FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Systém pro elektronické zajištění voleb v Mensa ČR
Student:	Vladimir Cherkezov
Vedoucí:	Ing. Tomáš Nováček
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Cílem práce je navrhnout a naimplementovat řešení projektu elektronického volebního systému Mensy ČR. Postupujte v těchto krocích:
- Seznamte se s problematikou realizace voleb přes internet, a to jak po technické, kryptografické, tak procesní stránce,
- prostudujte existující řešení pro elektronické hlasování Vědecké rady FIT ČVUT,
- seznamte se procesem voleb v organizaci Mensa ČR a analyzujte jeho odlišnosti od elektronického hlasování Vědecké rady FIT ČVUT,
- prodiskutujte s Mensou, jak upravit či přepracovat elektronické hlasování Vědecké rady FIT ČVUT tak, aby vyhovoval jejich potřebám,
- navrhněte, realizujte a zdokumentujte úpravu či přepracování elektronického hlasování Vědecké rady FIT ČVUT dle analýzy z předchozího bodu,
- ve spolupráci s Mensou proveďte akceptační test řešení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 6. prosince 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Systém pro elektronické zajištění voleb v Mensa ČR

Vladimír Cherkezov

Katedra softwarového inženýrství
Vedoucí práce: Ing. Tomáš Nováček

4. června 2020

Poděkování

Chtěl bych poděkovat Ing. Tomáši Nováčkovi, vedoucímu mé práce, za odborné vedení a konzultace, cenné rady, trpělivost a čas, který mi v průběhu zpracování bakalářské práce věnoval. Dále bych rád poděkoval: Ing. Michalu Valentovi, Ph.D. za poskytnutí aplikace Baletka za účelem její analýzy. V neposlední řadě rád bych poděkoval Zuzaně Polákové, Tomáši Kubešovi a Martinu Sedláčkovi za pěknou spolupráci a pomoc při implementaci výsledného systému ze strany Mensy. Velké poděkování náleží rovněž mé rodině a přátelům za psychickou podporu během psaní této práce. Také bych rád poděkoval výrobci energetických nápojů Red Bull GmbH za zajištění mé produktivity v nočních hodinách.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na jejímž základě se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ustanovení § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisu.

V Praze dne 4. června 2020 . . . . . .

Odkaz na tuto práci

Cherkezov, Vladimír. *Systém pro elektronické zajištění voleb v Mensa ČR.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato bakalářská práce se zabývá analýzou, návrhem a následnou implementací webové aplikace, jejímž účelem je zajištění voleb pro neziskovou organizaci Mensa ČR. Práce obsahuje detailní analýzu zabezpečení webových aplikací, v níž se lze seznámit jak s obecnými řešeními bezpečnostních problémů, tak i s konkrétními řešeními ve frameworcích Ruby on Rails a Symfony, které jsou popsány během analýzy aplikace Baletka, respektive v realizaci volebního systému pro organizaci Mensa ČR. Volební systém, který je výsledným produktem dané práce, je navržen a otestován ve spolupráci z organizací Mensa ČR. S těmito procesy se lze seznámit v příslušných sekcích a příloze, ve které se nacházejí wireframy této aplikace.

Klíčová slova elektronické volby, zabezpečení webu, webová aplikace, Symfony framework, elektronické hlasování, Mensa ČR

Abstract

This bachelor's thesis dedicates to analysis, design, and subsequent implementation of a web application, the goal of which is to ensure elections for the non-profit organization Mensa CZ. The thesis contains a detailed analysis of web application security, during which you can get acquainted with theoretical solutions to security problems and practical solutions in the frameworks Ruby on Rails and Symfony, which are described during the analysis of the application Baletka, respectively the implementation of the Election System for Mensa CZ. The voting system, which is the final product of the work, is designed and tested in cooperation with the organization Mensa CZ. These processes can be found in the relevant sections and the addition in which the wireframes of this application are located.

Keywords electronic voting, web security, web application, Symfony framework, electronic ballot, Mensa CZ

Obsah

Seznam obrázků

Úvod

V současné době existuje velké množství problémů, jejichž řešení se dá zjednodušit pomocí použití internetu. Jedním z takových problémů je hlasování. Elektronické hlasování se provádí na různých úrovních – v rámci nějaké organizace nebo celého státu. Většinou při provádění voleb existují body, které je potřeba garantovat: správné počítání hlasů, anonymizace výsledků, pokud jsou volby tajné, a další. Abychom mohli zachovat dané vlastnosti pro elektronické hlasování, potřebujeme použít správné technologie a vyřešit problémy, které jsou spojené jak s elektronickými volbami, tak i s webovými aplikacemi obecně.

Hlasování přes internet je velmi aktuální pro jednu z neziskových společností České republiky. Jedná se o organizaci Mensa ČR, pro kterou jsou jednou z důležitých aktivit volby do řídících orgánů. Pro zjednodušení volebního procesu této organizace bylo rozhodnuto vytvořit webovou aplikaci, která by měla zjednodušit proces hlasování v Mense.

Zřejmými výhodami tohoto řešení jsou: není potřebná distribuce a následující sbírání hlasovacích lístků, úspora času voličů k tomu, aby se dostali na místo konání voleb, úspora času spojená s organizací voleb a následným zpracováním údajů získaných od voličů.

Je potřeba zaznamenat, že volby v Mense se konají formou přímého, rovného a tajného hlasování. Zajištění těchto principů způsobuje určité náročnosti, především spojené s bezpečností, a jejich nedostatečně pečlivé vyřešení může vést v extrémních případech k porušení volebního procesu nebo zneplatnění výsledků voleb.

Cíl práce

Cílem práce je navrhnout a implementovat elektronický volební systém pro organizaci Mensa ČR. Tato práce je zaměřena na seznámení se s problémy, které jsou spojené s elektronickým hlasováním, a eliminaci jejich vzniku při reali-

zaci aplikace pro volby v Mense ČR. Pro detailnější představu problémů, které je potřeba řešit v rámci této práce, bude analyzována bezpečnostní stránka aplikace Baletka, která je používaná pro elektronické hlasování vědecké rady Fakulty informačních technologií Českého vysokého učení technického v Praze (FIT) a jejíž koncepce, jak později ukážeme, je shodná s koncepcemi aplikace, kterou potřebuje Mensa. Po seznámení s problémy budou dílčími cíli: správné navržení technologie a nástrojů, pomocí kterých budeme příslušné problémy řešit, navržení aplikace a poté její implementace.

Struktura práce

Práce je strukturována do pěti logických celků: analýza, návrh, realizace, testování a závěr.

První část je soustředěna na získání informací potřebných pro vhodný návrh a realizaci výsledné aplikace. Tato část začíná seznámením se s procesem hlasování v organizaci Mensa ČR a jeho porovnáním s procesem hlasování, který je realizován v systému Baletka. Po zjištění shody systému probíhá analýza bezpečnosti, v rámci které dojde k seznámení se s možnými hrozbami a pak, pro dosažení vyššího stupně porozumění problémům, provedení analýzy zabezpečení systému Baletka. Tuto kapitolu uzavírá analýza technologií a rozhodnutí o tom, které technologie bychom mohli použít při realizaci mensovní aplikace.

Druhá část se zabývá návrhem aplikace v souladu s analýzou požadavků poskytnutých Mensou. Tato část obsahuje detailní popis navrženého systému, znázorněný pomocí diagramů případů užití, wireframů a diagramů vybraných procesů.

Třetí část je věnovaná realizaci aplikace na základě získaných znalostí a pomocí technologií uvedených v analytické části. V této části jsou uvedeny použité technologie, popis nasazení aplikace do serveru Mensy, realizace zabezpečení a některá řešení, která byla označena jako netypická.

Čtvrtá část je spojena s testováním vytvořené aplikace, ve které jsou popsány použité testovací nástroje a typy jednotlivých testů, které byly provedeny.

V páté části shrneme, k čemu jsme dospěli v této práci, a navrhneme další možné zlepšení pro naši aplikaci.

Kapitola 1

Analýza

Cílem této části je získání informací, které budeme potřebovat pro návrh výsledné aplikace. Zde zjistíme, jak probíhá volební proces v organizaci Mensa, a porovnáme jej s procesem hlasování poskytnutým nám pro analýzu aplikace Baletka.

Na základě porovnání těchto procesů rozhodneme, jakou část poskytnuté aplikace potřebujeme analyzovat detailněji. Dále zjistíme, že bude přínosné provést analýzu zabezpečení systému Baletka. Ale před tím, než začneme analýzu zabezpečení Baletky, se potřebujeme seznámit s problémy, které nastávají při zabezpečení webových aplikací, abychom mohli dále provádět zkoumání toho, jak jsou dané problémy vyřešeny v poskytnuté aplikaci. Na konci této kapitoly v podkapitole 1.5 provedeme analýzu a zdůvodníme, jaké nastroje bychom chtěli použít pro realizaci volební aplikace.

1.1 Seznámení s problematikou elektronických voleb

Tato sekce je věnovaná analýze volebních systémů po technické a kryptografické stránce, seznámení se s procesy, které v nich probíhají.

Pokusy o provádění voleb elektronicky byly uskutečněny v Estonsku, Belgii, Německu, Švýcarsku a mnoha dalších státech. To způsobilo vznik různých systémů řešících volby. Každý ze systémů má odlišnou realizaci a řeší problémy s ní spojené. V dané sekci se soustředíme na analýzu voleb, které probíhají v Estonsku a Švýcarsku. Výběr těchto dvou systému je způsoben tím, že používají na rozdíl od ostatních postupy a technologie, které bychom mohli v určité míře použít i my.

1.1.1 Elektronické volby v Estonsku

Systém elektronických voleb v Estonsku byl vytvořen v roce 2005 a je založen na identifikačních kartách, které v sobě obsahují 2 PIN kódy. Jeden, ze kterého se identifikují voliči v systému, a druhý pro digitální podpis hlasu. Proces hlasování se dá popsat v následujících krocích:[4]

1. Naskenování ID karty voliče pomocí čtečky karet.
2. Verifikace identifikačního PINu.
3. Ověření, zda uživatel je zaregistrován do databáze.
4. Zobrazení volebního lístku uživateli.
5. Volba kandidátů a posílaní šifrovaného hlasu pomocí druhého PINu.
6. Počítání anonymních hlasů a smazání digitálního podpisu.

Jak je vidět z procesu estonského systému, ten zaručuje tajnost hlasování, tedy není zachováná žádná vazba mezi voličem a jeho hlasem. Kvůli odstranění té vazby vzniká hlavní problém daného přístupu – nelze poznat falzifikaci hlasu. Což vede ke snížení důvěry ze strany uživatele a způsobuje problém při využívání systému.

1.1.2 Elektronické volby ve Švýcarsku

Elektronické volby ve Švýcarsku probíhají do určité míry podobně jako v Estonsku, ale ve volebním procesu jsou podstatné odlišnosti. Proto je pro nás zajímavé tento proces prozkoumat. Proces hlasování se budeme snažit popsat v následujících krocích:

1. Volič dostává poštou volební lístek, volební kartu a volební dokumenty, které bude potřebovat pro elektronické hlasování. Volební dokumenty v sobě obsahují kódy, které bude volič potřebovat pro elektronické hlasování.
2. Pak zadá počáteční volební klíč do systému a dostane přístup k elektronickému volebnímu lístku.
3. Dále se posílá na server zašifrovaný hlas voliče a volební karta.
4. Volební karty a hlasy jsou ukládány na serveru zvlášť, a hlasy navíc jsou zamíchaný tak, aby nešlo identifikovat hlas podle pořadí.
5. V den voleb dostává volební komise přístup ke hlasům pomocí více komponentového společně vytvořeného klíče.

Volební komise není schopná identifikovat voliče, ale je schopná ověřit validitu hlasů podle volebních karet. Tím pádem je anonymita zachráněna a volební komise může ověřit, že hlasy jsou validní. Ale teď narážíme na stejný problém, který jsme zjistili během analýzy voleb v Estonsku. Zjistíme podrobněji, jak probíhá volební proces z pohledu voliče. Po volbě kandidátů se volební lístek odešle na server, kde budou ke každé jednotlivé volbě přidány kódy, a pak se odešle zpět voliči. Úkolem uživatele je zjistit, jestli kódy, které se objevily u jednotlivých hlasů, odpovídají kódům, které volič může najít na volebním lístku, jenž byl doručen poštou. Pokud jsou porovnané kódy stejné, volič zadá potvrzovací kód a odešle hlas. V opačném případě zamítne elektronické hlasování, spojí se s organizátory voleb a zúčastní se voleb jiným způsobem.

Po úspěšném odeslání hlasu bude volič potřebovat ověřit identifikátor, pod kterým byl přihlášen do systému, a v úspěšném případě bude hlas konečně započítán. Tento proces umožňuje voliči ověřit, jestli jeho volba byla úspěšné uložena.[5]

Závěr

V této sekci jsme se seznámili s problematikou elektronických voleb ve dvou státech a zjistili jsme, jak zaručit tajnost voleb, a problémy s tím spojené. Pro více detailních seznámení s daným tématem provedeme analýzu aplikace Baletka.

1.2 Analýza procesů hlasování

Tato kapitola je věnovaná porovnání dvou procesů: hlasování v organizaci Mensa a hlasování v systému Baletka. Po seznámení se s dvěma procesy a jejich porovnání budeme muset rozhodnout, jaká část analýzy Baletky bude mít největší přínos pro výsledný systém Mensy.

1.2.1 Analýza volebního procesu v organizaci Mensa ČR

Tato podkapitola je věnovaná detailům volebního procesu v organizaci Mensa České republiky.

Volby v organizaci Mensa ČR se konají jednou za dva roky. Jejich účelem je zvolení předsedy rady, členů rady a členů kontrolní komisi. Kandidáti na dané pozice jsou navrženi členy Mensy. Volby v Mense se konají formou tajného hlasování a volební proces vyžaduje účast orgánu, který odpovídá za řádný průběh voleb. Tento orgán se jmenuje volební komise.

Na volební proces můžeme pohlížet z pozice účastníka voleb, kterým je každý zletilý člen Mensy, a z pozice správce, kterým je člen volební komise.

Volební komise je zodpovědná za vyplnění údajů potřebných pro provedení hlasování. Pro budoucí hlasování se přidává informace o kandidátech, která

člení na povinné údaje, jako je jméno, členské číslo, proslov, e-mail, a nepovinné jako telefon, role a další informace, jako např. LinkedIn. Po vyplnění údajů potřebných pro hlasování začíná další fáze voleb.

V teto fázi voleb volební komise zajišťuje zaslání hlasovacích lístků všem členům Mensy.

Po vyplnění lístků a jejich odeslání se podle pokynů volební komise provádí sčítání hlasů. Pak jsou výsledky voleb vyhlášeny předsedou volební komise na valné hromadě, která je nejvyšším orgánem Mensy a skládá se ze všech členů Mensy. [6]

Z pohledu účastníka voleb vypadá proces poměrné triviálně. Volič dostává od členů volební komise hlasovací lístek a volí kandidáty na příslušné pozice. Účastník voleb může zvolit maximálně tolik kandidátů, kolik je míst v radě nebo kontrolní komisi. V případě voleb předsedy může účastník voleb zvolit právě jednoho kandidáta. Po vyplnění hlasovacího lístku jej volič odesílá podle pokynů volební komise.

1.2.2 Analýza procesu hlasování v systému Baletka

V následující kapitole provádíme analýzu procesu hlasování v systému Baletka poskytnutém FIT, abychom dále, po porovnání s volebním procesem Mensy, mohli analyzovat, jaké části Baletky by mohly být přínosem pro návrh a implementaci volební aplikace Mensy.

Baletka je aplikace, která slouží jako hlasovací systém pro vědeckou radu FIT. Ten se používá pro usnadnění rozhodovacího procesu, který se týká činnosti vědecké rady FIT. Pro klasické hlasování vědecké rady je potřeba naplánovat schůzi. Na této schůzi se musí členové sejít v jedné místnosti a poté hlasovat. Baletka umožňuje provádět hlasování vzdálené a v době, kdy je toto hlasování spuštěné.

Proces hlasování lze popsat ze dvou pozic. Tedy z pozice účastníka voleb, kterými jsou členové vědecké rady FIT, a z pozice osoby, zabývající se administrací voleb, kterou pojmenujeme pověřenou osobou.

Pověřená osoba vytváří nové hlasování, které může obsahovat jednu, nebo více otázek s třemi možnostmi odpovědí: „Ano“, „Ne“ a „Zdržuji se“. Hlasování se může provádět bud’ tajně, nebo veřejně. Po vytváření hlasování pověřená osoba spouští hlasování na předem stanovenou dobu. Pak se hlasování mohou zúčastnit členové vědecké rady. Členové vědecké rady vyplní hlasovací lístky a pošlou je na server pro další zpracování. Po vypršení doby stanovené pro hlasování Baletka povolí pověřené osobě přístup k výsledkům hlasování.

1.2.3 Porovnání procesů hlasování

Tato podkapitola je věnovaná odhalení shod a rozdílů dvou procesů. Shoda procesů se projevuje v následujících bodech:

Za prvé, účastníky obou procesů jsou určité skupiny osob. V organizaci Mensa jsou to členové Mensy a v systému Baletka jsou to členové vědecké rady FIT. Z toho plyne, že pro oba procesy je důležitá identifikace osob, aby nešlo zpřístupnit hlasování těm, kdo do skupiny voličů nepatří.

Za druhé, obě hlasování vyžadují tajnost. To znamená, že obojí musí zajišťovat tajnost rezultátů hlasování jednotlivých účastníků.

Za třetí, cílem obou hlasování je rozhodování o věcech spojených s korektním provedením činnosti organizace. Proto je důležité zabránit problémům, které by mohly ovlivnit proces hlasování nebo zneplatnit jeho výsledky.

Kromě shod lze při porovnání pozorovat i odlišnosti. Podstatným rozdílem obou procesů je to, o čem se rozhoduje v průběhu hlasování. Cílem hlasování v organizaci Mensa je volba lidí na určité pozice do řídicích orgánů, zatímco vědecká rada FIT řeší jednotlivé problémy, které nastávají během její činnosti.

Při porovnání procesů, které probíhají v organizace Mensa, a systému hlasování pro vědeckou radu FIT lze tvrdit, že probíhají podobně. Podobnost ale spočívá v problémech, které potřebují řešit porovnávané strany pro správné konání obou procesů. Ovšem samotné procesy mají určité odlišnosti. Pro nás to v podstatě znamená, že je zbytečné analyzovat celou aplikaci Baletka. Mnohem větší smysl má soustředit se na společné problémy v obou procesech. Takže by bylo velmi přínosné pro návrh mensovní aplikace prozkoumat bezpečnostní stránku aplikace Baletka. Proto se budeme v následujících sekcích této kapitoly zabývat analýzou zabezpečení systému Baletka a zabezpečením webových aplikaci obecně.

1.3 Analýza zabezpečení aplikací

V dané podkapitole budeme rozebírat bezpečnostní problémy, které mohou vzniknout v aplikaci, a způsoby jejich zabezpečení. Zjistíme, jaké jsou nejčastější bezpečnostní problémy, a zjistíme, jaká řešení těchto problémů existují.

Základem struktury této analýzy byla klasifikace nejčastějších bezpečnostních problémů pro verzi OWASP [1], která je v podstatě prezentovaná následujícím žebříčkem:

1. Injection
2. Broken Authentication
3. Sensitive data exposure
4. XML External Entities (XXE)

[1] Open Web Application Security Project – je projekt zabývající se bezpečností webových aplikací. Komunita OWASP zahrnuje korporace, vzdělávací organizace a jednotlivce z celého světa. Komunita pracuje na tvorbě článků, příruček, dokumentace, nástrojů a technologií volně přístupných z internetu.

5. Broken Access control
6. Security misconfigurations
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with known vulnerabilities
10. Insufficient logging and monitoring

V dané kapitole, pokud nebude řečeno jinak, je zdrojem instrukcí o tom, jak se bránit proti uvedeným problémům, organizace OWASP, ale také byly nalezeny informace v jiných zdrojích, které doporučovaly podobné řešení problémů. Tyto odkazy budou uvedeny v následujícím textu spolu z odkazem na instrukce OWASP.

Následující struktura byla doplněna dalšími podstatnými bezpečnostními problémy, které byly nalezeny v jiných zdrojích. Takže byly rozebrány problémy přihlášení a bezpečného přenosu dat po síti. I když jsou tyto problémy mimo tuto strukturu, stejně je potřebujeme řešit pro výslednou aplikaci.

1.3.1 Injection

SQL Injekce – druh injekce kódu, který vzniká, pokud očekáváte od uživatele na vstupu nějaké data, např. `username/userid`, ale místo toho uživatel zadá SQL příkaz, který bude spuštěn ve vaší databázi bez vašeho vědomí.[7]

Dále uveďme sadu instrukcí, jak zabránit injekci:

- Používejte bezpečné rozhraní API [2] , které se zcela vyhýbá použití internetu nebo poskytuje parametrizované rozhraní nebo migruje za použití nástrojů pro mapování relací objektů (ORM) [3].
- Používejte pozitivní nebo whitelist validace [4] vstupu na straně serveru. Toto není úplná obrana, protože mnoho aplikací vyžaduje speciální znaky.
- U všech zbývajících dynamických dotazů unikněte zvláštním znakům pomocí specifické syntaxe escape [5] pro tento interpreter.

[2] Application Programming Interface – popis metod, pomocí kterých jeden počítačový program může interagovat s jiným programem.

[3] Object-Relational Mapping – technologie, která spojuje databáze s koncepty objektově orientovaných programovacích jazyků a vytváří „virtuální objektovou databázi“

[4] Whitelist – druh kontroly založený na určení symbolů, které jsou dovolené. Například blacklist je druh kontroly, jehož cílem je určit, jaké symboly je potřeba odstranit.

[5] Escapování symbolu je postup nahrazení symbolů, které mají pro konkrétní jazyk speciální význam, jejich bezpečnými verzemi.

- Použijte LIMIT a další ovládací prvky SQL v rámci dotazů, abyste zabránili hromadnému zpřístupnění záznamů v případě SQL injekce.[8]

1.3.2 Broken Authentication

Podle OWASP má webová aplikace zranitelnost, která může být použita pro rozbití autentifikace, pokud:

- Povoluje brute force (tzv. útok hrubou silou) nebo jiné automatické útoky.
- Povoluje slabá nebo dobře známá hesla, např. Password1.
- Používá neefektivní postupy pro obnovení zapomenutých údajů, např. knowledge-based otázky [6].
- Slabé hashování, šifrování hesel nebo ukládaní je jako plain text (tzv. prostý text).
- Nevyužívá se nebo se využívá neefektivně dvoufaktorovou autentizací.
- Poskytuje identifikátor sezení v URL.
- Neotáčí identifikátor sezení po úspěšném přihlášení.
- Sezení uživatele nebo autentizace token nejsou správně zneplatněny během odhlášení nebo ve fázi nečinnosti.

Jak předcházet zranitelnosti spojené s autentizací:

- Pokud je to možné, implementujte dvoufaktorovou autentizaci, abyste zabránili automatizovaným útokům, brute force a opakovanému zneužití zjištěných přihlašovacích údajů.
- Neposílejte ani nenasazujte žádné výchozí přihlašovací údaje, zejména pro administrátory.
- Realizujte kontrolu slabých hesel, např. lze kontrolovat nová a změněná hesla podle seznamu 10 000 nejhorších hesel.
- Zarovnejte zásady délky hesla, složitosti a rotace s pokyny NIST 800-63 B v oddíle „Memorized Secrets“[9] nebo jiné moderní politiky hesel založené na důkazech.
- Zajistěte, aby registrace, obnovení přihlašovacích údajů a cesty vašeho API byly zabezpečené proti vyčíslení účtu.

[6] Speciální otázka, kterou musí uživatel zadat navíc pro ověření jeho identity, např. jméno svého domácího mazlíčka.

- Omezte nebo stále zpožďujte neúspěšné pokusy o přihlášení.
- Zaznamenejte všechna selhání a upozorněte administrátory, když jsou detekovány credential stuffing [7] , brute force nebo jiné útoky.
- Použijte zabezpečený integrovaný správce relací na straně serveru, který po přihlášení vygeneruje nové ID náhodné relace s vysokou entropií.[10]

1.3.3 Sensitive Data Exposure

Vystavení citlivých dat je jednou z nejhorších zranitelností. Vystavení dat se stává kvůli neefektivní nebo chybějící ochranné databázi, ve které se nacházejí citlivá data. Většinou je příčinou slabé šifrování nebo chybějící šifrování, slabiny softwaru a další problémy. Příkladem citlivých údajů jsou: hesla, čísla kreditních karet, autorizační údaje, informace umožňující identifikaci osob a další osobní údaje.[11]

Jak zabránit vystavení citlivých dat:[12]

- Neuchovávejte citlivá data zbytečně. Data, která nejsou uchovaná, nemohou byt odcizena.
- Je potřeba šifrovat uložená citlivá data. Zajistěte aktuálnost a bezpečnost standardních algoritmů, protokolů a klíčů.
- Šifrujte všechna přenášená data pomocí zabezpečených protokolů, jako je TLS [8] s perfektním dopředným tajemstvím [9] .
- Vynuťte šifrování pomocí směrnic, jako je HTTP Strict Transport Security (pojem bude vysvětlen v kapitole 1.3.13).
- Zakažte kešování pro odpovědi, které obsahují citlivá data.
- Ukládejte hesla pomocí silných adaptivních a solených hashovacích funkcí s pracovním faktorem, jako je Argon2, scrypt, bcrypt nebo PBKDF2.
- Nezávisle ověřte účinnost konfigurace a nastavení.

[7] Credential stuffing – je automatická injekce porušených dvojic uživatelských jmen a hesel k podvodnému získání přístupu k uživatelským účtům. Jedná se o podmnožinu kategorie útoku hrubou silou.

[8] Transport Layer Security – je kryptografický protokol poskytující možnost zabezpečené komunikace na internetu.

[9] Perfect forward secrecy – vlastnost zabezpečených komunikačních protokolů, která zaručuje, že prozrazení soukromého klíče či hesla neohrozí dávnou komunikaci.

1.3.4 XML External Entities

Externí entity XML je bezpečnostní problém aplikace, která parsuje XML vstup. Tento problém nastává, když vstup XML, obsahující odkaz na externí entitu, je zpracován slabě nakonfigurovaným analyzátorem XML.

Jak zabránit problému s externími entitami XML:

- Pokud je to možné, používejte méně složité formáty dat, jako je JSON, a vyhněte se serializaci [10] citlivých dat.
- Zakažte zpracování externí entity XML a zpracování DTD [11] ve všech analyzátorech XML v aplikaci, podle OWASP Cheat Sheet XXE Prevention [13].
- Implementujte pozitivní whitelisting ověření vstupu, filtrování nebo dezinfekci na straně serveru, abyste zabránili nepřátelským datům v XML dokumentech, hlavičkách nebo uzlech.

1.3.5 Broken Access Control

Zabezpečení webových stránek řízení přístupu znamená omezení počtu sekcí nebo stránek, na které se návštěvníci mohou dostat, v závislosti na jejich potřebách. Zranitelnost řízení přístupu může dát útočníkovi přístup k neoprávněným funkcím nebo datum, zobrazit citlivé soubory, povolit útočníkovi měnit přístupová práva, upravovat údaje ostatních uživatelů atd.

Jak zabránit problémům s řízením přístupu:

- S výjimkou veřejných zdrojů je přístup defaultně zakázán.
- Mechanismy řízení přístupu implementujte jednou a znovu je používejte v celé aplikaci, včetně minimalizace využití CORS [12] .
- Omezte rychlost přístupu ke kontroléru a API, abyste minimalizovali škodu od automatizovaných nástrojů pro útoky.
- Vypněte možnost vylistování adresáře vašeho web serveru a ověřte si, že metadata (např. .git) a záložní soubory se nevyskytují v kořenovém adresáři vaší webové aplikace.

[10] Serializace je proces převádění objektu v paměti na proud bajtů pro posílání přes síť nebo ukládání na disk.

[11] Počítačový jazyk, který se používá k zaznamenání skutečných pravidel syntaxe pro jazyk XML.

[12] Cross-Origin Resource Sharing – technologie prohlížečů, která dovoluje webové stránce poskytnout vám přístup ke zdrojům z jiné domény.

- Tokeny JWT [13] by měly být na serveru zneplatněny po odhlášení.

1.3.6 Security Misconfigurations

Hackeři vždy hledají způsoby, jak proniknout na webové stránky, a nesprávná konfigurace zabezpečení může být snadnou cestou. Zde je několik příkladů věcí, které hackeři obvykle využívají, aby získali neoprávněný přístup: neopravené vady, výchozí konfigurace, nevyužité stránky, nechráněné soubory a adresáře, zbytečné služby. K nesprávné konfiguraci zabezpečení může dojít na kterékoli úrovni aplikací, včetně síťových služeb, platformy, webového serveru, aplikačního serveru, databáze, frameworku, vlastního kódu, předinstalovaných virtuálních strojů, kontejneru úložného prostoru.

Jak mít bezpečně instalovaný systém:

- Minimalizujte platformu a odstraňte z ní zbytečné prvky, nepoužité frameworky a komponenty, příklady a zbytečnou dokumentaci.
- Kontrolujte a aktualizujte konfigurace v souladu s bezpečnostními pokyny, obnoveními a záplatami.
- Automatizujte proces kontroly efektivity konfigurací a nastavení ve všech prostředích.

1.3.7 Cross-Site Scripting (XSS)

Cross-Site Scripting – injekce škodlivého skriptu, který se bude spouštět při načtení webové stránky. K útoku skriptování mezi weby k XSS dochází, když:

- Data se dostanou do webové aplikace prostřednictvím nedůvěryhodného zdroje, nejčastěji prostřednictvím webového požadavku.
- Data jsou zahrnuta v dynamickém obsahu, který je odeslán uživateli webu, aniž by byl ověřen na škodlivý obsah.

Útoky XSS lze obecně rozdělit do dvou kategorií podle směrů působení útoku: perzistentní a dočasné. Existuje třetí, mnohem méně známý typ útoku XSS nazvaný DOM [14] Based XSS.

Perzistentní útoky (XSS typu I) jsou útoky, kde je injektovaný skript trvale uložen na cílových serverech, například v databázi, ve fóru zpráv, v protokolu

[13] JSON Web Token – otevřený standard (RFC 7519) pro vytváření přístupových tokenů založených na formátu JSON. Obvykle se používá pro přenos dat pro ověření v aplikacích klient-server. Tokeny jsou vytvářeny serverem, podepisovány tajným klíčem a posílány klientovi, který tento token dále používá k potvrzení své identity.

[14] Document Object Model – nezávislé na platformě a jazykově programovací rozhraní, které umožňuje programům a skriptům přístup k obsahu dokumentů HTML, XHTML a XML a také umožňuje měnit obsah, strukturu a podobu takových dokumentů.

návštěvníků atd. Oběť poté načte škodlivý skript ze serveru, když požaduje uloženou informaci.

Dočasné útoky (XSS typu II) jsou útoky, ve kterých se injektovaný skript odráží mimo webový server, například v chybové zprávě, výsledku vyhledávání nebo v jakékoli jiné odpovědi. Dočasné útoky jsou obětem doručovány jinou cestou, například v e-mailové zprávě nebo na jiné webové stránce. Když je uživatel oklamán klikáním na škodlivý odkaz, odesláním speciálně vytvořeného formuláře nebo dokonce procházením škodlivého webu, injektovaný kód putuje na zranitelný web, což odráží útok zpět do prohlížeče uživatele. Prohlížeč poté provede kód, protože pochází z důvěryhodného serveru.

DOM Based XSS (XSS typu 0) je zranitelnost, která se objeví v DOM (Document Object Model) namísto části HTML. U perzistentních a dočasných útoků padá užitečné zatížení zranitelnosti na vrácenou webovou stránku oproti DOM Based XSS, ve které útok neovlivňuje HTML kód a k zatížení útoku nelze nalézt v odpověď. To lze pozorovat pouze za běhu nebo prozkoumáním DOM stránky.

Jak zabránit XSS útokům:

- Pomocí frameworků, které umějí escapovat XSS, jako např. Ruby on Rails nebo React JS. Zjistěte, proti kterým možnostem XSS-útoku se zvolený framework nemůže bránit, a tyto možnosti ošetříte sami.
- Escapujte nedůvěryhodná data z HTTP požadavku v kontextu vstupu HTML (tělo, atributy, JavaScript, CSS, URL). To vyřeší problém s perzistentními a dočasnými útoky.
- Použití je závislé na kontextu escapování při modifikaci dokumentu na straně klienta a působí proti DOM based XSS.
- Použijte Content Security Policy (CSP) jako zmírňující kontrolu proti XSS.

1.3.8 Insecure Deserialization

Nezabezpečená deserializace je zranitelnost, ke které dochází, když jsou nedůvěryhodná data použita ke zneužití logiky aplikace, k útoku na odmítnutí služby [15] nebo dokonce k provedení libovolného kódu po jeho deserializaci.

Jak se vyhnout nezabezpečené deserializaci:

[15] Denial of Service attack – druh útoku na počítačový systém za účelem jej znefunkčnit, tj. vytvořit takové podmínky, že reální uživatelé nebudou mít přístup k poskytovaným prostředkům, nebo bude tento přístup obtížný.

- Implementujte kontrolu integrity, jako jsou digitální signatury pro jakékoliv serializované objekty, aby se zabránilo tvorbě nepřátelských objektů nebo manipulaci s daty.
- Zajistěte striktní typizaci během deserializace před vytvořením objektu.
- Sledování deserializace, upozornění, pokud uživatel neustále deserializuje.

1.3.9 Using Components with known vulnerabilities

Známými zranitelnostmi jsou zranitelnosti, které byly objeveny v komponentách s otevřeným zdrojovým kódem a zveřejněny v National Vulnerability Database (NVD) [14], bezpečnostních doporučeních nebo systémech sledování problémů.

Jak zabránit používání komponent se známými zranitelnostmi:

- Odstraňte nepoužité závislosti, nepotřebné funkce, součásti, soubory a dokumentaci.
- Průběžně inventarizujte verze komponent na straně klienta i serveru (např. frameworky, knihovny) a jejich závislosti pomocí nástrojů, jako jsou Versions Maven Plugin, DependencyCheck, retire.js atd. Průběžně sledujte zdroje, jako jsou Common Vulnerabilities and Exposures (CVE) [15] a National Vulnerability Database (NVD), na zranitelnost v komponentách. K automatizaci procesu použijte nástroje pro analýzu složení softwaru.
- Komponenty získávejte pouze z oficiálních zdrojů přes zabezpečené odkazy. Preferujte podepsané balíčky, abyste snížili šanci na zahrnutí modifikovaných škodlivých komponentů.
- Monitorujte knihovny a součásti, které nejsou udržovány nebo nevytvářejí záplaty zabezpečení pro starší verze.
- Pokud oprava není možná, zvažte rozmístění virtuální záplaty pro sledování, detekci nebo ochranu před objeveným problémem.

1.3.10 Insufficient Logging and Monitoring

Je to bezpečnostní problém spojený s neschopností detekování a efektivního zpracování hrozeb, které vznikají kvůli nepostačujícímu logování a monitorování, pokud má systém nedetekovanou zranitelnost, kterou může použít útočník.

Jak se vyhnout problémům spojeným s nedostatečným protokolováním a monitorováním:

- Zajistěte, aby všechna selhání přihlašování, kontroly přístupu a selhání ověření vstupu na straně serveru mohla být zaznamenána s dostatečným uživatelským kontextem pro identifikaci podezřelých nebo škodlivých účtů a aby byla uchovávána po dostatečně dlouhou dobu tak, aby umožnila opožděnou analýzu.

- Zajistěte, aby byly protokoly generovány ve formátu, který lze snadno spotřebovat pomocí centralizované správy protokolů.

- Zajistěte efektivní monitorování a upozorňování tak, aby podezřelé činnosti byly detekovány a reakce na ně byla včasná.

- Zajistěte nebo osvojte si reakci na incidenty a plán obnovení, například NIST 800-61 verzi 2 nebo novější.

Existují bezpečnostní problémy, které nepatří do této klasifikace, ale velmi často se vyskytují v jiných; zanalyzujeme nejčastější z nich.

1.3.11 Cross Site Request Forgery (CSRF)

Cross Site Request Forgery (CSRF) – je druh útoku, ke kterému dochází, když škodlivá webová stránka, e-mail nebo program donutí prohlížeč uživatele provést nežádoucí akci na důvěryhodném webu, do kterého je uživatel aktuálně přihlášen. Nežádoucí akce bude provedena, pokud uživatel klikne na odkaz URL, který mu poskytne útočník např. v e-mailové zprávě. Útok CSRF nutí prohlížeč oběti, aby odeslala padělaný HTTP požadavek, včetně session cookie oběti a jiné zahrnuté autentifikační informace do zranitelné webové aplikace.

Jak zabránit CSRF útokům:

- Inicializace session pro každého uživatele, který náhodou navštívil webovou stránku. Na serveru se generuje dlouhý náhodný řetězec znaků pro každou relaci, která se nazývá token CSRF. Když uživatel vyplňuje formulář, který provádí citlivou akci, vloží se tento token jako skryté pole. Po odeslání formuláře je porovnána hodnota skrytého pole a hodnota uložená na serveru. Pokud se neshodují, žádost bude zamítnuta.

- Pouze pro vysoce citlivé operace doporučujeme také ochranu založenou na interakci uživatele (opakovaná autentizace, jednorázový token nebo CAPTCHA [16]).

[16] Completely Automated Public Turing test to tell Computers and Humans Apart – plně automatický veřejný Turingův test k odlišení počítačů a lidí.

1.3.12 Insecure Direct Object References (IDOR)

Insecure Direct Object References nastává, když aplikace vystavuje odkaz na vnitřní objekt realizace. Tímto způsobem odhalí skutečný identifikátor a formát použitý v prvku na straně backendu úložiště. Nejběžnějším příkladem je identifikátor záznamu v úložném systému (databáze, souborový systém atd.). IDOR umožňuje útočníkovi provést vyčíslení za účelem vyzkoušení přístupu k souvisejícím objektům.

Aby zranitelnost IDOR mohla být využita, musí se kombinovat s problémem řízení přístupu viz kapitola 1.3.5. Protože tento problém dovoluje útočníkovi získat přístup k tomuto objektu, jehož identifikátor byl uhádnut. [16]

Dle:[17] se lze zranitelnosti IDOR vyhnout následovně:

- Používejte odkazy na vnitřní objekty, které jsou nahrazeny kryptograficky silnými náhodnými hodnotami, které mapují původní hodnoty.
- Navrhněte řízení přístupu tak, aby uživatelé nemohli provádět žádné nedovolené akce.

1.3.13 Zabezpečení komunikace v počítačové síti

Obsahem této sekce je informace o tom, jak zabezpečit komunikaci pomocí Hypertext Transfer Protocol Secure (HTTPS) a zjistit, proč je doporučené používat HSTS, a vysvětlit, co to je.

HTTPS je rozšířením protokolu HTTP o podporu šifrování pro zvýšení bezpečnosti. Dané rozšíření využívá mechanismy SSL a TSL. HTTPS je určený pro zabezpečení proti útokům založeným na poslechu síťového připojení. To v podstatě znamená, že se stále používá HTTP protokol, ale pomocí bezpečného spojení SSL/TLS, které šifruje a dešifruje požadavky a odpovědi.

Spojení SSL/TLS mezi klientem a serverem vzniká pomocí procesu, který se jmenuje „potřesení rukou“ ruky (tzv. handshake). V tomto procesu musejí proběhnout následující akce:

- Klient musí být přesvědčen, že mluví se správným serverem, a někdy je vyžadováno i zpětné přesvědčení.
- Strany se musejí dohodnout na sadě šifer včetně algoritmu, který bude použit pro šifrování přenášených dat.
- Strany se musejí dohodnout na všech klíčích nezbytných pro tento algoritmus.

Jakmile je spojení uskutečněno, obě strany mohou používat vybraný algoritmus šifrování a klíče pro další bezpečnou komunikaci. [18]

Ovšem může vzniknout scénář, ve kterém se dá bezpečnost HTTPS spojení obejít. Tento útok je založen na obecně známém principu man-in-the-middle. [17]

Rozebereme si proces komunikace po HTTPS na příkladu připojení do poštovního servisu Google.

1. Klient se chce přihlásit na adresu `http://mail.google.com` pomocí HTTP.

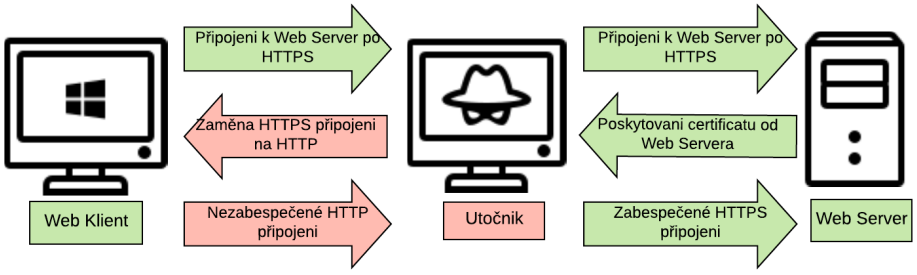
2. Server přesměruje klienta na HTTPS verzi poštovního servisu s kódem 302 přesměrování.

3. Klient se připojí na adresu `https://mail.google.com`.

4. Proběhne proces „potřesení rukou“ popsaný výše.

5. Vytvoří se šifrované připojení.

Nejvíce zranitelnou je operace přesměrování na HTTPS. Existuje speciální nastroj SSLStrip [19] který zachycuje data přenášená mezi klientem a serverem. Ve chvíli, kdy detekuje HTTPS URL adresu, nahradí jej HTTP odkazem a pak zachovává přenos všech dat, která přes toto spojení procházejí. Stroj využívající tento nástroj poskytuje data webovému serveru a představuje klienta. Data ze zabezpečeného web serveru jsou poskytována klientovi. Na obrázku 1.1 je znázorněn postup využitý útočníkem při dané zranitelnosti. [20, 21, 22]



Obrázek 1.1: Man-in-the-middle útok

Výše popsanému problému se dá vyhnout provedením příslušné politiky HSTS.

[17] Man in the middle – je druh útoku, kdy útočník tajně předává a v případě potřeby mění spojení mezi dvěma stranami, které se domnívají, že komunikují mezi sebou přímo.

HTTP Strict Transport Security (HSTS) – to je standard pro zabezpečení klienta, který zaručuje to, že browser se vždy musí připojovat k určité web adrese pomocí HTTPS. Tím pádem vylučuje scénář s přesměrováním z `http://` do `https://` [23].

1.3.14 Autentifikace

Tato podkapitola je věnována autentifikaci. Jejím cílem je popsat možné způsoby autentifikace, případné problémy s tím spojené a najít vhodný způsob řešení těchto problémů.

Autentifikace a autorizace

Autentifikace a autorizace jsou běžnými pojmy ve světě správy identit a přístupu (IAM) [18]. I když tyto pojmy mohou znít podobně, oba jsou odlišné bezpečnostní procesy a pochopení rozdílu mezi nimi je klíčem k úspěšné implementaci řešení IAM.

Autentifikace je akt, který potvrzuje, že uživatelé jsou tím, za koho se vydávají. Hesla jsou nejčastějším autentizačním faktorem – pokud uživatel zadá správné heslo, systém předpokládá, že identita je platná a poskytne přístup.

Autorizace v zabezpečení systému je proces udělování oprávnění uživatele přístupu ke konkrétnímu prostředku nebo funkci. Tento termín je často používán zaměnitelně s řízením přístupu nebo s právy klienta. Dobrým příkladem je dát někomu povolení ke stažení určitého souboru na serveru nebo poskytnout jednotlivým uživatelům administrativní přístup k aplikaci. [24]

HTTP authentication

HTTP podporuje použití několika autentizačních mechanismů pro řízení přístupu na stránky a další zdroje. Všechny tyto mechanismy jsou založeny na použití stavového kódu 401 a hlavičky WWW-Authenticate v odpovědi. Nejčastěji používanými mechanismy autentizace HTTP jsou:

Basic je princip, když klient odešle uživatelské jméno a heslo jako nešifrovaný text kódování base64. Měl by být používán pouze s HTTPS, protože heslo lze snadno zachytit a znovu použít přes HTTP.

Digest je princip, když klient odešle zahashované heslo na server. I když heslo nelze zachytit přes HTTP, je možné znovu přehrát požadavky pomocí hashe daného hesla.

[18] Identity and Access Management – sada přístupů, postupů, technologií a speciálních softwarových nástrojů pro správu pověření uživatele.

NTLM se používá jako bezpečný mechanismus požadavek/odpověď, který zabraňuje zachycení hesla nebo brání opakovaným útokům přes HTTP. Ověřování se však vztahuje na konkrétní připojení a bude fungovat pouze u trvalých připojení HTTP/1.1. Z tohoto důvodu nemusí fungovat ve všech proxy serverech HTTP a může znamenat velké množství obousměrného zpoždění, pokud jsou připojení k webovému serveru pravidelně uzavřená. [25]

Hesla nejsou bezpečná a neměla by se používat pro autentizaci samotná. Je těžké si je pamatovat, takže uživatelé mají pokušení používat slabá hesla a znovu je používat na více webech. I když je heslo silné, je to stále jen krátký řetězec, který uživatelé znají. Pokusíme se najít další mechanismy autentizace, které jsou považovány za více bezpečné. [26]

Autentifikace pomocí certifikátu

Autentifikace klienta pomocí certifikátu je proces toho, jak uživatelé bezpečně přistupují k serveru nebo vzdálenému počítači výměnou digitálního certifikátu. Digitální certifikát je považován za digitální ID a používá se ke kryptografické svázanosti totožnosti zákazníka, zaměstnance nebo partnera s jedinečným digitálním certifikátem (obvykle včetně jména, názvu společnosti a umístění vlastníka digitálního certifikátu). Digitální certifikát lze poté namapovat na uživatelský účet a použít k zajištění řízení přístupu k síťovým prostředkům, webovým službám a webům. [27]

Autentifikace pomocí certifikátu může být nevhodnou ze dvou důvodu. Autentifikace klienta předpokládá proces registrace, který není triviální ze strany klienta. Certifikát se vyrábí na jednom zařízení, takže pokud se klient bude chtít přihlásit z jiného zařízení, bude ho muset bezpečně přenést, což je také těžké zaručit. [28]

Dvoufaktorová autentifikace

Dvoufaktorová autentifikace je další metoda zabezpečení, která se používá jako doplněk k vašim přihlašovacím údajům na webech, které ji mají povolenou. Vyžaduje, abyste potvrdili, že se přihlašujete pomocí fyzického zařízení, jako je například mobilní zařízení, a prostřednictvím SMS nebo aplikace. [29]

Jednou z hlavních nevýhod dvoufaktorové autentizace je, že nemůže zajistit 100% ochranu před útoky hackerů. Navíc samotná technologie má slabá místa, která mohou kybernetičtí zločinci použít k obcházení její bezpečnostní vrstvy. Hackeři používají různé typy útoků, jako je malware, phishing, man-in-the-middle (MitM), nebo dokonce schémata obnovy účtu, aby se vyhnuli funkci 2FA nebo zachytili jednorázová hesla a softwarové tokeny.

Dvoufaktorová autentifikace přidává další krok, který musíte provést pokaždé, když se pokusíte přihlásit. Celkový proces ověření může trvat až jednu minutu, ale přesto může některé uživatele obtěžovat. [30]

Autentifikace pomoci tokenu

Tato metoda autentifikace se nejčastěji používá při vytváření distribuovaných systémů Single Sign-On (SSO), kde jedna aplikace (service provider) deleguje funkci ověřování uživatelů na jinou aplikaci (identity provider).

Implementace této metody spočívá v tom, že identity provider poskytuje spolehlivé informace o uživateli ve formě tokenu a aplikace service provider používá tento token k identifikaci, ověření a autorizaci uživatele.

Existuje několik standardů, které přesně definují protokol interakce mezi klienty a aplikacemi typu identity provider/service provider a formát podporovaných tokenů. Mezi nejpopulárnější standardy patří OAuth, OpenID Connect, SAML a WS-Federation. [31]

Standardy OAuth a OpenID Connect

OAuth je protokol autorizace, který dovoluje poskytnout třetí straně omezený přístup k zabezpečeným zdrojům uživatele bez poskytnutí přihlašovacích údajů (uživatelské jméno/heslo). Aktuální verze protokolu je OAuth 2.0, která byla publikovaná v 2012 roce.[1]

V dokumentaci protokolů OAuth jsou definované 4 role:

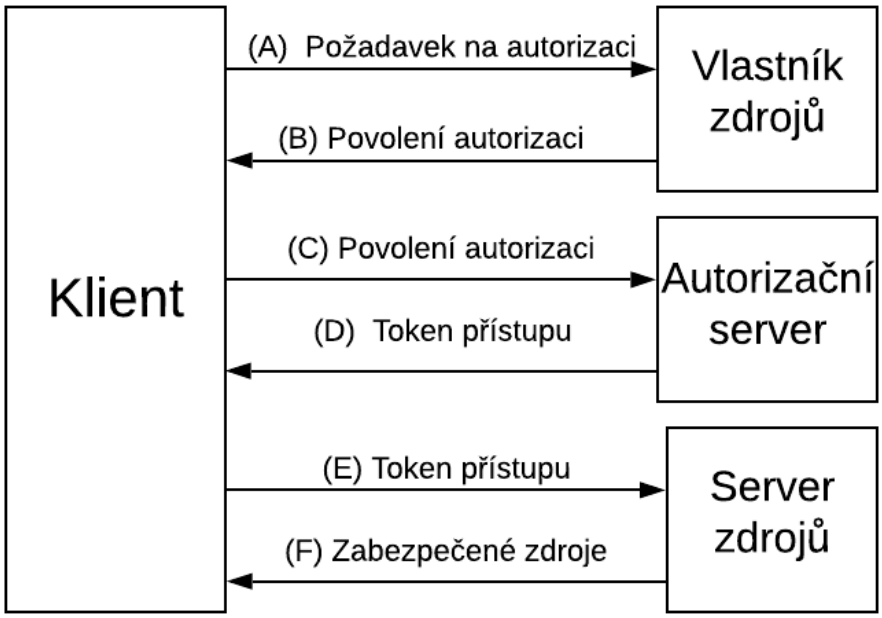
- Vlastník zdrojů – subjekt, který je schopen poskytnout přístup k zabezpečeným zdrojům.
- Server zdrojů – server, na kterém jsou rozmístěny zabezpečené zdroje.
- Klient – aplikace, která požaduje přístup k zabezpečeným zdrojům z pověření vlastníka zdrojů.
- Autorizační server – server, který vydává tokeny přístupu (tzv. access tokens) klientovi po úspěšné autentifikaci vlastníka zdrojů a povolení k přístupu.

Pro kompletní vysvětlení popíšeme interakci mezi těmito rolemi. Interakce se dá jednoduše rozdělit na následující kroky, které jsou znázorněny na obrázku 1.2:

(A) Klient posílá požadavek na autorizaci vlastníků zdrojů. Požadavek může být poslán přímo vlastníkovi, nebo přes autorizační server, který v tomto případě bude hrát roli zprostředkovatele.

(B) Klient dostává povolení k autorizaci, která reprezentuje autorizaci vlastníka zdrojů.

(C) Klient požaduje autorizační server o token přístupu na základě získaného povolení k autorizaci.



Obrázek 1.2: Abstraktní tok protokolu OAuth[1]

(D) Server autorizace identifikuje klienta a kontroluje, jestli má platné povolení. Pokud ano, vydává klientovi token přístupu.

(E) Klient žádá server zdrojů o zabezpečené zdroje na základě tokenu přístupu.

(F) Server zdrojů kontroluje platnost tokenu a zpracovává požadavek.

Dále se pokusíme popsat protokol OpenID.

OpenID Connect (OIDC) je ověřovací protokol založený na specifikacích rodiny OAuth 2.0. Používá jednoduché JSON webové tokeny (JWT), které můžete získat pomocí toků vyhovujících specifikacím OAuth 2.0.

Zatímco OAuth 2.0 je o přístupu ke zdrojům a sdílení, OIDC je hlavně o autentizaci uživatelů. Jeho účelem je poskytnout vám jedno přihlášení pro více webů. Pokaždé, když se potřebujete přihlásit na web pomocí protokolu OIDC, budete přesměrováni na web OpenID, na který se přihlásíte, a poté přejdete zpět na web. [32]

Shrnutí analýzy bezpečnosti

V dané kapitole jsme se seznámili s nejčastějšími bezpečnostními problémy ve světě webových aplikací a zjistili jsme jejich možná řešení. Teď jsme schopni analyzovat Baletku a zjistit, jak je navržena její bezpečnostní stránka.

1.4 Analýza bezpečnostní stránky aplikace Baletka

Předtím, než začneme analyzovat aplikaci Baletka, je potřeba zmínit, že Baletka je napsaná v jazyce Ruby z využitím frameworku Ruby on Rails, které budou popsány podrobněji v kapitole 1.5 V předchozí podkapitole jsme prozkoumali nejčastější problémy, které vznikají při zabezpečení aplikace, a zjistili jsme možná řešení těchto problémů. Kvůli shodě procesů hlasování porovnaných v sekci 1.2.2 je pro nás přínosné prozkoumat implementaci aplikace Baletka a podívat se na řešení bezpečnostních problémů. V této sekci prozkoumáme řešení, které používá Baletka na základě znalostí získaných v předchozí kapitole.

Je potřeba uvést, že cílem této kapitoly bylo obecné seznámení se s principy zabezpečení systému Baletka, abychom mohli použít tyto postupy k zabezpečení libovolné aplikace. Z tohoto důvodu nebyly některé části aplikace z předchozí kapitoly analyzovány. Předpokládáme zatím, že nástroje a způsoby jsou docela specifické, a záleží na použitém jazyce a frameworku, který je použit. Pokrytí těchto sekcí bude rozebráno při implementaci volebního systému v souvislosti s k tomu použitými nástroji.

Také vzhledem k časové náročnosti nasazení a zajištění správného fungování služeb jako GoSMS, ReCaptcha bylo rozhodnuto odkazovat v některých případech na jinou bakalářskou práci věnovanou analýze zabezpečení Baletky. [33]

1.4.1 Zabezpečení aplikace Baletka proti Injection

Cílem SQL injection je ovlivnit výsledky SQL dotazu, který místo očekávaného vstupu od uživatele dostane kus SQL kódu, jenž bude interpretován a proveden na serveru. Tedy útočník očekává, že slabým místem v aplikaci bude komunikace s databází. V aplikaci Baletka je tato komunikace zajištěna pomocí frameworku Active Record v Ruby on Rails.

Ruby on Rails má vestavěný (tzv. build-in) způsob obrany proti SQL Injections, který zajišťuje escapování symbolů jako ' , ", NULL, a řádkový zlom. V některých metodách escapování se používá automaticky, např.

```
User.find_by_some_attribute(attribute)
```

Ale ve fragmentech kódů obsahujících podmínky `(where("..."))`,

```
User.find_by_sql()
```

by se mělo zabezpečení provádět manuálně.

Pro vyčistění dat mohou být použity následující konvence:

```
Model.where("login = ? AND password = ?", entered_user_name, entered_password).first
```

Tato konvence v sobě obsahuje pole, ve kterém první polovina je fragmentem SQL s otazníky, za které budou dosazeny proměnné z druhé poloviny pole.

Ke stejným účelům můžeme využít alternativní syntaxi, která místo pole využívá hash mapu [34].

```
Model.where(login: entered_user_name, password: entered_password).first
```

Obě tyto konvence jsou využity při zabezpečení aplikace Baletka. Z toho důvodu nebyly po provedené analýze kódů nalezeny zranitelnosti spojené se SQL Injection.

1.4.2 Zabezpečení aplikace Baletka proti Broken Authentication

V předchozí kapitole v sekci 1.3.2 jsme definovali seznam kritérií pro rozhodnutí, jestli má webová aplikace zranitelnosti, které mohou být použité pro rozbití autentizace. Zkontrolujeme, jestli aplikace Baletka odpovídá těmto kritériím.

Přihlášení do aplikace Baletka probíhá ve dvou částech. První část autentizace je zajištěna pomocí uvedení e-mailu a hesla uživatelem a druhá probíhá zasíláním alfanumerického SMS kódu, který uživatel musí uvést do systému pro úspěšné dokončení přihlášení.[33]

Jak už víme z kapitoly 1.3.14 dvoufaktorová autentifikace zabraňuje provedení automatických útoku, brute force a opakovanému zneužití údajů. Takže v aplikaci Baletka jsou nastavena další omezení, která komplikují prolomení autentifikace. Z nich důležité jsou omezení počtu pokusů pro oba faktory, zavedení dočasného zablokování účtu v případě, že limit pokusu na přihlášeni je vyčerpán, a nastavení doby po vypršení, po které se uživatel musí znovu přihlásit.

```
1 config.max_login_attempts = 5
2 config.maximum_attempts = 5
3 config.unlock_in = 5.minutes
4 config.timeout_in = 30.minutes
```

Dalším kritériem je bezpečnost hesla.

Heslo má omezení na minimální a maximální délku (8 až 128 znaků) a také se kontroluje síla hesla a jestli se dané heslo vyskytuje v seznamu nejznámějších hesel. V aplikaci Baletka je za tuto činnost zodpovědný gem `devise_zxcvbn`.

Pro hashování v Baletce se používá bcrypt,[19] což je algoritmus, který má dobrou pověst a nebyl kompromitován.

Lze tvrdit, že aplikace Baletka je zabezpečena proti rozbití autentifikace.

1.4.3 Zabezpečení aplikace Baletka proti Sensitive Data Exposure

Jednou z důležitých vlastností volebního systému by měla být ochrana citlivých dat před neoprávněným přístupem.

Volební systém by měl chránit citlivé údaje a data před neoprávněným přístupem, modifikací či zničením. Navíc by měl chránit celistvost a skutečnost dat přenášených přes veřejné sítě.

Předtím, než začneme diskutovat o zabezpečení citlivých dat, musíme určit, co považujeme za citlivá data v případě volebních aplikací. Citlivost údajů demonstruje to, jaký potenciální dopad bude mít jejich zveřejnění. Není tajné, že citlivým údajem pro aplikace jsou hesla, protože je to způsob prokázání identity uživatele. I když pro přihlášení jako externí pracovník je potřeba dostat SMS kód, získání hesla velmi usnadňuje práci, kterou musí udělat útočník, aby se přihlásil do aplikace.

Hesla v Baletce jsou ukládána jenom pro externí pracovníky, jak bude uvedeno v podkapitole 1.4.9

Přihlášení pro zaměstnance ČVUT probíhá pomocí přesměrování na autorizační server FITu – Zuul – a autorizace je prováděna pomocí protokolu OAuth 2.0. Jak víme, jednou z důležitých odlišností daného protokolu je to, že přihlašovací údaje nejsou přenášeny po síti, ale jsou uložené na autorizačním serveru. Pro uložení hesla v databázi Baletky použit algoritmus hashování bcrypt, který byl podrobně popsán v podkapitole 1.4.2.

Další citlivý údaj je informace o volbě uživatele v případě, že hlasování je tajné. V databázi Baletky existuje tabulka `votes`, která reprezentuje vazby mezi voliči a odpověďmi. Naštěstí je pro tajné hlasování místo informace o voliči uváděna null hodnota. Takže při tajném hlasování nelze zkonfrontovat konkrétní odpověď a uživatele, protože mezi nimi není o tom uložena žádná vazba.

Dále popíšeme, jak je realizován přenos citlivých údajů po síti. Z kapitoly 1.3.13 víme, že pro zabezpečení komunikace pomoci HTTPS proti man-in-the-middle útokům se velmi doporučuje využití HSTS. Abychom mohli tvrdit, že Baletka má správně zabezpečenou komunikaci, musíme zjistit, jak správně nakonfigurovat HSTS pro Ruby on Rails aplikaci.

V Ruby on Rails je použití HSTS definovano v konfiguraci jako `config.force_ssl = true`. Stejnou konfiguraci lze pozorovat i v aplikaci Baletka:

[19] Bcrypt – kryptografická hashovací funkce odvození klíče. Funkce je založena na šifře Blowfish. Byla navržena Nielsem Provosem a Davidem Mazièresem a poprvé prezentovaná v roce 1999.

```
# Force all access to the app over SSL, use Strict-Transport-Security...
config.force_ssl = true
```

Tato konfigurace provede po její aktivaci sadu následujících akcí:[35]

- Označit všechny soubory cookie, nastavené aplikací jako zabezpečené.
- Odezva bude obsahovat záhlaví zabezpečení HSTS a donutí prohlížeč uživatele k provedení požadavků pouze prostřednictvím protokolu HTTPS.
- Všechny požadavky HTTP budou přesměrovány na HTTPS.

1.4.4 Zabezpečení aplikace Baletka proti Broken Access control

Řízení přístupu v aplikaci Baletka je zajištěno pomocí filtru. Filtry jsou metody, které jsou spouštěny „before“, „after“ nebo „around“ akce ovladače.

Filtry jsou zděděny, takže pokud zapnete filtr ApplicationController, bude spuštěn na každém řadiči ve vaší aplikaci. Filtry „before“ mohou zastavit cyklus žádostí. Běžný filtr „before“ je takový, který vyžaduje, aby byl uživatel přihlášen k provedení akce. [36]

```
class ApplicationController < ActionController::Base
  before_action :require_login

  private

  def require_login
    unless logged_in?
      flash[:error] = "You must be logged in to access this section"
      redirect_to new_login_url # halts request cycle
    end
  end
end
```

V Ruby on Rails lze snadno dostat seznam všech zvenku dostupných URL adres pomocí zadání `/rails/info/routes` od kořenové URL naší aplikace.

Testování přístupu už bylo prováděno v bakalářské práci Petra Nohejla, který nalezl zranitelnost na adrese začínající `ballot_templates`. [33] Při znovu provedeném testování bylo zjištěno, že tento problém byl úspěšné vyřešen, a nebyly nalezeny žádné další problémy s přístupovými body.

1.4.5 Zabezpečení aplikace Baletka proti Cross-Site Scripting

Jak už bylo zmíněno v kapitole 1.4.7 Ruby on Rails má vestavěný mechanismus obrany proti XSS, který je založen na automatickém escapování HTML pro všechna data, která jsou přenášena z Rails do HTML. Pro escapování v Rails je použita funkce `ERB::Util#html_escape`, která nahrazuje speciální znaky pro HTML jejich bezpečnými verzemi.

```
{
    '&' => '&amp;',
    '>' => '&gt;',
    '<' => '&lt;',
    '"' => '&quot;',
    "'" => '&#39;'
}
```

Například pro následující kód při zadání parametru „name“ hodnoty:

```
<script>alert(1)</script>
```

pro následující kód:

```
<b>Hello <%= params[:name] %></b>
```

budou výsledky HTML vypadat nasledovné:

```
<b> Hello &lt;script&gt;alert(1)&lt;/script&gt;gt; </b>
```

Jak je vidět, všechny speciální symboly pro HTML byly escapovány. [37]

1.4.6 Zabezpečení aplikace Baletka proti Insufficient Logging and Monitoring

Logovaní v systému Baletka je realizované pomocí Rails Loggeru, který je nastaven v defaultním formátu.

```
config.log_formatter = ::Logger::Formatter.new
```

Zajímavé ale je, že logger filtruje citlivá data, což zaručuje, že heslo a ověřovací kód se nevyskytnou v logu.

Dané chování je zajištěno pomocí funkce `make_random_identifier`,

```
1 # Configure sensitive parameters which will be filtered from the log file.
2
3 Rails.application.config.filter_parameters += [:password]
4 Rails.application.config.filter_parameters += [:code]
```

Je těžké rozhodnout, jestli je protokolování v aplikaci Baletka implementováno dostatečně, ale je rozumně nastavené a je ve snadno čitelném formátu, který lze bezproblémově použít pomocí centralizované správy protokolu, pokud by to vyžadovala situace.

1.4.7 Zabezpečení aplikace Baletka proti Cross Site Request Forgery (CSRF)

Zabezpečení proti CSRF v Rails je stanoveno na unikátním tokenu, který je součástí HTML stránky poslané uživateli a také je zachován v session cookie. Když uživatel zadá požadavek POST, token CSRF z HTML se odešle spolu s tímto požadavkem. Pokud se tokeny shodují na serveru, požadavek bude schválen. [38]

Realizace zabezpečení z pohledu programátora probíhá ve dvou krocích: přidání do `application_controller.rb` jednoduchého řádku kódu

```
protect_from_forgery with: :exception
```

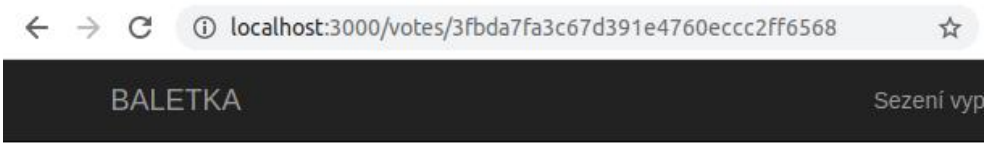
a přidání jednoho řádku do souboru `application.html.rb` [38]

```
<%= csrf_meta_tags %>
```

Předchozí dva kroky lze pozorovat i v aplikaci Baletka, z čehož se dá odvodit, že aplikace je zabezpečena proti CSRF útokům.

1.4.8 Zabezpečení aplikace Baletka proti Insecure Direct Object References (IDOR)

V aplikaci Baletka je realizované zabezpečení proti IDOR pomocí využití nepřímé mapy referencí (Indirect Reference Map). Tato metoda nahrazuje faktické identifikátory uživatelů alternativními identifikátory. Z obrázku 1.3



Obrázek 1.3: Znázorněni zabezpečeni v URL Baletky

je vidět, že místo reálného identifikátoru hlasování v URL se vyskytuje vygenerovaná posloupnost symbolu, která nedovoluje uživateli zjistit původní identifikátor.

Dané chování je zajištěno pomocí funkce `make_random_identifier`,

```
1     def make_random_identifier
2     random_string = SecureRandom.hex
3     while Ballot.find_by(vote_id_string: random_string).present?
4       random_string = SecureRandom.hex
5     end
6     self.vote_id_string = random_string
7   end
```

která pomocí funkce `SecureRandom.hex` generuje náhodný řetězec a pak jej ukládá do databáze.

Je potřeba zmínit, že takový postup je použit jenom pro uživatele systému, zatímco v URL pověřené osoby se vyskytují reálné identifikátory. Dá se předpokládat, že při návrhu Baletky bylo rozhodnuto, že korektní řešení je skrývat reálná data od pověřené osoby. Tím pádem můžeme tvrdit, že zabezpečení proti danému problému v Baletce je navržené korektně.

1.4.9 Proces přihlášení do systému Baletka

V této kapitole popíšeme nejdůležitější momenty spojené s přihlášením do aplikace Baletka. Více podrobný popis lze nalézt v bakalářské práci Petra Nohejla.[33]

Baletka umožňuje dva typy přihlášení: pro zaměstnance fakulty a externí pracovníky. Způsoby přihlášení se také liší v závislosti na uživatelích. Zaměstnanec ČVUT je pro přihlášení přesměrován na autorizační server FITu – Zuul – a autorizace je prováděna pomocí autorizačního protokolu OAuth 2.0.

Přihlášení externích pracovníků je realizované pomocí dvoufaktorové autentifikace. První faktor představuje klasickou kombinaci e-mailové adresy a hesla. Další část autentifikace je realizována zasláním SMS alfanumerického kódu, který uživatel potřebuje uvést do aplikace pro dokončení procesu přihlášení.

Zabezpečení aplikace Baletka je dobrým kompromisem mezi zajištěním bezpečnosti přihlášení a její použitelností. Postup přihlášení Baletky by mohl být dobrou inspirací pro budoucí návrh výsledného systému.

Shrnutí analýzy Baletky

Během analýzy aplikace Baletka jsme se setkali s konkrétními řešeními, které zajišťovaly bezpečnost systému. Později se ukáže, že principy a nástroje, které byly použity pro zabezpečení Baletky v Ruby on Rails, mají realizaci i ve frameworku Symfony, na kterém bylo pak rozhodnuto vyvíjet volební systém pro Mensu ČR. Analýza Baletky měla docela silný vliv na zabezpečení výsledné aplikace a lze pozorovat, že některá řešení uplatněná v Baletce posloužila jako inspirace pro návrh a realizaci mensovního volebního systému.

1.5 Analýza použitých technologií

V této sekci zanalyzujeme technologie, které bychom chtěli použít pro implementaci výsledné aplikace. Tedy provedeme analýzu frameworků a vybereme nejvíc vhodný z nich. Dále se seznámíme s technologiemi, které framework využívá.

1.5.1 Analýza frameworku

V této kapitole budou rozebrané jazyky a frameworky, které by bylo možno použít pro implementaci aplikace, a bude provedeno jejich porovnání, na základě něhož bude zvolen konkrétní framework.

Ruby on Rails

Ruby on Rails je framework pro vývoj webových aplikací napojených na databázi, používající architekturu model-view-controller. Vytvořil jej dánský programátor David Heinemeier Hansson při práci na projektu Basecamp.

Vše v Rails je založeno na jazyce Ruby. Na něm je založen Ajax v šablonách (view), odpovědi v controllerech i architektura aplikace v modelech obalujících databázi. Ke spuštění aplikace je třeba jen databáze.

Mezi základní principy Rails patří „Konvence má přednost před konfigurací“, tedy že programátor konfiguruje pouze ty části aplikace, které se liší od běžného nastavení. Vytvoří-li tedy např. model `Person`, aplikace bude data automaticky hledat v tabulce `People`. Chce-li, aby aplikace načítala data z tabulky staff, musí tak učinit výslovně.

Rails jsou postaveny na bázi architektury model-view-controller, která odděluje části aplikace zodpovědné za čtení a ukládání dat včetně manipulace s nimi (model) od zobrazení grafického rozhraní aplikace (view) a od části přijímající vstupy od uživatele a řídící zobrazení dat na výstupu (controller). [39]

Jedním z hlavních důvodů použití Ruby on Rails byla už realizovaná v tomto frameworku aplikace pro provedení hlasování vědecké rady FIT. Kvůli shodě aplikace Baletka s problémem vznikajícím při hlasování v organizaci Mensa lze očekávat, že některé principy a technologie realizované v Ruby on Rails je možné použít i při implementaci mensovní aplikace. To by mohlo ušetřit čas na vývoj aplikace.

Dalšími důvody použití Ruby on Rails jsou: [40]

- nástroje a knihovny, pomocí kterých lze realizovat vetší funkcionalitu za kratší dobu.
- umožňuje automatizaci testování.
- výmluvnost a výstižnost jazyka Ruby, a tedy možnost rychlejšího vývoje aplikací.

Existují však momenty, se kterými je potřeba počítat při volbě Ruby on Rails:

- Pro méně populární gemy může být obtížné najít dobrou dokumentaci. [41]
- Neznalost a nezvyklost syntaxe a principu fungování jazyku Ruby a frameworku Ruby on Rails. Vzhledem k tomu, že musíme navrhnout aplikaci řešící hlavně problémy spojené s bezpečností použití málo známého jazyka, může se zvětšovat náročnost implementace a komplikovat proces vývoje. V souvislosti s omezeným časem to může hrát důležitou roli.

Symfony

Symfony je framework, který je založený na open-source PHP projektech jako Propel, Doctrine, PHPUnit, Twig a Swift Mailer. Přesto má své komponenty jako Symfony YAML, Symfony Event Dispatcher, Symfony Dependency Injector a Symfony Templating. Od roku 2005 rostla společnost Symfony jako stále spolehlivější a vyspělejší framework. Používá se hlavně pro komplexní podnikové projekty. [42]

Důvody pro použití jsou následující:

1. Symfony je stabilní a osvědčené prostředí s pravidelnými aktualizacemi. Nejnovější verze zůstávají dlouhodobě podporovány a kompatibilní s ostatními verzemi: u některých verzí až 3 roky. [43]
2. Společnost SensioLabs, která se zabývá vývojem a podporou frameworku Symfony, navíc poskytuje celoživotní podporu v otázkách souvisejících s bezpečností. [44]
3. Symfony se integruje s knihovnou pro testování PHPUnit, která umožňuje unit a funkcionální testy. [44]
4. Symfony kombinuje preventivní bezpečnostní opatření k zabránění útoků a SQL injection. Integruje tyto bezpečnostní procesy jako vestavěná opatření. [45]

Dalším důležitým argumentem pro jsou vlastní zkušenosti. Framework Symfony byl používán v rámci předmětu BI-TWA, což způsobilo získání základních znalostí a pochopení principu fungování tohoto frameworku. Jazyk PHP běžné používám a mám znalosti o něm získané z předmětů BI-PHP a BI-TWA. Tohle by mohlo ušetřit čas na vývoj aplikace a možnost soustředit se na řešení problémů nezávislých na volbě jazyka a frameworku.

Symfony má samozřejmě i určité nedostatky: [46]

1. Navzdory své skvělé funkčnosti je Symfony spojena s poměrně nízkým výkonem, ke kterému dochází, když web nebo aplikaci používá mnoho

uživatelů současně. Ale vzhledem k tomu, že v Mense se účastní hlasování kolem 300 lidi během 2 měsíců, pravděpodobnost toho, že aplikace nebude schopna poskytnout své služby kvůli zátěži, je extrémně malá.

2. Symfony používá návrhové vzory, které vyžadují hodně dovedností a technických znalostí, jejich vytvoření a spuštění může chvíli trvat, například vytvoření kódu aplikace pro další části systému. Přestože jsou tyto funkce velmi užitečné při používání aplikací po spuštění, zaberou ve fázi návrhu hodně času.

Volba frameworku

Oba frameworky mají své určité výhody při použití a nedostatky, které podle mého názoru nejsou kritické. Výsledná aplikace by mohla byt implementovaná v libovolném z nich. Myslím si, že Symfony má vlastnosti, které bychom očekávali od frameworku vhodného pro implementaci volebního systému. Takovými vlastnosti jsou stabilita a osvědčenost frameworku, vhodné nastroje pro testování, vestavěné míry zabezpečení atd. Také mohou jako argument sloužit osobní zkušenosti se Symfony.

1.5.2 Další použité technologie

V této sekci budou popsány pomocné nástroje, které budou použité při implementaci.

Doctrine

Doctrine je ORM framework pro jazyk PHP. Nabízí vysokou míru abstrakce databázové vrstvy s použitím objektového přístupu. Umožňuje tak pracovat s daty jako s objekty. Jedna z klíčových vlastností Doctrine je psaní databázových dotazů použitím Doctrine Query Language (DQL), který vychází z ORM frameworku Hibernate určeného pro Javu. Doctrine je šířen pod licencí LGPL. [47]

Twig

Twig je šablonovací systém pro programovací jazyk PHP. Jeho syntaxe pochází z Jinja a Django šablon. Jedná se o produkt s otevřeným zdrojovým kódem licencovaný na základě licence BSD a spravovaný společností Fabien Potencier. Původní verzi vytvořil Armin Ronacher. Symfony PHP framework přichází s dodávanou podporou pro Twig od verze 2. [48]

Composer

Composer je správce balíčků na úrovni aplikace pro programovací jazyk PHP, který poskytuje standardní formát pro správu závislostí softwaru PHP a poža-

dovaných knihoven. Vyvinuli jej Nils Adermann a Jordi Boggiano, kteří tento projekt i nadále řídí. Vývoj zahájili v dubnu 2011 a poprvé jej vydali 1. března 2012. Composer je silně inspirován Node.js „npm“ a Ruby „bundler“. [49]

1.6 Shrnutí analýzy

V této části jsme prezentovali informace, které budeme potřebovat pro návrh výsledné aplikace. Probrali jsme oba volební procesy, což vedlo k analýze zabezpečení Baletky. Také jsme provedli analýzu zabezpečení webových aplikací. Tím pádem jsme získali postačující informace pro to, abychom byli schopni navrhnout bezpečný systém. Na konci jsme také zvolili framework a další technologie, které bychom využili při implementaci dané aplikace.

Kapitola 2

Návrh

Tato kapitola je věnována porozumění požadavkům organizace Mensa pro daný systém a na základě těchto požadavků sestavení návrhu volebního systému.

2.1 Analýza požadavků

V této sekci jsou zpracované požadavky organizace Mensa ČR, na základě kterých budeme navrhovat volební systém.

2.1.1 Aktéři

Pro návrh zavedeme tři druhy aktérů. Prvním druhem aktéra je nepřihlášený uživatel, ten může provést v podstatě jednu jedinou akci – přihlásit se. Po přihlášení se z nepřihlášeného uživatele mohou stát dva další druhy aktérů. Dalšími druhy aktérů jsou volič a člen volební komise. Cílem voliče v tomto systému je vyplňovaní hlasovacího lístků a odesílaní do výsledné aplikace, zatímco člen volební komise řeší administraci voleb.

2.1.2 Funkční požadavky

V této podkapitole budou rozebrány funkční požadavky na výsledný systém. Tedy identifikujeme jednotlivé úkony, aktivity a akce, které musí být vykonány.

F1 Autentifikace

1. Nepřihlášený uživatel bude mít možnost se přihlásit.
2. Volič a člen volební komise budou mít možnost se odhlásit.

F2 Volby

1. Volič bude mít možnost volit předsedu rady, radu a kontrolní komisi (KK).
2. Volič bude mít možnost zvolit maximálně tolik kandidátů, kolik je míst v radě nebo KK. V případě voleb předsedy rady může uživatel zvolit právě jednoho kandidáta.
3. Po skončení elektronického hlasování musí mít člen volební komise možnost přidat hlasy členů Mensy, kteří hlasovali na valné hromadě, ale pouze v den konání valné hromady.

F3 Backoffice

1. Člen volební komise bude mít možnost ukládat informaci o kandidátech.
2. Člen volební komise bude mít možnost změnit nastavení hlasování. Například změnit maximum kandidátů, které mohou uživatelé vybrat do rady nebo KK.
3. Člen volební komise bude mít možnost dostat výstup systému po skončení hlasování (seznam kandidátů a jejich hlasů, a seznam lidí, kteří volili).

2.1.3 Nefunkční požadavky

V této podkapitole budou rozebrány nefunkční požadavky na výsledný systém. Tedy určíme omezení a standardy kladená na systém.

N1 Systém musí být srozumitelný.

1. Volič musí být upozorněn na to, že jeho hlas je uložen.
2. Volič musí před finálním uložením vždy potvrdit svoji volbu.
3. Volič může hlasovat pouze jednou za dobu voleb a po odeslání své volby na server ji už nemůže měnit.
4. V každém kroku musí být jednoznačně a srozumitelně popsáno, co se chce v tomto kroku učinit.

N2 Systém musí být kompatibilní.

1. Systém musí podporovat mobilní verzi browseru.
2. Systém musí podporovat následující verze prohlížečů: Google Chrome verze 63, Mozilla Firefox 58, Edge 16, Explorer 11, a Safari 11.

N3 Systém musí mít vysokou anonymizaci.

N4 Systém musí mít nastavené přihlášení pomocí dvoufaktorové autentifikace, které je možné v případě potřeby vypnout.

2.2 Případy užití

V této sekci popíšeme jednotlivé případy užití. Případ užití zachycuje vztah mezi aktéry a funkčními požadavky. Pomocí případů užití se budeme snažit popsat nejdůležitější části výsledné aplikace a znázornit je pomocí diagramů případů užití.

2.2.1 Autentizace

Tato část pokrývá požadavky F1 a N5.

Hlavní scénář: Přihlášení s vypnutou dvoufaktorovou autentifikací

1. Nepřihlášený uživatel zadá své přihlašovací údaje, tedy konkrétně členské číslo (číslo, které jednoznačné identifikuje člena Mensy) a heslo a odešle je na server.

2. Proběhne verifikace na API Mensy, které je mimo volební systém a je připojené k databázi s údaji členů Mensy. Účelem tohoto API je rozhodování na základě získaného požadavku od volební aplikace, týkajícího se přihlášení, zda jsou přihlašovací údaje platné a zda daný člen Mensy může volit (jednou z příčin, proč nemůže volit, je například věk uživatele).

3. Mensovní API pošle odpověď volebnímu systému, který ji zpracuje a poskytne uživateli přístup do systému.

4. Nepřihlášený uživatel se přihlásí jako volič nebo jako člen volební komise.

5. Přihlášený uživatel, kterým je buď voličem, nebo členem volební komise, se může odhlásit, čímž se z něj stane nepřihlášený uživatel.

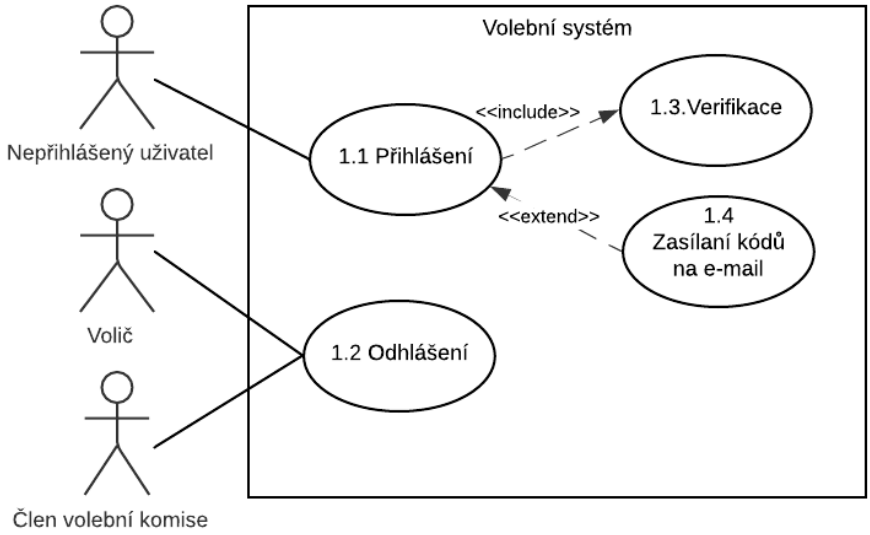
Alternativní scénář: Přihlášení se zapnutou dvoufaktorovou autentifikací

1. Scénář začíná ve 3. kroku hlavního scénáře. Volební systém dostane odpověď od mensovního API, a pokud verifikace proběhla v pořádku, uživateli bude poslán na e-mail ověřovací kód.

2. Uživatel tento kód zadá a pošle jej na server volebního systému.

3. V tuto chvíli o platnosti už rozhodne volební systém, a pokud je ověřovací kód správný, uživatel se přihlásí jako volič nebo jako člen volební komise.

4. Další krok je stejný jako krok 5 ve hlavním scénáři.



Obrázek 2.1: Use Case diagram Autentizace

2.2.2 Volby

Tato část pokrývá požadavky F2 a N4.

Hlavní scénář: Elektronické hlasovaní

1. Voliči se zobrazí volební lístek, ve kterém jsou uvedeni kandidáti pro různé orgány. Tedy kandidáti na předsedu, do rady a do kontrolní komise. Spuštěného hlasování se mohou zúčastnit pouze voliči.

2. Volič provede svoji volbu. To znamená, že dá svůj hlas určitým kandidátům. Volič může vybrat maximálně tolik kandidátů, kolik je míst v konkrétním orgánů. Například pokud se rada Mensy skládá z 11 lidí, volič může ve volebním lístku zvolit maximálně 11 kandidátů.

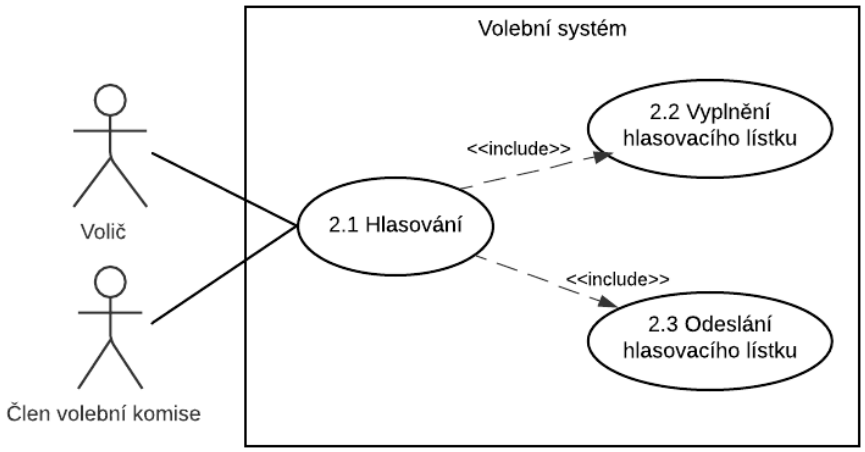
3. Volební lístek se odešle na server, kde bude zpracován.

Alternativní scénář: Hlasování na valné hromadě

1. Případ užitíi se začíná po skončení elektronického hlasování v den konání valné hromady. Na valné hromadě ti lidé, kteří se nezúčastnili elektronických voleb, mohou hlasovat fyzicky, to znamená, že musejí vyplnit volební lístek a hodit ho do urny.

2. Člen volební komise během dne přidá hlasy lidí, kteří volili na valné hromadě.

3. Při přidávání hlasů člen volební komise prochází hlavním scénářem, dokud nepřidá všechny hlasy.



Obrázek 2.2: Use Case diagram Volby

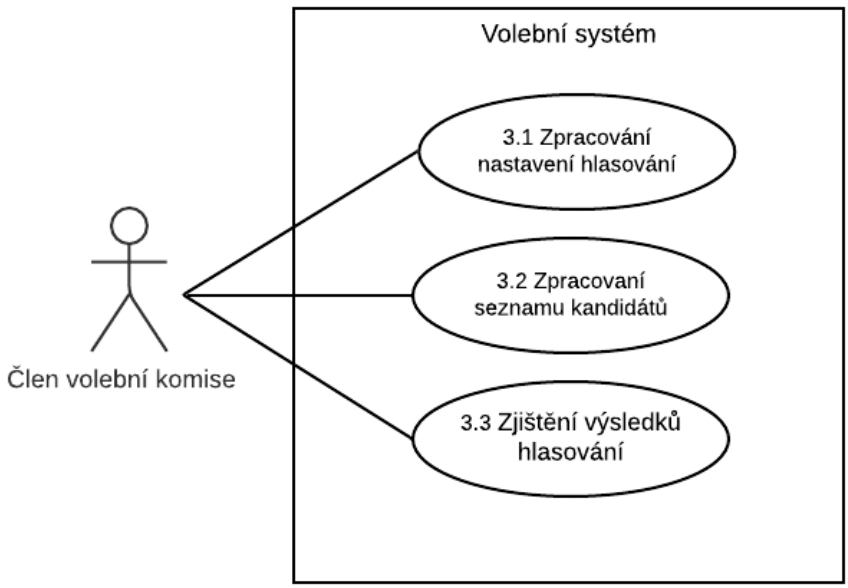
2.2.3 Backoffice

1. Člen volební komise přidá před konáním voleb do systému informace potřebné pro konání voleb. Jsou mezi nimi nastavení elektronického hlasování: doba, ve které bude hlasování proběhat, a počet míst v radě a kontrolní komisi. Dalšími informacemi, které musí uvést člen volební komise, jsou informace o kandidátech, což jsou údaje kandidáta, role v organizaci Mensa, proslov a číslo pořadí kandidáta v seznamu. Poslední údaj se generuje náhodně, ale není v rámci systému, proto je potřeba ho zadávat.

2. Člen volební komise vyplní volební lístek pomocí operací: vytváření, mazání a úprava jednotlivých kandidátů.

3. Po skončení elektronické fáze voleb bude člen volební komise oprávněn získat seznam členských čísel voličů, kteří se zúčastnili elektronického hlasování, a předběžnou informaci o výsledcích voleb.

4. Po proběhnutí valné hromady a uvedení do systému hlasů lidí, kteří fyzicky volili na valné hromadě, systém přepočítá přidělení hlasů mezi kandidáty a dovolí stáhnout finální verzi výsledku voleb.



Obrázek 2.3: Use Case diagram Backoffice

2.3 Wireframe

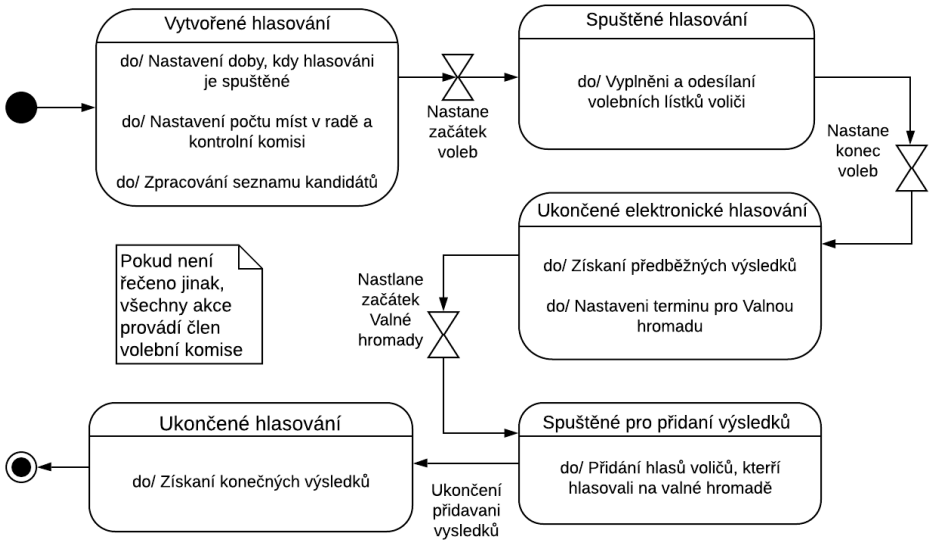
V rámci této bakalářské práce byly pro usnadnění komunikace se zákazníkem vytvořeny i wireframy, ale podrobně je probírat nebudeme. V případě zájmu je možné se je najít na přiloženém disku.

2.4 Diagramy vybraných procesů

V této sekci budou popsány diagramy nejsložitějších procesů, které potřebujeme vysvětlit před tím, než začneme popisovat implementaci volebního systému.

2.4.1 Fáze hlasování

Pro znázornění fáze hlasování byl vybrán stavový diagram. Na daném diagramu jsou uvedeny jednotlivé stavy, kterými musí hlasování projít, aby bylo korektně ukončené, a přechody mezi jednotlivými stavy, které znázorňují akci, kterou je potřeba provést pro přechod z jednoho stavu do druhého. Samotné stavy obsahují akce, které se provádějí v daném stavu a jsou uvedeny pod názvem stavu s využitím syntaxe „do/“, která je běžně používaná k těmto účelům[50, 51] .



Obrázek 2.4: Diagram stavů hlasovaní

Dále vysvětlíme, co popisuji jednotlivé stavy, jaké důležité akce v nich probíhají a kdo může tyto akce provádět. Podrobněji budou nejsložitější z nich znázorněny a popsány v aktivity diagramech, které budeme probírat v další podsekci.

Vytvořené hlasování

Tento stav reprezentuje hlasování, které ještě není spuštěné, to znamená, že voliči ještě nemají možnost hlasovat. V tomto stavu se členové volební komise musí postarat o přidání všech údajů potřebných pro hlasování. To znamená, že musejí nastavit dobu, odkdy dokdy by hlasování mělo být spuštěné, počet míst v radě a kontrolní komisi a přidat informaci o kandidátech. Pokud jsou nějaké údaje zadány špatně, člen volební komise je může opravit do té doby,

dokud nenastane začátek voleb. Spouštěním volebního procesu se dostáváme do stavu „Spuštěné hlasování“.

Spuštěné hlasování

V tomto stavu jsou hlavními účastníky voliči, kteří se mohou zúčastnit hlasování. Spuštěné hlasování se ukončí podle nastavené doby a stane se z něj „Ukončené elektronické hlasování“

Ukončené elektronické hlasování

Tento stav reprezentuje ukončenou první fázi hlasování. Je potřeba zmínit, že po ukončení elektronického hlasování členové Mensy, kteří nevolili elektronicky, mohou hlasovat fyzicky v den konání valné hromady. V tomto stavu již volič nemůže volit elektronicky a člen volební komise dostává možnost stáhnout předběžné výsledky voleb.

Další akcí, kterou musí provést člen volební komise, je nastavit termín konání valné hromady. Ve chvíli, kdy nastane termín konání valné hromady, členům volební komise bude zpřístupněna možnost přidávat hlasy voličů, kteří hlasovali na valné hromadě, do volební aplikace. Takže proces se přepne do stavu „Spuštěné pro přidání výsledků“

Spuštěné pro přidání výsledků

Z popisu předchozího stavu se může zdát, že člen volební komise může přidávat v den konání valné hromady libovolný počet hlasů. Naštěstí tomu není tak a počet hlasů, které mohou přidat členové volební komise, je omezen počtem lidi, kteří se zúčastnili fyzických voleb. Po ukončení přidání výsledku do systému se přejde na další stav – „Ukončené hlasování“.

Ukončené hlasování

V tomto stavu mohou členové volební komise dostat finální verzi výsledků voleb a žádné další akce se neprovádějí.

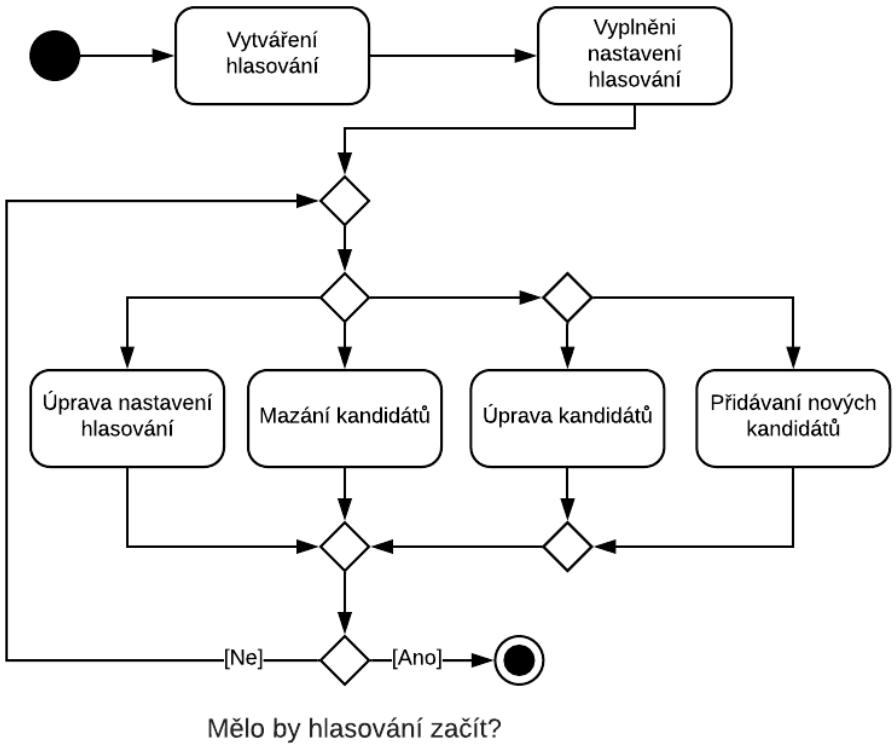
2.4.2 Znázornění stavů pomocí diagramu aktivit

V této podsekci znázorníme nejvíce složité stavy a podrobně je popíšeme. Těmito stavy pro volební systém jsou „Vytvořené hlasování“ a „Spuštěné pro přidání výsledků“.

Vytvořené hlasování

Jak je vidět z obrázku 2.5 po vytváření hlasování sledujeme vyplnění nastavení hlasování. Pro korektní chování hlasování je potřeba zadat dobu, odkdy dokdy se hlasování spouští, a maximální počet kandidátů, jež může volič zvolit pro

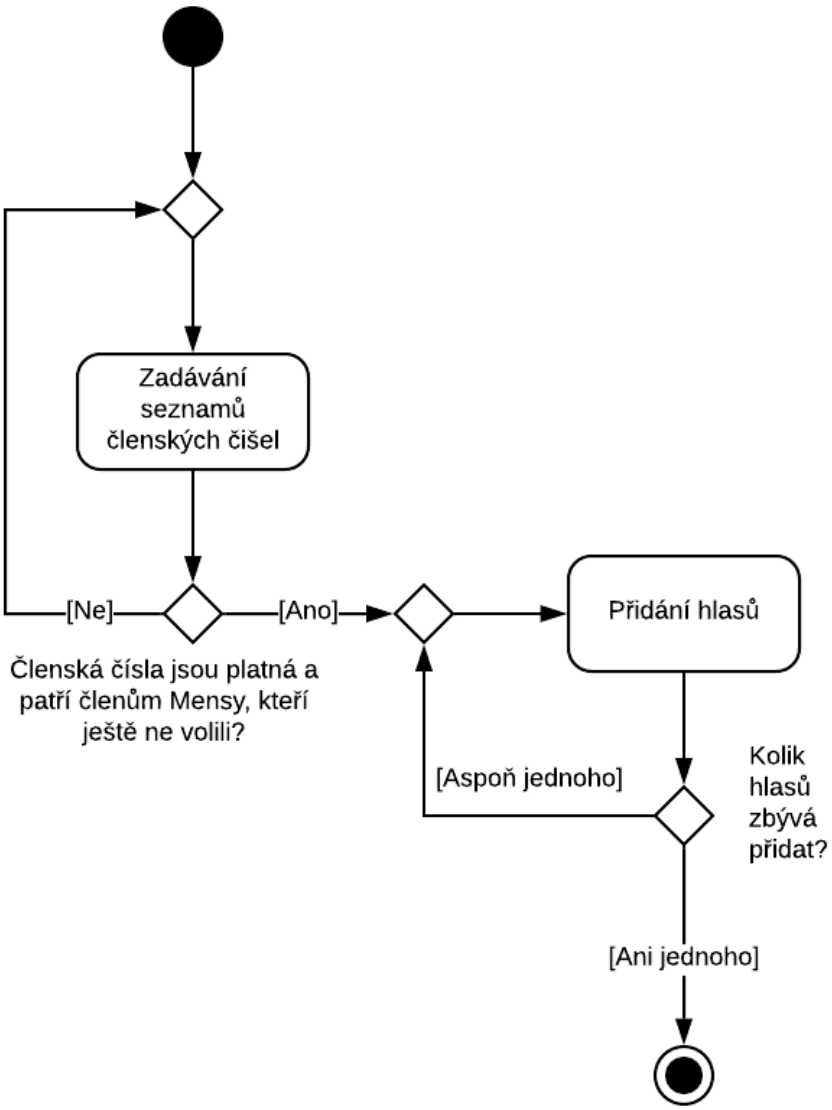
konkrétní orgán, tedy pro radu a kontrolní komisi. V případě volby předsedy se dá zvolit jenom jeden kandidát. Dále jsou zobrazeny čtyři události, které člen volební komise může provádět po vyplnění nastavení hlasování. Tyto události se mohou provádět opakovaně a nezávisle na ostatních, dokud nezačne hlasování.



Obrázek 2.5: Diagram aktivit pro vytváření hlasováni

Spuštěné pro přidání výsledků

V tomto stavu je první akcí, kterou musí člen volební komise provést, zadat do systémů členská čísla osob, které volily fyzicky na valné hromadě, jinak mu systém nedovolí přidávat hlasy jednotlivých voličů. Pokud člen volební komise zadá platná členská čísla lidí, kteří ještě nevolili, systém zpřístupní možnost přidávat hlasy. Hlasů lze přidat tolik, kolik členských čísel bylo zadáno předtím. Přidání hlasů probíhá stejným způsobem jako u voliče, který se zúčastnil voleb elektronicky.



Obrázek 2.6: Diagram aktivit pro přidaní hlasů lidi, které se zúčastnili voleb na Valné hromadě

Kapitola 3

Realizace

V této kapitole popíšeme technologie, které jsme použili pro realizaci frontendu a backendu, popíšeme, jak je implementované zabezpečení volebního systému, a ukážeme několik netypických řešení, která jsme měli realizovat kvůli požadavkům na tento systém a zlepšení použitelnosti dané aplikace.

3.1 Technologie a jejich použití

V této sekci popíšeme jazyky, frameworky a nejvíce zajímavé balíčky a knihovny, které jsme použili v této aplikaci. Ukážeme také, jak je použít, a označíme, jakou motivaci jsme měli při volbě určitých technologií.

3.1.1 Frontend

Na první pohled nebylo UI navržené pro volební systém složité, a proto nebyl k těmto účelům použit příliš komplikovaný framework. Bylo rozhodnuto použít pro vytváření uživatelského rozhraní populární řešení pro Symfony, kterým je šablonovací systém Twig. Dále byly pro stylizaci a animaci využity Bootstrap a JQuery spolu s jazyky CSS a JavaScript.

Twig

Je potřeba říct, že jsme definovali Twig jako jeden z dalších nástrojů v sekci 1.5 a nyní v doplnění k této informaci popíšeme princip a výhody použití.

Jádrem Twigu jsou šablony, které přidávají funkcionalitu do statické HTML stránky. V šabloně můžeme mimo statický HTML využívat podmínky, cykly, definovat makra a další pomocné syntaxe. Na obrázku 3.1 je vidět jednoduchou Twig šablonu. Twig využívá dva druhy oddělovačů `{% ... %}` a `{{ ... }}`. První z nich lze použít pro řídící konstrukce jako například podmínky. Druhý se používá pro výsledek příkazu, jako například `user.name` vyvolá funkci `getName()` u objektu třídy user a vrátí jméno uživatele.

```
<!DOCTYPE html>
<html>
    <head>
    <title>Welcome to Symfony!</title>
    </head>
    <body>
        <h1>{{ page_title }}</h1>

        {% if user.isLoggedIn %}
            Hello {{ user.name }}!
        {% endif %}

        {# ... #}
    </body>
</html>
```

Obrázek 3.1: Ukázka šablony Twig, převzata z [2]

Šablona Twig se vykreslí v kontroleru ve funkci render, do které lze jako druhý argument přidat pole proměnných, které chcete v této šabloně použít. Ukázku funkcí `render` lze najít na obrázku 3.2.

```
<?php
...
    return $this->render(
        'backoffice/candidate_new.html.twig',
        ['form' => $form->createView(),
            'foto' => '/images/icon-candidate.jpg',
            'backLink' => '/candidate-list-setting']
    );
```

Obrázek 3.2: Ukázka vytváření Twig šablony

Jak lze pozorovat z popisu Twig, jde v podstatě o způsob, jak vizualizovat a strukturovat staticky HTML lépe. K těmto účelům byl tedy Twig použit během implementace.

CSS a využití Bootstrap

Cascading Style Sheets (CSS) – jazyk, který popisuje styl HTML dokumentů. CSS je založen na selektorech, kterých je velké množství, ale jejich hlavním

účelem je popsat skupinu HTML elementů, na které musejí být aplikované určité styly. Na obrázku 3.3 jsou uvedeny tři nejpopulárnější typy selektorů, každý z nich pro zjednodušení ukázky definuje stejná pravidla, a to konkrétně zarovnání textu doprostřed, jeho obarvení modrou barvou a nastavení jeho rozměrů na 16 pixelů.

```
1  p{
2      text-align: center;
3      color: blue;
4      font-size: 16px;
5  }
6  #par1{
7      text-align: center;
8      color: blue;
9      font-size: 16px;
10 }
11 .center{
12     text-align: center;
13     color: blue;
14     font-size: 16px;
15 }
```

Obrázek 3.3: CSS ukazka selectoru

První selektor `p` aplikuje CSS pravidla na všechny elementy s tagem <p> v HTML dokumentu. Následující dva selektory aplikují CSS pravidla na HTML elementy s atributy `id="par1"` respektive `class="center"`.

Bootstrap – sada nástrojů pro tvorbu webových aplikací, která v sobě obsahuje šablony založené na HTML a CSS. Byl použit pro zjednodušení procesu stylizace HTML stránky a zmenšil náročnost stylizace pomocí CSS.

Využití čistého CSS je docela náročné, proto jsou použity nástroje jako Bootstrap, které poskytují předem definované atributy CSS třídy, jež mají předem nastavená CSS pravidla, což zbavuje programátora nutnosti psaní obrovských CSS souborů.

JavaScript a využití JQuery

JavaScript – „multi-paradigm“ programovací jazyk. Podporuje objektově orientované, imperativní a funkcionální styly. Je realizací standardu ECMAScript (ECMA-262)[20]. JavaScript je běžně používán ke zlepšení interaktivity webových stránek. Pomocí JavaScript je také často realizována validace vstupu

[20]ECMAScript – programovací jazyk, používaný jako základ pro vytváření jiných jazyků jako např JavaScript.

na straně klienta. Tyto dva způsoby použití byly uplatněny při realizaci uživatelského rozhraní volebního systému.

jQuery – knihovna JavaScript, která je soustředěná na manipulaci s dokumenty HTML, zpracování události a animaci. Je velice populární kvůli lakonické a jednoduché syntaxi. Na obrázku 3.4 je ukázka, jak zadat do HTML elementu s atributem `id="demo"` text „Hello World!“ z využitím JQuery a bez. JQuery zjevně zjednodušuje kód a využívá syntaxi CSS selektoru, která usnadňuje přístup k jednotlivým HTML elementům.

```
1 // JavaScript code
2 document.getElementById('demo').innerHTML = 'Hello, World!'
3
4 // Same with JQuery
5 $('demo').html('Hello, World!')
```

Obrázek 3.4: Porovnaní syntaxi JavaScript a JQuery [3]

3.1.2 Backend

V této podsekci budou popsány jak frameworky použité pro realizaci aplikace, tak i nejzajímavější balíčky, pomocí kterých byly realizovány takové části aplikace jako: logování, mailer a dvoufaktorová autentifikace.

Composer

Jak už bylo uvedeno jednou v sekci 1.5 Composer je balíčkovací systém, který se používá pro správu závislosti v PHP projektu. Všechny závislosti projektu jsou uvedeny v souboru `composer.json` který je vidět na obrázku 3.5. Tento soubor je klíčový při práci s Composerem. Správu závislosti lze provádět jak manuálně v souboru, tak i s použitím příkazové řádky, což velmi usnadňuje práci se závislostmi, a Composer je proto velmi užitečným nástrojem.

Doctrine

S Doctrine jsme se také setkávali v sekci 1.5 a víme o ní, že je to ORM, který usnadňuje komunikaci s databází tím, že zajišťuje vysokou míru abstrakce databázové vrstvy použitím objektového přístupu. Využití v Symfony projektu je realizováno jednoduše pomocí využití anotací.

Na obrázku 3.6. lze pozorovat příklad třídy entity (Entity Class) `Candidate`. Třída entity reprezentuje tabulku v databázi, zatímco entita je objekt PHP, který reprezentuje konkrétní záznam v dané tabulce.

```
{
    "type": "project",
    "license": "proprietary",
    "require": {
        "php": "^7.2.5",
        "ext-ctype": "*",
        "ext-curl": "*",
        "ext-iconv": "*",
        "ext-json": "*",
        "components/jquery": "^3.4",
        "doctrine/annotations": "^1.10",
        "myclabs/php-enum": "^1.7",
        "scheb/two-factor-bundle": "^4.15",
        "sensio/framework-extra-bundle": "^5.5",
```

Obrázek 3.5: Příklad souboru composer.json

Pomocí anotace `@ORM|Entity` je zajištěno, že tato třída bude namapovaná na tabulku v databázi. Mapování sloupců je vyřešeno pomocí přiřazení proměnným třídy anotace `@ORM|Column`, pomocí níž lze zadávat i další omezení, jako například typ sloupců, zda může být sloupec prázdný, a další. Doctrine na základě anotací vygeneruje tabulky v databázi, se kterými pak budeme moci pracovat.

Je potřeba zmínit, že Doctrine podporuje sadu databází a je nezávislá na tom, jakou z nich používáte, což je dalším argumentem, proč ji použít ve své aplikaci.

3.2 Realizace zabezpečení

V teto sekci uvádím informace o tom, jaká řešení byla použita během implementace, abychom zajistili bezpečnost naší aplikace. Sekce je strukturovaná podle bezpečnostních problémů, se kterými jsme se seznámili během analýzy.1.3

3.2.1 Zabezpečení přístupu do aplikací

Zabezpečení přístupu spočívá hlavně v omezení počtů stránek v aplikaci, kam se uživatel může dostat. Ukážeme několik způsobů, jak v Symfony omezit přístup uživatele k určitým zdrojům.

První způsob je vhodně anotovat jak jednotlivé metody v kontroleru, tak i celé kontrolery. Omezení přístupu je řešeno na základě rolí, které jsou přidělovány jednotlivým uživatelům. Na obrázku 3.7 je vidět, že omezení zajišťuje

```php
<?php
...
/**
 * @ORM\Entity(repositoryClass="App\Repository\CandidateRepository")
 */
class Candidate
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @Assert\NotBlank(message="entity.value.not_blank")
     * @ORM\Column(type="string", length=255)
     */
    private $name;
```

Obrázek 3.6: Příklad třídy entity (Entity Class) v Symfony

anotaci `@IsGranted("ROLE_ADMIN")`, která se vztahuje k celému kontroleru. Díky tomu není nikdo schopen přistoupit k žádnému URI, které se vyskytuje v tomto kontroleru, kromě těch, kdo mají nastavenou roli `"ROLE_ADMIN"`.

Pokud je přístup potřeba řešit pro několik rolí v rámci jedné metody kontroleru, lze to vyřešit způsobem ukázaným na obrázku 3.8. Pomocí metody `denyAccessUnlessGranted()` je zajištěné přesměrování všech uživatelů z rolí, které nejsou uvedeny v této metodě, na stránku přihlášení. Další omezení jsou přidávána metodou `IsGranted()`. Pomocí této metody se dá zkontrolovat, jestli má uživatel nějakou roli.

Dané metody jsou součástí kontroleru v Symfony a spolu s anotací poskytují kompletní zabezpečení přístupu do aplikace.

Dalším způsobem, jak omezit přístup, je přidat do souboru z konfigurace zabezpečení `security.yaml` jednotlivých cest a rolí, které mohou k těmto cestám přistoupit. Ve volebním systému je takovýmto způsobem zajištěn přístup k cestám, které jsou spojeny s přihlášením.

```php
<?php
...
/**
* @IsGranted("ROLE_ADMIN")
*/

class BackOfficeController extends AbstractController
{
...
    /**
     * @Route("/vote-setting",name="vote_setting",methods={"GET","POST"})
     * @param Request $request
     * @return Response
     */
    public function voteSetting(Request $request)
    {
```

Obrázek 3.7: Omezení přístupu pomocí anotací

3.2.2 Validace uživatelského vstupu

Jak už víme z analýzy zabezpečení aplikací a analýzy bezpečnostní stránky aplikace Baletka, validace vstupu je důležitou částí zabezpečení aplikace a v této části potřebujeme zajistit ochranu volebního systému proti SQL injekcím a XSS útokům.

Validace uživatelského vstupu se provádí jak na straně klienta, tak i na straně serveru. Pro oba tyto procesy ji lze jednoduše zajistit pomocí frameworku Symfony.

Validace na straně klienta je určena hlavně pro oznámení uživateli, že má chybu ve vyplněných údajích, předtím než je odešle na server. Ovšem lze to snadno obejít, například pokud vypnete JavaScript ve vašem browseru, a proto by se neměla používat bez validace na straně serveru.

V Symfony je validace uživatelského vstupu na straně klienta těsně vázaná na formuláře, do kterých uživatel zadává nějakou informaci. Tato validacemse provádí v třídě, která je zodpovědná za sestavení nějaké formy.

Jak je vidět z obrázku 3.9 validace klienta je definovaná v metodě `buildForm` pomocí definování typu u jednotlivých atributů, které očekáváme od uživatele. Například číslo pořadí kandidátů při výpisu, tedy proměna `serialNumber`, je omezená na celé číslo typem `IntegerType`, a pokud se uživatel pokusí zadat vstup, který není číslem validace na straně klienta, pak ho na to upozorní a nedovolí odeslat formu na server.

```php
<?php
...
/**
 * @Route("/", name="homepage",methods="GET")
 *
 */
public function ballot()
{
    $entityManager = $this->getDoctrine()->getManager();

    // if user is not autorized he will be redirected to login
    $this->denyAccessUnlessGranted('ROLE_ADMIN','ROLE_USER');

    // call entity manager
    $ballots = $entityManager->getRepository(Voting::class)->findAll();
    $currentUser = $this->security->getUser();
    // check roles
    if ($this->security->isGranted('ROLE_USER')) {
```

Obrázek 3.8: Omezení přístupu pro jednotlivou metodu kontroleru

Je vidět, že posledním parametrem funkce `add()` je pole parametrů, které definuje další pomocné parametry. Například parameter `required` nastavený na hodnotu `false` dovoluje uživateli nechat pole formuláře nevyplněné.

Dále popíšeme, jak probíhá validace na straně serveru.

Validace na straně serveru je zajištěná několika způsoby, které popíšeme v těchto bodech.

Za prvé, zabezpečení proti SQL injekci je zajištěné ve frameworku Doctrine. Je potřeba říct, že je zabezpečena jenom sada API, včetně všech hodnot uložených nebo změněných pomocí `Doctrine\ORM\EntityManager#persist()` a všech „find“ metod v `Doctrine\ORM\EntityRepository` [52]. Uvedené body pokrývají ve volebním systému použité funkce pro komunikaci s databází a dá se tvrdit, samozřejmě po provedení testování, že implementovaný systém je zabezpečen proti SQL Injections.

Zadruhé, v Symfony je validace uživatelského vstupu na straně serveru těsně vázána na třídy, pro jejichž instance provádíme validaci. Jedním ze způsobů, jak zajistit validaci v Symfony, je pomocí anotací. Jak je vidět z obrázku 3.10 jsou to anotace, které začínají na `@Assert`.

Symfony poskytuje sadu anotací, pomocí nichž se dá jednoduše a důkladně popsat omezení pro libovolnou proměnnou, například anotace `@Assert|Positive` definuje, že uvedené číslo musí být větší než nula. Validace v Symfony podpo-

```php
<?php
...
class CandidateType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('name', TextType::class,[])
            ->add('serialNumber',IntegerType::class,[
            ])
            ->add('memberNub', IntegerType::class,[
            ])
            ->add('phone', TextType::class,[
                'required'   => false,
            ])
            ->add('email', EmailType::class,[])
            ->add('contacts', TextareaType::class,[
                'required'   => false,
            ])
```

Obrázek 3.9: Validace vstupu na straně klienta v Symfony

ruje i regulární výrazy, pomocí nichž lze například definovat, jak musí vypadat telefonní číslo.

Také Symfony má vestavěné escapování speciálních symbolů pro HTML a nahrazení jejich bezpečnými verzemi 1.4.5, což je nejdůležitějším bodem pro zabezpečení proti XSS útokům.

Za třetí, validace v Symfony je mocným nástrojem pro validaci jednotlivých proměnných, ale nebyl nalezen nástroj pro validaci vztahů mezi jednotlivými proměnnými. Proto se validace vztahu mezi začátkem a koncem elektronických voleb řešila zvlášť. Jinak se žádné další problémy s validací neobjevily a dá se říct, že validaci v Symfony lze realizovat velmi snadno.

3.2.3 Zabezpečení citlivých dat

Za citlivá data budeme považovat stejně jako při analýze aplikací Baletka hesla a informace o tom, jak hlasovali jednotliví kandidáti. Ukládání hesel je realizováno jinak pro členy volební komise a jinak pro voliče. Pro voliče probíhá autentifikace pomocí mensovního API, a proto nejsou jednotlivá hesla voličů ukládána do databáze. Pro člena volební komise probíhá autentifikace standardním způsobem a pro hashování a solení hesel je využita funkce bcrypt, stejně jako v aplikaci Baletka.

```php
<?php
...
/**
 * @Assert\NotBlank(message="entity.value.not_blank")
 * @Assert\Positive(message="entity.number.positive")
 * @ORM\Column(type="integer")
 */
private $memberNub;

/**
 * @ORM\Column(type="string", length=255, nullable=true)
 * @Assert\Regex("/^\+?([0-9][- ]?){5,14}([0-9])$/",
            message="entity.phone.invalid_format")
 */
private $phone;

/**
 * @Assert\NotBlank(message="entity.value.not_blank")
 * @Assert\Email(message="entity.email.invalid_email")
 * @ORM\Column(type="string", length=255, nullable=true)
 */
```

Obrázek 3.10: Validace vstupu na straně serveru v Symfony realizovaná pomocí anotací

Zabezpečení hlasů voličů je navržené tak, že informaci o hlasech neukládáme do databáze, ale pamatujeme si počet získaných hlasů u jednotlivých kandidátů. Tímto způsobem je zajištěné jak zabezpečení citlivých údajů, tak i tajnost hlasování.

3.2.4 Přenos dat

Veškerá komunikace volebního systému probíhá s využitím HTTPS. V Symfony se dá jednoduše nastavit donucení použití HTTPS v souboru annotations.yaml nastavením parametru https pro všechny možné metody kontroleru, které používají anotace pro zadání URL adresy. Konfiguraci volebního systému se zapnutým HTTPS lze pozorovat na obrázku3.12.

Jednoduše lze ověřit, že donucení použití HTTPS probíhá s využitím HSTS. Pokud se pokusíte přihlásit zadáním nezabezpečené URL adresy, budete přesměrováni pomocí kódu 307, což je korektní chování browseru, pokud je na webové stránce zapnuto HSTS.

```php
<?php
...
if ($form->get('dateFrom')->getData() >= $form->get('dateTo')->getData()) {

                return $this->render('backoffice/vote_setting.html.twig', [
                    'form' => $form->createView(),
                    'dateTimeErrorMessage' => $this->translator
                        ->trans("Neni spravne nastaveny interval dvou dat")
                ]);
            }
```

Obrázek 3.11: Validace vstupu na straně serveru v Symfony řešena manuálně

```
controllers:
    resource: ../../src/Controller/
    type: annotation
    defaults:
        schemes: [https]
```

Obrázek 3.12: Konfiguraci volebního systému z zapnutým HTTPS

3.2.5 Autentifikace

Autentifikace probíhá dvěma způsoby v závislosti na roli uživatele systémů. Pro členy volební komise je realizované standardní přihlášení v Symfony, podrobnější informaci o něm můžete najít zde: [53]

Takže byl implementován Guard Authenticator, což je třída, která má na starosti proces přihlášení, jejž jsme potřebovali upravit, abychom byli schopni naimplementovat přihlášení voličů pomocí mensovního API.

Podrobněji bude proces přihlášení voličů rozebrán v podsekci „Propojení s API Mensy“. 3.3

Dalším zajímavým bodem je přidání druhého faktoru autentifikace do volebního systému. Tímto faktorem je zasílaní kódu e-mailem uživatelům systému. Dvoufaktorovou autentifikaci v Symfony zajišťuje balíček `scheb/two-factor-bundle`, který umožňuje upravovat a konfigurovat dvoufaktorovou autentifikaci podle vašich potřeb. Pro volební systém jsme nakonfigurovali dvoufaktorovou autentifikaci, tak aby umožňoval zaslání kódu e-mailem.

Na obrázku 3.13 lze pozorovat konfiguraci nastavenou pro volební systém. Je vidět, že jsou nastavitelné údaje jako délka kódu, zapínaní a vypínaní autentifikace, což v případě potřeby umožní zjednodušit neboli vypnout druhý

faktor autentifikace.

```
# See the configuration reference at https://github.com/scheb/two-factor-bundle/...

scheb_two_factor:
    security_tokens:

        # If you're using guard-based authentication, you have to use this one:
         - Symfony\Component\Security\Guard\Token\PostAuthenticationGuardToken
    email:
#        how long is code
        digits: 6
        enabled: false # If email authentication should be enabled, default false
        mailer: cus.mailer
```

Obrázek 3.13: Konfigurace dvoufaktorové autentifikace volebního systému

3.3 Netypická řešení

Během realizace volebního systému jsme většinu problémů řešili standardně pro framework Symfony a měli pokyny pro jejich řešení. Bylo i několik věcí, které jsme potřebovali řešit specificky.

Plovoucí záhlaví

Dané řešení bylo využito při realizaci volebního lístku. Problém byl v tom, že podle jednoho z požadavků Mensy by se měl volební lístek zobrazovat na jedné stránce a obsahovat kandidáty pro všechny pozice (tedy kandidáty na předsedu, do rady a kontrolní komise Mensy). Měli tedy být rozděleni do tří sekcí. Zdrojem problému bylo velké množství informací o kandidátech v jednotlivých sekcích a potřeba pamatovat si, kolik kandidátů už je zvoleno. Řešením bylo navrhnout tzv. plovoucí záhlaví, které by obsahovalo informace o tom, v jaké sekci se nachází volič a kolik může být v této sekci zvoleno kandidátů. Chování záhlaví bylo zkomplikováno manipulací, kterou je schopen provádět uživatel s rozměry okna v prohlížeči a zobrazením více informací o kandidátech. Korektním řešením bylo aktualizovat polohu záhlaví v závislosti na několika událostech, které může provádět uživatel, což bylo realizováno z využitím jazyku JavaScript.

Propojení s API Mensy

Mensovní API mělo na starosti verifikaci voličů, kteří se snaží přihlásit do volebního systému. Proces verifikace se skládá v podstatě ze zasílání přihlašovacích údajů volebním systémem do mensovního API a rozhodování o schválení přihlášení na základě od něj získané odpovědi.

Posílání požadavku je realizované pomocí sady funkci `curl_`, které simulují stejnojmenný příkaz v Bash. Odpovědí je jednoduchý json, který v případě úspěchu obsahuje platné údaje člena Mensy. V opačném případě obsahuje chybovou hlášku, kterou volební systém pak zobrazí uživateli. Kvůli specifičnosti daného postupu bylo nutné přepsat defaultní přihlášení a rozšířit jej o přihlášení pomocí mensovního API, což bylo snadno realizované pomocí Guard Autentificatoru, který Symfony nabízí pro tyto účely.

Kapitola 4

Testování

Tato kapitola je věnovaná testování volebního systému. Kapitola je strukturovaná do třech logických celků podle způsobu testování a k tomu použitých nástrojů. V dané kapitole je také popsané uživatelské testování, které bylo provedeno ve spolupráci s Mensou ČR.

Pro testování webové aplikace byl využit framework PHPUnit, který je populárním frameworkem pro testování systémů napsaných v jazyce PHP. Daný framework umožňuje provádět jak Unit testování, tak i testování integrační.

Během testování byly primárně prováděny integrační testy, které umožňovaly simulovat spolupráci jednotlivých celků aplikace. Dokonce nám to umožnilo kompletně nasimulovat proces hlasování od momentu jeho vytváření až do získání výsledků.

4.1 Integrační testy

Pomocí integračních testů byly testovány procesy, které probíhají během konání voleb jak pro voliče, tak i pro člena volební komise. Pomocí PHPUnit byl nasimulován uživatel, který prováděl různé činnosti jako odesílání formulářů, přechod mezi jednotlivými stránkami a další. Na obrázku 4.1 je vidět proces vytváření hlasování.

Z fragmentu kódu, který je na obrázku, lze pozorovat proces vytváření nového hlasování. Na začátku se uživatel, který je členem volební komise, přihlásí do systému a pošle GET požadavek na hlavní stránku volební aplikace. Dále se provede kontrola, že se opravdu přihlásil na domovskou stránku, a hlasování není vytvořené. Pomocí funkce `assertSame` zkontroluje, jestli jste opravdu dorazili na správnou stranku, tedy objeví se HTML element `<h1>`, který je nastaven na hodnotu „Nové hlasování“.

Dále je pomocí funkce `checkBallotSetting` zajištěna správnost vytváření a odesílání formy, obsahující v sobě všechny údaje potřebné pro vytvoření

```php
<?php
...
public function testBeforeBallotStarts(){
// login administrator
$this->client =  $this->fakeLogin->loginAdmin($this->client);
// delete current voting if exists
$this->DBCommunicationService->deleteBallotIfExist();
// go to homepage and check if ballot is not created
$crawler = $this->client->request('GET', '/');
$this->assertSame(Response::HTTP_OK, $this
->client->getResponse()->getStatusCode());
$this->assertSame('Nové hlasováni', $crawler->filter('h1')->text());

// check are routers OK
$this->checkAccessBeforeCreatingBallot();
// check form validation and create ballot
$this->checkBallotSetting
    ('/vote-setting','Vytvořit','Nastavení parametrů hlasování proběhlo úspešně!');

$crawler = $this->client->request('GET', '/');
$this->assertSame(Response::HTTP_OK, $this
->client->getResponse()->getStatusCode());
$this->assertSame('Vytvořené hlasováni', $crawler->filter('h1')->text());
```

Obrázek 4.1: Ukázka testování volebního procesu

hlasování. Fragment kódu vytváření formy ve funkci `checkBallotSetting` lze pozorovat na obrázku 4.2

Po vytvoření formy lze pozorovat, že při zasílání GET požadavku znovu na hlavní stránku se obsah elementu `<h1>` změnil na „Vytvořené hlasování“ což znamená, že jsme se úspěšné přepnuli do dalšího stavu hlasování.

Podobné integrační testy byly provedeny i pro další procesy, včetně procesu hlasování z pohledu uživatele, procesu získání výsledku a dalších. Pomocí integračních testů bylo ověřeno korektní chování systému během jeho použití.

4.2 Testování bezpečnosti

Testování bezpečnosti aplikace bylo prováděno v rámci integračních testů. Testování se týkalo hlavně řízení přístupu během volebního procesu, což proběhlo zasíláním sady požadavků na všechny přístupové body a ověření správnosti odpovědi volebního systému. Na obrázku 4.3 lze pozorovat fragment kódu, který měl na starosti kontrolu řízení přístupu.

```php
<?php
...
    $form = $crawler->selectButton($btnsubmit)->form();

    $form['voting_settings[dateFrom][date][month]']->select($monthFrom);
    $form['voting_settings[dateFrom][date][day]']->select($dayFrom);
    $form['voting_settings[dateFrom][date][year]']->select($yearFrom);
    $form['voting_settings[dateFrom][time][hour]']->select($hourFrom);
    $form['voting_settings[dateFrom][time][minute]']->select($minuteFrom);
    $form['voting_settings[dateTo][date][month]']->select($monthTo);
    $form['voting_settings[dateTo][date][day]']->select($dayTo);
    $form['voting_settings[dateTo][date][year]']->select($yearTo);
    $form['voting_settings[dateTo][time][hour]']->select($hourTo);
    $form['voting_settings[dateTo][time][minute]']->select($minuteTo);
    $form['voting_settings[chairmanNum]']->setValue($chairmanNum);
    $form['voting_settings[membersCounsilNum]']->setValue($counsilNum);
    $form['voting_settings[membersKKNum]']->setValue($comissionNum);

    $this->client->submit($form);

    $this->assertContains(
        $exceptedmessage,
        $this->client->getResponse()->getContent()
    );
```

Obrázek 4.2: Simulace odesílání formuláře v PHPUnit

Většina útoků, o kterých víme z analýzy, byla vyřešena na úrovni frameworku a stačilo jenom ověřit, jestli je zabezpečení proti některým útokům nastavené a funguje korektně. Do sady takových problémů patřily SQL injection, XSS útoky, CSRF, zabezpečení citlivých dat a nastavení HSTS. V případě zájmu lze najít detailní informace o tom, jak byla zabezpečena aplikace proti daným útokům, v příslušných sekcích kapitoly realizace 3

Útokem, kterému bychom měli zabránit, bylo použití komponent se známými zranitelnostmi. Kontrolu stability verze komponent, které přidáváte do svého projektu, má na starosti framework Composer, ve kterém se dá nastavit minimální stabilita verze komponentů, které chcete používat ve svém projektu, explicitně je tento parametr nastaven na `stable`, což znamená přidávání do projektu pouze stabilních verzí. Další nástroj pro kontrolu komponent v projektu poskytuje framework Symfony. Jedná se o webovou stránku[54], na které lze zkontrolovat, jestli soubor `composer.json` v sobě obsahuje komponenty se známými zranitelnosti.

```php
<?php
...
$this->assertRoteAccess('DELETE', '/candidate-delete/1',Response::HTTP_FORBIDDEN);
// if someone admin can to success page it is ok
$this->assertRoteAccess('GET', '/success-vote-setting',Response::HTTP_OK);
$this->assertRoteAccess('GET', '/mem-numbers-setting',Response::HTTP_FORBIDDEN);
$this->assertRoteAccess('GET', '/date-after-setting',Response::HTTP_FORBIDDEN);
$this->assertRoteAccess('GET', '/date-after-alterable',Response::HTTP_FORBIDDEN);
```

Obrázek 4.3: Testování řízení přístupu

Dalšími problémy, které ovšem nejsou zranitelnostmi, byly nedostatečné monitorování a logování a IDOR.

Pro volební systém bylo nastaveno standardní logování, které poskytuje framework Symfony, a stejně jako v aplikaci Baletka byly nastaveny výjimky pro zobrazení citlivých dat v logách. Je problematické otestovat, jestli je dané logování postačující pro Mensu, ale v případě potřeby umožní budoucímu správci systému relativně rychle vyhledat potřebné informace v logách aplikace.

Pro navržení zabezpečení proti IDOR bylo potřeba mít nějaký parametr, který by se vyskytoval v URI pro voliče. Ovšem jednotlivé URL, ke kterým volič může přistoupit, v sobě neobsahují žádné parametry, a tím pádem se dá tvrdit, že aplikace je z pozice voliče zabezpečena proti IDOR. Z pozice člena volební komise zabezpečení aplikace proti IDOR nemělo smysl, protože má přístup ke všem zdrojům, které může uhádnout vysledováním všech parametrů v konkrétních URL dané aplikace.

4.3 Uživatelské testování

V této sekci popíšeme jednotlivé problémy, které byly zjištěny během uživatelského testování, a jaký dojem učinila aplikace na uživatele.

Uživatelské testování probíhalo v organizaci Mensa a samotného testování se zúčastnilo celkem pět lidi, včetně mého vedoucího. Uživatelům byly přiřazovány úkoly jak z pozice Voliče, tak i s pozice členů Volební komise podle toho v jakém stavu aktuálně je hlasování. Nejvíce častý úkol se týkal vytváření hlasování, přidávaní kandidátů. Pak se uživatel měl přepnout do účtu, který patřil voliči, a se zúčastnit hlasování.

Většina chyb se týkala uživatelského rozhraní a byla spojena s nekorektním zobrazováním chybových hlášek (většinou v angličtině), neboli se jednalo o nevyhovující popisy u jednotlivých elementů (pole a tlačítka). Většina z těchto chyb byla opravena přímo v Mense a zbytek byl vyřešen poté.

Bylo také zjištěno několik neočekávaných a nepřivětivých chování systému, kterými byly zvětšení fotografie kandidáta, z čehož měl jeden z účastníků testování pocit, že je přesměrován na jinou webovou stránku, a zobrazení hlášky „potvrdit odeslání formy“ dříve, než se zobrazovaly chybové hlášky na straně klienta. Což také bylo vhodné upravit.

Také bylo zjištěno několik funkcí, které byly v systému nefunkční před konáním uživatelského testování. Jednou z těchto funkcí byl proces přihlášení do aplikace. Chyba byla způsobena změnou API ze strany Mensy, bylo potřeba provést aktualizaci přihlášení ve volebním systému a byla odstraněna na začátku uživatelského testování.

Další podstatnou chybou bylo nezapočítaní výsledků voleb. Tuto chybu způsobil překlep v anotaci metody, související s počítáním hlasů, což bylo také opraveno během voleb.

Uživateli byly navrženy i další změny v aplikaci, ale kvůli časovým náročnostem bylo rozhodnuto vyřešit je mimo bakalářskou práci. Podrobněji o nich se lze dozvědět v závěru práce 4.3

Celkově byli uživatelé spokojeni a volební systém splnil poslední z pokynů uvedených v zadání práce.

Závěr

Cílem práce bylo navrhnout a naimplementovat řešení projektu elektronického volebního systému Mensy ČR. Pro tyto účely jsme se seznámili s procesy, které probíhaly v aplikaci pro hlasování Vědecké rady FIT ČVUT, a porovnali je s procesem voleb v organizaci Mensa.

Na základě výsledků porovnání jsme došli k závěru, že je potřeba navrhnout odlišný systém, který by ale velmi podobně zajišťoval bezpečnost systému. Abychom mohli detailně prostudovat zabezpečení Baletky, potřebovali jsme se seznámit se zabezpečením webových aplikací obecně. Pro zjištěni potřebné teorie jsme prostudovali bezpečnostní stránku aplikace Baletka a narazili jsme na řešení, které ovlivnilo návrh a následující implementaci volebního systému.

Podle požadavků a pravidelných diskusí s organizací Mensa jsme navrhli a poté i naimplementovali volební systém, zajistili jeho nasazení na server Mensy a propojili s API, pomocí nějž Mensa vyžadovala zajistit verifikaci jednotlivých členů Mensy. Provedli jsme potřebné testování ve spolupráci s organizací Mensa.

Na základě provedených prací můžeme tvrdit, že jsme splnili původně zamýšlený cíl. To ale neznamená, že nemůžeme naše řešení zlepšit. Jedním z možných směrů by mohlo být zlepšení UI častí, které v aktuálním stavu splňují požadavky klienta, ale podle mého názoru nedostatečně přívětivě. Během uživatelského testování bylo navrženo několik drobností, které by neměly mít vliv na korektnost fungování volebního systému, ale usnadnily by volební proces. Například přidání validace data na straně klienta.

Další část, kam můžeme směřovat, je omezit provádění akcí, jako je přidávání, úprava a mazání, ze strany členů volební komise. Účelem těchto omezení je zmírnit činnost útočníků, ale tato omezení mohou způsobovat i komplikace a při jejich nesprávné implementaci mohou přinášet další nebezpečí. Proto bylo rozhodnuto je řešit velmi opatrně a jenom v případě potřeby.

Také jedním z návrhu, který zazněl v průběhu vypracování daného systému, bylo z jedno bezpečnostních vylepšení. Jedná se o hašování členských čísel, které můžeme také považovat za citlivá data.

Toto ale není úplný seznam věci, které by měly být vyřešené pro výsledný systém. V tom ale, doufám, budu pokračovat až po dokončení bakalářského studia.

Literatura

[1] Brian Eaton, D. H. A. T., Yaron Y. Goland: The OAuth 2.0 Authorization Framework. Dostupné z: https://tools.ietf.org/html/rfc6749

[2] SAS, S.: Creating and Using Templates [online]. Dostupné z: https://symfony.com/doc/current/templates.html

[3] Rascia, T.: How to Use jQuery, a JavaScript Library [online]. Dostupné z: https://www.taniarascia.com/how-to-use-jquery-a-javascript-library

[4] Chaeikar Saman, Z. M. A. M., Chukwuekezie Christian: Electronic Voting Systems for European Union Countries [online]. Dostupné z: https://www.researchgate.net/publication/260481185_Electronic_Voting_Systems_for_European_Union_Countries

[5] Ltd, P. C.: E-voting [online]. Dostupné z: https://www.evoting.ch/en

[6] ČR, M.: Stanovy Mensy ČR [online]. Dostupné z: http://www.mensa.cz/mensa/stanovy

[7] W3C: SQL Injection [online]. Dostupné z: https://www.w3schools.com/sql/sql_injection.asp

[8] OWASP: A1:2017-Injection [online]. Dostupné z: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A1-Injection

[9] of Standarts, N. I.; Technology: Memorized Secrets [online]. Dostupné z: https://pages.nist.gov/800-63-3/sp800-63b.html#memsecret

[10] OWASP: A2-Broken Authentication [online]. Dostupné z: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A2-Broken_Authentication

[11] NortonLifeLock: Sensitive data exposure [online]. Dostupné z: https://us.norton.com/internetsecurity-privacy-sensitive-data-exposure-how-its-different-from-data-breach.html

[12] OWASP: A3-Sensitive Data Exposure [online]. Dostupné z: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A3-Sensitive_Data_Exposure

[13] OWASP: Cheat Sheet Series[online]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html

[14] of Standarts, N. I.; Technology: National Vulnerability Database [online]. Dostupné z: https://nvd.nist.gov/

[15] Corporation, T. M.: Common Vulnerabilities and Exposure [online]. Dostupné z: https://cve.mitre.org/

[16] OWASP: Cheat Sheet Series[online]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

[17] Biako, E.: Insecure Direct Object Reference IDOR. Dostupné z: https://owasp.org/www-chapter-ghana/assets/slides/IDOR.pdf

[18] Heaton, R.: How does HTTPS actually work?[online]. Dostupné z: https://robertheaton.com/2014/03/27/how-does-https-actually-work/

[19] sslstrip. © 2012. Dostupné z: https://moxie.org/software/sslstrip/

[20] GlobalSign: What Is HSTS and How Do I Implement It?[online]. Dostupné z: https://www.globalsign.com/en/blog/what-is-hsts-and-how-do-i-use-it

[21] Ltd, N.: 95% of HTTPS servers vulnerable to trivial MITM attacks[online]. Dostupné z: https://news.netcraft.com/archives/2016/03/17/95-of-https-servers-vulnerable-to-trivial-mitm-attacks.html

[22] Wikipedie: Ataka posrednika [Man in the middle][online]. Dostupné z: https://ru.wikipedia.org/wiki/%D0%90%D1%82%D0%B0%D0%BA%D0%B0_%D0%BF%D0%BE%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B8%D0%BA%D0%B0

[23] government, T. U. S.: HTTP Strict Transport Security [online]. Dostupné z: https://https.cio.gov/hsts/

[24] Okta, I.: Authentication vs. Authorization [online]. Dostupné z: https://www.okta.com/identity-101/authentication-vs-authorization/

[25] Limited, N.: HTTP Authentication [online]. Dostupné z: https://www.httpwatch.com/httpgallery/authentication/

[26] Sevcsik-Zajácz, A.: Authentication using HTTPS client certificates [online]. Dostupné z: https://medium.com/@sevcsik/authentication-using-https-client-certificates-3c9d270e8326

[27] Johnson, K.: What is Client Authentication and Why Do I Need It? [online]. Dostupné z: https://medium.com/@sevcsik/authentication-using-https-client-certificates-3c9d270e8326

[28] Why is client certificate authentication not more common. In *Stack Exchange Inc.* Dostupné z: https://security.stackexchange.com/questions/198837/why-is-client-certificate-authentication-not-more-common?rq=1

[29] McKinnon, J.: Why Two-Factor Authentication Isn't Always Totally Secure [online]. Dostupné z: https://medium.com/@sevcsik/authentication-using-https-client-certificates-3c9d270e8326

[30] TSYKTOR, V.: Pros and Cons of Two-Factor Authentication [online]. Dostupné z: https://cyberpulse.info/pros-and-cons-of-two-factor-authentication/

[31] Vyrostkov, D.: Obzor sposobov i protokolov autentifikacii v web-prilozheniyah [Overview of authentication methods and protocols in web applications] [online]. Dostupné z: https://m.habr.com/ru/company/dataart/blog/262817/

[32] Auth0: OpenID Connect [online]. Dostupné z: https://auth0.com/docs/protocols/oidc

[33] Nohejl, P.: Zabezpečení hlasovací aplikace Baletka. 2018, vedoucí práce: doc. Ing. Štěpán Starosta, Ph.D. Dostupné z: https://alfresco.fit.cvut.cz/share/proxy/alfresco/api/node/content/workspace/SpacesStore/e671f2dd-94ac-4a7f-a6f2-35af02c28b02

[34] Hansson, D. H.: Securing Rails Applications [online]. Dostupné z: https://guides.rubyonrails.org/security.html

[35] Carletti, S.: Using HTTPs with Ruby on Rails [online]. Dostupné z: https://www.pluralsight.com/guides/using-https-with-ruby-on-rails

[36] Hansson, D. H.: Action Controller Overview [online]. Dostupné z: https://guides.rubyonrails.org/action_controller_overview.html

[37] Ltd, N.: Preventing Cross-site Scripting Vulnerabilities When Developing Ruby on Rails Web Applications [online]. Dostupné z: https://www.netsparker.com/blog/web-security/preventing-xss-ruby-on-rails-web-applications/

[38] Taylor, A.: A Deep Dive into CSRF Protection in Rails [online]. Dostupné z: https://medium.com/rubyinside/a-deep-dive-into-csrf-protection-in-rails-19fa0a42c0ef

[39] Wikipedia: Ruby on Rails [online]. Dostupné z: https://cs.wikipedia.org/wiki/Ruby_on_Rails

[40] Ganguly, S.: Pros Cons you must know before using Ruby on Rails for your startup.[online]. Dostupné z: https://hackernoon.com/pros-cons-you-must-know-before-using-ruby-on-rails-for-your-startup-234ecd631aaf

[41] Garg, N.: The Good and The Bad of Ruby Development [online]. Dostupné z: https://hackernoon.com/the-good-and-the-bad-of-ruby-development-8cr30fj

[42] BROTHERS, A.: Laravel vs Symfony in 2020 – which framework choose for your project? [online]. Dostupné z: https://asperbrothers.com/blog/laravel-vs-symfony/

[43] Lilly021: 7 good reasons to use Symfony framework for your start up project [online]. Dostupné z: https://lilly021.com/7-reasons-to-use-symfony-for-start-up-project/

[44] SHUKLA, S.: Five Reasons to Use Symfony Framework for Your Next Project [online]. Dostupné z: https://www.netsolutions.com/insights/symfony-framework-features/

[45] Krify: Disadvantages of Symfony [online]. Dostupné z: https://krify.co/tag/disadvantages-of-symfony/

[46] EXPERTSFROMINDIA.COM: The Good and the Bad of Symfony Web Development [online]. Dostupné z: https://www.expertsfromindia.com/tech-talk/technology/the-good-and-the-bad-of-symfony-web-development.html

[47] Wikipedia: Doctrine (PHP) [online]. Dostupné z: https://cs.wikipedia.org/wiki/Doctrine_(PHP)

[48] Wikipedia: Twig (template engine) [online]. Dostupné z: https://en.wikipedia.org/wiki/Twig_(template_engine)

[49] Wikipedia: Composer (software) [online]. Dostupné z: https://en.wikipedia.org/wiki/Composer_(software)

[50] SmartDraw: State Diagramy [online]. Dostupné z: https://www.smartdraw.com/state-diagram/

[51] Lynch, W.: State Diagram Comprehensive Guide with Examples [online]. Dostupné z: https://medium.com/@warren2lynch/state-diagram-comprehensive-guide-with-examples-e08b6d1c70fe

[52] Doctrine: Security[online]. Dostupné z: https://www.doctrine-project.org/projects/doctrine-orm/en/2.7/reference/security.html

[53] SAS, S.: Security [online]. Dostupné z: https://symfony.com/doc/current/security.html#b-authenticating-your-users

[54] Symfony: PHP security vulnerabilities monitoring[online]. Dostupné z: https://security.symfony.com/

Příloha A

Seznam použitých zkratek

API Application Programming Interface

BSD Berkeley Software Distribution

CVE Common Vulnerabilities and Exposures

CSRF Cross Site Request Forgery

CSP Content Security Policy

CSS Cascading Style Sheets

CORS Cross-Origin Resource Sharing

DTD Document Type Definition

DOM Document Object Model

DoS Denial of Service attack

DQL Doctrine Query Language

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

HSTS HTTP Strict Transport Security

IDOR Insecure Direct Object References

IAM Identity and Access Management

JSON JavaScript Object Notation

JWT JSON Web Token

JS JavaScript

KK Kontrolní komise

LGPL GNU Lesser General Public License

NVD National Vulnerability Database

NTLM NT LAN Manager

OWASP Open Web Application Security Project

ORM Object-Relational Mapping

OIDC OpenID Connect

SQL Structured Query Language

SSL Secure Sockets Layer

SSO Single Sign-On

TLS Transport Layer Security

URL Uniform Resource Locator

UI User Interface

XXE XML External Entities

XSS Cross-Site Scripting

XML Extensible Markup Language

XHTML eXtensible HyperText Markup Language

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF