

Autor **Adam Petříček**

Obor **Informační technologie**

Vedoucí práce **Mgr. Michal Stehlík**

Školní rok **2020/2021**

Střední průmyslová škola strojní   
a elektrotechnická a Vyšší odborná škola, Liberec 1, Masarykova 3

APLIKACE SOUČÁSTKOVÁ ZÁKLADNA

Maturitní práce

Anotace (Resumé)

Práce se zabývá ...

Vychází z ...

Přináší ..., atp.

Summary

This work ...

Čestné prohlášení

Prohlašuji, že jsem předkládanou maturitní/ročníkovou práci vypracoval(a) sám(a) a uvedl jsem veškerou použitou literaturu a bibliografické citace.

V Liberci dne

Adam Petříček

Obsah

[Úvod 1](#_Toc459976514)

[1 První kapitola 1](#_Toc459976515)

[1.1 První podkapitola 1](#_Toc459976516)

[1.1.1 První podpodkapitola 1](#_Toc459976517)

[1.2 Druhá podkapitola 1](#_Toc459976518)

[1.2.1 Další podpodkapitola 1](#_Toc459976519)

[1.2.2 Ještě další podpodkapitola 1](#_Toc459976520)

[2 Kapitola 1](#_Toc459976521)

[2.1 Podkapitola 1](#_Toc459976522)

[Závěr 1](#_Toc459976523)

[Seznam obrázků 1](#_Toc459976524)

[Použitá literatura 1](#_Toc459976525)

[A. Seznam přiložených souborů 1](#_Toc459976526)

[B. Další příloha 1](#_Toc459976527)

Úvod

Aplikace Součástková základna byla vytvořena podle zadání firmy Jablotron. Její vývoj začal v květnu 2019 (v rámci povinné praxe studentů ve firmách) a pokračuje až do teď. Aplikace se používá pro správu základny pro elektronické součástky na dvou odděleních ve firmě. Díky všem funkcím aplikace poskytuje kontrolu nad součástkami v elektronické podobě.

Použití šablony

Nastavte název dokumentu a autora v nabídce Soubor/Vlastnosti.

Pro vkládání zdrojů použijte Reference/Spravovat prameny.

1. Návrh systému

Po obecném zadání aplikace od vedoucího daného oddělení a upřesnění technických možností od IT oddělení bylo třeba určit strukturu aplikace a použité technologie. Částečnou inspiraci jsem čerpal z předchozí verze systému, který se ve firmě používal do té doby.

* 1. Backend technologie

Systém, co se ve firmě používal předtím byl napsán v čistém PHP. Pro moji aplikaci používám PHP framework Laravel. Tento framework je použit z důvodu normalizace technologií pro všechny aplikace ve firmě, nevybíral jsem ho. Oproti čistému PHP přináší ulehčení v mnoha ohledech – např. zabezpečení aplikace, práce s databází, přehlednost kódu, ovšem za cenu menší kontroly nad kódem.

* 1. Frontend technologie

Za účelem zjednodušení designu obsahuje aplikace knihovnu Bootstrap, která umožňuje používat předem vytvořené styly. Pro použití ikon se v aplikací nachází FontAwesome. Co se funkční stránky týče, pro zobrazování tabulek z databáze uživateli používám JavaScript knihovnu DataTables. V neposlední řadě je pro jednodušší JavaScript kód použita knihovna jQuery.

* 1. Obecná struktura aplikace

Aplikace je rozdělena na dvě základní části – **„Prototypová dílna vývoje“** a **„Servis“**. Toto rozdělení je z důvodu, že aplikace běží na dvou odděleních ve firmě, kde každá má jiný způsob uložení součástek. Mezi nimi lze přepínat přes switch v horní části aplikace. Každá část má své vlastní administrátory, kteří systém spravují.

* 1. Konkrétní struktura aplikace

Nejdůležitější stránky aplikace jsou **„Seznam součástek“** a **„Grafické zobrazení“**, které se používají pro spravování součástek. Dále obsahuje sekci **„Můj seznam“**, kde si lze uložit vlastní seznam součástek, pro případné nejasnosti uživatelů **„Manuál“** a pro nastavování aplikace sekci **„Administrace“**.

* + 1. Seznam součástek

Pokud uživatel zná název součástky a potřebuje zjistit její pozici na stěně, použije tuto část aplikace. Hlavní částí stránky je vlastní full-text vyhledávací systém. Lze vyhledávat podle více parametrů anebo hledat jen v určité kategorii součástek. Výsledek vyhledávání je poté zformátován do tabulky se všemi dostupnými informacemi o součástce.

* + 1. Grafické zobrazení

Naopak, v případě, že uživatel zná pozici součástky a potřebuje nalézt její jméno, popř. ji upravit, použije tuto část aplikace. Grafické zobrazení slouží jako elektronická vizualizace stěny, tudíž by měla vypadat stejně, jako když se uživatel na stěnu kouká fyzicky. Po kliknutí na konkrétní skříňku se zobrazí detail, kde jsou vidět již jednotlivé šuplíky.

Tři základní způsoby skladování součástek jsou v grafickém zobrazení odděleny barevně. Červená barva značí klasické součástky na stěně, uložené v šuplících. Modrá barva značí stejné skříňky jako jsou na stěně, ovšem na posuvných stojanech. Zelená barva značí uložení součástek v přepravkách a na kotoučích ve skladové místnosti.

* + 1. Můj seznam

Myšlenka za touto částí aplikace je taková, že si uživatel ve své kanceláři do seznamu přidá součástky, které potřebuje, poté přijde k součástkové základně a na tabletu vedle si zobrazí svůj seznam. Podle něj si poté může součástky pohodlně najít a nemusí si je pokaždé psát na kus papíru.

* + 1. Administrace

Do administrace mají přístup pouze určení uživatelé, kteří se o systém starají. Lze zde nastavovat některé parametry aplikace, přidávat nové přepravky do skladu, testovat aplikaci jako uživatel bez administračních práv, měnit práva jiných uživatelů, a především se starat o nahlašovací systém šuplíků.

1. Požadavky na systém

Požadavky byly zadávány ze dvou stran. Tou první byl šéf oddělení Prototypová dílná vývoje, který zadával jednotlivé funkce aplikace, které očekávají. Úkolem bylo vytvořit databázový systém, ve kterém půjdou evidovat součástky, nahlašovat je v případě vyprázdnění šuplíku a vyhledávat v nich. Pokud to bude možné, grafické vyobrazení celé aplikace bude jen bonusem. Celkové zadání jsem si mohl dost upravit podle svých potřeb, stačilo to jen prokonzultovat se zadavatelem. Postupem času se přidaly další funkce, které se používáním aplikace ukázaly jako potřebné. Druhou zadávající stranou bylo IT oddělení firmy, které zadávalo aplikaci z funkční stránky – v jakém jazyce má být naprogramována, jaké technologie mohu použít a jak projekt nahrát na live server.

* 1. Autentifikační modul a uživatelské API

K přihlašování uživatelů, kteří aplikaci používají jsem využil autentifikační modul již vytvořený a používaný od Jablotronu. Jde o custom Middleware, který detekuje přihlášení do interního aplikačního portálu podle cookies. V případě, že je uživatel přihlášen, tak jeho údaje uloží do session, aby byly použitelné v celé aplikaci. V opačném případě přesměruje uživatele na přihlašovací stránku mimo mou aplikaci.

Další mnou nevytvořená věc použitá v aplikaci je API pro zjištění informací o uživatelích. Podle unikátního čísla uživatele o něm mohu zjistit informace jako jeho jméno, mail, nadřízené aj.

* 1. Git

Celý projekt je od začátku vývoje uložen na soukromém GitLabu pro Jablotron. Díky tomu lze dohledat historii commitů a jednotlivé změny v kódu. Počet commitů do repozitáře se pohybuje kolem stovky, v průběhu vývoje byl taktéž vytvořen branch „dev“ pro testování nových funkcí.

* 1. Propojení s tabletem

Po dokončení základního vývoje a vytvoření funkční aplikace přišel návrh na propojení s tabletem. Tablet je fyzicky umístěn na uzamykatelném stojanu vedle součástkové základny a aplikace na něm běží 24/7. Díky tomu si kdokoliv, kdo si přijde pro součástku, může najít její pozici v součástkové základně rovnou na místě a nemusí používat svůj počítač. Další funkce tabletu je sekce „Můj seznam“ v aplikaci, kde si uživatel může sestavit seznam různých součástek a na tabletu si ho poté jen zobrazit a podle souřadnice součástky nalézt. Tablet má nastavenou statickou IP adresu a v aplikaci je pro něj výjimka v Middleware, která vytváří pro tablet „stinný účet“, pro zajištění správné funkčnosti v ostatních částech aplikace.

1. Framework Laravel

Díky frameworku Laravel bylo vytvoření celé práce jednodušší a jeho funkce ušetřily spoustu času. Jde o lepší řešení i do budoucna, protože je kód přehlednější – lépe se v něm tedy vyzná i někdo jiný, kdo by ho chtěl v budoucnu upravovat. Laravel je MVC framework, to znamená že používá architekturu Model-View-Controller. Díky takovému rozdělení dokáže aplikace rozlišit jednotlivé logické části.

* 1. Controller

Stará se o komunikaci uživatele s aplikací. Přichází do něj požadavky od routeru na jednotlivé funkce. Tyto funkce zpracují uživatelův požadavek a předají ho do View, který vrací uživateli jako výsledek. V aplikaci mám logiku, že každý View má svůj Controller s příslušnými metodami pro zpracování dat.

* 1. View

Jde o blade šablonu s HTML kódem. Může dostat od Controlleru nějaká data jako parametry, ty poté zobrazí. Díky blade šabloně lze používat v kódu PHP tagy ve zjednodušené formě, která je přehlednější společně s HTML kódem. Blade kód se poté převádí na čisté PHP.

* 1. Model

Fungují zpravidla jako zprostředkovatelé připojení k databázi. Logika modelu je taková, že jeden model představuje jednu databázovou tabulku a instance tohoto modelu je jeden záznam v této tabulce. Když tedy chceme vytvořit nový záznam v databázi, vytvoříme novou instanci modelu a použijeme na něj metodu pro uložení. Podobným způsobem se dají i záznamy upravovat, či mazat.

* 1. Routování

O to, aby uživatel dostal data, o jaké si žádá se stará soubor „routes/web.php“. V něm jsou vypsané jednotlivé URL adresy aplikace a každá má přiřazenou metodu v Controlleru, která se volá při zadání dané aplikace. Jsou zde také rozlišené GET a POST metody, podle způsobu předání dat.

* 1. Příkazy

Celý Laravel se spouští a ovládá přes příkazy v příkazové řádce. Pro zjednodušení používání umožňuje framework i vytvoření vlastních příkazů. V aplikaci jsem vytvořil celkem 3 nové příkazy.

* php artisan generate:tags
* php artisan fill:units
* php artisan fill:dimensions

Tyto příkazy lze volat přímo z konzole a Laravel poskytuje příkaz na vygenerování šablony pro vytvoření vlastního příkazu.

1. Databáze

Pro ukládání dat využívám MySQL, na jejich správu poté phpMyAdmin na lokálním serveru a MySQL Workbench pro připojování k produkčnímu serveru. Tento systém jsem zvolil především z důvodu jeho velké rozšířenosti a jednoduchosti.

* 1. Obecná struktura databáze

V aplikaci jsou použité dvě databáze. Jedna z toho je interní firemní databáze se skladovými kartami součástek, kterou používám pouze na čtení (pro zjednodušení vyhledávání). Druhá databáze obsahuje data aplikace, ta byla vytvořena mnou a aplikace do ní zapisuje i z ní čte. Obsahuje celkem 9 tabulek. 3 z toho jsou společné pro celou aplikaci (**users, saved\_units, storage\_units**), 3 jsou jen pro část aplikace „Servis“ (**servis\_dimensions, servis\_reported\_units, servis\_units**) a 3 jsou jen pro část aplikace „Prototypová dílna vývoje“ (**dimensions, reported\_units, units**).

* 1. Konkrétní struktura tabulek

Všechny tabulky, ve kterých se může vyskytnout více řádků se stejnými daty, obsahují pro jednodušší vybírání dat sloupec s primárním unikátním klíčem, který se jmenuje „id“. Tento sloupec se nenastavuje ručně, ale má vlastnost „AUTO\_INCREMENT“ – o jeho nastavování se tedy stará samo SQL, a to tím způsobem, že ho zvýší o 1 při každé nově vytvořené položce. Díky tomu bude pokaždé vytvořeno nové unikátní číslo pro identifikaci řádků. Jednotlivé sloupce mají poté nastaven konkrétní datový typ, podle typu jeho obsahu, popř. i výchozí hodnotu a informaci, zda může být sloupec nulový.

Výjimkou jsou tabulky **dimensions** a **servis\_dimensions**, které obsahují pouze jeden řádek. Tyto tabulky slouží pro uložení rozměrů stěny (počet sloupců a řádků), díky tomu lze aplikaci použít časem i na jinou stěnu, popř stěnu rozšiřovat.

* 1. Migrace

Pro bezproblémové nasazení aplikace jsou ve složce „database/migrations/“ umístěny soubory s migracemi databáze. V nich jsou detailně popsány všechny tabulky, co se mají v databázi vytvořit, jejich datové typy, defaultní hodnoty a další atributy. Díky tomu lze migrovat pouze určité tabulky. Práce s migracemi taky přidává možnost přehledně migrace vracet (rollbackovat) – díky tomu nemusíme při chybě v jedné tabulce mazat celou databázi, ale stačí pouze poslední migrace.

Závěr

Tak jsem to dokázal! A teď ještě, co jsem mohl udělat jinak a jak by se dalo v práci pokračovat.

Seznam obrázků

**Nenalezena položka seznamu obrázků.**

Použitá literatura

1. **učitelé SPŠSE.** Úvod. *SPŠSE a VOŠ Liberec.* [Online] 01. 09 2016. [Citace: 01. 09 2016.] https://www.pslib.cz.

1. Seznam přiložených souborů

Text

1. Další příloha