

Střední průmyslová škola a Vyšší odborná
škola, Písek,

Karla Čapka 402

397 11 Písek

Školní rok: 2024/2025

Obor vzdělání: 18-20-M/01 Informační technologie

Maturitní práce

System pro hlášení závad

Téma číslo: 25

Jméno žáka: Antonín Kožmín

Třída: B4.I

Vedoucí práce: Bc. Bartuška Martin, DiS.

Anotace

Tato maturitní práce se zaměřuje na vytvoření systému pro hlášení závad ve škole. Celé to poběží ve webovém rozhraní s přihlášením jednotlivých uživatelů a možností přidání závady. Dále se zde bude nacházet archiv pro správce, který bude moci upravovat jednotlivé závady podle potřeby. Práce zahrnuje emailovou komunikaci pro obě strany, a také propojení s inventarizačním systémem. Hlavním cílem bylo vytvořit přehledný, bezpečný a efektivní nástroj s intuitivním uživatelským rozhraním.

Klíčová slova: systém pro hlášení závad, PHP, MySQL, web, emailová komunikace, inventární čísla, uživatelské rozhraní

Annotation

This thesis focuses on creating a system for reporting faults in schools. The entire system will run on a web interface with individual user logins and the ability to submit a fault report. Additionally, there will be an archive for administrators, who can modify individual fault reports as needed. The work includes email communication for both parties, as well as integration with an inventory system. The main goal was to create a clear, secure, and efficient tool with an intuitive user interface.

Keywords: fault reporting system, PHP, MySQL, web, email communication, inventory numbers, user interface

Poděkování

Rád bych poděkoval Bc. Bartuška Martin, DiS. za rady, pozitivní přístup, pomoc a profesionální vedení při tvorbě mé dlouhodobé maturitní práce.

Licenční smlouva o podmínkách užití školního díla

ve smyslu zákona č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění (dále jen „Autorský zákon“), uzavřená mezi smluvními stranami:

1. Autor práce (dále jen „Autor“):

jméno a příjmení: Antonín Kožmín
bytem: Chrastiny 73, Dolní Novosedly 39701
datum narození: 4.11.2005

a

2. Nabyvatel

Střední průmyslová škola a Vyšší odborná škola, Písek, Karla Čapka 402
397 11 Písek, Karla Čapka 402 (dále jen „Nabyvatel“)
zastoupená ředitelem školy: Ing. Jíří Uhlík

Článek 1

Vymezení pojmů

1.1 **Školním dílem dle § 60 Autorského zákona** se pro účely této smlouvy rozumí dílo vytvořené žákem/studentem ke splnění školních nebo studijních povinností vyplývajících z jeho právního postavení ke škole.

1.2 **Licencí** se pro účely této smlouvy rozumí oprávnění k výkonu práva školní dílo užít v rozsahu a za podmínek dále stanovených.

Článek 2

Dílo

2.1 Předmětem této smlouvy je poskytnutí licence k užití školního díla - maturitní práce.

Název práce (dále jen „**Dílo**“): Systém pro hlášení závad
vedoucí práce: Bc. Bartuška Martin, DiS.
odevzdané Nabyvateli v elektronické formě dne 31.3.2025.

2.2 Autor prohlašuje, že:

- vytvořil Dílo, specifikované touto smlouvou, samostatnou vlastní tvůrčí činností;
- při zpracovávání Díla se sám nedostal do rozporu s Autorským zákonem a předpisy souvisejícími;
- Dílo je dílem původním;
- neposkytl třetí osobě výhradní oprávnění k užití Díla v rozsahu licence poskytnuté Nabyvateli dle této smlouvy před podpisem této smlouvy;
- je si vědom, že před zamýšleným poskytnutím výhradního oprávnění k užití Díla v rozsahu licence poskytnuté Nabyvateli dle této smlouvy třetí osobě, je povinen informovat tuto třetí osobu o skutečnosti, že již poskytl nevýhradní licenci k užití Díla Nabyvateli.

2.3 Dílo je chráněno jako Dílo dle Autorského zákona a dle zákona č. 89/2012 Sb., občanský zákoník (dále jen „**Občanský zákoník**“).

Článek 3

Poskytnutí licence

3.1 Licenční smlouvou Autor poskytuje Nabyvateli na dobu neurčitou oprávnění k výkonu práva Dílo užít pro účely výuky na Střední průmyslové škole a Vyšší odborné škole, Písek, Karla Čapka 402, a pro vnitřní potřebu této školy, ze které neplyne škole hospodářský výsledek.

3.2 Licence je poskytována pro celou dobu trvání autorských a majetkových práv k Dílu.

3.3 Autor poskytuje Nabyvateli oprávnění užít Dílo způsoby podle 3.1 neomezeně.

3.4 Autor poskytuje Nabyvateli oprávnění užít Dílo bezúplatně za splnění podmínky, že Nabyvatel nebude užívat Dílo za účelem dosažení zisku a nebude-li v budoucnu dohodnuto písemně jinak.

Článek 4 **Údaje o autorství**

4.1 Nabyvatel se zavazuje, že uvede údaje o autorství Autora dle Autorského zákona.

Článek 5 **Další ujednání o užití Díla**

5.1 Pokud to není v rozporu s oprávněnými zájmy Nabyvatele, licence je poskytována jako **nevýhradní**. Nabyvatel je oprávněn postoupit tuto licenci třetí osobě a udělovat podlicence za splnění podmínek uvedených v § 48 Autorského zákona.

5.2 Autor může své Dílo užít či poskytnout jinému licenci, není-li to v rozporu s oprávněnými zájmy Nabyvatele, za podmínky, že Nabyvatel (dle této licenční smlouvy) je oprávněn po Autoru Díla požadovat, aby přiměřeně přispěl na úhradu nákladů, tak, jak je stanoveno v § 60 odst. 3 Autorského zákona.

5.3 Nabyvatel není povinen Dílo užít.

5.4 Nabyvatel smí dílo nebo jeho název upravit či jinak měnit, je oprávněn Dílo spojovat s jinými díly i zařadit Dílo do díla souborného. Autor dává svolení k tomu, aby Nabyvatel pořídil pro účely užití uvedené v této smlouvě překlad Díla.

5.5 V případě, že z Díla plyne hospodářský výsledek Autorovi nebo Nabyvateli, rozdělení zisku bude řešeno dodatkem k této smlouvě.

Článek 6 **Výpověď smlouvy**

6.1 Každá smluvní strana může smlouvu kdykoliv písemně vypovědět bez udání úvodu.

6.2 Výpověď musí být učiněna doporučeným dopisem nebo datovou schránkou doručeným druhé smluvní straně. Výpovědní lhůta je stanovena na jeden rok a začíná běžet prvním dnem kalendářního měsíce následujícího po měsíci, v němž byla výpověď doručena druhé smluvní straně.

6.3 Autor nahradí Nabyvateli škodu, která mu odstoupením od smlouvy podle odstavce 1 a 2 vznikla. Účinky odstoupení nastanou nahrazením škody nebo poskytnutím dostatečné jistoty.

Článek 7 **Závěrečná ustanovení**

7.1 Smlouva je sepsána ve dvou vyhotoveních s platností originálu, která budou vložena do dvou výtisků Díla (práce), z toho Nabyvatel i Autor obdrží po jednom vyhotovení.

7.2 Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí Autorským zákonem a Občanským zákoníkem, popř. dalšími právními předpisy.

7.3 Smlouva byla uzavřena podle svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoli v tísní a za nápadně nevýhodných podmínek.

7.4 Smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Písku dne: 28.3.2025

.....
Autor

.....
Nabyvatel

Obsah

Úvod	7
1 Teoretická část	8
1.1 Php.....	8
1.2 HTML	8
1.3 Javascript.....	9
1.4 CSS	9
1.5 MySQL	9
1.6 phpMyAdmin.....	10
1.7 SMTP	11
1.7.1 smtp4dev	12
1.8 Apache.....	12
1.9 LDAP	13
1.10 XAMPP.....	13
1.11 diagrams.net	13
2 Praktická část.....	14
2.1 Databáze	14
2.2 Vytvoření databáze	14
2.3 Implementace databáze	15
2.4 Přihlášení	15
2.4.1 LDAP	16
2.4.2 Napojení na databázi	17
2.4.3 Login.....	17
2.5 Formulář pro uživatele.....	19
2.6 Posílání emailů.....	21
2.7 Historie uživatele.....	22

2.8	Stránka pro správce	23
2.9	CSS	24
	Závěr	26
	Seznam obrázků	28
	Seznam literatury	29

Úvod

Tématem této práce je vytvoření systému pro hlášení závad. Moderní a efektivní správu pro správce, a také uživatelsky přívětivé rozhraní pro uživatele ve webovém rozhraní. Systém bude realizován v PHP a v MySQL pro správu databáze. Správce bude mít k dispozici archiv závad, ve kterém bude moci upravovat jednotlivé závady, a uvidí také podrobnosti o každé závadě. Důležitým bodem je propojení s inventarizačním systémem spolu s obousměrnou emailovou komunikací.

1 Teoretická část

Tato část popisuje jednotlivé nástroje a programy, které jsou využity v mém projektu pro lepší orientaci v praktické části.

1.1 Php

Je hodně oblíbený open source programovací jazyk, který slouží především pro tvorbu dynamických webových stránek ve formátu HTML, XHTML, WML a webových aplikací. Zpracovává se na straně serveru, na rozdíl od jazyků jako JavaScript, které se vykonávají už v prohlížeči. [1]

PHP vyvinul Rasmus Lerdorf v roce 1994 jako sadu skriptů napsaných v jazyce C, určených k zaznamenávání návštěvnosti jeho webových stránek. První verze byla vydána v roce 1995. Od té doby se PHP vyvinulo v jeden z nejpopulárnějších programovacích jazyků na světě, používaný pro vývoj webových stránek a aplikací. Od verze 5 podporuje PHP objektově orientované programování, což umožňuje lepší strukturování a údržbu kódu. Tato vlastnost zlepšuje čitelnost a opakovatelnost kódu, což je v dnešní době důležité kvůli minimální obsazenosti paměti, a tudíž maximální efektivitě. [1]

Je kompatibilní s různými webovými servery, včetně Apache a Nginx, a podporuje širokou škálu platforem a operačních systémů, což znamená, že může běžet na různých operačních systémech, jako jsou Windows, Linux nebo macOS. PHP umožňuje vývojářům vytvářet dynamický obsah, pracovat s databázemi, zpracovávat formuláře a spravovat soubory na serveru. Jeho flexibilita a široká podpora různých databázových systémů, jako je MySQL, umožňuje efektivní správu a manipulaci s daty. Nabízí velké množství knihoven a rozšíření, která zjednodušují vývoj, mezi ně zařadíme právě MySQL, FTP, SMTP i LDAP. [2]

V Česku je PHP široce využíváno pro tvorbu různých webových aplikací, od osobních blogů po rozsáhlé e-commerce platformy. Jeho popularita a dostupnost zdrojů v češtině usnadňují jeho znalost a implementaci v lokálních projektech.

1.2 HTML

Základní značkovací jazyk pro vytváření a strukturování obsahu webových stránek. Používá se pro definování struktury webové stránky pomocí série elementů a tagů.

Do značek patří nadpisy, odstavce, obrázky, tabulky atd. Podle nichž se sdělují prohlížeči, jakým způsobem mají být jednotlivé části obsahu interpretovány a zobrazeny. HTML dokument je většinou rozdělen do dvou částí, hlavičky a těla. Hlavička obsahuje metadata, informace pro prohlížeč nebo napojení na CSS soubor, zatímco tělo samotná data pro uživatele. [3]

1.3 Javascript

Jedná se o víceúrovňový programovací jazyk, který umožňuje vývojářům manipulovat s obsahem webových stránek v reálném čase. Třeba vytvářet animace nebo reagovat na uživatelské chování. Má schopnost být spouštěn přímo v klientském prohlížeči, což je lepší pro server a pro plynulost stránky. [4]

1.4 CSS

Kaskádové styly, známé pod zkratkou CSS (Cascading Style Sheets), představují jazyk určený k definování vizuálního vzhledu webových stránek. Umožňují oddělit obsahovou strukturu dokumentu, vytvořenou pomocí HTML od jeho zobrazení, čímž zajišťují flexibilitu a efektivitu při tvorbě a údržbě webových projektů.

Můžeme zde nastavit vlastnosti písma, jako je barva, styl, velikost nebo definovat rozmístění jednotlivých prvků na stránce včetně okrajů, výplní a pozic. Umožňuje vytvářet layouty, které se poté přizpůsobují různým velikostem obrazovek. Jednotlivé definované styly lze opakovat pro více stránek, abychom dosáhli jednotného vzhledu. Externí CSS soubory jsou převážně cachovány v prohlížeči kvůli rychlejšímu načítání. [5]

1.5 MySQL

Jedná se o relační databázový systém pro správu databází (RDBMS), který využívá strukturovaný dotazovací jazyk (SQL) pro přístup a manipulaci s daty. Jeho název vznikl kombinací jména "My", což je jméno dcery spoluzakladatele Michaela Wideniuse, a "SQL", zkratky pro Structured Query Language. [6]

Byl vytvořen švédskou společností MySQL AB, kterou založili David Axmark, Allan Larsson a Michael "Monty" Widenius. Původní vývoj začal v roce 1994 a první verze byla vydána 23. května 1995. V roce 2008 společnost Sun Microsystems obdržela

MySQL AB, a následně v roce 2010 byla Sun Microsystems akvizována společností Oracle Corporation. [7]

MySQL je navrženo pro rychlost, spolehlivost a snadné použití. Podporuje více uživatelů a umožňuje současný přístup k databázím bez výrazného snížení výkonu. Díky své modulární architektuře umožňuje použití různých enginů, jako jsou InnoDB a MyISAM (starší, nahradil InnoDB). Které nám poskytují ukládání a zajištění integrity dat v databázi. [6]

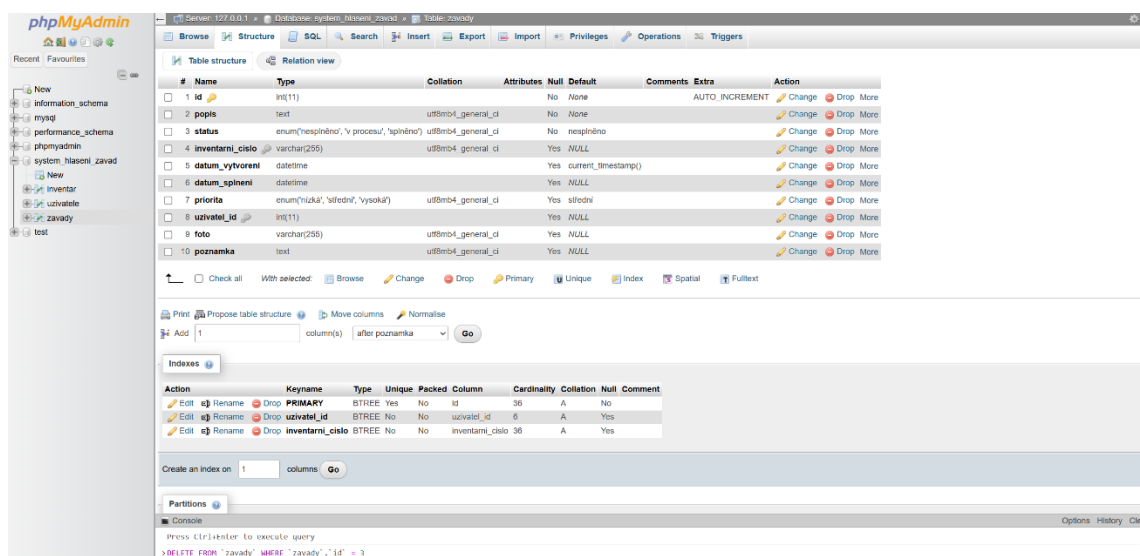
Nachází se právě často součástí populárního softwarového balíčku LAMP (Linux, Apache, MySQL, PHP/Perl/Python), který je široce používán pro vývoj a nasazení dynamických webových aplikací. Díky své schopnosti efektivně zpracovávat velké objemy dat a podpoře více uživatelů je MySQL preferovanou volbou pro mnoho webových aplikací, včetně systémů pro správu obsahu.

1.6 phpMyAdmin

Jde o bezplatnou webovou aplikaci napsanou v jazyce PHP, která slouží k efektivní správě MySQL a MariaDB databází prostřednictvím přehledného grafického rozhraní. Poskytuje funkce pro vytváření, úpravu a odstraňování databází, tabulek, sloupců, relací, indexů a uživatelských oprávnění. Podporuje import dat z různých formátů (CSV, SQL) a export do běžně používaných aplikací a formátů (Spreadsheet, Word, Excel, LaTeX, PDF). Dále můžeme ovládat více databázových serverů z jednoho rozhraní, a také monitorovat využití CPU paměti či sledovat aktivitu na serveru. [8]

Projekt phpMyAdmin zahájil v roce 1998 Tobias Ratschiller, inspirován nástrojem MySQL-Webadmin od Petera Kuppelwiesera. V roce 2000 Ratschiller projekt opustil kvůli nedostatku času, ale do té doby se phpMyAdmin stal jedním z nejpopulárnějších nástrojů pro správu MySQL s rozsáhlou komunitou uživatelů a přispěvatelů. V roce 2001 převzala vývoj skupina tří vývojářů – Olivier Müller, Marc Delisle a Loïc Chapeaux, kteří projekt zaregistrovali na platformě SourceForge. [9]

Tento nástroj se často využívá studenty a vývojáři díky své jednoduchosti, snadné instalaci a bohatým možnostem administrace databází, což zjednodušuje práci s daty v rámci webových projektů.



Obrázek 1 phpMyAdmin

Zdroj: vlastní zpracování

1.7 SMTP

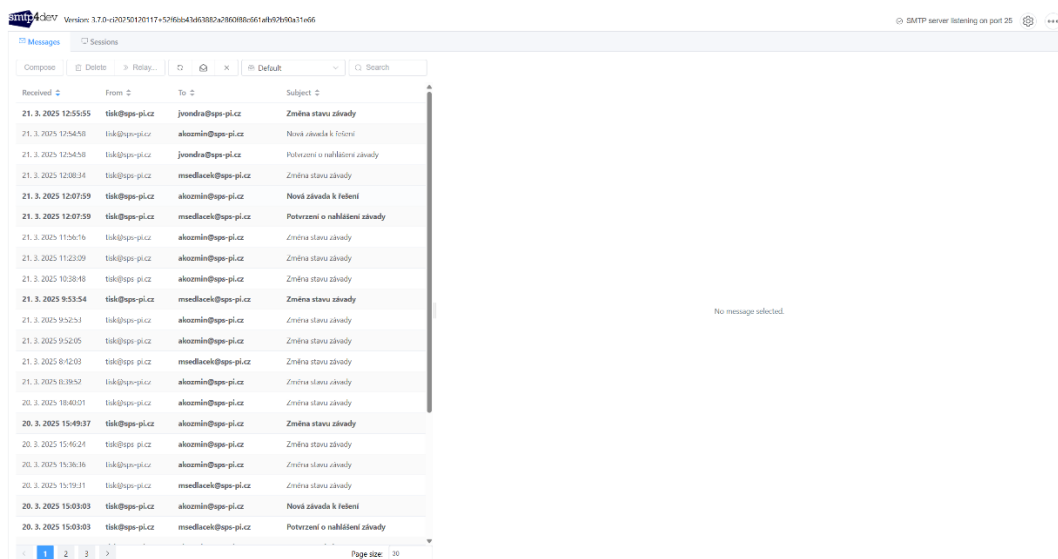
Simple Mail Transfer Protocol (SMTP) je standardní komunikační protokol pro přenos elektronické pošty (e-mailů) přes internet. Je navržen tak, aby umožňoval spolehlivý a efektivní přenos e-mailových zpráv mezi servery a klienty. SMTP je textově orientovaný protokol, kde odesílatel (klient) komunikuje se příjemcem (serverem) prostřednictvím příkazů a odpovědí. Typická SMTP relace zahrnuje tři hlavní fáze:

1. **Zahájení relace:** Klient naváže spojení se serverem a identifikuje se pomocí příkazu HELO nebo EHLO.
2. **Přenos zprávy:** Klient specifikuje odesílatele (MAIL FROM) a příjemce (RCPT TO) zprávy, následovaný příkazem DATA, po kterém následuje samotný obsah e-mailu.
3. **Ukončení relace:** Po úspěšném přenosu zprávy klient ukončí spojení pomocí příkazu QUIT.

SMTP používá Transmission Control Protocol (TCP) a standardně operuje na portu 25 pro komunikaci mezi servery a na portu 587 pro odesílání zpráv ze služeb třetích stran. [10]

1.7.1 smtp4dev

Bezplatný nástroj určený pro vývojáře, který simuluje SMTP server v lokálním prostředí. Namísto skutečného odesílání e-mailů zachytává a ukládá odchozí zprávy, což umožňuje ověřit správnost a formát e-mailů bez rizika odeslání reálným uživatelům. Je dostupný pro všechny PC platformy.



Obrázek 2 smtp4dev

Zdroj: vlastní zpracování

1.8 Apache

Apache HTTP Server, běžně označovaný jako Apache, je open-source webový server vyvinutý a udržovaný Apache Software Foundation. Byl vytvořen proto, aby poskytoval bezpečný, efektivní a rozšiřitelný server, který poskytuje HTTP služby v synchronizaci s aktuálními standardy HTTP. Jedná se o multiplatformní server, který umožňuje dynamické načítání modulů, což znamená že můžu přidávat nebo odebírat funkce dle potřeby. Pro správce je velkou výhodou možnost detailně nastavovat konfigurační soubory za účelem zabezpečení či správy virtuálních hostitelů.

Projekt Apache začal v roce 1995 jako nástupce NCSA HTTPd serveru. Skupina vývojářů začala vytvářet sadu oprav (patchů) pro NCSA HTTPd, což vedlo k vytvoření "a patchy" serveru, z čehož vznikl název Apache. Od té doby se stal jedním z nejpoužívanějších webových serverů na internetu díky schopnosti integrovat různé programovací jazyky a databázové systémy. [11]

1.9 LDAP

Jedná se o standardní aplikační protokol pro přístup a údržbu na adresářovém serveru. Položky jsou zde ve stromové struktuře a server je optimalizován pro data, která nejsou tak často aktualizována například uživatelská jména, hesla, email.

Podporuje operace jako je autentizace (Bind), vyhledávání a získávání záznamů (Search), porovnávání atributů (Compare) a samozřejmě také přidávání, mazání a modifikaci. Adresářové služby umožňují sdílení informací o uživateli, systémech, sítích a aplikacích napříč sítí. Využití LDAP je nezbytné pro efektivní správu a přístup k distribuovaným adresářovým informacím v moderních síťových prostředích. [12]

1.10 XAMPP

Multiplatformní softwarový balíček, který obsahuje vše potřebné pro lokální vývoj a testování webových aplikací. Balíček zahrnuje Apache, MySQL, PHP, phpMyAdmin a další nástroje, které společně umožňují jednoduše a rychle vytvořit funkční lokální server bez nutnosti složité konfigurace jednotlivých komponent. Je dostupný na všech platformách, a proto je oblíbeným nástrojem mezi vývojáři pro rychlé prototypování a testování webových aplikací. Aby se snížilo riziko chyb při jejich nasazení do reálného prostředí.

1.11 diagrams.net

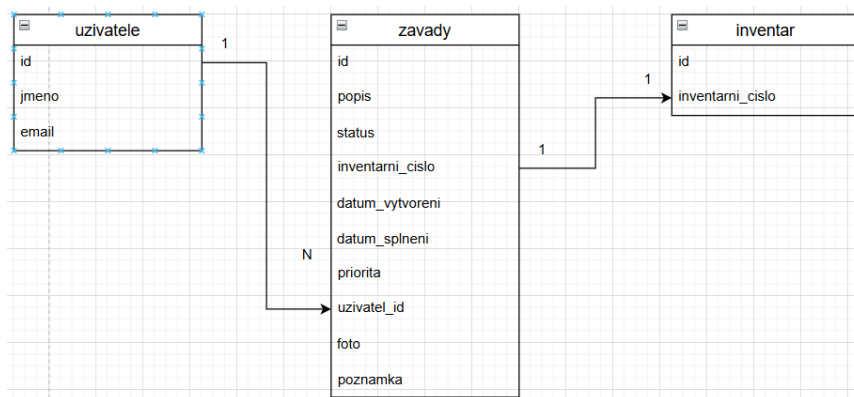
Dříve známý jako draw.io, je bezplatná aplikace pro tvorbu diagramů, která umožňuje uživatelům vytvářet různé typy diagramů, včetně Entity-Relationship (ER) diagramů. ER diagramy jsou klíčovým nástrojem při návrhu databází, protože vizualizují strukturu dat a vztahy mezi jednotlivými entitami.

2 Praktická část

V této kapitole se zaměřím na postup mého konkrétního řešení pro systém hlášení závad. Do detailu rozeberu každou část zadání od vytvoření databáze až po automatickou emailovou komunikaci.

2.1 Databáze

Databázi jsem si navrhl pomocí Entity-Relationship diagramu do vizuální podoby v diagrams.net, abych si to jednodušeji představil. Musel jsem brát v potaz, že budu určitě potřebovat tabulku uživatelů a závad, a to se vzájemným vztahem 1:N, protože jeden uživatel bude mít více závad, a naopak jedna závada bude mít jen jednoho uživatele. Dále jsem ještě doplnil tabulku pro inventární čísla kvůli jednoduššímu importu dat do databáze. Jednotlivé atributy v tabulkách jsem vytvořil podle toho, aby databáze byla komplexní a sbírala všechna podstatná data.



Obrázek 3 ER model

Zdroj: vlastní zpracování

2.2 Vytvoření databáze

Ve VSCode jsem si napsal jednotlivé inserty tabulek, jež níže vysvětlím, abych to poté mohl celé zkopírovat a vložit do phpMyAdmin a vytvořit databázi. První řádek vytváří databázi se jménem *system_hlaseni_zavad* poté se hned do ní vstupuje, aby se mohli vytvářet tabulky. Dále se vytváří tabulka se jménem *uzivatele* s atributem *id*, který je číslo s automatickým doplněním, a také primární klíč. Ten jednoznačně určuje danou instanci z relace, a tento klíč musí být unikátní pro tuto tabulku a nesmí být nulový. *Varchar* v atributu *jmeno* říká, že to bude maximálně 100 znaků, a že je toto pole povinné, protože nesmí být nulové. Poslední v tabulce hodně podobný *jmenu* je *email*,

jen může mít více znaků a musí být unikátní, protože nemůže mít jeden email více uživatelů v mém projektu. Za závorkou je příkaz pro využití úložného enginu *InnoDB* a pro použití cizích klíčů.

Druhá tabulka s názvem *inventar* není nijak zvláštní jen má jiné názvy atributů. Poslední tabulka *zavady* má samozřejmě svůj *primary key* ale třeba u *popis* je *text*, což umožňuje delší počet znaků než 255. *Status* má předdefinované tři stavy, a když by správce nevybral nic z těchto tří, tak se automaticky nastaví na *nesplněno*. *Datum_vyvoreni* používá funkci, která umožňuje ukládat datum a čas, a zároveň se automaticky vloží, když je vytvořen nový záznam. *Datum_splneni* je hodně podobné předchozímu jen může být nulový, jelikož když vytvořím závadu, tak nemám hned datum splnění. *Priorita* má stejný princip jako *status*, *uzivatel_id* je cizím klíčem. *Foto* má povolený řetězec 255 znaků kvůli názvu souboru, avšak je nulové, protože to není povinné. *Poznamka* je stejná jako *popis* jen není povinná.

Klíče jsou napsané jako poslední, první doplňuje do *uzivatel_id* z tabulky *uzivatele* jejich *id*. Druhý dělá to samé jen s inventárními čísly, a navíc když smažeme nějaké inventární číslo z tabulky *inventar*, tak se automaticky smažou tyto záznamy v tabulce *zavady*, abychom nepracovali se záznamy, které už neexistují. Obdobně to funguje s *update* jen se záznamy aktualizují.

2.3 Implementace databáze

Napsanou databázi vkládám do prostředí phpMyAdmin, které musím spustit pomocí virtuálního serveru XAMPP, ve kterém rozeběhnu MySQL. Až poté se pomocí adresy 127.0.0.1/phpmyadmin v prohlížeči dostanu do tohoto rozhraní, ve kterém otevřu Console a vložím popsany kód.

2.4 Přihlášení

Zprvu jsem vůbec přihlášení pro uživatele nechtěl řešit jen jsem chtěl udělat login pro admina, aby se dostal na svoji stránku pro správu závad. Jenže to by každý uživatel musel do formuláře zadávat svojí emailovou adresu, a mohl by také někdo nahlašovat závadu pod cizím emailem, což by nebylo úplně optimální. Tak jsem se rozhodl použít LDAP server pro autentizaci všech, který má škola k dispozici, abych omezil falešné závady, které by správce jen zdržovali.

2.4.1 LDAP

LDAP údaje mi poskytl správce sítě, a tudíž jsem je použil v kodu *ldap_auth.php*, který vysvětlím. Nejdříve nastavím proměnné do *putenv* proto, abych tyto citlivé údaje nevolal přímo z hlediska bezpečnosti. Ve funkci *getLdapConfig* zjišťuji, jestli jsou všechny položky definovány, a pokud nějaká položka chybí, tak to vypíše, o jakou se jedná. Na konci této funkce se vrací konfigurace v asociativním poli, se kterým pak pracují další funkce pro lepší orientaci.

Funkce pro logování mi zapisuje do souboru každou chybu nebo důležitou informaci. Má jednotný formát pro datum, čas, potom *configid* což je default, a nakonec chybovou hlášku.

```
2025-03-19 10:18:03 [default] User not found in LDAP
2025-03-20 11:50:45 [default] Admin bind failed
2025-03-20 11:51:24 [default] User authentication failed
2025-03-20 11:51:32 [default] User authentication failed
```

Obrázek 4 Logování do textového editoru

Zdroj: vlastní zpracování

Hlavní autentizační funkce *ldapAuth* nejprve načte celou funkci *getLdapConfig* do proměnné *\$config* poté sestrojí URL pro připojení do proměnné. Dále se připojí k LDAP serveru a nastaví LDAP protokol na verzi 3 a vypne *referrals* odkazy na další servery kvůli stabilitě spojení. Když funkce *\$ldap_conn* nenaváže spojení, tak vypíše chybovou hlášku. Poté se načtou informace a skript se pokouší přihlásit k LDAP serveru pomocí údajů administrátora. Pokud se akce nezdaří, tak se vypíše hláška, vykřičník označuje negaci a zavináč potlačuje chybové hlášení generované volanou funkcí.

Nyní můžeme hledat uživatele a vyrobíme si filtr pro jejich hledání pomocí uživatelského jména. Dále už hledá podle existujícího připojení, pozice v LDAP struktuře, filtru a poslední znamená co chci získat (mail a celé jméno), pokud ne tak se vypíše hláška. Celé se to uloží do *\$result*. Další funkce získá výsledky z hledání a převede je na pole pro přehlednější formát. Pokud v poli nic není tak uživatel neexistuje a vypíšeme hlášku.

Získám *DN* uživatele, protože každý ho má jedinečný a později ho použiji pro autentizaci. Získání emailu z atributu *mail* není tak jednoduché, tato hodnota se zde

nikde nenachází a já ji poté potřebuji pro automatickou emailovou komunikaci. I když se pokoušíme hledat první možnou adresu z pole tak nic nenajdeme a dva otazníky znamenají, že pokud tam nic není tak se vrátí hodnota *null*. Dále načítám obdobným způsobem celé jméno, pokud by i tady nic nebylo, tak se místo toho použije uživatelské jméno. Další *if* říká, že pokud je hodnota *\$userEmail null* nebo je nalezená hodnota emailu neplatná, protože neobsahuje znak zavináč, tak se *email* vytvoří automaticky podle šablony: uživatelské jméno a přípona @sps-pi.cz.

V poslední části se uživatel ověřuje vůči LDAP serveru, pokud údaje nesouhlasí vypíše se hláška. Hned poté se funkce uzavírá, jelikož už není dále potřeba. Po správné autentizaci vrátí funkce pole s *CN* a *email*. Úplně poslední je funkce pro kontrolu admina.

2.4.2 Napojení na databázi

Popisuji zde kód *db_connect.php*. Nejdříve si nastavím informace potřebné pro připojení do databáze. V tomto případě se jedná napojení pouze na lokální server, při implementaci na reálný server jen změním údaje. *DSN* řetězec definuje hodnoty z proměnných a dále jej použijem při vytváření *PDO*. Konfigurace *PDO* obsahuje: možnost pro vypisování výjimek při chybách, načítání dotazů jako asociativní pole nebo zpracovávání přípravných dotazů serverem, tím zvyšujeme bezpečnost a lepší komunikaci. Při nezdárném pokusu o připojení nám funkce *try* vypíše druh chyby, jinak se spojení naváže.

2.4.3 Login

Na tuhle stránku se dostanu zadáním adresy 127.0.0.1/dlouhodobka/ a díky „naindexování“ mě to přesune přímo sem. Zde je postup vytváření kódu *login.php*. *Session_start* umožňuje uchovávat informace o uživateli mezi stránkami. *Require* načítá soubory, které budeme potřebovat pro vykonání funkcí. Zpracování *POST* dat začne až po odeslání formuláře a uživatelské jméno se navíc ořízne od zbytečných mezer funkcí *trim*. Dále se volá autentizační funkce, která když proběhne, tak se do session uloží email a uživatelské jméno malými písmeny.

Poté hledáme, jestli uživatel už existuje v naší databázi podle emailu vytvořeného v *ldap_auth.php*. Když ne, tak pomocí dotazu vkládáme informace o uživateli do tabulky

uzivatele, na konci se získá nové *ID* uživatele a uloží se do *session*. Ovšem když jeho *ID* existuje, tak se načte pouze jeho *ID*.

Další podmínka rozhoduje, jestli je uživatel admin nebo jenom user, a to už se děje na konci *ldap_auth.php*. Pokud se tedy přenesení *true* value, tak je přesměrován na dashboard pouze pro správce, ostatní míří do *user_dashboard*. Pokud při autentizaci dojde k jakékoliv chybě, tak se výjimka zachytí a zpráva o chybě se uloží do proměnné *\$error*, která se může dále zobrazit.

V hlavičce HTML se nachází CSS soubor, právě pro login stránku, ve kterém si nejdříve pojmenuji barvy, s jakými budu pracovat, aby to zjednodušilo práci. Dále se tu nachází nastavení fontu písma, obrázek pro pozadí, jak budou jednotlivé položky zarovnány atd. Jak budou vypadat tlačítka a jaké mají zarovnání, velikost, tohle všechno zde najdeme.

V body neboli těle, se nejdříve dostáváme do kontejneru pro stylování. Zobrazuje se základní nadpis a pokud máme v *\$error* nějakou hlášku, tak se nám vypíše přímo pod nadpisem, ve formátu nastaveného *\$error* v CSS. Dále se dostáváme do klasického formuláře, který když vyplníme, tak posíláme *username* a *password* pomocí metody *POST* na server. Tato odeslaná data se na serveru zachytí pomocí proměnných *\$_POST*. Obě okna se samozřejmě musí vyplnit, proto je tam *required* a u hesla jsem použil *input type password*, který zajišťuje, že heslo se zobrazuje jako hvězdičky pro lepší bezpečnost. Na konci je samotné tlačítko pro odeslání formuláře, které při najetí kurzorem změní pozadí na žlutou barvu.



Obrázek 5 Login stránka

Zdroj: vlastní zpracování

2.5 Formulář pro uživatele

Do formuláře pro uživatele se dostaneme pomocí přesměrování z *login.php*. Tady se zaměřím na popis funkcí ve formuláři pro uživatele z *user_dashboard.php*. Nejprve se znovu inicializuje session, připojíme se k databázi načteme knihovny a funkci pro odesílání emailu. Poté definujeme použití knihovny, aby bylo možné použít třídy *PHPMailer* a *Exeption* přímo. Dále se kontroluje, zda je uživatel přihlášený, a to pomocí toho, jestli existuje session proměnná email, pokud ne je přesměrován na login, aby se zabránilo neoprávněnému přístupu.

Příkazem do databáze, aby se mi načetla všechna inventární čísla z tabulky *inventar* vzestupně, dostanu pole, které později použiji pro výběr možností ve formuláři. Prázdné proměnné připravuji proto, aby poté sloužili k informačním hláškám při odesílání formuláře.

If začíná kontrolou, jestli je odeslán formulář metodou *POST* a existuje pole *popis*. Pokud uživatel zvolí „bez inventárního čísla“ nastaví se hodnota na *null*. Jinak se zadané číslo načte a ořízne, pokračuji příkazem pro to, jestli zadané číslo skutečně existuje, když ne vypíše zprávu. Dále zpracovávám nahrávání souborů, nejprve zkontroluji, zda uživatel něco opravdu nahrál, pokud ne tak se tato část přeskočí. Pokud ano, tak se počítá, jestli jich je více než tři, protože maximum jsou tři soubory pro nahrání, jinak vypíše chybovou hlášku.

Obrázky se budou ukládat do složky *uploads/*, pokud neexistuje automaticky se vytvoří s úplným přístupem, a kdyžtak i všechny nadřazené složky. Cyklus *for* prochází každý nahraný soubor jeden po druhém. Vytvoří se jedinečný název souboru pomocí funkce *time* podtržítka a původní název souboru. Dále ověřuji, jestli se jedná o správný typ souboru, aby uživatel nenahrál nebezpečný soubor. Poté probíhá nahrávání souboru na server z dočasného umístění do složky *uploads/*. Tato cesta k nahranému obrázku se uloží do pole *\$imagePaths*. Cesty obrázků se uloží jako text oddělený čárkami. Jako další se hledá id uživatele podle uloženého emailu v *session*. Otazník se později nahradí hodnotou *\$email* pomocí *execute*, a pokud byl nalezen, tak se uloží jeho *id*, jinak bude *null*.

Pro vložení záznamu do databáze si nejdříve připravíme dotaz a poté jej provedeme se zadanými nebo získanými hodnotami. Nakonec vypíše zprávu o úspěšném odeslání. Ihned poté se odesílá potvrzovací email pro toho, který formulář vyplnil a pro správce,

že má nahlášenou novou závadu. Kdyby se stala nějaká chyba při vkládání údajů do databáze, budeme upozorněni, o jakou chybu se jedná.

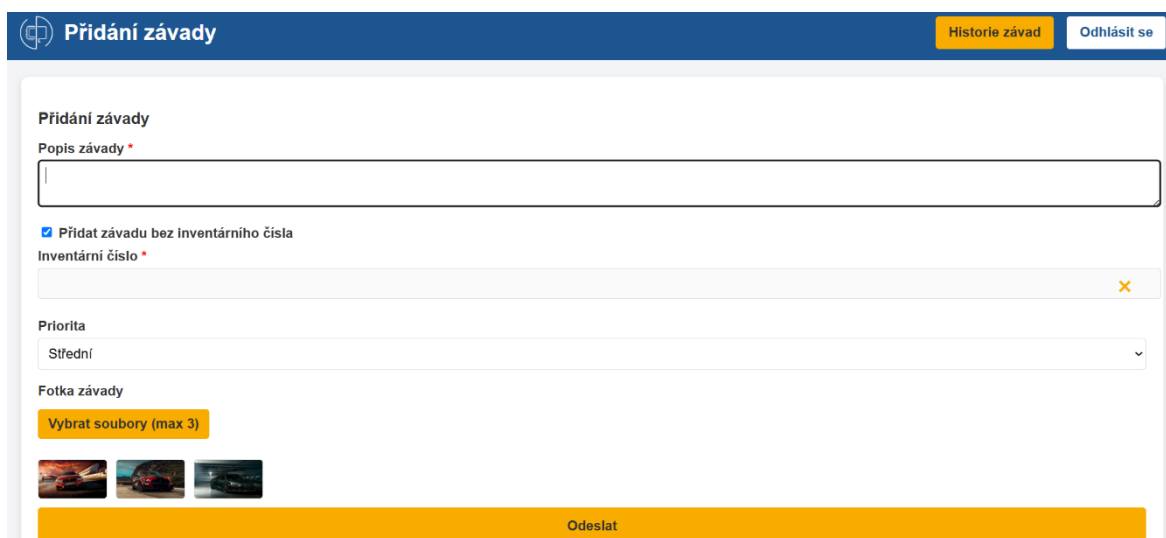
Dostáváme se do HTML, v němž na začátku odkazujeme na CSS, aby byl kód čistší. Vkládám tlačítka s odkazy pro odhlášení a zobrazení historie. Vše má svoji třídu, která má nadefinovaný vzhled v CSS, a také díky jednomu souboru můžu tyto layouty použít vícekrát na různých stránkách, a hlavně v jednotné formě. Ještě jsem v hlavičce použil *viewport* pro přizpůsobení stránky mobilním zařízením. Pod hlavičkou stránky, kterou mi vytvoří naformátované CSS, tak je prostor pro chybové nebo potvrzovací zprávy v zelené nebo červené barvě. Na začátku formuláře jsem musel použít *enctype="multipart/form-data"*, aby bylo možné nahrávat obrázky níže. Jako další je *checkbox* pro možnost zadávat závady i bez inventárního čísla.

Inventární číslo je pomocí *for* propojené přímo pomocí jeho *id*. Třída *required* dělá červenou hvězdičku za popiskem, aby uživateli bylo jasné, že je toto pole povinné. Vyplňování inventárního čísla probíhá pomocí toho, že se uživateli rozjede celá nabídka inventárních čísel, a když začne psát, tak se mu automaticky nabízejí ta, která obsahují znaky, co zadal. Tudíž mám v *inputu list* a důležité je mít i *id* a *name* pro propojení s Javascriptem níže, a abych věděl pod kterým klíčem se odešle formulář. Pod tím je tlačítko pro vymazání políčka, které je také napojené na Javascript níže. V *datalistu* je php část právě pro nabízení možností, takové „našeptávání“. Když začnu psát znaky, tak nezáleží na velkých či malých písmenech.

Priorita nabízí volbu mezi přednastavenými hodnotami a předem je označená jako střední. Při kliknutí na tlačítko vybrat soubory se otevře výběr souborů a můžeme vybrat maximálně tři, ale kontrola probíhá trochu níže. Vybrané obrázky se zobrazí v minimalizované podobě také díky JS. Tlačítko odeslat odešle formulář a data uloží do databáze.

První funkce v JS vykonává to, že když se tlačítko v *checkboxu* zaškrtně, tak vymaže hodnotu v textovém poli a deaktivuje pole, aby se do něj nedalo nic zadávat. Je to proto, aby nám nevznikala nekonzistence dat, tedy aby nebylo vybráno inventární číslo a zároveň zaškrtnuté políčko. Po změně hodnoty v poli *inventarni_cislo* script porovnává hodnotu s položkami v *datalistu*, když nalezne shodu nastaví se na „jen pro čtení“. Důvod je, aby se nestalo, že se zadá hodnota inventárního čísla, které neexistuje. Dále je tlačítko pro vymazání inventárního čísla, na které když kliknu,

tak se vymaže hodnota v poli a změní se na editovatelné. Jako další se zpracovává nahrávání souborů a zobrazení náhledů obrázků. Před zobrazením se obsah pro náhledy vymaže, poté se kontroluje, jestli není souborů více jak tři. Když je podmínka splněna pokračuje *if* dále a pomocí cyklu se jednotlivé soubory kontrolují, jestli řetězec začíná na *image/*. Pro každý obrázek se pomocí *FileReader* načte soubor ve formátu URL. Hned po načtení se vytvoří *img* element, jehož adresou je URL. Obrázek se poté zobrazí v minimalizované formě, a když tam máme ty obrázky, tak se kontejner zobrazí jinak ne, aby tam nebylo “prázdný soubor”.



Obrázek 6 User dashboard

Zdroj: vlastní zpracování

2.6 Posílání emailů

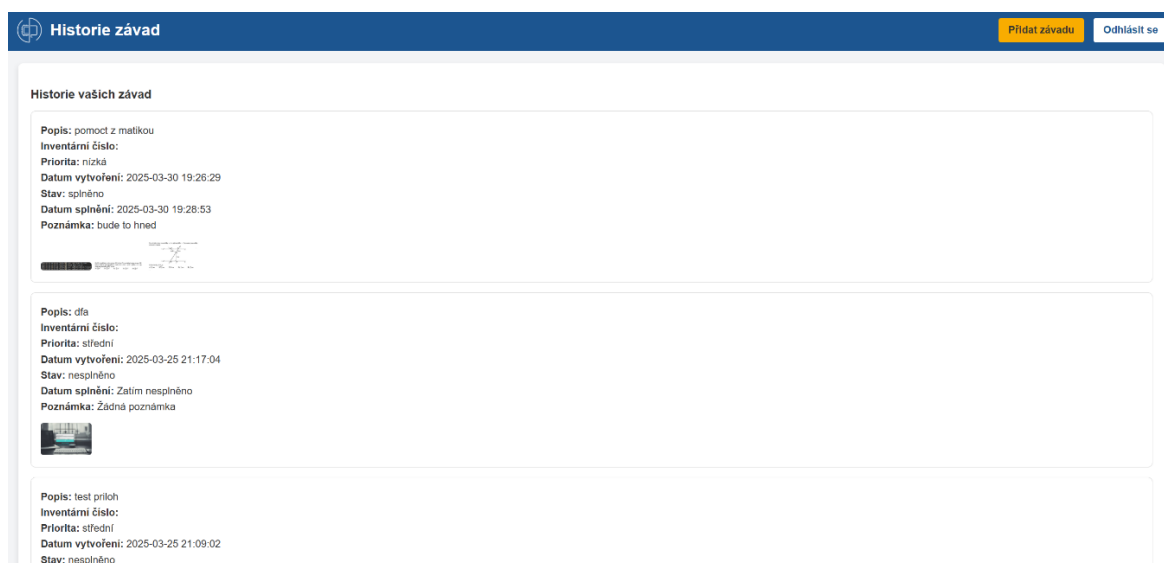
Pro vývojářské účely použijí smtp4dev. Jedná se o soubor *sendEmail.php*, jenž nejdříve umožní používat *PHPMailer* a *Exception* pro výjimky co může *PHPMailer* vyhazovat. Funkce je dost jednoduchá a přijímá tři parametry. Hned se vytvoří nová instance třídy s parametrem *true*, aby byly při chybách zobrazovány výjimky. Jako další se nastavuje používání SMTP, server, vypínání autentizace (na localhostu není potřeba), port, znaková sada, adresa a jméno odesílatele. Ještě se přidá adresát, určí se, že zpráva nebude v HTML a nastaví se tělo a předmět zprávy. Teď už odesíláme email, pokud se to nepovede vypíše se výjimka s podrobnou chybovou zprávou do logu, jinak se zapíše úspěšná zpráva.

2.7 Historie uživatele

Sem se dostanu díky odkazu v *user_dashboard* proto, aby každý uživatel mohl vidět svůj archiv závad, a také průběh řešení závady. Na začátku máme klasické zahájení *session*, připojení k databázi a ověření, zda je uživatel přihlášen. Jako další je získání *emailu*, a podle něj vyhledání *id*, které už je vysvětleno výše. Když nalezneme *id*, tak lze vykonat dotaz, v němž hledám všechny závady přihlášeného uživatele pomocí *where* a ještě je seřadím od nejnovějších po nejstarší. Výsledky jsou načteny do pole a uloženy do proměnné *\$zavady*.

HTML hlavička je totožná jen s jinými odkazy. V hlavním obsahu nejdříve ověřuji, jestli máme vůbec nějaké závady, pokud ne zobrazí se nápis. Pomocí *foreach* se prochází pole a pro každou závadu se zobrazují jednotlivé údaje z databáze. Když se nachází nějaká hodnota v poli *foto*, tak vím, že jsou jednotlivé obrázky oddělené čárkou. Pomocí funkce *explode* se text rozdělí na jednotlivé cesty, a díky tomu můžeme vytvořit *img* element. Každý obrázek se po kliknutí zobrazí v originální velikosti pomocí JS.

Modal okno se zobrazí jako okno z části překrývající stránku. *Span* je tlačítko odkazující na funkci pro zavření okna. URL obrázku předává funkce *openModal* a díky *block* se právě zobrazí obrázek v modálním okně. Ještě je tu možnost zavřít obrázek kliknutím mimo obrázek, a to pomocí funkce, která to kontroluje a nastaví se případně na *none*.



Obrázek 7 Historie uživatele

Zdroj: vlastní zpracování

2.8 Stránka pro správce

Začátek *admin_dashboard.php* funguje úplně stejně jako *userdashboard.php*. Poté se ověřuje, jestli je opravdu přihlášen admin pomocí *true value* z *ldap_auth.php*, kdyby nebyl tak je přesměrován na *login* stránku. Dále je funkce pro řazení, v základu je *DESC* tedy od nejnovější závady po nejstarší, jinak když se vstupní hodnota změní na *ASC*, tak je to obráceně. Dále probíhá filtrace závad na základě předaných hodnot metodou *GET*, vytváří se prázdné pole a do něj se přidávají hodnoty, pokud byly zvoleny ve formuláři. Připraví se dotaz, který vybírá vše z tabulky závady, a ještě přidává data z tabulky *uzivatele*, aby měl správce nejvíce možných údajů o daném uživateli. Pokud jsme zadali nějaký filtr, tak se vytváří podmínka *where* s počtem otazníků. Každý otazník odpovídá jednomu filtru, třeba v procesu nebo splněno, takže maximálně tam budou tři a minimálně nula, protože není povinnost filtrovat. V předešlé proměnné si to spočítám a oddělím čárkami, aby to databáze zvládla zpracovat. Pokud není filtr zadán, tak se vylučují závady se stavem splněno, aby správce viděl opravdu závady, které řeší nebo jsou nově přidáné. Na konci se doplní dotaz pro řazení, které je právě zvolené.

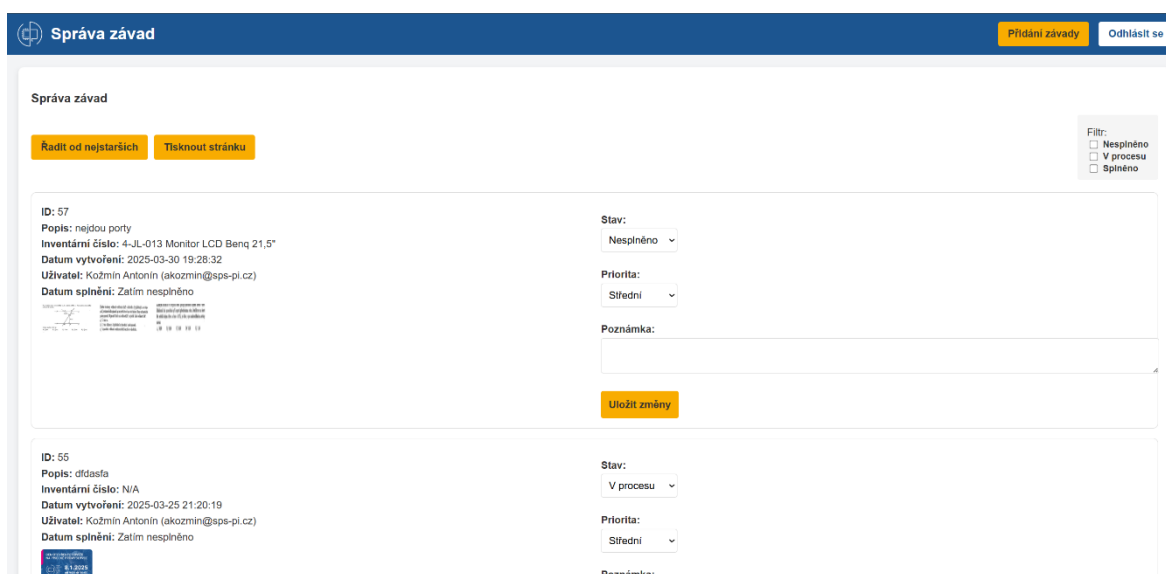
Vytvoří se přípravný dotaz, abych zabránil SQL injection. Dále se už vykoná dotaz, pokud je filtr prázdný, tak se nic nedoplní, jinak ano pomocí *\$filter_statuses*. Metoda *fetchAll* načte vše do proměnné *\$zavady*. Další je funkce pro úpravu závady a je spuštěna jen tehdy, když byl odeslán formulář. Pro aktualizaci záznamu v databázi se připraví dotaz, buď aby se doplnil aktuální čas, když je označené splněno, jinak jen změna údaje a *datum_splneni* zůstane s hodnotou *null*. Na konci se dotaz vykoná pomocí *execute*.

Jako další je na řadě odeslání emailu uživateli, co závadu nahlásil, ale nejdříve musíme získat jeho adresu, a to pomocí cizího klíče navázaného na závadu. Pokud se podaří najít jeho adresu tak se uloží se do proměnné. Vyplním předmět a zprávu, abych mohl zavolat funkci *sendEmail*. Do zprávy se rovnou vypisují aktuální změny, takže status, priorita a poznámka, aby se uživatel nemusel přihlašovat a koukat se do historie co vlastně bylo změněno.

Dále je script pro kontrolu a udržení filtrů, zda jsou v URL nastaveny filtry podle stavu závady, pokud ano připojí je do proměnné. Díky tomu zůstávají filtry zachované při změně nějaké závady.

Hlavička HTML je stejná, v body je na začátku tlačítko pro řazení a filtr, s napojením do URL. Ještě je zde tlačítko pro tisk stránky. Jako další je výpis závad a jejich detailů, při existenci obrázku se spustí funkce *modal* pro zvětšení, jak už jsem vysvětloval výše. Formulář pro aktualizaci závady má skrytý vstup, kde je *id*, které potřebujeme identifikovat abychom mohli závadu aktualizovat. Přidal jsem pole pro poznámku, aby si sem správce mohl psát třeba nějaké podrobnosti nebo i díky tomu informovat uživatele, jaký bude další postup. Na konci je tlačítko pro uložení změn, které spustí funkci, jenž jsem popisoval.

Funkce *openModal* pracuje úplně totožně jako v *userdashboard.php*. Když někdo zmáčkne tlačítko pro odhlášení, tak se automaticky *session* zničí a přesměruje ho to na login stránku.



The screenshot shows the 'Správa závad' (Issue Management) admin dashboard. At the top, there's a blue header with the title 'Správa závad' and two buttons: 'Přidání závady' (Add issue) and 'Odhlásit se' (Logout). Below the header, there's a section with two buttons: 'Řadit od nejstarších' (Sort by oldest) and 'Tisknout stránku' (Print page). On the right, there's a filter section with three checkboxes: 'Nesplněno' (Not completed), 'V procesu' (In process), and 'Splněno' (Completed). The main content area displays two issue cards. The first card (ID: 57) shows details for a monitor issue, including its description, inventory number, creation date, user, and completion status. It has dropdown menus for 'Stav' (Status) and 'Priorita' (Priority), a text area for 'Poznámka' (Note), and an 'Uložit změny' (Save changes) button. The second card (ID: 55) shows details for a mouse issue, with similar fields and a 'Uložit změny' button.

Obrázek 8 Admin dashboard

Zdroj: vlastní zpracování

2.9 CSS

Vyberu jen nejzajímavější řádky ze *style2.css*, protože jinak je to pořád stejně se opakující systém. V *.dashbord* nebo *.dashboard-controls* se používá *flexbox* pro automatické zarovnávání podle dostupného místa stránky na obrazovce, aby byla dobrá reakce při změně velikosti okna. Při přidávání souboru máme standardní tlačítko pro výběr souboru neviditelné, protože má omezené možnosti formátování. Díky kontejneru, ve kterém se nachází, je kontejner naformátovaný podle mých představ, ale přitom tam to tlačítko je a funguje při zmáčknutí. Kontejnery jsou takovou

složkou, ve které mám třeba jednotlivá tlačítka a hodně ulehčují práci při zarovnávání, a celkovém vzhledu stránky.

Závěr

V praktické části práce jsem podrobně popsal celý postup realizace webového systému pro hlášení závad. Hlavním cílem bylo vytvořit jednoduchý a efektivní nástroj pro správu závad na škole, s důrazem na uživatelskou přívětivost, bezpečnost a funkčnost.

Nejdříve jsem navrhl databázi s využitím Entity-Relationship diagramů, abych jasně definoval vztahy mezi jednotlivými tabulkami uživatelů, závad a inventárních čísel. Díky tomu jsem zajistil optimální strukturu dat, dostatečnou flexibilitu pro budoucí úpravy a snadnou správu informací. Databáze byla implementována pomocí phpMyAdmin v rámci virtuálního serveru XAMPP, což výrazně usnadnilo následnou správu a testování jednotlivých funkcí.

Z hlediska zabezpečení systému jsem zvolil autentizaci prostřednictvím školního LDAP serveru. Díky tomu jsem mohl efektivně předcházet neoprávněným či falešným hlášením závad, jelikož uživatelé využívají své existující školní přihlašovací údaje. Citlivé informace o připojení k LDAP jsou bezpečně uloženy pomocí funkce *putenv*, čímž se minimalizují potenciální bezpečnostní rizika a zároveň je zachována přehlednost kódu.

Významnou součástí řešení byla implementace automatické emailové komunikace. Pro tento účel jsem využil knihovnu *PHPMailer*, která umožňuje spolehlivé a rychlé odesílání notifikačních emailů jak uživatelům, kteří závalu nahlásili, tak správci systému. Díky této automatizaci jsou všichni uživatelé průběžně informováni o stavu svých závad a správce má okamžitý přehled o nových požadavcích.

Při návrhu uživatelského rozhraní jsem kladl velký důraz na jeho intuitivnost a jednoduchost použití. Implementoval jsem možnost nahrávání fotografií závad s automatickým zobrazením náhledů ve formuláři, což výrazně usnadňuje popis a identifikaci problému. Dále jsem přidal funkcionalitu automatického doplňování inventárního čísla, kdy systém při zadávání jednotlivých znaků dynamicky nabízí odpovídající položky z databáze. Také systém umožňuje snadné filtrování a řazení závad, čímž správce může rychleji a efektivněji reagovat na jednotlivé požadavky.

Naučil jsem se spousty nových dovedností v oblasti programování Javascriptu a PHP, kde jsem musel používat obsáhlejší podmínky a zamezit všem kritickým situacím,

které mohli ohrozit chod systému. Seznámil jsem se s bezpečnou autentizací přes LDAP nebo automatizovanou emailovou komunikací.

Celkově jsem vytvořil funkční, praktický a uživatelsky přívětivý nástroj, který splňuje všechny stanovené požadavky a je připraven k nasazení v reálném provozu školy. Zároveň je celý systém navržen tak, aby byl snadno rozšiřitelný a přizpůsobitelný podle případných budoucích potřeb školy a uživatelů.

Seznam obrázků

Obrázek 1 phpMyAdmin	11
Obrázek 2 smtp4dev	12
Obrázek 3 ER model.....	14
Obrázek 4 Logování do textového editoru.....	16
Obrázek 5 Login stránka	18
Obrázek 6 User dashboard.....	21
Obrázek 7 Historie uživatele	22
Obrázek 8 Admin dashboard.....	24

Seznam literatury

- [1] *PHP*, N.d.. Online. Wikipedie. Dostupné z: <https://cs.wikipedia.org/wiki/PHP>. [cit. 2025-03-26].
- [2] *PHP*, N.d.. Online. PHP kosek. Dostupné z: <https://www.kosek.cz/php/>. [cit. 2025-03-26].
- [3] *HTML*, N.d.. Online. Vavyskov. Dostupné z: <https://web.vavyskov.cz/znackovaci-jazyk.html>. [cit. 2025-03-26].
- [4] *Javascript*, N.d.. Online. Webglobe. Dostupné z: <https://www.webglobe.cz/poradna/co-je-javascript?>. [cit. 2025-03-26].
- [5] *CSS*, N.d.. Online. Vavyskov. Dostupné z: <https://web.vavyskov.cz/kaskadove-styly.html>. [cit. 2025-03-26].
- [6] *MySQL*, N.d.. Online. Wikipedie. Dostupné z: <https://cs.wikipedia.org/wiki/MySQL>. [cit. 2025-03-26].
- [7] *MySQL*, 2017. Online. EXADEL. Dostupné z: <https://exadel.com/news/old-reliable-mysql-history/>. [cit. 2025-03-26].
- [8] *PhpMyAdmin*. Online. PhpMyAdmin. Dostupné z: <https://www.phpmyadmin.net/>. [cit. 2025-03-27].
- [9] *PhpMyAdmin*, N.d.. Online. Wikipedie. Dostupné z: <https://cs.wikipedia.org/wiki/PhpMyAdmin>. [cit. 2025-03-26].
- [10] *SMTP*, 2024. Online. Geeksforgeeks. Dostupné z: <https://www.geeksforgeeks.org/simple-mail-transfer-protocol-smtp/>. [cit. 2025-03-27].
- [11] *Apache*, N.d.. Online. APACHE. Dostupné z: https://httpd.apache.org/ABOUT_APACHE.html. [cit. 2025-03-27].
- [12] *Ldap*, N.d.. Online. LDAP. Dostupné z: <https://ldap.com/basic-ldap-concepts/>. [cit. 2025-03-27].