Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Vývoj webové aplikace eVlastníci

Bc. Michal Šotek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Michal Šotek

Informatika

Název práce

Vývoj webové aplikace eVlastníci

Název anglicky

Development of web application eVlastníci

Cíle práce

Hlavním cílem teoretické části práce je definovat základní pojmy a postupy z oblasti vývoje webových aplikací ve skriptovacím programovacím jazyce PHP za použití aplikačního frameworku Symfony. Dílčím cílem práce je porovnání a zhodnocení jednotlivých PHP frameworků a analýza již existujících aplikací, které se věnují hlasování společenství vlastníků jednotek. V poslední řadě pak analýza právních norem souvisejících s uskutečňováním legitimního a legálního online hlasování.

Cílem praktické části práce je vývoj webové aplikace, jež umožní hlasování společenství vlastníků jednotek online. Webová aplikace eVlastníci bude vznikat ve spolupráci s firmou Externity s. r. o. a výstup z této práce bude firmou dále využíván v praxi.

Metodika

Úvodní část práce bude založena na studiu informačních zdrojů z oblasti tvorby webových aplikací v jazyce PHP a frameworku Symfony. Dále proběhne analýza již existujících frameworků pro jazyk PHP. V další části bude proveden rozbor dosavadních aplikací, které se věnují hlasování SVJ. Na závěr teoretické části práce budou nastudovány právní normy nezbytné pro možnost online hlasování SVJ. Informace získané v teoretické části budou využity v praktické části práce pro tvorbu webové aplikace eVlastníci za spolupráce s firmou Externity s. r. o.

Doporučený rozsah práce

60-80 stran

Klíčová slova

webová aplikace, eVlastníci, SVJ, Symfony, PHP

Doporučené zdroje informací

POTENCIER, Fabien. Symfony 5: The Fast Track. 6th edition. Clichy: Symfony SAS, 2020. ISBN 978-2-918390-37-4.

STAUFFER, Matt. Laravel: Up & Running. 2nd edition. Sebastopol: O'Reilly Media, 2019. ISBN 978-1-492-04121-4.

WELLING, Luke a Laura THOMSON. Mistrovství PHP a MySQL. Brno: Computer Press, 2017. ISBN 978-80-251-4892-1.

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Dana Vynikarová, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 31. 10. 2022

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 28. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 31. 03. 2023

Čestné prohlášení

Prohlašuji, že svou diplomovou práci „Vývoj webové aplikace eVlastníci" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31. 03. 2023 _____

Poděkování

Rád bych touto cestou poděkoval své vedoucí diplomové práce Ing. Daně Vynikarové, Ph.D. za věnovaný čas při konzultacích a za cenné připomínky a odborné vedení práce.

Vývoj webové aplikace eVlastníci

Abstrakt

Diplomová práce je zaměřená na vývoj webové aplikace eVlastníci s důrazem na její backendovou část. Na úvod teoretické části je proveden rozbor právních norem, které specifikují společenství vlastníků jednotek a umožňují legitimní a legální online hlasování. Dále je provedena analýza již existujících aplikací zabývajících se stejnou tématikou. Hlavní část teoretické práce je zaměřena na skriptovací programovací jazyk PHP a analýzu frameworků Symfony a Laravel, kde bylo jako vhodné prostředí pro tento projekt vybráno Symfony. Úvod praktické části se zabývá dotazníkovým šetřením na téma hlasování SVJ, kterého se zúčastnilo přes 800 participantů. Následně jsou popsány diagramy a UX/UI design celé aplikace včetně ukázek a uživatelského testování v prototypu Figma. Závěrečná část práce je zaměřena na backendový vývoj webové aplikace, kde je popsáno její fungování, struktura a ukázky důležitých částí kódu. Výstupem práce tak je elegantní softwarové řešení, které umožňuje vlastníkům jednotek hlasovat online. Aplikace je od dubna 2022 v ostrém provozu a za tuto dobu eviduje 273 registrovaných uživatelů a 32 uskutečněných hlasování. Veškerá funkcionalita je k dispozici zcela zdarma. Tato diplomová práce vznikla ve spolupráci s firmou Externity s. r. o., kde jsem již druhým rokem zaměstnán na pozici junior vývojáře.

Klíčová slova: webová aplikace, eVlastníci, SVJ, Symfony, PHP, backend, Laravel, softwarový vývoj, online hlasování, Externity s. r. o.

Development of web application eVlastníci

Abstract

The diploma thesis is focused on the development of a web application called eVlastníci with an emphasis on its backend part. The theoretical part begins with an analysis of legal norms that specify the community of owners of units and allow legitimate and legal online voting. Further, an analysis of existing applications dealing with the same topic is carried out. The main part of the theoretical work is focused on the scripting programming language PHP and an analysis of the Symfony and Laravel frameworks, where Symfony was chosen as a suitable environment for this project. The practical part begins with a questionnaire survey on the topic of voting in owners' associations, which was participated in by over 800 participants. Subsequently, diagrams and UX/UI design of the entire application, including samples and user testing in the Figma prototype are described. The final part of the thesis is focused on the backend development of the web application, where its functioning, structure, and important parts of the code are described. The output of the work is an elegant software solution that allows owners of units to vote online. The application has been in operation since April 2022 and has 273 registered users and 32 completed votes during this time. All functionality is available completely for free. This diploma thesis was created in cooperation with the company Externity s. r. o., where I have been employed as a junior developer for the second year.

Keywords: web application, eVlastníci, COA, Symfony, PHP, backend, Laravel, software development, online voting, Externity s. r. o.

Obsah

1 Úvod

Moderní aplikace zasahují v dnešní době do téměř všech sfér lidského života. Za pomoci moderních technologií lze výrazně usnadnit lidskou práci a je možné aplikovat taková technická řešení, která byla dříve naprosto nepředstavitelná. Stále se ale najdou oblasti, kde se moderní aplikace příliš nevyužívají, ačkoliv je jejich použití extrémně výhodné, a to jak časově, finančně, tak i z hlediska lidských zdrojů.

Shromáždění společenství vlastníků jednotek se koná zpravidla jednou za rok. Značná část těchto shromáždění není usnášeníschopná, neboť se zasedání nezúčastní zákonný počet vlastníků bytových jednotek. Ovšem zasedání usnášeníschopného orgánu je klíčové a zásadní pro další rozvoj i údržbu bytových domů. Důvody, proč lidé na zasedání nechodí jsou různé. Někteří se zúčastnit z nějakého objektivního důvodu nemohou, další zkrátka na shromáždění nechodí a nechávají odpovědnost na druhých. Takové shromáždění pak nemůže hlasovat a je nezbytné uspořádat další schůzi, kde opět neexistuje jistota, že se sejde dostatečné množství členů. Může nastat situace, kdy je nezbytné připravit korespondenční hlasování, což s sebou přináší další časovou náročnost a samozřejmě i další nevyhnutelné náklady. Nahrazením klasického zasedání webovou, případně mobilní aplikací, která by takovéto hlasování zprostředkovala, by se mohlo stát velmi výhodným, efektivním a snadno dostupným nástrojem, který může celý proces zjednodušit a zrychlit. Elektronizace v dnešní době vstupuje do všech sfér dnešního života s cílem zjednodušit nastavené procesy a ušetřit čas i peníze a samozřejmě ani tato oblast nemusí být výjimkou.

I když legislativa České republiky hlasování v elektronickém prostředí za určitých podmínek již umožňuje, v současné době nejsou podobné aplikace příliš rozšířené, společenství vlastníků jednotek a jejich orgány spíše inklinují ke konzervativnímu zastaralému řešení. Roční náklady za provoz aplikace by přitom byly naprosto zanedbatelné, případně pro menší sdružení úplně nulové. Náklady na pořízení a provoz by v konečném důsledku s přihlédnutím k ušetřenému času nebyl zřejmě v podstatě žádným významným zásahem do rozpočtu. I přesto se velká část společenství podobným řešením brání, ať už argumentují finanční náročností nebo složitostí vyplývající z obav před moderními informačními technologiemi. Dnes již i starší část populace disponuje chytrým telefonem, nebo alespoň přístupem k internetu a obdobné aplikace dokáže bez problémů obsluhovat, a tudíž jsou podobné obavy neopodstatněné. U většiny těchto uživatelů půjde spíše o strach z neznámého, nežli o objektivní technickou překážku. Tento závěr ostatně koresponduje i se

skutečností, jakým způsobem se zrychlil rozvoj elektronických technologií v době pandemie COVID-19 a i jejich odpůrci se rychle a bez potíží přizpůsobili. I pro shromáždění společenství vlastníků jednotek lze pandemii považovat za jakýsi bod zvratu, kdy vyvstala potřeba zajistit fungování bez možnosti ustanovit zasedání. A právě v tomto období vznikl nápad na vytvoření aplikace eVlastníci.

Tato diplomová práce vzniká za spolupráce s firmou Externity s. r. o., kde jsem již druhým rokem zaměstnán na pozici junior vývojáře. Hlavním cílem práce je vytvořit webovou aplikaci, která umožní online hlasování společenství vlastníků jednotek. Konkrétně bude kladen důraz především na backendovou část, ačkoliv dojde ke stručnému popisu i dalších náležitostí, které byly při procesu vývoje vytvořeny.

Práce se dělí na teoretickou a praktickou část. V úvodu teoretické části jsou popsány základní charakteristiky společenství vlastníků jednotek dle příslušných zákonů a je proveden rozbor právních norem, které umožňují legitimní a legální online hlasování. Dále je provedena analýza již existujících aplikací, které online hlasování sdružení umožňují. Následuje rozbor základních pojmů a postupů z oblasti vývoje webových aplikací ve skriptovacím programovacím jazyce PHP. Na závěr teoretické části je provedena analýza největších PHP frameworků, které mohou být při vývoji použity. Praktická část práce je zaměřena na vývoj webové aplikace eVlastníci, která umožní legální a legitimní online hlasování, a to především s důrazem na vývoj backendové části aplikace.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem teoretické části práce je definovat základní pojmy a postupy z oblasti vývoje webových aplikací ve skriptovacím programovacím jazyce PHP za použití aplikačního frameworku Symfony. Dílčím cílem práce je porovnání a zhodnocení jednotlivých PHP frameworků a analýza již existujících aplikací, které se věnují hlasování společenství vlastníků jednotek. V poslední řadě pak analýza právních norem souvisejících s uskutečňováním legitimního a legálního online hlasování.

Cílem praktické části práce je vývoj webové aplikace, jež umožní hlasování společenství vlastníků jednotek online. Webová aplikace eVlastníci bude vznikat ve spolupráci s firmou Externity s. r. o. a výstup z této práce bude firmou dále využíván v praxi.

2.2 Metodika

Úvodní část práce bude založena na studiu informačních zdrojů z oblasti tvorby webových aplikací v jazyce PHP a frameworku Symfony. Dále proběhne analýza již existujících frameworků pro jazyk PHP. V další části bude proveden rozbor dosavadních aplikací, které se věnují hlasování SVJ. Na závěr teoretické části práce budou nastudovány právní normy nezbytné pro možnost online hlasování SVJ. Informace získané v teoretické části budou využity v praktické části práce pro tvorbu webové aplikace eVlastníci za spolupráce s firmou Externity s. r. o.

3 Teoretická východiska

3.1 Společenství vlastníků jednotek

Společenství vlastníků jednotek je v současném pojetí platných právních norem právnická osoba, které občanský zákoník přiznává v § 20 právní osobnost, tedy schopnost být nositelem a vykonavatelem práv a povinností, a to od svého vzniku do svého zániku. „*Právnická osoba je organizovaný útvar, o kterém zákon stanoví, že má právní osobnost, nebo jehož právní osobnost zákon uzná. Právnická osoba může bez zřetele na předmět své činnosti mít práva a povinnosti, které se slučují s její právní povahou.*“ Obecné ustanovení § 20 pak doplňuje ustanovení § 1194. „*Společenství vlastníků je právnická osoba založená za účelem zajišťování správy domu a pozemku; při naplňování svého účelu je způsobilé nabývat práva a zavazovat se k povinnostem.*“ Obligatorně se zapisuje do veřejného seznamu, kterým je rejstřík společenství vlastníků jednotek dostupný na oficiálních webových stránkách a zápis, respektive výmaz je okamžikem vzniku, respektive zániku, tedy okamžikem rozhodným pro možnost vstupovat do právních vztahů vedoucích k naplňování účelu vzniku, a to prostřednictvím svého statutárního orgánu. Účelem je zajišťování správy domu s bytovými jednotkami, případně pozemku, tedy správa cizího majetku, jehož vlastníky jsou jednotliví členové společenství. „*Správa domu a pozemku zahrnuje vše, co nenáleží vlastníku jednotky a co je v zájmu všech spoluvlastníků nutné nebo účelné pro řádnou péči o dům a pozemek jako funkční celek a zachování nebo zlepšení společných částí. Správa domu a pozemku zahrnuje i činnosti spojené s údržbou a opravou společných částí, přípravou a prováděním změn společných částí domu nástavbou, přístavbou, stavební úpravou nebo změnou v užívání, jakož i se zřízením, udržováním nebo zlepšením zařízení v domě nebo na pozemku sloužících všem spoluvlastníkům domu.*“ Platí, že členové společenství vlastníků jsou povinně všichni vlastníci jednotek v domě a členství je neoddělitelně spojeno s vlastnictvím jednotky. (Česko, 2012)

V současnosti vzniká společenství vlastníků jednotek zakladatelským právním jednáním, a to přijetím stanov všemi vlastníky jednotek v mezích a podle pravidel daných občanským zákoníkem, které musí mít povahu veřejné listiny, tedy vzniknou formou notářského zápisu. Povaha veřejné listiny není vyžadována pro stanovy, kde je zakladatelem jediná osoba. Toto právní jednání, tedy vznik společenství vlastníků jednotek je dobrovolné,

povinně zákon vyžaduje založení jen, pokud je v domě alespoň pět jednotek, z nichž alespoň čtyři jsou ve vlastnictví čtyř různých vlastníků. (Česko, 2012)

Obecný právní rámec, nezbytnosti pro vznik a zánik právnických osob, vymezení účelu, obecná ustanovení o orgánech a způsob jednání stanovuje občanský zákoník v díle 3, oddíle 1, ustanoveních § 118 až 209. Další relevantní hmotněprávní normy jsou obsaženy v oddíle 5, kde je specifikováno bytové spoluvlastnictví, společné části nemovité věci, je obsažen zákonný rámec vzniku jednotky, práva a povinnosti vlastníka jednotky, normy týkající se zcizení, tedy převodu jednotky, definice a pravidla pro správu domu a pozemku a je tak dána možnost správy domu a pozemku prostřednictvím společenství vlastníků jednotek. Vymezení pojmu společenství vlastníků a účelu, pro který mohou být založena, pravidla fungování, pravidla pro založení, orgány společenství vlastníků a způsob rozhodování jsou uzákoněny v ustanoveních § 1194 až § 1222. Tyto normy nabyly účinnosti 1.1.2014. (Česko, 2012)

V současné době existují také společenství vlastníků jednotek vzniklých před účinností nového občanského zákoníku v zákonných mezích zákona o vlastnictví bytů ve znění novely z roku 2000. Přesto, že je tento předpis již zrušen, zůstávají právní normy vůči účastníkům závazkových a vlastnických vztahů nadále účinné, a to v souladu s přechodnými ustanoveními občanského zákoníku. „*Není-li dále stanoveno jinak, řídí se ustanoveními tohoto zákona i právní poměry týkající se práv osobních, rodinných a věcných; jejich vznik, jakož i práva a povinnosti z nich vzniklé přede dnem nabytí účinnosti tohoto zákona se však posuzují podle dosavadních právních předpisů.*“ (Česko, 2012)

Společenství vlastníků jednotek vzniklých před rokem 2014, která nepřijala stanovy podle pravidel nového občanského zákoníku se i nadále řídí ve svých vnitřních poměrech normami zákona o vlastnictví bytů ve znění novely z roku 2000. I těmto právnickým osobám zákon přiznával právní osobnost, tedy přiznával jim způsobilost mít práva a povinnosti spojené se správou, provozem a opravami společných částí domu. Rozdílem oproti společenstvím vlastníků jednotek vzniklých za dnešní právní úpravy je, že nebyly zřizovány fakultativně, ale přímo zákonem, při splnění podmínky existence alespoň pěti jednotek v domě, z nichž nejméně tři jsou ve vlastnictví tří různých vlastníků. I tato společenství jsou zapsána do rejstříku společenství vlastníků jednotek vedeného soudem. (Česko, 1994)

3.1.1 Právní rámec elektronického hlasování

Rozhodování společenství vlastníků jednotek se řídí občanským zákoníkem ustanoveními § 1206 až § 1214. Zákon prioritně předpokládá hlasování nejvyššího orgánu společenství vlastníků, tedy shromáždění, které tvoří všichni vlastníci jednotek, na zasedání svolaném statutárním orgánem společenství, jiné možnosti však nejsou a priori vyloučeny. Zároveň v ustanoveních § 1211 až 1214 zákon připouští i rozhodování mimo zasedání a upravuje pravidla pro tuto situaci. (Česko, 2012)

3.1.1.1 Vymezení v občanském zákoníku

Pro stanovení právního rámce a existence zákonné možnosti elektronického hlasování společenství per rollam je zásadní ustanovení § 1211 odstavce 1 občanského zákoníku, který podmiňuje rozhodování mimo zasedání tím, že svolané shromáždění není způsobilé usnášet se. Zároveň je v druhé větě stanoveno, že v jiných případech lze navrhnout přijetí rozhodnutí mimo zasedání, pokud to připustí stanovy. Z tohoto vyplývá, že stanovy společenství vlastníků jednotek mohou upravovat širší využití rozhodování mimo zasedání, nežli pouze v případě neusnášeníschopného shromáždění. (Česko, 2012)

Občanský zákoník dále v ustanovení § 1212 stanovuje, že k platnosti hlasování se vyžaduje vyjádření vlastníka jednotky s uvedením dne, měsíce a roku, kdy bylo učiněno, podepsané vlastní rukou na listině obsahující plné znění návrhu rozhodnutí. Z tohoto ustanovení se zdá být na první pohled možnost platného elektronického hlasování vyloučena, ale je nutné posoudit jej v širším kontextu. V občanském zákoníku jsou v oddíle 3 zakotveny formy právních jednání, při čemž ustanovení § 562 odstavec 1 obsahuje normu, že písemná forma je zachována i při právním jednání učiněném elektronickými nebo jinými technickými prostředky umožňujícími zachycení jeho obsahu a určení jednající osoby. Zákonodárce tak v tomto ustanovení rozšiřuje možnosti, jakými lze učinit platné právní jednání, pro které zákon vyžaduje písemnou formu. Odstavec 2 téhož paragrafu pak stanovuje podmínky, za kterých právní jednání v elektronickém systému a priori považuje za spolehlivé, tedy provádějí-li se záznamy údajů systematicky a posloupně a jsou-li chráněny proti změnám. (Česko, 2012)

3.1.1.2 Vymezení v zákoně o službách vytvářejících důvěru pro elektronické transakce

Zákon č. 297/2016 Sb., o službách vytvářejících důvěru pro elektronické transakce upravuje Nařízení Evropského parlamentu a Rady (EU) č. 910/2014, o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu (dále jen „nařízení eIDAS“), tedy předpis vzniklý na půdě Evropské unie mající přímou platnost a závaznost pro použití ve svých členských státech. Tento zákon v ustanoveních § 5 až § 6 upravuje elektronické podepisování dokumentů, při čemž rozlišuje mezi úkonem, který činí veřejnoprávní subjekt nebo je činěn vůči veřejnoprávnímu subjektu a úkony jiné, tedy mezi soukromoprávními osobami, kde umožňuje v ustanovení § 7 použít zaručený elektronický podpis, uznávaný elektronický podpis, případně jiný typ elektronického podpisu. Zároveň jsou stanovena pravidla pro poskytování služeb, jejichž prostřednictvím lze získat certifikát k používání zaručeného a uznávaného elektronického podpisu. S tímto způsobem podpisu lze jednoznačně spojit osobu, která jej činí, protože jsou stanoveny nároky na vytváření a zabezpečení těchto podpisů. (Česko, 2016; Nařízení Evropského parlamentu a Rady, 2014)

Definice pojmů jsou obsaženy ve výše uvedeném nařízení eIDAS, při čemž pouhým elektronickým podpisem se rozumí data v elektronické podobě, která jsou připojena k jiným datům v elektronické podobě nebo jsou s nimi logicky spojena a která podepisující osoba používá k podepsání. Z této definice je patrné, že i běžnou e-mailovou komunikaci s uvedením dat, tedy například jména a příjmení osoby, lze považovat za podepsaný elektronický dokument, ačkoli nenaplňuje požadavky kladené na vyšší formy elektronických podpisů. V důvodové zprávě k zákonu č. 297/2016 Sb., o službách vytvářejících důvěru pro elektronické transakce se uvádí, že v případě soukromoprávních jednání je možné použít všechny typy elektronických podpisů, které nařízení eIDAS zná, tj. elektronický podpis, zaručený elektronický podpis, zaručený elektronický podpis založený na kvalifikovaném certifikátu pro elektronický podpis nebo kvalifikovaný elektronický podpis. Zároveň je uvedeno: „*Zákon tak rozšiřuje paritu s vlastnoručním podpisem i na tyto typy elektronických podpisů.*“ Z tohoto vyplývá, že v soukromoprávních vztazích mají všechny typy elektronických podpisů právní účinky, které jim nemohou být upírány. Ostatně ani hlasování společenství vlastníků jednotek per rollam v klasické listinné podobě neobsahuje zákonný požadavek opatřit listinu úředně ověřeným podpisem hlasující osoby, proto lze vztáhnout analogii i na elektronický podpis bez vyšší formy kvalifikace nebo záruky. (Česko, 2016; Nařízení Evropského parlamentu a Rady, 2014)

3.1.1.3 Shrnutí

Protože se při výkonu hlasovacích práv v rámci rozhodování společenství vlastníků jednotek vymezuje oblast soukromého práva, vyplývá z výše uvedeného závěr, že elektronické hlasování per rollam zákon nevylučuje. Ostatně tento závěr plně koresponduje s dnešním fungováním společnosti obecně, kde je kladen důraz na elektronizaci, a tudíž i zefektivnění procesů, ať už v soukromé či veřejné sféře. Písemná forma tak zůstává při splnění stanovených technických podmínek zachována, taktéž i účinky vlastnoručního podpisu. Překážkou, kterou však lze snadno odstranit, může být, že SVJ bude muset přijmout úpravu svých stanov tak, aby rozhodování mimo zasedání umožňovaly. (Česko, 2012; Nařízení Evropského parlamentu a Rady, 2014)

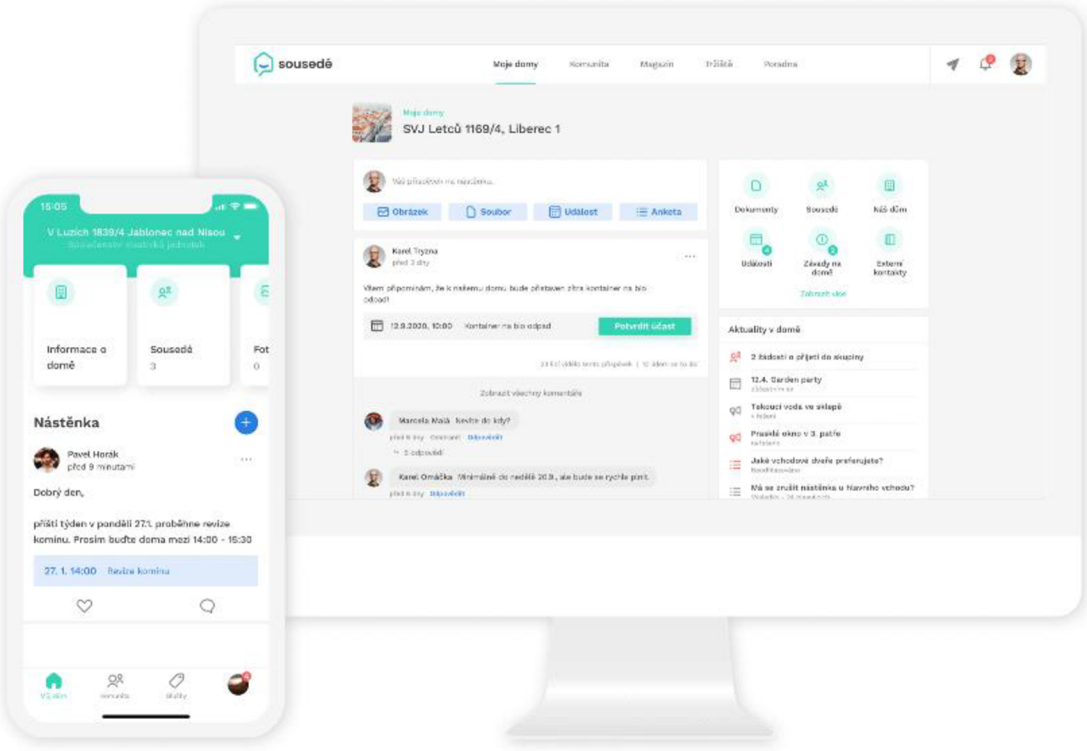
Aplikace bude navržena tak, aby bylo právní jednání vlastníka při elektronickém hlasování co nejvíce prokazatelné a bylo možné jej zpětně ověřit. Vlastník při registraci uvede své osobní a kontaktní údaje, které budou následně zpětně verifikovány. Verifikace bude probíhat i při samotném hlasování v aplikaci. Postup bude obdobný jako například při verifikaci transakce v bankovním prostředí.

3.2 Analýza existujících řešení

Online aplikace jsou využívány v téměř všech oblastech lidského života, a proto se i výbory SVJ čím dál více zajímají o taková moderní řešení. Možnosti online hlasování se začaly do povědomí dostávat především s příchodem pandemie COVID-19, kdy nebyla umožněna účast velkého množství lidí na schůzích a sdružení začala hledat jiné cesty, jak hlasování uskutečnit. Online hlasování s sebou přináší i výhody jako jsou pohodlí, efektivita a menší časová náročnost. Aplikace pro online hlasování SVJ jsou založeny na principu hlasování per rollam, které je podrobně popsáno v sekci 3.1.1. Přestože již vzniklo několik aplikací, které se touto problematikou zabývají, ať už jako primární cíl, nebo jako dílčí nástroj aplikace, jež má prioritně jiný smysl, existuje na trhu značný prostor pro nové projekty. Konkrétně existují v České republice 3 největší aplikace, které jsou Sousedé, SVJ Aplikace a SV online. Většina SVJ stále hlasuje fyzicky na schůzích, jež se konají zpravidla jednou za rok. (Sousedé, c2023; SVJ Aplikace, c2018-2023; SV Online, c2021)

3.2.1 Sousedé

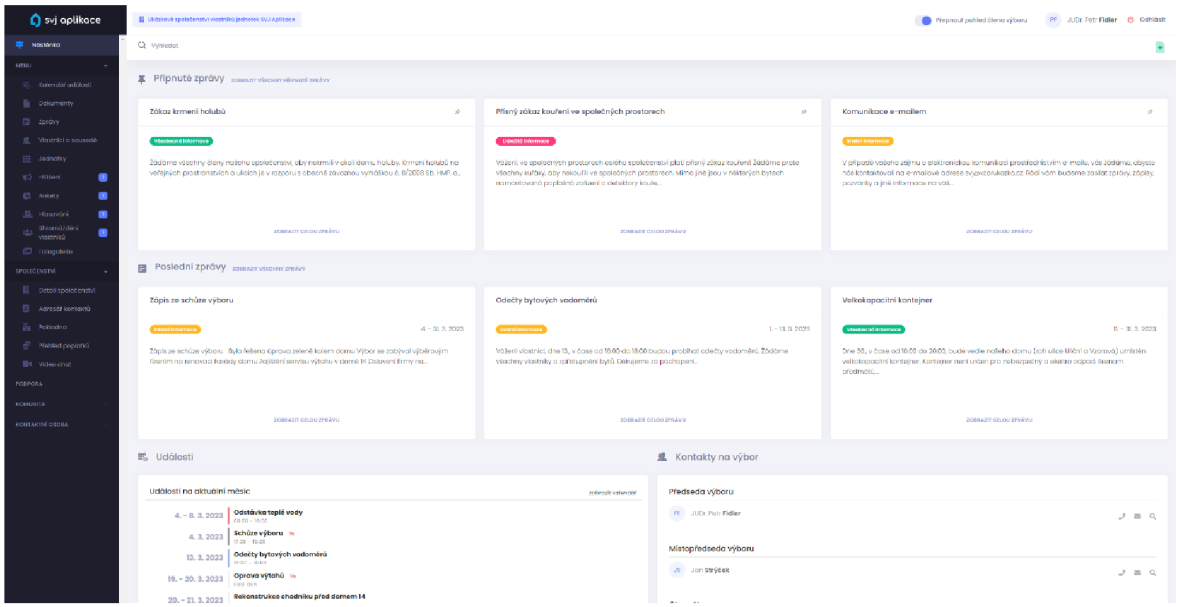
Aplikace Sousedé patří k nejrozšířenějšímu nástroji pro online hlasování v České republice, ačkoliv samotné online hlasování nebylo původním cílem aplikace a bylo přidáno teprve před několika lety. V počátcích sloužili Sousedé prioritně jako komunitní nástroj pro komunikaci a sdílení souborů v rámci SVJ. Vznikli již v roce 2015 a vytvořila je stejnojmenná společnost Sousedé s. r. o. Kromě samotného online hlasování nabízí i celou řadu dalších funkcí a nástrojů pro zlepšení procesů a komunikace v rámci bytových jednotek. Umožňují například vyhledávání úklidových služeb, účetního poradenství, pojištění a dalších podobných činností, ačkoliv jejich nabídka je značně omezená. Dle oficiálních stránek využívá tuto aplikaci přes 2 000 bytových domů a více než 45 000 uživatelů. K přístupu do aplikace je možné využít jak webový prohlížeč, tak mobilní aplikaci, která je dostupná na obě hlavní platformy iOS a Android. Podle recenzí na Googlu mají sousedé hodnocení 3,2 z 5, což značí, že má aplikace značné rezervy a nedostatky. Základní cena pro bytovou jednotku za rok je 180 kč, nad rámec se dají pořídit i extra služby, jako jsou ověření exekucí, hlídání insolvence, nebo třeba rozšířené uložiště pro dokumenty. Tyto doplňkové služby jsou ovšem dostupné pouze za příplatky. (Sousedé, c2023)



Obrázek 1 – Aplikace Sousedé (Sousedé, c2023)

3.2.2 SVJ Aplikace

Druhou největší aplikaci, která se zabývá touto problematikou je SVJ Aplikace. Dle oficiálních stránek má aplikace aktuálně více než 1 000 společenství a více než 29 000 registrovaných uživatelů. Obdobně jako Sousedé nabízí SVJ Aplikace pro své uživatele nástěnku, kalendář událostí, sdílení dokumentů, katastr nemovitostí, nebo právě online hlasování. Největší rozdíl je v ceně služby. Zatímco Sousedé mají zpoplatněnou již základní verzi, SVJ Aplikace umožňuje využití verze zdarma, ta ovšem obsahuje pouze základní nástroje. Pokud by chtěli uživatelé možnost online hlasování, museli by si za aplikaci zaplatit 190 kč na rok za bytovou jednotku. Další značnou výhodou oproti konkurenci je možnost vyzkoušet si demo verzi na oficiálních stránkách. Případní zájemci si tak mohou již dopředu zkusit kompletní fungování aplikace, a to bez jakýchkoliv závazků, nebo registrace. Ačkoliv je webová aplikace plně responzivní a podporuje tak všechny druhy zařízení, nemá svojí vlastní mobilní aplikaci, kterou by si mohli uživatelé stáhnout přímo do svého zařízení, což může mít negativní dopad na pohodlí uživatele. (SVJ Aplikace, c2018-2023)

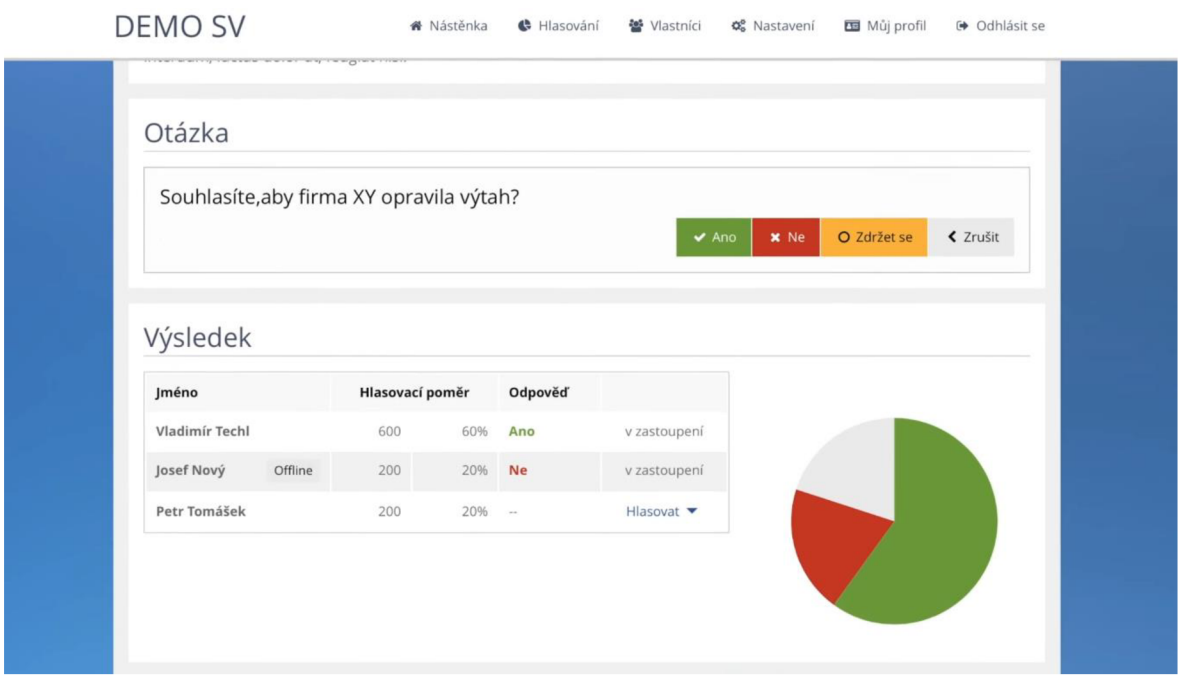


Obrázek 2 – SVJ Aplikace (SVJ Aplikace, c2018-2023)

3.2.3 SV online

Aplikace SV online je primárně zaměřena pouze na online hlasování SVJ. Nenabízí tak příliš rozsáhlou řadu funkcí jako předchozí projekty a v aplikaci se nachází pouze

nástěnka a sekce pro hlasování. Díky tomu může pro uživatele působit více přívětivěji, neobsahuje tolik rušivých elementů a uživatelé se mohou primárně zaměřit pouze na důležité části obsahu. Roční licence pro SVJ stojí 2 500kč, a navíc se platí 10kč za každý hlas uživatele. K dispozici je pouze responzivní webová aplikace stejně jako u SVJ Aplikace. Údaje o rozsahu klientely nejsou na oficiálních stránkách k dispozici. (SV Online, c2021)



Obrázek 3 – SV online (SV Online, c2021)

3.2.4 Shrnutí

Ačkoliv se na trhu nachází několik aplikací, které umožňují online hlasování, je zde značný prostor pro nové projekty, neboť nejsou příliš rozšířené. Všechny aplikace jsou zpoplatněny, nicméně každá z nich využívá jiný systém plateb. Největší uživatelskou základnu mají Sousedé a umožňují také největší rozsah funkcionalit, ačkoliv SVJ Aplikace obsahuje také velké množství nástrojů, výhodou je především možnost vyzkoušet aplikaci v demo verzi a zároveň umožňuje využití verze zdarma, ta je ale značně omezená. SV online oproti tomu nenabízí velké množství nástrojů, ale soustředí se pouze na online hlasování. Nicméně je zde nutná platba za každý odeslaný hlas a díky tomu může být aplikace značně finančně náročná. (Sousedé, c2023; SVJ Aplikace, c2018-2023; SV Online, c2021)

3.3 Jazyk PHP

PHP je univerzální skriptovací jazyk na straně serveru, který byl navržen speciálně pro tvorbu webových stránek. Ačkoliv se jedná o poměrně starý jazyk, je na něm založena naprostá většina dnešních moderních webových stránek. Základem téměř všech webů je HTML stránka, do které lze vkládat kód jazyka PHP, jenž se provede při každém obnovení této stránky. PHP je navíc open source, což umožňuje přístup k jeho zdrojovému kódu, který lze používat, měnit a distribuovat. Původní význam zkratky PHP bylo Personal Home Page, ale postupně se začal využívat název PHP Hypertext Preprocessor. Aktuální verzí je PHP 8.2. (Welling, 2017; Achour, c2001-2023)

V dnešní době existuje značné množství produktů, které umožňují vytvářet webové stránky. Nejdůležitějšími aspekty, které je potřeba zohlednit při výběru vhodného prostředí jsou použitý hardware, operační systém, webový server a databázový systém. Tento výběr může být poněkud složitý, neboť některé části spolu nemusí být kompatibilní. Jazyk PHP má ovšem hlavní výhodu v tom, že funguje na všech hlavních operačních systémech, a to dokonce i na velkém množství méně rozšířených. Zároveň je kód přenositelný mezi operačními systémy. Většinu kódu lze tedy napsat tak, aby bylo možné ho použít pro jakýkoliv operační systém, nebo webový server. (Welling, 2017)

Jedná se o relativně starý jazyk, který začal vznikat již v roce 1994. Na začátku vývoje stál přitom jen jeden člověk, kterým byl Rasmus Lerdorf. K jeho práci se následně přidávali další autoři a jazyk do značné míry přepsali, než z něj vznikl tak rozšířený produkt, jakým je PHP dnes. V roce 2013 na něm bylo založeno více než 75 % všech webových stránek. Toto číslo stále stoupá a již v roce 2016 vzrostl počet na 82 %. Mezi nejznámější webové stránky založené na tomto jazyce patří: Facebook, Yahoo, Wikipedia, Wordpress a celá řada dalších. PHP tak rozhodně není „mrtvý“ jazyk a alespoň základní znalosti jsou nezbytné pro každého programátora webových stránek. (Welling, 2017)

3.3.1 Přednosti PHP

Hlavními konkurenty jazyka PHP jsou Python, Ruby, Perl, Java, Node a .NET. Následující seznam vypisuje hlavní výhody PHP oproti konkurenčním prostředím.

- Výkon – Jedna z hlavních předností jazyka PHP je jeho výkon. Díky vysoké rychlosti je možné obsluhovat miliony návštěv webové stránky denně pouze za pomoci

jediného levného serveru. Zvládne přitom zpracovávat jak malé stránky s několika formuláři, tak složité stránky s obrovským množstvím dat, jako je například Facebook.

- Škálovatelnost – Škálovatelnost lze nazvat také jako rozšiřitelnost a jedná se tak o schopnosti systému pracovat s náhlými změnami. Díky tomu je možné PHP levně a efektivně implementovat do velkého množství webových serverů.
- Integrace s databází – Další velmi zásadní výhodou je možnost nativní podpory většiny databázových systémů. PHP tak umožňuje připojit databáze MySQL, PostgreSQL, Oracle, MongoDB, MSSQL, nebo třeba SQLite. PHP také podporuje standard Open Database Connectivity, jenž dává možnost připojit se k téměř jakémukoliv databázovému systému.
- Vestavěné knihovny – Primárním úkolem jazyka PHP bylo použití na webu a tomu je také přizpůsobena většina vestavěných funkcí, které umí vykonávat užitečné webové úlohy. Pomocí několika řádků kódu tak lze docílit dynamického generování obrázků, odesílání e-mailů, parsovat kód jazyka XML, pracovat s cookies, nebo například generovat PDF dokumenty.
- Učení – Syntaxe jazyka PHP je velice podobná jiným obdobným programovacím jazykům zejména pak C a Perl. Uživatel, který zvládá práci s těmito jazyky, dokáže být téměř okamžitě produktivní i v PHP.
- Objektově orientované programování – Od verze 5 je možné využívat navržené objektově orientované prvky, které byly následně verzí 7 ještě značně vylepšeny. Je tak možné využívat dědičnost, soukromé a chráněné vlastnosti a metody, konstruktory a destruktory, iterátory a traity, nebo třeba abstraktní třídy a metody.
- Přenositelnost – Jak již bylo zmíněno, PHP je k dispozici na většinu operačních systémů. Lze jej tedy využít na operačních systémech Windows od Microsoftu, na OS X a na volně dostupných unixových systémech jako jsou Linux, nebo FreeBSD.
- Zdrojový kód – Jelikož je jazyk PHP open source, umožňuje nahlédnout a případně i upravit zdrojový kód. Kdokoliv tak může kód upravit a není nutné čekat na další kroky výrobce.
- Dokumentace a podpora – Kompletní dokumentace je k dispozici zdarma na stránkách výrobce a stejně tak i kontakt na podporu. Jelikož je ale jazyk PHP velice rozšířený, je možné v případě problému zažádat o pomoc i komunitu na některém z diskuzních fór. Reakce bývá většinou rychlá a věcná.

- Cena – Poslední zásadní výhodou PHP je jeho cena. Je k dispozici zdarma a kdokoliv si jej může stáhnout přímo z oficiálních stránek PHP. (Welling, 2017)

3.3.2 Základní syntaxe

Syntaxe jazyka PHP je velice jednoduchá a vkládá se přímo do HTML kódu. Aby bylo možné rozpoznat, o kterou část kódu se jedná, odděluje se PHP kód značkami „<?php“ pro začátek a „?>“ pro konec PHP části. Když dochází ke zpracování skriptu, tyto části kódu jsou vyjmuty z HTML, zpracovány a až následně převedeny zpět a zobrazeny na stránce. Existuje ale i opačný postup, kdy základní soubor je psán pouze v jazyce PHP a HTML část je načítána ze separovaných souborů, které se nazývají šablony. Tyto šablony jsou pak psané výhradně v HTML. (Kruliš, 2009)

3.3.2.1 Definování proměnných

Proměnné se v PHP definují pomocí znaku „$“. Po uvedení znaku následuje identifikátor proměnné, který zpravidla obsahuje pouze písmena a čísla. Výhodou PHP je, že se proměnné nemusí deklarovat zvlášť, ale definují se při prvním přiřazení do proměnné. Pokud nebyla proměnná deklarována, nelze ji použít ani uvnitř výrazu. Při práci s proměnnými lze využít funkce „isset“ a „unset“. První z nich ověří, zda je proměnná deklarována a druhá naopak proměnnou odstraní. (Kruliš, 2009)

3.3.2.2 Datové typy

Stejně jako v obdobných skriptovacích jazycích existují v PHP čtyři základní skalární typy. První z nich je „string“, tedy řetězec obsahující libovolný počet znaků. Dalším je „integer“, což je celé číslo, jehož délka je závislá na platformě, ale většinou bývá omezen na 32bitovou hodnotu. Následuje další číselný datový typ „float“, který je na rozdíl od „integeru“ reálným číslem s plovoucí desetinnou čárkou. Posledním skalárním typem je „boolean“, tedy proměnná, která nabývá pouze hodnot „true“ a „false“. (Kruliš, 2009)

Mimo skalární datové typy existují také dva složené typy. Těmi jsou pole a objekty. Pole jsou v PHP dynamická a dokáží uchovávat více hodnot v jedné proměnné. Mohou tak obsahovat základní skalární typy, a dokonce do nich lze uložit i jiné pole. Objekty jsou vytvářeny pomocí tříd a mají své vlastnosti a metody. Tyto vlastnosti jsou uloženy jako

proměnné a metody jsou funkce, které se mohou volat na objektech. Každý objekt vytvořený z třídy je instance této třídy. (Kruliš, 2009)

Na závěr existují ještě dva speciální datové typy. Prvním z nich je „resource“, který představuje odkaz na externí zdroj dat. Těmi mohou být různé typy souborů, databáze, nebo třeba obrázky. Posledním datovým typem je „null“. Zajímavé na tomto typu je, že může nabývat pouze jediné hodnoty. Jak už napovídá název datového typu, touto hodnotou je „null“, což v praxi znamená, že proměnná nemá žádnou hodnotu. (Kruliš, 2009)

3.3.2.3 Přetypování

V jazyce PHP je možné provádět přetypování v průběhu skriptu, což znamená dynamicky měnit proměnnou na jiný datový typ, než byla původně. Přetypování probíhá implicitně. Není tedy nutné zadávat speciální příkaz, ale stačí proměnné přiřadit jinou hodnotu. Přetypování má ovšem svá pravidla, a ne všechny datové typy lze převést na jiné. Nejjednodušší práce je u převádění na „boolean“, neboť na ten je možné přetypovat všechny ostatní datové typy. Pravidla jsou přitom poměrně jednoduchá. Hodnoty „null“, 0, prázdný řetězec a prázdné pole se převádí jako „false“, vše ostatní pak nabývá hodnoty „true“. (Kruliš, 2009)

3.3.2.4 Operátory

Operátory v PHP umožňují provádět širokou škálu úkonů jako jsou aritmetické výpočty, přiřazení proměnné, vyhodnocování podmínek, nebo třeba logické operace. Kompletní výčet operátorů by byl extrémně zdlouhavý, proto budou zmíněny jen ty nejdůležitější. Základní aritmetické operátory jsou: „+, -, *, /, %“ a mají stejné významy jako u většiny ostatních jazyků. Pro přiřazení do proměnné se používá operátor „=“. Navíc PHP umožňuje přiřazení jedné hodnoty do více proměnných pomocí jednoho zápisu „$a = $b = $c = 1;“. V tomto případě budou všechny tři proměnné nastaveny na hodnotu 1. Logické operátory jsou: „AND, OR, XOR, NOT“ a značí se: „&, |, ^, ~“. Pro „boolovské“ hodnoty se využívá stejné operátory mimo „XOR“ ale jejich značení se u „AND“ a „OR“ dvojí tedy: „&&, ||“ a pro negaci se používá „!“. Poslední operátory, které stojí za zmínku, jsou operátory pro vyhodnocování podmínek. Rovnost a nerovnost se značí: „==, !=“, identita a neidentita: „===, !==“ a pro porovnávání: „<, >, <=, >=“, které znamenají v tomto pořadí menší, větší, menší nebo rovno, větší nebo rovno. Jak již bylo zmíněno PHP obsahuje značné

množství operátorů a jejich kompletní výčet lze nalézt přímo na oficiálních stránkách v sekci dokumentace. (Kruliš, 2009)

3.3.2.5 Řetězce

Řetězce se v PHP používají k reprezentaci textových dat. Pomocí nich lze uchovávat a manipulovat s informacemi, jako jsou jména, adresy, čísla atd. Kromě toho se řetězce dají použít i ke generování HTML a XML formátů za pomocí šablon. Přímo v řetězci je také možné deklarovat proměnné, nebo třeba zpracovávat data z webového formuláře. K zápisu se používají apostrofy a všechny znaky uvnitř jsou považovány za znaky řetězce. (Kruliš, 2009)

3.3.2.6 Podmínky

Podmínky, nebo také řídící struktury, slouží v PHP k řízení toku programu. Existují dva typy řídících struktur, a těmi jsou příkazy a cykly. Neexistuje způsob, jak určit, která z podmínek je nejlepší. Vše závisí na konkrétním problému, přičemž by mělo být vybráno takové řešení, které je pro danou situaci nejlépe čitelné. Volba tak závisí převážně na zkušenostech programátora. (Welling, 2017)

3.3.2.6.1 Příkazy

Základní konstrukce, pomocí které dokáže program rozhodovat se nazývá podmíněný příkaz. Nejjednodušším a zároveň nejvyužívanějším příkazem je „if“. Ten umožňuje zadat podmínku, která je následně vyhodnocena a pokud bude pravdivá, provede se operace uvnitř bloku této podmínky. Pokud je ovšem nutné vykonat jinou operaci, v případě že je podmínka vyhodnocena jako nepravdivá, využije se příkaz „else“. PHP také podporuje vnořování příkazů do sebe, pokud tedy bude první příkaz „if“ vyhodnocen jako pravda, může následovat další podmínka. K využití více možnosti než pravda a nepravda slouží příkaz „elseif“. V takovém případě prochází program všechny možnosti a hledá první podmínku, která je vyhodnocena jako pravda. Obdobně lze využít i příkaz „switch“, který umí vyhodnocovat více možností za předpokladu, že jsou jednoduchého datového typu (textový řetězec, celé, nebo desetinné číslo). (Welling, 2017)

3.3.2.6.2 Cykly

Základní myšlenkou cyklu je opakování určitého úkonu, dokud nenastane požadovaná situace. Nejjednodušším cyklem, který existuje v PHP, je „while“. Principiálně funguje stejně jako příkaz „if“, tedy vyhodnocuje danou podmínku. Rozdíl spočívá v tom, že když je podmínka pravdivá, cyklus provede akci uvnitř bloku a vrátí se zpět na začátek a vyhodnocuje znovu, dokud je podmínka stále pravdivá. Obdobným cyklem je také „for“. Obě varianty fungují stejně, ale zápis pomocí „for“ může být o něco kompaktnější a ušetřit několik řádků kódu v závislosti na složitosti problému. Nicméně neexistuje návod, který z nich zvolit a vše opět záleží na samotném programátorovi. Jako poslední existuje příkaz „do…while“. Ten již funguje trochu odlišně. Na rozdíl od dvou předchozích, tento cyklus vyhodnocuje podmínku až na konci iterace. Cyklus tedy provede požadovaný úkon, na konci vyhodnotí podmínku a pokud je tato podmínka pravdivá, dochází k jeho opakování. (Welling, 2017; Kruliš, 2009)

3.3.3 Ukládání dat

V PHP existují dva základní způsoby, jak ukládat data. Triviálnější způsob je pomocí prostého souboru. Tento způsob je sice lehčí, ale do jisté míry také značně omezující. Data se většinou ukládají do textového souboru, kdy se na každém řádku nachází jeden zápis. Při práci s větším množství dat je žádoucí využít některý ze systémů pro správu databází. (Welling, 2017)

3.3.3.1 Ukládání pomocí souborů

Ukládání, potažmo načítání dat ze souboru probíhá zpravidla ve třech krocích. Nejprve dochází k otevření souboru, následně jsou data zapsána, případně načtena ze souboru a na závěr je soubor zavřen. Při otevírání souboru je nutné specifikovat, co se bude se souborem dále provádět. Následně je nezbytné ověřit, zda má uživatel práva k užívání souboru daným způsobem. Otevření probíhá pomocí funkce „fopen“. Krom lokálních soborů je navíc možné otevírat i soubory vzdálené za pomocí protokolů FTP a HTTP. Samotné zapisování do souboru je poměrně snadné a využívá se k němu funkce „fwrite“. Po dokončení úprav souboru je nezbytné soubor také řádně zavřít. K tomu slouží funkce „fclose“, která vrací hodnotu „true“, pokud byl soubor řádně zavřen, případně „false“ pokud nebyl. (Welling, 2017)

Užívání prostého souboru může mít své opodstatnění v případě ukládání malého množství dat. S narůstajícím počtem těchto dat se ale začnou objevovat jeho hlavní slabiny. Čím více dat v něm bude, tím pomalejší bude jeho zpracování. Dále je velice obtížné v něm vyhledávat konkrétní záznamy. Soubory se špatně vypořádávají s větším množství procesů najedou. V případě sdíleného souboru se tak může stát, že se jej bude snažit otevřít větší množství uživatelů najednou a začne docházet ke dlouhým čekacím dobám. Soubor je navíc zpracováván sekvenčně, tedy čte data ze začátku na konec. Upravování záznamu, který se nachází uprostřed souboru, může být složité a může představovat značné režijní náklady. (Welling, 2017)

3.3.3.2 Ukládání pomocí databáze

Téměř všechny problémy ukládání dat do souborů lze vyřešit využitím systému pro správu relačních databází. Hlavní výhodou je mnohem rychlejší a efektivnější přístup k datům. Vyhledávání je možné pomocí dotazování, kdy relační databáze najde pouze ty data, která splňují daná kritéria. Přístup k datům je náhodný, není tedy nutné procházet data sekvenčně. Navíc obsahují databázové systémy vestavěný systém oprávnění, který umožní autorizaci uživatele. (Welling, 2017)

3.3.4 MySQL

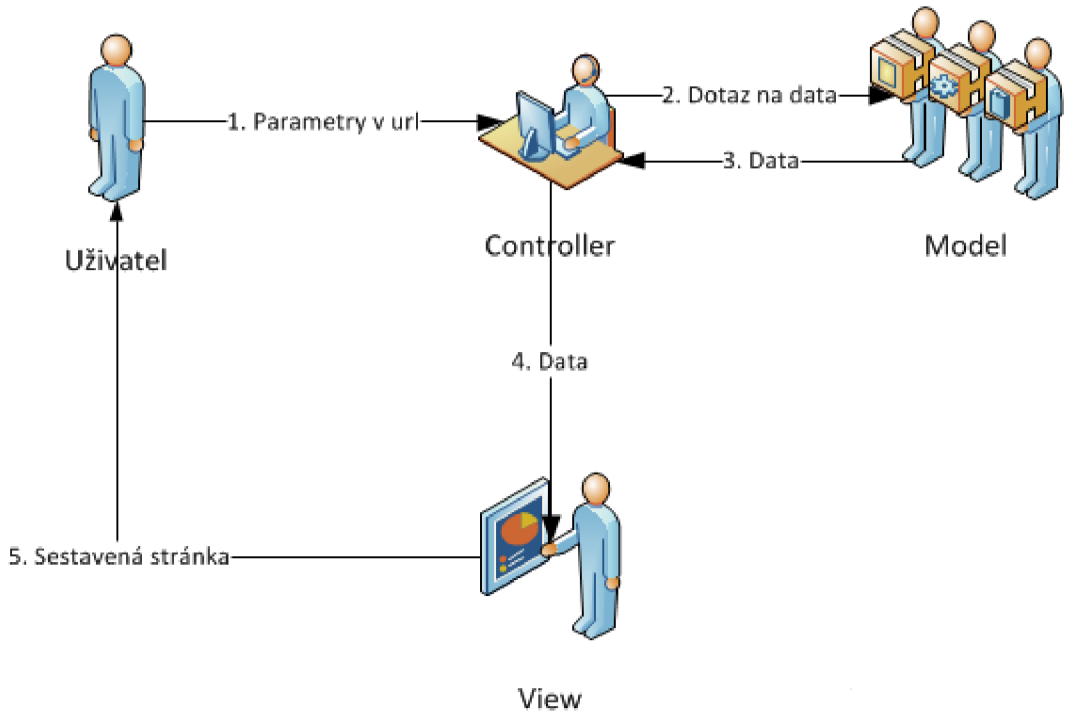
Ačkoliv existuje velké množství systémů pro správu databází, ve spojení s PHP je MySQL zdaleka nejrozšířenější a nejefektivnější databáze s otevřeným zdrojovým kódem. Počátky jeho vývoje sahají již do roku 1979, ale veřejně dostupné bylo až od roku 1996. Původně stála za projektem malá švédská firma MySQL AB, nyní jej vlastní společnost Oracle Corporation. Za hlavní autory jsou považováni Michael Widenius a David Axmark. Na databázích MySQL fungují největší organizace světa, jako jsou Facebook, Twitter, Booking, Intel a řada dalších. (Oracle, 2023)

Databáze MySQL umí celou řadu funkcí, jako jsou vyhledávání, ukládání, řazení a načítaní dat. Přístup k datům zajišťuje server. Díky tomu může k datům přistupovat velké množství uživatelů současně a zároveň jsou ověřeny jejich oprávnění. Kromě volně dostupné verze s otevřeným zdrojovým kódem existuje také placená komerční verze. Mezi hlavní přednosti databáze patří vysoký výkon, nulová cena, snadné použití, neboť většina databází využívá shodně jazyk SQL a přenositelnost v rámci operačních systémů. (Oracle, 2023)

Základem relační databáze jsou tabulky, kdy každá tabulka má svůj vlastní název. V tabulkách se pak vyskytují sloupce a řádky. Sloupce tabulky reprezentují proměnné a obsahují jedinečný název, datový typ a případné další atributy, pokud jsou nezbytné. Řádky představují záznamy konkrétních dat. Každý záznam v tabulce má svůj jednoznačný identifikátor, kterým bývá většinou ID. K identifikaci sloupců v tabulce slouží tzv. klíče. Ty se dále dělí na primární a cizí. Primární klíč je identifikátorem jedné tabulky, pokud bude ale pomocí tohoto klíče propojena další tabulka, v té bude představovat cizí klíč. Díky tomu je možné propojovat tabulky a zároveň určit, která data z jedné tabulky souvisí s daty z jiné tabulky. (Oracle, 2023)

3.3.5 Architektura MVC

Jak již název napovídá, vzor se skládá ze tří základních komponent: „model, view, controller“. Ačkoliv původně vznikl pro desktopy, dnes se hojně využívá zejména při tvorbě webů. Lze jej charakterizovat jako softwarovou architekturu, která rozděluje datový model aplikace, uživatelské rozhraní a řídící logiku takovým způsobem, kdy úprava jedné z těchto části nemá vliv, nebo má jen minimální, na jakoukoliv další část aplikace. Základní myšlenkou architektury je tedy oddělení logiky od výstupu. To pomáhá zvýšit přehlednost kódu, jeho psaní a snadnější udržování a rozšiřování. (Čápka, 2016)



Obrázek 4 – Architektura MVC (Čápka, 2016)

3.3.5.1 Model

Model v architektuře reprezentuje veškerou logiku aplikace. Řadí se sem výpočty, databázové dotazy, validace atd. Jelikož je architektura rozdělena, model se vůbec nestará o výstupy, ale pouze přijímá parametry a následně vydává data. Model se ani nezajímá o to, odkud data v parametrech pochází, nebo jak budou následně formulována a vypsána. V případě využívání objektově relačního mapování modely přímo korespondují s databázovými tabulkami. (Čápka, 2016)

3.3.5.2 View

Tuto vrstvu je možné přeložit do českého jazyka jako zobrazení. Jedná se o část, která se stará o zobrazení výstupu uživateli. Zpravidla to bývá PHTML šablona, která obsahuje stránku HTML a tagy značkovacího jazyka, díky kterému je možné do šablony vkládat proměnné, provádět iterace a podmínky. Jelikož je PHP přímo značkovacím jazykem, je umožněn takový styl zápisu, kdy bude zachována struktura stránky HTML. Stejně jako v modelu se zobrazení vůbec nestará o to, odkud data přišla, ale soustředí se výhradně na jejich promítnutí uživateli. (Čápka, 2016)

3.3.5.3 Controller

Jedná se o mezivrstvu, která izoluje zobrazení od modelu a tím umožňuje, aby se model vyvinul nezávisle na zobrazení. Kromě zobrazení a modelu s ním komunikuje i samotný uživatel a drží celý systém pohromadě. Při vznesení požadavku je vybrán kontroler, který voláme. Ten následně pozná, jaká akce se po něm chce a předá požadavek modelu. Model provede daný úkon a vrátí kontroleru požadovaná data. Ten následně předá data do zobrazení, které je vloží do připravené šablony a kompletní stránku promítne uživateli. (Čápka, 2016)

3.4 Analýza PHP frameworků

Frameworky jsou nástroje pro vývoj aplikací, díky kterým mají vývojáři možnost efektivnějšího vývoje za použití menšího úsilí. Součástí frameworků jsou implementované sady knihoven a nástrojů, které umožňují vyhnout se duplicitní práci a zaměřit se na specifické funkce aplikace. Využívání frameworků je běžnou praxí a více aplikací již vzniká

za jejich pomoci nežli bez nich. Výběr vhodného frameworku může být značně komplikovaný, neboť každý nabízí specifické funkce a je zaměřen na lehce odlišný typ aplikací. Příkladem může být framework CodeIgniter, který je poměrně jednoduchý a vhodný pro menší aplikace. Naopak pro komplikovanější projekty lze využít rozsáhlejší frameworky, jako jsou Symfony, nebo Laravel. (Morris, 2023)

PHP je velmi vysoký jazyk, který má sám o sobě implementovanou celou řadu užitečných funkcí. I přesto obsahuje značné mezery, s některými funkcemi se špatně pracuje, případně v něm chybí úplně. Většina programátorů v čistém PHP tak po několika projektech zjistí, že téměř v každém z nich je nezbytné vytvořit celou řadu základních funkcí, které jsou prakticky totožné. V takovém případě je vhodné sáhnout po některém z frameworků, který již tyto základní funkce obsahuje. Díky tomu je možné ušetřit až 50 % kódu a tedy i 50 % času programátora. Navíc práce nebude tak stereotypní a programátor se u psaní kódu bude více bavit. Následující analýza se zaměří pouze na nejrozšířenější a největší frameworky Symfony a Laravel, neboť právě ty jsou vhodné pro takto složitý projekt. (Máca, c2023)

3.4.1 Symfony

Symfony je aktuálně jedním z nejpopulárnější „full-stack“ PHP frameworků, který obsahuje sadu komponent, jenž zásadně usnadňují tvorbu webových aplikací. Využívá se jak u menších projektů, tak pro obrovské aplikace ve velkých korporátních firmách. Aktuální verzí je Symfony 6.2, která byla vydána 1. února 2023, první verze však vznikla již v roce 2005. Jedná se o „open-source“ projekt, jehož vývoj je sponzorován francouzskou firmou Sensio Labs. Podobně jako ostatní PHP frameworky je založen na architektonickém vzoru MVC. (Potencier, 2020)

3.4.1.1 Instalace

Jednou z hlavních výhod Symfony je jeho snadná instalace a podpora všech hlavních operačních systémů. Symfony tak lze bez problému spustit na operačních systémech Windows, macOS a Linux. Další důležitou součástí je vývojové prostředí, které je plně v kompetenci programátora, ačkoliv je doporučené využít prostředí PhpStorm. Jeho ovládání je velice snadné a umožňuje využití různých druhů nástrojů, díky kterým je psaní kódu ještě jednodušší a intuitivnější. Samotná instalace Symfony je možná za využití příkazové řádky operačního systému, nebo přímo v PhpStormu a to dvěma způsoby. První z nich je za pomoci Composeru, což je nástroj třetí strany pro instalaci a správu závislostí

projektů v PHP. Příkaz je zobrazen na obrázku 5. Druhou možností je využití přímo Symfony binárních souborů, jenž je zobrazena na obrázku číslo 6. Oběma postupy lze docílit stejného výsledku. (Potencier, 2020)

```
# run this if you are building a traditional web application
$ composer create-project symfony/skeleton:"6.2.*" my_project_directory
$ cd my_project_directory
$ composer require webapp

# run this if you are building a microservice, console application or API
$ composer create-project symfony/skeleton:"6.2.*" my_project_directory
```

Obrázek 5 – Vytvoření Symfony projektu pomocí Composeru (Symfony, 2022)

```
# run this if you are building a traditional web application
$ symfony new my_project_directory --version="6.2.*" --webapp

# run this if you are building a microservice, console application or API
$ symfony new my_project_directory --version="6.2.*"
```

Obrázek 6 – Vytvoření Symfony projektu pomocí binárních souborů (Symfony, 2022)

3.4.1.2 Git, Docker

Při práci se Symfony lze využívat velkou řadu dalších nástrojů, které usnadňují práci programátora. V dnešní době je již téměř nepostradatelný verzovací nástroj Git. Ten umožňuje jednoduše spravovat a distribuovat konkrétní verze projektu. Dále obsahuje kompletní historii projektu, větvení, místní i vzdálený repozitář a díky němu může na projektu pracovat neomezené množství pracovníků najednou. Git je navíc zdarma, open source a podporuje všechny základní operační systémy. Vznikl již v roce 2005 a jeho tvůrcem byl Linus Torvalds. Git lze ovládat více způsoby. Asi nejvyužívanějším je přímo pomocí příkazů v příkazovém řádku. PhpStorm navíc umožňuje ovládání Gitu rovnou ve svém vývojovém prostředí. Poslední možností jsou externí aplikace, jako je např. Sourcetree, které umí jednotlivé projekty a jejich větve vizualizovat, což umožňuje větší přehlednost

celého projektu. Navíc není nutné znát syntaxi konkrétních úkonů, ale pouze jejich názvy. (Valkovič, c2023)

Druhým nástrojem, který rozhodně stojí za zmínku je virtualizační nástroj Docker. Díky němu je možné celou aplikaci zabalit do tzv. kontejneru, který je pak možné spustit bez ohledu na prostředí. Lze jej tak bez problému spustit na operačních systémech Windows, Linux, nebo macOS. Kontejnery využívají virtualizaci jádra operačního systému a obsahují jen nejnutnější soubory pro běh dané aplikace. To umožňuje snížit velikost a tím pádem ušetřit výpočetní výkon systému. Stejně jako Git je i Docker k dispozici zdarma a má otevřený zdrojový kód. Vznikl v roce 2013 a je napsaný v programovacím jazyce GO. (Vondra, c2023)

3.4.1.3 Základní struktura projektu

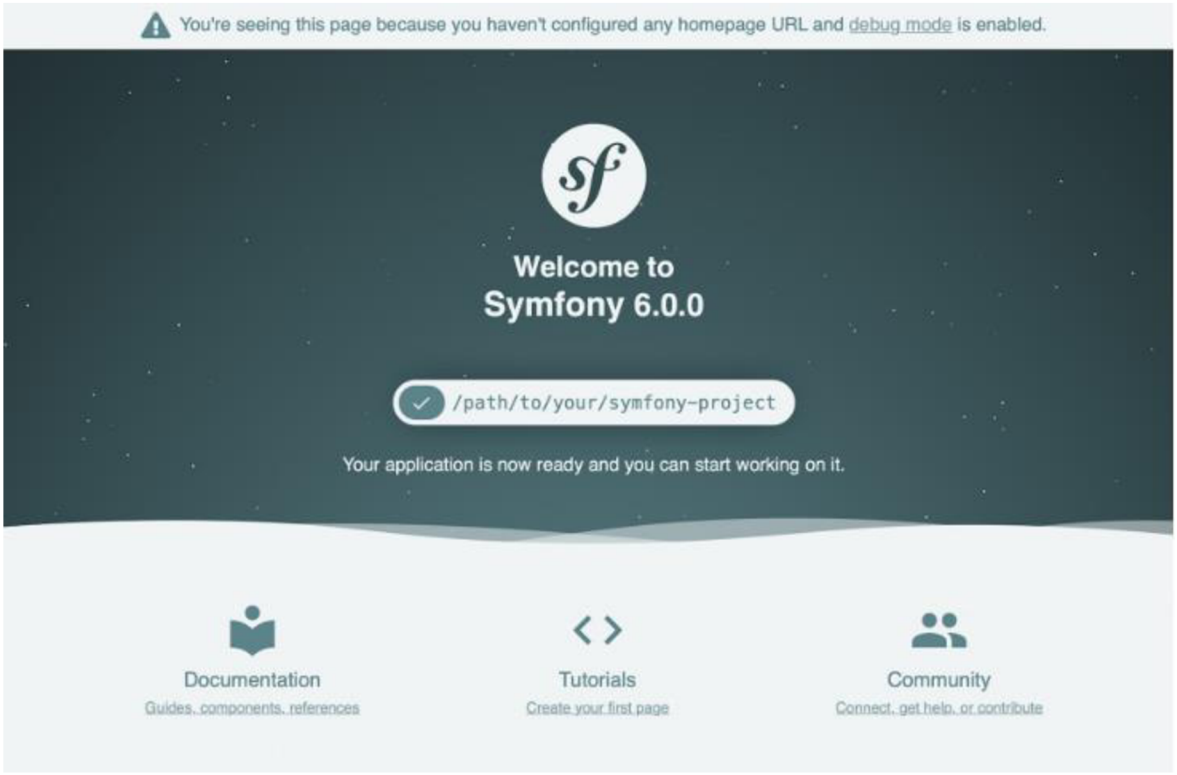
Po instalaci Symfony vygeneruje základní strukturu a konfigurační soubory, které jsou nezbytné pro hladký běh celého projektu. Ve výchozím adresáři se tak zobrazí několik složek a souborů, jenž jsou popsány níže.

- Složka bin – Obsahuje hlavní vstupní bod pro příkazovou řádku.
- Složka config – V této složce se nachází konfigurační soubory. Ty mají své výchozí hodnoty, které zpravidla fungují velice dobře. Pokud není nutná změna, jenž přímo vyžaduje konkrétní projekt, doporučuje se je neupravovat a ponechat ve výchozím nastavení.
- Složka public – Jedná se o kořenovou složku webu, která obsahuje také soubor „index.php“ a právě ten je výchozím bodem pro všechny dynamické zdroje HTTP. Cokoliv, co se nachází v této složce, je dostupné pro prohlížeč. Z tohoto důvodu by se zde neměly vyskytovat žádné citlivé části aplikace.
- Složka src – Zde bude programátor trávit většinu svého času. Jedná se o složku, které obsahuje většinu kódu aplikace. Zároveň tato část kódu není dostupná pro prohlížeče a nedá se tak zneužít.
- Složka var – Obsahuje mezipaměti, protokoly a soubory generované během chodu aplikace.
- Složka vendor – V poslední složce se nachází veškeré balíčky, které byly nainstalovány a to včetně Symfony samotného. Programátor by do této složky neměl vůbec vstupovat a nic zde upravovat.

- Soubory .lock – Vytváří a spravuje tzv. zámky, což jsou mechanismy, které poskytují přístup ke sdíleným prostředkům. Umožňují tak například zajištění toho, že příkaz bude proveden jen jednou ve stejnou dobu. (Potencier, 2020)

3.4.1.4 Spuštění lokálního serveru

Pro spuštění Symfony v lokálním prostředí existuje opět několik postupů, které se od sebe drobně liší, ale docílí stejného výsledku. Prvním z nich je spuštění pomocí vestavěného PHP serveru příkazem: „php -S 127.0.0.1:8000 -t public“. Druhou možností je využít prostředí Symfony, které má také svůj vlastní server: „symfony server:start -d“. Dále lze využít externí aplikace, které jsou k tomu určené. Vhodným pomocníkem je například Docker, který z aplikace vytváří tzv. kontejnery. Pro začátek je nezbytné kontejner postavit, k čemuž slouží příkaz: „docker-compose -p my_project build“. Po vytvoření je již možné kontejner spustit příkazem: „docker-compose -p my_project up“. Kromě zde popsaných řešení existuje i celá řada dalších možností, jak projekt spustit a preference jednotlivých přístupů je spíše na uživateli samotném, ačkoliv právě Docker poskytuje celou řadu prostředků pro usnadnění práce. (Máca, c2023; Potencier, 2020)



Obrázek 7 – Úvodní stránka projektu v Symfony (Symfony, 2022)

3.4.1.5 Kontrolery

Jedná se o základní stavební kameny celého frameworku Symfony a zároveň hlavní bod pro implementaci aplikační logiky. Jsou zodpovědné za zpracování HTTP požadavku a vrácení odpovědi, kterou může být v podstatě cokoliv od HTML, nebo XML stránky až po stažení souboru, přesměrování, nebo i chyby 404. Zajišťují také napojení uživatelského rozhraní na backendovou logiku aplikace. V Symfony lze kontroler vytvořit pomocí příkazu, případně lze manuálně vytvořit PHP třídu a do ní vložit vše potřebné. (Potencier, 2020)

Jako vhodný popis fungování kontrolerů poslouží kód. Který se nachází níže. Při vytváření je nezbytné, aby třída zahrnovala „AbstractController". Díky tomu Symfony pozná, že se jedná právě o třídu s kontrolery. Součástí každého kontroleru jsou anotace, které zahrnují URL adresu, název, metody a případné další části, jsou-li nezbytné. Prvním kontrolerem na obrázku je „homepage", který je poměrně jednoduchý a pouze generuje šablonu pro domovskou obrazovku. Druhý s názvem „new" je již poměrně rozsáhlejší a zajímavější. Jeho úkolem je vytvořit nový produkt a vložit jej do databáze. Nejprve dochází k vytvoření proměnné produkt, která ale v tuto chvíli neobsahuje žádné hodnoty. Následně je vytvořen formulář, kam uživatel zadává potřebná data o produktu. Ve chvíli, kdy jsou data odeslána a zároveň jsou validní, dochází k naplnění podmínky. Data z formuláře jsou načtena do produktu a o vše ostatní se postará „Entity manager", což je Symfony komponenta pro práci s daty v databázi. Programátor mu pouze přidělí příslušná data a zadá, co s nimi má udělat. Poté co je produkt vytvořen dochází k přesměrování na domovskou stránku. (Potencier, 2020; Symfony, 2022)

```
<?php

namespace App\Controller;


class ProductController extends AbstractController
{
    /**
     * @Route ("/", name="homepage", methods={"GET"})
     */

    public function hopemage()
    {
        return $this->render('homepage.html.twig');
    }

    /**
     * @Route ("/add", name="add", methods={"GET", "POST"})
```

```
     */

    public function new(Request $request, EntityManagerInterface
$entityManager): Response
    {
        $product = new Product();
        $form = $this->createForm(AddFormType::class, $product);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid())
        {   $product = $form->getData();
            $product->setActivity(true);
            $entityManager->persist($product);
            $entityManager->flush();


            return $this->redirectToRoute('homepage');
        }
        return $this->render('add.html.twig', ['form' => $form-
>createView()]);
    }
}
```

(Symfony, 2022)

3.4.1.6 Entity

Jako entitu lze v Symfony označit třídu, která reprezentuje objekt v databázi. Práce s databází je možná díky anotacím Doctrine, které umožňují mapování objektů na jednotlivé tabulky v databázi. S objekty je tedy možné pracovat stejně jako s daty samotnými. Hlavní výhodou využívání entit v Symfony tak je snadná manipulace s daty, která umožňuje věnovat více času důležité logice aplikace před tvorbou databází a SQL dotazů. Příklad entity je zobrazen v následujícím kódu, kde je znázorněna entita produktu, která obsahuje id, jméno a metody get a set pro práci s nimi. (Potencier, 2020; Symfony, 2022)

```
<?php

namespace App\Entity;

/**
 * @ORM\Entity(repositoryClass=ProductRepository::class)
 */
class Product
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
```

```
    private $id;
    /**
     * @ORM\Column(type="string", length=255)
     */
    private $name;
    public function getId(): ?int
    {
        return $this->id;
    }
    public function getName(): ?string
    {
        return $this->name;
    }
    public function setName(string $name): self
    {
        $this->name = $name;
        return $this;
    }
}
```
(Symfony, 2022)

3.4.1.7 Šablony

Pro tvorbu šablon v Symfony se používá moderní jazyk Twig. Jedná se o samostatný produkt s otevřeným zdrojovým kódem vytvořeným pro PHP. Kromě Symfony je součástí velké řady dalších frameworků a nástrojů. Navíc není nutné psát kompletní šablonu, ale je možné využít již předdefinované šablony, které jsou běžně k dispozici. Twig podporuje celou řadu funkcí, jako jsou zobrazování dat z databáze, výpis chybových hlášení, nebo třeba vkládání CSS a JS souborů. Syntaxe je velice snadná a intuitivní, takže psaní šablon v Twigu nebude problém ani pro programátory, kteří se s tímto jazykem zatím nesetkali. Níže je ukázka jednoduché Twig šablony, která umí vygenerovat formulář, obsahuje tlačítko pro odeslání formuláře a ve druhé části se nachází seznam historie cen. Každý Twig soubor musí rozšiřovat základní šablonu „base.html.twig“ a zároveň se musí nacházet ve složce „templates“, která je pro to přímo určena. (Potencier, 2020; Symfony, 2022)

```
{% extends 'base.html.twig' %}
{% block body %}

      {{ form_start(form) }}
      {{ form_widget(form) }}
            <button class="btn btn-primary">Uložit</button>
            {{ form_end(form) }}


        <table id="productPrices" class="table">
            <tbody>
```

```
                {% for productPrices in prices %}
                    <tr><p>{{ productPrices.price }}kč</p></tr>
                {% endfor %}
            </tbody>
        </table>
{% endblock %}
```

(Symfony, 2022)

3.4.2 Laravel

Laravel je velmi podobný framework jako Symfony, ačkoliv mezi nimi existují drobné rozdíly. Vznikl v roce 2011 vývojářem Taylorem Otwelle, jako reakce na složitost a zdlouhavost vývoje webových aplikací v čistém PHP. Jedná se o open source projekt, takže se poměrně rychle rozrůstal především díky početné komunitě vývojářů. Obdobně jako u Symfony je žádoucí využít některé nástroje, které dokáží usnadnit práci při vývoji, jako jsou Docker, nebo Git. (Stauffer, 2019)

3.4.2.1 Instalace

Instalace frameworku je velice jednoduchá a je možné jej spustit na všech operačních systémech, které mají nainstalovaný jazyk PHP. Důležitou součástí instalace je vývojové prostředí, které bude použito. Stejně jako u Symfony je vhodným softwarem PhpStorm, který dokáže ulehčit velkou řadu úkonů. Instalace Laravelu je možná několika způsoby. Pokud bude instalace probíhat pomocí příkazové řádky, lze využít Composer příkaz pro jednorázovou instalaci, který je zobrazen na obrázku 8. Jestliže bude uživatel vytvářet více projektů, může příkaz nainstalovat do adresáře v proměnném prostředí a následně volat přímo Laravel, což je znázorněno na obrázku číslo 9. K instalaci je možné využít právě i vývojové prostředí PhoStormu. (Welling, 2017)

```
composer create-project laravel/laravel example-app
```

Obrázek 8 – Vytvoření Laravel projektu pomocí Composeru (Laravel c2011-2023)

```
composer global require laravel/installer

laravel new example-app
```

Obrázek 9 – Instalce Laravelu do adresáře v proměnném prostředí (Laravel c2011-2023)

3.4.2.2 Struktura

Aplikace je rozdělena na několik adresářových složek, které reprezentují jednotlivé části aplikace. Tyto složky jsou vytvořeny již při generování projektu a uživatel by neměl měnit jejich základní strukturu.

- Adresář bootstrap – Obsahuje základní logiku aplikace, bez které by ji nebylo možné spustit.
- Adresář config – V této složce se nachází konfigurační soubory. Ty mají své výchozí hodnoty, které zpravidla fungují velice dobře. Pokud není nutná změna, jenž přímo vyžaduje konkrétní projekt, doporučuje se neupravovat je a ponechat výchozí hodnoty.
- Adresář database – Zde jsou umístěny migrace databázového schématu a logika pro naplnění schématu konkrétními daty.
- Adresář public – Představuje kořenový adresář dokumentů aplikace. Nachází se zde soubor „index.php“, což je vstupní bod do aplikace, který spouští celý framework Laravel. Dále obsahuje různé šablony stylů, obrázky atd.
- Adresář resources – Obsahuje pohledy, internacionalizační soubory a další prostředky aplikace.
- Adresář storage – V tomto adresáři se nachází dočasná data, která si zde ukládá framework, nebo aplikace. Může se jednat o mezipaměti, relační data, nebo třeba zkompilované pohledy.
- Adresář test – Obsahuje automatické testy, které si může uživatel sám vytvořit.
- Adresář vendor – Obsahuje závislosti, které byly nainstalovány prostřednictvím Composeru tedy i framework samotný. Programátor by zde neměl nic upravovat, jinak riskuje, že naruší chod frameworku.

- Adresář app – Zde se nachází veškerá logika aplikace. V tomto adresáři tak programátor stráví většina času při vývoji aplikace. (Welling, 2017)

3.4.2.3 Kontrolery

Kontrolery v Laravelu fungují velice obdobně jako u Symfony. Jsou zodpovědné za zpracování HTTP požadavku a interagují s ostatními části aplikace, jako jsou model a zobrazení (pohledy). Jedná se o části projektu pro uspořádání logiky aplikace, které seskupují související části do jediné třídy. Dokáží tak zpracovávat příchozí požadavky a následně zobrazovat, vytvářet, editovat, nebo mazat data, přesměrovat uživatele na jinou stránku, nebo třeba stáhnout soubor. Kontrolery se nachází ve složce: „app/Http/Controllers“ a k vytvoření základního kontroleru pro entitu uživatele stačí použít příkaz: „php artisan make:controller UserController“. Kód níže znázurňuje jednoduchý příklad kontroleru, který má za úkol zobrazit uživatele podle jeho id. (Welling, 2017; Laravel c2011-2023)

```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\View\View;

class UserController extends Controller
{
    /**
     * Show the profile for a given user.
     */
    public function show(string $id): View
    {
        return view('user.profile', [
            'user' => User::findOrFail($id)
        ]);
    }
}
```

(Laravel c2011-2023)

3.4.2.4 Modely

Ve frameworku Laravel představují modely to samé, co jsou v Symfony entity. Používají se pro práci s databází a následnou reprezentaci dat. Většinou se tak jedná o databázové tabulky, které obsahují proměnné, atributy a příslušné hodnoty. Model definuje schéma tabulky v databázi a poskytuje základní metody pro práci s daty.

Implementace modelů probíhá pomocí komponenty Eloquent, která poskytuje přístup k objektově relačnímu mapování. K přístupu k databázi a operacemi s ní se tudíž nepoužívají SQL dotazy, ale objektově orientované přístupy. Stejně jako u vytváření kontroleru stačí použít jednoduchý příkaz: „php artisan make:model User“ a Laravel vytvoří model automaticky. Následující kód znázorňuje jednoduchý model uživatele, který obsahuje jméno, e-mail a heslo. (Welling, 2017; Laravel c2011-2023)

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    use HasFactory;

    protected $fillable = [
        'name',
        'email',
        'password'
    ];
}
```

(Laravel c2011-2023)

3.4.2.5 Pohledy

K prezentaci dat slouží v Laravelu pohledy. Ty dokáží generovat složitý výstup, který může být například ve formátu HTML, za pomoci načtených dat a výpočtů z kontroleru. Ačkoliv je možné vytvářet pohledy pouze za pomoci nástroje frameworku Laravel, je vhodné využít šablonovací systém Blade. Ve výchozím nástroji se s pohledy pracuje pomocí čistého jazyka PHP, který může být v případě implementace logiky zbytečně zdlouhavý a složitý. Systém Blade oproti tomu poskytuje celou řadu užitečných funkcí, které dokáží vytvářet pohledy i bez složité logiky. Jednou z hlavních výhod je schopnost dědit šablony a definované sekce, které tak není nutné vytvářet v každém pohledu zvlášť. Navíc systém Blade dokáže pracovat i s klasickým PHP kódem, což některé jiné obdobné nástroje nedovolují. Vytvoření jednoduché Blade šablony pro uživatele s parametry jméno a e-mail je zobrazeno níže. (Welling, 2017; Laravel c2011-2023)

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <div class="card">
            <div class="card-header">{{ __('User Profile') }}</div>

            <div class="card-body">
                <div class="form-group row">
                    <label for="name" class="col-md-4 cot-form-label
                    text-md-right">{{ __('Name') }}</label>

                    <div class="col-and-6">
                        <input id="name" type="text" class="form-
                        control" name="name" value="{{ $user->name
                        }}" readonly>
                    </div>
                </div>

                <div class="form-group row">
                    <label for="email" class="col-md-4 col-form-label
                    text-md-right">{{ __('E-Mail Address') }}</label>

                <div class="col-and-6">
                   <input id="email" type="email" class="form-
                   control" name="email" value="{{ $user->email }}"
                   readonly>
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```
(Laravel c2011-2023)

3.4.3 Závěr

Ačkoliv si jsou oba frameworky velice podobné, existují mezi nimi nepatrné rozdíly. Laravel cílí více na jednoduchost a snaží se poskytnout takové řešení, které je snadněji využitelné. Symfony je oproti tomu více modulární, obsahuje větší množství samostatných komponent a obecně jej lze považovat za složitější framework. Laravel má větší komunitu, stejně tak i dokumentace je považována za přívětivější, a i díky tomu je vhodnější pro začátečníky. Mezi největší rozdíly patří využívání jednotlivých nástrojů. Pro komunikaci s databázemi využívá Laravel nástroj „eloquent“, který je přímo integrován do frameworku, zatímco v Symfony slouží k těmto účelům knihovna „doctrine“, která umožňuje pracovat s databázovými záznamy jako s běžnými objekty. Obdobně je tomu u šablon, kde také každý framework využívá svůj vlastní systém. V Laravelu se aplikuje šablonovací systém Blade,

kdežto v Symfony obdobný systém Twig. Pro tento projekt se jeví jako vhodnější framework Symfony, neboť je více zaměřen na rozsáhlejší aplikace a obsahuje větší množství funkcionalit a nástrojů, které mohou být během vývoje použity.

4 Vlastní práce

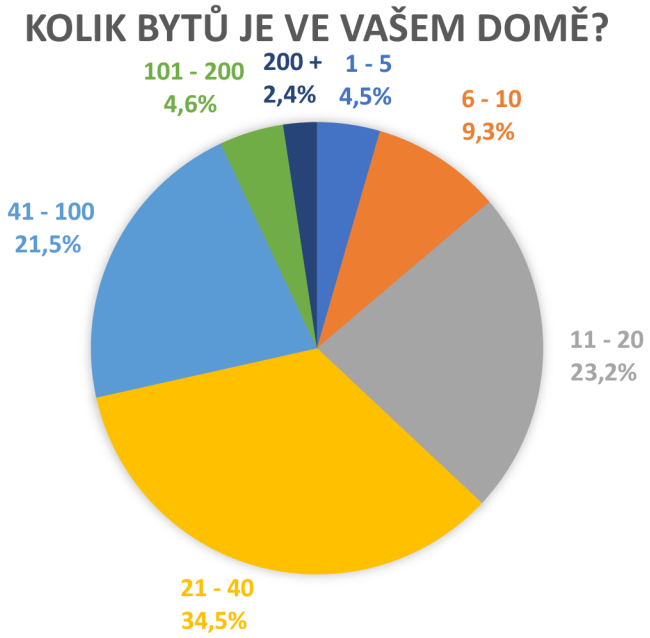
4.1 Přípravy pro vývoj aplikace eVlastníci

Nápad na vývoj aplikace eVlastníci vznikl již v roce 2021. V této době panovala pandemie COVID-19, která zasáhla téměř všechny oblasti lidského života. V České republice platil výjimečný stav a byla omezena některá práva za účelem zamezit šíření pandemie. Jedním z nich bylo i omezení počtu lidí na jednom místě, které se zásadně dotklo i shromáždění SVJ. Některá z nich přitom nebyla usnášeníschopná již před pandemií a problémy s hlasováním se tak začaly projevovat více, než kdy dříve. Ve firmě Externity s. r. o. jsme tak začali přemýšlet nad moderním řešením, které by tyto problémy vyřešilo a zároveň zvýšilo komfort pro všechny členy hlasování. Tímto moderním řešením měla být webová a mobilní aplikace, jež by umožnila legální a legitimní hlasování SVJ online.

4.1.1 Dotazníkové šetření

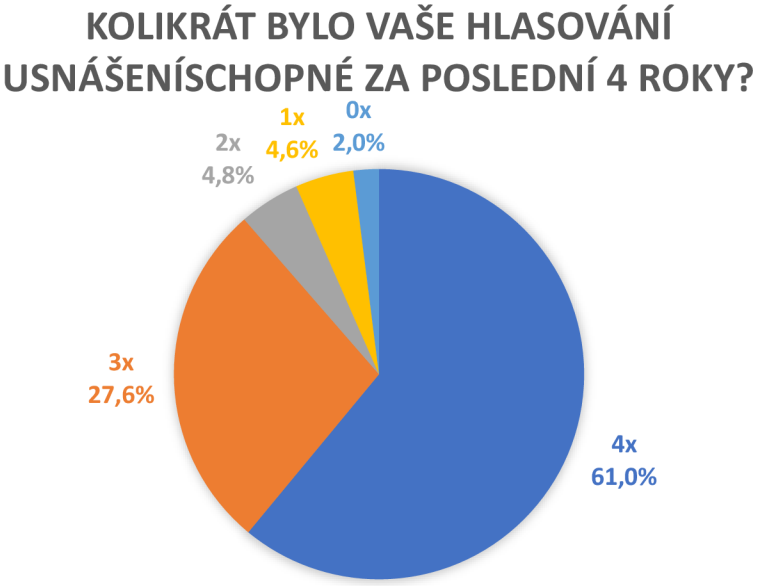
Před zahájením vývoje samotné aplikace bylo nutné provést několik analýz, ze kterých by bylo patrné, že si aplikace skutečně najde v praxi své uplatnění. Jednou z nich byl i průzkum trhu pomocí kvantitativního dotazníkového šetření. To probíhalo formou Facebook kampaně a zúčastnilo se ho 807 korespondentů. Některé otázky byly koncipovány formou výběru jedné odpovědi, jiné formou multiple choice v závislosti na charakteru konkrétní otázky. Z průzkumu byly vybrány nejvíce vypovídající otázky, které byly zahrnuty do této práce.

Graf 1 znázorňuje počet bytů v domě a je z něj patrné, že nejvíce korespondentů žije ve středně velkých domech s počtem bytů mezi 21–40. Přes 20 % respondentů žije v 11–20 bytech na dům a s drobným odstupem následuje 41–100 bytů. Přibližně 14 % respondentů žije v malých domech pod 10 bytů a zhruba 7 % bydlí ve velkých bytových domech.



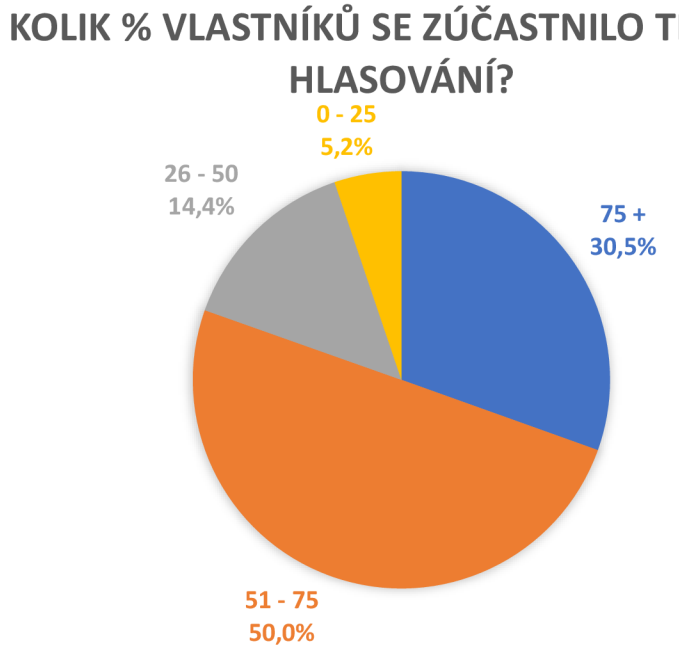
Graf 1 – Počet bytů

Z grafu 2 je patrné, že přes 60 % hlasování bylo za poslední 4 roky usnášeníschopné. 27,6 % hlasování bylo usnášeníschopné 3x, 4,8 % hlasování 2x, 4,6 % 1x a pouze 2 % hlasování nebylo usnášeníschopné ani jednou. Ačkoliv se poměr usnášeníschopnosti zdá vcelku vysoký, i tak je zde značný prostor pro zlepšení. I taková sdružení, která nemají s docházkou členů problém, by mohla aplikace začít využívat, především díky většímu komfortu, menší časové náročnosti a vyšší přehlednosti.



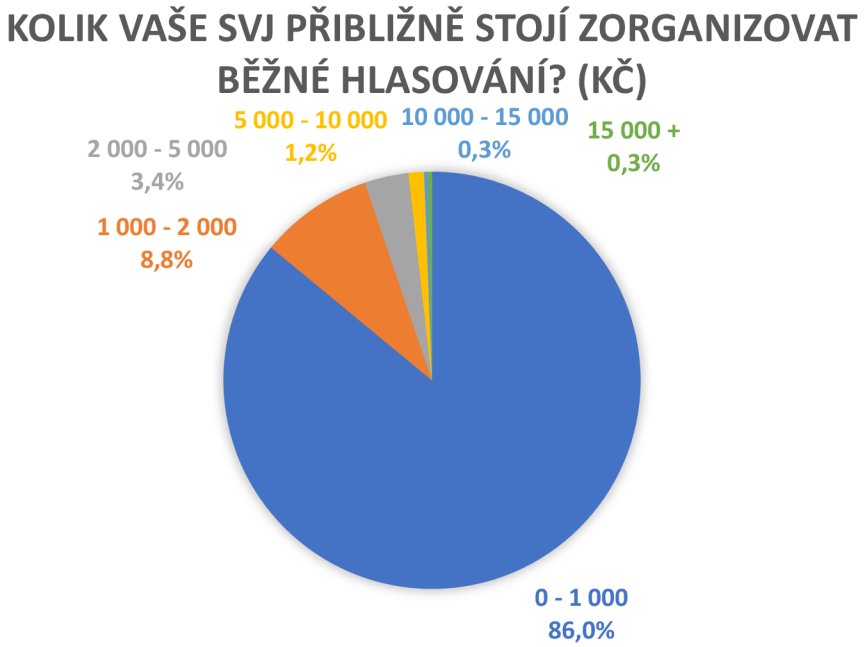
Graf 2 – Usnášeníschopnost hlasování

Docházka vlastníků do jisté míry reflektuje usnášeníschopnost hlasování. Nejvyšší zastoupení má účast mezi 51–75 %. Pouze na 30,5 % schůzí je účast vyšší než 75 % a necelých 20 % připadá na účast pod 50 %, tedy hranicí usnášeníschopnosti hlasování.



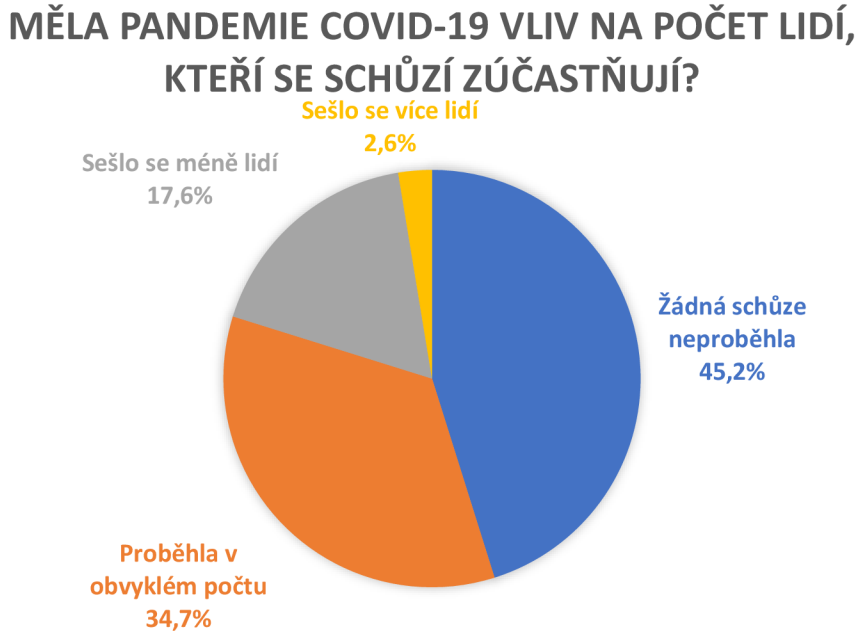
Graf 3 – Účast na hlasování

Cena za organizaci hlasování se v 86 % pohybuje mezi 0–1000 Kč. Čím více stoupá cena hlasování, tím menší zastoupení v grafu má. Dá se předpokládat, že čím je sdružení větší a obsahuje více vlastníků, tím budou náklady na hlasování vyšší. Zároveň je nezbytné počítat s možnou odchylkou, neboť členové SVJ nemusí vůbec tušit, jaké jsou náklady na zorganizování hlasování.



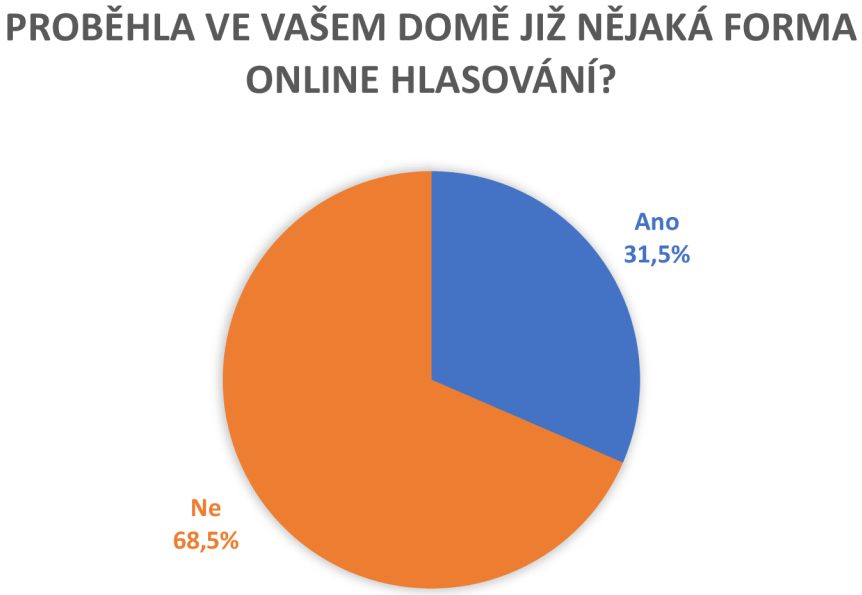
Graf 4 – Cena hlasování

Vliv pandemie COVID-19 na hlasování SVJ je naprosto zřejmý. 45,2 % hlasování vůbec neproběhlo, čímž můžou vznikat společenství značné komplikace. 34, 7 % hlasování proběhlo v obvyklém počtu a v 17,6 % se sešlo méně lidí, než bylo obvyklé. Ve 2,6 % se dokonce sešlo více lidí. To může být zapříčiněno tím, že někteří lidé pracovali z domova a měli více času, aby se hlasování zúčastnili.



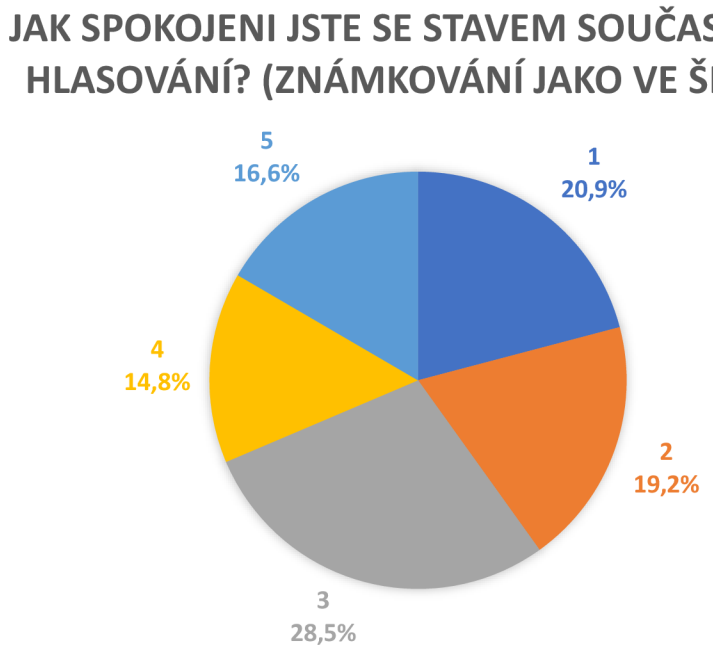
Graf 5 – Vliv pandemie COVID-19

Z grafu 6 je patrné, že u 31,5 % korespondentů již proběhla nějaká forma online hlasování. Dá se předpokládat, že toto číslo se zvýšilo nejvíce v době pandemie. Většina členů nicméně s online hlasováním stále zkušenosti nemá.



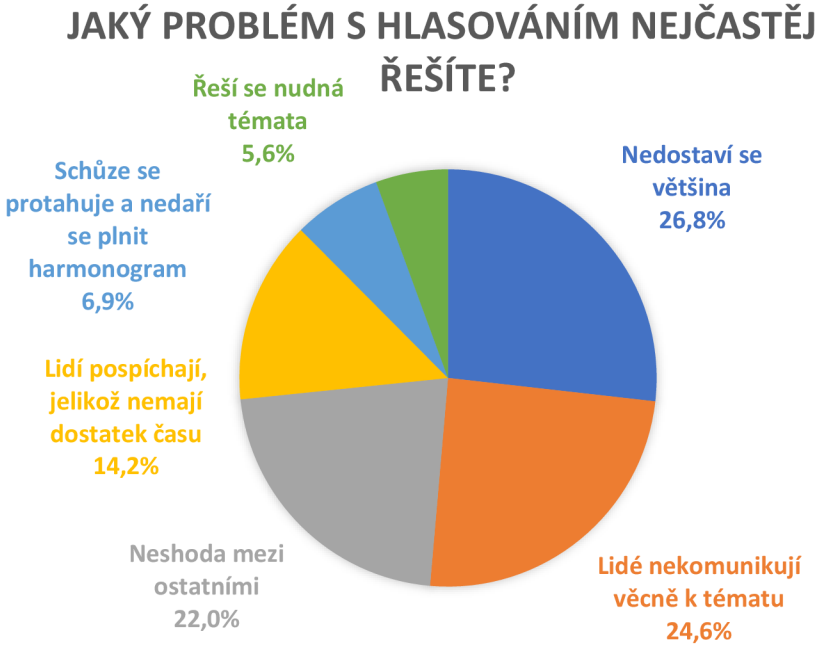
Graf 6 – Zkušenost s online hlasováním

Ve spokojenosti se současným stavem hlasování korespondenti uváděli známky, které mají stejný význam jako ve škole. Graf je do značné míry rozdělen vcelku rovnoměrně. Největší část zajímá hodnocení za 3 s 28,5 %. Známku 1 si vysloužilo pouze 20,9 % hlasování a známka 2 dosahuje 19,2 %. Nejhorší známku 5 udělilo 16,6 % hlasujících a známka 4 dosahuje hodnoty 14,8 %. Z tohoto grafu je patrné, že většina členů SVJ není spokojena se současným stavem hlasování a existuje zde značný prostor pro nové možnosti, které by zvýšily celkovou spokojenost.



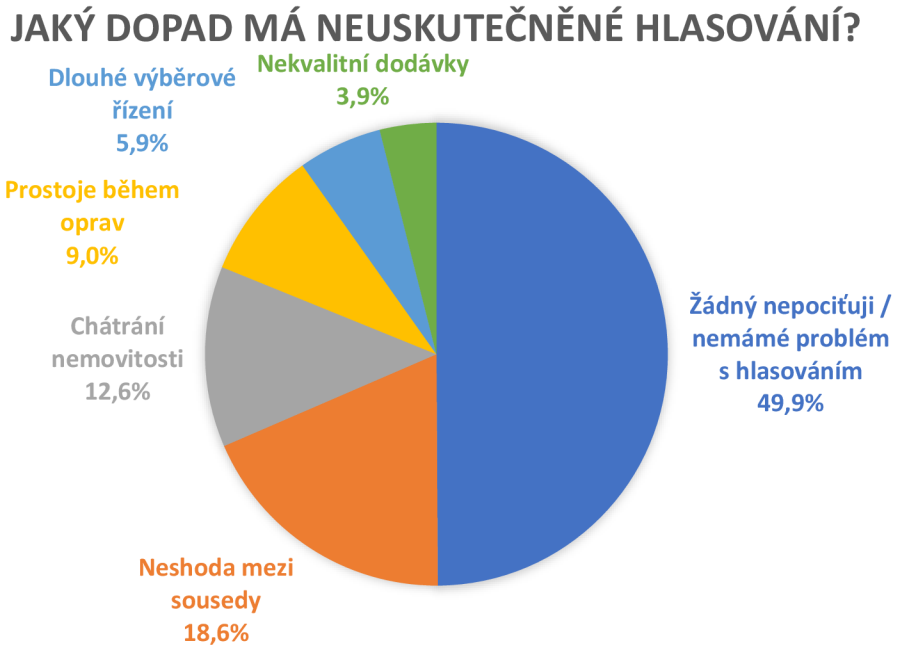
Graf 7 – Spokojenost se současným hlasováním

Graf 8 znázorňuje nejčastější problémy v rámci hlasování a probíhal formou multiple choice. Nejvíce korespondentů shledává problém v tom, že se nedostavila většina. Následuje komunikace, která neprobíhá věcně a neshoda mezi ostatními. Poměrně vysokého počtu odpovědí dosáhl i nedostatek času. Nejméně korespondentů pak opovědělo, že se nedaří plnit harmonogram a řeší se nudná témata.



Graf 8 – Problémy hlasování

Dopad na neuskutečněné hlasování 49,9 % korespondentů vůbec nepociťuje. Zde se jedná pravděpodobně o tu část, která nemá s usnášeníschopností žádný problém. Neshody mezi sousedy vnímá 18,6 % korespondentů a 12,6 % hlasujících pak pociťuje chátrání nemovitosti. Prostoje během oprav jsou dopadem u 9 %, dlouhé výběrové řízení u 5,9 % a nekvalitní dodávky u 3,9 % hlasujících.

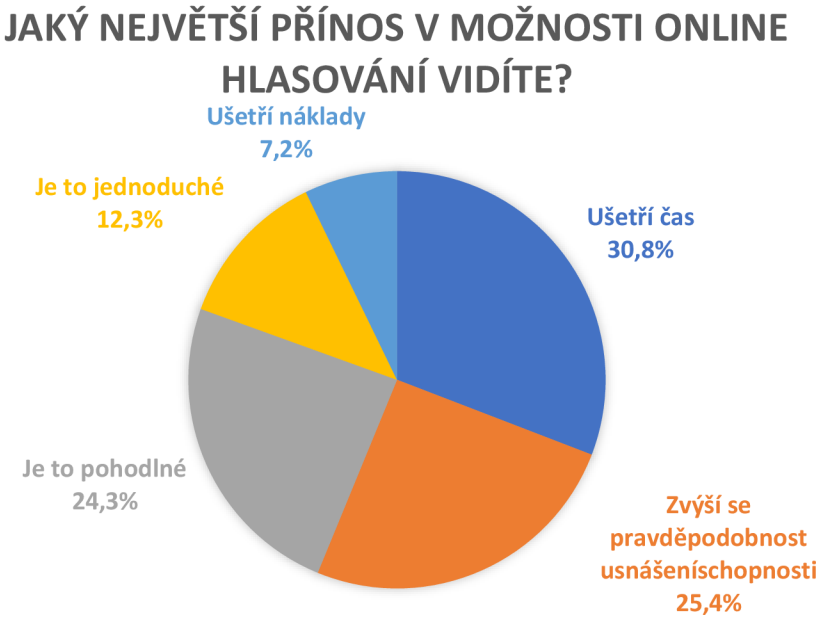


Graf 9 – Dopad neuskutečněného hlasování

U otázky, zda by se líbila členům možnost online hlasování, odpovědělo 53,3 % hlasujících jako ano a zbylých 46,7 % jako ne. Tento výsledek je poměrně optimistický a dá se usuzovat, že by jednotlivá sdružení měla o aplikaci skutečně zájem. Vzhledem k trendům dnešní doby je možné, že tento zájem bude postupně ještě narůstat. Nezájem o aplikaci budou mít z pravidla starší lidé, kteří tolik nevyužívají moderní technologie a menší sdružení, kde není s usnášeníschopností problém. Naopak větší zájem by mohl být u mladší generace a u lidí, kteří žijí ve velkých panelových domech, kde se nachází značné množství vlastníků a není tak snadné hlasování uspořádat.

Graf 10 – Názor na online hlasování

Poslední graf je zaměřen na přínos online hlasování a probíhal formou multiple choice. Nejvíce korespondentů shledává výhody v úspoře času, následuje zvýšení pravděpodobnosti usnášeníschopnosti, více pohodlí, jednoduchost a nejmenší část hlasujících se domnívá, že by aplikace snížila náklady.



Graf 11 – Přínos online hlasování

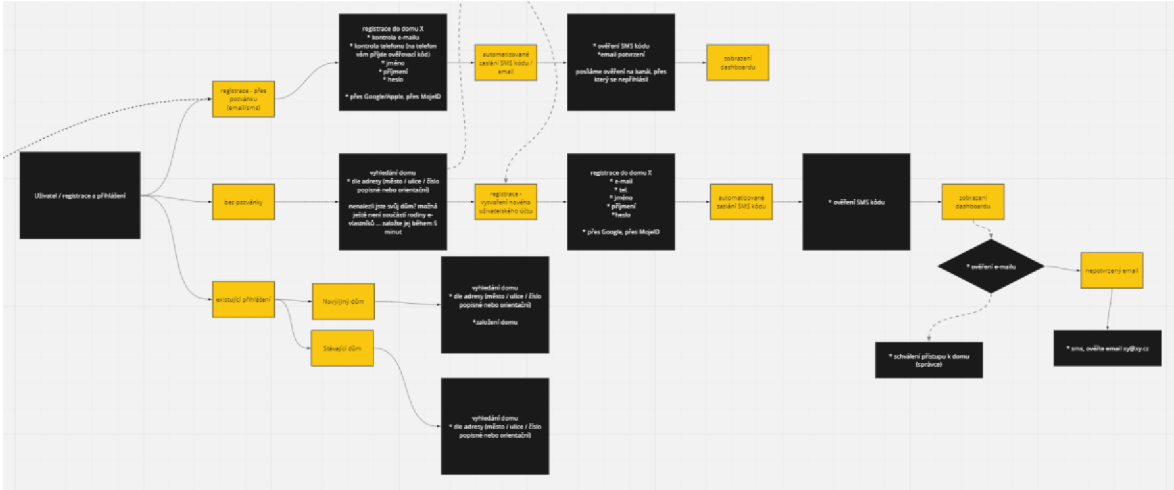
4.1.1.1 Shrnutí

Z provedeného průzkumu je zřejmé, že některá společenství mají skutečně s hlasováním vážný problém. Ačkoliv většina schůzí je usnášeníschopná, lze předpokládat, že online aplikace by si našla na trhu své místo. Takováto aplikace by se líbila více než polovině korespondentů a shledávají v ní značné přínosy. Po kvantitativním šetření byl proveden i kvalitativní výzkum, kdy proběhl osobní rozhovor s pěti předsedy SVJ. Z něj bylo patrné, že ačkoliv je většina SVJ usnášeníschopná, o aplikaci by uvažovali a vznesli návrh o jejím nasazení na schůzi.

4.1.2 Diagramy

Před zahájením návrhu designu bylo vytvořeno také několik diagramů, které slouží k lepšímu porozumění funkcím a procesům uvnitř aplikace. Všechny diagramy byly vytvářeny pomocí vizualizačního nástroje Miro. Nejvýznamnější část zabraly data flow a user flow diagramy. Oba uvedené dosahovaly poměrně značných rozměrů, a proto bude pro demonstraci uvedena pouze část user flow diagramu, který je zobrazen na obrázku 10. Konkrétně se jedná o část registrace a přihlášení uživatele. Na úvod se diagram dělí do tří větví, kdy první z nich znázorňuje registraci přes pozvánku, druhá registraci bez pozvánky a poslední přihlášení zaregistrovaného uživatele. Při registraci přes pozvánku uživatel musí nejdříve vyplnit formulář, který obsahuje kontrolu e-mailu, kontrolu telefonu, jméno, příjmení a heslo. Následně je uživateli zaslán ověřovací kód prostřednictvím SMS a potvrzení na e-mail. Po ověření se uživatel dostává na dashboard a může postupovat dále.

Při registraci bez pozvánky musí uživatel nejprve vyhledat dům podle adresy. Následně probíhá registrace, kde uživatel opět vyplní údaje obdobně jako při registraci přes pozvánku, ovšem ty v tomto případě nejsou předvyplněny. Poté probíhá ověření pomocí SMS kódu a zobrazení dashboardu. Uživatel musí v tomto případě ověřit také svůj e-mail a následně získá přístup k domu jako správce. Pokud neověří e-mail je mu odeslána další SMS, která žádá toto ověření. Pro existující uživatele probíhá pouze přihlášení s následnou možností volby přihlášení se ke stávajícímu domu, nebo vytvoření nového pomocí vyhledávače podle adresy.

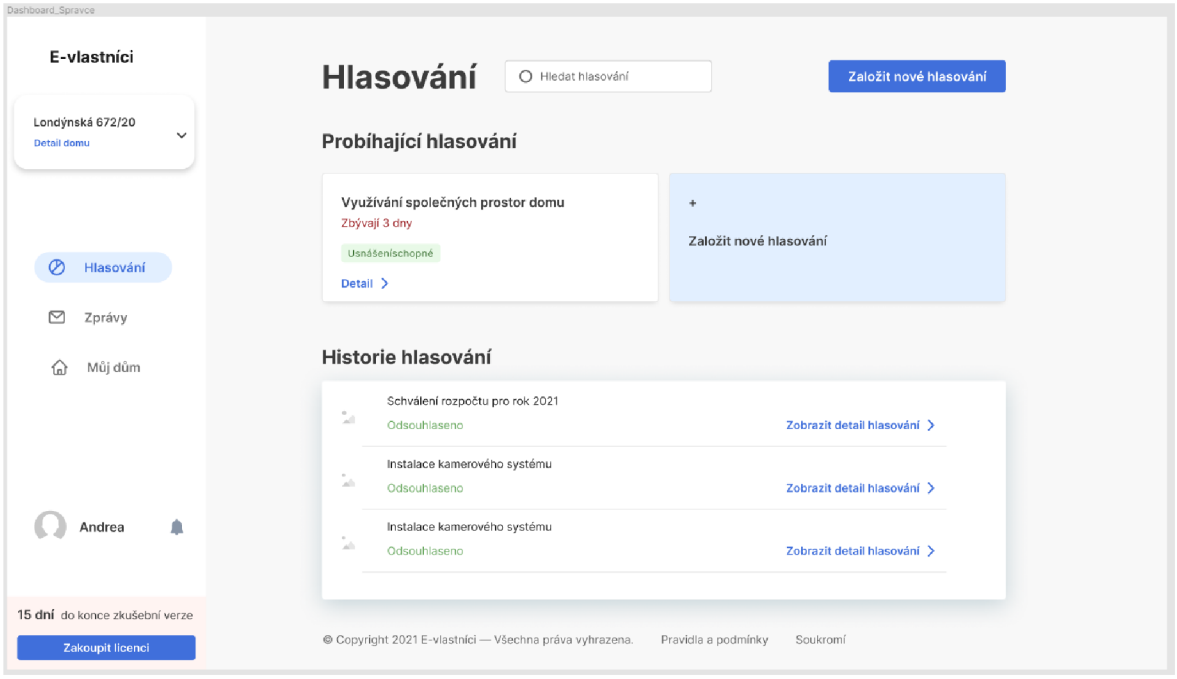


Obrázek 10 – User flow diagram

4.1.3 UX/UI design

V rámci přípravné fáze byly vytvořeny interaktivní prototypy webové a mobilní aplikace, aby bylo zajištěno, že výsledný produkt bude pro uživatele co nejpřívětivější a pohyb po aplikaci pro něj bude rychlý, efektivní a intuitivní. Podcenění UX designu by mohlo mít za následek, že se uživatelé budou v aplikaci a jejích funkcích ztrácet, což může způsobit odliv klientely. Prototyp byl vytvořen pomocí nástroje Figma, jenž je profesionálním softwarem pro tvorbu wireframů, návrhu uživatelského rozhraní a tvorby interaktivních prototypů. Uživatelské rozhraní z tohoto návrhu do jisté míry vycházelo a bylo obohaceno o moderní vzhled, díky příjemným barvám, které dokáží zdůraznit důležitost jednotlivých komponent, ale nejsou pro uživatele příliš kontrastní. Většina pracovní plochy zůstala ve formě bílé a několika odstínů šedé barvy. Menu bylo odlišeno tmavě modrou barvou a tlačítka pro interakci tyrkysovou. Některé prvky, které bylo nutné barevně zvýraznit, jako například příchozí notifikace, jsou zobrazovány v oranžové barvě.

Jak již bylo zmíněno v přechozím odstavci, návrh designu byl vytvořen pomocí softwarového nástroje Figma. V něm probíhal především návrh logické části aplikace, ačkoliv již zde byly zahrnuty některé grafické prvky. Nejedná se tedy o čistý bezbarvý wireframe, ale v rámci přípravné fáze zastával tento prototyp stejnou funkci. Na obrázku 11 je demonstrováno, jak takovýto prototyp vypadá. Konkrétně se jedná o sekci hlasování z pohledu správce. Všechny části prototypu je možné procházet, logicky na sebe navazují.



Obrázek 11 – Prototyp aplikace v nástroji Figma

4.1.3.1 Vzorové persony

Pro návrh aplikace byly vytvořeny 3 vzorové persony, které mají za úkol co nejlépe reprezentovat jednotlivé typy uživatelů. Primární persona bude představovat uživatele, který nejčastěji využije tuto aplikaci. Je nižšího věku a disponuje znalostmi a schopnostmi s ovládáním různých druhů aplikací. Sekundární persona bude představovat předsedu SVJ, který bude mít za úkol v aplikaci vytvářet jednotlivá hlasování. Z průzkumu vyplývá, že takovýto předseda je většinou ve věku 40–50 let a disponuje průměrnými schopnostmi v práci s moderními aplikacemi. Poslední personou bude starší vlastník, který sice disponuje přístupem k počítači a chytrým mobilním telefonem, ovšem v jejich obsluze má značné nedostatky.

Jan Svoboda

- Věk: 29 let
- Vzdělání: Vysokoškolské
- Zaměstnání: Manager ve firmě zaměřené na prodej léčiv
- Zájmy: hry na PC, akční filmy, vášeň pro automobily

- Typický den: Ráno vstává okolo 6 hodiny a většinu pracovního dne tráví v automobilu při cestě za klienty firmy po celé České republice. Svůj volný čas tráví většinou s přáteli mimo domov, případně se zabaví hraním na PC nebo sledováním televize. Využívá celou řadu moderních zařízení a aplikací a s jejich obsluhou nemá žádný problém.

Josef Novák

- Věk: 50 let
- Vzdělání: Střední škola s maturitou
- Zaměstnání: Dispečer v dopravní firmě
- Zájmy: Rodina, procházky se psem, televizní seriály, vaření, zahrada
- Typický den: Do práce vstává kolem 7 hodiny a většinu dne tráví prací na počítači a telefonními hovory s ostatními členy firmy. Ve svém volném čase se věnuje především své rodině, procházkami se psem a prací na zahradě. Považuje se za středně zdatného ve využívání moderních aplikací.

Jaroslav Kašpárek

- Věk: 71 let
- Vzdělání: Základní škola
- Zaměstnání: Důchodce, dříve prodavač v masně
- Zájmy: Čtení knih, sledování televize, luštění křížovek, procházky
- Typický den: Vzhledem k důchodovému věku nemusí ráno nikam vstávat, a tak spí někdy i do 8 hodin. Většinu dne kouká s partnerkou na televizi, případně vyrazí na procházku do nedalekého lesa. Nevlastní chytrý telefon a jediný přístup k internetu má prostřednictvím stolního počítače, na kterém zvládá pouze základní obsluhu a není příliš pokročilý uživatel. Ve svém věku již nemá příliš dobrý zrak a některé menší prvky pro něj mohou být špatně čitelné.

4.1.3.2 Uživatelské testování

Pro vývoj aplikace vznikly interaktivní prototypy v nástroji Figma. Pomocí těchto prototypů bylo možné provést uživatelské testování, ještě před vznikem samotné aplikace. Prvotního testování se zúčastnilo pět participantů v rozmezí od 27 do 57 let, kteří disponovali různými schopnostmi a zkušenostmi s ovládáním aplikací. Nejzdatnější zúčastnění využívali

aplikace jako Facebook, WhatsApp, internetové bankovnictví, případně specifické aplikace denně, naopak jeden z participantů se nepovažoval za příliš pokročilého uživatele a s aplikacemi podobného druhu neměl příliš zkušeností. Participanti dostali za úkol splnit 7 různých úkonů, které měly ověřit především správnost rozložení jednotlivých prvků a přehlednost celého systému. Testování trvalo u každého participanta přibližně 45 minut.

4.1.3.2.1 Registrace uživatele

Uživateli byl odeslán e-mail s informací o implementaci aplikace eVlastníci do SVJ. Jeho úkolem je přečíst si informace a prostřednictvím odkazu v e-mailu provést registraci. Tento krok zvládli všichni participanti a registrace jim přišla přehledná. Až na jednoho uživatele byli všichni spokojeni s předvyplněnými údaji v registračním formuláři. Jeden z uživatelů namítal možné nedodržení GDPR a také to, že někteří mohou chtít použít jiné údaje k registraci než ty předvyplněné, i když je jejich změna ve formuláři možná. Na počátku registrace by někteří participanti chtěli vidět informace o jednotce, které se registrace týká. Dále proběhly náměty na zobrazení zadaného hesla při registraci, které v prototypu chybělo.

4.1.3.2.2 Hlavní stránka

Uživatel se nachází na hlavní stránce a jeho úkolem je projít si obsah a odhadnout, k čemu budou jednotlivé části sloužit. Respondenti se shodli, že celá hlavní stránka působí přehledně, ačkoliv někteří z nich navrhli, že by se informace mohly rozdělit do podstránek, aby jich nebylo tolik najednou. Na první pohled není jasný význam grafů, ačkoliv po zamyšlení jsou schopni grafy popsat, ovšem ne vždy správně. Dále některý z uživatelů navrhl větší důraz na odlišení dokončených hlasování od těch, které stále probíhají.

4.1.3.2.3 Hlasování

Uživatel má za úkol zjistit, jaká hlasování nemají dostatek hlasů na to, aby byla právně závazná. Jedno z nich si následně vybrat a libovolně v něm odhlasovat. Neusnášeníschopná hlasování odhalili všichni uživatelé kromě jednoho. Grafy u jednotlivých otázek popisují uživatelé narozdíl od těch na hlavní stránce správně. Všichni uživatelé zvládli v aplikaci úspěšně odhlasovat a chválili posloupnost kroků a celkový dojem z hlasování. Pozitivní ohlas zanechala i potvrzovací SMS a závěrečný detail o provedeném hlasování. Nicméně po odhlasování si uživatelé nebyli jistí tím, jaké změny v sekci

s hlasováními nastaly. Někteří odhalili změnu pořadí jednotlivých hlasování, ale nebyli si jisti, podle jakého klíče proběhlo.

4.1.3.2.4 Odeslání podkladů e-mailem

V další části mají uživatelé najít podklady k libovolnému již dokončenému hlasování z minulosti a zaslat si je na e-mail. Archiv hlasování se podařilo najít všem uživatelům, ale správně odeslat podklady zvládli pouze dva. Největší problém byl shledán v tlačítku s třemi tečkami místo popisu. Vzhledem k tomu, jací uživatelé budou aplikaci používat, je mnohem vhodnější používání slovního popisu než grafického znázornění. Na tomto názoru se shodli všichni participanti. Uživatelům, kteří zvládli podklady úspěšně poslat, chybí možnost stáhnou všechny podklady najednou a také jejich tisk přímo z aplikace.

4.1.3.2.5 Přepínání mezi domy

Uživatelé se dostávají do situace, kdy vlastní dvě bytové jednotky v různých domech. Jejich úkolem je změnit stávající dům na druhý vlastněný. Změnu domu zvládli všichni uživatelé zčásti i díky tomu, že se v aplikaci po celý čas testování pohybují a tuto část měli na očích. Dva uživatelé ovšem po kliknutí na druhý dům nečekali automatickou změnu a začali klikat na tlačítko „přidat další dům“.

4.1.3.2.6 Notifikace

V této části mají uživatelé za úkol zjistit, zda mají nějaké nepřečtené upozornění, najít jej a otevřít. Tato úloha byla pravděpodobně nejproblémovější z celého testování. Uživatelé měli potíže notifikace najít a zobrazit, což zvládli jen dva z nich, a i ti se shodli, že je řešení poměrně nepřehledné. Vhodnou změnou bude přesun notifikací z profilové sekce přímo na hlavní stránku a lepší kontrast, aby byla nová upozornění jasně patrná. Jeden participant měl problém pochopit zadání, neboť nevěděl, co přesně znamená slovo notifikace.

4.1.3.2.7 Kontakty na SVJ

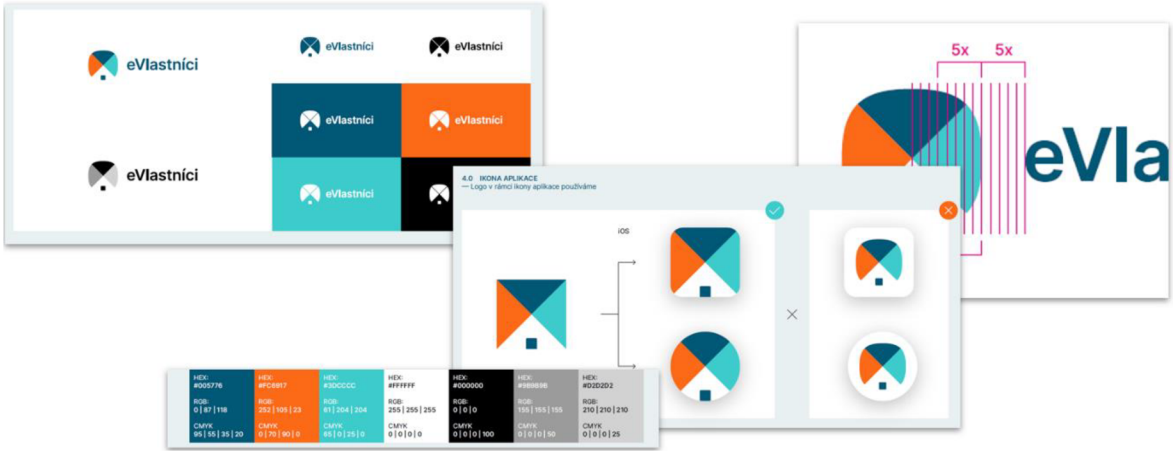
V posledním úkolu měli uživatelé najít kontakt na předsedu výboru SVJ. Všichni směřovali do správné sekce k detailu domu a s nalezením kontaktu neměli problém. Někteří dokonce navrhovali, jaké další informace by se mohly k detailu domu přidat. Nejčastěji byly zmiňovány mapy objektů, mapy a informace z katastru nemovitostí.

4.1.3.2.8 Shrnutí

Na závěr měli participanti ohodnotit aplikaci známkami jako ve škole. Aplikaci udělili známky 1, 1–2, 2, 2, 2–3. Obecně lze říci, že mladší a zdatnější část zvládala pohyb po aplikaci lépe a od toho se odvíjelo i pozitivnější hodnocení. Naopak uživatelé, kteří nedokázali splnit některé kroky, dávali aplikaci horší známku. Nicméně i přes to se všichni uživatelé shodli, že aplikaci by používali a vidí v ní značné výhody. Aplikaci je pouze nutné zpřehlednit, přesunout a zvýraznit některé funkce, které dělaly uživatelům problémy, jako bylo například hledání notifikací. Samozřejmě prototyp má svá omezení a výsledný produkt, který z něj vznikne, bude jednoznačně přehlednější, prvky budou lépe uspořádány, popsány a barevně odlišeny. Jako testovací nástroj posloužil prototyp velice dobře a následný vývoj bude využívat poznatky od uživatelů, aby eliminoval chyby a nedostatky, které se v prototypu nacházely.

4.1.3.3 Jméno a logo

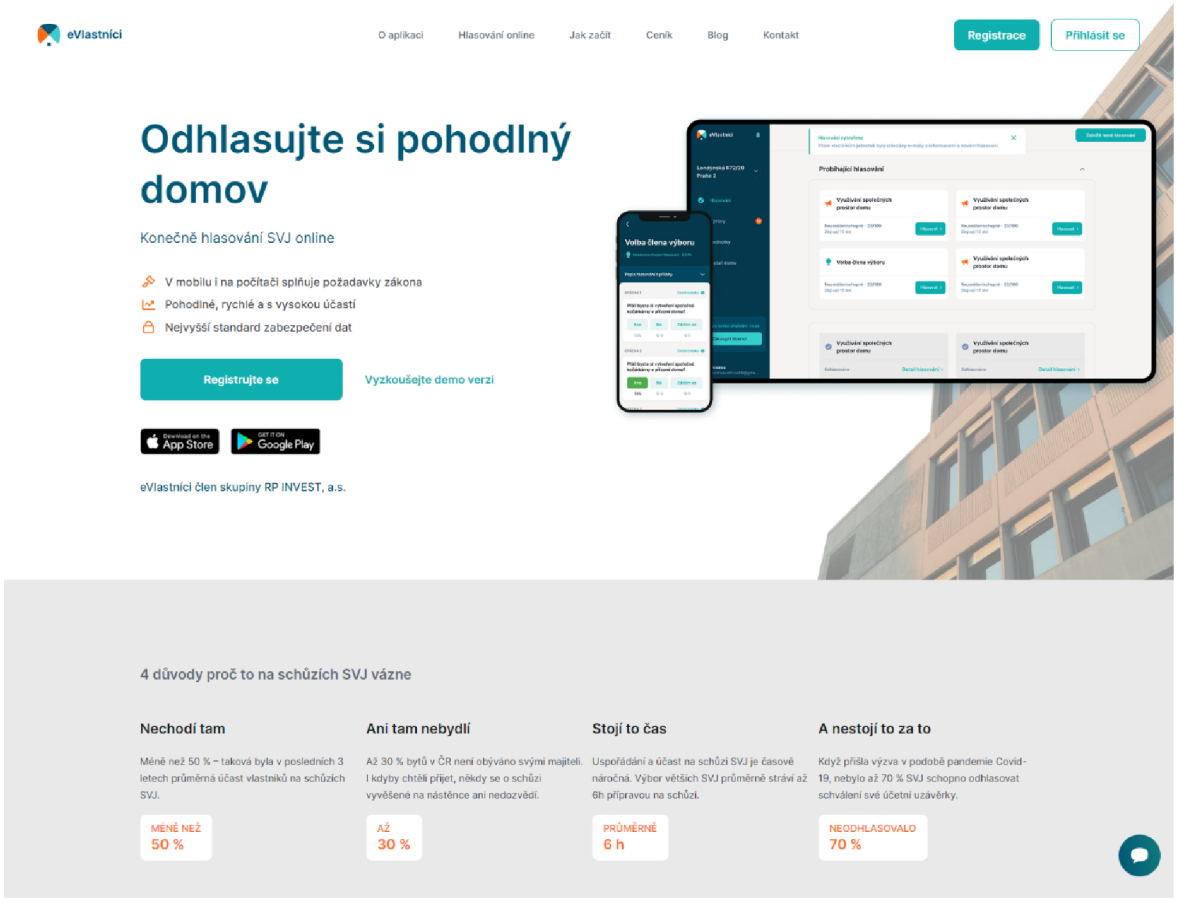
Při počátcích vývoje byla aplikace pracovně pojmenována jako eVlastníci, tedy elektronická forma vlastníků. Postupem času se začalo nad názvem více diskutovat a byly vytvořeny nové návrhy, nicméně po uvážení různých možností vznikl závěr, že původní pracovní označení se zalíbilo natolik, aby bylo zachováno i v produkční verzi. Návrh loga vycházel primárně z barev, které jsou použity i v samotné aplikaci. Design byl primárně cílen na moderní a příjemný vzhled.



Obrázek 12 – Návrh loga

4.1.3.4 Web design

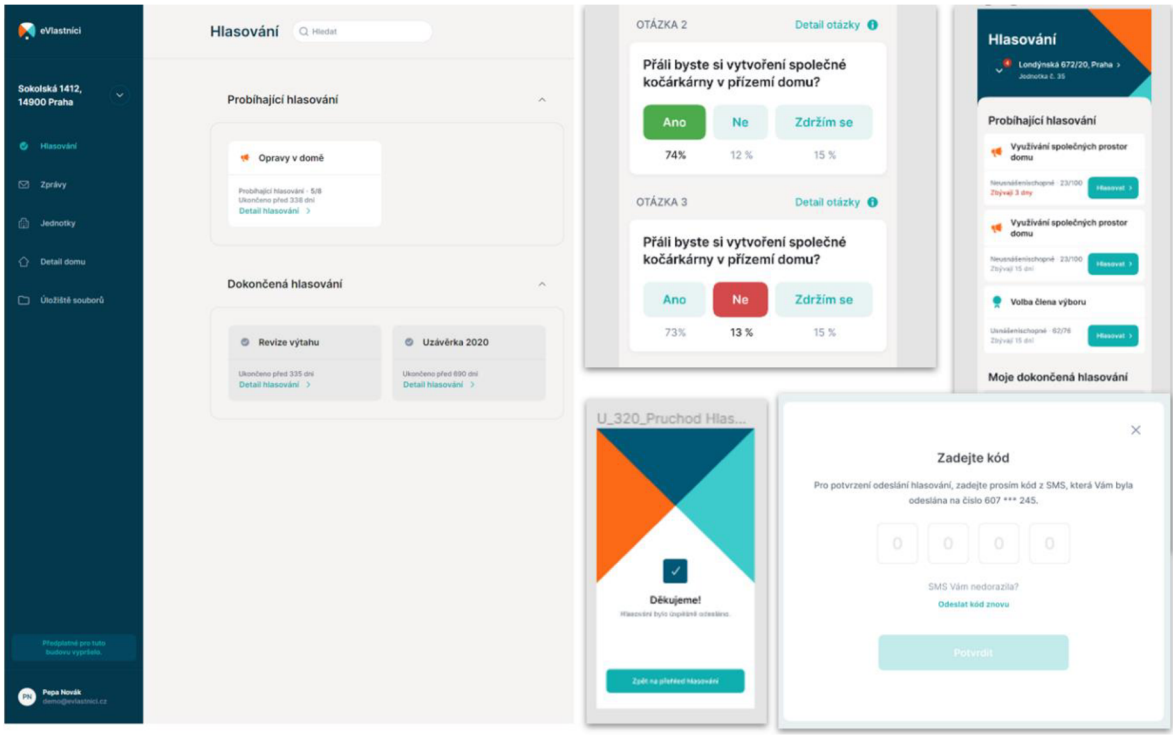
Hlavní webová stránka eVlastníků byla vytvořena tak, aby působila na první pohled moderně a jednoduše stejně jako samotná aplikace. V horní části obrazovky se nachází scrollovací menu, které uživatele posune po stránce na požadovanou sekci, případně přesměruje na další stránku v závislosti na konkrétní interakci. Sekce o aplikaci, hlasování online, jak začít a ceník se nachází na titulní stránce, v případě blogu a kontaktu dochází k přesměrování na další stránku, tlačítka registrace a přihlášení otevírají nové okno s webovou aplikací. Dále se na stránce nachází jednotlivé sekce s texty a obrázky, odkaz na demo verzi, kde si uživatelé mohou aplikaci vyzkoušet a odkazy na stažení mobilní aplikace z Google Play, nebo App Store. Webová stránka také obsahuje chatovací nástroj, kde se mohou uživatelé přímo zeptat na konkrétní otázky. Ve spodní části stránky se vyskytuje footer, který obsahuje všechny sekce jako hlavní menu a důležité odkazy včetně sociálních sítí.



Obrázek 13 – Web design

4.1.3.5 Design aplikace

Samotná webová a mobilní aplikace využívá obdobný vzhled se stejnými barvami jako logo a webová stránka tak, aby vše působilo jako jeden celek. Menu se nachází na levé straně pracovní plochy a je barevně odděleno od zbytku stránky. Na něm uživatel nalezne odkazy na všechny důležité části aplikace, jako jsou výběr domu, hlasování, zprávy, detail domu a úložiště dokumentů. Ve spodní části menu se navíc nachází jméno přihlášeného uživatele, které po kliknutí zobrazí další detaily a možnosti úpravy jeho profilu. Na hlavní části obrazovky se zobrazují informace dle konkrétní sekce, ve které se uživatel právě nalézá. V případě hlasování je v horní části zobrazeno vyhledávání, jež umožňuje selektovat konkrétní hlasování podle názvu, dále sekce s probíhajícími hlasováními, kde mohou uživatelé zaznamenat svůj hlas. Správce se navíc může podívat i na detail hlasování s průběžnými výsledky a počtem odhlasovaných uživatelů. V poslední sekci se pak nachází dokončená hlasování, kde mohou vidět detaily všichni uživatelé.



Obrázek 14 – Design aplikace

4.2 Backend vývoj webové aplikace eVlastníci

Vývoj celého projektu eVlastníci byl zahájen firmou Externity s. r. o. v červenci 2021 a trval zhruba 9 měsíců. Za tuto dobu vznikla webová stránka, webová aplikace a mobilní aplikace. Tato část je věnována především vývoji backendové části webové aplikace, který probíhal v PHP frameworku Symfony. Protože obsahuje backend mimo automaticky vygenerovaných částí přes 800 souborů, tudíž je jejich kompletní popis v této práci nereálný, bude obsažen pouze stručný výčet několika ukázek. Frontendová část vznikala kompletně odděleně a byla vytvořena v JavaScriptové knihovně pro tvorbu uživatelského rozhraní React. Veškerá komunikace mezi vrstvami probíhá přes REST API a všechna data jsou při přesunu mezi vrstvami převedeny do souboru ve formátu JSON.

4.2.1 Struktura projektu

Veškerá důležitá logika aplikace se nachází ve složce „src“, rozdělené dále do několika podsložek.

- Command – Tato část obsahuje příkazy, které je možné spouštět přes příkazovou řádku, což umožňuje zautomatizovat některé činnosti.
- Common – Zde se nachází pomocné třídy a funkce dále využívané napříč aplikací.
- DataFixtures – Slouží k přednastavení dat, která jsou vložena do databáze během migrace. Ty se využívají v testovacím prostředí, nebo například pro demo aplikace.
- Integration – Používá se k synchronizaci mezi jednotlivými vrstvami aplikace, tedy například dekóduje obsah requestu na JSON formát.
- Module – Tato složka je nejrozsáhlejší částí a zároveň právě na ní bude zaměřen zbytek této práce. Jednotlivé části aplikace jsou zde uspořádány do modulů obsahující související části, jako jsou kontrolery, entity, eventy apod. Dělí se na modul pro budovy, faktury, registr nemovitostí, hlasování, uživatele a předplatné. Poslední modul pro předplatné je v tuto chvíli nepředmětný, neboť aplikace je zdarma a veškerá logika, která se zde nachází není dále využívána, nicméně v budoucnu by mohla být použita pro zavedení placených částí.
- Security – Obsahuje třídy a soubory související s bezpečností aplikace. Kromě obecného zabezpečení se zde nachází například autentizace a autorizace uživatele.
- Service – V této složce jsou umístěny služby, které poskytují další funkcionalitu pro konkrétní části aplikaci. Využívají se například při načítání dat ze souborů JSON.

- Utils – Obdobně jako servisny poskytují další funkcionalitu, ta se ale týká obecně aplikace a neváže se na konkrétní části.

4.2.2 Moduly

Jak již bylo zmíněno, nejzajímavější část logiky se odehrává v 6 samostatných modulech. Každý modul se dále větví na vrstvy doménovou, prezenční a entity. V doménové vrstvě se nacházejí třídy, jež definují objekty a jejich chování, se kterými aplikace dále pracuje. Vyskytují se zde například repositáře, eventy a handlery. Prezenční vrstva obsahuje třídy starající se o zpracování dat z vrstvy doménové. Obsahuje tak především kontrolery a pomocné třídy pro input a output dat. Poslední entity část slouží k reprezentaci dat v databázi. Obsahuje jejich vlastnosti a metody pro práci s nimi.

4.2.3 Kontrolery

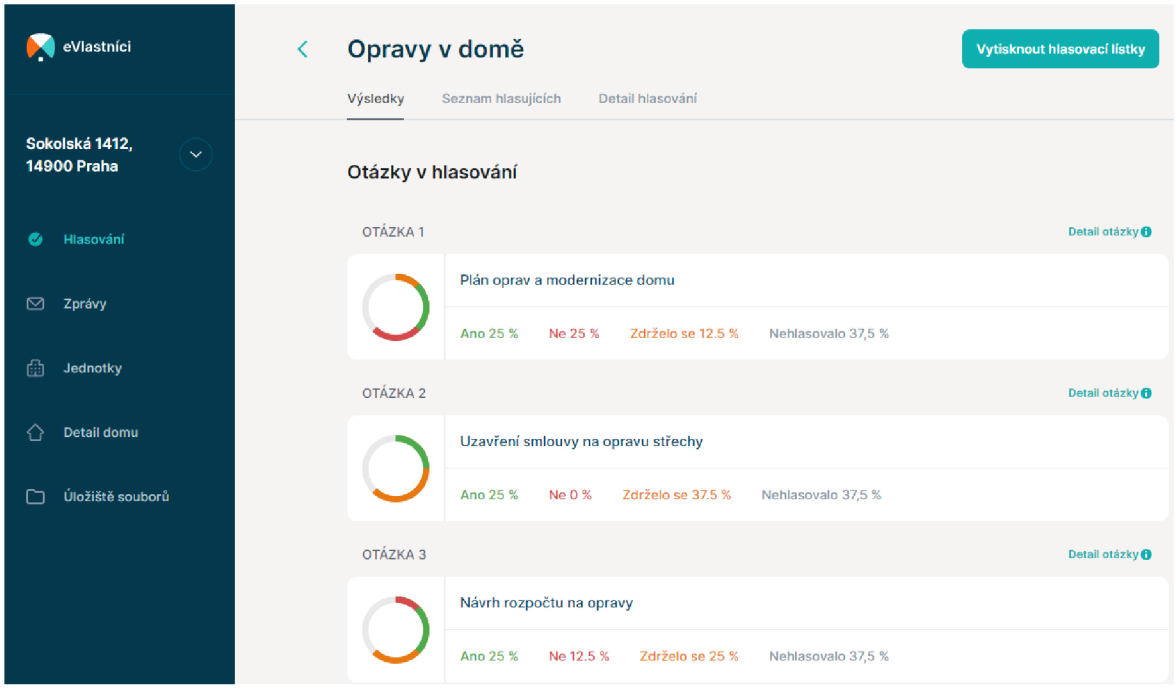
Kontrolery jsou základem prezenční vrstvy aplikace a řídí veškerý tok dat. Ovšem kontrolery v eVlastnících se od těch běžně používaných v Symfony do jisté míry odlišují. Běžně zobrazují kontrolery pohledy, tedy nejčastěji vrací Twig šablonu s konkrétními daty. V tomto případě je ale oddělen backend od frontendu a tento Symfony projekt tak žádné Twig šablony neobsahuje. Tedy pro upřesnění několik šablon zahrnuje, ty ale slouží například k posílání e-mailu, nebo třeba k vygenerování PDF dokumentu, nikoliv jako šablony skutečné stránky. Kontrolery tedy nevrací šablony jako takové, ale pouze outputy a inputy s daty, jež jsou dále převedeny do formy JSON a prostřednictvím REST API odeslána na frontent.

Jako ukázkový kód byl vybrán kontroler pro samotné hlasování, který má za úkol připravit data pro náhled na konkrétní hlasování. Z názvu složky je patrné, že se kontroler skutečně nachází ve složce s moduly, poté následuje název modulu, kterého se kontroler týká, což je hlasování a na závěr je umístěn v prezenční vrstvě. Na úvod obsahuje kód atributu, kde se nachází cesta lišící se v závislosti na ID budovy a ID hlasování. Dále je zde umístěn název a požadavky, které kontroler potřebuje, tedy jako v cestě ID budovy a uživatele. Následuje metoda, kterou je v tomto případě „get“. Poté jsou v atributu uvedena práva potřebná k tomu, aby se uživatel k datům dostal, tedy být členem dané budovy. Na závěr jsou uvedeny atributy pro deklaraci API koncového bodu a výstupních dat.

Po atributech následuje název funkce, tedy zobrazení a veškeré parametry, které funkce během svého procesu přijímá a zpracovává. Těmi jsou ID budovy a uživatele,

bezpečnostní ověření uživatele a služby k vytváření datových objektů pro odpověď uživateli nebo administrátorovi a repositář pro získání dat z databáze. Do proměnné „poll“ jsou načteny data pomocí metody „find“ nad instancí třídy „pollRepository“ s parametrem „pollId“. Pokud není nalezena požadovaná entita, je vrácena výjimka se zprávou o chybě. Na závěr se v kódu nachází podmínka, která vrací data podle toho, zda je uživatel administrátorem, nebo nikoliv.

```
namespace App\Module\Poll\Presentation\Poll;
    #[Route(
        '/building/{buildingId}/poll/{pollId}',
        name: 'view',
        requirements: ['buildingId' => RegExpPattern::ULID, 'pollId'
=> RegExpPattern::ULID],
        methods: ['GET']
    )]
    #[IsGranted(BuildingPermission::IS_MEMBER, subject:
'buildingId')]
    #[IsPartOf('pollId', 'buildingId')]
    #[Api\Endpoint(description: 'View poll')]
    #[Api\Output(
        description: 'View for member of building',
        code: Response::HTTP_OK,
        output: PollViewMemberOutput::class
    )]
    #[Api\Output(
        description: 'View for admin of building',
        code: Response::HTTP_OK,
        output: PollViewAdminOutput::class
    )]
    public function view(
        BuildingId $buildingId,
        PollId $pollId,
        #[CurrentUser] SecurityUser $user,
        PollViewMemberOutputFactory $pollViewOutputFactory,
        PollViewAdminOutputFactory $pollViewAdminOutputFactory,
        ProfileRepositoryInterface $profileRepository
    ): Response {
        $poll = $this->pollRepository->find($pollId) ?? throw $this-
>createNotFoundException();
        if ($profileRepository->isAdminForBuilding($user-
>getProfileId(), $buildingId)) {
            return $this->response($pollViewAdminOutputFactory-
>single($poll, $user->getProfileId()));
        }
        return $this->response($pollViewOutputFactory->single($poll,
$user->getProfileId()));
    }
```



Obrázek 15 – Detail hlasování

4.2.4 Entity

Každá entita v projektu reprezentuje jednu řádku v tabulce databáze. S entitami se pracuje pomocí objektově relačního mapování s nástrojem Doctrine. Dohromady se v celém projektu nachází přes 40 různých entit, které reprezentují různé části informací o budovách, uživatelích, hlasováních apod. Jako příklad byla zvolena entita, která představuje odeslaný hlas v hlasování. Ukázka kódu je zobrazena níže v této sekci.

Soubor se stále nalézá v modulu hlasování a složce pro entity. Na úvod se v entitě nachází anotace označující, že se jedná právě o třídu s entitami, dále jsou specifikovány unikátní sloupce pro ID otázky a jednotky. Na závěr je v anotaci vytvořen index pro sloupec „verify_at“. Následuje samotná třída pro hlas, které jsou přidány vlastnosti pro náhodně generovaný unikátní identifikátor (UID) a pro uchování datumu a času vytvořeného objektu. Poté jsou opět využity anotace k zápisu jednotlivých vlastností entity, jako jsou otázky, hlasující, jednotka, odpověď atd. První tři vlastnosti využívají relace pro napojení na další entitu a to konkrétně „ManyToOne“. V případě otázky to například znamená, že může být u jedné otázky více hlasů. Anotace dále obsahují datový typ vlastnosti, případně délku a to, zda může nabývat nulové hodnoty.

Následně se v kódu nachází konstruktor, který přijímá parametry a předává je dále jako závislosti následně přiřazené ke svým odpovídajícím vlastnostem. Kromě toho

konstruktor také generuje hodnoty pro vlastnosti ID a datum vytvoření. V poslední části entity se nachází funkce sloužící k získání informací o dané třídě. Pro příklad funkce „getId“ vrátí ID konkrétního hlasování. Poslední funkce s názvem „verify“ pak potvrzuje a nastavuje čas pro hlasování.

```
namespace App\Module\Poll\Entity;
#[ORM\Entity]
#[ORM\UniqueConstraint(name: 'question_unit', columns:
['question_id', 'unit_id'])]
#[ORM\Index(columns: ['verify_at'], name: 'verifyAt')]
class Vote
{
    use EntityUlidIdProperty;
    use EntityCreatedAt;
    #[ORM\ManyToOne(targetEntity: Question::class, inversedBy:
'votes')]
    #[ORM\JoinColumn(nullable: false)]
    private Question $question;
    #[ORM\ManyToOne(targetEntity: Profile::class)]
    #[ORM\JoinColumn(nullable: false)]
    private Profile $voter;
    #[ORM\ManyToOne(targetEntity: Unit::class)]
    #[ORM\JoinColumn(nullable: false)]
    private Unit $unit;
    #[ORM\Column(type: 'string', length: 255)]
    private string $answer;
    #[ORM\Column(type: 'app_datetime', nullable: true)]
    private ?Date $verifyAt = null;
    #[ORM\Column(type: 'string', length: 255)]
    private string $voteSource;
    #[ORM\Column(type: 'string', length: 255)]
    private string $voteAs;
    public function __construct(
        Question $question,
        Profile $voter,
        Unit $unit,
        AnswerEnum $answer,
        VoteSource $voteSource,
        VoteAs $voteAs,
        ?Date $verifyAt = null,
        ?VoteId $id = null
    ) {
        $this->id = ($id ?? VoteId::make())->asString();
        $this->createdAt = Date::now();
        $this->question = $question;
        $this->voter = $voter;
        $this->unit = $unit;
        $this->answer = $answer->value;
        $this->voteAs = $voteAs->value;
        $this->voteSource = $voteSource->value;
        if ($verifyAt) {
            $this->verify($verifyAt);
        }
```

```
    }
    public function getId(): VoteId
    {
        return VoteId::fromString($this->id);
    }
    public function getQuestion(): Question
    {
        return $this->question;
    }
    public function getVoter(): Profile
    {
        return $this->voter;
    }
    public function getUnit(): Unit
    {
        return $this->unit;
    }
    public function getAnswer(): Answer
    {
        return Answer::from($this->answer);
    }
    public function isVerify(): bool
    {
        return null !== $this->verifyAt;
    }
    public function verify(Date $verifyAt): self
    {
        Assert::null($this->verifyAt);
        $this->verifyAt = $verifyAt;
        return $this;
    }
}
```

4.2.5 Outputy

Outputy slouží v tomto projektu jako definice výstupních datových struktur a používají se při komunikaci s API. Pro demonstraci byl zvolen output, který se používá pro zobrazení hlasování. Jednotlivé vlastnosti jsou zapsány opět v anotacích, obsahují příklad dané hodnoty, datový typ a název.

```
namespace App\Module\Poll\Presentation\Poll\Output;
final class PollIndexOutput
{
    public function __construct(
        #[Api\Property(example: '1Bh7xEwcnpxooGcvrQVntB')]
        public readonly string $id,
        #[Api\Property(example: 'Poll')]
        public readonly string $name,
        #[Api\Property(example: 'draft|publish|active|finish')]
        public readonly string $state,
        #[Api\Property(example: '2022-01-03')]
```

```
        public readonly ?string $startAt,
        #[Api\Property(example: '2022-01-03')]
        public readonly ? string $endAt,
        #[Api\Property(example: 'false')]
        public readonly bool $canVote,
        #[Api\Property(example: 'false')]
        public readonly bool $hasVoted,
        #[Api\Property(example: 'false')]
        public readonly bool $isDraft,
        #[Api\Property(example: 'false')]
        public readonly bool $isFinished,
        #[Api\Property(example: 'false')]
        public readonly bool $isResultPublished,
        #[Api\Property(type: QuorateOutput::class.'[]')]
        public readonly QuorateOutput $quorate,
    ) {
    }
}
```

4.2.6 Interface repositářů

Interface repositáře slouží jako definice metod, které lze provést nad danou entitou. Tyto metody si lze představit jako konkrétní funkce umožňující manipulaci s daty v databázi. Jako příklad byl opět zvolen interface pro hlasování. Jelikož jsou zde definovány objekty, nachází se interface v doménové vrstvě pod složkou hlasování.

V interface existuje několik funkcí, které mají za úkol různé operace. Nalézají se zde funkce „get“ pro získání hlasování dle ID a podle datumu pro aktivní hlasování a ukončené hlasování. Dále je zde „find“ pro hledání konkrétního hlasování a funkce pro získání hlasování určeného pro administrátora nebo běžného uživatele. Poslední funkce je zodpovědná za nalezení jednotek, k nimž je přiřazena možnost hlasovat.

```
namespace App\Module\Poll\Domain\Poll;
interface PollRepositoryInterface
{
    public function get(PollId $id): Poll;
    public function find(PollId $id): ?Poll;
    public function getAllForActivateByDate(Date $date): iterable;
    public function getAllForFinishByDate(Date $date): iterable;
    public function forAdmin(BuildingId $id, ?int $limit = null, ?int
$offset = null): array;
    public function forMember(BuildingId $id, ?int $limit = null,
?int $offset = null): array;
    public function getUnitsWhichCanBeVotedFor(PollId $pollId,
ProfileId $profileId): array;
}
```

4.2.7 Repositáře

V předchozí části byl představen interface repositáře a nyní bude následovat ukázka z části repositáře samotného. Zatímco interface obsahuje pouze výčet funkcí pro další použití, v repositáři se již nachází i logika jednotlivých funkcí. Pro příklad byl zvolen repositář stejného interface, tedy pro hlasování. Následující kód byl pro demonstraci zkrácen a obsahuje pouze funkce „get“ pro získání hlasování dle ID a podle aktivních a dokončených hlasování. Soubor se nalézá ve stejné doménové složce jako interface. Hned z úvodu kódu je zřejmá provázanost obou souborů, neboť repositář implementuje rozhraní interface. První funkce vrací hlasování dle ID, případně vrátí chybu, pokud nebylo hlasování nalezeno. Funkce pro aktivní hlasování vytváří dotazy pomocí dotazovacího jazyka, které filtrují pouze ta hlasování, jež jsou právě aktivní. Poslední funkce funguje velice obdobně, ale filtruje hlasování již ukončená.

```
namespace App\Module\Poll\Domain\Poll;
class PollRepository implements PollRepositoryInterface
{
    /** @var EntityRepository<Poll> */
    private EntityRepository $repository;
    public function __construct(EntityManagerInterface $em)
    {
        $this->repository = $em->getRepository(Poll::class);

    }
    public function get(PollId $id): Poll
    {
        return $this->find($id) ?? throw
RecordNotFoundException::entity(PollId::class, $id);
    }
    public function getAllForActivateByDate(Date $date): iterable
    {
        return $this->repository->createQueryBuilder('p')
            ->andWhere('p.pollState = :state')
            ->andWhere('p.startAt = :startAt')
            ->setParameter('state', PollState::PUBLISH->value)
            ->setParameter('startAt', $date->format('Y-m-d'))
            ->getQuery()
            ->toIterable()
            ;
    }
    public function getAllForFinishByDate(Date $date): iterable
    {
        return $this->repository->createQueryBuilder('p')
            ->andWhere('p.pollState = :state')
            ->andWhere('p.endAt = :endAt')
            ->setParameter('state', PollState::ACTIVE->value)
            ->setParameter('endAt', $date->subDay()->format('Y-m-d'))
            ->getQuery()
```

```
            ->toIterable()
            ;
    }
```

4.2.8 E-mailové šablony

Jak již bylo zmíněno, tento projekt neobsahuje téměř žádné šablony, neboť frontendová část je řešená zvlášť. Nachází se zde pouze šablony pro odesílání e-mailů a generování PDF dokumentů. Pro demonstraci bude použit kód, který obsahuje šablonu, jež se odešle uživateli na e-mail při vytvoření nového hlasování. Šablona je vytvořena v nástroji Twig, jenž je velmi podobný HTML a umožňuje vkládání proměnných, podmínek a cyklů. Šablona rozšiřuje základní layout, který obsahuje například styly a footer e-mailu. Kromě prostého textu se zde nachází několik proměnných jako například název hlasování, nebo jeho nejzazší možný termín. Dále obsahuje „for“ cyklus pro seznam otázek včetně názvu, kompletního znění a popisu. Na závěr e-mailu je přidáno tlačítko s odkazem na konkrétní hlasování.

```
{% extends 'email/layout.html.twig' %}

{% block content %}
    <h1>Nové hlasování</h1>
    <hr />
    <h2>Právě bylo otevřeno nové online hlasování SVJ vašeho domu.
Zvládnete to naklikat za pár minut.</h2>
    <p>
        Dobrý den,<br />
        je tu nové hlasování pro váš dům. O čem to bude tentokrát?
        <br />
        <br />
        Název hlasování: {{ name }}
        <br />
        Termín - nejpozději do: {{ endAt|date('d. m. Y') }}
        <br />
    </p>
    Seznam návrhů usnesení otevřeného hlasování:
    <ol>
        {% for question in questions  %}
            <li>
                <b>Název otázky: </b> {{ question.name }} <br />
                <b>Celé znění navrhovaného ustanovení</b> <br />
                {{ question.description }} <br />
                {% if question.descriptionExtra is not null %}
                    <b>Popis otázky</b> {{
question.descriptionExtra|raw }}
                {% endif %}
            </li>
        {% endfor %}
    </ol>
```

```
    <p>
        Co můžete odhlasovat dnes, neodkládejte na zítřek.
    </p>

    <row>
        <columns large="5" small="12">
            <button class="radius expanded" href="{{
frontend_url('frontend_poll', {pollId: pollId}) }}">Pojďme
hlasovat</button>
        </columns>
        <columns large="7" small="12"></columns>
    </row>
    <p>
        Hezký den,
    </p>
{% endblock %}
```

4.3 Aktuální stav

Aplikace byla do ostrého provozu uvedena v dubnu 2022. Za tuto dobu eviduje 99 registrovaných SVJ a 273 uživatelů. Toto číslo se může jevit jako poměrně nízké, ale je nutné si uvědomit, že shromáždění se koná zpravidla jednou za rok a nasazení aplikace tak může být relativně zdlouhavý proces. Od uvedení do provozu bylo uspořádáno 32 hlasování. Průměrná velikost registrovaného SVJ byla 33 jednotek. Je tedy zřejmé, že se v tuto chvíli ještě nezaregistrovala většina vlastníků, kterých je dohromady 4 588 a jejich profil byl založen správcem SVJ. Veškeré funkce jsou k dispozici zdarma, kterékoli SVJ tak může aplikace využívat bez dalšího finančního zatížení, a to i v budoucnu, neboť se plánuje, že aplikace zdarma také zůstane. Do budoucna se chystá přidání nových funkcí, jako je biometrické ověření uživatele, nebo například možnost hlídání insolvenčních pohledávek, které by mohly vzniknout SVJ kvůli jednotlivým neplatícím členům. Takovou pohledávku pak musí statutární orgán SVJ přihlásit nejpozději 2 měsíce po zahájení insolvence u insolvenčního soudu, což může být z hlediska časového i právní náročnosti pro SVJ komplikované. Předpokládá se, že tyto funkce by byly zpoplatněny. V současné době probíhá především zvyšování povědomí o eVlastnících, a to jak reklamní kampaní, tak oslovováním přímo vlastníků a investorů, aby se mohla aplikace a její uživatelská základna dále rozrůstat.

5 Závěr

Úvod teoretické části slouží jako seznámení s příslušnými zákony, které charakterizují společenství vlastníku jednotek a je proveden rozbor právních norem umožňujících legitimní a legální online hlasování. Další kapitola se zabývá již existujícími aplikacemi, jež disponují funkcí online hlasování, zároveň je provedena jejich analýza. Následuje literární rešerše z oblasti vývoje webových aplikací za pomocí skriptovacího programovacího jazyka PHP, kde jsou popsány základní funkcionality a postupy práce v tomto jazyce. Na závěr teoretické části je provedena analýza hlavních PHP frameworků Symfony a Laravel, kde bylo jako vhodné prostředí vybráno Symfony.

Praktická část práce je zaměřena na vývoj webové aplikace eVlastníci, který probíhal ve firmě specializované právě na vývoj aplikací a informačních systémů Externity s. r. o., kde jsem již druhým rokem zaměstnán. Úvod vlastní práce se zabývá dotazníkovým šetřením ohledně hlasování společenství vlastníků jednotek, jež probíhalo formou Facebook kampaně a zúčastnilo se jej přes 800 participantů. Zde byly shromážděny důležité poznatky sloužící jako impulz pro zahájení vývoje. Následně jsou popsány procesy, které bylo nutné aplikovat před samotným vývojem. Nachází se zde tvorba diagramů, kde byla jako příklad představena část user flow diagramu, návrhy UX/UI designu a uživatelské testování prototypu, kterého se zúčastnilo 5 participantů. Výsledky z testování byly zpracovány a aplikace přizpůsobena, aby byla uživatelsky přívětivou. Poslední část je zaměřena na backendový vývoj webové aplikace, kde je popsáno její fungování, struktura a dále obsahuje ukázky důležitých částí kódu.

Tato práce byla zaměřena především na vývoj webové aplikace, ale kompletní projekt eVlastníci zahrnuje také webovou stránku a mobilní aplikaci na operační systémy Android a IOS. Všechny tyto části jsou od dubna 2022 v ostrém provozu a veškeré funkcionality jsou k dispozici zcela zdarma. Aplikaci aktuálně využívá 273 uživatelů, kteří již uspořádali 32 hlasování. Hlavním úkolem je nyní zvýšit povědomí o aplikaci, aby se mohla klientela dále rozrůstat, k čemuž slouží probíhající reklamní kampaň podpořená aktivním oslovováním jednotlivých společenství vlastníků jednotek. V budoucnosti se také plánuje přidání nových funkcionalit, z nichž některé budou zpoplatněny. Mohlo by se jednat například o biometrické přihlašování nebo hlídání insolvenčních pohledávek.

6 Seznam použitých zdrojů

ACHOUR, Mehdi, Friedhelm BETZ, Antony DOVGAL, Nuno LOPES, Hannes MAGNUSSON, Georg RICHTER, Damien SEGUY a Jakub VRANA (c2001-2023). PHP Manual [online]. [cit. 2023-01-31]. Dostupné z: https://www.php.net/

BARÁŠEK, Jan (2020). Proč a jak používat frameworky a knihovny [online]. [cit. 2023-02-07]. Dostupné z: https://php.baraja.cz/proc-pouzivat-frameworky

ČÁPKA, David (2016). MVC architektura [online]. [cit. 2023-02-09]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor

ČESKO (1994). Zákon č. 72/1994 Sb., o vlastnictví bytů. [online]. [cit. 2023-02-14]. Dostupné z: https://www.zakonyprolidi.cz/cs/1994-72

ČESKO (2012). Zákon č. 89/2012 Sb., občanský zákoník. In: Sbírka zákonů. ISSN 1211-1244

ČESKO (2016). Zákon č. 297/2016 Sb., o službách vytvářejících důvěru pro elektronické transakce. In: Sbírka zákonů. ISSN 1211-1244

GILMORE, W. J. (2005). Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály. Brno: Zoner Press. Encyklopedie webdesignera. ISBN 80-868-1520-X.

KRULIŠ, Martin (2009). PHP – stručná reference syntaxe a funkcí [online]. [cit. 2023-02-03]. Dostupné z: https://ksvi.mff.cuni.cz/~krulis/vyuka/php/download/php.pdf

LARAVEL (c2011-2023). Laravel – The PHP Framework For Web Artisans. [online]. [cit. 2023-03-03]. Dostupné z: https://laravel.com/docs/10.x.

MÁCA, Jindřich (c2023). Úvod do Symfony frameworku pro PHP [online]. [cit. 2023-02-14]. Dostupné z: https://www.itnetwork.cz/php/symfony/zaklady/uvod-do-symfony-frameworku-pro-php

MORRIS, Will (2023). Best PHP Frameworks For Beginner to Pro Developers [online]. [cit. 2023-02-10]. Dostupné z: https://www.hostinger.com/tutorials/best-php-framework

NAŘÍZENÍ EVROPSKÉHO PARMALEMTNU A RADY (2014). č. 910/2014, o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu (eIDAS) [online]. [cit. 2023-02-14]. Dostupné z: https://eur-lex.europa.eu/legal-content/CS/TXT/?uri=celex%3A32014R0910

ORACLE (2023). Guide to MySQL as an Embedded Database [online]. California, [cit. 2023-02-08]. Dostupné z: https://www.mysql.com/why-mysql/white-papers/guide-to-mysql-as-an-embedded-database/

POTENCIER, Fabien (2020). Symfony 5: The Fast Track. 6th edition. Clichy: Symfony SAS. ISBN 978-2-918390-37-4.

SOUSEDÉ (c2023). Sousedé [online]. [cit. 2023-03-03]. Dostupné z: https://www.sousede.cz/

STAUFFER, Matt (2019). Laravel: Up & Running. 2nd edition. Sebastopol: O'Reilly Media. ISBN 978-1-492-04121-4.

SV ONLINE (c2021). Shromáždění SV online [online]. [cit. 2023-03-03]. Dostupné z: https://www.svonline.cz/

SVJ APLIKACE (c2018-2023). SVJ Aplikace [online]. [cit. 2023-03-03]. Dostupné z: https://svjaplikace.cz/

SYMFONY (2022). Symfony Documentation 6.3. [online]. [cit. 2023-02-15]. Dostupné z: https://symfony.com/doc/6.3/index.html

ŠULEK, Marcel (2014). Škálovatelný PHP hosting s podporou Git deploymentu [online]. [cit. 2023-02-01]. Dostupné z: https://www.root.cz/clanky/skalovatelny-php-hosting-s-podporou-git-deploymentu/

VALKOVIČ, Patrik (c2023). Git – Historie a principy [online]. [cit. 2023-02-10]. Dostupné z: https://www.itnetwork.cz/programovani/git/git-tutorial-historie-a-principy

VONDRA, Marek (c2023). Docker – Teorie a instalace [online]. [cit. 2023-02-16]. Dostupné z: https://www.itnetwork.cz/site/docker/docker-teorie-a-instalace

WELLING, Luke a Laura THOMSON (2017). Mistrovství PHP a MySQL. Brno: Computer Press. ISBN 978-80-251-4892-1.

7 Seznam obrázků, tabulek, grafů a zkratek

7.1 Seznam obrázků

7.2 Seznam grafů

7.3 Seznam použitých zkratek

SVJ – Společenství vlastníků jednotek
PHP – Hypertext preprocessor, hypertextový preprocesor
HTML – Hypertext markup language, hypertextový značkovací jazyk
XML – Extensible markup language, rozšiřitelný značkovací jazyk
MVC – Model-view-controller, model-zobrazení-kontroler
HTTP – Hypertext transfer protocol, hypertextový přenosový protokol
FTP – File transfer protocol, souborový přenosový protokol
UX – User experience, uživatelská zkušenost
UI – User interface, uživatelské rozhraní

SMS – Short message service, Služba krátkých textových zpráv
GDPR – General data protection regulation, obecné nařízení o ochraně osobních údajů
PDF – Portable document format, přenosný formát dokumentu
UID – Unique identifier, unikátní identifikátor