Replace the contents of this file with official assignment. Místo tohoto souboru sem patří list se zadáním závěrečné práce.

Diplomová práce

MOBILNÍ APLIKACE PRO ZOBRAZENÍ VÝSLEDKŮ HLASOVÁNÍ POSLANECKÉ SNĚMOVNY

Bc. Lukáš Dang

Fakulta informačních technologií Katedra webového inženýrství Vedoucí: Ing. Ondřej John 6. února 2023

České vysoké učení technické v Praze Fakulta informačních technologií

 $\ensuremath{{\mathbb O}}$ 2023 Bc. Lukáš Dang. Všechna práva vyhrazena.

Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení, je nezbytný souhlas autora.

Odkaz na tuto práci: Dang Lukáš. *Mobilní aplikace pro zobrazení výsledků hlasování Poslanecké sněmovny*. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Obsah

Poděkování	viii
Prohlášení	ix
Abstrakt	x
Shrnutí	xi
Seznam zkratek	xii
1 Cîle	1
Poslanecká sněmovna 2.1 Poslanecká sněmovna	1
3 Funkční a nefunkční požadavky 3.1 Funkční požadavky	
4 Analýza existujících řešení 4.1 politiscope	
5 Analýza zdrojových dat 5.1 Zdroj	

iv Obsah

6	Náv			1
	6.1	Uživat	telské rozhraní	. 1
	6.2	REST	`API	. 2
	6.3	Datab	pázové datové struktury	. 4
7	Imp	lemen	ntace	1
	7.1		ní aplikace	. 1
		7.1.1	Použité nástroje a technologie	
		7.1.2	Implementace uživatelského rozhraní	
		7.1.3	Implementace prezentační vrstvy	
		7.1.4	Implementace doménové vrstvy	
		7.1.5	Implementace datové vrstvy	
		7.1.6	Implementace síťové vrstvy	
	7.2	-	end	
	1.2	7.2.1		
			Použité nástroje a technologie	
		7.2.2	Prezentační vrstva	
		7.2.3	Doménová vrstva	
		7.2.4	Databázová vrstva	
	7.3	-	ování dat	
		7.3.1	Stahování zdrojových souborů	
		7.3.2	Extrakce datových souborů	. 19
		7.3.3	Pročištění dat	. 20
		7.3.4	Parsování dat	. 20
		7.3.5	Transformace dat	. 21
8	Test	ování		1
	8.1		ní aplikace	. 1
	8.2		end	
	0.2	Bacher		
9	Nas	azení		1
	9.1	Aplika	ace	. 1
	9.2	Backer	end	. 1
10	Spu	štění		1
	_		ace	. 1
		-	end	
11	Záv			1
11	Lav	er.		
A	Příl	oha		3
Ol	osah	přilož	ženého média	17

Seznam obrázků

4.1 4.2	Android aplikace politiscope									
A.1	Seznam hlasování								. 3	3
A.2	Vyhledávání v seznamu hlasování									
A.3	Obrazovka pro seznam hlasování									
A.4	Detail hlasování									
A.5	Jak hlasovaly kluby									
A.6	Jak hlasovaly kluby									
A.7	Obrazovky pro detail hlasování									
A.8	Seznam poslanců									
A.9	Vyhledávání v seznamu poslanců									
	Obrazovka pro seznam poslanců									
	Detail poslance									
	2 Jak hlasoval/a poslanec/kyně									
	3 Obrazovky pro detail poslance									
	Seznam nastavení									
	5 Nastavení volebního období									
	6 Obrazovka pro nastavení									
	'Adresářová struktura mobilní aplikace									
		Se	21	ıa	m	. t	ā	ıb	ulek	
3.1 3.2	Funkční požadavky pro mobilní aplikaci								. 2	2
3.2	Funkční požadavky pro back-end								. 2	2
									. 22	2
3.2 3.3 3.4 5.1	Funkční požadavky pro back-end								. 22	2
3.2 3.3 3.4 5.1 5.2	Funkční požadavky pro back-end								. 22	2
3.2 3.3 3.4 5.1	Funkční požadavky pro back-end								. 22	2 2 3 3
3.2 3.3 3.4 5.1 5.2 5.2 5.3	Funkční požadavky pro back-end. Nefunkční požadavky pro mobilní aplikaci. Nefunkční požadavky pro back-end. Typy dat sloupců v tabulkách. Tabulka typ_organu Tabulka typ_organu Tabulka organy.								. 22	22334
3.2 3.3 3.4 5.1 5.2 5.2 5.3 5.4	Funkční požadavky pro back-end. Nefunkční požadavky pro mobilní aplikaci. Nefunkční požadavky pro back-end. Typy dat sloupců v tabulkách. Tabulka typ_organu Tabulka typ_organu Tabulka organy. Tabulka osoby.								. 22	
3.2 3.3 3.4 5.1 5.2 5.2 5.3 5.4 5.5	Funkční požadavky pro back-end. Nefunkční požadavky pro mobilní aplikaci. Nefunkční požadavky pro back-end. Typy dat sloupců v tabulkách Tabulka typ_organu Tabulka typ_organu Tabulka organy Tabulka osoby Tabulka zarazeni								. 22	
3.2 3.3 3.4 5.1 5.2 5.2 5.3 5.4 5.5	Funkční požadavky pro back-end. Nefunkční požadavky pro mobilní aplikaci. Nefunkční požadavky pro back-end. Typy dat sloupců v tabulkách Tabulka typ_organu Tabulka typ_organu Tabulka organy Tabulka osoby Tabulka zarazeni Tabulka zarazeni								. 22 . 33 . 44 . 44 . 55 . 66 . 6	
3.2 3.3 3.4 5.1 5.2 5.2 5.3 5.4 5.5	Funkční požadavky pro back-end. Nefunkční požadavky pro mobilní aplikaci. Nefunkční požadavky pro back-end. Typy dat sloupců v tabulkách Tabulka typ_organu Tabulka typ_organu Tabulka organy Tabulka osoby Tabulka zarazeni								. 22 . 33 . 34 . 44 . 45 . 66 . 66 . 66 . 66	

5.8 5.9	Tabulka hl_poslanec	
A.1 A.2 A.3 A.4 A.5 A.6 A.7	Struktura agency	7
	Seznam výpisů	kódu
7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 7.10	Příklad použití coroutiny Praktický příklad použití coroutiny Příklad použití flow Příklad intermediate operátoru pro flow Příklad použití DI pomocí knihovny Hilt Příklad konfigurace závislostí pro Hilt Příklad použití composable funkce Text. Příklad skládání composable funkcí. Vykreslí ikonku a text vedle sebe. Příklad parametrů pro změnu vzhledu a chování. Příklad composable funkce používající lokální stav.	3 4 4 5 5 6 6 7
7.11 7.12 7.13 7.14 7.15	Příklad composable funkce používající stav z ViewModelu. Třída MainActivity. (Soubor psp_fe/app/MainActivity). Ukázka využití view modelu. Ukázka využití use caseu pro získání seznamu hlasování. Ukázka využití use caseu pro získání detailu hlasování.	8 8 10 11
7.17 7.18 7.19 7.20 7.21	Ukázka datové vrstvy pro data o hlasováních Ukázka datového zdroje Ukázka použití knihovny Retrofit pro získání seznamu hlasování z backendu Ukázka kódu pro vytvoření endpointu Ukázka endpointu s request parametrem Ukázka nastavení hlaviček pro stránkování	12 12 14 15 15
7.23 7.24 7.25 7.26	Ukázka kódu pro získání detailu poslance Ukázka doménové vrstvy pro vrácení seznamu poslanců Ukázka kódu pro sestavení objektu pro stránkování Ukázka dopočtu statistik pro detail hlasování za běhu v doménové vrstvě Entita Vote reprezentující hlasování Repozitář pro hlasování	16 16 17 17
7.28 7.29 7.30 7.31 7.32	Ukázka query metody pro dotazování se nad stránkovaným obsahem	18 18 19 19 19

7.34	Parsování datového souboru omluvy.unl	21
7.35	Parsování datového souboru omluvy.unl	21
7.36	Transformace objektu Omluva na databázový objekt Excuse	21
A.1	Tělo odpovědi pro dotaz GET /api/app	3
A.2	Tělo odpovědi pro dotaz GET /api/vote	4
A.3	Tělo odpovědi pro dotaz dGET /api/votei	4
A.4	Tělo odpovědi pro dotaz GET /api/party/vote/1	4
A.5	Tělo odpovědi pro dotaz "	5
A.6	Tělo odpovědi pro dotaz GET /api/member/1	6
A.7	Tělo odpovědí pro dotaz GET /api/member/1/vote	6

Rád bych tímto poděkoval svému vedoucímu, Ing. Ondřej John, za jeho vstřícnost, trpělivost a čas, který mi věnoval při vedení mé diplomové práce. Dále bych chtěl poděkovat své rodině, která mě při psaní podporovala.

	10.0	1/	_
Pr	Oh.	láše	mı
	OII.	last	

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.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen "Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 6. února 2023	
V Praze dne 6. února 2023	

Abstrakt

Diplomová práce popisuje návrh a implementaci mobilní aplikace, která slouží k zobrazení výsledků hlasování poslanců Poslanecké sněmovny Parlementu ČR. Součástí práce je i návrh a implementace REST API, který bude poskytovat data pro mobilní aplikaci. V teoretické části specifikuji požadavky po aplikaci a backendu, analyzuji již existující či podobná řešení a popisuji potřebné koncepty pro pochopení praktické části. V praktické části pak popisuji implementaci mobilní aplikace v jazyce Kotlin a REST API v jazyce Java.

Klíčová slova poslanecká sněmovna, poslanecká sněmovna parlamentu ČR, hlasování, poslanec, REST, backend

Abstract

Keywords

Shrnutí

Poslanecká sněmovna

Analýza existujících řešení

V první kapitole analyzuji existující řešení, která řeší stejný nebo podobný problémm.

Funkční a nefunkční požadavky

V této kapitole popisuji funkční a nefunkční požadavky kladené na mobilní aplikaci a backendu.

Analýza

V této kapitole analyzuji aplikace, které se také zabývají agregací a zobrazování výsledků hlasování. Konkrétně se podíváme na mobilní aplikace politiscope, Election Tracker - US election a Election Polls US. Budu analyzovat především zobrazovaná data a uživatelské rozhraní. V další části analyzuji zdrojová data, které poslanecka sněmovna poskytuje na jejich oficiálním portálu www.psp.cz.

Návrh

Na základě funkčních a nefunkčních požadavků a analýzy zdrojových dat navrhnu v této kapitole uživatelské rozhraní mobilní aplikaci. Součástí

návrhu jsou wireframy. Následně popíši návrh REST API pro poskytování dat pro mobilní aplikaci. REST API bude navrženo tak, aby co nejlépě vyhovělo požadavkům mobilní aplikace a jejímu uživatelskému rozhraní. Na konci popíši návrh datové struktury backend služby, tedy jaká data a v jakém formátu budou v databázi na backendu ukládána. Zároveň popíši transformaci zdrojových dat do této struktury.

Implementace

V této kapitole popisuji implementaci mobilní aplikace a backendové služby. Popíši zde strukturu a architekturu projektu, způsob implementace na základě návrhů, odůvodnění některých implementačních rozhodnuí a konkrétne využité technologie.

Testování

V této kapitole popisuji testy pro ověření korektnosti mobilní aplikace a backendové služby.

Shrnutí

V této kapitole shrnuji celou práci a ujasňuji, jaké byly na práci kladeny požadavky a které požadavky byly splněny. Uvádím možná vylepšení do budoucnosti.

Seznam zkratek

PSP Poslanecká sněmovná Parlamentu ČR

 ${\bf REST} \quad {\bf Representational \ state \ transfer}$

API Application Programming Interface

Kapitola 1

Cíle

Prvním cílem práce je analýza existujích řešení v zahraničí. Druhým cílem je specifikace funkčních a nefunkčních požadavků, které jsou kladeny na mobilní aplikaci a backendu. Dalším cílem je návrh, implementace, otestování a nasazení mobilní aplikace pro zobrazení výsledků hlasování poslanecké sněmovny pro operační systém Android. Následujícím cílem je návrh, implementace, otestování a nasazení backendu, který bude pravidelně stahovat zdrojová data a transformovat je pro vhodné použití mobilní aplikací. Posledním cílem je shrnutí práce a diskuze ohledně splnění požadavků a budoucích vylepšení.

2 Cíle

Kapitola 2

Poslanecká sněmovna

Tato kapitola slouží jako úvod do tématiky hlasovánív Poslanecké sněmovně (dále jen PS). Popíši politický systém v ČR v kontextu hlasování v PS, k čemu slouží a jak funguje hlasování v poslanecké sněmovně. Poté popíši, jakým způsobem je průběh a výsledek hlasování poskytnut veřejnosti. Nakonci uvedu motivaci k vytvoření mobilní aplikace pro sledování průběhu hlasování.

2.1 Poslanecká sněmovna

.

Základní prvky politického systému ČR představuje prezident, vláda, Parlament a ústavní soud. Ústava ČR dělí moc na zákonodárnou – Parlament, který je složen z Poslanecké sněmovny a Senátu, výkonnou – prezident, vláda a státní zastupitelství a soudní – Ústavní soud a obecné soudy. [1]

Parlament České republiky se skládá ze dvou komor – Poslanecké sněmovny (dolní komora) a Senátu (horní komora). Poslanecká sněmovna se skládá z 200 poslanců a je volena na čtyři roky na základě poměrného volebního systému.[1]

2.2 Hlasování v poslanecké sněmovně

Komora PS je usnášeníschopné, pokud je přítomna alespoň jedna třetina jejích členů. K přijetí usnesení (tzn. ke schválení zákona) je nutný souhlas nadpoloviční většiny přítomných poslanců, pokud ústava nestanoví jinak. [1]

Proces návrhu a schvalování zákona je komplexní a řídí se podle určitých pravidel. Pro účely této práce se však budu zabývat pouze schvalovacím procesem v PS. Více informací ohledně procesu přijímání zákonů lze najít na https://www.psp.cz/sqw/hp.sqw?k=173.

2.3 Webový portál psp.cz

Hlavním zdrojem pro výsledky a průběhy hlasování je oficiální webový portál psp.cz. Tento portál poskytuje mnoho informací, pro účely této práce však budu čerpat především strojově zpracovatelná data, která budou nutná pro implementaci mobilní aplikace.

2 Poslanecká sněmovna

2.4 Motivace pro tuto práci

Web PSP obsahuje veškeré informace ohledně hlasováních, nicméně není responzivní a přizpůsobený pro mobilní zařízení, a tudíž pro uživatele mobilních zařízení je web nepřehledný. Zároveň srovnatelná aplikace na českém trhu práce ještě neexistuje, a tudíž by se uživatelovi v ČR taková aplikaci mohla hodit. V neposlední řadě touto prací podporuji to, abychom měli informované voliče, zajímající se o to, jak jimi volení zástupci hlasují.

Kapitola 3

Funkční a nefunkční požadavky

V této kapitole popisuji funkční a nefunkční požadavky na mobilní aplikaci a backendu. Funkční požadavky specifikují funkcionality, které by měl daný software poskytovat. Nefunkční požadavky určují omezení kladená na daný software.

3.1 Funkční požadavky

V této podkapitole uvádím funkční požadavky pro mobilní aplikaci (3.1) a backend (3.2). Ke každému požadavku uvádím identifikátor pro pozdější odkazování k požadavku.

Funkční požadavky pro mobilní aplikaci		
ID požadavku	Popis požadavku	
FP_01	Aplikace bude umět zobrazit seznam výsledků hlasování. Kromě výsledku budou jednotlivá hlasování v seznamu obsahovat také název hlasování, a datum a čas, kdy bylo odhlasováno.	
FP_02	Aplikace bude umět zobrazit detail hlasování. Detail hlasování bude obsahovat název, datum a čas, odkaz na stenoprotokol a celkovou statistiku hlasování. Celkovou statistikou hlasování rozumíme počet hlasování pro ano, ne, nepřihlášeno, omluveno a zdrženo od hlasování. Dále bude obsahovat to, jak v daném hlasování hlasovaly jednotlivé poslanecké kluby a členy těchto klubů.	
FP_03	Aplikace bude umět zobrazit seznam členů poslanecké sněmovny. Prvky v tomto seznam budou obsahovat stručné informace o daném poslanci. Tyto informace budou obsahovat jméno a příjmení, volební kraj, název klubu a profilovou fotku.	
FP_04	Aplikace bude umět zobrazit detail poslance. Detail poslance bude obsahovat jméno a příjemní, datum narození, profilovou fotku, datum nabytí statusu poslance, poslanecký klub a volební kraj. Dále bude obsahovat seznam výsledků hlasování a to, jak v nich hlasoval daný poslanec	

FP_05	Aplikace bude poskytovat možnost nastavení volební období, při kterém se nastaví hlasování a poslanci daného volebního období.
FP_06	Aplikace bude poskytovat možnost vyhledávání hlasování podle jeho názvu.
FP_07	Aplikace bude poskytovat možnost vyhledávání poslance / poslankyně podle jeho / jejího jména.

Tabulka 3.1 Funkční požadavky pro mobilní aplikaci.

Funkční požadavky pro back-end	
ID požadavku	Popis požadavku
FP_01	Backend bude prostřednictvím volně dostupného rozhraní poskytovat mobilní aplikaci všechna data, která bude potřebovat.

■ Tabulka 3.2 Funkční požadavky pro back-end.

3.2 Nefunkční požadavky

V této podkapitole uvádím nefunkční požadavky pro mobilní aplikaci (3.3) a backend (3.4).

Nefunkční požadavky pro mobilní aplikaci			
ID požadavku Popis požadavku			
NP_00	Aplikace nebude provádět výpočetně náročná zpracování dat z důvodu šetření aplikace. Toto zpracování bude delegováno na backend.		
NP_01	Aplikace bude data stahovat pouze po spuštění aplikace z důvodu úspory dat a šetření BE.		
NP_02	Aplikace bude podporovat pouze čas v ČR, jelikož cílí na hlasování o zákonech v ČR. Tj. pokud používáme aplikaci v zahraničí, zobrazené časy budou stále lokální vzhledem k ČR.		
NP_03	Aplikace bude mít jednoduché a intuitivní uživatelské rozhraní.		
NP_04	Aplikace bude fungovat na zařízeních s OS Android 5.1 a výš pro cílení většího množství uživatelů.		
NP_05	Aplikace nebude sbírat uživatelská data.		
NP_06	Aplikace bude používat architekturu do- poručenou v oficiální dokumenteci Androidu (https://developer.android.com/topic/architecture).		

NP_07	Backend bude zdrojová data zpracovávat tak, aby se nemusely zpracovávat v mobilní aplikaci. Zpracování dat na straně klienta by mohlo aplikaci zpomalit. Část dat, jejichž zpracování bude trvat příliš dlouho (např. půl dne), lze zpracovávat za běhu, pokud to nebude trvat příliš dlouho, a tím zhoršovat user experience uživatele. Rozhodnutí o to, kdy budou data zpracvávána při startu backendu a kdy až za běhu, je na pocitu programátora user testování mobilní
	aplikace.

Tabulka 3.3 Nefunkční požadavky pro mobilní aplikaci.

Nefunkční požadavky pro back-end		
ID požadavku	Popis požadavku	
NP_01	Backend bude data stahovat z oficiální portálu PSP.	
NP_02	Backend bude data stažená z portálu PSP transformovat do databázového modelu, který bude popsán v kapitole o návrhu aplikace a backendu. Cílem je, aby data byla předzpracovaná a připravená pro rychlý přistup z mobilní aplikace.	
NP_03	Backend bude ztransformovaná data ukládat do databáze pro jejich perzistenci a připravu pro použití mobilní aplikací kdykoliv.	
NP_04	Backend bude ztransformovaná a uložena data vystavovat prostřednictvím REST API.	
NP_03	Backend bude každý den stahovat nová data z portálu PSP a aktualizovat databázi.	
NP_05	Backend bude data vystavovat ve formatu JSON.	

Tabulka 3.4 Nefunkční požadavky pro back-end.

Kapitola 4

Analýza existujících řešení

V rámci této kapitoly provádím rešerši podobných řešení s cílem zjistit jejich funkčnosti a UI.

4.1 politiscope

Autor: Android Politiscope Developer

Počet stažení: více než 10 000

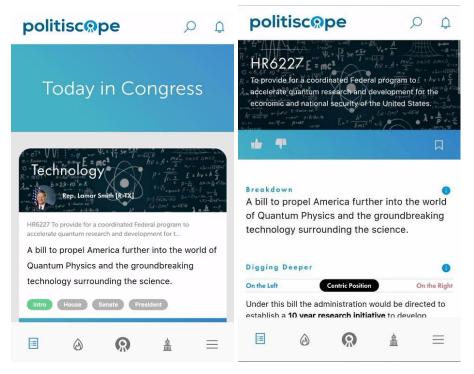
Analyzovaná verze: 2.4 (26. 1., 2023)

Aplikace politiscope [2] dle popisu na Google Play poskytuje informace ohledně politiky ve Spojených Státech v zjednodušené formě. Aplikace poskytuje informace i politicích a jejich rozhodnutích v hlasováních. Informace jsou podávány jednodušší formou, přičemž se snaží udržet objektivitu podaných informací. Aktuální témata jsou barevně označena pro lepší UX. Uživatelé mají možnost uložit si návrh zákona a sledovat průběh hlasování. Uživatelé mají také možnost sledovat konkrétní politiky. Návrhy zákonů jsou označeny tagy pro snazší vyhledání. U témat jsou i oficiální sumarizace a odkazy na oficiální zdroje. Lze také sledovat průběh voleb a kampaně.

Aplikace čerpá data z API poskytuných z následujícíh portálů:

- https://api.propublica.org/ ProPublica je nezávislá, nezisková redakce. [3]
- https://theunitedstates.io/ @unitedstates je projekt poskytující data ohledně Spojených Států veřejností a pro veřejnost. [4]
- https://www.congress.gov/ Congress.gov je oficiální portál pro informace z Kongresu a orgánů státní správy. [5]
- https://api.open.fec.gov/ OpenFEC je oficiální portál vlády Spojených Států. [6]

Výše uvedené informace jsou čerpány čistě z popisu a screenshotů aplikace na Google Playi. Do aplikace se mi nepodařilo dostat. Pro přístup je potřeba se zaregistrovat a přihlásit se. Při registraci mě to však automaticky přesměruje na obrazovka pro přihlášování. Při zadání přihlašovacích údaju to však píše, že účet se zadanými přihlašovacími údaji neexistují. Aplikaci jsem testoval na dvou různých zařízeních a na obou nastal ten samý problém. Aplikace má přesto přes 10 000 stažení, a tudíž ve většině případech funguje. Tipuji, že problém souvisí s geografickou lokací mobilního zařízení.



Obrázek 4.1 Android aplikace politiscope

4.1.1 Zhodnocení

Přestože aplikaci se mi nepodařilo zprovoznit, stálo za podle mého názoru ji sem dát kvůli jejímu rozsáhlému výčtu funkčností. Další výhodou této aplikace je také přívětivé uživatelské rozhraní a fakt, že data získává z API. Poslanecká sněmovna, jak později ukážu, API pro svoje data neposkytuje, ale data poskytuje ve formě CSV souborů.

4.2 Congress

Autor: Eric Mill

Počet stažení: více než 500 000

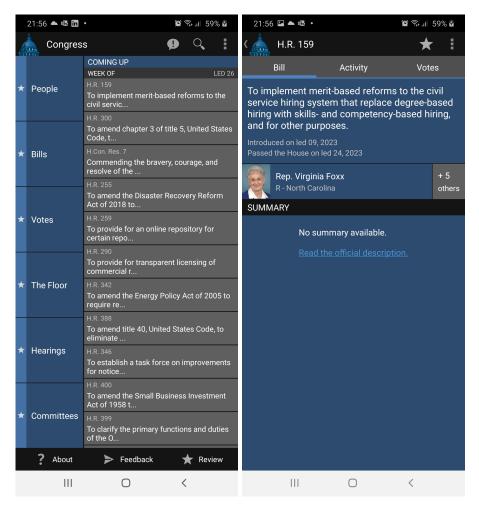
Analyzovaná verze: 4.9.2 (27. 1., 2023)

Aplikace Congress [5] dle popisu na Google Play poskytuje informace ohledně politických reprezentantů a jejich hlasováních, a návrhů zákonů ve Spojených Státech. Návrhy a hlasování umožňuje vyhledávat.

Při spuštění aplikace uvidíme domovskou obrazovku, která obsahuje menu pro seznam politiků, návrhů zákonů, výsledků hlasování, aktivity v kongresu, schůzky komisí a seznam komisí. Na domovské stránce uvidíme také seznam nejnovějších návrhů zákonů.

Obrazovka pro seznam politiků obsahuje seznam politiků, které aktuálně sledujeme, seznam politických reprezentantů rozdělených podle států, sněmovny a senátu. Na obrazovce konkrétního politického reprezentanta uvidíme jméno, politickou stranu, příslušný stát, telefonní číslo, jak hlasoval, které zákony navrhnul, ke kterým komisím přísluší, odkaz na oficiální stránku s informacemi o něm a jeho biografii.

Congress 3



Obrázek 4.2 Android aplikace politiscope

Obrazovka pro návrhy zákonů obsahuje seznam návrhů, které sledujeme, seznam aktivních návrhů a seznam nových návrhů.

Na obrazovce pro výsledky hlasování je popis, výsledek datum a čas, a údaj o tomm, zda se hlasovalo ve Sněmovně nebo Senátu. Obrazovka s detailem hlasování obsahuje výsledek hlasování, počet hlasování pro a proti, a počet lidí, kteří nehlasovali. Dále obsahuje informace o tom, kolik lidí je potřeba být ve fyzické přítomnosti, aby hlasování bylo platné, a jak hlasoval který politik.

Obrazovka pro události v kongresu obsahuje seznam událostí seřazené sestupně podle data času. Události jsou rozdělené podle toho, zda nastaly ve Sněmovně nebo v Senátu.

Obrazovka pro schůzky komisí byla v době analýzy aplikace prázdná. Obrazovka pro seznam komisí byla v době analýzy aplikace prázdná. Nejspíš proto je i obrazovka pro schůzky komisí prázdná.

4.2.1 Zhodnocení

Můj první dojem z této aplikace je to, že je velmi propracovaná z hlediska různorodosti informací, které poskytuje. Přitom díky dobře navrženému uživatelskému rozhraní nepůsobí nepřehledně. Naopak působí velmi intuitivně. Z menu se dostaneme na hlavní obrazovky, které jsou dále rozděleny na taby. U návrhů můžeme snadno vidět výsledek, jak kdo hlasoval, proces schvalování

návrhu a aktuální stav. Politici můžeme snadno vyhledat podle klíčových slov, státu a příslušnosti ve Sněmovně nebo Senátu. Politikz a návrhy zákonu můžeme sledovat a nastavit si notifikaci, takže budeme vždy notifikovani o nových změnách.

Kapitola 5

Analýza zdrojových dat

5.1 Zdroj

Zdrojová data PS jsou volně ke stažení na https://www.psp.cz/sqw/hp.sqw?k=1300. Data jsou strukturovaná a pochází z agend PS a Senátu jako např. agenda poslanců, osob, hlasování a tisků. Pro účely této práce nás však budou zajímat pouze podmnožina dat agend z PS, které popíši později.

5.2 Formát dat

Data jsou poskytována v souborech ve formátu UNL, tj.:

- Každý řádek v souboru odpovídá jednom řádku v databázi.
- Oddělovačem je znak roury (—).
- Pokud je sloupec prázdný, je jeho hodnota typu null.
- V sloupcích jsou používány tzv. escape sekvence k zápisu speciálních znaků s úvodním znakem (backslash) následovaný znakem.

Tyto soubory jsou podle typu seskupeny do souborů ve formátu zip, např. poslanci.zip pro data o poslancích a hl-2021ps.zip pro data o hlasováních v 9. volebním období.

5.3 Aktualizace

Data obsahují úplný stav, rozdílové aktualizace nejsou poskytovány. To pro nás znamená, že při aktualizaci dat musíme rozdíly mezi zdrojovými daty a daty v databázi najít sami a podle toho aktualizovat databázi. Důležité při tom je to, aby data, která na sobě závisí, byla aktualizována tak, aby byla zaručena jejich konzistence. Tedy pokud při aktualizaci nějakého údaje musíme aktualizovat i všechny údaje, které na tom údaji závisí.

Pokud bude strunktura dat doplňována, budou nové sloupce přidávány na konec. Nové sloupce pro nás nebudou důležitá. Budeme pracovat pouze s daty, které tam jsou v době psaní diplomové práce.

5.4 Kódování

Kódování je windows-1250. Ten obsahuje mimo jiné všechny znaky z české abecedy. Na to bude potřeba brát ohled při ukládání dat do databáze, aby se toto kódování zachovalo.

5.5 Datové typy

Na stránce je uvedena tabulka obsahující typy dat sloupců v tabulkách a popis jejich významu.

Typy dat sloupců v tabulkách		
Тур	Popis	
int	integer	
char(X)	textový řetězec, s blíže neuvedenou délkou	
char(N)	textový řetězec, s konktrétní délkou	
date	datum, ve formátu DD.MM.YYYY	
datetime(year to hour)	datum a čas, do úrovně hodin, ve formátu YYYY-MM-DD HH	
datetime(year to second)	datum a čas, do úrovně vteřin, ve formátu YYYY-MM-DD HH:TT:SS	
datetime(, fraction)	Doplnění formátu o zlomky vteřiny, odděleno tečkou od původního formátu	
datetime(hour to minute)	čas, ve formátu HH:MM	

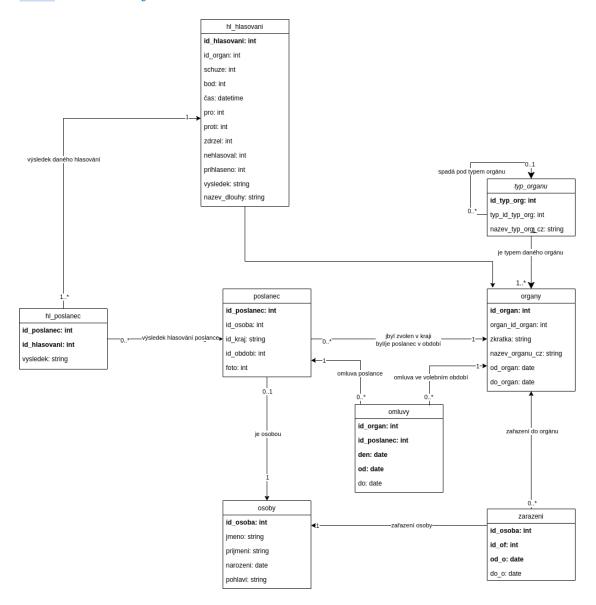
■ Tabulka 5.1 Typy dat sloupců v tabulkách

5.6 Licence

Data jsou poskytována bezplatně, využití dat je podmíněno uvedením zdroje dat a případně datem zpracování dat. Mobilní aplikace a backend budou implementovány ve dvou různých repozitářích. V každém z nich uvedeno, odkud data pocházela.

Tabulky 3

5.7 Tabulky



5.7.1 typ_organu

Orgány mají svůj typ, tyto typy mají hiearchickou strukturu.

■ Tabulka 5.2 Tabulka typ_organu

Tabulka typ_organu		
Sloupec	Тур	Použití a vazby
id_typ_org	int	Identifikátor typu orgánu

■ Tabulka 5.2 Tabulka typ_organu

Tabulka typ₋organu			
Sloupec	Тур	Použití a vazby	
typ_id_typ_org org	int	Identifikátor nadřazeného typu orgánu (typ_organu:id_typ_org), pokud je null či nevyplněno, pak nemá nadřazený typ	
nazev_typ_org_cz	char(X)	Název typu orgánu v češtině	
nazev_typ_org_en	char(X)	Název typu orgánu v angličtině	
typ_org_obecny	int	Obecný typ orgánu, pokud je vyplněný, odpovídá záznamu v typ_organu:id_typ_org. Pomocí tohoto sloupce lze najít např. všechny výbory v různých typech zastupitelských sborů.	
priorita	int	Priorita při výpisu	

5.7.2 organy

Některé orgány mají nadřazený orgán a pak je položka organy:organ_id_organ vyplněna, přičemž pouze v některých případech se tyto vazby využívají.

■ Tabulka 5.3 Tabulka organy

Tabulka organy			
Sloupec	Тур	Použití a vazby	
id_organ	int	Identifikátor orgánu	
organ_id_organ	int	Identifikátor nadřazeného orgánu, viz organy:id_organ	
id_typ_organu	int	Typ orgánu, viz typ_organu:id_typ_organu	
zkratka	char(X)	Zkratka orgánu, bez diakritiky, v některých připadech se zkratka při zobrazení nahrazuje jiným názvem	
nazev_organu_cz	char(X)	Název orgánu v češtině	
nazev_organu_en	char(X)	Název orgánu v angličtině	
od_organ	date	Ustavení orgánu	
do_organ	date	Ukončení orgánu	
priorita	int	Priorita výpisu orgánů	
cl_organ_base	int Pokud je nastaveno na 1, pak při výpisu členů se nezobrazují záznamy v tabulkce zarazeni kde cl_funkce == 0. Toto chování odpovídá tomu, že v některých orgánech nejsou členové a teprve z nich se volí funkcionáři, ale přímo se vol do určité funkce.		

Tabulky 5

5.7.3 osoby

Obsahuje jména osob, které jsou zařazeni v orgánech. Vzhledem k tomu, že k jednoznačnému rozlišení osob často není dostatek informací, je možné, že ne všechny záznamy odkazují na jedinečné osoby, tj. některé osoby jsou v tabulce vícekrát.

■ Tabulka 5.4 Tabulka osoby

Tabulka	Tabulka osoby		
Sloupec	Тур	Použití a vazby	
id_osoba	int	Identifikátor osoby	
pred	char(X)	Titul pred jmenem	
jmeno	char(X)	Jméno	
prijmeni	char(X)	Příjmení, v některých případech obsahuje i dodatek typu "st.", "ml."	
za	char(X)	Titul za jménem	
narozeni	date	Datum narození, pokud neznámo, pak 1.1.1900.	
pohlavi	char(X)	Pohlaví, "M"jako muž, ostatní hodnoty žena	
zmena	date	Datum posledni změny	
umrti	date	Datum úmrtí	

5.7.4 zarazeni

Obsahuje data zařazení v orgánu nebo data funkcí osoby v orgánu. Pokud je zarazeni:do_o typu null, pak jde o aktuální zařazení.

■ Tabulka 5.5 Tabulka zarazeni

Tabulka zarazeni		
Sloupec	Тур	Použití a vazby
id_osoba	int	Identifikátor osoby, viz osoba:id_osoba
id_of	int	Identifikátor orgánu či funkce: pokud je zároveň nastaveno zarazeni:cl_funkce == 0, pak id_o odpovídá organy:id_organ, pokud cl_funkce == 1, pak odpovídá funkce:id_funkce.
cl_funkce	int	Status členství nebo funce: pokud je rovno 0, pak jde o členství, pokud 1, pak jde o funkci.
od_o	datetime(year to hour)	Zařazení od
do_o	datetime(year to hour)	Zařazení do

■ Tabulka 5.5 Tabulka zarazeni

Tabulka	Tabulka zarazeni		
Sloupec	Тур	Použití a vazby	
od_f	date	Mandát od. Nemusí být vyplněno a pokud je vyplněno, pak určuje datum vzniku mandátu a zarazeni:od_o obsahuje datum volby.	
do_f	date	Mandát do. Nemusí být vyplněno a pokud je vyplněno, určuje datum konce mandátu a zarazeni:do_o obsahuje datum ukončení zařazení.	

5.7.5 poslanec

■ Tabulka 5.6 Tabulka poslanec

Tabulka poslanec			
Sloupec	Тур	Použití a vazby	
id_poslanec	int	Identifikátor poslance	
id_osoba	int	Identifikátor osoby, viz osoba:id_osoba	
id_kraj	int	Volební kraj, viz organy:id_organu	
id_kandidatka	int	Volební strana/hnutí, viz org:id_organu, pouze odkazuje na stranu/hnutí, za kterou byl zvolen a nemusí mít souvislost s členstvím v poslaneckém klubu.	
id_obdobi	int	Volební období, viz organy:id_organu	
web	char(X)	URL vlastních stránek poslance	
ulice	char(X)	Adresa regionální kanceláře, ulice.	
obec	char(X)	Adresa regionální kanceláře, obec.	
psc	char(X)	Adresa regionální kanceláře, PSČ.	
email	char(X)	E-mailová adresa poslance, případně obecná posta@psp.cz.	
telefon	char(X)	Adresa regionální kanceláře, telefon.	
fax	char(X)	Adresa regionální kanceláře, fax.	
psp_telefon	char(X)	Telefonní číslo do kanceláře v budovách PS.	
facebook	char(X)	URL stránky služby Facebook.	
foto	int	Pokud je rovno 1, pak existuje fotografie poslance.	

Tabulky 7

5.7.6 hl_hlasovani

■ Tabulka 5.7 Tabulka hl_hlasovani

Tabulka hl_hlasovani			
Sloupec	Тур	Použití a vazby	
id_hlasovani	int	Identifikátor hlasování	
id₋organ	int	Identifikátor orgánu, viz organy:id_organ	
schuze	int	Číslo schůze	
cislo	int	Číslo hlasování	
bod	int	Bod pořadu schůze; je-li menší než 1, pak jde o procedurální hlasování nebo o hlasování k bodům, které v době hlasování neměly přiděleno číslo.	
datum	date	Datum hlasování	
čas	datetime(hour to minute)	Čas hlasování	
pro	int	Počet hlasujících pro	
proti	int	Počet hlasujících proti	
zdrzel	int	Počet hlasujících zdržel se, tj. stiskl tlačítko X	
nehlasoval	int	Počet přihlášených, kteří nestiskli žádné tlačítko	
prihlaseno	int	Počet přihlášených poslanců	
kvorum	int	Kvórum, nejmenší počet hlasů k přijetí návrhu	
druh_hlasovani	char(X)	Druh hlasování: N - normální, R - ruční (nejsou známy hlasování jednotlivých poslanců), E - vinou technické závady nejsou dostupná všechna data k hlasování, např. výsledky hlasování jednotlivých poslanců.	
vysledek	char(X)	Výsledek: A - přijato, R - zamítnuto, jinak zmatečné hlasování	
nazev_dlouhy	char(X)	Dlouhý název bodu hlasování	
nazev_kratky	char(X)	Krátký název bodu hlasování	

5.7.7 hl_poslanec

Tabulka zaznamenává výsledek hlasování jednotlivého poslance.

■ Tabulka 5.8 Tabulka hl_poslanec

Tabulka hl_poslanec		
Sloupec	Тур	Použití a vazby
id_poslanec	int	Identifikátor poslance, viz poslanec:id_poslanec
id_hlasovani	int	Identifikátor hlasování, viz hl_hlasovani:id_hlasovani
vysledek	char(X)	Hlasování jednotlivého poslance. 'A' - ano, 'B' nebo 'N' - ne, 'C' - zdržel se (stiskl tlačítko X), 'F' - nehlasoval (byl přihlášen, ale nestiskl žádné tlačítko), '@' - nepřihlášen, 'M' - omluven, 'W' - hlasování před složením slibu poslance, 'K' - zdržel se/nehlasoval. Viz úvodní vysvětlení zpracování výsledků hlasování.

5.7.8 omluvy

Tabulka zaznamenává časové ohraničení omluv poslanců z jednání Poslanecké sněmovny.

■ Tabulka 5.9 Tabulka omluvy

Tabulka omluvy		
Sloupec	Тур	Použití a vazby
id_organ	int	Identifikátor volebního období, viz organy:id_organ
id_poslanec	int	Identifikátor poslance, viz poslanec:id_poslanec
den	date	Datum omluvy
od	datetime(hour to minute)	Čas začátku omluvy, pokud je null, pak i omluvy:do je null a jedná se o omluvu na celý jednací den.
do	datetime(hour to minute)	Čas konce omluvy, pokud je null, pak i omluvy:od je null a jedná se o omluvu na celý jednací den.

Kapitola 6

Návrh

6.1 Uživatelské rozhraní

Seznam hlasování

Na obrázku (A.1) je návrh obrazovky pro seznam hlasování. Ta je složena z hlavičky, seznamu hlasování a dolní navigace. Hlavička obsahuje titul identifikující danou obrazovku, aktuálně nastavené volební období a tlačítko pro vyhledávání hlasování. Titul slouží pro snazší orientaci v aplikaci. Volební období slouží pro snazší kontrolu, jaké volební období je aktuálně nastaveno. Jednotlivá hlasování v seznamu jsou rozdělena do jednotlivých boxů. Každý box obsahuje krátký popis návrhu zákona, datum a čas hlasování, výsledek hlasování znázorněný ikonkou a textem, a indikátor pro kliknutí na dané hlasování. Dolní navigace pak slouží pro navigaci mezi hlavními obrazovkami, tj. mezi obrazovkou pro seznam hlasování, seznam poslanců a nastavení.

Kliknutím na tlačítko pro vyhledávání se zobrazí vyhledávací pole (A.2), které slouží pro vyhledávání seznamu hlasování podle jejich popisu. Do pole uživatel zadává klíčová slova. Pole obsahuje placeholder text, tlačítko pro smazání textu a tlačítko pro schování vyhledávacího pole.

Detail hlasování

Obrazovka pro detail hlasování (A.5) je složena z hlavičky a obsahu. Hlavička obsahuje tlačítko pro navigaci zpět. Obsah je rozdělen do dvou tabů, mezi kterými lze navigovat pomocí dvou tlačítek pod hlavičkou. V prvním tabu se nachází obecné údaje o daném návrhu zákona a výsledích jeho hlasování. Tyto údaje zahrnují popis návrhu zákona, datum a čas hlasování, odkaz na oficiální portál se stenoprotokolem a tabulku s informacemi ohledně toho, jak se hlasovalo. Každý typ hlasování (ano, ne, nepřihlášen, omluven, zdržel se) je popsáno textově i pomocí ikonky.

Druhý tab (A.5) obsahuje seznam poslaneckých klubů v daném volebním období, rozdělených do boxů. V každém boxu je název klubu, jeho logo, pokud je k dispozici, a indikátor pro expandování boxu pro zobrazení informací o tom, jak daný klub a jeho členové hlasovali (A.5). Expandovaný box obsahuje navíc tabulku se statistikou hlasování jako v prvním tabu, ale pro konkrétní poslanecký klub. Pod tabulkou je seznam členů klubu a to, jak pro daný návrh zákona hlasovali. Výsledek hlasování členů je znázorněno ikonkou.

2 Návrh

Seznam poslanců

Obrazovka pro seznam poslanců (A.9) vypadá podobně jako obrazovka pro seznam hlasování. Obsahuje hlavičku s titulem, aktuálně nastaveným volebním obdobím a tlačítkém pro vyhledávání, seznam poslanců a dolní navigaci. Poslanci jsou rozděleny do boxů, které obsahují profilovou fotku poslance, jméno a příjmení, volební kraj, poslanecký klub a indikátor pro kliknutí. Kliknutím se dostaneme na obrazovku s detailem daného poslance.

Pomocí vyhledávacího pole (A.9) lze poslance filtrovat podle jejich jména a příjmení.

Detail poslance

Na obrazovce s detailem poslance (A.11) můžeme vidět údaje o daném poslanci a informace o tom, jak hlasoval pro návrhy zákonů. Obrazovka je rozdělena na hlavičku a obsah. Hlavička obsahuje tlačítko pro navigaci zpět. Obsah je rozdělen do dvou tabů. První z tabů obsahuje údaje o daném poslanci, tj. profilovou fotku, jméno a příjmení, datum a narození, datum mandátu poslance, poslanecký klub, do kterého přísluší, a volební kraj. Druhý tab (A.12) obsahuje údaje o tom, jak daný poslanec hlasoval pro jednotlivé návrhy zákonů. Návrh je podobný jako návrh seznamu hlasování, obsahuje navíc údaj o tom, jak hlasoval daný poslanec.

Nastavení

Na obrazovce pro nastavení (A.15) je seznam nastavení dané aplikace. V době psaní této práce obsahuje pouze jedno možné nastavení, kterým je volební období. Nastavení obsahuje ikonku znázorňující typ nastavení (v tomto případě volební období), název nastavení a text nastaveného volebního obdobi. Kliknutím na toto nastavení naskočí okno (A.15) se seznamem volebních období. Po zvolení volebního období uživatel může kliknout na tlačítko Uložit, kterým se dané volební období nastaví, nebo Zrušit, čímž se zruší aktuální výběr v seznamu.

6.2 REST API

Mobilní aplikace komunikuje s backendem pomocí REST API. Tato kapitola popisuje endpointy této API, její vstupy a výstupy.

HTTP hlavička

- prev Odkaz na předchozí stránku. Null, pokud aktuální stránka je první.
- next Odkaz na následující stránku. Null, pokud aktuální stránka je poslední.
- last Odkaz na poslední stránku.
- self Odkaz na aktuální stránku.

Endpointy

GET /api/app

Vrací následující informace o stavu aplikace (A.1):

Volební roky

REST API 3

GET /api/vote

Vrací seznam hlasování s následujícími informace (A.2):

- identifikátor hlasování
- datum a čas hlasování
- popis hlasování
- výsledek hlasování (A přijato, R zamítnuto, jinak zmatečné hlasování)

GET /api/vote{id}

Vrací následující informace o detailu hlasování (A.3):

- identifikátor hlasování
- datum a čas hlasování
- popis hlasování
- výsledek hlasování (A přijato, R zamítnuto, jinak zmatečné hlasování)
- URL odkaz na příslušný stenoprotokol
- počet hlasování pro
- počet hlasování proti
- počet nepřihlášených
- počet omluvených
- počet zdržených
- volební rok

GET /api/party/vote/{id}

Vrací následující informace o hlasováních poslaneckých klubů v daném hlasování (A.4):

- název klubu
- URL odkaz na logo klubu
- identifikátor hlasování
- výsledky hlasování klubu
- výsledky hlasování členů klubu

GET /api/member

Vrací seznam poslanců s následujícími informacemi:

identifikátor

4 Návrh

- jméno a příjmení
- poslanecký klub
- URL odkaz na profilovou fotku
- volební kraj
- volební rok

GET /api/member/{id}

Vrací následující informace o detailu poslance:

- identifikátor
- jméno a příjmení
- pohlaví (M muž, Ž ostatní)
- poslanecký klub
- začátek mandátu
- konec mandátu
- datum narození
- volební kraj
- URL odkaz na profilovou fotku, pokud existuje, jinak null
- volební rok

GET /api/member/1/vote

Vrací následující informace o hlasováních daného poslance:

- obecné informace o hlasování
- \blacksquare výsledek hlasování poslance (A ano, B nebo N ne, C zdržel se, F nehlasoval, @ nepřihlášen, K zdržel se / nehlasoval)

6.3 Databázové datové struktury

Struktura agency.

Struktura agency_type.

Struktura excuse.

Struktura member.

Struktura member_vote.

Struktura membership.

Struktura party.

Struktura vote.

Kapitola 7

Implementace

Tato kapitola se zabývá implementací mobilní aplikace a backendu. Kapitola je rozdělena na dvě podkapitoly: jednu pro popis implementace mobilní aplikace a druhou pro popis implementace backendu.

.

7.1 Mobilní aplikace

V rámci této podkapitoly bude popsána implementace mobilní aplikace. Kapitola je rozdělena do několika sekcí. Na začátku budou popsány nástroje a technologie použité pro implementaci. Následné bude popsána adresářová struktura projektu. Poté bude popsána implementace uživatelského rozhraní. Na konci bude popsána implementace síťové vrstvy pro komunikaci s backendem.

7.1.1 Použité nástroje a technologie

V této sekci budou popsány použité nástroje a technologie, včetně použitých knihoven pro implementaci mobilní aplikace.

Android Studio

Mobilní aplikace byla vytvořena ve vývojovém prostředí Android Studio, což je oficiální IDE pro vývoj mobilních aplikací pro Android. Některé jeho vlastnosti:

- Podpora pro Android Pro vývoj mobilních aplikací pro Android je Android Studio velmi uživatelsky přívětivé. Po spuštění IDE se hned zobrazí průvodce pro instalaci Android SDK obsahující nástroje potřebné pro vývoj aplikací. Po instalaci lze SDK rovnou použít. Není tedy třeba Android SDK manuálně instalovat z internetu a nakonfigurovat s IDE. IDE Dále poskytuje průvodce pro vytvoření projektu, pomocí kterého lze vybrat jednu z existujích šablon pro různé typy projektů. Konfiguraci projektu si uživatel tedy nakliká a IDE se postará o zbytek, tedy vytvoří projektový adresář se všemi potřebnými konfiguračními soubory a počáteční kostru aplikace. Po vytvoření projektu je již k dispozici zatím prázdná, ale spustitelná aplikace.
- Emulátor Android Studio poskytuje vestavěný emulátor, který emuluje fyzicka mobilní zařízení. Díky tomu může uživatel testovat své aplikace na různých zařízeních s různými konfiguracemi (např. rozměry zařízení, verze Androidu).

■ Klávesové zkratky - Jeden z hlavních důvodů pro používání IDE založených od JetBrains jsou klávesové zkratky, které jsou stejné pro všechna IDE od JetBrains. Tyto zkratky zrychlují navigaci v kódu a zvyšují produktivitu vývojáře.

zdroj: https://developer.android.com/studio

Gradle

Gradle je nástroj pro automatizaci sestavování programu, tj. automatizace kompilace zdrojového kódu a zabalení výsledného binárního kódu spolu s dalšími zdroji, které zdrojový kód využívá. Toto jsou další výhody Gradlu:

- Gradle Pluginy Poskytují Gradlu další nástroje jako např. možnost kompilovat Kotlinu, parsovat anotace nebo generovat kód na základě konfiguračního souboru.
- **Externí knihovny** Pomocí Gradlu lze specifikovat externí knihovny, které se mají automaticky nainstalovat a naimportovat do aplikace pro použití.
- Konfigurace Gradle umožňuje konfigurovat pluginy a Android. konfigurace pluginu závisí na daném pluginu. U Androidu lze konfigurovat např. SDK verzi, verzi Kotlinu, aktivaci Jetpack Compose toolkitu. Konfigurací se detailněji zabývá podkapitola o konfiguraci Androidu.

zdroj: https://gradle.org/

Alternativou nástroje Gradle je Maven, který lze také použít pro vývoj mobilní aplikace pro Android. Oba mají své výhody a nevýhody pro různé situace. Rozhodl jsem se však pro Gradle, jelikož při vytváření projektu v Android Studiu nebyla možnost výběru mezi Gradlem a Mavenem. Projekt byl automaticky nakonfigurován pomocí Gradlu. Předpokládám tedy, že Google preferuje Gradle jako nástroj pro tento účel. Maven by se musel nakonfigurovat manuálně, což by pro zprovoznění aplikace bylo časově náročné, a nejspíš i zbytečné. Pokud by však byla jednoduchá možnost, jak zvolit Maven jako nástroj, přesto bych zvolil Gradle, jelikož konfigurace v Gradlu je psána v programovacím jazyce Groovy, který je pro mě mnohem čitelnější než XML, v kterém je psána konfigurace pro Maven.

Kotlin

Pro mobilní aplikaci byl použit programovací jayzk Kotlin, který je od roku 2017 preferovaným jazykem pro Android. Původním programovacím jayzkem pro Android byla Java. Kotlin má však oproti němu několik výhod. Toto jsou výhody Kotlinu:

- Je stručný Kotlin umožňuje vytvořit singleton pomocí klíčového slova object.
- Je bezpečný Kotlin rozlišuje null a non-null datové typy. Non-null typy lze dereferencovat vždy, null typy pouze po kontrole výskytu hodnoty null. To je vynuceno typovým systémem Kotlinu. Díky tomu není možné zkompilovat kód, v kterém by se dereferencovala hodnota null, kvůli čemuž by aplikace spadla.
- Je expresivní Kotlin byl navržen s důrazem na výstižnost kódu, což přispívá k čitelnosti kódu

Dalším důvodem pro použití Kotlinu souvisí s dlouze běžícími blokujícími operacemi jako např. síťovými a databázovými operacemi. Aplikace běží defaultně na hlavním vláknu, které má na starosti vykreslování obrazovky a obsluhu událostí (např. kliknutí na tlačítko a scrollování). Pokud na něm provedeme dlouze běžící blokující operaci, vlákno se zablokuje na delší dobu a nebude moct obsluhovat události. Uživatelovi se pak aplikace jeví jako zamrznutá. Možným řešením pro tento problém je vytvoření nového vlákna, které bude běžet paralelně s hlavním

Mobilní aplikace 3

vlákenm a které bude provádět danou blokující operaci. Operace bude blokovat nově vytvořené vlákno, nebude však blokovat hlavní vlákno, díky čemuž bude uživatel stále moct interagovat s aplikací, i když blokující operace stále běží. Tvorba vláken a jejich správa jsou však drahé operace.

Kotlin Coroutines - Alternativním řešením jsou proto Kotlin Coroutines, které umožňují provádět operace paralelně bez nutnosti vytvoření nového vlákna. Coroutina je kus kódu, který je suspendovatelný. To znamená, že ho lze pomocí určitých metod pozastavit (také suspendovat), dokud nebude operace hotová, a tím uvolnit aktuální vlákno pro použití jinde. To se hodí např. pro síťové operace, která stahuje data z internetu. Fungovalo by to tak, že danou síťovou operaci obalíme do coroutiny a až pak operaci provedeme. Ve chvíli kdy operace začne stahovat data z internetu, coroutinu suspendujeme, čímž uvolníme aktuální vlákno pro použití jinde. Jakmile jsou data stažena, kód v suspendované coroutině automaticky pokračuje. To vše probíhá na jednom vlákně. Ukážeme si příklad použití coroutiny je na (7.1).

Výpis kódu 7.1 Příklad použití coroutiny

```
fun main() = runBlocking {
  launch {
    delay(1000L)
    println("World!")
  }
  println("Hello")
}
```

Metoda runBlocking vytvoří tzv. scope, v rámci kterého je vytvořena coroutina pomocí metody launch. Scope vymezuje životnost coroutinů, které v něm běží. To je užitečné pro zabránění úniku dat. Např. když uživatel spustí aplikaci a pomocí coroutinu se začnou stahovat data z internetu, a aplikaci hned zavře, díky scopu coroutina přestane stahovat data a je zrušena. V rámci metody launch specifikujeme funkci, která je suspendovatelná, tj. lze ji pozastavit, a tím uvolnit aktuálkní vlákno. Coroutina v tuto chvíli běží sekvenčně se zbytkem kódu, dokud není suspendována. V tomto případě je suspendována pomocí funkce delay, která ji suspenduje na 1 sekundu. V tuto chvíli je aktuální vlákno uvolněno a je proveden kód následující bezprostředně za coroutinou a vypíše se Hello. V tuto chvíli program ještě neskončí, scope totiž vždy čeká na to, až všechny její coroutiny doběžely. Po 1 sekundě suspendovaná coroutina opět pokračuje a vypíše řetězec World.

Kód (7.2) ilustruje praktičtější příklad, kdy voláme funkci fetchDataFromApi(), která stahuje data z API a interně operaci suspenduje, díky čemuž se uvolní vlákno a funkce handleUserInteraction() pro interakci s uživatelem může být spuštěna ještě před tím, než přijdou data. Až přijdou data, coroutina opět poběží a na základě získaných dat aktualizuje UI pomocí funkce updateUi(). Klíčové slovo suspend u deklarace funkce znamená, že je suspendovatelná a lze z ní volat funkce, které jsou také suspendovatelné.

Výpis kódu 7.2 Praktický příklad použití coroutiny

```
fun main() = runBlocking {
  launch {
    val data = fetchDataFromApi()
      updateUi(data)
  }
  handleUserInteraction()
}
suspend fun fetchDataFromApi() {...}
```

Výhoda coroutines oproti vláknům je tedy menší paměťová náročnost a menší zátěž na procesoru.

Kotlin Flow - Pro zvýšení čitelnosti a expresivnosti kódu a implementaci stránkování je v aplikaci používán datový typ Kotlin Flow, který reprezentuje sekvenci hodnot, které jsou produkovány a konzumovány asynchronně pomocí suspend funkcí. Příkladem použití je získání aktuálního volebního roku každou sekundu (7.3).

Výpis kódu 7.3 Příklad použití flow

```
fun fetchData(): Flow<Int> = flow {
   while(true) {
     val year = fetchLatestElectionYear()
     emit(year)
     delay(1000)
   }
}
fun main() = {
   fetchData().collect { data -> println(data) }
}
```

Funkce flow vytvoří flow, který v nekonečném cyklu získá data, emitne je a počká 1 s, než postup zoopakuje. Vidíme tedy, že funkce emit vrací data, ale funkce pokračuje dál. Funkce skončí až tehdy, kdy doběhne na konec. Funkce vrací flow, v tuto chvíli však kód dané flow ještě neběží. Až po zavolání funkce collect začne flow provádět svůj kód a emitovat data. Takovým funkcím, které spustí kód ve flow říkáme terminální operátory. Terminální operátor konzumuje všechny hodnoty emitované danou flow, s kterými lze dále pracovat. Výhodou kotlin flow je tedy jednoduchá implementace pro zpracovávání potenciálně nekonečných streamů dat, které jsou produkovány a konzumovány asynchronním způsobem pomocí coroutinů.

Další výhodou kotlin flowů je možnost použití tzv. intermediate operátorů, které s hodnoty emitované flowem nějakým způsobem manipulují. Mohou je např. filtrovat, mapovat na jiné hodnoty, kešovat je a další. V (7.4) voláme funkci, getMembers(), která vrací flow Votů. Na ní se zavolá intermediate operátor filter, který z hodnot emitovaných danou flow vyfiltruje takové voty, jejichý popis začíná písmenem A. V tuto chvíli flow ještě neběží. Následně se zavolá terminální operátor collect, který flow spustí a zkonzumuje.

Kotlin flows se používají také pro implementaci stránkování, které bude poposáno v sekci Implementace uživatelského rozhraní.

Výpis kódu 7.4 Příklad intermediate operátoru pro flow

```
fun main() = {
  getVote()
    .filter(vote -> vote.description.startsWith("A"))
    .collect { vote -> updateUi(vote)
  }
  fun getVote(): Flow<Vote> {...}
}
```

Android komponenty

Aplikace v Androidu jsou složeny z Android komponentů, které mají na starosti různé odpovědnosti:

Application - Reprezentuje celou aplikaci. V rámci práce je používaná pouze pro nakonfigurování DI pomocí Hiltu, které bude popsáno v následující sekci.

Mobilní aplikace 5

 Activity - Reprezentuje obrazovku. Je to vstupní bod pro vytvoření UI a pro počáteční inicializace.

ViewModel - Reprezentuje komponentu, která definuje business logiku a drží si stav UI elementů. Změna stavu komponenty ViewModel způsobí i změnu UI. ViewModel si pamatuje data i po konfiguračních změnách (např. otočení mobilu).

Hilt

Hilt je DI knihovna pro Android. Poskytuje kontejnery pro každou Android komponenty. Tyto kontejnery spravují objekty a jejich závislosti a jsou použity pro injektování závislostí. Příkladem injektování kódu lze vidět na (7.5). Do třídy GetAppStateUseCase injektujeme pomocí anotace @Inject implementataci rozhraní AppStateRepository.

Výpis kódu 7.5 Příklad použití DI pomocí knihovny Hilt

```
class GetAppStateUseCase @Inject constructor(
  val appStateRepository: AppStateRepository,
) {...}
interface AppStateRepository {...}
```

Pro konfiguraci závislostí, na základě kterého Hilt sestaví kontejnery, jsou používány tzv. moduly - objekty pro konfiguraci závislostí. V konfigurace (7.6) říkáme, že do proměnných typu AppStateRepository budou injektován objekt AppStateRepositoryImpl. Objekt je přiom instanciován jako singleton. Objekt zároveň potřebuje závislost na PspRemoteDataSource, který musí být opět někde nakonfigurován, aby tu mohl být injektován. Buď v tom samém modulu nebo v jiném, pokud chceme moduly pro čitelnost nějakým způsobem rozdělit.

Výpis kódu 7.6 Příklad konfigurace závislostí pro Hilt

```
@Module
@InstallIn(SingletonComponent::class)
object DataModule {

    @Provides
    fun provideAppStateRepository(
    pspRemoteDataSource: PspRemoteDataSource
    ): AppStateRepository {
       return AppStateRepositoryImpl(pspRemoteDataSource)
    }
}
```

zdroj: https://kotlinlang.org/docs/flow.html zdroj: https://kotlinlang.org/docs/object-declarations.html zdroj: https://kotlinlang.org/docs/object-declarations.html zdroj: https://kotlinlang.org/ zdroj: https://techcrunch.com/2019/05/is-now-googles-preferred-language-for-android-app-development/ zdroj: zdroj: https://www.oracle.com/java/zdroj: https://developer.android.com/kotlin/coroutine

7.1.2 Implementace uživatelského rozhraní

V této sekci bude popsána implementace uživatelského rozhraní.

Jetpack Compose

Jetpack Compose je Androidem doporučený způsob pro implementaci uživatelského rozhraní. Rozhraní je implementováno voláním tzv composable funkcí, což jsou funkce anotované anotací

@composable a reprezentují nějaký UI element na obrazovce. Jetpack Compose nám již poskytuje nativní UI elementy jako text a tlačítko nebo kontejnery pro seskupení element do řádku či sloupců. Tyto nativní UI elementy lze kombinovat, a tím vytvořit složitější komponenty. Tyto složitější komponenty lze pak zapouzdřit do vlastně definované composable funkce. Důsledkem je to, že celé UI popisováno pomocí composable funkcí, které volají další composable funkce. Díky tomu je kód pro vytvoření UI modulární.

Jelikož composable funkce jsou vlastně jenom funkce napsané v programovacím jazyce, lze jim předávat parametry, které určují vzhled či chování daného UI elementu. Které parametry composable funkce přijímá a co určuji závisí na implementaci dané funkce. Jsou parametry, které mění např. barvu a velikost daného elementu. Dále jsou parametry, kterým předáváme funkci. Tato funkce může být volána např. když je na daný element kliknuto. Také jsou parametry, které určují textový obsah nějakého textového elementu. Výhodou použití knihovny Jetpack Compose je možnost definovat UI pomocí programovacího jazyka. To znamená, že lze využít cykly (např. pro vytvoření seznamu elementů) nebo podmínky (např. pro podmíněné vykreslování). Implementace rozhraní je díky tomu jednoduchá a intuitivní.

Alternativou k Jetpack Composu jsou XML layouty, které k popisu UI používají externí XML soubor, kde je rozhraní popisováno pomoxí XML tagů a jejich atributů. Hlavní myšlenkou tohoto přístupu je to, že oddělujeme popis uživatelského rozhraní od programového kódu, díky čemuž je kód čitelnější a udržitelnější. Míchání popisu UI a programového kódu se však nikdy nevyhneme, jelikož layoutům se musí minimálně předat aspoň data. Toho lze dosáhnout pouze propojením programového kódu s layoutem. Pokud implementujeme dynamický seznam, situace pro XML layouty je ještě horší, neboť implementace vyžaduje hodně boilerplate kódu. Pro Jetpack Composu je vytvoření takového seznamu otázkou pár řádků kódu.

Z výše uvedených důvodů jsem se rozhodl uživatelské rozhraní implementovat pomocí Jetack Compose.

zdroj: https://developer.android.com/develop/ui/views/layout/recyclerviewzdroj: https://developer.android.com/getpack/compose

Composable funkce

Základním stavebním kamenem pro tvorbu uživatelského rozhraní pomocí knihovny Jetpack Compose jsou composable funkce. V (7.7) lze vidět použití nativní composable funkce Text, která slouží pro vytvoření textového UI elementu. Vidíme, že funkce přijímá parametr text, který určuje textový obsah.

Výpis kódu 7.7 Příklad použití composable funkce Text.

```
Text(text = "Vysledky hlasovani")
```

Composable funkce lze kombinovat pro vytvoření složitějších composable funkcí (7.8). Vytváříme zde ikonku pomocí funkce Icon a text. Ikonka přijímá parametr specifikující objekt reprezentující danou ikonku a text používaný pro přístupové služby. Oba elementy jsou pak seskupeny do řádku pomocí kontejnerové composable funkce Row.

Výpis kódu 7.8 Příklad skládání composable funkcí. Vykreslí ikonku a text vedle sebe.

```
Row {
   Icon(
     imageVector = Icons.Filled.ArrowBack,
     contentDescription = "Tlacitko zpet",
)
   Text(text = "Detail hlasovani)
}
```

Mobilní aplikace 7

Pomocí parametrů lze měnit i vzhled a chování composable funkce. (7.9). Vytváříme zde tlačítko s textem a černým okrajem, které při kliknutí vypíše do konzole text.

Výpis kódu 7.9 Příklad parametrů pro změnu vzhledu a chování.

```
Button(
  onClick = { println("Kod pro ulozeni") },
  border = BorderStroke(0.dp, Color.Black)
) {
  Text(text = "Ulozit")
}
```

Kompozice a rekompozice

Kompozice funkce je proces, kdy se zavolá. Rekompozice funkce je pak proces, kdy se zavolá znovu v reakci na změnu jejího lokálního stavu nebo stavu ve ViewModelu, na kterém je závislý, nebo na rekompzici rodiče. Lokální stav composable funkce je popsána v následující sekci. Důsledkem je to, že změna composable funkce (jejího stavu) spustí rekompozici pouze této funkce a všech funkcí, které volá. Ostatní funkce zůstanou nedotčené. Díky tomu dojde k rekompozici pouze u funkcí, kde se něco změnilo nebo potěnciálně změnilo.

Stav composable funkce

Composable funkce mohou v sobě držet lokální stav. Kód (7.10) ukazuje příklad jeho použití. Zde vidíme text a tlačítko. Text se vykresluje podmíněně podle aktuální hodnoty proměnné expanded. Defaultní hodnota proměnné expanded je false, a tudíž při první kompozici funkce se text nevykreslí. Tlačítko v reakci na kliknutí této proměnné nastaví opačnou hodnotu, což spustí rekompozici funkce. Aby změna této proměnné spustila rekompozici, musí být typu State. Aby se proměnná však vůbec dala měnit, musí být typu MutableState, a proto je defaultní hodnota false obalena do funkce mutableStateOf. Aby se při rekompozici nenastavila opět defaultní hodnota false, ale nová hodnota, musí si composable funkce tento stav pamatovat napříř rekompozicemi, a toho dosáhneme pomocí funkce remember. Klíčové slovo by je syntactic sugar, který deleguje vrácení hodnoty na funkci remember. Pro nás to znamená jenom to, že typ proměnné expanded je Boolean a ne State¡Boolean¿. Pracuje se s tím pak lépe.

Výpis kódu 7.10 Příklad composable funkce používající lokální stav.

```
@Composable
private fun MyExpandableContent() {
   var expanded by remember {mutableStateOf(false)}

Row {
    if (expanded) {
        Text(text = "Zbytek obsahu")
    }
    Button(onClick = { expanded = !expanded },
    ) {
        Text(text = "Klikni pro zobrazeni zbytku obsahu")
    }
  }
}
```

Mnohem obvyklejší je však stav zapouzdřit do ViewModelu, který je určený k ukládání stavu UI, jak je ukázáno na (7.11). ViewModel si pamatuje stav i při navigaci na jinou obrazovku a

zpět. Díky může data stažená např. z internetu přepoužít a nemusí je zbytečně stahovat vícekrát, pokud se data málo mění.

Výpis kódu 7.11 Příklad composable funkce používající stav z ViewModelu.

```
private fun MyExpandableContent(viewModel: MyViewModel) {
    val expanded = viewModel.expandeed
    Row {
      if (expanded) {
        Text(text = "Zbytek obsahu")
      }
      Button(onClick = { viewModel.toggleExpanded() },
      ) {
        Text(text = "Klikni pro zobrazeni zbytku obsahu")
    }
}
class MyViewModel : ViewModel() {
  val expanded = mutableStateOf(false)
  fun toggleExpanded() {
    expanded = !expanded
  }
}
```

zdroj: https://developer.android.com/jetpack/compose/mental-model

Activity

Aktivita je jedna z hlavních komponent aplikace a reprezentuje obrazovku, do které vkládáme naše uživatelské rozhraní. V rámci aplikace je použita především jako vstupní bod pro vytvoření UI a pro počáteční inicializace stavu aplikace. Aktivita se může nacházet v různých stavech podle toho, zda ji má uživatel v popředí nebo pozadí, nebo zda ji vypíná. Nás bude zajímat především stav, kdy je aktivita poprvé vytvořena, tedy když je aplikace spuštěna. Pro detekci tohoto stavu poskytuje aktivita funkci onCreate, která se zavolá, když je aktivita vytvořena. V této funkci lze pak provádět žádané operace (7.12). Funkce setcontent vezme composable funkci PspApp, zakomponuje ji do aktuální aktivity a nastaví ji jako kořenovou composable funkci.

Výpis kódu 7.12 Třída MainActivity. (Soubor psp_fe/app/MainActivity)

```
class MainActivity : ComponentActivity() {
    ...
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    ...
    setContent {
            PspApp()
        }
    }
}
```

Mobilní aplikace 9

7.1.3 Implementace prezentační vrstvy

Popis implementace prezentační vrstvy je rozděleno na popis implementace uživatelského rozhraní a popis implementace držitelů stavů a logiky. Nejdřív bude popsána první implementace a po ní druhá.

Uživatelské rozhraní bylo implementováno pomocí knihovny Jetpack Compose. Pro implementaci byly použity následující composable funkce:

- Column Sloupcový kontejner pro elementy. Hodí se pro pozicování UI elementů na obrazovce. Nehodí se pro dlouhé seznamy elementů ve sloupci, jelikož nedělá rekompozici pouze viditelných elementů, ale všech. Nejde jím scrollovat. Používám ji ve všech místech, kde je potřeba napozicovat UI elementy pod sebou.
- LazyColumn Sloupcový kontejner pro element zoptimalizovaný pro dlouhé seznamy. Dělá rekompozici pouze pro viditelné elementy. Lze jím scrollovat. Používám ji pro zobrazení seznamů, tedy na obrazovce pro seznam hlasování, seznam poslanců, seznam hlasování jednotlivých klubů a seznam hlasování poslance.
- Row Řádkový kontejner pro elementy.
- Text Textový element.
- Image Obrázkový element. Obrázek lze získat pomocí URL adresy. Používám ho pro zobrazení profilové fotky poslance a logů klubů.
- Icon Funkce pro různé ikonky, např. pro ikonky v hlavičce nebo v dolní liště.
- IconButton Tlačítko, který má místo textu ikonky. Používám ji hlavičce pro vytvoření ikonky pro vyhledávání v seznamu hlasování nebo poslanců.
- OutlinedButton Tlačítko s kontrastem barev mezi pozadím a obvodem.
- OutlinedTextField Textové pole s kontrastem barev mezi pozadím a obvodem.
- FloatingActionButton Plovoucí tlačítko. Používám ji pro skok na začátek seznamu.
- Scaffold Kontejner umožňující jednoduchým způsobem sdílet UI element mezi více obrazovkami. Umožňuje také jednoduchým způsobem přidat plovoucí tlačítko.
- TopAppBar Horní lišta.
- Spacer Vytváří mezeru mezi dvěma UI elementy. zda je mezera horizontální nebo veritkální, specifikujeme skrz parametr.
- Divider Oddělovač. Dá se nastavit na vertikální nebo horizontální.
- CompositionLocalProvider Umožňuje přepsat hodnotu kontextu. Používám ji pro lokální přepsání globálních barev.
- Box Kontejner umožňující skládat UI elementy na sebe. Používám ho pro vytvoření popupu s nastavením volebního období v nastavení. Na pozadí je seznam nastavení a na něm se objeví daný popup.
- SettingsMenuLink Composable funkce z knihovny Alorma pro snadné vytvoření prvku v seznamu nastavení přesně podle návrhu. Knihovna ji definuje defaultní vzhled.
- ListItemPicker Composable funkce z knihovny Alorma reprezentující seznam hodnot, kterým lze scrollovat. Knihovna ji definuje defaultní vzhled.

 Card - Karta obsahující libovolný obsah a prvek pro akci. Používám ji např. pro elementy v seznam hlasování.

- TabRow Lišta s taby. Používám ji na obrazovce s detailem hlasování a na obrazovce s detailem poslance.
- Tab Reprezentuje konkrétní obrazovku v rámci tabu.
- HorizontalPager Layout umožňující horizontálne scrollovat obsahem. Používám ho pro scrollování mezi taby.

Stav UI a logika pro obsluhu změn UI jsou zapouzdřeny v držitelích stavů a logiky, což jsou view modely. Jsou to objekty, s kterými komunikuje Jetpack Compose pro získání dat nebo pro informování o nějaké události. Na (7.13) lze vidět ukázku využití view modelu. Na začátku view modelu se nachází stav pro seznam hlasování. PagingData je datový typ pro stránkovaný obsah. MutableStateFlow je datový typ pro flow, který lze měnit. Flow je vyžadován pro implementaci stránkování pomocí PagingData. Vně view je viditelná pouze imutabilní verze stavu StateFlow. Pro bezpečnost se se stavem dá manipulovat pouze uvnitř view modelu. Při spuštění view modelu se proveden coroutinová operace v rámci scopu view modelu. To znamená, že pokud je view model odstraněn, zruší se i coroutinová operace. Coroutina získává data pomocí funkce getVotes. Ta vyžaduje volební rok a klíčová slova pro filtrování seznamu. Jakmile se změní jedna z těchto dvou hodnot, měly by se stáhnout nová data. Obě hodnoty jsou implementeovány jako flow, a proto je nejdřív kombinujeme a zavoláme na nich terminální operátor collectLatest. Emitované hodnoty se pomocí cachedIn(viewModelScope)zakešují v rámci view modelu. To znamená, že pokud se na daném flow opět zavolá terminální operátor v jiné části kódu, vrátí se zakešovaná hodnota.

Výpis kódu 7.13 Ukázka využití view modelu

```
// Soubor VoteListViewModel.kt
class VoteListViewModel(
): ListViewModel(...) {
  private val _votes: MutableStateFlow < PagingData < Vote >> =
  MutableStateFlow(PagingData.empty())
  val votes: StateFlow<PagingData<Vote>> = _votes
  init {
    viewModelScope.launch {
      currentElectionYear.combine(searchText, ::Pair)
      .collectLatest { pair ->
        getVotes(pair.first, pair.second)
        .cachedIn(viewModelScope)
        .collect { votes ->
           _votes.value = votes
      }
    }
  }
}
```

Navigace mezi obrazovkami je implementována pomocí composable funkce NavHost, která definuje jednotlivé destinace. Destinace reprezentuje obrazovku, ke které lze pomocí této funkce navigovat.

Mobilní aplikace 11

Destinace je reprezentována řetězcem. Např. destinace pro seznam hlasování se nazývá "votes". Pro samotnou navigaci se používá objekt NavHostController, na které zavoláme metodu navigate a předáme ji název destinace, ke které chceme navigovat. Pro navigaci k detailu hlasování nebo poslance používá sepcifičtější název destinace, např. "votes/1234"nebo "member/5678".

7.1.4 Implementace doménové vrstvy

Ukázka implementace doménové vrstvy pro seznam hlasování (7.14). Objekt Pager je vstupním bodem pro stránkovací mechanizmus. Nastavujeme mu defaultní velikost stránky, v době psaní práce je to 20. Dále mu předáme VotesPagingSource reprezentující zdroj stránkovaných dat. Pager poskytuje proměnnou flow, která bude emitovat stránky. flowOn nastaví, aby sbírání flow probíhalo na speciálním vlákně IO, aby neprobíhalo na hlavním vlákně. Není to nutné díky používání coroutinů, ale je to dobrá praxe to používat pro síťové operace.

Výpis kódu 7.14 Ukázka využití use caseu pro získání seznamu hlasování

```
// Soubor GetVotesUseCase.kt
class GetVotesUseCase @Inject constructor(...) {

  operator fun invoke(electionYear: Int, searchText: String) = Pager(
    PagingConfig(pageSize = DEFAULT_PAGE_SIZE)
) {
    VotesPagingSource(
        voteRepository = voteRepository,
        electionYear = electionYear,
        searchText = searchText
    )
}
.flow
.flowOn(Dispatchers.IO)
}
```

Doménové vrstvy pro obrazovky, které nepoužívají seznam, jsou přímočařejší. Ukázku takového use caseu lze vidět na (7.15).

Výpis kódu 7.15 Ukázka využití use caseu pro získání detailu hlasování

```
// Soubor GetVoteDetailUseCase.kt
class GetVoteDetailUseCase @Inject constructor(...) {
  operator fun invoke(id: Int): Flow<VoteDetails> =
    voteRepository.getVoteDetail(id)
    .flowOn(Dispatchers.IO)
}
```

7.1.5 Implementace datové vrstvy

Datová vrstva slouží pro abstrahování doménové vrstvy od konkrétních datových zdrojů. V aplikaci je použit jediný datový zdroj, a to vzdálený pro získání data z backendu. V datové vrstvě jsou získávána data z datového zdroje v podobě síťových entit. Ty jsou následně namapována na doménové entity pro použití doménovou vrstvou. Ukázka použití je na (7.16).

Výpis kódu 7.16 Ukázka datové vrstvy pro data o hlasováních

```
class VoteRepositoryImpl @Inject constructor(...) : VoteRepository {
```

```
override suspend fun getVotes(
   page: Int,
   size: Int,
   electionYear: Int,
   searchText: String
): List<Vote> =
   pspRemoteDataSource.getVotes(
    page = page,
    size = size,
   electionYear = electionYear,
   searchText = searchText
).map { it.toDomain() }
...
}
```

Datové zdroje jsou pouze abstrakcí nad konkrétním rozhraním pro získání daného zdroje. Vzdálený datový zdroj tedy abstrahuje datovou vrstvu od přímé komunikace s knihovnou pro získání dat z internetu. Na (7.17) je ukázka datového zdroje. Funkce <code>getAppState()</code> pouze zavolá funkci definovanou pomocí knihovny Retrofit, která bude popsána v následující sekci.

Výpis kódu 7.17 Ukázka datového zdroje

```
class PspRemoteDataSourceImpl @Inject constructor(
  private val pspApi: PspApi
) : PspRemoteDataSource {
  override suspend fun getAppState() =
     pspApi.getAppState()
  ...
}
```

7.1.6 Implementace síťové vrstvy

Síťová vrstva je implementována pomocí knihovny Retrofit. Je to HTTP klient pro komunikaci se zdroji na internetu. Aplikaci ho používá pro získání veškerých dat z backendu. Na (7.18) je ukázka použití. URL adresu daného zdroje specifikujeme pomocí anotace @GET. Pomocí anotací Query specifikujeme query parametry pro jako číslo stránky a počet stránek. Návratovou hodnotou je seznam API entit. Implementaci této funkce má na starosti knihovna. Data z backendu jsou ve formátu JSON a jsou namapována na objekt v návratovém typu.

Výpis kódu 7.18 Ukázka použití knihovny Retrofit pro získání seznamu hlasování z backendu

Backend 13

}

zdroj: https://square.github.io/retrofit/

7.2 Backend

7.2.1 Použité nástroje a technologie

Intellij IDEA

Backend byl vyvíjen ve vývojovém prostředí Intellij IDEA. Hlavní důvody pro výběr tohoto IDE:

- Vestavěný inicializátor Spring Boot aplikací Pomocí tohoto inicializátor lze nakonfigurovat
 a nastavit potřebné závislosti ve Spring Boot¹ aplikaci jednoduše naklikáním v průvodci.
- Klávesové zkratky Toto IDE je vyvíjeno společností JetBrains, a tudíž obsahuje stejné klávesové zkratky jako Android Studio.

zdroj: https://www.jetbrains.com/idea/

Maven

Maven je stejně jako Gradle nástroj pro automatizaci sestavování programu. Původně byl použit Gradle kvůli čitelnější syntaxi. Maven byl zvolen z historických důvodů. Původně bylo v plánu backend nasadit na Cloud Azure. Ten poskytoval plugin pro maven, který umožňoval backend nasadit a zprovoznit pomocí jednoho příkazu. Kvůli omezeného free tieru však byl backend nakonec nasazen na jiný cloud (více v kapitole o nasazení). Maven je tedy pozůstatek historického rozhodnutí. Na funkčnosti aplikace to však nemá žádný vliv, a proto se už nepřešlo z Mavenu na Gradle.

zdroj: https://maven.apache.org/ https://azure.microsoft.com/en-us

Spring Boot

Před popisem technologie Spring Boot si popíšeme technologii Spring, na které je Spring Boot založen. Spring je open-source framework pro vývoj enterprise aplikací. Je to seskupení různých nástrojů pro řešení různých problémů. Pro účely této práce byly využity nástroje pro vytvoření webových aplikacím, ale umožňují implementovat i backend s REST API. Hlavními benefity Springu je množství použitelných nástrojů a dependency injection pomocí anotací.

Spring Boot je framework, který je postavený na Springu a který má za cíl redukci boilerplate kódu a nutnost konfigurace, a tím urychlit vývoj aplikace. Toho dosahuje pomocí autokonfigurace, což je vlastnost Spring Bootu, kdy jsou jednotlivé komponenty (např. rozhraní pro komunikaci s databází, webový server, ORM) automaticky nakonfigurovány Spring Bootem pomocí defaultních hodnot. Defaultní hodnoty jsou Spring Bootem nastaveny na nejčastější hodnoty, aby se tím pokrylo co nejvíce situací, na které může vývojář narazit. Důsledkem je to, že pro zprovoznění komponent je potřeba pouze nainstalovat jejich závislost. Pokud bude potřeba použít jinou konfiguraci (např. pro připojení k databázi chceme použít jiný než defaultní port), Spring Boot to umožňuje. Pointa je však, že knihovnu pouze stáhneme bez nutnosti konfigurace nebo s minimální konfigurací a aplikace lze hned spustit.

Alternativou ke knihovně Spring Boot je knihovna Ktor, pomcí které lze taktéž vytvořit backend s REST API. Výhodou této knihovny je, že je určená pro psaní v Kotlinu. Lze tedy využít všechny výhody tohoto jazyka. Nevýhodou je, že nepodporuje defaultně dependency injection.

¹Technologie Spring Boot bude popsána později.

Pro něj se musí zvlášť stáhnout knihovna a a tu nakonfigurovat. Další nevýhodou oproti Spring Bootu je chybějící autokonfigurace. Vše se tedy musí manuálně nakonfigurovat.

```
zdroj:
https://spring.io/
https://spring.io/why-spring
https://spring.io/web-applications
https://ktor.io/
```

Java

Java byla od začátku hlavním programovacím jazykem pro vývoj aplikací pomocí technologie Spring Boot. Od roku 2017 přišla integrace jazyka Kotlin do Spring Bootu a v dokumentaci jsou ukázkové kódy psány jak v Javě tak i Kotlinu. Java má oproti Kotlin větší podporu v komunitě, co se týče vývoji ve Spring Bootu. Když jsem při psaní v Javě narazil na nějaký problém, mnohem šlo najít na internetu řešení než při psaní v Kotlinu. Jedním řešením je hledat řešení v Javě a javovský kód pomocí automatického nástroje ztransfomovat do Kotlinu, výsledný kód bylo však potřeba vždy prošistit, jelikož u všech proměnných vždy obsahoval datové typy, které lze v Kotlinu v některých případech vynechat. Zároveň výstupní kód někdy nebyl kvůli nějaké drobnosti kompilovatelný. Z toho důvodu jsem se rozhodl pro použití Java. Zdá se však, že podpora pro psaní aplikací Spring Boot pomocí Kotlinu je čím dál tím větší. Když bych měl možnost backend přepsat, znovu bych zauvažoval o použití Kotlinu, jelikož je to velmi dobrý programovací jazyk.

```
zdroj: https://spring.io/guides/tutorials/spring-boot-kotlin/https://spring.io/blog/2017/01/04/introducing-kotlin-support-in-spring-framework-5-0zdroj: https://docs.spring.io/spring-boot/docs/current/reference/html/index.html
```

MySQL

Na databázi backendu nebyly kladeny velké nároky. Pouze bylo potřeba data perzistentní uložit, aby byly připravené pro použití mobilní aplikací. Zvolil jsem tedy MySQL, s kterým jsem měl zkušenosti. Bylo by však možné použít např. i PostreSQL. Na funkčnosti backendu by to však nemělo vliv.

7.2.2 Prezentační vrstva

V prezentační vrstvě byly implementovány jednotlivé endpointy REST API, které přijímají dotazy od mobilní aplikace a vrací ji předzpracovaná data uložena v databázi. Ukázku implementace endpointu pro získání detailu konkrétního hlasování lze vidět na (7.19). Endpoint pro HTTP GET požadavky je ve Spring Bootu implementován tak, že vytvoříme metodu s libovolným názvem a označíme ji anotací <code>@GetMapping</code>. V parametru této anotace pak specifikujeme URL adresu daného endpointu. Celá metoda se musí nacházet ve tříde s anotací <code>@RestController</code>. Endpointy jsou rozděleny do několika tříd podle typu dat (jedna třída pro entity hlasování, druhá pro entity poslance a třetí pro entity klubů). Pokud URL adresa obsahuje parametr identifikující daný zdroj, musí být obsažen v parametru metody a oanotován pomocí <code>@PathVariable</code>

Výpis kódu 7.19 Ukázka kódu pro vytvoření endpointu

```
// Soubor VoteController.java
@RestController
public class VoteController {
   private final VoteService service;
   private final VoteMapper mapper;
```

Backend 15

```
...
@GetMapping("/vote/{id}")
public DetailedVote getVote(@PathVariable Integer id) {
   Vote vote = service.getVote(id);
   return mapper.toDetailedVote(vote);
}
```

Query parametry jsou implementovány pomocí anotace RequestParam, kterému je v parametru předán název parametru:

Výpis kódu 7.20 Ukázka endpointu s request parametrem

Podobným způsobem jsou implementovány query parametry pro stránkování (číslo a velikost stránky). Kromě povinného parametru pro název query parametru lze specifikovat i parametr required pro nastavení volitelnosti nebo defaultValue pro nastavení defaultní hodnoty query parametru.

Na dvou předchozích ukázkách lze vidět použití hodnot service a mapper. První reprezentuje doménovou vrstvu, která získá data z databázové vrstvy a druhá obsahuje metody pro mapování objektů, pokud je to potřeba (pokud je struktura entit uložených v databázi jiná než struktura entit pro REST API). Takto jsou implementovány všechny controllery. Controllery, které obsahují endpoint, který vrací seznam, navíc sestavují hlavičky pro HTTP odpověď. K tomu je použita třída HttpHeaders, kterou poskytuje Spring framework. Instanci této třídy se jednoduchým způsobem nastaví potřebné hodnoty a objekt je controllorem vracen:

Výpis kódu 7.21 Ukázka nastavení hlaviček pro stránkování

```
// Soubor PaginationHeaderGenerator.java
public static HttpHeaders buildHeaders(...) {
   HttpHeaders responseHeaders = new HttpHeaders();
   ...

responseHeaders.set(previousPageString, String.valueOf(prevPage));
responseHeaders.set(nextPageString, String.valueOf(nextPage));
responseHeaders.set(lastPageString, String.valueOf(lastPage));
return responseHeaders;
```

```
// Soubor VoteController.java
@GetMapping("/vote")
public ResponseEntity < List < General Vote >> getVotes(...) {
    ...

HttpHeaders headers = GenericAndPaginationHeaderGenerator
    .buildHeaders(pagedResult.getTotalPages(), page);
    ...

return ResponseEntity
    .ok()
    .headers(headers)
    .body(general Votes);
}
```

7.2.3 Doménová vrstva

Doménová vrstva má na starosti business logiku aplikace a abstrahování prezentační vrstvy od implementačních detailů databázové vrstvy. V našem případě backend neobsahuje business logoiku, pouze filtruje data. Ukázka kódu pro získání detailu poslance:

Výpis kódu 7.22 Ukázka kódu pro získání detailu poslance

```
// Soubor MemberService.java
public Member getMember(int id) {
  return memberRepository
  .findById(id)
  .orElseThrow(() -> new MemberNotFoundException(id));
}
```

Pro získání seznamů je použito stránkování, které je implementováno pomocí třídy Pageable. Ta se předá repozitáři, který bude popsán v následující sekci:

Výpis kódu 7.23 Ukázka doménové vrstvy pro vrácení seznamu poslanců

```
// Soubor MemberService.java
public Page<Member> getMembers(PagingParams pagingParams) {
   Pageable pageable = PageableGenerator.buildPageable(pagingParams);
   if (filterName == null) {
      return memberRepository
      .findByElectionYear(electionYear, pageable);
   } else {
      // kod pro filtrovani poslancu
   }
}
```

Instance třídy Pageable je vytvořena následovně:

Výpis kódu 7.24 Ukázka kódu pro sestavení objektu pro stránkování

Backend 17

```
public static Pageable buildPageable(...) {
   Pageable pageable;
   ...
   pageable = PageRequest.of(page, size, sort);
   // napr. page = 2, size = 20, sort = Sort.by("dateTime").descending())
   return pageable;
}
```

Doménová vrstva pro ostatní entity jsou implementovány obdobně jako bylo popsáno výše. Výjimkou je metoda pro získání detailu hlasování. Jedním z atributů hlasování jsou statistky hlasování. Výpočet počtu omluvených a nepřihlášených poslanců příliš zpomalovalo zpracování zdrojových dat přes uložením do databáze. Z toho důvodu je tento výpočet prováděn až za běhu, kdy se mobilní aplikace nad daným detailem hlasování dotazuje:

Výpis kódu 7.25 Ukázka dopočtu statistik pro detail hlasování za běhu v doménové vrstvě

```
public Vote getVote(int id) throws IOException {
    ...
    int excusedCount = ...
    int loggedOffCount = ...
    ...
    // nastaveni hodnot excusedCount a loggedOffCount
    // vraceni vysledku
}
```

7.2.4 Databázová vrstva

V databázové vrstvě získávám data z databáze pomocí dotazů. Pro komunikaci s databází je použita knihovna Hibernate, která poskytuje objektově-relační mapování, díky kterému jsme abstrahování od databázových tabulek a místo toho pracujeme s objekty (entitami), které jsou danou knihovnou namapovány na dané tabulky. Příkladem takového objektu je entita pro hlasování, jejíž ukázka je na (??). Pomocí anotace Entity říkáme, že daný objekt je databázovou entitou. Hibernate nám automaticky vytvoří tabulku s danými atributy v databázi. Anotace @Getter je z knihovny Lombok a slouží pro vygenerování getterů pro všechny atributy. Anotací @Id specifikujeme atribut, který má být primárním klíčem v tabulce.

Výpis kódu 7.26 Entita Vote reprezentující hlasování

```
// dalsi anotace
@Entity(name = VOTE)
@Getter
public class Vote {
    @Id
    private int id;
    private LocalDateTime dateTime;
```

```
// zbytek atributu
}
```

Pro dotazování se nad daty používám knihovnu spring-data-jpa, která poskytuje rozhraní JpaRepository, který obsahuje základní metody pro manipulaci s danou entitou jako např. findAll() pro získání všech záznamů z tabulky nebo findById() pro získání záznamu s daným id. Největší sílou této knihovny je však možnost vytvoření vlastních metod, kterým se říká query metody. Dotaz se sestaví na základě pojmenování query metody podle určitých pravidel. Např. následující metoda vrací seznam hlasování ve volebním období, který začal daným volebním rokem:

Výpis kódu 7.27 Repozitář pro hlasování

```
// Soubor VoteRepository.java
@Repository
public interface VoteRepository extends JpaRepository < Vote, Integer > {
    ...
    List < Vote > findByElectionYear(int electionYear);
}
```

Přitom nebylo potřeba metodu implementovat. Knihovna si na základě názvu metody implementaci vygeneruje. Metodu lze modifikovat, aby vracela stránkovaný obsah:

Výpis kódu 7.28 Ukázka query metody pro dotazování se nad stránkovaným obsahem

```
// Soubor VoteRepository.java
Page < Vote > findByElectionYear(int electionYear, Pageable pageable);
```

Lze i vytvořit komplexnější dotaz, který se dotazuje nad entitou na základě dvou podmínek:

Výpis kódu 7.29 Ukázka query metody s dvěma podmínkami

```
// Soubor MembershipRepository.java
boolean existsByPersonIdAndAgencyId(int personId, int agencyId);
```

zdroj: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/

7.3 Zpracování dat

Backend každý den o půl noci aktualizuje databázi podle zdrojových dat na webu PSP. Aktualizace probíhá v následujících krocích:

- Stažení zdrojových souborů Zdrojové soubory jsou ve formátu zip a jsou stažené z webu PSP (https://www.psp.cz/sqw/hp.sqw?k=1300).
- Extrakce datových souborů Ze zdrojových souborů jsou vyextrahovány datové soubory, které jsou ve formátu UNL.
- Pročištění dat Některé datové soubory obsahují duplicitní záznamy. Ty jsou pro snadnější parsování odstraněny.
- Parsování dat Datové soubory jsou zparsovány a načteny do Java objektů.
- Transformace dat Objekty jsou ztransformovány do požadované podoby.
- Uložení dat do databáze Ztransfomovaná data jsou perzistentně uložena do databáze.

Zpracování dat 19

7.3.1 Stahování zdrojových souborů

Stahování zdrojových souborů má na starosti třída následující třída:

Výpis kódu 7.30 Třída pro stahování zdrojových souborů

```
// Soubor PspFilesDownloader.java
public class PspFilesDownloader {
  public static void downloadFiles() throws IOException {
    downloadHlasovani();
    downloadPoslanci();
}

// dalsi metody
}
```

Tato třída pro stahování souborů na základě URL interně používá knihovnu commons-io (7.31). Knihovna poskytuje metodu copyURLToFile, která akceptuje parametr pro URL zdrojového souboru a parametr pro lokální soubor, do kterého se má zdrojový soubor nakopírovat.

Výpis kódu 7.31 Ukázka stahování dat pomocí knihovny commons-io

```
// Soubor FileDownloader.java
public class FileDownloader {

  public static void download(
    String downloadUrlString,
    String downloadDestination) {

    File file = new File(downloadDestination);
    URL downloadUrl = new URL(downloadUrlString);

    FileUtils.copyURLToFile(downloadUrl, file);
    ...
}
```

zdroj: https://commons.apache.org/proper/commons-io/

7.3.2 Extrakce datových souborů

Pro extrakci datových souborů byla použita knihovna zip4j. Ta poskytuje třídu ZipFile pro vytvoření objektu reprezentujícího lokální soubor ve formátu zip. Konstruktoru této třídy je předána lokace souboru v souborovém systému. Tato třída poskytuje metodu extractFile() pro extrakci souborů ze zipu. Akceptuje tři parametry:

- Název souboru, který se má vyextrahovat.
- Adresář, do které se má soubor vyextrahovaný soubor uložit.
- Výsledný název ukládaného souboru. V našem případě je stejný jako název před vyextrahováním souboru.

Zde je ukázka kódu pro popsanou extrakci souborů:

Výpis kódu 7.32 Ukázka extrakce souborů ze zipu

```
// Soubor ZipExtractor.java
```

```
public class ZipExtractor {

  public static void extract(
    String pathToZip,
    String fileToExtract,
    String destinationDir
) {
    ZipFile zipFile = new ZipFile(pathToZip)
    zipFile.extractFile(fileToExtract, destinationDir, fileToExtract);
}
```

zdroj: https://github.com/srikanth-lingala/zip4j

7.3.3 Pročištění dat

Některé datové soubory obsahují duplicitní záznamy. Ty jsou odstraněny pomocí skriptu napsaného v jazyce Bash:

Výpis kódu 7.33 Skript pro odstranění duplicitních řádků

```
// Soubor removeDuplicates.sh.java
for FILE in "$1"/*; do
   sort "$FILE" | uniq > 'tmp.unl'
   mv 'tmp.unl' "$FILE"
   rm 'tmp.unl'
done
```

Skript funguje následovně:

- Skript předpokládá ve svém prvním argumentu cestu ke adresáři, kde se nachází datové soubory.
- Na začátku se iteruje přes všechny soubory v daném adresáři.
- Každý soubor se pomocí příkazu sort seřadí vzestupně podle abecedy.
- Ze seřazeného souboru se odstraní duplicitní řádky jdoucí za sebou pomocí příkazu uniq. Vy tuto chvíli jsou ze souboru odstraněny všechny duplicity.
- Zbytek kódu již je pouze přesouvání obsahů souborů tak, aby datové soubory s odstraněnými duplicity měly jejich originální název.

Skript je volán ze souboru PspFilesCleaner. Důvodem pro odstranění duplicit pomocí jazyka Bash a ne přímo pomocí Javy je to, že v Bashi jsou operace jednodušší a rychlejší než v Javě.

7.3.4 Parsování dat

Pro parsování zdrojových dat používám knihovnu opencsv. Ta umožňuje parsovat CSV soubory tak, že každý řádek v souboru namapuje na objekt. Mapování sloupců v souboru na atributy objektu je implementováno přidáním anotací k příslušným atributům. Vysvětlíme si na příkladu. Nejdříve je potřeba vytvořit třídu, do jejíž instancí se napamatují záznamy ve zdrojovém souboru. Ta bude mít u atributů anotaci určující pozici neboli číslo sloupce záznamu. Hodnota tohoto sloupce se pak napamatuje na tento atribut:

Zpracování dat 21

Výpis kódu 7.34 Parsování datového souboru omluvy.unl

```
// Soubor Omluva.java
public class Omluva {
   @CsvBindByPosition(position = 0)
   private int idOrgan;

   @CsvBindByPosition(position = 1)
   private int idPoslanec;

// dalsi atributy
}
```

Nyní lze datový soubor zparsovat a namapovat na tuto třídu. Knihovna poskytuje třídu CsvToBeanBuilder, které předáme objekt typu —Reader—. Ta slouží pro čtení znaků ze streamu. V tomto případě je do streamu posílány data z datového souboru. Třídá CsvToBeanBuilder poskytuje metody, kterými specifikujeme:

- Třídu, na kterou se mají záznamy namapovat.
- Oddělovač hodnot v záznamu.
- Určení, kdy je hodnota null. V tomto případě hodnota interpretována jako null, pokud je prázdná.
- Ignorování složených závorek.

Výpis kódu 7.35 Parsování datového souboru omluvy.unl

```
// Soubor OmluvyParser.java
public class OmluvyParser extends UnlParser {
   public List<Omluva> read() {
      String filePath = PspPath.Unl.OMLUVY;
      BufferedReader reader = getReader(filePath);

      return new CsvToBeanBuilder<Omluva>(reader)
      .withType(Omluva.class)
      .withSeparator(UNL_SEPARATOR)
      .withFieldAsNull(UNL_NULL_SEPARATOR)
      .withIgnoreQuotations(true)
      .build()
      .parse();
}
```

zdroj: https://opencsv.sourceforge.net/

7.3.5 Transformace a uložení dat

Po zpasrování datových souborů a namapování záznamů na objekty lze tyto objekty začít ztransformovat do žádané podoby. Kód pro transformaci má u všech typů dat podobný tvar:

Výpis kódu 7.36 Transformace objektu Omluva na databázový objekt Excuse

```
// Soubor ExcuseLoader.java
```

Před samotným parsováním jsou smazány všechny záznamy v příslušné tabulce. Poté probíhá parsování datových souborů a seznamu namapovaných objekt. Z tohoto seznamu je vytvořen paralelní stream, což stream, který potenciálně zpracuje objekt v seznamu paralelně. Každý objekt je ztransformován a namapován na databázový objekt. Konkrétní implementace transformace se liší od typu entity. Ztransformovaná data jsou perzistetně uložena do databáze. Ukládání je optimalizováno tak, že se neukládá po jednom ale po skupinách. Každá skupina obsahuje 1000 objektů. Ukládáme tedy po 1000 objektech.

Kapitola 8

Testování

- 8.1 Mobilní aplikace
- 8.2 Backend

2 Testování

Nasazení

- 9.1 Aplikace
- 9.2 Backend

2 Nasazení

Kapitola 10 Spuštění

10.1 Aplikace

10.2 Backend

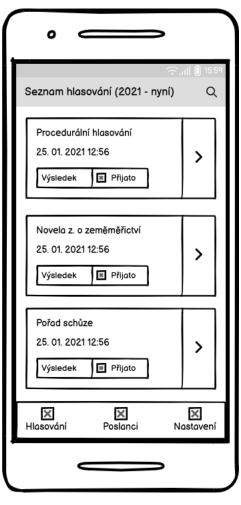
2 Spuštění

• •	 	 Kapitola 11
		Závěr

2 Závěr

Příloha A

Příloha





- Obrázek A.1 Seznam hlasování
- Obrázek A.2 Vyhledávání v seznamu hlasování
- Obrázek A.3 Obrazovka pro seznam hlasování

Výpis kódu A.1 Tělo odpovědi pro dotaz GET /api/app.

```
1
     "election_years": [
2
     2021,
3
4
     2017,
     2013,
5
     2010,
6
     2006,
7
     2002,
     1998,
9
     1996,
10
     1992
11
12
13
```

Výpis kódu A.2 Tělo odpovědi pro dotaz GET /api/vote

```
Г
1
2
       "id": 1,
3
       "date_time": "16. 12. 2022 13:29",
4
       "description": "Hlasovani 1",
5
       "result": "A"
6
7
8
       "id": 2,
9
       "date_time": "16. 12. 2022 13:26",
10
       "description": "Hlasovani 2",
11
       "result": "A"
12
13
14
```

■ Výpis kódu A.3 Tělo odpovědi pro dotaz dGET /api/votei

```
1
     "id": 1,
2
     "date_time": "16. 12. 2022 13:29",
3
     "description": "Hlasovani 1,
4
     "result": "A",
5
6
     "steno_protocol_url": "http://www.psp.cz/eknih/2021ps/stenprot/
        048 schuz/s048109.htm #h76",
     "yes_count": 100,
7
     "no_count": 0,
     "logged_off_count": 64,
9
     "excused_count": 0,
10
     "refrained_count": 36,
11
     "election_year": 0
12
13
```

Výpis kódu A.4 Tělo odpovědi pro dotaz GET /api/party/vote/1

```
1 [
2 {
```

```
"party_name": "Nazev klubu",
3
       "logo_url": "https://www.psp.cz/pics/klub/l-cps.jpg",
4
       "vote_id": 1,
5
       "party_results": {
6
         "yes_count": 2,
7
8
         "no_count": 0,
         "logged_off_count": 1,
9
         "excused_count": 0,
10
         "refrained_count": 0
11
       "member_results": [
13
14
           "member_name": "Poslanec 1",
           "vote_result": "@"
16
17
18
19
           "member_name": "Poslanec 2",
           "vote_result": "C"
20
21
22
           "member_name": "Poslanec 3",
23
           "vote_result": "A"
24
25
26
           "member_name": "Poslanec 4",
           "vote_result": "A"
29
30
31
32
```

Výpis kódu A.5 Tělo odpovědi pro dotaz "

```
Е
1
2
       "id": 1,
3
       "name": "Poslanec 1",
4
       "party": "ANO",
       "photo_url": "https://www.psp.cz/eknih/cdrom/2021ps/eknih/202
6
          1ps/poslanci/i6474.jpg",
       "election_region": "Volebni kraj 1",
7
       "election_year": 2021
8
9
10
       "id": 2,
11
       "name": "Poslanec 2",
12
       "party": "ODS",
13
       "photo_url": "https://www.psp.cz/eknih/cdrom/2021ps/eknih/202
14
          1ps/poslanci/i6804.jpg",
       "election_region": "Volebni kraj 2",
       "election_year": 2021
16
17
18
```

Výpis kódu A.6 Tělo odpovědi pro dotaz GET /api/member/1

```
1
     "id": 1,
2
     "name": "Poslanec 1",
3
     "gender": "M",
4
     "party": "Poslanecky klub",
5
     "member_from": "12. 10. 2021",
     "member_to": null,
7
     "date_of_birth": "25. 09. 1970",
8
     "election_region": "Volebni kraj 1",
9
     "photo_url": "https://www.psp.cz/eknih/cdrom/2021ps/eknih/2021
10
        ps/poslanci/i6474.jpg",
     "election_year": 2021
11
12
```

■ Výpis kódu A.7 Tělo odpovědi pro dotaz GET /api/member/1/vote

```
1
2
       "vote": {
3
         "id": 1,
4
         "date_time": "16. 12. 2022 13:29",
5
         "description": "Hlasovani 1",
6
         "result": "A"
7
       "how_member_voted": "@"
9
10
11
       "vote": {
12
         "id": 2,
13
         "date_time": "16. 12. 2022 13:26",
14
         "description": "Hlasovani 2",
15
         "result": "A"
16
17
       "how_member_voted": "@"
18
19
20
```

■ Tabulka A.1 Struktura agency

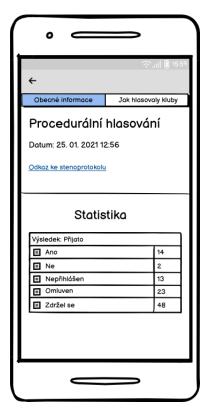
Název	Тур	Popis
id	int	identifikátor orgánu
abbreviation	varchar(255)	zkratka názvu orgánu
end_date	date(255)	datum zániku orgánu
name	varchar(255)	název orgánu
start_date	date	datum založení orgánu
type_id	int	identifikátor typu orgánu

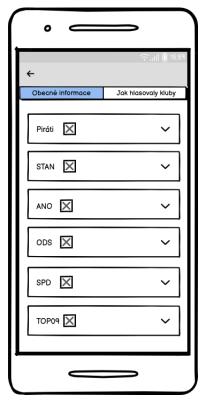
■ Tabulka A.2 Struktura agency_type

Název	Тур	Popis
id	int	identifikátor typu orgánu
name	varchar(512)	název typu orgánu
superior_agency_type_id	int	identifikátor nadřazeného typu orgánu

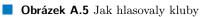
■ Tabulka A.3 Struktura excuse

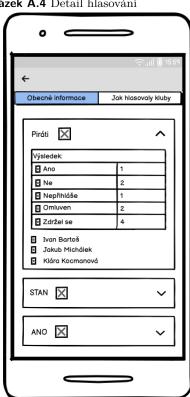
Název	Тур	Popis
member_id	int	identifikátor poslance, který je omluven
date	date	datum, kdy je poslanec omluven
start_time	time	čas, od kterého byl poslanec omluven
end_time	time	čas, do kterého byl poslanec omluven
election_year	int	první rok volebního období



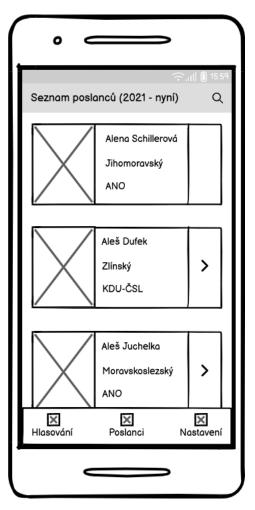


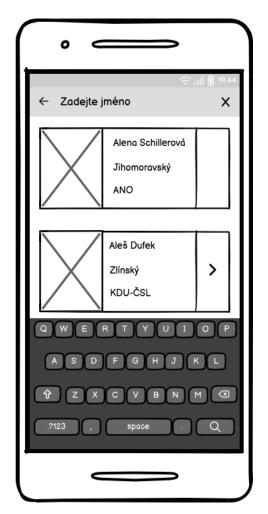
Obrázek A.4 Detail hlasování



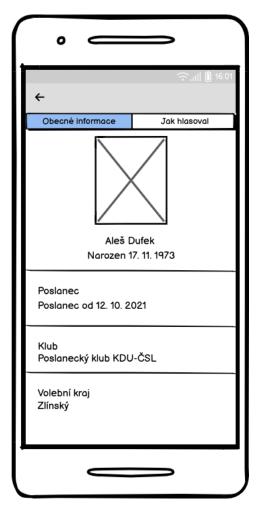


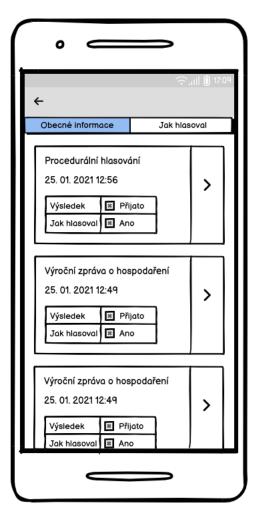
- Obrázek A.6 Jak hlasovaly kluby
- Obrázek A.7 Obrazovky pro detail hlasování



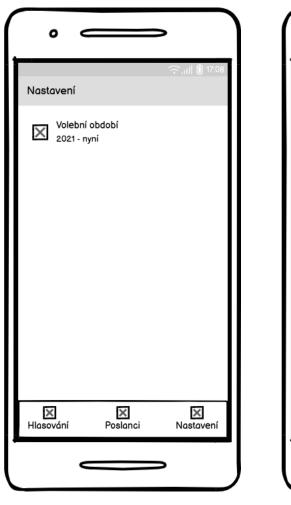


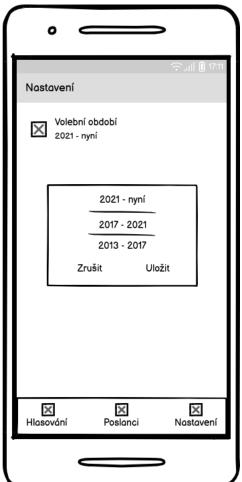
- Obrázek A.8 Seznam poslanců
- Obrázek A.9 Vyhledávání v seznamu poslanců
- Obrázek A.10 Obrazovka pro seznam poslanců





- Obrázek A.11 Detail poslance
- Obrázek A.12 Jak hlasoval/a poslanec/kyně
- Obrázek A.13 Obrazovky pro detail poslance





- Obrázek A.14 Seznam nastavení
- Obrázek A.15 Nastavení volebního období
- Obrázek A.16 Obrazovka pro nastavení

■ Tabulka A.4 Struktura member

Název	Тур	Popis
id	int	identifikátor poslance, který je omluven
date_of_birth	date	datum narození
election_region	varchar(255)	volební kraj
election_year	int	první rok volebního období
gender	varchar(255)	pohlaví
member_from	date	datum začátku členství
member_to	date	datum konce členství
name	varchar(255)	jméno
person_id	int	identifikátor osoby
photo_url	varchar(255)	URL profilové fotky
party_election_year	int	první rok volebního období
party_party_id	int	identifikíátor poslaneckého klubu, jehož je členem

■ Tabulka A.5 Struktura member_vote

Název	Тур	Popis
result	varchar(255)	jak hlasoval poslanec
member_id	int	jak identifikátor poslance
vote_id	int	identifikátor hlasování

■ Tabulka A.6 Struktura membership

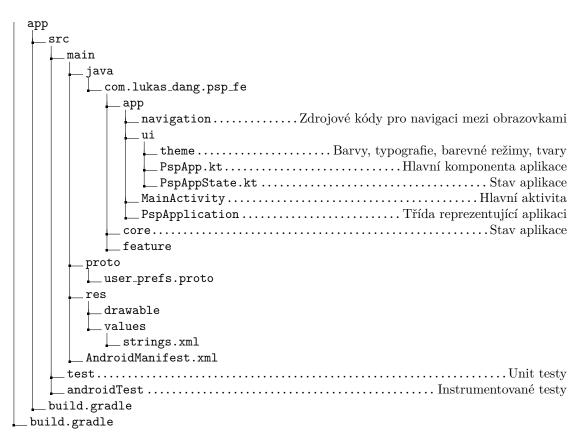
Název	Тур	Popis
end_date	datetime	datum a čas konce zařazení
agency_id	int	identifikátor orgánu
person_id	int	identifikátor osoby
start_date	datetime	datum a čas začátku zařazení

■ Tabulka A.7 Struktura party

Název	Тур	Popis
abbreviation	varchar(255)	zkratka pro název klubu
name	varchar(255)	název klubu
election_year	int	první rok volebního období
party_id	int	identifikátor klubu

Tabulka A.8 Struktura vote

Název	Тур	Popis
date_time	datetime	datum a čas hlasování
description	varchar(255)	popis hlasování
election_year	int	první rok volebního období
excused_count	int	počet omluvených
logged_off_count	int	počet nepřihlášených
meeting_number	int	bod hlasování
no_count	int	počet hlasování proti
refrained_count	int	počet zdržených
result	varchar(255)	výsledek hlasování
steno_protocol_url	varchar(255)	stenoprotokol
yes_count	int	počet hlasování pro
number	int	číslo hlasování
id	int	identifikátor hlasování



Obrázek A.17 Adresářová struktura mobilní aplikace

Bibliografie

- 1. HUŠEK, Petr; SMOLÍK, Josef. POLITICKÝ SYSTÉM A POLITICKÉ STRANY ČESKÉ REPUBLIKY. Zemědělská 1, 613 00 Brno: Mendelova univerzita v Brně, 2019. ISBN 978-80-7509-665-4.
- 2. DEVELOPER, Android Politiscope. politiscope [online]. Jan 2020. [cit. 2023-01-26]. Dostupné z: https://play.google.com/store/apps/details?id=com.junkie.android.politiscope&hl=en&gl=US.
- 3. [online]. [B.r.]. [cit. 2023-01-26]. Dostupné z: https://www.propublica.org/.
- 4. [online]. [B.r.]. [cit. 2023-01-26]. Dostupné z: https://theunitedstates.io/.
- 5. MILL, Eric. Congress [online]. Jan 2019. [cit. 2023-01-27]. Dostupné z: https://play.google.com/store/apps/details?id=com.sunlightlabs.android.congress&hl=en&gl=US
- 6. [online]. [B.r.]. [cit. 2023-01-26]. Dostupné z: https://api.open.fec.gov/developers/.

16 Bibliografie

Obsah přiloženého média

	readme.txt	stručný popis obsahu média
1	exe	adresář se spustitelnou formou implementace
1	src	
	impl	zdrojové kódy implementace
	thesis	zdrojové kódy implementace zdrojová forma práce ve formátu L ^A T _E X
1	text	text práce
	thesis.pdf	text práce ve formátu PDF