STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Aplikace součástková základna

Adam Petříček
Liberecký kraj

Liberec 2021

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Aplikace součástková základna

Application parts base

Autoři: Adam Petříček
Škola: Masarykova 3, 460 84 Liberec 1
Kraj: Liberecký kraj
Konzultant: Mgr. Michal Stehlík

Liberec 2021

Anotace (Resumé)

Práce se zabývá vytvořením webového systému v PHP frameworku Laravel pro správu součástkové základny ve firmě Jablotron. Popisuje prostředky a způsoby použité k vytvoření aplikace. Součástí práce je také detailní popis jejich částí a detail jejího vývoje.

Summary

The work deals with the creation of a web system in the PHP framework Laravel for the management of the parts base in the company Jablotron. It describes the resources and methods used to create the application. Part of the work is also a detailed description of their sections and a detail of its development.

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Liberci dne 22.3.2021 ……
Adam Petříček

Obsah

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.

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. Částečná inspirace byla čerpána z předchozí verze systému, který se ve firmě používal do té doby.

1.1 Backend technologie

Systém, který se ve firmě používal dříve, byl napsán v čistém PHP. Pro moji aplikaci používám open-source PHP framework Laravel. Tento framework je použit z důvodu normalizace technologií pro všechny aplikace ve firmě, nevybíral jsem ho já. Oproti čistému PHP přináší ulehčení v mnoha ohledech – např. zabezpečení aplikace, práce s databází, přehlednost kódu, ovšem za cenu menší kontroly nad kódem. Při začátku vývoje aplikace šlo o můj první kontakt s tímto frameworkem – učil jsem se tedy všechny potřebné věci sám, především z dokumentace Laravelu.

1.2 Frontend technologie

Za účelem zjednodušení designu obsahuje aplikace CSS a JS knihovnu Bootstrap. Tato knihovna umožňuje používání předem vytvořených stylů pro různé prvky webu, jako jsou například tlačítka nebo formuláře. Mimo jiné Bootstrap mění také některé defaultní nastavení webu, jako je např. font. V neposlední řadě zjednodušuje práci s marginem a paddingem.

Pro zjednodušení uživatelské interakce s tlačítky používám knihovnu FontAwesome. Ta přidává kolekci několika stovek ikon. Tyto ikony se na web implementují v tagu `<i>`. V aplikaci jsou ikony použité skoro ve všech tlačítkách místo textu, protože poskytují intuitivní cestu, jak uživateli říct, co tlačítko dělá.

V aplikaci se nachází několik tabulek, do kterých se vybírají data z databáze. Pro zobrazování těchto dat uživateli jsem se rozhodl použít JavaScript knihovnu DataTables. Ta umožňuje přehledné zobrazování dat, které ji aplikace předá. Díky této knihovně lze zobrazovat pouze určité množství ze všech položek v SQL dotazu a poté lze mezi nimi listovat přes pagination, popř. vyhledávat.

Pro jednodušší práci s JavaScriptem bylo zvoleno použití knihovny jQuery. Ta zlepšuje čitelnost kódu a celkovou interakci JavaScriptu s ostatními částmi webu.

1.3 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.

Sekce „**Prototypová dílna vývoje**“ má stěnu, stojan a sklad. První dvě části fungují na stejném principu – jde o součástky ve skříňkách, kde má každá svou vlastní souřadnici. Naopak sklad má součástky v přepravkách a jediná informace, kterou součástka má, je číslo přepravky.

Sekce „**Servis**“ se liší tím, že mají pouze stojany. Z tohoto důvodu je v této části aplikace změněn systém souřadnic, aby seděl na strukturu bez stěny.

1.4 Konkrétní struktura aplikace

Nejdůležitější stránky aplikace jsou **„Seznam součástek“** a **„Grafické zobrazení“**, které se používají pro spravování součástek, vyhledávání v nich a zobrazování grafické interpretace součástkové stěny. Dále aplikace obsahuje sekci **„Můj seznam“**, kde si lze uložit vlastní seznam součástek a dále s ním pracovat. Pro případné nejasnosti uživatelů slouží sekce **„Manuál“** a pro nastavování a celkovou správu aplikace sekce **„Administrace“**.

1.5 Vyhledávací systém

U vyhledávacího systému bylo třeba učinit rozhodnutí, zda bude vyhledávání probíhat na straně serveru nebo klienta. Hlavní rozdíl je ten, že pokud vyhledávání probíhá na straně klienta, do front-endu aplikace se posílají všechna data (typicky SQL dotaz, který vybírá vše z jedné tabulky). Poté se všechna data načtou do front-endu (v mém případě přes JavaScriptovou knihovnu DataTables), front-end aplikace poté reguluje zobrazení pouze určitého množství dat a zajišťuje vyhledávání v nich. Výsledný HTML kód tedy může mít až statisíce řádků, podle počtu položek v databázi. Výhodou tohoto způsobu je rychlost vyhledávání a přepínání stránek, protože data už jsou načtena, je to tedy téměř instantní. Nevýhodou je pomalé první načítání – odvíjí se od počtu položek v databázi a typicky trvá několik vteřin.

Druhým způsobem je tzv. server-side search, který je použit v aplikaci. Vyhledávání probíhá na straně serveru, posílá se tedy nový SQL dotaz po každé změně vyhledávacího

pole, přepnutí navigace na jinou stránku, nebo změny filtrování. V tomto SQL dotazu jsou poté vrstveny klauzule WHERE, aby se vyfiltrovaly pouze položky podle parametru vyhledávání a použity klauzule LIMIT, a OFFSET pro vybrání pouze určitého počtu prvků. Výhodou je to, že uživatel nemusí čekat na první načítání – protože mu vždycky přijde pouze malé množství dat, které se načte rychle. Nevýhodou je časté posílání SQL dotazů a složitější kód.

2 Požadavky na systém

Požadavky byly zadávány ze dvou stran. Tou první byl vedoucí oddělení Prototypová dílná vývoje, ten zadával jednotlivé funkce aplikace, které jsou očekávány. 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 mohlo být dost upraveno od původní verze, 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 produkční server.

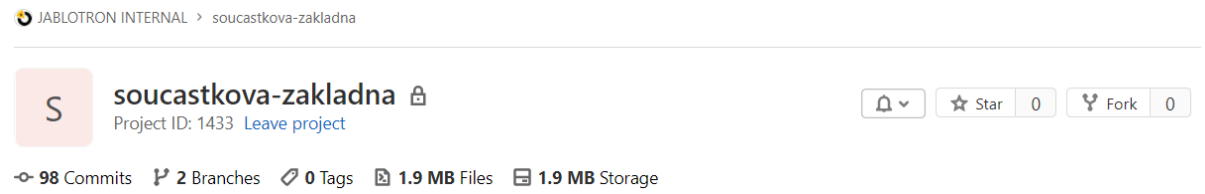
2.1 Autentifikační modul a uživatelské API

K přihlašování uživatelů jsem využil autentifikační modul již vytvořený a používaný od Jablotronu. Jde o vlastní 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.

2.2 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. Na GitLab jsem nahrával jednotlivé commity přes SSH, pro které jsem si ve složce Windows profilu vygeneroval klíč. Tento „SSH klíč“ jsem poté uložil do svého profilu na GitLabu. Díky tomuto postupu jsem mohl nahrávat pokaždé bez nutnosti zadávat přihlašovací údaje na GitLab. Druhý způsob nahrávání je přes protokol HTTP, u něj se ovšem musí pokaždé zadávat přihlašovací údaje.



Obrázek 1 - Popis projektu na Gitu

2.3 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, 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 využívající data o přihlášeném uživateli v ostatních částech aplikace.

Obrázek 2 - Detail tabletu

2.4 Použití JavaScriptu a CSS

V aplikaci se nachází několik JavaScript a CSS souborů, které jsou vytvořeny od firmy a používám je pro normalizaci zobrazení ve všech firemních aplikacích. Příkladem je soubor „public/css/coma.css“, který upravuje zobrazení některých formulářových prvků, a především knihovny DataTables, u které mění zobrazení navigace. Jednotlivé soubory jsou rozděleny do složek podle typu (JavaScript / CSS).

2.5 Souřadnice šuplíků

Způsoby označování šuplíku se dělí podle sekce aplikace. V sekci **„Prototypová dílna vývoje“** má každý šuplík, co se na stěně nachází svou unikátní souřadnici. Tato souřadnice se skládá ze dvou písmen a ze dvou čísel. První dvě písmena značí souřadnici skříňky (první písmeno je sloupec, druhé je řádek). Poté čísla značí souřadnici konkrétního šuplíku v určité skříňce (opět je první písmeno sloupec, druhé řádek. Souřadnice tedy může mít formát např. „**A-B-3-1**“. Tento šuplík se nachází ve skřínce ve sloupci A, řádku B, konkrétní šuplík je poté ve sloupci 3 a řádku 1.

Sekce aplikace „**Servis**“ nemá stěnu, ale pouze stojany. Zvolili si tedy alternativní způsob značení. Písmeno u nich značí jednotlivé stojany a číslo konkrétní skříňky ve stojanu. Způsob značení šuplíků zůstává je podobný jako v předchozím případě, ovšem místo dvou čísel používají písmeno a číslo. Příkladem je tedy např. souřadnice „**A-4-C-1**“. V tomto případě jde o čtvrtý šuplík ve stojanu A, ve kterém se nachází šuplík ve sloupci C a řádku 1.

Obrázek 3 - Detail stojanu

2.6 Nahlašování šuplíků

Jednou za čas se stane, že součástky v daném šuplíku dojdou. Pro tento případ bylo potřeba navrhnout systém, který se o toto dokáže starat. Kdokoliv tedy může nahlásit prázdný šuplík, což pošle informační mail (příjemce lze nastavit v souboru „.env“), ve kterém je informace o pozici šuplíku. ID šuplíku se uloží do databáze a nahlášené šuplíky se poté spravují v administraci webu. Zde se šuplík nejdříve označí jako objednaný. Až objednávka dorazí, a součástka se dá do šuplíku, šuplík se označí jako doplněný, a je možné ho znovu nahlásit.

Uživatel si také může vybrat, aby mu po doplnění šuplíku přišel mail s informací, že už byl šuplík naplněn a že může si pro danou součástku přijít. Maily se posílají přes firemní e-mailový server. K nahlášení lze i připsat poznámku (např. informace o konkrétním druhu rezistoru).

3 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.

3.1 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 se vrací uživateli jako výsledek. V aplikaci používám logiku, že každý View má svůj Controller s příslušnými metodami pro zpracování dat. Controller obsahuje až stovky řádků kódu a až desítky metod pro zpracování dat.

3.2 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. Lze tedy např. opakovat několikrát HTML kód pomocí cyklů, či používat proměnné bez nutnosti neustále používat tagy, pro označení PHP kódu. Blade kód se poté převádí na čisté PHP, jde tedy pouze o zjednodušení psaní a přehlednosti kódu.

3.3 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, nastavíme mu parametry a použijeme na něj metodu pro uložení. Podobným způsobem se dají i záznamy upravovat, či mazat. V modelu se dají nastavit různé proměnné, díky kterým lze měnit připojenou tabulku, způsob práce s primárním klíčem atd.

3.4 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. Každá URL adresa má také své jméno, díky které lze na ni odkazovat přímo přes metodu.

3.5 Způsob fungování

Uživatel zadá do internetového prohlížeče určitou URL adresu. Tyto adresy jsou vypsané v souboru „routes/web.php“. Podle této adresy se vybere přiřazený Controller a jeho konkrétní metoda. Tato metoda se zavolá, jako parametr metody lze posílat třídu „Request“ – to se používá např. v případě vyřizování výsledku formuláře. Metoda v Controlleru zpracuje data např. použitím Modelů a vrací View (blade šablonu). Do View lze také přidat nějaké parametry, jako např. pole prvků, přes které se poté např. iterovat cyklem foreach.

3.6 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 pomocný příkaz na vygenerování šablony pro vytvoření vlastního příkazu.

3.6.1 Příkaz na generování štítků

Příkaz „`php artisan generate:tags`“ je v aplikaci použit pro generování textových souborů, které lze poté poslat do tiskárny. Používají speciální formát zapsání jednotlivých řádků a jsou využity pro tisk štítků se souřadnicí šuplíku, který má každý šuplík na své zadní straně. Tento způsob označování šuplíků je použit, aby se při odebrání několika šuplíku ze stěny poté nevracely na špatné místo. Příkaz vygeneruje textové soubory pro

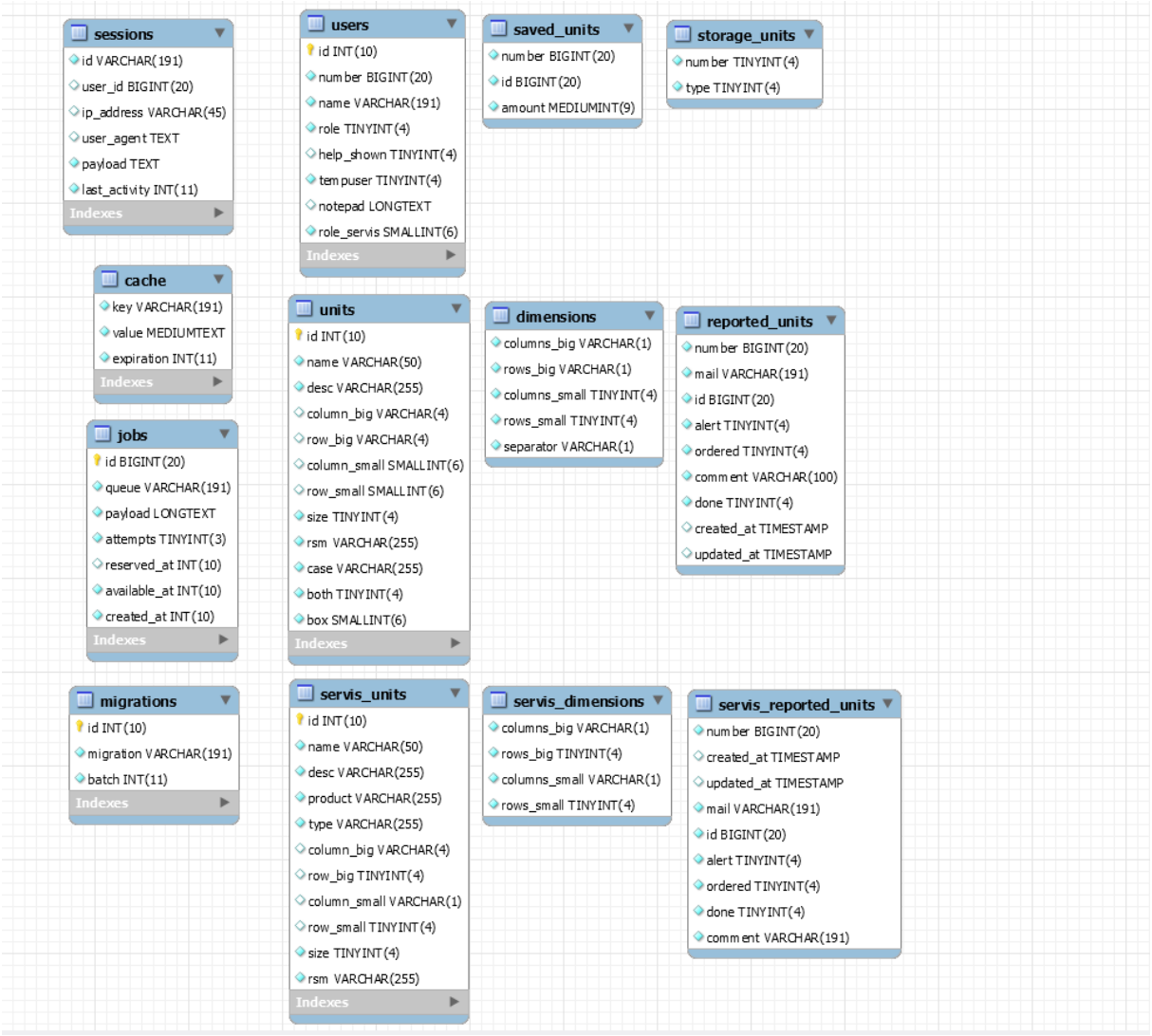
všechny skříňky na stěně, lze ovšem vygenerovat i textový soubor pro každou skříňku zvlášť – v sekci „Grafické zobrazení“.

3.6.2 Příkazy pro prvotní naplnění tabulek

Příkazy „`php artisan fill:units`“ a „`php artisan fill:dimensions`“ se používají pro naplnění dat do tabulek v databázi. Prakticky jde o vytvoření prázdných šuplíků. Tyto příkazy je nutné provést při prvotním nasazení aplikace, protože jinak aplikace popírá logiku šuplíků – na stěně vždy jsou, i když jsou prázdné. Nejdříve se zadává příkaz, který podle parametrů naplní tabulku dimensions danými rozměry. Poté druhý příkaz nejdříve přečte data z tabulky dimensions a podle daných rozměrů vygeneruje určitý počet prázdných šuplíků.

4 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.



Obrázek 4 - Schéma databáze

4.1 Obecná struktura databáze

V aplikaci jsou použité dvě databáze. Jedna z toho je interní firemní databáze se skladovými kartami součástek, kterou používám pouze pro čtení (na 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**).

4.2 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.

V tabulce pro uživatele jsou dva sloupce, které určují uživatelovu roli. Sloupce jsou dva kvůli rozdělení aplikace na části. Možné hodnoty jsou buď normální uživatel (hodnota 1), nebo administrátor (hodnota 2). Administrátor má oproti normálnímu uživateli rozšířené funkce a přístup do administrace.

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.

4.3 Migrace

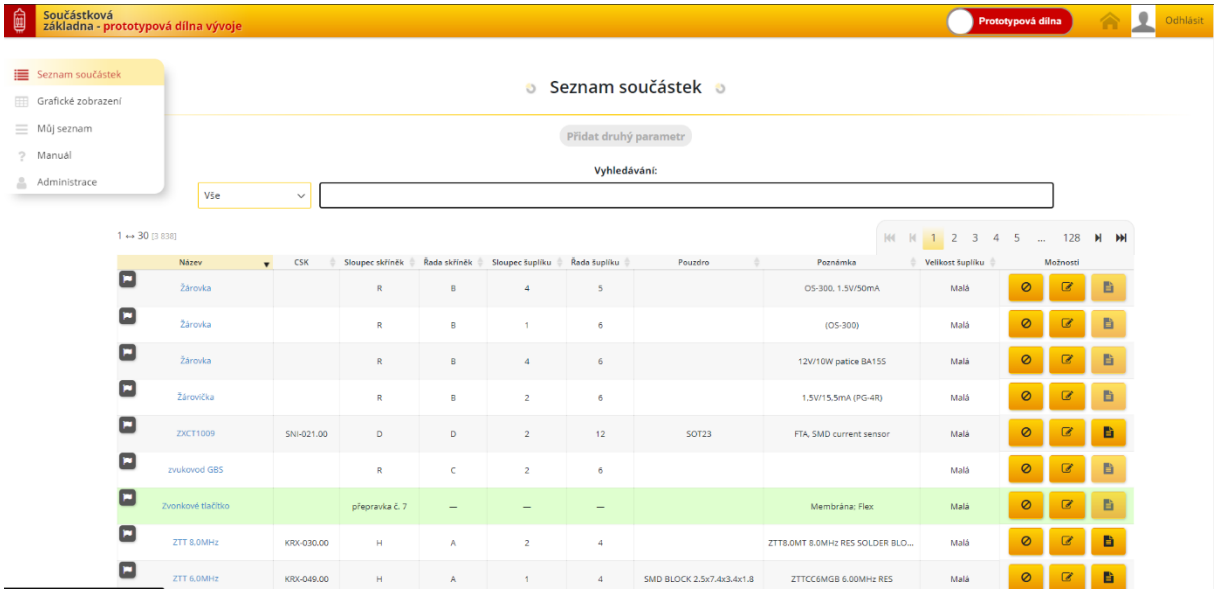
Pro bezproblémové nasazení aplikace jsou ve složce „database/migrations/“ umístěny soubory s migracemi databáze. V nich jsou detailně popsány všechny tabulky, co se mají v databázi vytvořit, jejich datové typy, defaultní hodnoty a další atributy. Díky tomu lze migrovat pouze určité tabulky. Práce s migracemi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

5 Sekce aplikace

Aplikaci jsem rozdělil do jednotlivých sekcí tak, aby bylo ovládání co nejvíce intuitivní. Uživatel si tedy buď potřebuje najít součástku podle jejího jména, nebo potřebuje najít součástku podle její pozice. Podle toho si vybírá, do které ze dvou základních částí aplikace zamíří.

5.1 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, popř. hledat jen v určité kategorii součástek (podle začátku CSK). Výsledek vyhledávání je poté zformátován do tabulky se všemi dostupnými informacemi o součástce. Tabulka s výsledkem vyhledávání obsahuje na pravé straně navigační panel, kde lze přepínat mezi jednotlivými stránkami. Na levé straně tabulky se nachází informace o celkovém počtu prvků a o počtu filtrovaných prvků. Uživatel si také může nastavit, aby byly výsledky vyhledávání seřazeny podle určitého sloupce. Pokud je uživatel bez administrátorských práv, může nahlásit prázdný šuplík, přidat si součástku do seznamu a zobrazit si její datasheet. Administrátor může navíc upravovat informace o součástce, vyprázdnit šuplík.



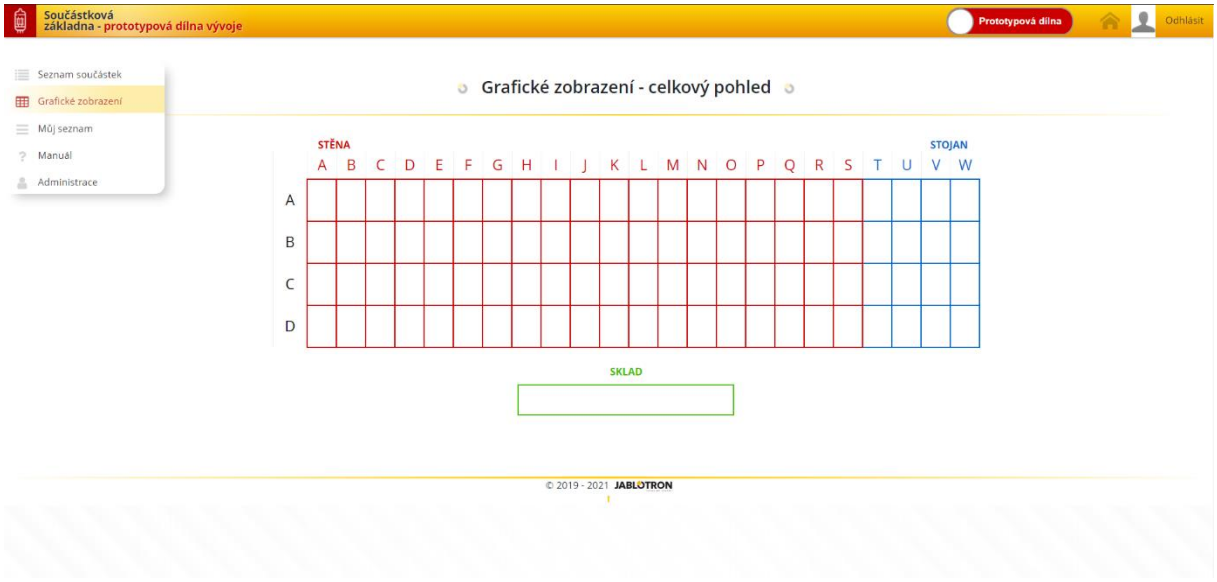
Obrázek 5 - Screenshot vyhledávání

5.2 Grafické zobrazení

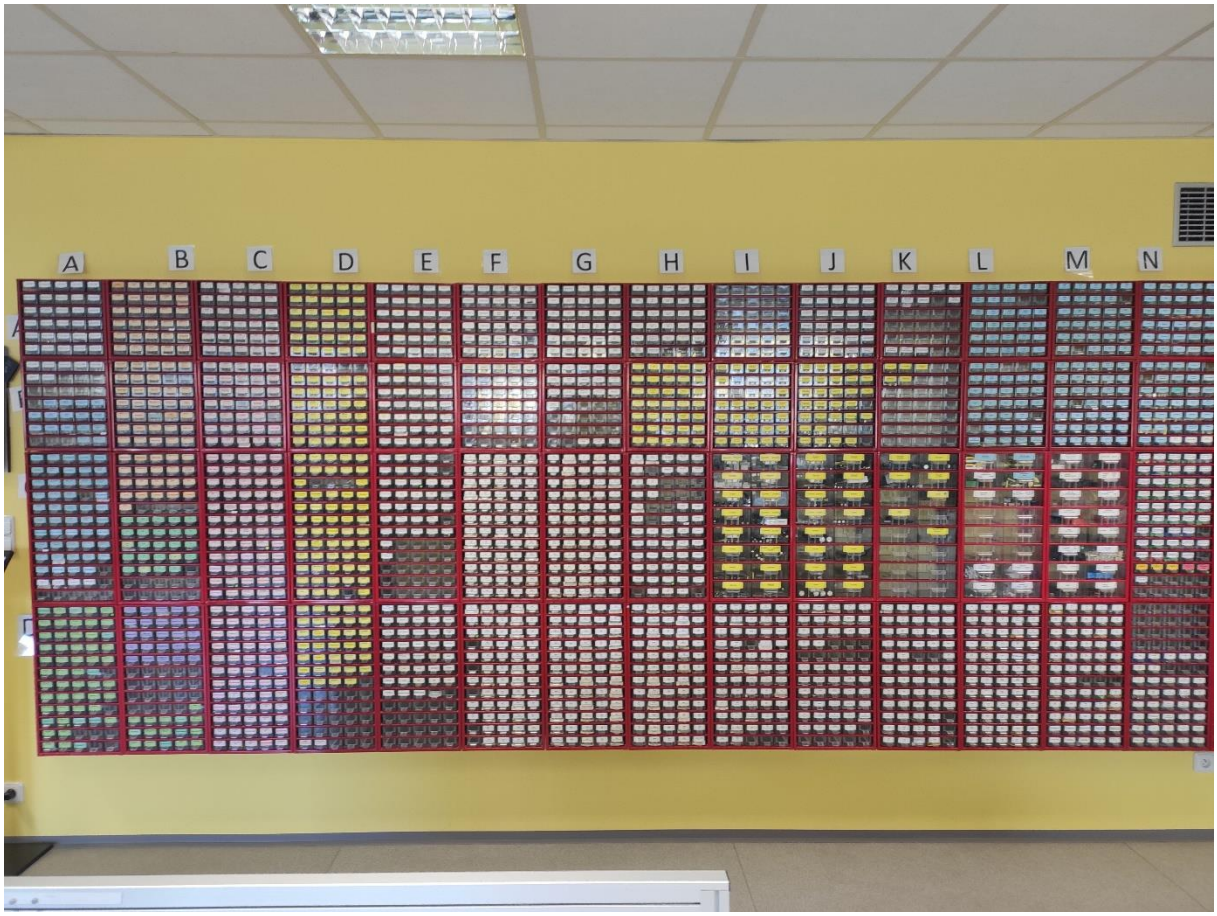
Naopak, v případě, že uživatel zná pozici součástky a potřebuje nalézt její jméno, popř. ji upravit, použije tuto část aplikace. Grafické zobrazení slouží jako elektronická vizualizace

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.



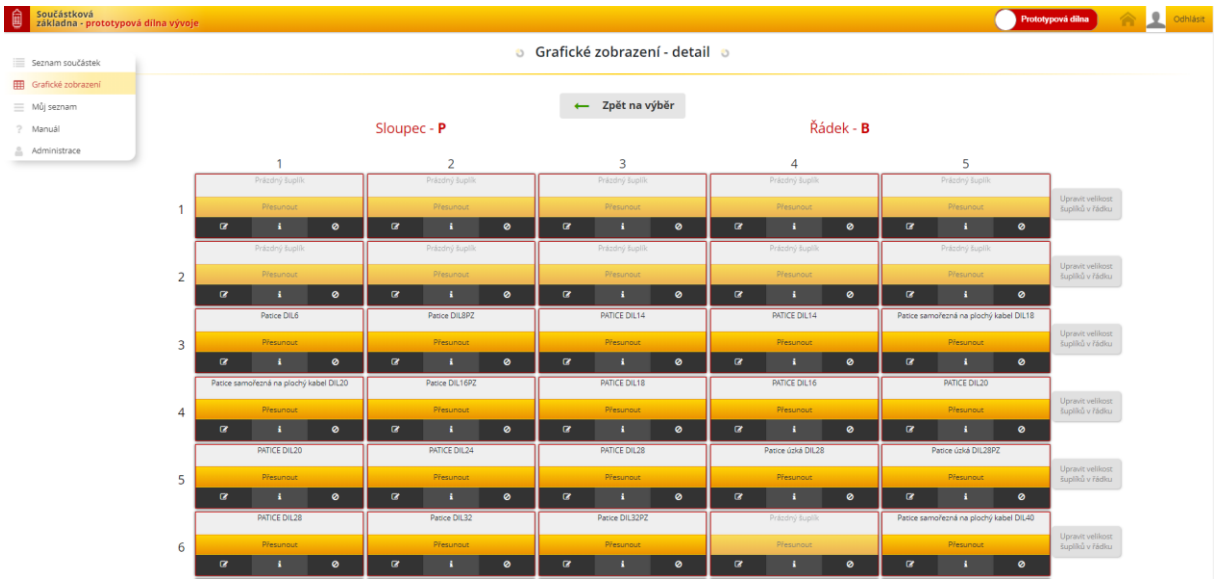
Obrázek 6 - Screenshot grafického zobrazení



Obrázek 7 - Stěna součástkové základny

5.2.1 Detailní zobrazení

Po kliknutí na určitou skříňku se otevře detailní zobrazení této skříňky, kde jsou vidět jednotlivé šuplíky. Lze měnit jejich velikost (malý – 5 vedle sebe, střední – 2 vedle sebe, velký – 1 šuplík na řádek), ovšem pouze v případě, že jsou prázdné. Dále lze řádky přidávat a mazat. Díky těmto funkcím lze vytvořit přesně takovou strukturu šuplíků, jako se může vyskytnout na stěně – protože každá skříňka může vypadat trochu jinak. Jednotlivým šuplíkům lze také upravovat jejich vlastnosti, zobrazit si informace u šuplíku, přesouvat je (i mezi skříňkami) a v neposlední řadě šuplík vyprázdnit. Další funkcí detailního zobrazení je možnost vygenerovat textový soubor pro tisk štítků na konkrétní šuplík.



Obrázek 8 - Detail skříňky v grafickém zobrazení

Obrázek 9 - Detail skříňky se součástkami

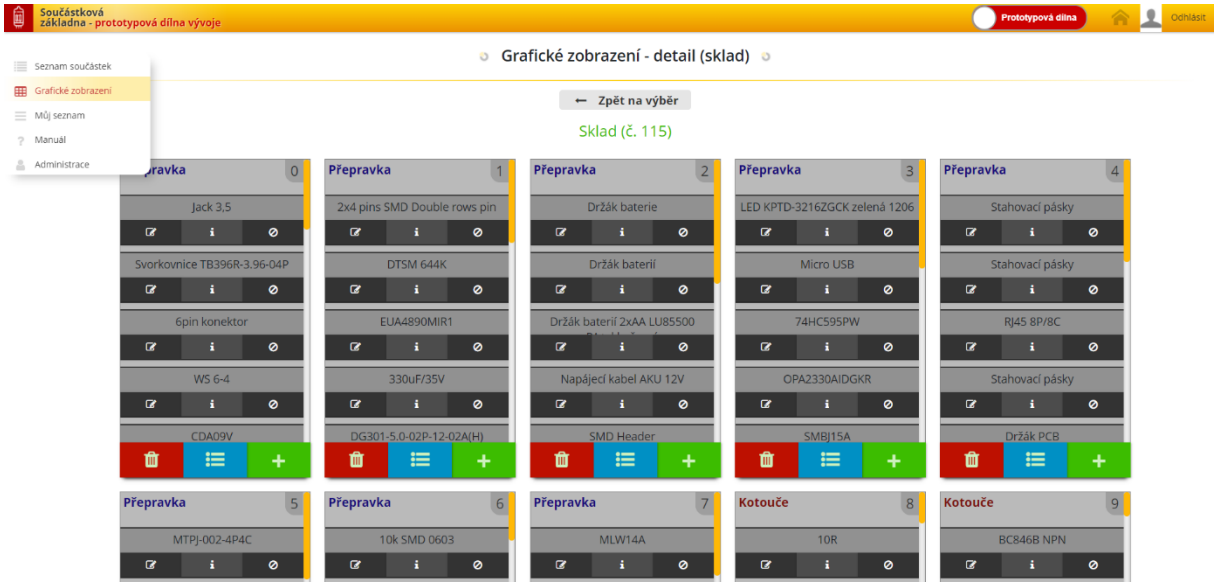
5.2.2 Sklad

Sklad se fyzicky nachází za stěnou součástkové základny. Jde o místnost, ve které jsou uloženy některé ze součástek pro budoucí doplňování. Součástky jsou uloženy buď v přepravkách, nebo na kotoučích (typicky rezistory). V aplikaci jsou tyto možnosti uložení vyobrazeny a součástky lze mezi nimi přetahovat systémem drag and drop – poté se ukládají do databáze přes AJAX.

Ve skladu lze vymazat obsah celé přepravky / kotouče, zobrazit si seznam položek v přehledném konkrétním zobrazení a přidávat položky do přepravky / kotouče. S položkami jde i manipulovat konkrétně, tedy mazat, editovat a zobrazovat si o nich informace.

Součástky mají v interakci se skladem dva stavy. Buď jsou pouze ve skladu (typicky méně používané součástky) – ve vyhledávání jsou znázorněny zelenou barvou řádku. Místo souřadnic takové součástky mají pouze zapsané pouze přepravky a souřadnice jsou proškrtané. Druhým stavem je, že se součástka nachází jak ve skladu, tak na stěně. Takové součástky jsou označeny normálními souřadnicemi ze stěny, ovšem navíc u sebe mají zelenou tečku. Toto označení je zvoleno z toho důvodu, že stejným způsobem jsou označeny fyzicky na svém štítku na stěně. Toto rozlišení je důležité z důvodu, že když

jedna součástka na stěně dojde a je ve skladu, nemusí se objednávat, ale může se rovnou doplnit.

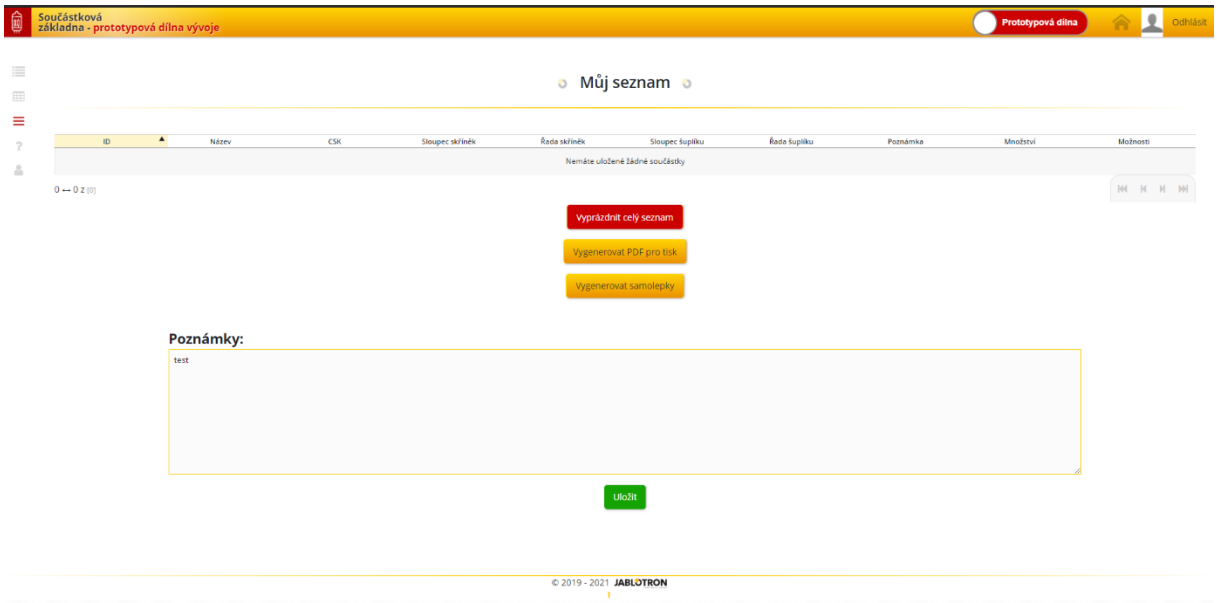


Obrázek 10 - Screenshot zobrazení skladu

Obrázek 11 - Přepravky a kotouče ve skladu

5.3 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. Seznam obsahuje možnost si pro vybrané součástky vygenerovat přehledný PDF soubor, který se poté dá použít např. na tisk. Administrátor také může z vybraných součástek vygenerovat textový soubor pro tisk štítků. Každý uživatel si také může napsat cokoliv do poznámek, které také lze zobrazit v tabletu.



Obrázek 12 - Screenshot sekce Můj seznam

5.4 Manuál

V manuálu jsou popsané tři základní části aplikace pro nové uživatele (např. nové členy firmy), kteří začínají aplikaci používat. Aplikace je ovšem navržena tak, aby pro většinu uživatelů manuál nebyl ani potřeba, díky jejím jednoduchým a intuitivním funkcím. Pokud v aplikace přibude větší množství funkcí, lze z Administrace zobrazit všem uživatelům hlášku, aby si přečetli manuál.

5.5 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.

5.5.1 Konkrétní popis rozšiřujících funkcí administrace

Díky funkci „Dočasná role uživatele“ lze aplikaci testovat z pohledu uživatele. Například v sekci „Seznam součástek“ administrátor nevidí tlačítka pro přidání do seznamu, místo nich tam má tlačítka pro nahlášení součástky.

Rozdělovač je písmenko abecedy z části aplikace „Prototypová dílna vývoje“. Jeho maximální hodnotu určuje databázová tabulka dimensions, konkrétně sloupec „columns_big“. Písmenko určuje, kde se označení sloupců na stěně mění na označení sloupců na stojanu. Jediná funkce je prakticky barevné odlišení v sekci „Grafické zobrazení“, jinak funkce sloupců na stěně a na stojanu zůstává stejná.

Funkce „Restartovat upozornění na manuál“ se používá v případě, že v aplikaci se přidá za krátkou dobu velké množství změn a zároveň se zapíšou do sekce „Manuál“. V tomto případě každému uživateli po prvním spuštění aplikace vyskočí pop-up upozornění. Funkčnost zajišťuje sloupec „help_shown“ v tabulce users, který se po prvním spuštění aplikace uživatelem aktualizuje.

6 Řešení problémů

Pro znázornění některých postupů v aplikaci jsem se rozhodl zvýraznit a popsat některé části kódu. Na příkladech lze názorně vidět, jaké jsem zvolil postupy při řešení určitých problémů. Dále jsem vybral některé z hlavních problémů, které jsem při vývoji aplikace řešil, a popsal způsob jejich řešení.

6.1 Problém s přesouváním šuplíků

Při tvorbě aplikace jsem narazil na problém s měněním velikostí šuplíku. Když jsem z malého šuplíku udělal střední šuplík, který byl poté přesunut na místo jiného malého šuplíku, v daném řádku najednou nastal zakázaný stav – na jednom řádku bylo více druhů velikostí šuplíku. Tento problém byl ošetřen tím, že šuplík při přesouvání nenese svou velikost, nýbrž pouze informace o něm (název, poznámka, CSK…). Lze tedy přesunout šuplík na jinou pozici, která má rozdílnou velikost a šuplíky si zanechají své velikosti.

6.2 Problém s evidencí počtu součástek v šuplíku

Během vymýšlení možných funkcí v aplikaci přišel návrh, že by každý šuplík mohl mít navíc informaci o počtu součástek v něm. Tím by odpadla nutnost mít tlačítko pro nahlášení prázdného šuplíku, jelikož by se šuplík nahlásil sám, když počet součástek v něm klesne na nulu. Nevýhoda tohoto systému je ovšem fakt, že musí uživatelé evidovat přesný počet součástek, které odeberou. Pokud některý z nich udělá chybu, celý systém začne být nepřesný, je tedy potřeba dělat pravidelné kontroly. Z důvodu těchto nevýhod, které převyšují pozitiva, byla nakonec tato funkce zamítnuta.

6.3 Problém se skladem

V první verzi aplikace byla pouze stěna a stojan. Ty jsou v podstatě totožné, stojan je pouze rozšířením stěny. Když jsme se později rozhodli rozšířit evidenci i pro sklad, byl problém se způsobem zobrazování skladu. Jelikož už nešlo o formát, kde každá součástka měla svou souřadnici, nebylo možné aplikovat pro sklad stejné zobrazování, jako má stěna a stojan. Zvolil jsem tedy jednotlivé kontejnery (přepravky a kotouče), které neobsahují součástky se souřadnicemi, ovšem každá součástka nese pouze informaci, v jakém kontejneru se nachází. Tato informace je v databázi ve sloupci „col_big“, kde se normálně nachází písmenko sloupce skříněk – ostatní sloupce pro souřadnice jsou NULL. Díky tomu

lze rozeznat součástky ve skladu, protože mají vyplněný pouze jeden databázový sloupec mezi souřadnicemi.

Vzhledem k jednoduššímu zobrazení jednotlivých kontejnerů ve skladu lze součástky mezi nimi přesouvat jednoduše, pomocí metody „drag and drop“. V JavaScriptu jsou vytvořeny funkce pro umožnění hýbání s HTML prvkem. Když ho uživatel poté pustí do jiné přepravky, JavaScript pošle AJAX požadavek na server, který změní u dané součástky číslo jejího kontejneru.

6.4 Refaktorování kódu

Po nějaké době používání aplikace a pravidelného vytváření změn v aplikaci jsem se rozhodl o „refactoring kódu“. Jde o změnu kódu, která nemá vliv na funkčnost systému (uživatel nevidí změnu). Od tohoto postupu lze očekávat větší přehlednost kódu pro vytváření budoucích změn někým jiným.

První změnou bylo vytvoření nových šablon jako náhrada za formuláře, které se v back-end kódu často opakovaly a zabíraly tam příliš mnoho místa. Vytvořil jsem tedy šablony, které se nacházejí ve složce „resources/views/layouts“. Pro každý formulář použitý v back-endu byla vytvořena jedna šablona. Do této šablony lze jako parametr přenést např. informace o součástce. Šablony fungují jako klasický View, používám v nich i Blade šablony. Díky tomuto kroku se jednodušeji orientuje v kódu Controllerů, především v kódu vyhledávacího systému, kde jsou podobné formuláře použity často.

V aplikaci je často používáno testování, zda je uživatel admin. Z toho důvodu jsem přidal metodu na Model třídu User, která testuje, zda je uživatel admin podle momentální spuštěné verze aplikace (Prototypová dílna vývoje / Servis). Díky této metodě jsem zpřehlednil testování uživatele na administrátorská práva.

Pro jednodušší nastavování proměnných (např. při novém nasazení aplikace) jsem se rozhodl více využít konfigurační soubor „.env“. Přidal jsem do něj celkově 5 nových položek, kterými jsou:

- E-maily pro posílání informací o nahlášení součástek ze Servisu
- E-maily pro posílání informací o nahlášení součástek z Prototypové dílny vývoje
- Statická IP adresa tabletu
- URL adresy API, ze kterých aplikace čte data o uživatelích

Posledním krokem v refaktorování kódu bylo odstranit zbytečné PHP z Blade šablon ve View a přesunout ho do Controlleru, kde by se měla odehrávat většina logiky aplikace. V Controlleru se tedy data vypočítají a poté se do View pošlou jako parametry, tam jsou pak přístupné v proměnné.

6.5 Problém s uživatelskými účty

Při propojení aplikace s tabletem, který je fyzicky umístěn vedle stěny součástek nastal problém s uživatelskými účty. Aby aplikace fungovala, na tabletu musel být někdo přihlášen. Ovšem vzhledem k faktu, že tablet není žádná osoba, nemohl pro něj z bezpečnostních důvodů být vytvořený běžný účet, jako mají ostatní uživatelé. Tento problém byl nakonec vyřešen vytvořením podmínky v Middleware, která detekuje statickou IP adresu tabletu. V tomto případě obejde klasické přihlašování a do cookies místo toho uloží místo jména uživatele „Tablet“ a ostatní údaje (email, jméno nadřízený) zapíše jako prázdná. Díky tomu lze z tabletu nahlašovat prázdné šuplíky.

Závěr

Výsledkem této práce je funkční a v praxi používaná aplikace firmou Jablotron. Od začátku vývoje až do dnešní doby jsem na práci strávil odhadem 700 hodin. Aplikace stále není v konečném stavu a stále se rozšiřuje podle potřeb firmy. Díky refaktorování kódu a zvýšení jeho čitelnosti na aplikaci můžou pracovat i ostatní členové firmy a dělat na ní případné úpravy.

Při začátku vývoje aplikace jsem se framework Laravel učil od úplného minima (bez jakékoliv znalosti tohoto frameworku). Aplikaci jsem tvořil především podle oficiální dokumentace Laravelu, která je přehledná a dostačující. Kdybych aplikaci tvořil s momentálními znalostmi, některé věci bych udělal jinak (PHP kód ve View, migrace). Většinu z nich jsem v aplikaci ovšem již opravil.

S prací jsem spokojen hlavně díky faktu, že se ve firmě aktivně využívá. Aplikace je připravena na budoucí rozšiřování a případné změny. Znalosti PHP frameworku Laravel, které jsem získal při tvoření práce se také budou hodit.

Seznam obrázků

Použitá literatura

1. **Laravel.** Laravel dokumentace. [Online] Laravel, 1. 5 2020. https://laravel.com/docs/.

2. **Bootstrap.** Bootstrap dokumentace. [Online] Bootstrap, 1. 5 2020. https://getbootstrap.com/docs/.

3. **SpryMedia Ltd.** Datatables dokumentace. [Online] 1. 5 2020. https://datatables.net/manual/.

4. **Font Awesome.** Font Awesome - seznam ikon. [Online] 1. 5 2020. https://fontawesome.com/v4.7.0/.

5. **Oracle Corporation.** MySQL dokumentace. [Online] 1. 5 2020. https://dev.mysql.com/doc/.

A. Seznam přiložených souborů

Obsahem datového nosiče je:

- Složka „app“ – obsahuje kód aplikace
- Složka „images“ – obsahuje obrázky použité v dokumentaci
- Dokumentaci ve formátech .docx a .pdf