

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 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 i mění některé defaultní nastavení webu, jako je např. font nebo 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 intuitvní 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 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 jsem zvolil i použití knihovny jQuery. Ta zlepšuje čitelnost kódu a celkovou interakci JavaScriptu s ostatními částmi webu.

* 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.

Sekce „**Prototypová dílna vývoje**“ má stěnu, stojan a sklad. První dvě části fungují na stejném principu, jsou pouze vizuálně rozděleny. Naopak sklad funguje trochu jiným způsobem, protože místo skříněk jsou v něm přepravky a kotouče.

Sekce „**Servis**“ se liší tím, že mají pouze stojany. Z tohoto důvodu od nich přišel požadavek na změnu systému souřadnic.

* 1. 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. Vyhledávací systém

U vyhledávacího systému jsem se rozhodoval, 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). V mém případě se všechna data načtou přes JavaScriptovou knihovnu DataTables, která pak zobrazí pouze určité 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.

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

1. Požadavky na systém

Požadavky byly zadávány ze dvou stran. Tou první byl šéf 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 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 produkční 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. 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 open-source JavaScriptová knihovny DataTables, u které mění zobrazení navigace. Jednotlivé soubory jsou přehledně rozděleny do složek podle toho, kdo je jejich autorem. Vytvořil jsem složky pro moje soubory, pro soubory od firmy a pro volně dostupné knihovny.

* 1. 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.

* 1. 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 může si pro danou součástku přijít. K nahlášení lze i připsat poznámku (např. informace o konkrétním druhu rezistoru).

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. Způsob fungování

Uživatel zadá do internetového prohlížeče určitou URL adresu. Tyto adresy jsou vypsané v souboru „routs/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.

* 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. 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 každý šuplík má 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“ aplikace.

* + 1. 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. Příkaz nejdříve přečte data z tabulky dimensions. Poté podle daných rozměrů vygeneruje určitý počet prázdných šuplíků.

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.

1. 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.

* 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. Detailní zobrazení

Po kliknutí na určitou skříňku se otevře detailní zobrazení této skřínky, 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. Poslední funkcí detailního zobrazení je možnost vygenerovat textový soubor pro tisk štítků na konkrétní šuplík.

* + 1. 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.

* 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. 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.

* 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. 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 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.

1. Řešení problémů

Závěr

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

Seznam 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