ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

Systém pro anonymní hlasování na FEL ČVUT

Štěpán Bořek
Softwarové inženýrství a technologie

Květen 2022
Vedoucí práce: Ing. Ondřej Votava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Bořek** Jméno: **Štěpán** Osobní číslo: **483841**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Systém pro anonymní hlasování na FEL ČVUT

Název bakalářské práce anglicky:

System for Secret Voting at FEE CTU

Pokyny pro vypracování:

Seznamte se s nástroji a frameworky pro tvorbu interaktivních webových aplikací. Zaměřte se na možnost využití jedné aplikace jak pro desktop, tak pro mobilní zařízení. Implementujte nástroj, který umožní anonymní hlasování jak pro interní uživatele - přihlášení přes SSO, tak pro externí spolupracovníky. Aplikace umožní více kolová hlasování, každé kolo může nabízet jinou formu hlasování (výběr 1 až N z M položek apod.). Výsledný nástroj otestujte.

Seznam doporučené literatury:

[1] Databázový systém PostgreSQL - www.postgresql.org
[2] Vue.js - Progressive JavaScript Framework - v3.vuejs.org
[3] Socket.IO - Bidirectional and low-latency communication for every platform - www.socket.io
[4] Keycloak - Open Source Identity and Access Management - www.keycloak.org
[5] OpenID Connect https://openid.net/connect/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Ondřej Votava katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **02.02.2022** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2023**

Ing. Ondřej Votava podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování / Prohlášení

Chtěl bych především poděkovat vedoucímu mé bakalářské práce, panu Ing. Ondřeji Votavovi, za jeho vedení, konzultace a vstřícný přístup.

Dále bych také chtěl poděkovat všem účastníkům uživatelského testování, kteří svými postřehy přispěli k finalizaci vzniklé aplikace.

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

V Praze dne 20. 5. 2022

.....

Abstrakt / Abstract

Cílem této práce bylo analyzovat problematiku anonymního hlasování pomocí webových aplikací, existující řešení, a možné technologie a frameworky vhodné pro implementaci takové aplikace.

Dále také navrhnout, implementovat a otestovat single-page webovou aplikaci pro anonymní hlasování vhodnou pro využití na Fakultě elektrotechnické Českého vysokého učení technického v Praze.

Klíčová slova: anonymní hlasování; tajné hlasování; single-page web aplikace; Vue.js; Node.js; Socket.IO; Keycloak; PostgreSQL.

The main goal of this thesis was to analyze the issue of anonymous voting using webapplications, existing solutions, and technologies and frameworks which can be used to implement such applications.

Then design, implement and test a single-page webapplication usable for anonymous voting at Faculty of Electrical Engineering of Czech Technical University in Prague.

Keywords: anonymous vote; secret vote; single-page web application; Vue.js; Node.js; Socket.IO; Keycloak; PostgreSQL.

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

V mnoha případech je potřeba o určité otázce rozhodnout pomocí anonymního hlasování. Neexistuje mnoho webových aplikací pro jeho uskutečnění a je těžké ověřit, zda-li nakládají s daty opravdu tak, aby byla zajištěna anonymita. Cílem této práce je najít vhodné technologie pro tvorbu aplikací pro anonymní hlasování, analyzovat existující řešení, a na základě těchto poznatků navrhnout, implementovat a otestovat webovou single-page aplikaci vhodnou pro použití v rámci Fakulty elektrotechnické Českého vysokého učení technického v Praze (FEL ČVUT).

Toto téma mě zaujalo především možností vytvořit aplikaci s plně praktickým využitím, implementovanou za použití moderních webových technologií. Alternativní systémy mají mnoho nedostatků a proto je výhodné vytvořit řešení na míru požadavkům FEL ČVUT. Vytvořená aplikace může poskytovat funkce, které neposkytuje konkurence, a pro ověření uživatelů používat přihlášení skrze účet ČVUT a zároveň umožnit hlasování i hostům mimo ČVUT.

V rámci této práce se nejprve zaměřuji na problematiku anonymního hlasování a rozbor již existujících webových aplikací pro anonymní hlasování. Na základě této analýzy a zadání práce poté formuluji analýzu požadavků a případů užití.

Dále se zabývám návrhem architektury aplikace na základě předchozí analýzy, návrhem řešení a obrazovek. Popisuji kroky implementace navrženého řešení a způsob nasazení na server. Také uvádím informace o testování aplikace a jeho výstupech.

V závěru práce shrnuji naplnění cílů stanovených v zadání a možnosti pro další vývoj vytvořené webové aplikace.

Kapitola 2
Popis problematiky

2.1 Principy anonymního hlasování

Hlavním principem anonymního hlasování (také označovaného jako tajné hlasování) je nemožnost najít spojení mezi hlasující osobou a tím, pro co hlasovala [1]. Pouze konkrétní hlasující tedy ví, pro co hlasoval, nikdo jiný nesmí mít jakoukoliv možnost tuto informaci zjistit. Tato anonymita je narušena pouze v několika případech: všichni hlasující hlasují pro stejnou možnost, nikdo nehlasoval, hlasovala pouze jedna osoba, nebo hlasovaly pouze 2 osoby.

Tyto možnosti narušení anonymity lze omezit dvojitě anonymním hlasováním - nejenže nelze zjistit, kdo hlasoval pro jakou možnost, ale také nelze zjistit, zda vůbec daný hlasující hlasoval a kdo je přizván k hlasování [2]. Anonymita takového hlasování je narušena pouze ve 2 případech, a to pokud všichni možní hlasující hlasovali a to pro stejnou možnost, nebo nikdo nehlasoval.

2.2 Problematika webové aplikace pro anonymní hlasování

Mimo základních principů tvorby kvalitních a bezpečných aplikací je v případě online aplikace pro anonymní hlasování potřeba dbát především o správné nakládání s daty, jak vyplývá z principů anonymního hlasování.

Je třeba zajistit, aby samotný systém nejenže nezobrazoval, kdo jak hlasoval, ale také tuto informaci ani neukládal. Dále je třeba zajistit, že pouze osoby autorizované k účasti v daném hlasování mohou zaslat svůj hlas a to pouze jednou.

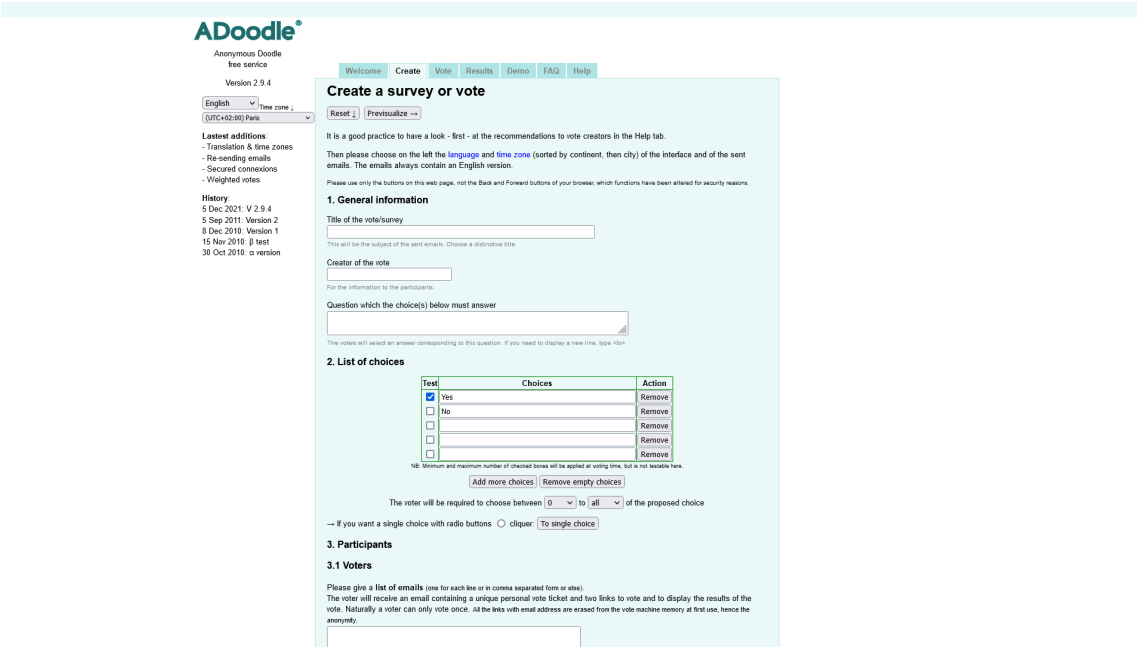
Při zpracování požadavku o provedení hlasování je tedy potřeba zkontrolovat autorizaci hlasujícího (tedy že má oprávnění hlasovat v rámci tohoto hlasování a ještě v něm nehlasoval), uložit informaci o tom, že právě provedl hlasování, a konečně v rámci hlasování uložit anonymní informaci o tom, pro co hlasující hlasoval.

V případě použití online hlasování pro velmi důležité volby (např. státní volby), při kterých hrozí velké riziko pokusu o kompromitaci systému, je velmi důležité důkladné zabezpečení, auditovatelnost a imutabilita výsledků, a důsledné ověření voličů. Je velmi těžké implementovat opravdu bezpečný systém a ačkoliv se v rámci implementace snažím o co nejlepší zabezpečení celé aplikace, tak mnou vyvíjená aplikace není určena pro velmi důležité volební procesy, nýbrž pouze pro anonymní hlasování na nižší úrovni.

Kapitola 3
Existující webové aplikace pro anonymní hlasování

V této kapitole se věnuji analýze již existujících řešení. Webových aplikací obecně umožňujících online hlasování existuje celá řada, pouze malá část je ale přímo založena na principech anonymního hlasování, zaměřuji se tedy pouze na tuto skupinu aplikací.

3.1 ADoodle



Obrázek 3.1. Screenshot tvorby nového hlasování v aplikaci ADoodle, zdroj [2]

Anonymous Doodle je webová aplikace umožňující anonymní hlasování. Funguje na principu dvojitě anonymního hlasování a je dostupná zdarma. Každý uživatel může bez registrace vytvořit hlasovací dotazník, který se skládá z názvu, autora, jedné otázky, možností odpovědí, nastavení odpovědí na single-choice a multiple-choice výběr, emailových adres hlasujících a pozorovatelů, časové hranice hlasování, a nastavení zobrazení výsledků.

Každý uživatel může hlasovat pouze jednou na základě unikátního klíče, který mu přijde na email uvedený při tvorbě hlasování. Informace spojující email s hlasovacím klíčem je smazána ihned při přístupu k hlasování. Seznam emailů hlasujících je smazán po ukončení hlasování.

Aplikace také umožňuje přístup k výsledkům hlasovaní i přihlížejícím (specifikovaným při tvorbě hlasování), kteří nemají možnost hlasovat.

Výhody aplikace ADoodle jsou především bezplatné použití, dvojitě anonymní hlasování a systém přihlížejících. Nevýhody spočívají v možnosti jedné otázky v rámci jednoho hlasování a fakt, že systém dvojité anonymity také nemusí být vhodný, pokud je třeba mít přehled o tom, kdo již hlasoval a kdo nikoliv.

3.2 Polys

Webová aplikace Polys umožňuje anonymní hlasování na principu blockchain technologií, čímž se odlišuje od ostatních kompetitorů. Blockchain zaručuje především transparentnost a nemožnost měnit data po provedení hlasování [3]. Vytvoření hlasování vyžaduje reigstraci.

Hlavní výhodou aplikace Polys je především imutabilita dat způsobená použitím blockchain technologií a možnost podrobného nastavení způsobu hlasování. Nevýhody spočívají především v ceně při hlasování ve středním nebo větším měřítku a možnost pouze jedné otázky v rámci jednoho hlasování.

3.3 Další aplikace

Existují také další aplikace, určené převážně pro volební procesy v komerčním sektoru. Mají pouze mírně odlišné funkcionality, pro použití ve středním a větším měřítku jsou placené a nelze ověřit, zda opravdu nakládají s daty tak, aby byla zajištěna anonymita hlasování i na straně serveru.

Mezi takovéto aplikace patří například Anonyvoter [4], Election Runner [5], nebo ElectionBuddy [6].

3.4 Shrnutí

Jak je vidět na výčtu existujících aplikací, webové aplikace pro anonymní hlasování sice existují, ale jejich nabídka je limitovaná. Z hlediska funkcionalit jsou existující aplikace jen mírně odlišné a pouze jediná z nich je plně použitelná zdarma.

Implementace vlastního řešení umožňuje vytvořit systém s podporou vícekolových hlasování, více otázkami v rámci jednoho kola, lepší integrací do systémů ČVUT, a také možností dalšího rozvoje aplikace dle potřeb uživatelů.

Kapitola 4
Analýza požadavků webové aplikace

V této kapitole se zabývám hlavními požadavky na aplikaci a případy užití z hlediska jednotlivých typů uživatelů. Přesné stanovení všech požadavků je kritické pro samotný návrh a vývoj aplikace.

4.1 Funkční požadavky

Funkční požadavky ukazují, jaké hlavní funkce musí aplikace poskytovat, aby plnila svou funkci [7]. Může jít například o uživatelské funkce, požadavky na autentizaci či autorizaci, nebo shromažďování dat.

Některé funkční požadavky vycházejí už ze zadání, další z analýzy problematiky a již existujících aplikací. Výčet funkčních požadavků aplikace uvádím v tabulce 4.1. Každý požadavek obsahuje ID (unikátní identifikátor), název a popis.

4.2 Nefunkční požadavky

Nefunkční požadavky představují kvalitativní atributy aplikace a nevztahují se přímo k jejím funkcím [7]. Jde většinou o požadavky na dostupnost, výkon, použitelnost, zabezpečení, nebo plnění právních předpisů.

Seznam nefunkčních požadavků aplikace uvádím v tabulce 4.2. Každý požadavek obsahuje ID, název a popis.

ID	Název	Popis
NP01	Anonymita hlasování	Aplikace nebude ukládat informace umožňující určit, jak který hlasující hlasoval.
NP02	Responsivní design	Uživatelské rozhraní aplikace se bude přizpůsobovat dle typu uživatelova zařízení.
NP03	Kompatibilita	Aplikace bude kompatibilní s prohlížeči Mozilla Firefox a Google Chrome.
NP04	Uživatelská přívětivost	Aplikace umožní uživateli snadnou navigaci mezi sekcemi pomocí navigačního panelu.
NP05	Zabezpečení	Aplikace bude dostupná pouze skrze zabezpečené připojení. SSL (Secure Sockets Layer) certifikát bude ověřený důvěryhodnou autoritou. Přístupová hesla nebudou uložena v čistém textu, ale budou uložena zahashována.

Tabulka 4.2. Nefunkční požadavky.

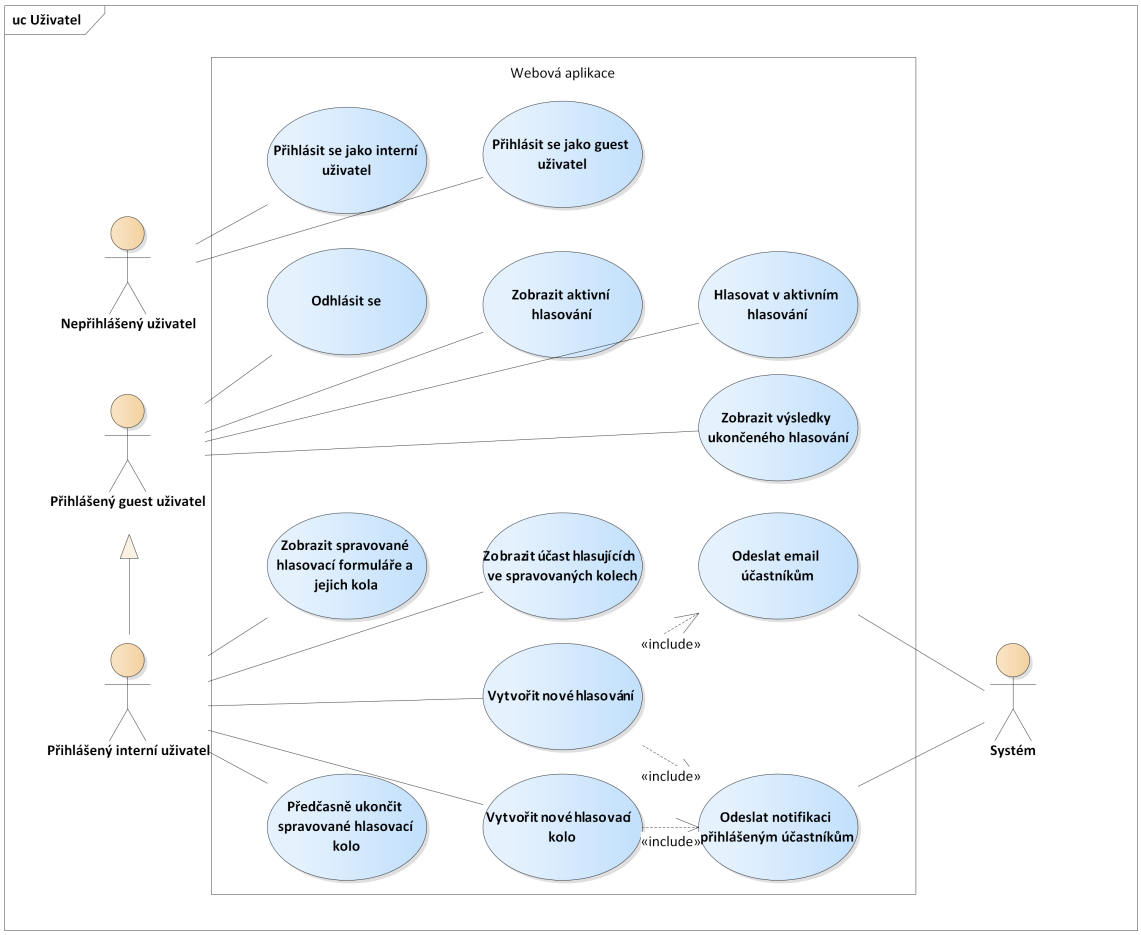
ID	Název	Popis
FP01	Přihlášení interního uživatele	Aplikace umožní přihlášení internímu uživateli ČVUT skrze ČVUT SSO (Single sign-on) bránu.
FP02	Přihlášení guest uživatele	Aplikace umožní přihlášení guest uživateli pomocí přihlašovacího jména a hesla.
FP03	Odhlášení uživatele	Aplikace umožní odhlásit se internímu i guest uživateli.
FP04	Zobrazení aktivních hlasovacích kol	Aplikace umožní internímu i guest uživateli zobrazit aktivní hlasovací kola, ve kterých má právo hlasovat.
FP05	Hlasování v aktivním hlasovacím kole	Aplikace umožní internímu i guest uživateli hlasovat v aktivním hlasovacím kole, ve kterém má právo hlasovat a ještě v něm nehlasoval.
FP06	Zobrazení výsledků ukončených hlasovacích kol	Aplikace umožní internímu i guest uživateli zobrazit výsledky ukončených hlasovacích kol, ve kterých měl právo hlasovat, nebo která založil.
FP07	Vytvoření nového hlasovacího formuláře	Aplikace umožní internímu uživateli vytvořit nový hlasovací formulář a přizvat účastníky k hlasování.
FP08	Zobrazení spravovaných hlasovacíh formulářů	Aplikace umožní internímu uživateli zobrazit jím spravované formuláře včetně informace o tom, kdo již hlasoval.
FP09	Vytvoření nového hlasovacího kola	Aplikace umožní internímu uživateli vytvořit nové hlasovací kolo jím spravovaného hlasovacího formuláře.
FP10	Ukončení spravovaných hlasovacíh kol	Aplikace umožní internímu uživateli předčasně ukončit jím spravované hlasovací kolo.
FP11	Notifikace přihlášených uživatelů	Aplikace notifikuje přihlášené uživatele, jakmile vznikne nový hlasovací formulář nebo kolo, ve kterém mají oprávnění hlasovat.
FP12	Přizvání hlasovacích účastníků	Aplikace odešle účastníkům nového hlasování email s přihlašovacími údaji k novému hlasovacímu formuláři při jeho vytvoření.

Tabulka 4.1. Funkční požadavky.

4.3 Případy užití

Diagram případů užití ukazuje chování aplikace z pohledu koncového uživatele. Znázorňuje akce, které uživatel může provést a jejich případné napojení na další aktéry (např. aplikaci samotnou, jiné systémy, nebo čas), ale neobsahuje přesnou informaci o tom, jakým způsobem jsou tyto akce implementovány, ani pořadí akcí nutných pro dosažení specifického cíle [8].

Diagram 4.1 ukazuje případy užití aplikace a propojení aktérů. Tabulky 4.3, 4.4, 4.5, 4.6 obsahují popisy jednotlivých případů užití pro určité aktéry.



Obrázek 4.1. Diagram případů užití

ID	Název	Aktér	Popis
UC01	Přihlásit se jako guest uživatel	Nepřihlášený uživatel	Uživatel vyplní získanými přihlašovacími údaji přihlašovací formulář a odešle jej. Po úspěšném přihlášení jsou mu zpřístupněny další funkce aplikace.
UC02	Přihlásit se jako interní uživatel	Nepřihlášený uživatel	Uživatel klikne na tlačítko „SSO Login“, čímž bude přesměrován na ČVUT SSO bránu, kde se přihlásí svým ČVUT účtem. Po úspěšném přihlášení je přesměrován zpět do aplikace a jsou mu zpřístupněny další funkce aplikace.

Tabulka 4.3. Tabulka případů užití - Nepřihlášený uživatel

UC03	Odhlásit se	Přihlášený guest uživatel	Uživatel klikne na tlačítko „Logout“ a je odhlášen ze systému.
UC04	Zobrazit aktivní hlasování	Přihlášený guest uživatel	V sekci aktivních hlasování se uživateli zobrazí seznam aktivních hlasovacích formulářů a jejich kol, ve kterých má oprávnění hlasovat. Uživateli jsou také zobrazeny detaily vybraného hlasovacího formuláře a vybraného kola.
UC05	Hlasovat v aktivním hlasování	Přihlášený guest uživatel	Po zobrazení aktivního hlasování (viz UC04), ve kterém ještě uživatel nehlasoval, může uživatel vybrat své volby a odeslat je tlačítkem „Vote“.
UC06	Zobrazit výsledky ukončeného hlasování	Přihlášený guest uživatel	V sekci výsledků se uživateli zobrazí seznam již ukončených hlasovacích formulářů a kol, ke kterým má uživatel přístup. Uživatel si může zobrazit detaily výsledků hlasovacího kola, výsledné počty a poměry hlasování jsou znázorněny textem i grafem.

Tabulka 4.4. Tabulka případů užití - Guest uživatel

UC07	Zobrazit spravované hlasovací formuláře a jejich kola	Přihlášený interní uživatel	V sekci správy hlasování se uživateli zobrazí seznam hlasovacích formulářů a kol, která vytvořil. Uživatel si může také zobrazit přehled detailů vybraného hlasovacího formuláře.
UC08	Zobrazit účast hlasujících ve spravovaných kolech	Přihlášený interní uživatel	V sekci správy hlasování (viz UC07) se u každého kola uživateli zobrazuje, kdo již hlasoval a kdo nikoliv.
UC09	Vytvořit nový hlasovací formulář	Přihlášený interní uživatel	V sekci tvorby nového hlasovacího formuláře může uživatel vytvořit nový hlasovací formulář s prvním hlasovacím kolem. Specifikuje detaily nového hlasovacího formuláře a prvního kola, vyplní emailové adresy přizvaných hlasujících a formulář vytvoří tlačítkem „Create“.
UC10	Vytvořit nové hlasovací kolo	Přihlášený interní uživatel	V sekci správy hlasování (viz UC07) může uživatel v rámci již existujícího hlasování vytvořit nové kolo vybraného hlasovacího formuláře. Vyplní detaily nového kola a vytvoří ho kliknutím na tlačítko „Create“.
UC11	Předčasně ukončit spravované hlasovací kolo	Přihlášený interní uživatel	V sekci správy hlasování (viz UC07) může uživatel v přehledu kol předčasně ukončit ještě neukončené kolo spravovaného hlasovacího formuláře.

Tabulka 4.5. Tabulka případů užití - Interní ČVUT uživatel

UC12	Odeslat email účastníkům	Systém	Po vytvoření nového hlasovacího formuláře (viz UC09) systém odesílá emailem pozvánku k hlasování přizvaným hlasujícím. Pozvánka obsahuje údaje pro přihlášení jako guest uživatel.
UC13	Odeslat notifikaci přihlášeným uživatelům	Systém	Po vytvoření nového hlasovacího formuláře (viz UC09) nebo nového kola (viz UC10) systém odesílá notifikaci všem přihlášeným uživatelům, kteří mají oprávnění v novém hlasování hlasovat.

Tabulka 4.6. Tabulka případů užití - Systém

4.4 Shrnutí

V této kapitole jsem stanovil funkční a nefunkční požadavky, které musí aplikace splňovat. Na základě těchto požadavků jsem dále stanovil případy užití aplikace z hlediska různých aktérů.

Vzniklý soubor požadavků a případů užití tvoří základ návrhu samotné aplikace, dle kterého v dalších částech práce rozebírám návrh řešení a implementaci funkcí.

Kapitola 5
Architektura webové aplikace

V této kapitole se zabývám volbou vhodné architektury pro vyvíjenou aplikaci. Architektura určuje propojení komponent celého systému, ukazuje tedy jakým způsobem mezi sebou interagují uživatelská rozhraní, databáze (DB), middleware, nebo servery v rámci systému [9]. Zvolení vhodné architektury pro aplikaci je důležitým krokem od kterého se poté odvíjí návrh řešení a implementace aplikace.

5.1 Backend

Monolitický

Monolitická BE (Backend) architektura je stále značně rozšířená, ačkoliv je používána už velmi dlouhou dobu. Aplikace zajišťující BE funkce je plně implementována vývojáři jako jeden celek, který je poté spuštěn na serveru. Toto řešení je náročnější na vývoj a údržbu než serverless architektura a hůře se škáluje než microservice architektura, ale umožňuje největší volnost implementace funkcionalit a správy dat.

Microservice

Aplikace založená na microservice architektuře se skládá z mnoha volně propojených dílčích komponent, z nichž každá je udržována samostatně [10]. To umožňuje jednodušší škálování a znovupoužitelnost komponent ve více aplikacích. Běžně FE (Frontend) komunikuje s API (Application Programming Interface) bránou, která se posléze dotazuje na další API jednotlivých komponent.

Serveless

Při použití serverless architektury je delegován běh a funkce BE na jiného poskytovatele [9]. Vývojáři stačí poté implmentovat FE část aplikace a nastavit, jak se má chovat poskytovaný backend. Toto řešení snižuje množství práce vývojářů aplikace a umožňuje dobrou škálovatelnost a stabilitu (zajištěnou poskytovatelem služby).

Serverless architektura je rozdělena do 2 kategorií - Backend-as-a-Service (BaaS) a Function-as-a-Service (FaaS):

- BaaS je model, ve kterém jsou všechny aspekty BE části delegovány na poskytovatele BaaS služby [11]. Poskytovatel poté zajišťuje pro FE všechny potřebné věci, jako správu databáze, cloudové úložiště, hosting, autentizaci, notifikace apod. Tento přístup umožňuje vývojářům plně se soustředit jen na FE část a co nejméně se zabývat BE částí.
- FaaS je model, ve kterém se poskytovatel služby stará o běh serveru, databáze apod., zatímco vývojáři aplikace implementují FE a samostatně spustitelné funkce v rámci FaaS, které se pak z FE volají [12]. To umožňuje vývojářům nemuset se starat o běh BE serveru, ale pouze o jeho funkcionality. Tento přístup je často používán při vývoji microservice aplikací.

5.2 Frontend

Single-page

SPA (single-page aplikace) nenačítá každou stránku webové aplikace samostatně, ale pracuje nad jedinou stránkou, přičemž FE na pozadí provádí dotazy na BE API a dynamicky dle výsledků zobrazuje obsah, čímž velmi zvyšuje uživatelský komfort a plynulost navigace v aplikaci [9].

Progresivní

PWA (progresivní webové aplikace) jsou založeny na principu SPA, ale rozšiřují ho o mnoho klíčových požadavků, které by měly zaručit co nejjednodušší použití pro koncové uživatele [13]. Mezi tyto požadavky patří např. objevitelnost veškerého obsahu internetovými vyhledávači, nezávislost na internetovém připojení, dostatečná funkcionalita i na starších zařízeních, zabezpečená komunikace, plně responsivní design, možnost instalace do zařízení, nebo sdílení skrz odkaz.

5.3 Shrnutí

Microservice architektura není úplně vhodná pro vývoj aplikace, jelikož vyvíjená aplikace se nedotazuje na mnoho jiných API a znovupoužitelnost jejích částí nemá velký význam. Z tohoto důvodu také nemá smysl využívat FaaS, jelikož ten je vhodný především k tvorbě microservice aplikací. BaaS by delegoval funkce BE na poskytovatele, ale pro lepší správu práce s daty jsem se rozhodl pro monolitickou BE architekturu. Ta také není závislá na poskytovateli serverless cloud služeb.

V rámci architektury FE části jsem se rozhodl pro architekturu SPA, přičemž aplikaci bude v budoucnosti možné rozšířit o dodatečné funkce pro splnění všech požadavků na architekturu PWA. Pro funkcionalitu vyvíjené aplikace není implementace všech požadavků na PWA kritická, avšak ty nejdůležitější by splňovat měla, jde především o zabezpečení veškeré komunikace a responsivní design.

Kapitola 6
Návrh řešení

V rámci této kapitoly se zabývám výběrem vhodných technologií pro implementaci aplikace na základě zadání, analýzy požadavků a vybrané architektury. Dále zde také rozebírám návrh datového modelu a komunikace mezi FE a BE částí.

6.1 Databáze

6.1.1 Typ

Nejprve je třeba rozhodnout, jestli by použitý databázový systém měl být relační, nebo NoSQL. Oba typy databází mají své výhody i nevýhody.

Relační

Relační databáze jsou mnohem starší, dobře odzkoušené a široce podporované. Data jsou v nich ukládána do relačně propojených tabulek. Mají fixní schéma, jsou vertikálně škálovatelné (zvýšení rychlosti zvýšením výkonu serveru) a pro práci s daty je potřeba mapování na datové objekty.

Mezi relační databázové systémy patří například Oracle, MySQL, Microsoft SQL Server, nebo PostgreSQL [14].

NoSQL

NoSQL databáze jsou novější a rychlejší, nejsou založeny na relacích, tedy místo propojení fixně navržených tabulek jsou většinou tvořeny JSON dokumenty, daty propojenými do grafu, nebo key-value páry.

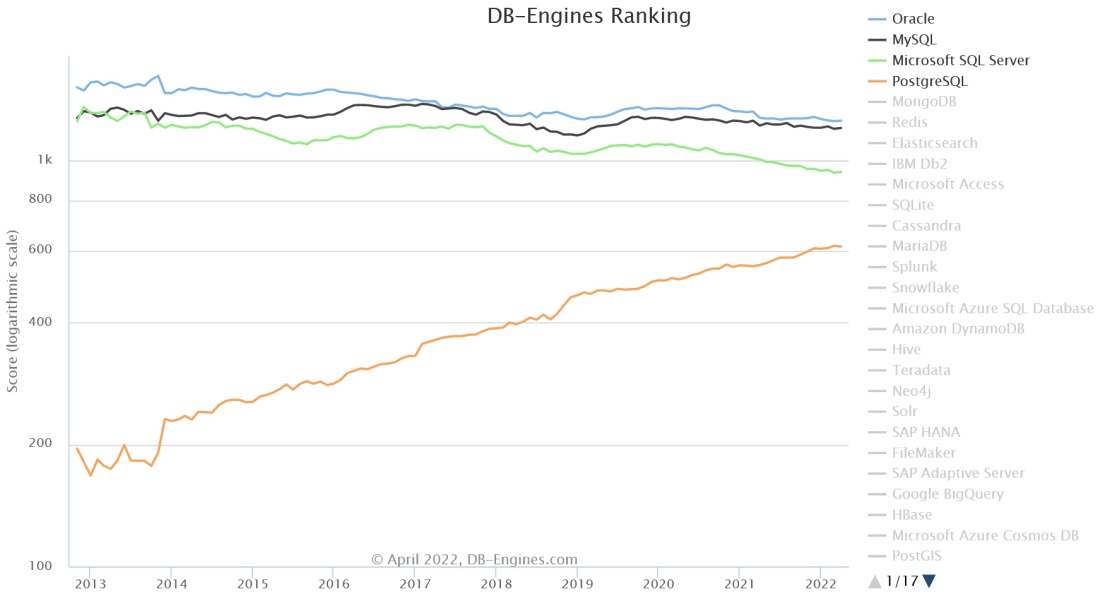
Vynikají především při velkém množství dat, nutnosti častých změn schématu a vynikající horizontální škálovatelností (zvýšení rychlosti zvýšením počtu serverů). Oproti relačním databázím se avšak v NoSQL databázích velmi špatně řeší atomické změny na více záznamech naráz.

Mezi NoSQL databázové systémy patří například MongoDB, CouchDB, Redis, DynamoDB, Cassandra,HBase, Neo4j, nebo Amazon Neptune [14].

Shrnutí

Rozhodl jsem se pro vývoj použít relační databázi, jelikož relační databáze jsou velmi dobře podporované, plně dostačují vyvíjené aplikaci a mám s nimi více zkušeností než s NoSQL databázemi.

6.1.2 Technologie



Obrázek 6.1. Srovnání popularity databázových systémů, převzato z [15]

V rámci relačních databázových systémů se nabízí mnoho různých řešení. Pro vývoj této aplikace jsem se rozhodoval mezi MySQL, Oracle, PostgreSQL a Microsoft SQL Server. Jde o 4 nejpopulárnější DB systémy dle dat DB-Engines [15], jak ukazuje obrázek 6.1.

MySQL

MySQL je open-source relační databázový systém spravovaný firmou Oracle Corporation.

Je široce používaný, škálovatelný, jednoduchý pro použití, a je využíván firmami jako Google, Amazon nebo Twitter [16]. Všechny základní funkcionality jsou dostupné zdarma, ale Oracle Corporation nabízí placené doplňkové služby [17].

Oracle

Oracle je komerční relační databázový systém spravovaný firmou Oracle Corporation.

Je určený pro velké korporáty a jeho hlavní výhodou je velmi dobré zabezpeční proti chybám způsobených správou systému [16]. Pro studijní nekomerční účely nabízí omezenou verzi zdarma.

PostgreSQL

PostgreSQL je open-source relační databázový systém.

Jde o velmi univerzální řešení s širokou podporou datových typů, velkým množstvím funkcionalit a možností ukládat data bez schématu [17].

Microsoft SQL Server

Microsoft SQL Server je komeční relační databázový systém spravovaný firmou Microsoft.

Zahrnuje velké množství dodatečných nástrojů použitelných pro snadný reporting, analytiku, nebo integraci s jinými technologiemi od Microsoftu [17]. Je velmi vhodný pro použití v prostředí používajícím velké množství Microsoft technologií. Jde o komerční

produkt, verze zdarma je silně omezena výkonem a je určena pouze pro malé firmy nebo vývojáře.

Shrnutí

Ačkoliv mám velké množství zkušeností s Microsoft SQL Server, tak vzhledem k jeho limitacím při použití neplacené verze není tento databázový systém vhodný pro vývoj této aplikace. Oracle také není vhodným kandidátem, jelikož jde o komerční systém zaměřený na korporátní využití a jeho neplacená verze je pouze pro studijní účely.

S PostgreSQL mám více zkušeností než s MySQL, zároveň je PostgreSQL široce využíván na ČVUT a byl mi doporučen v zadání. Pro lepší integraci vyvíjené aplikace mezi ostatní systémy používané na ČVUT jsem tedy zvolil PostgreSQL.

6.2 Backend

BE část aplikace bude zajišťovat komunikaci s FE a DB, práci s daty, autentizaci a notifikace uživatelů. Nabízí se vícero řešení, podrobněji rozebírám Spring, ASP .NET Core a Node.js.

Spring

Spring je multiplatformní open-source Java framework pro vývoj enterprise aplikací [18]. Vývojářům poskytuje infrastrukturu, na které lze jednoduše implementovat potřebné funkce, jako jsou dotazy do DB, tvorba HTTP (Hypertext Transfer Protocol) endpointů pro vývoj API, message handlerů apod. Propracovaný systém pro dependency injection také velmi usnadňuje vývoj.

ASP .NET Core

ASP .NET Core je multiplatformní open-source .NET Core framework pro vývoj moderních internetových aplikací [19]. Poskytuje jednoduchou možnost vyvíjet jak API, tak UI (User Interface) webových aplikací, umožňuje jednoduché testování a s použitím Blazer frameworku pro FE dovoluje psát klientskou logiku nejen v JavaScriptu, ale i v jazyce C#.

Node.js & Express

Node.js je asynchronní event-driven JS runtime. Nevyužívá pro každé spojení jedno vlákno, nýbrž při každém dotazu volá callback, čímž umožňuje paralelní provoz [20]. Tím předchází jakýmkoliv problémům s deadlockem, jelikož není potřeba nic zamykat.

Pro Node.js existuje velké množství knihoven, jednou z nejpoužívanějších pro tvorbu BE API je Express. Jde o velmi rychlou, jednoduchou a flexibilní knihovnu, díky které lze implementovat BE API extrémně rychle [21].

6.2.1 Shrnutí

Ačkoliv mám se Spring a ASP .NET Core více zkušeností než s Node.js, tak jsem pro vývoj aplikace zvolil Node.js, a to pro jeho velmi dobře zdokumentovanou kompatibilitu s ostatními použitými technologiemi.

6.3 ORM systém

Pro práci s DB daty v rámci BE je třeba mapovat data na objekty. Je možné implementovat vlastní systém pro dotazy do DB a následné mapování na objekty, ale použití již existujícího systému má nesporné výhody. Existující ORM systémy řeší většinu možných problémů, jako je optimalizace dotazů, migrace nebo sanitizace dat [22].

Existuje mnoho ORM systémů pro Node.js, dále rozebírám 4 z nich - Knex, Sequelize, TypeORM a Prisma.

Knex.js

Knex.js je ORM systém pro Node.js, sloužící především jako builder SQL dotazů. Je jednoduše přenosný a použitelný, podporuje asynchronní dotazy pomocí callback i promises, vytváření schémat, transakce a connection pooling [23].

Sequelize

Sequelize je ORM systém pro Node.js a Typescript. Podporuje např. transakce, relace, replikaci, eager i lazy loading, nebo migrace [24]. Asynchronní dotazy v rámci JS fungují na principu promises.

TypeORM

TypeORM je ORM podporující mnoho JS frameworků i vanilla JS od verze ES5 (ECMAScript 5). Poskytuje přístup k DB 2 různými patterny (Active Record nebo Data Mapper), čímž umožňuje implementaci volně provázaných, dobře škálovatelných a udržitelných aplikací [25]. Má velké množství funkcionalit a podporuje značné množství DB systémů včetně NoSQL MongoDB.

Prisma

Prisma je nový typ ORM systému, od ostatních ORM se velmi odlišuje. Tradiční ORM systémy pracují na systému mapování tabulek na třídy definované v BE, což může vést k problémům v důsledku nesouladu tříd a schématu. Prisma oproti tomu definuje datový model, který poté slouží jako jediný zdroj jak pro DB schéma, tak pro objekty v rámci BE [26].

Prisma se skládá ze 3 komponent:

- Prisma Client: Automaticky generovaný builder SQL dotazů pro Node.js a TypeScript
- Prisma Migrate: Pokročilý migrační systém
- Prisma Studio: GUI (Graphical User Interface) pro zobrazení a editaci dat v DB

Na základě schématu je vygenerován Prisma Client a tabulky v databázi. Takto vygenerovaný Client umožňuje velmi jednoduše přistupovat k záznamům v DB jako k objektům.

Ukázka update dotazu použitím Prisma Client:

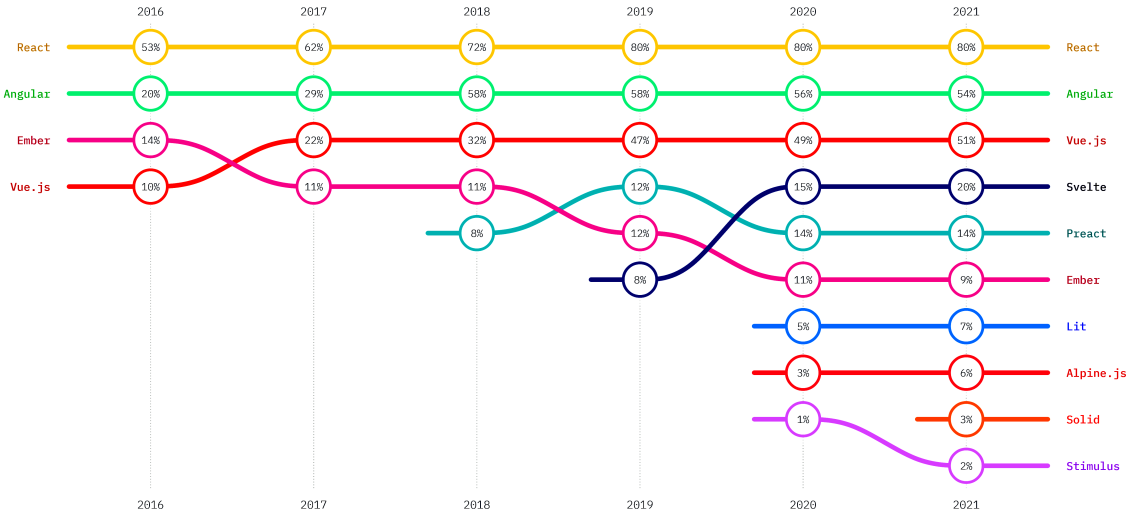
```
var result = await prisma.round.updateMany({
  where: {
    form: {
      creatorId: 6
    }
  },
  data: {
```

```
      terminated: true,
      endTime: new Date()
    },
  });
```

6.3.1 Shrnutí

Z probraných ORM systémů jsem zvolil systém Prisma, a to pro jeho moderní přístup, velmi snadné migrace DB a jednoduché použití.

6.4 Frontend



Obrázek 6.2. Srovnání četnosti použití FE frameworků v čase, hodnocení na základě dotazníku, hodnoty vypočítány jako $\frac{chtějí\ použít\ znovu\ +\ nechtějí\ použít\ znovu}{celkový\ počet\ hlasujících}$, převzato z [27]

Pro vývoj FE existuje velké množství možností. První otázkou je, zda programovat všechny funkcionality v čistém (vanilla) JS, nebo zda použít některý z mnoha frameworků. Dle dat ze stateofjs.com [27] jsou tři nejvíce používané frameworky React, Angular a Vue.js, jak ukazuje obrázek 6.2.

Vanilla JS

Před vydáním verze ES6 (ECMAScript 6) v roce 2015 byly oproti vanilla JavaScriptu často preferovány frameworky. Po 6 letech od vydání předchozí verze přinesl ES6 řadu funkcionalit, díky kterým lze pohodlně psát SPA i bez použití frameworků, např. arrow funkce, třídy, iterátory nebo asynchronní systém Promises [28].

Vývoj ve vanilla JS je výhodný především v rychlosti aplikací, vanilla JS je o hodně rychlejší než jakýkoliv FE framework [29]. Nevýhodou je nutnost implementovat celý zobrazovací, datový i komunikační systém od nuly.

React

React je open-source FE framework spravovaný společností Meta a je světově nejpoužívanějším FE frameworkem [27]. React se specializuje na skládání UI ze znovupoužitelných komponent, na jejichž prvky jednostranně mapuje data [30].

Velkou výhodou je také existence mnoha rozšiřujích balíčků třetích stran, což umožňuje snadné přizpůsobení různým použitím. Nevýhodou je zaměření pouze na zobrazení dat, oboustranné mapování dat a naslouchání eventům je nutno implementovat samostatně. Kvůli tomu má React také střední křivku učení.

Angular

Angular je open-source FE framework spravovaný společností Google, byl vydán v roce 2010 jako vůbec první framework pro vývoj SPA [30]. Angular je komplexnější než React a je uzpůsobený k vývoji kompletní SPA včetně správy dat apod., umožňuje např. oboustranné mapování dat, nebo propisování dat mezi komponentami rodičů a jejich potomků.

Stejně jako React existuje i pro Angular mnoho balíčků třetích stran. Hlavní výhodou je možnost použití Angularu pro celý FE, není třeba samostatně vyvíjet mapování dat nebo naslouchání eventům. Nevýhodou je velmi strmá křivka učení, způsobená komplexností frameworku.

Vue.js

Vue.js je open-source MVVM (používající architekturu model–view–viewmodel) FE framework pro vývoj UI a SPA, který vydal Evan You a je spravován komunitou [30]. Vue.js lze použít pro kompletní vývoj SPA čistě použitím Vue komponent v rámci Vue projektu, ale jde také integrovat do vanilla JS aplikace pouze pro určité části UI. Umožňuje oboustranné mapování dat, UI skládá ze samostatných propojitelných komponent, kde každá komponenta má vlastní HTML (HyperText Markup Language) template, JS logiku a zapouzdřený CSS (Cascading Style Sheets) styl.

Mezi hlavní výhody Vue.js patří jeho modularita, velmi propracovaná dokumentace, dobrá nabídka balíčků (oficiálních i od třetích stran) a nízká křivka učení. Jde stejně dobře využít jak pro malé a jednoduché aplikace, tak pro velmi komplexní systémy.

6.4.1 Shrnutí

Pro vývoj aplikace jsem zvolil framework Vue.js. Byl mi doporučen v zadání práce a jeho nízká křivka učení je ideální, protože ani s jedním z probíraných frameworkům jsem neměl žádné zkušenosti. Vanilla JS by sice poskytoval možnost vyvinout mnohem rychlejší aplikaci, ale vývoj některých funkcionalit by byl mnohem náročnější.

6.5 Komunikace FE a BE

Komunikace FE klienta a BE serveru lze realizovat různými způsoby. Nejtypičtějším řešením pro SPA aplikace je jednosměrná komunikace skrze REST API (Representational State Transfer API) vystavené na BE serveru, na které se poté FE dotazuje.

Pro dosažení oboustranné komunikace lze použít protokol WebSocket, nebo na něm založené technologie. V rámci zadání mi byla doporučena knihovna Socket.IO, která je nadstavbou nad WebSocket protokolem.

REST API

REST není komunikační protokol, nýbrž sbírka architektonických požadavků na komunikaci, implementace se tedy mohou lišit.

Při komunikaci skrz REST API jsou používány standardní HTTP metody GET, POST, UPDATE a DELETE, pro každý zdroj dat existuje samostatný endpoint, komunikace je bezstavová a pro přenos dat se používají většinou formáty JSON nebo XML (Extensible Markup Language) [31].

WebSocket

WebSocket je komunikační protokol poskytující oboustranný komunikační kanál skrze jedno TCP připojení. Pro kompatibilitu s HTTP připojením se otevírá použitím HTTP Upgrade hlavičky při navázání TCP spojení skrz normální HTTP porty 80 nebo 443 [32].

Je podporován prakticky ve všech moderních prohlížečích, jako Google Chrome, Firefox, Microsoft Edge a Safari [33].

Socket.IO

Socket.IO je nadstavbou WebSocketu umožňující real-time obousměrnou event-based komunikaci mezi klientem a serverem [34]. Primárně existuje jako serverová knihovna pro Node.js a klientská knihovna pro JS, ale existují implementace i pro jiné jazyky.

Výhodou Socket.IO je jednodušší použití než implementace celé event-based komunikace jen na základě WebSocket protokolu. Nevýhodou je, že ačkoliv Socket.IO komunikuje použitím WebSocket protokolu, tak s ním není zpětně kompatibilní, WebSocket klient se tedy nemůže připojit přímo k Socket.IO serveru a naopak.

Ukázka použití implementace Socket.IO komunikace, převzato z [34]:

```
Server:
import { Server } from "socket.io";
const io = new Server(3000);
io.on("connection", (socket) => {
  socket.emit("hello from server", 1, "2", { 3: Buffer.from([4]) });
});

Klient:
import { io } from "socket.io-client";
const socket = io("ws://localhost:3000");
socket.on("hello from server", (...args) => {
  var a = args;
});
```

6.5.1 Shrnutí

Pro dosažení oboustranné komunikace mezi serverem a klientem jsem se rozhodl použít knihovnu Socket.IO jako hlavní způsob komunikace.

Jak uvádím dále v sekci 6.6, před spojením prostřednictvím Socket.IO je nutné provést autentizaci uživatele. Proto jsem se rozhodl implementovat taktéž API určené výhradně pro účely autentizace.

6.6 Autentizace

Systém autentizace je nutné navrhnout dle analýzy požadavků a návrhu ostatních částí aplikace. Pro komunikaci mezi FE a BE je používán Socket.IO, autentizace tedy musí fungovat pro Socket.IO spojení. Dále je nutné poskytnout možnost přihlášení jak interním uživatelům ČVUT, tak guest uživatelům mimo ČVUT.

6.6.1 Autentizační metody

Pro autentizaci uživatelů se používají primárně 2 autentizační metody, a to buď ověření pomocí cookie, a nebo tokenu.

Cookie

Při autentizaci pomocí cookie server po úspěšném přihlášení vytváří session, generuje session ID, ukládá jej do DB a posílá jej zpět uživateli jako cookie. Mimo session ID může obsahovat také expiraci, doménu a další parametry. Uživatelský prohlížeč cookie se session ID uloží a přikládá jej ke všem dalším dotazům v hlavičce dotazu. Server ověřuje přiložené session ID proti DB, pokud DB obsahuje dané platné session ID, uživatel je ověřen [35].

Cookie autentizace je výhodná v případě, že je třeba rozlišovat každou session na serveru a ukládat k ní dodatečné informace. Nevýhody spočívají ve špatné škálovatelnosti (server musí držet v DB údaje o každé session) a problematické pooužití pro autentizaci vůči API [35].

Token

Při autentizaci pomocí tokenu server po úspěšném přihlášení vytváří podepsaný token, který posílá zpět uživateli. Token je uložen prohlížečem a je přiložen při všech dalších dotazech v hlavičce dotazu, server poté ověřuje validitu kryptografického podpisu, datum expirace a případné další parametry [35].

Existuje více standardů a protokolů pro autentizaci pomocí tokenů, jde například o OAuth (Open Authorization), SAML (Security Assertion Markup Language), OIDC (OpenID Connect), CIBA (Client Initiated Backchannel Authentication), nebo JWT (JSON Web Token) [36].

6.6.2 Autentizace Socket.IO spojení

Socket.IO podporuje registraci middleware pro autentizaci, výběr existujících modulů ovšem není velký. Alternativně lze middleware moduly pro Express použít i v rámci Socket.IO pomocí wrapper funkce, ale pouze s omezenou funkcionalitou [37].

Knihovna socketio-jwt je middleware modul pro Socket.IO a umožňuje jednoduchou autentizaci pomocí JWT, podporuje nejnovější verzi Socket.IO a stále dostává nové aktualizace [38]. Vychází ze staršího a již neudržovaného modulu socketio-jwt-auth [39] a je prakticky jediným aktuálním modulem pro autentizaci Socket.IO spojení.

6.6.3 Autentizace interního ČVUT uživatele

Ze zadání a analýzy požadavků vyplývá, že by aplikace měla podporovat přihlášení interním zaměstnancům ČVUT pomocí SSO brány zprostředkované ČVUT Keycloak serverem.

Keycloak je open-source správce identit a přístupů používající SSO autentizační schéma a podporuje protokoly OpenID Connect, OAuth 2.0 a SAML [40].

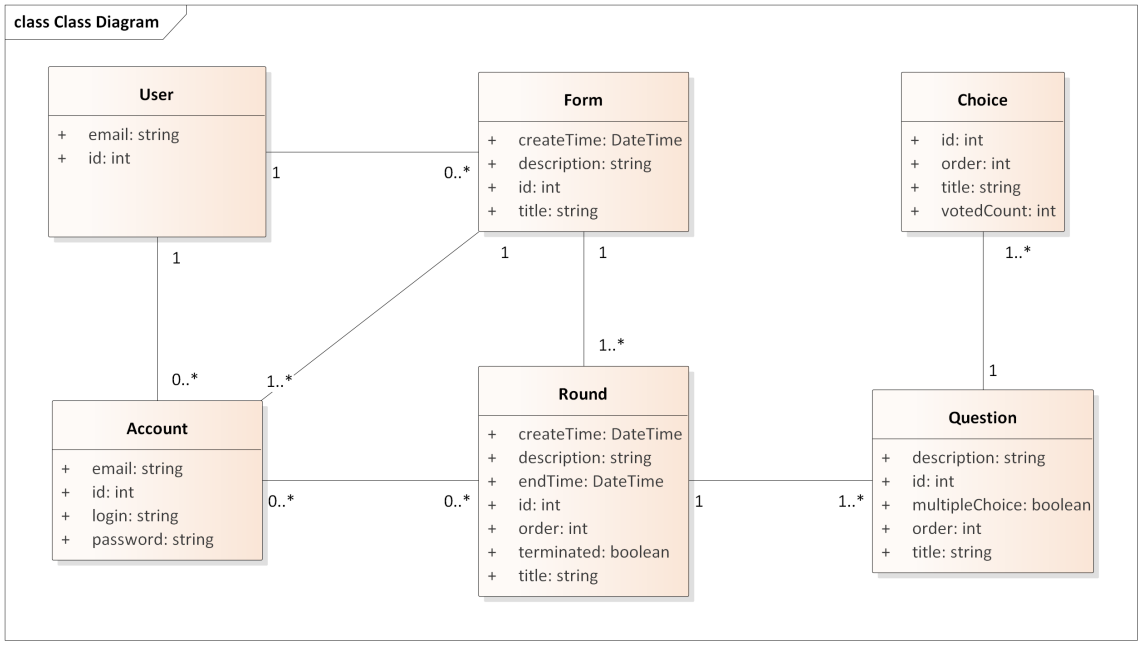
6.6.4 Shrnutí a výběr metody autentizace

Jelikož implementace vlastního autentizačního middleware pro Socket.IO by byla náročná na implementaci, zvolil jsem jako princip autentizace použití JWT s pomocí knihovny socketio-jwt.

Přihlášení guest uživatelů proběhne skrze první endpoint REST API, pokud přihlášení proběhne v pořádku, BE vydá uživateli podepsaný JWT token pro autentizaci Socket.IO spojení knihovnou socketio-jwt.

Přihlášení interních uživatelů ČVUT proběhne skrze ČVUT SSO Keycloak server, který uživateli vydá OIDC token. Ten nelze jednoduše ověřit pomocí knihovny socketio-jwt, jelikož ta nepodporuje ověřování více typů tokenů od různých vydavatelů. Pro autentizaci Socket.IO spojení tedy BE vydá JWT token na základě OIDC tokenu vydaného ČVUT serverem, k čemuž bude sloužit druhý z endpointů REST API.

6.7 Diagram tříd



Obrázek 6.3. Diagram tříd

Diagram tříd definuje strukturu systému z hlediska tříd a jejich atributů [41]. Obrázek 6.3 ukazuje diagram znázorňující návrh datových tříd, vytvořený na základě analýzy požadavků.

Níže rozebírám jednotlivé datové třídy, jejich funkce, atributy a relace. Atributy a typy vychází z vytvořeného schématu pro Prisma ORM, dle něhož jsou posléze generovány datové třídy a DB schéma.

6.7.1 User

Třída User představuje uživatele, definovaného jeho emailem.

- id (Int) - unikátní identifikátor generovaný databází
- email (String) - email uživatele
- forms (Form[]) - kolekce hlasovacích formulářů, které uživatel spravuje
- accounts (Account[]) - kolekce účtů, které patří uživateli

6.7.2 Account

Třída Account představuje hlasovací účet uživatele. Každý účet patří určitému uživateli a je napojen na hlasovací formulář, ve kterém je oprávněn hlasovat.

- id (Int) - unikátní identifikátor generovaný databází
- email (String) - email účtu
- login (String) - generované přístupové uživatelské jméno pro jednorázový přístup
- password (String) - hash generovaného přístupového hesla pro jednorázový přístup
- userId (Int) - identifikátor uživatele, pod který účet spadá
- user (User) - uživatel, pod který účet spadá
- formId (Int) - identifikátor hlasovacího formuláře, ke kterému účet náleží
- form (Form) - hlasovací formulář, ke kterému účet náleží
- voted (Round[]) - kolekce kol hlasovacích formulářů, ve kterých již účet hlasoval

6.7.3 Form

Třída Form představuje hlasovací formulář. Formulář je vždy spravován jedním uživatelem a je tvořen 1 až N hlasovacími koly, ve kterých mohou oprávnění uživatelé hlasovat.

- id (Int) - unikátní identifikátor generovaný databází
- title (String) - název hlasovacího formuláře
- description (String?) - nepovinný popis hlasovacího formuláře
- createTime (DateTime) - čas vytvoření hlasovacího formuláře
- creatorId (Int) - identifikátor uživatele, který spravuje hlasovací formulář
- creator (User) - uživatel, který spravuje hlasovací formulář
- accounts (Account[]) - kolekce účtů, které mají oprávnění hlasovat v kolech hlasovacího formuláře
- rounds (Round[]) - kolekce hlasovacích kol, které spadají pod hlasovací formulář

6.7.4 Round

Třída Round představuje hlasovací kolo hlasovacího formuláře. Každé kolo obsahuje kolekci účtů, které již hlasovaly, a je tvořeno 1 až N otázkami.

- id (Int) - unikátní identifikátor generovaný databází
- order (Int) - pořadí kola v rámci hlasovacího formuláře
- title (String) - název hlasovacího kola
- description (String?) - nepovinný popis hlasovacího kola
- createTime (DateTime) - čas vytvoření hlasovacího kola
- endTime (DateTime) - čas ukončení hlasovacího kola
- terminated (Boolean) - indikátor předčasného ukončení hlasovacího kola
- formId (Int) - identifikátor hlasovacího formuláře, pod který hlasovací kolo spadá
- form (Form) - hlasovací formulář, pod který hlasovací kolo spadá
- questions (Question[]) - kolekce otázek, které spadají pod hlasovací kolo
- voted (Account[]) - kolekce účtů, které již v hlasovacím kole hlasovaly

6.7.5 Question

Třída Question představuje otázku hlasovacího kola. Každá otázka obsahuje kolekci možných odpovědí a indikátor, zda je možné hlasovat pro více odpovědí, nebo jen pro jednu.

- id (Int) - unikátní identifikátor generovaný databází
- order (Int) - pořadí otázky v rámci hlasovacího kola
- title (String) - název otázky
- description (String?) - nepovinný popis otázky
- multipleChoice (Boolean) - indikátor možnosti hlasovat v rámci této otázky pro více než 1 odpověď
- roundId (Int) - identifikátor hlasovacího kola, pod které otázka spadá
- round (Round) - hlasovací kolo, pod které otázka spadá
- choices (Choice[]) - kolekce odpovědí, které spadají pod otázku

6.7.6 Choice

Třída Choice představuje možnou odpověď na otázku. V rámci odpovědi je také uložena informace o počtu hlasujících pro tuto odpověď.

- id (Int) - unikátní identifikátor generovaný databází
- order (Int) - pořadí odpovědi v rámci otázky
- title (String) - název odpovědi
- votedCount (Int) - počet účtů hlasujích ve prospěch této odpovědi
- questionId (Int) - identifikátor otázky, pod kterou odpověď spadá
- question (Question) - otázka, pod kterou odpověď spadá

6.8 Návrhy obrazovek

Pro nastínění vizuální podoby aplikace jsem vytvořil návrhy obrazovek aplikace. Zaměřil jsem se na low-fidelity prototyp pro mobilní zařízení, zabývám se rozložením komponent, nikoliv přesným designem. Návrh pro desktopové zobrazení by byl odlišný pouze rozložením hlavního menu a nabídky dostupných hlasovacích formulářů.

Pro tvorbu návrhů jsem použil webovou aplikaci Moqups [42], která umožňuje velmi komplexní tvorbu návrhů i ve verzi zdarma. Ukázka vzniklých návrhů se nachází v příloze B.

Kapitola 7
Implementace

V této kapitole se zabývám popisem implementace částí webové aplikace na základě provedené analýzy požadavků a návrhu aplikace. Zaměřuji se na hlavní kroky v implementačním procesu a snažím se nezacházet do zbytečných detailů.

Implementaci jsem začal založením FE a BE projektů a instalací potřebných technologí. Poté jsem vytvořil a propojil základní DB, BE a FE, čímž vzniklo jádro SPA. Dále jsem se zaměřil na autentizaci a rozvoj funkční FE a BE. Pro odhalení chyb a zlepšení uživatelského komfortu jsem v průběhu implementace aplikaci manuálně testoval, jak zmiňuji dále v kapitole 8.

7.1 Vývojové prostředí

Pro efektivní vývoj aplikace jsem využil IDE (Integrated development environment) Visual Studio Code [43]. Jde o multiplatformní open-source IDE vyvíjené firmou Microsoft, v základu podporující JavaScript, TypeScript a Node.js, přičemž podporu mnoha dalších jazyků a runtime prostředí lze přidat pomocí pluginů [44].

Veškeré textové příkazy použité v průběhu implementace jsem spouštěl skrz Windows Terminal, což je multiplatformní open-source alternativa k základní příkazové řádce systému Windows s velkými možnostmi přizpůsobení, podporou záložek a unicode textu [45]. Příkazy spouštěné na serveru jsem spouštěl skrz SSH (Secure Shell Protocol) připojení.

7.2 Založení projektu

Soubory projektu jsem rozdělil na FE a BE části do dvou adresářů. Vzhledem k tomu, že obě části aplikace jsou založeny na JS technologiích, byla inicializace obou částí podobná.

Pro instalaci potřebných balíčků jsem použil NPM, což je registr balíčků, které lze jednoduchými příkazy nainstalovat do adresáře (nebo globálně) a vytvořit záznam o závislosti aplikace na daných balíčcích [46].

7.3 Backend

Pro implementaci BE jsem použil Node.js společně s dalšími knihovnami, které mi velmi usnadnily vývoj. BE část jsem rozdělil do několika modulů - hlavní modul zajišťující aplikační logiku a komunikaci mezi BE a FE, modul pro přístup k databázi a modul pro odesílání emailů uživatelům.

Začal jsem vytvořením DB schématu pro Prisma ORM, poté jsem příkazem `npx prisma generate` vytvořil Prisma Client s definicí databázových objektů, kterými lze přistupovat k databázi. V rámci modulu pro přístup k databázi jsem skrze Prisma Client implementoval DB dotazy potřebné pro všechny funkce aplikace.

Na to jsem navázal implementací samotné logiky aplikace v hlavním modulu. Komunikaci mezi FE a BE pomocí API endpointů jsem implementoval použitím knihovny Express, na implementaci Socket.IO spojení jsem použil oficiální knihovnu pro Node.js.

Komunikace mezi FE a BE částí začíná procesem autentizace skrze API. Pro ověření OIDC tokenu při přihlášení pomocí ČVUT SSO jsem použil knihovnu aws-jwt-verify [47], pro generaci JWT knihovnu node-jsonwebtoken [48]. Po zahájení Socket.IO spojení je JWT ověřen middleware knihovnou socketio-jwt [38], skrze ověřené spojení posléze probíhá obousměrná komunikace. Pro každý typ dotazu jsem vytvořil metodu, ve které probíhá vyhodnocení dotazu a poté odeslání odpovědi. Pro lepší správu odpovědí posílám v každé z nich pole s číselným HTTP status kódem, na jehož základě posléze určuji, zda-li byl dotaz úspěšný.

Jako poslední jsem implementoval modul pro odesílání emailů uživatelům. Posílání probíhá skrze ČVUT SMTP (Simple Mail Transfer Protocol) server, pro implementaci jsem použil knihovnu Nodemailer [49].

7.4 Frontend

Pro implementaci FE jsem použil JS framework Vue.js. Začal jsem vytvořením Vue projektu pomocí Vue CLI (command-line interface) příkazem `npm vue create ctu_balloting_webapp`. Pro každou sekci aplikace jsem implementoval samostatnou komponentu, pro routing mezi komponentami jsem použil Vue Router [50], který zajistil i správnou historii procházení aplikací v rámci prohlížeče.

Implementaci Socket.IO spojení s BE mi velmi ulehčila knihovna Vue-Socket.IO [51], která automaticky globálně registruje Socket.IO instanci přístupnou ze všech komponent, v každé komponentě je pak možné navěsit potřebné listenery.

Pro autentizaci interních uživatelů ČVUT jsem použil knihovnu vue-oidc-client [52], kterou jsem nastavil na ověření vůči ČVUT Keycloak serveru. Po přihlášení skrze ČVUT SSO a získání OIDC tokenu provádím dotaz s tokenem na BE API, které vrátí JWT token pro navázání Socket.IO spojení.

Po vizuální stránce jsem se řídil připravenými návrhy obrazovek (viz sekce 6.8) a zvolil jsem jednoduchý vzhled, který lze případně v budoucnu vylepšit.

Pro urychlení vývoje jsem pro účely průběžného manuálního testování měl FE projekt spuštěný ve Vue CLI příkazem `npm run serve -- --port 80`, funkce Hot Reload [53] v takovém případě aktualizuje právě upravené komponenty za běhu aplikace a bez přenačtení stránky.

7.5 Nasazení

Pro nasazení systému jsem použil virtuální server se systémem Ubuntu 20.04 běžící na platformě Google Cloud Platform (GPC). GPC nabízí typ „e2-micro“ své instance zdarma včetně veřejné IP adresy [54]. Instance má dedikovanou 1/4 jádra procesoru, 1GB (Gigabyte) RAM (Random Access Memory) a HDD (Hard Disk Drive) úložiště o kapacitě 30GB. Tyto parametry nejsou vysoké, ale pro běh jednoduchých webových aplikací plně dostačují.

7.5.1 Databáze

DB systém PostgreSQL jsem nainstaloval dle tutoriálu na DigitalOcean [55]. Poté jsem vytvořil DB pro data aplikace. Tabulky jsem vygeneroval skrz ORM systém Prisma použitím příkazu `npx prisma migrate dev --name MigrationName` v rámci vývoje BE části jak uvádím v sekci 7.3.

7.5.2 Backend

Zdrojové soubory BE jsem umístil do samostatného adresáře a nainstaloval závislosti pomocí příkazu `npm install`. Pro automatický běh Node.js serveru jsem použil systémového správce systemd. Vytvořil jsem konfigurační soubor s nastavením BE služby, kterou jsem posléze aktivoval příkazem `systemctl enable ctu_balloting_webapp_node` a spustil příkazem `systemctl start ctu_balloting_webapp_node`.

Ukázka konfigurace BE služby:

```
[Unit]
Description="Node.js backend for CTU Balloting webapplication"
After=network.target

[Service]
Environment=NODE_PORT=15100
Typ=simple
User=root
ExecStart=/usr/bin/node /var/www/balloting.borek.cc/backend/server.js
WorkingDirectory=/var/www/balloting.borek.cc/backend/
Restart=always
RestartSec=10
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=CtuBallotingBackend

[Install]
WantedBy=multi-user.target
```

BE server používá pro zabezpečenou komunikaci s FE klientem SSL certifikát od Let's Encrypt, což je nezisková organizace vydávající X.509 certifikáty zdarma [56]. Instalaci certifikátu popisuji níže v sekci 7.5.3. Pro přístup BE serveru k souborům certifikátu spouštím službu jako root uživatel.

7.5.3 Frontend

Pro nasazení FE jsem použil Apache, což je open-source HTTP server pro UNIX and Windows systémy [57]. Instalaci a nastavení Apache serveru jsem provedl dle návodu na webu DigitalOcean [58].

Pro nasazení Vue projektu je nejdříve nutné provést build zdrojových souborů příkazem `npm run build`, výsledné soubory jsou umístěny ve složce `dist`. Vzniklé soubory jsem umístil na server do samostatné složky vedle BE části. Pro tuto složku jsem poté nastavil virtuální instanci Apache serveru dle návodu od DigitalOcean [59].

Vzhledem k tomu, že jde o SPA, musel jsem také nastavit přesměrování všech cest webové aplikace na `index.html`, aby správně fungoval Vue Router, který se stará o přechody mezi jednotlivými sekcemi aplikace. Pro přesměrování jsem použil konfigurační soubor `.htaccess`, kterým lze specifikovat přístup k souborům vystaveným Apache serverem a to globálně i pro jednotlivé adresáře.

Ukázka použitého .htaccess souboru:

```
<IfModule mod_rewrite.c>
  RewriteEngine On
  RewriteBase /
  RewriteRule ^index\.html$ - [L]
  RewriteCond %{REQUEST_FILENAME} !-f
```

```
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule . /index.html [L]
  </IfModule>
```

Komunikaci s aplikací jsem zabezpečil pomocí SSL certifikátu od Let's Encrypt [56]. Pro automatickou instalaci a správu certifikátů jsem použil nástroj Certbot [60]. Po nastavení virtuální Apache instance aplikace stačí Certbot spustit pomocí příkazu `certbot --apache` a projít dialogem pro nastavení certifikátu, Certbot poté požádá Let's Encrypt o vystavení certifikátu a nastaví virtuální instanci aplikace tak, aby certifikát používala.

7.6 Závěr

V této kapitole jsem přiblížil postup implementace single-page aplikace a její nasazení na virtuální server se systémem Ubuntu 20.04. Screenshoty vzniklé aplikace lze najít v příloze C.

Pro zvěřejnění aplikace jsem vytvořil subdoménu v rámci mé domény vedené u registrátora Porkbun [61]. Aplikace je dostupná na adrese `https://balloting.borek.cc`.

Kapitola 8
Testování

Pro zajištění správné funkčnosti aplikace je nutné otestovat její funkce. Existují různé způsoby testování, zaměřil jsem se na manuální a uživatelské testování.

8.1 Manuální

V průběhu vývoje jsem průběžně manuálně testoval aplikaci pro ověření správné funkce všech komponent a odhalení případných problémů. Tímto způsobem jsem odhalil a opravil kritické chyby.

Před začátkem uživatelského testování jsem provedl podrobný manuální test aplikace, zaměřený nejen na funkčnost všech komponent, ale také na krajní případy, které mohou ojediněle nastat při používání aplikace. Cílem bylo odhalit co nejvíce možných chyb, abych se poté mohl zaměřit jen na menší problémy odhalené testery v rámci uživatelského testování.

8.2 Uživatelské

Uživatelské testování je kvalitativní forma manuálního testování, při které zkušební uživatelé (testeři) prochází připravené průchody aplikací v reálných podmínkách [62]. Tímto způsobem lze otestovat funkčnost aplikace při normálním používání a nasbírat zpětnou vazbu na její použití.

Vzhledem k nutnosti ČVUT účtu pro přístup ke všem funkcím aplikace jsem rozdělil scénáře do 2 skupin - pro guest uživatele a pro interní uživatele ČVUT. Každý scénář obsahuje identifikátor, popis testovacích dat a kroků, a očekávaný výstup.

Pro účely testování aplikace jsem oslovil malou skupinu lidí, část z nich s funkčním ČVUT účtem, aby bylo možné otestovat všechny funkce aplikace.

Scénáře jsou navrženy tak, aby je po obdržení pozvánky k hlasování mohl každý tester projít samostatně bez nutnosti kooperace s ostatními. Toto řešení sice snižuje počet možných scénářů, ale velmi usnadňuje práci testerům. Každému testerovi guest funkcí jsem poskytl přihlašovací údaje ke guest účtu s jedním hlasováním, které obsahovalo jedno neukončené a jedno ukončené kolo. Testy pro interní ČVUT uživatele jsou navrženy tak, aby je bylo možné projít bez předchozí pozvánky k hlasování.

8.2.1 Scénáře pro guest uživatele

Scénář G01 - Přihlášení

Testovací data: Nepřihlášený uživatel, přihlašovací údaje
Testovací kroky:

1. Uživatel přijde na úvodní stranu aplikace s login formulářem.
2. Uživatel vyplní přihlašovací formulář přihlašovacími údaji a odešle jej tlačítkem „Login“.

Očekávaný výstup: Po úspěšném odeslání formuláře je uživatel přihlášen do aplikace, v sekci „Home“ se mu zobrazí úvodní text a navigační menu obsahuje položky „Home“, „Vote“, „Results“ a „Logout“.

Scénář G02 - Zobrazení dostupných hlasování

Testovací data: Přihlášený guest uživatel
Testovací kroky:

1. Uživatel přejde do sekce „Vote“ pomocí navigačního menu.
2. Pokud není automaticky vybrán formulář a kolo, které chce uživatel zobrazit, tak jej vybere pomocí nabídky hlasovacích formulářů a kol.

Očekávaný výstup: Uživatel vidí detaily vybraného hlasovacího formuláře a kola s hlasovacími možnostmi.

Scénář G03 - Provedení hlasování

Testovací data: Přihlášený guest uživatel
Testovací kroky:

1. Uživatel přejde do sekce „Vote“ pomocí navigačního menu.
2. Pokud není automaticky vybrán formulář a kolo, ve kterém chce uživatel hlasovat, tak jej vybere pomocí nabídky hlasovacích formulářů a kol.
3. V sekci se zvoleným kolem uživatel vybere možnosti, pro které chce hlasovat.
4. Uživatel odešle vybrané možnosti kliknutím na tlačítko „Vote“.

Očekávaný výstup: Pod detaily vybraného hlasovacího kola je zobrazena zpráva „Voting successful“ a možnosti nelze již vybírat, ani odesílat.

Scénář G04 - Zobrazení výsledků hlasování

Testovací data: Přihlášený guest uživatel
Testovací kroky:

1. Uživatel přejde do sekce „Results“ pomocí navigačního menu.
2. Pokud není automaticky vybrán formulář a kolo, jehož výsledky chce uživatel zobrazit, tak jej vybere pomocí nabídky hlasovacích formulářů a kol.

Očekávaný výstup: V detailech kola zobrazených pod nabídkou kol jsou zobrazeny výsledky hlasování.

Scénář G05 - Odhlášení

Testovací data: Přihlášený guest uživatel
Testovací kroky:

1. Uživatel klikne na tlačítko „Logout“ v navigačním menu.

Očekávaný výstup: Uživatel je přesměrován na úvodní stranu aplikace s login formulářem a na navigačním menu je pouze záložka „Home“.

8.2.2 Scénáře pro interní ČVUT uživatele

Scénář I01 - Přihlášení

Testovací data: Nepřihlášený uživatel, účet ČVUT
Testovací kroky:

1. Uživatel přijde na úvodní stranu aplikace a je mu zobrazen přihlašovací formulář.

2. Uživatel klikne na tlačítko „CTU SSO Login“ pro přesměrování na ČVUT SSO bránu.
3. Uživatel vyplní přihlašovací formulář na ČVUT SSO bráně a odešle jej tlačítkem „SSO Login“. Tento krok je přeskočen, pokud má uživatel v prohlížeči platnou ČVUT SSO session.

Očekávaný výstup: Po úspěšném ověření v rámci ČVUT SSO brány je uživatel přesměrován zpět do aplikace, v sekci „Home“ se mu zobrazí úvodní text a navigační menu obsahuje položky „Home“, „Vote“, „Results“, „Create“, „Manage“ a „Logout“.

Scénář I02 - Povolení notifikací

Testovací data: Přihlášený interní ČVUT uživatel
Testovací kroky:

1. Uživatel přejde do sekce „Home“ pomocí navigačního menu.
2. Uživatel zobrazí dialog prohlížeče pro povolení notifikací kliknutím na tlačítko „Allow notifications“.
3. Uživatel v dialogu prohlížeče povolí zobrazování notifikací.

Očekávaný výstup: Tlačítko „Allow notifications“ je neaktivní a nelze znovu použít.

Scénář I03 - Vytvoření nového hlasovacího formuláře

Testovací data: Přihlášený interní ČVUT uživatel
Testovací kroky:

1. Uživatel přejde do sekce „Create“ pomocí navigačního menu.
2. Uživatel vyplní ve formuláři povinné položky pro hlasovací formulář, může vyplnit také nepovinné položky. Výčet položek:
   - Pro hlasovací formulář povinné textové pole („Title“), nepovinné textové pole („Description“).
   - Pro 1. hlasovací kolo povinné textové pole („Title“), povinné pole s datem a časem ukončení hlasování „Round end“ (datum a čas musí být v budoucnosti), nepovinné textové pole („Description“).
   - Pro každou otázku povinné textové pole („Title“), nepovinné textové pole („Description“).
   - Pro každou odpověď povinné textové pole se zněním odpovědi.
3. Uživatel vyplní svůj vlastní email patřící k jeho ČVUT účtu do pole pod nadpisem „Voters“, poté ho přizve k hlasování stisknutím tlačítka „Invite“.
4. Uživatel vytvoří nový hlasovací formulář a jeho první kolo kliknutím na tlačítko „Create“.

Očekávaný výstup: Pole formuláře jsou znovu prázdná a je viditelná zpráva „New form created“. Uživateli přijde do emailové schránky pozvánka s údaji k přihlášení jako guest uživatel.

Scénář I04 - Zobrazení dostupných hlasování

Testovací data: Přihlášený interní ČVUT uživatel, vytvořený hlasovací formulář
Testovací kroky:

1. Uživatel přejde do sekce „Vote“ pomocí navigačního menu.
2. Pokud není automaticky vybrán formulář a kolo, které chce uživatel zobrazit, tak jej vybere pomocí nabídky hlasovacích formulářů a kol.

Očekávaný výstup: Uživatel vidí detaily vybraného hlasovacího formuláře a kola s hlasovacími možnostmi.

Scénář I05 - Provedení hlasování

Testovací data: Přihlášený interní ČVUT uživatel, vytvořený hlasovací formulář
Testovací kroky:

1. Uživatel přejde do sekce „Vote“ pomocí navigačního menu.
2. Pokud není automaticky vybrán formulář a kolo, ve kterém chce uživatel hlasovat, tak jej vybere pomocí nabídky hlasovacích formulářů a kol.
3. V sekci se zvoleným kolem uživatel vybere možnosti, pro které chce hlasovat.
4. Uživatel odešle vybrané možnosti kliknutím na tlačítko „Vote“.

Očekávaný výstup: Pod detaily vybraného hlasovacího kola je zobrazena zpráva „Voting successful“ a možnosti nelze již vybírat, ani odesílat.

Scénář I06 - Zobrazení spravovaných formulářů

Testovací data: Přihlášený interní ČVUT uživatel, vytvořený hlasovací formulář
Testovací kroky:

1. Uživatel přejde do sekce „Manage“ pomocí navigačního menu.
2. Pokud není automaticky vybrán formulář a kolo, které chce uživatel spravovat, tak jej vybere pomocí nabídky hlasovacích formulářů a kol.

Očekávaný výstup: Jsou zobrazeny detaily vybraného hlasovacího formuláře a jeho kola.

Scénář I07 - Vytvoření nového hlasovacího kola

Testovací data: Přihlášený interní ČVUT uživatel, vytvořený hlasovací formulář, povolené notifikace
Testovací kroky:

1. Uživatel přejde do sekce „Manage“ pomocí navigačního menu.
2. Pokud není automaticky vybrán formulář, který chce uživatel spravovat, tak jej vybere pomocí nabídky hlasovacích formulářů.
3. Uživatel zobrazí formulář pro tvorbu nového hlasovacího kola kliknutím na tlačítko „New“ v nabídce hlasovacích kol.
4. Uživatel vyplní ve formuláři povinné položky pro hlasovací kolo:
   - Pro nové hlasovací kolo povinné textové pole („Title“), povinné pole s datem a časem ukončení hlasování „Round end“ (datum a čas musí být v budoucnosti), nepovinné textové pole („Description“).
   - Pro každou otázku povinné textové pole („Title“), nepovinné textové pole („Description“).
   - Pro každou odpověď povinné textové pole se zněním odpovědi.
5. Uživatel vytvoří nové hlasovací kolo kliknutím na tlačítko „Create“.

Očekávaný výstup: Formulář pro tvorbu nového kola je skrytý, v nabídce kol je označeno nové kolo a jsou zobrazeny jeho detaily. Prohlížeč zobrazil notifikaci o dostupnosti nového hlasovacího kola.

Scénář I08 - Předčasné ukončení hlasovacího kola

Testovací data: Přihlášený interní ČVUT uživatel, vytvořený hlasovací formulář s probíhajícím kolem
Testovací kroky:

1. Uživatel přejde do sekce „Manage“ pomocí navigačního menu.

2. Pokud není automaticky vybrán formulář a kolo, které chce uživatel spravovat, tak jej vybere pomocí nabídky hlasovacích formulářů a kol.
3. Uživatel ukončí kolo kliknutím na tlačítko „Terminate“ v detailech kola zobrazených pod nabídkou kol.

Očekávaný výstup: Položka „Terminated“ obsahuje hodnotu „true“.

Scénář I09 - Zobrazení výsledků hlasování

Testovací data: Přihlášený uživatel, ukončené hlasovací kolo
Testovací kroky:

1. Uživatel přejde do sekce „Results“ pomocí navigačního menu.
2. Pokud není automaticky vybrán formulář a kolo, jehož výsledky chce uživatel zobrazit, tak jej vybere pomocí nabídky hlasovacích formulářů a kol.

Očekávaný výstup: V detailech kola zobrazených pod nabídkou kol jsou zobrazeny výsledky hlasování.

Scénář S10 - Odhlášení

Testovací data: Přihlášený uživatel
Testovací kroky:

1. Uživatel klikne na tlačítko „Logout“ v navigačním menu.

Očekávaný výstup: Pokud je uživatel přihlášen pomocí ČVUT SSO s platnou session, tak je přesměrován na logout stránku ČVUT SSO brány. Pokud již vypršela platnost ČVUT SSO session, tak je uživatel přesměrován na úvodní stranu aplikace s login formulářem a na navigačním menu je pouze položka „Home“.

8.2.3 Výstupy

Testování se zůčastnilo celkem 9 osob, z toho 6 jako guest uživatelé a 3 jako interní uživatelé ČVUT. Výstupy průchodů testovacími scénáři uvádím v tabulkách 8.1 a 8.2. Scénáře jsou označeny svým identifikátorem, úspěšnost průchodu daného uživatele je označena A pro úspěšný průchod a N pro neúspěšný. Uživatelé mohli také dobrovolně podat slovní zpětnou vazbu ke každému testovacímu scénáři a aplikaci jako celku.

ID	UG01	UG02	UG02	UG04	UG05	UG06
G01	A	A	A	A	A	A
G02	A	A	A	A	A	A
G03	A	A	A	A	A	A
G04	A	A	A	A	A	A
G05	A	A	A	A	A	A

Tabulka 8.1. Výstupy uživatelského testování - guest uživatelé.

Ve všech případech byly všechny testovací scénáře projity úspěšně, v rámci slovní zpětné vazby bylo identifikováno několik chyb a návrhů pro zlepšení. Na jejich základě jsem po ukončení uživatelského testování provedl následující úpravy:

- V rámci tvorby nového hlasovacího formuláře či kola:
  - Přesunutí tlačítek pro přidání dalších odpovědí z pravé strany na levou.

ID	UI01	UI02	UI02
I01	A	A	A
I02	A	A	A
I03	A	A	A
I04	A	A	A
I05	A	A	A
I06	A	A	A
I07	A	A	A
I08	A	A	A
I09	A	A	A
I10	A	A	A

Tabulka 8.2. Výstupy uživatelského testování - interní ČVUT uživatelé.

  - Úprava pole pro výběr data a času ukončení hlasování. Pole lze nyní vyplnit ručním zapsáním data a času ve správném formátu, v případě výběrů skrz prvek s kalendářem a hodinami jsou hodnoty aktualizovány okamžitě a není nutné jejich změnu potvrzovat. Výchozí čas změněn z T na $T + 1h$, kde T je aktuální čas.

- V rámci všech zobrazení otázek a možností odpovědí vybraného hlasovacího kola:
  - Viditelnější oddělení otázek a lepší zobrazení možných odpovědí.
- Oprava chyby způsobující zobrazení některých prvků ve špatném pořadí v prohlížeči Google Chrome.

Některé návrhy vyžadují větší množství úprav a určité chyby se mi nepodařilo reprodukovat. Implementace těchto vylepšení a oprav je možná v rámci případného budoucího vývoje aplikace, který je více rozebrán v sekci 9.1. Mezi možné úpravy vyplývající z uživatelského testování patří:

- Lepší nastavení možností otázek, především u otázek s možností více odpovědí naráz lze mít např. možnost nastavit minimální počet odpovědí.
- Sjednocení grafického vykreslení všech prvků mezi prohlížeči Google Chrome a Mozilla Firefox, některé prvky jsou vykresleny odlišně.
- Vylepšení grafického znázornění pokusu o opětovné připojení Socket.IO po přenačtení stránky.
- Oprava chyby, způsobující zaseknutí přihlašovací obrazovky ve stavu probíhajícího přihlašování v případě, že uživatel zvolil možnost přihlášení skrze ČVUT SSO, byl přesměrován na ČVUT SSO bránu a poté se bez provedení přihlášení vrátil zpět na přihlašovací obrazovku aplikace použitím tlačítka prohlížeče „Zpět“. Tuto chybu se mi nepodařilo zreprodukovat.

8.3 Shrnutí

Aplikaci jsem otestoval pomocí manuálního a uživatelského testování. V rámci manuálního testování jsem odhalil a opravil chyby spojené s funkčností a zabezpečením aplikace, v rámci kvalitativního uživatelského testování jsem se primárně zaměřil na zlepšení uživatelského komfortu. Funkcionalita byla tímto testováním ověřena a aplikaci lze tedy plnohodnotně používat.

Z časových důvodů jsem neimplementoval žádnou variantu automatického testování, v případě implementace automatických testů bych zvažoval použití nástrojů Selenium [63] nebo Cypress [64].

Kapitola 9
Závěr

Cílem této práce bylo najít vhodné technologie pro tvorbu aplikací pro anonymní hlasování, analyzovat již existující řešení, a na základě těchto poznatků navrhnout, implementovat a otestovat webovou single-page aplikaci vhodnou pro použití na FEL ČVUT.

Nejdříve jsem provedl analýzu principů anonymního hlasování a rešerši existujících řešení. Na základě získaných poznatků jsem provedl analýzu požadavků a případů použití potřebných pro návrh SPA, umožňující anonymní hlasování pro uživatele v rámci ČVUT a pozvané hosty. Poté jsem navrhl moderní architekturu SPA podporující mimo jiné i ověření pomocí ČVUT SSO a oboustrannou komunikací mezi klientem a serverem.

Dle takto získané specifikace jsem implementoval navrženou aplikaci, použitelnou na desktop i mobilních zařízeních. Vzniklou aplikaci jsem nasadil na virtuální linuxový server se systémem Ubuntu 20.04 a zpřístupnil na webové adrese `https://balloting.borek.cc`. Zdrojové kódy jsem poskytl svému vedoucímu práce ve školním GitLab repozitáři pro možnost případného dalšího vývoje a použití na FEL ČVUT. Na základě manuálního a uživatelského testování jsem identifikoval a opravil podstatné chyby a upravil grafické rozhraní pro větší uživatelský komfort.

Všichni uživatelé mohou hlasovat v přístupných hlasováních a zobrazit výsledky již proběhlých hlasování. Interní uživatelé ČVUT se mohou do aplikace přihlásit pomocí ČVUT SSO a mají navíc možnost vícekolová hlasování vytvářet a spravovat. K hlasování lze přizvat také uživatele mimo ČVUT, všichni přizvaní uživatelé se mohou do aplikace přihlásit pomocí přihlašovacích údajů zaslaných v emailové pozvánce. Aplikace neukládá informaci o tom, kdo pro co hlasoval, čímž je dosaženo plné anonymity. Veškerá komunikace mezi klientem a serverem je zabezpečená SSL certifikátem podepsaným důvěryhodnou autoritou a přístupová hesla jsou uložena v zahashované podobě.

Tvorba této webové aplikace pro mě byla velkým přínosem, jelikož jsem nikdy neimplementoval takto komplexní webovou aplikaci plně založenou na JavaScript technologiích. Naučil jsem se pracovat s mnoha moderními webovými technologiemi a zdokonalil jsem se v nasazování aplikací na server.

9.1 Budoucnost aplikace

Ačkoliv je vyvinutá aplikace plně funkční, tak poskytuje pouze omezené množství funkcí a je zde mnoho prostoru pro její vylepšení.

Z hlediska funkcionalit lze aplikaci rozšířit například o lepší nastavitelnost forem hlasování pro každou otázku, možnost přizvat hlasující do již probíhajícího hlasování, rozšíření typů uživatelů o pozorovatele hlasování, nebo poskytnout další možnosti pro přihlášení (např. použitím účtů Google nebo Meta). Vzhledem k možnosti používání webových aplikací na velkému počtu různých zařízení s různými webovými prohlížeči v různých verzích je možné se zaměřit na opravu grafických chyb způsobených rozdílnou konfigurací zařízení. Také se nabízí vylepšení vzhledu aplikace, který je sice funkční, ale velmi jednoduchý.

Z technického hlediska lze zjednodušit nasazování aplikace dockerizací, zlepšit odezvu optimalizací aplikační logiky, zpřehlednit kód větší modularizací, nebo obohatit aplikaci o automatické testy. Nabízí se také refaktorovat kód z jazyka JavaScript do jazyka TypeScript, který je na JavaScriptu založen, ale je silně typovaný. V případě velkého používání a rozšiřování aplikace by bylo také výhodné nastavit automatizovaný CICD (Continuous Integration and Continuous Delivery) proces, který by usnadnil nasazování nových verzí.

Literatura

[1] *Wikipedia - Secret ballot* [online]. [vid. 2022-04-07]. Dostupné na http://www.bowlesfluidics.com/capabilites/technology/.

[2] *ADoodle* [online]. [vid. 2022-04-07]. Dostupné na https://adoodle.org/ .

[3] *Polys* [online]. [vid. 2022-04-07]. Dostupné na https://polys.me/ .

[4] *Anonyvoter* [online]. [vid. 2022-05-03]. Dostupné na https://anonyvoter.com/ .

[5] *Election Runner* [online]. [vid. 2022-05-03]. Dostupné na https://electionrunner.com/ .

[6] *ElectionBuddy* [online]. [vid. 2022-05-03]. Dostupné na https://electionbuddy.com/ .

[7] *The App Solutions - Functional vs Non-functional Requirements: Main Differences & Examples* [online]. [vid. 2022-04-20]. Dostupné na https://theappsolutions.com/blog/development/functional-vs-non-functional-requirements/ .

[8] *Visual Paradigm - What is Use Case Diagram?* [online]. [vid. 2022-04-20]. Dostupné na https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ .

[9] *Simform - An Ultimate Guide to Web Application Architecture* [online]. [vid. 2022-04-23]. Dostupné na https://www.simform.com/blog/web-application-architecture/ .

[10] *MuleSoft - Microservices vs Monolithic Architecture* [online]. [vid. 2022-04-26]. Dostupné na https://www.mulesoft.com/resources/api/microservices-vs-monolithic .

[11] *Cloudflare - What is BaaS?* [online]. [vid. 2022-04-26]. Dostupné na https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/ .

[12] *Cloudflare - What is Function-as-a-Service (FaaS)?* [online]. [vid. 2022-04-23]. Dostupné na https://www.cloudflare.com/learning/serverless/glossary/function-as-a-service-faas/ .

[13] *MDN web docs - Introduction to progressive web apps* [online]. [vid. 2022-04-24]. Dostupné na https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Introduction .

[14] *MongoDB - NoSQL vs SQL Databases* [online]. [vid. 2022-04-26]. Dostupné na https://www.mongodb.com/nosql-explained/nosql-vs-sql .

[15] *DB-Engines - DB-Engines Ranking - Trend Popularity* [online]. [vid. 2022-04-26]. Dostupné na
https://db-engines.com/en/ranking_trend .

[16] *JavaPoint - Difference between MySQL and Oracle* [online]. [vid. 2022-04-26]. Dostupné na
https://www.javatpoint.com/mysql-vs-oracle .

[17] *Jelvix - Choosing between MySQL vs PostgreSQL vs SQL Server* [online]. [vid. 2022-04-26]. Dostupné na
https://jelvix.com/blog/mysql-postgresql-sql-server .

[18] *Spring - Overview of Spring Framework* [online]. [vid. 2022-04-26]. Dostupné na
https://docs.spring.io/spring-framework/docs/5.0.0.RC2/spring-framework-reference/overview.html .

[19] *Microsoft - Overview to ASP.NET Core* [online]. [vid. 2022-04-26]. Dostupné na
https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0 .

[20] *Node.js - About* [online]. [vid. 2022-04-26]. Dostupné na
https://nodejs.org/en/about/ .

[21] *Express* [online]. [vid. 2022-04-26]. Dostupné na
https://expressjs.com/ .

[22] *Medium - Learn all about Different ORM in Node JS 2021* [online]. [vid. 2022-04-28]. Dostupné na
https://medium.com/tkssharma/learn-all-about-different-orm-in-node-js-2021-38536828c1f0 .

[23] *Knex.js* [online]. [vid. 2022-04-28]. Dostupné na
https://knexjs.org/ .

[24] *Sequelize* [online]. [vid. 2022-04-28]. Dostupné na
https://sequelize.org/ .

[25] *TypeORM* [online]. [vid. 2022-04-28]. Dostupné na
https://typeorm.io/ .

[26] *Prisma - What is Prisma?* Dostupné na
https://www.prisma.io/docs/concepts/overview/what-is-prisma.

[27] *State of JS - Front-end Frameworks 2021* [online]. [vid. 2022-04-27]. Dostupné na
https://2021.stateofjs.com/en-US/libraries/front-end-frameworks/ .

[28] *Medium - JavaScript brief history and ECMAScript(ES6.* [vid. 2022-04-27]. Dostupné na
https://madasamy.medium.com/javascript-brief-history-and-ecmascript-es6-es7-es8-features-673973394df4 .

[29] *Plain English - JavaScript Frameworks.* [vid. 2022-04-27]. Dostupné na
https://javascript.plainenglish.io/javascript-frameworks-performance-comparison-2020-cd881ac21fce .

[30] *SitePoint - The 5 Most Popular Front-end Frameworks Compared* [online]. [vid. 2022-04-27]. Dostupné na
https://www.sitepoint.com/most-popular-frontend-frameworks-compared/ .

[31] *RedHar - What is a REST API?* [online]. [vid. 2022-05-03]. Dostupné na
https://www.redhat.com/en/topics/api/what-is-a-rest-api .

[32] *Wikipedia - WebSocket* [online]. [vid. 2022-05-03]. Dostupné na https://en.wikipedia.org/wiki/WebSocket .

[33] *MDN web docs - The WebSocket API (WebSockets)* [online]. [vid. 2022-05-03]. Dostupné na https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API .

[34] *Socket.IO - Introduction* [online]. [vid. 2022-05-03]. Dostupné na https://socket.io/docs/v4/ .

[35] *Medium - Web Authentication: Cookies vs. Tokens* [online]. [vid. 2022-05-03]. Dostupné na https://blog.bitsrc.io/web-authentication-cookies-vs-tokens-8e47d5a96d34 .

[36] *PingIdentity - Ultimate Guide to Token-based Authentication* [online]. [vid. 2022-05-03]. Dostupné na https://www.pingidentity.com/en/resources/blog/posts/2021/ultimate-guide-token-based-authentication.html .

[37] *Socket.IO - Middlewares* [online]. [vid. 2022-05-03]. Dostupné na https://socket.io/docs/v4/middlewares/ .

[38] *Github - Thream - socketio-jwt* [online]. [vid. 2022-05-03]. Dostupné na https://github.com/Thream/socketio-jwt .

[39] *Github - adcentury - socketio-jwt-auth* [online]. [vid. 2022-05-03]. Dostupné na https://github.com/adcentury/socketio-jwt-auth .

[40] *Keycloak - About* [online]. [vid. 2022-05-03]. Dostupné na https://www.keycloak.org/about .

[41] *Wikipedia - Class diagram* [online]. [vid. 2022-04-28]. Dostupné na https://en.wikipedia.org/wiki/Class_diagram .

[42] *Moqups* [online]. [vid. 2022-05-11]. Dostupné na https://moqups.com/ .

[43] *Visual Studio Code* [online]. [vid. 2022-05-05]. Dostupné na https://code.visualstudio.com/ .

[44] *Visual Studio Code - Getting Started* [online]. [vid. 2022-05-05]. Dostupné na https://code.visualstudio.com/docs .

[45] *Github - Microsoft - Terminal* [online]. [vid. 2022-05-05]. Dostupné na https://github.com/Microsoft/Terminal .

[46] *NPM - About* [online]. [vid. 2022-05-05]. Dostupné na https://docs.npmjs.com/about-npm.

[47] *Github - awslabs - aws-jwt-verify* [online]. [vid. 2022-05-17]. Dostupné na https://github.com/awslabs/aws-jwt-verify .

[48] *Github - auth0 - node-jsonwebtoken* [online]. [vid. 2022-05-17]. Dostupné na https://github.com/auth0/node-jsonwebtoken .

[49] *Nodemailer* [online]. [vid. 2022-05-11]. Dostupné na https://nodemailer.com/about/ .

[50] *Vue.js - Router* [online]. [vid. 2022-05-12]. Dostupné na https://router.vuejs.org/ .

[51] *Github - MetinSeylan - Vue-Socket.io* [online]. [vid. 2022-05-12]. Dostupné na https://github.com/MetinSeylan/Vue-Socket.io .

[52] *Github - soukoku - vue-oidc-client* [online]. [vid. 2022-05-12]. Dostupné na https://github.com/soukoku/vue-oidc-client .

[53] *Vue.js - Hot Reload* [online]. [vid. 2022-05-12]. Dostupné na https://vue-loader.vuejs.org/guide/hot-reload.html .

[54] *Google Cloud Platform - Free trial and free tier* [online]. [vid. 2022-05-05]. Dostupné na https://cloud.google.com/free.

[55] *DigitalOcean - How To Install and Use PostgreSQL on Ubuntu 20.04* [online]. [vid. 2022-05-05]. Dostupné na https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-20-04 .

[56] *Let's Encrypt* [online]. [vid. 2022-05-05]. Dostupné na https://letsencrypt.org/ .

[57] *Apache* [online]. [vid. 2022-05-05]. Dostupné na https://httpd.apache.org/ .

[58] *DigitalOcean - How To Install the Apache Web Server on Ubuntu 20.04* [online]. [vid. 2022-05-05]. Dostupné na https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04 .

[59] *DigitalOcean - How To Set Up Apache Virtual Hosts on Ubuntu 20.04* [online]. [vid. 2022-05-05]. Dostupné na https://www.digitalocean.com/community/tutorials/how-to-set-up-apache-virtual-hosts-on-ubuntu-20-04 .

[60] *Certbot* [online]. [vid. 2022-05-05]. Dostupné na https://certbot.eff.org/ .

[61] *Porkbun* [online]. [vid. 2022-05-05]. Dostupné na https://porkbun.com/ .

[62] *TestingXperts - User Testing Guide* [online]. [vid. 2022-05-11]. Dostupné na https://www.testingxperts.com/blog/user-testing .

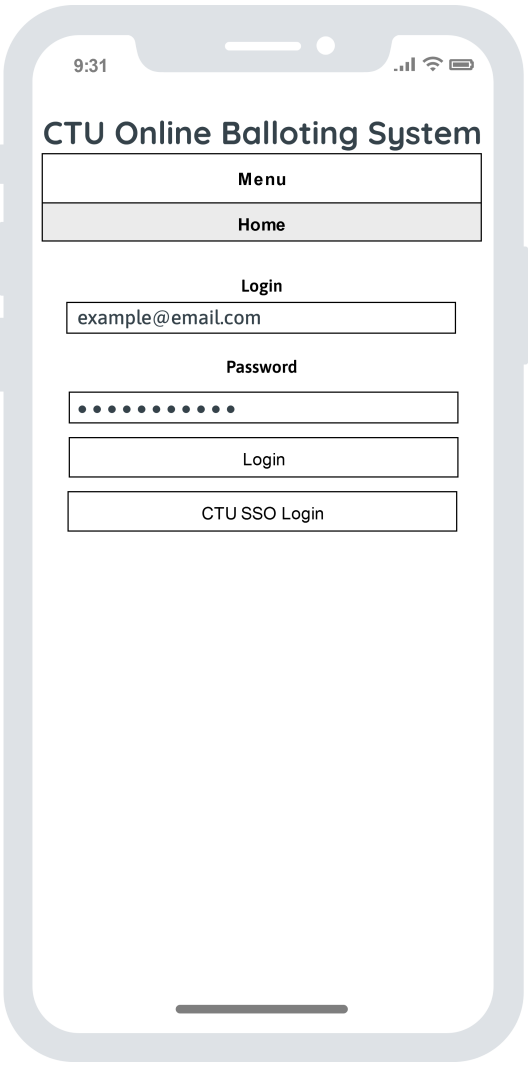
[63] *Selenium* [online]. [vid. 2022-05-10]. Dostupné na https://www.selenium.dev/ .

[64] *Cypress* [online]. [vid. 2022-05-10]. Dostupné na https://www.cypress.io/ .

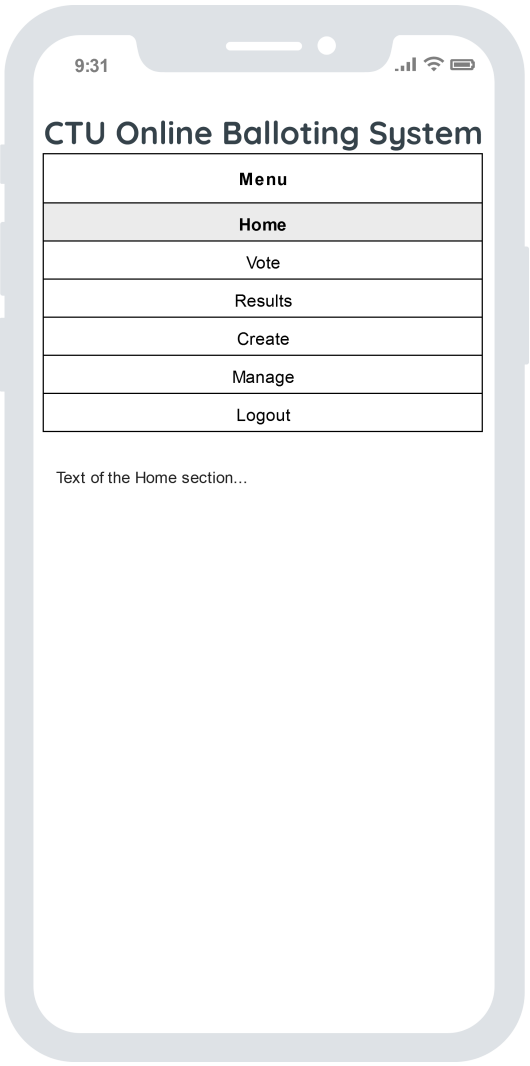
Příloha A
Zkratky

API	Application Programming Interface
BE	Backend
BaaS	Backend-as-a-Service
CICD	Continuous Integration and Continuous Delivery
CSS	Cascading Style Sheets
CTU	Czech technical university in Prague
ČVUT	České vysoké učení technické v Praze
DB	Databáze
ES5	ECMAScript 5
ES6	ECMAScript 6
FEL	Fakulta elektrotechnická
FE	Frontend
FaaS	Function-as-a-Service
GB	Gigabyte
GPC	Google Cloud Platform
GUI	Graphical User Interface
HDD	Hard Disk Drive
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated development environment
ID	Unikátní identifikátor
JSON	JavaScript Object Notation
JS	JavaScript
JWT	JSON Web Token
ORM	Object–relational mapping
PWA	Progresivní webová aplikace; Progressive web application
RAM	Random Access Memory
REST	Representational State Transfer
SPA	Single-page aplikace
SSH	Secure Shell Protocol
SSL	Secure Sockets Layer
UI	User Interface
XML	Extensible Markup Language

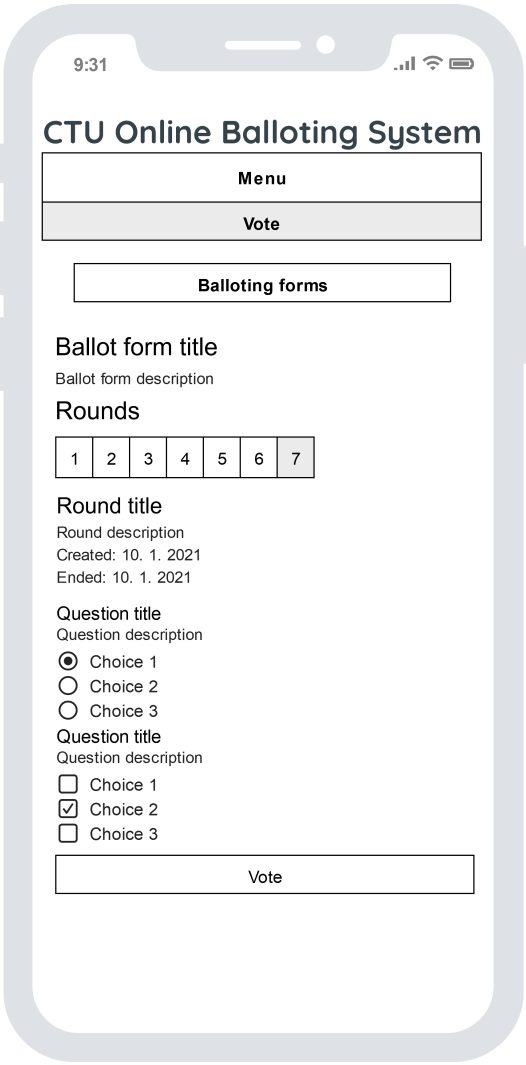
Příloha B
Návrhy obrazovek



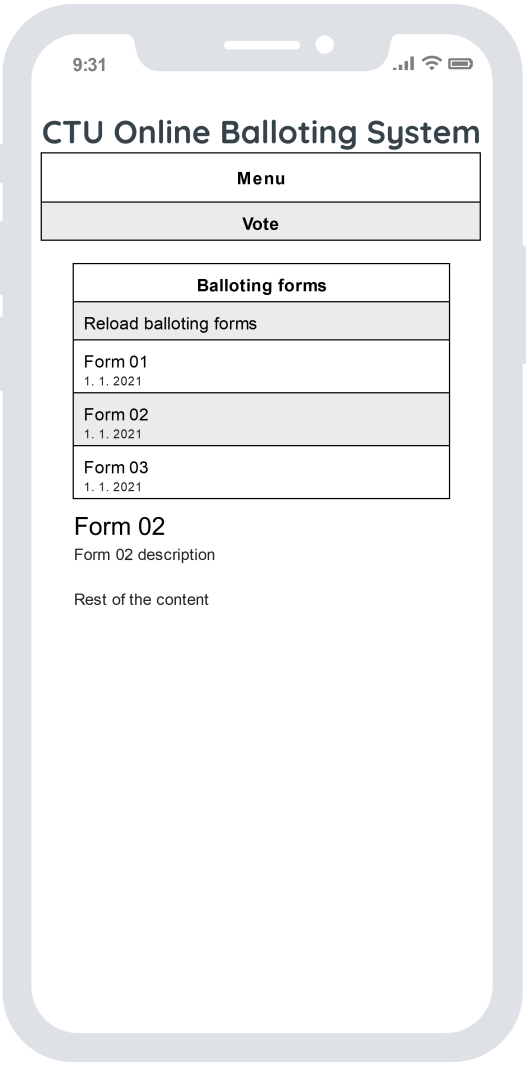
Obrázek B.1. Návrh obrazovky - Login.



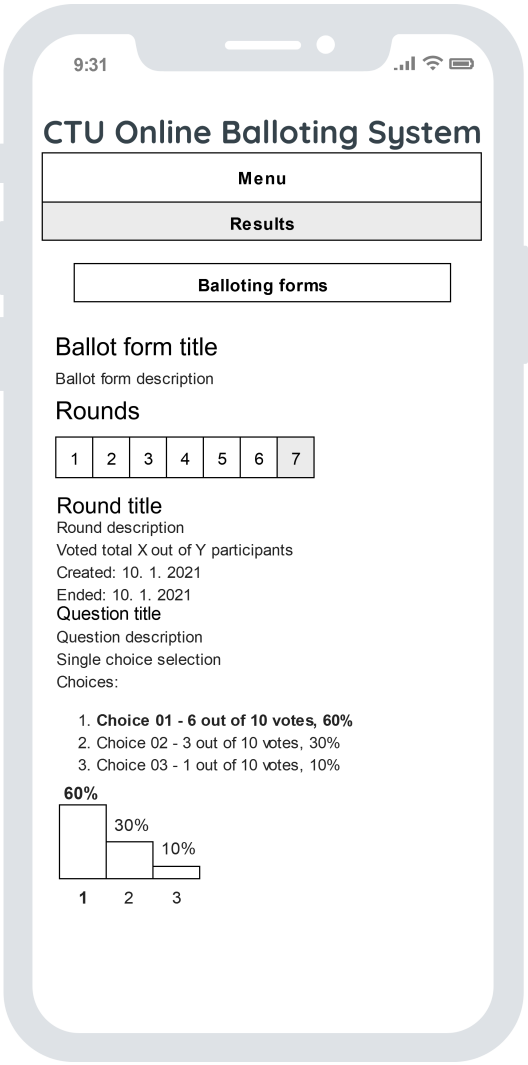
Obrázek B.2. Návrh obrazovky - Domovská stránka.



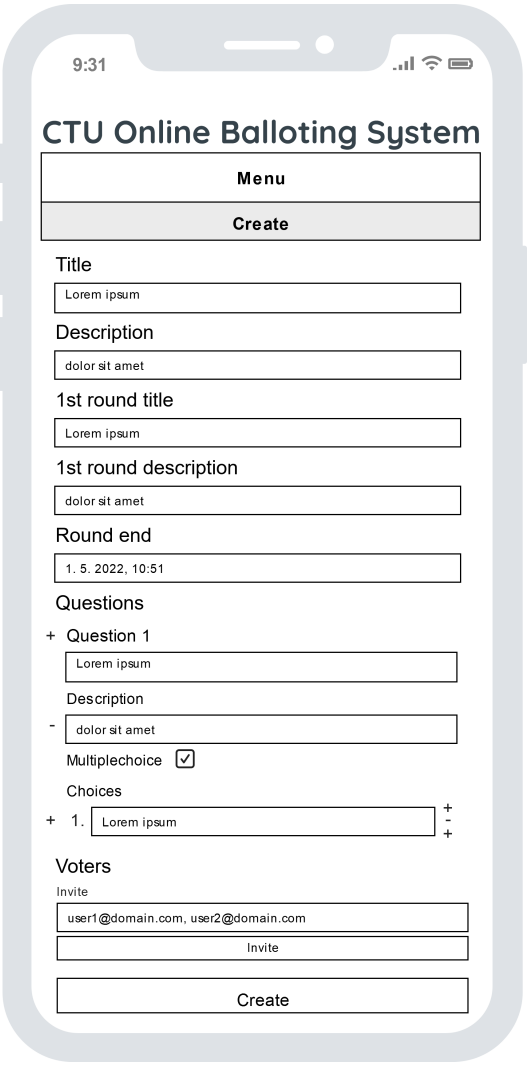
Obrázek B.3. Návrh obrazovky - Hlasovat.



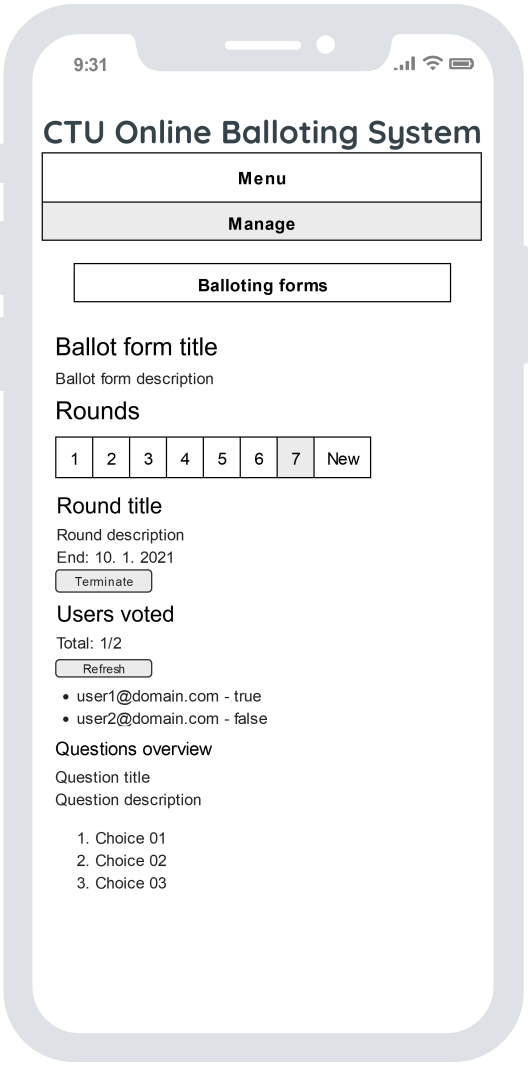
Obrázek B.4. Návrh obrazovky - Menu pro výběr hlasovacího formuláře.



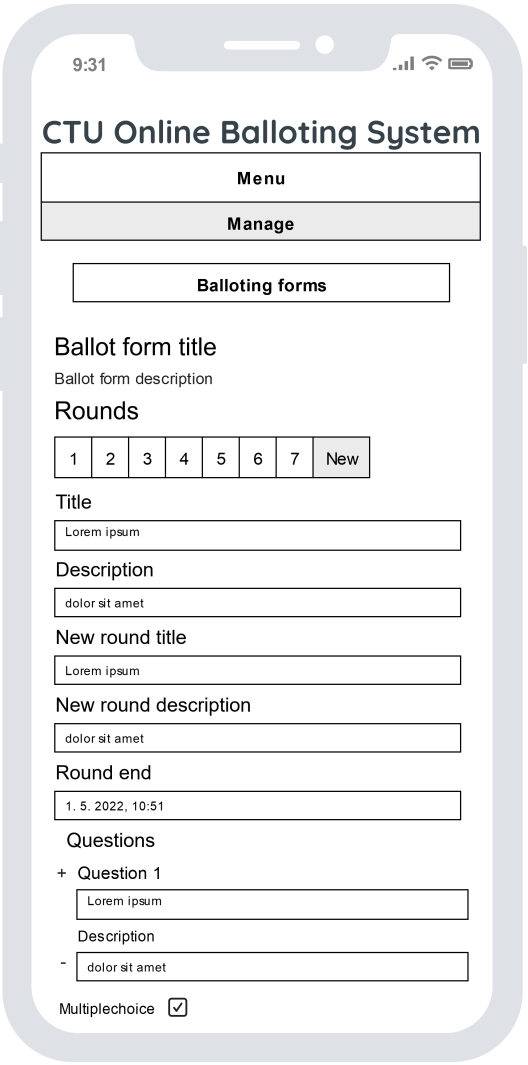
Obrázek B.5. Návrh obrazovky - Výsledky.



Obrázek B.6. Návrh obrazovky - Nový hlasovací formulář.



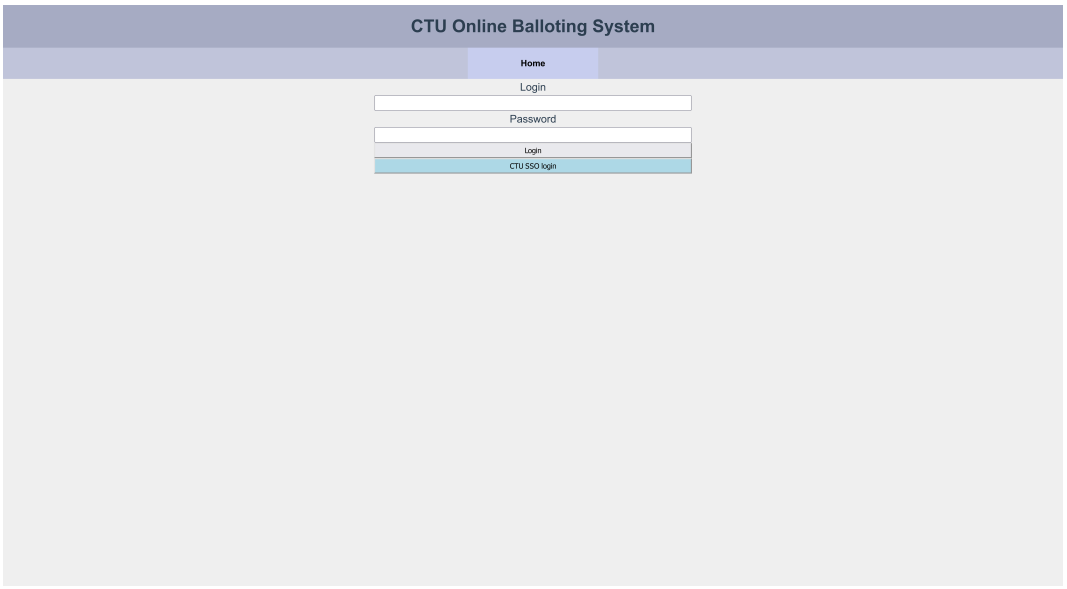
Obrázek B.7. Návrh obrazovky - Správa hlasování - Existující kolo.



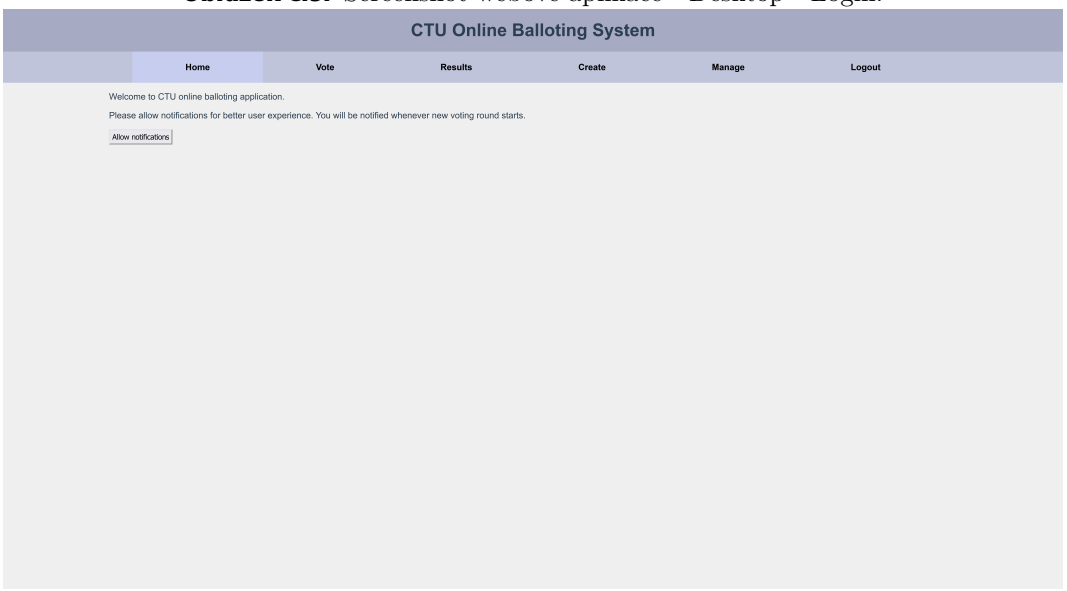
Obrázek B.8. Návrh obrazovky - Správa hlasování - Nové kolo.

Příloha C
Screenshoty webové aplikace

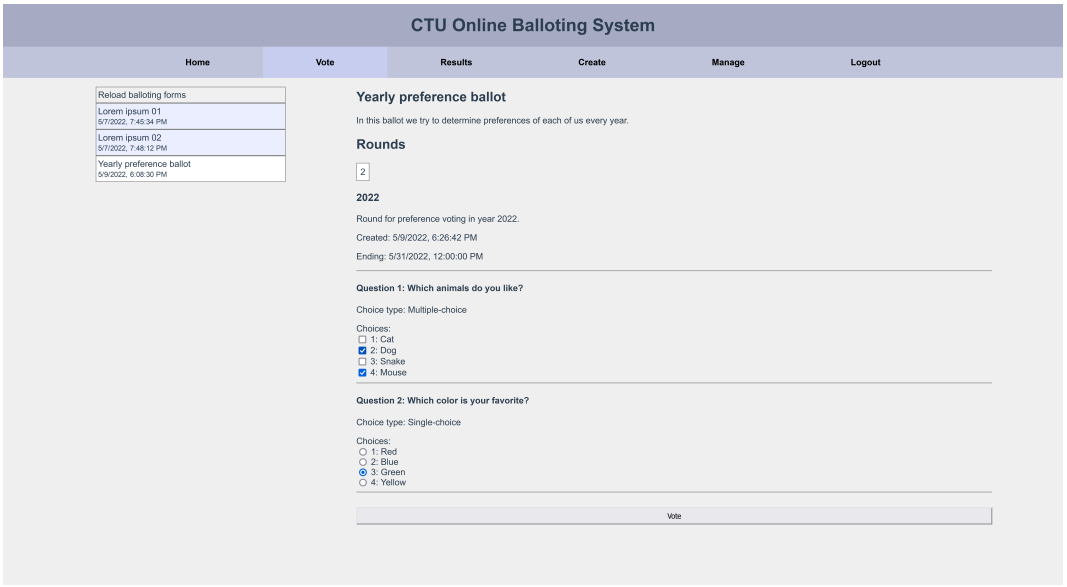
C.1 Desktop



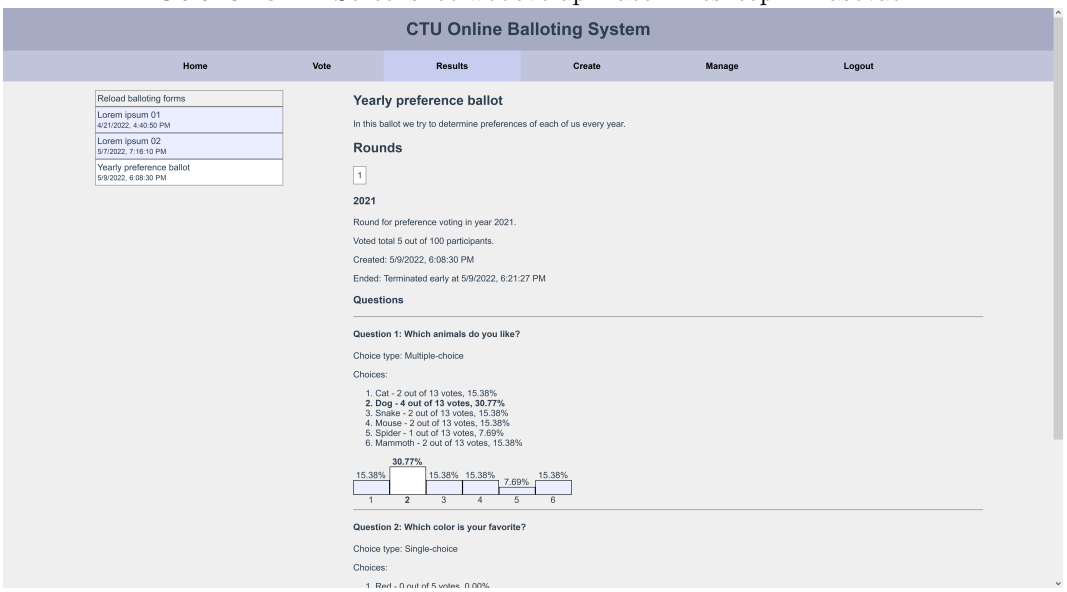
Obrázek C.9. Screenshot webové aplikace - Desktop - Login.



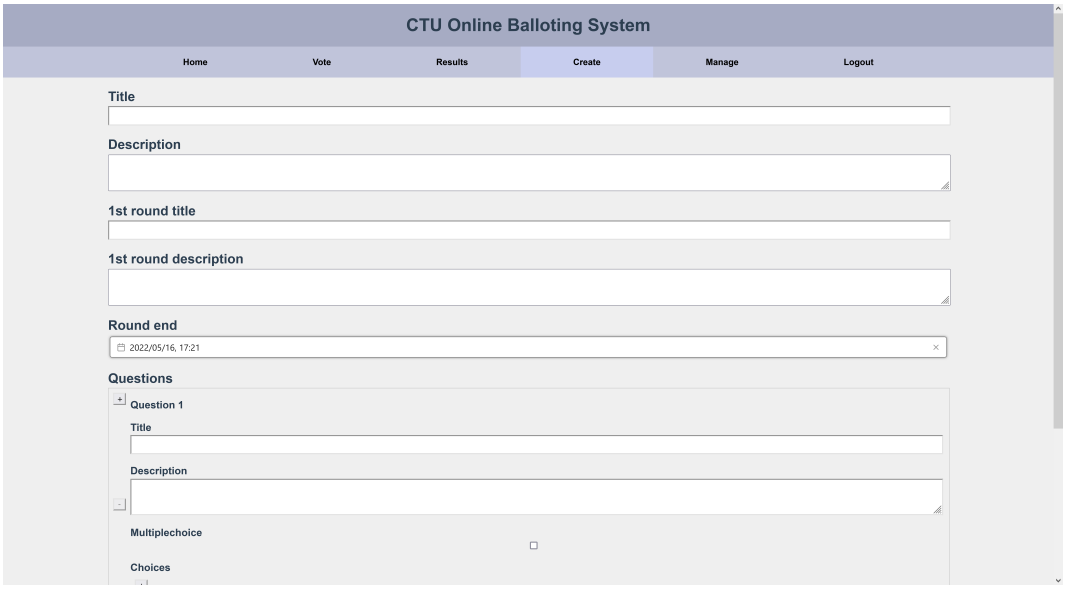
Obrázek C.10. Screenshot webové aplikace - Desktop - Úvodní stránka.



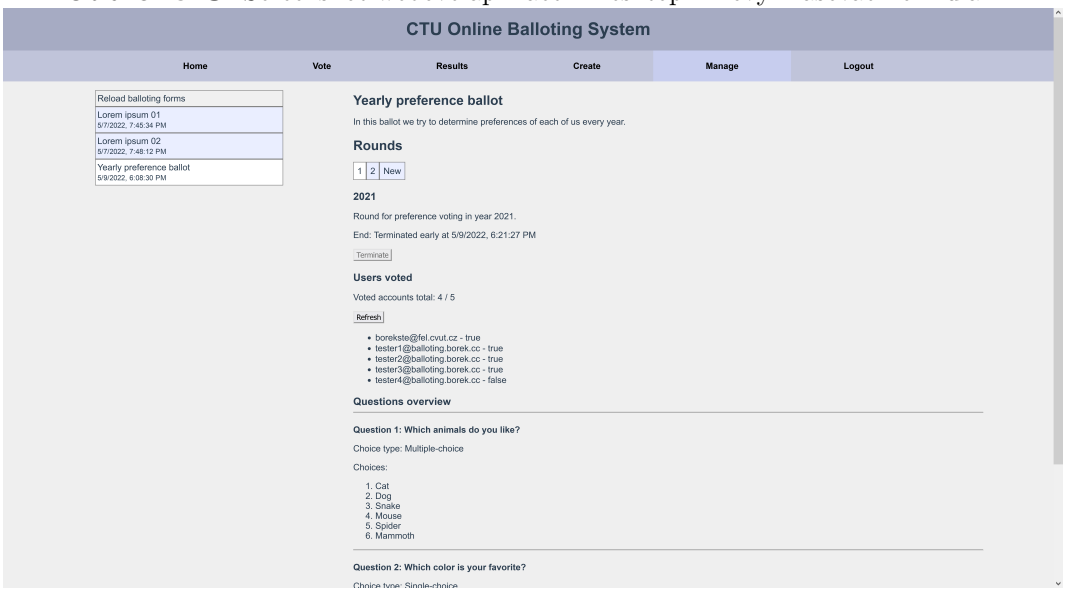
Obrázek C.11. Screenshot webové aplikace - Desktop - Hlasovat.



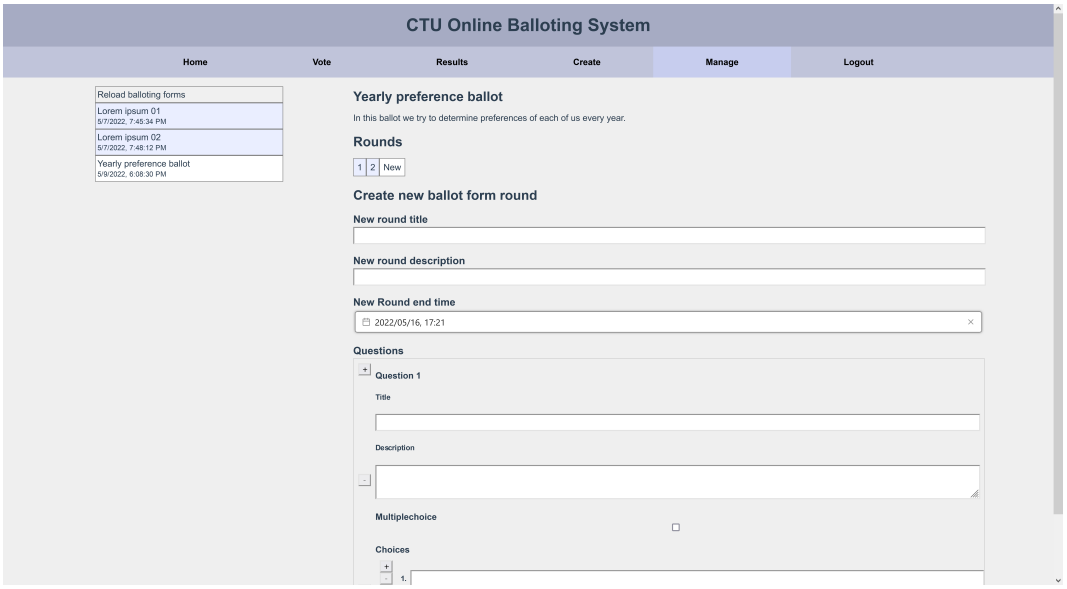
Obrázek C.12. Screenshot webové aplikace - Desktop - Výsledky.



Obrázek C.13. Screenshot webové aplikace - Desktop - Nový hlasovací formulář.

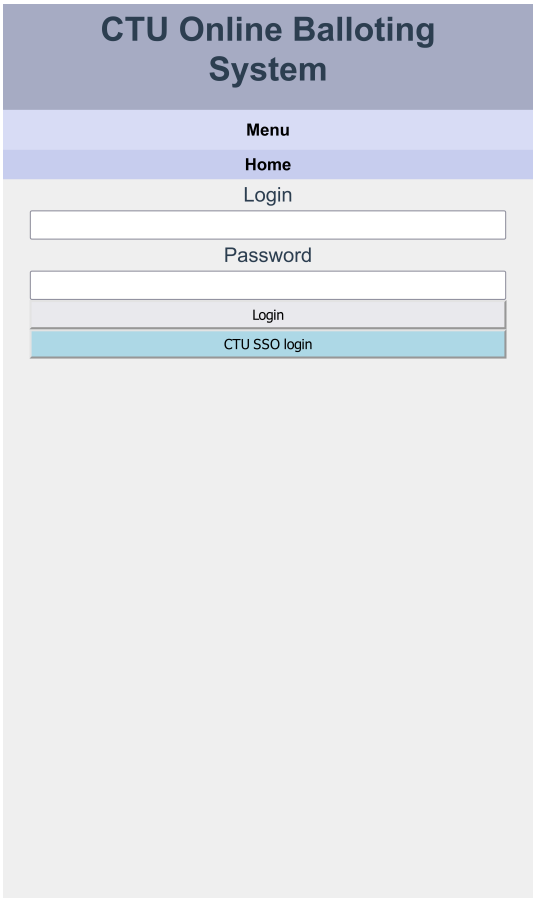


Obrázek C.14. Screenshot webové aplikace - Desktop - Správa existujících hlasovacích formulářů - Existující kolo.

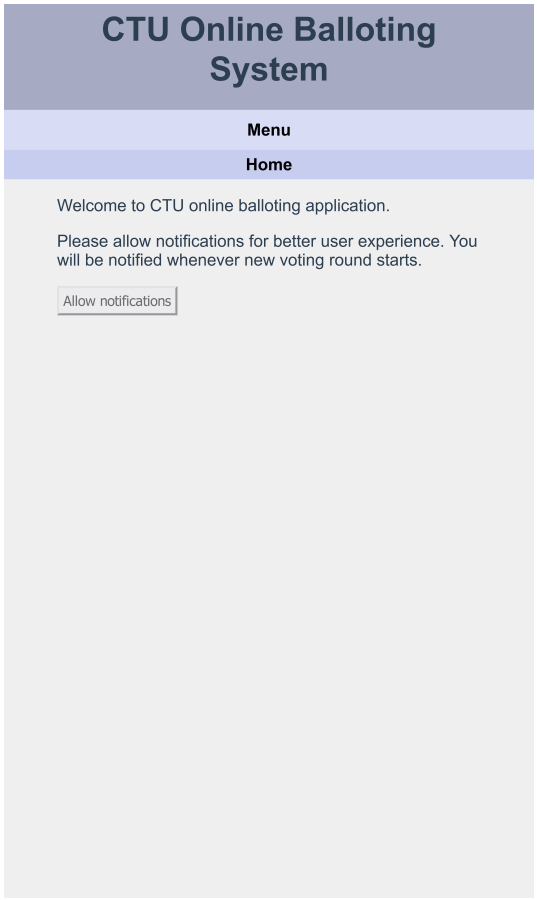


Obrázek C.15. Screenshot webové aplikace - Desktop - Správa existujících hlasovacích formulářů - Nové kolo.

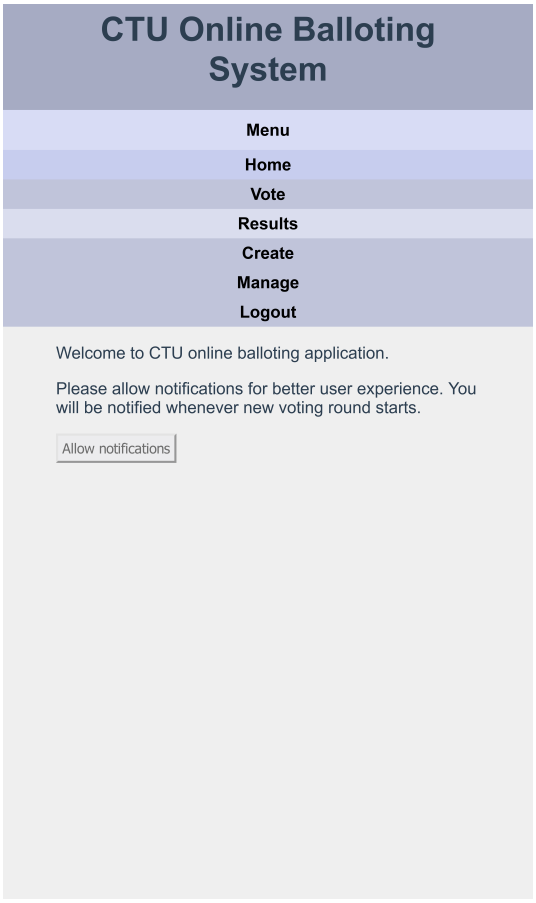
C.2 Mobilní zařízení



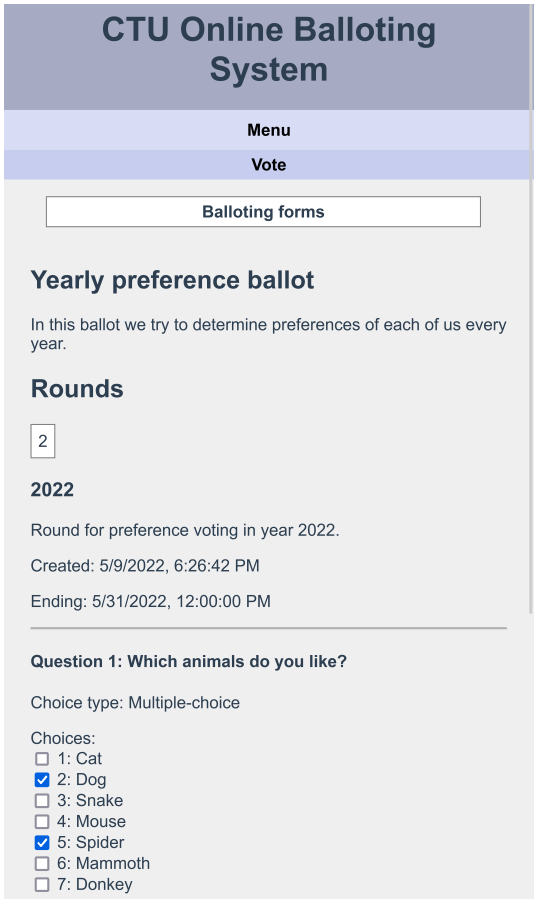
Obrázek C.16. Screenshot webové aplikace - Mobilní zařízení - Login.



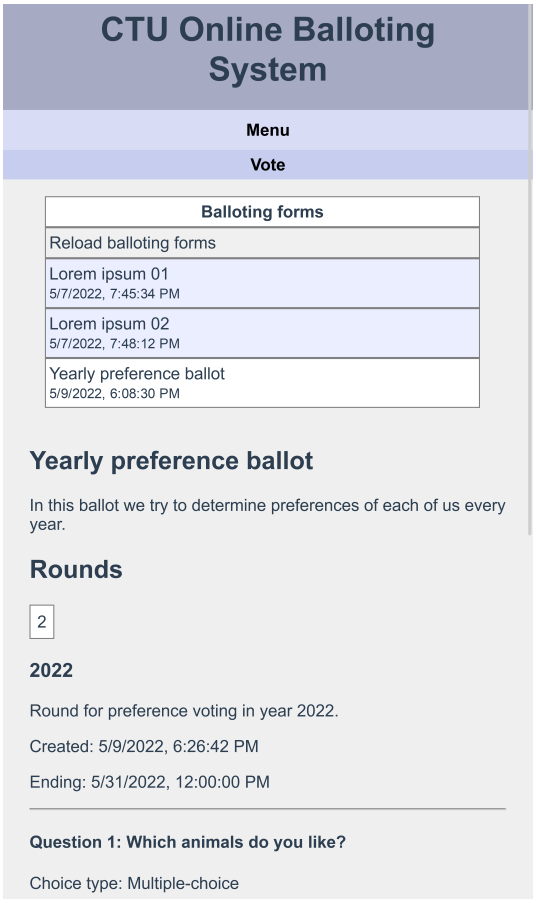
Obrázek C.17. Screenshot webové aplikace - Mobilní zařízení - Úvodní stránka.



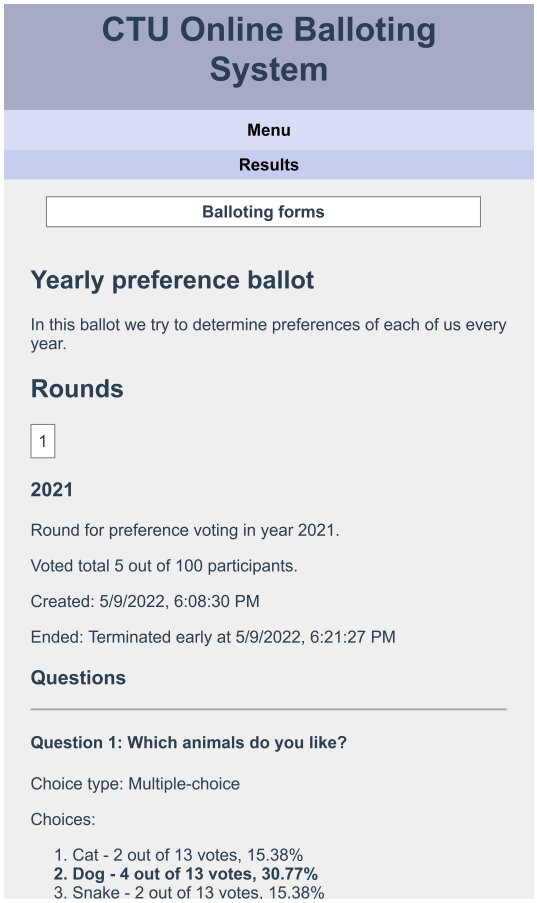
Obrázek C.18. Screenshot webové aplikace - Mobilní zařízení - Hlavní menu.



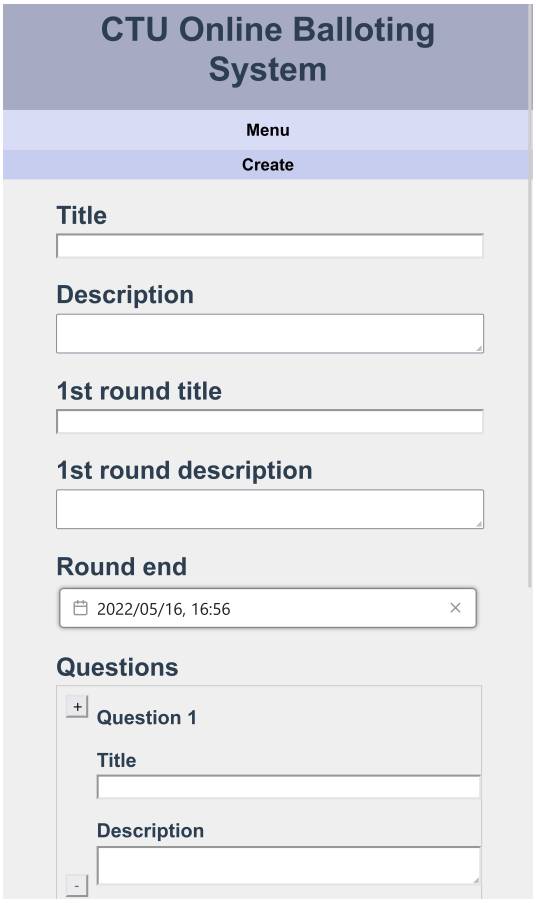
Obrázek C.19. Screenshot webové aplikace - Mobilní zařízení - Hlasovat.



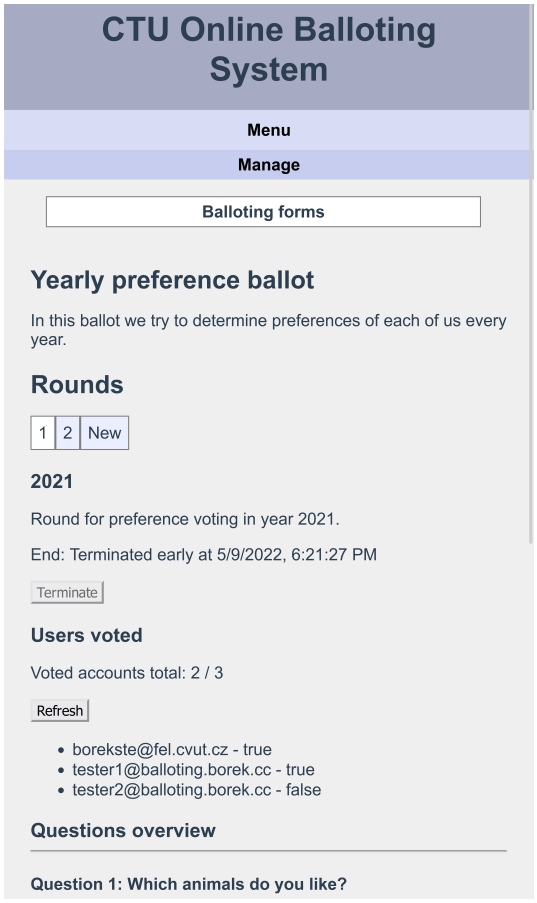
Obrázek C.20. Screenshot webové aplikace - Mobilní zařízení - Hlasovat - Nabídka hlasovacích formulářů.



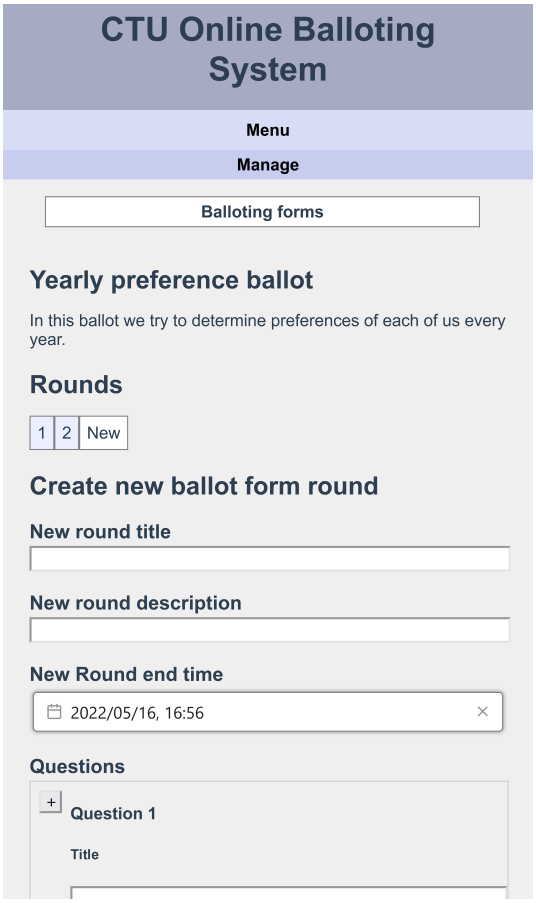
Obrázek C.21. Screenshot webové aplikace - Mobilní zařízení - Výsledky.



Obrázek C.22. Screenshot webové aplikace - Mobilní zařízení - Nový hlasovací formulář.



Obrázek C.23. Screenshot webové aplikace - Mobilní zařízení - Správa existujících hlasovacích formulářů - Existující kolo.



Obrázek C.24. Screenshot webové aplikace - Mobilní zařízení - Správa existujících hlasovacích formulářů - Nové kolo.