

ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA PRÁCA

Študijný odbor: Informatika

Adam Bušfy

Správa majetku

Vedúci práce: doc. Ing. Michal Kvet, PhD.

Registračné číslo: 1227/2020

Žilina, 2021

Čestné vyhlásenie

Prehlasujem, že som bakalársku prácu Správa majetku vypracoval samostatne pod vedením doc. Ing. Michala Kveta, PhD. a uviedol v nej všetky použité literárne a iné odborné zdroje v súlade s právnymi predpismi, vnútornými predpismi Žilinskej univerzity a vnútornými aktmi riadenia Žilinskej univerzity a Fakulty riadenia a informatiky.

V Žiline, dňa

.....

Meno Priezvisko

Pod'akovanie

Na tomto mieste by som chcel poďakovať vedúcemu bakalárskej práce doc. Ing. Michalovi Kvetovi, PhD. za cenné pripomienky, ktorými prispel k vypracovaniu tejto bakalárskej práce.

ABSTRAKT

BUŠFY, Adam: *Správa majetku*. [Bakalárska práca]. – Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Katedra informatiky. – Vedúci: doc. Ing. Michal Kvet, PhD. – Stupeň odbornej kvalifikácie: bakalár. – Žilina: FRI ŽU, 2021 – 48s.

Cieľom práce je vytvoriť informačný systém pre evidenciu majetku s dôrazom na temporálny aspekt. Výsledná práca umožňuje evidovať majetok pomocou webovej aplikácie v rámci akejkoľvek organizácie, ktorú je možno hierarchicky rozdeliť na menšie celky. Používateľ má možnosť bezplatnej registrácie a následného spravovania majetku v lokalite, ktorá mu je pridelená správcom resp. administrátorom systému danej organizácie.

Kľúčové slová: Evidencia majetku. Správa majetku. Webová aplikácia.

ABSTRACT

BUŠFY, Adam: *Asset management*. [Bachelor thesis]. – The University of Žilina. Faculty of Management Science and Informatics; Department of informatics. – Tutor of the bachelor thesis: doc. Ing. Michal Kvet, PhD. – Level of professional qualifications: bachelor. – Žilina: FRI ŽU, 2021 – 48pgs.

The aim of the thesis is to create an information system for the registration of assets with emphasis on the temporal aspect. Final product allows users to manage assets of any organization using web application. This organization can be hierarchically divided into smaller units. Users have the option of free registration and management of locations which were assigned to them by system administrator of the organization.

Key words: Asset registry. Asset management. Web application.

Obsah

Zoznam obrázkov	8
Zoznam tabuliek	9
Zoznam skratiek	10
ÚVOD	11
1 Analýza súčasného stavu.....	12
1.1 Postup pri analýze súčasného stavu	12
1.2 Evidencia a správa majetku	12
1.3 Nástroj na evidenciu majetku Sortly	13
1.4 Nástroj na evidenciu majetku Asset panda	13
1.5 Nástroj na evidenciu majetku Money S3 inventory	14
1.6 Analýza správy majetku v prostredí UNIZA	15
1.7 Záver získaný analýzou.....	15
2 Požiadavky na systém	17
2.1 Všeobecné požiadavky.....	17
2.2 Bezpečnosť	17
2.3 Užívateľské rozhranie	18
2.3.1 Administrátorské rozhranie	18
2.3.2 Správcovské rozhranie	18
3 Výber programovacieho jazyka	19
3.1 PHP	19
3.2 Javascript.....	20
4 Použité technológie	21
4.1 Virtualizácia	21
4.1.1 Docker	21
4.2 MySQL databáza	22
4.2.1 MySQL Workbench	22
4.3 Webový server Nginx	22
4.4 Ajax volania	22
4.5 PHP composer.....	23
4.6 Twig	23
4.7 Doctrine.....	23
4.8 Git	23
4.9 Chart.js	23
5 Architektúra aplikácie	25
5.1 MVC architektúra	25

5.2	Symfony.....	26
6	Dátový model	27
6.1	Popis entít.....	27
6.2	Popis dátového modelu	29
7	Štruktúra projektu	31
7.1	Balík Entity	31
7.1.1	Mapovanie entít.....	31
7.2	Balík Repository	32
7.3	Balík Controller	32
7.4	Balík Form	33
7.5	Balík Template.....	34
8	Frontend	35
8.1	Prihlásenie.....	35
8.2	Registrácia.....	35
8.3	Domovská stránka.....	36
8.4	Items.....	37
8.4.1	Unassigned items	37
8.4.2	Assigned items	38
8.4.3	Import / Export.....	39
8.5	Discarded Items	40
8.6	Users	40
8.7	Categories a Locations.....	40
8.8	Notifications.....	41
8.9	Formuláre.....	41
8.9.1	Add form	41
8.9.2	Edit form	42
8.10	Item Detail	42
	Záver.....	44
	Zoznam použitej literatúry	45
	Zoznam príloh.....	46
	Prílohy	47
	Príloha A: Obsah DVD.....	48

Zoznam obrázkov

Obr. 1.1 Nástroj na evidenciu majetku Sortly	13
Obr. 1.2 Nástroj na evidenciu majetku Asset Panda	14
Obr. 1.3 Nástroj na evidenciu majetku Money S3	15
Obr. 2.1 Diagram základných prípadov použitia.....	17
Obr. 3.1 Typová deklarácia v jazyku PHP	19
Obr. 3.2 Arrow funkcia v jazyku PHP	20
Obr. 4.1 Inicializácia lineárneho grafu	24
Obr. 5.1 MVC architektúra.....	25
Obr. 6.1 Dátový model systému	29
Obr. 7.1 Príklad Doctrine ORM mapovania.....	31
Obr. 7.2 Príklad Repository triedy	32
Obr. 7.3 Príklad Controller triedy.....	33
Obr. 7.4 Príklad Form triedy	33
Obr. 7.5 Šablóna pre tlačidlo Detail	34
Obr. 8.1 Formulár na prihlásenie.....	35
Obr. 8.2 Formulár na registráciu	36
Obr. 8.3 Domovská stránka	37
Obr. 8.4 Unassigned items.....	38
Obr. 8.5 Assigned items	39
Obr. 8.6 Import / Export items	39
Obr. 8.7 Podstránka Locations	40
Obr. 8.8 Notifications podstránka	41
Obr. 8.9 Formulár na pridávanie predmetu	42
Obr. 8.10 Formulár na úpravu kategórie	42
Obr. 8.11 Podstránka Item Detail	43

Zoznam tabuliek

Tabuľka 6.1 Popis entity – Item	27
Tabuľka 6.2 Popis entity – Category	28
Tabuľka 6.3 Popis entity – Location	28
Tabuľka 6.4 Popis entity – User	28
Tabuľka 6.5 Popis entity – Notification	29
Tabuľka 6.6 Popis entity – History	29

Zoznam skratiek

OS	O perating S ystem
API	A pplication P rogramming I nterface
URL	U niform R esource L ocator

ÚVOD

Obrovský rozmach v odvetví hromadnej výroby so sebou prináša nespornú potrebu mať veci pod kontrolou. Keďže evidencia majetku sa stáva čoraz náročnejšia a na jeho samotnú správu už dávno nestačia ručne písané papierové záznamy, preniesli sa tieto nástroje do sveta technológií.

Aj z tohto dôvodu sme sa rozhodli navrhnúť systém zameraný na evidenciu a správu majetku. Softvér bol tvorený so zámerom možného využitia v prostredí Žilinskej univerzity, ktorá podobným systémom aktuálne nedisponuje.

V posledných rokoch sa dostáva do popredia najmä oblasť, ktorá sa zaoberá tvorbou webových stránok. Dôvodom je hlavne jednoduchý prístup k informáciám bez nutnosti dodatočnej inštalácie softvéru. Spoločnosti sa čoraz viac obracajú na tvorcov webových aplikácií so žiadosťou spracovania ich požiadaviek.

Takto spracované riešenia majú hneď niekoľko výhod. K ich použitiu stačí obyčajný webový prehliadač, ktorý zvykne byť súčasťou každého zakúpeného zariadenia. Obsah stránky je tak možné prezerať ako na počítači, tak aj na každom smartfóne. Práve preto sme sa pri spracovaní zadania tiež vydali cestou webového rozhrania, ktoré v sebe zahŕňa najnovšie použité technológie.

Výsledný produkt tak poskytuje užívateľsky prívetivé prostredie. Oproti mnohým iným nástrojom, ten náš nevyžaduje kúpu licencie či žiadnu inú formu platby za používanie. Vďaka tomu, že je celý softvér zasadený do prostredia prehliadača, nie je obmedzená ani užívateľská platforma a stránku je taktiež možné navštíviť s použitím viacerých operačných systémov.

Cieľom tejto práce je oboznámiť čitateľa s procesom návrhu, implementácie a výsledným produktom. V úvodných kapitolách sa bližšie venujeme analýze aktuálne existujúcich systémov pre správu majetku. Porovnávame všetky klady a zápory existujúcich nástrojov a poskytujeme tak čitateľovi ucelený pohľad na danú problematiku.

Práca ďalej obsahuje špecifikáciu požiadaviek, opis použitých technológií a venuje sa aj dátovému modelu a použitej architektúre.

V závere práce vysvetľujeme funkčnosť jednotlivých podstránok, ktoré sa v aplikácii nachádzajú.

1 Analýza súčasného stavu

Skôr než sme sa pustili do samotnej tvorby systému bolo potrebné vykonať analýzu a preskúmať tak aktuálnu ponuku softvérov dostupných na trhu.

1.1 Postup pri analýze súčasného stavu

Analýzou sme chceli dospieť k preskúmaniu čo najširšieho spektra systémov ponúkaných na trhu. Najčastejším problémom pri samotnej analýze, bola nutnosť zakúpenia licencie pre prístup ku základným funkcionalitám softvéru a tiež chýbajúca voľne dostupná testovacia verzia. Preto sme sa obracali na video dokumentáciu a články poskytujúce bližší popis systému. Zahrnúť sme chceli produkty zahraničných ale aj domácich poskytovateľov.

Medzi niektoré z nich patria napríklad systémy:

- Sortly
- Asset Panda
- Money S3

Avšak skôr ako si opíšeme jednotlivé systémy, je potrebné povedať čo všetko so sebou evidencia a správa majetku prináša.

1.2 Evidencia a správa majetku

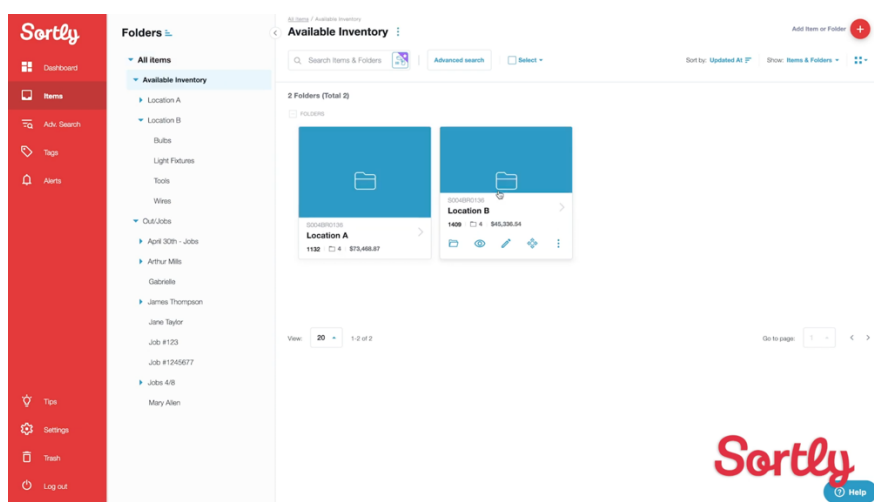
Pod evidenciu a správu majetku spadá mnoho oblastí, ktoré musíme spojiť do jedného celku. Každý spravovaný predmet sa vyvíja v čase, čo zahŕňa nevyhnutnosť monitorovať jeho históriu zmien. Predmet má svoj názov, kategóriu, lokalitu či napríklad správcu ktorý naň dohliada. O predmetoch je teda potrebné viesť záznamy s pravidelným zachytávaním sledovaných dát.

Systém pre správu preto musí obsahovať spojenie všetkých oblastí tak, aby sa výsledné riešenie dalo použiť pre akúkoľvek organizáciu, ktorá disponuje majetkom a potrebou jeho evidencie.

1.3 Nástroj na evidenciu majetku Sortly

Softvér Sortly je veľmi jednoduchý a prehľadný nástroj na správu akýchkoľvek predmetov. Rozhranie tohto systému je zasadené do prehľadného prostredia, ktoré poskytuje príjemný užívateľský zážitok. Keďže sa Sortly radí ku webovým aplikáciám je veľkým pozitívom aj platformová nezávislosť.

Prostredie webového prehliadača poskytuje okrem iného aj možnosti ako skenovanie čiarového kódu a QR kódu pomocou fotoaparátu čo prináša zásadne rýchlejšie pridávanie záznamov do systému. Výhodou môže byť aj dokumentácia, ktorá je voľne dostupná vo forme krátkych video návodov.



Obr. 1.1 Nástroj na evidenciu majetku Sortly

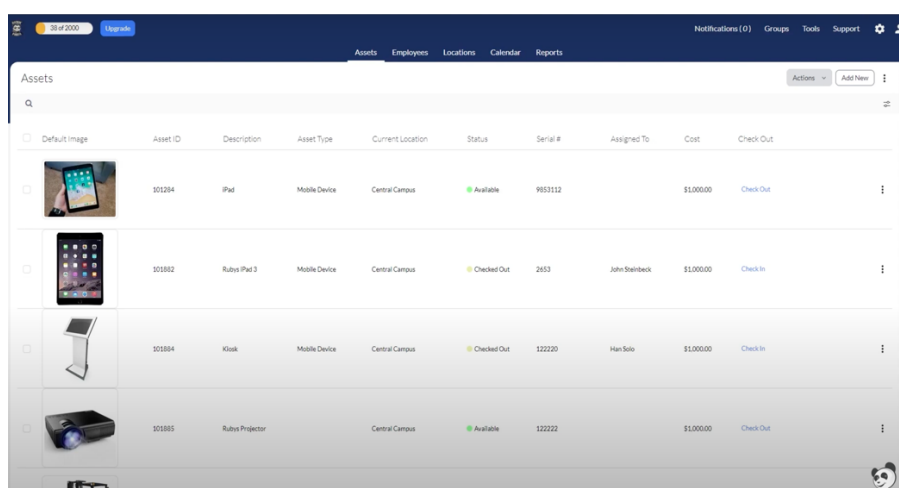
Jednou z nevýhod systému Sortly je, že ak by sme sa rozhodli túto službu používať, museli by sme pristúpiť na spoplatnený mesačný plán. Na výber je hneď niekoľko takýchto plánov, no časť z nich je obmedzená počtom funkcionalít. Plnú slobodu používania nám ponúka až najdrahší plán, ktorého cena sa však šplhá do vysokých čísel.

1.4 Nástroj na evidenciu majetku Asset panda

Ďalším z nástrojov je Asset Panda. Jedná sa o systém, ktorý je určený predovšetkým väčším spoločnostiam, v ktorých sa správa a evidencia majetku vzhľadom na jeho množstvo stáva nemožná.

Celá štruktúra aplikácie je zameraná najmä na synchronizáciu a súhrn medzi pracovníkmi, ktorí ju používajú. Ak chceme rozprávať o flexibilita tak Asset Panda ju vo veľkej miere poskytuje.

V prostredí nástroja môžeme nájsť aj veľké množstvo štatistík, ktoré sú tvorené na základe údajov množstva či histórie pohybu spravovaných predmetov. Štatistiky je možné taktiež v prehľadnej forme exportovať napríklad vo forme tabuliek.



The screenshot shows the 'Assets' page of the Asset Panda application. It features a table with columns for Asset ID, Description, Asset Type, Current Location, Status, Serial #, Assigned To, Cost, and Check Out. There are four rows of assets listed, each with a corresponding image icon on the left. The status of each asset is indicated by a colored dot (green for Available, yellow for Checked Out).

Asset ID	Description	Asset Type	Current Location	Status	Serial #	Assigned To	Cost	Check Out
101284	iPad	Mobile Device	Central Campus	Available	9853112		\$1,000.00	Check Out
101882	Ruby iPad 3	Mobile Device	Central Campus	Checked Out	2653	John Steinbeck	\$1,000.00	Check In
101884	Kiosk	Mobile Device	Central Campus	Checked Out	122220	Han Solo	\$1,000.00	Check In
101885	Ruby Projector		Central Campus	Available	122222		\$1,000.00	Check Out

Obr. 1.2 Nástroj na evidenciu majetku Asset Panda

Ako aj pri predchádzajúcom nástroji nevýhodou tohto je absencia používania zadarmo. Avšak vďaka skúšobnému obdobiu je pred rozhodnutím o zakúpení užívateľovi poskytnutý krátky pohľad do vnútra systému.

1.5 Nástroj na evidenciu majetku Money S3 inventory

Nástroj Money S3 sa radí k najväčším slovenským systémom pre správu majetku. Medzi hlavné zamerania softvéru patrí najmä skladová agenda. Jedným z hlavných dôvodov jeho popularity je prepojenie systému s internetovým obchodom. Vďaka predajom z minulého obdobia softvér presnejšie odhadne, koľko tovaru bude potrebné v jednotlivých obdobiach zaobstarať. Okrem toho Money S3 poskytuje aj vedenie účtovníctva vrátane odpisových plánov či vytvárania faktúr. Prepojenie účtovníctva s prípravou prehľadných grafov umožňuje užívateľovi získať manažérsky pohľad na svoje dáta.

Stratka	Stav zásoby	Obstarávacia cca PLU	Číslo katalogu	Čiarový kód
AVG Antivirus	30.0000	0.0000	000014	AV00000001
Cestovné náhrady	0.0000	0.0000	000012	SLUCE0001
Expedičné náklady	0.0000	0.0000	000013	SLUCEP001
Kábel čierny	4.0000	0.2500	000018	KABCE0001
Kábel prepisovací	7.0000	0.9500	000020	KABPRE001
Kancelársky papier	3 508.0000	0.1000	000019	KON0001
Klávesnica multimedialná	70.0000	15.0000	000004	KEYMUL001
Klávesnica štandardná	40.0000	5.0000	000003	KEYSTD001
Mechanika CD	124.0000	9.7000	000006	MECCD001
Mechanika DVD	69.0000	24.4325	000007	MECCDV001
Mechanika FDD	65.0000	9.0495	000005	MECFDD001
Monitor LCD	12.0000	118.2203	000002	MONLCD001
Monitor STD	19.0000	69.0000	000001	MONSTD001
Myš optická	22.0000	8.5000	000009	MYSOPT001
Myš štandardná	36.0000	1.5000	000008	MYSTD001
PC zostava - komplet	0.0000	0.0000	000015	ZOSP0001
Podčiarka pod reči	42.0000	0.6000	000010	MYSPOD001
Podčiarka pod reči 10k - sada	0.0000	0.0000	000016	MYSSAD001
Servisné služby	0.0000	0.0000	000011	SLUSER001
Snímač čiarových kódov - VC	6.0000	88.9500	000017	SKLAS001

Obr. 1.3 Nástroj na evidenciu majetku Money S3

Slabou stránkou tohto nástroja je menej intuitívne užívateľské rozhranie, ktoré môže na prvý pohľad pôsobiť zastarane. Na záver treba ešte podotknúť, že Money S3 vyžaduje stiahnutie softvéru do počítača čo obmedzuje voľnosť v použití ľubovoľnej platformy.

1.6 Analýza správy majetku v prostredí UNIZA

Žilinská univerzita sa radí k najväčším študentským inštitúciám na Slovensku a preto je zrejmé, že disponuje veľkým množstvom majetku. Či už sa jedná o výbavu učební, prednáškových sál alebo iných priestorov školy, je potrebné viesť záznamy o každom existujúcom predmete ktorý sa v priestoroch nachádza alebo o každom, ktorý je zakúpený a následne do priestorov priradený. Celá univerzita má hierarchickú štruktúru, ktorá je zložená z viacerých fakúlt. Tie sú ďalej členené na katedry. V súčasnosti neexistuje žiaden ucelený systém, ktorý by otázku správy majetku v rámci školy riešil čo celú evidenciu sťažuje.

1.7 Záver získaný analýzou

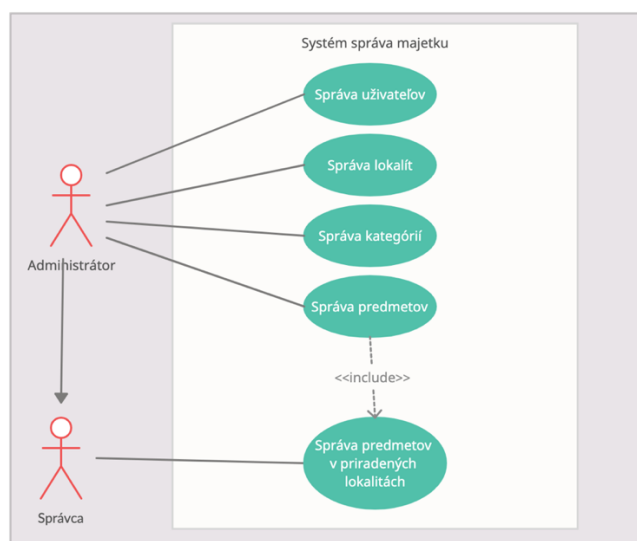
Po podrobnom vykonaní analýzy, dôslednom preskúmaní aktuálneho stavu a zistení, že Žilinská univerzita nedisponuje žiadnym podobným systémom sme dospeli k záveru, že má zmysel takýto softvér vytvoriť. Pri vývoji sme sa snažili vyhýbať nedostatkom existujúcich nástrojov pričom sme sa snažili dbať na jednoduchosť

používania. Otázka správy majetku so sebou prináša okrem iného aj potrebu zabezpečiť zdieľané dáta v systéme. Každý užívateľ má svoju rolu, na základe ktorej má prístup k funkciám systému. Hierarchická štruktúra vyžaduje aj rolu administrátora, ktorý je správcom celého systému. Jeho úlohou je vytvoriť štruktúru organizácie a zadefinovať kategórie predmetov. Následne má administrátor možnosť prerozdeliť užívateľom prístup k správe lokalít a spraviť z nich tak správcov daných lokalít. Pri návrhu sme sa snažili dbať na nezávislosť užívateľskej platformy a chceli sme aby boli všetky funkcie dostupné zadarmo bez potreby dodatočného zakúpenia licencie.

2 Požiadavky na systém

2.1 Všeobecné požiadavky

Pri skúmaní problematiky sme zisťovali akú funkčnosť by novo vytvorený systém mal spĺňať. V systéme nesmie chýbať možnosť zadefinovať základnú štruktúru, nad ktorou bude mať administrátor plnú kontrolu. Menším celkom hierarchickej štruktúry sa bude dať následne priradiť správcov, ktorí ich budú mať na starosti. O predmetoch sa musí viesť história, ktorá zaznamenáva celý priebeh vývoja každej zmeny. Predmety v systéme sa musia deliť na voľne dostupné, tie ktoré sú priradené do existujúcej lokality, vyradené alebo úplne zmazané.



Obr. 2.1 Diagram základných prípadov použitia

2.2 Bezpečnosť

Užívateľovi by malo byť poskytnuté konto, ktoré si má možnosť vytvoriť pomocou registračného formulára a prístup do systému by mal byť následne dostupný skrz prihlasovací formulár. Heslo zadané pri registrácii musí byť zakódované podľa dnešných štandardov a bezpečne uložené v databáze rovnako ako všetky údaje, ktoré sú pri registrácii zadané.

2.3 Užívateľské rozhranie

Ako sme si všimli už pri práci s pozorovanými nástrojmi základom je jednoduché rozhranie a prístup do systému bez nutnosti dodatočného sťahovania softvéru. V nástroji by sa nemali nachádzať žiadne rušivé prvky, či už vo forme reklám a podobne. Používateľa by po prihlásení mala privítať úvodná domovská stránka s prehľadnými údajmi o systéme. Keďže sa v systéme vyskytujú dva druhy užívateľov, kde jeden má na starosti správu celého systému a druhý len správu lokalít, ktoré mu boli pridelené je nutné zadať rolu administrátora a správcu. Vzhľadom na rozličnosť ich úloh sa bude líšiť aj ich rozhranie.

2.3.1 Administrátorské rozhranie

Ako už bolo spomenuté správu celého systému má na starosti Administrátor. Ten by mal vedieť vkladať resp. importovať predmety do systému. Tento proces musí zahŕňať aj vkladanie väčšieho množstva dát naraz. Najčastejší spôsob takéhoto hromadného vkladania je pomocou csv súborov, ktorých štruktúra sa vopred zadefinuje. Predmety spadajú pod kategórie, ktorých vytváranie a správu má mať na starosti taktiež administrátor. Okrem iného by ako najvyšší správca mal byť schopný tie najdôležitejšie údaje a štatistiky systému zobrazit' v prehľadných štatistikách, prípadne si ich vygenerovať vo forme PDF súboru. Štatistiky by mali zahŕňať celkový počet predmetov, počet spravovaných lokalít či ukazovateľ všetkých užívateľov.

2.3.2 Správcovské rozhranie

Správcovia by mali mať možnosť priradiť akýkoľvek voľne dostupný predmet do lokality, ktorú spravuje alebo naopak ho z nej odstrániť. Pri vyradení predmetu má mať možnosť napísať doplňujúce údaje s dôvodom vyradenia. Každému správcovi má byť taktiež poskytnutý podrobnejší pohľad na svoje vlastné štatistiky a taktiež si má vedieť zobrazit' detail voľného, vyradeného či zmazaného predmetu.

3 Výber programovacieho jazyka

Ako vieme, programovacie jazyky slúžia na opis objektov a ich vlastností. Pri tomto výbere sme brali do úvahy, ktoré programovacie jazyky budú najviac vhodné na opis nášho systému. Zohľadňovali sme predovšetkým aktuálnosť daného programovacieho jazyka a schopnosť splniť požiadavky na systém. Keďže náš systém vyžaduje webové rozhranie, rozhodli sme sa pre jazyk PHP, ktorý má širokú históriu používania. S týmto jazykom sme mali dokonca aj predchádzajúce skúsenosti pri vývoji iných webových stránok, čo nás vo výbere utvrdilo. Navyše je PHP jazyk veľmi flexibilný a jednoduchý na používanie. Disponuje širokou dokumentáciou a veľkými komunitami používateľov na programátorských fórach. O vizuálnu stránku sme sa postarali značkovacím jazykom HTML. Ten sme doplnili o jazyk CSS, ktorý slúži na rozšírenie a formátovanie HTML elementov.

Na dosiahnutie toho aby bola naša stránka interaktívna sme použili skriptovací programovací jazyk Javascript.

3.1 PHP

Čo sa týka jazyka PHP, rozhodli sme sa pre verziu 7.4, ktorá sa aktuálne radí k najnovším stabilným verziám tohto jazyka. Oproti predošlým verziám poskytuje hneď niekoľko nových vylepšení a možností. Verzia 7.4 je prvou verziou jazyka PHP, ktorá podporuje typy atribútov.

```
1 <?php
2
3 class Uzivatel
4 {
5     public int $id;
6     public string $meno;
7 }
```

Obr. 3.1 Typová deklarácia v jazyku PHP

Tento príklad hovorí o tom, že atribúty triedy *Uzivatel* sú typovo viazané. Atribút *id* môže nadobúdať teda len číselné hodnoty a atribút *meno* len hodnoty typu *string*. Vďaka kompatibilitě s predchádzajúcimi verziami jazyka PHP je však stále možnosť použiť slabé

typovanie, pri ktorom by atribúty v tomto príklade neboli viazané na žiadnu hodnotu čo prináša programátorovi široké možnosti využitia.

```
1 <?php
2 $konstanta = 10;
3 $cisla = array_map(
4
5     fn($cisla) => $cisla * $konstanta, [1, 2, 3, 4]
6
7 );
8
9 // $cisla = array(10, 20, 30, 40);
10 ?>
```

Obr. 3.2 Arrow funkcia v jazyku PHP

Ďalšou novinkou tejto verzie sú arrow funkcie, ktoré sú najčastejšie využívané pri anonymných funkciách. Poskytujú skrátenú syntax pre zapísanie implicitných funkcií. Príklad zobrazuje použitie funkcie `array_map`, ktorá aplikuje spätné volania na prvky poľa. Tie sú vynásobené vopred definovanou konštantou.

3.2 Javascript

Jazyk Javascript je skriptovacím jazykom, ktorý sme si vybrali z dôvodu aby sme vytvorili interaktívnu stránku. Využíva zabudované objekty a objekty prehliadača čo prináša programátorovi mnohé možnosti jeho použitia. To, že sa zdrojový kód písaný v tomto jazyku vykonáva až v prehliadači, umožňuje napríklad zmenu obsahu bez nutnosti znova načítania stránky.

Čo sa týka nášho systému, Javascript bol skvelým pomocníkom pri tvorbe farebných animácií, pohyblivého menu či vykresľovaní štatistík vo forme koláčového alebo lineárneho grafu.

4 Použité technológie

Výsledný systém sa skladá z viacerých celkov. Pri návrhu sme zohľadňovali aj zapojenie externých knižníc, či iných pomocných nástrojov, ktoré by nám pomohli zjednodušiť celkový vývoj. Snažili sme sa vybrať tie, ktoré sú stabilné aktuálne a najviac používané.

4.1 Virtualizácia

Existuje viacero techník virtualizácie. My sme si v našom prípade zvolili OS-level virtualizáciu, v ktorej oddeľujeme systémové zdroje.

4.1.1 Docker

Docker sa radí k nástrojom, ktoré sprostredkovávajú virtualizáciu. Jeho obrovskou výhodou je multiplatformová nezávislosť. Okrem iného ponúka jednoduché ovládanie pre správu kontajnerov. Kontajnery indikujú nádoby na prepravu tovaru. Znamená to teda, že kontajnery zapuzdria aplikáciu tak aby sa dala jednoducho prepraviť do iných systémov. Ak však chceme takýchto kontajnerov vytvoriť viacero naraz je vhodné použiť Docker Compose.

Nástroj Docker Compose nám dáva možnosť pomocou jedného príkazu spustiť viacero pripravených kontajnerov. Dá sa tak spraviť vytvorením yaml súboru, ktorý má svoju konkrétnu štruktúru a v ktorom bližšie špecifikujeme požiadavky, ktoré chceme zapuzdriť.

V priebehu krátkeho času je vďaka tomuto nástroju možné vytvoriť komplexnú infraštruktúru kontajnerov, ktoré môžeme jednoducho spravovať. Každý kontajner pritom zabezpečuje samostatné oddelenie pre chod aplikácie.

Ako teda už bolo spomenuté k tým najväčším výhodám patrí jednoduché zdieľanie pripravených kontajnerov medzi vývojármi. Môžeme ich taktiež nasadiť do cloudu a následne škálovať podľa potreby.

V našom projekte sme taktiež použili tento spôsob virtualizácie. Kontajnery tak zaobalujú napríklad databázu či webový server. Prípadné nasadenie výslednej aplikácie do prevádzky sa vďaka tejto virtualizácii výrazne zjednodušuje.

4.2 MySQL databáza

Čo sa ukladania dát týka, rozhodli sme sa pre MySQL databázu. V rámci fungovania nášho systému vzniká veľké množstvo prepojení a vzťahov. Každý užívateľ má spojenie so svojím účtom, každý predmet je viazaný na kategóriu, priradenú lokalitu a podobne. MySQL je relačná databáza. V našom systéme sa vyskytuje mnoho vzťahov medzi entitami a práve MySQL databáza ich dokáže spracovať. Dátovému modelu sa budeme bližšie venovať v kapitole 6.

4.2.1 MySQL Workbench

Nástroj MySQL Workbench nám poskytol skvelé prostredie na testovanie funkčnosti nášho dátového modelu. Vďaka funkcií reverzného inžinierstva, ktorú softvér ponúka sme mali možnosť vygenerovať vizuálny dátový model. Získali sme tak externý pohľad na systém.

4.3 Webový server Nginx

Webový server zvláda obsluhu viacerých návštevníkov stránky. Stará sa o predávanie súborov nášmu prehliadaču. Napríklad keď prehliadač pošle požiadavku na načítanie webovej stránky, webový server túto požiadavku obsluží a vráti všetky požadované súbory (obrázky, HTML súbory...) späť prehliadaču pomocou protokolu HTTP. Funguje ako prostredník medzi serverom a našim zariadením. Aj keď sa nazývajú ako webové servery, v skutočnosti ide o software bežiaci na fyzickom alebo virtuálnom serveri. Takýto webový server musí zvládnuť obslužiť niekoľko naraz pripojených používateľov, ktorí sa dopytujú na súbory z fyzického servera. (Nemeček, 2019)

4.4 Ajax volania

Ajax je technika používaná pri komunikácii s webovým serverom. Zo servera si vyžiada údaje, ktoré následne pomocou Javascriptu zobrazí na stránke. Výhodou týchto volaní je, že nie je nutné stránku stále načítavať, ale potrebné údaje sa zobrazia dynamicky. Ajax volania poskytujú užívateľovi lepší pocit pri návšteve stránky.

V našom prípade sme túto techniku využili najmä pri tvorbe filtrov a tabuliek, o ktorých si povieme neskôr pri implementácii.

4.5 PHP composer

Je jedným z najdôležitejších nástrojov, ktoré uľahčili vývoj nášho systému. Jedná sa o pomôcku na správu závislostí použitých v projekte. Definície všetkých závislostí sa nachádzajú v priečinku `composer.json`, v ktorom sa nachádzajú konkrétne verzie všetkých závislostí a knižníc použitých v projekte. Vďaka tomuto správcovi nedochádza ku konfliktom závislostí, ktoré nemajú kompatibilné verzie.

4.6 Twig

Twig je šablónovací systém, ktorý vychádza z použitia frameworku Symfony. Vďaka tomuto princípu sme zamedzili použitiu PHP kódu v tele HTML čím sa zvyšuje bezpečnosť stránky.

4.7 Doctrine

Doctrine zaisťuje mapovanie objektov na relačnú databázu čo v praxi znamená, že vďaka vopred vytvoreným objektom dokáže vytvoriť databázu s tabuľkami. Každému atribútu je priradený stĺpec v tabuľke. Použitie tejto technológie výrazne urýchlilo vytvorenie dátového modelu systému.

4.8 Git

Nástroj na správu verzií Git sme používali najmä pri vytváraní záloh. Jednotlivé verzie fungujú ako záchytné body a ponúkajú pohľad na celý priebeh vývoja s možnosťou vrátiť sa do ktorejkoľvek zálohovanej verzie.

4.9 Chart.js

Je to bezplatná knižnica a slúži na grafickú vizualizáciu údajov. Výber knižnice bol založený na základe toho, že ide o javascriptovú knižnicu, ktorej použitie je veľmi jednoduché. V našom prípade sme ju použili pri zobrazovaní štatistiky predmetov na domovskej stránke a vizualizácií cenového vývoja v detaile predmetu. O tomto využití bližšie hovoríme v kapitole 8. Okrem koláčového a lineárneho grafu, ktoré sme použili nám táto knižnica ponúka množstvo iných spôsobov grafickej vizualizácie.

Inicializácia grafov funguje prostredníctvom elementu canvas umiestnenom v tele HTML stránky. Takýto element možno označiť identifikátorom slúžiacim na lokalizáciu v javascriptovom kóde. Po lokalizácii a vytiahnutí kontextu elementu canvas metódou *getElementById()* a *getContext()* môžeme vytvoriť graf, ktorému nastavíme typ, naplníme ho údajmi a poskytneme mu nastavenia. Tie zahŕňajú veľkosť plátna, úpravu farieb grafu, nastavenia x-ovej a y-ovej osi a mnoho iných.

```
11 var ctx = document.getElementById( elementId: 'history_prices_line_chart').getContext('2d');
12
13 let chart = new Chart(ctx, {
14   type: 'line',
15   data: {
16     datasets: [{
17       label: 'Price history',
18       data: parsedHistoryPrices,
19       backgroundColor: [
20         'rgba(153, 102, 255, 0.5)',
21       ],
22       borderColor: [
23         'rgba(153, 102, 255, 1)',
24       ],
25       borderWidth : 1
26     }],
27     labels: parsedHistoryPrices.map((price, index) => index)
28   },
29   options: {
30     scales: {
31       yAxes: [{
32         ticks: {
33           suggestedMin: 50,
34           suggestedMax: 100
35         }
36       }]
37     }
38   }
39 }
```

Obr. 4.1 Inicializácia lineárneho grafu

Na uvedenom príklade je zobrazená inicializácia lineárneho grafu pre vývoj ceny predmetu. Graf vytvorený pomocou knižnice Chart.js prijíma do konštruktora kontext elementu canvas a nastavenia obsahujúce informácie pre inicializáciu.

5 Architektúra aplikácie

Pri výbere vhodnej architektúry pre náš systém sme mali na výber z viacerých možností. Ako už bolo spomenuté pri vývoji sme chceli použiť tie najnovšie technológie a princípy, ktoré sa aktuálne používajú. Jedným zo štandardov a najčastejšie využívaných architektúr vo veľkých spoločnostiach je MVC architektúra.

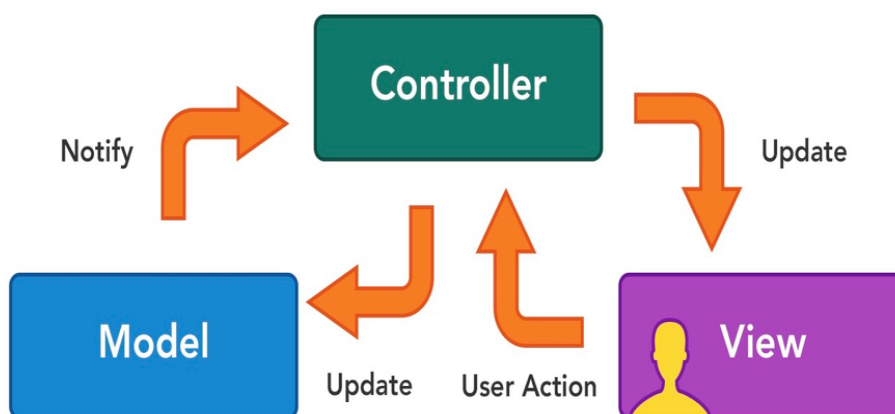
5.1 MVC architektúra

Model-View-Controller alebo skratene MVC je návrhový vzor, ktorý rozdeľuje aplikáciu do troch logických celkov. Každá táto časť je navrhnutá tak aby ovládala samostatný aspekt aplikácie.

Model v sebe obsahuje všetku logiku vykonávanú pri práci s databázou.

View sa používa pri vykresľovaní rozhrania aplikácie, s ktorou prichádza do styku koncový užívateľ. Ide teda o vizuálnu reprezentáciu spracovanej požiadavky.

Controller je časť architektúry, ktorá sa stará o spracovanie všetkých požiadaviek od užívateľa. Je prostredníkom medzi celkami Model a View. Vykonáva všetku potrebnú logiku na pozadí. Vyberá vhodný model pre spracovanie dát a rozhoduje o tom ktorý View sa užívateľovi zobrazí po spracovaní požiadavky.



Obr. 5.1 MVC architektúra

5.2 Symfony

Symfony je veľmi populárnym PHP frameworkom čo sa týka vytvárania webových stránok a aplikácií. Výber tejto technológie a jej zakomponovanie do vývoja bolo ovplyvnené predchádzajúcim výberom návrhového vzoru. Symfony totižto vychádza z MVC architektúry.

6 Dátový model

Prvým krokom k vytvoreniu funkčného dátového modelu bolo definovať entity, ktoré sa v systéme budú nachádzať. Na základe požiadaviek, ktoré má systém spĺňať sme následne vytvorili dátový model pomocou nástroja MySQL Workbench.

6.1 Popis entít

V tabuľkách sú použité skratky a to konkrétne:

- PK – Primary key (primárny kľúč)
- FK – Foreign key (cudzí kľúč)

Názov atribútu	Popis atribútu
id (PK)	Jedinečné identifikačné číslo predmetu
parent_id (FK)	Referencia na komponent, pod ktorý predmet patrí
category_id (FK)	Referencia na kategóriu, do ktorej predmet patrí
location_id (FK)	Referencia na lokalitu, v ktorej sa predmet nachádza
name	Názov predmetu
date_create	Dátum kedy bol predmet pridaný do systému
price	Cena predmetu
state	Stav v akom sa predmet nachádza
discard_reason	Dôvod prečo bol predmet vyradený z užívania

Tabuľka 6.1 Popis entity – Item

Názov atribútu	Popis atribútu
Id (PK)	Jedinečné identifikačné číslo kategórie
parent_id (FK)	Referencia na hierarchicky nadradenú kategóriu
name	Názov kategórie

is_active	Informácia o tom, či je kategória stále aktívna
-----------	---

Tabuľka 6.2 Popis entity – Category

Názov atribútu	Popis atribútu
id (PF)	Jedinečné identifikačné číslo lokality
parent_id (FK)	Referencia na hierarchicky nadradenú lokalitu
name	Názov lokality
is_active	Informácia o tom, či je lokalita stále aktívna

Tabuľka 6.3 Popis entity – Location

Názov atribútu	Popis atribútu
id (PK)	Jedinečné identifikačné číslo užívateľa
date_create	Dátum kedy bol užívateľ zaregistrovaný do systému
name	Užívateľské meno
email	Užívateľský e-mail
roles	Užívateľská rola
password	Užívateľské heslo
is_active	Informácia o tom, či je užívateľské konto aktívne

Tabuľka 6.4 Popis entity – User

Názov atribútu	Popis atribútu
id (PK)	Jedinečné identifikačné číslo notifikácie
user_id (FK)	Referencia na užívateľa, ktorému patrí konkrétna notifikácia
type	Typ notifikácie
status	Stav, v ktorom sa notifikácia nachádza (prečítaná / neprečítaná)

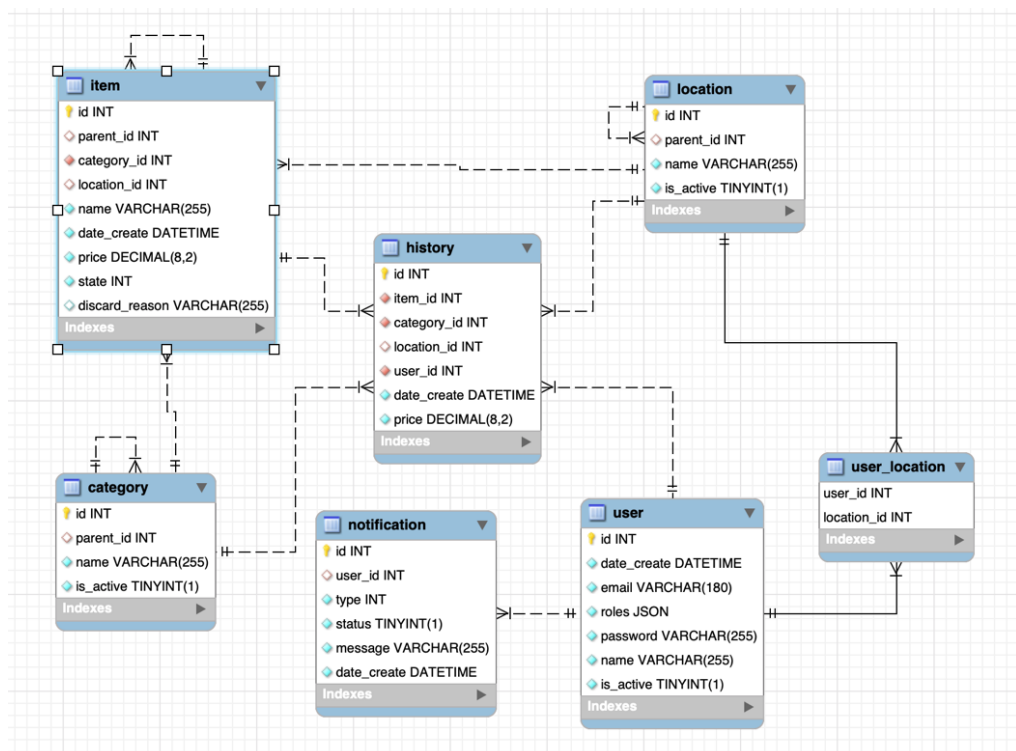
message	Správa s dôvodom notifikovania
date_create	Dátum kedy bola notifikácia vytvorená

Tabuľka 6.5 Popis entity – Notification

Názov atribútu	Popis atribútu
id (PK)	Jedinečné identifikačné číslo notifikácie
item_id (FK)	Referencia na predmet
category_id (FK)	Referencia na kategóriu
location_id (FK)	Referencia na lokalitu
user_id (FK)	Referencia na užívateľa
date_create	Dátum kedy bol záznam vytvorený
price	Cena, ktorú predmet v danom okamihu mal

Tabuľka 6.6 Popis entity – History

6.2 Popis dátového modelu



Obr. 6.1 Dátový model systému

User predstavuje užívateľa aplikácie. Jednou z požiadaviek bolo mať okrem správcov aj administrátora, ktorý celý systém spravuje. Práve preto bolo nutné do entity pridať atribút, na základe ktorého bude možné tieto dva typy užívateľov rozlíšiť a to užívateľskú rolu. Pri každom správcovi evidujeme aj dátum vytvorenia a stav či je aktívny.

Lokalite je priradená tabuľka **location** s atribútom `parent_id` ktorý ukazuje na nadradenú lokalitu. Tento vzťah nám tak dáva možnosť vytvoriť hierarchickú štruktúru organizácie, ktorú chceme spravovať. Správca s rolou administrátora dokáže priradiť existujúce lokality pod správu ľubovoľného užívateľa. To znamená, že viacero správcov môže spravovať viacero lokalít a naopak. Ako možno vidieť na obrázku túto funkčnosť zabezpečuje tabuľka **user_location**.

Hierarchickú štruktúru je potrebné vytvárať aj pri kategóriách (tabuľka **category**) alebo pri predmetoch (tabuľka **item**), čím sme zabezpečili požiadavku na to aby mohol mať každý predmet priradené komponenty.

Tabuľka **history** je viazaná na všetky ostatné tabuľky. Prostredníctvom tejto entity dokážeme uchovávať záznamy o každej zmene, ktorá bola vykonaná pri predmete.

Notification tabuľka má za úlohu uchovávať záznamy, ktoré sa používajú pri informovaní užívateľa o podstatných udalostiach, ktoré sa v systéme udiali. Môže existovať viacero typov notifikácií. Každá obsahuje správu pre užívateľa a informáciu o tom, či už bola prečítaná alebo nie.

7 Štruktúra projektu

Štruktúra projektu sa vo veľkej miere odvíjala od použitej architektúry. Projekt obsahuje niekoľko dôležitých celkov, ktoré si opíšeme.

7.1 Balík Entity

V balíčku Entity sa nachádzajú PHP triedy, ktoré sme vytvárali pomocou nástroja Doctrine ORM. Ten ich následne mapuje na databázové entity. Nachádzajú sa tu triedy User, Item, Location, Category, History a Notification.

7.1.1 Mapovanie entít

Ako už bolo spomenuté mapovanie na databázové entity funguje vďaka nástroju Doctrine. Triedu možno vytvoriť pomocou príkazu *make:entity*. Tento príkaz nás prevedie krátkym formulárom, v ktorom možno zodpovedať informácie o entite a jej atribútoch. Výsledkom je trieda, ktorá obsahuje nami zadané informácie vo formulári.

```
12 use Doctrine\ORM\Mapping as ORM;
13
14 /**
15  * @ORM\Entity(repositoryClass=ItemRepository::class)
16  */
17 class Item
18 {
19     /**
20      * @ORM\Id
21      * @ORM\GeneratedValue
22      * @ORM\Column(type="integer")
23      */
24     private $id;
25 }
```

Obr. 7.1 Príklad Doctrine ORM mapovania

Na príklade možno vidieť triedu Item, ktorá využíva Doctrine ORM mapovanie. Veľkú úlohu tu zohrávajú anotácie. Práve vďaka nim Doctrine ORM vie, že sa jedná o entitu a že atribút *id* má byť typu integer.

Takto vytvorené a zmapované entity možno následne previesť do reálnych databázových tabuliek a to vytvorením migrácie príkazom *make:migration* a následným vykonaním migrácie pomocou príkazu *doctrine:migrations:migrate*. Všetky takto vytvorené migrácie

sa ukladajú do samostatnej tabuľky, vďaka čomu máme možnosť vrátiť sa ku staršej verzii dátového modelu.

7.2 Balík Repository

Doctrine sa postará aby ku každej vytvorenej entite prislúchala aj Repository trieda. Tá slúži na vytváranie vlastných query príkazov pomocou metódy *createQueryBuilder*, ktoré môžeme zabaliť do metód.

```
5 use App\Entity\Item;
6 use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
7 use Doctrine\Persistence\ManagerRegistry;
8
9 /**
10  * @method Item|null find($id, $lockMode = null, $lockVersion = null)
11  * @method Item|null findOneBy(array $criteria, array $orderBy = null)
12  * @method Item[] findAll()
13  * @method Item[] findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
14  */
15 class ItemRepository extends ServiceEntityRepository
16 {
17     public function __construct(ManagerRegistry $registry)
18     {
19         parent::__construct($registry, Item::class);
20     }
21 }
```

Obr. 7.2 Príklad Repository triedy

Každá Repository trieda už obsahuje niekoľko štandardných volaní uvedených nad definíciou triedy pomocou anotácií. Jedná sa o metódy:

- `find()` – vráti jeden záznam z tabuľky na základe identifikačného čísla
- `findOneBy()` – vráti jeden záznam z tabuľky, ktorý spĺňa zadané kritéria
- `findAll()` – vráti všetky záznamy z tabuľky
- `findBy()` – vráti všetky záznamy z tabuľky, ktoré spĺňajú zadané kritéria

7.3 Balík Controller

V tomto balíčku sa nachádzajú všetky Controller triedy, ktoré sa starajú o spracovávanie žiadostí (Requests) od užívateľa. Každý controller môže mať viacero metód, pričom každá z nich spracováva žiadosti na inej *url api* adrese. Tieto triedy po prečítaní žiadosti vytvoria odpoveď (Response) a to zväčša vo forme HTML stránky, súboru na stiahnutie alebo presmerovania na inú stránku.


```

1 <?php
2
3 class ExampleController extends AbstractController
4 {
5     /**
6      * @Route("/example", name="example")
7      * @param Request $request
8      * @return Response
9      */
10    public function index(Request $request): Response
11    {
12        $exampleForm = $this->createForm(Example::class);
13        $exampleForm->handleRequest($request);
14
15        if ($exampleForm->isSubmitted() && $exampleForm->isValid()) {
16            //do something
17        }
18
19        return $this->render('page/example.html.twig');
20    }

```

Obr. 7.3 Príklad Controller triedy

Príklad ktorý sme uviedli spracováva žiadosti na podstránke *example*. Po prečítaní žiadosti *handleRequest()* a vykonaní potrebnej logiky vracia odpoveď pomocou metódy *render()* vo forme HTML stránky nachádzajúcej sa v priečinku *page*.

7.4 Balík Form

Ako už vypovedá samotný názov v tomto balíčku sa nachádzajú všetky triedy obsahujúce formuláre, ktoré užívateľ na stránke potvrdzuje. Tvorba HTML formulárov môže byť často komplikovaná a zdĺhavá. Taktiež pri nich vzniká veľká duplicita kódu a ich kontrola a validácia sa stáva komplikovaná. Aby sme tomuto predišli využili sme funkciu na vytváranie formulárov, ktorú nám ponúka samotný framework Symfony. Ide o použitie Form tried. Každá Form trieda prijíma do konštruktora *FormBuilderInterface*, vďaka ktorému máme možnosť vytvoriť ľubovoľnú šablónu pre formulár.

```

1 <?php
2
3 class AddLocation extends AbstractType
4 {
5     public function buildForm(FormBuilderInterface $builder)
6     {
7
8         $builder
9             ->add('name', TextType::class)
10            ->add('parent', EntityType::class, [
11                'class' => Location::class,
12                'choice_label' => 'name',
13                'required' => false,
14            ])
15            ->add('submitButton', SubmitType::class, [
16                'label' => 'Add',
17                'attr' => ['class' => 'btn btn-success btn-xs']
18            ]);
19    }
20
21 }

```

Obr. 7.4 Príklad Form triedy

7.5 Balík Template

Tu sa nachádza vizuálna časť nášho systému. Súbor s názvom *base.html.twig* obsahuje základné rozloženie stránky ako bočné menu (side navigation menu) a horný panel (navbar).

V pod priečinku page nájdeme všetky HTML stránky, ktoré sa zobrazujú užívateľovi a do pod priečinku layout sme umiestnili šablóny pre tlačidlá a modal okná, ktoré sa na stránke vyskytujú. Použitie týchto pripravených tlačidiel zamedzuje duplicite kódu.

```
1 <span data-toggle="tooltip" data-placement="top" title="Detail">
2   <a href="{{ url }}" class="btn btn-info mr-1 btn-sm">
3     <i class="c-icon fas fa-eye"></i>
4   </a>
5 </span>
```

Obr. 7.5 Šablóna pre tlačidlo Detail

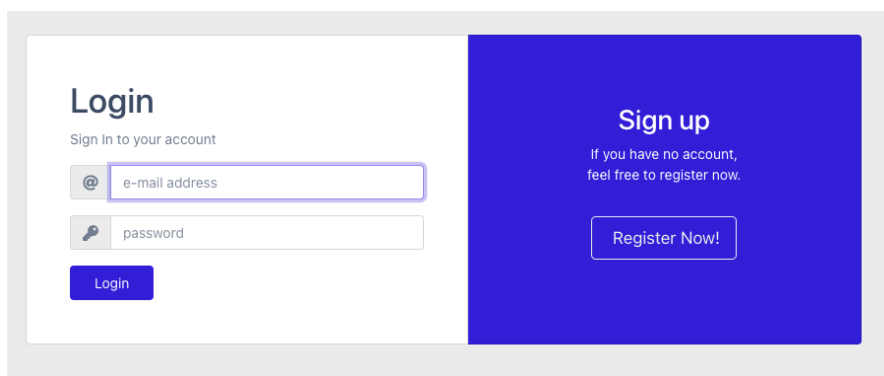
Na obrázku 7.5 je zobrazená šablóna pre tlačidlo detailu. Túto šablónu môžeme jednoducho použiť na každej HTML stránke pomocou twig príkazu *include*. Takýto príkaz však musí byť v dvojitéch kučeravých zátvorkách.

8 Frontend

Vizuálna stránka je dôležitým aspektom pri vývoji každého informačného systému. Prostredie stránky by malo byť intuitívne a jednoduché na použitie. O to sme sa pokúsili aj pri spracovaní nášho rozhrania. V nasledujúcich kapitolách si povieme o použitých grafických knižniciach a ukážeme si rozhranie systému.

8.1 Prihlásenie

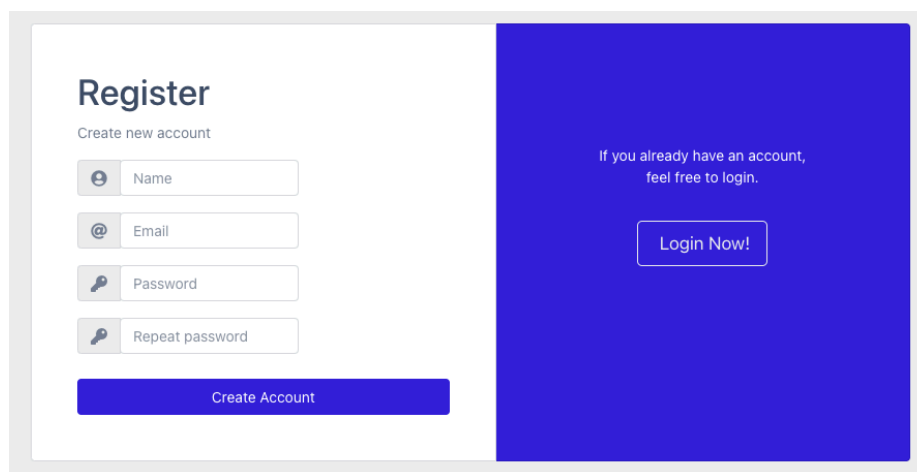
Pri návšteve stránky sa užívateľovi zobrazí formulár na prihlásenie. Po potvrdení prihlasovacích údajov, ktoré zahŕňajú mailovú adresu a heslo nás server presmeruje na domovskú stránku systému. V prípade, že užívateľ ešte nie je zaregistrovaný, má možnosť registrácie kliknutím na tlačidlo Register Now.



Obr. 8.1 Formulár na prihlásenie

8.2 Registrácia

Registračný formulár ma podobný tvar ako prihlasovací. Po správnom vyplnení potrebných údajov registrácie teda užívateľského mena, mailovej adresy a hesla nám server vytvorí systémový účet, automaticky nás prihlási a presmeruje na domovskú stránku. V prípade, že užívateľské konto už máme vytvorené, máme možnosť prekliknúť sa k prihlasovaciemu formuláru tlačidlom *Login Now*.



The image shows a web registration form. On the left, there's a white box with the title 'Register' and the subtitle 'Create new account'. Below this are four input fields: 'Name' (with a person icon), 'Email' (with an '@' icon), 'Password' (with a key icon), and 'Repeat password' (with a key icon). At the bottom of this box is a blue button labeled 'Create Account'. To the right of this box is a large blue sidebar. Inside the sidebar, there's white text that says 'If you already have an account, feel free to login.' and a white button labeled 'Login Now!'.

Obr. 8.2 Formulár na registráciu

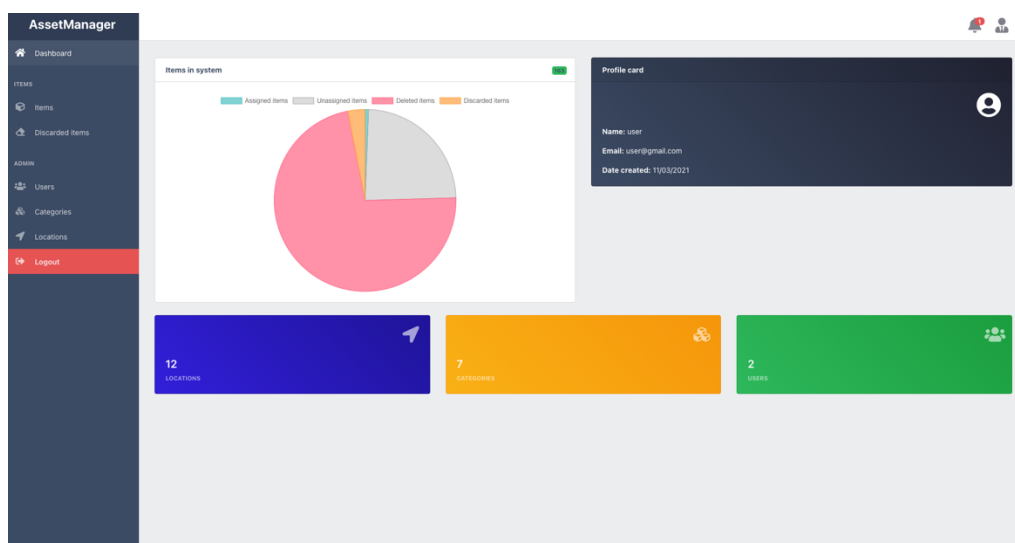
8.3 Domovská stránka

Domovská stránka ponúka všeobecný prehľad a poskytuje základné informácie týkajúce sa systému. V ľavej časti sa nachádza spomínané bočné menu, ktoré slúži na prechod medzi podstránkami systému. V hornej časti je takzvaný navbar, kde je umiestnené tlačidlo s notifikáciami.

Samotný obsah domovskej stránky je tvorený prehľadnými štatistikami s údajmi o predmetoch v systéme, počte spravovaných lokalít, existujúcich kategórií či množstve registrovaných užívateľov. Profilová karta umiestnená v pravom hornom rohu zobrazuje základné údaje prihláseného užívateľa. Koláčový graf znázorňuje rozdelenie predmetov podľa ich stavu.

- Assigned items – vyjadruje počet predmetov, ktoré má prihlásený užívateľ pod správou
- Unassigned items – vyjadruje počet predmetov, ktoré nemá prihlásený užívateľ pod správou a zároveň sú voľne dostupné
- Discarded items – vyjadruje počet predmetov, ktoré boli vyradené zo systému (napríklad z dôvodu poškodenia)
- Deleted items – vyjadruje počet predmetov, ktoré boli odstránené zo systému

Pri tvorbe tohto grafu sme využili grafickú knižnicu Chart.js, o ktorej sme hovorili v kapitole 4.9.



Obr. 8.3 Domovská stránka

8.4 Items

Podstatou tejto stránky je práca s predmetmi v systéme. Telo tejto stránky sa skladá z troch záložiek. Každá záložka má inú funkciu a slúžia pre lepšiu orientáciu na stránke Items.

8.4.1 Unassigned items

Prvá záložka má názov Unassigned items. Vo vrchnej časti je umiestnené tlačidlo odkazujúce na stránku s pridávaním predmetov do systému. Pod ním sa nachádza filter s tabuľkou nezaradených predmetov. Tabuľka je zložená zo stĺpcov pre identifikačné číslo predmetu, názov, lokalitu v ktorej je predmet umiestnený, kategóriu, cenu a dátum zaradenia do systému. V poslednom stĺpci tabuľky sa nachádzajú tlačidlá s funkciami.

Filter nad tabuľkou slúži na zobrazenie predmetov spĺňajúcich zadané kritéria. V tabuľke je taktiež možnosť predmety zoradiť podľa niektorých stĺpcov. Množstvo predmetov, ktoré sa v tabuľke zobrazia je možné nastaviť vľavo nad ňou.

Zobrazenie tlačidiel závisí od role prihláseného užívateľa, pretože administrátor má na rozdiel od bežného správcu možnosť predmety editovať či úplne zo systému vymazať.

Tlačidlo *Assign* slúži na priradenie predmetu do lokality. Po kliknutí sa nám zobrazí okno s možnosťou výberu lokality, do ktorej chceme predmet priradiť. Zobrazenie výberu lokalít závisí od množstva spravovaných lokalít. V prípade, že správca nemá pod správou žiadnu lokalitu nemá možnosť predmet nikam priradiť.

Tlačidlo *Detail* presmeruje užívateľa na podstránku s podrobnými informáciami o predmete. Tejto podstránke sa venujeme v kapitole 8.10.

Tlačidlo *Edit* taktiež užívateľa presmeruje na podstránku, v ktorej možno upraviť informácie o predmete. Toto tlačidlo je sprístupnené iba administrátorovi.

Tlačidlo *Delete* zobrazí potvrdzovací formulár s otázkou či si skutočne prajeme predmet natrvalo odstrániť zo systému. Táto funkčnosť je sprístupnená iba administrátorovi.

Unassigned items

Filter

Name: Category:

from: to:

Show 25 entries

ID	Name	Location	Category	Path	Price	Timestamp	Actions
2	Stolička	empty location	Nábytok	Stolička	80.00	13/04/2021	
4	Kávovar	empty location	Spotrebiče	Kávovar	99.00	13/04/2021	
5	Reprodukory	empty location	Elektronika	Reprodukory	60.00	13/04/2021	
6	Macbook	empty location	Počítače	Macbook	899.00	13/04/2021	

Showing 1 to 4 of 4 entries (filtered from 6 total entries)

First Previous **1** Next Last

Obr. 8.4 Unassigned items

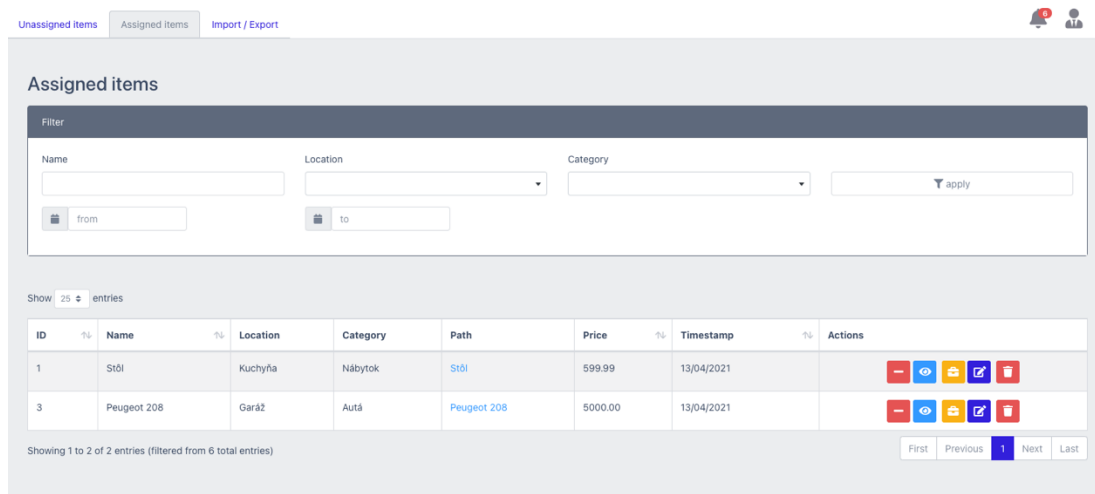
8.4.2 Assigned items

Druhá záložka Assigned items sa skladá taktiež z tabuľky s filtrom. Hlavným rozdielom sú predmety, ktoré sa v tejto tabuľke zobrazujú. Jedná sa o predmety nachádzajúce sa v lokalitách, ktoré sú pod správou užívateľa. To znamená, že ak má užívateľ napríklad pod správou lokalitu A, budú sa v tejto tabuľke zobrazovať predmety patriace len do tejto lokality. Ďalším podstatným rozdielom sú tlačidlá s akciami umiestnené v tabuľke. Okrem už opísaných tlačidiel *Detail*, *Edit* a *Delete* sa tu nachádza ešte *Unassign* a *Discard*.

Tlačidlo *Unassign* slúži na odstránenie predmetu z lokality. Po kliknutí sa nám zobrazí potvrdzovací formulár.

Tlačidlo *Discard* slúži na vyradenie predmetu zo systému. Po kliknutí sa nám zobrazí potvrdzovací formulár s možnosťou napísať dôvod vyradenia. Po vyradení sa predmetu

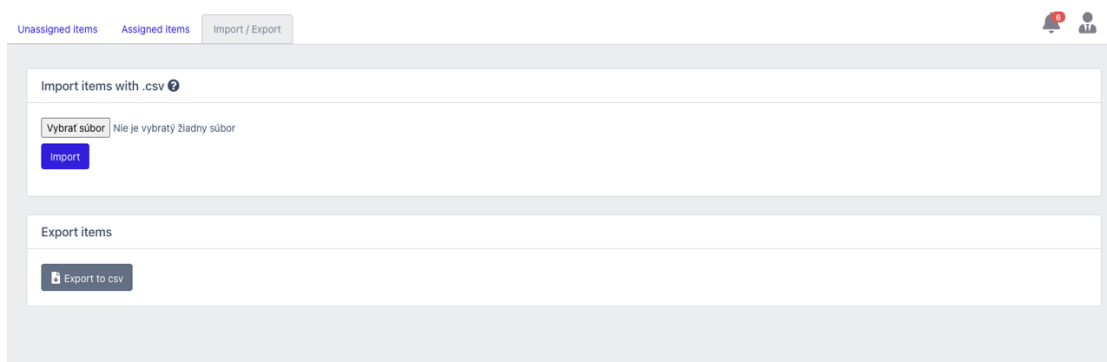
zmení stav z *Active* (aktívny) na *Discarded* (vyrađený) a nebude ďalej možné ho priradiť do lokality.



Obr. 8.5 Assigned items

8.4.3 Import / Export

Tretia záložka Import / Export poskytuje možnosť importovať a exportovať predmety v rámci systému.



Obr. 8.6 Import / Export items

Importovanie predmetov je možné pomocou .csv súboru, ktorého štruktúra je popísaná vo vysvetlivke v hlavičke import formulára.

Export vygeneruje .csv súbor obsahujúci všetky aktívne predmety, ktoré sa v systéme aktuálne nachádzajú.

8.5 Discarded Items

Sekcia *Discarded items* obsahuje tabuľku predmetov, ktoré boli vyradené zo systému. Z tabuľky je možné prekliknúť sa do detailu vyradeného predmetu. V prípade administrátora je vyradený predmet možno zo systému úplne odstrániť.

8.6 Users

Users predstavuje podstránku pre prácu užívateľov. Je prístupná administrátorovi a obsahuje dve záložky.

Pod prvou záložkou sa nachádza tabuľka s možnosťou filtrovania aktívnych užívateľov. Tlačidlo *Attach*, ktoré sa v tabuľke nachádza presmeruje administrátora na podstránku, kde je možné užívateľovi priradiť alebo naopak odstrániť zo správy existujúce lokality.

Pod druhou záložkou *Export* má administrátor možnosť vyexportovať údaje týkajúce sa užívateľov a im priradených lokalít. Exportovaný súbor je možné stiahnuť vo formáte PDF.

8.7 Categories a Locations

Obsah týchto dvoch podstránok má podobnú funkciu a to zobrazit' aktuálne vytvorené kategórie a lokality, ktoré v systéme existujú.

Obe obsahujú tabuľku s filtrom a tlačidlo odkazujúce na pridávací formulár.

Locations

Filter

Name

Show 25 entries

ID	Name	Path	Actions
1	Dom	Dom	<input type="button" value="↓"/> <input type="button" value="✎"/> <input type="button" value="✖"/>
2	Kuchyňa	Dom > Kuchyňa	<input type="button" value="↓"/> <input type="button" value="✎"/> <input type="button" value="✖"/>
3	Obývačka	Dom > Obývačka	<input type="button" value="↓"/> <input type="button" value="✎"/> <input type="button" value="✖"/>
4	Izba	Dom > Izba	<input type="button" value="↓"/> <input type="button" value="✎"/> <input type="button" value="✖"/>
5	Garáž	Garáž	<input type="button" value="↓"/> <input type="button" value="✎"/> <input type="button" value="✖"/>

Showing 1 to 5 of 5 entries

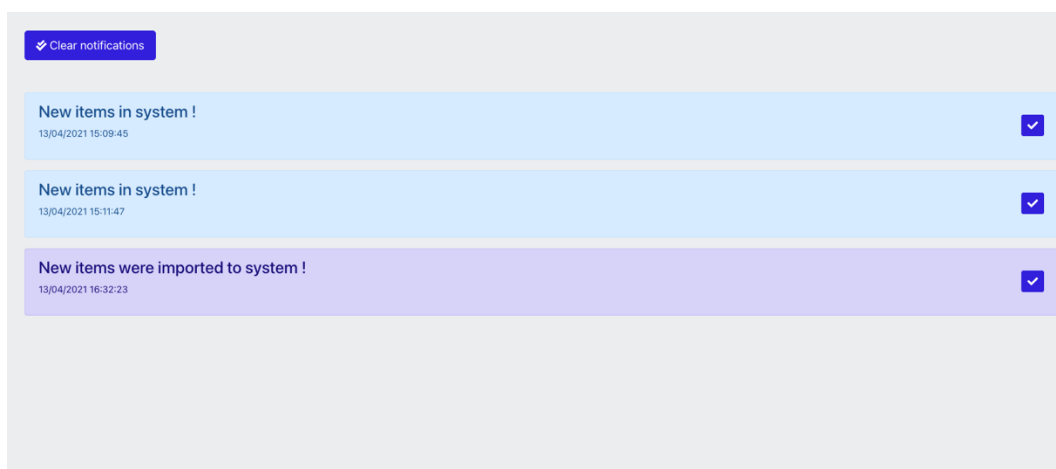
First Previous 1 Next Last

Obr. 8.7 Podstránka Locations

8.8 Notifications

Na podstránku notifications sa užívateľ dostane kliknutím na zvonček nachádzajúci sa v hornom menu. Tento zvonček má zároveň za úlohu informovať prihláseného užívateľa o nových notifikáciách v systéme. V prípade, že užívateľ má neprečítanú notifikáciu, pri zvončeku sa zobrazí číslo s počtom takýchto notifikácií.

Samotná podstránka sa skladá zo zoznamu notifikácií, ktoré obsahujú správu týkajúcu sa zmeny v systéme a dátum kedy k takejto zmene nastalo. Užívateľ má následne možnosť označiť notifikáciu za prečítanú čím sa z tohto zoznamu odstráni.



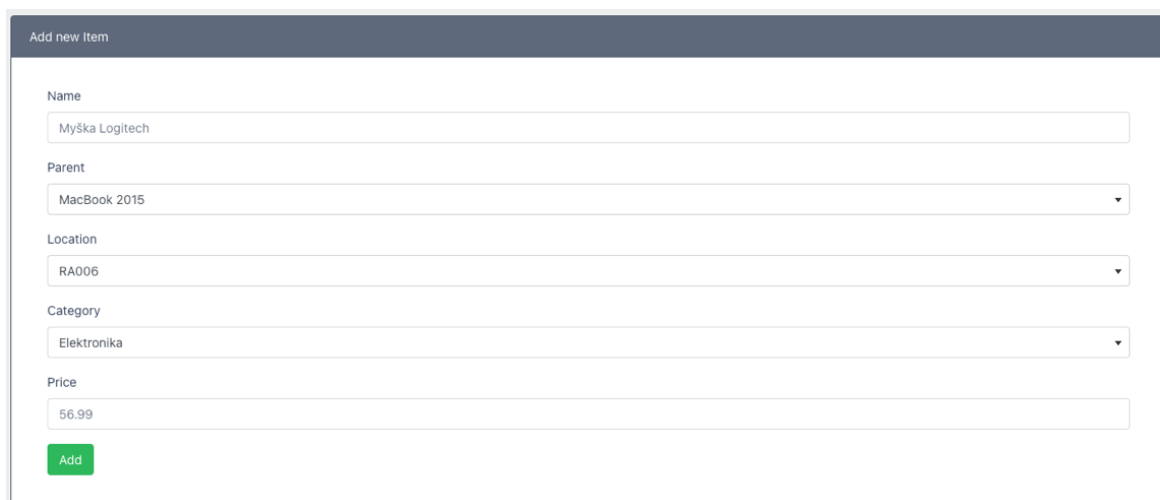
Obr. 8.8 Notifications podstránka

8.9 Formuláre

V našom systéme sa nachádza niekoľko formulárov. Ide najmä o formuláre na pridávanie a editovanie. Ako sme už písali v predchádzajúcich kapitolách k podstránkam, na ktorých sa formuláre nachádzajú sa vieme dostať z podstránky *Items*, *Categories* a *Locations* kliknutím na tlačidlo *Add new*, alebo *Edit*.

8.9.1 Add form

Formulár na pridávanie má jednoduchú formu aby bolo zrejmé čo sa od užívateľa vyžaduje. Napríklad pri pridávaní predmetu do systému má užívateľ možnosť vyplniť názov predmetu, komponent ktorému je predmet podradený, lokalitu v ktorej sa nachádza a podobne.

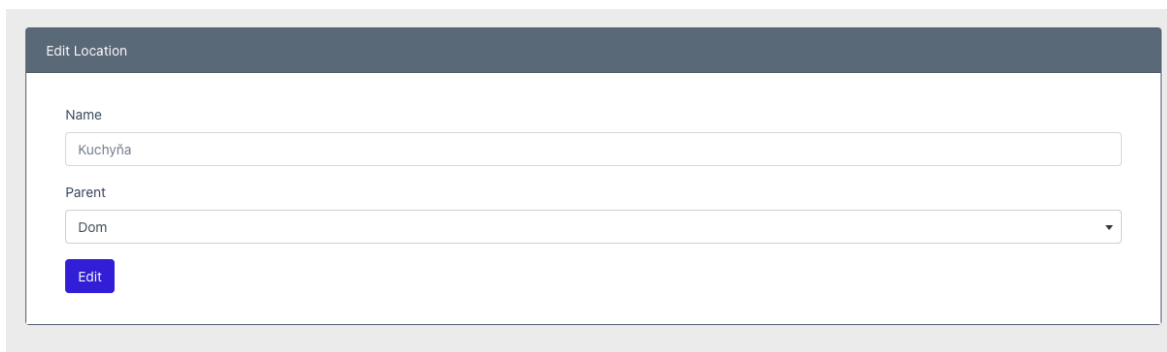


Obr. 8.9 Formulár na pridávanie predmetu

Pri vytváraní kategórie a lokality sa vo formulári nachádza len názov pre vytváraný subjekt a možnosť nastaviť mu nadradenú lokalitu / kategóriu.

8.9.2 Edit form

Formulár na úpravu existujúcich predmetov, lokalít a kategórií už obsahuje pred vyplnené údaje, ktoré možno zmeniť a následne potvrdiť tlačidlom *Edit*.

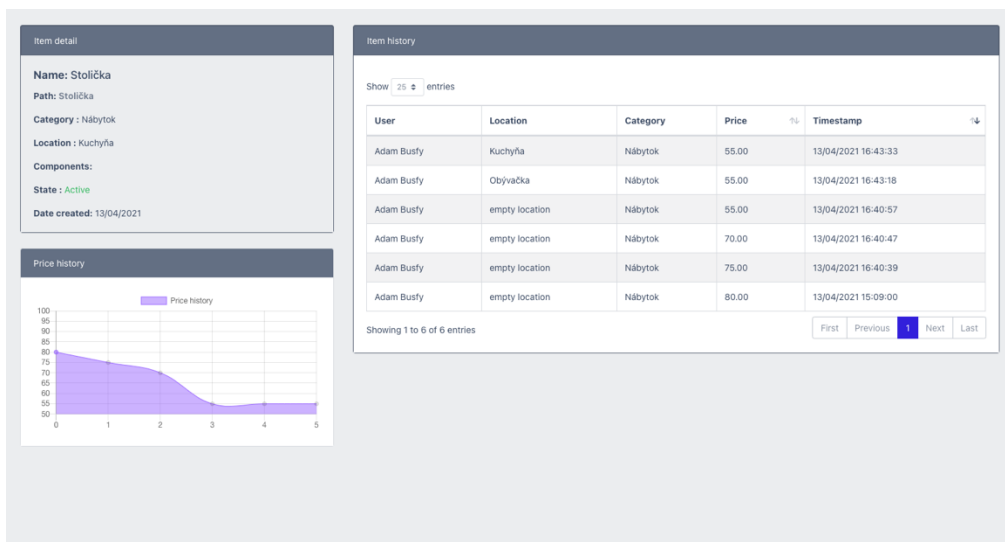


Obr. 8.10 Formulár na úpravu kategórie

8.10 Item Detail

Obsah podstránky *Item Detail* slúži na zobrazenie údajov o predmete. V ľavej časti sa nachádza tabuľka so všetkými informáciami o predmete. Okrem názvu, kategórie a lokality tu je zobrazený aj zoznam komponentov a aktuálny status predmetu. V prípade, že sa jedná o vyradený predmet, je možnosť zobrazit' si informáciu prečo bol zo systému vyradený.

Pod túto tabuľku sme umiestnili čiarový graf, zobrazujúci vývoj histórie ceny predmetu. V pravej časti tejto podstránky je tabuľka, v ktorej sú záznamy zachytávajúce vykonané zmeny týkajúce sa predmetu. Tabuľka zachytáva zmenu kategórie, lokality a ceny. Záznamy je možné zoradiť od najnovších či najstarších kliknutím na obojstrannú šípku v stĺpci *Timestamp*.



Obr. 8.11 Podstránka Item Detail

Záver

Cieľom práce bolo vytvoriť informačný systém pre evidenciu majetku s dôrazom na temporálny aspekt. V práci sme popísali a analyzovali súčasný stav existujúcich nástrojov a venovali sme sa aj analýze správy majetku v prostredí UNIZA.

Ďalej sme definovali požiadavky na systém a vysvetlili prečo sme si vybrali technológie použité pri vývoji. Súčasťou práce je taktiež architektúra aplikácie a navrhnutý dátový model.

Jednou z najťažších častí práce bolo okrem návrhu vytvoriť prehľadné užívateľské rozhranie, ktoré bude intuitívne a jednoduché na použitie. Veríme, že sa nám túto funkčnosť podarilo splniť.

Počas práce na tejto bakalárskej práci sme nadobudli mnoho poznatkov z oblasti webového vývoja. Zoznámili sme sa s novými technológiami a knižnicami ako napr. *Chart.js* či *FPDF generátor*.

Pri vývoji sme sa stretli s mnohými problémami či už pri návrhu alebo samotnej implementácii, no dokázali sme ich vyriešiť. Systém je navrhnutý tak aby dokázal zvládnuť aj väčší objem dát.

Aplikácia bola navrhnutá tak aby bola implementácia prípadných rozšírení čo možno najjednoduchšia. Takto vytvorený systém ponúka ešte niekoľko funkcií, ktoré by sa dali do systému zakomponovať. Niektorými z nich sú napríklad:

- možnosť nahrat' k predmetom aj obrázky
- rozšíriť zobrazované štatistiky o ďalšie grafy
- možnosť vopred upraviť tvar exportovaných dát
- pridať administrátorovi možnosť vytvoriť vlastnú notifikáciu

Zoznam použitej literatúry

1. **Bhanwar Gupta.** Introduction to AJAX: Bringing Interactivity & Intuitiveness into Web Applications. [online]. August 2010. <https://www.free-ebooks.net/computer-internet/AJAX-Introduction>.
2. PHP indtroduction. [online]. <https://www.w3schools.com/php>.
3. Javascript introduction. [online]. <https://www.w3schools.com/js>.
4. Databases and the Doctrine ORM. [online]. <https://symfony.com/doc/current/doctrine.html>
5. Symfony Documentation. [online]. <https://symfony.com/doc/current/index.html>
6. Chart.js. [online]. <https://www.chartjs.org/docs/latest/>
7. Twig Documentation. [online]. <https://twig.symfony.com/doc/3.x/>
8. Virtualizácia. [online]. <https://magazin.kpi.fei.tuke.sk/2019/02/docker-a-jeho-pouzitie-pri-kontajnerizacii/>

Citované diela

- Nemeček, A. (14. Január 2019). *Najpoužívanejšie webové servery – Apache vs. Nginx*. Dostupné na Internete: <https://www.websupport.sk/blog/2019/01/najpouzivanejsie-webove-servery-porovnanie-apache-vs-nginx/>

Zoznam príloh

Príloha A Obsah DVD

Prílohy

Príloha A: Obsah DVD

Priložené DVD obsahuje:

- Prácu v elektronickej podobe (formát PDF)
- Zdrojový kód práce