {
      "name": "Acme co.",
      "street": "Hollywood 14",
```

```
      "zip": "900 00",
      "city": "L.A.",
      "country": "US",
      "identificationNumber": "12345678",
      "VAT": "CZ12345678",
      "phone": "+1 555-1234",
      "fax": "+1 555-1234",
      "email": "acme@example.com",
      "contactPerson": "Wile E. Coyote",
      "customerProfile": ""
   },
   "billing": {
      "name": "Acme co. Bank",
      "street": "Hollywood 15",
      "zip": "900 00",
      "city": "L.A.",
      "country": "US",
      "identificationNumber": "12345678",
      "VAT": "CZ12345678",
   },
   "history": [
      {
         "id": 112999,
         "caseNumber": "HTC6543211",
         "type": "pair",
         "registeredDate": "2014-10-01"
      },
      {
         "id": 123456,
         "caseNumber": "HTC6543210",
         "type": "last",
         "registeredDate": "2014-10-01"
      }
   ],
   "material": [
      {
         "partNumber": "77ABC987-00X",
         "partName": "LCD",
         "serialNumber": null,
         "pieces": 1,
         "status": "RW",
         "beforeOrderDate": "2014-09-25",
         "orderDate": "2014-09-25",
         "receivedDate": "2014-09-29"
      },
      {
         "partNumber": "77DEF654-00Y",
         "partName": "Front speakers",
         "serialNumber": "XYZ-123",
         "pieces": 1,
         "status": "PO",
         "beforeOrderDate": null,
         "orderDate": null,
         "receivedDate": "2014-09-30"
      }
   ],
   "work": [
      {
         "type": "VD001"
         "name": "Diagnostics"
      },
      {
         "type": "VDP02"
         "name": "Work 1 hour"
      }
   ]
}
```


"warranty" can reach values:

value	warrantor	in warranty	description
IWV	vendor	YES	In Warranty of Vendor
OOW	customer	NO	Out Of Warranty
CID	customer	NO	Customer Induced Defect
RER	service	YES	Repeated Repair
null	-	-	Unknown warranty

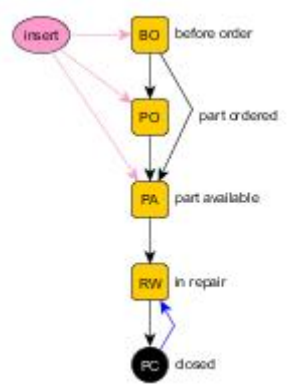
"result" can reach values:

value	description
REP	Repaired
XCH	Exchanged
ACC	Accessories exchanged
DIA	Diagnostics only
QRE	Quotation rejected
QLE	Quotation lapsed
NTF	No trouble found
NOA	No action (unsupported device)
SCR	Scrapped
QNC	Canceled
null	Not available

The **"symptomCode"** is the code of the defect from a range agreed with partner's company.

In **"repairCode"** will be code of the defect defined by engineer.

Part **"status"** property will contain two characters long string expressing the position in workflow schema:



If **"billing"** data equals with the delivery, it will bring a value of *null* instead of described fields.

The **"history"** array contains history of device repair cases ordered from the oldest. The **"type"** property can reach value of **common** for common repair case, **pair**for case that is repaired in parallel with the main case (e.g. with different warranty) and **last** for the last repair. Be aware that this array contains all cases, so even the one requested.

8.1.3.3 Create new case

To create the case, you will need an extra permission.

8.1.3.3.1 Request

URL: **/unit**
METHOD: **POST**
Header: **Content-type: application/json**

8.1.3.3.2 Request body JSON properties:

Download case-new.schema.json.

(string) **"identifier":** Your company case identifier. It is up to you, what you will choose as identifier. It only should be unique per case. Its maximal length is 16 characters.
(string) **"serialNumber":** The serial number of the device. Either **"serialNumber"** or **"imei"** is [required].
(string) **"imei":** The IMEI number of the device.
(string) **"imei2":** The second IMEI number if the device has it.
(string) **"vendor":** [required] The vendor of the device (eg. HTC, Brother).
(string) **"productNumber":** [required] The product number of the device.

(bool) **"inWarranty":** Your assumption whether the repair will be warranty. *null* when it is not known.

(string) **"symptomCode":** Code from a range agreed with partner's company that describes certain device symptom.

(string) **"symptomDescription":** Defect description. Either **"symptomCode"** or **"symptomDescription"** is [required].

(string) **"visualAppearance":** Outer appearance of the device upon receipt.

(date) **"purchaseDate":** The date when the end customer bought the device.

(date) **"claimDate":** When the date of the begin of complaint differs from today (ie. the day of case creating) it should be defined.

(string[]) **"accessories":** The list of attached accessories.

(string) **"notice":** The internal b2b notice.

(object) **"delivery":** [required] The delivery address information:

(string) **"name":** [required] Full name of end customer / company.

(string) **"street":** [required] Street and number.

(string) **"zip":** [required] Postal code / ZIP.

(string) **"city":** [required] City.

(string) **"country":** [required] ISO 3166-1 2-letter country code.

(string) **"phone":** Customer's phone number.

(string) **"fax":** Customer's fax number.

(string) **"email":** Customer's e-mail address.

(string) **"contactPerson":** The name of person available on given phone/e-mail contacts.

(string) **"idNumber":** National identification number. Used for billing if no billing ID provided.

(string) **"VAT":** Value added tax identification number. Used for billing if no billing VAT provided.

(object) **"billing":** The billing address information. If it is not attached, it is assumed to be the same as delivery.

(string) **"name":** [required] Full name of end customer / company.

(string) **"street":** [required] Street and number.

(string) **"zip":** [required] Postal code / ZIP.

(string) **"city":** [required] City.

(string) **"country":** ISO 3166-1 2-letter country code. Default is the same as delivery.

(string) **"idNumber":** National identification number.

(string) **"VAT":** Value added tax identification number.

(float) **"fund":** Preliminary agreement with a price limit for repair, given in a customer's country currency.

8.1.3.3.3 Request body example


```
{
   "identifier": "123456",
   "serialNumber": "DEF012345678",
   "imei": "543210987654323",
   "vendor": "HTC",
   "productNumber": "99ABC012-00",
   "inWarranty": true,
   "symptomCode": "F123",
   "symptomDescription": "Customer's defect description...",
   "purchaseDate": "2014-08-15",
   "claimDate": "2014-10-01",
   "accessories": [
      "headphones",
```

```
      "battery"
    ],
    "delivery": {
      "name": "Acme co.",
      "street": "Hollywood 14",
      "zip": "900 00",
      "city": "L.A.",
      "country": "US",
      "phone": "+1 555-1234",
      "email": "acme@example.com",
      "contactPerson": "Wile E. Coyote"
    },
    "fund": 80
}
```

8.1.3.3.4 Example Response

```
{
    "message": "Case with identifier 123456 was successfully logged in under the case number HTC0123456.",
    "id": 178901,
    "caseNumber": "HTC0123456"
}
```

8.1.3.4 Update case

8.1.3.4.1 Request

URL: /unit/<caseNumber>
URL: /unit/<id>
METHOD: PUT
Header: Content-type: application/json
example: /unit/HTC0123456

8.1.3.4.2 Request body JSON properties

Download case-update.schema.json.

(object) **"delivery":** The delivery address information:
(string) **"name":** Full name of end customer / company. To update address, full one is required (including country).
(string) **"street":** Street and number.
(string) **"zip":** Postal code / ZIP.
(string) **"city":** City.
(string) **"country":** ISO 3166-1 2-letter country code.
(string) **"phone":** Customer's phone number. To update contact information, all **"phone"**, **"fax"** and **"email"** are required.
(string) **"fax":** Customer's fax number.
(string) **"email":** Customer's e-mail address.
(string) **"symptomDescription":** Defect description.
(string) **"visualAppearance":** Outer appearance of the device upon receipt.

8.1.3.4.3 Request body example

```
{
    "symptomDescription": "Modified customer's defect description...",
    "delivery": {
      "name": "Acme Ltd",
      "street": "Hollywood 14",
      "zip": "900 00",
      "city": "L.A.",
      "country": "US"
    }
```

```
}
```

8.1.3.4.4 Example Response

```
{
   "message": "Case HTC0123456 was successfully updated."
}
```

8.1.3.5 Approve quotation

2016-03-22 Approve quotation deprecated

Set the case to the approved state when the customer has accepted quotation.

8.1.3.5.1 Request

URL: /unit/approve-quotation/<caseNumber>
URL: /unit/approve-quotation/<id>
METHOD: PUT
example: /unit/approve-quotation/HTC0123456

8.1.3.5.2 Example Response

```
{
  "message": "Quotation for case HTC0123456 was successfully approved."
}
```

8.1.3.6 Release case

Set the case to the released state after unit has been handed to the customer.

8.1.3.6.1 Request

URL: /unit/close/<caseNumber>
URL: /unit/close/<id>
METHOD: PUT
example: /unit/close/HTC0123456

8.1.3.6.2 Request body JSON properties

Download case-close.schema.json.

(datetime) **"closedDate":** The date optionally with time when the unit was released.

8.1.3.6.3 Request body example

```
{
  "closedDate": "2014-10-12"
}
```

8.1.3.6.4 Example Response

```
{
  "message": "Case HTC0123456 was successfully closed."
}
```

8.1.3.7 List quotation for approval

Available from 2016-03-22

More detailed information about price quotation can be obtained as follows:

8.1.3.7.1 Request

URL: /price-quotation/
URL: /price-quotation/<caseNumber>
METHOD: GET
example: /price-quotation/HTC0123456

8.1.3.7.2 Example response


```
{
   "count": 2,
   "case":
   {
      "HTC0123456":
      {
         "price": 77.55,
         "currency": "EUR"
      },
      "HTC0123789":
      {
         "price": 3500.60,
         "currency": "CZK"
      },
   }
}
```


8.1.3.8 Decision price quotation

Available from 2016-03-22

Set a decision on the price quotation - approved [QPA] or rejected [QPR].

8.1.3.8.1 Request

URL: **/price-quotation/decision/**
METHOD: **PUT**
example: **/price-quotation/decision/**

8.1.3.8.2 Request body JSON properties

Download decision-price-quotation.schema.json.

(string) **"caseNumber":** The unique identifier for a case.
(float) **"price":** Approved on an amount.
(decision) **"caseNumber":** Your decision to price quotation. QPA (accept) or QPR (rejection).

8.1.3.8.3 Request body example


```
{
   "caseNumber": "HTC0123456",
   "price": "58.30",
   "decision": "QPA"
}
```


8.1.3.8.4 Example Response


```
{
   "message": "Quotation for case HTC0123456 was successfully approved."
}
```


8.1.4 Transfer order

8.1.4.1 List range of transfer orders

8.1.4.1.1 Request

URL: **/order?since=<dateSince>[&to=<dateTo>][&page=<page>][&pageLimit=<pageLimit>]**

METHOD: GET
example: /order?since=2014-11-10&to=2014-11-20

Orders are ordered from newest.

Server will return JSON object with property **"orders"** that contains array of orders that met the conditions. Each order is object as the one in an example:

8.1.4.1.2 Example response


```
{
   "count": 1,
   "totalCount": 1,
   "currentPage": 1,
   "pagesCount": 1,
   "pageLimit": 30,
   "orders": [
      {
         "orderNumber": "ABC-12345",
         "status": "open",
         "country": "CZ",
         "address": [
            "Acme Ltd.",
            "Hollywood 7",
            "901 00"
         ],
         "emails": [
            "acme@example.com"
         ],
         "createdDate": "2012-02-15",
         "dispatchMethod": "PPL",
         "priority": 2,
         "shipmentNumber": "123456",
         "courier": "PPL",
         "price": 99.35,
         "duty": 14.1,
         "material": [
            {
               "partNumber": "AAAA123456",
               "name": "SCREW",
               "quantity": 25,
               "swap": false
            }
         ]
      }
   ]
}
```


Status can reach values: **open**, **closed** or **dispatched**.

Address parts count is variable.

8.1.4.2 Get single transfer order

8.1.4.2.1 Request

URL: /order/<orderNumber>
METHOD: GET
example: /order/ABC-12345

The response is one order object, similar as the one in an example response at section 4.1.

8.1.4.3 Create new transfer order

8.1.4.3.1 Request

URL: /order
METHOD: POST
Header: Content-type: application/json
Header: Expect:

8.1.4.4 Request body JSON properties:

Download material-transfer-order.schema.json.

Request is JSON array that contains one or more order objects with following properties:

(string) **"orderNumber":** [required] The order number.
(string) **"country":** [required] ISO 3166-1 2-letter country code.
(string) **"address":** Address where parts are separated with newline.
(array) **"address":** Or array of address parts. One specification of address is [required].
(array) **"emails":** List of partner e-mail addresses.
(string) **"dispatchMethod":** Dispatch method.
(int) **"priority":** Priority from 1 to 3 (1 = highest).
(array) **"material":** [required] Array of part objects with following properties:
(string) **"partNumber":** [required] Part number.
(int) **"quantity":** [required] Count of parts.

8.1.4.4.1 Request body example

```
[
  {
    "orderNumber": "ABC-12345",
    "country": "CZ",
    "address": "Acme Ltd.\nHollywood 7\n901 00",
    "emails": [
      "acme@example.com"
    ],
    "dispatchMethod": "PPL",
    "priority": 2,
    "material": [
      {
        "partNumber": "AAAA123456",
        "quantity": 25
      },
      {
        "partNumber": "BBBB123456",
        "quantity": 4
      }
    ]
  }
]
```

8.1.4.4.2 Example Response

```
{
  "message": "Order ABC-12345 was successfully logged. Booked 29 pieces of material."
}
```

8.1.4.5 Remove transfer order

8.1.4.5.1 Request

URL: /order/<orderNumber>
METHOD: DELETE

Order can be removed only if it is not processed yet.

8.1.5 Stock levels

8.1.5.1 List stock levels

8.1.5.1.1 Request

URL: /stock?vendor=<vendor>&stock=<stock>
METHOD: GET
Example: /stock?vendor=HTC[&stock=H01]

Server will return JSON object with property **"material"** that contains array of parts in stock.

8.1.5.1.2 Example response

```
{
   "count": 2,
   "material": [
      {
         "partNumber": "AAAA123456",
         "partName": "SCREW",
         "freeQuantity": 170,
         "bookedQuantity": 25,
         "orders":
         [
            "orderNumber": "ABC-12345"
         ]
      },
      {
         "partNumber": "BBBB123477",
         "partName": "NUT E-300",
         "freeQuantity": 350,
         "bookedQuantity": 0,
         "orders":
         [
         ]
      }
   ]
}
```

8.1.6 Parts information

8.1.6.1 The status of parts in stock

8.1.6.1.1 Request

URL: /part?part=<partNumber>
METHOD: GET
example: /part?part=ABCD123-45

Server will return JSON object with property **"cases"** that contains array of open cases with the specified part.

8.1.6.1.2 Example response


```
{
   "partNumber": "AAAA123-45",
   "partName": "SCREW",
   "vendor": "HTC",
   "stock": {
      "PA": 274,
      "RW": 7,
      "PC": 4534,
      "CA": 3
   },
   "cases": [
   [
      "HTC012345",
      "HTC012346",
      "HTC013235",
      "HTC014346",
      "HTC015457"
   ]
}
```


8.1.6.2 The number of parts in a given status

8.1.6.2.1 Request

URL: /part?part=<partNumber>&status=<status>
METHOD: GET
example: /part?part=ABCD123-45&status=RW

8.1.6.2.2 Example response


```
{
   "partNumber": "AAAA123-45",
   "partName": "SCREW",
   "vendor": "HTC",
   "stock": 7
}
```


8.1.6.3 Consumption of parts on case

URL: /part-consumption/<caseNumber>
METHOD: GET
example: /part-consumption/ABC0123456

Server will return JSON object with property **"parts"** that contains array of parts consumed in the case.

8.1.6.3.1 Example response


```
{
   "caseNumber": "ABC0123456",
   "parts": [
      {
         "partNumber": "ABCD123-45",
         "status": "PC"
      },
      {
         "partNumber": "ABDE456-78",
         "status": "PC"
      },
      {
         "partNumber": "83ABC016-23M",
```

```
        "status": "RW"
      }
    ]
}
```

8.1.6.4 Consumption of parts for a specified period

URL: `/part-consumption?since=<dateSince>&to=<dateTo>`
METHOD: `GET`
example: `/part-consumption?since=2015-09-01&to=2015-09-10`

The maximum period is **10 days**.

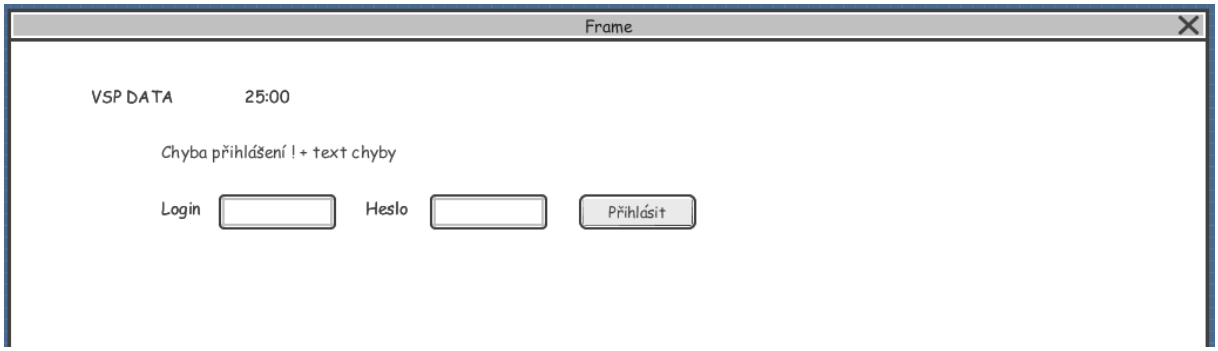
Server will return JSON object with property **"parts"** that contains array of parts consumed for a specified period.

8.1.6.4.1 Example response

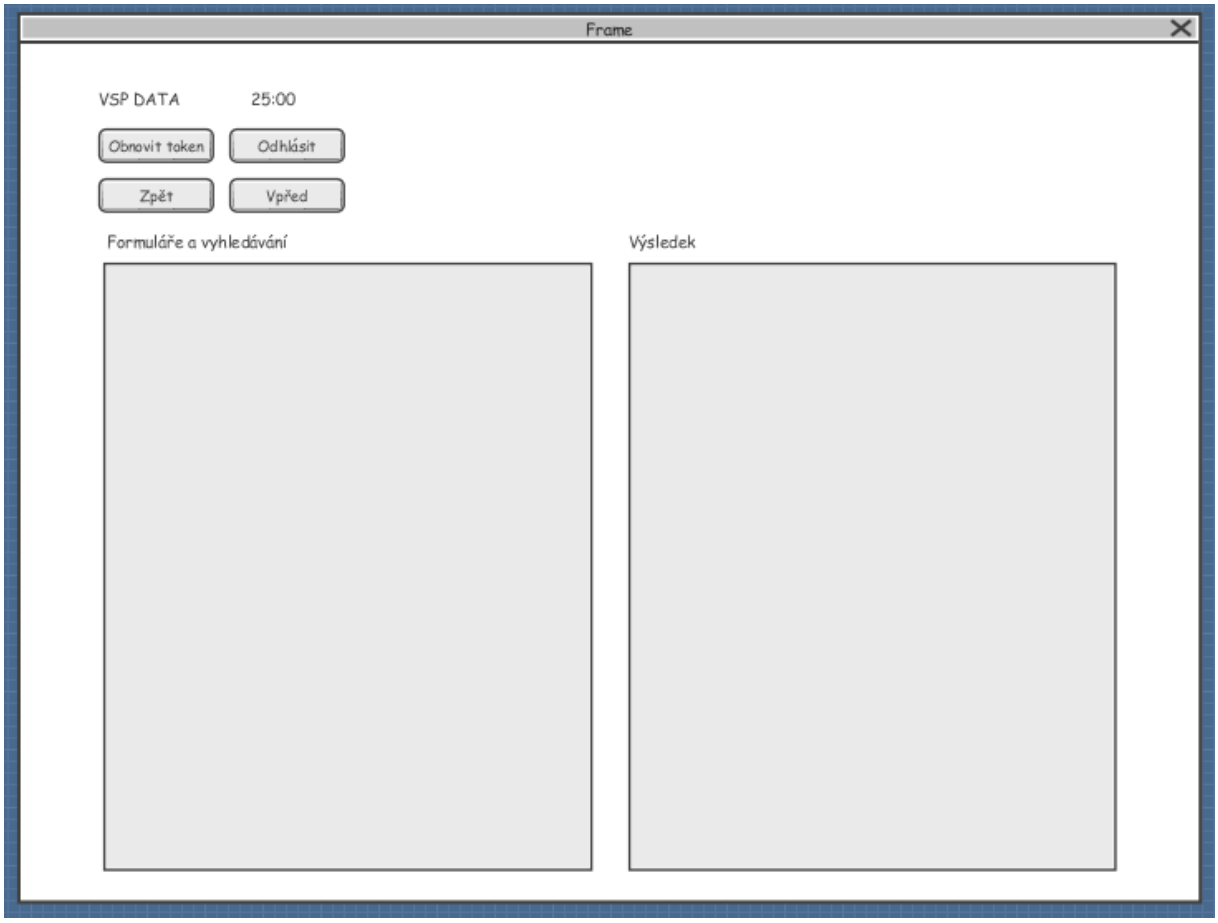

```
{
  "parts": [
    {
      "caseNumber": "ABC1234567",
      "partNumber": "ABCD123-45",
      "status": "PC"
    },
    {
      "caseNumber": "CD00011236",
      "partNumber": "ABDE456-78",
      "status": "RW"
    },
    {
      "caseNumber": "CEA14788999",
      "partNumber": "83ABC016-23M",
      "status": "DP"
    }
  ]
}
```

8.2 Příloha 2 – Prototyp UI aplikace

Obrázek 11 - Úvodní obrazovka před přihlášením



Obrázek 10 - Po neúspěšném pokusu o přihlášení



Obrázek 12 – Po úspěšném pokusu o přihlášení

8.3 Příloha 3 – Aplikace

Příloha číslo 3 je externí příloha a obsahuje zdrojové kódy naší aplikace. Soubor je „aplikace.zip“. Aplikace není spustitelná ani na při nahrání na hosting kvůli jejímu zabezpečení.