FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Portál pro videoreporty
Student:	Dmitry Belov
Vedoucí:	Ing. Jan Matoušek
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je návrh a implementace webového nástroje pro snadnou tvorbu a plnění video reportů, jejichž účelem je prověřit stav fyzického objektu (např. nemovitosti, kdy zájemce nemá čas na osobní prohlídku), který je popsaný v zadání reportu. Služba by měla být dostupná uživatelům z internetu skrze webový prohlížeč.

Postupujte dle následujících kroků:

1) Proveďte průzkum stávajících řešení.
2) Analyzujte potřeby cílových uživatelů.
3) Na základě analýzy specifikujte funkční a nefunkční požadavky.
4) Metodami softwarového inženýrství navrhněte vhodné řešení.
5) Implementujte prototyp softwarového řešení pomocí vhodně zvolených technologií.
6) Prototyp podrobte vhodným testům.
7) Na základě poznatků z testování navrhněte další směřování vývoje.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 28. ledna 2022 v Praze.

Bakalářská práce

PORTÁL PRO VIDEOREPORTY

Dmitry Belov

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Jan Matoušek
11. května 2022

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Rád bych poděkoval vedoucímu své bakalářské práce Ing. Janu Matouškovi za jeho pomoc, rady a nápady. Byl ochotný mou bakalářskou práci převzít od jiného vedoucího a za to mu velice děkuji.
Taky bych rád poděkoval své rodině a přátelům, kteří mě podporovali po dobu studia na vysoké škole. Vysokou školu bych bez jejich podpory a pomoci nebyl schopen dokončit.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2022 .....

Abstrakt

Práce se zaměřuje na vývoj prototypu webového portálu. Nejdříve jsou pomocí analýzy konkurence stanoveny požadavky kladené na prototyp. Následně je pomocí metod softwarového inženýrství navrženo vhodné řešení pro splnění požadavků. Navrženy jsou jak technologie použité v implementaci, tak i potřebný funkcionál prototypu. Po návrhu je uvedena ukázka použití nejdůležitějších technologií v implementaci prototypu. Celá implementace je psána v JavaScriptu s využitím TypeScriptu a dalších nástrojů a knihoven jako je například Express a React. Výsledná implementace je následně otestována a jsou stanoveny nedostatky implementovaného prototypu.

Výsledkem práce je funkční prototyp splňující identifikované požadavky. Na práci by se dalo navázat doděláním prototypu do produkční verze, jeho obohacením o další funkcionál a odstraněním nedostatků prototypu, které vyplynuli z testování.

Klíčová slova video report, inzerce, webový portál, prototyp, metody softwarového inženýrství, JavaScript, TypeScript, React, Express

Abstract

Thesis focuses on the development of a web portal prototype. First, the requirements for the prototype are determined using competition analysis. Subsequently, using software engineering methods, a suitable solution is designed to meet the requirements. Both the technologies used in the implementation and the necessary prototype functional are designed. After the design, I will demonstrate the use of the most important technologies in the implementation of the prototype. The entire implementation is written in JavaScript using TypeScript and other tools and libraries like Express and React. The finished implementation is then tested and the shortcomings of the implemented prototype are determined.

The result is a functional prototype fulfilling the identified requirements. The work could be followed by finishing the prototype in to the production version, enriching it with another functionality and by fixing shortcomings of prototype.

Keywords video report, advertising, web portal, prototype, software engineering methods, JavaScript, TypeScript, React, Express

Seznam zkratek

3D	three dimensions
API	Application Programming Interface
AVI	Audio Video Interleave
AWS	Amazon Web Services
S3	Simple Storage Service
CSS	Cascading Style Sheets
DRY	Don't repeat yourself
EA	Enterprise Architect
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
JWT	JSON Web Token
KISS	keep it simple, stupid
MIT	Massachusettském technologickém institutu
MKV	Matroska Video
MOV	Apple QuickTime Multimedia File
MP4	MPEG-4 Part 14
MVC	Model-view-controller
npm	Node Package Manager
OOP	Object-oriented programming
ORM	Object–relational mapping
PDF	Portable Document Format
REST	Representational State Transfer
SASS	Syntactically Awesome Style Sheets
SEO	Search Engine Optimization
SHA	Secure Hash Algorithm
SOAP	Simple Object Access Protocol
SPA	Single-Page App
UI	User Interface
WMV	Windows Media Video
YAML	YAML Ain't Markup Language

Kapitola 1
Úvod

V dnešní době vyřizujeme spoustu věcí z pohodlí domova jako například virtuální prohlídky bytu nebo aut v inzercích pomocí řady služeb. A na jejich základě se často rozhodujeme a účastníme osobních prohlídek před koupí. Osobní prohlídky ale mohou být časově náročné nebo se nacházet ve vzdálených oblastech. Dalším problémem může být i nedostatečná kvalita inzercí nebo třeba i klamavé inzerce.

Chtěl bych proto vytvořit webový portál, který by nabízel alternativu účasti na osobní prohlídce. Portál by umožnil inzerovat osobní prohlídky, které by lidé potřebovali nahradit, a protistrana čili náhradníci, by následně natočili video report prohlídky a dostali za něj finanční ohodnocení. Video report by zachycoval požadovanou osobní prohlídku objektu z inzerce. Lidé, kteří by potřebovali nahradit svoji účast na osobní prohlídce, by tak mohli ušetřit čas a náhradníci by mohli využívat své lokality a volného času k přivýdělku.

Hlavním cílem mé bakalářské práce je vytvoření prototypu webové aplikace pro snadnou tvorbu a plnění video reportů, jejichž účelem je zachycení stavu fyzického objektu dle zadání. Zadáním se rozumí prohlídka inzerovaného objektu a její zachycení na video.

Práce je logicky rozdělená do kapitol. V kapitole *Průzkum* jsou uvedena současná řešení a z nich plynoucí požadavky na tvorbu aplikace. V kapitole *Návrh* pak vytvořím podklady a navrhnu vhodné řešení pro implementaci prototypu pomocí metod softwarového inženýrství. Při návrhu bude cílem navrhnout řešení splňující požadavky stanovené v kapitole *Průzkum*. Poté v kapitole *Implementace prototypu* uvedu technologie, jejich praktické využití v aplikaci a postup implementace prototypu. Na závěr prototyp podrobím testům v kapitole *Testování* a z něho pak vyvodím další možné směřování vývoje v kapitole *Závěr*.

Kapitola 2
Průzkum

Hlavním cílem této kapitoly je získání požadavků cílových uživatelů na vyvíjenou aplikaci. Tyto požadavky považuji jako jeden ze základních stavebních kamenů pro vytvoření kvalitní aplikace s reálným využitím. Nejdříve sestavím průzkum současných řešení a specifikaci produktu, na jejichž základě následně sepíšu vypozorované požadavky cílových uživatelů. Závěrem pak shrnu dopady průzkumu na následující návrh a vývoj aplikace.

2.1 Specifikace produktu

Portál bude vyvíjen pro český trh. Obecně má aplikace působit jako platforma umožňující zadávat a plnit video reporty, které mají zachytit na video stav objektu (oblast, nemovitost, auto...) dle zadaní. Za splnění pak dostane plnitel práce finanční ohodnocení od zadavatele práce. Uživatelé portálu budou moci takové zakázky buď sami vytvářet nebo je plnit. Dnešní doba dovoluje plnit zakázky skoro každému díky vysokému rozšíření chytrých mobilních zařízení s kvalitními fotoaparáty. Ty vlastní v dnešní době přes 70 % Čechů dle článku na mediaguru.cz [1]. Počítám proto s tím, že většina uživatelů bude využívat portál skrze mobilní zařízení. Verze pro počítače bude také podporována a nebude se od té mobilní funkcionálem nikterak lišit. Přístup k aplikaci by pak měl být skrze webový prohlížeč.

Při tvorbě portálu se budu soustředit hlavně na video reporty, které mají zachytit předměty z inzercí. Předmětem práce bude tedy účast plnitele práce na osobní prohlídce inzerovaného objektu, kterou natočí a následně poskytne report zadavateli práce. V práci bude proto její zadavatel moci pro zjednodušení procesu připojit odkaz na inzerci.

Jako předměty z inzercí, které má za úkol video report zachytit, budu uvažovat především nemovitosti a automobily, zejména kvůli jejich ceně, která většinu uživatel nabádá k opatrnějšímu výběru a osobní prohlídce, kterou má portál buď zcela nahradit nebo poskytnout dostatek informací pro rozhodnutí o její účasti.

2.2 Současná řešení

V následující části je popsána identifikace a následné ohodnocení stávajících řešení, která jsou považována za konkurenci. Tato konkurence poslouží především jako inspirace a osnova pro další sestavení požadavků cílových zákazníků.

2.2.1 Identifikace současných řešení

Pří identifikaci konkurence budu vycházet ze *Specifikace produktu* uvedené v 2.1. Podobné produkty se tak budou soustředit především na náhradu účasti v osobních prohlídkách automobilu a nemovitostí.

Nebudu ale hledat pouze řešení, které se zabývá touto problematikou. Do hledání zapojím i služby umožňující například přidání virtuální / video prohlídky k inzerci. Tyto dodatečné informace k inzerátu do značné míry usnadňují rozhodnutí o účasti na osobní prohlídce a lze je považovat i za její alternativu.

Dle specifikace výše jsem nalezl produkty uvedené v tabulce 3.1. Hledání bylo provedeno prostřednictvím vyhledávače Google. K produktům je také v tabulce uvedená i skupina služeb, které zastupují. Hledané výrazy odpovídají textu popisující skupinu v tabulce k datu 10. 3. 2022.

Tabulka 2.1 Identifikovaná řešení

Název služby	Zastupuje služby
autobezobav.cz	pomoc s prohlídkou auta
remax-czech.cz	pomoc s prohlídkou nemovitosti a její inzerce
facebook.com	sociální sítě

2.2.2 Evaluace současných řešení

V dané podkapitole ohodnotím identifikovaná současná řešení uvedená v tabulce 3.1. Každé z nich zastupuje skupinu služeb, které pro aplikaci představují konkurenci. Hodnocení bude do určité míru aplikovatelné i pro celou skupinu, kterou produkt zastupuje.

Hodnotit jednotlivé produkty budu dle technického stavu stránek a konkurenceschopnosti s vyvíjenou aplikací.

V konkurenceschopnosti se budu zaměřovat především na srovnaní produktu s vyvíjenou aplikaci. Srovnávat budu především jakou výhodu či nevýhodu by neslo využití takového řešení při náhradě účasti osobní prohlídky. Nejedná se tedy tolik o celkové srovnaní produktu jako spíš o porovnání nabízených služeb. Na závěr srovnání vždy uvedu obecné shrnutí.

U technického stavu budu hodnotit celkovou kvalitu stránek pomocí nástroje Lighthouse. Lighthouse objektivně shrnuje kvalitu stránky podle řady parametrů. Předtím něž ohodnotím samotné produkty bych chtěl podrobněji popsat tento nástroj.

Lighthouse

Tento nástroj od společnosti Google umožňuje jednoduše, objektivně a rychle hodnotit kvalitu webových stránek. Hodnocení bude probíhat na základě těchto kategorií:

Performance (výkon) Tato část je zaměřena na to, jak je webová stránka optimalizovaná pro interakci s uživatelem. Jsou v ní zahrnuty i doporučení pro optimalizaci daného aspektu.

Best Practises Daná část se zabývá dodržením best practises, které mají značný dopad na kvalitu kódu. Krom ohodnocení nabízí nástroj i tipy pro možná zlepšení kódu.

Accessibility (přístupnost) Má za úkol zjistit, zda všichni uživatelé mohou stránky efektivně využívat a procházet. Stejně jako předchozí testy nabízí i tento tipy na zlepšení.

SEO Zjišťuje, zda je stránka optimalizovaná pro hodnocení výsledků vyhledávačem. Na základě tohoto hodnocení vyhledávač řadí výsledky dle relevance.

Daný popis byl získán volným překladem ze stránky web.dev [2]. Každá z uvedených kategorií je ohodnocená na stupnici 0-100. Indikátorem úspěchu v testu je zabarvení skóre do zelené barvy. Oranžová pak označuje, že se oblast dá zlepšit a červená má značit velký nedostatek stránky, který je potřeba opravit.

Je nutno dodat, že daný nástroj hodnotí jen a pouze technický stav. Nemusí tedy nutně říkat nic o tom, jak se stránky používají a jakou zkušenost s nimi mají uživatele. Tento parametr je zkoumán především pro srovnání technického stavu v závěru práce.

2.2.3 autobezobav.cz

Služba se soustředí na pomoc s výběrem ojetých aut. Nabízí náhradu účasti na osobní prohlídce, konzultace ohledně vozu nebo i pomoc s jeho výběrem. Služba funguje pro zákazníky jako platforma, na níž si bez registrace vyberete technika z oblasti, kde se auto z inzerátu nachází a on pak provede osobní prohlídku po jejímž závěru je následně zákazníkovi sdělen verdikt ohledně auta. Konzultace je u všech techniků zdarma a až za fyzickou prohlídku se platí ke dni 24. 3. 2022 průměrně 3 000 korun českých. Komunikace mezi klientem a technikem probíhá mimo stránky.

Stránka umožňuje práci všem technikům, kteří splňují podmínky a nebere si žádný podíl na výdělku slouží tak pouze jako prostředník. Každý technik má hodnocení, které mu uděluje zákazník pro dokončení objednávky.

Výhody a nevýhody řešení

Výhody:

- verdikt profesionála k prohlídce
- systém hodnocení techniků
- moderní, kvalitně zpracovaný a přehledný web
- nevyžaduje registraci
- nulové poplatky pro technika

Nevýhody:

- nedostatečná filtrace a řazení techniků
- přísné podmínky pro přijetí techniků
- cena

Na rozdíl od tohoto produktu je vyvíjený portál více všeobecný a necílí na poskytnutí pomoci s výběrem, ale pouze na nahrazení účasti na prohlídce. Tento portál je tedy vhodný pro někoho, kdo chce pomoci s výběrem od profesionála na rozdíl od pouhého nahrazení fyzické účasti. Profesionalita techniků se projevuje i v přísných podmínkách pro jejich přijetí a v ceně za prohlídku, pohybující se ke dni 24.3.2022 okolo částky 3 000 korun českých.

Stránky působí profesionálně a přehledně. Jednoduše jsem na nich nalezl dostatek informací o službě. K systému hodnocení také nemám výhrady a považuji jej za dobře zpracovaný. Postrádal jsem ale filtrace a řazení techniků, které šlo filtrovat pouze podle oblasti a místa.

Lighthouse

Daný produkt má nejhorší hodnocení ze všech srovnávaných. Jeho výsledky však nejsou nijak špatné. Nejhorší hodnocení má produkt v položce Accessibility, což je způsobeno chybějícími atributy u obrázků a odkazů. Kategorie Best Practices je také poměrně slabá zejména kvůli zranitelnosti použitých jQuery knihoven. SEO a Performance hodnocení je naopak na dostatečně dobré úrovni.

Obrázek 2.1 Hodnocení Lighthouse pro autobezobav.cz

2.2.4 remax-czech.cz

Všechny informace ohledně Remax byly čerpány ke dni 26.3.2022 ze stránek remax-czech.cz. Remax je jednou z největších realitních služeb nejen v České republice ale i na evropském trhu. Nabízejí všechny služby, co se týče nemovitostí, od pomoci s prodejem až po jejich inzercí, kterých mají přes 6 000. Pracuje na nich i přes 1 300 realitních makléřů, kteří mohou pomoc s výběrem i prohlídkou nemovitosti. Krom toho nabízejí i virtuální prohlídky některých nemovitostí v inzercích. Vzhledem k povaze vyvíjené aplikace se zaměřím pouze na realitní makléře a inzerce k jejichž prohlídce není třeba registrace. Komunikace s klientem pak probíhá mimo službu.

Realitní makléř nabízí nejen náhradu účasti na osobních prohlídkách, ale komplexně se stará i o celé zařizování procesu koupě, prodeje nebo pronájmu nemovitosti.

Co se týče virtuálních prohlídek většinou se jedná o naskenovaný 3D model nemovitosti nebo její video prezentace. Takovéto prohlídky však nalezneme zejména u dražších nemovitostí a u valné většiny inzercí jsou přítomny pouze obrázky a popis.

Výhody a nevýhody řešení

Výhody:

- silné postavení na trhu
- velké množství inzercí a makléřů
- dobrá filtrace, řazení a vyhledávaní makléřů / inzerátů
- není potřeba registrace

Nevýhody:

- osobní prohlídky pouze nemovitostí, které nabízí Remax
- nepřehledné a designově zastaralé stránky

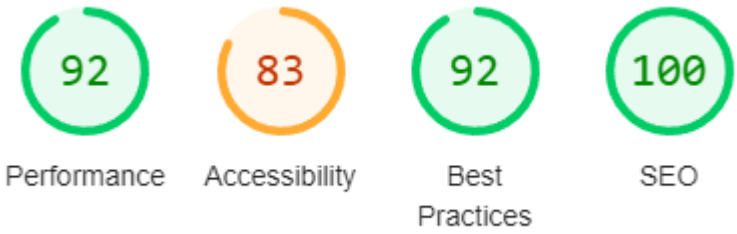
Remax je nepochybně velkým hráčem na českém trhu a nabízí velké množství makléřů a inzercí nemovitostí. Tito makléři, ale většinou umožňují pouze prohlídky nemovitostí inzerovaných na stránce Remax. Vzhledem k počtu těchto inzercí to ale nepovažuji za velký nedostatek. Navíc u nich bývá častým předmětem sporu i objektivita, která může být do značné míry ovlivněná například výší provize pro makléře. To však není faktem ale spíše spekulací, která se týká obecné důvěry v makléře jako takové.

Makléři nejsou ale jedinou stránkou konkurence, jsou jim i virtuální prohlídky. Jak již bylo řečeno v 2.2.1, tak do značné míru snižují potřebu osobní prohlídky. Předpokládám proto, že uživatele vyvíjeného portálu budou našich služeb využívat spíše k prohlídkám nevirtualizovaných nebo nedostatečně zdokumentovaných / kvalitních inzercí.

Stránky jsou kvalitně zpracované, ale kvůli objemu služeb i nepřehledné. Filtrace, řazení a vyhledávání inzercí a realitních makléřů je na výborné úrovni.

Lighthouse

Daný produkt získal nejlepší výsledky ze všech tří. V kategorii SEO dosahuje v hodnocení maximálního skóre což je skvělý výsledek. Nejslabší částí je Accessability, ve které ale také dosahuje dobrých čísel. Důvodem je nízký kontrast s pozadím a chybějícími parametry u odkazů.



Obrázek 2.2 Hodnocení Lighthouse pro remax-czech.cz

2.2.5 facebook.com

Tato sociální síť je jednou z nejpopulárnějších na českém trhu a v Česku jí používá přes 70 % populace, podle článku publikovaném na mediaguru.cz [3]. Pro jeho používání vyžaduje Facebook registraci. Umožňuje posílaní zpráv, videí, dokumentů. Nabízí toho ale mnohem víc, jako je třeba tvorba skupin pro uživatelé se stejnými zájmy. Tyto skupiny umožňují uživatelům, kteří jsou členy, vytvářet příspěvky, na které můžou ostatní členové dávat reakce a komentovat je.

S tímto funkcionálem je možné kompletně nahradit vyvíjený portál. Kde si stačí založit skupinu, kam budou lidí dávat své zadání a nabídky. Po dokončení by si mohli přímo na platformě předat video report. Toto není jedinou možností a jistě lze pomocí Facebooku vymyslet více způsobů náhrady za vyvíjený portál.

Výhody a nevýhody řešení

Výhody:

- rozšíření
- množství kvalitních nástrojů umožňujících komunikaci

Nevýhody:

- chybí systém hodnocení
- chybí filtrace a řazení příspěvku ve skupinách
- není přizpůsobený k této činnosti

Facebook by jistě mohl platformu do určité míry nahrazovat. Jeho výhodou je vysoké rozšíření a jednoduché používaní. Facebook je tedy silným konkurentem, který ale není bez nedostatků. Jeho velkou nevýhodou je například nepřizpůsobenost k takovému způsobu použití (tedy jakožto portál pro video reporty).

Uvažujme nyní že každou prací je zpráva někomu nebo příspěvek ve skupině. V takovém případě bude chybět filtrování, řazení a přehled prací, na kterých se uživatel podílí. K tomu všemu uživatel postrádá i jakýkoliv systém hodnocení, který by mohl usnadnit například výběr plnitele práce. Tyto všechny nedostatky mohou vést, s rostoucím počtem prací a uživatelů, k zahlcení pracemi ve skupině a k obecnému znepřehlednění. Navíc jsou dané vlastnosti dle mého názoru pro vyvíjený portál klíčové.

Zpracování stránek považuji za skvělé. Zobrazení na mobilních zařízeních sice trochu zaostává, ale na druhou stranu Facebook poskytuje pro Android i iOS aplikaci, což danou skupinu zařízení pokrývá.

Lighthouse

Ze všech tří dopadl Facebook průměrně. Hodnocení Performance dosahuje maximálního počtu bodů. Ostatní parametry jsou přijatelné. Nejhůř jsou na tom parametry Accessibility a SEO s 83 body. Přitom v SEO je toto skóre způsobeno pouze blokací crawleru pro určité odkazy. Crawler je nástrojem pro hledání odkazů odpovídajících hledanému výrazu vyhledávačem. Můžeme mu zakázat přístup k určitým odkazům stránky. To je zřejmě uděláno z bezpečnostních důvodu. Accessibility pak dopadlo hůř zejména kvůli nedostatečnému kontrastu popředí a pozadí.

Obrázek 2.3 Hodnocení Lighthouse pro facebook.com

2.3 Požadavky zákazníků

V této části uvedu požadavky cílových zákazníků, vyplývající z předchozího průzkumu a ze specifikace produktu. Ty, které vycházejí ze současných řešení, budou stanoveny především na základě výhod a nevýhod. Požadavky se budou týkat funkcionálu aplikace a jejího technického stavu. Jednotlivé požadavky rozdělím do dvou částí podle jejich typu na Funkční a Nefunkční.

2.3.1 Funkční požadavky

V této části uvádím funkční požadavky. Funkční požadavky, jak již vyplývá z názvu, jsou požadavky, které se zaměřují na funkcionál aplikace. Při stanovení těchto funkčních požadavků se tedy soustředím na to, co musí aplikace umět nebo co bude umožňovat / jaký funkcionál by měla mít.

U každého funkčního požadavku uvedu kromě stručného popisu i jeho Složitost a Prioritu. Ty ohodnotím na stupnici: vysoká, střední, nízká. U Složitosti budu vycházet z osobních zkušenosti. Prioritu pak určím na základě klíčovosti pro naplnění potřeb zákazníků a provázanosti s ostatními požadavky.

F01 – Registrace a přihlašování

Portál by měl poskytovat základní registraci prostřednictvím emailu, uživatelského jména a hesla. Přihlašování pak bude uskutečněno pomocí uživatelského jména a hesla.

Složitost: Nízká
Priorita: Vysoká

F02 – Obnovení hesla pomocí emailu

Portál nabídne možnost obnovení hesla pomocí registrovaného emailu. Na registrovaný email bude zaslán odkaz, pomocí něhož půjde zadat nové heslo.

Složitost: Střední
Priorita: Střední

F03 – Editace údajů profilu

Portál umožní editovat údaje profilu. Údaje budu dělit na povinné (zadané při registraci) a nepovinné. Nepovinné údaje mají za úkol popsat uživatele a jeho zájmy a nebudou po registraci vyplněné.

Složitost: Střední
Priorita: Vysoká

F04 – Vytvoření a spravování prací

Portál umožní registrovaným uživatelům vytvářet a spravovat práce. Při jejím vytvoření se bude mimo jiné požadovat i odkaz na inzerci, která se práce týká. Pod spravováním je pak myšleno výběr plnitele ze zájemců, úprava objednávky, nahrání výsledků a její kompletní zrušení.

Složitost: Vysoká
Priorita: Vysoká

F05 – Plnění práce

Portál umožní registrovaným uživatelům plnění práce. Součástí plnění je projevení zájmu o plnění, zrušení zájmu, nahrání výsledku a vypovězení plnění práce.

Složitost: Vysoká
Priorita: Vysoká

F6 – Notifikace o změně stavu práce

Při změně stavu práce, jako je například: odeslání výsledků, přijetí objednávky plnitelem..., bude na registrovanou emailovou adresu zaslán uživateli email s informací o změně stavu.

Složitost: Vysoká
Priorita: Střední

F07 – Nahrání a přehrání videa

Výsledkem splněné práce bude video dle jejího zadání. Proto bude potřeba aby portál umožnil jeho nahrání a následné přehrání. Cílem bude umožnit přehrání v co nejlepší kvalitě.

Složitost: Vysoká
Priorita: Vysoká

F08 – Udělování recenzí

Účastníci práce se budou moct po splnění objednávky vzájemně ohodnotit. Hlavními parametry hodnocení bude samotná recenze a ohodnocení na škále od jedné do pěti. Tyto recenze budou veřejné přístupné a nepůjdou editovat.

Složitost: Střední
Priorita: Vysoká

F09 – Procházení, filtrování a řazení prací

Uživatel portálu bude moct procházet, filtrovat a řadit práce dle jejich parametrů. Práce rozdělím do dvou hlavních kategorií: ty na kterých se podílí uživatel a na ty zveřejněné od ostatních uživatelů, které zatím nikdo neplní. Zveřejněné práce budou moct procházet i nepřihlášení uživatelé. Pro účast na nich bude ale vyžadována registrace nebo přihlášení.

Složitost: Vysoká
Priorita: Vysoká

F10 – Zobrazení profilu uživatele a jeho recenzí

Ke každému uživateli společně s jeho profilem půjdou zobrazit i jemu udělené recenze, které bude možno řadit podle ohodnocení na škále. U recenzí bude vždy uvedeno skóre, samotná recenze, kdo a kdy jí udělil a identifikační číslo objednávky, které se týká.

Složitost: Nízká
Priorita: Vysoká

F11 – Administrativní část

Má za úkol umožnit plynulé a jednoduché spravování webového portálu. Administrativní část aplikace umožní smazání nevhodných recenzí nebo prací. Umožněna bude i úplná blokace uživatelů nebo přiřazení administrátorských práv.

Složitost: Vysoká
Priorita: Nízká

2.3.2 Nefunkční požadavky

V této části uvedu nefunkční požadavky. Nefunkční požadavky se týkají obecného fungování systému. Kladou tedy omezení na systém a říkají, jakým způsobem by měl být navržen. Každý z nefunkčních požadavků by měl odpovídat na otázku: Jaký by systém měl být?

N01 – Dostupnost z internetu prostřednictvím prohlížeče

Aplikace bude dostupná přes webový prohlížeč. Při implementaci prototypu budu cílit hlavně na podporu prohlížeče Chrome. Ten jsem zvolil, protože je to statisticky nejpoužívanější prohlížeč na světě, jak je vidět na obrázku 2.4 a používá jej přes 63 % uživatelů internetu. Avšak v další fázi vývoje aplikace, bude potřeba otestovat fungování i v jiných prohlížečích a vydat případné opravy.

Obrázek 2.4 Podíl internetových prohlížečů na celosvětovém trhu dle [4]

N02 – Rozšiřitelnost

Aplikace by měla být jednoduše rozšiřitelná a udržovatelná. Toho dosáhneme vhodným návrhem architektury a dodržováním Best Practices jako je například DRY (Don't Repeat Yourself) nebo KISS (Keep It Simple Stupid). K naplnění požadavku dále využiji svých znalostí, které jsem získal během studia z předmětu jako jsou například BI-SI1, BI-SI2 a BI-OOP.

N03 – Responzivita

Responzivita je aktuálně velice důležitou součástí každé webové stránky. Zajišťuje zobrazení na všech velikostech obrazovek se zachováním funkcionality. Vyvíjená aplikace by proto měla být také responzivní. To vyplývá z potřeby podporovat zobrazování na mobilních zařízení. Ty navíc aktuálně představuji nadpoloviční většinu uživatel, jak plyne ze statistky uvedené na obrázku 2.5.

Obrázek 2.5 Podíl zařízení na celosvětovém trhu dle [5]

Při její implementaci budu přizpůsobovat zobrazení aplikace pro mobilní zařízení, tablety a počítače. U každé z těchto skupin jsem na základě statistik podílu velikosti obrazovek na celosvětovém trhu, ke dni 26.3.2022 ze statcounter.com, zvolil nejběžnější rozlišení, pro které budu aplikaci přizpůsobovat:

- počítač: 1920 x 1080
- tablet: 768 x 1024
- mobilní telefon: 360 x 800

N04 – Česká lokalizace

Vzhledem k cílovému trhu je potřeba aby byla webová aplikace lokalizována do českého jazyka. To znamená, že všechny ovládací prvky, texty, popisy... na webové stránce by měly být v češtině.

2.4 Shrnutí

Cílem této kapitoly bylo především získání klíčových požadavků kladených na prototyp aplikace. Tyto požadavky se mi podařilo identifikovat za pomoci průzkumu současných řešení a specifikace produktu v části Požadavky zákazníků 2.3. Identifikované požadavky využiji pro další návrh a vývoj prototypu. Při návrhu a implementaci se budu soustředit hlavně na splnění funkčních požadavků s vysokou a střední prioritou a všech nefunkčních požadavků. Na požadavky s nízkou prioritou bych se chtěl soustředit po implementaci prototypu v další iteraci vývoje projektu.

Kromě požadavků jsem ale zároveň získal dostatek informaci o technické stránce současných řešení. Ty poslouží také k následnému srovnání v kapitole Testování 5 s již vyvinutým prototypem aplikace.

Kapitola 3
Návrh

Cílem této kapitoly je návrh vhodného řešení pro následující implementaci prototypu. K návrhu použiji metody softwarového inženýrství jako jsou například případy užití, diagram aktivit a doménový model.

Na návrh fungování prototypu, se podívám v částech Případy užití a Diagram aktivit, které mi dají jasnou představu, jak by měl vyvíjený prototyp fungovat. Budu zde vycházet především z funkčních a nefunkčních požadavků uvedených v kapitole 2. Následně na základě návrhu fungování se podívám na návrh technologií. V něm uvedu, jakým způsobem a pomocí jakých nástrojů budu prototyp implementovat. Závěrem kapitoly shrnu její dopad na navazující implementaci.

3.1 Případy užití

V části Případy užití definuji vzájemnou interakci aplikace s jednotlivými aktéry, které identifikuji v 3.1.1. Vzájemné interakce uvedu v jednotlivých bodech (případech), které budou vycházet z funkčních požadavků uvedených v 2.3. Vzhledem k množství těchto bodů jsem se rozhodl je sdružit do podkapitol tykajících se stejného tématu.

Každý z bodů bude obsahovat popis případu a seznam aktérů, kteří se ho účastní. U každé podkapitoly uvedu i model případu užití pro lepší vizualizaci. Modely jsem vytvářel pomocí nástroje EA (Enterprise Architect), se kterým jsem byl v průběhu studia seznámen.

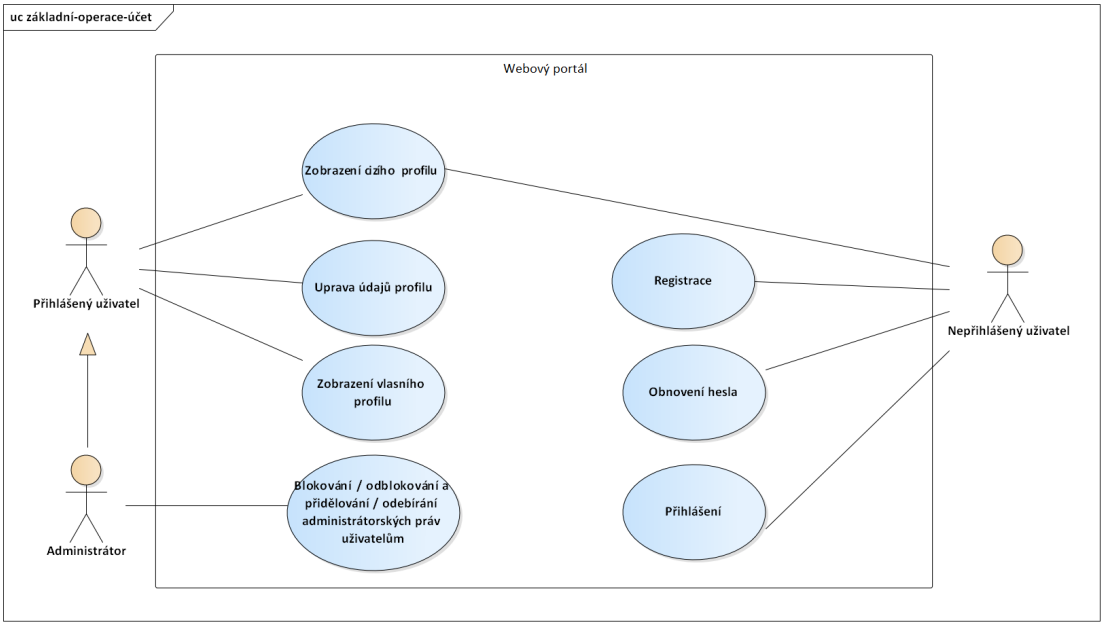
3.1.1 Identifikace aktérů

Toho, kdo se účastní interakce s aplikací nazveme aktérem. Na základě funkčních požadavků z kapitoly 2 jsem stanovil následující aktéry:

- přihlášený uživatel portálu
- nepřihlášený uživatel portálu
- zájemce – uživatel portálu, který má zájem práci plnit
- přiřazený uživatel – uživatel portálu, který byl vybrán zadavatelem k plnění práce
- plnitel – uživatel portálu, který plní / plnil vytvořenou práci
- zadavatel – uživatel portálu, který zadal práci
- administrátor – uživatel portálu, který může moderovat portál (mazat nevhodné recenze, blokovat uživatele...)

3.1.2 Základní operace s uživatelským profilem a účtem

V této podkapitole jsou uvedený všechny interakce týkající se operací s uživatelským profilem a účtem. Na obrázků 3.1 je pak uveden Model případu užití týkající se těchto operací.



Obrázek 3.1 Model pro základní operace s uživatelským profilem a účtem

U01 - Registrace

Případ užití umožňuje nepřihlášenému uživateli registraci do portálu. Systém bude po uživateli vyžadovat vyplnění uživatelského jména, emailu, hesla a jeho zopakování. Uživatelské jméno musí být unikátní a zadaný email nesmí používat jiný uživatel registrovaný v portálu. Heslo by mělo obsahovat minimálně 8 znaků, které musí být kombinací čísel a písmen a musí se také shodovat s jeho zopakováním. Po validním vyplnění těchto údajů do formuláře bude vytvořen účet a uživatel se bude moct přihlásit.

V případě chybného vyplnění, ať už v políčku určeném pro email, uživatelské jméno nebo heslo, bude uživatel upozorněn a vyzván k opravě nevalidních údajů. Pokud bude email nebo uživatelské jméno již zabráno bude na tento fakt uživatel také upozorněn.

Aktéři: nepřihlášený uživatel portálu

U02 – Přihlášení

Případ užití umožňuje nepřihlášenému uživateli přihlášení do portálu. Pro přihlášení bude potřeba zadat pouze své uživatelské jméno a heslo do přihlašovacího formuláře. Uživatel bude při neúspěšném přihlášení upozorněn. Pokud uživatel heslo nebude znát, bude moct jednoduše z formuláře přejít na formulář pro obnovu hesla. Po úspěšném přihlášení bude uživatel přesměrován do portálu.

Aktéři: nepřihlášený uživatel portálu

U03 – Úprava údajů

Případ užití umožňuje přihlášenému uživateli v nastavení měnit údaje zadané při registraci (jen heslo a email) a také i údaje dodatečné. Jimi jsou: popis profilu, telefonní číslo, kraj, ve kterém se uživatel nachází a specializace. Specializacemi jsou kategorie prací, na které se uživatel primárně soustředí. Základní kategorie budou tři: auta, nemovitosti a ostatní. Tyto dodatečné nebo také nepovinné údaje nebudou po registraci vyžadovány.

Aktéři: přihlášený uživatel portálu

U04 – Zobrazení profilu uživatele

Případ užití umožňuje přihlášenému uživateli zobrazit si vlastní nebo cizí profil. Vlastní profil se bude zobrazovat na úvodní obrazovce po přihlášení. Cizí profily pak půjdou zobrazit u publikované práce, kde bude dostupné zobrazení profilu toho, kdo ji vytvořil. Pokud se jiný uživatel přihlásí k plnění mnou vytvořené práce, bude pro mě také umožněno zobrazení jeho profilu v dané práci.

V profilu uživatele bude uvedeno uživatelské jméno, popis, email, telefonní číslo, specializace, počet vytvořených a splněných objednávek a skóre vycházející z jemu udělených recenzí a jejich celkový počet. Nalezneme zde i možnost zobrazení udělených recenzí, které půjdou řadit podle skóre a stáří.

Aktéři: přihlášený uživatel portálu

U05 – Obnovení hesla

Případ užití umožňuje nepřihlášenému uživateli obnovit zapomenuté heslo. Pro jeho obnovení bude potřeba vyplnit email, kterým se uživatel registroval. Na něj bude zaslán odkaz, pomocí kterého bude uživatel moct zadat nové heslo. Tento odkaz bude mít platnost 15 minut. Po vypršení této platnosti bude uživatel při pokusu o nastavení nového hesla upozorněn na neplatnost odkazu. Pro zadání nového hesla uživatel vyplní a odešle formulář, který bude přístupný skrze zaslaný odkaz. Při jeho volbě platí stejná pravidla jako při tvorbě hesla během registrace.

Aktéři: nepřihlášený uživatel portálu

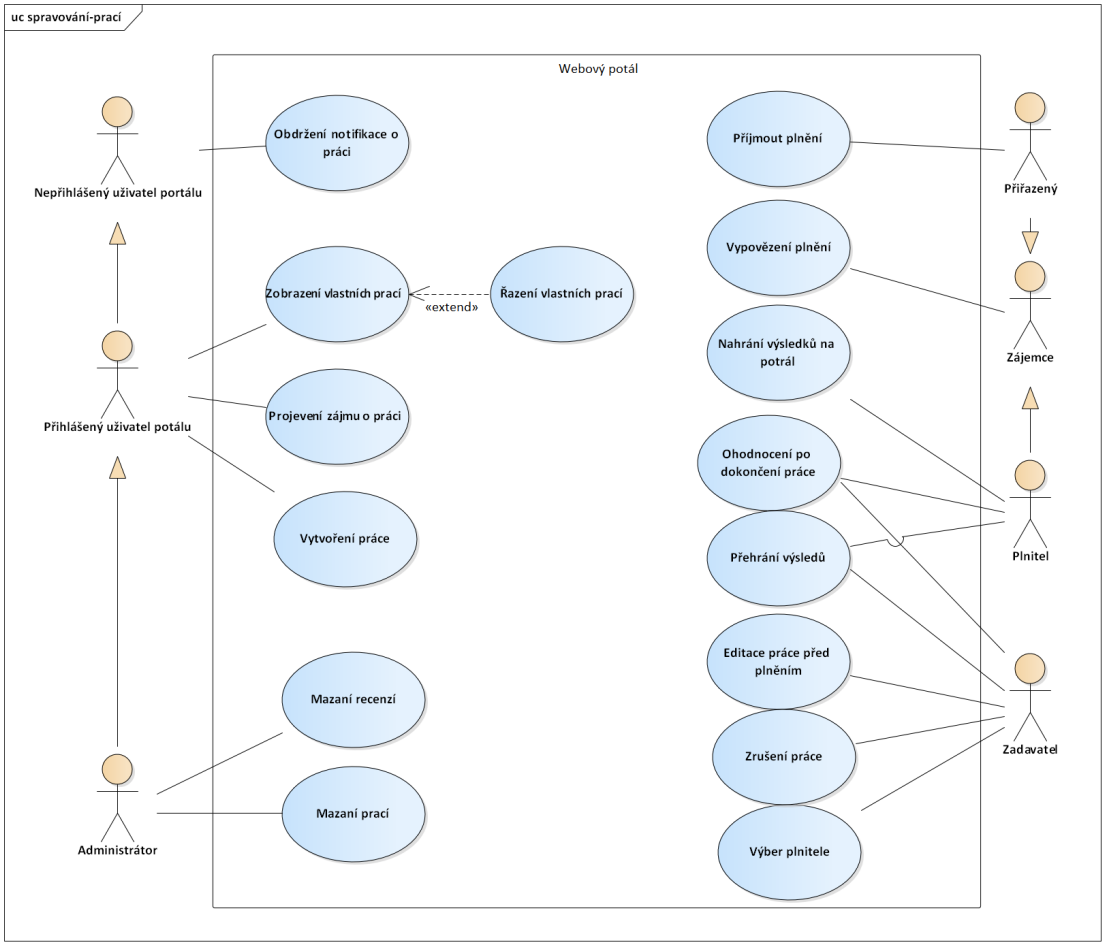
U06 – Blokování účtu a přidělování administrátorských práv uživatelům

Případ užití umožňuje administrátorovi blokovat účty uživatelů nebo jim dávat administrátorská práva. Administrátor si bude moct zobrazovat profily všech uživatel. V případě nevhodného chování uživatele bude moct administrátor zablokovat účet daného uživatele. Účet bude administrátor moct i odblokovat a sebrat mu administrátorská práva.

Aktéři: administrátor

3.1.3 Spravování prací

V této podkapitole jsou uvedený všechny interakce týkající se spravování prací. Na obrázku 3.2 je pak uveden model případu užití.



Obrázek 3.2 Model pro spravování prací

U07 – Zobrazení vlastních prací

Případ užití umožňuje přihlášenému uživateli zobrazit práce, na kterých se podílí. Práce budou rozděleny podle typu účasti v nich. Dělit je budu na dvě kategorie: zadané a vzaté uživatelem.

Do zadaných budou řazeny všechny práce, ve kterých je uživatel v roli zadavatele. Ve vzatých pak budou ty, kde je uživatel v roli plnitele. Protože práce mohou mít různé stavy, budou pro větší přehlednost v kategoriích rozděleny do skupin podle jejich stavu. Stav práce může být: volná, přiřazená, probíhající a ukončená. Ve skupinách si bude uživatel moct práce řadit podle ceny, data splatnosti a stáří.

Každá práce, na které se uživatel podílí, půjde otevřít pro zobrazení podrobnějších informací. Budou v ní vždy uvedeny detaily práce. Tyto detaily jsou všechny parametry zadávané při jejím vytváření, dále v jakém je stavu, kdo je plnitelem (pokud je přiřazen) a zadavatelem. Podle stavu a toho, v jaké roli je uživatel se však bude lišit obsah pod těmito detaily ten bude odpovídat možným interakci při roli uživatele a aktuálním stavu práce.

Aktéři: přihlášený uživatel portálu

U08 – Vytvoření práce

Případ užití umožňuje přihlášenému uživateli vytvoření práce. K jejímu vytvoření bude zapotřebí vyplnit formulář s následujícími parametry: název a popis práce, cena, kontaktní email a telefonní číslo pro komunikaci, adresa, město a kraj, kde se bude prohlídka konat, odkaz na inzerci, kategorie a očekávané datum, do kterého by měla být práce splněna. Všechny tyto údaje, až na popis práce a odkaz, jsou povinné a nepůjde bez nich práci vytvořit. Po jejím vytvoření přechází práce do stavu volná a uživatel, který jí vytvořil, se stává zadavatelem práce.

Aktéři: přihlášený uživatel portálu

U09 – Projevení zájmu o práci

Případ užití umožňuje přihlášenému uživateli projevit zájem o publikovanou práci ve stavu volná. K projevení zájmu bude stačit práci otevřít a přihlásit se k zájmu o její plnění pomocí tlačítka. Projeveným zájmem uživatel souhlasí s plněním a stává se zájemcem o danou práci.

Aktéři: přihlášený uživatel portálu

U10 – Notifikování o změně stavu práce

Případ užití umožňuje přihlášenému i nepřihlášenému uživateli portálu dostávat notifikace o změně stavu nebo jeho role v pracích, na kterých se podílí. Je v nich tedy v roli zadavatele, plnitele, přiřazeného uživatele nebo zájemce. Notifikace bude uživatel dostávat na emailovou adresu, kterou vyplnil při registraci.

Aktéři: přihlášený uživatel portálu, nepřihlášený uživatel portálu

U11 – Editace práce před začátkem plnění

Případ užití umožňuje zadavateli upravovat parametry již vytvořené práce ve stavu volná. Bude umožněna úprava všech parametrů práce zadaných při jejím vytváření.

Aktéři: zadavatel

U12 – Výběr plnitele práce

Případ užití umožňuje zadavateli práce, ve stavu volná, vybrat ze seznamu se všemi zájemci plnitele práce. U každého z těchto zájemců budou uvedeny základní informace, jako jeho aktuální skóre, počet recenzí, uživatelské jméno a specializace. Půjde si zobrazit jejich profil a ze seznamu si bude moct zadavatel vybrat jednoho plnitele. Po jeho výběru, zájemce přechází do role přiřazený a bude muset v práci potvrdit její plnění Práce přechází do stavu přiřazená. Po potvrzení se přiřazený uživatel stává plnitelem dané práce a ta přejde do stavu probíhající. V případě odmítnutí se práce přesouvá zpět do stavu volná.

Aktéři: zadavatel

U13 – Zrušení práce zadavatelem

Případ užití umožňuje zadavateli práce ve stavu volná zrušit práci. Taková práce bude kompletně smazána ze systému.

Aktéři: zadavatel

U14 – Vypovězení plnění práce

Případ užití umožňuje zájemci, přiřazenému uživateli a plniteli vypovědět plnění práce. Práce po vypovězení přejde znovu do stavu volná.

Aktéři: zájemce, plnitel, přiřazený uživatel

U15 – Mazaní prací administrátorem

Případ užití umožňuje administrátorovi mazat nevhodné práce, ať už budou v jakémkoliv stavu. Administrátor si tak bude moc zobrazit všechny práce a v případě, že by byly nevhodné je mazat. Při smazaní práce dojde k odstranění i všech recenzí a výsledků spojených s prací.

Aktéři: administrátor

U16 – Nahrání výsledků na portál

Případ užití umožňuje plniteli práce, ve stavu probíhající, po dokončení prohlídky a natočení videa nahrát výsledné video. Video se nahraje pomocí stisknutí tlačítka vybrat video, které umožní video vybrat a poté nahrání potvrdí stisknutím tlačítka nahrát. Pokud je plnitel spokojen s nahranými výsledky potvrdí odeslání zadavateli, tím přejde práce do stavu ukončená. Pokud by nahrál špatné video bude moct před odesláním zadavateli plnitel video přemazat nahráním nového.

Aktéři: plnitel

U17 – Přehrání výsledků

Případ užití umožňuje zadavateli i plniteli nahrané video přehrát přímo v ukončené práci. Nebude, ale umožněno stažení videa.

Aktéři: zadavatel

U18 – Vzájemné ohodnocení po dokončení práce

Případ užití umožňuje na konci každé dokončené práce se vzájemně ohodnotit plniteli a zadavateli. Při ohodnocení bude potřeba vyplnit hodnocení na škále od 1 do 5 a samotnou recenzi. Hodnotící škála by měla odrážet celkovou spokojenost se spoluprací.

Aktéři: zadavatel, plnitel

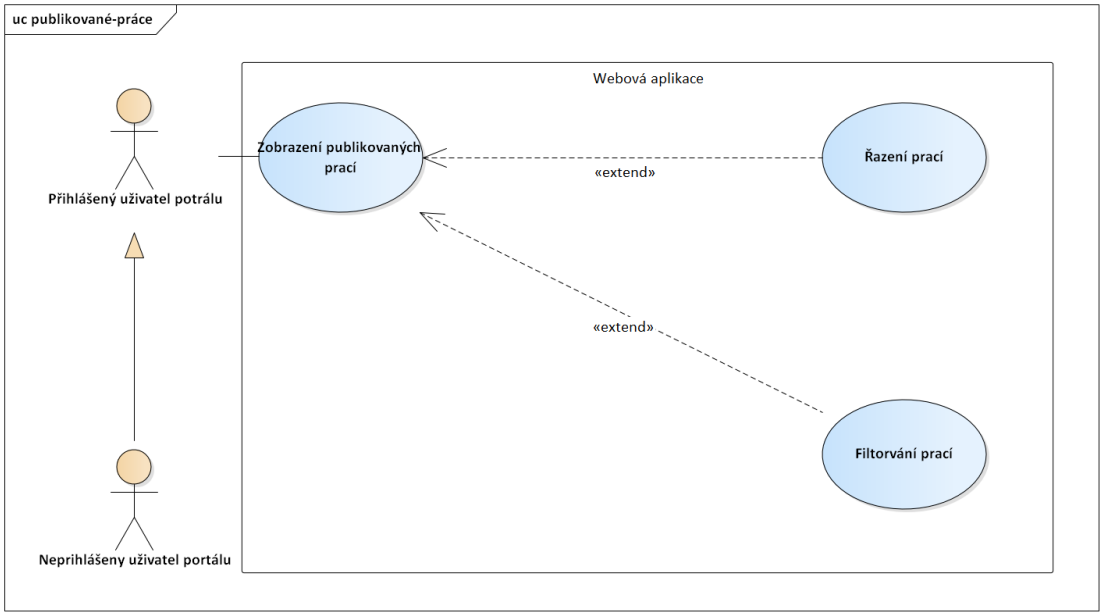
U19 – Mazaní recenzí administrátorem

Případ užití umožňuje administrátorovi mazat nevhodné recenze. Administrátor si tak bude moct zobrazit všechny recenze a v případě, že by byly nevhodné je mazat.

Aktéři: administrátor

3.1.4 Publikované práce

V této podkapitole jsou uvedený všechny interakce týkající se publikovaných prací. Na obrázku 3.3 je pak uveden model případu užití.



Obrázek 3.3 Model pro publikované práce

U20 – Zobrazení publikovaných prací

Případ užití umožňuje přihlášenému i nepřihlášenému uživateli zobrazit všechny publikované práce. Tedy práce, jenž je ve stavu volná. Každá z nich půjde otevřít a zobrazit si tak její detaily, včetně toho, kdo jí zadal. Bude umožněno zobrazení profilu zadavatele. Pod uvedenými detaily práce bude moct přihlášený uživatel projevit zájem o plnění práce.

Aktéři: přihlášený uživatel portálu, nepřihlášený uživatel portálu

U21 – Filtrování prací

Případ užití umožňuje přihlášenému i nepřihlášenému provádět různé typy filtrace nad publikovanými pracemi. Filtrace bude probíhat podle: ceny, data splnění, kraje a kategorie objednávky. Filtry půjdou kombinovat.

Aktéři: přihlášený uživatel portálu, nepřihlášený uživatel portálu

U22 – Řazení prací

Případ užití umožňuje přihlášenému i nepřihlášenému řadit publikované práce podle: ceny, data splnění, data vytvoření a jejich stáří. Řadit ale bude umožněno podle jednoho parametru.

Aktéři: přihlášený uživatel portálu, nepřihlášený uživatel portálu

3.2 Tabulka pokrytí

V následující části uvedu tabulku pokrytí, pomocí které se ujistím, že jsou všechny funkční požadavky nezbytné a jsou zohledněny v případech užití. Pokud nebude některý z funkčních požadavků pokrývat případ užití, tak se na požadavek buď zapomnělo nebo je zbytečný.

Tabulka 3.1 Tabulka pokrytí

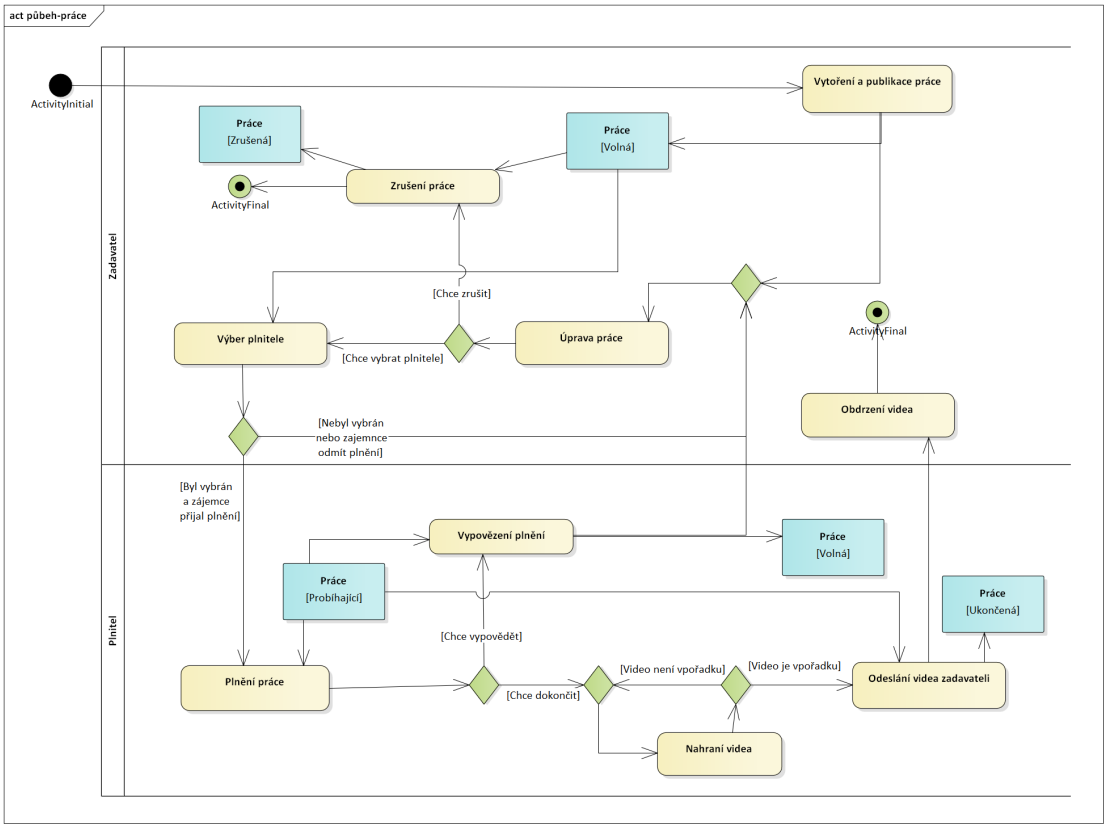
X	F01	F02	F03	F04	F05	F06	F07	F08	F09	F10	F11
U01	✓										
U02	✓										
U03			✓								
U04										✓	
U05		✓									
U06											✓
U07				✓							
U08				✓							
U09					✓						
U10						✓					
U11				✓							
U12				✓							
U13				✓							
U14					✓						
U15											✓
U16							✓				
U17							✓				
U18								✓			
U19											✓
U20									✓		
U21									✓		
U22									✓		

3.3 Diagram aktivit

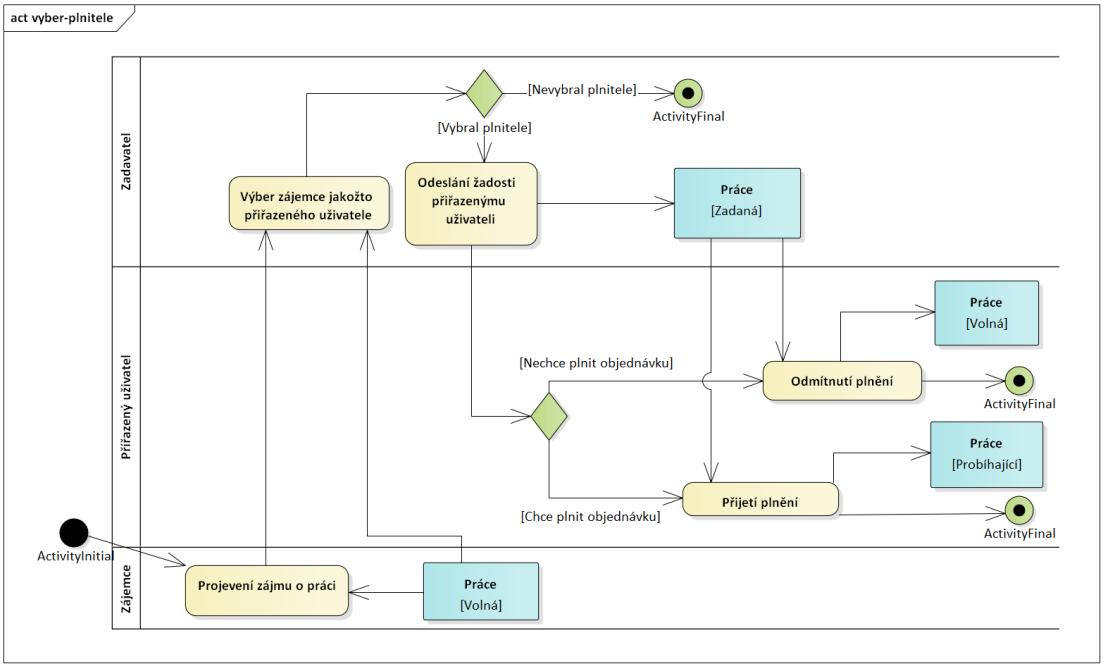
Diagram aktivit má za úkol zachytit vykonávanou činnost. Slouží především pro vizualizaci složitějších procesů. Rozhodl jsem se udělat diagram aktivit pro průběh práce. K jeho tvorbě využiji zejména případy užití uvedené v 3.1. Důvodem je složitost průběhu práce, změny jejího stavu a množství akcí v určitých fázích práce pro plnitele i zadavatele. Diagram budu vytvářet pomocí nástroje EA se kterým jsem byl seznámen v době studia.

Na obrázku 3.4 naleznete diagram aktivit k průběhu práce. Hlavně je zde zachycena interakce mezi zadavatelem a plnitelem objednávky. Práce začíná jejím vytvořením zadavatelem, který jí muže následně zrušit nebo upravit. Pokud bude zadavatel mít nějaké zájemce o práci pak v aktivitě Výběr plnitele vybere uživatele, který by měl plnit práci. Uživatel buď potvrdí plnění nebo jej odmítne. Při přijetí pokračuje práce dál k plnění a mění svůj stav na probíhající. Uživatel, který plnění potvrdil získává roli plnitele. Pokud odmítne bude moct zadavatel práci upravit a vybrat jiného uživatele ke splnění práce nebo ji zrušit. Při plnění se uživatel zúčastní osobní prohlídky, kterou natočí a nahraje na portál. Video bude moct přemazat jeho opětovným nahráním. Pokud je s ním spokojen odešle video plniteli a tím se práce považuje za ukončenou a nabývá stavu ukončená. V případě, že by si to plnitel rozmyslel, tak v průběhu práce může plnění vypovědět. Tím se zase práce vrátí zpět do stavu volná a zadavatel může vybrat jiného plnitele.

Zmiňoval jsem se o aktivitě Výběr plnitele vzhledem k tomu, že se i v ní mění stav práce a role uživatele, rozhodl jsem se vytvořit diagram i pro tuto aktivitu uvedený na obrázku 3.5. Vše začíná tím, že uživatel portálu projeví zájem o plnění práce a stane se tak zájemcem. Zadavatel pak tohoto zájemce vybere a odešle mu žádost o plnění práce. Zájemce se v tu chvíli stane přiřazeným uživatelem práce a práce změní svůj stav z volné na zadanou. Přiřazený uživatel pak plnění buď příjme, stane se z něho plnitel a práce změní svůj stav na probíhající nebo odmítne. V tom případě bude práce uvedena znovu do stavu volná a zadavatel musí vybrat jiného zájemce.



■ **Obrázek 3.4** Diagram aktivit pro průběh práce



■ **Obrázek 3.5** Diagram aktivit pro výběr plnitele

3.4 Architektura implementace

Implementaci jsem se rozhodl rozdělit na dvě hlavní části, a to na frontend a backend. Toto rozdělení se řídí server-client architekturou. Server-client je síťová architektura, která odděluje klienta (často aplikaci s grafickým uživatelským rozhraním) a server, kteří spolu komunikují přes počítačovou síť podle [6]. Toto rozdělení bylo vybráno na základě potřeb a podstaty aplikace.

3.5 Backend

V dané části popíšu, jak budu postupovat při implementaci Backendu. Backend je část implementace programu, která odpovídá za chod aplikace na pozadí. Budu se zde věnovat hlavně volbě technologií a nástrojů použitých pro jeho vytvoření.

Cílem je navrhnout vhodné řešení pro vytvoření backendu, se kterým by komunikoval frontend. Zvolené řešení by mělo naplňovat cílové požadavky uživatel, definované v 2.3 a umožnit provedení případů užití uvedených v 3.1. Při jeho návrhu budu vycházet z dříve zmíněných požadavků a případů užití.

3.5.1 Volba technologií

Ve volbě technologií bych se chtěl zaměřit na volbu programovacího jazyka, ve kterém budu backend vyvíjet. Při jeho volbě jsem zvažoval jak vhodnost použití, tak míru zkušeností, kterou mám. Na základě předchozího návrhu a povahy projektu jsem identifikoval body, které je potřeba aby jazyk splňoval:

- snadná rozšiřitelnost a škálovatelnost
- jednoduchost pro tvorbu prototypu
- jednoduchá a rychlá implementace API
- dobrá udržitelnost
- dostatek nástrojů pro implementaci

Na základě těchto bodů jsem stanovil programovací jazyky, které je splňují. Jazyky jsou uvedeny v tabulce 3.2, kde ke každému je uvedena moje zkušenost s daným jazykem. Zkušenost budu hodnotit na škále 1-5. Nejvyšší hodnocení je tedy 5 a značí velkou míru zkušenosti s jazykem. Toto hodnocení bude odrážet především počet hodin, které jsem s daným jazykem strávil. Dalším hodnocením je popularita jazyka. Údaje o ní budu čerpat ze statistik uvedených na stránkách stackoverflow.com pro rok 2021 [7]. Na jejich základě uvedu v tabulce pozici, na které se jazyk umístil v kategorii popularita.

Tabulka 3.2 Nalezené programovací jazyky

Programovaní jazyk	Moje zkušenost s nim	Umístění v žebříčku popularity
Java / Kotlin	2	5.
Node.js	5	6.
Python	2	3.
PHP	1	11.

Jak vyplývá z tabulky největší zkušenost mám s Node.js, který si nevede špatně ani v žebříčku popularity. Myslím si, že bude skvělou volbou jako programovací jazyk pro vývoj backendu. Node.js je škálovatelný a skvělý pro rychlou tvorbu prototypu. V kombinaci s TypeScriptem

bude navíc i dostatečně udržitelný. Zároveň je pro Node.js dostupné velké množství nástrojů pomocí npm nebo yarn. Není však bez chyb a například není vhodné jej používat pro vývoj aplikací s vysokou výpočetní náročností. Především kvůli tomu, že se jedná o single-threaded jazyk. Tedy jazyk využívající jen jedno vlákno. Kvůli tomu neumožňuje paralelní programování v dostatečné míře, které může složité výpočty značně urychlit. Ve vyvíjené aplikaci se žádné složité výpočty provádět nebudou, proto mi dané omezení jazyka nevadí. Pro využití v tomto projektu je tedy Node.js více než dostatečný. Předtím něž se posunu k další části návrhu, tak bych se chtěl krátce zmínit o samotném Node.js a jeho vztahu s JavaScriptem a TypeScriptem. Také bych se chtěl zmínit i o samotném TypeScriptu.

Node.js

Node.js je programovací jazyk, který umožňuje použití JavaScriptu pro tvorbu aplikací běžících na serveru. Je používaný zejména pro tvorbu dat intenzivních real-time aplikací podle [8]. Při psaní aplikace v Node.js tedy vlastně budu psát aplikaci v JavaScriptu. JavaScript je sám o sobě dost neudržitelný, protože nenabízí dostatek nástrojů pro typování. Proto jej použiji v kombinaci s TypeScriptem, který do JavaScriptu přidává nástroje pro typovaní a mnoho dalších nástrojů zvyšujících udržitelnost.

TypeScript

TypeScript byl vytvořen firmou Microsoft již v roce 2012 a je vydáván jako open source. Jedná se o nadstavbu jazyka JavaScript, který ho rozšiřuje o statické typování, třídy, rozhraní a další věci známé z OOP podle [9].

3.5.2 API

Nyní bych přešel k tomu, jakým způsobem budu v Node.js backend psát. Je potřeba zejména zvolit jaký typ API budu implementovat. API (Application Programming Interface) je způsob, jakým se bude dát s backendem komunikovat. V implementovaném prototypu bude s backendem komunikovat především frontend. Tento způsob komunikace je psán podle typu zvolené architektury. Je několik populárních možností:

- REST API
- SOAP API
- GraphQL

Já jsem zvolil REST API, zejména kvůli jeho rozšířenosti a tomu, že s ním již mám zkušenost. Je ale potřeba zmínit, že i přes to, že budu psát REST API, tak nebudu dogmaticky dodržovat všechny jeho aspekty. Například formát koncových bodů (dále endpointů) aplikace není podle RESTu zcela korektní. Uvedu nyní stručně v bodech, co představuje REST API:

- Pro přístup k datům použijeme endpointy. Každý přístup má pak právě jeden endpoint. Např. GET /users - vrátí seznam uživatel, GET /jobs - vrátí seznam prací atd.
- Pro volání těchto bodu použijeme HTTP metody jako POST, PUT, GET, DELETE a PATCH podle jejich významu.
- Zpět kromě dat vracíme i HTTP status kód podle významu.
- Komunikace s klientem je bez stavová.

O správném psaní REST API by se toho jistě dalo napsat mnoho. Tohle je pouze pár základních bodů a podrobnější zkoumání REST API bohužel není předmětem této bakalářské práce.

Podle zvoleného typu API jsem následně navrhl endpointy. Vzhledem k jejich množství jsem uvedl jenom pár pro ukázku v tabulce 3.3 , kde je vždy uveden endpoint, metoda HTTP a stručný popis. Ostatní endpointy jsou k nalezení v příloze. Po návrhu endpointů jsem následně zdokumentoval celé API pomocí Swaggeru.

Tabulka 3.3 Ukázka endpointů

metoda HTTP	endpoint	popis
POST	/users	vytvoří nového uživatele
GET	/users	vrátí seznam uživatelů
POST	/job	vytvoří práci
POST	/job/:id	upraví práci s identifikačním číslem :id
GET	/job/:id	vrátí práci s identifikačním číslem :id
DELETE	/job/:id	odstraní práci s identifikačním číslem :id

Jak jsem již dříve zmínil budu implementovat backend v Node.js. Pro implementaci REST API jsem se rozhodl využít frameworku, který ji značně usnadní. Zvolil jsem framework Express, protože se mi líbila jeho provázanost s TypeScriptem, velká rozšířenost a jednoduchost v osvojení. Express umožňuje vytvářet REST API (nejenom) a to jednoduše a rychle.

Ke zdokumentování API jsem využil Swagger. V následující části bych chtěl tento nástroj krátce popsat.

Swagger

Swagger je dokumentační nástroj, který umožňuje zachytit rozhraní API. Používá se zejména k dokumentaci Open API a REST API. Dokumentace muže být psána pomoci YAML a JSON formátu. Jeho hlavní výhodou je, že ze sepsané dokumentace lze generovat řadu věcí. Při implementaci prototypu jej využiji ke generaci klienta a interaktivní dokumentace API, která vizualizuje a umožní interakci s endpointy. Klienta budu generovat pomocí nástroje swagger-typescript-api. Tento nástroj umožňuje generovat TypeScript klienta ze swagger dokumentace. Volil jsem tento nástroj, protože mi přišlo, že kód, který generoval, byl srozumitelný a čitelný. Na rozdíl od jiných nástrojů, které jsem zkoušel. Navíc poskytoval řadu možností, jakým způsobem bude kód generován.

3.6 Bezpečnost

V implementaci je potřeba, kromě samotných funkcionalit, řešit i bezpečnost. Nebudu se snažit dosáhnout maximální bezpečnosti aplikace ani vymýšlet si vlastní zabezpečení. Pokusím ale o zabezpečení aspektů, které považuji za důležité, pomocí známých a osvědčených metod. Identifikovanými aspekty jsou:

- uchovávání hesla
- komunikace frontendu a backendu
- autentizace a autorizace uživatel

V následujících částech bych se chtěl věnovat jednotlivým bodům. V každé části stručně uvedu problematiku, proč by se měla řešit a navrhnu vhodné řešení.

3.6.1 Uchovávání hesla

Heslo bych jistě neměl v databázi uchovávat v plaintextu (nepozměněný text). Při takovém to uchování by totiž při úniku dat došlo k narušení bezpečnosti celé aplikace. Hesla by mohl pachatel využit k přístupu do aplikace, což je obzvlášť nebezpečné bude-li mít přístup k administrátorským profilům. Proto je potřeba hesla, před jejich uložením, šifrovat, aby je pachatel nemohl použít při jejich úniku. Pro volbu způsobu šifrování jsem použil znalosti z předmětu BI-BEZ.

K šifrování využiji jednosměrné šifry SHA256. Jednosměrnost znamená, že heslo půjde jednoduše zašifrovat, ale jeho dešifrace je výpočetně náročná. Přitom zašifrováním jednoho a toho samého hesla vznikají stejné šifry nebo taky hashe. Proto pokud bude potřeba heslo porovnat (například k validaci při přihlášení), tak zašifruji porovnávaný řetězec a porovnám hashe. Pokud se shodují, je porovnávaný řetězec stejný jako heslo.

3.6.2 Komunikace frontendu a backendu

Při komunikaci backendu a frontendu může docházet k jejímu odposlechu. V jeho důsledku by pak mohly uniknout citlivé údaje jako je například heslo uživatele, který se pokouší přihlásit. Je tedy zapotřebí zabezpečit přenos dat a zabránit odposlechu. Toho dosáhnu použitím protokolu HTTPS (Hypertext Transfer Protocol Secure). HTTPS zajišťuje autentizaci, důvěrnost přenášených dat a jejich integritu podle [10].

3.6.3 Autentizace a autorizace uživatel

Při komunikaci frontendu a backendu je potřeba autorizovat a autentizovat uživatele. Tedy zjišťovat, zda má uživatel právo provádět operace (autorizace) a o koho, se při provádění operace jedná (autentizace). To je potřeba například pokud bude chtít uživatel změnit údaje svého profilu nebo provést jakoukoliv operaci s prací. Toho lze dosáhnout mnoha způsoby jako například:

- udržování session
- pomocí JWT
- pomocí cookies

Každý z těchto způsobu je validní a má svoje pro a proti. Já budu při implementaci volit JWT, protože s danou technologií mám největší zkušenost a líbí se mi jeho jednoduchost. Je to ale jen moje preference a stejně korektní by bylo zvolit si jakýkoliv ze zmíněných způsobu. Nyní bych chtěl krátce popsat JWT v následující části.

JWT

JWT (JSON Web Tokens) je způsob ověřování komunikace pomocí JWT tokenů. Princip, který používá je jednoduchý a je popsán v následujících bodech:

1. Na začátku interakce klienta s backendem se pošle dotaz pro získaní autorizačního tokenu. Ta může být například v podobě přihlášení.
2. Backend token vygeneruje a vrátí klientovi.
3. Při dalším dotazovaní backendu posílá klient autorizační token v hlavičce dotazu. Token se při dotazu (requestu) na backend validuje. Pokud je validace v pořádku provede se požadovaná operace.

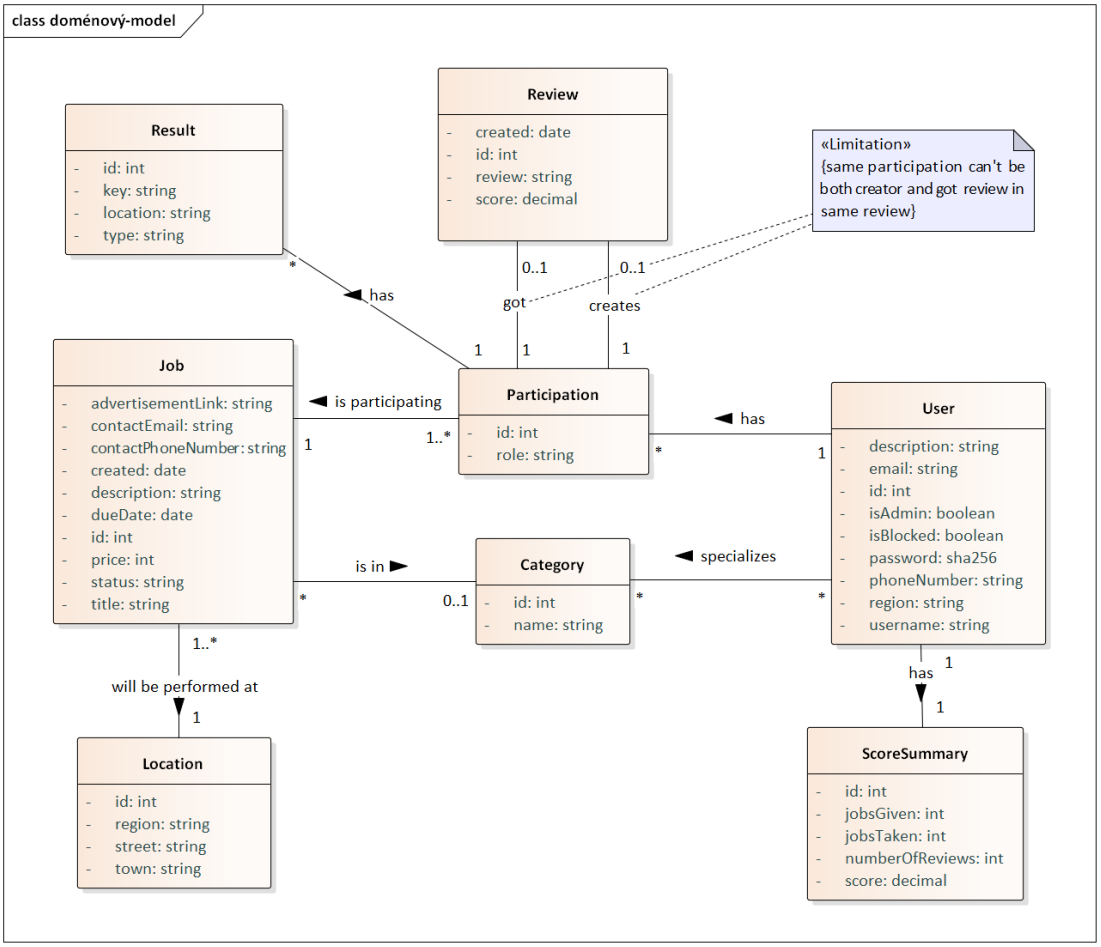
Je důležité zmínit, že tyto tokeny mají dobu platnosti, která se nastavuje. Kromě autorizačního tokenu, tak do hry vstupuje i refresh token, který je poslán na začátku interakce spolu s autorizačním tokenem. Pokud skončí platnost autorizačního tokenu, je potřeba získat token nový. Ten získáme posláním refresh tokenu, který má značně vyšší expirační dobu. Refresh tokeny se typicky pro účely zneplatnění ukládají na straně backendu a po odhlášení jsou odstraněny.

Při implementaci v Node.js lze využit nástroje jsonwebtoken, který umožňuje tokeny vytvářet a validovat. Pro ukládaní refresh tokenu na backendu by se mohl použít Redis což je velice rychlá in-memory databáze.

3.7 Databáze

V této části popíšu návrh databáze. Pod jejím návrhem se rozumí doménový model a volba typu databáze, který bude při implementaci použit. Další jeho důležitou součástí je i práce s databází v backendu, který by s ní měl komunikovat a modifikovat v ní data.

3.7.1 Doménový model



Obrázek 3.6 Doménový model

Na základě údajů uvedených v diagramech aktivit, funkčních požadavcích a případech užití jsem vytvořil doménový model uvedený na obrázku 3.6. Model by měl zachycovat vztahy jednotlivých objektů. Tento model využiji zejména k tvorbě databáze. Modelování probíhalo pomocí EA, stejně jako diagram aktivit a případy užití v předchozích částech.

3.7.2 Volba databáze

Jakožto typ databáze volím PostgreSQL. Volbu jsem prováděl na základě potřeb implementovaného prototypu. Důvody této volby jsou následující:

- open source projekt
- velká komunita a rozšířenost
- objektovo-relační databáze

Pro práci s databází na backendu využiji ORM přístup, se kterým mám zkušenost. Objektově relační zobrazení (ORM) je programovací technika v softwarovém inženýrství, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem podle [11]. ORM jsem volil, protože šetří čas při vývoji a umožňuje jednoduchou migraci při změně typu databáze. Pro Node.js existuje řada knihoven, které se danou problematikou zabývají, můj výběr se zastavil u knihovny TypeORM. Vybral jsem jí, protože má skvělou podporu, dokumentaci a do značné míry se podobá Hibernate knihovně v Javě, se kterou mám zkušenosti.

3.8 Ukládání videa

V aplikaci je potřeba vzhledem k funkčním požadavkům a podstatě aplikace řešit i ukládání souboru, konkrétně videí. Ukládaní souborů lze řešit například následujícími způsoby:

- lokální úložiště
- cloudové úložiště
- ukládání do databáze

Vzhledem k tomu, že bude zapotřebí ukládat videa, která můžou zabírat i stovky megabitů, tak je nutné uvažovat řešení, které nebude mít problém s takovým objemem dat. Ukládání do databáze je proto zcela jistě nevhodné. Další možností je využít lokální úložiště. Toto řešení je validní z hlediska vývoje, ale pro produkční verze už nikoliv, takže stejně budu muset řešit jiný způsob ukládání. Tím se dostávám k mé volbě, a tím je cloudové úložiště, kde se data ukládají na server třetí strany.

Zvolil jsem cloudové úložiště od společnosti Amazon. Řešení se jmenuje AWS S3 a je na rok zcela zdarma, poté je zpoplatněno. Vybral jsem ho, kvůli tomu, že je zdarma a pro manipulaci s ním existuje pro Node.js řada řešení. Při implementaci prototypu, tak můžu použít například knihovnu multer-s3 pro nahrávaní a awk-sdk pro odstraňování a získávání dat.

U videí je kromě jejich ukládání potřeba řešit i formáty, které budou dovoleny. Jelikož nemám zkušenosti s video formáty, tak jsem se rozhodl vyhledat nejpoužívanější a ty podporovat. Nalezl jsem seznam populárních formátů s jejich popisem od Adobe [12] a vybral 5, které podle popisu odpovídají potřebám vyvíjeného prototypu: MP4, MOV, AVI, MKV, WMV.

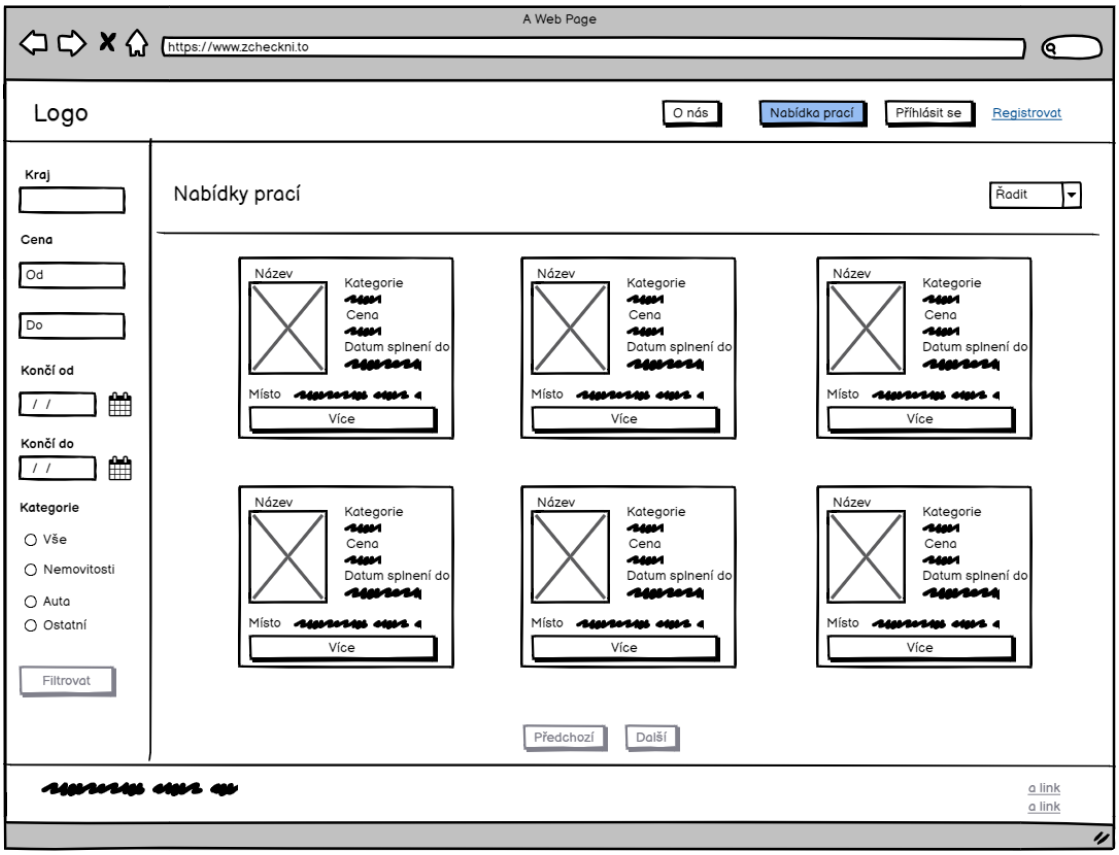
3.9 Frontend

V dané častí nahrnu, jakým způsobem budu implementovat frontend. Navrhnu zde jeho vizuální návrh v podkapitole 3.9.1. Po návrhu vizuální části vyberu technologie, pomocí kterých budu prototyp implementovat. Hlavním cílem frontendu je vývoj grafického rozhraní webové aplikace, ke kterému bude přistupovat uživatel.

3.9.1 Wireframe

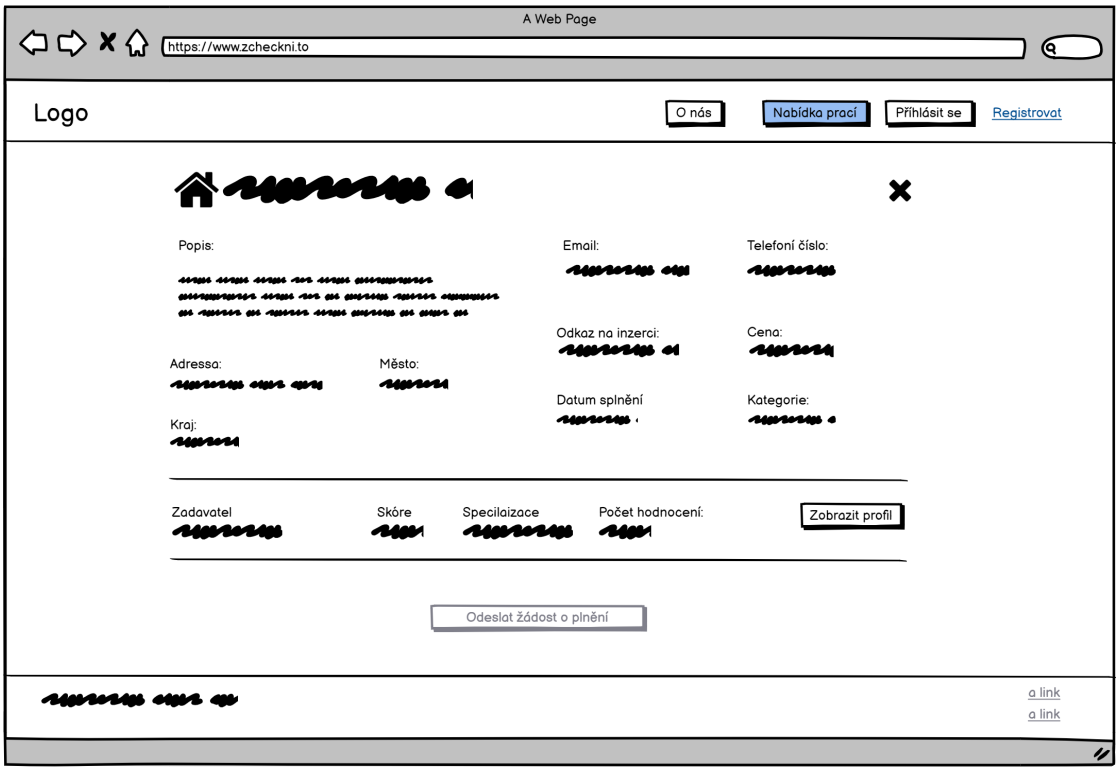
V této části uvedu postup tvorby wireframu a následně popíšu úvodní obrazovku s publikovanými pracemi a obrazovku s prací samotnou. Soubor s celým návrhem a ukázku dalších wireframu naleznete v příloze.

Wireframe slouží k vytvoření si představy o podobě stránek výsledné implementace. Dává informaci o rozložení prvků na stránkách a zjednodušuje tak postup při jejich implementaci. Navrhovat jej budu pomocí nástroje Balsamiq, který umožňuje rychlý a jednoduchý návrh a jeho následný export do PDF formátu. Vytvořený návrh, pomocí tohoto nástroje, je interaktivní a je možno jím proklikávat jednotlivé stránky pomocí tlačítek v exportovaném PDF. Nástroj volím, protože s ním mám dobrou zkušenost a navrhoval jsem pomocí něj již pro 4 projekty. Budu navrhovat pouze obrazovky pro počítačovou verzi. Při implementaci pak z návrhu pro počítače vyvodím i verzi pro mobily a tablety. Tento postup se mi osvědčil u řady projektů. Při návrhu využiji zejména svých zkušeností z předchozích projektů. Jednou z nich je například umisťovat prvky stránky na obvyklá místa, tedy tam kde by je uživatel očekával.



Obrázek 3.7 Wireframe - nabídky prací

Vzhledem k množstvím navržených obrazovek zde uvedu 2 hlavní. Pro ukázku zde popíši obrazovku s nabídkami prací, která je uvedena na obrázku 3.7 a samotnou práci. Navigační prvky stránky jsem umístil nahoře, kde vlevo na kraji je logo a vpravo tlačítka s přesměrováním na přihlášení, registraci a stránku, kde se nepřihlášený uživatel může dozvědět více o produktu. Po levé straně pak uživatel může nalézt i lištu s filtry a nahoře, nad nabízenými pracemi, je umístěno tlačítko pro řazení prací. Dole pod pracemi je pak přepínání další a předchozí stránky. Samotné nabízené práce pak prezentuji jako čtverce se základními informacemi o práci. Po kliknutí na práci se uživateli zobrazí samotná stránka s více informacemi o ní.



Obrázek 3.8 Wireframe - práce

Na obrázku 3.8 je uveden návrh zobrazení nabídky práce po jejím otevření. Pod lištou s menu aplikace je umístěna ikonka, která odpovídá kategorii práce a název práce, která je zobrazena. Pod názvem jsou informace o vytvořené práci, které byly zadány při jejím vytvoření. Následují údaje profilu uživatele, který práci vytvořil, s tlačítkem pro zobrazení celého profilu. Posledním v pořadí je tlačítko pro odeslání žádosti o plnění práce, to ale půjde použít pouze po přihlášení do portálu. Práce v ostatních stavech, jako je například ukončená, jsou pak dost podobné. Vždy jsou uvedeny detaily práce pod nimiž jsou tlačítka s akcemi, které může uživatel v daném stavu provést, případně nahrané video.

3.9.2 Volba technologií

Pro implementaci uživatelského rozhraní existuje spousta možností. Já jsem zvolil, že budu psát frontend v knihovně React s TypeScriptem a to z následujících důvodů:

- mám s ním hodně zkušeností
- velká komunita, aktuálnost a rozšířenost

- frontend bude psán ve stejném jazyce jako backend
- množství nástrojů a knihoven psaných pro React
- dobrá kompatibilita s TypeScriptem

Mohl jsem sice volit i jiné podobné nástroje jako je Vue.js nebo Next.js, s nimi ale zatím nemám žádnou zkušenost, a proto jsem radši sáhl po Reactu, se kterým jsem strávil více než 600 hodin. Předtím než se podívám na stylování v Reactu, tak bych se o něm chtěl krátce zmínit.

React

React je nástroj vytvořený společností Facebook a slouží k tvorbě SPA (Single-Paged Application). SPA ve zkratce znamená, že se celé uživatelské rozhraní rendruje (vykresluje) na klientské straně s pomocí JavaScriptu.

Základní stavební kámen zde tvoří tzv. komponenty (components), což jsou různé znovupoužitelné HTML elementy se zapouzdřenou funkcionalitou, jejichž skládáním vzniká komplexní UI (User Interface) aplikace. Tyto komponenty pak mají své vlastnosti (props) a spravují svůj vnitřní stav (state) dle [13]. Spravování stavu komponenty je pak řešen pomocí hooks.

Problém Reactu je především v jeho otevřenosti. Neboli existuje množství způsobů, jak lze pomocí něj implementovat. Proto vyžaduje velkou míru zkušeností, aby byl projekt naimplementován dobře.

3.9.3 Stylování

Know-how každé webové aplikace je její design (styl) a na ten se zaměřím v této části. Existuje řada způsobu, jak stylovat projekty v Reactu, ať už staré dobré CSS, SASS nebo komponentové knihovny. Právě třetí způsob jsem zvolil pro implementaci prototypu, a to hlavně proto, že daný způsob šetří čas a umožňuje se lépe soustředit na důležitější aspekty při implementaci jako je například funkcionalita.

Jako takovou knihovnu jsem zvolil Material UI, která je inspirována Material designem od společnosti Google. Tato knihovna poskytuje velké množství nastylovaných komponent, které pokrývají veškeré potřeby při implementaci plynoucí z vizuálního návrhu. Je také do značné míry konfigurovatelná a flexibilní, což umožní navrhnout vlastní design stránek odlišitelný od konkurence.

3.9.4 Komunikace s backendem

Frontend bude komunikovat s backendem za účelem provádění operací: získávání, modifikování a ukládání dat. Frontend komunikuje s backendem pomocí klienta, který komunikaci zprostředkovává.

V Reactu jsou dvě nejpopulárnější možnosti komunikace pomocí Fetch a Axios klientů. V případě implementace prototypu budu volit Axios a to kvůli jeho jednoduchosti v používání a automatické transformaci odpovědí do JSONu formátu. S Axios mám také hodně zkušeností, což o Fetch říci nemůžu. Klienta v Axiosu budu kompletně generovat pomocí swaggeru a swagger-typescript-api.

Při komunikaci je často také potřeba ukládat globální stav, pro předání dat ostatním komponentám. K tomu se dá v Reactu použít Redux nebo Recoil knihovna. Já preferuji Recoil, hlavně kvůli jeho jednoduchosti. Redux mi přijde komplikovaný kvůli jeho boilerplatu.

3.10 Distribuce prototypu

Prototyp bych chtěl umožnit spustit co nejednodušším způsobem. To usnadní především vývoj v týmu v budoucnu. Cílem tedy bude co nejvíc usnadnit instalaci. Toho dosáhnu pomocí Dockeru, který umožní instalovat a spustit projekt pomocí jednoho příkazu.

Docker umožňuje tvorbu kontejnerů, ve kterých bude běžet výsledná implementace. Tedy backend, frontend, a databáze v případě prototypu. Protože je implementace rozdělená takto do více částí rozhodl jsem se využit Docker Compose, který usnadní jejich orchestraci a umožní tak jednoduché spuštění skrze jeden příkaz.

3.11 Shrnutí

Cílem této kapitoly bylo vytvořit návrh vhodného řešení pro následující implementaci. Toho jsem úspěšně dosáhl a navrhl jsem fungování prototypu a také i jeho technickou stránku, kterou se budu řídit při implementaci. V návrhu jsem zohlednil všechny požadavky definované v kapitole 2. I přesto bych rád opět zdůraznil, že funkční požadavky s nízkou prioritou budu implementovat až v další fázi vývoje aplikace. Tedy po implementaci prototypu.

V navazující kapitole Implementace prototypu se podívám na implementaci prototypu pomocí navrženého řešení.

Kapitola 4
Implementace prototypu

V této kapitole uvedu postup a detaily implementace prototypu. Při tvorbě implementace využiji zejména kapitoly 3, kde je popsáno, co vše by měla implementace prototypu zahrnovat a jaké technologie budou použity. Protože je výsledná implementace vcelku rozsáhlá a má dohromady přes 15 000 řádků, tak se zaměřím především na způsob použití zvolených technologií v implementaci. Přičemž se zaměřím na technologie, které jsou v implementaci nejdůležitější.

Kapitolu jsem rozdělil do 4 částí, kde v první části popíšu postup a v druhé strukturu a formátovaní, kterou následuje popis implementace backend a frontend části.

4.1 Postup implementace

Implementaci jsem se rozhodl rozdělit do 2 separátních částí, a to na backend a frontend. Jako první jsem se rozhodl implementovat backend a poté na něj navázat implementací frontendu.

Vzhledem k tomu že jsem implementoval projekt sám, zvolil jsem postup, při kterém budu implementovat projekt po částech, a to iterativním způsobem. Implementaci jsem tedy rozdělil podle funkcionalit do iterací, které budu plnit. Tento přístup mi přišel vhodný zejména, protože jsem měl přesně stanovené požadavky, které chci splnit a věděl jsem, že se v průběhu nebude nic měnit. Také mi to umožnilo dobře naplánovat postup a odhadnout časovou náročnost implementace. Pro verzování projektu jsem pak volil nástroj Gitlab.

4.2 Struktura a formátování

V této části popíšu strukturu implementace a nástroje, které jsem použil ke zkvalitnění kódu. Pro dodržování best practises jsem využil Eslintu, který po správné konfiguraci upozorní na jejich nedodržení a vynutí si opravu. Eslint jsem používal jak pro backend, tak pro frontend.

Pro formátování kódu jsem využil nástroje Prettier. Formátovaní společně s Eslintem tak dává hezký a čitelný kód, který dodržuje best practises.

Pro strukturování backend části jsem se rozhodl využít architektury MVC (Model-view-controller). Princip MVC je rozdělení backendu do 3 částí:

Model: Je datová reprezentace informací. V případě implementace je uložená ve složce entities, kde jsou namapovány jednotlivé tabulky pomocí TypeORM.

View: Je část která odpovídá za interakci s uživatelem. V případě backendu je interakcí volání endpointů. Ty jsou uloženy v adresáři routes.

Controller: Pak reaguje na změny v modelu a interakce uživatele. Controller jsem rozdělil na servisy, které odpovídají za logiku a controllery, které volají potřebné servisy.

Pro strukturaci frontendu jsem využil svých předchozích zkušeností a rozdělím ho na klienta pro volání endpointů, stránky, které se budou zobrazovat a komponenty.

4.3 Backend

Hlavním cílem backendu je vytvořit REST API, pomocí kterého bude frontend s backendem komunikovat. Kromě komunikace bude ale potřeba řešit i jeho bezpečnost, ukládání videí a komunikaci s databází.

V této části popíšu, jak jsem využil jednotlivé nástroje k dosažení tohoto cíle. Vybrány byly pouze nejdůležitější nástroje. V každé z následujících částí vždy nástroj krátce popíšu a uvedu ukázku kódu, ve které jsem ho použil. Ukázku také krátce popíšu.

4.3.1 Express a implementace API

Express je framework sloužící k jednoduché a rychlé implementaci API. Tento framework je zcela zdarma pod MIT licencí. Na následující ukázce kódu 4.1 je implementace jednoho z mnoha endpointů REST API použitých v aplikaci.

Výpis kódu 4.1 Ukázka Express

```
const jobsRouter = express.Router()
const jobsController = new JobsController()

jobsRouter.post(
    "/api/v1/jobs",
    body(CreateJobRequired, "required").notEmpty(),
    validateRequest,
    authenticateToken,
    jobsController.createJob
)
```

Tento kód vytvoří endpoint s HTTP metodou POST a cestou /api/v1/jobs. Nejdříve je potřeba vytvořit objekt pro vytváření endpointů pomocí příkazu express.Router(). Poté voláním metod nad tímto objektem jako je .post(...), .get(...), .delete(...)... se vytvoří endpoint s odpovídající HTTP metodou podle názvu metody. Jako první parametr se vždy specifikuje cesta endpointu, která bude společně s HTTP metodou volána. Následují názvy metod, které budou použity jako middleware. Middleware dále zpracovává data, která byla zaslána a odesílá odpovědi na dotazovaný endpoint. Middlewary se můžou takto řetězit a přecházejí vždy jeden do druhého. Zde například middleware authenticateToken zkontroluje JWT token a po kontrole následuje middleware jobsController.createJob, který práci vytvoří. Podrobněji popíšu jejich fungování v 4.3.3, kde rozeberu právě authenticateToken.

4.3.2 TypeORM a práce s databází

K práci s databázemi jsem využil TypeORM, které využívá ORM pro práci nad databází. Zvolený nástroj umožňuje vytvářet databázové tabulky a manipulovat s daty databáze pomocí objektů, které reprezentují tabulky v databázi. V následujících ukázkách kódu 4.2 a 4.3 se vyskytuje třída reprezentující tabulku databáze a manipulace s daty databáze. Tato třída reprezentuje tabulku results v databázi a je definována pomocí anotací začínajících @. Tyto anotace slouží pro namapování třídy na databázi. Každou anotaci lze blíže specifikovat v kulatých závorkách. Následující body popisují jednotlivé anotace:

@Entity: Určuje, že daná třída je mapováním třídy na tabulku v databázi. V závorkách je pak uveden název tabulky v databázi v tomto případě results.

@PrimaryGeneratedColumn: Definuje, že daný parametr je automaticky generován a je primárním klíčem tabulky. Jedná se o sloupec v tabulce.

@Column: Definuje, že parametr je sloupcem v tabulce. V závorkách lze pak specifikovat typ a omezení kladená na sloupec.

@ManyToOne: Tato definice specifikuje vztah ManyToOne s jinou tabulkou. Obdobně se definuje OneToOne, OneToMany a ManyToMany. Jako první parametr v závorce je název cizí tabulky, se kterou má vztah. Druhým parametrem určuji název sloupce pro napojení na tabulku. A třetím je další konfigurace.

Obecně jsou anotace a jejich konfigurace obšírné téma, kterému je věnována podstatná část dokumentace TypeORM a tohle je jen zlomek možností, které TypeORM poskytuje.

Výpis kódu 4.2 Ukázka TypeORM třída reprezentující tabulku databáze

```
export enum ResultType { VIDEO = "video", }

@Entity("result")
class Result extends BaseEntity {
  @PrimaryGeneratedColumn()
  id: number

  @Column()
  key: string

  @Column()
  location: string

  @Column({ type: "enum", enum: ResultType })
  type: ResultType

  @ManyToOne("participation", "results", {
    nullable: false,
    onDelete: "CASCADE",
  })
  participant: Participation
}
```

Výpis kódu 4.3 Ukázka TypeORM manipulace s daty databáze

```
#userRepository = dataSource.getRepository(User)
// create repository
#scoreSummaryRepository = dataSource.getRepository(ScoreSummary)
    ...
// encrypts the password vith sha256
const hashedPassword = await bcrypt.hash(password, 10)

// save data to database via. repository
const scoreSummary = await this.#scoreSummaryRepository.save({
  score: 0, jobsGiven: 0, jobsTaken: 0, numberOfReviews: 0,
})

await this.#userRepository.save({
  email, username, scoreSummary,
  password: hashedPassword,
  isAdmin: false,
  isBlocked: false,
})
```

V ukázce 4.3 je příklad, ve kterém do databáze pomocí TypeORM ukládám nového uživatele. Pro manipulaci s daty v TypeORM používám repositories. Ty se vytvoří pro každou tabulku zvlášť pomocí dataSource.getRepository(...), kde uvnitř závorek je uvedena třída namapovaná na tabulku v databázi. Metoda getRepository je volána nad instancí připojené databáze. To nám vrátí repository pomocí kterého se dále manipuluje s daty tabulky. Jako například tady pomocí metod .save(...) ukládám data do databáze. Další takovou metodou je například .delete(...), která maže data z tabulky nebo .find(...), která vyhledává data v tabulce. V závorce se pak specifikuje operace. V tomto případě, jsou to data co se mají uložit do databáze. Stejně jako v případě anotací je tohle pouze ukázka práce s repository a manipulaci s daty databáze je věnována značná část dokumentace TypeORM.

4.3.3 JWT

Pro práci s JWT využiji knihovny jsonwebtoken, která umožňuje JWT tokeny validovat a vytvářet. V ukázce kódu 4.4 je právě validace tokenů pomocí middleware.

Výpis kódu 4.4 Ukázka ověřování JWT tokenu

```
const authenticateToken =
  (req: Request, res: Response, next: NextFunction) => {
  const authHeader = req.headers.authorization
  const token = authHeader && authHeader.split(" ")[1]

  if (token || authHeader) {
    jwt.verify(
      token || authHeader,
      process.env.AUTH_TOKEN_SECRET_KEY || "key",
      (error, user) => {
        if (error) {
          res.status(403).send("invalid auth token")
        } else {
          res.locals = { authData: user as TokenPayload }
          next()
        }
      }
    )
  } else {
    res.status(403).send("invalid auth token")
  }
}
```

Nejdříve vysvětlím obecné fungovaní middleware na této ukázce. Middleware je funkce, která bere 3 parametry:

req: Obsahuje příchozí data dotazu (requestu), které byly odeslány na endpoint. Obsahuje data z hlavičky, těla i query parametrů odeslaných s voláním. V ukázce se pomocí req.headers.authorization přistupuje k tokenu, který je v hlavičce.

res: Tento parametr se využívá zejména k odesílání odpovědi, kde se specifikuje odpověď nebo HTTP status kód, který má být odeslán. Pomocí res parametru se přistupuje k metodám, které to umožní. V ukázce res.status(403).send(”invalid auth token”) odešle odpověď s HTTP status kódem 403 a zprávou ”invalid auth token”.

next: Je používán pro přesun do dalšího middlewaru. Pokud dojde k zavolání next(), tak bude po dokončení průběhu middlewaru spuštěn další middleware v pořadí.

Po získání tokenu z hlavičky requestu je token validován pomocí jwt.verify(...), kde je specifikován použitý klíč pro ověření. V případě neúspěchu ověření je vracen odpovídající HTTP status kód s chybovou hláškou. Pokud validace uspěje, uloží se informace obsazené v tokenu do res.locals, kde k nim bude mít přístup další middleware a zavolá se next() pro přesun na další middleware.

4.3.4 Práce s videi

Video se bude ukládat na AWS S3. K tomu využiji multerS3 a aws knihovny. K nahraní souboru je použit multerS3, aws pak slouží pro připojení k AWS S3 cloudovému úložišti. V ukázce kódu 4.5 je konfigurace middlewaru pro uložení videa do AWS S3 úložiště. Ten je pak použit při definici endpointu pro ukládání videí.

Výpis kódu 4.5 Ukázka ukládání videí

```
const s3 = new aws.S3({
  accessKeyId: process.env.AWS_ACCESS_KEY_ID,
  secretAccessKey: process.env.AWS_SECRET_KEY,
})

const uploadVideoS3 = multer({
  storage: multerS3({
    s3,
    bucket: "zchecknito-videos",
    acl: "public-read",
    metadata(req, file, cb) {
      cb(null, { fieldName: file.fieldname })
    },
    key(req, file, cb) {
      cb(null, `${Date.now()}-${file.originalname}`)
    },
  }),
  fileFilter: fileFilterVideo,
})
```

Nejdříve je vytvořena konfigurace pro připojení k AWS S3 pomocí new aws.S3(...), kde jsou specifikovány přihlašovací klíče vygenerované na stránkách AWS. Následně je pomocí multeru vytvořen middleware pro ukládání. Multer slouží pro získávání a ukládání souborů z příchozích dat na endpoint. Při jeho konfiguraci je pak storage definován pomocí multerS3, který získaná data přesměruje do AWS S3. MulterS3 vyžaduje dříve vytvořené spojení, které bylo uloženo do proměněné s3. Také je potřeba nastavit jméno složky, do které se soubor bude ukládat v parametru bucket. Zde je to "zchecknito-videos". Jako další zajímavý parametr je key funkce, která bude nastavovat jméno ukládaného souboru. Parametrem metadata se pak nastavují metadata předávaná AWS S3 a acl určuje práva pro práci se souborem. Aby bylo jméno souboru unikátní přidává se k němu aktuální čas. FileFilter pak stanovuje omezení na typ souboru, který je možno ukládat.

4.3.5 Emaily a notifikovaní

K zasílání emailu, pro notifikovaní uživatele nebo zaslání odkazu pro obnovu hesla, jsem využil knihovnu nodemailer. Která umožňuje připojit se k emailu a z něj odesílat emaily na jiné adresy. V ukázce kódu 4.6 je uvedeno odesílání vytvořeného odkazu pro obnovu hesla.

V mailOptions je uložena konfigurace emailu, který bude odeslán. Součástí konfigurace je předmět, kdo mail odesílá, komu bude doručen a text emailu. Konfigurace je následně předána nodemailerTransporter, což je jenom nakonfigurovaný nodemailer, a odeslána pomocí metody sendMail. V případě výskytu chyby bude vyhozena výjimka.

Výpis kódu 4.6 Ukázka odesílání emailu

```
const mailOptions = {
  from: "zchecknito@gmail.com",
  to: email,
  subject: "Zapomenute heslo",
  text: `Dobry den, zasilame Vam link na obnoveni hesla
  plati po dobu 15 minut: \n${process.env.CLIENT_URL}
  /reset-password?resetToken=${resetPaswordToken}`,
}

await nodemailerTransporter.sendMail(mailOptions).catch(err => {
  throw new Error(err)
})
```

4.4 Frontend

Cílem frontendu je vytvoření uživatelského rozhraní prototypu. Toto rozhraní by mělo umožnit uživateli přihlášení, registrace atd. Toho se dosáhne implementací wireframu navrženého v 3.9.1 a jeho následné napojení na backend pro ukládání, provádění operací a dalších manipulací s daty.

V následujících částech popíšu, jak jsem využil technologie pro tvorbu frontendu navržené v kapitole 3, přičemž z nich vyberu ty nejdůležitější. V každé části uvedu vždy její stručný popis a ukázku kódu, na které technologii předvedu a rozeberu obdobně jako v backend části.

4.4.1 React

React je knihovna vytvořená společností Facebook a slouží jako platforma pro tvorbu frontendu aplikace. Umožňuje tvorbu jednotlivých stránek s jejich funkcionálem. V ukázce kódu 4.7 je implementace komponenty. Skládáním takových komponent se pak tvoří jednotlivé stránky a funkcionalita v nich. Vytvořená komponenta v ukázce pak například odpovídá za vykreslení textu.

Výpis kódu 4.7 Ukázka React komponenty

```
type InformationTextProps = {
  color?: string
}

const InformationText: FC<InformationTextProps> = ({
  color = "primary",
  children,
}) => {
  return (
   <Typography variant="h4" color={color} textAlign={"center"} py={"7%"}>
      {children}
   </Typography>
  )
}
```

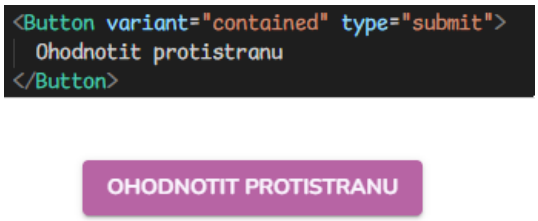
Tento kód vytváří takzvanou funkční komponentu, která muže být dále používána v dalších funkčních komponentách a je volána pomocí tagů. Tato komponenta přijímá parametry color a children. Color se pak předává jako parametr jiné komponentě Typography a určuje barvu textu. Parametr children pak slouží pro předání všeho, co bude uvnitř tagu komponenty. Tedy v tomto případě vše uvnitř tagů <InformationText>...</InformationText> se předá jako children. Jak

je vidět na komponentě Typography uvnitř prvního otevíracího tagu jsou nastaveny parametry jako variant, color, textAlign a py, které budou komponentě předány.

Důležitým konceptem Reactu jsou hooky. I když v ukázce nejsou používány rád bych je krátce popsal. Hooky jsou určeny k psaní logiky, která bude aplikovaná v životním cyklu komponenty. Umožňují udržovat a měnit stav v jeho průběhu a mnoho dalšího. Nejznámějším takový hookem je useState(...), který dovoluje již zmíněnou změnu a udržování stavu.

4.4.2 Material UI

V předchozí částí jsem ukázal tvorbu komponent. Abych ale zbytečně neztrácel čas vytvářením a stylováním jednotlivých komponent, jako jsou tlačítka atd., využiji komponentové knihovny Material UI, která poskytuje již vytvořené a nastylované komponenty. Ty už pak jen skládám v Reactu jako stavebnici lego a vytvářím pomocí nich jednotlivé stránky. Například v předchozí ukázce kódu jsem použil Typography, které je součástí komponentové knihovny Material UI. Na obrázku 4.1 je pro ilustraci uvedeno použití komponenty Button pro tvorbu tlačítka a jeho výsledná podoba na stránce.



Obrázek 4.1 Ukázka Material UI Button komponenty

4.4.3 Axios

Pro komunikaci frontendu s backendem využiji Axios klienta, který bude odesílat požadavky na endpointy backendu. Pro generaci tohoto klienta jsem použil swagger a nástroj swagger-typescript-api. V ukázce kódu 4.8 je použit vygenerovaný klient.

Výpis kódu 4.8 Ukázka použití Axios klienta

```
...
const api = new Users()

const res = await api.loginUser(data)
...
```

Generace klienta spočívá ve vytvoření tříd, které umožňují komunikaci s endpointy. Nad jejími objekty se pak volají metody, které odešlou dotaz na endpoint. V ukázce jsem nejdříve vytvořil objekt třídy Users, který umožňuje volání endpointů začínajících na api/v1/users. Nad instancí třídy jsem pak zavolal metodu loginUser, které jsem předal data k přihlášení. Volaná metoda pak zařídí odesílaní dat na endpoint api/v1/users/login a vrátí data, která vrátil endpoint.

4.5 Shrnutí

Cílem této kapitoly bylo popsat implementaci a využití navržených technologií. Ty jsem jednotlivě popsal i když ne všechny a soustředil jsem při tom na to, zachytit ty nejdůležitější.

Kromě technologií jsem zde uvedl i strukturu implementace a jak jsem při ní postupoval.

Kapitola 5
Testování

V této kapitole otestuji implementovaný prototyp. Budu se zde zabývat návrhem testů, jejich aplikací a následnou analýzou jejich výsledků. Kromě toho otestuji aplikaci pomocí nástroje Lighthouse, pro zhodnocení technické stránky. Výsledkem by měla být získaná a zanalyzovaná testovací data, která využiji pro určení nedostatků prototypu.

5.1 Návrh a postup testování

Testování jsem se rozhodl vést spíše kvalitativní nežli kvantitativní formou, a to na základě uživatelského testování. Před každým testováním jsem se rozhodl krátce aplikaci popsat, abych uživatele uvedl do problematiky. Uživatelské testování bych chtěl vést formou kombinace osobního pohovoru a plnění mnou zadaných úkolu v aplikaci.

Tyto úkoly se budou řídit testovacími scénáři, které mají za účel, aby si testovací subjekt vyzkoušel nejdůležitější aspekty aplikace. V jednotlivých scénářích bude uživateli řečeno, co má v aplikaci udělat. Po splnění scénáře následně ohodnotím, jak ho subjekt zvládl. Po tom, co projdeme všechny scénáře, položím testovacímu subjektu otázky týkající se jeho zkušeností s prototypem.

Jelikož jsem volil formu testovaní, která je spíše kvalitativní, a tedy i více časově náročná počítám s 5-6 testovacími subjekty. Dle mého odhadu by každý test měl zabrat 20-30 minut.

5.2 Scénáře testování

V dané části nejdříve navrhnu jednotlivé scénáře a následně vysvětlím, jak budu hodnotit úspěšnost jejich splnění. Tyto scénáře mají za účel zachytit zkušenost uživatele pří používání prototypu.

5.2.1 Volba scénářů testování

Při volbě scénářů se zaměřím na pokrytí klíčových aspektů prototypu. Těmito aspekty pro mě budou funkční požadavky uvedené v 2.3 a na jejich základě a na základě jejich priorit budu navrhovat scénáře. Testovat budu hlavně požadavky s vysokou a střední prioritou. Následující body popisují jednotlivé scénáře, kde je uveden úkol, který má uživatel splnit a funkční požadavky, které testuje. Body budou na sebe navazovat a zadávat je budu v uvedeném pořadí.

T01 – Registrace a přihlášení

Uživatel se zaregistruje a následně přihlásí. Registrační email bude uživateli přidělen. (Protože se s ním dále pracuje v některých bodech.)

Požadavky: F01

T02 – Úprava profilu

Uživatel si upraví údaje v profilu a doplní údaje, které nejsou po registraci vyplněny.

Požadavky: F03

T03 – Tvorba práce

Uživatel vytvoří práci a následně se ji pokusí najít v nabídkách práce.

Požadavky: F04, F09

T04 – Přihlášení k plnění

Uživatel nalezne specifickou práci v nabídkách prací. (Bude mu zadán její název, cena, kraj a datum.) Nalezenou práci si otevře a přihlásí se k jejímu plnění. Před přihlášením si zobrazí profil zadavatele.

Požadavky: F05, F09, F10

T05 – Potvrzení plnění

Uživatel si zobrazí práce, které vzal a jsou připravené k potvrzení. Následně nalezne práce, které jsou mu přiřazené a u práce z bodu T04 potvrdí její plnění.

Požadavky: F05, F09

T06 – Nahrání a odeslání výsledků

Uživatel nalezne práce, které vzal a plní. Následně nalezne práci, u které v bodě T05 potvrdil plnění a nahraje video. Po nahraní uživatel odešle výsledky.

Požadavky: F05, F07, F09

T07 – Ohodnocení protistrany

Uživatel nalezne práce, které vzal a splnil. Následně nalezne práci, u které v bodě T06 odeslal výsledky. Výsledky si přehraje a udělí recenzi protistraně.

Požadavky: F05, F07, F08, F09

T08 – Splnění zadané práce

Uživatel nalezne práce, které vytvořil a nejsou zadané plnitelům. Následně nalezne práci, kterou v bodě T03 vytvořil. Změní parametry zadané práce a následně nalezne v zájemcích uživatele podle jména a zobrazí si jeho profil. Zobrazeného si uživatel přiřadí k plnění.

Požadavky: F04, F09, F10

T09 – Ohodnocení splnění zadané práce

Uživatel nalezne práce, které vytvořil a jsou splněné. Následně nalezne práci, kterou v bodě T08 přiřadil. U ní si přehraje výsledky a udělí protistraně recenzi.

Požadavky: F04, F07, F08, F09

T10 – Obnovení hesla

Uživatel si obnoví heslo pomocí emailu, který zadal při registraci. Po obnovení se pokusí přihlásit.

Požadavky: F02

5.2.2 Hodnocení scénářů testování

U každého bodu budu hodnotit, s jaký úspěchem uživatel zvládl scénář. Hodnotím vždy na stupnici:

- zvládl bez problému - 3
- zvládl - 2
- nezvládl - 1

Ke každému z hodnocení je uvedeno i číslo odpovídající bodovému ohodnocení. V případě hodnocení "zvládl"bylo potřeba uživatele nějakým způsobem popostrčit nebo mu vykonání scénáře trvalo moc dlouho. Pokud je hodnocení "nezvládl"vyskytly se komplikace kvůli nimž nešlo scénář dokončit anebo bylo potřeba uživateli hodně pomoct. Nejlepším případem je potom, když uživatel zvládl scénář bez problému.

5.3 Diskuze

Po dokončení scénářů budu navazovat diskuzí o zkušenostech práce s prototypem. Diskuze si budu nahrávat a ve výsledcích testování shrnu jejich obsah. V diskuzích se zaměřím jak na pohodlnost při používání prototypu, tak na celkovou přehlednost a podněty ke zlepšení. Nejdůležitější ale bude celková spokojenost nebo nespokojenost s používáním. Diskuze povedu formou otázek, které se budou soustředit na pokrytí popsaných témat zaměření diskuze výše. Kromě úvodní otázky, která se zaměří na míru zkušeností uživatele. Vytvořil následující otázky:

1. Jakou máte zkušenost s používáním počítače? Ohodnoťte se na škále od 1 do 10.
2. Přišla Vám aplikace přehledná (intuitivní) a co se Vám na ní líbilo?
3. Co Vám v testovaném prototypu nejvíce chybělo?
4. Co Vám na testovaném prototypu nejvíce vadilo?
5. Používali byste podobnou aplikaci?

5.4 Výsledky testování

V této části shrnu, jak dopadlo testování. Výsledky jsem se rozhodl rozdělit do dvou částí, a to na výsledky plnění scénářů a výsledky dotazování. Testování se zúčastnilo 5 lidí.

5.4.1 Výsledky scénářů

Výsledky plnění scénářů řeknou především, jak je implementovaný prototyp uživatelsky přívětivý a dá uživateli možnost si udělat představu o prototypu. Také odhalí značné nedostatky a chyby implementace. V tabulce 5.1 je uvedeno bodové hodnocení jednotlivých úkolů od každého testera.

Tabulka 5.1 Výsledky testování scénářů

Uživatel	T01	T02	T03	T04	T05	T06	T06	T07	T08	T09	T10
Uživatel 1	3	2	3	2	2	2	2	2	3	3	2
Uživatel 2	3	2	3	3	3	3	3	2	2	3	3
Uživatel 3	3	2	2	2	3	3	2	3	3	3	1
Uživatel 4	3	2	3	3	3	3	3	3	3	2	3
Uživatel 5	3	2	2	2	3	2	2	2	3	2	2

Testy scénářů dopadly dobře a až na jeden případ všichni testeři úspěšně splnili všechny scénáře. Jak je vidět T02 dostával konzistentně za 2 a to, protože tlačítko pro vytvoření práce nebylo dostatečně výrazné. Dalším poznatkem, který můžeme z testování scénářů získat, je nepřehlednost stránek s mými pracemi. Většina uživatel musela relativně dlouho hledat správný filtr a často docházelo k proklikávání všech filtrů, než testeři nalezli hledanou práci.

Nevhodné se ukázalo i to, že domovská obrazovka uživatele je zároveň jejich profilem. Spoustu uživatelů to mátlo a hledali svůj profil v nabídce menu, společně s nastavením.

Nakonec bych se chtěl pozastavit u jednoho nezdaru, a to konkrétně u uživatele 3 na scénáři T10, který se mu nepodařilo dokončit. Jednalo se o test pro obnovu hesla a jeho neúspěch spočíval v neplatném odkazu pro obnovení hesla.

To bylo záhy opraveno a příčinou chyby se ukázalo, že expirace validity odkazu měla příliš krátkou dobu platnosti. Tato hodnota byla omylem nastavena na 3 minut místo 15. Během toho, než se odkaz podařilo otevřít tak došlo k jeho expiraci.

5.4.2 Výsledky dotazování

Po splnění všech scénářů a potom, co si uživatelé udělali názor na aplikaci, nastal čas na dotazování. Jednotlivé otázky bych chtěl v následujících částech rozebrat. Ke každé otázce uvedu jednotlivé názory uživatelů. V závěru každé části pak odpovědi shrnu.

Otázka 1

Otázka: Jakou máte zkušenost s používáním počítače? Ohodnoťte se na škále od 1 do 10.

Uživatel 1: 10, **Uživatel 2:** 10, **Uživatel 3:** 9, **Uživatel 4:** 10, **Uživatel 5:** 6

Celkově považuji všechny uživatele, kteří testovali aplikaci, za zkušené, s výjimkou jednoho. Což pokrývá všechny skupiny, na které aplikace cílí, a to až na úplně nezkušené jako například na důchodci.

Otázka 2

Otázka: Přišla Vám aplikace přehledná (intuitivní) a co se Vám na ní líbilo?

Uživatel 1: "Celkově aplikace působila dobrým dojmem a přišla mi intuitivní. Líbil se mi hlavně design aplikace."

Uživatel 2: "Aplikace byla intuitivní, ale stránka Moje Práce mi přišla zbytečně komplikovaná. Také moc nechápu, proč se mi při přihlášení zobrazí můj profil a jak to, že není v menu společně s nastavením. Líbil se mi hlavně design a uživatelsky přívětivé prostředí."

Uživatel 3: "Intuitivní mi přišla hlavně část s přihlášením a zobrazením všech prací. Musím si stěžovat na umístění tlačítka pro zobrazení vlastního profilu, které bylo nevhodné. V aplikaci se mi líbilo především, že si na nic nehraje a má hezký a přívětivý design."

Uživatel 4: "Aplikace působila celkově intuitivně. Jsem zkušenější uživatel a líbilo se mi využití cloudových služeb pro ukládaní videí."

Uživatel 5: "Intuitivnost hodnotím jako průměrnou, hlavně kvůli komplikované stránce Moje Práce. Líbil se mi design aplikace."

Všem přišla aplikace vesměs intuitivní, několikrát ale zaznělo, že stránka Moje práce je komplikovaná a umístění tlačítka pro zobrazení vlastního profilu je nevhodné. Na aplikaci se uživatelům líbil především její design.

Otázka 3

Otázka: Co Vám v testovaném prototypu nejvíce chybělo?

Uživatel 1: "Chyběli mi notifikace přímo v aplikaci a považuji za velký nedostatek, že se neukazují notifikace o provedených změnách. Notifikace zasílané prostřednictvím emailu mi přišly nedostačující. Další věcí, která mi chyběla, byla možnost vyhledávání prací podle názvu."

Uživatel 2: "Především mi chyběly notifikace v aplikaci. Také bych chtěl mít možnost komunikovat s protistranou v aplikaci. Potom, co jsem zjistil, že notifikace dostávám emailem považuji za chybu, že o tom v aplikaci není ani zmínka. Chyběla mi také možnost vyhledávání práce podle názvu."

Uživatel 3: "Chtěl bych mít možnost komunikovat s protistranou přímo na stránce. Notifikace mi přijdou dostačující prostřednictvím emailu, ale opět jako v ostatních případech mi chybělo vyhledávání práce podle názvu."

Uživatel 4: "Chybělo mi zobrazení, kde bych viděl všechny moje práce najednou a mohl bych si podle libosti filtrovat. Notifikace v aplikaci bych taky ocenil, ale nejsou pro mě až tak podstatné. Horší už to vidím u toho, že nemám možnost komunikace s protistranou přímo na stránkách."

Uživatel 5: "Jediné co musím vytknout je, že mi chybí komunikace s protistranou."

Skoro každý z dotazovaných uvedl hlavně chybějící komunikaci s protistranou, a to ať už v podobě chatu nebo formulářů v aplikaci. Kromě toho hodně uživatelům chyběly notifikace v aplikaci a hledání objednávek podle názvu.

Otázka 4

Otázka: Co Vám na testovaném prototypu nejvíce vadilo?

Uživatel 1: "Hlavně nepřehledná stránka Moje práce. Chtěl bych, aby byly filtry více odlišitelné nebo někde schované. Tlačítko vytvořit práci je zapadlé a není dostatečně viditelné. Také mi vadila absence notifikací na stránce."

Uživatel 2: ”Stránka Moje práce na mně působila komplikovaně a obtížně se v ní hledalo tlačítko pro vytvoření práce. Dále mi vadila chybějící zmínka o notifikacích na email a absence vypnutí notifikací.”

Uživatel 3: ”Nelogické umístění tlačítka pro zobrazení vlastního profilu. Po přihlášení bych očekával, že budu přesunut k nabídkám prací, a ne na vlastní profil. Zbytečně komplikovaná stránka s vlastními pracemi stačilo by tlačítko s filtry.”

Uživatel 4: ”Především nemožnost komunikace s klientem a stránka Moje práce. Stejně jako uživateli 3 by mi stačilo tlačítko s filtry.”

Uživatel 5: ”Nevýrazné tlačítko pro vytvoření práce a nelogické umístění tlačítka s vlastním profilem.”

Všichni uživatele se shodují na nevhodném řešení stránky Moje práce, která je zbytečně komplikovaná. Dalším bodem je pak nevýraznost tlačítka pro vytvoření práce, absence komunikace protistran a nelogické umístění tlačítka profilu spolu s tím, že je zobrazen jako úvodní obrazovka.

Otázka 5

Otázka: Používal byste podobnou aplikaci?

Uživatel 1: ”Ano, ale spíše jako někdo, kdo práce plní. Za plnění bych byl ochoten zaplatit 400 korun a za to bych i klidně práce plnil.”

Uživatel 2: ”Ano, za 500 korun bych bral obojí.”

Uživatel 3: ”Z pohledu toho, že bych je plnil, určitě. Ale nesvěřil bych osobní prohlídku nikomu jinému. Za plnění bych požadoval 300 až 400 korun.”

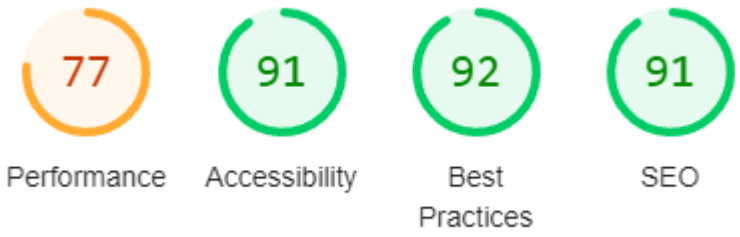
Uživatel 4: ”Ne z ani jednoho pohledu. Hlavně kvůli mé nedůvěře v lidi a absenci volného času.”

Uživatel 5: ”Ano, primárně bych ale chtěl práce plnit a ne zadávat. Částka, kterou bych považoval za férovou, je 600 korun za plnění.”

Celkově o aplikaci zájem, ale spíše převládá zájem o plnění prací než jejich zadávaní. Častým důvodem je nedůvěra v ostatní lidi. Cena za práci by se tak podle testovaných subjektů měla pohybovat kolem 450 korun.

5.5 Technická stránka

Technickou stránku prototypu budu měřit pomocí nástroje Lighthouse, který jsem uvedl v kapitole 2. Na obrázku 5.1 se nachází výsledky měření.



Obrázek 5.1 Hodnocení Lighthouse pro implementaci prototypu

Hodnocení Lighthouse dopadlo ve srovnáním s identifikovanou konkurencí v kapitole 2 průměrně a jistě je co zlepšovat. Obzvláště špatně dopadlo hodnocení Performance, kvůli velké velikosti JavaScript bundleů, což způsobovalo delší dobu načítání stránek. Bude tedy potřeba provést optimalizace vedoucí k jeho zmenšení. Toho lze dosáhnout například minifikací JavaScript souborů. Také by se tato kategorie hodnocení dala zlepšit preloadem obrázků.

5.6 Nalezené nedostatky

Z testovacích dat bych chtěl nyní uvést několik nalezených nedostatků, které by se měly v další iteraci vývoje opravit:

Chybějící komunikace s protistranou: V průběhu testování vyšlo najevo, že řadě uživatelům chyběla komunikace s protistranou, při plnění objednávky. Při průzkumu a stanovování požadavků byla komunikace s protistranou opomenuta. Jak ale vyplynulo z testů, tak je to potřeba a měla by to být hlavní prioritou v další fázi vývoje. Řešit by se to dalo například přidáním chatu k pracím.

Komplikovaná stránka Moje práce: Stránka Moje práce, která odpovídá za zobrazení prací, na kterých se uživatel podílí, je nepřehledná. Je potřeba její re-design a zjednodušení. Ideální by bylo zbavit se kolonky s filtrem stavu prací a přesunout filtry do tlačítka, pomocí kterého by se filtrace prováděla.

Nevhodné umístění elementu: Obzvláště tlačítko pro zobrazení vlastního profilu uživatelé očekávali společně v menu s nastavením a odhlášením. Po přihlášení by se také měla zobrazit stránka s vlastními pracemi nebo nabídkami prací místo profilu. Jako poslední je drobnost, která spočívá v nevýraznosti tlačítka pro vytváření nových prací.

Chybějící hledaní prací podle názvu: Chybělo hledání prací podle názvu a pouze filtry a řazení se ukázali jako nedostatečné řešení pro vyhledávání prací. Bude tedy potřeba přidat hledání podle názvu.

Notifikace: Uživatelům přišlo nedostatečné notifikovaní pomocí emailu a požadovali i notifikace v aplikaci. Také chyběla zmínka o notifikacích pomocí emailu a možnost jejich vypnutí či zapnutí.

Dlouhá načítající doba: Dlouhé načítání stránek jsem řešil v 5.5, kde jsem navrhl i vhodnou optimalizaci.

Kromě těchto nedostatků je také potřeba doimplementovat požadavek s nízkou prioritou na administrátorské prostředí. Pro něj už jsou vytvořeny endpointy a chybí jen napojení frontendu.

5.7 Shrnutí

Cílem této kapitoly bylo navrhnout testy pro implementaci a také ji otestovat. To se mi podařilo navržením testů, které kombinovali plnění scénářů a následné dotazování. Také jsem otestoval aplikaci pomocí nástroje Lighthouse, který dává objektivní data o technickém stavu implementace.

Získané poznatky z testů jsem následně využil a stanovil jsem nedostatky prototypu, které je potřeba opravit v další části vývoje.

Kapitola 6
Závěr

Cílem práce bylo navržení a implementace prototypu webového portálu pro tvorbu video reportu. Nejdříve byla provedena analýza požadavků cílových zákazníků. Tyto požadavky se podařilo získat pomocí analýzy stávajících řešení a byly rozděleny na funkční a nefunkční. Po získaní požadavků bylo pomocí metod softwarového inženýrství navrženo vhodné řešení pro naplnění získaných požadavků. To zahrnovalo jak návrh funkcionálu, tak i technickou stránku, ve které byly určeny nástroje a postupy pro následující implementaci prototypu.

Pro nejdůležitější zvolené technologie bylo následně ukázáno jejich praktické použití v implementaci prototypu. Výsledný prototyp byl následně podroben uživatelskému testování a testovaní pomocí nástroje Lighthouse. Obojí odhalilo řadu chyb a chybějících funkcionalit prototypu, které byly popsány v nalezených nedostatcích 5.6.

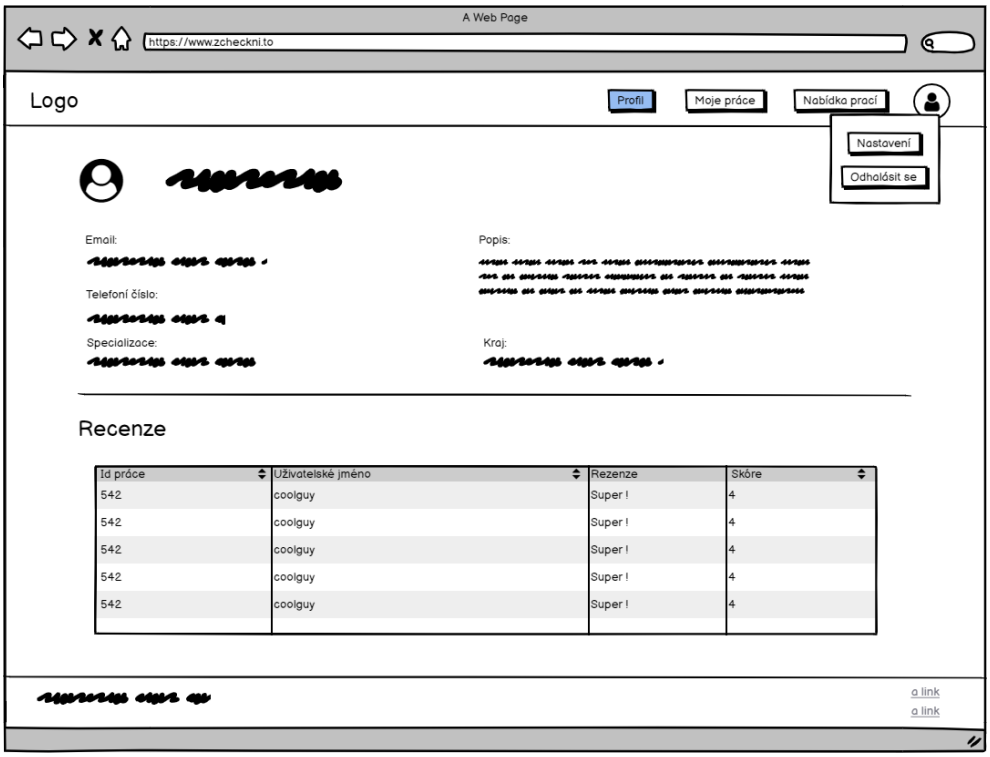
Celkově implementace dopadla dobře a podařilo se mi implementovat prototyp, který splňuje stanovené požadavky. Opomenu-li požadavky s nízkou prioritou. A splňuje i cíle stanovené v zadání práce.

Zároveň jsem získal nové požadavky z testování na chybějící funkcionál aplikace, který bude řešený v další iteraci vývoje portálu, společně s požadavky s nízkou prioritou. Kromě toho by se dalo na prototyp navázat, jeho dovedením do produkčního stavu a následným nasazením.

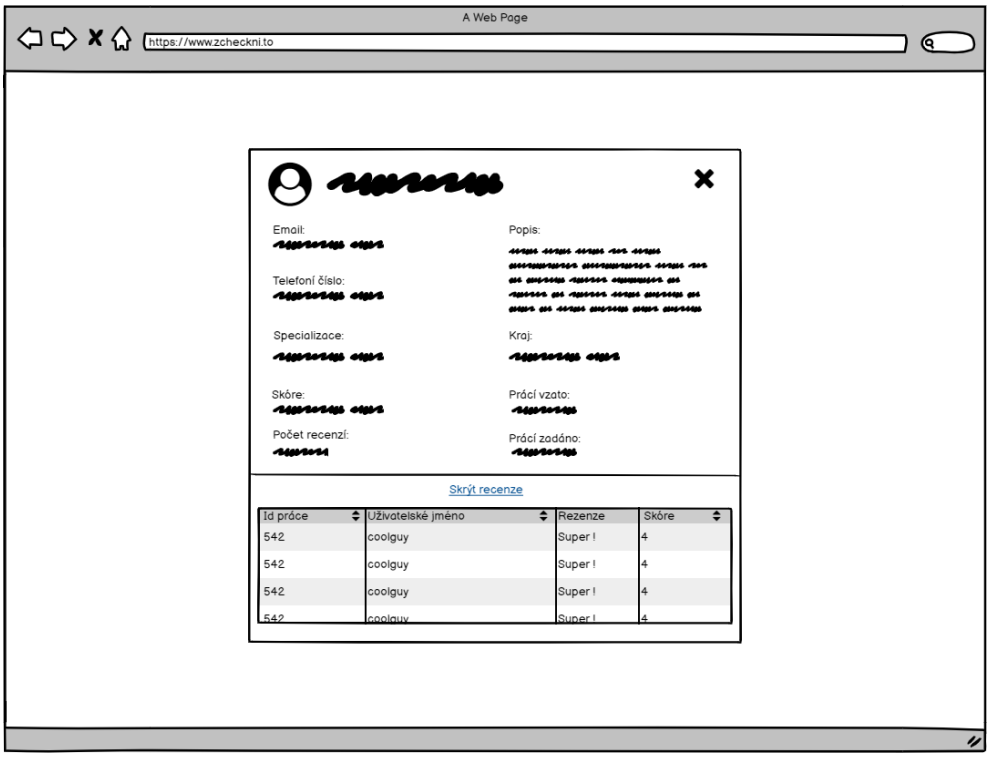
Navázání na bakalářskou prací by pak mělo být jednoduché, díky dodržení best practices, dobrému návrhu a rozšiřitelností implementovaného prototypu.

Příloha A

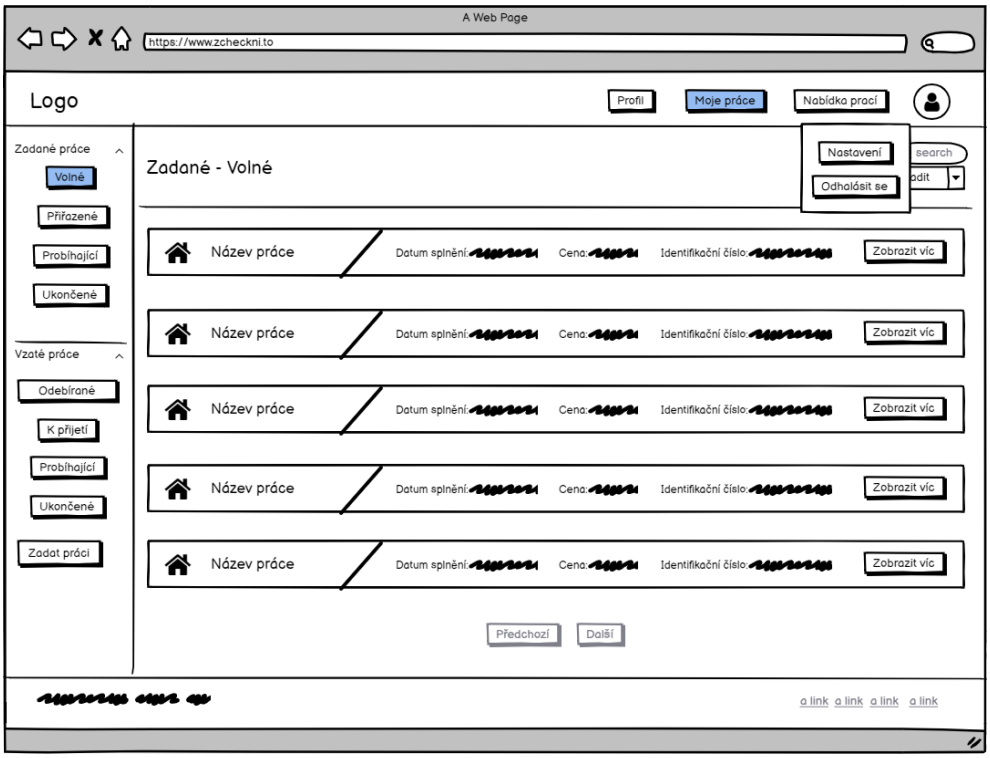
Wireframe



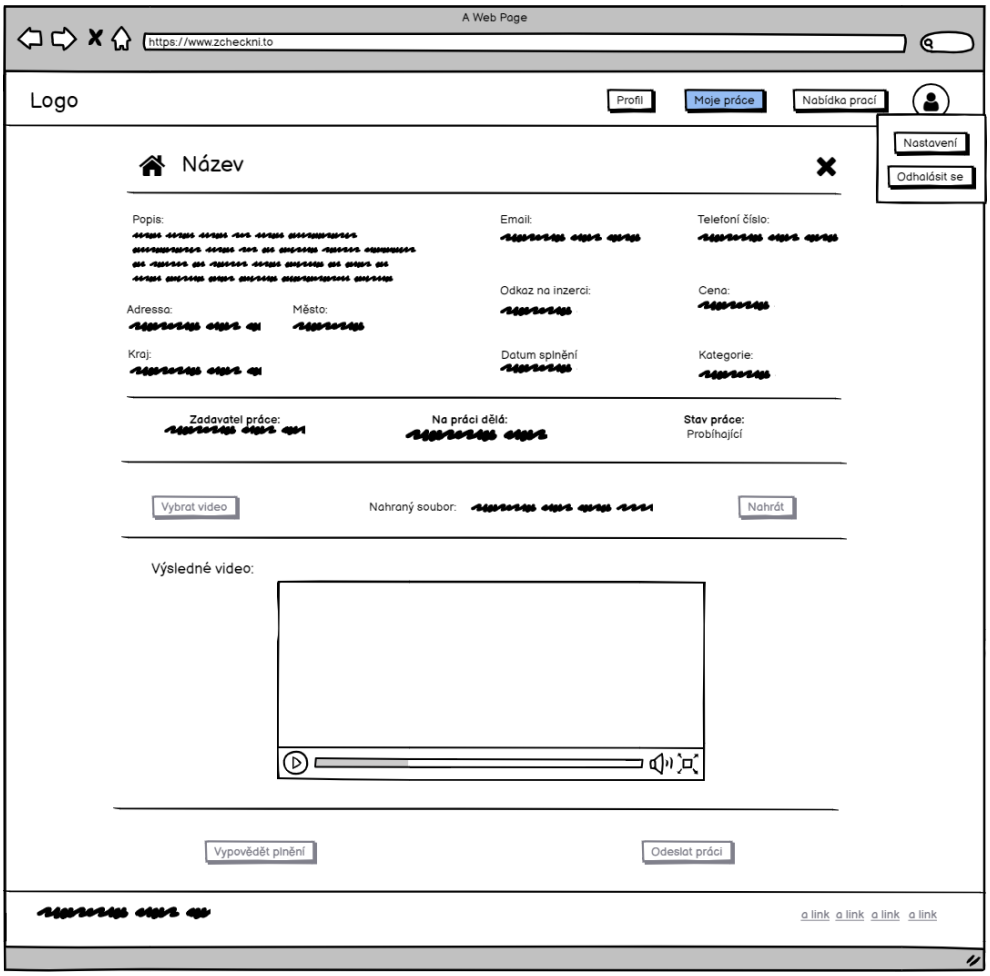
Obrázek A.1 Vlastní profil



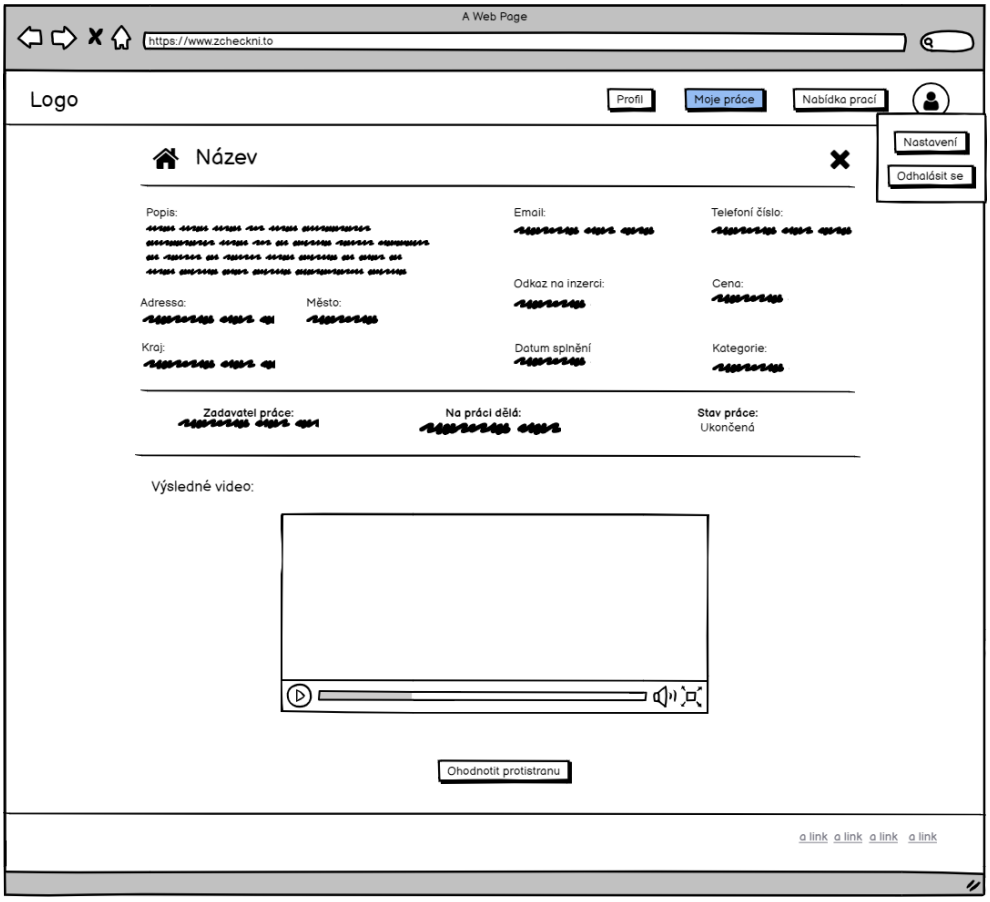
Obrázek A.2 Cizí profil



Obrázek A.3 Moje práce



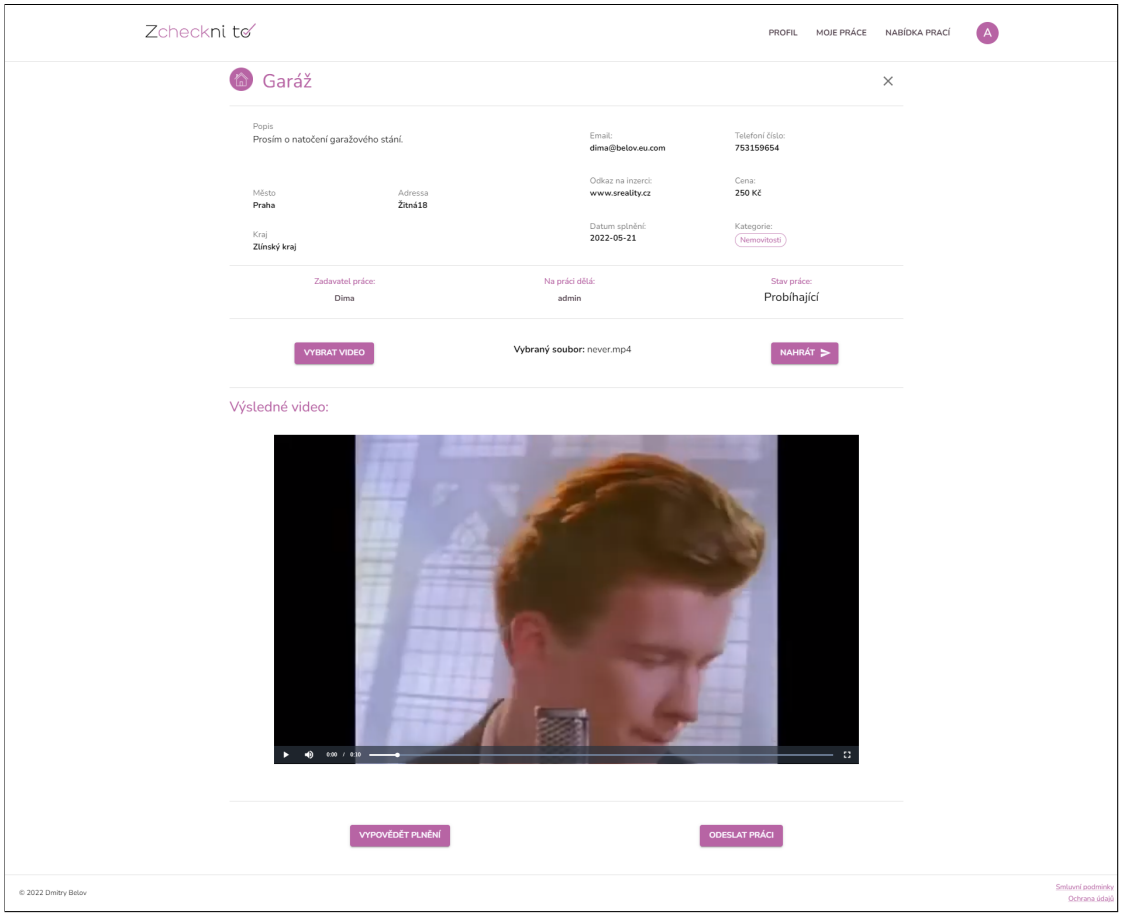
Obrázek A.4 Probíhající práce plnitele



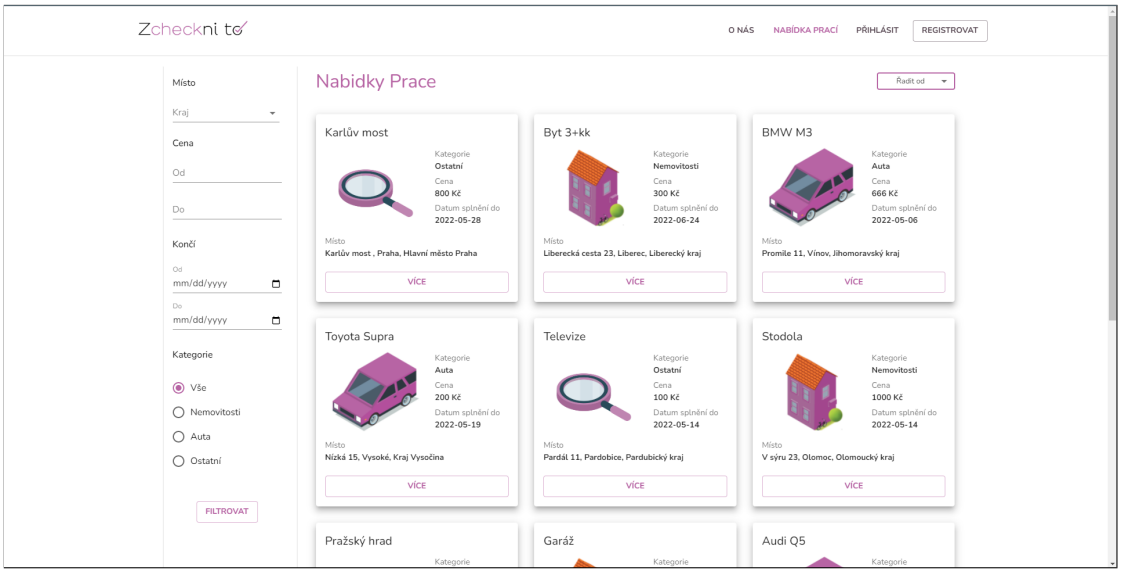
Obrázek A.5 Splněná práce

Příloha B

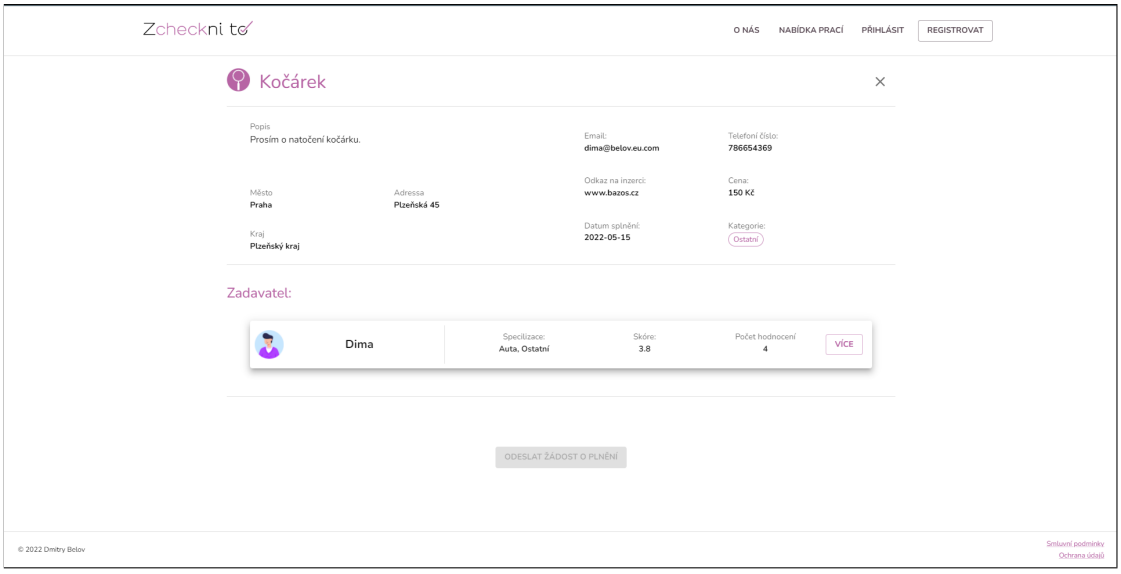
Ukázky prototypu



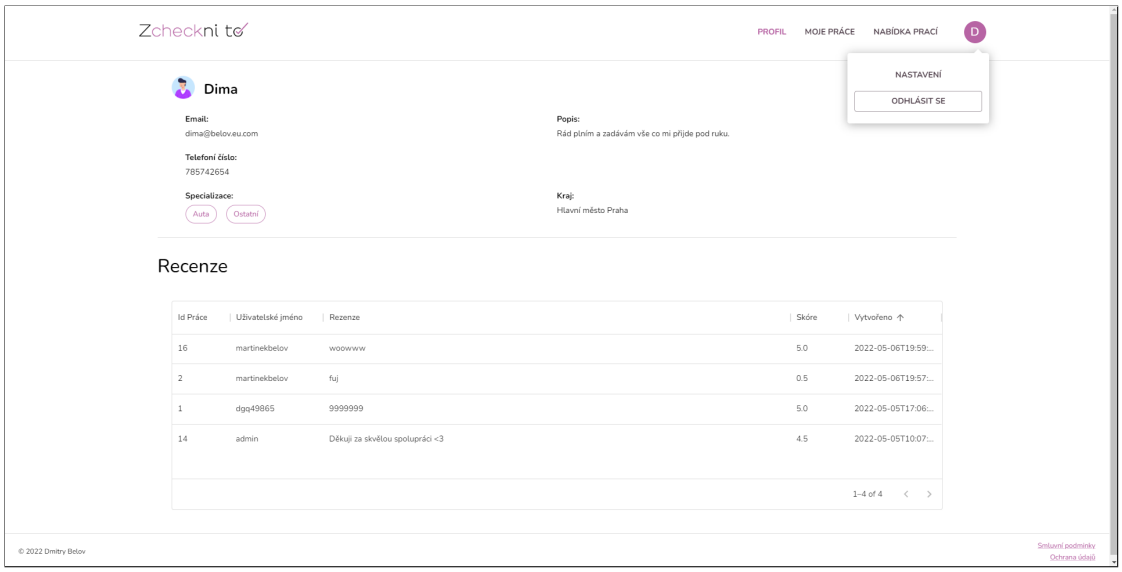
Obrázek B.1 Probíhající práce plnitele



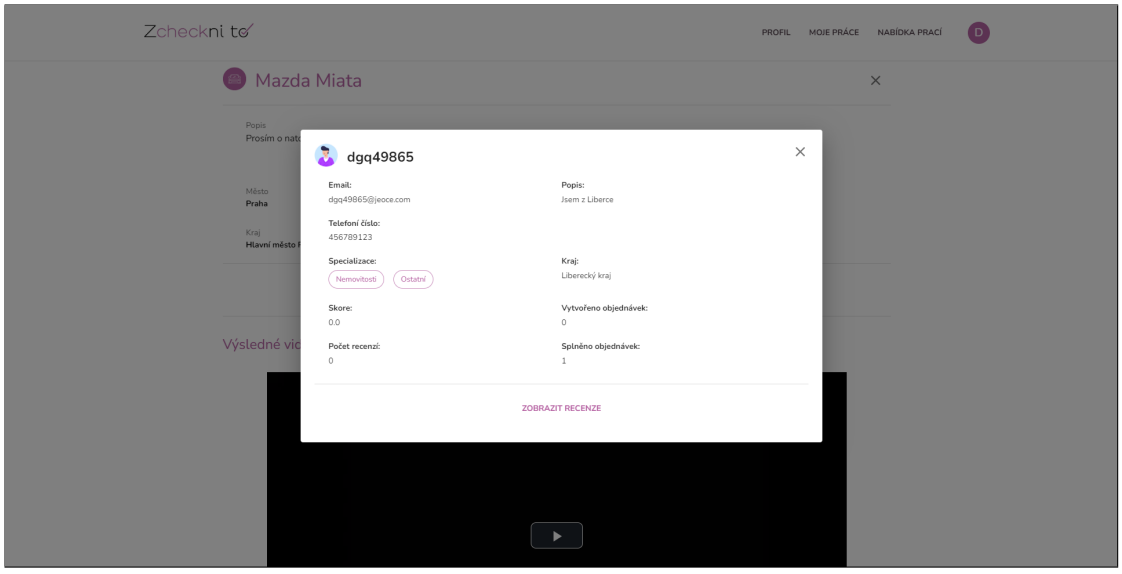
Obrázek B.2 Nabídky prací



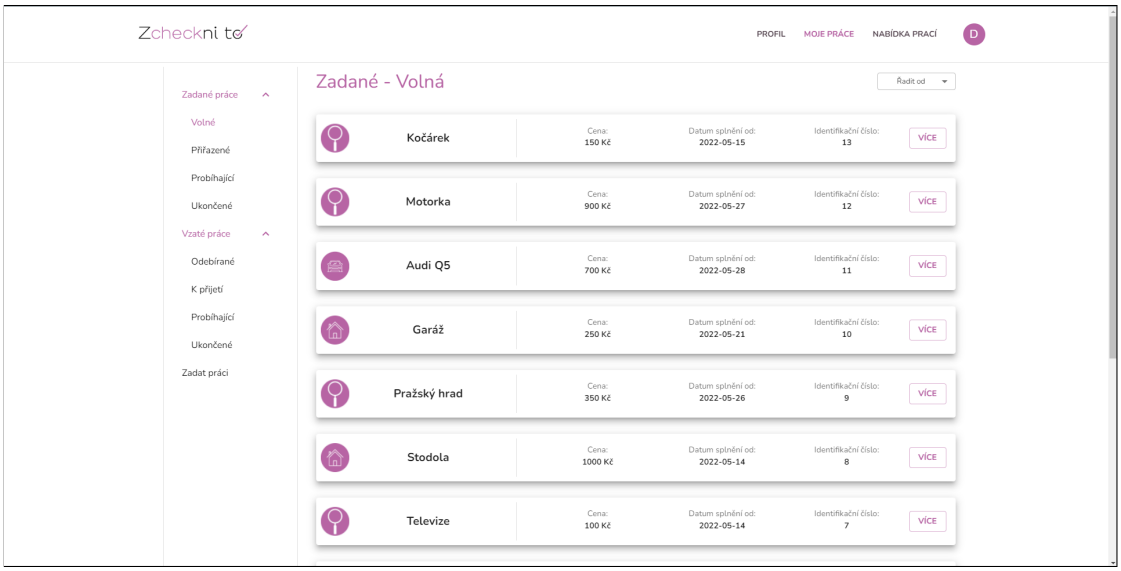
Obrázek B.3 Nabízená práce



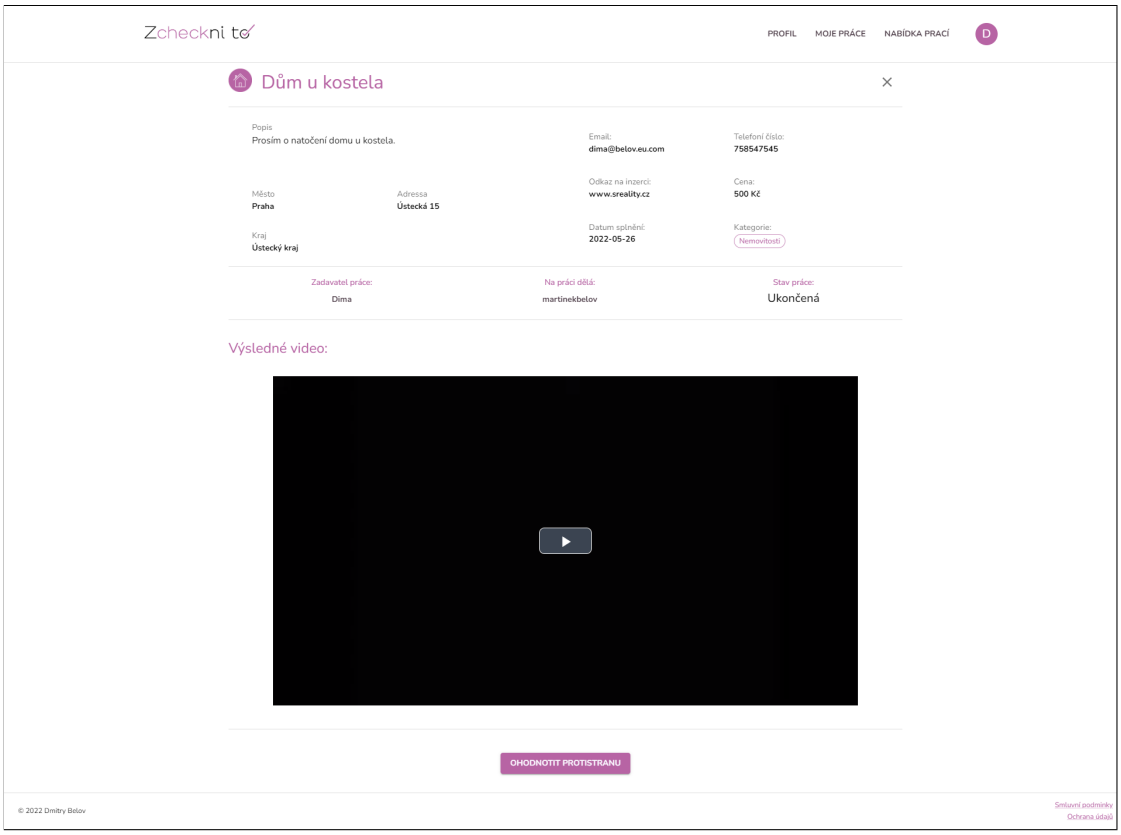
Obrázek B.4 Vlastní profil



Obrázek B.5 Cizí profil



Obrázek B.6 Moje práce



Obrázek B.7 Splněná práce

Příloha C

REST API

Tabulka C.1 Endpointy pro operace uživatele

metoda HTTP	endpoint	popis
POST	/users	vytvoří uživatele
GET	/users	vrátí seznam uživatelů
POST	/users/login	přihlásí uživatele
POST	/users/refresh-auth-token	obnoví autorizační token uživatele
POST	/users/logout	odhlásí uživatele
POST	/users/forgot-password	odešle na email odkaz pro obnovení hesla
POST	/users/reset-forgotten-password	obnoví zapomenuté heslo
POST	/users/:username	upraví data uživatele s uživatelským jménem :username
GET	/users/:username	vrátí uživatele s uživatelským jménem :username
POST	/users/:username/change-password	změní heslo uživateli s uživatelským jménem :username
POST	/users/:username/reviews	vytvoří recenzi pro uživatele s uživatelským jménem :username
GET	/users/:username/reviews	vrátí seznam recenzí uživatele
DELETE	/users/:username/reviews/:reviewId	odstraní recenzi s identifikačním číslem :reviewId

Tabulka C.2 Endpointy pro operace administrátora

metoda HTTP	endpoint	popis
POST	/admins	přidá uživateli administrátorská práva
DELETE	/admins/:username	zbaví uživatele s uživatelským jménem :username administrátorský práv
POST	/admins/user-managment/blocked	zablokuje uživatele
GET	/admins/user-managment/blocked	vrátí seznam zablokovaných uživatelů
DELETE	/admins/user-managment/blocked/:username	odblokuje uživatele s uživatelským jménem :username

Tabulka C.3 Endpointy pro operace nad pracemi

metoda HTTP	endpoint	popis
POST	/jobs	vytvoří práci
GET	/jobs	vrátí seznam prací
POST	/jobs/:jobId	upraví práci s identifikačním číslem :jobId
GET	/jobs/:jobId	vrátí práci s identifikačním číslem :jobId
DELETE	/jobs/:jobId	odstraní práci s identifikačním číslem :jobId
GET	/jobs/:jobId/creator	vrátí zadavatele práce s identifikačním číslem :jobId
POST	/jobs/:jobId/assignee	přiřadí uživatele k plnění práce s identifikačním číslem :jobId
GET	/jobs/:jobId/assignee	vrátí přiřazeného uživatele k práci s identifikačním číslem :jobId
DELETE	/jobs/:jobId/assignee	odmítne plnění práce s identifikačním číslem :jobId
POST	/jobs/:jobId/fulfiller	potvrdí plnění práce s identifikačním číslem :jobId
GET	/jobs/:jobId/fulfiller	vrátí plnitele práce s identifikačním číslem :jobId
DELETE	/jobs/:jobId/fulfiller	vypoví plnění práce s identifikačním číslem :jobId
POST	/jobs/:jobId/subscribers	projeví zájem o práci s identifikačním číslem :jobId
GET	/jobs/:jobId/subscribers	vrátí seznam zájemců o práci s identifikačním číslem :jobId
DELETE	/jobs/:jobId/subscriber/:username	zruší projevení zájmu o práci s identifikačním číslem :jobId
GET	/jobs/:jobId/results	vrátí seznam identifikačních čísel výsledků práce s identifikačním číslem :jobId
POST	/jobs/:jobId/results/submit	odešle výsledky práce s identifikačním číslem :jobId
POST	/jobs/:jobId/results/videos	uloží video, které bylo nahráno jako výsledek práce s identifikačním číslem :jobId
POST	/jobs/:jobId/results/videos/:videoId	aktualizuje nahrané video s identifikačním číslem :videoId
GET	/jobs/:jobId/results/videos/:videoId	vrátí odkaz na stažení videa s identifikačním číslem :videoId

Bibliografie

1. *ČSÚ: Chytrý Telefon Používá 70 % čechů, Přibývá seniorů* [online]. 2020 [cit. 2022-03-26]. Dostupné z: https://www.mediaguru.cz/clanky/2020/03/csu-chytry-telefon-pouziva-70-cechu-pribyva-senioru/.
2. *Lighthouse* [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://web.dev/learn/#lighthouse.
3. *V česku Nejdynamičtěji Roste Tiktok, vrací SE i Snapchat* [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://www.mediaguru.cz/clanky/2021/06/v-cesku-nejdynamicteji-roste-tiktok-vraci-se-i-snapchat/.
4. *Browser market share worldwide* [online]. 2022 [cit. 2022-01-21]. Dostupné z: https://gs.statcounter.com/browser-market-share.
5. *Desktop vs mobile VS tablet market share worldwide* [online]. 2022 [cit. 2022-01-21]. Dostupné z: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet.
6. *Klient–server* [online]. Wikimedia Foundation, 2022 [cit. 2022-04-23]. Dostupné z: https://cs.wikipedia.org/wiki/Klient%E2%80%93server.
7. *Stack overflow developer survey 2021* [online]. 2021 [cit. 2022-04-23]. Dostupné z: https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language.
8. *JavaScript vs. node.js - javatpoint* [online]. 2021 [cit. 2022-04-23]. Dostupné z: https://www.javatpoint.com/javascript-vs-nodejs.
9. KVAPIL, Jiří. *Lekce 1 - úvod do typescriptu* [online]. 2018 [cit. 2022-04-23]. Dostupné z: https://www.itnetwork.cz/javascript/typescript/uvod-do-typescriptu.
10. *HTTPS* [online]. Wikimedia Foundation, 2022 [cit. 2022-04-23]. Dostupné z: https://cs.wikipedia.org/wiki/HTTPS.
11. *Objektově relační mapování* [online]. Wikimedia Foundation, 2021 [cit. 2022-04-23]. Dostupné z: https://cs.wikipedia.org/wiki/Objektov%C4%9B_rela%C4%8Dn%C3%AD_mapov%C3%A1n%C3%AD.
12. *How to pick the Best Video File Format — adobe* [online]. Adobe, 2020 [cit. 2022-04-23]. Dostupné z: https://www.adobe.com/ca/creativecloud/video/discover/best-video-format.html.
13. MÁCA, Jindřich. *Lekce 1 - úvod do react* [online]. itnetwork, 2019 [cit. 2022-04-23]. Dostupné z: https://www.itnetwork.cz/javascript/react/zaklady/uvod-do-react/.

Obsah přiloženého média

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