UNIVERZITA PARDUBICE

NÁZEV FAKULTY

BAKALÁŘSKÁ / DIPLOMOVÁ PRÁCE

2022 Jméno a příjmení autora práce

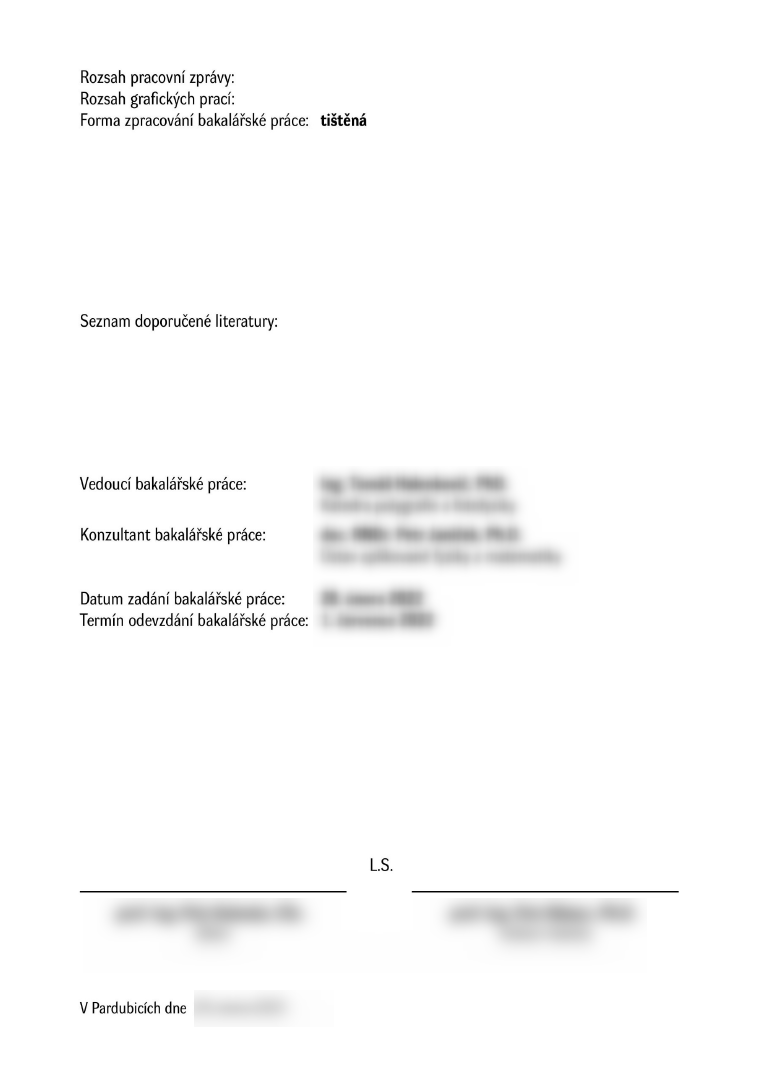
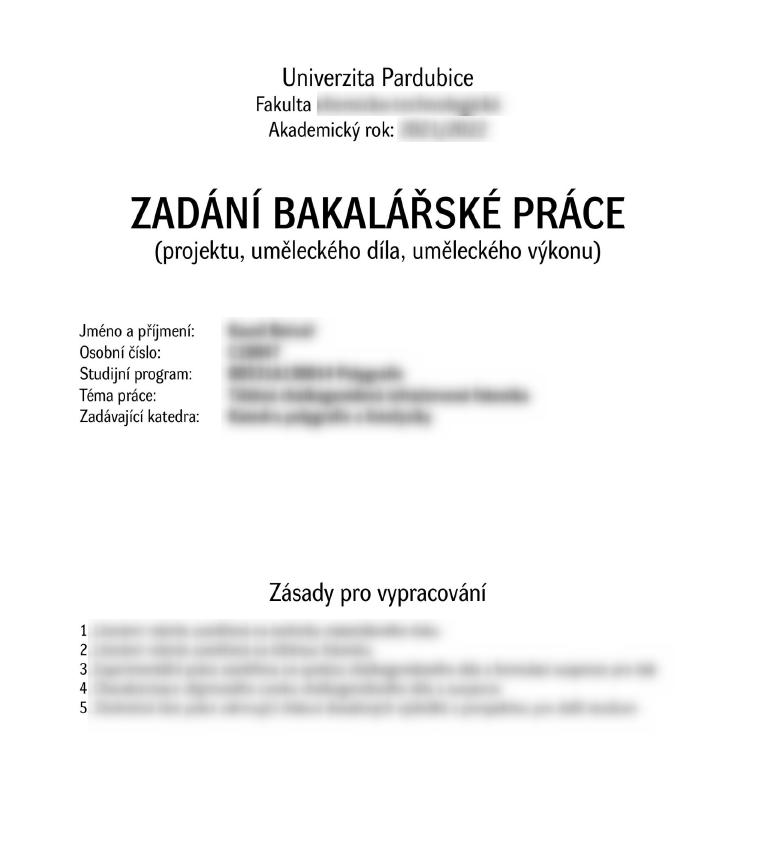
Univerzita Pardubice

Název fakulty

Název práce

Bakalářská / Diplomová práce

2022 Jméno a příjmení autora práce



Prohlašuji:

Práci s názvem ………………………………………………………………………………… jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne XX. XX. 20XX

Jméno a příjmení autora v.r.

**PODĚKOVÁNÍ**

**ANOTACE**

Červený se vedle modrého a zeleného zdá trochu menší, ale to je nejspíš jen optický klam, a i kdyby byl skutečně o něco málo menší, tak vážně jen o trošičku. Vítr skoro nefouká, a tak by se na první pohled mohlo zdát, že se balónky snad vůbec nepohybují. Jenom tak klidně levitují ve vzduchu. Jelikož slunce jasně září a na obloze byste od východu k západu hledali mráček marně, balónky působí jako jakási fata morgána uprostřed pouště.

**KLÍČOVÁ SLOVA**

slunce, vítr, balónky, barvy, pohyb, automatický generátor textu

**TITLE**

Red seems to be a little smaller next to blue and green

**ANNOTATION**

Red seems to be a little smaller next to blue and green, but that's probably just an optical illusion, and even if it's actually a little smaller, it's really just a tiny bit. The wind is barely blowing, and at first glance it might seem that the balloons are not moving at all. They just levitate in the air. As the sun shines brightly and you would be looking for a cloudless in the sky from east to west, the balloons act like a sort of mirage in the middle of the desert.

**KEYWORDS**

sun, wind, balloons, colors, motion, automatic text generator

**OBSAH**

[SEZNAM ILUSTRACÍ A TABULEK 11](#_Toc195448940)

[SEZNAM ZKRATEK A ZNAČEK 12](#_Toc195448941)

[TERMINOLOGIE 13](#_Toc195448942)

[ÚVOD 14](#_Toc195448943)

[1 Webová aplikace 15](#_Toc195448944)

[1.1. Architektura řešení při vývoji webových aplikací 15](#_Toc195448945)

[1.1.1 Architektonický vzor MVC 15](#_Toc195448946)

[1.1.2 Model-View-Presenter 18](#_Toc195448947)

[1.1.3 Model-View-ViewModel (MVVM) 19](#_Toc195448948)

[1.1.4 Model-View-ViewModel-Coordinator (MVVM-C) 20](#_Toc195448949)

[1.1.5 Monolitická architektura 20](#_Toc195448950)

[1.1.6 Mikroslužbová architektura 21](#_Toc195448951)

[1.1.7 Zhodnocení architektur 21](#_Toc195448952)

[2 Technologie používané při vývoji webových aplikací 23](#_Toc195448953)

[2.1 Backendové technologie 23](#_Toc195448954)

[2.1.1 Srovnání technologií pro backend 24](#_Toc195448955)

[2.2 Frontendové technologie 25](#_Toc195448956)

[2.2.1 Srovnání technologií pro frontend 26](#_Toc195448957)

[2.3 Relační databáze 27](#_Toc195448958)

[2.3.1 PostgreSQL 27](#_Toc195448959)

[2.4. Docker a Docker Compose 28](#_Toc195448960)

[2.5 Java Spring Boot 28](#_Toc195448961)

[2.5.1 Architektura a principy Spring Boot 28](#_Toc195448962)

[2.5.2 Výhody a přínosy použití Spring Boot 29](#_Toc195448963)

[2.5.3 Srovnání se standardním Spring frameworkem 29](#_Toc195448964)

[2.5.4 Omezení a perspektivy budoucího vývoje 29](#_Toc195448965)

[2.5.5 Z čeho se skládá Spring Boot projekt 30](#_Toc195448966)

[2.6 Vue.js 30](#_Toc195448967)

[2.6.1 Architektura a principy Vue.js 30](#_Toc195448968)

[2.6.2 Výhody a přínosy použití Vue.js 30](#_Toc195448969)

[2.6.3 Nedostatky Vue.js 31](#_Toc195448970)

[2.6.4 Z čeho se skládá Vue projekt 31](#_Toc195448971)

[PRAKTICKÁ ČÁST 32](#_Toc195448972)

[3. Vybrané komunitní platformy a intranety 32](#_Toc195448973)

[3.1 Facebook 32](#_Toc195448974)

[3.2 Reddit 32](#_Toc195448975)

[3.3 Telegram 32](#_Toc195448976)

[3.4 LinkedIn 33](#_Toc195448977)

[3.5 Instagram 33](#_Toc195448978)

[4 Analýza 34](#_Toc195448979)

[4.1 Uživatelské chování a zkušenosti 34](#_Toc195448980)

[4.1.1 Shrnutí 38](#_Toc195448981)

[4.2 Přednástupová fáze a informační bariéry 39](#_Toc195448982)

[4.2.1 Shrnutí 40](#_Toc195448983)

[4.3 Požadavky na aplikaci 42](#_Toc195448984)

[4.3.1 Shrnutí 45](#_Toc195448985)

[4.4 Porovnání odpovědi mezí rusky a česky mluvících studentu 46](#_Toc195448986)

[5. Požadavky 48](#_Toc195448987)

[5.1 Funkční požadavky 48](#_Toc195448988)

[5.2 Nefunkční požadavky 49](#_Toc195448989)

[6 Databáze 50](#_Toc195448990)

[7 Diagramy aktivit 51](#_Toc195448991)

[7.1 Hlavní diagram 51](#_Toc195448992)

[7.2 Diagram autorizace 52](#_Toc195448993)

[7.3 Diagram fóra 53](#_Toc195448994)

[7.4 Diagram bazaru 54](#_Toc195448995)

[7.5 Diagram skupin 55](#_Toc195448996)

[1.1.3 Obhajoba volby modelu Web 2.0 56](#_Toc195448997)

[ZÁVĚR 59](#_Toc195448998)

[POUŽITÁ LITERATURA 60](#_Toc195448999)

# SEZNAM ILUSTRACÍ A TABULEK

[Obrázek 1: Balonky 14](#_Toc24459062)

[Tabulka 1: Příklad tabulky 15](#_Toc116377311)

# SEZNAM ZKRATEK A ZNAČEK

# TERMINOLOGIE

# ÚVOD

Očividně je uvázaná dostatečně pevně, protože balónky skutečně neucházejí. To ale není nic zvláštního. Překvapit by však mohl fakt, že nikdo, snad krom toho, kdo balónky k obloze vypustil, netuší, jakou má ona stuha barvu. Je totiž tak lesklá, že za světla se v ní odráží nebe a za tmy zase není vidět vůbec. Když svítí slunce tak silně jako nyní, tak se stuha třpytí jako kapka rosy a jen málokdo vydrží dívat se na ni přímo déle než pár chvil. Jak vlastně vypadají ony balónky? Ptají se často lidé.

Inu, jak by vypadaly – jako běžné pouťové balónky střední velikosti, tak akorát nafouknuté. Červený se vedle modrého a zeleného zdá trochu menší, ale to je nejspíš jen optický klam, a i kdyby byl skutečně o něco málo menší, tak vážně jen o trošičku. Vítr skoro nefouká, a tak by se na první pohled mohlo zdát, že se balónky snad vůbec nepohybují. Jenom tak klidně levitují ve vzduchu. Jelikož slunce jasně září a na obloze byste od východu k západu hledali mráček marně, balónky působí jako jakási fata morgána uprostřed pouště. Zkrátka široko daleko nikde nic, jen zelenkavá tráva, jasně modrá obloha a tři křiklavě barevné pouťové balónky, které se téměř nepozorovatelně pohupují ani ne moc vysoko, ani moc nízko nad zemí. Kdyby pod balónky nebyla sytě zelenkavá tráva, ale třeba suchá silnice či beton, možná by bylo vidět jejich barevné stíny – to, jak přes poloprůsvitné barevné balónky prochází ostré sluneční paprsky. Jenže kvůli všudy přítomné trávě jsou stíny balónků sotva vidět, natož aby šlo rozeznat, jakou barvu tyto stíny mají. Uvidět tak balónky náhodný kolemjdoucí, jistě by si pomyslel, že už tu takhle poletují snad tisíc let. Stále si víceméně drží výšku a ani do stran se příliš nepohybují.

Proti slunci to vypadá, že se slunce pohybuje k západu rychleji než balónky, a možná to tak skutečně je. Nejeden filozof by mohl tvrdit, že balónky se sluncem závodí, ale fyzikové by to jistě vyvrátili. Z fyzikálního pohledu totiž balónky působí zcela nezajímavě. Nejvíc bezpochyby zaujmou děti – jedna malá holčička zrovna včera div nebrečela, že by snad balónky mohly prasknout. A co teprve ta stuha. Stuha, kterou je každý z trojice balónků uvázán, aby se nevypustil. Očividně je uvázaná dostatečně pevně, protože balónky skutečně neucházejí. To ale není nic zvláštního.

# 1 Webová aplikace

Webová databázová aplikace představuje softwarové řešení, které umožňuje uživatelům přistupovat k centralizované databázi prostřednictvím webového rozhraní. Tyto aplikace integrují správu dat a webovou prezentaci, usnadňuje aktualizaci, údržbu i zabezpečení informací. V současnosti jsou webové databázové aplikace klíčovým prvkem při správě informací ve vzdělávacích institucích i firmách, protože umožňují dynamické a interaktivní služby založené na standardních protokolech, jako jsou HTTP/HTTPS, a využívají technologie, například AJAX a RESTful API, pro efektivní komunikaci mezi klientem a serverem [1].

Webové aplikace fungují na principu modelu „klient-server“ a využívají standardní protokoly HTTP/HTTPS. Vývoj těchto aplikací je charakterizován přechodem od statických webových stránek k dynamickým systémům, které jsou postaveny na modulární architektuře se snadnou integraci nových funkcí a efektivní škálovatelnost. Technologie RESTful API standardizují výměnu dat mezi klientem a serverem a usnadňuje vývoj a údržbu aplikací. Historický vývoj webových technologií dokládá postupný přechod od Web 1.0 k interaktivním řešením Web 2.0 a dále k experimentálním přístupům Web 3.0, přičemž nejrozšířenější a nejspolehlivější jsou právě aplikace založené na modelu Web 2.0 [3]. Moderní frameworky pro frontend, jako jsou Vue, React nebo Angular, v kombinaci s backendovými technologiemi, například Spring Boot nebo Laravel, poskytuje vytvářet flexibilní, rychlé a spolehlivé aplikace, které splňují požadavky současného uživatele [1].

## 1.1. Architektura řešení při vývoji webových aplikací

Architekturní řešení ve vývoji webových aplikací hrají klíčovou roli při formování celkové struktury a funkčnosti systému. Správná volba architektury určuje nejen výkon a škálovatelnost komponent v budoucnu. V dnešním světě, kde se software neustále vyvíjí, se architektonická rozhodnutí stávají základem pro rychlou reakci na nové požadavky a technologické výzvy. Je důležité si uvědomit, že architektura webové aplikace není jen souborem komponent, ale souhrnem podložených návrhových rozhodnutí, která ovlivňují celý životní cyklus produktu. Tento přístup snižuje pravděpodobnost vzniku chyb, minimalizovat náklady na údržbu a zajistit vysokou úroveň bezpečnosti dat, a to je obzvláště aktuální pro systémy využívané ve vzdělávacím i korporátním prostředí [1].

### 1.1.1 Architektonický vzor MVC

Model-View-Controller (MVC) je základní architektonický vzor, který se stal nedílnou součástí vývoje webových aplikací. Tento vzor rozděluje aplikaci do tří hlavních komponent – modelu, view a controlleru – přičemž každá má jasně vymezenou roli. Model představuje datovou a business logiku; stará se o správu, validaci a ukládání dat, a zároveň definuje pravidla, která s těmito daty souvisejí. View, tedy prezentační vrstva, zodpovídá za vizuální zobrazení informací, které jsou získány z modelu, a to ve formě uživatelského rozhraní. Controller slouží jako prostředník mezi modelem a view; přijímá vstupy od uživatele, interpretuje je a na jejich základě volá odpovídající metody modelu, přičemž následně vyvolává aktualizace ve view.

Toto oddělení odpovědností přináší řadu výhod. Díky jasnému rozdělení funkcí, jednotlivé části aplikace lze vyvíjet a testovat nezávisle a výrazně usnadnovat údržbu a škálovatelnost systému. Například změny v uživatelském rozhraní (view) nevyžadují zásahy do business logiky (model), a naopak. Tento přístup také umožňuje týmům pracovat efektivněji, a soustředit pozornost na konkrétní komponenty bez nutnosti hlubokého porozumění celkové implementaci.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek 1*:* Common model-view-controller pattern (vl zdroj)

Výzkumy, jako například studie klasifikace MVC softwarových aplikací pomocí samoorganizujících map (Self-Organizing Maps, SOM) [4], ukazují, že správná implementace MVC vzoru výrazně ovlivňuje kvalitu softwaru. Analýza MVC aplikací pomocí metrik jako počet řádků kódu, duplicity, složitost a designová kvalita poskytuje cenné informace o tom, jak různé implementační strategie ovlivňují udržovatelnost a robustnost systému. Tato analýza umožňuje identifikovat potenciální problémy (tzv. "code smells") a optimalizovat architektonická rozhodnutí, zásadní milníky pro dlouhodobou úspěšnost a efektivitu softwarových řešení.

Z praktického hlediska je MVC vzor obzvláště cenný v prostředích, kde je potřeba zajistit vysokou míru interaktivity a dynamickou aktualizaci obsahu. Tyto požadavky jsou dnes typické pro webové aplikace. Ve spojení s technologiemi jako AJAX pro asynchronní komunikaci, RESTful API pro standardizovanou výměnu dat a moderními frameworky poskytuje MVC pevný a flexibilní základ pro vývoj systémů, které jsou uživatelsky přívětivé a snadno udržovatelné [4].

### 1.1.2 Model-View-Presenter MVP

**Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.**

Obrázek *2:* Common model-view-presenter pattern (vl zdroj)

Model-View-Presenter (MVP) je evoluční variantou klasického MVC, která vznikla s cílem snížit úzké propojení mezi prezentační vrstvou a business logikou. Ve vzoru MVP je View definováno jako pasivní komponenta, která nezpracovává žádnou prezentační logiku, ale pouze zobrazuje data a předává uživatelské akce do Presenter. Presenter funguje jako prostředník – přijímá vstupy z View, komunikuje s Modelem a následně aktualizuje View. Tento přístup umožňuje jasné oddělení odpovědností, lepší testovatelnost a snadnější údržbu aplikace.[5]

Presenter v tomto vzoru hraje klíčovou roli při zpracování uživatelských interakcí a aktualizaci UI. Na rozdíl od MVC, kde Controller často přímo manipuluje s View, v MVP Presenter nemá přímý přístup k View. Místo toho komunikuje prostřednictvím rozhraní, což umožňuje snadnější testování a modularitu komponent.​[5][8][33]

Empirické studie zaměřené na výkon nativních Android aplikací ukazují, že MVP může mít výhodu v oblasti správy zdrojů – například některé experimenty uvádějí, že architektura MVP vykazuje nižší spotřebu paměti ve srovnání s alternativními vzory. Mezi její hlavní přednosti patří jednoduchá implementace a dobrá testovatelnost [8]. V praxi ale může docházet ke zvýšení složitosti kódu kvůli soustředění business logiky do komponenty Presenter, a tím i ke ztíženému dalšímu rozvoji a údržbě systému [5].​

### 1.1.3 Model-View-ViewModel MVVM

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *3:* Common model-view-viewmodel pattern (vl zdroj)

Model-View-ViewModel (MVVM) je architektonický vzor, který se stal oblíbeným zejména díky podpoře obousměrné datové vazby a reaktivního programování. V MVVM se aplikace dělí na tři hlavní komponenty: Model, který spravuje data a business logiku; View, jež představuje uživatelské rozhraní; a ViewModel, který funguje jako prostředník mezi Modelem a View a transformuje data do formy vhodné pro zobrazení. Díky datové vazbě se změny v Modelu automaticky promítají do View a změny v uživatelském rozhraní jsou reflektovány ve ViewModelu bez nutnosti explicitní synchronizace.

Rozsáhlé literární přehledy a studie, jako je například práce "Evaluating Android Architectural Patterns: A Deep Dive into MVP, MVVM, and MVI" [7], potvrzují, že MVVM zvyšuje testovatelnost a udržovatelnost aplikací. Použití reaktivní datové vazby přispívá k rychlejší odezvě a efektivnějšímu využití CPU, přičemž zároveň však může vést k mírně vyšší spotřebě paměti kvůli implementaci dodatečné knihovny pro datovou vazbu [8].

Ve srovnání s MVP, kde Presenter přímo komunikuje s View prostřednictvím rozhraní, MVVM nabízí volnější propojení mezi komponentami. ViewModel nemá přímou referenci na View, čím umožňuje lepší oddělení odpovědností a usnadňuje testování. MVVM podporuje opětovné použití ViewModelu v různých View, což zvyšuje flexibilitu a škálovatelnost aplikací [5][33].

MVVM se hojně využívá především v mobilních a desktopových aplikacích, kde je podpora datové vazby běžná, ale v oblasti webového vývoje může být MVVM součástí širší architektury MVC, například jako součást front-endové architektury. Vue.js, Angular.js implementuje MVVM přístup prostřednictvím reaktivních komponent a dvoucestné datové vazby [25].​

### 1.1.4 Model-View-ViewModel-Coordinator MVVM-C

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *4:* Common model-view-viewmodel-coordinator pattern (vl zdroj)

MVVM-C představuje rozšíření standardního MVVM vzoru o dodatečnou komponentu, a to koordinátora (Coordinator), která se zaměřuje na správu navigačních toků a řízení přechodů mezi obrazovkami. V této architektuře zůstávají základní principy MVVM zachovány, tzn. že Model spravuje data, View zobrazuje uživatelské rozhraní a ViewModel obsahuje prezentační logiku – avšak koordinátor zajišťuje centralizovanou správu navigace. Díky tomu může ViewModel zůstat čistý a zaměřený pouze na správu dat a logiku, aniž by se musel starat o implementaci přechodů mezi různými částmi uživatelského rozhraní. Koordinátor také odpovídá za vytváření závislostí, jako jsou instance View a ViewModel, a jejich správné propojení, čímž se snižuje duplicitní kód a minimalizuje riziko chyb při řízení toků mezi různými moduly. Tato architektura se ukazuje jako zvláště vhodná pro rozsáhlé projekty, kde je potřeba efektivně řídit interakce mezi komponentami a zároveň podporovat agilní vývoj a škálovatelnost [7] [34].​

MVVM-C je často využíván při vývoji mobilních aplikací, zejména pro platformu iOS, kde koordinátor pomáhá udržet ViewController jednoduchý a zaměřený pouze na zobrazení uživatelského rozhraní. Tuto architekturu lze aplikovat i ve webovém vývoji. Například ve frameworku Vue.js může být Vue Router použit jako koordinátor, který spravuje navigaci mezi komponentami. V tomto kontextu router funguje jako samostatná entita, která vytváří a propojuje komponenty View a ViewModel, čímž naplňuje principy MVVM-C i ve frontendovém vývoji [15].

### 1.1.5 Model-View-Intent MVI

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *5:* Common model-view-intent pattern (vl zdroj)

MVI se skládá ze tří hlavních komponent: Model, View a Intent. View – pasivně zobrazuje aktuální stav a odesílá uživatelské akce jako Intenty. Intent – zachycuje uživatelské akce (např. kliknutí) a předává je Modelu k vyhodnocení. Model – každá změna vytváří nový stav, eliminuje nejednoznačnosti a zajišťuje konzistenci. Po zpracování v Modelu se stav automaticky promítne do View. Tímto způsobem data obíhají „v kruhu“, přičemž je zajištěn jednosměrný tok dat a neměnnost stavu aplikace. Tento vzor je obzvláště vhodný pro aplikace s komplexními interakcemi a asynchronními událostmi, kde je důležitá předvídatelnost a snadné ladění. Nicméně pro jednodušší aplikace může být implementace MVI zbytečně složitá a náročná na údržbu [5] [7].

Ačkoliv MVI vzniklo v kontextu mobilního vývoje, jeho principy jsou aplikovatelné i ve webovém prostředí, zejména při použití frameworku Vue.js. Vuex, oficiální knihovna pro správu stavu ve Vue.js, implementuje jednosměrný tok dat a centralizované řízení stavu. Tento přístup odpovídá principům MVI. Použití Vuex tak přináší výhody MVI do webového vývoje, zejména v projektech, kde je důležitá konzistence stavu a předvídatelnost chování aplikace [15].

### 1.1.6 Monolitická architektura

Monolitická architektura představuje tradiční přístup k vývoji webových aplikací, kdy je celá business logika, správa dat a prezentace uživatelského rozhraní implementována v jediné kódové bázi. V tomto přístupu nejsou aplikace rozděleny do nezávislých služeb – všechny komponenty běží v rámci jednoho procesu a využívají společnou databázi. Mezi hlavní výhody monolitu patří jeho relativní jednoduchost při vývoji, testování, ladění a nasazení, protože není třeba implementovat složité mechanismy mezi-procesové komunikace. Blinowski a spol. [9] uvádí, že v jedno-serverových nasazeních může monolit vykazovat vyšší výkon díky minimálním režijním nákladům spojeným s vnitřním voláním metod. Nicméně s rostoucí velikostí a složitostí aplikace vznikají problémy se škálovatelností – změna v jednom modulu může ovlivnit celý systém a horizontální škálování často vyžaduje kompletní replikaci aplikace. Tento přístup vede k neefektivnímu využití zdrojů [9]. Udržení modulární struktury a správa závislostí se navíc stává náročnější, a tím i obtížnější adaptace systému na měnící se požadavky [1], [3].

### 1.1.7 Mikroslužbová architektura

Na rozdíl od monolitické architektury mikroslužbový přístup rozděluje aplikaci na sadu nezávislých, samostatně provozovaných služeb, přičemž každá služba zodpovídá za konkrétní funkci. Každá mikroslužba může mít vlastní obchodní logiku, vlastní databázi a je vyvíjena a nasazována nezávisle na ostatních službách. Tento přístup umožňuje flexibilní škálování – je možné zvýšit počet instancí pouze u těch služeb, které jsou zatíženy nejvyšším počtem požadavků, a právě tento model je vhodný zejména pro rozsáhlé systémy s vysokou mírou paralelních dotazů [9]. Mikroslužby také zvyšují odolnost systému, protože selhání jedné služby neznamená ztrátu funkčnosti celé aplikace. Distribuovaná povaha mikroslužeb však přináší další složitosti, zejména v oblasti komunikace mezi službami. Ta může zvýšit režii a zkomplikovat monitorování, především při vysokém zatížení [9]. Použití standardních protokolů, jako jsou HTTP/HTTPS a RESTful API, napomáhá standardizaci výměny dat mezi službami, zároveň však vyžaduje dodatečnou infrastrukturu pro orchestraci a správu služeb – například prostřednictvím API bran nebo systémů pro objevování služeb [2], [9].

### 1.1.8 Zhodnocení architektur

Výběr mezi monolitickou a mikroslužbovou architekturou závisí na mnoha faktorech, včetně velikosti a složitosti aplikace, požadavků na škálovatelnost a odolnost systému, a také na schopnosti týmu efektivně spravovat a vyvíjet systém. Jak uvádí [9], monolitická architektura může v jednoserverových prostředích vykazovat lepší výkon díky absenci nákladů na meziprocesovou komunikaci. Monolitické aplikace jsou navíc levnější na vývoj a údržbu. S rostoucím zatížením a narůstajícími požadavky však mohou být vhodnějším řešením mikroslužby. U rozsáhlých systémů s dynamicky se měnícím zatížením nabízí mikroslužbový přístup významné výhody v oblasti škálovatelnosti a nezávislého řízení jednotlivých částí systému. Použití RESTful API a moderních technologií pro automatizaci a orchestraci napomáhá minimalizaci režijních nákladů spojených s distribuovanou povahou mikroslužeb. Hlavními nevýhodami mikroslužbové architektury jsou však vyšší nároky na systémové prostředky a potřeba profesionálních dovedností spolu se sehraným týmem pro vytvoření kvalitní aplikace [3], [9].

Na základě přehledu výhod a nevýhod jednotlivých architektonických přístupů vyplývá, že volba optimální architektury pro konkrétní projekt musí vycházet z jeho specifických požadavků, jako je velikost systému, očekávané zatížení a potřeba rychlých iterací. Pro projekt, který se zaměřuje na relativně malý až středně velký rozsah funkcí a nepočítá s masivním nárůstem uživatelské základny, se jako nejvhodnější volba jeví monolitická architektura. Díky své jednoduchosti, nižším režijním nákladům na interní volání a snadnější implementaci umožňuje rychlejší vývoj, testování a nasazení aplikace. To je klíčové pro dosažení stabilního a efektivního řešení. I když mikroslužby nabízejí výhody v oblasti škálovatelnosti a nezávislého nasazení jednotlivých komponent [3], [9].

# 2 Technologie používané při vývoji webových aplikací

Vývoj webových aplikací je komplexní proces, který vyžaduje integraci řady technologií, přičemž klíčovým prvkem je správná volba technologického stacku. Tento stack zahrnuje frontendové technologie, které zajišťují interaktivní a responzivní uživatelské rozhraní, a backendové technologie, kde je implementována business logika, správa dat a zabezpečení. Kromě toho hraje roli i integrace s databázemi a externími službami prostřednictvím RESTful API nebo jiných komunikačních protokolů. Taková řešení umožňují dynamickou výměnu informací mezi klientem a serverem a představují základ pro škálovatelné a udržitelné systémy [3] [9] [10].

## 2.1 Backendové technologie

Backendové technologie tvoří „motor“ webových aplikací, mezi nejčastěji používané patří Node.js, PHP (Laravel), Python (Django) a robustní enterprise platformy jako Spring Boot. Node.js, využívající JavaScript, je založený na asynchronní event-driven architektuře, což umožňuje efektivní zpracování mnoha souběžných požadavků [10]. PHP s frameworkem Laravel je oblíbený pro svou jednoduchost, rychlý vývoj a širokou komunitní podporu, avšak při vyšších nárocích může vykazovat omezení výkonu [10]. Python s frameworkem Django nabízí přehlednost a robustní strukturu, avšak může mít omezení při vysokém zatížení [10]. Enterprise platformy jako Spring Boot či MS.NET Core integrují pokročilé mechanismy pro správu databází, transakcí a RESTful API, čímž zajišťují vysokou úroveň bezpečnosti, výkonu a škálovatelnosti [12][13].

### 2.1.1 Srovnání technologií pro backend

Tabulka 1: Srovnání backendových technologií. Zdroj:[1] [9] [10] [12] [13]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kritéria | Express (Node.js) | Laravel (PHP) | Spring Boot (Java) | Django (Python) |
| Výkon a škálovatelnost | Vysoká efektivita při velkém množství paralelních požadavků díky asynchronní architektuře | Dobrá pro menší a středně velké projekty, škálovatelnost však klesá při extrémním zatížení | Výborná škálovatelnost a stabilní výkon i při vysokém zatížení díky pokročilým mechanismům správy transakcí | Vhodný pro menší až střední aplikace, škálovatelnost limitována při náročném zatížení |
| Vývojářská přívětivost a údržba | Podporuje modulární a flexibilní struktury, náročnější na znalosti asynchronního programování | Jednoduchá údržba, vhodný pro rychlý vývoj a prototypování, komunita usnadňuje vývoj | Pokročilá podpora a dlouhodobá údržba komplexních systémů, vyžaduje znalost Javy | Jednoduchá struktura a rychlá integrace, ideální pro prototypy, méně vhodné pro rozsáhlé projekty |
| Komplexnost a integrace | Ideální pro aplikace s vysokým počtem I/O operací, integrace vyžaduje manuální nastavení | Omezenější flexibilita při integraci složitějších systémů a více databází | Integrované pokročilé nástroje pro transakce, komunikaci mezi komponentami a API | Snadné základní integrace, složitější scénáře vyžadují doplňkové nástroje a knihovny |
| Náklady a dostupnost odborných znalostí | Zvyšující se dostupnost specialistů, nicméně vyžaduje hlubší odborné znalosti | Dobrá dostupnost odborníků díky široké komunitě a historii frameworku | Vyžaduje specializované znalosti, často vyšší náklady na vývoj, avšak výborná dlouhodobá návratnost | Velmi dobrá dostupnost odborníků díky silné komunitě, ale mohou nastat omezení při složitých projektech |

Výběr optimální backendové technologie závisí na konkrétních požadavcích projektu. Pro aplikace s vysokým počtem souběžných uživatelů a náročnými I/O operacemi je vhodné využít asynchronních platforem, jako je Express (Node.js), které zajišťují nízkou latenci a efektivní využití zdrojů, zatímco rozsáhlé podnikové systémy s přísnými požadavky na bezpečnost, robustnost a integraci se nejlépe realizují pomocí enterprise řešení, například Spring Boot. Menší a jednodušší projekty pak mohou být efektivně vyvíjeny s využitím frameworku Laravel (PHP), který podporuje rychlý vývoj, nebo Django (Python), jenž je vhodný zejména pro prototypování a méně náročné aplikace [1] [9] [10] [12] [13].

## 2.2 Frontendové technologie

Frontendové technologie představují klíčovou složku vývoje moderních webových aplikací, neboť zajišťují interaktivní a uživatelsky přívětivé rozhraní, které reaguje na dynamické změny v reálném čase. Hlavním cílem využívání frontendových frameworků je usnadnění vývoj**e.** Před rozšířením těchto technologií byly webové stránky vytvářeny pomocí staku HTML, CSS a JavaScriptu, který často ztěžoval práci. Dnes existuje dostatek různorodých frameworků, díky nimž si lze snadno vybrat takový, který nejlépe odpovídá konkrétním požadavkům. Mezi nejrozšířenější patří JavaScriptové frameworky, jako jsou React.js, Angular.js a Vue.js, které umožňují tvorbu komplexních jednostránkových aplikací (SPA). Tyto technologie využívají pokročilé techniky, například virtuální DOM pro efektivní aktualizaci uživatelského rozhraní, obousměrnou datovou vazbu zajišťující synchronizaci mezi modelem a zobrazením, a komponentový přístup podporující opětovné použití kódu. Díky těmto vlastnostem se frontendové technologie staly nepostradatelným nástrojem pro dosažení kvalitní uživatelské zkušenosti a rychlého načítání obsahu. Vzhledem k rostoucím nárokům uživatelů na rychlost a interaktivitu webových aplikací je tento aspekt stále důležitější [14].

### 2.2.1 Srovnání technologií pro frontend

Tabulka 2: Srovnání frontendových technologií. Zdroj: [14] [15] [19]

|  |  |  |  |
| --- | --- | --- | --- |
| Kritéria | Angular.js | React.js | Vue.js |
| Robustnost | Robustní a komplexní rámec vhodný pro rozsáhlé podnikové aplikace | Flexibilní řešení pro rychlý vývoj modulárního kódu | Lehké a přehledné řešení ideální pro menší až středně velké projekty |
| Učební křivka | Strmější učební křivka s vyššími nároky na konfiguraci | Nízká učební křivka, usnadňuje rychlý start | Velmi přívětivá pro začátečníky díky jednoduchosti a architektuře MVVM |
| Správa stavu | Podporuje komplexní správu stavu vhodnou pro náročnou logiku | Efektivní správa stavu umožňuje rychlou aktualizaci a údržbu kódu | Reaktivní datová vazba usnadňuje synchronizaci dat mezi komponentami |
| Komunita a podpora | Silná komunita a rozsáhlá podpora, ideální pro velké projekty | Velká a aktivní komunita | Rostoucí a aktivní komunita |
| Rychlost vývoje | Vyžaduje více konfigurace, může zpomalit vývoj, zejména u menších projektů | Velmi agilní, vhodné pro prototypování a rychlý vývoj | Rychlý vývoj díky nízké složitosti kódu a jednoduché konfiguraci |
| Dokumentace | Podrobná a komplexní dokumentace, usnadňuje pochopení složitých konceptů, ale může být náročná na studium | Dobrá dokumentace s rozsáhlým ekosystémem knihoven, avšak některé aspekty se spoléhají na komunitní zdroje | Výjimečně přehledná a srozumitelná dokumentace, usnadňuje učení a implementaci, |

Srovnání jednotlivých frontendových technologií odhaluje, že každý framework má své specifické přednosti i omezení, které ovlivňují volbu vhodného řešení pro daný projekt. Rozsáhlé a komplexní aplikace často těží z robustního rámce Angular.js, zatímco agilní vývoj modulárního kódu je efektivně řešen pomocí React.js. Vue.js pak kombinuje jednoduchost a přehlednost s moderními prvky, jako je reaktivní datová vazba a jej činí ideální volbou pro menší a středně velké projekty, kde je kladen důraz na rychlost vývoje a nízkou složitost kódu [14] [15] [19].

## 2.3 Relační databáze

Relační databáze představují základ správy dat, jejichž koncepty se ustálily již více než padesát let od zásadního příspěvku E.F. Codda, který v roce 1970 definoval relační model, založený na reprezentaci dat v řádcích a sloupcích. Tento model umožnil oddělit logickou strukturu dat od jejich fyzické reprezentace a otevřel cestu pro deklarativní dotazovací jazyk SQL, který si dodnes udržuje dominantní postavení díky své jednoduchosti a efektivitě při zpracování dotazů. Historický vývoj databázových systémů zaznamenal postupný přechod od hierarchických a síťových modelů, které byly omezeny svou flexibilitou, k relačnímu přístupu, jenž umožnil optimalizaci vyhledávání i manipulaci s daty pomocí sofistikovaných optimalizátorů, které překládají dotazy do efektivních exekučních plánů. Navzdory kritikám, například kvůli absenci úplné ortogonality či problémům s null hodnotami, se relační databáze staly osvědčeným a široce používaným řešením díky své robustnosti, konzistenci a možnosti správy velkých objemů dat ve vysoce konkurenčním prostředí. S nástupem nových požadavků na horizontální škálovatelnost a dynamickou správu dat s flexibilním schématem se na trhu objevují i systémy NoSQL, které částečně uvolňují některá omezení relačních databází, avšak díky dlouhodobému využívání a široké podpoře se relační databáze, jako MySQL, PostgreSQL, Microsoft SQL Server, SQLite či Oracle Database, stále řadí mezi nejpoužívanější systémy pro stěžejní aplikace [16].

### 2.3.1 PostgreSQL

PostgreSQL je open-source relační databázový systém, který je známý svou robustností, vysokou úrovní standardizace a rozšířenými funkcemi umožňujícími efektivní správu komplexních datových operací. Tento systém podporuje pokročilé SQL funkce, jako jsou rekurzivní dotazy, window funkce či možnost definice vlastních datových typů [20]. Ve srovnání s MySQL, přímým a zároveň jedním z nejpopulárnějších alternativních řešení, se PostgreSQL vyznačuje lepší schopností zvládat složitější datové struktury a náročné transakční operace. To potvrzují i srovnávací studie zaměřené na výkonnost a specifika jednotlivých databázových systémů [21]. Díky aktivní komunitě, rozsáhlé dokumentaci a pravidelným aktualizacím představuje PostgreSQL stabilní volbu pro aplikace, kde je prioritou zajištění konzistence a bezpečnosti dat. Tento systém nabízí komplexní řešení kombinující vysoký výkon, flexibilitu a rozšiřitelnost, a je proto často preferovaným nástrojem pro vývoj webových aplikací.

## 2.4. Docker a Docker Compose

Docker je open-source platforma, pro vývoj, nasazení a správu aplikací pomocí kontejnerů. Kontejnery představují přenosná a izolovaná prostředí, ve kterých aplikace běží konzistentně napříč různými systémy a infrastrukturami. Ten přístup zjednodušuje migraci a škálování software, umožňuje plně se soustředit na psaní kódu a zkracuje dobu vývoje a nasazení aplikací, eliminuje nutnost řešení složitých konfigurací operačního systému [17].

Docker Compose pak rozšiřuje funkčnost Dockeru přes definice a koordinace vícekontejnerové aplikace prostřednictvím jediného konfiguračního souboru. Tento nástroj usnadňuje a automatizuje nasazení a správu závislostí mezi jednotlivými službami, jako jsou databáze, webové servery a další komponenty. Díky vzájemné komunikaci mezi kontejnery lze komplexní systémy spravovat mnohem efektivněji, aniž by bylo třeba každou službu nastavovat ručně [18].

Celkově Docker a Docker Compose představují moderní přístup k vývoji a správě softwaru, který podporuje agilní metody, kontinuální integraci a nasazení (CI/CD) pro DevOps flow s moznosti rychlou adaptaci na měnící se požadavky trhu. Platforma Docker dává jednoduchý přístup efektivní využití hardwarových zdrojů a zvyšuje spolehlivost a konzistenci běhu aplikací, zatímco Docker Compose koordinuje více kontejnerových služeb do jednotného, snadno spravovatelného celku [17] [18]. Tímto způsobem jsou tyto nástroje flexibilní a vysoce škálovatelné.

## 2.5 Java Spring Boot

Spring Boot je framework vyvinutý společností Pivotal, pro vývoj a nasazení aplikací založených na platformě Spring. Díky své auto-konfigurační povaze a přednastaveným inicializačním „starterům“ umožňuje rychle vytvářet samostatné, produkčně připravené aplikace s minimálním množstvím konfigurace.

### 2.5.1 Architektura a principy Spring Boot

Spring Boot staví na robustní architektuře tradičního Spring frameworku, avšak výrazně zjednodušuje konfiguraci díky mechanismu automatické konfigurace. Ten na základě dostupných knihoven a aktuálního kontextu prostředí volí vhodné nastavení, čímž zajišťuje optimální běh aplikace. Tato architektura podporuje vývoj mikroslužeb a modulárních systémů se snadnou správu a škálování. Auto-konfigurační moduly, tzv. „starters“, a vestavěný server (např. Tomcat) urychluje celý proces vývoje a nasazení aplikací, eliminují potřebu složitého manuálního nastavování [22].

### 2.5.2 Výhody a přínosy použití Spring Boot

Díky automatickým a přednastaveným konfiguracím je při vývoji možné soustředit se přímo na psaní aplikační logiky. Samostatné aplikace, které lze spustit jako běžné Java programy, usnadňují nasazení v cloudovém prostředí. Spring Boot zároveň umožňuje snadnou integraci s databázemi, messaging, security a dalšími enterprise systémy. Dlouhodobou udržitelnost projektů podporuje rozsáhlá dokumentace, množství dostupných tutoriálů a aktivní komunita, která řešení Spring Boot pravidelně aktualizuje a rozvíjí [23].

### 2.5.3 Srovnání se standardním Spring frameworkem

Spring Boot představuje rozšíření standardního Spring frameworku, přičemž přidává vestavěnou automatickou konfiguraci a výrazně usnadňuje nasazení aplikací. Zatímco tradiční Spring aplikace vyžadují detailní konfiguraci pomocí XML souborů nebo anotací, Spring Boot tyto složitosti eliminuje díky svému auto-konfiguračnímu mechanismu. Tím dochází k rychlejšímu zahájení vývoje a zároveň se snižuje riziko chyb při nastavování aplikace. Kromě toho zjednodušuje jak samotný vývoj, tak i následnou údržbu. Díky vestavěnému serveru a předdefinovaným závislostem (tzv. *starters*) se není nutné zabývat ruční správou verzí a konfigurací externích knihoven [22], [24].

### 2.5.4 Omezení a perspektivy budoucího vývoje

Existují určité oblasti, kde může být přizpůsobení Spring Boot specifickým požadavkům náročnější. Například u velmi komplexních aplikací může být automatická konfigurace omezující, zejména v případech, kdy je zapotřebí detailní ladění nebo specifické optimalizace. Přestože Spring Boot výrazně zjednodušuje vývoj, zůstává přímým nástupcem rozsáhlého a komplexního Spring frameworku, jehož zvládnutí může být pro začínající vývojáře náročné. V případech, kdy projekt nevyžaduje pokročilé enterprise funkce, může být pro začátek vhodnější zvolit jednodušší backendová řešení, jako jsou Django, Laravel, Node.js nebo Flask. Pokud se však vývojář rozhodne využívat Spring Boot i přes jeho složitost, je pravděpodobné, že mnohé potenciální problémy a omezení bude možné v budoucnu překonat. Díky otevřenému zdrojovému kódu a aktivní komunitě dochází k pravidelným aktualizacím a rozšiřování funkcí. Výhled do budoucna je velmi slibný – Spring Boot totiž neustále posiluje svou integraci s cloudovými platformami a nástroji pro vývoj mikroslužeb, a zůstává tak i nadále klíčovou technologií pro enterprise vývoj [22], [24].

2.5.5 Z čeho se skládá Spring Boot projekt  
Typická struktura projektu ve Spring Boot je navržena tak, aby podporovala logické rozdělení kódu a usnadňovala jeho správu. Hlavním vstupním bodem je třída obsahující metodu main, která se nachází v hlavním balíčku a inicializuje Spring kontext. Zdrojový kód se obvykle umisťuje do složky ***src/main/java***, zatímco konfigurační soubory, jako je application.properties nebo application.yml, jsou umístěny v adresáři ***src/main/resources***. Součástí projektu jsou také složky pro statické zdroje, jako jsou obrázky a styly, a pro šablony, pokud aplikace využívá serverové renderování. Projekt je obvykle organizován do logických vrstev, kde jsou odděleny kontrolery, služby, repozitáře a modely. Testy se pak nacházejí v adresáři ***src/test/java***. Tato struktura je často generována pomocí nástroje Spring Initializr [32], který poskytuje správně nakonfigurovaný základ pro jejich projekty [22] [23] [24].

## 2.6 Vue.js

Vue představuje moderní přístup k vývoji uživatelských rozhraní, který klade důraz na jednoduchost, modularitu a vysokou výkonnost díky svým inovativním principům a architektonickým řešením [19].

### 2.6.1 Architektura a principy Vue.js

Vue je postaveno na architektuře MVVM, která zajišťuje oddělení datové logiky od prezentační vrstvy prostřednictvím reaktivní datové vazby s automatickou aktualizaci uživatelského rozhraní při změnách v datech. Framework využívá virtuální DOM, který optimalizuje manipulaci se skutečným DOM a tím zvyšuje výkon aplikací. Dále je Vue založen na přístupu orientovaném na komponenty, kdy každá komponenta obsahuje šablonu, skript a styly v jednom souboru, podporuje modularitu a opakované použití kódu. Vue také přináší statickou kontrolu a robustnější vývoj podporou TypeScript, a prostřednictvím platformy Weex lze Vue využít i pro tvorbu mobilních aplikací [25].

### 2.6.2 Výhody a přínosy použití Vue.js

Vue nabízí rychlý start i pro začátečníky díky své značné flexibilitě, intuitivní syntaxi a přehledné dokumentaci, která výrazně urychluje vývoj. Reaktivní systém frameworku zajišťuje bezproblémový, efektivní a plynulý chod aplikace, díky němuž se jakákoli změna dat okamžitě projeví v uživatelském rozhraní. Velkou výhodou je rovněž možnost efektivně vyvíjet a spravovat aplikace, jak menší, tak i středně velké bez nárůstu složitosti kódu. Díky komponentové architektuře Vue lze snadno strukturovat i rozsáhlejší projekty a udržet jejich čitelnost a přehlednost [26].

### 2.6.3 Nedostatky Vue.js

I přes řadu výhod má Vue i své omezení. Ve srovnání s některými konkurenty může být ekosystém a komunita menší. To může vést k omezené dostupnosti specializovaných knihoven a rozšíření. Další výzvou je rychlý vývoj frameworku, vedoucí k častým aktualizacím a nutnosti přizpůsobovat stávající projekty novým verzím a může to komplikovat dlouhodobou údržbu. Tyto faktory mohou představovat určité potíže zejména v komplexnějších enterprise projektech [26].

### 2.6.4 Z čeho se skládá Vue projekt

Typický Vue projekt je strukturován tak, aby podporoval modulární vývoj a efektivní správu kódu. Hlavním vstupním bodem je soubor ***main.js*** (nebo ***main.ts***, pokud je využit TypeScript), kde se inicializuje instance Vue a nastavují se globální konfigurace, včetně integrace routeru a správy stavu, například pomocí Vuex. Složka komponent obsahuje jednotlivé .vue soubory, kde každá komponenta zahrnuje šablonu, logiku a styly. Dále projekt obvykle obsahuje složku pro definici tras, která zajišťuje navigaci mezi komponentami, a složku s assety, kde jsou umístěny statické zdroje jako obrázky, styly a další soubory [25].

# PRAKTICKÁ ČÁST

# 3. Vybrané komunitní platformy a intranety

Intranet představuje vnitřní síť, určenou pro výměnu informací, spolupráci a automatizaci procesů uvnitř organizace nebo vzdělávací instituce. Díky omezenému přístupu k těmto systémům je zajištěna vysoká úroveň bezpečnosti a centralizovaného řízení dat. To je zvláště důležité pro vzdělávací instituce, kde má ochrana osobních údajů studentů a učitelů zásadní význam. Možnosti a výhody intranetových systémů jsou široce reflektovány v odborné literatuře a standardech [1]

## 3.1 Facebook

Tato sociální globální síť, známá pod označením Facebook od společnosti Meta využívají jednotlivci, rodiny, firmy, neziskové organizace i vládní instituce. Uživatelé si zde vytvářejí osobní profily, sdílejí fotografie, videa a příspěvky, komunikují prostřednictvím soukromých zpráv a mohou se zapojovat do skupin, které podporují komunitní interakci. Kromě toho Facebook nabízí Marketplace pro nákup a prodej zboží přímo na platformě, a řadu dalších nástrojů pro event management, reklamu a analytiku. Přestože platforma poskytuje rozsáhlé možnosti, pro některé uživatele může představovat omezení zejména z hlediska ochrany osobních údajů a moderace obsahu [27].

## 3.2 Reddit

Reddit – komunitně orientovaná diskusní platforma, provozovaná společností Reddit Inc., využívají studenti, nadšenci, profesionálové a experti z různých oborů. Uživatelé Redditu se zapojují do diskusí ve specializovaných fórech, tzv. subredditech, kde mohou publikovat příspěvky, hlasovat a komentovat obsah. Tato platforma umožňuje relativní anonymitu a podporuje živé a dynamické diskuse pro sdílení znalostí a názorů. Na druhou stranu však může být Reddit náchylný k problémům s moderací a občas nese riziko šíření kontroverzního nebo polarizujícího obsahu [28].

## 3.3 Telegram

Komunikační aplikace Telegram, kterou využívají jednotlivci, skupiny, firmy, a dokonce i vládní agentury, a je známá především svou vysokou úrovní zabezpečení a šifrování. Uživatelé Telegramu mohou využívat nejen standardní funkce pro posílání zpráv, ale také skupinové chaty, kanály pro vysílání informací a integrované miniaplikace (tg Mini Apps), které umožňují provádět například nákupy nebo rezervace přímo v rámci aplikace. Tato rozšířená funkčnost podporuje inovativní využití platformy, ale může také čelit regulacím v určitých zemích a potenciálním bezpečnostním rizikům, zejména v případě veřejně přístupných kanálů [29].

## 3.4 LinkedIn

Profesionální sociální síť LinkedIn, vlastněná společností Microsoft, kterou používají pracovníci, personalisté, odborníci a firmy pro budování kariérních kontaktů a profesní komunikaci. Platforma umožňuje vytváření detailních profilů, sdílení odborných článků a informací o pracovních příležitostech, a také podporuje účast ve skupinách zaměřených na konkrétní odvětví. Mezi klíčové funkce patří možnost vyhledávání pracovních nabídek, navazování kontaktů a budování profesionálních sítí. Omezení LinkedInu spočívají zejména ve formálním prostředí, které může omezovat osobní interakce ve srovnání s volnějšími sociálními sítěmi [30].

## 3.5 Instagram

Instagram od Meta je vizuálně orientovaná sociální síť, kterou využívají jednotlivci, firmy a influenceři pro sdílení fotografií, videí a příběhů. Uživatelé zde vytvářejí atraktivní profily a prostřednictvím funkcí jako Stories, Reels nebo živých přenosů mohou efektivně komunikovat se svým publikem. Platforma nabízí také řadu nástrojů pro podniky, jako jsou analytika, placené reklamy a možnost správy kampaní. Ačkoliv Instagram efektivně podporuje budování vizuální identity a zapojené komunity, platforma zároveň čelí kritice v souvislosti s ochranou osobních údajů, nedostatečnou moderací obsahu a jistými omezeními pro uživatele i firmy [31].

# 4 Analýza

Během práce byla provedena analýza dotazů od studentů prostřednictvím dotazníku. Dotazník byl sestaven ve dvou jazycích – češtině a ruštině – pro porovnání požadavků mezi zahraničními a českými studenty prezenční formy. Byla také připravena verze dotazníku pro studenty Erasmu v angličtině, avšak z nedostatku dat nebude v této práci zohledněna. Dotazníky byly distribuovány prostřednictvím skupinových chatů na sociálních sítích, studentské pošty a tištěných letáků na fakultách. Na základě výsledků dotazníku budou formovány požadavky na aplikaci. Dále je důležité upřesnit, že u otázek s odpověďmi od 1 do 5, 1 znamená nejnižší hodnocení a 5 nejvyšší. Toto upřesnění je nutné kvůli rozdílům mezi českým a rusky mluvícím hodnotícím systémem (v rusky mluvících zemích 5 bodů odpovídá 1 bodu v Česku).

## 4.1 Uživatelské chování a zkušenosti

Zaměřeno na to, jak studenti interagují s univerzitou, jaké mají zkušenosti a jaké problémy vnímají

Изображение выглядит как текст, снимок экрана, диаграмма, число

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *6: Do jaké míry se cítíte začleněni do univerzitního života? (*vl. zdroj)

Изображение выглядит как текст, снимок экрана, диаграмма, число

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *7: Jak snadné pro vás je získávat informace o událostech, které se na univerzitě konají?* (vl. zdroj)

*Изображение выглядит как текст, снимок экрана, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.*

Obrázek *8: Jak často navštěvujete univerzitní akce (kulturní, vědecké, sportovní)?* (vl. zdroj)

Изображение выглядит как текст, снимок экрана, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *9: Do jaké míry jste ochotni sdílet své studijní materiály* (vl. zdroj)

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *10: Jste členem nějaké studentské skupiny (klubu) se společnými zájmy?* (vl. zdroj)

*Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.*

Obrázek *11: S jakými hlavními obtížemi jste se potýkali před či při nástupu na univerzitu?* (vl. zdroj)

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *12: Co vám nejvíce pomohlo s adaptací na univerzitu?* (vl. zdroj) Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *13: Odkud se nejčastěji dozvídáte o univerzitních událostech?* (vl. zdroj)

Изображение выглядит как текст, снимок экрана, линия, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *14: O které typy událostí máte největší zájem?* (vl. zdroj)

### 4.1.1 Shrnutí

Na základě analýzy dotazníku lze konstatovat, že studenti mají relativně nízký pocit začlenění do univerzitního života, průměrné hodnocení činí jen lehce nad hodnotou 2 z 5 možných. To koresponduje s nízkou četností jejich účasti na univerzitních akcích, většina respondentů navštěvuje tyto akce jen velmi zřídka. Výsledky naznačují, že hlavní příčinou není nedostatek informací, jelikož respondenti hodnotí dostupnost informací o událostech relativně kladně. Dominantním zdrojem informací jsou sociální sítě, zejména Instagram. Tato zjištění zdůrazňují relevanci navrhované univerzitní platformy, která může centralizovat informace o událostech a nabízet snadnější způsoby komunikace mezi studenty. Vysoká ochota studentů sdílet studijní materiály a identifikovaná potřeba komunitního propojení poukazují na potenciál plánované aplikace ke zvýšení jejich zapojení do univerzitního života. Jelikož studenti nejvíce oceňují kulturní a společenské akce, aplikace by měla klást důraz nejen na akademické aktivity, ale také na vytváření komunit a usnadnění sociální interakce. Navržená platforma tak může efektivně řešit problém nízkého začlenění studentů do univerzitního života prostřednictvím cíleného propojení komunikačních nástrojů, sdílení zdrojů a podpory neformální interakce mezi studenty.

## 4.2 Přednástupová fáze a informační bariéry

Zaměřeno na problémy a potřeby studentů před nástupem na univerzitu

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *15: Co se týče období PŘED nástupem na univerzitu potíže s vyhledáváním informací?* (vl. zdroj)

*Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.*

Obrázek *16: S jakými hlavními obtížemi jste se potýkali před či při nástupu na univerzitu?* (vl. zdroj)

Изображение выглядит как текст, снимок экрана, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *17: Do jaké míry by vám pomohla taková platforma ještě před nástupem?* (vl. zdroj)

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *18:* *Kdybyste se mohli obrátit na další studenty ještě před nástupem...* (vl. zdroj)

### 4.2.1 Shrnutí

Výsledky analýzy odpovědí odhalují, že studenti před nástupem na univerzitu čelí značným potížím při získávání informací o životě na škole, ubytování, kroužcích a administrativních postupech. Průměrné hodnocení těchto potíží se pohybuje kolem 2,9 bodu ze 5, a tohle naznačuje, že bariéry nejsou zanedbatelné. Respondenti opakovaně uvádějí nedostatek podpůrné komunity a osobních kontaktů, které by jim usnadnily adaptaci v novém prostředí. Studentům schází spolehlivá informační podpora, která by umožnila rychlejší orientaci v prostředí univerzity.

Důležitou roli hraje očekávání přínosu online komunity, která by studentům poskytla potřebné informace ještě před nástupem na fakultu. Průměrné hodnocení přínosu takové platformy dosahuje hodnoty přibližně 3.7 bodu, to svědčí o pozitivním postoji vůči integraci informací do jednoho centrálního systému. Téměř všichni dotazovaní (99 z 118) potvrzují, že možnost konzultovat záležitosti s dalšími uchazeči prostřednictvím uzavřeného fóra by usnadnila adaptaci na univerzitní prostředí.

Tato zjištění podtrhují význam navrhované platformy, jejímž cílem je překonat informační bariéry a poskytnout studentům prostor pro přednástupovou komunikaci. Platforma by měla centralizovat informace o univerzitním životě, nabídnout přístup ke zkušenostem starších studentů a umožnit vzájemnou výměnu zkušeností mezi uchazeči. Navržené řešení tak usiluje o vytvoření prostředí, které podpoří rychlou a efektivní adaptaci nových studentů. Celkově výsledky analýzy potvrzují, že vytvoření jednotné online komunity může eliminovat nedostatek informací, posílit mezilidské vazby a výrazně přispět k lepší integraci studentů do akademického prostředí.

## 4.3 ****Požadavky na aplikaci****

Otázky, které přímo souvisejí s návrhem funkcí a vlastností navrhované platformy

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *19:* *Pokud by existovala jednotná univerzitní platforma... usnadnilo by vám to...?* (vl. zdroj)

**Изображение выглядит как текст, снимок экрана, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.**

Obrázek *20: Do jaké míry by vám pomohla taková platforma ještě před nástupem?* (vl. zdroj)

**Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.**

Obrázek *21:* *Kdybyste se mohli obrátit na další studenty ještě před nástupem...* (vl. zdroj)

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *22:* *Chtěli byste mít přístup k základním informacím o ostatních studentech...?* (vl. zdroj)

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *23: Považujete za užitečné, aby platforma byla propojena s vaším studijním rozvrhem...?* (vl. zdroj)

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *24*: *Začali byste aktivně zveřejňovat vlastní materiály nebo příspěvky...?* (vl. zdroj)

*Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.*

Obrázek *25*: *Byla by podle vás užitečná vizualizace kampusu...?* (vl. zdroj)

Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *26*: *Uvítali byste, kdyby platforma usnadňovala sdílení studijních materiálů...?* (vl. zdroj)

### 4.3.1 Shrnutí

Odpovědi na otázky zaměřené na požadavky na navrhovanou platformu ukazují, že studenti mají velký zájem o sjednocení různých funkcí do jednoho systému. Respondenti vyjadřují potřebu mít centralizovaný přehled událostí, diskusního fóra a tržiště, abych ta aplikace umožnila rychlou orientaci a snadnou komunikaci. Jednotlivci dávají přednost nástroji, který by jim nabídl možnost získat základní informace o ostatních studentech, přehled studijního rozvrhu a upozornění na nadcházející zkoušky či akce katedry. Odezva na myšlenku aktivního zveřejňování vlastních materiálů a příspěvků naznačuje, že studenti očekávají interaktivitu a ochotu podílet se na obsahu. Interaktivní vizualizace kampusu, například formou mapy budov, knihoven či učeben, bývá vnímána jako užitečný prvek, který pomůže novým i stávajícím studentům lépe se orientovat v prostředí fakulty.

Výsledky z dotazníku zdůrazňují zájem o online komunitu, jež nabídne přehled informací ještě před nástupem na fakultu. Mělo bych pozitivní dopad na adaptaci do nového prostředí, že by takový systém usnadnil komunikaci mezi uchazeči a studenty, z představ respondentu. Uživatelé preferují propojení aplikace s jejich studijním rozvrhem pro plánování a lepší organizaci času, a rovněž podporu sdílení studijních materiálů v rámci uzavřených či veřejných skupin. Zájem o tyto funkce svědčí o potřebě jednotného řešení, které zjednoduší přístup k důležitým informacím a podpoří komunitní vazby mezi studenty.

Celkový přístup k návrhu platformy vyplývá z touhy překonat fragmentaci informací. Navrhované řešení by mělo nabídnout moderní prostředí, které usnadní interakci, posílí mezilidské vztahy a přispěje k celkové integraci studentů do akademického života. Tyto poznatky poskytují solidní základ pro další vývoj a optimalizaci univerzitní aplikace, která by splňovala reálné potřeby cílové skupiny.

## 4.4 Porovnání odpovědi mezí rusky a česky mluvících studentu

Pro úplnost je vhodné upřesnit, že pod označením „rusky mluvící studenti“ nebo ru-studenty nejsou míněni pouze občané Ruské federace, ale širší skupina studentů hovořících rusky jako svým dorozumívacím jazykem ve skupině – tedy například studenti z Ukrajiny, Běloruska, Kazachstánu, Uzbekistánu a dalších států postsovětského prostoru. Tito studenti tvoří nezanedbatelnou část univerzitní komunity.

Изображение выглядит как текст, снимок экрана, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *27*: *Do jaké míry jste ochotni sdílet své studijní materiály?ru-studenty* (vl. zdroj)

Mezi ruskými a českými studenty byly identifikovány určité významné rozdíly, které je nutné zohlednit při návrhu univerzitní online platformy. Nejvýraznějším zjištěním je vyšší ochota rusky mluvících studentů sdílet studijní materiály, kde průměrná hodnota dosáhla 4,06 oproti 3,68 u českých studentů. Tento rozdíl naznačuje, že platforma by měla být navržena s důrazem na nástroje pro jednoduché a bezpečné sdílení obsahu, přičemž u českých uživatelů bude nezbytné tuto funkcionalitu zároveň doprovodit edukací a podporou komunity, která výší důvěru v její používání.

Изображение выглядит как текст, снимок экрана, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *28*: *Do jaké míry se cítíte začleněni do univerzitního života? ru-studenty* (vl. zdroj)

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.**

Obrázek *29*: *Co vám nejvíce pomohlo s adaptací na univerzitu? ru-studenty* (vl. zdroj)

V otázce integrace do univerzitního života se ruská skupina hodnotila mírně výše (2,31 bodu) než česká (2,07 bodu), ukazuje na obecnou potřebu zlepšit prostředí pro začlenění studentů. Tomu odpovídá také vyšší míra využívání osobních kontaktů a přátelských vazeb jako klíčového adaptačního nástroje. U ruských studentů to uvedlo 85,7 %, zatímco u českých 65,3 %. Pomoc od starších studentů zafungovala u 42,9 % ruskojazyčných respondentů, ale jen u 16,9 % českých. Z těchto dat vyplývá důležitost zavedení mentorského systému a podpory přirozené interakce mezi studenty napříč ročníky.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *30*: *Odkud se nejčastěji dozvídáte o univerzitních událostech? ru-studenty* (vl. zdroj)

Rozdíly jsou patrné také ve zdrojích informací o univerzitních událostech. Zatímco Instagram dominuje u obou skupin, mezi ruskými studenty ho uvedlo 53,1 %, oproti 47,5 % u českých. Čeští studenti však více využívají oficiální web univerzity (18,6 %) než ruští (10,2 %), zatímco ruská skupina více čerpá z offline reklamy (letáky apod.). Tim, aplikace vyžaduje přizpůsobení formy komunikace cílovým zvyklostem jednotlivých skupin.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *31*: *O které typy událostí máte největší zájem? ru-studenty* (vl. zdroj)

V preferencích typů událostí převládá u obou skupin zájem o zábavné akce, avšak u rusky mluvících je tato preference ještě silnější (67,3 %) oproti českým studentům (54,2 %). Kulturní a vědecké akce mají srovnatelnou váhu, a to potvrzuje, že univerzitní prostředí musí nabízet pestrou škálu aktivit nejen akademického, ale i komunitního charakteru, ideálně s podporou nástroje pro vytváření a správu komunitních skupin a kalendářů.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *32*: *S jakými hlavními obtížemi jste se potýkali před či při nástupu na univerzitu? ru-studenty* (vl. zdroj)

V otázce obtíží při nástupu je situace podobná. Významný rozdíl však tvoří jazyková bariéra – u ruských studentů ji uvedlo 40,8 %, u českých jen 3,4 % (a myslím, že to byli také cizí studenti). Pro platformu to znamená nutnost jazykové lokalizace a podpory vícejazyčné komunikace. Také absence známých či komunity byla častěji zmíněna mezi ruskými studenty (63,3 % vs. 48,3 % u českých), potvrzuje to nutnost komunitně orientovaného řešení s důrazem na prvky přednástupového propojení uchazečů a mentorů.

**Изображение выглядит как текст, снимок экрана, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.**

Obrázek *33*: *Do jaké míry by vám pomohla taková platforma (online komunita) ještě před nástupem, abyste se více dozvěděli o univerzitním životě? ru-studenty* (vl. zdroj)

Přínos online platformy před nástupem byl pozitivně hodnocen v obou skupinách, ale ruská skupina častěji udělovala nejvyšší známky – celkem 38,8 % uvedlo hodnotu 5, stejně tak i 38,8 % hodnotu 4. U českých studentů byla větší rozptýlenost odpovědí, přičemž 41,5 % zvolilo 4 a 19,5 % nejvyšší hodnocení. Tato zpětná vazba potvrzuje potřebu přístupného, přehledného a centralizovaného nástroje pro komunikaci, sdílení informací a zapojení nově příchozích studentů ještě před samotným nástupem.

Ve všech ostatních oblastech byly rozdíly mezi skupinami marginální a pohybovaly se ve statisticky nevýznamném rozpětí, a proto nebyly v tomto rozboru podrobněji hodnoceny. Všechny výše uvedené rozdíly by však měly být zohledněny v architektuře, lokalizaci, návrhu rozhraní i komunitních funkcích univerzitní aplikace.

Na základě osobní zkušenosti lze navíc doplnit, že rusky mluvící komunita se často přirozeně odděluje od českého prostředí ve fázi navazování vztahů na univerzitě. Tato tendence není hodnocena ani pozitivně, ani negativně, je však zřejmé, že jedním z důvodů bývá komplikované začlenění na začátku studia, a tím i menší ochota navazovat kontakt mimo jazykovou skupinu. Z tohoto důvodu má navrhovaná platforma i sekundární – osobnější – cíl: propojit jazykově odlišné skupiny v rámci univerzitní komunity a podpořit integraci studentů se zahraniční zkušeností do širšího akademického prostředí. Platforma se tak může stát nejen praktickým nástrojem, ale i prostředím pro vzájemné porozumění a podporu napříč kulturami.

# 5. Požadavky

Na základě provedené analýzy dotazníků a diskusí se studenty vyšlo najevo, že hlavním problémem je roztříštěnost informací a nedostatek centralizace jednotlivých dat a informací, zejména pro nově přijaté a zahraniční studenty. Studenti projevují zájem o centrální platformu, která by sdružovala funkce pro komunikaci, plánování událostí a sdílení studijních materiálů. Klíčovou roli hraje také možnost získat základní informace o univerzitě ještě před nástupem, včetně přehledné prezentace rozvrhu a možností zapojení do zájmových skupin. Tato kapitola proto shrnuje požadavky na navrhovanou aplikaci, které vycházejí z uvedených zjištění. Nejprve jsou představeny funkční požadavky, jež popisují, co by systém měl umět, a poté nefunkční požadavky, zaměřené na aspekty bezpečnosti, dostupnosti či použitelnosti. Následně bude v dalších podkapitolách objasněna struktura databáze a popsáno vizuální modelování procesů prostřednictvím diagramů aktivit a případů užití.

## 5.1 Funkční požadavky

Tabulka 3: Funkční požadavky webové aplikace (zdroj vlastní)

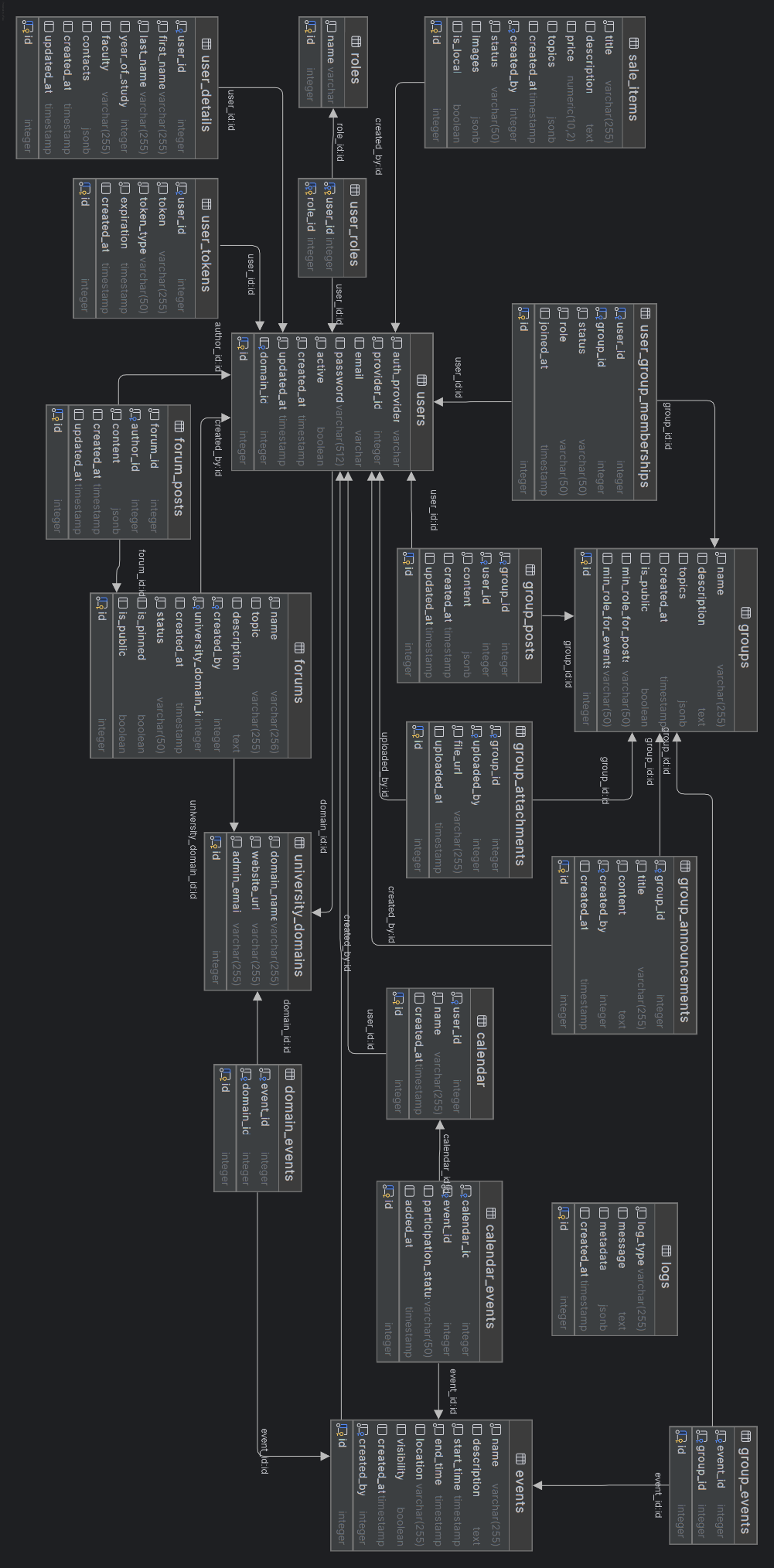
|  |  |  |
| --- | --- | --- |
| F1 | Registrace uživatelů | Aplikace musí umožňovat registraci uživatelů pomocí univerzitního e-mailu či ověřené identity (např. LDAP). |
| F2 | Přihlášení | Aplikace musí zprostředkovat bezpečné přihlášení (login/heslo, případně 2FA) do systému. |
| F3 | Správa profilů | Aplikace musí umožnit úpravu uživatelského profilu (jméno, foto, studijní skupina) a zobrazení profilu. |
| F4 | Fórum a diskuse | Aplikace musí umožňovat vytváření diskusních témat, odpovědí a zobrazování vláken ve fóru. |
| F5 | Komunitní skupiny | Aplikace musí umožňovat zakládat a řídit skupiny (přidávat a odebírat členy, řídit obsah). |
| F6 | Kalendář univerzitních akcí | Aplikace musí poskytovat modul pro plánování a sledování událostí (přednášky, setkání, akce) s možností registrace. |
| F7 | Integrace studijního rozvrhu | Aplikace musí **podporovat načítání rozvrhu studenta** (např. z univerzitního systému) pro zobrazení v kalendáři. |
| F8 | Bazar | Aplikace musí umožňovat vložení a správu inzerátů (prodej, výměna, hledání), a to výhradně přihlášenými uživateli. |
| F9 | Systém notifikací | Aplikace musí umět zasílat notifikace (např. o nových příspěvcích, odpovědích, akcích) buď e-mailem, nebo jinou formou (push, interní upozornění), dle preferencí uživatele. |
| F10 | Lokalizace | Aplikace musí mít překlady do českého a anglického jazyků |

## 5.2 Nefunkční požadavky

Tabulka 4: Nefunkční požadavky webové aplikace (zdroj vlastní)

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | N1 | | Bezpečnost | Aplikace musí implementovat šifrovanou komunikaci (HTTPS), chránit se proti útokům (XSS, CSRF, SQL injection) a umožnit řízení přístupových práv (role). |
| N2 | Dostupnost | |  | | --- | | Aplikace by měla být dostupná 24/7 s definovanou maximální dobou výpadku. Je nutné zavést pravidelné zálohy databáze a případné hot-standby řešení. | |
| N3 | Použitelnost (UX/UI) | Rozhraní musí být responzivní, přehledné a konzistentní. Orientace v aplikaci má být intuitivní pro studenty i správce, s ohledem na zásady UX designu. |
| N4 | Provozní prostředí | Aplikaci musí být možné spustit a nasadit na běžných serverech s OS Linux či Windows (případně kontejnerizovaně), s minimálními softwarovými závislostmi. |
| N5 | Lokalizace a jazyk | Aplikace by měla umožnit vícejazyčné rozhraní (primárně CZ, případně EN) podle nastavení uživatele. |
| N6 | API rozhraní | Aplikace by měla nabízet REST API pro mobilní klienty či jiné systémy (autentizovaný přístup k datům). |
| N7 | Rozšiřitelnost a modularita | Architektura aplikace (např. vícevrstvá MVC) má být navržena tak, aby usnadňovala přidávání nových modulů (např. integrace messengeru, rozhraní pro mobilní aplikace). |

# 6 Databáze



# 7 Diagramy aktivit

V této části jsou představeny hlavní aktivity uživatele při práci s navrženou komunitní platformou, a to včetně autorizace, správy fóra, práce s bazarem a fungování skupin. Jednotlivé diagramy aktivit poskytují přehled klíčových kroků a rozhodovacích bodů, které uživatel nebo systém provádí v rámci konkrétních funkcí.

## 7.1 Hlavní diagram

Изображение выглядит как диаграмма, План, Технический чертеж, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *35*: Hlavní diagram aktivit (vl. zdroj)

Hlavní diagram aktivit znázorňuje obecný postup uživatele po spuštění aplikace. Po případné autorizaci (pokud je vyžadována) může uživatel přistupovat k jednotlivým funkcím, jako je fórum, bazar, skupiny, osobní údaje či správa obsahu. Rozhodovací body umožňují rozlišit, zda je uživatel autorizován, a podle toho mu nabídnout příslušné možnosti (například editaci profilu či vytváření nových příspěvků). Diagram tak poskytuje komplexní pohled na to, jak se uživatel v aplikaci pohybuje a jaké funkce jsou pro něj dostupné v závislosti na úrovni přístupu.

## 7.2 Diagram autorizace

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *36*: Diagram autorizace (vl. zdroj)

Tento diagram se zaměřuje na proces vytváření uživatelského účtu a následné přihlášení. Nejprve uživatel rozhoduje, zda si chce založit nový účet, nebo se přihlásí pomocí stávajících přihlašovacích údajů či externí služby (např. OAuth). Po úspěšné registraci či autentizaci může pokračovat v plném využívání aplikace. Diagram zobrazuje rozhodovací uzly, kdy se ověřuje správnost zadaných údajů, a v případě neúspěchu umožňuje vrátit se k zadávání přihlašovacích či registračních dat.

## 7.3 Diagram fóra

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *37*: Diagram fóra (vl. zdroj)

V diagramu aktivit pro fórum jsou popsány možnosti, které má uživatel k dispozici v rámci diskusních vláken. Může si prohlédnout již existující fórum, vytvářet nová fóra nebo uzavírat stávající (typicky pouze s příslušnými oprávněními). Dále je zde popsána možnost přispívat do fóra, což zahrnuje vkládání příspěvků či komentářů k daným tématům. Výsledkem je ukončení práce ve fóru, ať už formou odhlášení se z diskuse, nebo přechodem k jiné funkci aplikace.

## 7.4 Diagram bazaru

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *38*: Diagram bazaru (vl. zdroj)

Aktivity související s bazarem se zaměřují na práci s inzeráty. Uživatel s příslušnými oprávněními může inzeráty vytvářet nebo je naopak odstraňovat. Všichni uživatelé pak mají možnost prohlížet si aktuální inzeráty a kontaktovat prodávajícího, pokud je zaujme nabízené zboží či služba. Stejně jako u fóra je v závěru aktivity ukončení práce v bazaru, po němž se uživatel může vrátit k jiným sekcím aplikace nebo aplikaci opustit.

## 7.5 Diagram skupin

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Obrázek *39*: Diagram skupin (vl. zdroj)

Diagram aktivit pro skupiny zobrazuje proces správy a účasti ve skupinách. Uživatel může skupinu prohlédnout, vstoupit do ní či z ní odejít, vytvářet nové skupiny nebo v případě vyšších oprávnění skupinu spravovat (například přidávat další účastníky, plánovat události či měnit nastavení skupiny). Tato funkce je užitečná pro organizaci a komunikaci v rámci menších i větších komunit. Po ukončení práce ve skupinách se uživatel může přesunout do dalších sekcí aplikace nebo zcela ukončit svou činnost.

Jak vlastně vypadají ony balónky? Ptají se často lidé. Inu, jak by vypadaly – jako běžné pouťové balónky střední velikosti, tak akorát nafouknuté. Červený se vedle modrého a zeleného zdá trochu menší, ale to je nejspíš jen optický klam, a i kdyby byl skutečně o něco málo menší, tak vážně jen o trošičku. Vítr skoro nefouká, a tak by se na první pohled mohlo zdát, že se balónky snad vůbec nepohybují. Jenom tak klidně levitují ve vzduchu. Jelikož slunce jasně září a na obloze byste od východu k západu hledali mráček marně, balónky působí jako jakási fata morgána uprostřed pouště. Zkrátka široko daleko nikde nic, jen zelenkavá tráva, jasně modrá obloha a tři křiklavě barevné pouťové balónky, které se téměř nepozorovatelně pohupují ani ne moc vysoko, ani moc nízko nad zemí. Kdyby pod balónky nebyla sytě zelenkavá tráva, ale třeba suchá silnice či beton, možná by bylo vidět jejich barevné stíny - to jak přes poloprůsvitné barevné balónky prochází ostré sluneční paprsky. Jenže kvůli všudy přítomné trávě jsou stíny balónků sotva vidět, natož aby šlo rozeznat, jakou barvu tyto stíny mají.



Obrázek 1: Balonky (Min, 2016)

Uvidět tak balónky náhodný kolemjdoucí, jistě by si pomyslel, že už tu takhle poletují snad tisíc let. Stále si víceméně drží výšku a ani do stran se příliš nepohybují. Proti slunci to vypadá, že se slunce pohybuje k západu rychleji než balónky, a možná to tak skutečně je. Nejeden filozof by mohl tvrdit, že balónky se sluncem závodí, ale fyzikové by to jistě vyvrátili. Z fyzikálního pohledu totiž balónky působí zcela nezajímavě. Nejvíc bezpochyby zaujmou děti – jedna malá holčička zrovna včera div nebrečela, že by snad balónky mohly prasknout. A co teprve ta stuha. Stuha, kterou je každý z trojice balónků uvázán, aby se nevypustil. Očividně je uvázaná dostatečně pevně, protože balónky skutečně neucházejí. To ale není nic zvláštního. Překvapit by však mohl fakt, že nikdo, snad krom toho, kdo balónky k obloze vypustil, netuší, jakou má ona stuha barvu. Je totiž tak lesklá, že za světla se v ní odráží nebe a za tmy zase není vidět vůbec.

Tabulka 1: Příklad tabulky

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Zdroj: (Min, 2016)

Když svítí slunce tak silně jako nyní, tak se stuha třpytí jako kapka rosy a jen málokdo vydrží dívat se na ni přímo déle než pár chvil. Jak vlastně vypadají ony balónky? Ptají se často lidé. Inu, jak by vypadaly – jako běžné pouťové balónky střední velikosti, tak akorát nafouknuté. Červený se vedle modrého a zeleného zdá trochu menší, ale to je nejspíš jen optický klam, a i kdyby byl skutečně o něco málo menší, tak vážně jen o trošičku. Vítr skoro nefouká, a tak by se na první pohled mohlo zdát, že se balónky snad vůbec nepohybují. Jenom tak klidně levitují ve vzduchu. Jelikož slunce jasně září a na obloze byste od východu k západu hledali mráček marně, balónky působí jako jakási fata morgána uprostřed pouště. Zkrátka široko daleko nikde nic, jen zelenkavá tráva, jasně modrá obloha a tři křiklavě barevné pouťové balónky, které se téměř nepozorovatelně pohupují ani ne moc vysoko, ani moc nízko nad zemí. Kdyby pod balónky nebyla sytě zelenkavá tráva, ale třeba suchá silnice či beton, možná by bylo vidět jejich barevné stíny - to jak přes poloprůsvitné barevné balónky prochází ostré sluneční paprsky.

# ZÁVĚR

Překvapit by však mohl fakt, že nikdo, snad krom toho, kdo balónky k obloze vypustil, netuší, jakou má ona stuha barvu. Je totiž tak lesklá, že za světla se v ní odráží nebe a za tmy zase není vidět vůbec. Když svítí slunce tak silně jako nyní, tak se stuha třpytí jako kapka rosy a jen málokdo vydrží dívat se na ni přímo déle než pár chvil. Jak vlastně vypadají ony balónky? Ptají se často lidé. Inu, jak by vypadaly – jako běžné pouťové balónky střední velikosti, tak akorát nafouknuté. Červený se vedle modrého a zeleného zdá trochu menší, ale to je nejspíš jen optický klam, a i kdyby byl skutečně o něco málo menší, tak vážně jen o trošičku. Vítr skoro nefouká, a tak by se na první pohled mohlo zdát, že se balónky snad vůbec nepohybují. Jenom tak klidně levitují ve vzduchu. Jelikož slunce jasně září a na obloze byste od východu k západu hledali mráček marně, balónky působí jako jakási fata morgána uprostřed pouště. Zkrátka široko daleko nikde nic, jen zelenkavá tráva, jasně modrá obloha a tři křiklavě barevné pouťové balónky, které se téměř nepozorovatelně pohupují ani ne moc vysoko, ani moc nízko nad zemí. Kdyby pod balónky nebyla sytě zelenkavá tráva, ale třeba suchá silnice či beton, možná by bylo vidět jejich barevné stíny – to, jak přes poloprůsvitné barevné balónky prochází ostré sluneční paprsky. Jenže kvůli všudy přítomné trávě jsou stíny balónků sotva vidět, natož aby šlo rozeznat, jakou barvu tyto stíny mají.

Uvidět tak balónky náhodný kolemjdoucí, jistě by si pomyslel, že už tu takhle poletují snad tisíc let. Stále si víceméně drží výšku a ani do stran se příliš nepohybují. Proti slunci to vypadá, že se slunce pohybuje k západu rychleji než balónky, a možná to tak skutečně je. Nejeden filozof by mohl tvrdit, že balónky se sluncem závodí, ale fyzikové by to jistě vyvrátili. Z fyzikálního pohledu totiž balónky působí zcela nezajímavě. Nejvíc bezpochyby zaujmou děti – jedna malá holčička zrovna včera div nebrečela, že by snad balónky mohly prasknout.

A co teprve ta stuha. Stuha, kterou je každý z trojice balónků uvázán, aby se nevypustil. Očividně je uvázaná dostatečně pevně, protože balónky skutečně neucházejí. To ale není nic zvláštního. Překvapit by však mohl fakt, že nikdo, snad krom toho, kdo balónky k obloze vypustil, netuší, jakou má ona stuha barvu. Je totiž tak lesklá, že za světla se v ní odráží nebe a za tmy zase není vidět vůbec. Když svítí slunce tak silně jako nyní, tak se stuha třpytí jako kapka rosy a jen málokdo vydrží dívat se na ni přímo déle než pár chvil.

# POUŽITÁ LITERATURA

[1] ACKERMANN, Philip. *Full Stack Web Development: The Comprehensive Guide.* Rheinwerk Computing, 2023.

[2] FISHER, Derek. *Application Security Program Handbook: A guide for software engineers and team leaders.* Manning, 2022.

[3] JACKSI, Karwan and Shakir M. Abass. “Development History Of The World Wide Web.” *International Journal of Scientific & Technology Research*, 2019. Dostupné z: <https://www.semanticscholar.org/paper/Development-History-Of-The-World-Wide-Web-Jacksi-Abass/a6b82f4a6caabd0010343793c834e9d5de6cc09c>

[4] CLASSIFYING MODEL-VIEW-CONTROLLER SOFTWARE APPLICATIONS USING SELF-ORGANIZING MAPS, Guamán et al., 2021. Dostupné z: <https://www.semanticscholar.org/paper/Classifying-Model-View-Controller-Software-Using-Guam%C3%A1n-Delgado/cf2875b64995fd753c2be54c3d33ef30f48236e0>

[5] YODGORBEK Komilov. **Android Architecture Patterns: Comparing MVC, MVP, MVVM, and MVI** (2024)., Medium Dostupné z: [https://medium.com/%40YodgorbekKomilo/android-architecture-patterns-comparing-mvc-mvp-mvvm-and-mvi-b282712cfb46](https://medium.com/%40YodgorbekKomilo/android-architecture-patterns-comparing-mvc-mvp-mvvm-and-mvi-b282712cfb46%20)

[6] **KURAPATI, Lakshmanarao.** (2024). Different Types of Architectures in Frontend Design and Development. *International Journal for Multidisciplinary Research*, IJFMR240528707, September–October 2024. Dostupné z: <https://www.researchgate.net/publication/388486719_Different_Types_of_Architectures_in_Frontend_Design_and_Development>

[7] **ASIMIYU, Zainab.** (2024, November). Evaluating Android Architectural Patterns: A Deep Dive into MVP, MVVM, and MVI. Dostupné z: <https://www.researchgate.net/publication/385492745_Evaluating_Android_Architectural_Patterns_A_Deep_Dive_into_MVP_MVVM_and_MVI>

[8] **Performance Comparison of Native Android Application on MVP and MVVM**, Bambang Wisnuadhi, Ghifari Munawar, & Ujang Wahyu. (2020). Dostupné z: <https://www.researchgate.net/publication/348060919_Performance_Comparison_of_Native_Android_Application_on_MVP_and_MVVM>

[9] Choma, D., Chwaleba, K., & Dzieńkowski, M. (2022). The Efficiency and Reliability of Backend Technologies: Express, Django, and Spring Boot. *International Journal of Advanced Computer Science and Applications*, 13(5). Dostupné z: <https://www.researchgate.net/publication/376709512_THE_EFFICIENCY_AND_RELIABILITY_OF_BACKEND_TECHNOLOGIES_EXPRESS_DJANGO_AND_SPRING_BOOT>

[10] Shaikh, A. (2025). Node.js vs Django vs Laravel: Which is the best back-end web framework? *Peerbits Blog*. Dostupné z: <https://www.peerbits.com/blog/nodejs-vs-django-vs-laravel-back-end-web-framework.html>

[11] Odeniran, Q. (2023). *Comparative Analysis of Fullstack Development Technologies: Frontend, Backend and Database*. Georgia Southern University.

[12] Kramer, N. (2024). Backend Frameworks List: Choosing the Right One. *Daily.dev Blog*. Dostupné z: <https://daily.dev/blog/backend-frameworks-list-choosing-the-right-one>

[13] Framework Training. (2024). Spring Boot vs. Node.js vs. .NET Core vs. Django: a “celebrity face-off”. *Framework Training News & Insights*. Dostupné z: <https://www.frameworktraining.co.uk/news-insights/spring-boot-vs-node-js-versus-dot-net-core-vs-django/>

[14] Odeniran, Qozeem. „Comparative Analysis of Fullstack Development Technologies: Frontend, Backend and Database.“ Georgia Southern University, 2023. Dostupné z: <https://digitalcommons.georgiasouthern.edu/etd/2663>

[15] Li, Nian a Bo Zhang. „The Research on Single Page Application Front-end development Based on Vue.“ Journal of Physics: Conference Series, 2021. Dostupné z: <https://www.researchgate.net/publication/351145453_The_Research_on_Single_Page_Application_Front-end_development_Based_on_Vue>

[16] CHAMBERLIN, Donald. "50 Years of Queries." Communications of the ACM, vol. 67, no. 8, August 2024, pp. 110–121. doi:10.1145/3649887.

[17] Docker, Inc. "Four Ways Docker Boosts Enterprise Software Development." Dostupné z: <https://www.docker.com/blog/four-ways-docker-boosts-enterprise-software-development/#:~:text=For%20workloads%20targeting%20on,are%20still%20dominant>

[18] BMC Software. "State of Containers." Dostupné z: <https://www.bmc.com/blogs/state-of-containers/#:~:text=The%20primary%20reason%20for%20leveraging,%E2%80%9D>

[19] Zero To Mastery. "Angular vs React vs Vue: Which Framework Should You Choose?" Dostupné z: <https://zerotomastery.io/blog/angular-vs-react-vs-vue/#:~:text=Angular%20vs%20React%20vs%20Vue%3A,Detailed%20Documentation%3A%20Vue%27s>

[20] PostgreSQL. Dostupné z: <https://www.postgresql.org/>

[21] Integrate.io. "PostgreSQL vs MySQL: Which One Is Better for Your Use Case?" Dostupné z: <https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/#:~:text=Here%20are%20some%20of%20the,defining%20characteristics%20of%20PostgreSQL>

[22] Spring Boot. Dostupné z <https://spring.io/projects/spring-boot>

[23] Spring. Dostupné z <https://spring.io/>

[24] IJFMR. (2024). Spring Boot, introduced by Pivotal, enhances productivity in enterprise application development. International Journal for Multidisciplinary Research, IJFMR28930, Dostupné z: <https://www.ijfmr.com/papers/2024/5/28930.pdf>

[25] Vue.js. Dostupné z: <https://vuejs.org/>

[26] Invedus. "Angular vs React vs Vue.js – A Comparative Study." Dostupné z: <https://invedus.com/blog/angular-vs-react-vs-vue-js-a-comparative-study/>

[27] Facebook. Dostupné z: <https://www.facebook.com>

[28] Reddit. Dostupné z: <https://www.reddit.com>

[29] Telegram. Dostupné z: <https://telegram.org>

[30] LinkedIn. Dostupné z: <https://www.linkedin.com>

[31] Instagram. Dostupné z: [https://www.instagram.com](https://www.instagram.com/)

[32] Spring Initializr Dostupné z <https://start.spring.io/>

[33] TRĘTOWICZ M., ZALEWSKI A., TURKOWSKI J., MVC MVP and MVVM architecture pattern – Introduction, itCraft, Dostupné z <https://itcraftapps.com/blog/mvc-mvp-and-mvvm-architecture-pattern-introduction/>

[34] BUDHABHOOSHAN Patil, Understanding MVVM-C, LinkedIn, Dostupné z: <https://www.linkedin.com/pulse/understanding-mvvm-c-budhabhooshan-patil/>

ГОЛ работы на будущее

<https://www.researchgate.net/publication/357842727_Web_Development_and_performance_comparison_of_Web_Development_Technologies_in_Nodejs_and_Python#:~:text=>,

В ожидании полной работы

<https://www.researchgate.net/publication/343421705_JSP_Application_Architectures>

в ожидании полной работы, в описаны JAVA EE JSP

<https://zerotomastery.io/blog/angular-vs-react-vs-vue/#:~:text=Angular%20vs%20React%20vs%20Vue%3A,Detailed%20Documentation%3A%20Vue%27s>

- статья сравнения ангуляр и вью и реакта

Синим – дополнить цитирование

Красным – источник под вопросом

Зеленым – Всё заебись

Желтый – стоит разобраться или хотя бы почитать источник

SEZNAM PŘÍLOH

Příloha A: Název přílohy

**PŘÍLOHA A*:* Název přílohy**

Vložit přílohu, popis přílohy A, zdroj.