Jihočeská univerzita
v Českých Budějovicích

Přírodovědecká fakulta

Webový portál pracovních příležitostí pro studenty JU

Bakalářská práce

Libor Matásek

Vedoucí práce: Mgr. Geyer Jakub

Školitel: Ing. Monika Maříková, Ph.D.

České Budějovice 2024

Přírodovědecká fakulta
Faculty of Science

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: Matásek Libor
(jméno, příjmení, tituly)

Program studia / specializace: Aplikovaná informatika
specializace Web a multimédia

Pracoviště PřF JU: Katedra informatiky

Školitel: Mgr. Jakub Geyer
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, e-mail)

Garant z PřF JU:
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: Ing. Monika Maříková, Ph.D.
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, e-mail)

Téma bakalářské práce: Webový portál pracovních příležitostí pro studenty JU

Cíle práce:

Cílem práce je vytvoření webového portálu, který umožní firmám nabídnout pracovní příležitosti či stáže/praxe studentům/absolventům univerzity. Portál bude současně prezentovat zájem studentů o určité oblasti a zprostředkuje výměnu kontaktů. Portál bude napojen na univerzitní systémy IDM a STAG. Systém IDM bude sloužit pro přihlášení aktuálních studentů a zaměstnanců (vybraní zaměstnanci budou moci také vkládat nabídky či moderovat nabídky firem). Ze systému STAG budou načteny informace o studentovi, který bude moci vybrané údaje (např. studijní program, průměr, kvalifikační práce, apod.) prezentovat na svém profilu (veřejně či pouze firmám, o jejichž nabídky projeví zájem). Portál bude umožňovat notifikace studentů o nových nabídkách ve vybraných oblastech a notifikace firem o zájmu studenta o určitou pozici. Portál musí být přístupný jako webová aplikace, a to i pro mobilní zařízení (responzivní design). Součástí práce budou bezpodmínečně následující dokumenty:

- *Analýza funkčních a nefunkčních požadavků*
- *Analýza a výběr vhodných softwarových prostředků*
- *Dokumentace kódu, Testovací dokumentace, Uživatelská dokumentace*

Přírodovědecká fakulta
Faculty of Science

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Základní doporučená literatura:

Zvláštní poznámky pracoviště:

Financování práce:

Školitel práce ..... podpis: .....

U externích vedoucích fakultní garant práce ..... podpis: .....

Garant programu/specializace[1][2] ..... podpis: .....

Vedoucí pracoviště PřF JU, kde proběhne obhajoba ..... podpis: .....

Souhlas vedoucího ústavu AV nebo jiné instituce[3] ..... podpis: .....

V Českých Budějovicích dne 03. 1. 2022 Podpis studenta: .....

[1] v případě prací v programu bakalářském Biologie není podpis garanta programu vyžadován
[2] v případě magisterských prací v programech učitelství pro SŠ podpis proděkana pro učitelské obory
[3] v případě, že práce bude vypracovávána jinde než prostorách PřF, například na ústavu AV

Bibliografické údaje

Matásek, L., 2024: Webový portál pracovních příležitostí pro studenty JU [Web portal of job opportunities for JU students, Bc. Thesis, in Czech] – 85 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic

Anotace

Tato bakalářská práce se zabývá vytvořením nového webového portálu pro inzerci práce určeného studentům Jihočeské univerzity (JU) s cílem vylepšit stávající řešení provozované Kariérním centrem JU. Nový portál přináší několik významných vylepšení oproti současnému, mezi která patří: vyšší míra automatizace zavedením samoobsluhy pro inzerenty a studenty; pokročilejší vyhledávání pracovních nabídek; profily studentů a možnost přímo reagovat na inzerované nabídky; integraci se systémem STAG.

Annotation

This bachelor thesis deals with the creation of a new job advertisement web portal for students of the University of South Bohemia (JU) in order to improve the existing solution operated by the JU Career Centre. The new portal brings several significant improvements over the current solution, including: a higher level of automation by introducing self-service for advertisers and students; more advanced job search; student profiles and the ability to respond directly to advertised jobs; integration with the STAG system.

Prohlašuji, že jsem autorem této kvalifikační práce a že jsem ji vypracoval(a) pouze s použitím pramenů a literatury uvedených v seznamu použitých zdrojů.

V Českých Budějovicích dne 12.4.2024 _____

Libor Matásek

Poděkování

Rád bych poděkoval svému vedoucímu práce Mgr. Jakubovi Geyerovi za jeho čas, vstřícnost, trpělivost a odborné vedení. Dále chci poděkovat Kariérnímu centru Jihočeské univerzity za milou spolupráci, především pak své školitelce paní Ing. Monice Maříkové, Ph.D. Také chci poděkovat Ondřeji Doktorovi, Bc. DiS., za úvod k možnostem autentizace prostřednictvím univerzitních systémů. Nakonec, děkuji své rodině a blízkým za podporu.

Obsah

1 Úvod

1.1 Motivace

Motivací pro vytvoření karierního portálu byl především impuls a zájem Karierního centra Jihočeské univerzity o realizaci tohoto portálu. V analýze existujících řešení na vybraných univerzitách se ukázalo, že univerzity se téměř vždy do různé míry zabývají inzercí pracovních nabídek (kapitola 1.5.2). Provozování karierního portálu, prohlubování spolupráce mezi univerzitou a firmami, a propojování studentů s firmami je jednou z logických činností, kterou se organizace univerzit, jako je Kariérní centrum, zabývají, a činností, kterou si současný student pod organizací s tímto názvem představí. Provozováním karierního portálu může univerzita zlepšit své služby studentům, zvýšit svou prestiž, nabízet pozice pro své nové zaměstnance, navazovat novou spolupráci a posilovat stávající spolupráci s firmami a v neposlední řadě získávat finanční prostředky.

Portál může ovlivnit značné množství studentů, z průzkumu banky Equa bank vyplynulo, že při studiu pracuje 9 z 10 studentů ve věku od 16 do 26 let [1]. Z dotazníkového šetření provedeného v práci *How Does a University of Choice Come to Student's Mind? From the Aspect of the E-Servicescape of University's Website* vyplynul významný vliv internetových služeb na rozhodnutí uchazeče zvolit danou školu [2]. V konkurenčním souboji škol o studenty je proto důležité studentům poskytovat kvalitní služby. I když portál pracovních nabídek cílí spíše na stávající studenty a absolventy, než-li na uchazeče o studium, je jejich spokojenost s poskytovanými službami jistě neméně důležitá a mj. se může odrazit v jejich referování o univerzitě.

1.2 Cíl a úkoly práce

Cílem práce je vytvoření webového portálu, který umožní firmám nabídnout pracovní příležitosti či stáže/praxe studentům/absolventům univerzity. Portál bude současně prezentovat zájem studentů o určité oblasti a zprostředkuje výměnu kontaktů. Portál bude napojen na univerzitní systémy IDM a STAG. Systém IDM bude sloužit pro přihlášení aktuálních studentů a zaměstnanců (vybraní zaměstnanci budou moci také vkládat nabídky či moderovat nabídky firem). Ze systému STAG budou načteny informace o studentovi, který bude moci vybrané údaje (např. studijní program, průměr, kvalifikační

práce, apod.) prezentovat na svém profilu (veřejně, či pouze firmám, o jejichž nabídky projeví zájem). Portál bude umožňovat notifikace studentů o nových nabídkách ve vybraných oblastech a notifikace firem o zájmu studenta o určitou pozici. Portál musí být přístupný jako webová aplikace, a to i pro mobilní zařízení (responsivní design). Součástí práce budou bezpodmínečně i následující dokumenty: analýza funkčních a nefunkčních požadavků, analýza a výběr vhodných softwarových prostředků, dokumentace kódu, testovací dokumentace, uživatelská dokumentace.

Z cílů práce vycházejí následující úkoly:

- Analyzovat existující kariérní portály.
- Analyzovat funkční a nefunkční požadavky.
- Analyzovat a vybrat vhodné softwarové technologie.
- Implementovat řešení formou webové aplikace vhodné k používání i na mobilních zařízeních. Integrovat univerzitní systémy STAG a IDM.
- Implementovat automatizované testy.
- Provést uživatelské testování aplikace.
- Zdokumentovat důležité oblasti práce.
- Vytvořit uživatelskou a testovací dokumentaci.

Výstupem naplnění cílů bude webový portál, který bude v různých směrech, a především v míře automatizace, lepší než stávající řešení inzerce pracovních nabídek Karierního centra, které je popsáno v kapitole 1.4. Nové řešení bude oproti stávajícímu řešení nabízet především:

- Zvýšenou automatizaci implementací uživatelských účtů a samoobsluhy firem.
- Zvýšenou automatizaci implementací uživatelských účtů a samoobsluhy studentů.
- Lepší schopnost vyhledávat relevantní pracovní nabídky a porovnávat je mezi sebou.

- Komfortněji navázat komunikaci mezi studentem a firmou přímo prostřednictvím portálu.

1.3 Úvod do internetové inzerce práce

Webové portály inzerce práce jsou internetové stránky, které primárně zobrazují pracovní nabídky a propojují zájemce o práci s inzerenty. Nabídky mohou být na portálech inzerovány zdarma nebo za úplatu. Nabídnuty mohou být i další služby, jako například: zobrazování nabídky na předních pozicích; zvýraznění nabídky; propagace na sociálních sítích; úprava obsahu profesionálem; možnost přímo oslovit registrované zájemce.

Pro úspěšnost inzerce je kromě jejího obsahu důležitý i dosah, tedy aby se dostala k co možná největšímu množství lidí. To přirozeně zvyšuje šanci na úspěšnost inzerce. Internet poskytuje širokou základnu uživatelů a je nejrychleji rostoucím moderním médiem. V USA trvalo internetu pouze 4 roky získat 50 milionů uživatelů, což je krátká doba v porovnání s televizí, které to trvalo 13 let a rádiem s 38 lety [3].

Jedním ze zdrojů návštěvníků pro inzeráty práce jsou i sociální sítě, které představují aplikace budující komunity lidí. Nejpopulárnější sociální sítě disponují obrovským množstvím uživatelů, a proto je jejich využití pro inzerci logickým krokem. Sociální sítě nabízí různé úrovně služeb týkajících se inzerce práce: od možnosti vytvářet příspěvky s nabídkou, reklamní propagace, až po úplné služby inzertního portálu. Největší sociální sítí zaměřenou především na kariéru je LinkedIn, vlastněný společností Microsoft, který spojuje celosvětově přes miliardu lidí [4]. Obecněji zaměřená sociální síť Facebook dříve nabízela vlastní inzertní portál práce, který však časem zrušila. Pro případ sdílení a propagace vlastních webových stránek na sociálních sítích mohou být tyto stránky vybaveny speciálním kódem, který je určen k poskytnutí metadat síti, která mohou upravovat způsob zobrazení příspěvku na síti, včetně jeho textu, obrázku apod. [5].

Dalším zdrojem návštěvníků webových portálů jsou internetové vyhledávače. I pro ně lze vybavit stránku speciálním kódem, který upravuje zobrazení ve vyhledávání. Webové stránky bojují o co nejdřívější umístění ve výsledcích vyhledávání. Tato komplexní problematika se nazývá SEO (Search Engine Optimization) a může se lišit pro různé vyhledávače, protože každý z nich používá odlišný algoritmus indexace. V roce 2021 byl v Čechách nejpopulárnějším vyhledávačem Google s necelými 84 % podílem na trhu [6].

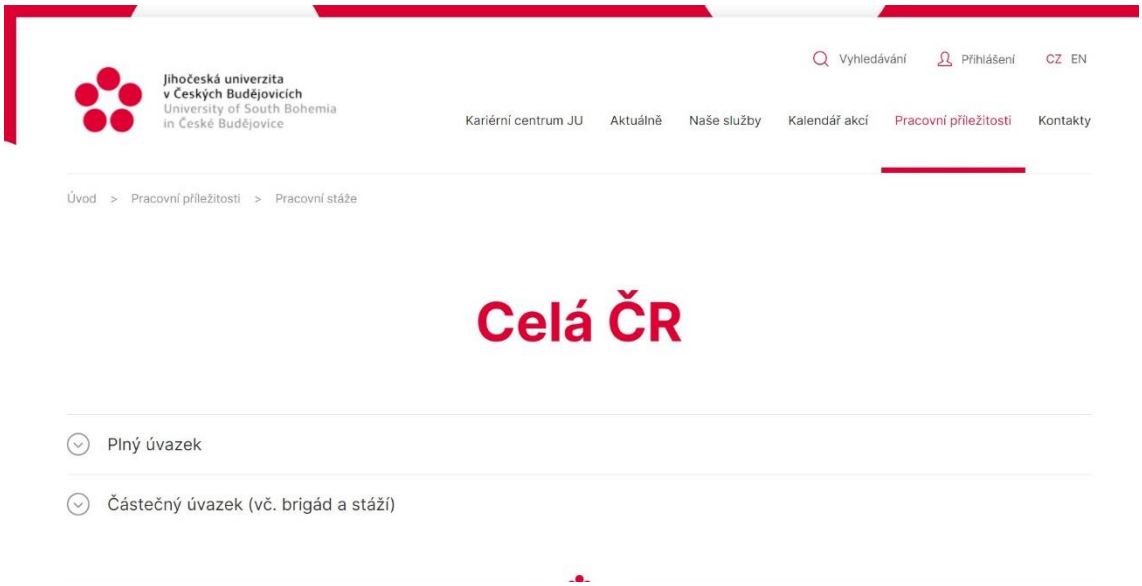
Úspěšnost umístění na předních pozicích ovlivňuje: relevantnost obsahu; důvěryhodnost webové stránky; reference z ostatních stránek; technická kvalita. Pro Google je obsah důležitější než technická kvalita, která však může rozhodnout u podobně relevantních stránkách, a jíž faktory jsou: přívětivost pro mobilní zařízení; bezpečnost (např. použití šifrovaného protokolu pro komunikaci); Core Web Vitals (Metriky měřící uživatelskou zkušenost, jako je rychlost načítání, vizuální stabilita a interaktivita.) [7]. Technická kvalita webové stránky nemá vliv pouze na úspěšnost pořadí ve vyhledávání, ale i na délku pobytu návštěvníků. „V BBC zjistili, že přijdou o 10 % uživatelů za každou další sekundu než se jejich stránka načte.“ [8]

1.4 Kariérní centrum Jihočeské Univerzity a současné řešení

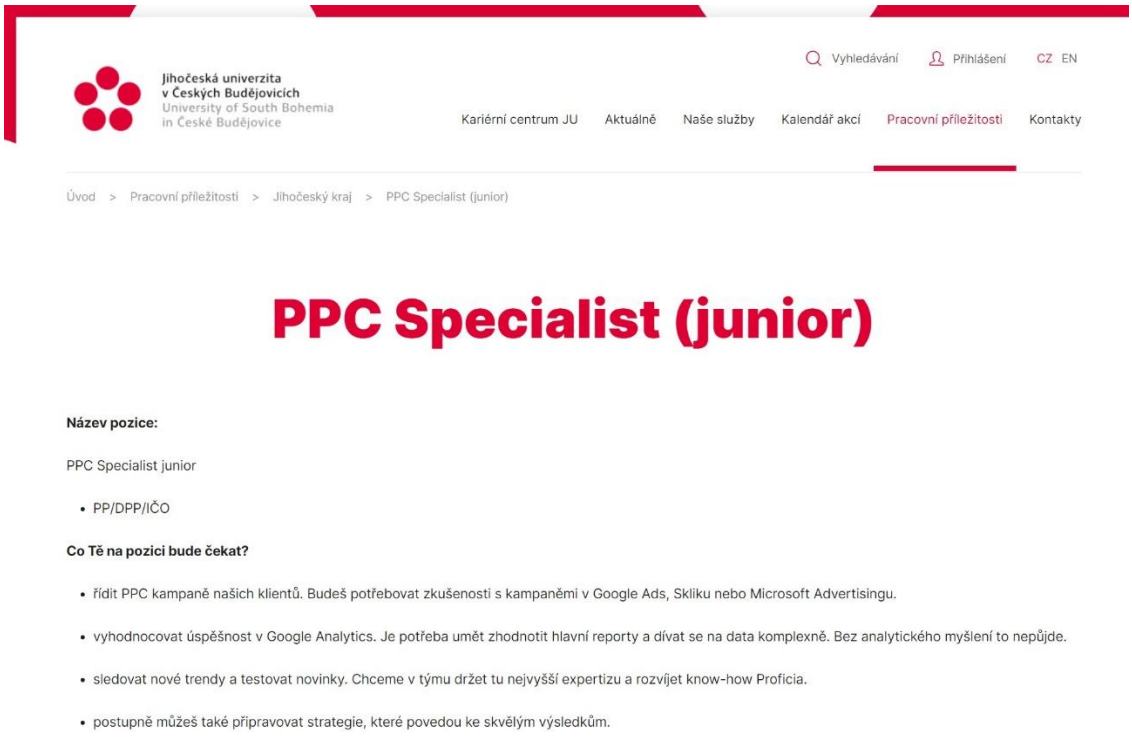
Tato práce je vypracována ve spolupráci s Kariérním centrem Jihočeské univerzity (dále jen „KC“). Jedná se o organizaci poskytující kariérní služby všem studentům Jihočeské univerzity, a to nezávisle na jejich příslušnosti k fakultě. KC vzniklo za účelem propojení akademického a pracovního prostředí. Mezi nabízené služby patří: pořádání workshopů a seminářů; poskytování kariérního poradenství v oblastech, jako je výběr vhodného zaměstnání, tvorba životopisů, motivačních dopisů či LinkedIn profilu; zprostředkování odborných stáží nebo možnosti vypracovat závěrečnou práci ve spolupráci se zaměstnavateli… [9]

Kariérní centrum provozuje své webové stránky na adrese https://www.kc.jcu.cz. Zde vystavuje i jednoduchou inzerci pracovních nabídek. Existující řešení však postrádá složitější funkcionalitu. Web disponuje minimem automatizace, neobsahuje samosprávu inzerentů ani studentů, nabízí pouze základní prezentační funkcionalitu a správu pouze ze strany členů kariérního centra. Možnost navigovat se v inzerci je základní, bez možnosti filtrovat podle více atributů či komfortně zobrazit důležité aspekty pozice. Nabídky jsou organizovány v proklikové hierarchii: *pracovní příležitosti* > *oblast (Jihočeský kraj)* > *úvazek (plný / částečný)* > *detail nabídky* (Obrázek 1). Web je realizován v jazyce PHP a používá CMS (redakční systém pro správu obsahu) Joomla.

Obrázek 1 Prokliková navigace



Obrázek 2 Detail nabídky



1.5 Analýza existujících řešení

Na internetu lze nalézt množství portálů zabývajících se inzercí práce, které provozují jak soukromé subjekty, tak i státní organizace. V rámci Jihočeské univerzity je možné se s inzercí pracovních nabídek setkat na různých místech a kanálech. Takovými místy jsou: hlavní web univerzity [10]; weby jednotlivých fakult [11]; weby kateder a ústavů [12];

web Karierního centra [13]; kariérní portály jako jobs.cz [14][15][16]; web Úřadu práce [17]; ale i místa fyzická, jako jsou nástěnky v prostorách vstupu do akademické knihovny.

Nevýhodou vystavování nabídek univerzitou u soukromých subjektů je nutnost platby za inzerci a boj o zobrazení v zástupu ostatních nabídek. Jedna konkrétní pracovní pozice může být vystavována i na několika pracovních portálech, čímž se náklady sčítají. Další nevýhodou vystavování inzerce na portálech komerčních subjektů je, že nelze počítat s implementací individuálních požadavků, napojením univerzitních systémů, či možným komerčním zhodnocením ve prospěch univerzity. Výhodou inzerce na komerčních portálech je jejich dosah díky širokým základnám uživatelů a zažité značce. Podobně jako má člověk zažitou oblíbenou aplikaci, například pro rezervaci hotelů Booking, a již nevyužívá jiné zdroje, může mít stejně zažitý i portál pro nabídky práce, například jobs.cz.

Kromě portálů zaměřujících se primárně na nabízení práce uchazečům o práci existují také portály, kde je primární zaměření opačným směrem, a sice jsou inzerováni uchazeči. Portály nabízející databázi uchazečů jsou například weby zivotopisy.cz a navolnenoze.cz.

Webovou nabídku práce provozuje i státní instituce Úřad Práce ČR [18]. Pro možnost inzerce práce na tomto portále je však nutné se zaregistrovat v interní databázi u místně příslušného kontaktního pracoviště Úřadu práce [19].

1.5.1 Analýza existujících řešení soukromých subjektů

Byla provedena analýza předních českých komerčních portálů dle portálu jenprace.cz [20]. Analýza byla zaměřena na použité technologie, nabízenou funkcionalitu a náklady na inzerci. Data o nákladech na inzerci byla získána z příslušných webů a ceníků. Pro srovnání byla vybrána nabídka odpovídající základní inzerci na 30 dnů, avšak příplatky mohou výslednou částku výrazně zvýšit. Portály si nechávaly připlatit za různorodé služby, přičemž příklady takových služeb již byly uvedeny dříve v kapitole 1.3. Ve srovnání nejsou zohledněny tarify, množstevní ani individuální slevy.

Tabulka 1 Ceny inzerce práce vybraných komerčních portálů [21][22][23][24][25][26][27][28]

Portál	Cena v Kč bez DPH
jobs.cz	Od 7 390 Kč za 1 inzerát na 30 dnů

prace.cz	Od 4 790 Kč za 1 inzerát na 30 dnů
jenprace.cz	Od 3 590 Kč za 1 inzerát na 30 dnů
startupjobs.cz	Od 6 749 Kč za 1 inzerát na 30 dnů
dobraprace.cz	Od 1 050 Kč za 1 inzerát na 30 dnů
easy-prace.cz	2 000 Kč za 1 inzerát na 40 dnů
hitprace.cz	zdarma
volnamista.cz	zdarma

Z Tabulka 1 je vidět, že se základní inzerce na 30 dnů pohybuje v částce do deseti tisíc Kč bez DPH na jeden inzerát. V případě neúspěšné inzerce v daném časovém období může být nutné vynaložit další náklady na prodloužení inzerce, často za zvýhodněnou cenu. V tabulce jsou i poskytovatelé nabízející inzerci zdarma. Tito poskytovatelé mohou vydělávat na zobrazované reklamě, nebo dokáží zhodnotit sbíraná data.

Představu o použitých technologiích lze získat různými nástroji, které dokáží některé technologie rozpoznat. Technologie mohou zanechávat své stopy v kódu nebo komunikaci (HTTP hlavičky). Některé technologie však žádné stopy nezanechávají, a nelze je tak detekovat. Byly použity nástroje BuiltWith a Wappalyzer, které se však neukázaly být stoprocentně spolehlivé. Například nedokázaly na některých stránkách rozpoznat frontendový framework React, který byl s určitostí použit, protože ho detekoval jeho oficiální Chrome plugin. Pro množství technologií použitých na jednotlivých webových stránkách a nemožnost spoléhat se na detekci byl konkrétní seznam pro každý web vynechán. Pro získání úplného seznamu použitých technologií by musel být celý projekt zveřejněn jako otevřený kód, nebo by musela být informace získána komunikací s ochotným provozovatelem. Dříve zmíněné nástroje pro detekci rozpoznaly z hlavních jazyků, frameworků a knihoven následující: PHP, Nette, Symfony, Java, TypeScript, JavaScript, React, Next.js, jQuery, Vue, Nuxt, Node.js, Bootstrap, Tailwind, MUI. Jak již bylo zmíněno, výčet je pouze orientační, všechny technologie rozhodně nemusely být rozpoznány.

1.5.2 Analýza existujících řešení univerzit

Byla provedena analýza pracovních portálů Českých univerzit na internetu. Do analýzy byly vybrány univerzity Jihočeského kraje, protože v něm působí i Jihočeská Univerzita. V Jihočeském kraji působí pouze pět univerzit (JČU, VŠTE, VŠERS, FAMO, VŠE [29]), bylo proto dále vybráno několik dalších autorovi známých univerzit.

Bylo zjišťováno jaké typy inzerce univerzita nabízí, především zda nabízí komplexní portál inzerce práce. Za takový portál byl považován web, který disponoval pokročilou funkcionalitou, jako je částečná automatizace, samoobsluha inzerentů a studentů, pokročilé vyhledávání inzerátů, možnost reagovat na nabídky z portálu. Za komplexní portál nabídek práce nebyl považován takový portál, kde k zadávání práce zjevně dochází prostřednictvím redakčního systému a realizuje tak minimální míru automatizace. Analýza byla také zaměřena na komerční modely komplexních portálů a zda jsou přístupné veřejnosti.

Tabulka 2 Analýza pracovních portálů univerzit [30; 31] [32; 33; 34][13; 10; 11][35; 36; 37; 38; 39][40; 41][42; 43][44; 45][46; 47][48][49; 50; 51][52]

Univerzita	Komplexní portál práce	Jednoduchý portál inzerce práce	Komplexní portál má veřejně přístupné nabídky	Komerční model	Poznámka
VŠTE	Nenalezeno	Ano		Nenalezeno	
VŠERS	Nenalezeno	Nenalezeno			
FAMO	Nenalezeno	Nenalezeno			
VŠE	Ano	Ano	Ne	Zpoplatněno, ceny nejsou veřejně vystaveny na internetu.	Nebyla nalezena stránka pro registraci inzerenta, pouze kontaktní email.
JČU	Nenalezeno	Ano			Jednoduchou inzerci je možné nalézt na hlavním webu, webu fakult i kariérním centru.
MUNI	Ano	Ano	Ne	Ano od 5 000 Kč bez DPH za 1 inzerát na 30 dnů	Inzerent získá přístup do portálu až po zaplacení služeb a schválení. Portál ukazuje, na kolik procent uchazeči vyhovují nabídce.
ČVUT kariérní centrum	Nenalezeno	Ano		Ano od 3.500 Kč / 3 měsíce	Inzerent objednává inzerci přes email odesláním na uvedený kontaktní email.

ČVUT FIT	Ano		Ano	Možnost inzerce nabídek práce je zahrnuta v programu partnerství od 30 000,- Kč bez DPH, nebo programu sponzorství 70 000,- Kč bez DPH.	Bez registrace studentů. Na inzeráty může reagovat i veřejnost. Využívá pro řazení nabídek algoritmus strojového učení.
Univerzita Karlova Kariérní centrum	Nenalezeno	Ano		Nenalezeno	Inzerent projevuje zájem o inzerci přes email odesláním na uvedený kontaktní email.
Univerzita Karlova Právnická fakulta	Částečně		Ano	Zdarma, možnost topování nabídky za 2 500 Kč s DPH.	Zadávání inzerce bez registrace inzerentů či studentů.
Univerzita Karlova Pedagogická fakulta	Nenalezeno	Ano, formou sociální sítě.			Odkazuje pro inzerci na Facebookovou skupinu. Ta mj. odkazuje na nefunkční web https://chciucit.pedf.cuni.cz.
ZČU v Plzni	Částečně	Ano	Ano	Ano, dle ceníku jobs.cz viz Tabulka 1	Má částečně komplexní portál. Umožňuje firmám po registraci zadávat praxe, semestrální práce a závěrečné práce. Avšak pracovní nabídky importuje z portálu jobs.cz, a to všechny ze Západočeského kraje.
UHK	Ano		Ano	Nenalezeno	Veřejně přístupná bez přihlášení je i odpověď na nabídku.

Provedená analýza existujících řešení inzerce práce na vybraných univerzitách v České republice ukázala značnou rozmanitost v přístupu k tomuto tématu. Lze shrnout, že:

- U univerzit v Jihočeském kraji s výjimkou VŠE, která ovšem působí především v Praze, nebyl nalezen žádný komplexní portál inzerce práce.
- V rámci celé České republiky je již situace odlišná: většina vybraných univerzit se zabývá komplexní či částečně komplexní inzercí práce, což může být způsobeno výběrem větších a známějších univerzit.
- Téměř všechny analyzované univerzity provozují na internetu inzerci práce, ať už pro své potenciální zaměstnance nebo studenty.
- Některé univerzity provozují komplexní portály práce, zatímco jiné se omezují pouze na jednoduchou inzerci.
- Komplexní a částečně komplexní portály práce provozované univerzitami bývají často komerčně zaměřené, existují ale i příklady méně komerčních modelů, jako například u Právnické fakulty Univerzity Karlovy, kde je inzerce zdarma, avšak s možností dokoupení prémiové služby.
- Některé komplexní portály byly veřejnosti nepřístupné.
- Inzerce napříč univerzitou a jejími fakultami bývá nekonzistentní: některé provozují komplexní portály, zatímco jiné pouze jednoduchou inzerci. To se ukázalo například u univerzit ČVUT a Univerzity Karlovy, kde určitá fakulta může disponovat komplexním portálem práce, zatímco univerzita jako celek, kariérní centrum nebo její ostatní fakulty podobný portál nemají, nebo inzerci řeší jinak.
- Kromě vlastních webových portálů se objevila i inzerce práce formou skupin na sociálních sítích.
- Jednoduché inzerce nabídek byly většinou realizovány prostřednictvím CMS (Wordpress, Joomla, Umbraco).

1.5.3 Analýza existujících programových řešení

Cílem této analýzy bylo najít hotové programové řešení pro kariérní portál, které by splňovalo zadání práce. Bylo však zřejmé, že nalézt řešení plně odpovídající zadání, bude obtížné – kvůli integraci univerzitního systému STAG. Bylo však stále možné najít řešení alespoň částečné, které by mohlo být použito a rozšířeno.

Hledání probíhalo na platformě Github s využitím klíčových slov "job list", "job board" a "job advertisement". Byla zohledněna pouze řešení s hodnocením minimálně 25 hvězdiček a aktualizovaná od roku 2020, aby byla zajištěna aktuálnost a bezpečnost nalezených řešení. Po ručním odfiltrování nerelevantních výsledků bylo nalezeno přibližně 15 repozitářů. [53][54]

Většina nalezených repozitářů nenabízela demo a když ano, tak často nefunkční. Proto bylo pro získání představy o funkcionalitě nutné procházet kód. Většina projektů se zaměřovala pouze na základní funkce, jako je zobrazování a vytváření inzerátů. Nebyl nalezen kód, který by obsahoval jak základní funkcionalitu, tak i profil inzerenta, správu uživatelů a řízení přístupu podporující více rolí a administraci. I když se některé repozitáře blížily požadované funkcionalitě, žádný z nich nebyl shledán jako vhodný pro použití v této práci. Často se rozsahem a kvalitou jednalo spíše o technologická dema než o fungující a otestovaná řešení vhodná pro produkční nasazení. Potřebnému řešení se nejvíce blížily repozitáře: tramcar/tramcar, Pythondeveloper6/django-job-board, Sany07/Job-Portal-Django, Automattic/WP-Job-Manager.

Kromě Githubu byly zkontrolovány i tržiště pluginů pro CMS Joomla (současné řešení inzerce KC) a CMS WordPress (nejčastější CMS na webu [55]).

Pro CMS Joomla bylo nalezeno více pluginů, od nejjednodušších po rozsáhlé. Požadavkům nejvíce odpovídal plugin JSJobs s rozsáhlou funkcionalitou, která se přibližovala funkcionalitě v této práci. Plugin má také vystavené funkční demo na internetu [50]. JSJobs nabízí verzi zdarma a placenou verzi „Pro“. Mezi přibližně stovkou recenzí lze nalézt jednotky až desítky negativních. Kvalita projektu je diskutabilní, verze zdarma ke stažení neobsahuje žádné testy. Soubor s modelem inzerátu *job.php* obsahuje 3410 řádků a mnoho duplikací. Například metoda získání dat o práci podle identifikátoru *getJobById* z tohoto souboru je dlouhá 121 řádků. Jeden z recenzentů zmiňuje, že placená verze je, na rozdíl od verze neplacené, formou uzavřeného kódu, což je pro tuto práci nežádoucí vlastnost. [56] [57] [58]

Pro CMS WordPress bylo také nalezeno několik pluginů v různém rozsahu funkcionality. Z nich požadované funkcionalitě opět nejvíce vyhovoval plugin JSJobs, avšak tentokrát ve verzi pro WordPress. [59]

Autor se v této práci rozhodl nepoužít již hotové programové řešení a to z několika důvodů:

- Reference: Žádný komplexní kariérní portál z analyzovaných existujících řešení soukromých subjektů (kapitola 1.5.1) a veřejných institucí (kapitola 1.5.2) nevykazoval známky použití již hotového programového řešení, což však nelze vyloučit vzhledem k nedostupnosti otevřených kódů.

- Bezpečnost: Nepopulární řešení od nedůvěryhodných autorů by bylo nutné podrobit důkladné analýze z pohledu bezpečnosti a funkčnosti. Nalezená řešení ve výhodném rozsahu se vyskytovala především formou pluginů do známých CMS, avšak např. WordPress je svou populárností lákavým cílem útočníků.

- Rozsah a kvalita: Analyzovaná existující řešení nepokrývala zadání projektu v plném rozsahu. Mnohdy se jednalo pouze o technologická dema. Kvalita kódu byla často diskutabilní.

- Redundance – Existující programová řešení, včetně CMS a pluginů, aby postihla co nejvíce případů užití, obsahují množství nepotřebného kódu a zbytečné konfigurace. Minimalizace množství kódu a složitosti jsou jedny z nejdůležitějších aspektů vývoje a údržby SW.

2 Implementace

2.1 Analýza požadavků

Velká část požadavků přirozeně vyplývá z tématu práce. Požadavky dále vycházejí ze zadání práce a ze schůzek s KC a vedoucím práce.

2.1.1 Funkční požadavky

- Výpis a hledání inzerátů: Veřejně přístupný stránkovaný výpis aktivních inzerátů včetně možného filtrování výsledků.

- Detail inzerátu: Veřejně přístupná stránka zobrazující informace o nabídce včetně odkazu na detail inzerenta. Administrátorovi a autorovi nabídky je zobrazeno rozšířené UI o proveditelné akce správy. Studentovi je umožněno odpovědět na nabídku.

- Odpověď na nabídku: Student má na detailu inzerátu možnost odpovědět na nabídku odesláním formuláře. Formulář umožňuje připojit více příloh např. CV nebo certifikát. Po odeslání formuláře je odeslán email na emailovou adresu inzerenta, kterou zadal při vytvoření inzerátu a kopií studentovi. Tento email má jako adresu pro odpověď nastavenou osobní emailovou adresu studenta, aby mohl inzerent studenta kontaktovat zpět. Je použita osobní emailová adresa studenta, protože student může časem přijít o přístup ke svému univerzitnímu emailu vlivem ukončení studia. Student se může ve formuláři rozhodnout udělit inzerentovi přístup na svůj profil po dobu 60 dnů, který je v takovém případě odkazem připojen do těla emailu.

- Statistika: Administrátor má možnost zobrazit počet odpovědí na nabídku a počet zobrazení stránky nabídky.

- Detail inzerenta: Veřejně přístupná stránka zobrazující informace o inzerentovi, včetně výpisu jeho aktivních nabídek. Administrátorovi a danému inzerentovi je zobrazeno rozšířené UI o proveditelné akce správy.

- Úprava profilu inzerenta: Inzerent může upravit svůj profil včetně nahrání loga, a tím i upravit data zobrazená na stránce detailu inzerenta či jinde. Administrátor má možnost upravit profily všech inzerentů.

- Výpis a hledání inzerentů: Veřejně přístupný stránkovaný výpis inzerentů.

- Registrace inzerenta: Veřejně přístupná stránka pro vytvoření účtu inzerenta.

- Ověření inzerenta: Pro dokončení registrace inzerenta je po odeslání registračního formuláře odeslán email s kódovaným podepsaným odkazem na zadanou adresu, jehož znalostí a navštívením inzerent prokáže vlastnictví emailové adresy. Tím je

znemožněno blokovat cizí emailové adresy, které jsou jinak v rámci účtů unikátní. Navíc je ztíženo případné vytváření zbytečných/škodlivých účtů.

- Vytvoření inzerenta administrátorem: Administrátor má možnost vytvořit inzerenta, např. po vzájemné komunikaci s ním. V takovém případě není ověřována emailová adresa inzerenta.

- Změna hesla uživatele: Uživatel má možnost si změnit své heslo. Administrátor má možnost změnit heslo jakéhokoli uživatele. To je vhodné například v případě, že byl účet založen právě administrátorem, a ten chce později předat přístup opravdovému uživateli.

- Změna emailu uživatele: Uživatel má možnost změnit si svůj email, v takovém případě následuje ověření vlastnictví emailu. Administrátor má možnost změnit email jakéhokoli uživatele, v tomto případě není ověřována nová emailová adresa uživatele. To je vhodné například v případě, že byl účet založen právě administrátorem, a ten chce později předat přístup opravdovému uživateli.

- Přihlášení emailem a heslem: Všem uživatelům, inzerentům, studentům a administrátorům je umožněno se přihlásit emailem a heslem.

- Vytvoření a úprava inzerátu: Inzerenti a Administrátoři mají možnost vytvářet a upravovat inzeráty. Administrátor má možnost kromě své univerzity vytvářet a upravovat i inzeráty u jakéhokoli jiného inzerenta, např. na základě vzájemné komunikace s ním. Po vytvoření či úpravě inzerentem přejde inzerát do stavu „čekající na schválení“, nezobrazuje se ve výpisu, ale vyžaduje nejprve schválení administrátorem. Administrátoři jsou upozorněni emailem na potřebu schválit novou či upravenou nabídku. Nabídka je automaticky schválena, pokud je autorem změny administrátor, a v takovém případě není ani odeslán notifikační email o nutnosti schválení.

- Schválení a zamítnutí inzerátu: Administrátor má možnost schvalovat jednotlivé inzeráty, nebo je případně odmítnout s odůvodněním, které je u nabídky uživateli zobrazeno. V obou případech je inzerentovi odeslán informativní email.

- Archivace inzerátu: Inzerent má možnost archivovat své inzeráty. Administrátor má možnost archivovat inzeráty všech inzerentů. Nabídky jsou, kromě ruční archivace, archivovány automaticky systémem po vypršení jejich platnosti, kterou lze nastavit maximálně na 90 dnů od vytvoření nebo úpravy inzerátu, a to z toho důvodu, aby nezůstávaly na portálu zapomenuté nabídky. Možnost archivovat existuje, aby uživatel nepřišel o pracně vyplněný obsah. Nabídku je z archivu možné obnovit její úpravou, čímž dojde k opětovnému odeslání na schválení, případně přepsáním dat do nového inzerátu.

- Odstranění inzerátu: Inzerent a administrátor mají možnost odstranit inzerát.

- Registrace studenta: Student má možnost vytvoření účtu prostřednictvím univerzitního systému STAG. Tím je ověřeno, že je dotyčný opravdu student. V rámci registrace je po studentovi vyžadován osobní email, z toho důvodu, aby měl student přístup do svého účtu i po ukončení studia, nebo výpadku STAG. Registrace je dokončena po ověření vlastnictví emailové adresy.

- Přihlášení prostřednictvím STAG: Studeni a administrátoři disponující univerzitním účtem mají možnost se kromě hesla přihlásit i prostřednictvím STAG, a využít tak SingleSignOn funkcionalitu.

- Úprava profilu studenta, CRUD studia a kvalifikačních prací: Student má možnost vyplnit si svůj profil. Má možnost ručně vyplnit své studium na škole a kvalifikační práce. Ruční vyplnění je zde pro případ ztráty přístupu do STAG po ukončení studia, nebo pro možnost úplného vyplnění profilu při studiu na jiných univerzitách.

- Import dat ze STAG: Student má možnost importovat si na svůj profil data ze STAG. Studia, která absolvoval, volitelně včetně známek a kvalifikační práce. Importovaná data jsou označená jako důvěryhodná. Je umožněna i úprava importovaných dat, v takovém případě jsou viditelně označena jako upravená studentem.

- Oznámení studentům o nových nabídkách: Student si může zapnout emailové oznámení na nové nabídky, nastavit frekvenci denně/týdně/měsíčně a omezit nabídky dle filtru, aby dostával upozornění pouze na nabídky, které ho zajímají.

- Odhlášení uživatele: Uživatelé se mohou tlačítkem odhlásit.

- Obnova zapomenutého hesla: Uživatelé si mohou obnovit heslo zadáním svého emailu. Na email přijde podepsaný, šifrovaný, unikátní URL odkaz pro nastavení nového hesla.

- Odstranění účtu uživatele: Uživatel má možnost odstranit svůj účet po zadání ověření záměru ve formuláři. V takovém případě jsou odstraněny všechny související záznamy a soubory (logo). Administrátor má možnost odstranit účet jakéhokoli uživatele.

2.1.2 Nefunkční požadavky

- Bezpečnost: Bude vyvinuto úsilí k zabezpečení aplikace. Použitá bezpečnostní opatření shrnuje kapitola **Chyba! Nenalezen zdroj odkazů.**.

- Vzhled a ovládání: Aplikace bude vzhledově působit jako jedna ze stránek Jihočeské univerzity, čímž bude působit známě, důvěryhodně a jednotně. Ovládání bude snadné a intuitivní. Naplnění požadavku bylo ověřeno v kapitole 2.20.1.

- Kompatibilita zařízení: Aplikace bude naprogramována jako webová aplikace, čímž bude dostupná na celé řadě různých zařízení. Web bude poskytovat podobný uživatelský zážitek na různých zařízeních a různých velikostech zobrazení. Naplnění požadavku bylo ověřeno v kapitole 2.20.1.

- Dokumentace: K aplikaci bude vytvořena uživatelská dokumentace. Web rozsahu a složitosti této práce by měl být dostatečně intuitivní, aby běžní uživatelé v roli inzerenta a studenta nemuseli tuto dokumentaci použít. Dokumentace však bude výhodná pro přehledné shrnutí funkcionalit portálu a může být prospěšná zejména pro administrátory, kterým může sloužit jako návod pro správu portálu.

- Snadná údržba a rozšiřitelnost: Kód aplikace by neměl být složitý a měl by být snadno rozšiřitelný. Naplnění tohoto požadavku by měla zajistit volba vhodné architektury a technologií (kapitola 2.3), použité zásady vývoje (kapitola 2.2), a

také se ho dotkla kapitola „2.3.8 Slepá větev vývoje“ popisující přepsání a redukci složitého kódu na menší a jednodušší.

2.2 Zásady vývoje a kvalita kódu

Při vývoji software je vhodné dodržovat dobře známé zásady vývoje, které zaručí určitou minimální kvalitu výstupu a do budoucna přijatelné náklady na údržbu, úpravy a rozšíření. Z hlediska kvality kódu je žádoucí, aby kód splňoval následující vlastnosti:

- Zjistitelnost: Hledání kódu souvisejícího s určitou funkcionalitou aplikace by mělo být co nejsnazší.
- Čitelnost: Kód by měl být dobře čitelný, a to i pro nově příchozí vývojáře.
- Soudržnost: Kód který se sebou souvisí by se měl nacházet pospolu.
- Nízkou komplexitu: Kód by měl být co možná nejméně složitý a v minimálním možném množství.
- Konzistence: Kód by měl být konzistentní a tím i očekávatelný.
- Snadnou udržitelnost: Kód by měl být snadný na změnu a údržbu.
- Malá kognitivní nálož: Pro dokončení vývojového úkolu by mělo být potřeba udržet v hlavě malé množství věcí naráz.
- Bezpečnost: Kód by měl být bezpečný.

K dosažení těchto vlastností byly vybrány následující zásady, které byly do určité míry naplňovány:

- KISS (Keep It Simple Stupid): Snaha udržet komplexitu minimální, nekomplikovat zbytečně kód.
- DRY (Don’t Repeat Yourself): Snaha neduplikovat kód. Po duplikaci vzniká potřeba při případných úpravách upravovat kód na více místech. Při přílišné duplikaci je možné dostat projekt do stavu, kdy ho nadále nelze v rozumném čase udržovat. Duplikaci se lze vyhnout např. zavedením vhodné abstrakce.

- YAGNI (You Aren't Gonna Need It): Neprogramovat nepotřebnou funkcionalitu, nezobecňovat kód pro všechny potencionální případy, nebo nezavádět zbytečně abstrakce dokud to není vhodné.

- SOLID: Je soubor pěti zásad, v této práci bylo zohledněna především první a sice Single responsibility principle, která říká, že každá třída nebo funkce by měla mít pouze jednu zodpovědnost, jeden důvod ke změně, nebo-li dělat jen jednu věc a dělat ji dobře.

- Vyhnutí se předčasné optimalizaci: Nesnažit se veškerý kód ihned optimalizovat. Přehnaná optimalizace stojí čas, zanáší do kódu komplexitu a nemusí být nakonec ani potřeba.

- Eliminovat „mrtvý" kód: Neponechávat v projektu kód, který se nepoužívá a je proto tzv. „mrtvý". Takový kód bývá ponecháván pro případ, že by se v budoucnu mohl hodit. Mrtvý kód mate vývojáře, protože na první pohled není zřejmé, zda se kód (ne)používá. Navíc při vývoji překáží, protože se objevuje ve vyhledávání a snižuje tak zjistitelnost.

- Před komentáři upřednostnit dobře čitelný kód: Dobře čitelný kód je dokumentací sám o sobě. Zahrnuje správně zvolené názvy, nízkou komplexitu, krátké funkční bloky a další vlastnosti. Na rozdíl od komentářů netrpí desynchronizací. Komentáře časem nemusí být aktuální vzhledem ke kódu, ke kterému se vztahují, např. v důsledku úprav a následné neaktualizaci komentáře.

2.3 Hlavní architektura a výběr technologií

Architektura a výběr technologií je nedílnou součástí každého SW. Zahrnuje důležitá rozhodnutí, která může být velmi těžké a nákladné později měnit. Rozhodnutí, která ovlivňují celý řetěz následujících rozhodnutí. Termín vznikl v dobách, kdy byl vývoj velmi rigidní a podobný tomu ve stavitelství, kde bylo potřeba učinit množství rozhodnutí před samotnou konstrukcí, neboť později by bylo nákladné překopávat základy či měnit materiál zdí. Jednotlivé architektonické volby mají své klady a zápory, a je mezi nimi hledán přijatelný kompromis. Návrhu architektury může být vyčleněna i samostatná pracovní pozice – SW architekt. Takový člověk by měl být zkušeným expertem, mít

širokou ale i hlubokou znalost ve svém oboru. Architektura se formuje především na základě požadavků. Některé požadavky mohou klást velký vliv na architektonická rozhodnutí, zatímco jiné minimální. [60]

Dopady výběru nevhodné architektury a technologií demonstruje kapitola 2.3.8, protože se této práci v původní implementaci nevyhnuly.

Technologie je možné vybírat na základě různých kritérií. Jedním z nejdostupnějších kritérií pro porovnání dvou technologií řešící podobný problém je jejich popularita. Otevřený kód bývá hostovaný na portálech jako je GitHub. Ten nabízí možnost ocenit projekt udělením hvězdičky. Populární technologie mívají stovky, spíše však tisíce a více hvězdiček. Na GitHubu je zobrazen i počet vývojářů podílejících se na kódu a lze vyčíst aktuálnost projektu z datumů posledních změn. Lze očekávat, že vyšší popularita signalizuje, že je daná technologie dobrá, a proto si získala oblibu. Větší komunita okolo dané technologie také poskytuje určitý dohled nad bezpečností, hlásí a opravuje chyby, tvoří dokumentaci, doplňky a rozšíření, atd. Ostatní kritéria, která by bylo možné hodnotit, jako je složitost, učící křivka, rychlost vývoje, DX, vhodnost pro různé typy projektů a podobně se již porovnávají obtížně, protože nebývají dostupná. Případně jsou založena na názorech různých autorů. Dalším vhodným ukazatelem mohou být existující řešení, avšak ty mohou mít uzavřený kód, nebo disponovat odlišnými zdroji pro vývoj.

2.3.1 Základy aplikace

Na počátku projektu se bylo potřeba rozhodnout, zda rozšířit o novou funkcionalitu současné řešení, nebo budovat projekt nový tzv. na „zelené louce“. Obě možnosti mají svá pro a proti.

Následující výhody byly identifikovány pro možnost budování nového projektu:

- Možnost volit libovolné technologie.
- Žádná omezení plynoucí z míry (ne)rozšiřitelnosti stávající aplikace a jejího datového modelu.
- Žádné výkonnostní omezení existujícího řešení.

- Separátní životní cyklus aplikací. Pokud by došlo k problému v inzerci práce, stále by byl funkční prezentační web karierního centra a naopak.
- Kód portálu by byl menší o existující kód současného řešení.

Následující výhody byly identifikovány pro možnost rozšířit stávající řešení:

- Existující infrastruktura – nebylo by nutné shánět a zprovoznit novou.
- Vše týkající se kariéry by bylo na jednom místě a na jedné doméně.
- Mohou být již vyřešeny různé oblasti vývoje webu jako je vývojové prostředí, tvoření a zpracování formulářů, systém pro řízení přístupu...

Při analýze existujících řešení univerzit bylo pozorováno, že většina komplexních karierních portálů jsou samostatné projekty na vlastní doméně. Byly zváženy pro a proti obou přístupů a i s přihlédnutím k referenčním řešením bylo rozhodnuto o stavbě nové aplikace.

2.3.2 Struktura

Jedním z prvních architektonických rozhodnutí je zvolení členění projektu. Ke kódu je možné se chovat jako k celku, nebo ho rozdělit na menší prvky. Ustálenými modely, které se zabývají touto problematikou jsou model Monolith a Microservices.

Monolith je organizace aplikace do jednoho rigidním celku, který se obvykle nasazuje jako celek a všechen související kód je tradičně napsán ve stejném jazyce a využívá stejných technologií. Jde o nejjednodušší možnou architekturu, která je vhodná pro menší a nekomplexní projekty. S rostoucí velikostí projektu se mohou objevit nevýhody jako nízká produktivita, výkonnostní problémy s editory kódu, těžší orientace, vyšší chybovost a chyby ovlivňující zdánlivě nesouvisející část projektu. U monolithu se škáluje celek, což vyžaduje zbytečně více zdrojů než by bylo nutné, protože obvykle je potřeba škálovat jen nejvyužívanější části aplikace. [60]

Tyto nevýhody řeší Microservices architektura, avšak má své vlastní nevýhody. V tomto modelu je kód rozdělen do menších samostatně nasaditelných částí – služeb, kde každá může využívat různé technologie a architekturu. Různé funkcionality však mohou

vyžadovat spolupráci více služeb, což přináší složitost v nutnosti řešit komunikaci mezi službami. Je potřeba ošetřit případy selhání komunikace a mechanismy vzpamatování se z chybového stavu. Další komplexita spočívá v eventuální konzistenci dat způsobené ať už komunikací mezi službami, nebo nemožností obalit vykonání scénáře v rámci více služeb do transakce. Je těžší udržet konzistenci kódu napříč různými službami a je nutné řešit konfiguraci a vývojové prostředí pro každou službu zvlášť. [60]

Z důvodu jednoduchosti byl zvolen model Monolith. Projekt je tak konzistentní, nároky na znalost vývojářů jsou nižší, nasazování je jednodušší.

Obrázek 3 Významné položky a popis souborové struktury projektu

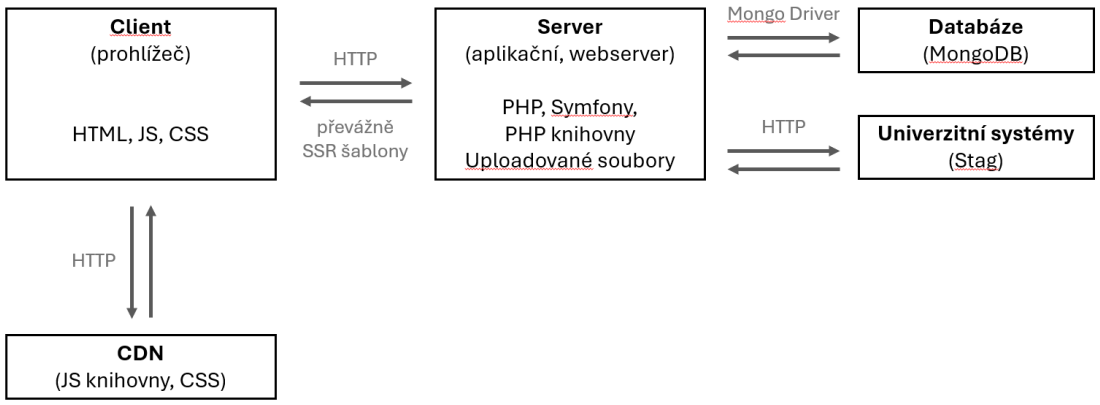
```
├── bin  # obsahuje CLI program Symfony
├── config  # obsahuje konfigurační soubory Symfony
├── db # obsahuje DB příkazy pro vytvoření účtu inzerenta JU a       .
.       přidání administrátora
├── docker # konfigurační soubory PHP a webserveru
├── docs
├── e2e  # obsahuje automatizované end-to(2)-end testy
├── public  # obsahuje veřejné soubory přístupné přes URL adresu/
│   ├── bundles  # obsahuje veřejně přístupné soubory knihoven Symfony
│   ├── upload  # obsahuje uživateli uploadované soubory
│   └── static  # obsahuje např. ikonky webu
├── src  # obsahuje PHP zdrojový kód/
│   ├── Command  # obsahuje CLI příkazy
│   ├── Controller  # obsahuje controllery (obsluhu jednotlivých URL)
│   ├── Core  # obsahuje základní kód sdílený napříč kódem
│   ├── Document  # obsahuje datové modely
│   ├── EventListener  # obsahuje obsluhy událostí
│   ├── Form  # obsahuje formuláře, jejich rozšíření a formulářové
.             prvky
│   ├── Repository  # obsahuje repozitáře obsahující metody pro práci
.                     s databází a ukládání datových modelů
│   ├── Security  # obsahuje kód týkající se bezpečnosti, Votery apod.
│   ├── Uploader  # obsahuje kód týkající se uploadu
│   └── Validator  # obsahuje kód týkající se validace
├── templates  # obsahuje šablony Twig (frontend aplikace)
├── tests  # obsahuje php testy
└── vendor  # obsahuje nainstalované PHP knihovny
```

2.3.3 Architektura komunikace

V této práci byla použita pro web tradiční architektura Client-Server. Aplikační server obsahuje většinu kódu, logiky a složitosti. Je použita metoda generování frontendového

kódu na serveru, který je distribuován do prohlížeče clienta protokolem HTTP. Frontendový kód byl psán jednoduchý a obsahuje minimum logiky a JavaScriptu. Po doručení frontendového kódu na clienta se mezi serverem a clientem ve vetšině případů již žádná komunikace neodehrává. Komunikace mezi clientem, serverem, a ostatními částmi řešení včetně hlavních technologií je znázorněna na Obrázek 4.

Obrázek 4 Schéma komunikace, prvků a hlavních technologií portálu



2.3.4 Výběr technologií frontendu

Pro frontend je možné vybrat jen takové jazyky a technologie, se kterými dokážou webové prohlížeče pracovat. Tradiční webová stránka je tvořená trojicí jazyků HTML, CSS, JavaScript. HTML je značkovací jazyk, který definuje strukturu a obsah dokumentu. CSS je jazyk sloužící k ovlivnění vzhledu dokumentu. JavaScript je programovací jazyk, který přidává dynamičnost, interaktivitu a dokáže modifikovat jak strukturu dokumentu (HTML), tak i jeho vzhled (CSS). Co se interaktivity a logiky týká, prohlížeč dokáže pracovat i s dalšími jazyky prostřednictvím technologie WebAssembly, ale i tento kód je nakonec načítán a spouštěn skrze JavaScript [61]. Je možné volit různé nástavby zmíněných jazyků, které jsou však nakonec nutně transformovány do zde uvedených nativních jazyků. V této práci byly použity přímo nativní jazyky:

- HTML
- CSS
- JavaScript

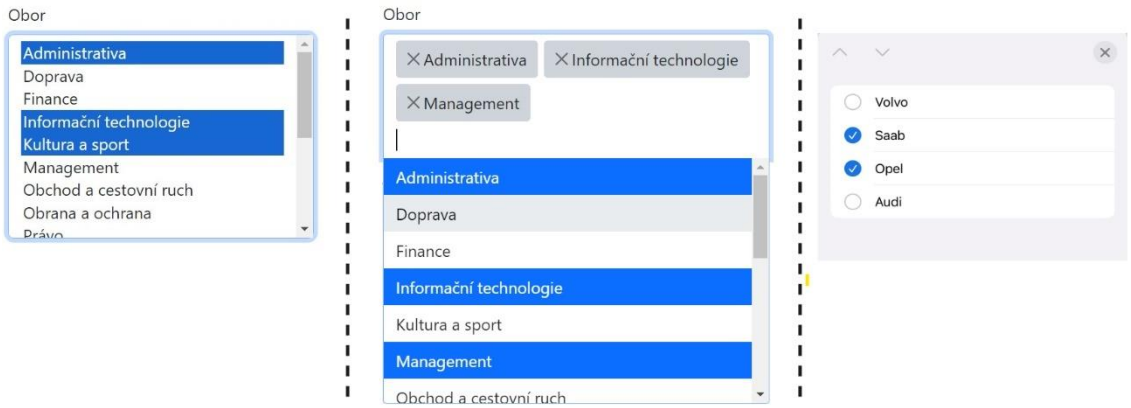
Tyto jazyky však nabízejí velký výběr různých knihoven a frameworků. Jak již bylo vidět na Obrázek 4, knihovny jsou v této práci načítány z CDN (Content Delivery Network), což je nejjednodušší způsob „instalace“. Pokud identickou adresu CDN používá i jiný web, nemusí uživatel knihovnu stahovat a prohlížeč mu ji může poskytnout z cache. Při načítání knihoven byl využit atribut *integrity* umožňující kontrolovat, že nedošlo k nežádoucí změně obsahu souboru na dané URL.

V této práci byly použity tyto významnější knihovny:

- Bootstrap: knihovna obsahující různorodou funkcionalitu a předpřipravené komponenty včetně podpory interakce prostřednictvím JavaScriptu.
- Select2: uživatelsky přívětivé formulářové výběry z nabídky hodnot (<*select*> elementy).
- Select2-bootstrap-5-theme: adaptování vzhledu Select2 výběrů do vzhledu knihovny Bootstrap.
- Jquery: knihovna obsahující různorodou JavaScript funkcionalitu. Vyžadováno knihovnou Select2. V dřívějších verzích vyžadováno i knihovnou Bootstrap.

Zajímavá je instalace knihovny Select2. Zatímco jsou prohlížeče schopny obstojně zobrazit i komplexnější typy vstupů, jako je například prvek pro výběr datumu, tak jeden z nejběžnějších formulářových prvků, a sice výběr z více hodnot, není příliš vhodně zobrazen, jak ukazuje Obrázek 5 (alespoň v prohlížeči Chrome na OS Windows). Nativní výběr více hodnot vyžaduje znalost uživatele, že musí držet klávesu CTRL, navíc není zřejmé, že lze vybrat více hodnot. Autor si nevzpomíná, kde by se naposledy setkal s nativním zobrazením tohoto prvku. Takto běžný prvek by mohl být již v prohlížečích řešen lépe.

Obrázek 5 Vlevo nativní zobrazení výběru více hodnot v prohlížeči Chrome na Windows. Uprostřed zobrazení s knihovnou Select2. Vpravo nativní zobrazení na mobilu s IOS.



2.3.5 Výběr technologií backendu

U technologií backendu existuje množství kombinací výběru technologií, od samotného HW přes výběr OS až po výběr jazyka, frameworků a knihoven.

Mnoho funkcí se napříč webovými projekty stále opakuje. Jedná se o funkcionalitu jako je tvorba a zpracování formulářů, validace, routování, autentizace, řízení přístupu, upload souborů, datové modelování, atd. Z tohoto důvodu vznikly frameworky, které nabízí ucelená řešení. Mohou mít za sebou mnohaletý vývoj a tisíce přispěvovatelů kódu [62]. Jsou tak solidním základem, na kterém lze stavět. Budování vlastního frameworku by zabralo velké množství času, zejména pak v rozsahu a kvalitě populárních frameworků. Typicky existuje minimálně jeden framework pro každý jazyk používaný k webovému vývoji.

Důležitější, než vybrat vhodný jazyk, proto může být výběr vhodného frameworku, který bude podporovat veškerou potřebnou funkcionalitu. Pro tuto práci byly zvažovány frameworky z Tabulka 3.

Tabulka 3 Popularita frameworků a jejich jazyků v průzkumu na webu StackOverflow v roce 2022, převzato z [63]

Framework	Popularita frameworku [%]	Popularita jazyka [%]
Symfony (PHP)	3.9	20.8
Laravel (PHP)	10.3	20.8
ASP.NET Core (C#)	20.7	27.9
Django (Python)	13.5	48.0
Ruby On Rails (Ruby)	6.3	6.0

Express (JavaScript)	22.9	65.3

Z možných frameworků byl vybrán framework Symfony. Sice je jeho popularita v průzkumu StackOverflow nižší, avšak jazyk PHP je stále populární. Tento framework byl vybrán především pro dlouholetou zkušenost autora s tímto frameworkem, a to včetně produkčně nasazených aplikací. Autor si byl jistý, že framework nabízí vše potřebné pro realizaci projektu. Navíc Symfony již dobře znal, což bylo klíčové po neúspěchu s původním řešením (kapitola 2.3.8).

Byly zvažovány i ostatní frameworky. ASP.NET nebylo vybráno pro nutnost kompilace, což bylo považováno za zpomalující a svazující faktor při vývoji. Z tabulky nejpopulárnější framework Express nebyl vybrán, protože obsahuje minimum funkcionality oproti ostatním frameworkům. Ruby On Rails pohánějící známé weby jako je GitHub nebyl vybrán pro nízkou popularitu jazyka Ruby.

Funkce frameworku byly doplněny o některé další funkce instalací knihoven. Ty byly instalovány za použití manažera závislostí Composer. Seznam významnějších použitých knihoven:

- Doctrine/doctrine-bundle, doctrine/mongodb-odm-bundle: knihovna pro práci s databází, mapování objektů apod.
- Vich/uploader-bundle: upload souborů.
- Symfonycasts/verify-email-bundle: generování a validace digitálně podepsaných URL.
- Stof/doctrine-extensions-bundle: rozšíření funkcionality Doctrine např. o automatické aktualizování pole *updatedAt* dokumentu při uložení.
- Babdev/pagerfanta-bundle, pagerfanta/twig, pagerfanta/doctrine-mongodb-odm-adapter: stránkování záznamů.
- Phpunit/phpunit: testovací framework pro psaní PHP testů.

2.3.6 Architektura aplikačního kódu

Server může být vybudován jako:

1. UI nezávislé API, které vrací data v přenosném formátu (JSON). Takové řešení osahuje komplexnější frontend, který řeší komunikaci se serverem a zobrazování těchto dat.

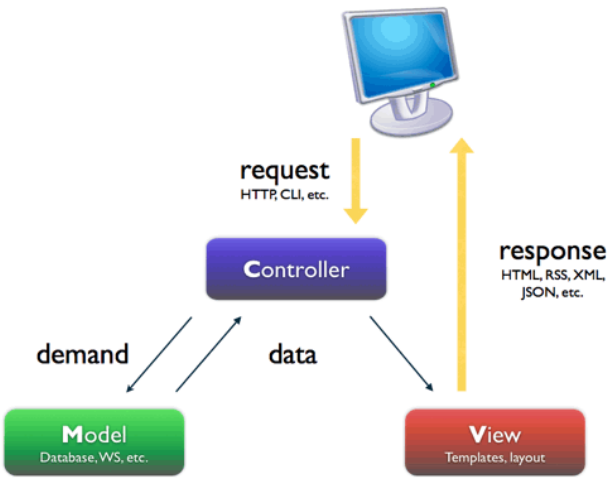
2. Server vracející hotové HTML (HTML, CSS, JavaScript).

V této práci bylo nakonec zvoleno řešení z bodu 2, tedy server vracející z většiny již hotový frontendový kód. Jde o implementačně jednodušší variantu s jednoduchým frontendem. Toto řešení vede k "silnému" serveru, tedy lze očekávat více kódu, zátěže, komplexity a logiky na serverové části. Naopak původní řešení (kapitola 2.3.8) volilo přístup UI nezávislého API.

Aplikační kód používá architekturu MVC (Obrázek 6). Tato architektura rozděluje kód do třech hlavních vrstev:

- Model: Řeší datové modely.
- View: Řeší grafické rozhraní a frontend.
- Controller: Orchestrační aplikační kód. Řeší obsluhu HTTP requestů; vrací HTTP response, často naplněním šablon vrstvy View daty z datových modelů vrstvy Model; řeší řízení přístupu…

Obrázek 6 Schéma architektury MVC



2.3.7 Databáze

Vybraný framework Symfony volitelně integruje projekt Doctrine, který nabízí ORM (Object Relation Mapper) nad vybranými SQL databázemi a ODM (Object Document Mapper) nad NoSQL MongoDB databází. Doctrine je použita či zmíněna na mnoha místech dokumentace frameworku, a je tak prezentována jako standardní způsob, jak ve frameworku pracovat s databází. Z tohoto důvodu byla použita Doctrine s databází MongoDB i v této práci. Databáze MongoDB byla vybrána především proto, že jde o populární NoSQL databázi, a není to databáze SQL. Autorovi se líbí přístup NoSQL databází více než SQL databáze, především protože jsou volnější na pravidla a nedochází tolik k duplikacím mezi aplikačním kódem a databází. MongoDB byla použita i v původním neúspěšném řešení (kapitola 2.3.8) a jako jedna z mála technologií nebyla nakonec považována za špatně zvolenou.

2.3.8 Slepá větev vývoje

Původně byly pro tuto práci vybrány jiné technologie a architektura. V pokročilém stádiu vývoje, po implementaci většiny funkčních požadavků, došlo k rozhodnutí přepsat projekt do současných tradičních technologií. Hlavní důvody pro toto rozhodnutí byly:

- Veliká složitost technologií a náročnost na znalosti vývojářů v porovnání s výsledným řešením.
- Veliký objem kódu. Objemem kódu byl frontend na neprázdné řádky přibližně 5x objemnější než výsledné řešení a backend přibližně 2x. Vedle složitého backendu tedy bylo potřeba vyvíjet velmi složitý frontend v porovnání s „hloupým“ frontendem výsledného řešení.
- Pomalý vývoj, náročná údržba, častá chybovost, mnoho problematiky k řešení.

Jedním z hlavních kritérií výběru byla populárnost těchto technologií. Byl vybrán jazyk TypeScript (typovaná verze JavaScriptu) jak pro frontend, tak i pro backend, což umožňovalo i sdílení kódu mezi backendem a frontendem. Jako hlavní technologie pro frontend byly zvoleny React a NextJs s nasazením do cloudu Vercel. Jako hlavní technologie backendu byl zvolen Nodejs bez frameworku, s nasazením do AWS cloudu

prostřednictvím infrastruktury definované kódem s použitím knihovny AWS CDK (Cloud Development Kit) a databáze Mongodb nasazená v cloudu Mongodb Atlas.

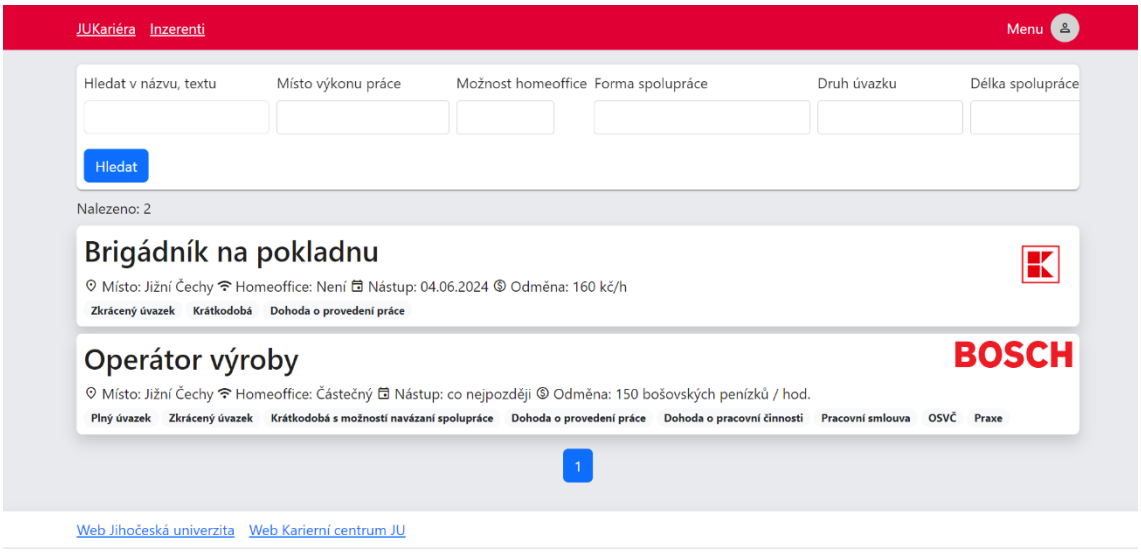
Dle průzkumu provedeného známým vývojářským portálem StackOverflow v roce 2022, byl JavaScript nejpopulárnější jazyk, React nejpopulárnější frontendovou technologií, Nodejs nejpopulárnější serverovou technologií, MongoDb čtvrtou nejpopulárnější databází a AWS nejpopulárnějším cloudem [63]. Několik těchto technologií se objevilo i v analýze existujících řešení.

Backend architektura byla zvolena jako UI nezávislý server poskytující JSON HTTP API. A sice z důvodu, že by se v budoucnu mohla najít jiná univerzitní služba, která by mohla interagovat s backendem.

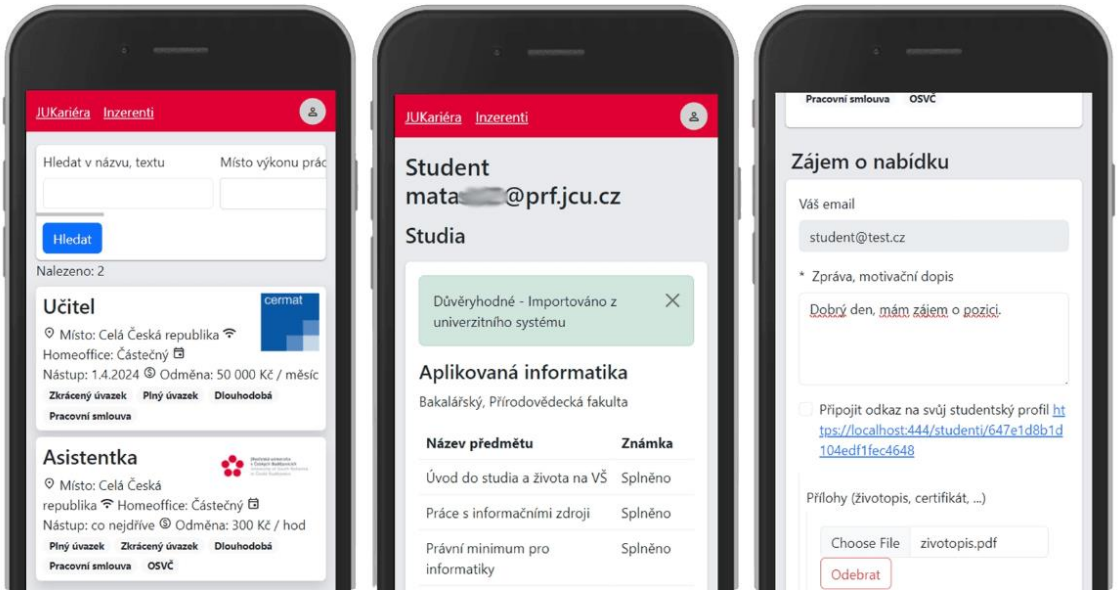
2.4 Design

Byl použit jednoduchý design inspirovaným systémem Material design společnosti Google, který je ukázán na Obrázek 7 a Obrázek 8. Vzhled je ovlivněn použitím knihovny Bootstrap a jejích předpřipravených UI prvků. Při implementaci byl kladen důraz na jednoduchost, která usnadňuje bezproblémové zobrazení portálu i na mobilních zařízeních. Barvy textu a výrazného navigačního pruhu jsou stejné jako barvy použité na hlavním webu univerzity a na webu KC. Prohlédnout vzhled všech stránek je možné v uživatelském manuálu.

Obrázek 7 Vzhled portálu - výpis nabídek



Obrázek 8 Vzhled portálu - mobilní zařízení



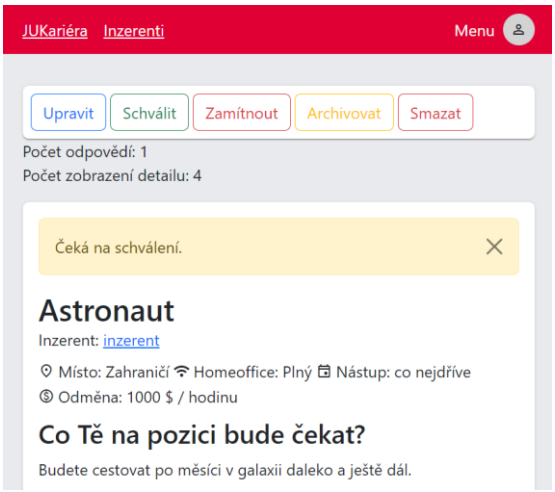
2.5 Administrátorské rozhraní

Pro administrátorského rozhraní byly zvažovány dva přístupy:

1. Separátní rozhraní administrace (typicky za postfixem */admin*).
2. Rozšíření stávajícího rozhraní o administrátorské rozhraní.

Pro tuto práci byl zvolen přístup z bodu 2 (Obrázek 9). Ten pouze přidává administrační prvky k již vytvořeným šablonám. Administrační rozhraní je tak umístěno přímo vedle zobrazovaných dat, ke kterým se vztahuje. Další výhodou je pravděpodobně menší množství kódu než by bylo v případě tvorby separátní administrace. Nevýhodou tohoto řešení je, že jsou šablony složitější o logiku (ne)zobrazení administrátorského rozhraní.

Obrázek 9 Rozšíření UI o administrátorské akce



2.6 Vývojové prostředí

2.6.1 Docker

Vytvoření vývojového prostředí je dobré co nejvíce automatizovat. Kdykoli může nastat potřeba vytvořit další, ať už selháním vývojového stroje, nebo pokud se má na projektu podílet nová osoba. Automatizace navíc slouží jako dokumentace, protože někde musí být definována. Z definice vývojového prostředí je i v čase patrné, co vše je potřeba ke spuštění projektu. Za tímto účelem byla použita kontejnerizační technologie Docker.

Kromě použití připraveného lokálního vývojového prostředí formou Dockeru je možná i vlastní instalace a zprovoznění technologií použitých v této práci.

K nainstalování a spuštění vývojového prostředí stačí mít nainstalovaný a funkční kontejnerizační program Docker, následně ve složce se zdrojovým kódem spustit příkaz *docker compose up*. Tím se nainstalují a spustí kontejnery obsahující databázový server, webserver a další pomocné kontejnery potřebné pro vývoj. Po spuštění je aplikace přístupná přes webový prohlížeč na adrese http://localhost. Mohlo by dojít ke konfliktu portů, pokud je již na stroji nějaký stejný port použitý, v takovém případě je potřeba upravit porty v definici kontejnerů. Porty, na kterých jsou spuštěny jednotlivé služby, je možné prohlédnout v souborech *docker-compose*.

Vývojové prostředí běží zcela na stroji vývojáře. Definice vývojového prostředí vychází z kódu na Githubu, na který odkazuje dokumentace Symfony. [64] [65]

Prostředí vytvořené s použitím Dockeru je přenositelné, a lze jej proto snadno přenášet z vývojového stroje na produkční server, a v podstatě všude tam, kde je nainstalován Docker. [66]

V případě této práce se spouští celkem 6 virtuálních kontejnerů obsahující:

- webserver kontejner,
- PHP kontejner,
- databázový kontejner,
- kontejner se službou Smtp4dev zachytávající odchozí emaily při vývoji a umožňující je zobrazit na adrese http://localhost:1080,
- Playwright kontejner pro spouštění end-to-end testů.

Definice kontejnerů se nachází v souborech *docker-compose.yml, docker-compose.override.yml* a *Dockerfile*. Soubory *docker-compose* orchestrují kompozici více kontejnerů a mohou se odkazovat na soubor Dockerfile. [20]

2.6.2 Použité prostředí pro vývoj

Práce byla programována na strojích s operačním systémem Ubuntu a s operačním systémem Windows, na kterém však byla nainstalována technologie WSL2 (Windows Subsystem for Linux) a systém Ubuntu. Vývoj probíhal na systému Linux, protože bylo předpokládáno nasazení na server s operačním systémem Linux. Windows Server nebyl zvažován kvůli minimalizaci nákladů na provoz – jde na rozdíl od operačního systému Linux o licencovaný komerční produkt [67]. Vývojové prostředí je dobré co nejvíce přiblížit produkčnímu prostředí, je tak možné předejít problémům, které by se vyskytovaly pouze na jednom z odlišných prostředí.

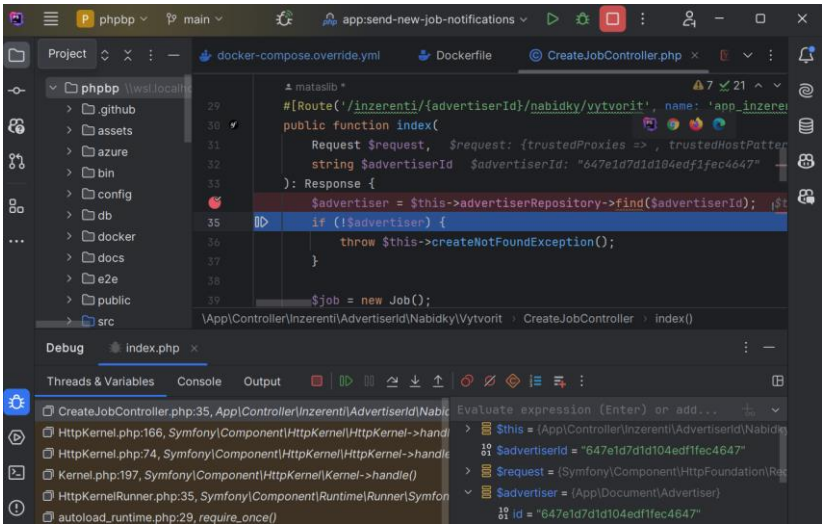
Bylo použito integrované vývojové prostředí PhpStorm pro vývoj kódu. Jedná se o jedno z nejpopulárnějších IDE, které nabízí širokou paletu funkcionalit, inteligentní asistence a vylepšenou podporu frameworku Symfony po instalaci pluginu.

Při vývoji aut. testů byl použit editor VSCode pro jeho snazší nastavení debuggování.

2.6.3 Ladění PHP kódu

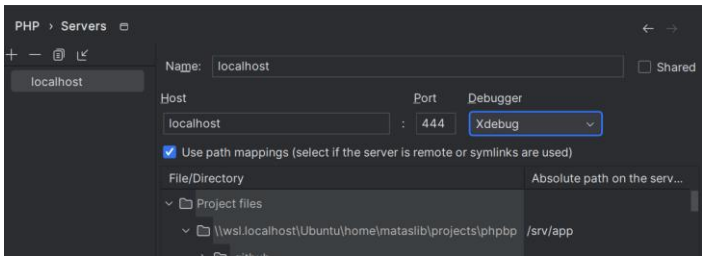
Pro ladění kódu je v PHP obrazu nainstalován debugger Xdebug. Ten umožňuje krokovat kód, prohlédnout či pozměnit hodnoty proměnných (Obrázek 10)... Pro ladění HTTP requestů pomocí Xdebug je kromě jeho instalace nutné řešit i konfiguraci. Při debuggování spolu komunikuje editor a debugger, kteří mohou být i na rozdílných strojích. Je proto nutné řešit správnou konfiguraci debuggeru jak v editoru, tak i na serveru a nasměrovat je na sebe. [68]

Obrázek 10 Debuggování a krokování requestu v PhpStorm IDE s debuggerem Xdebug



Konfigurace debuggeru v IDE PhpStorm je vidět na Obrázek 11. Je zde specifikována komunikace (host, port) a mapování cest mezi strojem s editorem a odpovídající cestou */srv/app* na serveru (Docker kontejneru). V případě IDE PhpStorm je také nutné aktivovat naslouchání Xdebug připojení viz **Chyba! Nenalezen zdroj odkazů.**.

Obrázek 11 Nastavení Xdebugu v IDE PhpStorm



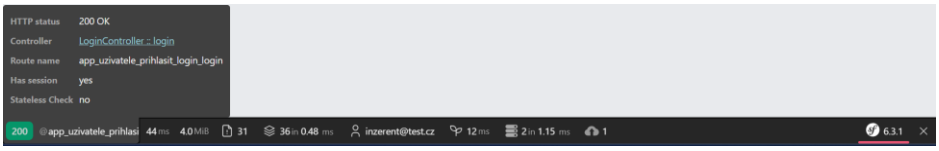
Obrázek 12 Tlačítko pro zapnutí naslouchání Xdebug spojení v IDE PhpStorm

Protože komunikaci mezi editorem a serverem zahajuje Xdebug na serveru, je potřeba i na serveru specifikovat síťovou adresu a port editoru. Xdebug by měl při zapnutí konfigurace *xdebug.discover_client_host* zjistit IP adresu stroje, který zahájil komunikaci z HTTP requestu. Tato konfigurace však nefungovala, pravděpodobně z důvodu použití Dockeru. Bylo potřeba konkrétně specifikovat adresu stroje s editorem konfigurací na serveru *xdebug.client_host='host.docker.internal'*. V základu je Xdebug nastaven tak, aby se aktivoval, pouze pokud HTTP request obsahuje určitou signalizující vlastnost. Tuto vlastnost může nastavovat doplněk v prohlížeči, ale lze ji přidávat i ručně. Existuje i automatický režim *xdebug.start_with_request=yes*, který debugger aktivuje s každým HTTP requestem. [68]

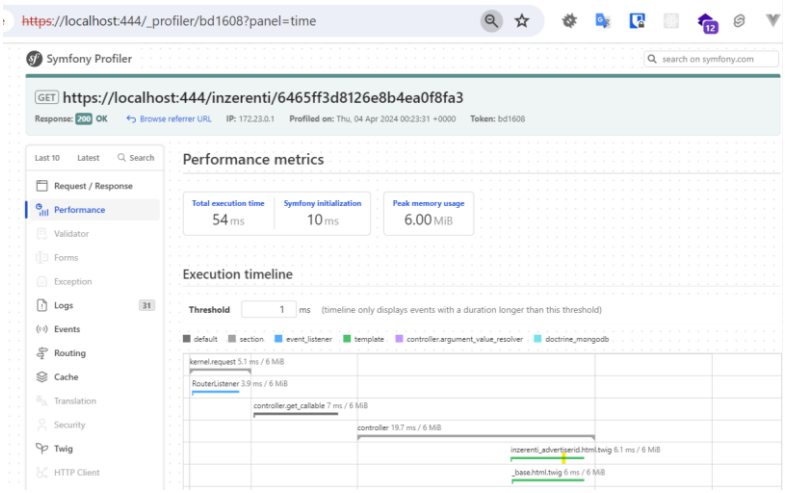
2.6.4 Symfony Debug toolbar

Framework Symfony zlepšuje ladění tím, že obohacuje šablony ve vývojovém režimu (v produkčním nikoli) o tzv. Debug toolbar (Obrázek 13). Jde o lištu pro ladění aplikace obsahující užitečné informace včetně odkazu do nástroje profilování (Obrázek 14). V těchto nástrojích lze nalézt různé užitečné informace jako je aktuální obsluhující Controller, analýza rychlosti načítání, vykonané databázové dotazy a jejich doba trvání, použité šablony apod.

Obrázek 13 Symfony Debug Toolbar



Obrázek 14 Nástroj profilování



2.7 Datové modelování

V rámci této práce byly zváženy 3 přístupy:

1. Data nemodelovat
2. Modelovat
    a. Anemické modely
    b. Rich domain modely

Jedním z možných přístupů k datovému modelování je neprovádět žádné modelování struktury dat. Tento přístup, kdy se vykonávají příkazy databáze, bez použití datových struktur jako mezičlenů, bývá vhodný spíše pro velmi jednoduché aplikace a scripty. Modelování dat do aplikace vnáší strukturu, znuvupoužitelnost, konzistenci a dokumentaci. Pro vývojáře tak přináší vyšší čitelnost a pochopitelnost. Kvůli uvedeným výhodám bylo rozhodnuto data modelovat.

Struktura dat může být definována jak na straně aplikačního kódu, tak i na straně databáze. U tradičních webových aplikací, využívající jako uložiště relační databáze, může docházet k duplikaci definice struktury a pravidel mezi aplikačním kódem a databází. Aby byly oba modely synchronní, využívá se např. ORM přístupu, kdy se v aplikačním kódu definují mapování objektů na koncepty databáze a z aplikačního kódu lze následně databázi generovat. Nesynchronní struktury by mohly vést k chybám. Tento

přístup bývá obtížnější na změny, například je potřeba řešit migrace tak, aby nové změny začaly v ideálním případě platit v aplikaci i databázi ve stejný okamžik.

Tím, že byla zvolena NoSQL databáze, nutnost synchronizace struktur odpadá, avšak protože tato databáze neumožňuje a nevyžaduje specifikovat strukturu dat, je o to důležitější definovat a tím i dokumentovat tuto strukturu alespoň v aplikačním kódu.

Zbývalo rozhodnout, zda data modelovat anemická, nebo domain rich, a tím i zda budou modely obsahovat hodně, či málo logiky.

Anemické modely slouží výhradně jako datové schránky, často bez jakékoli další logiky. V takových třídách modelů je minimum chování a metod. [69]

Naproti tomu domain rich modely obsahují rozsáhlejší množství chování a pravidel. Kromě datové struktury modelují také životní cyklus, možné stavy a přechody mezi nimi. Díky tomu je možné konzistentně vynutit pravidla objektu napříč aplikací. Tyto modely jsou vhodné pro velké a komplexní projekty, kde je kritické vynucovat pravidla a zabránit nekonzistenci dat. [69]

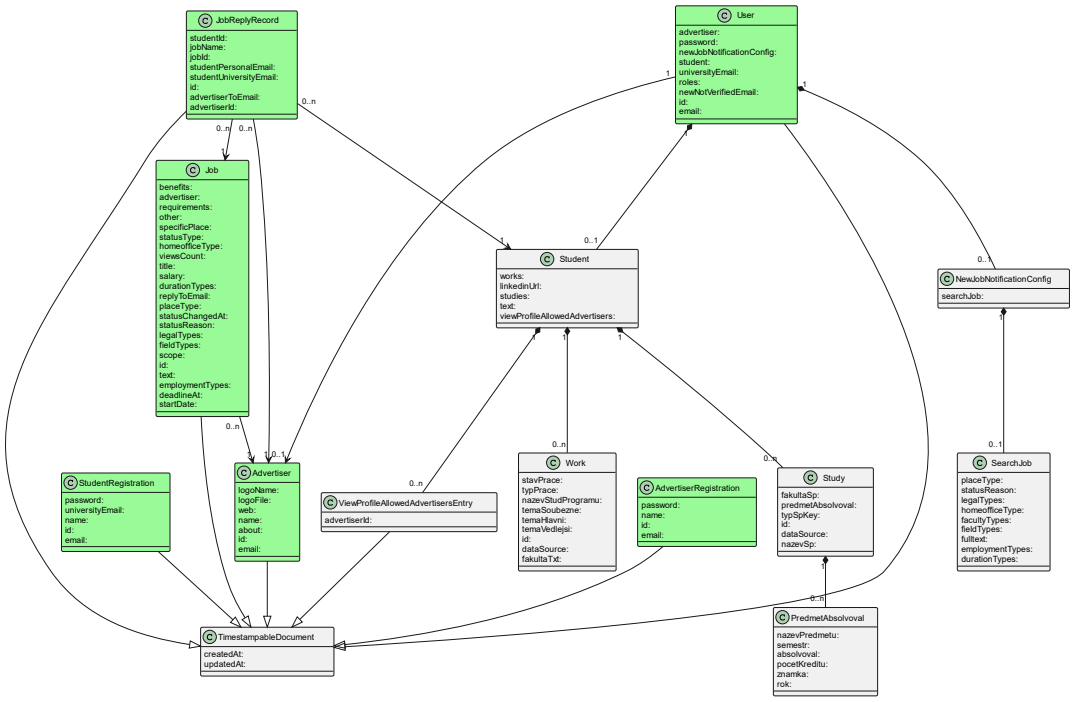
Anemické modely jsou naopak vhodné pro jednoduché a CRUD aplikace, jelikož je snadné dostat objekt do jakéhokoli stavu a mapovat na jiné koncepty, jako je request nebo formulář. Avšak nevýhodou anemických modelů je, že pokud se logika a pravidla nenachází přímo v modelu, tak se nutně musí nacházet jinde. To může vést k náročnějšímu udržování konzistence práce s objektem napříč aplikací, opakování kódu a obtížnějšímu odhalování a vynucování pravidel těchto modelů, protože mohou být v aplikaci roztroušena na různých místech. [69]

Martin Fowler označuje anemické modely za procedurální styl a domain rich modely za objektově orientovaný styl. [69]

V práci byl použit přístup modelování anemických modelů především pro jeho jednoduchost. Anemický model je oproti domain rich modelu nenáročný na schopnosti a znalosti vývojáře.

Na Obrázek 15 je k prohlédnutí class diagram datových modelů. Modely, které jsou vlastní dokumenty v databázi, a nejsou tak součástí ostatních dokumentů, jsou podbarveny zeleně.

Obrázek 15 Class diagram datových modelů.



Dokument univerzitního inzerenta (*Advertiser*) je modelován stejně jako účty ostatních inzerentů, není téměř ničím speciální. Pouze se na něj odkazuje více uživatelů (administrátorů) než je tomu v případe standardního inzerenta, kde je to vždy uživatel pouze jeden.

2.8 Doctrine ODM, propojení frameworku a databáze

Doctrine umožňuje mapovat objekty jazyka PHP na datovou reprezentaci v databázi, a tím automatizuje nutnost transformace a mapování mezi aplikační a databázovou datovou reprezentací. Nabízí mnoho dalších funkcionalit: eventy v životním cyklu entity, repository pattern, implicitní generování identifikátorů, cache, transparentní transakce, query builder, migrace, objektové API nad databází, ochranu před SQL injection… Některá zmíněná funkcionalita se týká pouze SQL databází, buď zatím v NoSQL nebyla implementována, protože je NoSQL novější koncept a ve světě PHP historicky méně populární a používaná, nebo takový problém ani nemusí v NoSQL existovat – např. potřeba synchronizovat aplikační schéma se schématem v databázi (MongoDB schéma nemá).

Na příkladu ve Fragment 1 je vidět výhoda mapování pomocí Doctrine. Vlastnost *advertiser* je objekt typu *Advertiser*, i když je ve skutečnosti v databázi uložena pouze

jako ID na dokument v kolekci *Advertiser* (Fragment 2). Doctrine transparentně naplní vlastnost *adverrtiser* daty z její vlastní kolekce, nebo do ní data uloží při uložení nabídky.

To je komfortní způsob práce s modely.

Doctrine na pozadí dělá i další věci, např. transparentně transformuje PHP objekty na databázové objekty MongoDb driveru a naopak. Vývojář tak může pracovat se standardními objekty PHP světa, jeko je string pro identifikátory či DateTime objekt pro čas. Při uložení Doctrine tyto objekty přetransformuje na objekty driveru jako je *MongoDB\BSON*ObjectId a *MongoDB\BSON\UTCDateTime*. To by jinak vyžadovalo množství transformujícího kódu.

Na příkladu ve Fragment 2 je také vidět využití eventů životního cyklu entity u vlastnosti *createdAt*, která je prostřednictvím anotace *Timestampable* při uložení do databáze transparentně naplněna aktuálním datem.

Fragment 1 Příklad aplikační datové reprezentace – PHP objektu Job a jeho mapování na databázovou reprezentaci

```
class Job
{
    #[ODM\Id]
    public string $id;

    #[ODM\ReferenceOne(
        targetDocument: Advertiser::class,
        storeAs: "ref")]
    public Advertiser $advertiser;

    #[ODM\Field(type: "date")]
    #[Gedmo\Timestampable(on: "create")]
    public DateTime $createdAt;
```

Fragment 2 Data v databázi odpovídající aplikační reprezentaci z Fragment 1

```
# collection Job
{
  _id: ObjectID(„63933b95ef42a2b174abaf36“),
  createdAt: Date(2022-12-09T13:43:49.979+00:00),
  advertiser: {
    _id: ObjectId(„632d781f8439d4577dbffcd3“)
  }
  …,
```

```
}



# collection Advertiser
{
  _id: ObjectID(„632d781f8439d4577dbffcd3“),
  name: “Jihočeská univerzita“,
  …
}
```

2.9 Řízení přístupu

Důležité funkce portálu jako jsou registrace, výpis a hledání nabídek či inzerentů, detail nabídky a detail inzerenta jsou veřejně přístupné. Záměrem je, aby se portál objevoval i ve výsledcích vyhledávačů, a přilákal tak co největší množství uživatelů.

Neveřejné funkce aplikace vyžadovaly implementaci řízení přístupu. Za tímto účelem byl použit mix modelu RBAC (Role Based Access Control), který rozhoduje o udělení oprávnění na základě rolí a modelu ABAC (Attribute Based Access Control), který rozhoduje o udělení oprávnění na základě atributů datového modelu. Byla použita kombinace obou těchto přístupů, protože mnohdy nebylo možné rozhodnout o oprávnění pouze na základě role, ve které uživatel vystupuje, ale bylo potřeba pracovat i s jinými souvisejícími atributy.

RBAC model byl implementován přidáním možnosti přiřadit uživatelům role. Pomocí rolí lze uživatele snadno roztřídit do určitých skupin a řídit přístup za základě jejich rolí. Toho bylo docíleno vytvořením atributu *roles* v dokumentu *User,* obsahující pole možných rolí:

- ROLE_ADMIN: administrátorská role,
- ROLE_ADVERTISER: inzerentská role,
- ROLE_STUDENT: studentská role.

Framework Symfony poskytuje podporu pro RBAC model. Po implementací interface *UserInterface* v entitě *User*, je vyžadována implementace mj. funkce *getRoles*, která má vrátit role uživatele. Framework následně tuto funkci využívá v jeho systému řízení

přístupu. Různé možnosti použití jsou vidět na Fragment 3. Pomocí řízení přístupu lze omezit přístup k různě velkým částem kódu, nebo i celé stránce.

Fragment 3 Ukázky použití řízení přístupu na základě role

```
$isAllowed = $security->isGranted('ROLE_ADVERTISER');
$security->denyAccessUnlessGranted('ROLE_ADVERTISER');

#[IsGranted('ROLE_ADVERTISER'')]
#[Route('/guardedRoute ')]
public function guardedRoute()
{
}

# pouziti v Twig
{% if is_granted('ROLE_STUDENT') %}
{% endif %}
```

Jak již bylo dříve zmíněno, v některých případech však pouhá role nedostačuje k rozhodnutí o udělení oprávnění, a je potřeba použít atributy. V takovém případě je možné psát vlastní logiku přímo do kódu, která by však trochu narušila dosavadní konzistenci v používání řízení přístupu pomocí frameworku, a pravděpodobně by skončila roztroušená v různých částech projektu.

Vhodnějším způsobem, umožňující logiku udělení přístupu centralizovat, je použití tzv. *Voterů*. Implementace *Votera*, který zapříčiní, že jakékoli volání funkce řízení přístupu Symfony skončí pro uživatele s admin rolí úspěšně, je vidět na Fragment 4. Jde o třídy, které implementují Symfony interface *VoterInterface*, nebo dědí třídu *Voter*. Kód těchto tříd je volán při každém použití některé funkce řízení přístupu Symfony (*isGranted, denyAccessUnlessGranted,* …), a může se tak účastnit rozhodnutí o (ne)udělení přístupu. Ve *Voterech* je nutné implementovat metodu *voteOnAttribute*, která obsahuje logiku (ne)udělení přístupu. Nakonec také metodu *supports*, která rozhoduje o tom, zda se *Voter* má či nemá účastnit rozhodnutí o daném oprávnění. [70]

Fragment 4 Implementace Votera, který povoluje adminům veškerá oprávnění

```
// Allows everything for user with role admin
class AdminVoter extends Voter
{
    protected function supports(string $attribute, mixed $subject):
bool
```

```
    {
        return true; // vote on everything
    }

    protected function voteOnAttribute(string $attribute, mixed
$subject, TokenInterface $token): bool
    {
        $user = $token->getUser();
        // if the user is anonymous, do not grant access
        if (!$user instanceof User) {
            return false;
        }

        return in_array(Role::ADMIN->value, $user->roles);
    }
}
```

Kombinací rolí, atributů, *Voterů*, a funkcí řízení přístupu frameworku Symfony lze velmi snadno a elegantně řídit přístup v aplikaci. V Aplikaci byly implementovány tyto Votery:

- UserVoter: Zabývá se rozhodováním o oprávněních týkající se uživatelů.
- AdminVoter: Povoluje administrátorům veškerá oprávnění.
- AdvertiserVoter: Zabývá se rozhodováním o oprávněních týkající se operací inzerentů.
- StudentVoter: Zabývá se rozhodováním o oprávněních týkající se operací studentů.
- JobVoter: Zabývá se rozhodováním o oprávněních týkající se nabídek.

2.10 Integrace s IS/STAG

„IS/STAG je informační systém studijní agendy pro vysoké školy a univerzity. Systém vznikl a je vyvíjen Centrem informatizace a výpočetní techniky – Střediskem informačních systémů na Západočeské univerzitě v Plzni. Poprvé se IS/STAG použil v roce 1993 na ZČU. […] Systém v současnosti používá 13 škol v České republice. Z toho je 11 veřejnoprávních vysokých škol a 2 jsou soukromé vysoké školy.“ [71]

S informačním systémem STAG ve svém studiu pracuje každý student. Mj. si zde zapisuje předměty, zkoušky apod. V této práci byl integrován jako autentizační metoda studentů či administrátorů, a za účelem automatizace tvorby profilu studenta.

Student má možnost do svého profilu importovat data ze STAG. Jedná se především o data o studentovi, absolvovaných studijních programech, kurzech a kvalifikačních pracích.

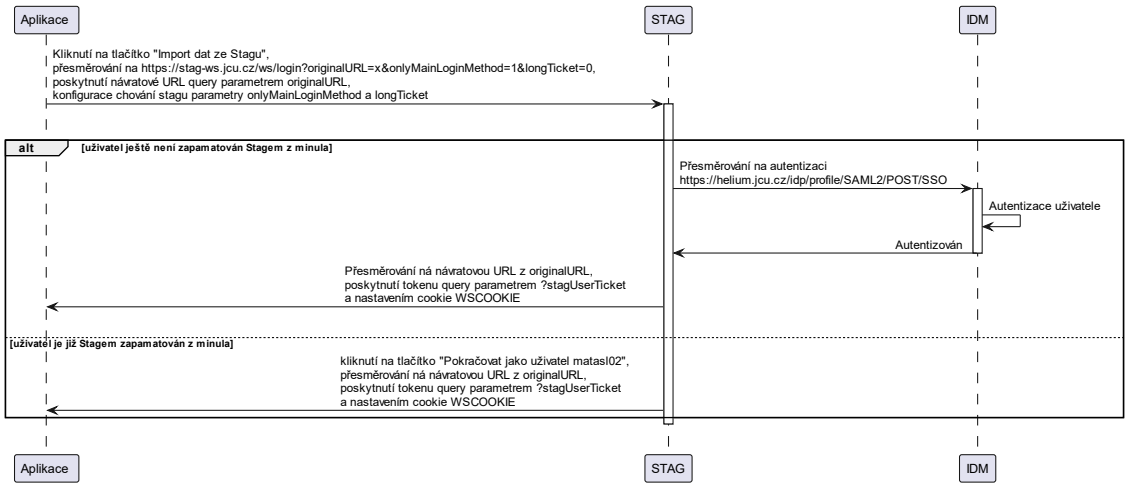
Data jsou získána skrze HTTP API STAG. Komunikace s API probíhá prostřednictvím identity studenta, který má oprávnění jen na svá data, je tak dosaženo větší bezpečnosti oproti implementační alternativě využití systémového účtu, který by měl přístup k datům všech studentů.

Jakmile je studentovi ukončeno studium, ztrácí přístup ke STAG a nemůže již využívat ani integraci STAG v této práci. Z tohoto důvodu jsou studenti nuceni při registraci zadávat svou osobní emailovou adresu, aby mohli portál stále používat s autentizací prostřednictvím osobní emailové adresy a hesla.

STAG JČU má webové služby vystavené na adrese https://stag-ws.jcu.cz. Toto je i základ URL adresy při provolávání služeb API. Při přistoupení přes webový prohlížeč jsou na této adrese dostupné užitečné odkazy jako je odkaz na dokumentaci nebo seznam změn. Také je odtud možné najít seznam s dokumentací služeb a možnost je provolávat přímo v prohlížeči, což je velmi komfortní způsob pro rychlé testování, zorientování se a prozkoumání výstupních dat. Některé Služby mohou nejprve vyžadovat přihlášení se do STAG.

Aplikace potřebuje pro použití API STAG přístupový token uživatele. Proces získání tokenu je vizualizován v sekvenčním diagramu viz Obrázek 16. Token je získán přesměrováním na STAG a poskytnutím návratové URL zpět do aplikace, kam bude uživatel přesměrován po úspěšné autentizaci ve STAG. Při přesměrování na autentizační stránku STAG je možné konfigurovat jeho chování. Volit mezi krátkodobým (30 minut) či dlouhodobým ticketem (až 90 dnů) nebo omezit autentizační metodu pouze na hlavní (přihlášení omezit na IDM). Kód přesměrování z aplikace do STAG je ukázán ve Fragment 5. Po úspěšné autentizaci je jednak nastavena cookie WSCOOKIE, jejíž hodnotou je přístupový token, navíc je token vracen ještě jako parametr v návratové URL. [72]

Obrázek 16 Sekvenční diagram zachycující proces získání tokenu uživatele autentizací ve STAG



Fragment 5 Konstrukce návratové URL ze STAG a přesměrování do STAG

```
$redirectUrl = $this->generateUrl(
"app_studenti_studentid_studium_importovat_importstudy_import",
    [
        "userId" => $user->id,
        "withGrades" => $formData["withGrades"],
    ],
    UrlGeneratorInterface::ABSOLUTE_URL,
);
$queryString = http_build_query(["originalURL" => $redirectUrl]);

return $this->redirect("https://wstag.jcu.cz/ws/login?$queryString");
```

Jakmile má aplikace k dispozici token, připojuje ho již bez interakce uživatele při volaní API služeb v HTTP hlavičce Authorization viz Fragment 6. Z bezpečnostního hlediska token není aplikací ukládán, je získáván pouze po explicitní akci studenta, a je žádán pouze krátkodobý token s platností 30 minut.

Fragment 6 Získání přístupového ticketu z URL po návratu ze STAG. Konstrukce STAG API URL a poslání požadavků

```
$stagUserTicket = $request->query->get('stagUserTicket');

$getStagForActualUserResponse = $this->httpClient->request(
    "GET",
    "https://stag-
ws.jcu.cz/ws/services/rest2/help/getStagUserForActualUser?outputFormat
=JSON",
    [
        "auth_basic" => [
            "username" => $stagUserTicket,
```

```
            "password" => "",
        ],
    ]
);
$stagForActualUserResData = $getStagForActualUserResponse->toArray();

# Odpověď z API:
{
  "userName": "B191XY",
  "role": "ST",
  "roleNazev": "Student",
  "fakulta": "FPR",
  "katedra": null,
  "ucitIdno": null,
  "osCislo": "B191XY",
  "email": "mataslXY@prf.jcu.cz"
}
```

V práci jsou využity následující služby STAG a jeho API:

- https://wstag.jcu.cz/ws/login: přihlašovací stránka;
- https://stag-ws.jcu.cz/ws/services/rest2/help/getStagUserForActualUser: data o uživateli (email, osobní číslo…);
- https://stag-ws.jcu.cz/ws/services/rest2/kvalifikacniprace/getKvalifikacniPrace: data o kvalifikačních pracích;
- https://stag-ws.jcu.cz/ws/services/rest2/student/getStudentInfo: data uživatele zahrnující i studijní data (email, fakulta, program, …);
- https://stag-ws.jcu.cz/ws/services/rest2/student/getStudentPredmetyAbsolvoval: data o předmětech studenta;

2.11 Šablony

PHP vzniklo jako šablonovací jazyk reakcí na nepraktičnost vývoje dynamických webů prostřednictvím jazyků jako je jazyk C. Časem se z jazyka PHP stal obecný programovací jazyk. Začaly vznikat šablonovací jazyky i pro jazyk PHP, protože se ukázalo, že ani PHP není nejlepším šablonovacím jazykem. Vznikl například šablonovací jazyk Smarty, který limitoval schopnosti PHP. Neumožnoval například použití databázových dotazů a

striktně tak oddělil databázovou a pohledovou vrstvu, což je často cílem dobré architektury. Použití dalších šablonovacích jazyků však kromě výhod přináší i nevýhody jako potřebu učit se další jazyk. Časem se ukázalo, že výhody převažují nevýhody. [73]

Při pohledu na nejpopulárnější PHP frameworky je vidět, že použití šablonovacích jazyků pro PHP převážilo výhodami. Symfony používá šablonovací jazyk Twig [74], Nette jazyk Latte [75], Laravel jazyk Blade. [76] Tato práce používá šablonovací jazyk Twig (Obrázek 17).

Obrázek 17 Ukázka části Twig šablony

```
<h1 class="m-0">{{ job.title }}</h1>
   <div>
      Inzerent: <a href="{{
path('app_inzerenti_advertiserid_advertiserdetail_index', {
   advertiserId: job.advertiser.id
}) }}">{{ job.advertiser.name }}</a>
```

Hlavní výhodou šablonovacích jazyků je zabudovaná automatická bezpečnost před útokem XSS. Standardní vypsání obsahu v PHP bez explicitního volání bezpečnostních funkcí jako je funkce *htmlspecialchars()* je nebezpečné a může dojít ke spuštění cizího kódu. Zatímco v PHP musí být ošetření před útokem XSS explicitní, šablonovací jazyky fungují opačně, a chrání implicitně. Automaticky tedy ošetřují vypisované HTML tzv. „escapováním“. Některé šablonovací jazyky jako je Latte dokáží i zohlednit kontext, do kterého je obsah vypisován. Zda pouze do běžných elementů HTML, nebo do JavaScriptového bloku *<script></script>* apod. V každém kontextu se totiž nebezpečné symboly liší. Implicitní ochranu lze při výpisu vypnout použitím filtru *|raw*, který vypíše obsah tak jak je, a otevírá tak dveře pro útok XSS. Autoři Latte poukazují na nepraktičnost jazyka Twig, který se svou syntaxí více než PHP podobá Pythonu, s čímž se autor práce ztotožňuje. [73]

2.12 Formuláře

Formuláře jsou standardně součástí většiny webových stránek. Tato práce obsahuje přes 15 formulářů. Formuláře jsou jednou ze složitějších oblastí vývoje webových stránek. Část jejich funkcionality se odehrává na serveru a část na clientu. V ideálním případě se obě částí chovají synchronně.

Clientská část by měla být pro uživatele přívětivá, zabývat se uživatelským rozhraním, měnit strukturu formuláře v závislosti na interaktivitě a zobrazovat validační chyby. Serverová část by měla sloužit jako zdroj pravdy, ochrana stavu systému, a obsahovat logiku zpracovávající data odeslaná z clienta.

Prostřednictvím formulářů dochází ke změně stavu systému uživatelem, a proto jsou formuláře jedním z možných vektorů útoku. Je proto nutné myslet i na bezpečnost.

Formuláře mohou být statické nebo v čase dynamické, kdy se v závislosti na interaktivitě mění jejich struktura. Náročné na implementaci jsou především formuláře dynamické.

Existuje několik možných způsobů jak k implementaci formulářů přistupovat. Framework Symfony nabízí poměrně komplexní systém pro tvorbu formulářů. Výhodou tvorby formulářů prostřednictvím Symfony je, že je řešena validace, jsou rozšiřitelné, kód formuláře je poměrně pěkně centralizován do jediného kódu na serveru, umožňují generovat frontendovou část formuláře (Fragment 9), zároveň umožňují i individuální úpravy generovaného kódu. Pokud v průběhu projektu dojde k rozhodnutí, že má nějaký vstup vypadat jinak nebo se jinak chovat, např. textové pole, pak je díky Symfony snadné upravit funkcionalitu pro všechny formuláře napříč celým webem.

2.12.1 Propojení HTTP requestu, formuláře a datového modelu

Formulářový systém transparentně mapuje data HTTP requestu na objekt formuláře při zavolání metody $form->handleRequest($resquest). *Formulář je následně schopný vytvořit a naplnit daty novou instanci typu určeným atributem* data_class *(*

Fragment 8). Případně může pouze data mapovat na již existující instanci, pokud je mu předána, což se hodí pro případ úpravy dat již existujícího záznamu. Není proto potřeba psát žádný objemný mapující kód mezi HTTP requestem, formulářem a datovým modelem za předpokladu, že se názvy a struktura polí requestu, formuláře a modelu shodují.

Pokud formulář nespecifikuje žádnou třídu atributem *data_class*, na kterou by se měl mapovat, ani mu není předána instance objektu, pak je jeho výstupem asociativní pole.

2.12.2 Validace

Formulář při zavolání zpracování requestu validuje data na základě:

- Anotací Constraints* použitých na vlastnostech třídy propojené s formulářem parametrem *data_class* (Fragment 7).

Specifikace atributu contraints *u formulářového pole (*

- Fragment 8).

Ostatních atributů jako je choices *u formulářového prvku* ChoiceType *(*

- Fragment 8). Framework při zpracování formuláře na serveru, při použití inputu výběru ze seznamu hodnot *ChoiceType*, vyhodnotí, zda je příchozí hodnota přípustná dle seznamu možných hodnot atributu *choices*. Případně zdali nebyla uživatelem na clientu podstrčena hodnota mimo definovaný seznam přípustných hodnot.

Symfony navíc při generování frontendu formuláře zobrazí u nevalidních polí chybovou hlášku.

Fragment 7 Definování validace anotací u vlastnosti třídy

```
class Job
{
    #[Constraint\NotBlank]
    public string $title;
```

Fragment 8 Ukázka definice formuláře pro vytvoření inzertní nabídky

```
class CreateUpdateJobType extends AbstractType
{
    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => Job::class,
        ]);
    }

    public function buildForm(FormBuilderInterface $builder, array
$options): void
    {
                ->add('legalTypes', ChoiceType::class, [
                    'label' => 'Forma spolupráce',
                    'required' => true,
                    'multiple' => true,
```

```
                    'choices' => array_combine(LegalType::values(),
LegalType::values()),
                ])
                ->add('deadlineAt', DateType::class, [
                    'label' => 'Deadline/Platnost nabídky',
                    'required' => true,
                    'help' => "Po tomto datu bude nabídka automaticky
archivovaná.",
                    'widget' => "single_text",
                    'attr' => [
                        'min' => (new DateTime('midnight'))-
>format('Y-m-d'),
                        'max' => (new DateTime('midnight + 90 days'))-
>format('Y-m-d'),
                    ],
                    'constraints' => [
                        new Range([
                            "min" => 'midnight',
                            "max" => 'midnight + 90 days',
                        ]),
                    ],
                ])
                // …
                // …
```

Vygenerování clientského kódu pro formulář z

Fragment 8 je díky frameworku velmi jednoduché a je vidět na Fragment 9, kdy ve skutečnosti těchto 7 řádků kódu vygeneruje přes 150 řádků clientského HTML kódu, který by jinak musel být napsán.

Fragment 9 Vygenerování clientského kódu formuláře z

Fragment 8 v šabloně

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

Jak má vypadat generovaný frontendový kód lze modifikovat v souboru _my_form_theme.html.twig. Ten byl nastaven v konfiguračním souboru twig.yaml jako

základní formulářová šablona. Jde jen o jeden z možných způsobů, jak generování ve frameworku ovlivnit. Symfony již v základu nabízí formulářovou šablonu pro knihovnu komponent Bootstrap. Jelikož tato práce knihovnu Bootstrap používá, tak i formulářová šablona _my_form_theme.html.twig rozšiřuje šablonu bootstrap_5_layout.html.twig, která se stará o to, aby se všechny základní formulářové typy vykreslovaly ve struktuře a vzhledu knihovny Bootstrap.

2.12.3 Zabezpečení proti CSRF

Formulářový systém Symfony za vývojáře na pozadí řeší také bezpečnost proti útoku CSRF (Cross Site Request Forgery). Jde o ochranu proti útokům, kdy útočník využívá toho, že se některé věci v prohlížeči dějí automaticky, že se některá data „pamatují“ napříč různými HTTP requesty. Útočník je schopen tohoto využít, a podstrčit zbylá data požadavku například tak, aby došlo k nechtěné změně stavu na serveru – smazání účtu, převodu peněz a podobně závažné akci. Data, která se napříč požadavky běžně “pamatují”, jsou například data zajišťující přihlášení uživatele. Není totiž komfortní, aby se uživatel neustále znovu přihlašoval.

Automatika, které se tyto útoky týkají, je především technologie zvaná Cookies. Jde o uložiště prohlížeče, na které si může server odkládat data. Prohlížeč následně tato data automaticky posílá s každým dalším požadavkem zpět tomuto serveru. Typicky si server do cookies uloží identitu přihlášeného uživatele, zasláním cookies v každém dalším požadavku server ví, o jakého uživatele se jedná, a tedy že je již ověřený.

Nejjednodušší formou CSRF útoku může být podstrčení requestu oběti emailem např. prostřednictvím odkazu s adresou *https://aplikace.cz/smaz-ucet* maskovaným pod jiným záměrem. Jakmile uživatel odkaz v emailu otevře, tak pokud je na serveru pamatován jeho přihlášený stav, doje ke smazání účtu. Tento útok bude úspěšný pouze u špatně napsaných aplikací. Obecné doporučení je, že žádná citlivá a stav měnící operace nesmí být dovolena provést prostřednictvím requestu typu GET. Ten má sloužit pouze ke získávání dat ze serveru a nikoli ke změnám dat na serveru. Odkazy vždy provádějí HTTP request typu GET.

Další forma útoku CSRF je, že si útočník umístí formulář na svůj web. Útočník v tomto případě nějakým způsobem donutí oběť navštívit svůj web (odkaz v emailu) a odeslat bez

vědomí uživatele sebou upravený formulář na server, kde byl uživatel v minulosti přihlášen. Takto upravený formulář může být odeslaný dokonce i automaticky JavaScriptem bez jakékoli interakce uživatele.

Podobnou možností je odeslání asynchronního požadavku kódem. Takový útok by však při správném nastavení neměl uspět kvůli obraně SOP (Same Origin Policy) a CORS (Cross Origin Resourse Sharing) prohlížečů. Prohlížeče implementují tuto ochranu, kdy dovolují ve výchozím stavu odesílat requesty pouze na servery se stejnou doménovou adresou. Pokud má být asynchronnímu requestu umožněno směřovat na server s jinou adresou, pak je nutné, aby takový server explicitně povoloval CORS v HTTP hlavičce Access-Control-Allow-Origin response.

Symfony chrání svou cookie vlastnostmi HttpOnly a SameSite (Obrázek 18). Vlastnost HttpOnly zakazuje přístup clientskému kódu (JavaScriptu), čímž ji chrání před ukradnutím při útoku XSS (Cross Site Scriptu). Dále má nastavenou vlastnost *SameSite=lax,* která zajistí, že se tato cookie neposílá z webů jiných domén, čímž zabraňuje jejímu zneužití při útoku CSRF. [77]

Obrázek 18 Vlastnosti Symfony cookie

Name	Value	Domain	Path	Expires / Max-A...	Si...	HttpOnly	Secure	SameSite
PHPSESSID	c3n1v13h0vk...	localhost	/	Session	35	✓	✓	Lax

Framework však nespoléhá pouze na SameSite ochranu a volitelně poskytuje další vrstvu ochrany formulářů proti CSRF. Tato ochrana přichází vhod především pro případ starších prohlížečů nepodporující SameSite cookies. Ochrana funguje na principu Naive Double-Submit Cookie Pattern. Symfony si do cookies při každé komunikaci uloží náhodný řetězec a stejný přidá i do vygenerovaného clientského kódu formuláře jako skrytý input typu *hidden* (Fragment 10). Při odeslání dat formuláře na server jsou oba řetězce porovnány a pokud jsou stejné, tak formulář ochranou projde (Fragment 11). Tím je zřejmé, že formulář pochází se stejného serveru. [78]

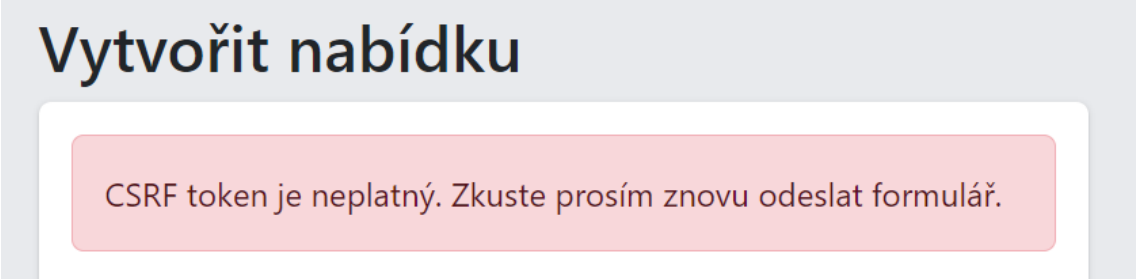
Fragment 10 Input s CSRF tokenem vygenerovaný Symfony

```
<input type="hidden" id="create_update_job__token"
name="create_update_job[_token]"
value="924e91202b7ffc7af2bc2c.laNVpzIhExKI3_qQa0AxRZh40OJNVJFQHy16M0jH
GUA.pPET6QBuQyrOhr7WXi17HPYL5IYhA6E5a3QTcj-WIQPw9zbhSldUaMqplQ">
```

Fragment 11 Zdrojový kód Symfony porovnávající hodnotu ze session cookie a z formuláře

```
return hash_equals($this->storage->getToken($namespacedId), $this-
>derandomize($token->getValue()));
```

Obrázek 19 Chybová hláška při chybějícím či nevalidním CSRF tokenu



2.12.4 Zabezpečení proti XSS

Vstup formulářů je jednou z možností, jak může dojít k útoku XSS (Cross Site Scripting), kdy útočník vloží do vstupu svůj kód, a ten může být při nevhodném zacházení vykonaný, ať už na serveru, nebo v prohlížeči při návštěvě webové stránky, která by obsah tohoto pole vypisovala třeba jako nadpis. Škodlivý kód útočníka poté může dělat cokoli, co je kódu umožněno.

Jedna z možností ochrany proti XSS je použití šablon a implicitního ošetřování vypisovaných dat na stránku (kapitola 2.11). Ta však řeší situaci po tom, co se již nebezpečný kód do systému dostal.

Symfony nabízí možnost preventivně očistit vstupy formulářů nastavením atributu *sanitize_html=true* u definic prvků formuláře. Při použití této možnosti by měl být výsledný kód bezpečným HTML. Algoritmus funguje na bázi návrhu standardu - HTML Sanitizer W3C Standard Proposal. [79]

Autor práce chtěl tuto možnost využít u všech vstupů, avšak ukázalo se, že to není možné. Sice byl škodlivý kód jako *<script>destroyPage();</script>* správně odstraňován ze vstupů, avšak problém nastal u vstupů typu jako je email. V takovém případě byl symbol zavináče @ kódován do odpovídající bezpečné HTML entity @ - taková adresa je však nevalidní pro odesílání emailů či zobrazení na webu. Tato funkce by se tedy musela zapnout pouze jen u některých vstupů, nebo zapnout u všech a dořešovat tyto případy

individuálně. Kvůli konzistenci se autor rozhodl prozatím funkci nevyužít, ale může být v budoucnu kdykoli poměrně snadno aktivována. [79]

2.13 Upload souborů

V práci bylo potřeba u požadavku na nahrátí loga inzerenta vyřešit nahrávání souborů na server. Symfony poskytuje základní podporu pro nahrávání souborů, avšak pro elegantní implementaci byla doinstalována knihovna *vich/uploader-bundle*, která navíc poskytuje integraci s ORM/ODM Doctrine, typ formulářového pole a další funkcionalitu.

Nejprve je potřeba vytvořit definici mapování v konfiguračním souboru *vich_uploader.yaml* (Fragment 12). V mapování se nastavuje, kam se bude uploadovaný soubor ukládat, jakým způsobem se pojmenuje, jak se bude vytvářet URL k tomuto souboru, zda se má při odstranění databázového objektu smazat i soubor a další konfigurace. [80]

Fragment 12 Konfigurace knihovny VichUploaderBundle

```
vich_uploader:
    db_driver: mongodb

    mappings:
        advertiserLogo:
            uri_prefix: /upload/advertiser-logo
            upload_destination:
'%kernel.project_dir%/public/upload/advertiser-logo'
            namer: App\Uploader\AdvertiserLogoNamer
            inject_on_load: true
            delete_on_update: true
            delete_on_remove: true
```

Následně se použíjí anotace z knihovny v databázové entitě (Fragment 13).

Fragment 13 Použití anotací v databázové entitě

```
#[Vich\Uploadable()]
class Advertiser
{
      #[Vich\UploadableField(mapping: 'advertiserLogo',
fileNameProperty: 'logoName')]
      private ?File $logoFile = null;
```

Databázovou entitu je potřeba označit anotací *Vich\Uploadable*. Dále se označí anotací *Vich\UploadableField* vlastnost třídy, která má reprezentovat nahraný soubor. Tyto anotace způsobí zaregistrování Doctrine event listenerů, kteří naslouchají na databázové události jako je uložení či odstranění dokumentu. Tyto eventy jsou důležité pro funkcionalitu synchronizace mezi databázovým záznamem a soubory. Díky událostem je možné aktivovat automatiku, která odstraní nahraný soubor při odstranění entity z databáze, nebo přehrání jiným souborem. Jde o transparentní a komfortní způsob práce s nahranými soubory.

Anotace *UploadableField* ve Fragment 13 říká, aby se použila konfigurace mapování s identifikátorem *advertiserLogo*, a že pole dokumentu v databázi se má jmenovat *logoName* (

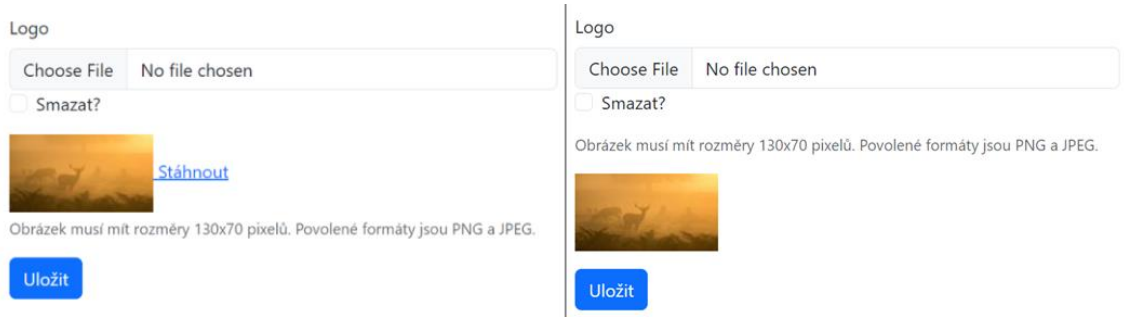
Fragment 14) – v databázi se bude držet pouze název souboru. Property třídy *$logoFile* je naplňována instancí třídy File, která oproti názvu souboru poskytuje více dat a metod.

Fragment 14 Reprezentace nahraného souboru v dokumentu v databázi

```
{
        logoName: 647e1d7d1d104edf1fec4647.jpg
        …
}
```

Knihovna nabízí i komponenty do formulářového systému Symfony. Generování jejich HTML bylo lehce modifikováno v souboru *_my_form_theme.html.twig* viz Obrázek 20.

Obrázek 20 HTML formulářového prvku pro upload obrázku (vlevo před úpravou, vpravo po úpravě)



2.14 Emaily

Na různých místech aplikace dochází k odeslání emailu. Může jít o zájem o nabídku inzerenta, notifikační email adminovi o nové nabídce ke schválení apod.

Možnosti jak řešit odeslání emailů byly identifikovány dvě:

1. Odesílat email přímo z kódu.

2. Odesílat email prostřednictvím služby 3. strany.

Odesílání emailu prostřednictvím služby 3. strany může probíhat použitím různých transportních protokolů, třeba SMTP nebo HTTP. Výhodou použití těchto služeb je, že mývají vyvinutou komplexní službu kolem posílání emailů; dokáží zobrazovat statistiky; monitorovat (ne)doručitelnost emailů; implementují opakované pokusy o doručení; udržují dobrou pověst jejich serverů, takže by zprávy neměly končit ve spamu apod. Nevýhodou je nutnost tyto služby platit. Z ekonomických důvodu v této práci tyto služby nebyly využity.

Kromě služeb 3. stran bylo možné odesílat emaily přímo z kódu:

1. PHP funkcí *mail*, která na pozadí používá program *sendmail*, případně protokol SMTP. Program *sendmail* musí být nainstalován a může být již v základu předinstalován v distribuci OS. [81].

2. Prostřednictvím Symfony služby *Mailer*. Výhodou této služby je komfortní rozhraní pro použití a možnost v budoucnu snadno přepínat mezi různými transporty a protokoly upravením jediného řádku konfigurace.

V této práci byl pro posílání emailů použit způsob odesílání prostřednictvím Symfony služby. Po nasazení se předpokládá připojení univerzitní emailové schránky protokolem SMTP.

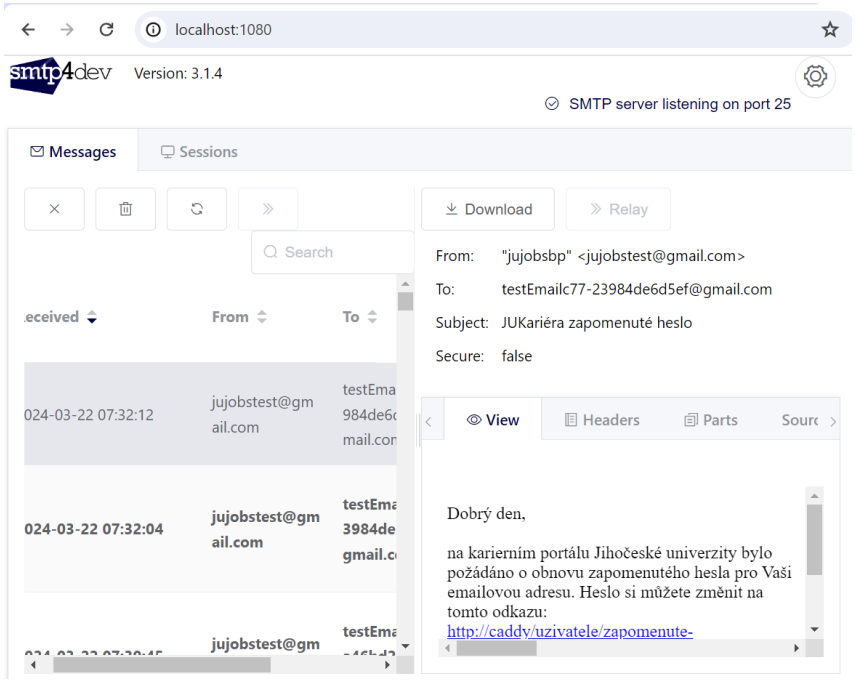
Při lokálním vývoji není zájem, aby emaily odcházely do schránek adresátů. Lze očekávat, že se při vývoji používají emaily pouze pro testování, a není vhodné adresáty spamovat. Z tohoto důvodu byla služba Mailer na lokálním vývojovém prostředí nasměrována na Docker kontejner s obrazem rnwood/smtp4dev (Fragment 15Fragment 16). Tento kontejner pak zachytává emaily na smtp://0.0.0.0:1025 a poskytuje webové UI pro prohlížení emailů na http://localhost:1080 (

Obrázek 21). Služba podporuje protokol IMAP, díky kterému lze v testech získat obsah emailů a testovat tak např. úplný proces registrace uživatele včetně ověření emailové adresy.

Fragment 15 Soubor .env umožňující nastavit prostředek transportu pro odesílání emailů – v tomto případě nasměrováno na Docker kontejner se službou smtp4dev.

```
###> symfony/mailer ###
MAILER_DSN=smtp://smtp4dev:1025
###< symfony/mailer ###
```

Obrázek 21 Uživatelské rozhraní služby smtp4dev



Odesílání emailů lze konfigurovat v souboru *mailer.yaml* (Fragment 16). Zde je možné konfigurovat základní emailovou adresu, která je použita pokud při odeslání není specifikována odchozí adresa. Dále je možné nastavit adresu administrátorů, na kterou má aplikace posílat zprávy jako je oznámení o nových nabídkách ke schválení.

Fragment 16 Soubor mailer.yaml pro konfiguraci emailů

```
framework:
    mailer:
        # use email transport defined in .env
        dsn: '%env(MAILER_DSN)%'
        envelope:
            # default email address used if „from“ is not set
            sender: 'jujobstest@gmail.com'
parameters:
```

```
    # email address of admins to send notifications to
    notifications_email: 'startuju@jcu.cz'
```

Seznam emailů, které jsou z aplikace posílány:

- Ověření vlastnictví emailové adresy inzerenta.
- Ověření vlastnictví osobní emailové adresy studenta.
- Ověření vlastnictví nové emailové adresy po změně uživatelem.
- Oznámení administrátorům o nové nabídce ke schválení.
- Oznámení administrátorům o upravené nabídce ke schválení.
- Oznámení inzerentovi o schválení nabídky.
- Oznámení inzerentovi o zamítnutí nabídky.
- Oznámení inzerentovi o zájmu studenta o nabídku.
- Obnova zapomenutého hesla s odkazem na možnost resetovat heslo.
- Oznámení studentům o nových nabídkách.

2.15 Stránkování

Byla implementována základní optimalizační technika stránkováním záznamů. Díky načítání dat tzv. po stránkách lze omezit množství dat v paměti či v přenosu a urychlit tak vykonání kódu, zrychlit načítání stránek, nebo udělat zobrazení v prohlížeči plynulejší.

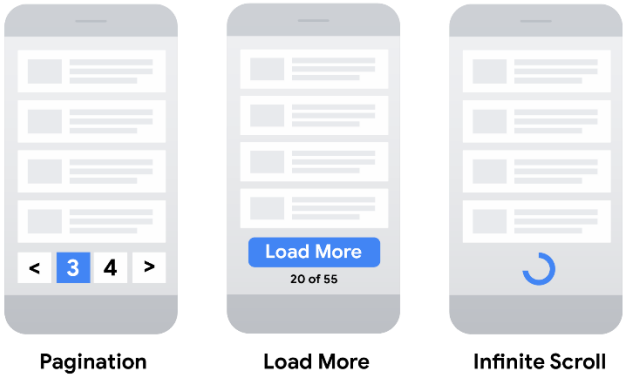
Z uživatelského hlediska lze optimalizaci omezení rozsahu dat, se kterými se pracuje, rozdělit na 3 typy:

1. Stránkování: Stránky dat se přepínají pomocí číselných tlačítek, nebo tlačítky „další“ a „předchozí“. Uživateli je vždy zobrazena pouze jedna stránka výsledků. [82]

2. Načíst další: Kliknutím na tlačítko „načíst další“ se rozšíří aktuálně zobrazená data (stránka) o další rozsah (stránku). [82]

3. Nekonečné rolování: Obdobné jako tlačítko načíst další z bodu 2, avšak vykonáno automaticky pozorováním dosažení konce okna aktuálně zobrazovaných dat. [82]

Obrázek 22 Typy stránkování z pohledu UX, převzato z [82]



V této práci bylo implementováno standardní stránkování z bodu 1. Jde o nejjednodušší a spolehlivou implementaci, která na rozdíl od bodu 2 a 3 nevyžaduje JavaScriptovou implementaci pro rozšíření aktuální struktury o další obsah, a nemá se tak v ní příliš co rozbít. Tato metoda již v základu podporuje SEO, protože stránkování může být implementováno prostřednictvím odkazového tagu *a*, se kterým indexovací roboti umí pracovat. SEO v případě metod implementovaných JavaScriptem je komplikovanější. Někteří indexovací roboti dokáží spouštět do určité míry JavaScript, ale stránky nejsou indexovány tak často a spolehlivě. Metoda stránkování z bodu 1 na stránce zobrazuje vždy jen omezené množství záznamů, což vede k plynulejšímu výkonu v prohlížeči a není tak pro velké množství záznamů potřeba implementovat další složité optimalizace typu virtuálních seznamů, kdy se zobrazují pouze ty položky, které se zrovna vejdou zobrazované plochy, což je nutnost především u na výkon náročnějších frontendových knihoven jako je React [83]. [82]

Stránkování má i více možností implementace na straně databáze.

Nejjednodušší varianta zahrnuje použití parametrů *limit* a *offset*, které zajistí omezení počtu výsledků a přeskočení potřebného množství stránek. Tato metoda je nevýhodná v tom, že čím více stránek je přeskočeno, tím je dotaz časově náročnější. [84] Tato varianta byla použita v této práci. Neočekává se, že by uživatelé procházeli velké množství stránek, nebo že by přeskakování mělo signifikantní vliv na výkon.

Jinou variantou by bylo využití seřazení výsledků například dle datumu vytvoření. V takovém případě se získají záznamy stránky prostřednictvím podmínky *where*, kdy se pro další stránku hledají záznamy s hodnotami následující po hodnotě posledního aktuálně zobrazeného záznamu. Není tak potřeba žádné záznamy přeskočit, protože nebudou vyhovovat podmínce. U této varianty je složitější odkazovat se na specifické stránky, když není k dispozici předcházející hodnota. Z tohoto důvodu bývají pro změnu stránky zobrazena pouze tlačítka „předchozí“ a „následující“. [84]

Implementace stránkování by měla řešit

- Renderování příslušného HTML. Běžné je označení aktuální stránky, odkaz na první a poslední stránku, několik stránek pro navigaci vpřed/vzad a signalizovat vynechané stránky. Možné jsou i jiné varianty jako výpis úplně všech stránek. Odkazy na další stránky musí zohledňovat aktuální parametry v URL, jinak se při změně stránky mohou ztratit hodnoty filtrovacího formuláře.

- Stránkování na úrovní databáze. Omezovat záznamy přidáním parametrů *limit* a *offset* na základě vybrané stránky. Počítat celkový odpovídající počet záznamů např. pro možnost přeskočit na poslední stránku.

Pro stránkování byly nainstalovány knihovny *babdev/pagerfanta-bundle, pagerfanta/doctrine-mongodb-odm-adapter, pagerfanta/twig.* Tyto knihovny integrují Doctrine a poskytují potřebnou funkcionalitu viz Fragment 17 a Fragment 18.

Fragment 17 Aplikace stránkování na databázový dotaz

```
$pager = (new Pagerfanta(
            new QueryAdapter($searchJobsQb),
        ))->setMaxPerPage(10)
          ->setCurrentPage($request->query->get('page', 1));
```

Fragment 18 Vyrenderování přepínání stránek v šabloně

```
{{ pagerfanta(pager) }}
```

2.16 Statistika

Jedním z požadavků byla statistika na počet odeslaných odpovědí firmám a počet zobrazení nabídky. Tato statistika je administrátorům k zobrazení v detailu nabídky.

- Statistika počtu odpovědí na nabídky je přesná. Při každé odpovědi na nabídku je vytvořen záznam v databázi.

- Statistika počtu zobrazení nabídky byla implementována naivně a jde tak pouze o orientační číslo, na které nelze spoléhat. Po načtení stránky a uplynutí krátké doby je odeslán požadavek na server, který zvýší počet zobrazení dokumentu nabídky v databázi o 1. Jde o základní implementaci funkce, která nebere v potaz, že by někdo škodil. Do budoucna je možno tuto implementaci nahradit sofistikovanějším řešením jako je např. analytika Google Analytics.

2.17 Bezpečnost

V této kapitole jsou shrnuty bezpečnostní opatření, některé z nich již byly zmíněny v ostatních kapitolách.

- Zpracování formulářů je chráněno proti útokům CSRF (kapitola 2.11).

- Cookie poskytující přístup k session datům na serveru je chráněna proti CSRF a ukradnutím XSS vlastnostmi HttpOnly a SameSite=lax (kapitola 2.11).

- Po uživatelích je vyžadováno úsilí, aby jejich heslo nebylo označeno za slabé dle doporučení OWASP, tedy délka alespoň 8 znaků [85].

- Hesla uživatelů jsou uložena v hashované podobě. Hashování je implementováno na straně frameworku a strategie je konfigurována v konfiguračním souboru security.yaml. Framework by měl při strategii auto používat nejbezpečnější algoritmus dostupný v systému. OWASP doporučuje použít nejlépe algoritmus Argon2id. Framework však používal pro hashování algoritmus Bcrypt, který OWASP označuje za vhodný pro starší systémy, i když byl framework schopen generovat hashe i pomocí algoritmu Argon2id. Strategie byla proto explicitně přenastavena na Argon2id, který vyžaduje knihovnu sodium. Framework navíc nabízí možnost automaticky při procesu přihlášení uživatele migrovat hesla na různé algoritmy hashování. Proces migrace mezi algoritmy Bcrypt a Argon2id při

přihlášení uživatele byl úspěšně otestován krokováním kódu v debuggeru a porovnáním změny hodnot hashů v databázi. Oba zmíněné algoritmy umožňují použít tzv. salt, kdy se při vytváření hashe hesla přidává náhodná hodnota. Dvě stejná hesla vytvořená stejným algoritmem tak vedou k unikátnímu hash, což zvyšuje bezpečnost hesla vůči nějaké existující databázi dvojic nehashovaných a hashovaných hesel. Přítomnost této funkcionality byla pro jistotu také otestována porovnáním dvou hashů stejného hesla vytvořených v různý čas. [86][87]

- Aplikace neukládá tokeny systému STAG a žádá tokeny s krátkodobou životností 30 minut (kapitola 2.9).

- Aplikace neukládá citlivá data jako je obsah zprávy a přílohy odpovědi na nabídku.

- Přístup na profil studenta, který může obsahovat citlivá data, je umožněn pouze inzerentům, a to jen po udělení explicitního souhlasu studentem při odpovědi na inzerát, a na omezenou dobu 60 dnů.

- Aplikace implementuje autentizaci, a řídí přístup ke zdrojům a operacím (kapitola 2.9).

- V aplikaci byl frontendový kód tvořen prostřednictvím šablon, které přinášejí implicitní ochranu před útokem XSS, kdy jsou proměnné hodnoty při výpisu na stránku ošetřovány proti obsahu kódu (kapitola 2.11).

- Odkazy ověřující vlastnictví emailové adresy jsou digitálně podepsány serverem. Nemohou tak být zneužity k ověření jakékoli adresy [88]. Z analýzy kódu knihovny vyšlo najevo, že používá jako klíč pro podpisy hodnotu parametru *secret* z konfiguračního souboru *framework.yaml*.

- Pro vytvoření aplikace byly použity v té době aktuální verze hlavních technologií jazyku PHP a frameworku Symfony. Instalovaná verze PHP 8.2.* deklaruje bezpečnostní aktualizace do 8. Prosince 2025 a Symfony 6.4.* do listopadu 2027. Aktuální verze by měly obsahovat nejnovější opravy bezpečnostních chyb. [89][90]

2.18 Pravidelná automatika

Byly implementovány 2 PHP CLI scripty, které by měly být pravidelně spouštěny systémem např. prostřednictvím Cronu. Framework Symfony myslí i na ovládání konzolovým rozhraním a nabízí základní podporu pro tvorbu příkazů. Implementované příkazy používají generování URL, protože však tyto příkazy nejsou spouštěny v rámci HTTP kontextu, nedokáží si odvodit základní URL, a je proto nutné jim tuto URL zadat v souboru *routing.yml* v parametru *default_uri*. [91][92]

Vytvořené příkazy:

- Automatická archivace nabídek s vypršelou platností: Spouští se příkazem *php bin/console app:archive-expired-jobs*.

- Odeslání emailových oznámení o nových nabídkách studentům: Spouští se příkazem *php bin/console app:send-new-job-notifications <frequency>*. Parametr *frequency* ovládá jak staré nabídky budou považovány za nové k oznámení. Tento parametr by měl korespondovat s nastavenou frekvencí spouštění scriptu. Validní hodnoty jsou: *monthly*, *weekly*, *daily*. Musí být nastaveno pravidelné spouštění tohoto příkazu ve všech třech variantách, aby byly podpořeny všechny možnosti nastavení frekvence oznámení, které si může uživatel zvolit.

2.19 Testování

Bylo implementováno automatizované testování typu end-to-end pro některé scénáře. End-to-end testy testují v ideálním případě funkcionalitu celkovou, odpovídající, nebo blížící se reálnému použití uživatele. Na rozdíl od jiných druhů testů se nezaměřují pouze na omezenou část funkcionality. Ve své podstatě jde o zautomatizování manuálního testování. Tímto způsobem lze snadno tvořit smysluplné testy pokrývající široké množství kódu a zároveň dosáhnout vysoké míry jistoty o funkčnosti všech částí aplikace, protože se simuluje chování uživatele. End-to-end testy v ideálním případě co nejvíce simulují reálné prostředí a chování uživatele, avšak při jejich psaní lze použít množství zkratek. Například je možné se přímo připojit do databáze a připravovat si vhodně data pro jednotlivé scénáře, jako testovací uživatele apod. Nebo lze zachytávat a podstrkávat

vlastní odpovědi na síťové požadavky aplikace. Byl použit testovací framework Playwright společnosti Microsoft umožňující:

- Definovat test a podmínky pro úspěch testu.
- Spouštět testy paralelně či sériově, a to i v grafickém rozhraní umožňující vizuální ladění testu.
- Programově ovládat a kontrolovat různé prohlížeče, jejich dialogová okna, rozměr zobrazení. zachytávat a pozměňování HTTP request a response, atd.
- Shromažďovat data o průběhu testů a později je ve webovém nástroji vizualizovat.
- A další funkce.

Testování scénářů bylo zaměřeno na běžný úspěšný průchod, bez testování všech možných chybových kombinací. Ostatní scénáře byly testovány pouze manuálně. Byly napsány testy pro následující scénáře.

- Registrace inzerenta.
- Dokončení registrace inzerenta ověřením emailu.
- Přihlášení inzerenta.
- Odhlášení inzerenta.
- Resetování zapomenutého hesla prostřednictvím odkazu v emailu.
- Odstranění uživatelského účtu.
- Změna emailu přihlášeného uživatele včetně ověření nové emailové adresy.
- Změna hesla přihlášeného uživatele.
- Možnost inzerenta upravit svou stránku.
- Možnost administrátora upravit stránku ostatních inzerentů.
- Vytvoření pracovní nabídky inzerentem.

- Upravení pracovní nabídky inzerentem.
- Vytvoření pracovní nabídky inzerenta administrátorem.
- Upravení pracovní nabídky inzerenta administrátorem.
- Registrace studenta prostřednictvím STAG.
- Ověření vlastnictví osobní emailové adresy studenta a dokončení registrace.
- Přihlášení studenta emailem a heslem.
- Odhlášení studenta.
- Přihlášení studenta prostřednictvím STAG.
- Importování studia studenta ze STAG.
- Importování kvalifikačních prací studenta ze STAG.
- Možnost studenta upravit svůj profil.
- Manuální vytvoření, editace, odstranění studia a označení jako studentem upravené.
- Manuální vytvoření, editace, odstranění kvalifikační práce a označení jako studentem upravené.
- Profil studenta není přístupný nepřihlášeným uživatelům.
- Profil studenta není přístupný inzerentům.
- Profil studenta není přístupný ostatním studentům.
- Odpověď na inzerát. Pokud student v odpovědi inzerentovi povolí přístup na svůj profil, je profil studenta danému inzerentovi přístupný.
- Formulář zapnutí emailových notifikací na nové nabídky studentem.
- Formulář vypnutí emailových notifikací na nové nabídky studentem.

Testy by měly být v ideálním případě rychle provedené, navzájem se neovlivňovat a být opakovatelné se stejným výsledkem.

Existuje několik technik, jak vlastností nezávislosti a opakovatelnosti docílit:

- Kontrolovat průběh a pořadí testů. Závodní podmínky mezi testy lze vyřešit přepnutím z paralelního vykonávání na sériové.

- Izolovat jednotlivé testy a nesdílet zdroje mezi nimi. Míra izolace může být různá, pro jednotlivé testy mohou být vytvářeny pouze individuální data (např. uživatel), ale i celá vlastní uložiště (databáze).

- Uvádět prostředí mezi testy do stejného stavu. Mazat zdroje vytvořené testy, naplňovat datová uložiště identickými daty…

- Definovat test a podmínky pro jeho úspěch specificky s ohledem na nezávislost a opakovatelnost. Např. generovat unikátní identifikátory rozlišující zdroje jednotlivých testů.

V práci byla použita kombinace zmíněných technik. Testy v této práci nevyužívají individuální datové uložiště, ale sdílejí stejnou databázi.

Při psaní testů byl částečně využit návrhový vzor PageObjectModel. Pro webové stránky aplikace jsou vytvořeny třídy, které je reprezentují. Ty mohou obsahovat metody popisující proveditelné akce dané stránky a vlastnosti reprezentující různé prvky na dané stránce, např. tlačítka. Tento vzor umožňuje vytvořit pro testování API vyšší abstrakční úrovně a zlepšit znovu použitelnost kódu. [93]

U scénářů vyžadujících použití email byl využit protokol IMAP k přístupu k emailům, které ve vývojovém prostředí před odesláním skutečným adresátům zachytávala služba Smtp4dev. Tímto způsobem bylo možné ověřit v plném rozsahu i funkcionalitu zahrnující emaily, jako je registrace uživatele - včetně ověření vlastnictví emailové adresy, kdy byla ze zachyceného emailu získána URL adresa pro ověření regulárním výrazem.

Některé testy v této práci využívají přímého přístupu do databáze, nebo plní databázi daty pro rychlost a jednoduchost uvedení systému do požadovaného stavu. Při spouštění je

možné zadat připojení do databáze enviromentální proměnnou *DB_CONNECTION_STRING*.

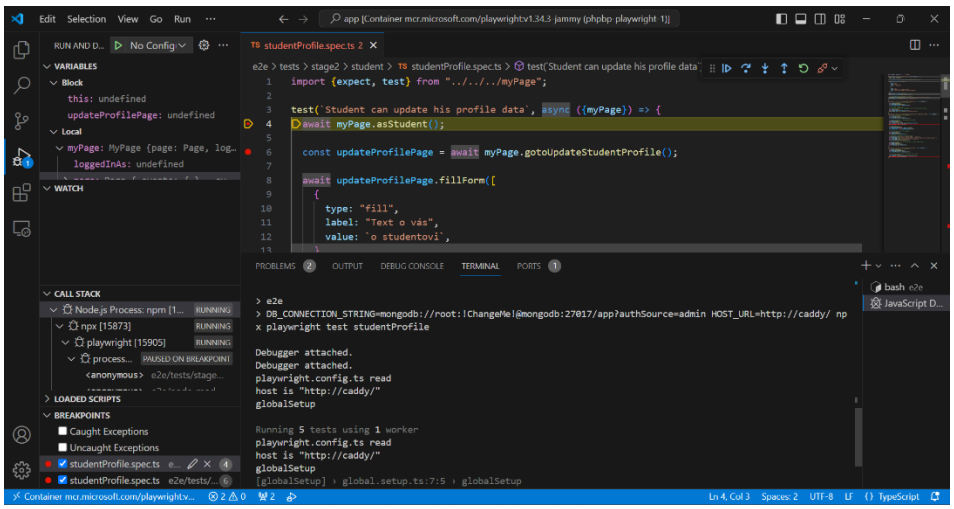
Scénáře testující funkce STAG vyžadují předání přístupových údajů do STAG environmentálními proměnnými *UNI_USERNAME*, *UNI_EMAIL*, *UNI_PASSWORD*.

Testy se nachází ve složce *e2e*. Soubor *packages.json* specifikuje knihovny používané v testech. Soubor *readme.md* ukazuje příkazy ke spuštění. Předpokládá se spouštění testů v čisté databázi.

2.19.1 Ladění aut. testů

Automatické testy lze krokovat debuggerem. Playwright je založený na technologii Nodejs, která má již debugger v sobě a nemusí být proto instalován.

Obrázek 23 Debuggování automatických testů Playwright v editoru VSCode



2.20 Uživatelské testování

2.20.1 Testování role inzerenta a studenta

Bylo provedeno uživatelské testování aplikace krátce před odevzdáním práce za přítomnosti autora s cílem získat zpětnou vazbu. Testování se účastnili tři účastnící ve věku od 26 do 31 let. Všichni účastníci měli dokončené vysokoškolské vzdělání na JČU a v průběhu svého studia pracovali. Dva z nich v době provádění testu stále studovali a při studiu pracovali, jeden se již plně věnoval zaměstnání. Účastníci pracovali v oblasti

informačních technologií, sociálních služeb a politiky. Účastníci byli jeden po druhém požádáni o vcítění se do role imaginárního, nebo současného zaměstnavatele, a poté do role studentů JČU. Následně byly požádáni o vykonání vybraných scénářů užití aplikace a byly zaznamenány jejich komentáře. Nakonec mohli aplikace volně zkoušet. Testovaná aplikace byla spuštěna v lokálním prostředí na počítači autora, účastníci otestovali scénáře na tomto počítači a následně ještě na svých mobilech – účastníci použili mobily s OS Android a iOS. Počínání účastníků bylo pozorováno a byly zaznamenány některé nápady ke zlepšení. Nakonec jim byly předloženy doplňující otázky a účastníci byli požádáni o upřímné hodnocení. Účastníci aplikaci před testováním nikdy neviděli, nebyl jim předložen uživatelský manuál, ani nebyli na práci s aplikací školeni. Protože jeden z účastníků již není studentem JČU, byl mu pro testování propůjčen existující STAG účet současného studenta.

Tabulka 4 Testování scénářů pro roli inzerenta

Vybraný scénář	Pozorování testerů	Opatření
Zaregistrujte se a přihlaste se jako inzerent pracovních nabídek.	Po přihlášení očekává přesměrování na domovskou stránku na místo stránky obsahující pouze oznámení o tom, že je přihlášený. Chybí mu možnost zobrazit heslo ve vstupu, avšak spokojuje se s ochranou před překlepem opakovaným zadáním. Při registraci udělal omylem chybu v emailu. Navrhuje přidat do hlášky o úspěšné registraci email, který byl použit.	
Vytvořte pracovní nabídku pro nového kolegu z řad studentů.	Líbí se mu volná možnost zadávat do některých vstupů libovolné hodnoty.	
	Prohlížeč mu podtrhává hodnoty vstupů červeně kvůli nastavenému jazyku pravopisu, jako by byly chybné.	Bylo způsobeno testováním v anonymním okně. Kontrola pravopisu je závislá na nastavení prohlížeče.
	Chybí mu u inzerátu vstup na možnost zadat smlouvu na dobu určitou nebo neurčitou.	
Upravte profil své firmy.	Ocenil by, aby bylo řešeno zmenšování obrázků již na straně aplikace.	

	Nedaří se mu zadat odkaz na svou webovou stránku, ocenil by příklad správné hodnoty vstupu.	Přidána vzorová hodnota pro vyplnění.
Zareagujte na zájem od studenta, který později projevíte při testování scénářů studenta.		

Tabulka 5 Testování scénářů pro roli studenta

Vybraný scénář	Pozorování testerů	Opatření
Zaregistrujte se a přihlaste se jako student prostřednictvím STAG.	Ptá se na možnost být student a inzerent zároveň.	Vysvětleno, momentálně nelze. Takový případ je možné nyní řešit pomocí 2 různých účtů.
	Obešel pravidlo aplikace zakazující použít univerzitní email jako soukromý.	Opraveno.
Vyplňte svůj studentský profil, importujte si současné studium a kvalifikační práce ze STAG.	Nerozumí proč by měla být možnost upravit importované záznamy.	Vysvětleno vzhledem k možné ztrátě přístupu do STAG.
	Chce udělat administrátorská tlačítka výraznější, vyplněné barvou, že je přehlédl, na místo současných průhledných s pixelovým okrajem.	
Doplňte na svůj profil studium a kvalifikační práci z jiné univerzity.	Poukazuje na omezený výčet hodnot obsahující hodnoty vztahující se k JČU, např. nemůže zadat fakultu, která není také na JČU.	Opraveno. Z výčtů udělány volná textová pole.
Najděte nabídku vytvořenou při testování scénářů inzerenta a odešlete zájem o nabídku.	Poukazuje na to, že se nabídky neřadí správně od nejmladších.	Opraveno.
	Není mu zřejmé, že zájem o nabídku inzerentovi přijde emailem. Hledá, a ptá se na přehled odpovědí na nabídky.	Upravena textace.

	Navrhuje zlepšit textaci vstupu z „Udělit přístup na 60 dnů a připojit odkaz na svůj studentský profil“ na „Připojit odkaz na můj studentský profil s přístupem pro tohoto inzerenta na 60 dnů.“	Upravena textace.
Zapněte si emailová oznámení na nové nabídky.	Navrhuje přejmenovat notifikace na „Emailová oznámení“.	Upravena textace.
	Obtížně pochopil možnost omezení výsledků filtrem.	Stránka upravena.

Při testování vybraných scénářů nebyla objevena žádná závažná chyba.

Tabulka 6 Hodnocení

Hodnocení jako ve škole (1 nejlepší až 5 nejhorší)	Vzhled na PC	Vzhled na Mobilu	Celkový uživ. zážitek na PC	Celkový uživ. zážitek na mobilu
Tester #1	1	1	1	1
Tester #2	2	1	2	2
Tester #3	2	1-	2	2

V hodnocení byli testeři spokojenější se vzhledem mobilní verze než se vzhledem verze počítačové, což dává smysl vzhledem k tomu, že byl vzhled aplikace na desktopu vytvářen s důrazem na snadné převedení do zobrazení pro menší obrazovky. Celkový uživatelský zážitek se mezi použitím na počítači a použitím v mobilu nelišil, tzn. nedošlo k situaci, kdy by se portál na mobilu hůře ovládal, bylo něco překryto, nebo se vzhled tzv. rozsypal.

Tabulka 7 Doplňující otázky

	Tester #1	Tester #2	Tester #3
Pracoval jste při studiu?	Ano	Ano	Ano
Hledal jste při studiu práci na internetu?	Ne	Ne	Ne
Dává Vám z pohledu inzerenta smysl, aby univerzita provozovala tuto službu?	Ano. Ve svém zaměstnání vyhledává cíleně studenty na nové pozice. Spolupracuje s kariérními centry a univerzitou při	Ano	Ano

	náboru nových zaměstnanců.		
Dává Vám z pohledu studenta smysl, aby univerzita provozovala tuto službu?	Ano	Ano	Spíše ano
Dává Vám smysl možnost vyplnit profil studenta a sdílet ho s inzerenty?	Ano.	Ano, ale HR inzerentů asi nebude tušit, jak moc jsou tato data pro danou pozici relevantní.	Ano
Máte návrh na nějakou další funkcionalitu?	Přehled odpovědí na nabídky v portálu.	Možnost kontaktovat firmy jinými kanály než je email (Skype, apod.).	Přehled odpovědí na nabídky v portálu. Hodnocení inzerentů studenty.
Působil na vás web dojmem jako by patřil k Jihočeské univerzitě?	Ano	Ano	Ano

Účastnici testování se v Tabulka 7 jednoznačně shodují na smysluplnosti provozování portálu inzerce práce univerzitou, avšak ani jeden z nich při studiu práci po internetu nehledal, i když všichni při studiu pracovali. Po požádání o rozvedení odpovědi na toto téma účastníci uvedli, že práce získali jinými způsoby a smysluplnost argumentovali především příklady specifického hledání studentů jimi, jejich zaměstnavateli a dokonce i blízkými. Pokud by se rozšířila skupina testerů, tak by se jistě našel student, který internet pro hledání práce při studiu použil, neboť i autor práce je takovým studentem.

Žádanou funkcí, které se dotklo i testování scénářů, je přehled odpovědí na nabídky. Taková funkcionalita nebyla autorem zprvu implementována schválně, a to především proto, že obsah zprávy považoval za citlivý, navíc neprvotní komunikace probíhá již bez vědomí aplikace, ta by proto mohla ukazovat pouze přehled prvotních zpráv. Na základě zpětné vazby byla tato funkce doimplementována. Zapracovány byly i některé další připomínky.

2.20.2 Testování všech rolí portálu členy KC

Práce byla pro otestování členy KC nasazena na virtuální server služby EC2 v cloudu poskytovatele AWS. KC mělo možnost testovat aplikaci bez účasti autora ve stavu

implementované většiny funkcionality. Před testováním byl sdílen uživatelský manuál. Z testování vzešly připomínky z nichž většina byla zapracována. Významné připomínky byly: neumožňovat administrátorům reakci na nabídky; možnost v reakci na inzerát poslat více souborů (cv, certifikát, …); zobrazovat nabídky na hlavní stránce. Zbylé připomínky nebyly reprodukovatelné, nebo se týkaly zlepšení intuitivnosti a úprav textace.

3 Závěr a diskuze

V práci byly implementovány všechny specifikované funkční požadavky uvedené v kapitole 2.1.1 vycházející ze zadání práce a konzultací. Mimo specifikované požadavky byla na základě uživatelského testování implementována funkcionalita přehledu odpovědí na nabídky pro studenty a inzerenty.

Práce obsahuje a naplňuje cíle a úkoly stanovené v kapitole 1.2. Byly provedeny analýzy existujících řešení soukromých subjektů, univerzit a programových řešení mapující problematiku inzerce pracovních nabídek na internetu a její různorodá řešení.

Poté byly vybrány vhodné technologie, stanovena architektura systému a portál naprogramován. Prvotní výběr technologií, provedený především na základě popularity a modernosti, byl neúspěšný a projekt byl pro jeho negativní vlastnosti, primárně složitost a objemnost, přepsán do tradičnějších technologií.

Po naprogramování řešení byly napsány automatizované testy automatizující manuální testování a imitující uživatele pro zvýšení spolehlivosti aplikace a sloužící především pro ověření funkčnosti částí aplikace po případném budoucím vývoji.

Nakonec bylo provedeno uživatelské testování aplikace členy KC a potencionálními uživateli. Uživatelské testování ověřilo naplnění požadavků, intuitivnost a spokojenost se vzhledem a uživatelským zážitkem na různých zařízeních. Významné připomínky ze zpětné vazby byly zapracovány. Nebyly však zapracovány úplně všechny připomínky a nápady na vylepšení např. zmenšení obrázků na straně aplikace či některé úpravy textace.

Aplikace je z pohledu autora ve stavu vhodném k nasazení na produkční prostředí a zahájení provozu. Bohužel se nepodařilo do odevzdání práce zajistit univerzitní servery a aplikaci nasadit.

Byla také vytvořena uživatelská dokumentace shrnující funkce portálu z pohledu uživatelů. Důležité technické oblasti jako je testování dokumentuje text této práce.

Do budoucna lze aplikaci různými způsoby vylepšovat a rozšiřovat. Z pohledu rozšíření se nabízí např. implementace komerčního modelu po tom, co aplikace získá zájem a významnou uživatelskou základnu. Bude potřeba udržovat technologický dluh. Je možné rozšiřovat automatizované testy a zlepšovat kvalitu kódu, protože ne vždy byly stanovené zásady vývoje dodrženy. Dále se nabízí např. integrace analytiky Google Analytics pro pokročilou a věrohodnou statistiku návštěvnosti. Lze také vylepšit vyhledávání nabídek.

S případným rostoucím provozem lze také očekávat potřebu optimalizovat výkon některých stránek aplikace. Práce se zabývala pouze základní optimalizací formou stránkování výsledků. Další optimalizace zahrnují vytváření indexů databáze, cachování (HTTP cache) či škálování částí sytému.

4 Seznam použitých zdrojů

Bibliografie

[1] Dnešní mládež se práce nebojí, na brigády chodí častěji než jejich rodiče. In: *Equa bank* [online]. [cit. 2022-05-30]. Dostupné z: https://www.equabank.cz/n/dnesni-mladez-se-prace-neboji-na-brigady-chodi-casteji-nez-jejich-rodice

[2] PARWATI, Kardina Yudha. HOW DOES A UNIVERSITY OF CHOICE COME TO STUDENTS'MIND? FROM THE ASPECT OF THE E-SERVICESCAPE OF UNIVERSITY'S WEBSITE. *Manajemen Bisnis*. 2020, (10.1), 90-99.

[3] THE EMERGING DIGITAL ECONOMY. *U.S. Department of Commerce* [online]. [cit. 2022-06-01]. Dostupné z: https://www.commerce.gov/sites/default/files/migrated/reports/emergingdig_0.pdf

[4] O společnosti LinkedIn. *O společnosti LinkedIn* [online]. © 2024 [cit. 2024-04-06]. Dostupné z: https://about.linkedin.com/cs-cz?lr=1

[5] A Guide to Sharing for Webmasters. In: *Meta for developers* [online]. [cit. 2022-06-03]. Dostupné z: https://developers.facebook.com/docs/sharing/webmasters/

[6] Search Engine Market Share Czech Republic. In: *Statcounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share* [online]. [cit. 2022-06-02]. Dostupné z: https://gs.statcounter.com/search-engine-market-share/all/czech-republic/2021

[7] Understanding page experience in Google Search results. In: *Google developers* [online]. [cit. 2022-06-02]. Dostupné z: https://developers.google.com/search/docs/advanced/experience/page-experience

[8] Why does speed matter? In: *Google Developers* [online]. [cit. 2022-06-02]. Dostupné z: https://web.dev/why-speed-matters/

[9] O nás. In: *KC - Home-KC* [online]. [cit. 2023-09-03]. Dostupné z: https://www.kc.jcu.cz/cz/karierni-centrum-ju

[10] Jihočeská univerzita v Českých Budějovicích - Volná místa. *Jihočeská univerzita v Českých Budějovicích - Úvod* [online]. © 2021 [cit. 2024-04-04]. Dostupné z: https://www.jcu.cz/cz/univerzita/volna-mista

[11] Volná místa - Přírodovědecká fakulta JU. *PŘF - Přírodovědecká fakulta JU* [online]. © 2024 [cit. 2024-04-04]. Dostupné z: https://prf.jcu.cz/cz/fakulta/volna-mista

[12] Nabídky pracovních příležitostí - Přírodovědecká fakulta JU. *PŘF - Přírodovědecká fakulta JU* [online]. © 2024 [cit. 2024-04-06]. Dostupné z: https://prf.jcu.cz/cz/fakulta/katedry/katedra-informatiky/nabidky-pracovnich-prilezitosti

[13] KC - Pracovní příležitosti. *KC - Home-KC* [online]. © 2021 [cit. 2024-04-04]. Dostupné z: https://www.kc.jcu.cz/cz/pracovni-prilezitosti

[14] Firma Jihočeská univerzita v Českých Budějovicích – Jobs.cz – Aktuální nabídka práce od zaměstnavatelů i personálních agentur. *Jobs.cz – skvělá šance na skvělý job – nabídka práce, volná pracovní místa, brigády i vzdělávání a rozvoj* [online]. [cit. 2024-04-06]. Dostupné z: https://www.jobs.cz/prace/?company=237895525

[1 Prace.cz - Technik/Technička - Jihočeská univerzita v Českých Budějovicích.
5] *Prace.cz - volná pracovní místa, aktuální nabídka práce v ČR i v zahraničí* [online]. [cit. 2024-03-03]. Dostupné z: https://www.prace.cz/nabidka/2000158338/?rps=77&searchId=8c4cc73a-1b61-4c83-884b-091c7584c66b

[1 Technik/Technička. *Jobs.cz – skvělá šance na skvělý job – nabídka práce, volná*
6] *pracovní místa, brigády i vzdělávání a rozvoj* [online]. [cit. 2024-03-03]. Dostupné z: https://www.jobs.cz/rpd/2000158338/?searchId=8e68945b-531a-4eea-a72f-aea23407675b&rps=233

[1 Technička. *Úřad práce ČR* [online]. [cit. 2024-03-03]. Dostupné z:
7] https://www.uradprace.cz/volna-mista-v-cr#/volna-mista-detail/67033004

[1 Volná místa v ČR. *Úřad práce ČR* [online]. [cit. 2024-03-06]. Dostupné z:
8] https://www.uradprace.cz/volna-mista-v-cr

[1 Zadávání volných míst. *Úřad práce ČR* [online]. [cit. 2024-03-06]. Dostupné z:
9] https://www.uradprace.cz/zadavani-volnych-mist

[2 TOP 10 pracovních portálů a 4 agregátory. *Práce a brigády, aktuální nabídka |*
0] *JenPráce.cz* [online]. 2023, 20.10.2023 [cit. 2024-03-03]. Dostupné z: https://www.jenprace.cz/magazin/top-10-portalu-v-cr-a-4-pracovni-agregatory

[2 Prace.cz pro firmy: výběr, hledání a získávání zaměstnanců, inzerce práce i zadávání
1] brigád. *Prace.cz pro firmy: výběr, hledání a získávání zaměstnanců, inzerce práce i zadávání brigád* [online]. [cit. 2024-03-03]. Dostupné z: https://firmy.prace.cz/

[2 Hledám zaměstnance – Jobs.cz. *Hledám zaměstnance – Jobs.cz* [online]. [cit. 2024-
2] 03-03]. Dostupné z: https://firmy.jobs.cz/

[2 Hledám zaměstnance | JenPráce.cz. *Práce a brigády, aktuální nabídka | JenPráce.cz*
3] [online]. [cit. 2024-03-03]. Dostupné z: https://www.jenprace.cz/inzerenti

[2 Firmy, které se posouvají vpřed | StartupJobs.cz. *UPgrade firem | StartupJobs.cz*
4] [online]. © 2012 – 2024 [cit. 2024-03-03]. Dostupné z: https://www.startupjobs.cz/pro-firmy

[2 Ceník inzerce nabídek práce na portálu DobraPrace.cz | Dobrá práce.cz. *Práce -*
5] *Dobra Prace CZ* [online]. [cit. 2024-03-03]. Dostupné z: https://www.dobraprace.cz/cenik.php

[2 Objednat kredit (ceník) | Easy-práce.cz. *Nabídka práce a volných pracovních míst |*
6] *Easy-práce.cz* [online]. © 2006 - 2024 [cit. 2024-03-03]. Dostupné z: https://www.easy-prace.cz/cenik-inzerce

[2 Najděte nové zaměstnance na HitPrace.cz. *HitPrace.cz* [online]. © 2020 [cit. 2024-
7] 03-03]. Dostupné z: https://www.hitprace.cz/pro-firmy/

[2 Vložit inzerát práce zdarma, Hledám zaměstnance, Volnámísta.cz. *Volnámísta.cz -*
8] *volná pracovní místa, nabídka práce, jobs a zaměstnání* [online]. © 1996–2024 [cit. 2024-03-03]. Dostupné z: https://www.volnamista.cz/promo-vkladani

[2 Výsledky vyhledávání: vysoké školy. *MŠMT* [online]. [cit. 2024-04-04]. Dostupné
9] z: http://stistko.uiv.cz/proavs/pro1n.asp?typ=10&okres=CZ031+&vnazvuv=&vnazvu=&vrid=&skupina=NIC&dalsi=&text=+vysok%E9+%9Akoly&SUB=HLEDEJ

[3 Nabídka pracovních míst. *Vysoká škola technická a ekonomická v Českých*
0] *Budějovicích* [online]. © 2024 [cit. 2024-04-04]. Dostupné z: https://www.vstecb.cz/nabidka-pracovnich-mist-1569-htm/

[3 Volná pracovní místa. *Vysoká škola technická a ekonomická v Českých*
1] *Budějovicích* [online]. © 2024 [cit. 2024-04-07]. Dostupné z: https://www.vstecb.cz/volna-pracovni-mista/

[3 Aktuální nabídka pracovních míst a stáží. *Kariérní centrum VŠE – Vysoká škola*
2] *ekonomická v Praze* [online]. © 2000 - 2024 [cit. 2024-04-04]. Dostupné z: https://kc.vse.cz/aktualni-nabidka-pracovnich-mist-a-stazi/

[3 Přihlášení - Kariérní portál VŠE. *Přihlášení - Kariérní portál VŠE* [online]. © 2024
3] [cit. 2024-04-04]. Dostupné z: https://jobs.vse.cz/

[3 Job Offers / Pracovní nabídky – Alumni & Corporate Relations – Prague University
4] of Economics and Business. *Alumni & Corporate Relations – Vysoká škola ekonomická v Praze* [online]. © 2000 - 2024 [cit. 2024-04-04]. Dostupné z: https://acr.vse.cz/english/job-offers/

[3 Vyberte si z volných míst na MUNI | Masarykova univerzita. *Masarykova*
5] *univerzita* [online]. © 2024 [cit. 2024-04-04]. Dostupné z: https://www.muni.cz/o-univerzite/kariera-na-mu/volna-mista

[3 Kariérní portál JobCheckIN | Kariérní centrum MUNI. *Kariérní centrum MUNI*
6] [online]. © 2024 [cit. 2024-04-04]. Dostupné z: https://kariera.muni.cz/pro-zamestnavatele/karierni-portal-jobcheckin

[3 Kariérní centrum MU - Obchodní centrum MU. *Informační systém* [online]. [cit.
7] 2024-04-04]. Dostupné z: https://is.muni.cz/obchod/fakulta/rect/karierni_centrum/

[3 JobCheckIN. *JobCheckIN* [online]. © 2016-2017 [cit. 2024-04-04]. Dostupné z:
8] https://www.jobcheckin.muni.cz/

[3 Volná místa | Přírodovědecká fakulta MUNI. *Přírodovědecká fakulta MUNI*
9] [online]. © 2024 [cit. 2024-04-04]. Dostupné z: https://www.sci.muni.cz/kariera-na-prf-mu/volna-mista-na-prf-mu

[4 Inzerce pracovních míst | Kariérní centrum ČVUT. *Úvod CVUT | Kariérní centrum*
0] *ČVUT* [online]. [cit. 2024-04-04]. Dostupné z: https://www.kariernicentrum.cz/inzerce-pracovnich-mist/

[4 Kariérní centrum. *Úvod CVUT | Kariérní centrum ČVUT* [online]. [cit. 2024-04-04].
1] Dostupné z: https://www.kariernicentrum.cz/wp-content/uploads/2023/03/Karierni-centrum-19.pdf

[4 Nabídky práce pro studenty - FIT ČVUT. *Fakulta informačních technologií ČVUT -*
2] *FIT ČVUT* [online]. Copyright 2024 [cit. 2024-04-04]. Dostupné z: https://fit.cvut.cz/cs/spoluprace/pro-studenty/nabidky-prace/partneri-a-sponzori

[4 Jak se stát partnerem nebo sponzorem - FIT ČVUT. *Fakulta informačních*
3] *technologií ČVUT - FIT ČVUT* [online]. Copyright 2024 [cit. 2024-04-04]. Dostupné z: https://fit.cvut.cz/cs/spoluprace/pro-prumysl/jak-se-stat-partnerem-nebo-sponzorem

[4 Nabídky práce, stáží - Kariérní centrum UK. *Kariérní centrum UK* [online]. © 2024
4] [cit. 2024-04-05]. Dostupné z: https://kariernicentrum.cuni.cz/KCUK-18.html

[4 Nabídka spolupráce - Kariérní centrum UK. *Kariérní centrum UK* [online]. © 2024
5] [cit. 2024-04-05]. Dostupné z: https://kariernicentrum.cuni.cz/KCUK-23.html

[4 Nabídka pracovních pozic | Právnická fakulta UK. *Právnická fakulta UK* [online]. ©
6] 2024 [cit. 2024-04-05]. Dostupné z: https://www.prf.cuni.cz/jobs

[4 Všeobecné obchodní podmínky | Právnická fakulta UK. *Právnická fakulta UK*
7] [online]. © 2024 [cit. 2024-04-05]. Dostupné z: https://www.prf.cuni.cz/vseobecne-obchodni-podminky

[48] Nabídky práce pro studenty a absolventy. *Nabídky práce pro studenty a absolventy - Pedagogická fakulta* [online]. © 2024 [cit. 2024-04-05]. Dostupné z: https://pedf.cuni.cz/PEDF-246.html

[49] *Úvod* [online]. Úvod [cit. 2024-04-04]. Dostupné z: https://spoluprace.zcu.cz/

[50] Nápověda pro firmy. *Úvod* [online]. [cit. 2024-04-04]. Dostupné z: https://spoluprace.zcu.cz/napoveda/napoveda-pro-firmy.html

[51] Volné pozice. *Západočeská univerzita v Plzni* [online]. © ZČU 1991–2024 [cit. 2024-04-04]. Dostupné z: https://www.zcu.cz/cs/University/Career/vacancies.html

[52] Kariérní web UHK. *Kariérní web UHK* [online]. © 2021-2024 [cit. 2024-04-04]. Dostupné z: https://kariera.uhk.cz/nabidky-prace

[53] Repository search results. *Https://github.com* [online]. © 2024 [cit. 2024-03-11]. Dostupné z: https://github.com/search?q=job+list+stars%3A%3E25++pushed%3A%3E2020-01-01&type=repositories&s=stars&o=desc&p=8

[54] Repository search results. *GitHub* [online]. © 2024 [cit. 2024-03-11]. Dostupné z: https://github.com/search?q=job+board+stars%3A%3E25++pushed%3A%3E2020-01-01&type=repositories&s=stars&o=desc

[55] CMS technologies Web Usage Distribution. *BuiltWith Web Technology Usage Statistics* [online]. [cit. 2024-03-11]. Dostupné z: https://trends.builtwith.com/cms

[56] JS Jobs Free Demo. *Demo Joomsky New* [online]. [cit. 2024-04-04]. Dostupné z: https://demo.joomsky.com/js-jobs/jm/free/

[57] JS Jobs, by Joom Sky - Joomla Extension Directory. *Joomla! Extensions Directory* [online]. © 2005 - 2024 [cit. 2024-03-11]. Dostupné z: https://extensions.joomla.org/extension/ads-a-affiliates/jobs-a-recruitment/js-jobs

[58] *JoomSky: Your Destination for Cutting-Edge Solutions and Products | Joomsky* [online]. © 2024 [cit. 2024-03-11]. Dostupné z: https://joomsky.com

[59] JS Job Manager – WordPress plugin | WordPress.org. *Blog Tool, Publishing Platform, and CMS – WordPress.org* [online]. [cit. 2024-03-12]. Dostupné z: https://wordpress.org/plugins/js-jobs/

[60] INGENO, Joseph. *Software Architect's Handbook: Become a successful software architect by implementing effective architecture concepts*. Birmingham, England: Packt Publishing, 2018. ISBN 9781788624060.

[61] WebAssembly Concepts. In: *MDN Web Docs* [online]. [cit. 2022-06-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts

[62] *Symfony/symfony: The Symfony PHP framework* [online]. [cit. 2023-10-08]. Dostupné z: https://github.com/symfony/symfony

[63] Stack Overflow Developer Survey 2022. *Stack Overflow Insights - Developer Hiring, Marketing, and User Research* [online]. 2022 [cit. 2024-02-27]. Dostupné z: https://survey.stackoverflow.co/2022#section-most-popular-technologies-web-frameworks-and-technologies

[64] Dunglas/symfony-docker: A Docker-based installer and runtime for Symfony. Install: download and `docker compose up`. *GitHub* [online]. [cit. 2024-03-25]. Dostupné z: https://github.com/dunglas/symfony-docker

[6 Using Docker with Symfony. *Symfony, High Performance PHP Framework for*
5] *Web Development* [online]. [cit. 2024-03-25]. Dostupné z:
https://symfony.com/doc/current/setup/docker.html

[6 The 10 Most Common Questions IT Admins ask About Docker. *Docker:*
6] *Accelerated Container Application Development* [online]. 2016, 2016-07-27 [cit.
2024-02-02]. Dostupné z: https://www.docker.com/blog/the-10-most-common-
questions-it-admins-ask-about-docker/

[6 Pricing and licensing for Windows Server 2022. *Microsoft – Cloud, Computers,*
7] *Apps & Gaming* [online]. [cit. 2024-01-31]. Dostupné z:
https://www.microsoft.com/en-us/windows-server/pricing

[6 Xdebug: Documentation » Step Debugging. *Xdebug - Debugger and Profiler Tool*
8] *for PHP* [online]. [cit. 2024-04-04]. Dostupné z: https://xdebug.org/docs/step_debug

[6 Anemic Domain Model. *Martinfowler.com* [online]. 25 November 2003n. l. [cit.
9] 2023-03-15]. Dostupné z:
https://martinfowler.com/bliki/AnemicDomainModel.html

[7 How to Use Voters to Check User Permissions (Symfony Docs). *Symfony, High*
0] *Performance PHP Framework for Web Development* [online]. [cit. 2024-02-08].
Dostupné z: https://symfony.com/doc/current/security/voters.html

[7 IS/STAG - Zájemci. *IS/STAG - Informační systém studijní agendy* [online]. [cit.
1] 2023-10-17]. Dostupné z: https://is-stag.zcu.cz/zajemci/

[7 ZABEZPEČENÉ OPERACE, PŘIHLAŠOVÁNÍ. *IS/STAG - Informační systém*
2] *studijní agendy* [online]. [cit. 2023-11-01]. Dostupné z: https://is-
stag.zcu.cz/napoveda/web-services/ws_prihlasovani.html

[7 Why Use Templates? *Nette – Comfortable and Safe Web Development in PHP*
3] [online]. © 2008, 2024 [cit. 2024-02-25]. Dostupné z: https://latte.nette.org/en/why-
use

[7 Creating and Using Templates. *Symfony, High Performance PHP Framework for*
4] *Web Development* [online]. [cit. 2024-02-25]. Dostupné z:
https://symfony.com/doc/current/templates.html

[7 Templates. *Nette – Comfortable and Safe Web Development in PHP* [online]. ©
5] 2008, 2024 [cit. 2024-02-25]. Dostupné z:
https://doc.nette.org/en/application/templates

[7 Templates. *Laravel - The PHP Framework For Web Artisans* [online]. © 2011-2024
6] [cit. 2024-02-25]. Dostupné z: https://laravel.com/docs/5.0/templates

[7 Co jsou SameSite cookie a proč je potřebujeme? *Zdroják - o tvorbě webových*
7] *stránek a aplikací* [online]. 2020 [cit. 2024-01-16]. Dostupné z:
https://zdrojak.cz/clanky/co-jsou-samesite-cookie-a-proc-je-potrebujeme/

[7 Cross-Site Request Forgery Prevention - OWASP Cheat Sheet Series. *Introduction -*
8] *OWASP Cheat Sheet Series* [online]. [cit. 2024-01-15]. Dostupné z:
https://cheatsheetseries.owasp.org/cheatsheets/Cross-
Site_Request_Forgery_Prevention_Cheat_Sheet.html

[7 *HTML Sanitizer* [online]. [cit. 2024-02-22]. Dostupné z:
9] https://symfony.com/doc/current/html_sanitizer.html

[8 VichUploaderBundle/docs/usage.md at master · dustin10/VichUploaderBundle.
0] *GitHub* [online]. © 2024 [cit. 2024-02-22]. Dostupné z:
https://github.com/dustin10/VichUploaderBundle/blob/master/docs/usage.md

[8 Requirements. *PHP: Hypertext Preprocessor* [online]. Copyright © 2001-2024 [cit.
1] 2024-02-22]. Dostupné z: https://www.php.net/manual/en/mail.requirements.php

[8 Pagination, incremental page loading, and their impact on Google Search. *Google*
2] *for Developers - from AI and Cloud to Mobile and Web* [online]. [cit. 2024-03-19]. Dostupné z: https://developers.google.com/search/docs/specialty/ecommerce/pagination-and-incremental-page-loading

[8 Virtualize large lists with react-window. *Web.dev* [online]. [cit. 2024-03-19].
3] Dostupné z: https://web.dev/articles/virtualize-long-lists-react-window

[8 Pagination - EF Core | Microsoft Learn. *Microsoft Learn: Získejte dovednosti, které*
4] *rozšíří vaše kariérní možnosti* [online]. [cit. 2024-03-18]. Dostupné z: https://learn.microsoft.com/en-us/ef/core/querying/pagination

[8 Authentication - OWASP Cheat Sheet Series. *Introduction - OWASP Cheat Sheet*
5] *Series* [online]. [cit. 2024-05-15]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

[8 Password Hashing and Verification (Symfony Docs). *Symfony, High Performance*
6] *PHP Framework for Web Development* [online]. [cit. 2024-04-08]. Dostupné z: https://symfony.com/doc/current/security/passwords.html

[8 Password Storage - OWASP Cheat Sheet Series. *Introduction - OWASP Cheat Sheet*
7] *Series* [online]. © 2024 [cit. 2024-04-08]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

[8 SymfonyCasts/verify-email-bundle: Simple, stylish Email Verification for Symfony.
8] *GitHub* [online]. © 2024 [cit. 2024-04-08]. Dostupné z: https://github.com/SymfonyCasts/verify-email-bundle

[8 Symfony releases, notifications and release checker. *Symfony, High Performance*
9] *PHP Framework for Web Development* [online]. [cit. 2024-04-08]. Dostupné z: https://symfony.com/releases

[9 PHP: Supported Versions. *PHP: Hypertext Preprocessor* [online]. © 2001-2024
0] [cit. 2024-04-08]. Dostupné z: https://www.php.net/supported-versions.php

[9 Console Commands. *Symfony, High Performance PHP Framework for Web*
1] *Development* [online]. [cit. 2024-04-10]. Dostupné z: https://symfony.com/doc/current/console.html

[9 Routing (Symfony 7.1 Docs). *Symfony, High Performance PHP Framework for*
2] *Web Development* [online]. [cit. 2024-04-10]. Dostupné z: https://symfony.com/doc/7.1/routing.html

[9 Page object models | Playwright. *Fast and reliable end-to-end testing for modern*
3] *web apps | Playwright* [online]. [cit. 2024-03-16]. Dostupné z: https://playwright.dev/docs/pom

Seznam obrázků

Seznam fragmentů kódu

Seznam tabulek

Příloha A

Součástí přílohy A je archiv obsahující uživatelskou dokumentaci a zdrojový kód práce.