VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRODEJCE PODAVAČŮ TYČÍ DO SOUSTRUHŮ
INFORMATION SYSTEM FOR A SELLER OF BAR FEEDERS FOR LATHES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PAVEL KALETA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO, 2019

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

21862

Student: **Kaleta Pavel**
Program: Informační technologie
Název: **Informační systém prodejce podavačů tyčí do soustruhů**
Information System for a Seller of Bar Feeders for Lathes
Kategorie: Informační systémy
Zadání:

1. Seznamte se s principy tvorby webových aplikací, dostupnými prostředími a frameworky.
2. Analyzujte požadavky na IS pro prodejce podavačů tyčí do soustruhů zahrnující objednávkový a archivační systém, který bude obsahovat sklad dokumentů, notifikace uživatelů, tvorbu grafů pro přehled o prodejích a efektivní vyhledávání v datech.
3. Navrhněte informační systém dle požadavků, použijte vhodné modelovací techniky.
4. Navržený systém implementujte a ověřte jeho funkčnost.
5. Zhodnoťte dosažené výsledky a diskutujte další možné pokračování tohoto projektu.

Literatura:

- Naramore, E., Gerner, J. et al: PHP 6, MySQL, Apache: Vytváříme webové aplikace. Computer Press, 2009. ISBN: 978-8-0251-2767-4.
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

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

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 26. října 2018

Abstrakt

Cílem této bakalářské práce je navrhnout a implementovat informační systém pro evidování stavu zakázek, uchování s nimi spjatých dokumentů a další ulehčení práce zaměstnanců firmy zabývající se prodejem podavačů tyčí do soustruhů. Webová aplikace postavená na nástroji Laravel umožňuje autentizaci uživatelů, vytváření nových zakázek a editaci již dříve vytvořených, přehledné zobrazení všech zakázek a efektivní vyhledávání v nich, nahrávání dokumentů k daným zakázkám a zobrazení statistik o všech zakázkách. V obsahu práce je popsán celý vývojový cyklus systému. Výsledkem této práce je plně funkční a v praxi nasazený informační systém.

Abstract

The aim of this bachelor's thesis is to design and implement an information system for monitoring the status of orders, keeping documents related to them and to further facilitate the work of employees of the company engaged in the sale of bar feeders for lathes. The Laravel web application allows users to authenticate, create new orders, edit previously created ones, clear view of all orders and efficient searches, upload documents to orders and view statistics on all orders. In the content of the thesis, the whole development cycle of the system is described. The result of this work is a fully functional and deployed information system.

Klíčová slova

Informační systém, webová aplikace, podavače tyčí, monitorování zakázek, uchování dokumentů, statistiky, Laravel, PHP, MySQL, JavaScript, HTML, Nginx, zabezpečení, softwarové testování

Keywords

Information systém, web application, bar feeders, order monitoring, document storage, statistics, Laravel, PHP, MySQL, JavaScript, HTML, Nginx, security, software testing

Citace

KALETA, Pavel. *Informační systém prodejce podavačů tyčí do soustruhů.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém prodejce podavačů tyčí do soustruhů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doktora Vladimíra Bartíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Pavel Kaleta
9. května 2019

Poděkování

Mile rád bych poděkoval svému vedoucímu práce panu Ing. Vladimíru Bartíkovi, Ph.D. za vedení mé práce, bezproblémovou komunikaci a cenné rady.

Obsah

1 Úvod

Informační systémy jsou v dnešní době velmi rozšířené. Vyskytují se všude kolem nás a ulehčují lidem život, jak i v práci, tak i ve volném čase. Jednou z firem, která dodnes pracuje bez vlastního informačního systému je společnost zabývající se prodejem podavačů tyčí do soustruhů. Zaměstnanci této firmy musí uchovávat veškeré dokumenty ve složité adresářové struktuře, musí si pamatovat, kde se jaký soubor nachází, veškeré informace uchovávají v nepřehledných tabulkách a často si musí jednotliví zaměstnanci mezi sebou posílat různé dokumenty, aby všichni měli aktuální verzi. Neexistuje u nich žádná synchronizace souborů, a tak musí vykonávat práci, kterou by za ně jednoduše zvládl informační systém.

Druhá kapitola je věnována seznámení se s informačními systémy a jejich rozdělení. Také obsahuje znalost, jak vybrat správný informační systém pro svou potřebu.

Třetí kapitola zachycuje analýzu požadavků na informační systém a principy použité při tvorbě návrhů. Čtenář zde nalezne informace ohledně konceptuálního modelování a také konkrétní návrhy systému.

Čtvrtá kapitola zachycuje použité technologie při vývoji tohoto informačního systému a detailněji seznamuje čtenáře s jednotlivými prvky. Dbá na detailní představení frameworku Laravel, jelikož se jedná o základní stavební kámen toho projektu. Také představuje jeho adresářovou strukturu a instalaci nového projektu. Píše se zde, proč jsem zvolil webový server Nginx místo tradičního Apache a proč se k tomuto projektu hodí použít databázový server MySQL.
V této kapitole je také popsána implementace v chronologickém postupu – co jsem krok po kroku musel provést. Vysvětluje se zde, jak Laravel vytváří tabulky v databázi, kde se uchovávají HTML šablony a co obsahují. Také se zde píše o tzv. cestách – jak framework interpretuje zadanou URL adresu uživatelem a o řadičích, které se starají o veškerou logiku aplikace. V neposlední řadě jsou zde popsány všechny funkce informačního systému pro výše zmíněnou firmu.

Pátá kapitola se věnuje zabezpečení informačních systému, jak vypadají nejčastější kybernetické útoky a jak se proti nim dá bránit. Nachází se zde také reálné statistiky, aby měl čtenář lepší přehled o skutečném dění.

Předposlední kapitola, šestá, seznamuje s testováním softwaru. Píše se zde, jak se takové testování dělí, kdy k němu dochází a k čemu je vůbec potřeba. Také zde zmiňuji mnou navržené testy, které jsem použil pro ověření funkcionality systému a systémové požadavky, aby se tyto testy daly spustit.

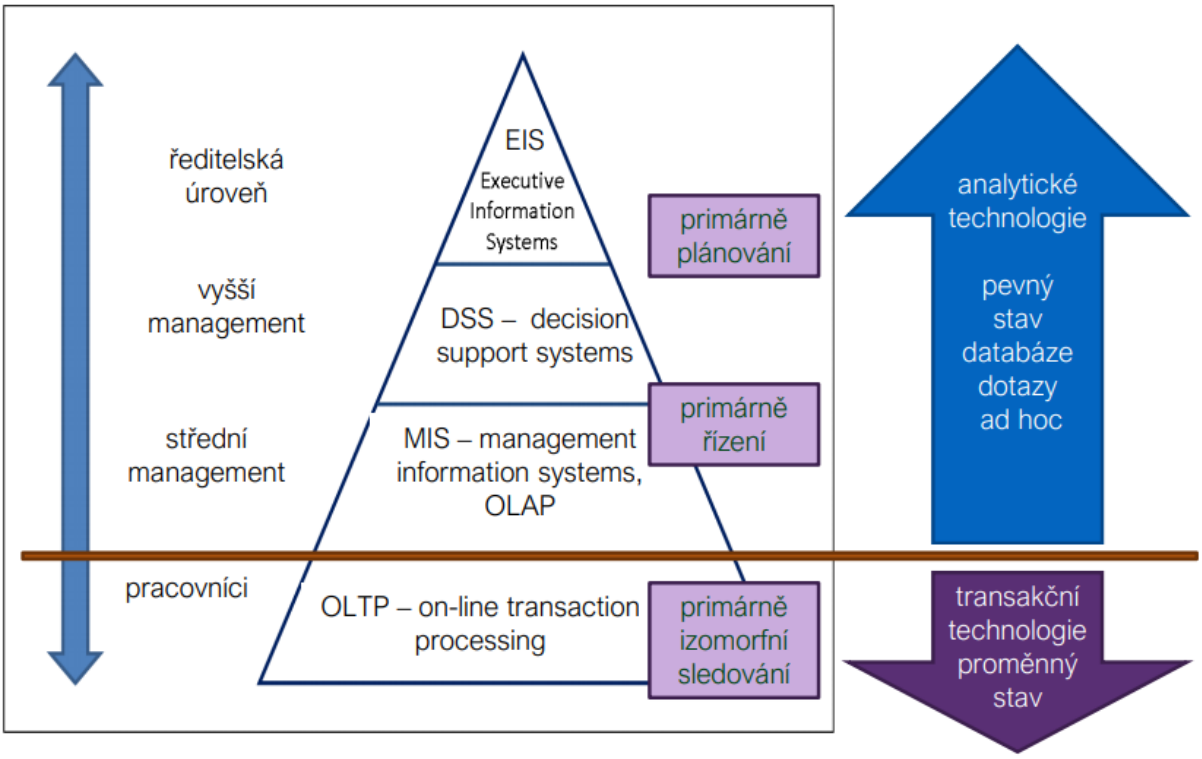
Poslední kapitola je věnována možným rozšířením informačního systému, která mě napadla během jeho vývoje, ale nebyla požadována od klienta. Nastiňuji zde, jak se dá použít obyčejných SMS zpráv pro vylepšení zabezpečení systému nebo další možnost upozornění uživatele. Také zmiňuji pár detailních rozšíření, která spíše souvisí s přehledností a je otázkou času, zdali budou užitečná.

2 Informační systémy

Aniž bychom si to uvědomili, informační systémy jsou v dnešní době všude kolem nás. Jen při cestě do obchodu narazíte hned na několik takových systémů. Pokud používáte aplikaci na zjištění, v kolik hodin vám odjíždí tramvaj, používáte informační systém. Jdete kolem měnící se tabule, díváte se na výstup informačního systému. V obchodě na kase prodavačka taktéž používá informační systém. Takové systémy ulehčují lidem práci a rozvíjí se čím dál více.

2.1 Rozdělení informačních systémů

V dnešní době existuje obrovské množství typů informačních systémů. Lidé se mnohokrát snažili o jejich klasifikaci, jenže ta byla platná pouze v době jejího vytvoření a ta, která vznikla před pár lety, už nemusí být dnes platná. Dalším faktorem ovlivňující tuto skutečnost je, že mnoho kategorií typů systémů nejsou disjunktní, to znamená, že jeden systém často spadá do více možných kategorií. Mezi nejstarší rozdělení patří tzv. Pyramidový model (viz Obrázek 1).



Obrázek 1: Pyramidový model

2.1.1 OLTP

Zkratka OLTP (z anglického Online Transaction Processing) označuje nejrozšířenější typ informačních systémů. Využívá technologii uložení dat v databázi a umožňuje tak snadnou a bezpečnou manipulaci s daty v mnohauživatelském prostředí. Změny v databázi se projevují v

reálném čase a systém okamžitě odpovídá na uživatelovy požadavky změnou stavu. Příkladem může být bankomat.

2.1.2 MIS

Další rozsáhlý typ systémů se skrývá za zkratkou MIS, tedy Management Information Systems, překládáme jako Informační systémy pro podporu řízení. Tyto systémy poskytují informace pro efektivní správu a řízení organizace. Údaje z databází jsou většinou interpretovány jako grafy, tabulky či sestavy.

2.1.3 DSS

Systémy pro podporu rozhodování (z anglického Decision Support Systems, DSS) využívají údaje z rutinních transakcí organizace a pomáhají uživatelům při rozhodování a řízení činnosti v podnikání. Systém nabízí uživateli zúžený rozsah voleb založený na teorii rozhodování a tím ulehčuje analýzu a kvantifikuje rizika. Samotné rozhodnutí je však na uživateli.

2.1.4 EIS

EIS (z anglického Executive Information Systems), překládáme jako Informační systémy pro exekutivu, se dají považovat za specializované systémy pro podporu rozhodování. Kladou důraz na grafické zobrazení informací a uživatelsky snadné ovládání. Často zahrnují celý podnik a reagují na změny v trendech, důležitých proměnných a monitorují výkonnost podniku a upozorňují na problémy či příležitosti.

Čerpáno z [1].

2.2 Výběr vhodného informačního systému

Výběr systému úzce souvisí s analýzou požadavků klienta. Mnoho klientů ani sami neví, co vlastně požadují nebo jen část, a tak je velmi důležité cílenými otázkami dostat ze zákazníka veškeré potřebné informace, na kterých můžeme stavět. Stejně důležité jako zjistit účel systémů je také prostředí, na kterém systém poběží a jak uživatelé k němu mají přistupovat. To může zásadně ovlivnit použitou technologii při vývoji systému. Mezi další velmi důležité prvky patří např. úroveň zabezpečení.

Spousta dodavatelských firem zná pouze svůj systém, a ne ty ostatní. Nezávislí konzultanti také většinou znají maximálně 2-3 systémy. Proto je téměř nemožné získat objektivní posouzení.

3 Analýza požadavků a návrh

Mezi nejdůležitější část správného postupu při tvorbě informačních systému patří dodržení vývojového cyklu projektu. Pokud bychom skákali z jedné etapy projektu ke druhé, brzy by se vyskytl zmatek, chyby či redundance. Než vůbec přistoupíme k první etapě, je nutné zjistit, zda bude investice do systému výhodná a zda jej lidé opravdu využijí. V případě, že se takový systém vyplatí, nastává první fáze – fáze analýzy.

3.1.1 Analýza

V této fázi životního cyklu zjišťujeme od klienta veškeré požadavky na systém. Klient málokdy vyjmenuje veškerou požadovanou funkcionalitu již při první komunikaci anebo později identifikujeme nejasné, nekompletní či protichůdné požadavky, a tak je důležité se doptat na kritické body systému, na kterých se dá stavět a vytvořit na nich konceptuální model.

2.1.1.1 Zjištěné požadavky na IS

Po první schůzce s klientem jsem zjistil následující požadavky. Jedná se o systém uchovávající informace o zakázkách a dokumentech s nimi spojenými, systém by měl umět:

- Monitorovat stav zakázek – uživatel systému by měl mít možnost vytvořit novou zakázku a uvést u ní základní detaily (název firmy a osoby, které se zakázka týká, z jakého státu firma pochází a jedná-li se o prodejce nebo koncového zákazníka). Poté by měl existovat přehled všech již vytvořených zakázek, ve kterém může uživatel sledovat jejich stavy.
- Upravovat detaily položek – pokud se u zakázky vyskytne potřeba upravit její detaily nebo přidat detaily nové (cena zboží, kódy z dokumentů, stav zakázky aj.) měl by systém tuto skutečnost umožnit.
- Přehledný seznam všech uskutečněných či potencionálních objednávek – možnost jednoduše rozlišit a zobrazit, v jakém stavu se jednotlivé zakázky nachází. Zakázky v systému budou postupně měnit svůj stav v závislost na stavu reálné zakázky a bylo by dobré, aby uživatel mohl zobrazit jak objednávky (jeden ze stavů zakázky) uskutečněné, tak teprve zakázky na objednávku čekající.
- Upozorňování na věci, které je potřeba udělat (kontaktovat klienta...) - jelikož denně vznikne hned několik nových zakázek, bylo by dobré mít nastavené automatické upozorňování po několika dnech od vytvoření nabídky, že se klient doposud neozval a zakázka je stále ve stejném stavu. To umožní uživatelům systému ihned vidět přehled všech zakázek ve stavu nabídky a kontaktovat zákazníky, zdali mají o zboží zájem.
- Možnost přikládat soubory k jednotlivým zakázkám – s každou zakázkou je spjat minimálně 1 soubor, a to dokument s nabídkou vytvořenou klientovi na míru. Pokud bude mít klient o nabídnuté zboží zájem, pošle nazpět další dokument s objednávkou. Během celého procesu obdrží uživatel systému ještě několik dalších dokumentů (potvrzení objednávky, fakturu aj.). Všechny tyto soubory je také potřeba uchovat v systému u příslušné zakázky.
- Digitální sklad dokumentů (PDF katalogy, faktury, obrázky...) - možnost vyhledávat mezi všemi dříve nahranými dokumenty.
- Inteligentní vyhledávání podle klíčových slov, filtrování – možnost jednoduše vyhledat potřebnou zakázku podle klíčových slov s ní spojenými (název firmy či osoby a unikátní kódy z příchozích dokumentů) a možnost vyfiltrovat zakázky podle kritérií.

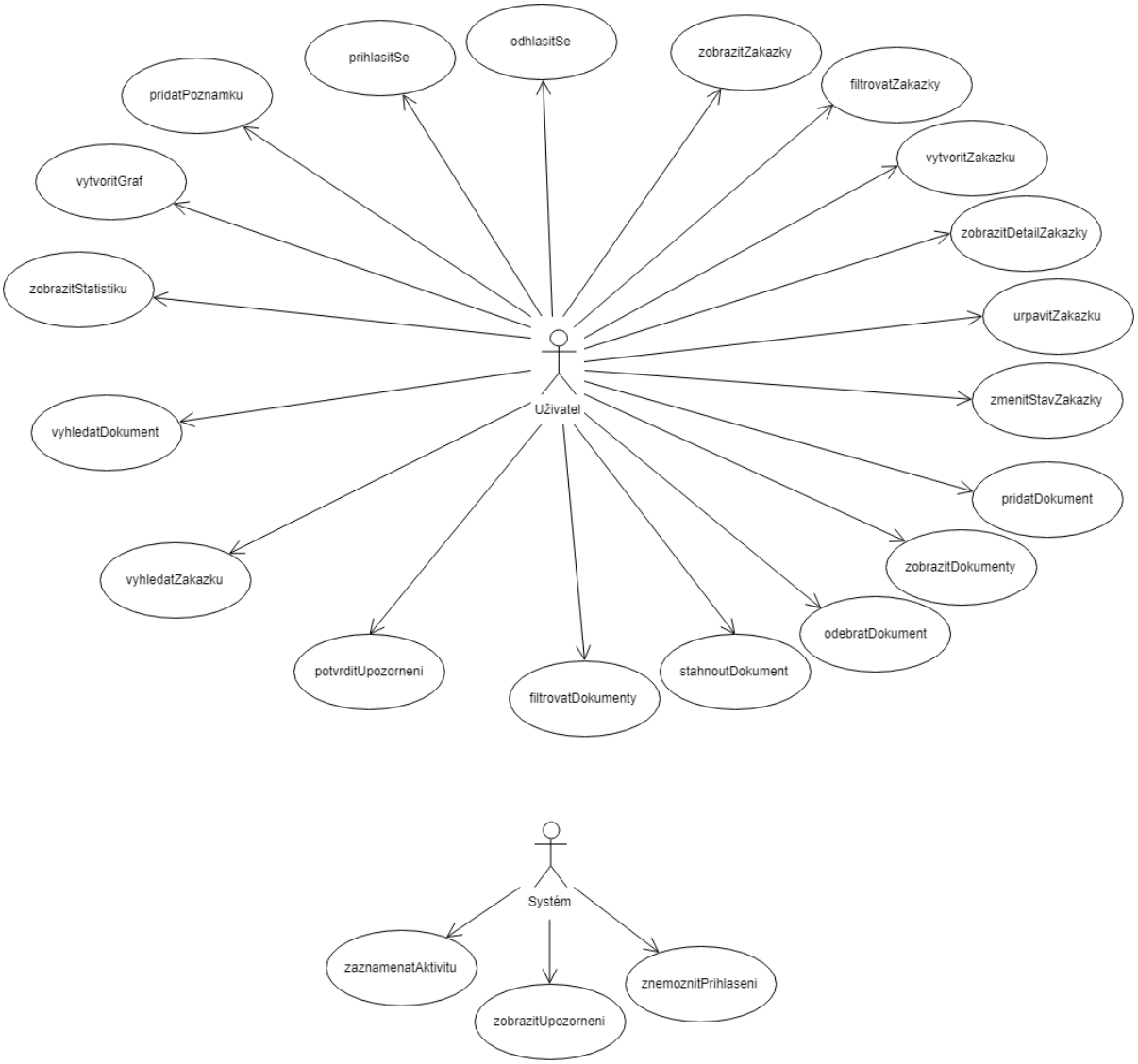
- Možnost vytvoření přehledu prodeje – detailní statistiky zobrazující informace o zakázkách.
- Online – přístup více lidí odkudkoliv, zabezpečení.
- Systém bude používán na PC s webovými prohlížeči Chrome a Safari.
- Praktičnost nade vše.
- Předem vytvořené účty pracovníků – není potřeba registrace.

Jelikož jsou požadavky velmi specifické a vyžadují přesné kroky při práci se zakázkami, je nutné navrhnout systém na míru, nikoliv použít nějaké již existující řešení.

2.1.1.2 Diagram případů užití

Pomocí diagramu případů užití (Obrázek 2) modeluji funkcionalitu informačního systému tak, aby byl srozumitelný i pro klienta. Všichni uživatelé systému mají stejná oprávnění a jejich účty jsou vytvořeny přímo zápisem do databáze.

Uživatel je při přístupu do systému vyzván k přihlášení, pokud několikrát v přihlášení neuspěje, systém kombinaci jeho IP adresy a přihlašovacích údajů zablokuje na určitou dobu. V případě úspěšného přihlášení se uživateli zobrazí všechny položky (zakázky) z databáze seřazené od nejnovějších. Položky v tabulce může vyfiltrovat a zobrazit tak pouze požadované. Podobný efekt má vyhledávání, které navíc doporučuje uživateli názvy firem, osob, dokumentů a jiných prvků. Uživatel může vytvořit novou položku, přidat k ní potřebné údaje a dokumenty. Dále může zobrazit podrobné informace jednotlivých položek a případně upravit jejich detaily či změnit jejich stav. Dokumenty lze zobrazit přímo v detailu položky, ke které patří nebo v celkovém přehledu dokumentů. Na hlavní stránce uživatel vidí systémem vygenerované upozornění, které může potvrdit a tím změnit stav položky. Na stránce se statistikami je možné zobrazit souhrn všech informací o položkách a grafy. Systém také monitoruje a zaznamenává důležitou aktivitu uživatelů.



Obrázek 2: Diagram případů užití

3.1.2 Tvorba konceptuálního modelu

Tato fáze zahrnuje zachycení skutečností v rámci modelu. Čerpá ze zjištěných požadavků z první fáze životního cyklu a definuje strukturu dat, s nimiž pracuje informační systém. Cílem je vytvořit datový model, kterým se vývojáři řídí při implementaci konkrétního systému. Takový datový model může být zobrazen pomocí jazyku UML či ERD.

2.1.1.3 UML

Unified Modeling Language neboli UML slouží v softwarovém inženýrství k specifikaci, vizualizaci a navrhování programových systémů. Mezi nejběžnější použití jazyka UML patří:

- Kreslení konceptu – základní možnost komunikace mezi vývojáři a zaznamenání myšlenek
- Kreslení detailních návrhů – kompletní návrh, např. pomocí diagramů případů užití. Na základě těchto diagramů by měl být programátor schopný systém implementovat bez dalších otázek.

- UML jako programovací jazyk – na základě nakreslených UML diagramů se vygeneruje přímo spustitelný kód. Je zapotřebí speciálních nástrojů a velmi přesného vyjadřování.

Čerpáno z [2].

2.1.1.4 ER model

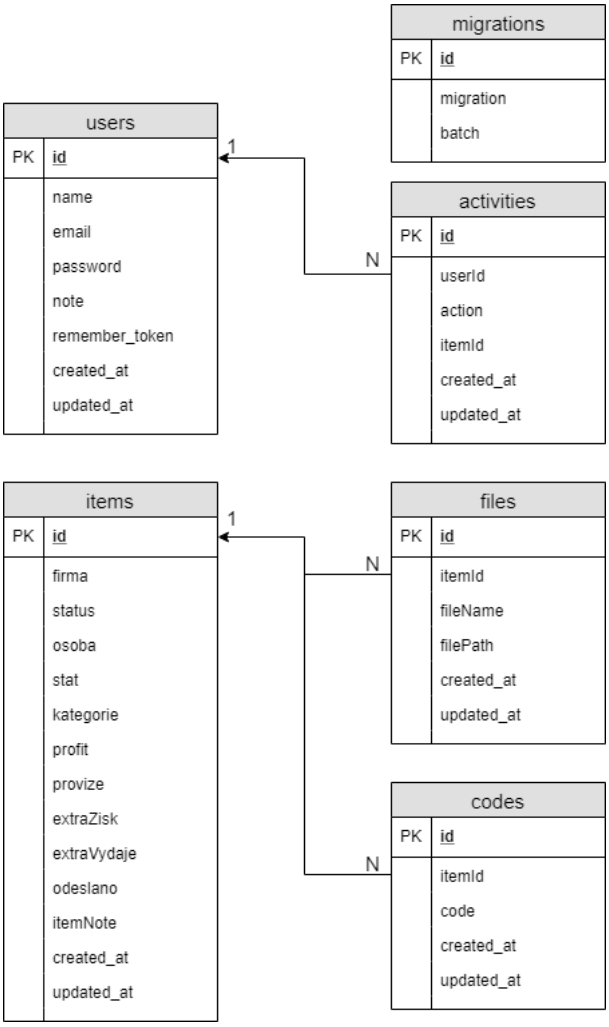
Entity-relationship model neboli entitně vztahový model je nyní „*základním modelem reprezentujícím požadavky na data uložená v databázi, z něhož vychází návrh tabulek relační databáze.*“ [3] Zapisuje se pomocí ER diagramu (ERD).

ERD obsahuje entity, atributy a vztahy:

- Entita – množina objektů reálného světa stejného typu
- Atribut – popisuje entitu, uchovává hodnotu
- Vztah – vztah mezi entitami

Pomocí ERD (Obrázek 3) modeluji databázi informačního systému, který obsahuje 6 entit:

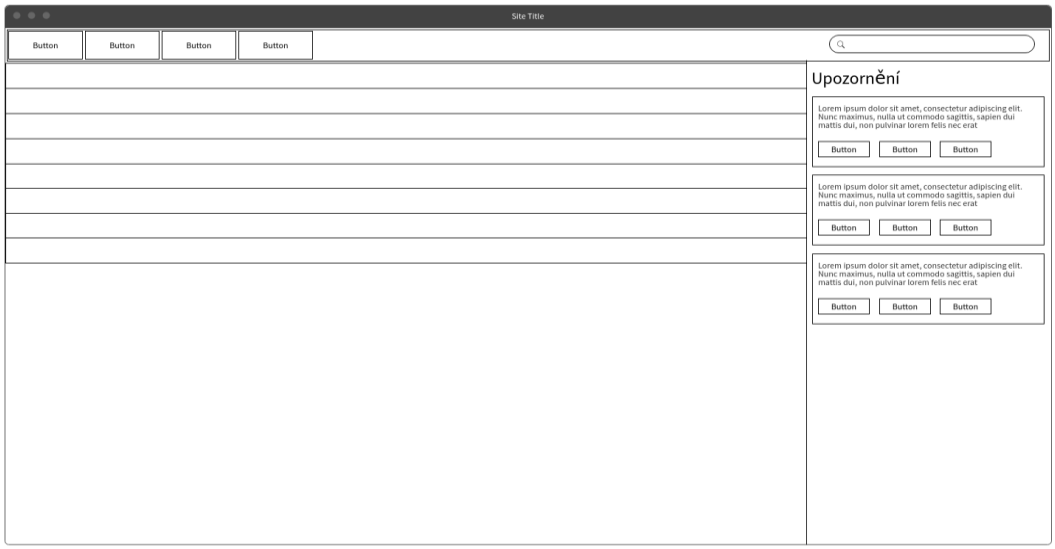
- users – tato entita uchovává informace o uživatelích systému. Atribut *id* je primárním klíčem a jednoznačně určuje každého uživatele. Atribut *name* uchovává celé jméno uživatele, *email* jeho e-mailovou adresu používanou pro přihlášení do systému. V atributu *password* je uloženo heslo v zašifrované podobě. Poznámky z hlavní stránky systému se ukládají uživateli do atributu *note*. Další *remember_token* slouží k ověření zapamatovaného uživatele v prohlížeči.
- items – do této entity se zaznamenávají jednotlivé položky (zakázky). Primárním klíčem, pomocí kterého se identifikují jednotlivé položky je opět *id*. Následují atributy uchovávající informace o jednotlivé položce.
- codes – entita uchovávající unikátní kódy z příchozích dokumentů. Používají se pro snadnější dohledání objednávky. Primárním klíčem je *id* a atribut *itemId* určuje, ke které položce kód patří. Samotný kód je uložen v *code*.
- files – enita zaznamenává informace o nahraných dokumentech, aby je bylo možné dohledat. Primární klíč je *id,* atribut *itemId* opět označuje, ke které položce dokument patří. Atributy *fileName* a *filePath* určují jméno dokumentu a místo uložení na serveru.
- activities – v této entitě jsou uchovány aktivity uživatelů. Primární klíč je *id, userId* označuje uživatele, který danou aktivity vykonal a *action* je číslo aktivity. Pokud se aktivita týká nějaké položky, je taktéž použit atribut *itemId.*
- migrations – poslední entita je automaticky vygenerována frameworkem Laravel a zaznamenává stav tabulek v databázi. Primární klíč je *id* a atribut *migration* uchovává jméno migrace (vysvětlení v podkapitole 3.1 Vytvoření databázové struktury). Atribut *batch* její stav.



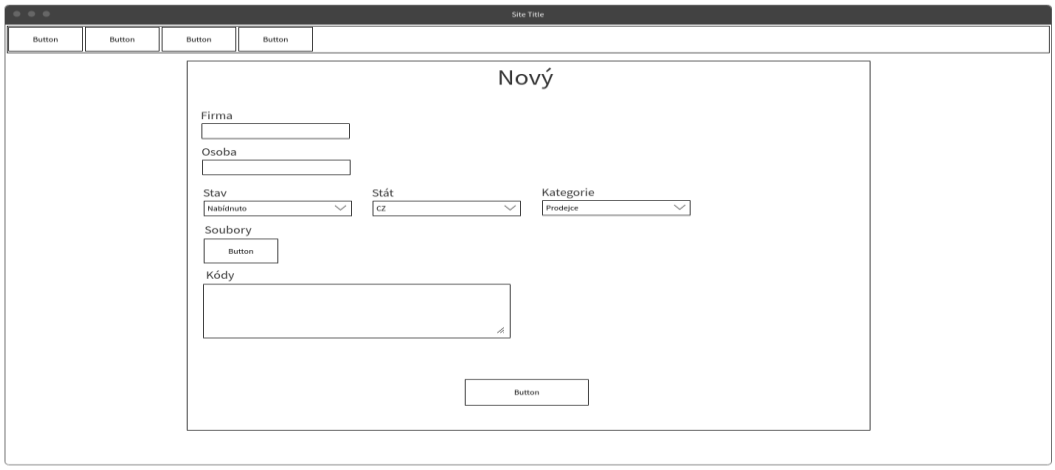
Obrázek 3: ERD

3.1.3 Návrh uživatelského rozhraní

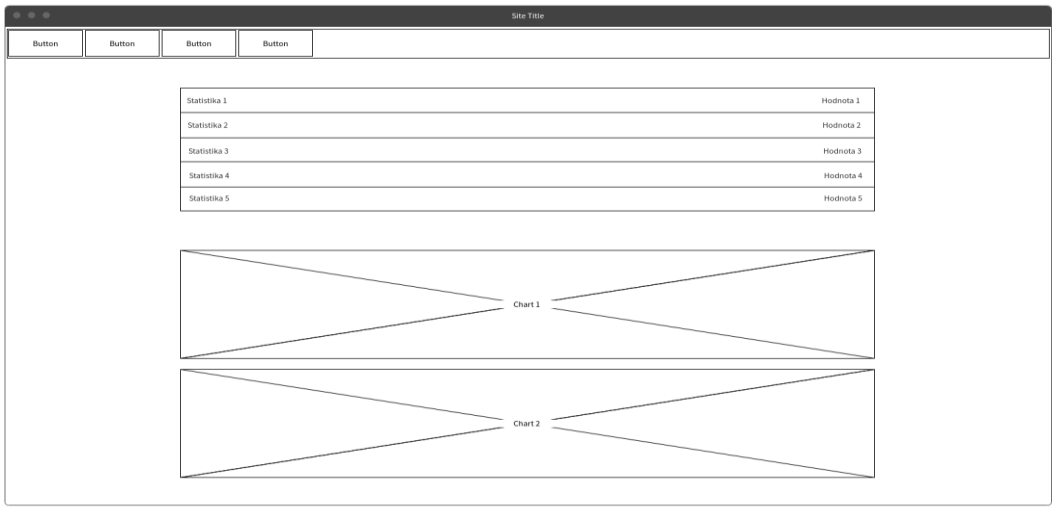
Před začátkem implementace jsem si vytvořil tzv. wireframe. Jedná se o návrh jednotlivých stránek webu, podle kterého jsem poté postupoval při vytváření HTML šablon. Jelikož zákazník uvedl, že zaměstnanci nejsou velmi zkušení v oblasti počítačů, musel jsem udělat návrh systému co nejpraktičtější a jednoduchý k použití. Zákazník preferoval praktičnost a možnost rychlého pohybu po systému nad designem. Ačkoliv jsem si wireframe připravil pečlivě, během implementace jsem přišel na několik prvků, které v návrhu chyběly. Také jsem během implementace několikrát konzultovat systém se zákazníkem a ten postupně přidával několik požadavků. Následuje několik ukázek návrhu wireframe.



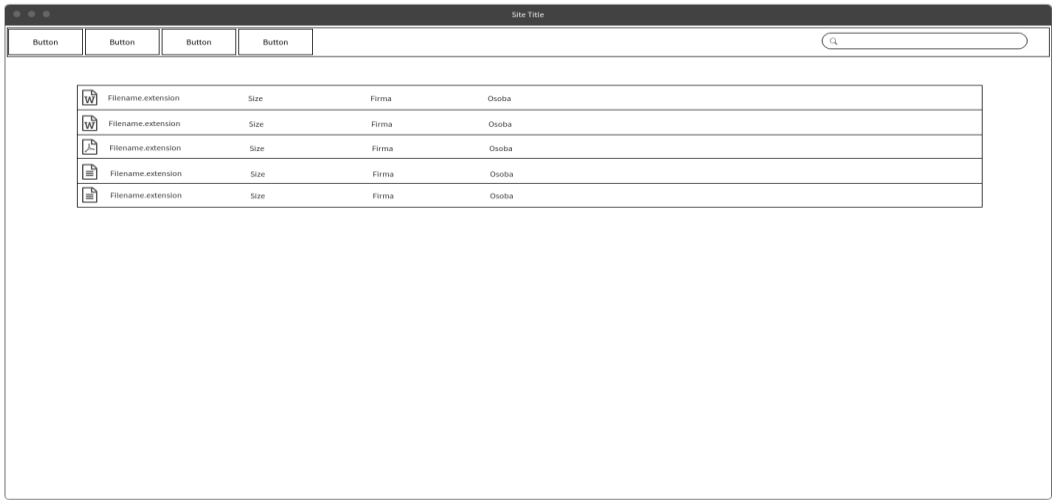
Obrázek 4: Wireframe přehledu



Obrázek 5: Wireframe tvorby nové zakázky



Obrázek 6: Wireframe statistik



Obrázek 7: Wireframe dokumentů

4 Použité technologie

Cílová skupina uživatelů mého systému je velmi malá, ale přesto vyžaduje umístění projektu na internet, aby byl jednoduše a rychle dostupný z celého světa. A proto je finální projekt umístěn na webu (World Wide Web). Ke správné funkčnosti systému běžícího na webu jsem použil tyto technologie:

- Skriptovací jazyk PHP a Javascript
- Framework Laravel
- HTML + kaskádové styly CSS
- Webový server Nginx
- Databázový server MySQL

Projekt jsem vyvíjel lokálně na operačním systému Windows s použitím multiplatformního softwarového balíčku XAMPP. Hotový projekt jsem nasadil na webový cloud hostovaný u firmy Forpsi. Na cloudu jsem zvolil operační systém Ubuntu Server pro své nižší požadavky na výpočetní výkon, tudíž i nižší cenu. Webový server Apache jsem zaměnil za Nginx, jelikož byl doporučován mnoha uživateli OS Ubuntu Server.

4.1 Skriptovací jazyk PHP

Jazyk PHP, označující Hypertextový preprocesor, se ve velké míře používá při programování dynamických webových stránek. Velmi úzce souvisí s HTML či XHTML, což je formát, ve kterém se zapisují webové stránky.

PHP skripty mohou být volány několika způsoby – přímo z příkazového řádku, pomocí dotazovacích metod HTTP nebo pomocí webových služeb. Dotazy jsou zpracovány na straně serveru a uživateli je vrácen výsledek dotazu, zobrazen pomocí webového prohlížeče.

Jazyk PHP je multiplatformní, je podporován většinou operačních systémů a rozdíly v jednotlivých verzích jsou minimální. Další skvělou vlastností PHP je fakt, že se jedná o velmi rozsáhlý programovací jazyk, tudíž existuje mnoho již vytvořených knihoven pro veřejné použití. Spolu s tím souvisí jednoduchý přístup a správa databáze. Tento skriptovací jazyk podporuje většinu rozšířených databázových serverů, např. MySQL, MariaDB, Oracle, PostgreSQL a internetových protokolů, např. HTTP, SMTP, POP3, LDAP aj. [4]

Pro svou obrovskou rozšířenost (79 % webových aplikací v březnu 2019 [5]) se PHP stal součástí zkratky LAMP, označující nejběžnější technologii při tvorbě webových aplikací. Zkratka LAMP označuje spojení Linux, Apache, MySQL, PHP (popřípadě Python nebo Pearl).

PHP je také využíván největšími internetovými projekty jako je Wikipedie či Facebook.

Během implementace bylo často nahlíženo do oficiální dokumentace jazyku PHP [6].

4.2 Javascript

Stejně jako PHP, Javascript je multiplatformní skriptovací jazyk používaný při tvorbě webových aplikací. Nejčastěji se zapisuje přímo do zdrojových souborů HTML stránek. Javascript, narozdíl od PHP, je vykonán po jeho úplném stažení na straně uživatele v prohlížeči. S tím souvisí bezpečnostní opatření – Javascript nemůže pracovat se soubory, aby neohrozil soukromí uživatele. [7]

Využívá se ke zlepšení uživatelské interakce s webovou stránkou. V poslední době se velmi rozšířilo používání technologie zvané Ajax, která umožňuje načtení nového obsahu webové aplikace bez nutnosti opětovného stažení celé stránky. Klient se při použití Ajax transparentně dotáže serveru, ten mu odpoví a pomocí Javascriptu se upraví potřebná část obsahu stránky. Uživatel tak nemusí čekat na zdlouhavé načítání celého obsahu a vylepší se tak uživatelský dojem.

Existuje několik velmi populárních knihoven pro Javascript, ze kterých jsem využil následující:

- jQuery v3.3.1 - rozšíření funkcionality Javascriptu, vyhledávání prvků v HTML a interakce s nimi, Ajax [8]
- typeahead.js 0.11.1 - dopředné vyhledávání (doporučování textu) [9]
- tokenizer.js - rozdělení slov na tokeny [10]
- excel-bootstrap-table-filter.js - Excelovské filtrování [11]
- Chart.js – grafy [12]

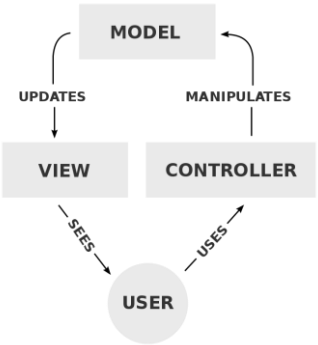
4.3 Framework Laravel

Laravel [13] patří v dnešní době k jednomu z nejpopulárnějších PHP frameworků používaného k vývoji webových aplikací. Je poskytován jako open source projekt pod licencí MIT. Laravel, založený na principu MVC (model-view-controller) architektury má spoustu výhod. Mezi největší z nich patří stále se rozrůstající komunita, což by mělo zajistit neustálou podporu a vývoj frameworku ze strany vývojářů. Další výhodou je jednoduchá integrace s knihovnami, spousta možností integrace s relačními databázemi, snadné nastavení cest a autentizace či cachování.

4.3.1 MVC

Model-view-controller je softwarová architektura, která rozděluje projekt na 3 komponenty:

- Model – dynamická datová struktura, s kterou aplikace pracuje. Přímo spravuje logiku, data a pravidla aplikace.
- View (pohled) - zobrazuje data z modelu do uživatelsky přívětivé a interaktivní podoby.
- Controller (řadič) - zpracovává povely (nejčastěji od uživatele) a zajišťuje změny v modelu, respektive pohledu.



Obrázek 8: Model-view-controller

4.3.2 Požadavky na server

Ke správné funkcionalitě frameworku Laravel musí server podporovat následující:

- PHP >= 7.1.3
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension
- Ctype PHP Extension
- JSON PHP Extension
- BCMath PHP Extension

4.3.3 Instalace nového projektu

K instalaci Laravelu je zapotřebí aplikaci Composer pro správu závislostí v projektu. Composer zajistí stažení správné verze balíčku a přiřazení k projektu.

Prvním krokem je stažení Laravel instalátoru a spuštění pomocí Composeru. Tento krok je vykonán jednoduchým příkazem:

```
composer global require laravel/installer
```

Dalším krokem je vytvoření nového projektu následujícím příkazem, přičemž NÁZEV nahradíme vlastním názvem projektu:

```
laravel new NÁZEV
```

Tímto je instalace dokončena. V tomto bodě bude nový projekt obsahovat následující adresářovou strukturu.

- Adresář *app* obsahuje jádro aplikace, modely a řadiče (controllers).
- Adresář *bootstrap* obsahuje všechny skripty zodpovědné za správné načtení aplikace při jejím spuštění.
- Adresář *config* obsahuje různá nastavení. Od jména aplikace, přes nastavení cache, hashování, systému ukládání souborů, až po možnosti připojení databáze.
- Adresář *database* uchovává informace o migracích, zodpovědných za správné vytvoření tabulek v databázi.

- Adresář *node_modules* je využíván aplikací Node.js, která umožňuje automatické překládání zdrojových souborů v reálném čase.
- Adresář *public* uchovává soubory, které jsou předloženy uživateli při navštívení webu aplikace. Také uchovává optimalizované a sjednocené Javascript a CSS soubory.
- Adresář *resources* obsahuje dokumenty formátu Sass (CSS obohacené o skripty), Javascript a šablony webových stránek ve formátu blade.php umožňující dynamické změny v obsahu.
- Adresář *routes* obsahuje všechny soubory definující cesty aplikace.
- Adresář *storage* obsahuje cache, nahrané soubory či soubory spjaté s uživatelským sezením.
- Adresář *tests* zahrnuje všechny testy aplikace.
- Adresář *vendor* obsahuje soubory potřebné pro správnou funkci aplikace Composer.
- Další důležitý soubor je *.env*, který definuje nastavení aplikace, připojení k databázi či e-mailovému serveru.

  Více o adresářové struktuře Laravel projektu naleznete v oficiální dokumentaci [14].

4.4 HTML

Hypertextový značkový jazyk (Hypertext Markup Language, HTML), je značkovací jazyk používaný při tvorbě webových stránek. Řadí se na první místo v nejpoužívanějších jazycích pro tvorbu stránek v systému World Wide Web, který umožňuje zveřejňování dokumentů na internetu. Jazyk HMTL je charakterizován množinou značek a jejich vlastností specifikovaných pro danou verzi. Původně byl HTML podmnožinou jazyka SGML, ale díky trendům internetu se postupně vyvíjel. Za specifikaci jazyka je odpovědná organizace W3C a aktuální verzí je HTML 5.3.

HTML dokument je strukturován do několika základních částí:

- Deklarace dokumentového typu – značka <!DOCTYPE html>
- Kořenový element – párová značka <html> a </html>, reprezentuje celé tělo dokumentu
- Hlavička dokumentu – párová značka <head> a </head> - obsahuje metadata dokumentu, např. kódování, titulek a popisek stránky či importování dalších souborů jako jsou Javascript či CSS.
- Tělo dokumentu – párová značka <body> a </body> - zahrnuje vlastní obsah dokumentu

Syntaxe toho jazyka je poměrně jednoduchá a zaměřuje se více na pestrou škálu možností prezentace informací (typy a váha písma, řádkování, tabulky aj.). Ke tvorbě webových stránek existují také různé grafické nástroje ulehčující práci nezkušeným uživatelům. V dnešní době si nevystačíme se statickým obsahem webu, a proto bývá obsah dynamicky vygenerován na straně serveru za použití např. PHP a odeslán uživateli k prohlížení. Pro zlepšení uživatelského dojmu jsou stránky formátovány pomocí CSS. [15]

4.5 CSS

Kaskádové styly (Cascading Style Sheets, CSS) je jazyk pro zapsání informace o způsobu zobrazení daného elementu zapsaného např. v HTML. Za standardizací jazyku stojí opět organizace W3C. Účelem vytvoření tohoto jazyka bylo oddělit vzhled dokumentu od jeho struktury a obsahu.

Kaskádové styly jsou aplikovány v HTML dokumentu různými způsoby:

- Přímým zápisem značkou <style> a </style>

- Importováním externího souboru značkou <link>
- Importování, externího souboru v http hlavičce
- Přímým zápisem atributem *style* dané značky

Čerpáno z [16].

4.6 Webový server Nginx

Nginx je webový server s otevřeným zdrojovým kódem. Podporuje nejmodernější protokoly jako HTTP, HTTPS, SMTP, POP3, IMAP a SSL. Jeho prioritou je vysoký výkon a nízké nároky na paměť. Pracuje na operačních systémech Unix, Linux, BSD, ale také existují verze pro macOS, Windows či Solaris.

Velkou výhodou serveru Nginx je nastavitelná cache, ve které se pokusí vyhledat odpověď na požadavek jako statický obsah a pokud jej najde, ihned odpoví. Pokud obsah nebude nalezen, obrátí se na jeden z nastavených serverů. Pokud mu ani tento server neodpoví, zeptá se serveru záložního. Když získá odpověď, uloží si ji do cache pro příští rychlejší zpracování.

Další nastavení umožňuje specifikovat maximální počet připojení z jedné IP adresy, čímž upevní zabezpečení svých aplikací proti DOS útokům. [17]

4.7 Databázový server MySQL

MySQL je multiplatformní databázový systém s otevřeným zdrojovým kódem. Pro svou vysokou výpočetní rychlost spolu s malými nároky na paměťový prostor se MySQL začal ve velkém využívat při tvorbě webových aplikací. Spolu s webovým serverem komunikují pomocí jazyka SQL (Structured Query Language). Správu údajů na straně databázového serveru zajišťuje tzv. DBMS (DataBase Management System). Ten komunikuje s klientem a odpovídá na jeho požadavky. Klient může být např. PHP skript běžící na webovém serveru, který se dotazuje pomocí SQL a zobrazuje poskytnuté výsledky.

K nahlédnutí do databáze existuje spoustu programů, mezi které můžeme zařadit také phpMyAdmin. Tento program je napsán v PHP a je většinou poskytován spolu s webovým serverem díky jeho nenáročnosti.

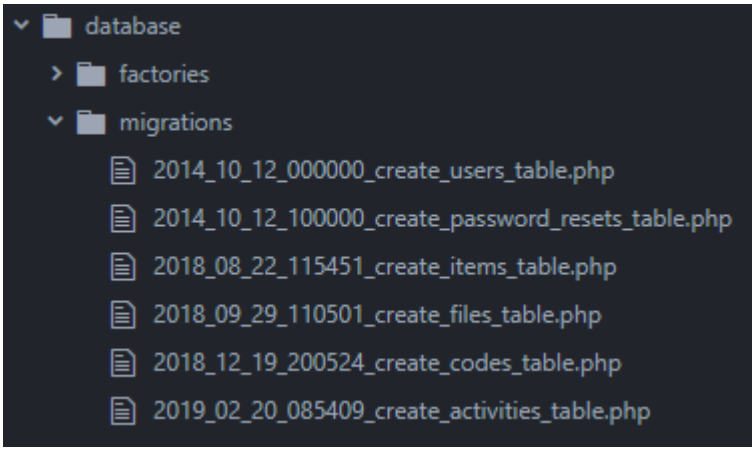
5 Implementace

Implementaci informačního systému bych rozdělil do několika kategorií podle chronologického postupu. Po nainstalování nového projektu jsem musel změnit patřičná nastavení, aby systém fungoval. Jako první jsem musel vytvořit novou databázi, se kterou bude aplikace komunikovat. Jelikož vývoj probíhal na OS Windows, spustil jsem databázový server MySQL pomocí balíčku programů XAMPP a přistoupil do správy databází skrz phpMyAdmin. Zde jsem vytvořil novou databázi a přidělil patřičná oprávnění.

Dalším krokem byla změna údajů v souboru *.env*, kde se nastavuje připojení k databázi a stav aplikace. Jakmile jsem vše nastavil a spustil webový server, bylo již možné navštívit uvítací stránku projektu. Databáze zatím byla prázdná, a tak jsem musel vytvořit její strukturu.

5.1 Vytvoření databázové struktury

Framework Laravel umožňuje vytvářet tabulky databáze pomocí třídy *Migration*. Za použití vestavěného rozhraní Artisan [13] jsem vytvořil příslušné migrace, které se ukládají v adresáři *database/migrations* (viz Obrázek 9), a jejich obsah upravil podle ER diagramu. Následné volání příkazu *php artisan migrate* vytvořilo v databázi požadované tabulky.

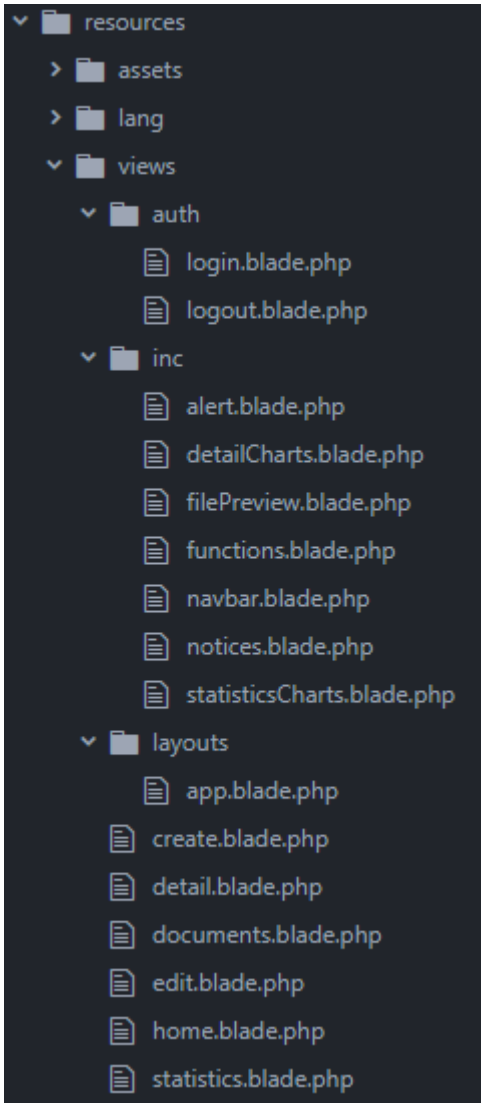


Obrázek 9: Migrace

5.2 Šablony webových stránek

V dalším kroku jsem vytvořil hlavní HTML šablonu aplikace, ze které čerpají všechny stránky mého systému. Tato šablona se nachází v *resources/view/layouts* a jmenuje se *app.blade.php*. Laravel pro šablony používá formát zápisu blade.php, což umožňuje nahrazení proměnných nebo celých sekcí stránky při kompilaci. Jednoduše řečeno, do jedné šablony lze vložit různé sekce podle toho, kterou stránku zrovna uživatel navštíví. V hlavní šabloně je deklarace hlavičky HTML dokumentu, kde jsou importovány Javascripty a CSS. Důležité bylo také nastavit metatag s názvem *robots* na hodnotu *noindex, nofollow*, aby vyhledávací roboti tento web ignorovali a nenabízeli ve výsledcích vyhledávání.

Další šablony (sekce) jsem vytvořil pro konkrétní stránky a jsou dynamicky vkládány do hlavní šablony. Pro lepší přehlednost jsem samostatné menší prvky šablon oddělil do vlastních souborů a jsou uloženy v podadresáři *inc*. Tak stejně šablony pro přihlašovací a odhlašovací stránku jsou uchovány v adresáři *auth*. Všechny šablony jsou na Obrázku 10. Tyto šablony se v MVC schématu považují jako pohled (view).



Obrázek 10: Šablony

5.3 Cesty

Cesty (routes) jsou důležitým prvkem MVC schématu. Řadič (controller) podle zadané URI adresy vygeneruje pohled (view) specifikovaný v souboru *routes/web.php*. Například cesta *Route::get('/documents', 'SearchController@documentsView');* nám říká, že při HTTP požadavku GET na URL stránky informačního systému následované cestou */documents* má řadič (konkrétně *SearchController*) zavolat metodu *documentsView*, která vrátí pohled *documents*, tedy webovou stránku skladu dokumentů.

Existují také dynamické cesty, kde se ze zadané URI adresy extrahuje příslušná část a předá se řadiči jako parametr.

Například *Route::get('/edit/{id}', 'SearchController@editView');* nám říká, že při návštěvě adresy */edit/42* řadič zavolá metodu *editView* s parametrem 42 a vrátí dynamicky vygenerovaný pohled pro úpravu položky s ID 42.

Cesty rozlišují metody HTTP požadavků a podle toho vedou k jiným cílům. Cesta pro HTTP požadavek s metodou GET může vést jinam než pro požadavek s metodou POST. Laravel podporuje následující metody požadavků: GET, POST, PUT, PATCH, DELETE, OPTIONS. [13]

5.4 Řadiče

Řadiče (controllers) jsou dalším prvkem MVC schématu. Místo toho, abychom definovali logiku příchozích požadavků v souboru s cestami, využijeme právě řadičů. Ty nám umožňují přehledné zapsání metod volaných v souboru s cestami. Řadiče také vykonávají veškeré dotazy do databáze, zpracovávají je a předávají pohledům. V konstruktoru každého řadiče využívám bezpečnostní mechaniku zabudovanou v Laravelu zvanou *middleware*. Ta umožňuje pouze autentizovaným uživatelům použít daný řadič a vstoupit "hlouběji" do aplikace. Pokud uživatel není přihlášený, tento mechanismus jej přesměruje na stránku pro přihlášení. [13]

Řadiče jsem opět vytvořil příkazem v rozhraní Artisan:

php artisan make:controller NÁZEV

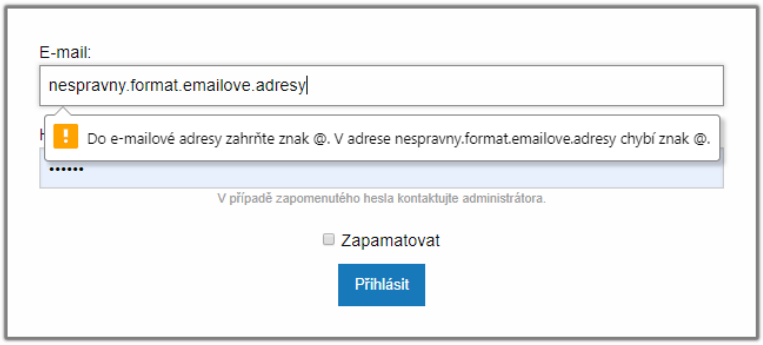
a rozdělil si je podle jejich využití. Řadič *SearchController* obsahuje metody pro zobrazení všech pohledů (views), metody pro vyhledání zakázek a dokumentů v databázi a metody pro dopředné doporučování slov při vyhledávání (typeahead). Řadič *ItemsController* obsahuje metody spojené s konkrétní položkou (vytvoření, upravení, nahrání a mazání dokumentů). Řadič *UserController* pro úpravu dat spjatých s jednotlivým uživatelem (poznámka). Řadič *ChartController* zpracovává požadavky na zobrazení grafů.

5.5 Implementované funkce systému

V této části kapitoly detailně popíšu všechny implementované funkce mého informačního systému. Veškerá funkcionalita je naprogramována v PHP s pomocí Javascriptu pro vylepšení uživatelského dojmu.

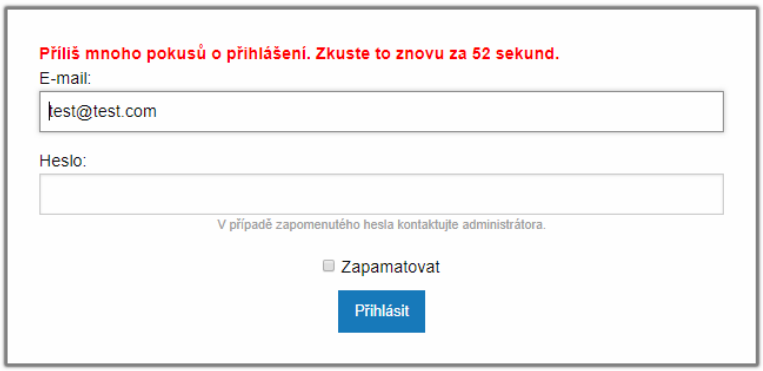
5.5.1 Přihlášení

Pokud není uživatel do systému přihlášen a pokusí se navštívit kteroukoliv stránku, bude přesměrován na přihlašovací stránku */login*. Zde musí uživatel zadat správnou e-mailovou adresu a heslo. Typ políčka pro e-mail (atribut *type* značky <input>) je nastaven na "email", což přikáže prohlížeči provést kontrolu správného formátu adresy (viz Obrázek 11) a upozornit uživatele v případě nevhodného formátu. Obě políčka mají také atribut *required*, což přikáže prohlížeči provést kontrolu, zdali se v polích nachází alespoň 1 znak. Pokud by uživatel záměrně zkoušel odeslat políčka prázdná (upravil by si zmiňovanou kontrolu v HTML), systém vrátí chybovou hlášku, že jsou políčka povinná.



Obrázek 11: Pokus o přihlášení s nesprávným formátem e-mailu

Pokud uživatel zadá 5x nesprávné údaje, je kombinace daného uživatelského jména a IP adresy zablokována na 1 minutu (viz Obrázek 12).



Obrázek 12: Příliš mnoho pokusů o přihlášení

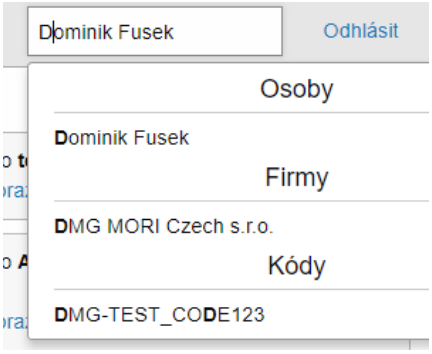
Uživatel může zvolit možnost “Zapamatovat”, v tom případě se uloží do databáze do tabulky *users* danému uživateli tzv. *remember_token* a stejný token se uloží do cookies v prohlížeči uživatele s platností 5 let.

Po úspěšném přihlášení je uživatel přesměrován na hlavní stránku (Přehled).

5.5.2 Přehled

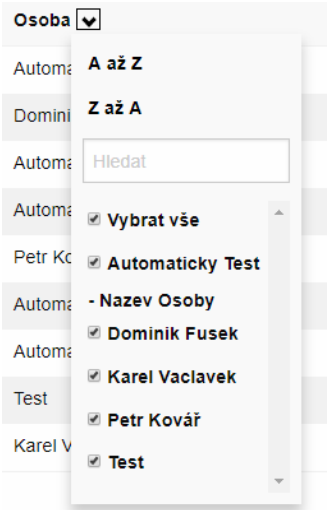
Na této stránce uživatel vidí hned několik bloků. Na horní části stránky se nachází menu a vyhledávání. Během implementace bylo přidáno tlačítko pro odhlášení ze systému, jelikož v návrhu chybí. Největší část zabírá přehled všech položek (zakázek) a po pravé straně se zobrazují upozornění a poznámky uživatele.

Po načtení stránky se pomocí Javascriptu a Ajaxu pošle několik požadavků. 3 požadavky jsou na načtení osob, firem a kódů z databáze pro dopředné doporučování slov (typeahead) při psaní ve vyhledávacím poli (viz Obrázek 13).



Obrázek 13: Dopředné doporučování slov ve vyhledávání

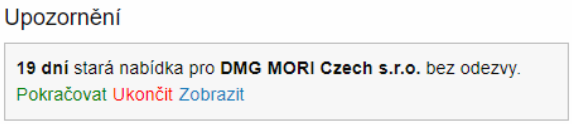
Jeden požadavek na načtení všech zakázek seřazených od nejnovější. Po úspěšném provedení Ajaxu je odpověď zpracována a upraven obsah tabulky. Stejný požadavek je odeslán při změně hodnoty ve vyhledávacím poli, s tím rozdílem, že požadavek také obsahuje vyhledávanou frázi. Řadič pak porovná záznamy v databázi na shodu vyhledávané fráze se jménem osoby, firmy či kódu a vrátí příslušné zakázky. Uživatel také může v tabulce nastavit filtry podle jednotlivých sloupců, stejně jako v programu Excel (viz Obrázek 14). K tomu je použita Javascript knihovna excel-bootstrap-table-filter.js.



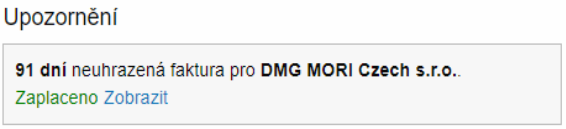
Obrázek 14: Filtrování v tabulce

Upozornění v pravé části stránky se objevují automaticky, a to ve dvou případech:

- Zakázka je ve stavu “Nabídnuto” déle jak 5 dní (viz Obrázek 15)
- Zakázka je ve stavu “Odesláno” déle jak 90 dní (viz Obrázek 16)



Obrázek 15: Upozornění na starou nabídku



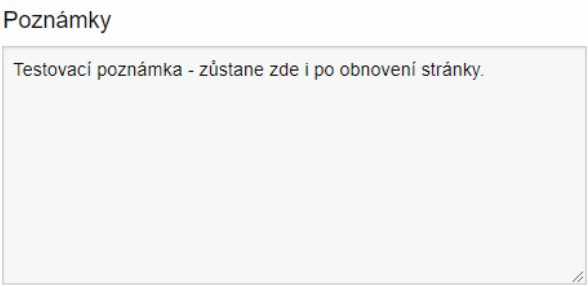
Obrázek 16: Upozornění na dlouho neuhrazenou fakturu

V prvním případě upozornění obsahuje 3 tlačítka. Kliknutím na zelené tlačítko “Pokračovat” se zakázce nastaví stav “Objednáno”. Kliknutím na červené tlačítko “Ukončit” se zakázce nastaví stav “Neúspěch” a kliknutím na modré tlačítko “Zobrazit” přejdete na detail dané položky.

V druhém případě upozornění nabízí pouze 2 tlačítka. První, zelené tlačítko “Zaplaceno”, nastaví stav zakázky na “Hotovo” a tím je zakázka uzavřena. Druhé, modré tlačítko “Zobrazit”, přesměruje uživatele na detail dané položky.

V obou případech použití tlačítek “Pokračovat”, “Ukončit” nebo “Zaplaceno” skryje pomocí Javascriptu dané upozornění a v budoucnu již nebude zobrazeno.

Pod upozorněními se nachází textové pole pro uživatelské poznámky. (viz Obrázek 17) Zapsáním textu do okna a následným opuštěním okna (tzv. “focusout”) se odešle Ajax požadavek a poznámka se uloží do databáze k danému uživateli. Tento prvek také nebyl součástí návrhu, zákazník požádal o přidání během konzultace.



Obrázek 17: Uživatelská poznámka

Nejdůležitější částí hlavní stránky (Přehledu) je však největší blok, a to tabulka se zakázkami (viz Obrázek 18). Zakázky jsou zde řazeny od nejnovější a uživatel může vidět:

- Pořadí zakázky
- Název firmy, se kterou zakázka probíhá
- Název osoby, se kterou o zakázce jedná
- Status (stav) zakázky
- Datum vložení zakázky do systému
- Stát, ze kterého zákazník pochází
- Kategorie zákazníka

Kliknutím na příslušný řádek zakázky přejde uživatel na stránku s detailem.

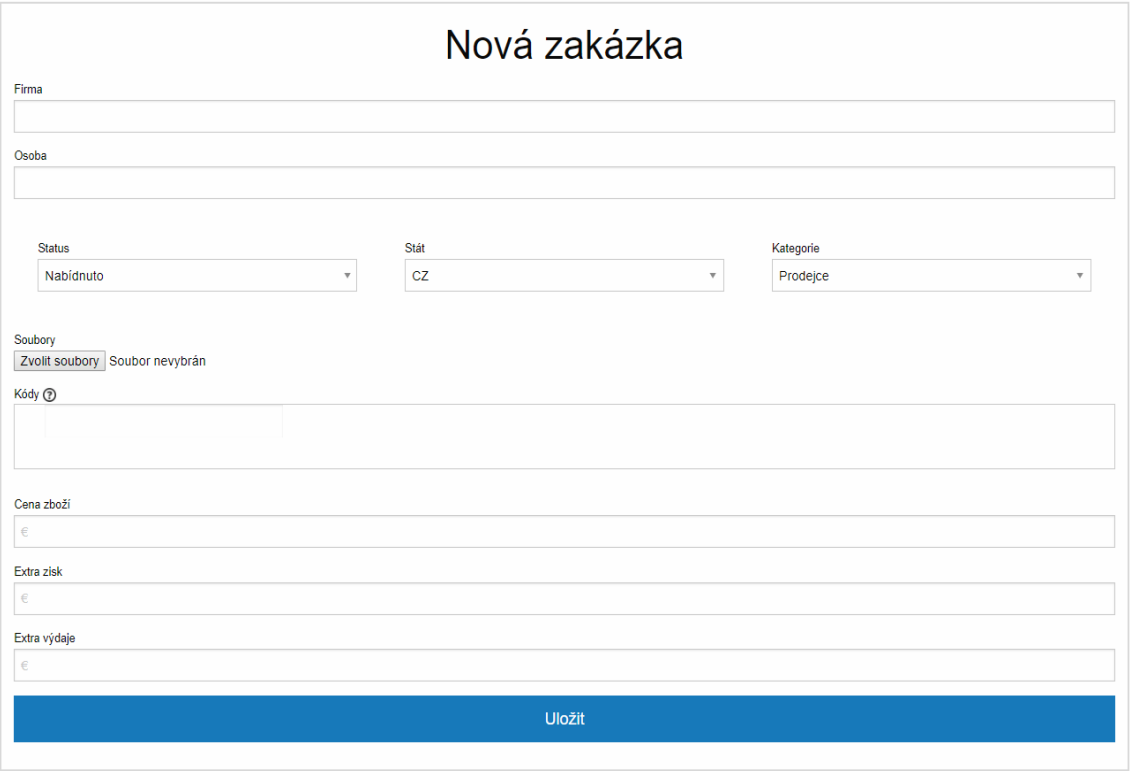
#	Firma	Osoba	Status	Datum	Stát	Kategorie	
10	Automaticky Test - Nazev Firmy	Automaticky Test - Nazev Osoby	Nabídnuto	27-02-2019	CZ	Prodejce	
9	DMG MORI Czech s.r.o.	Dominik Fusek	Objednáno	27-02-2019	CZ	Prodejce	
8	Automaticky Test - Nazev Firmy	Automaticky Test - Nazev Osoby	Objednáno	27-02-2019	CZ	Prodejce	
7	Automaticky Test - Nazev Firmy	Automaticky Test - Nazev Osoby	Objednáno	27-02-2019	CZ	Prodejce	
6	VUT v Brně	Petr Kovář	Objednáno	27-02-2019	CZ	Prodejce	
5	Automaticky Test - Nazev Firmy	Automaticky Test - Nazev Osoby	Objednáno	27-02-2019	CZ	Prodejce	
4	Automaticky Test - Nazev Firmy	Automaticky Test - Nazev Osoby	Objednáno	27-02-2019	CZ	Prodejce	
3	test	Test	Objednáno	27-02-2019	CZ	Prodejce	
1	DMG MORI Czech s.r.o.	Karel Vaclavek	Odesláno	12-12-2018	CZ	Prodejce	

Obrázek 18: Tabulka se zakázkami

5.5.3 Vytvoření nové zakázky

Tvorba nové položky je velmi intuitivní. Uživatel klepnutím na tlačítko v menu “Nový” přejde na formulář, kde stačí vyplnit pouze 2 povinná pole. Prvním je “Firma”, kde uživatel uvede název firmy spojené se zakázkou a druhým je “Osoba”, kde podobně uvede jméno osoby, se kterou jedná. Při psaní do těchto polí jsou automaticky nabízeny shody s již dříve vytvořenými položkami, aby uživatel nechtěně nevytvořil položku s podobným názvem (např. rozdíl v chybějící tečce či malém/velkém písmenu) a neudělal tak zmatek v systému. Další pole “Status”, “Stát” a “Kategorie” fungují na principu výběru z nabízeného seznamu.

V této chvíli by měla být již zákazníkovi vystavena nabídka, tudíž by uživatel měl tento soubor (popřípadě množinu souborů) vybrat a nahrát do systému. Nahrání může provést jednoduchým přetažením souboru na tlačítko “Zvolit soubory” nebo po kliknutí vyhledat soubory v adresářové struktuře počítače. V nabídce se také nachází její unikátní číslo, to by měl uživatel zaznamenat do pole “Kódy” pro snadnější vyhledání položky v budoucnu. Taktéž tam nalezne cenu zboží a měl by ji uvést do příslušného pole. Pokud v této fázi existují mimořádné výdaje či zisky spjaté se zakázkou, může uživatel využít poslední 2 pole. Tyto položky byly přidány později na žádost zákazníka. Veškeré uvedené informace se dají později upravit. Po vytvoření položky je uživatel přesměrován na hlavní stránku, kde v tabulce ihned uvidí novou zakázku.



Obrázek 19: Tvorba nové zakázky

5.5.4 Detail zakázky

Na stránce detailu zakázky může uživatel vidět podrobnější informace. Jako první upoutá jeho pozornost název firmy a osoba, se kterou je zakázka spjata. Hned vedle je barevně rozlišen stav, v jakém se zakázka nachází a pořadí zakázky. Stavy mohou být následující:

- Nabídnuto – oranžová, firmě byla udělána nabídka

- Objednáno – žlutá, firma nabídku přijala a zboží bylo objednáno
- Potvrzeno – žlutá, objednávka byla potvrzena a čeká se na expedici zboží
- Odesláno – žlutá, zboží bylo vyexpedováno, čeká se na platbu
- Hotovo – zelená, faktura za zboží byla zaplacena, zakázka je ukončena
- Neúspěch – červená, zakázka byla neúspěšně uzavřena

Pod indikátorem stavu se zobrazují unikátní kódy, které uživatel k zakázce přidá. Tyto kódy se nachází v příchozích dokumentech a ulehčují následné vyhledání zakázky.

V tabulce, mimo informace dostupné na hlavní stránce, může uživatel najít cenu zboží, automaticky vypočítanou provizi z prodeje (provize se liší v závislosti na kategorii zákazníka), extra zisk a extra výdaje spojené se zakázkou a celkový zisk ze zakázky. Všechny hodnoty jsou vedeny v eurech (€). Pokud je stav změněn na "Odesláno", automaticky se nastaví datum v řádku "Zboží odesláno".

Pod tabulkou se nachází textové pole na poznámky, které může uživatel použít v případě, potřebuje-li si něco poznamenat ke konkrétní zakázce. Tyto poznámky jsou při opuštění okna uloženy pomocí Javascriptu a Ajaxu do databáze k dané zakázce.

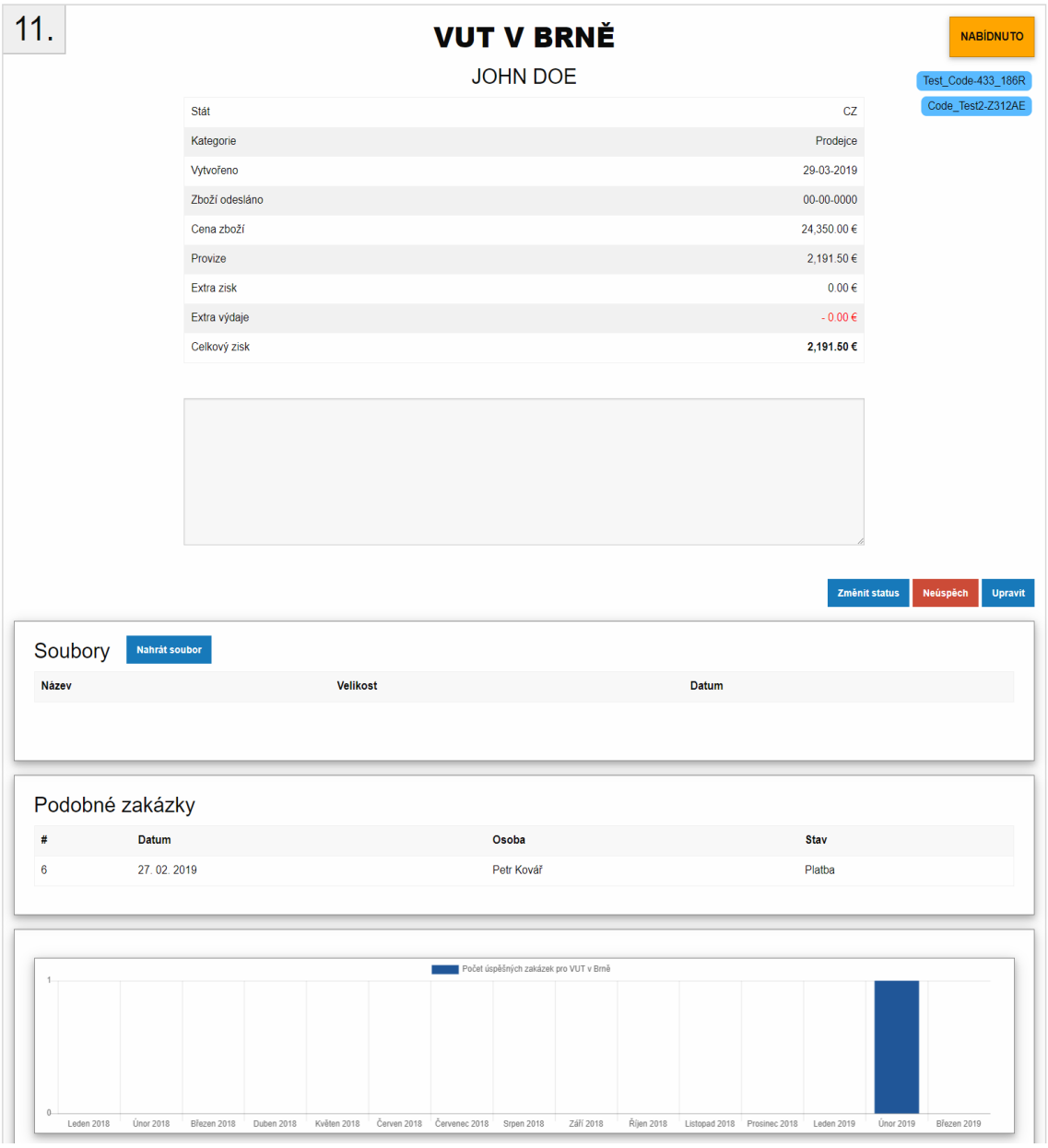
Dále může uživatel spatřit několik tlačítek. Jejich počet záleží na stavu, ve kterém se objednávka nachází, nicméně tlačítka jsou následující:

- Změnit status – nastaví stav zakázky na následující
- Neúspěch – nastaví stav zakázky na "Neúspěch"
- Upravit – přesměruje uživatele na stránku pro úpravu zakázky

Přímo pod tlačítky se nachází blok týkající se souborů souvisejících s danou zakázkou. Uživatel může intuitivně tlačítkem "Nahrát soubor" zvolit a následně nahrát množinu souborů z počítače. Ze souborů se při nahrávání extrahuje název, který se spolu s cestou, kam bude soubor na serveru uložen, zapíšou do databáze. Po úspěšném nahrání se v tabulce zobrazí všechny soubory. U každého souboru je zobrazena ikona reprezentující typ souboru, jeho název, velikost a datum nahrání. Pokud se jedná o typ souboru podporován online Google prohlížečem dokumentů, objeví se vedle názvu souboru ikona pro prohlédnutí tohoto dokumentu ve zvláštním okně. Pokud uživatel klikne na název souboru, bude soubor stažen do počítače. Na konci každého řádku se nachází ikona pro odstranění souboru. Použití této ikony nenávratně vymaže soubor ze serveru i příslušné záznamy z databáze.

Další blok obsahuje "Podobné zakázky". Tyto zakázky jsou vyhledány na základě podobnosti s položkou, kterou si uživatel prohlíží. Zejména jsou zobrazeny ty, které se týkají stejné firmy a jsou seřazeny od nejnovější. Zobrazováno je pouze rozumné množství. U každé zakázky může uživatel vidět datum jejího vytvoření, osobu, se kterou je položka spjata a stav, ve kterém se nachází.

Poslední část je vyhrazena pro grafy. Data grafů jsou načtena z databáze pomocí technologie Ajax a zobrazeny pomocí Javascript knihovny Chart.js. První graf udává počet úspěšných zakázek dané firmy za minulý a stávající rok. Graf druhý zobrazuje podíl jednotlivých stavů, ve kterých se nachází zakázky pro danou firmu za všechna období.



Obrázek 20: Detail zakázky

5.5.5 Úprava zakázky

Pokud chce uživatel upravit detaily zakázky, použije tlačítko “Upravit” na stránce detailu konkrétní zakázky. Bude přesměrován na jednoduchý formulář předvyplněný aktuálními informacemi, které může změnit. Při zadávání znaků do pole “Firma” a “Osoba” jsou automaticky doporučovány shody z databáze, aby uživatel omylem neuvedl jméno firmy či osoby s malou odchylkou (např. chybějící tečka) a neudělal tak v systému zmatek. Narozdíl od vytváření nové zakázky, pokud uživatel dodatečně upraví cenu zboží, musí také upravit hodnotu provize, pokud se změnila.

5.5.6 Dokumenty

V menu se také nachází položka “Dokumenty”. Kliknutím uživatel přejde do digitálního skladu všech dříve nahraných dokumentů. Zobrazení je podobné jako v detailu konkrétní zakázky, s tím rozdílem,

že na každém řádku je také uvedena klientská firma a osoba, se kterou dokument souvisí. Také vyhledávání na této stránce je jiné. Uživatel může vyhledávat dokumenty pouze podle jejich názvu nebo je filtrovat pomocí šipky vedle názvu jednotlivých sloupců tabulky.

Název	Velikost	Firma	Osoba	Datum
Proj_1.pdf	266.18 KB	VUT v Brně	Petr Kovář	14-03-2019 10:49:30
Sdeleni MSMT_terminy SC MZ_jaro 2019.pdf	131.78 KB	DMG MORI Czech s.r.o.	Dominik Fusek	14-03-2019 08:46:37
5246.txt	91.81 KB	DMG MORI Czech s.r.o.	Dominik Fusek	14-03-2019 08:41:35
ZI19-00012.pdf	131.75 KB	DMG MORI Czech s.r.o.	Karel Vaclavek	22-02-2019 11:43:47

Obrázek 21: Tabulku dokumentů

5.5.7 Statistiky

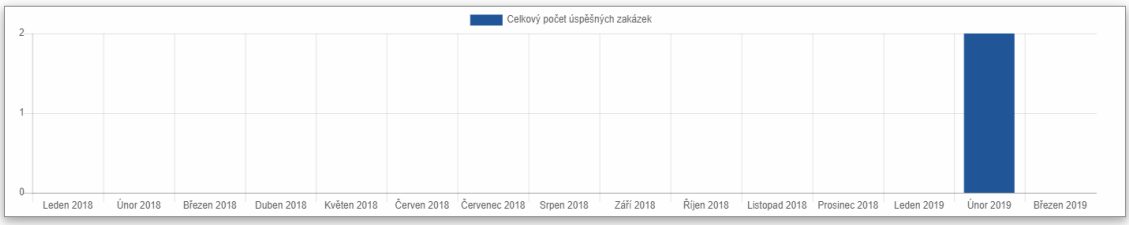
Poslední položkou menu jsou “Statistiky”. Na této stránce je k dispozici široký přehled informací o všech zakázkách. Uživatel si v horní části může zvolit, za jaké období si přeje statistiky zobrazit. Mezi monitorované údaje patří:

- Celkový počet zakázek
- Počet úspěšně uzavřených zakázek
- Počet neúspěšně uzavřených zakázek
- Počet probíhajících zakázek
- Počet neuhrazených odeslaných zakázek
- Celková cena zboží
- Celkové provize
- Celkový extra zisk
- Celkové extra výdaje
- Celkový zisk

Pod tabulkou se nachází několik užitečných grafů. První z nich zobrazuje počet úspěšně uzavřených zakázek za každý měsíc minulého a aktuálního roku. Druhý graf zachycuje počet úspěšně uzavřených zakázek s jednotlivými klientskými firmami. Graf třetí ukazuje celková zisk v jednotlivých měsících minulého a aktuálního roku a poslední graf ukazuje celkový zisk ze zakázek konkrétních firem.

Všechna období

Celkový počet zakázek	10
Počet úspěšně uzavřených zakázek	2
Počet neúspěšně uzavřených zakázek	0
Počet probíhajících zakázek	8
Počet neuhrazených odeslaných zakázek	2
Celková cena zboží	45,303.00 €
Celkové provize	4,100.00 €
Celkový extra zisk	0.00 €
Celkové extra výdaje	- 0.00 €
Celkový zisk	**4,100.00 €**



Obrázek 22: Statistiky

6 Zabezpečení

Důležitým prvkem informačních systémů je také jeho zabezpečení. V dnešní době přibývá moderních technologií a s tím souvisí i narůstající počet jak online, tak i off-line útoků. Studie na Marylandské univerzitě ukázala, že v průměru dojde k útoku na počítače s připojením k internetu každých 39 sekund. Velmi často (až 65 % případů) jsou útoky cíleny na malé a střední podniky. Ty si většinou nemohou z finančního hlediska dovolit experty na zabezpečení nebo nejsou až tak seznámeny s potenciální hrozbou. Naopak velké podniky mají infrastrukturu přizpůsobenou proti kyberútokům.

Článek z roku 2018 říká, že až 64 % společností zjistilo útok na jejich webové služby. 62 % zaznamenalo tzv. *phishing* (podvodná technika k získání citlivých údajů) nebo *social engineering* (způsob manipulace lidmi za účelem získání určité informace), 59 % společností objevilo nežádoucí kód ve svých aplikacích či aktivitu nevyžádaných robotů a 51 % zažilo výpadek systému v důsledku tzv. *DoS* (Denial of Service) útoku.

Společnost Cybersecurity Ventures uvedla, že dochází k ukradení 44 různých záznamů na internetu každou sekundu, což činí 158 727 ukradených záznamů každou hodinu.

Zajištění bezpečnosti bych rozdělil do 3 kategorií podle toho, čeho se týkají.

Čerpáno z [18].

6.1 Zajištění bezpečnosti dat

I když se nám povede zabezpečit uložená data proti kyberútokům, stále nám hrozí riziko fyzického poškození úložiště, s čímž souvisí ztráta dat. Tomu můžeme předejít pravidelným zálohováním dat na dvou či více fyzicky odlišných nosičích. Uživatelé využívající služeb webových hostingů, cloudů aj. by zpravidla nemusel tento krok trápit, jelikož mnoho společností poskytující tyto služby provádí zálohy automaticky.

Další důležitý prvek je nutnost zajistit důvěrnost dat. Pokud ukládáme citlivá data, měli bychom zajistit, aby se k nim mohly dostat pouze pověřené osoby. K tomu většinou slouží identifikace a autentizace uživatele. Nejběžnější formou autentizace v informačních systémech je přihlášení pomocí uživatelského jména a hesla. Tento způsob je však nejméně spolehlivý, jelikož v případě úniku hesla útočník ihned získává přístup k systému. Větší zabezpečení pak poskytuje tzv. dvoufázové ověření, kdy uživatel musí potvrdit přihlášení druhým nezávislým způsobem ověření totožnosti (např. zadat kód z SMS).

Čerpáno z [19].

6.2 Zajištění bezpečnosti služeb

Aby mohl uživatel službu používat, je nutno zajistit její dostupnost. Tu můžeme zajistit ochranou v případě výpadku elektrické energie (záložní zdroje), výpadku sítě (záložní síťová připojení) nebo

duplikací hardwaru a softwaru pro případ závady. Zde spadá také dříve zmíněný útok DoS, kterému můžeme zabránit správným nastavením síťové infrastruktury (firewall aj.). Avšak pokud pro své webové aplikace využíváte služeb jiných společností, spadá toto zabezpečení do jejich režie. Důležité je také zajistit spolehlivý a bezpečný přenos důvěrných informací mezi zdrojem a cílem.

Čerpáno z [19].

6.3 Zajištění bezpečnosti programových prostředků

Programy, které vývojář informačního systému zvolí, hrají důležitou roli v zabezpečení. Pokud použije zastaralé programy bez aktivní podpory jejich vývojářů, podstupuje riziko, že se v budoucnu mohou v programu objevit díry, které se dají zneužít a ohrozit tak bezpečnost celého systému. Zkušený vývojář by měl zvolit ty programy, které jsou prověřené a mají v budoucnu potenciál. Nestačí však program jednou nainstalovat a nestarat se o něj, důležité jsou pravidelné aktualizace softwaru. Správce systému by měl stále zůstat v obraze a zajímat se o nejmodernější technologie a postupně nahrazovat staré. Velmi důležitá je také správná konfigurace programu a jeho použití. Nesprávné nastavení může vytvořit trhlinu v zabezpečení, proto je důležité následovat dokumentaci programů.

I když použijeme nejspolehlivější programy a všechny je nastavíme správně, pořád hrozí riziko, že vývojář udělal nějakou chybu ve zdrojovém kódu aplikace, které si všimne potenciální útočník. Mezi takové chyby patří třeba nezajištění odolnosti proti chybným vstupním datům (buffer overflow, SQL injection), špatné nastavení přístupových práv k souborům nebo tzv, *cross-site request forgery* (CSRF).

Čerpáno z [19].

6.4 Nejběžnější útoky na informační systémy

V této kapitole bych se rád věnoval nejběžnějším útokům na informační systémy dopodrobna a jak se proti nim bránit.

6.4.1 Brute force

Útok hrubou silou (angl. Brute force attack) je technika, kdy se útočník snaží získat přístup do systému pomocí systematického testování všech možných kombinací nebo jejich podmnožiny. Tento způsob se často používá pro získání dvojice uživatelské jméno a heslo. Pokud se útočník snaží získat heslo pomocí předem připraveného seznamu (slovníku), jedná se o tzv. Slovníkový útok. Jelikož uživatelé často používají velmi jednoduchá a krátká hesla, je tento útok velmi úspěšný a velice rozšířený. [20]

Hlavní prevence proti útoku hrubou silou je dostatečně silné heslo. Jelikož útočník zkouší všechny možné kombinace systematicky, přidáním dalšího znaku do vašeho hesla zvýšíte potřebnou dobu k prolomení exponenciálně. Proto se velmi často požaduje po uživateli heslo alespoň o délce 8 znaků. Důležité je také použití v hesle kombinace velkých a malých písmen, číslic a speciálních

znaků. To znemožní útočníkovi zjistit heslo při použití jednodušších kombinací jako např. pouze malá písmena či pouze číslice. Uvádí se, že útok hrubou silou přestává být kvůli časové náročnosti použitelný při délce hesla 12 znaků.

Skvělým nástrojem na zjištění přibližné doby potřebné k prolomení daného hesla je tato webová stránka [21]. Zde se uživatel dozví, jestli se jeho heslo nenachází v seznamu uniklých nebo nejpoužívanějších hesel (v tomto případě je heslo prolomeno ihned) nebo jak dlouho by jeho heslo obstálo. Například heslo skládající se z 5 malých písmen by trvalo prolomit přibližně 0.0003 sec. Naopak heslo obsahující 8 znaků (velká a malá písmena, číslice a speciální symboly) by trvalo prolomit odhadem 9 hodin až několik dní.

I když uživatelé nepoužívají silná hesla, měl by být systém schopen zabránit útočníkovi v jejich prolomení, popřípadě zabránit v přístupu po úspěšném prolomení. To lze docílit například sledováním IP adresy uživatele, ze které se přihlašuje, a v případě její změny zablokovat daný účet nebo blokovat všechny příchozí požadavky z dané IP adresy po několika neúspěšných pokusech o přihlášení. Další možnosti je vést tzv. *whitelist* nebo *blacklist* IP adres, které se do systému mohou přihlásit, respektive nemohou. Tímto způsobem se dá také rozlišit přístupy podle geolokace. Na Obrázku 23 můžeme vidět 10 IP adres, ze kterých pochází nejvíce brute force útoků na webové služby uživatelů používajících plugin WordFence pro redakční systém WordPress za březen 2017. [22] Vidíme, že spousta z nich pochází z Ukrajiny a Ruska.

	Rank	IP	Total Attacks (MM)	Feb Rank	Jan Rank	Brute Force Attacks	Complex Attacks	ISP Name	Country Code	Country
1	1	185.159.36.6	15.86	2	3	0.00	15.86	Phoenix LLC	RU	Russia
2	2	193.201.224.205	12.63	1	1	0.00	12.63	PE Tetyana Mysyk	UA	Ukraine
3	3	91.200.12.15	9.81	24		9.75	0.06	Pp Sks-lugan	UA	Ukraine
4	4	91.200.12.159	7.46			7.41	0.05	Pp Sks-lugan	UA	Ukraine
5	5	91.200.12.103	6.02			5.98	0.04	Pp Sks-lugan	UA	Ukraine
6	6	192.151.148.50	5.86			5.82	0.04	DataShack, LC	US	United States
7	7	193.201.225.17	5.15			5.15	0.00	PE Tetyana Mysyk	UA	Ukraine
8	8	178.137.93.239	4.60			4.48	0.12	Kyivstar GSM	UA	Ukraine
9	9	192.187.98.42	4.56			4.48	0.08	DataShack, LC	US	United States
10	10	91.200.12.33	4.30			4.25	0.05	Pp Sks-lugan	UA	Ukraine

Obrázek 23: Tabulka IP adres, ze kterých pochází nejvíce útoků

Jelikož se uživatelé mého systému budou často přihlašovat z různých míst, blokovat přihlášení z různých IP adres je tedy nemožné. Zvolil jsem proto znemožnění přístupu na 1 minutu po několika nesprávných zadání uživatelského hesla. To znemožní použití útoku hrubou silou, jelikož po každých 5 špatných pokusech by musel útočník čekat 1 minutu, což by prodloužilo čas potřebný k získání údajů na nereálnou hodnotu.

6.4.2 Denial of Service

Odepření služby (angl. Denial of Service, DoS) je typ kyberútoku, při kterém je napadená webová služba zahlcena obrovským množstvím požadavků v krátkém čase, v důsledku čehož vypadne a je tak nepřístupná ostatním uživatelům. DoS útoky se dají provést několika způsoby. Mezi nejznámější patří následující. [20]

ICMP floods

Tento typ útoků používá k vyřazení napadnutých počítačů tzv. Internet Control Message Protocol pakety. Tyto pakety jsou jedny z nejběžnějších v internetové komunikaci a zajišťují např. odesílání chybových zpráv při dosažení nedostupné služby apod.

Smurf attack je typ DoS, kdy útočník zneužije špatně nastaveného systému, který umožňuje rozeslání paketu všem počítačům v síti pomocí broadcastu. Poté stačí poslat paket s dostatečnou velikostí, který nebude odfiltrován, a všechny počítače v síti jej budou muset zpracovat. [20]

Ping-flood je jednoduchý útok při kterém se zahltí cílový počítač požadavky o odezvu ping. [20]

SYN-flood je útok, při kterém útočník odesílá na cílový počítač TCP/SYN pakety s podvrženou hlavičkou odesílatele. Cílový počítač pakety zpracuje a odesílá nazpět TCP/SYN-ACK paket a čeká na odpověď TCP/ACK. Ta ovšem nikdy nepřijde, jelikož je hlavička původního paketu upravena a tato žádost nějakou dobu blokuje žádosti jiné. [20]

Distributed DoS

Útok DDoS je specifický tím, že narozdíl od útoku DoS přichází požadavky z většího množství různých počítačů. Tyto počítače mohou být klidně majetkem běžných uživatelů, kteří o útoku vůbec netuší a jejich počítače byly pouze infikovány programem nastaveným k útoku na cílovou IP adresu v daný čas. Dalším způsobem je infikovat velké množství počítačů programem (botem), ke kterému má útočník stále přístup. Infikovaný počítač se tak stává tzv. zombie a může být kdykoliv použit k dalšímu útoku. Celá síť takto infikovaných počítačů, které ovládá jeden bot se nazývá *botnet*. DDoS útok má oproti DoS několik výhod. Více počítačů může generovat mnohem více požadavků. Je těžší ošetřit takový útok, jelikož útočící stroje nejsou tak agresivní (menší počet požadavků, ale více strojů) a mohou se tak tvářit jako běžný uživatel. A v neposlední řadě je mnohem těžší vystopovat útočníka. [20]

Nukes

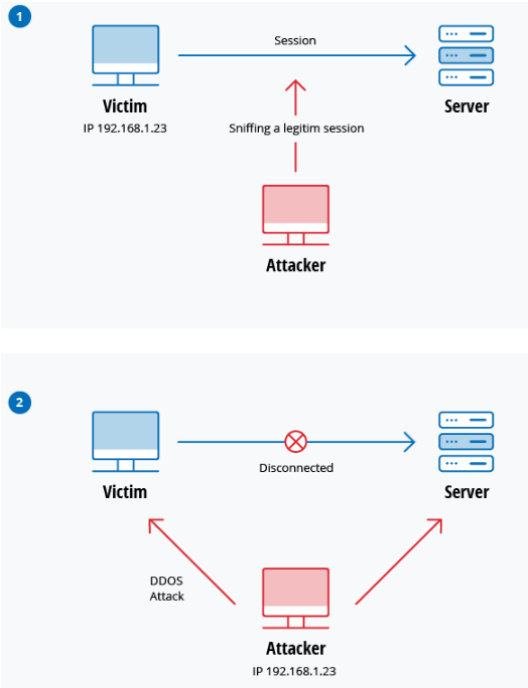
Slovem *nuke* je označován útok, který má za cíl nalézt chybu v systému a zaútočit přímo na ni. Systém pak může zkolabovat při zpracování jediného paketu. [20]

Prevence proti těmto útokům je monitorování sítě a správné nastavení filtračních pravidel provozu. Pokud už napadený systém zkolabuje, je dobré mít záložní systémy na různých místech v síti, které převezmou jeho funkcionalitu. Uživatelé používající dané webové služby tak nemusí důsledek útoku vůbec zaznamenat.

Tento problém řeší společnost Forpsi, na jejichž cloudu informační systém poběží.

6.4.3 Man-in-the-middle

MitM označuje takový typ útoku, kde útočník vstoupí mezi komunikujícího klienta a server. Poté co se klient připojí k serveru, převezme útočník kontrolu nad jeho počítačem a pomocí např. DoS útoku způsobí klientovi zahlcení, aby nemohl nadále komunikovat se serverem. Útočník pak nahradí svou IP adresu adresou napadeného počítače a nadále komunikuje se serverem. Ten si myslí, že stále udržuje komunikaci s původním klientem a nadále spolupracuje (viz obrázek 24). [20]



Obrázek 24: Útok Man-in-the-Middle

Takovému útoku se dá zabránit velmi těžce. Můžeme však ztížit útočníkovi provést DoS útok správným nastavením filtrovacích pravidel firewallu, abychom mohli dále komunikovat se serverem a odstřihnout tak od něj útočníka.

Proti tomuto útoku se musí každý uživatel systému bránit sám. Tento útok by mohl systém ohrozit, nicméně budou uživatelé nabádáni k používání bezpečných (nejlépe privátních) sítí k připojení k internetu.

6.4.4 Phishing

Tato technika spočívá v rozesílání falešných emailů s upravenou adresou odesílatele, tak aby vypadal jako by jej odeslal věrohodný zdroj. V emailu se snaží útočník získat citlivé informace, popřípadě poslat v příloze škodlivý malware. Pokud je email cílen přímo na vaši osobu a útočník si dá tu práci, aby o vás zjistil informace a email přizpůsobil na míru, říkáme tomu *Spear phishing*. V takových emailech často bývají odkazy na podvržené webové stránky, které vypadají totožně se stránkou věrohodnou, ale jejich úkolem je pouze získat vaše údaje. Příkladem může být email, který se tváří jako by jej odeslal administrátor vašeho informačního systému a píše se v něm, že si musíte obnovit heslo. Vy navštívíte podvodnou stránku, zadáte původní heslo a útočník tak získává přístup k vašemu systému. [20]

Spousta emailových klientů pozná, že se jedná o podvodný email a rovnou jej označí jako SPAM. Pokud se však takový email dostane k vám a jeho obsah vypadá podezřele, snaží se z vás vylákat citlivé informace nebo obsahuje podezřelé přílohy, je potřeba daný email ihned vymazat. Zběhlejší uživatelé mohou prozkoumat hlavičku emailu, kde zjistí, že se jedná o podvrh. Důležité je také ihned neklikat na odkazy vedoucí na cizí stránky. Po najetí kurzoru myši na odkaz ihned uživatel uvidí, kam odkaz vede a může tak předejít návštěvě nebezpečného webu.

Uživatelé systému budou o tomto problému informováni a ujištěni, že nikdy od nich nebudou požadovány přihlašovací údaje. Tím pádem se dají označit veškeré pokusy o získání těchto údajů za podvodné.

6.4.5 SQL injection

Tento typ útoku spoléhá na chybu v kódu webových aplikací, které spolupracují s databázemi. Když uživatel provádí na webu nějaké dotazy, webový server je převede na SQL dotazy a posílá je databázovému serveru. Problém nastává, když vývojář neošetří tyto SQL dotazy a umožní tak útočníkovi daný dotaz modifikovat vyplněním vstupních polí specifickými texty. Např. na místě, kde by měl uživatel vyplnit přihlašovací jméno, a to se následné vsune do SQL dotazu, zadá zbývající část SQL dotazu, která se následně vsune do původního. Tím se vytvoří zcela nový dotaz a pošle databázovému serveru. Ten odpoví na podvržený dotaz a útočník tak může získat citlivé údaje. [20]

Zabránit takovému útoku může vývojář použitím tzv. parametrizovaných dotazů (PDO), kdy se nevkládají data ze vstupních polí do SQL dotazu pomocí proměnné, ale odesílají se data a předem připravený dotaz zvlášť. Databázový server pak data sám bezpečně zpracuje. Většina moderních frameworků (např. Laravel) ošetřuje tento problém automaticky.

6.4.6 Cross-site request forgery

Cross-site request forgery (CSRF) je takový způsob útoku na webové aplikace, kdy jsou PHP skripty informačního systému volány z webového serveru třetí strany. Chceme, aby tyto dotazy byly ignorovány a server odpověděl pouze na dotazy přicházející ze stránek systému. K tomu slouží tzv. CSRF token, který je unikátní a vygeneruje se každému uživateli při návštěvě webové stránky systému (např. po přihlášení) a uloží se do cookies. Stejný token si pak pamatuje webový server pro dané sezení s daným uživatelem. Tento token je poté odeslán spolu s dotazem uživatele, webový server jej zkontroluje a pokud se shoduje, odpoví. [20]

Vývojář by měl v každém PHP skriptu, který chce ošetřit proti CSRF, kontrolovat shodu tokenu. Moderní frameworky automaticky CSRF token kontrolují, vývojář tak musí token nastavit uživateli po přihlášení a pokud používá Ajax volání, je nutné nastavit tomuto požadavku token také. Veškerá tato opatření jsem ve svém systému provedl.

7 Testování

Další důležitou součástí vývojového cyklu informačního systému je jeho testování. Zjednodušeně řečeno, testování softwaru znamená odhalování chyb v programech.

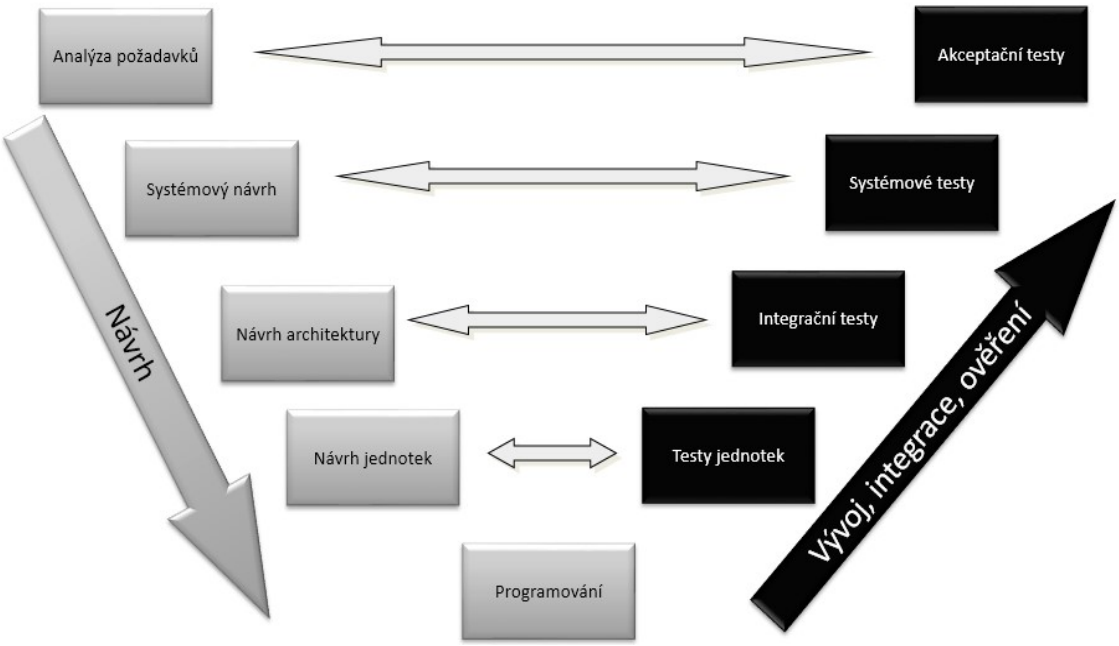
7.1 Rozdělení softwarového testování

Jelikož se v tomto oboru objevuje mnoho názvů i přístupů, často dochází k nedorozumění či nepochopení, co a kdy se vlastně testuje nebo jaký lze očekávat výsledek jednotlivých testování. Testování můžeme rozdělit podle několika kritérií.

7.1.1 Podle fáze testování

Testovat můžeme v různých krocích vývojového cyklu:

- Unit testy – nejmenší jednotka mezi testy. Píší je přímo programátoři a soustředí se na ověření funkčnosti kódu. Testuje se pouze jednotka samostatně (např. procedura, metoda, třída).
- Modul testy – tyto testy taktéž píší programátoři. Rozdíl mezi unit testy spočívá spíše v rozsahu – testuje se např. celá knihovna
- Integrační testy – jsou tvořeny testery po dokončení práce programátorů a zaměřují se na testování, zda jednotlivé moduly fungují dohromady.
- Funkční testy – funkční testování většinou zkoumá, co systém dělá, nikoliv nějakou funkci (metodu) třídy. Jsou to testy, které zkoumají, zda program správně plní všechny úkoly, pro které je vytvořen a zda odpovídá požadavkům zákazníka. Na tuto kategorii je kladen velký důraz, jelikož mají na bezporuchovost výsledného produktu významný vliv. Podle připravených scénářů se simulují různé kroky, které by mohly v reálné situaci nastat. Testování probíhá v několika kolech, pokud se objeví chyba, je opravena a funkčnost programu znovu otestována. Toto testování je nutností každého programu předtím, než se předá zákazníkovi.
- Akceptační testy – jedná se o testy na straně zákazníka. Ten si poté se svým týmem testerů provede vlastní testování. To je často provedeno pomoci předem připravených scénářů a probíhá v prostředí u zákazníka. Důležité je dohodnout si způsob oznamování nalezených chyb a nedostatků a jejich opravení a následné doručení a nasazení zpět u zákazníka v co nejkratší době. Pokud předešlé testy prošly bez komplikací, dalo by se očekávat, že i zákazník bude spokojen a jeho testy neobjeví žádné problémy. Opak je pravdou a zpravidla je zákazník rád, pokud jeho tým testerů objeví nějaké nedostatky. Spokojen přestává být ve chvíli, když je takových nedostatků mnoho nebo jsou kritické a ovlivňují funkčnost celého systému. Důležité je hlásit a opravovat chyby co nejrychleji, aby nedošlo k velkému zpoždění nasazení systému do provozu.



Obrázek 25: Fáze softwarového testování

7.1.2 Podle znalosti kódu

Tohle kritérium rozděluje testování pouze do 2 kategorií:

- White Box – testy jsou psány na základě znalosti kódu (např. unit testy)
- Black Box – tester nemá přístup ke kódu, testuje tak pouze, jak systém reaguje (např. funkční testy).

7.1.3 Podle způsobu realizace

Tady se testy dělí podle toho, jak jsou vykonávány:

- Manuální – testy provádí přímo tester podle testovacích případů. Dnes jsou takové testy na ústupu a nahrazují se testy automatizovanými.
- Automatizované – tyto testy jsou prováděny automaticky a zpravidla se opakují. Pomocí skriptovacích jazyků jsou vytvořeny programy testující scénáře zaměřené na různé prvky systému a ty jsou pak spouštěny.
- Exploratory testy – jedná se více méně o testy manuální, s tím rozdílem, že tester nemá připravené žádné testovací případy (scénáře), ale systém "objevuje". Cílem je prozkoumat, jak se daný program chová.

7.1.4 Podle dimenzí kvality

Dimenze kvality jsou definované normou ISO/IEC 25010:2011 a ukazují, že pohledů na kvalitu testovaného systému je více a že nestačí otestovat pouze jeho funkce.

- Funkčnost (Functionality) – správné chování funkcí systému podle definice funkčních specifikací (funcspec, FS).

- Výkon (Performance) – zda systém není pomalý a zvládne větší počet najednou pracujících uživatelů nebo naopak nezabírá příliš systémových zdrojů při obsluze maximálního počtu uživatelů
- Kompatibilita (Compatibility) – možnost kombinování více druhů softwaru nebo hardwaru
- Použitelnost (Usability) – zda je systém praktický a uživatelský přívětivý, zda se s ním dobře pracuje. I pokud se systém podle zákazníka "chová špatně", může se jednat o chování "správné", pak je chyba ve specifikaci
- Spolehlivost (Reliability) – zda se chová stejně za všech okolností, zvláště po přetížení, po výpadku či chybě, zda tyto stavy umí detekovat a hlásit.
- Bezpečnost (Security) – autentizace a autorizace, zabezpečení systému
- Udržitelnost – do jaké míry lze systém aktualizovat a testovat, nahrazovat jednotlivé komponenty
- Přenositelnost – nakolik je aplikace schopná fungovat na jiném HW, na jiných OS

Čerpáno z [23].

7.2 Vlastní testy informačního systému

Při vývoji informačního systému jsem se řídil správným vývojovým cyklem softwaru, tudíž jsem si po dokončení implementace vytvořil automatizované funkční testy, abych ověřil hlavní funkcionalitu systému. K tomu jsem využil program Behave [24], který umožňuje zapsat testovací scénáře simulující aktivitu uživatele v uživatelsky přívětivém jazyce a pozadí testů ve skriptovacích jazyce Python. Jelikož informační systém běží na webu, zvolil jsem nástroj Selenium, který simuluje veškerou interakci uživatele s webovou stránkou. Jelikož zákazník požadoval, aby byl systém optimalizován hlavně pro webový prohlížeč Google Chrome, nastavil jsem, aby Selenium využíval ovladač *chromedriver*.

7.2.1 Systémové požadavky

Testování provádím na OS Ubuntu Server. Pro úspěšné spuštění testů je nutné mít nainstalované následující programy:

- Behave 1.2.6
- Python 3.6.7
- Selenium WebDriver for Python 3.14.0
- ChromeDriver 74.0

Poté stačí v terminálu přejít do kořenové složky s testy a spustit testování příkazem *behave*.

7.2.2 Funkční testy

Testy jsou zaměřený na základní funkcionalitu systému. Testy obsahují scénáře (scenarios) simulující přihlášení (jak nesprávné přihlašovací údaje, tak správné), tvorbu nové zakázky, změnu stavu již vytvořené zakázky, úpravu již vytvořené zakázky, správnou funkčnost statistik, nahrávání souborů, mazání souborů, úpravu uživatelské poznámky a odhlášení ze systému.

Všechny testy jsou přizpůsobeny čerstvě nainstalovanému systému s jedním vytvořeným uživatelským účtem s přihlašovacím e-mailem *test@test.test* a heslem *123456*. Před opětovným

spuštěním testů je nutné vyprázdnit databázi a zachovat pouze tento uživatelský účet. Pro ulehčení jsem vytvořil skript, který tohle provede a následně spustí testování. Skript se nachází ve složce Laravelu určené pro testy, tudíž */tests/runTests.sh* a spustí se příkazem "*bash runTests.sh*". Výsledek testů uvidí uživatel v terminálu (viz Obrázek 26).

Jedinou věc, kterou testy odhalily, a to náhodné selhání některých testů, bych neoznačil ani tak za problém, jako spíš za názornou ukázku funkčnosti zabezpečení systému, kde se zablokuje příchozí požadavek, jelikož se jich odesílá velké množství ve velmi krátkém čase a mohou být považovány za DoS útok. Opětovné spuštění testů většinou pak proběhlo úspěšně. Testování na lokálním stroji bylo úspěšné vždy.

```
Feature: Prihlaseni # test.feature:1

  Scenario Outline: Prihlaseni -- @1.1 Uzivatele                    # test.feature:10
    Given Uzivatel je na prihlasovaci strance                       # steps/features.py:5 0.224s
    When Uzivatel se pokusi prihlasit s failtest@podajex.com 123456 # steps/features.py:10 0.480s
    Then uzivatel byl neprihlasen                                   # steps/features.py:15 0.021s

  Scenario Outline: Prihlaseni -- @1.2 Uzivatele                   # test.feature:11
    Given Uzivatel je na prihlasovaci strance                      # steps/features.py:5 0.087s
    When Uzivatel se pokusi prihlasit s test@test.test spatneheslo # steps/features.py:10 0.538s
    Then uzivatel byl neprihlasen                                  # steps/features.py:15 0.015s

  Scenario Outline: Prihlaseni -- @1.3 Uzivatele              # test.feature:12
    Given Uzivatel je na prihlasovaci strance                 # steps/features.py:5 0.095s
    When Uzivatel se pokusi prihlasit s test@test.test 123456 # steps/features.py:10 0.686s
    Then uzivatel byl prihlasen                               # steps/features.py:15 0.032s

  Scenario: Tvorba nove zakazky                    # test.feature:14
    When Uzivatel se pokusi vytvorit novou zakazku # steps/features.py:23 0.769s
    Then Nova zakazka je vytvorena                 # steps/features.py:28 0.071s

  Scenario: Zmena stavu zakazky                 # test.feature:18
    When Uzivatel se pokusi zmenit stav zakazky # steps/features.py:33 0.375s
    Then Zakazka zmeni stav                     # steps/features.py:38 0.346s

  Scenario: Uprava zakazky                            # test.feature:22
    When Uzivatel se pokusi upravit cenu zbozi zakazky # steps/features.py:43 0.522s
    Then Cena zbozi se zmeni                           # steps/features.py:48 0.243s

  Scenario: Statistiky               # test.feature:26
    When Uzivatel zobrazi statistiky # steps/features.py:53 0.186s
    Then Celkovy pocet zakazek je 1  # steps/features.py:58 0.345s

  Scenario: Nahrani souboru               # test.feature:30
    When Uzivatel se pokusi nahrat soubor # steps/features.py:63 0.707s
    Then Soubor byl nahran                # steps/features.py:68 0.064s

  Scenario: Smazani souboru                  # test.feature:34
    When Uzivatel se pokusi odstranit soubor # steps/features.py:73 0.493s
    Then Soubor by odstranen                 # steps/features.py:78 0.052s

  Scenario: Uprava uzivatelske poznamky      # test.feature:38
    When Uzivatel se pokusi upravit poznamku # steps/features.py:83 0.377s
    Then Poznamka byla zmenena               # steps/features.py:88 0.288s

  Scenario: Odhlaseni                # test.feature:42
    When Uzivatel se pokusi odhlasit # steps/features.py:93 0.390s
    Then Uzivatel je odhlasen        # steps/features.py:98 0.024s

1 feature passed, 0 failed, 0 skipped
11 scenarios passed, 0 failed, 0 skipped
25 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m7.432s
```

Obrázek 26: Výsledek testů v terminálu

8 Možná rozšíření

V této kapitole rozeberu možná rozšíření informačního systému, která by se mohla v budoucnu implementovat. Tyto nápady nebyly firmou požadovány a napadly mě až po čas implementace.

První věc, která mě napadla bylo dvoufázové ověření uživatele při přihlašování. Po zadání správného hesla by systém poslal uživateli SMS zprávu s kódem, který by musel uživatel zadat, aby mohl pokračovat. Bylo by nutné přidat v databázi v tabulce *users* sloupec pro telefonní číslo. Zkoumal jsem několik SMS bran, které poskytují HTTP API neboli propojení s webovými službami, avšak za každou odeslanou SMS by musel provozovatel systému platit.

Další nápad souvisí se zabezpečením. Pokud by systém zaznamenal narůstající počet pokusů o přihlášení a útočník by nebyl zablokován automaticky (obešel by veškerou ochranu), dostal by správce systému varovnou SMS zprávu a email, aby mohl včas zasáhnout a útočníkovi přístup odepřít.

Poslední rozšíření související s SMS je možnost aktivace zasílání přehledu upozornění (těch z hlavní stránky systému) pomocí SMS všem uživatelům, kteří si tuto službu aktivují. Pokud uživatel ví, že nebude mít delší dobu přístup k systému (např. kvůli vycestování mimo EU), mohl by si předem zapnout zasílání upozornění a každé ráno by obdržel SMS se všemi aktivními upozorněními pro daný den.

Další rozšíření, a to stránkování při velkém počtu zakázek v systému, je spíše otázkou přehlednosti. Záleží na uživatelích systému, zdali jim postupem času bude vadit velké množství zakázek zobrazených v přehledu na hlavní stránce systému nebo by raději uvítali stránkování, které rozdělí zakázky na jednotlivé stránky po např. 30 zakázkách, čímž ale zpomalí pohyb v systému. To stejné lze aplikovat na upozornění. V praxi po nasazení systému mohou po nějaké době uživatelé zjistit, že mají hlavní stránku zahlcenou upozorněními. K tomuto problému může dojít, pokud mnoho zakázek zůstane ve stavu “Nabídnuto” nebo “Odesláno”. Řešením by bylo shlukování různých upozornění týkající se stejné firmy do jednoho většího ukazujícího počet shluknutých upozornění.

9 Závěr

V rámci této bakalářské práce jsem vytvořil funkční informační systém pro firmu zabývající se prodejem podavačů tyčí splňující všechny požadavky firmy a také všechny body mého zadání. Systém jsem nasadil na webový cloud, ke kterému později udělím firmě přístup a uvedu systém do produkčního stavu, aby jej mohla začít využívat. Během předání také vytvořím požadované uživatelské účty.

Akceptační testy byly úspěšné, klient přišel jen s malým rozšířením, které požadoval doimplementovat. Systém lze plně využívat na všech zařízeních firmy, na kterých ho v budoucnu budou používat. S použitím jednotlivých funkcí systému problém nenastal. Praktičnost systému se klientovi zamlouvá a těší se na předání.

Systém jsem navíc také udělal responzivní, pro případ nutné návštěvy uživatelem z mobilního zařízení nebo pro případ modernizace firemního zařízení. Otestoval jsem jej na všech běžných webových prohlížečích a docílil na nich požadovaného vzhledu a funkčnosti.

Tato práce mě velmi obohatila ve znalostech informačních systémů a jejich zabezpečení. Nikdy předtím jsem nepracoval s frameworkem a jsem vděčný, že díky téhle zkušenosti jsem se naučil používat jeden z nejpopulárnějších frameworků Laravel. Tuto znalost určitě použiji i v budoucnu a budu se věnovat vývoji dalších informačních systému nebo jiných webových aplikací a zajisté mi pomůže v následujícím studiu. Také jsem si vyzkoušel následovat přesný vývojový cyklus softwaru. Tuto znalost nepochybně využiji při budoucí práci ve vývojářském týmu.

Literatura

[1] Burget, R.: Informační systémy. Pojem informačního systému, data, informace. Elektronická prezentace [online]. [cit. 2019-03-10]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIIS-IT%2Flectures%2Fp01_Informacni_systemy.pdf&cid=12157

[2] Čápka, D.: Úvod do UML. IT Network [online]. [cit. 2019-03-10]. Dostupné z: https://www.itnetwork.cz

[3] Zendulka, J., Rudolfová, I.: Databázové systémy. IDS. Studijní opora. FIT VUT v Brně. 2006, 217 s.

[4] PHP - Wikipedie [online]. [cit. 2019-03-14]. Dostupné z: https://cs.wikipedia.org/wiki/PHP

[5] Historical trends in the usage of server-side programming languages for websites, May 2019 [online]. Q.Success: c2009-2019 [cit. 2019-03-14]. Dostupné z: https://w3techs.com/technologies/history_overview/programming_language

[6] PHP: Hypertext Preprocessor [online]. [cit. 2018-11-23]. Dostupné z: https://www.php.net/

[7] JavaScript – Wikipedie [online]. [cit. 2019-03-14]. Dostupné z: https://cs.wikipedia.org/wiki/JavaScript

[8] JQuery [online]. [cit. 2018-12-28]. Dostupné z: https://jquery.com/

[9] Typeahead.js [online]. [cit. 2018-12-28]. Dostupné z: https://twitter.github.io/typeahead.js/examples/

[10] Tag / Token Input [online]. [cit. 2018-12-28]. Dostupné z: https://github.com/donmccurdy/input-tokenizer

[11] Excel-like Bootstrap Table Sorting And Filtering Plugin [online]. [cit. 2019-01-07]. Dostupné z: https://www.jqueryscript.net/table/Excel-like-Bootstrap-Table-Sorting-Filtering-Plugin.html

[12] Chart.js [online]. [cit. 2019-01-25]. Dostupné z: https://www.chartjs.org/

[13] Laravel – The PHP Framework For Web Artisans [online]. [cit. 2018-12-20]. Dostupné z: https://laravel.com/

[14] Directory Structure – Laravel - The PHP Framework For Web Artisans [online]. [cit. 2019-03-14]. Dostupné z: https://laravel.com/docs/5.8/structure

[15] Hypertext Markup Language – Wikipedie [online]. [cit. 2019-03-14]. Dostupné z: https://cs.wikipedia.org/wiki/Hypertext_Markup_Language

[16] CSS – Kaskádové styly [online]. [cit. 2019-03-14]. Dostupné z: https://www.jakpsatweb.cz/css/

[17] Nginx vs Apache – KeyCDN Support [online]. Proinity LLC: c2019, aktualizováno 2018-11-04 [cit. 2019-03-14]. Dostupné z: https://www.keycdn.com/support/nginx-vs-apache

[18] Milkovich, D.: 13 Alarming Cyber Security Facts and Stats. Cybint [online]. Cybrint: c2019, aktualizováno 2018-12-03 [cit. 2019-04-05]. Dostupné z: https://www.cybintsolutions.com/cyber-security-facts-stats/

[19] Metody zabezpečení informačních systémů | ISVS.CZ [online]. ADVICE.CZ s.r.o.: c2001-2015, aktualizováno 2007-04-11 [cit. 2019-04-10]. Dostupné z: http://2011-2015.isvs.cz/metody-zabezpeceni-informacnich-systemu/

[20] Melnick, J.: Top 10 Most Common Types of Cyber Attacks [online]. Netwrix Corporation: c2019, aktualizováno 2018-05-15 [cit. 2019-04-10]. Dostupné z: https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/

[21] How Secure Is My Password? [online]. Small Hadron Collider: c2019 [cit. 2019-04-10]. Dostupné z: https://howsecureismypassword.net/

[22] Maunder, M.: The March 2017 WordPress Attack Report [online]. DEFIANT INC: c2019, aktualizováno 2017-04-06 [cit. 2019-04-10]. Dostupné z: https://www.wordfence.com/blog/2017/04/march-2017-wordpress-attack-report/

[23] Kitner, R.: Typy testování software [online]. Radek Kitner: c2015 [cit. 2019-04-12]. Dostupné z: https://kitner.cz/testovani_softwaru/typy-testovani-software-trideni-testu/

[24] Welcome to behave! - behave 1.2.7.dev1 documentation [online]. [cit. 2019-03-28]. Dostupné z: https://behave.readthedocs.io/en/latest/

Seznam příloh

Příloha 1. CD/DVD se zdrojovými texty. Obsah:

- xkalet05.pdf – elektronická verze písemné zprávy
- xkalet05.docx – zdrojový tvar písemné zprávy
- zdrojove-kody.zip – komprimované zdrojové kódy aplikace
- manual.pdf – stručný návod