Platforma pro ukládání a sdílení hesel v pracovních týmech

Jakub Neuman

Bakalářská práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub Neuman**
Osobní číslo:	**A21211**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Platforma pro ukládání a sdílení hesel v pracovních týmech**
Téma práce anglicky:	**Platform for Storing and Sharing Passwords in Work Teams**

Zásady pro vypracování

1. Specifikujte požadavky na systém s ohledem na jeho zabezpečení.
2. Navrhněte bezpečný způsob sdílení hesel mezi jednotlivými uživateli.
3. Zpracujte funkce pro management sdílení hesel.
4. Navrhněte samotný systém pro online správu uživatelských hesel.
5. Navržený systém implementujte v testovacích prostředí a ověřte jeho funkčnost.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. MARTIN, Keith M. _Everyday cryptography: fundamental principles and applications_. Oxford: Oxford University Press, 2012, xxi, 530 s. ISBN 9780199695591.
2. BERNSTEIN, Daniel J., Johannes A. BUCHMANN a Erik DAHMEN, ed. _Post-quantum cryptography_. Berlin: Springer, [2009], viii, 245 s. ISBN 978-3-540-88701-0.
3. AWAD, Ali Ismail a Jemal H. ABAWAJY. _Security and privacy in the Internet of things: architectures, techniques, and applications_. Piscatway, NJ: IEEE Press, [2022], 1 online resource. Dostupné z: doi:9781119607755.
4. BRAY, Shannon. _Implementing cryptography using Python_. Indianapolis: Wiley, 2020, 1 online resource. ISBN 9781119612216. Dostupné také z: https://proxy.k.utb.cz/login?url=https://onlinelibrary.wiley.com/doi/book/10.1002/9781119612216.
5. ADAHMAN, Zillah, Asad WAQAR MALIK a Zahid ANWAR. An analysis of zero-trust architecture and its cost-effectiveness for organizational security. Computers & Security [online]. North Dakota State University (NDSU), USA; National University of Sciences and Technology (NUST), Islamabad, Pakistan: Elsevier, 2022, 5 May 2022, Revised 28 June 2022, 2022(122), 1-13 [cit. 2022-12-01]. Dostupné z: doi:[https://doi.org/10.1016/j.cose.2022.102911](https://doi.org/10.1016/j.cose.2022.102911 "https://doi.org/10.1016/j.cose.2022.102911")

Vedoucí bakalářské práce: **Ing. David Malaník, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **5. listopadu 2023**
Termín odevzdání bakalářské práce: **13. května 2024**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Jakub Neuman, v.r.
podpis studenta

ABSTRAKT

Táto bakalárska práca sa zaoberá problematikou zdieľania a ukladania hesiel v tímoch. Práca začína identifikáciou požiadaviek na systém. Ďalej sa zaoberá návrhom bezpečného zdieľania hesiel. Na základe tohto návrhu je vypracovaný manažment systému, ktorý spĺňa stanovené požiadavky. Tento návrh zahŕňa technické aspekty systému, ako aj jeho užívateľské rozhranie. Následne sa práca venuje implementácii navrhnutého systému. V závere sa vykonáva overenie funkčnosti systému v reálnom prostredí, pričom sa zhodnocuje jeho výkon a schopnosť spĺňať požiadavky na bezpečné zdieľanie hesiel v tímovom prostredí.

Klíčová slova: zdieľanie hesiel, kybernetická bezpečnosť, návrh systému, šifrovanie, bezpečné zdieľanie, MVC, C#

ABSTRACT

This bachelor thesis deals with the issue of sharing and storing passwords in teams. The thesis starts by identifying the requirements for the system. It then deals with the design of secure password sharing. Based on this design, a management system is developed to meet the specified requirements. This design includes the technical aspects of the system as well as its user interface. Subsequently, the thesis deals with the implementation of the proposed system. At the end, a functional verification of the system in a real environment is performed, evaluating its performance and its ability to meet the requirements for secure password sharing in a teams.

Keywords: password sharing, cybersecurity, system design, encryption, secure sharing, MVC, C#

Týmto spôsobom sa chcem poďakovať vedúcemu mojej práce Ing. Davidovi Malaníkovi Ph.D. za vecné a cenné rady pri písaní bakalárskej práce. Zároveň chcem poďakovať svojim rodičom a súrodencom za podporu pri celej dĺžke štúdia a v neposlednom rade kamarátom za odľahčenie nálady v rôznych situáciách.

Prohlašuji že při tvorbě této práce jsem použil/a nástroj generativního modelu AI [https://chatgpt.com/] za účelem řešení chybných kódů. Po použití tohoto nástroje jsem provedl kontrolu obsahu a přebírám za něj plnou zodpovědnost.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

V tejto digitálnej ére, keď sa práca stáva čoraz viac digitálnou a tímová spolupráca sa presúva online, je správa hesiel a citlivých prístupových údajov nevyhnutná pre úspešnú a bezpečnú prácu. V rámci pracovných tímov sa často stretávame s potrebou zdieľať prístupové údaje k rôznym aplikáciám, službám a sieťam. Tento proces však môže byť komplikovaný, najmä ak nie je správne zabezpečený a organizovaný.

S rastúcim počtom účtov, hesiel a prihlasovacích údajov sa zvyšuje aj riziko ich straty, zneužitia alebo úniku. Bezpečná správa týchto citlivých informácií je preto pre každý tím nevyhnutná. Z toho dôvodu vznikajú platformy a nástroje na ukladanie a zdieľanie hesiel.

Tieto platformy poskytujú riešenie, ktoré tímom umožňujú centrálne ukladať, spravovať a zdieľať prístupové údaje s dôrazom na bezpečnosť, jednoduchú dostupnosť a efektívnu spoluprácu. Odstraňujú zložitosti spojené s manuálnym zdieľaním hesiel, čím zvyšujú produktivitu a zlepšujú bezpečnosť pracovného prostredia.

Cieľom tejto práce je preskúmať a analyzovať požiadavky a kľúčové aspekty spojené s vývojom platformy na ukladanie a zdieľanie hesiel v pracovných tímoch. Ďalej sa zameriame na návrh a implementáciu takejto platformy, ktorá bude nielen funkčná, ale aj bezpečná a používateľsky prívetivá. Budeme skúmať spôsoby implementácie pokročilých bezpečnostných funkcií, ako je šifrovanie údajov, dvojfaktorová autentifikácia a auditné záznamy, ktoré pomáhajú chrániť citlivé informácie tímu.

Prostredníctvom tejto práce chceme poskytnúť praktický návrh a riešenie, ktoré bude slúžiť ako účinný nástroj na správu hesiel pre tímy, zvyšovať bezpečnosť, znižovať administratívnu záťaž a podporovať spoluprácu.

I. TEORETICKÁ ČÁST

1 ANALÝZA A NÁVRH ZABEZPEČENIA SYSTÉMU

Návrh systému je kritická časť plánovania projektu, kde budú vytvorené požiadavky, ktoré sa budeme snažiť implementovať do funkčného a bezpečného stavu. Na správnu funkcionalitu so všetkými potrebnými opatreniami a databázovými reláciami je nutné vytvoriť pomocou databázového modelovania tabuľky, ktoré budú reprezentovať jednotlivé časti stránky, prípadné spojenia a budú stredom všetkej funkcionality.

Pred tým než môžeme začať navrhovať stránku musíme sa oboznámiť s rôznymi hrozbami, ktoré by mohli ohroziť vytvorený systém. Následne si môžeme vytvoriť návrh systému, ktorý vie obmedziť tieto hrozby. Je potrebné si vypísať požiadavky na systém, na ktoré je potrebné brať ohľad pri spracovaní celej stránky. V aktuálnej kapitole bude vysvetlená potrebná funkčnosť stránky, ako bude stránka pracovať, aké metódy a funkcie bude využívať a na akej architektúre budeme celú stránku stavať.[1]

1.1 Návrh požiadavkou na platformu

Navrhnutie správnych kombinácií metód a funkcií na vytvorenie bezpečnej platformy pre zdieľanie hesiel by mali zahŕňať rýchle, ale bezpečné metódy, ktoré užívateľovi nebudú zhoršovať pocit z platformy. Pri návrhu potrebujeme myslieť aj na riadenie a hierarchiu užívateľov, potrebnú komunikáciu s databázou a druh implementácie tejto platformy. Na komunikáciu s databázou bude využitý relačný databázový systém MySQL. Na vytvorenie databázových tabuliek bude využitý Microsoft Entity Framework, kvôli rýchlemu a zrozumiteľnému vytvoreniu štruktúry.

1.1.1 Funkčné požiadavky systému

Systém by mal:

- mať možnosť vytvárať a spravovať používateľské účty
- priradzovať rôzne úrovne prístupu k heslám
- evidovať prístup k heslám a zmeny hesiel
- mať možnosť definovať skupiny hesiel
- mať schopnosť šifrovať a uchovávať heslá v bezpečnej forme
- upozorňovať na dlho nezmenené heslá
- mať možnosť tvoriť potrebné vzťahy medzi používateľom a heslami

1.1.2 Nefunkčné požiadavky systému

Systém:

- musí udržiavať integritu dát
- musí byť navrhnutý s dôrazom na bezpečnosť.
- musí byť navrhnutý tak, aby bol výkonný a schopný zvládnuť zvýšenú záťaž.
- by mal obsahovať možnosť obnovy dát.
- musí byť dostupný 24/7 s minimálnym výpadkom služby
- by mal poskytovať intuitívne a používateľsky prívetivé rozhranie
- bude využívať relačný databázový systém MySQL
- bude využívať architektúru MVVC
- bude odolný voči SQL Injections

1.2 Kľúčové aktivity na správu a zabezpečenie systému

Pre vytvorenie stránky nám nestačia len funkčné a nefunkčné požiadavky. Je nutné si vytvoriť základnú funkcionalitu. To ako sa budú ukladať dáta, akým spôsobom sa budú šifrovať používateľské funkcie, prípadne aj bezpečnostné opatrenia, ktoré je vhodné napísať v návrhu. Medzi tieto základné časti by mali patriť nasledujúce funkcie

1.2.1 Šifrovanie dát v prenose a v úložisku

Šifrovanie dát zabezpečuje, že citlivé informácie, ako sú heslá a prihlasovacie údaje sú chránené pred neoprávneným prístupom. Šifrovanie zabezpečuje, že aj keď útočník získa prístup k dátam, nebude schopný ich prečítať bez správneho dešifrovacieho kľúča. Pri prenose dát cez internet alebo siete je dôležité zabezpečiť, aby boli dáta chránené pred odpočúvaním a úpravou útočníkmi. Šifrovanie dát v prenose zabezpečuje, že aj keď sú dáta zachytené útočníkom, nebude schopný ich prečítať. [1]

1.2.2 Ukladanie hesiel

Ukladanie hesiel je kritickým aspektom v oblasti IT bezpečnosti. Heslá sú citlivé informácie, ktoré chránia prístup k osobným a dôverným údajom. Preto je nevyhnutné ukladať ich tak, aby boli chránené pred neoprávneným prístupom. To znamená, že heslá by nemali byť uložené v čitateľnej podobe, ale v šifrovanej forme. Šifrovanie zabezpečuje, že aj keď útočník získa prístup k úložisku hesiel, nebude schopný prečítať samotné heslá.[2]

1.2.3 Registrácia a prihlasovanie používateľa

Registrácia používateľa je prvým krokom k získaniu prístupu k systému alebo službám. Počas registrácie používateľ zvyčajne poskytuje svoje osobné údaje, ako sú meno, e-mailová adresa, prípadne telefónne číslo a ďalšie relevantné informácie. Tieto údaje sa potom využívajú na vytvorenie účtu v systéme a umožnia systému lepšie identifikovať používateľa. Cieľom registrácie je zabezpečiť, aby používateľské účty boli správne vytvorené a zabezpečené pred neoprávneným prístupom.

Prihlasovanie používateľa umožňuje systému identifikovať jednotlivých používateľov na základe ich prihlasovacích údajov, ako sú užívateľské meno a heslo. Tento identifikátor umožňuje systému priradiť prístupové práva a personalizovať používateľský zážitok. Zároveň pomáha chrániť citlivé údaje pred neoprávneným prístupom. Po prihlásení používateľa sú následne záznamy o úpravách presnejšie, keďže môžeme získať potrebné údaje o ňom.

1.2.4 Resetovanie hesla

Používatelia môžu zabudnúť svoje heslo, čo im bráni v prístupe k systému. Resetovanie hesla im umožňuje obnoviť prístup a pokračovať v používaní systému. V prípade, že došlo k porušeniu bezpečnosti a útočník získal prístup k heslám používateľov, resetovanie hesiel je nevyhnutné na zabezpečenie bezpečnosti účtov. Umožní používateľovi získať naspäť svoj účet. [2]

1.2.5 Auditné záznamy a sledovanie činností

Auditné záznamy a sledovanie umožňuje monitorovať činnosti používateľov a identifikovať potenciálne bezpečnostné hrozby. Tieto záznamy môžu pomôcť odhaliť neoprávnený prístup, nezvyčajné správanie alebo pokusy o neoprávnené akcie. Tieto záznamy umožňujú sledovať, kto a kedy pristupoval k určitým údajom alebo vykonával určité operácie v systéme. Tým sa zvyšuje zodpovednosť používateľov za ich činnosti a pomáha vyšetrovať prípady, kde je potrebné zistiť, kto bol zodpovedný za určité udalosti [3]

1.2.6 Dvojfázová autentifikácia (2FA)

Dvojfázová autentifikácia poskytuje vyššiu úroveň bezpečnosti ako tradičná autentifikácia na základe jedného faktora. Dvojfázová autentifikácia pomáha prevencii zneužitia ukradnutých alebo vytrhnutých údajov, ako sú heslá. Aj keď útočník získa heslo používateľa, bude potrebné ďalšie overenie identity[4]

1.2.7 Vytváranie tímov

Vytváranie tímov je dôležitým aspektom pre bezpečné a efektívne získavanie citlivých dát v rámci daného tímu. Umožňuje centralizovanú správu hesiel pre rôzne skupiny používateľov v systéme. To znamená, že heslá môžu byť zoskupené podľa projektov, oddelení alebo funkčných skupín, čo uľahčuje ich správu a kontrolu. Administrátori a manažéri systému môžu definovať prístupové práva pre každý tím, čo zabezpečuje, že len používatelia s dostatočnými právami majú prístup k zdieľaným heslám

1.2.8 Zdieľanie hesiel

Zdieľanie hesiel vie byť vyriešené viacerými spôsobmi napríklad vytvorením virtuálnych trezorov. Prístup k heslám je možné získať iba po autentifikácii môže to byť prihlásením používateľa s prístupovými právami alebo vytvorením jednorazového kľúča na autorizáciu prístupu do trezoru.

1.2.9 Správa prístupových práv

Znižuje riziko neoprávneného prístupu k citlivým údajom. Umožňuje systému definovať a riadiť, kto má prístup k akým údajom, a zabezpečuje, že len oprávnení používatelia majú prístup k určitým funkciám v systéme.[5]

1.2.10 Hľadanie a filtrovanie hesiel

Umožňuje používateľom rýchle nájdenie potrebných údajov bez zbytočného času stráveného prezeraním celého zoznamu hesiel. To zvyšuje efektivitu práce a umožňuje rýchlejšie vykonanie úloh. Používatelia môžu rýchlo pristupovať k správnym heslám, čím sa minimalizuje riziko použitia nevhodných údajov.

1.2.11 Obrana voči SQL Injection

Je to zraniteľnosť, ktorá umožňuje útočníkovi zaviesť (injektovať) neželaný kód SQL do dotazov vykonávaných softvérom. SQL injection je jednou z najzávažnejších zraniteľností z dôvodu, že databázy, ku ktorým pristupujú webové aplikácie, často obsahujú veľmi citlivé informácie, ktoré majú pre útočníkov najvyššiu hodnotu. Preto majú útočníci veľký záujem dostať sa k týmto údajom.[6]

2 BEZPEČNÉ ZDIEĽANIE HESLA

Zdieľanie hesiel je stále častejším javom, stalo sa kľúčovým pre efektívnu spoluprácu a prístup k zdieľaným zdrojom. V kontexte súčasných bezpečnostných hrozieb a zákonov o ochrane údajov je nevyhnutné venovať pozornosť tomu, ako ukladáme a zdieľame prihlasovacie údaje. Pri radách, ktoré sa dávajú na tvorbu hesla ako sú dĺžky a špeciálne znaky je ťažké si zapamätať všetky heslá na všetky účty, ktoré môžeme mať. S rastúcim počtom platforiem, vznikajú rôzne systémy na zdieľanie a ukladanie hesiel.

2.1 Metódy šifrovania

Pre bezpečné zdieľanie hesla potrebujeme otvorený text ako môžu byť heslá alebo prihlasovacie údaje zašifrovať metódou, ktorá dokáže udržať pri zdieľaní dáta dostatočne skryté a nečitateľné. Pri týchto metódach musíme sledovať moderné techniky šifrovania, keďže pri niektorých metódach sa našli zraniteľnosti alebo prelomenia použitím výkonnej výpočetnej techniky. Je teda potrebné nájsť šifrovacie metódy, ktoré obstáli aj proti novým spôsobom útokov. Aj keď existuje symetrická šifrovacia metóda, ktorá je často používaná aj v modernej dobe. Na naše účely by boli vhodné metódy asymetrického šifrovania. Keďže pri zdieľaní hesiel sa dajú zašifrovať verejným kľúčom a následne vedia byť odšifrované súkromným kľúčom. Kľúče sa teda nemusia odosielať alebo iným spôsobom zdieľať, takže je nízka pravdepodobnosť kompromitácie pri zdieľaní.

2.1.1 AES (Advanced Encryption Standard)

V skratke, AES je symetrický typ šifrovania, pretože používa rovnaký kľúč na šifrovanie aj dešifrovanie údajov. Používa algoritmus SPN (substitučná permutačná sieť), ktorý používa viacero cyklov na šifrovanie údajov. Tieto šifrovacie cykly sú dôvodom nepreniknuteľnosti AES, pretože je ich príliš veľa na prelomenie. Aj keď sa dĺžka kľúča tejto šifrovacej metódy mení (128, 192 a 256-bitov), veľkosť bloku - 128 bitov (alebo 16 bajtov) - zostáva pevná. Prelomenie 128-bitového šifrovacieho kľúča AES môže trvať až 36 kvadriliónov rokov.[7]

2.1.2 RSA

RSA je asymetrický typ šifrovanie. Jeho bezpečnosť závisí od veľkosti kľúčov; dlhšie kľúče poskytujú silnejšie zabezpečenie, ale môžu ovplyvniť výkon. založený na náročnosti faktorizácie veľkých prvočísel, vďaka čomu je bezpečný proti súčasným klasickým počítačom.[7]

2.1.3 ECC

ECC využíva matematické operácie na bodoch definovaných na eliptických krivkách nad konečnými poľami. Tieto krivky majú špeciálne vlastnosti, ktoré umožňujú bezpečné kryptografické operácie. Kľúčovým aspektom ECC je schopnosť vykonávať operácie ako sčítanie, násobenie a delenie bodov na eliptických krivkách. Tieto operácie sú základom algoritmov pre šifrovanie. Dôvod, prečo je ECC zaujímavá možnosť, je jej schopnosť dosiahnuť vysokú úroveň bezpečnosti pri použití oveľa kratších kľúčov v porovnaní s inými algoritmami, ako je RSA.[7]

2.2 Návrh bezpečného zdieľania hesiel

Zdieľanie hesiel by malo byť dostupné pre každého používateľa. Menil by sa iba počet a presnosť zdieľaných dát. Zatiaľ čo admin by mohol mať možnosť zdieľať každé dostupné heslo. Používateľ by mal právo zdieľať heslá, ktoré sám vytvoril alebo mu bolo pridelené právo na ich zdieľanie. Heslá by mali spĺňať moderné požiadavky dĺžky a rôznorodosti taktiež by mali byť šifrované metódou, ktorá je rýchla a bezpečná. Použitie prístupu na základe práv je vhodný spôsob ako bezpečne môžeme zdieľať a zobrazovať heslá, keďže ak by používateľ nemal dostatočné práva tak nedokáže zmeniť ani vymazať heslá. Ak udržíme všetku správu dát a hesiel na platforme, zvýšime jej bezpečnosť a znížime šancu na únik dát.

2.2.1 Zabezpečenie a ukladanie hesiel

Heslá by mohli byť pridávané do skupín/trezorov, ktoré budú mať vlastné šifrovacie kľúče. Čím by sa dokázalo znížiť percento zasiahnutých dát len na danú skupinu, keďže kľúče z jednej skupiny by nebolo možné použiť na ďalšiu skupinu pre odšifrovanie dát. Jeden z kľúčov by mohol byť skrytý do obrázku pomocou steganografie, vložený do databázy a pri zobrazení na platforme by bol komprimovaný čo by znemožnilo získanie kľúča z obrázku.

Zabezpečenie hesiel sa spolieha taktiež na vytvorenie trezorov, ktoré slúžia ako bezpečné úložisko pre citlivé údaje. Každý trezor je kategória alebo projekt a obsahuje rôzne typy hesiel a potrebných informácií

2.2.2 Zdieľanie hesiel

Návrh na bezpečné zdieľanie hesiel by spočíval vo vytvorení trezorov, do ktorých by sa ukladali heslá a iné informácie na prípadné prihlásenie alebo poznámky pre ľudí v tíme. Následne zdieľanie by bolo vyriešené pridávaním práv používateľom vidieť dané heslá. Podľa

toho ako a kým mu boli pridelené práva by dokázal aj spravovať údaje hesiel. Takto vytvorené trezory môžu mať aj vlastný list záznamov zmien, vytvorení jednorazových prístupov odoberaní a pridávaní používateľov, ktorý mali prístup k daným dátam. Čo by zjednodušilo sledovanie a auditáciu údajov.

2.2.3 Prideľovanie a odoberanie prístupu k heslám

Správa prístupu k heslám je proces riadenia prístupových práv užívateľov k citlivým údajom uloženým v systéme. Jeho cieľom je zabezpečiť, aby len používatelia s dostatočnými právami mali prístup k heslám, ktoré potrebujú. Prideľovanie a odoberanie prístupu by bolo možné pre používateľov, ktorý by mali na to priradené práva. Admin by mal možnosť pridať a odobrať akýkoľvek prístup. Používateľ by mohol pridať a odobrať prístup ak by mal na to právo a to iba do trezorov, do ktorých ma on sám prístup.

2.2.4 Správa hesiel

Správa hesiel je taktiež prvkom zabezpečenia citlivých údajov. Keďže správa utvrdzuje bezpečnosť dát. Pri správe sa vie zaznamenávať každý pohyb a zmena hesiel. Taktiež pri správe hesiel je potrebné myslieť na bezpečnosť zmien aby neautentifikovaný používateľ mohol meniť heslá iným užívateľom takže by mala byť dostupná hlavne pre administrátorov a používateľov, ktorí si chcú spravovať vlastné hesla ako je vytvorenie, zmazanie alebo editácia.

2.3 Kontrola prístupových práv a monitorovanie činnosti používateľov

Táto kontrola zabezpečuje, že len používatelia s určitými právami majú prístup k citlivým informáciám. Monitorovanie činnosti vie vopred predpovedať, či sa snaží niekto dostať neoprávnene na platformu a tým umožňuje rýchlu reakciu na bezpečnostné incidenty a ich vyšetrovanie. Identifikácia nezvyčajnej alebo podozrivej činnosti môže pomôcť platforme minimalizovať škody a zamedziť šíreniu hrozieb. Celkovým cieľom je zabezpečiť, aby prístupové práva boli riadené efektívne a transparentne, a aby sa zabezpečilo, že všetky činnosti užívateľov sú sledované a môžu byť analyzované pre účely bezpečnosti a auditu .V rámci kontroly prístupových práv existuje niekoľko metód, medzi ktoré patrí model riadenia prístupových práv na základe rolí (RBAC) a prístupové kontroly založené na atribútoch (ABAC) [5][8]

2.3.1 Riadenie prístupových práv na základe rolí (RBAC)

Keďže používateľom sa neprideľujú oprávnenia priamo, ale získavajú ich len prostredníctvom svojej roly, správa práv jednotlivých používateľov sa stáva otázkou jednoduchého priradenia príslušných rolí k používateľskému účtu; to zjednodušuje bežné operácie, ako je pridanie používateľa alebo zmena jeho oddelenia. Môžu sa použiť aj obmedzenia, roly sa môžu kombinovať v hierarchii. Tento prístup umožňuje jednoduchšiu a škálovateľnejšiu správu prístupových práv a zvyšuje bezpečnosť prostredníctvom presných a konzistentných prístupových kontrol.[5]

2.3.2 Prístupové kontroly založené na atribútoch (ABAC)

Model riadenia prístupových práv, ktorý priraďuje oprávnenia na základe atribútov používateľa, zdroja a prostredia. Tento model umožňuje flexibilnejšie riadenie prístupu na základe širokej škály faktorov, ako sú napríklad používateľské údaje, čas prístupu, poloha a ďalšie.

Keď sa na platformu prihlásia nové subjekty, pravidlá a objekty sa nemusia meniť, pokiaľ má subjekt pridelené atribúty potrebné na prístup k požadovaným objektom. Toto prispôsobenie sa externému používateľovi je jednou z hlavných výhod používania systému ABAC[8]

2.4 Zaznamenávanie činnosti používateľov

Ak sú užívatelia oboznámený o tom, že zmeny a správanie je na platforme sledované je väčšia pravdepodobnosť toho, že užívatelia budú viac obozretný o tom kde a ako zdieľajú dáta mimo platformu, na ktorej sa práve nachádzajú. Môže to mať aj presne opačný účinok na užívateľov. Prestanú dôverovať dostatočne aplikácií a niektorým sa nemusí páčiť to, že je sledovaný každý jeho krok. Používatelia by mali byť primerane informovaní o tom, ako a prečo sú ich činnosti sledované a poskytnú zrozumiteľné informácie o tom, aké údaje sa zbierajú a na čo sa používajú to môže zmierniť obavy a zvýšiť dôveru. Ďalšie riešenie tohto problému, je nechať užívateľa zvoliť si úroveň toho, aké všetky kroky budú sledované, s tým, že základne záznamy ako je kopírovanie a prístup k heslám, pridávanie, úprava a odstraňovanie povolení by boli povinné. [3]

3 MANAŽMENT ZDIEĽANIA HESIEL

Na začiatok tejto časti je potrebné sa rozhodnúť na akej technológií chceme našu platformu stavať. Pre zdieľanie hesiel bude potrebné vytvoriť aplikáciu, na ktorej vieme riešiť implementáciu do databázy a spojenie cez internet. Keďže väčšina ľudí, ktorí to potrebujú, to potrebujú na bezpečné zdieľanie hesiel tímom, ktoré nie sú napojené na hlavnú sieť, v ktorej by si bezpečne mohli zasielať heslá, ale tímom, ktoré sú v vzdialených častiach alebo na služobnej ceste. Preto sme si zvolili aplikáciu na technológií .NET C# s architektúrou MVC (Model-View-Controller) s dodatkom ViewModel-u. Niekedy sa táto architektúra môže nazývať MVVC (Model-View-Viewmodel-Controller), ViewModel používame na prenos dát medzi View a Modelom a autentifikáciu zadaných údajov.

3.1 Technológie na vytvorenie stránky

- MySQL

Relačný databázový systém, ktorý ponúka robustné možnosti správy dát. Dáta sú organizované do tabuliek, ktoré majú definované vzťahy medzi sebou. Tento model umožňuje efektívne ukladanie a manipuláciu s dátami pomocou SQL. MySQL je schopný spravovať veľké objemy dát a je škálovateľný pre rôzne úrovne zaťaženia. Vysoký výkon a spoľahlivosť robia z MySQL skvelú voľbu pre webové aplikácie [9]

- MVVC architektúra

Model-View-Viewmodel-Controller, v tejto verzií je jedna zmena oproti MVC a to je pridanie viewmodelu. Tento vzor je často používaný v softvérovom vývoji pre tvorbu aplikácií. Controller prenesie dáta z modelu do viewmodelu a view následne čerpá dáta z tohto novovytvoreného viewmodelu. To dokáže oddeliť view od modelu, čím sa zvýši bezpečnosť dát a dostanú sa iba dáta používateľa ku ktorým ma podľa vytvorených vzťahov právo. [10]

- Microsoft Visual Studio

Integrované vývojové prostredie (IDE) vyvinuté spoločnosťou Microsoft, ktoré poskytuje vývojárom nástroje a funkcie na vytváranie softvérových aplikácií. Poskytuje rôzne integrované nástroje, ktoré zjednodušujú proces vývoja softvéru. Medzi tieto nástroje patrí editor kódu s funkciami ako je syntax highlighting, IntelliSense (automatické dopĺňanie kódu), debugovanie, profilovanie výkonu [11]

- .NET

Vývojový rámec vyvinutý spoločnosťou Microsoft, ktorý poskytuje prostredie pre vytváranie a vykonávanie softvérových aplikácií. .NET podporuje viacero programovacích jazykov. Taktiež poskytuje širokú škálu nástrojov a knižníc poskytovaných v rámci .NET ekosystému. Zároveň podporuje tvorbu rôznych aplikácií. [12]

- C#

Objektovo orientovaný programovací jazyk vyvinutý spoločnosťou Microsoft. Podporuje koncepty ako triedy, objekty, dedičnosť taktiež poskytuje rôzne funkcie na ochranu aplikácií pred chybami a zraniteľnosťami. Patrí sem napríklad kontrola prístupu a rôzne výnimky. C# má veľkú a aktívnu komunitu vývojárov, z tohto dôvodu je dostupná aj pre začiatočníkov aj pokročilých keďže nové metódy v IT sú pomerne rýchle implementované do využiteľných knižníc. [13]

- Microsoft Entity Framework

Objektovo relačné mapovanie (ORM) pre .NET, ktoré umožňuje vývojárom definovať dátový model pomocou objektových tried a atribútov taktiež poskytuje funkcie pre automatické migrácie schémy databázy na základe zmien v dátovom modeli. Tieto migrácie umožňujú vývojárom ľahko aktualizovať štruktúru databázy a udržiavať konzistentnosť medzi dátovým modelom a reálnou databázou. [14]

- Bootstrap

Framework pre vývoj responzívnych a moderných webových stránok a aplikácií. Obsahuje flexibilný mriežkový (grid) systém, ktorý umožňuje jednoduché usporiadanie obsahu na webovej stránke pomocou stĺpcov a riadkov. Bootstrap obsahuje širokú škálu štýlových komponentov, ako sú tlačidlá, formuláre a ikony. Je to skvelý pomocník pri tvorbe webových stránok, z dôvodu jednoduchosti implementovania štýlov na určité časti stránok. [15]

- JavaScript

Vysokoúrovňový programovací jazyk pre tvorbu interaktívnych a dynamických webových stránok. Môže byť vložený priamo do HTML kódu webovej stránky alebo použitý ako externý súbor. Umožňuje vývojárom pridať rôzne interaktívne prvky a funkcionality do webových stránok, ako sú animácie a dynamické zmeny obsahu. Reaguje na udalosti a akcie používateľa, ako sú kliknutia, stlačenie kláves a ďalšie. Tieto udalosti môžu spúšťať funkcie a zmeny. [16]

- CSS

Jazyk používaný na definovanie vzhľadu a formátovania webových stránok, umožňuje vývojárom meniť vzhľad webových stránok bez zmeny samotného obsahu, čo zjednodušuje údržbu a zlepšuje flexibilitu a modularitu kódu.

3.2 Funkcie na správu a organizáciu zdieľaných hesiel

Organizácia a správa hesiel bude vyriešená „trezormi“ kde užívateľ alebo admin vie pridávať heslá. Na tieto trezory budú vytvorené vzťahy medzi používateľom a trezorom. Ak je vzťah vytvorený, používateľ má prístup k heslám. Odobratím alebo zmenami v týchto vzťahoch je prístup odopretý.

3.2.1 Manažment trezorov

Používateľ vie vytvoriť nový trezor, ktorému nastaví meno a popis pre lepšiu identifikáciu o aký trezor sa jedná. Do popisu sa môžu pridať rôzne poznámky k postupom na prihlásenie do platformy alebo admin môže pridať do nich informácie a upozornenia o potrebe zmeny hesla. Vytvoriť funkciu, ktorá bude sledovať, kedy bolo heslo naposledy zmenené/upravené ak to presiahne časový limit, v danom popise sa zobrazí upozornenie. Tieto upozornenia by sa nedali vymazať bežným používateľom iba ak by zmenil heslo, tým by sa splnila podmienka a upozornenie by sa odstránilo.

Používateľovi by bolo umožnené pridať aj fotku, na prispôsobenie daného trezoru a ešte jednoduchšieho hľadania potrebného trezoru na získanie hesla.

3.2.2 Manažment používateľov

Administrátor alebo oprávnený používateľ by mal mať možnosť vytvoriť nový používateľský účet, priradiť rôzne úrovne prístupových práv, zobraziť a aktualizovať informácie o používateľoch, ako sú užívateľské mená, heslá, e-mailové adresy, to zahŕňa aj možnosť vymazania účtu používateľa.

3.2.3 Správa rolí používateľov

Role používateľov, budú rozdelené nasledovne: Administrátor, Manažér, Užívateľ

Administrátor bude mať oprávnenia na správu celkového systému to zahŕňa vytváranie, upravovanie a odstraňovanie účtov používateľov taktiež úpravy spojení (používateľ – trezor), monitorovanie činnosti, prístup k záznamom v systéme. Úprava práv používateľov

napríklad zmena používateľa na manažéra alebo naopak. Vytváranie záznamov na audit v prípade nezvyčajného alebo podozrivého správania, bezpečnostných rizík a únikov dát. Administrátor bude mať aj právo vytvoriť ďalšieho administrátora s rovnakými oprávneniami.

Manažér bude mať oprávnenia na úpravu hesiel v trezoroch. Sledovať aktivitu vytvorenú pre trezory, v ktorých má prístup. Prípadne zmeniť člena v tíme (trezore) na manažéra v neprítomnosti. Zabezpečenie a kontrolu dát v trezore, upozornenie používateľa na zmenu hesla.

Používateľ bude mať obmedzené oprávnenia, môže pristupovať len k obmedzenému množstvu funkcií a údajov, získavať prístup k zdieľaným heslám a údajom, vytvárať a upravovať svoje vlastné heslá a pristupovať k záznamom svojej vlastnej aktivity.

3.2.4 Prideľovanie a odstraňovanie prístupu

Oprávnenie na prideľovanie a odstraňovanie prístupu budú mať role Admin a Manažér. Taktiež môžu pridať oprávnenie používateľovi na zmenu konkrétnych hesiel a údajov v trezore. Administrátor bude môcť prideliť tieto oprávnenia každému používateľovi na platforme, Manažér len používateľom v trezoroch kde má povolenia. Tieto funkcie umožnia administrátorovi a manažérovi efektívne spravovať prístupové práva a zabezpečiť, že len oprávnení používatelia majú prístup k citlivým údajom

3.2.5 Manažment hesiel

Vytváranie a odstraňovanie hesiel bude prístupné každému na platforme, ale podľa role budú na tom obmedzenia. Používateľ bude môcť vytvárať hesla v trezoroch kde bude mať prístup. Nim vytvorené heslá môže upravovať a vymazať iné heslá bude môcť len prečítať. Manažér bude mať možnosť správu hesiel v trezoroch kde má prístup. Administrátor má možnosť spravovať heslá v trezoroch. Používateľské heslá a údaje v prípade zabudnutia alebo úniku dát.

3.3 Správa hesiel a šifrovanie hesiel

3.3.1 Obnovenie hesla

Obnova hesla je dôležitou funkciou na našej platforme. Umožňuje používateľom získať prístup k ich účtom v prípade zabudnutia alebo kompromitácie hesla.

Existuje mnoho spôsobov, ktorým je možné funkčne implementovať obnovu hesla. Odoslanie overovacieho kódu na e-mail alebo telefón používateľa, ktorý sa zadá do formuláru na stránke, čim sa mu pridá povolenie na zmenu hesla. Ďalšia možnosť je odpovedať na kontrolnú otázku alebo overenie dvojfaktorovou autentifikáciou. Posledný spôsob je ten najbezpečnejší, keďže jediná kópia kódu je v aplikácií a každých 30 sekúnd sa mení.

3.3.2 História zmien hesiel

Umožňuje sledovať a monitorovať všetky zmeny hesiel v systéme. Táto história obsahuje záznamy o starých a nových hodnotách hesiel, dátumoch zmien a informácie o autoroch zmien. Uchovávanie histórie zmien hesiel pomáha v prípade potreby auditu, vyšetrovania bezpečnostných incidentov a dodržiavania regulatívnych požiadaviek. K tejto histórií bude mať prístup len administrátor platformy.

3.3.3 Šifrovanie hesla

Pri šifrovaní hesiel v trezore sa bude používať kombinácia RSA šifry a steganografie. Pri vytvorení trezoru sa vytvorí asymetrický pár kľúčov. Verejný kľúč sa uloží do databázy a bude slúžiť na šifrovanie hesiel. Pomocou LSB steganografie uložíme súkromný kľúč do obrázku. Pri zobrazovaní na platforme bude obrázok komprimovaný a zmenená veľkosť, čiže bude nemožné získať súkromný kľúč z obrázku na stránke. Pri zmene obrázku v trezore sa kľúč extrahuje a uloží do nového obrázku. [7][17]

Užívateľské heslá budú iba zahashované pomocou Argon2id[18] . Z dôvodu rýchlosti prihlasovania, keďže táto hashovacia metóda je dostatočne bezpečná. Bude k tomu pridaná hodnota neúspešných prihlásení. Po prekročení tohto limitu, bude nemožné sa prihlásiť, kým admin neresetuje danú hodnotu. Vyriešené by to mohlo byť odoslaním žiadosti z e-mailu zablokovaného účtu. Po overení admin zmení hodnotu na 0 a bude možné sa opäť prihlásiť.

II. PRAKTICKÁ ČÁST

4 IMPLEMENTÁCIA SYSTÉMU

Po analýze možných rizík a útokov na systém, spracovaní návrhu bezpečného zdieľania hesiel a návrhu funkcií sa pokúsime o implementáciu systému v testovacom prostredí. Ako testovacie prostredie sme si zvolili aplikáciu Microsoft Visual Studio, kvôli predošlým skúsenostiam v tomto prostredí. Budeme používať .NET C# MVVC architektúru s používaním Entity Framework-u.

Základný projekt sme si pomenovali PassHive na jednoduchšiu orientáciu vo vytvorených častiach. Následne sme vytvorili 3 zložky do ktorých budeme umiestňovať pomocné funkcie, metódy, databázové entity a základné ustanovenia priečinkov, s ktorými budeme pracovať pri tvorbe platformy.

4.1 Architektúra systému na ukladanie a zdieľanie hesiel

4.1.1 Model

Model predstavuje dáta, s ktorými aplikácia pracuje. Môžu to byť napríklad informácie získané z databázy, súbory uložené na disku alebo iné zdroje dát. Ale okrem dát zahŕňa model aj logiku aplikácie. Táto logika definuje, ako sú dáta manipulované a aké operácie sa s nimi vykonávajú. Môže to zahŕňať validáciu údajov, výpočty, transformácie dát atď. Model by mal byť nezávislý na používateľskom rozhraní (View) a riadiacom mechanizme (Controller). To znamená, že by nemal obsahovať žiadnu logiku, ktorá by sa týkala toho, ako sú dáta zobrazované alebo ako používateľ interaguje s aplikáciou.

K modelu je možné dopísať, že vo webových aplikáciách a iných implementáciách sa využíva aj ViewModel ten je sprostredkovateľom medzi Modelom a View. Jeho hlavnou úlohou je poskytovať dáta a spracovávať akcie vykonané používateľom, aby sa tieto akcie mohli odrážať v modeli.

4.1.2 View

View je časť aplikácie, ktorá je zodpovedná za zobrazovanie dát používateľovi. To môže zahŕňať všetko od jednoduchých textových informácií až po komplexné grafické rozhrania.

View nie je len pasívny zobrazovač údajov. Často tiež obsahuje nejakú úroveň logiky na formátovanie a zobrazenie dát. Napríklad, ak máme dátum uložený vo formáte času, View by sa mohol starať o jeho konverziu na čitateľný formát. View by mal byť nezávislý na

dátach (Model) a riadiacom mechanizme (Controller). To znamená, že by nemal priamo manipulovať s dátami alebo vykonávať akcie, ktoré ovplyvňujú stav aplikácie.

4.1.3 Controller

Controller je komponent, ktorý prijíma vstupy od používateľa a reaguje na ne vykonávaním príslušných akcií. Jeho hlavnou úlohou je spracovať vstupy od používateľa a riadiť tok dát medzi Modelom a View, k tomuto mu dopomáha aj ViewModel, ktorý bol predošle spomenutý. To znamená, že získava požiadavky od používateľa, komunikuje s modelom na získanie alebo aktualizáciu dát, ktoré predá Viewmodelu a potom aktualizuje View, aby zobrazilo aktuálny stav aplikácie. Controller by mal byť tenkým prízrakom, ktorý len koordinuje akcie, ale neobsahuje veľa autorizačnej ani zobrazovacej logiky. Jeho úlohou je skôr sprostredkovávať a koordinovať komunikáciu.

4.2 PassHive

Tento základný webový priečinok má v sebe všetky stránky, ktoré sa zobrazujú, pomocné metódy na úpravu funkcionality a vzhľadu webovej stránky (CSS, JavaScript a Bootstrap súbory). Obsahuje aj konfiguračné súbory, v ktorých je to ako sa stránka spustí, s akými nastaveniami a hodnotami, definovanie smerovania riadiaceho mechanizmu na iné stránky. V tomto priečinku sú aj takzvané oblasti, do ktorých sa stránka presúva. Tieto oblasti sú vytvorené z dôvodu bezpečnosti a čiastočného oddelenia administratívnej stránky od tej používateľskej.

Je to smerovaním oddelená časť od hlavnej stránky. Tieto oblasti sa tvoria z viacerých dôvodov jeden z nich je pri RBAC, to že do určitých oblastí sa dostane len používateľ s dostatočným povolením ako je Manager alebo Admin. Ďalší dôvod je rozdelenie funkcií. Pre jednoduchosť si stále chceme pomenovať stránky podobným spôsobom aby sme vedeli s akou stránkou práve pracujeme. To by nebolo možné s jedným hlavným controller-om, ktorý by musel spájať každú funkciu, ktorá by bola na platforme.

4.3 PassHive.Application

Táto skupina tried stojí za pomocnými metódami, hlavne na dostávanie veci do a z databázy do iných častí stránky. Čiže používateľ by nemal mať nikdy priamy kontakt s databázou. Taktiež uchováva ViewModel-y potrebné na bezproblémovú validáciu dát a komunikáciu medzi časťami.

4.4 PassHive.Domain

V tejto skupine tried sú entity potrebné na správne fungovanie stránky. Pri využívaní Entity Framework sú entity ako tabuľky databázy, ktorým sú nastavené parametre, z ktorých sa vytvorí databáza, ktorá je následne využívaná systémom. Pri zmene entít je potrebné vytvoriť migráciu. Migrácia vytvorí súbor, ktorý sa spustí a potrebné dáta sa vložia do databázy

4.5 Passhive.Infrastructure

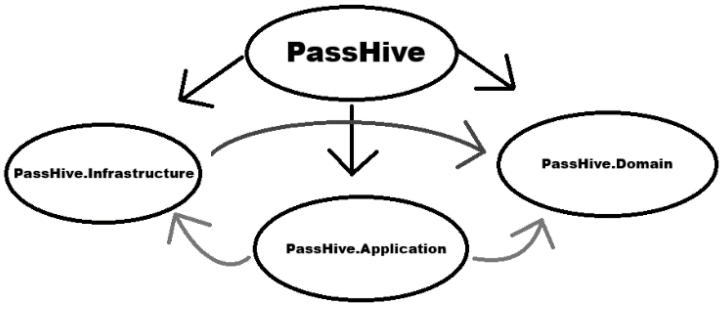
Zahŕňa databázový kontext na vytvorenie databázy a súbor na inicializáciu databázy, prípadne dát, ktoré budú vložené do databázy hneď po vytvorení. Napríklad ID rolí, zakladajúce účty ako je admin a manager. Taktiež sa v nej ukladajú všetky vytvorené migrácia na prípadný roll-back po zlej aktualizácií databázy.

4.6 Implementácia navrhnutého systému do testovacieho prostredia

Vytvorenie týchto priečinkov nám zabezpečí dostatočnú orientáciu v kóde ak by na tom pracovalo aj viac ľudí, keďže každý z týchto priečinkov obsahuje špecifické súbory. Ak by bol kód implementovaný do iného priečinku je možné, že by nemal dostatočné zdroje a nefungoval by. Na identifikáciu používateľa a správu účtov budeme používať balíček ASP.NET.Identity [19].

4.6.1 Prepojenie priečinkov

Jednotlivé priečinky musíme medzi sebou ešte prepojiť. Keďže pri programovaní funkcií potrebujeme názvy, dáta a pomocné funkcie na to aby sme vedeli danú funkciu naprogramovať do plnofunkčného stavu. Priečinok PassHive bude mať závislosť na každom z ostatných troch skupinách tried. PassHive.Application bude mať vytvorené závislosti na PassHive.Domain a PassHive.Infrastructure. PassHive.Infrastructure bude závislé na PassHive.Domain a PassHive.Domain nebude mať vytvorenú žiadnu závislosť.



Obrázok 1. Diagram prepojení

4.6.2 Entity

Entity boli vytvorené ako prvé keďže od nich závisela stavba funkcií. Do akých entít (tabuliek) budeme pridávať údaje pri tvorení nového trezoru alebo pridaní hesla do trezoru. Entity majú nasledovnú štruktúru. Väčšina z nich dedí z „Entity<int>“, je to entita, ktorá zabezpečuje, že každý nový vytvorený riadok v tabuľke je nový a má vlastný jedinečný identifikátor v danej tabuľke. Namespace v každej z tried je tam z dôvodu identifikácie kde je súbor uložený a akú cestu musia iné programy prístupnú, na získanie dát z tejto triedy.

Entita Credential má v sebe základné údaje potrebné na prihlasovanie sa do iných aplikácií a platforiem. Pri tvorbe hesla sa uloží aj Identifikátor trezoru v ktorom je heslo uložené:

```
namespace PassHive.Domain.Entities
{
    public class Credential : Entity<int>
    {
        public string? Username { get; set; }
        public string? Password { get; set; }
        public string? Website { get; set; }
        public int VaultId { get; set; }
    }
}
```

Entita UserVault bude využívaná na sledovanie prístupov pomocou identifikátorov trezoru a užívateľa taktiež je tam vytvorený vzťah na Vault ako prvok:

```
namespace PassHive.Domain.Entities
{
    public class UserVault
    {
        public int VaultId { get; set; }
        public Vault? Vault { get; set; }
        public int UserId { get; set; }
    }
}
```

Entita Vault uchováva väčšinu potrebných informácií na sledovanie. Taktiež má v sebe aj 2 definície typu ICollection, ktoré nie sú potrebné na tvorbu databázy, ale určujú vzťah a prístup k manipulácií s dátami. Pri tvorbe nového trezoru sú využité body Required – je potrebný údaj na vytvorenie objeku alebo NotMapped - Označuje, že vlastnosť alebo trieda by mala byť vylúčená z mapovania databázy Keďže na správne vytvorenie trezoru je potrebné meno a obrázok. Obrázok je potrebný na zašifrovanie kľúča pomocou steganografie.

```
namespace PassHive.Domain.Entities
{
    public class Vault : Entity<int>
    {
        [Required(ErrorMessage = "Názov a obrázok je povinný")]
        public required string Name { get; set; }
        public string? Description { get; set; }
        public string? ImageSrc { get; set; }
        public string? PublicKey { get; set; }
        [NotMapped]
        [FileContent("image")]
        public IFormFile? Image { get; set; }

        public ICollection<Credential>? Credentials { get; set; }
        public ICollection<UserVault>? UserVaults { get; set; }
    }
}
```

V entite IUser sú definované základne údaje potrebné na tvorbu a sledovanie účtu taktiež je definovaný v pod-súbore Interfaces z dôvodu využívania balíčku ASP.NET.Identity:

```
namespace PassHive.Domain.Entities.Interfaces
{
    public interface IUser
    {
        public int Id { get; set; }
        public string? UserName { get; set; }
        public string? Email { get; set; }
        public string? PhoneNumber { get; set; }
        public string? FirstName { get; set; }
        public string? LastName { get; set; }
    }
}
```

V Passhive.Infrastructure.Idenity je ešte dodatočná entita na používateľa ktorá dedí z IdentityUser, ktorý je definovaný v ASP.NET.Identity a vytvorenej IUser entity, zároveň je tam vytvorená trieda pre navigáciu používateľa a uservault-u.

```
namespace PassHive.Infrastructure.Identity
{
    public class ApplicationUser : IdentityUser<int>, IUser
    {
        public virtual string? FirstName { get; set; }
        public virtual string? LastName { get; set; }

        public ICollection<UserVault>? VaultUsers { get; set; }
    }

    public class UserVaults : UserVault
    {
    }
}
```

Pre vytvorenie rolí mám použitú enumeráciu a triedu ktorá z nej dedí a pri vytvorení databázy pridá hodnoty. Táto entita dedí taktiež z IdentityRole v balíčku ASP.NET.Identity.

```
namespace PassHive.Infrastructure.Identity.Enums
{
    public enum Roles
    {
        Admin,
        Manager,
        Customer
    }
}

namespace PassHive.Infrastructure.Identity
{
    public class Role : IdentityRole<int>
    {
        public Role(string role) : base(role)
        {
        }

        public Role() : base()
        {
        }
    }
}
```

4.6.3 Oblasti

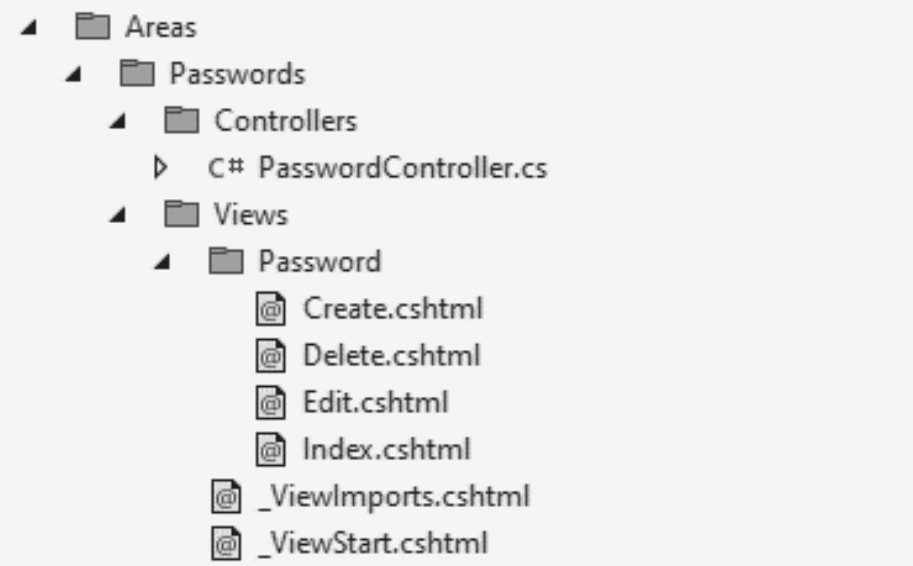
Vytvorili sme 4 oblasti na rozdelenie stránky. Passwords, UserPermissions, Users a Vaults. Každá z nich má funkcie na riadenie, smerovanie a modifikáciu pre vytvorenú entitu. Ich zdrojové súbory vyzerajú takto. Majú podobnú štruktúru kde v zložke Controllers by sa nachádzali všetky súbory na riadenie stránky v našom prípade je to len jedna trieda. V zložke Views sú uložené všetky stránky, na ktoré sa cez daný Controller vieme dostať a modifikovať cez neho údaje do a z databázy.

```
@{Layout = "_Layout";}
```

Tabuľka 1. ViewImport.cshtml pre oblasť Passwords

```
@using PassHive
@using PassHive.Models
@using PassHive.Domain.Entities
@using PassHive.Application.ViewModels
@using PassHive.Application.Implementation
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

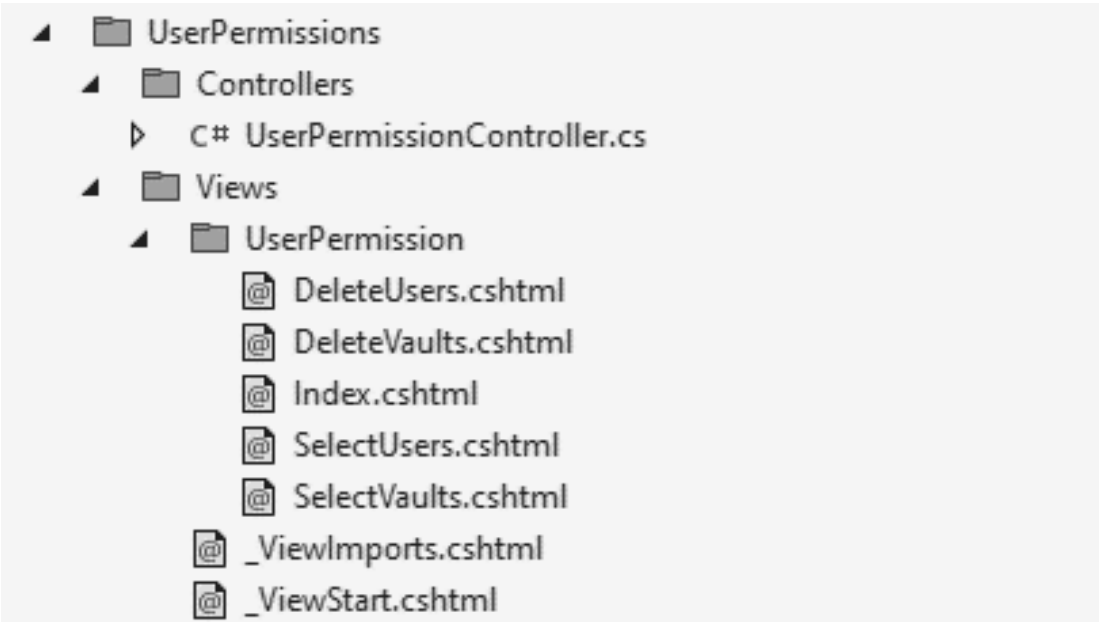
_ViewImports.cshtml má definované používané cesty k dátam, ktoré využíva. V _ViewStart je jednoduchý kód na to aby sa zobrazovalo menu



Obrázok 2. Zložka Passwords

Tabuľka 2. ViewImport.cshtml pre oblasť UserPermissions

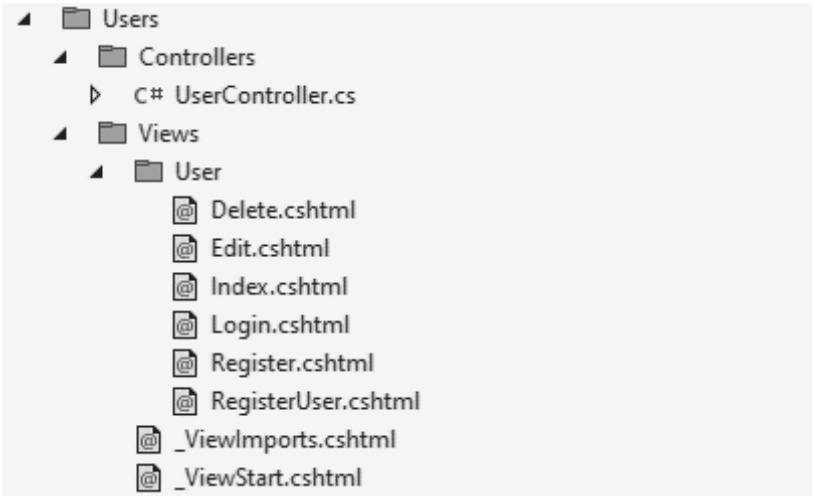
```
@using PassHive
@using PassHive.Models
@using PassHive.Domain.Entities
@using PassHive.Application.ViewModels
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```



Obrázok 3. Zložka UserPermissions

Tabuľka 3. ViewImport pre oblasť Users

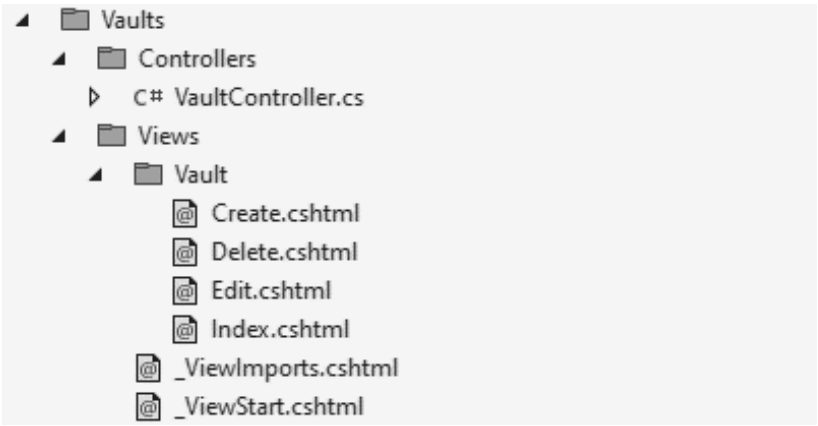
```
@using PassHive
@using PassHive.Models
@using PassHive.Domain.Entities
@using PassHive.Application.ViewModels
@using PassHive.Infrastructure.Identity;
@using PassHive.Infrastructure.Identity.Enums;
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```



Obrázok 4. Zložka Users

Tabuľka 4. ViewImport pre oblasť Vaults

```
@using PassHive
@using PassHive.Models
@using PassHive.Domain.Entities
@using PassHive.Application.ViewModels
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

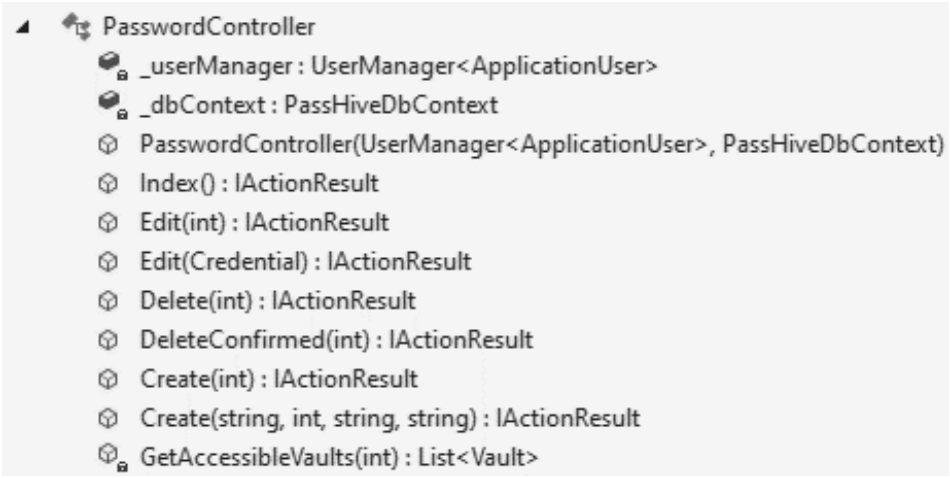


Obrázok 5. Zložka Vaults

4.7 Metódy, funkcie, viewmodely

V každej z týchto oblastí je Controller, ktorý riadi funkcionalitu v svojej oblasti. Controller využíva pomocné funkcie zo zložky PassHive.Application. Tieto funkcie sú následné definované v controlleri a používane vo funkciách taktiež z tejto zložky sú používane viewmodely na predávanie dát z modelu do view. Funkcie, ktoré sú v používané pre každý z controllerov. Značka ‚[Authorize]' bola pridaná do miest kde prístup bez prihlásenia je nemožný, to zaručí, že používateľ bude prihlásený do účtov, ku ktorým má prístup a sú v databáze, týmto spôsob je zabránené náhodnému alebo cielenému prístupu.

4.7.1 PasswordController



Obrázok 6. Metódy v PasswordController

- Index()

Ak je používateľ admin alebo manager, táto metóda načíta všetky trezory a heslá v nich. Ak je to používateľ, tak načíta iba trezory a heslá ku ktorým ma používateľ prístup.

- Edit() (GET)

Táto funkcia získa Id zvoleného hesla a odošle ho do View. Kde je možné ho upravovať.

- Edit() (POST)

Po upravení hesla sa v tejto funkcií uplatnia úpravy a používateľa presmeruje na index stránku kde uvidí vytvorenú zmenu.

- Delete()(GET)

Rovnako ako Edit() získa Id hesla zvoleného na vymazanie a odošle ho do View Delete kde používateľ potvrdí, že chce vymazať heslo.

- DeleteConfirmed()(POST)

Funkcia na potvrdenie odstránenia, heslo vymaže z databázy až po potvrdení používateľom. A controller ho presmeruje na index stránku.

- Create()(GET)

Táto funkcia získa aktuálneho používateľa, následne využije identifikátor používateľa a za pomoci tohto Id získa všetky dostupné trezory pre používateľa, ktoré sa následne zobrazia v liste, z ktorého si používateľ môže vybrať do akého trezoru chce pridať heslo.

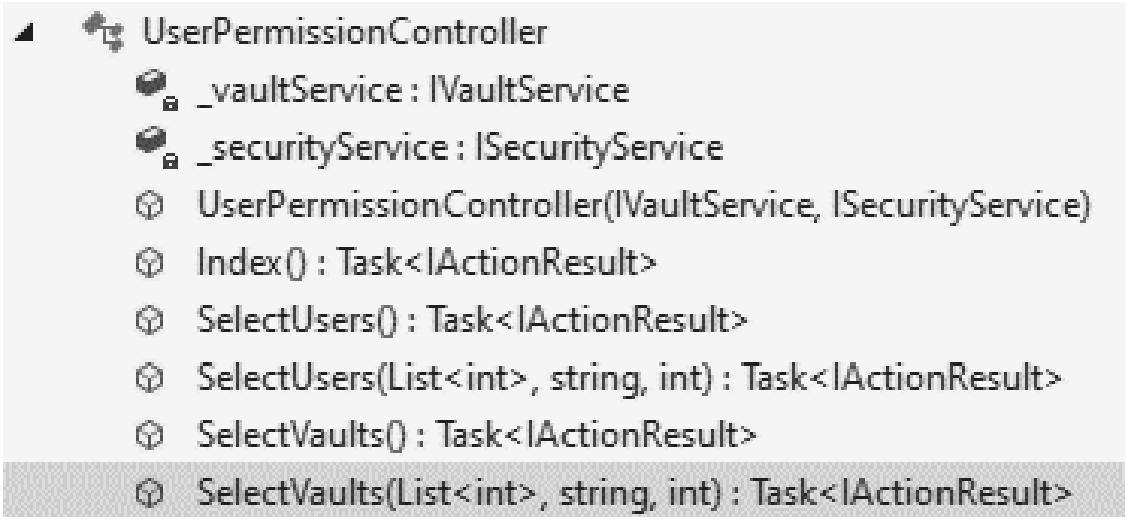
- Create()(POST)

Po získaní aktuálneho používateľa a jeho identifikátoru sa pozrie či má používateľ prístup k trezoru. Ak nemá zakáže mu pridanie hesla a prepošle ho na Index stránku. Po autorizácií či sú polia správne vyplnené a nie sú prázdne vytvorí heslo a uloží ho do databázy.

- GetAccessibleVaults(int)

Pomocná privátna funkcia na získanie dostupných trezorov pre používateľa.

4.7.2 UserPermissionController



Obrázok 7. Metódy v UserPermissionController

- Index()

Asynchrónne načíta všetkých používateľov a trezory, prevedie načítané údaje do ViewModelov, vytvorí kombinovaný ViewModel obsahujúci používateľov a trezory nakoniec odošle tento ViewModel do View.

- SelectUsers() (GET)

Keď si používateľ vyberie používateľa na pridanie prístupov tak získa identifikátor vybraného používateľa a uloží ho, asynchrónne načíta všetky trezory, konvertuje údaje trezorov do ViewModelu a odovzdá list týchto ViewModelov do View.

- SelectUsers() (POST)

Táto funkcia sa vykoná po odoslaní formulára na stránke SelectUsers. Prijíma vybrané trezory a ID používateľa, priradí používateľa k vybraným trezorom a potom presmeruje na index UserPermission , ak nie sú vybrané žiadne trezory vráti sa na SelectUsers.

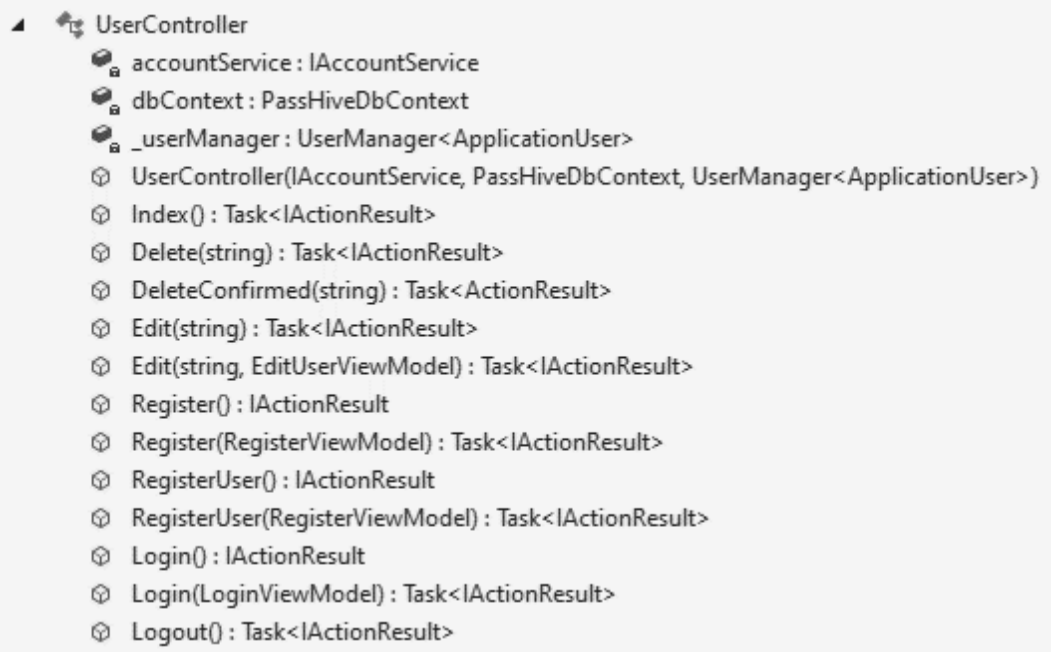
- SelectVaults() (GET)

Funguje na rovnakom princípe ako SelectUsers(), namiesto identifikátora používateľa, získa identifikátor trezoru, asynchrónne načíta všetkých používateľov, konvertuje ich do ViewModelu a vytvorí z nich list na odovzdanie View.

- SelectVaults() (POST)

Taktiež funguje na rovnakom princípe ako SelectUsers(). Kde prijme vybraných používateľov a ID trezoru, priradí trezor k vybraným používateľom a potom presmeruje na index UserPermission, ak nie je vybraný žiaden používateľ vráti sa na SelectVaults.

4.7.3 UserController



Obrázok 8. Metódy v UserController

- Index()

Táto funkcia je akcia na vykreslenie indexu UserController. Najprv sa skontroluje, či je aktuálny používateľ administrátor alebo manažér. Ak áno, načíta z databázy všetky trezory. Ak nie, načíta iba trezory prístupné aktuálnemu používateľovi. Potom načíta používateľov z databázy na základe ich rolí. Nakoniec skonštruuje ViewModel obsahujúci používateľov aj trezory a odovzdá ho do View.

- Delete() (GET)

Funkcia na zobrazenie potvrdzujúcej stránky pred vymazaním používateľa. Ako parameter prevezme ID používateľa, načíta používateľa z databázy a zobrazí View na potvrdenie.

- DeleteConfirmed() (POST)

Táto funkcia slúži na vymazanie použivateľa. Najprv vyhľadá používateľa podľa ID, odstráni ho zo všetkých rolí, odstráni všetky súvisiace UserVaulty a nakoniec odstráni samotného používateľa. V závislosti od výsledku presmeruje na index alebo používateľa opäť zobrazí.

- Edit() (GET)

Metóda na zobrazenie stránky Edit-u pre používateľa. Vyhľadá používateľa podľa ID a skonštruuje ViewModel na zobrazenie informácií o používateľovi na úpravu. Ak aktuálny

používateľ nie je správca alebo manažér a neplánuje upravovať svoj vlastný profil, presmeruje ho na indexovú stránku.

- Edit() (POST)

Táto funkcia najprv overí model, potom vyhľadá používateľa podľa ID a porovná ho s aktuálnym používateľom. Skontroluje znovu, či používateľ má správne práva na editáciu ak nie presmeruje ho na indexovú stránku. Ak používateľ má dostatočné práva, aktualizuje informácie o používateľovi, aktualizuje bezpečnostnú pečiatku a presmeruje na indexovú stránku.

- Register() a RegisterUser() (GET)

Obe metódy zobrazujú View na registrovanie používateľa. RegisterUser je vytvorený pre používateľov s Admin rolou. Je to na vytvorenie používateľa bez toho aby ho automaticky prihlásilo do nového účtu. S extra možnosťou vytvorenia nového Admin používateľa.

- Register() a RegisterUser() (POST)

Funkcia na registráciu nového používateľa. Najprv overí model a potom sa pokúsi zaregistrovať používateľa pomocou poskytnutých informácií. Ak je registrácia úspešná, používateľa prihlási a presmeruje na indexovú stránku. Ak nie, spracuje chyby a vráti View registrácie. RegisterUser funguje rovnako, len neprihlási nového používateľa a sleduje kolónku IsAdmin či má vytvoriť používateľa s Admin právami ak áno tak pridá používateľovi role Admina a Managera

- Login() (GET)

Metóda na zobrazenie prihlasovacej stránky.

- Login() (POST)

Metóda na spracovanie prihlásenia používateľa. Overí prihlasovací model a pokúsi sa prihlásiť používateľa. Ak je prihlásenie úspešné, presmeruje na indexovú stránku. Ak nie, nastaví značku označujúcu neúspech prihlásenia a vráti prihlasovacie zobrazenie.

- Logout() (GET)

Zavolá metódu odhlásenia zo služby účtu a presmeruje na prihlasovaciu stránku. Je používa funkciu [Authorize], aby sa zabezpečilo, že k nej budú mať prístup len overení používatelia.

4.7.4 VaultController

Obrázok 9. Metódy vo VaultController

- Index()

Metóda načíta aktuálneho používateľa a jeho roly, aby určila, ktoré trezory sa majú zobraziť. Ak je používateľ administrátor, načítajú sa všetky trezory, inak sa načítajú len trezory spojené s aktuálnym používateľom.

- Create() (GET)

Táto metóda vytvorí a vráti ViewModel na vytvorenie nového trezora

- Create() (POST)

Táto funkcia spracováva vytvorenie nového trezora. Overí platnosť zadaných údajov, nahrá súvisiaci súbor s obrázkom, vygeneruje kľúče RSA a vytvorí nový trezor. Taktiež vytvorí asociácie medzi vytvoreným trezorom a aktuálnym používateľom, ako aj všetkými používateľmi - administrátormi. Nakoniec presmeruje na indexovú stránku.

- Edit() (GET)

Táto metóda načíta trezor, ktorý sa má upraviť, na základe zadaného ID a odošle ho do View.

- Edit() (POST)

Metóda overí zadané údaje na aktualizáciu trezoru. Ak sú údaje platné, aktualizuje entitu trezoru a presmeruje na indexovú stránku. V opačnom prípade zaznamená chyby validácie a vráti sa do zobrazenia úprav.

- Delete() (GET)

Metóda načíta trezor na vymazanie na základe ID, zobrazí view kde používateľ potvrdí vymazanie.

- DeleteConfirmed() (POST)

Táto metóda natrvalo odstráni trezor identifikovaný zadaným ID a presmeruje na indexovú stránku.

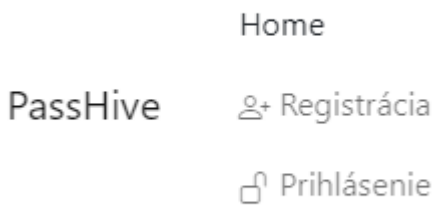
- UploadFileAsync(IFormFile fileToUpload, string folderNameOnServer)

Pomocná metóda asynchrónne nahrá súbor na server a vygeneruje jedinečný názov súboru na základe aktuálneho času. Potom uloží súbor do zadaného priečinka servera a vráti cestu k súboru.

4.8 Návrh a implementácia používateľského rozhrania

4.8.1 Menu

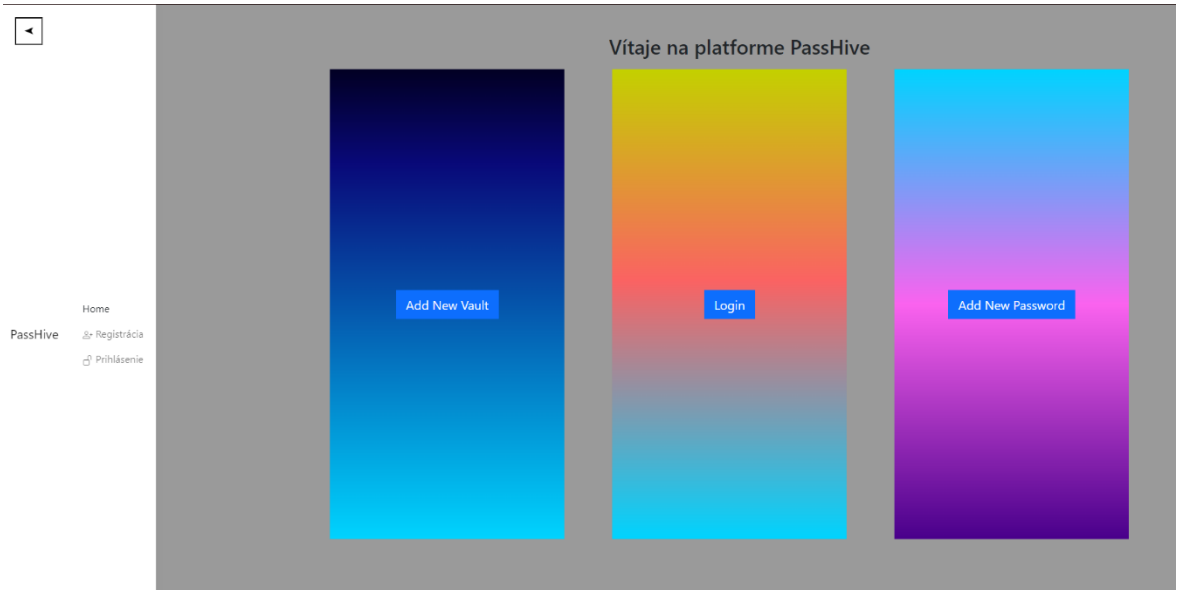
V menu sú tlačidlá na všetky časti stránky. Ak nie je používateľ prihlásený zobrazí sa mu tam len tlačidlo na Home, Prihlásenie a Registráciu. Po prihlásení sa menu zmení podľa toho akú ma používateľ rolu. Ak je to len bežný používateľ nezobrazí sa mu tam sekcia Users.



Obrázok 10. Menu stránky

4.8.2 Úvodná stránka

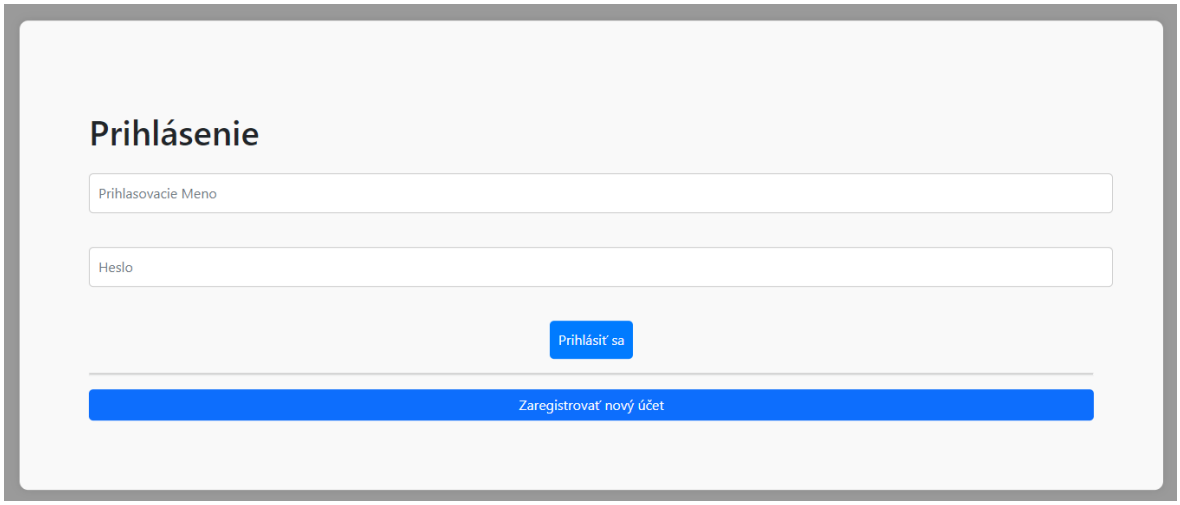
Na úvodnej stránke sa nachádzajú jednoduché tlačidlá s prepojením na časti stránky. Časť kde je Login sa po prihlásení zmení na Logout. Na obrázku je viditeľné aj menu, ktoré sa aktivuje tlačidlom v hornom-ľavom rohu.



Obrázok 11. Úvodná stránka

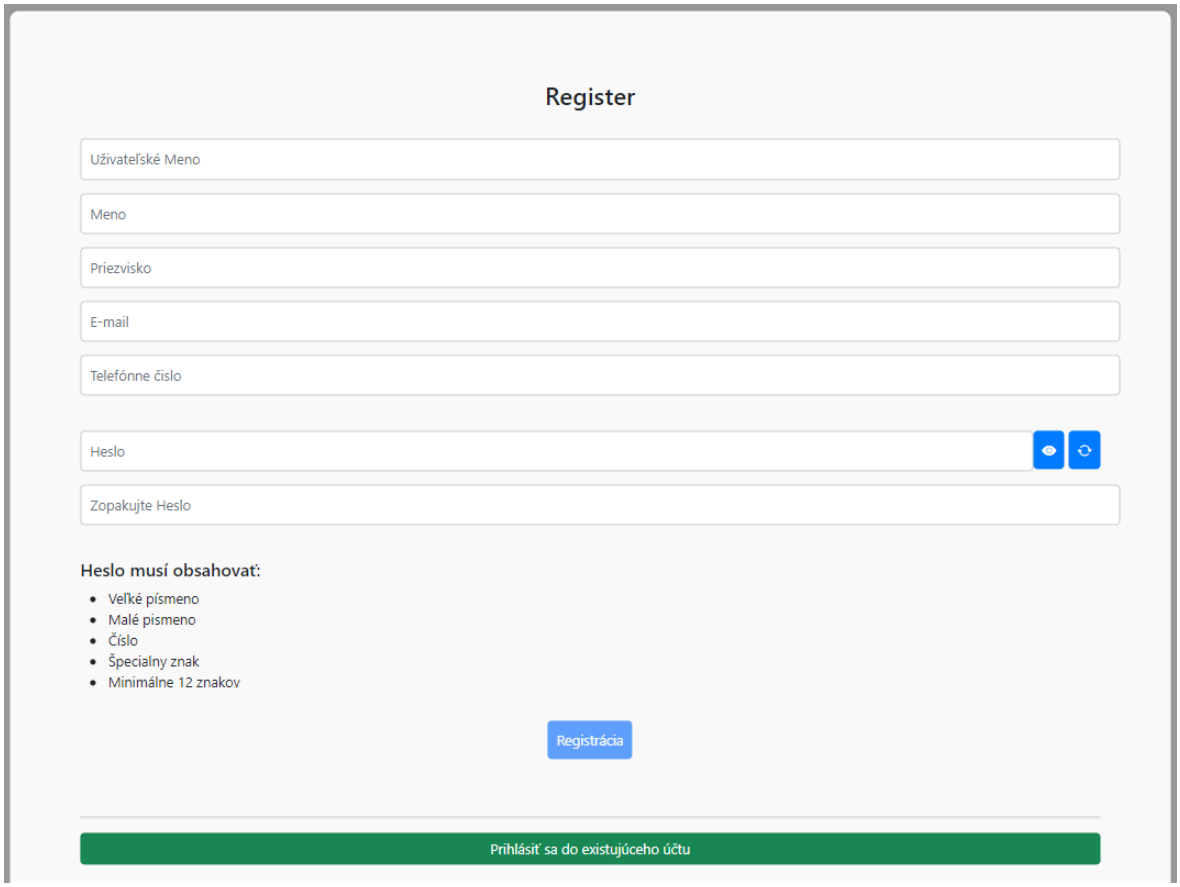
4.8.3 Stránka na prihlásenie a registráciu

Prihlasovací formulár s možnosťou zadania používateľského mena a hesla, registrácie nového účtu.



Obrázok 12. Prihlasovací formulár

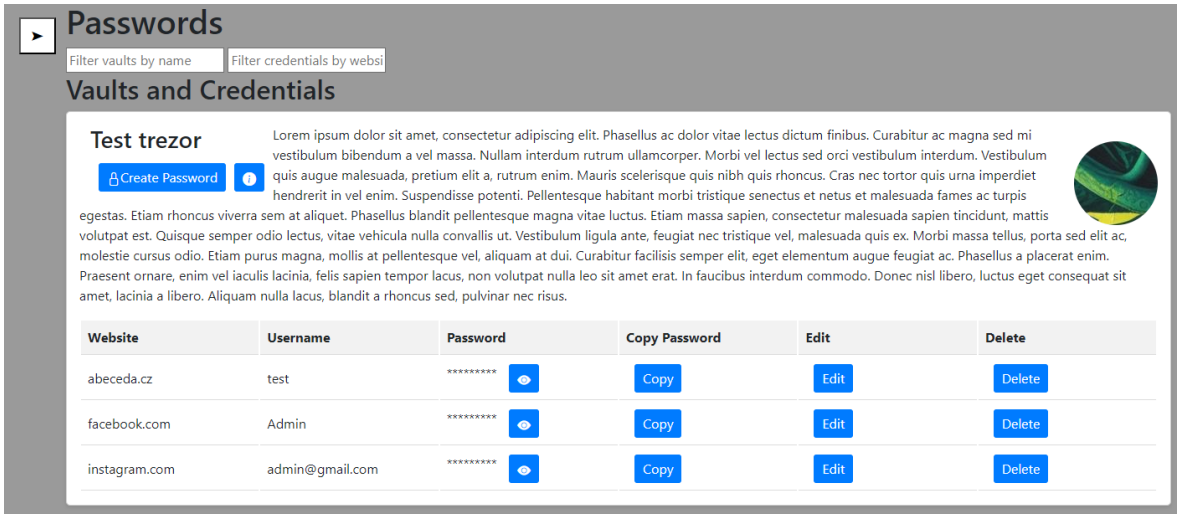
Formulár registrácie má základné polia a možnosť generácie hesla a odhalenia hesla, taktiež má kontrolu hesla či splňuje požiadavky. Kým nie sú splnené požiadavky nie je umožnená registrácia



Obrázok 13. Registračný formulár

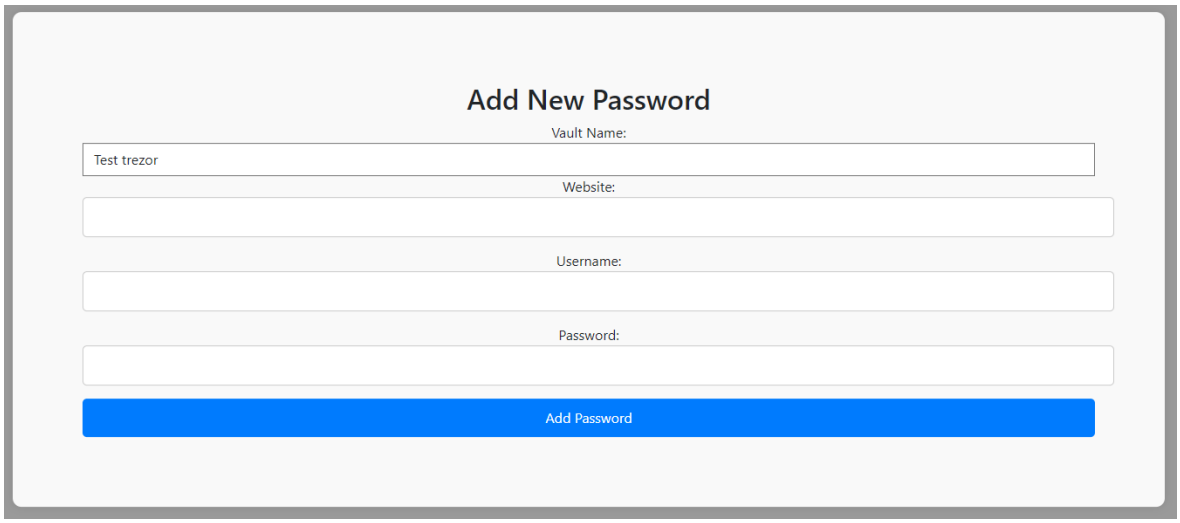
4.8.4 Podrobnosti o heslách a správa

Možnosť zobraziť podrobnosti o konkrétnom hesle vrátane používateľského mena, hesla a stránky na prihlásenie. Heslá sú skryté je ich možné odkryť pomocou tlačidla. Taktiež je vypnutá možnosť kopírovania a označovania textu. Na kopírovanie hesla slúži tlačidlo Copy a možnosť upraviť alebo odstrániť heslo. Je možné aj filtrovať na základe názvu trezoru, názve stránky alebo prihlasovacom mene. Popis je možné skryť tlačidlom vedľa ‘Create Password‘ v základe je popis skrytý.



Obrázok 14. Správa hesiel

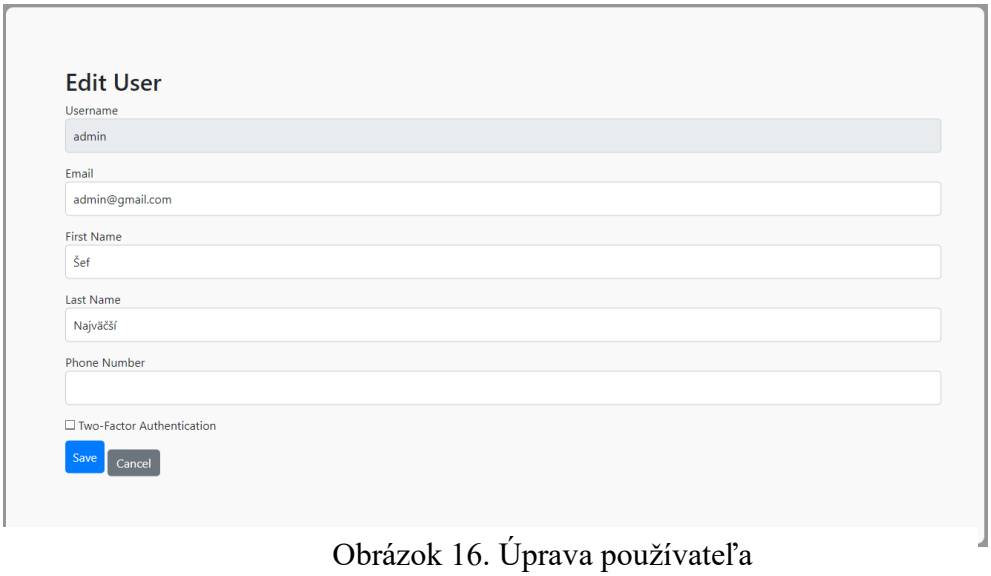
Tvorba hesla ma nasledujúci formulár kde je možné si vybrať do akého trezoru chceme pridať heslo, avšak ak klikneme na to tlačidlo nastaví sa ako základná hodnota trezor, v ktorom bolo tlačidlo stlačené.



Obrázok 15. Vytvorenie hesla

4.8.5 Správa používateľov

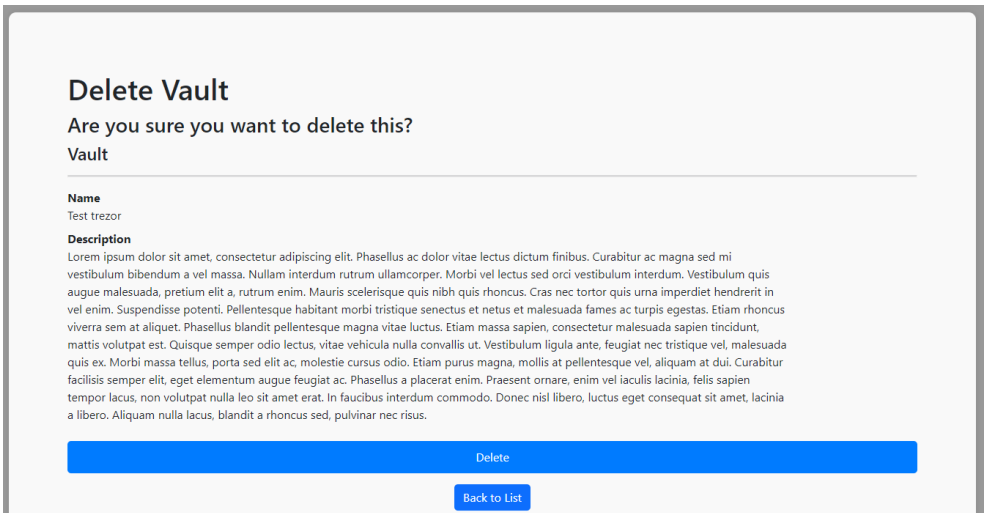
Na pridávanie používateľa je využitý formulár Register, tento formulár je upravený pridaním zaškrtávacieho poľa na vytvorenie admina. Vymazávanie má iba potvrdzovací formulár, v ktorom sa zobrazí meno a id používateľa na vymazanie. V edit formulári je možné zmeniť údaje okrem Username, je tam aj extra zaškrtávacie pole na povolenie dvojfaktorovej autentifikácie.



Obrázok 16. Úprava používateľa

4.8.6 Správa trezorov

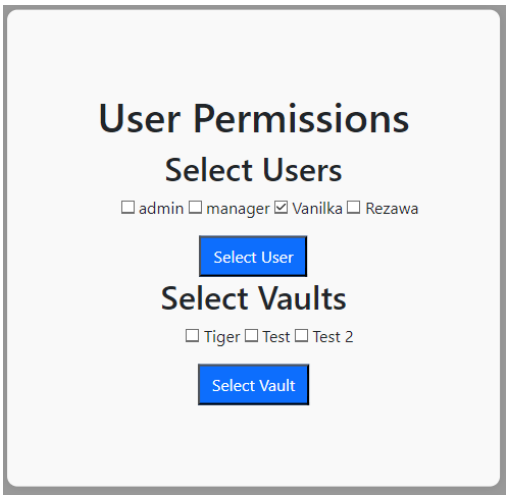
Na pridávanie nového trezoru je taktiež vytvorený jednoduchý formulár, ktorý ma v sebe pole na zadanie Mena, Popisu a pole na nahranie obrázku. Edit formulár ma v sebe zmenu mena a popisu. Tento formulár je nedostupný pre bežného používateľa. Vymazávanie je vyriešené rovnakým prístupom ako predošlé spôsoby, kde sa zobrazia základné údaje o trezore.



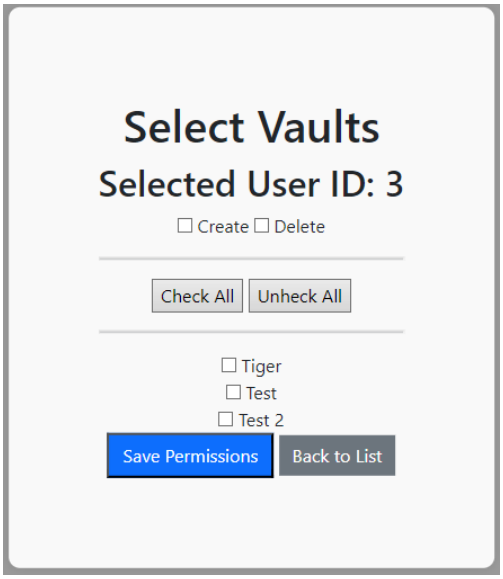
Obrázok 17. Vymazanie trezoru

4.8.7 Správa prístupov

Pridávanie prístupu je možné po zakliknutí jedného z polí a stlačenia jedného z tlačidiel. Ak sa stlačí jeden z používateľov a klikneme na ‘Select Vault‘ vybehne nám upozornenie o tom že máme vybrať jeden z trezorov. Funguje to aj v opačnom prípade ak vyberieme trezor a stlačíme tlačidlo ’Select User‘. Po správnej voľbe sa presmeruje na ďalšiu stránku na tejto stránke je možné zvoliť jeden alebo viacero polí. Ak sa zvolí používateľ bude to vyplnené trezormi a naopak. Sú tam aj tlačidla na zvolenie a zrušenie zvolenia polí. Po zakliknutí jedného z tlačidiel sa vykoná príslušná funkcia a to je vytvorenie alebo odstránenie prístupov.



Obrázok 18. Výber pridania prístupu



Obrázok 19. Výber možností pre pridanie prístupu

4.9 Vytvorenie databázy

Použitím entít, ktoré sme si vytvorili a vytvorením kódu na tvorbu databázy ďalším krokom je inicializovať si aj pár údajov priamo pri tvorbe ako je admin účet. Vytvorenie základnej časti databáze je možné vidieť na obr. Vo vnútri triedy PassHiveDbContext sú definované DbSety tieto predstavujú databázové tabuľky, ktoré budú vytvorené pre príslušné typy entít.

V rámci metódy OnModelCreating sa vytvorí inštancia triedy DatabaseInit. Táto trieda je zodpovedná za inicializáciu databázy s rolami, používateľmi a ich pridruženými rolami.

```
namespace PassHive.Infrastructure.Database
{
    public class PassHiveDbContext : IdentityDbContext<ApplicationUser, Role, int>
    {
        public DbSet<Vault> Vaults { get; set; }
        public DbSet<Credential> Credentials { get; set; }
        public DbSet<UserVault> UserVaults { get; set; }

        public PassHiveDbContext(DbContextOptions dbContextOptions) : base(dbContextOptions)
        {
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<UserVault>()
                .HasKey(uv => new { uv.UserId, uv.VaultId });

            DatabaseInit dbInit = new DatabaseInit();


            modelBuilder.Entity<Role>().HasData(dbInit.CreateRoles());


            (ApplicationUser admin, List<IdentityUserRole<int>> adminUserRoles) = dbInit.CreateAdminWithRoles();
            (ApplicationUser manager, List<IdentityUserRole<int>> managerUserRoles) = dbInit.CreateManagerWithRoles();
            (ApplicationUser user, List<IdentityUserRole<int>> userUserRoles) = dbInit.CreateUserWithRoles();


            modelBuilder.Entity<ApplicationUser>().HasData(admin, manager, user);


            modelBuilder.Entity<IdentityUserRole<int>>().HasData(adminUserRoles);
            modelBuilder.Entity<IdentityUserRole<int>>().HasData(managerUserRoles);
            modelBuilder.Entity<IdentityUserRole<int>>().HasData(userUserRoles);
        }
    }
}
```

Obrázok 20. Trieda PassHiveDbContext

V tejto ukážke je zobrazený hlavný list s tvorbou jednej roli, role je možné pridať akékoľvek.

```
public List<Role> CreateRoles()
{
    List<Role> roles = new List<Role>();

    Role roleAdmin = new Role()
    {
        Id = 1,
        Name = "Admin",
        NormalizedName = "ADMIN",
        ConcurrencyStamp = Guid.NewGuid().ToString("D")
    };
    roles.Add(roleAdmin);
    return roles;
}
```

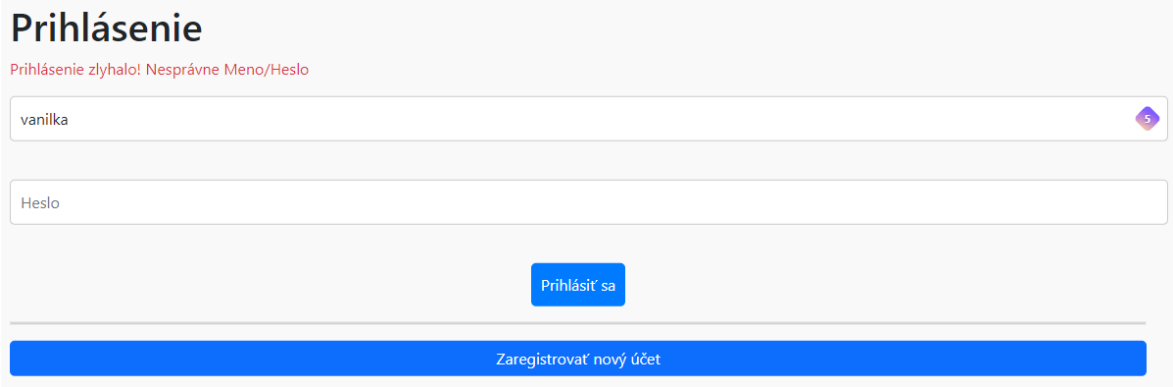
V tejto ukážke je vidieť definíciu Admin používateľa.

```
public (ApplicationUser, List<IdentityUserRole<int>>) CreateAdminWithRoles()
{
    ApplicationUser admin = new ApplicationUser()
    {
        Id = 1,
        FirstName = "Snoop",
        LastName = "Dogg",
        UserName = "admin",
        NormalizedUserName = "ADMIN",
        Email = "admin@gmail.com",
        NormalizedEmail = "ADMIN@GMAIL.COM",
        EmailConfirmed = true,
        PasswordHash = GenerateArgon2idHash("Admin123!"),
        SecurityStamp = (Guid.NewGuid().ToString("N")).ToUpper(),
        ConcurrencyStamp = Guid.NewGuid().ToString("D"),
        PhoneNumber = +421111222333,
        PhoneNumberConfirmed = true,
        TwoFactorEnabled = false,
        LockoutEnd = null,
        LockoutEnabled = true,
        AccessFailedCount = 0
    };

    List<IdentityUserRole<int>> adminUserRoles = new List<Identity-
UserRole<int>>()
    {
        new IdentityUserRole<int>()
        {
            UserId = 1,
            RoleId = 1
        },
        new IdentityUserRole<int>()
        {
            UserId = 1,
            RoleId = 2
        },
        new IdentityUserRole<int>()
        {
            UserId = 1,
            RoleId = 3
        }
    };

    return (admin, adminUserRoles);
}
```

5 OVEROVANIE FUNKČNOSTI SYSTÉMU A MOŽNOSTI ZLEPŠENIA

- Prihlásenie

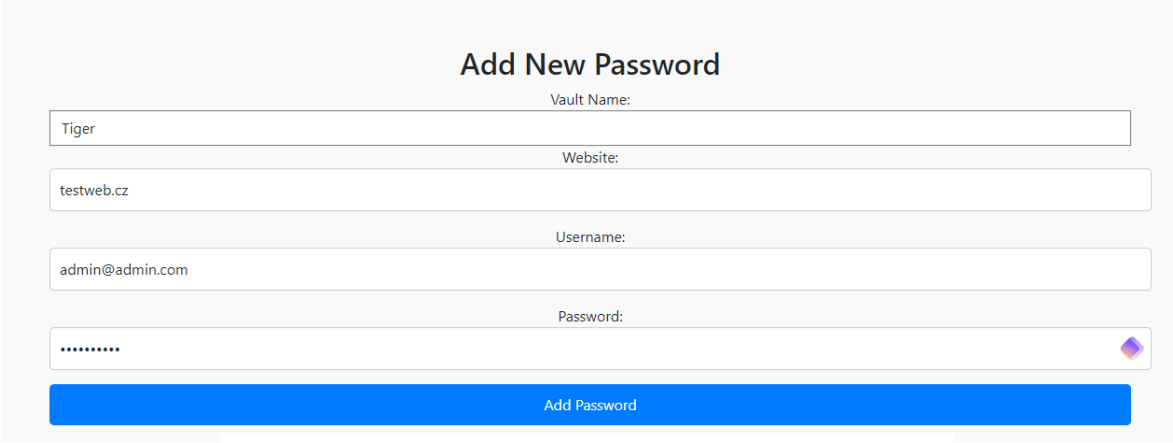
Ku prihláseniu by bolo vhodné pridať ešte metódu na 2FA. A metódu na zablokovanie prístupu po určitom množstve nesprávnych prihlásení. Inak je prihlasovanie plne funkčné, pri zadaní zlých údajov používateľa neprihlási.



Obrázok 21. Nesprávne prihlásenie

- Vytvorenie hesla

Vytvorenie hesla je funkčné. Pri vytvorení hesla sa vyberie trezor, do ktorého sa heslo pridá. Bohužiaľ bez implementácie šifrovania, čiže heslo sa uloží v otvorenom texte do databáze.



Obrázok 22. Zadané údaje na nové heslo

Username	Password	Website	VaultId
test	test extra	facebook.com	1
admin@admin.com	test hesla	testweb.cz	1
NULL	NULL	NULL	NULL

Obrázok 23. Záznamy v databáze

- Zdieľanie a odobratie trezoru a overenie či ho používateľ vidí

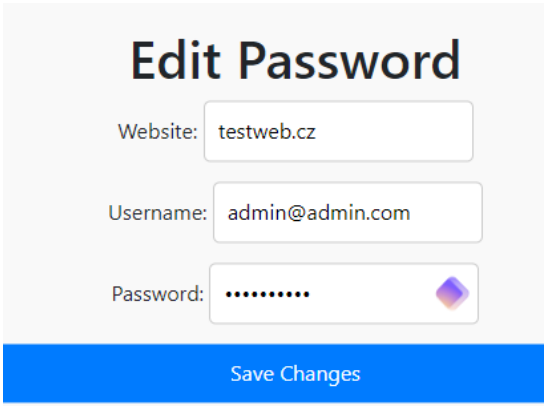
Zdieľanie je taktiež funkčné, používateľ, ktorý má k nemu prístup vidí všetky prístupné trezory a heslá v nich. Postup akým je vytvorené a odstraňované zdieľanie je vysvetlený v kapitole 4.8.7. Na obrázku číslo 24. je možné vidieť aktuálnu tabuľku vzťahov.

VaultId	UserId
1	1
2	1
3	1
1	2
2	2
3	2
2	3

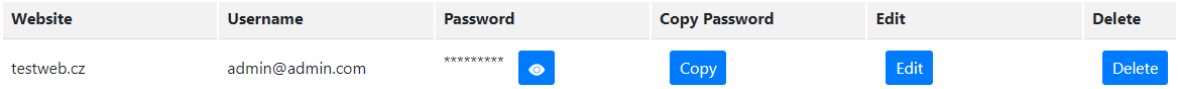
Obrázok 24. Tabuľka vzťahov z databázy

- Zmena a zobrazenie hesla

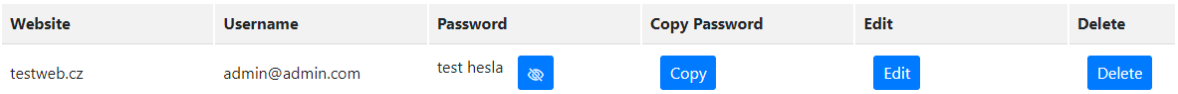
Zmena hesla v trezore je taktiež funkčná pomocou edit formuláru. Zobrazenie hesla je vyriešené pomocou javascriptu. Pridať možnosť kopírovania hesla do iného trezoru.



Obrázok 25. Zmena hesla



Obrázok 26. Skryté heslo



Obrázok 27. Odkryté heslo

5.1 Vyhodnotenie výsledkov a ďalšie vylepšenia

V priebehu implementácie sme úspešne dokončili funkcionalitu správy hesiel, zahŕňajúcu vytváranie, úpravu a mazanie hesiel. Tieto heslá sú uložené do trezorov, ktoré umožňujú ich zdieľanie medzi používateľmi. Na zabezpečenie účinného riadenia prístupu sme zaviedli správu prístupových práv prostredníctvom definovaných rolí. Taktiež sme implementovali funkciu správy používateľov, ktorá umožňuje vytváranie nových používateľských účtov.

V súčasnosti však existuje oblasť, v ktorej sme nedosiahli požadovaný výsledok - šifrovanie hesiel v trezoroch. Tento aspekt zostáva nedokončený a vyžaduje našu pozornosť v ďalšom vývoji.

Pre ďalšie vylepšenia by sme sa mali sústrediť na vyriešenie tohto nedostatku a zabezpečiť úplné šifrovanie hesiel v trezoroch pomocou metód objasnených v návrhu. Vytvorenie časového prístupu pre používateľa aby sa automaticky odobral prístup po vypršaní určitej doby. Implementácia upozornení, ktoré informujú členov tímu o zmene v zdieľaných heslách, aby sa zabezpečilo, že všetci v tíme sú vždy informovaní o aktuálnom stave a zmenách. Taktiež by sme mohli zvážiť pridanie ďalších bezpečnostných funkcií, ako je napríklad viacúrovňové overovanie používateľa, aby sme zabezpečili ešte vyššiu ochranu citlivých údajov a pridanie prihlásenia pomocou externých platforiem ako je Gmail alebo Facebook.

S touto sériou vylepšení sa domnievame, že naša aplikácia dosiahne vyšší štandard bezpečnosti a použiteľnosti, čo prispeje k zlepšeniu celkového užívateľského zážitku a dôveryhodnosti systému.

ZÁVĚR

V tejto bakalárskej práci sme sa zaoberali problematikou bezpečného ukladania a zdieľania hesiel v pracovných tímoch. Cieľom práce bolo navrhnúť a implementovať platformu, ktorá by splnila bezpečnostné štandardy a zároveň umožnila efektívnu správu hesiel.

Systém PassHive, ktorý bol vyvinutý ako súčasť tejto práce, bol úspešne integrovaný do testovacieho prostredia, kde sme overili jeho funkčnosť a bezpečnosť. Výsledky overovania potvrdili, že systém je schopný spravovať heslá, prístupové práva a prístupové údaje, ktoré vie zabezpečiť proti ich potenciálnym hrozbám.

Napriek úspešnému začleneniu spravovania prístupových práv, správy používateľov, ukladania a spravovania hesiel v trezoroch, niektoré aspekty, ako je integrácia s externými platformami, automatizácia auditných záznamov a šifrovanie hesiel v trezoroch, neboli realizované v rámci tejto práce.

Práca identifikuje možnosti pre ďalší vývoj systému. Tieto rozšírenia by mohli zlepšiť adaptabilitu systému v rôznych prostrediach a zároveň rozšíriť jeho použiteľnosť.

Záverom možno konštatovať, že práca úspešne demonštruje, ako môže byť systém prispôsobený na podporu bezpečnosti a efektívnosti v tímoch.

SEZNAM POUŽITÉ LITERATURY

[1] CISSP Study Guide, 2010. Online. Syngress. ISBN 978-1-59749-563-9. Dostupné z: https://doi.org/10.1016/C2009-0-61065-5. [cit. 2024-05-06].

[2] STAJANO, Frank; MJØLSNES, Stig F.; JENKINSON, Graeme a THORSHEIM, Per (ed.), 2016. Technology and Practice of Passwords. Online. Springer Cham. ISBN 978-3-319-29938-9. Dostupné z: https://doi-org.proxy.k.utb.cz/10.1007/978-3-319-29938-9. [cit. 2024-05-08].

[3] Audit Logging Overview, c2024. Online. DATADOG. Dostupné z: https://www.datadoghq.com/knowledge-center/audit-logging/. [cit. 2024-05-07].

[4] Two Factor Auth (2FA), c2015-2024. Online. BrainStation. Dostupné z: https://brainstation.io/cybersecurity/two-factor-auth. [cit. 2024-05-07].

[5] What is role-based access control (RBAC), 2023. Online. DigitalGuardian. Dostupné z: https://www.digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more. [cit. 2024-05-06].

[6] SQL injection (SQLi), c2024. Online. NIDECKI, Tomasz Andrzej, MUSSLER, Benjamin Daniel (ed.). Invicti. Dostupné z: https://www.invicti.com/learn/sql-injection-sqli/. [cit. 2024-05-07].

[7] Differences Between RSA, DSA, and ECC, 2021. Online. Sectigo. Dostupné z: https://www.sectigo.com/resource-library/rsa-vs-dsa-vs-ecc-encryption. [cit. 2024-05-07].

[8] Attribute-Based Access Control, 2015. Online. Computer. Roč. 2015, č. 48, s. 85-88. Dostupné z: https://doi.org/10.1109/MC.2015.33. [cit. 2024-05-06].

[9] What Is MySQL, 2024. Online. Hostinger. Dostupné z: https://www.hostinger.com/tutorials/what-is-mysql. [cit. 2024-05-07].

[10] MVVC Architecture, 2023. Online. Vtricks. Dostupné z: https://vtricks.in/lms/blog-details/mvvc-architecture. [cit. 2024-05-07].

[11] Visual Studio 2022, c2024. Online. Microsoft. Dostupné z: https://visualstudio.microsoft.com/cs/vs/. [cit. 2024-05-07].

[12] Introduction to .NET, 2024. Online. Microsoft. Dostupné z: https://learn.microsoft.com/sk-sk/dotnet/core/introduction?WT.mc_id=dotnet-35129-website. [cit. 2024-05-07].

[13] C# Introduction, c1999-2024. Online. W3schools. Dostupné z: https://www.w3schools.com/cs/cs_intro.php. [cit. 2024-05-09].

[14] Entity Framework Core, 2021. Online. Microsoft. Dostupné z: https://learn.microsoft.com/en-us/ef/core/. [cit. 2024-05-07].

[15] What Is Bootstrap?, 2023. Online. Hostinger. Dostupné z: https://www.hostinger.com/tutorials/what-is-bootstrap/. [cit. 2024-05-07].

[16] JavaScript, c1998–2024. Online. MDN Web Docs. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [cit. 2024-05-07].

[17] TAMANNA, Tahmina a SETHI, Ashwani, 2015. Steganography: A Juxtaposition between LSB DCT, DWT. Online. International Journal of Computer Applications. Roč. 126, č. 11, s. 6-10. Dostupné z: https://doi.org/10.5120/IJCA2015906215. [cit. 2024-05-06].

[18] Argon2, [2015]. Online. Dostupné z: https://www.password-hashing.net/argon2-specs.pdf. [cit. 2024-04-15].

[19] Identity on ASP.NET Core, 2024. Online. Microsoft. Dostupné z: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio. [cit. 2024-04-15].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

SQL	Structured query language
MVC	Model-View-Controller
2FA	Dvojfázové overovanie (2 Factor authentification)
AES	Advanced Encryption Standard
SPN	Substitution-Permutation Network
ECC	Elliptic-curve cryptography
RBAC	Role-Based Access Control
ABAC	Attribute-Based Access Control
MVVC	Model-View-Viewmodel-Controller
IDE	Integrated development environment
IT	Information Technology
ORM	Object–relational mapping
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
LSB	Least Significant Bit

.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

P I. CD s elektronickou verziou práce a kódom

PŘÍLOHA P I: CD S ELEKTRONICKOU VERZIOU PRÁCE A KÓDOM

Táto príloha zahŕňa elektronickú verziu práce vo formáte PDF/A a zip súbor “PassHive“ zahŕňajúci všetky zdrojové kódy stránky.