Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Možnosti využití frameworků pro tvorbu UA (Universal App) v modelových situacích

Bakalářská práce

Autor: Jiří Žák
Studijní obor: Aplikovaná informatika

Vedoucí práce: Mgr. Hana Rohrová

Hradec Králové duben 2024

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 17. 4. 2024

Jiří Žák

Poděkování:

Rád bych tímto poděkoval Mgr. Haně Rohrové, vedoucí mé bakalářské práce za cenné rady a věcné připomínky k jejímu metodickému vedení.

Abstrakt

Tato bakalářská práce se zabývá problematikou univerzálních (izomorfních) aplikací a jejich významem pro použití jako náhradu za aplikace jednostránkové. Za cíl si klade zejména zasvětit čtenáře do oblasti architektur webových stránek, představit několik základních technik jejich vykreslování a ty následně demonstrovat na několika běžných ukázkových aplikacích. Ukázkové jednostránkové aplikace využívají framework Vue.js a univerzální aplikace z nich vycházející jsou postaveny na frameworcích Quasar.js a Nuxt 3. Významem empirického výzkumu je v první řadě evaluovat výkonnostní rozdíly při použití univerzálních aplikací a jednostránkových aplikací. Mimo výkonnostní dopad se výzkum zabývá také klíčovou rolí univerzálních aplikací v oblasti SEO a podporou uživatelů bez JavaScriptu. V neposlední řadě práce naznačuje proces migrace z jednostránkové aplikace na univerzální a vytyčuje několik relevantních aspektů, které je vhodné brát v úvahu při výběru frameworku sloužícího pro tvorbu univerzálních aplikací.

Abstract

Title: Exploring the Applications of Universal App (UA) Frameworks in Model Scenarios

This bachelor thesis addresses the issue of universal (isomorphic) applications and their significance as replacements for single-page applications. Its main objective is to introduce the reader to the field of web architectures, present several basic techniques of their rendering, and subsequently demonstrate them on several common sample applications. The sample single-page applications utilize the Vue.js framework, while the universal applications derived from them are built on the Quasar.js and Nuxt 3 frameworks. The significance of empirical research is primarily to evaluate the performance differences between using universal applications and

single-page applications. In addition to performance impact, the research also examines the key role of universal applications in SEO and supporting users without JavaScript. Lastly, the thesis outlines the migration process from single-page applications to universal ones and highlights several relevant aspects to consider when selecting a framework for building universal applications.

Klíčová slova: jednostránkové aplikace (SPA), univerzální aplikace (UA), izomorfní aplikace (IA), techniky vykreslování webu, Vue.js, Quasar.js, Nuxt 3

Key words: single-page application (SPA), universal application (UA), isomorphic application (IA), web rendering techniques, Vue.js, Quasar.js, Nuxt 3

Obsah

Seznam obrázků

Seznam příkladů

Seznam tabulek

1 Úvod

"*If you want a great site, you've got to test. After you've worked on a site for even a few weeks, you can't see it fresh anymore. You know too much. The only way to find out if it really works is to test it.*" - Steve Krug

Když jsem před lety objevil framework Vue.js a společně s ním architekturu SPA (Single-Page Application), byl jsem unesen reaktivní provázaností proměnných a celkově snadnějším vývojem než u PHP (bez frameworku), se kterým jsem pracoval předtím. Nadšení z technologie bylo takové, že jsem si dlouhé měsíce nevšiml, že všechno dobré má i své stinné stránky. Tím špatným bylo v tomto případě SEO (Search Engine Optimization). Mé stránky byly z pohledu vyhledávačů prázdné a možným řešením byl přechod na architekturu UA (Universal Application). Dnes jsou jak SPA, tak UA již dobře známy a v praxi hojně využívány. Přestavují nedělitelný pilíř webového prostoru, a s každým dnem jejich popularita a vliv i nadále rostou. Proto budou v této práci evaluovány a výsledky budou předloženy čtenáři. V první řadě budou zachyceny informace o jednotlivých webových architekturách umožňující pochopení toho, co jsou a jak fungují architektury SPA a UA. Na tyto bude úzce navázáno popisem vybraných technik vykreslování na webu, z jejichž použití vychází typ zmíněné architektury, a které současně mají vliv i na další aspekty jako SEO, podpora uživatele bez JS (JavaScript) a výkonnost. Tyto vytyčené aspekty budou poté podrobeny testu v rámci empirického měření, na základě něhož bude následně rozhodnuto o přínosu UA architektury v porovnání s architekturou SPA.

Zvláštní pozornost bude poté věnována technice On-Demand ISR (Incremental Static Regeneration), jedné z novějších vykreslovacích technik aktuálně dostupných na trhu. Pro tuto jedinečnou techniku umožňující znovupoužitelnost vykreslení dynamických stránek, bude evaluována implementovatelnost a vytvořeno konkrétní řešení umožňující její de facto automatický provoz.

2 Cíl a metodika práce

Smyslem práce je uvést čtenáře do problematiky moderního webu popisem základních architektur, které lze v kontextu webu najít. Na webové architektury navázat stručným výčtem dnes používaných vykreslovacích technik, používaných nejen v rámci UA. Dále zvolit konkrétní frameworky, navrhnout a vytvořit SPA pro jednotlivé modelové situace, které poté migrovat na věrné kopie využívající architekturu UA, a to pro účely empirického kvantitativního šetření. V rámci šetření poté zpracovat a porovnat jednotlivé rozdíly spojené s vývojem, funkcionalitou a výkonem UA v kontrastu s SPA.

Proces migrace aplikace byl zvolen především k dosažení co nejvyšší míry podobnosti, mezi zdrojovými SPA a koncovými UA, ale také proto, že frameworky stavějící na určitém frameworku by se neměly v základech strukturálně a principiálně příliš vzdalovat od zdrojového frameworku, a to právě pro ulehčení procesu migrace. Modelové aplikace byly vybrány tak, aby co nejvěrněji imitovaly aplikace používané v reálném nasazení, a výsledky měření tak vypovídaly spíše o využití v praxi než vytvořením generické stránky přesně šité pro dosažení nejlepších výsledků v rámci testovaných aspektů. Výběr použitých frameworků byl proveden na základě osobního uvážení autora a výsledky zjištěné pro vybrané frameworky se mohou pro jiné frameworky lišit.

Vzhledem k současnému stavu zkoumané domény ve vědeckých materiálech, které nejsou z důvodu relativní čerstvosti domény a jejího rychlého vývoje ještě příliš obsažné, byly zvoleny především méně autoritativní zdroje dostupné volně na internetu. V rámci práce byla použita umělá generativní inteligence, a to k vygenerování obrázků použitých na modelových stránkách.

3 Architektury webových stránek

Architektura webové stránky je úzce spjata, ne-li přímo definována, použitým typem vykreslování na stránce. V současné době se můžeme setkat s dvěma široce uznávanými a již dobře známými architekturami SPA s MPA. Propojení těchto dvou architektur za pomoci moderních webových technologií dalo vzniknout třetí, prozatím širokou veřejností neuznávané architektuře UA. V této kapitole budou jednotlivé architektury, které lze při vývoji aplikací použít krátce představeny.

3.1 MPA (Multi-Page Application)

Na základě článku Cleveroad (Altynpara & Ropalo, 2023) v současnosti stále nejčastější architektura webových stránek, a to z důvodu, že mluvíme o první architektuře webových stránek. Jak již ve svém článku (Mamgain, 2022) naznačil D. Mamgain, front-end manažer ve společnosti KOCO, jedná se o aplikace, které se skládají z „*více stránek*“. Jedná se o validní argument, ale je to velice abstraktní pojem bez konkrétní výpovědní hodnoty, jelikož moderní webové aplikace typicky obsahují vizuální navigaci mezi více stránkami bez podmíněnosti architektury MPA. K jednoznačnému definování architektury je nutné zabřednout více do hloubky jejího fungování, v tomto případě konkrétně vykreslování stránky. P. Skólski (Neoteric, 2016) architekturu popsal jako „*tradiční*“, kde každé zobrazení „*dynamických*“ dat, či jejich odeslání vyžaduje nový požadavek na server, který vykreslí a vrátí stránku s novým obsahem. Podobně je architektura v tradičním slova smyslu definována na stránkách Cleveroad (Altynpara & Ropalo, 2023), kde oba zdroje současně uvádí její neplatnost v praxi, kde se již hojně využívá technologie AJAX (Asynchronous JavaScript and XML). Tato dle IBM (IBM, 2021) poskytuje metody komunikace mezi klientem a serverem právě bez zmíněného obnovení celé stránky. Což nás přivádí k následující architektuře webových aplikací. V českém jazyce je označována termínem „*vícestránkové aplikace*“.

3.2 SPA (Single-Page Application)

Dobře známý český web Itnetwork (Veverka, nedatováno) uvádí v lekci o technologii AJAX její využití u SPA architektury následujícím způsobem: *„AJAX stahuje nové stránky a data na pozadí. Poté JavaScript do části stránky, která má být měněna, vloží nový obsah“*. Ve zkratce lze architekturu definovat dle P. Skólski (Neoteric, 2016) a Cleveroad (Altynpara & Ropalo, 2023) jako jednoduchou stránku, co při navštívení načte všechen další obsah za pomocí JavaScriptu, většinu obsahu již není třeba při další

návštěvě stahovat, a je znovu využit. Všechny tři zmíněné články dále vyvozují, že tyto aplikace jsou nejčastěji tvořeny za pomoci nějakých JavaScript frameworků výraznou měrou usnadňujících jejich vývoj. Příklady takových frameworků jsou Angular, React nebo Vue, se kterým bude dále zacházeno v této práci. V českém jazyce je označována termínem „*jednostránkové aplikac*e“.

3.3 UA (Universal Application)

Také označována jako „*Isomorphic Application*“, vycházející z termínů „*Isomorphic JavaScript*“ a „*Universal JavaScript*“, které v kontextu webových aplikací, dle Airbnb (AirbnbEng, 2013) značí aplikační logiku, jenž je schopna běhu jak na straně serveru, tak klienta. Jak již bylo avizováno, nejedná se o ustálený termín. V minulých letech se objevili tací, kteří se o jasné vymezení těchto pojmů pokoušeli, jako např. G. Hengeveld v článku (Hengeveld, 2015). Bohužel do dnešního dne nebyl tento konflikt dořešen, a proto izomorfní a univerzální vystupují jako prostá synonyma dle (admin, nedatováno) a (Isomorphic react, nedatováno). V této práci bylo přikloněno k termínu univerzální z důvodu návaznosti na framework Nuxt 3 (Rendering Modes, nedatováno), který termín využívá. Samotná architektura spočívá v kombinaci MPA a SPA, jak uvádí autor knihy (Gordon, nedatováno), konkrétně pak tím způsobem, kdy první požadavek na načítání stránky je zpracován jako MPA, načež přebírá kontrolu SPA architektura, jak je uvedeno na stránkách Lullabot (NR, 2015). V českém jazyce je označována termíny „*univerzální aplikace*“ či „*izomorfní aplikace*“.

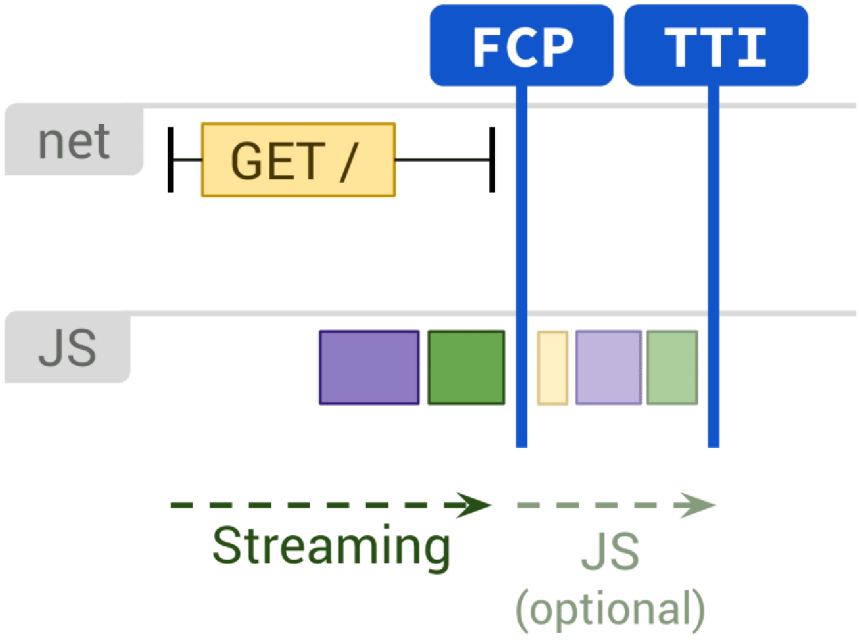
4 Vykreslení webové stránky

Proces vykreslování webové stránky je na základě slov S. Naylor, zaměstnance známé technologické společnosti Netlify (Naylor, 2023) proces, při kterém dochází ke generaci HTML (Hypertext Markup Language) značení, sloužícímu k vykreslení stránky. M. Ibsen ve svém článku o vykreslování na webu pro Medium (Ibsen, 2021), apeluje na nutnost znalosti základních vykreslovacích technik, jelikož nesprávný výběr těchto, může mít za následek přepisování částí kódu, či dokonce migraci na úplně jiný set technologií, čímž lze dle autorových slov, ušetřit nejen peníze a čas, ale také řadu starostí. Naylor (Naylor, 2023) dále uvádí dvě základní kategorie členění vykreslovacích technik jako následující: *„How, and most importantly, where“* (vol. přel. z angličtiny: jak, a především kde). Tyto mohou mít podle autorky zásadní dopad na uživatelský zážitek, výkon a SEO. Na základě zmíněných článků a dalších pozdějších zdrojů, byl sestaven seznam následujících základních technik.

4.1 Static Rendering (SR)

Jedná se o nejpřímočařejší a nejstarší vykreslovací techniku, která v minulých letech, kdy ještě nebyly nalezeny jiné techniky, byla tato, jak uvádí Naylor (Naylor, 2023) využita na všech webových stránkách. Definuje je jakožto kolekci ručně psaných HTML souborů, které jsou přímo bez dalšího zpracování odesílány uživateli. Načež navazuje definice vývojáře G. Cocca v článku (Cocca, 2023) pro freeCodeCamp, který ji definuje jako kolekci HTML, CSS (Cascading Style Sheets) a JS souborů odesílanou klientům bez zpracování na straně serveru a integrace databázového systému. I přes pokrok vykreslovacích technik se dle Naylor (Naylor, 2023) stále jedná o užitečnou techniku, která se mimo jiné hodí např. pro tvorbu úvodních stran.

Tato technika by měla, jak uvádí Osmani a Miller ve svém blogu (Osmani & Miller, 2019) poskytnout poměrně velmi dobrý TTFB (Time To First Byte), nízký čas FCP (First Contentful Paint) a stejně tak nižší TBT (Total Blocking Time). Důvodem je odesílání již vykreslené stránky, kterou klientu stačí pouze interpretovat. Techniku ilustruje následujícím obrázkem, imitujícím výkonnostní výstup ve vývojářských nástrojích:

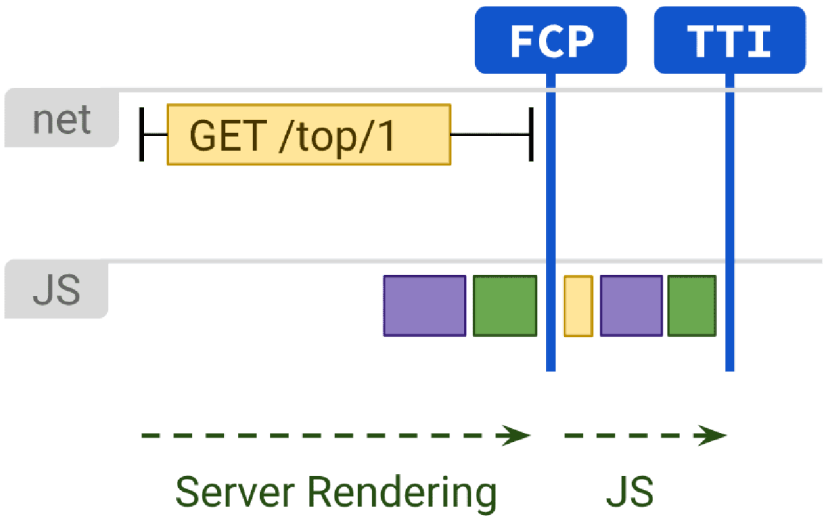


Obr. 1: Grafická ilustrace SR/SSG, zdroj (Osmani & Miller, 2019)

4.2 Server-Side Rendering (SSR)

V blogu (Osmani & Miller, 2019) vedeném primárně vývojáři prohlížeče Chrome, autoři techniku popisují jako generaci celého HTML značení stránky na straně serveru v reakci na navigaci, jinak řečeno, dle slov M. Ibsen *„Just In Time"* (JIT), tedy na základě P. Ram (Ram, 2021), vykreslování probíhá za běhu v čase požadavku. Což jak Ram dále uvádí, eliminuje potřebu dalšího získávání dat a šablonování na straně klienta. Naylor (Naylor, 2023) definuje motiv vzniku této techniky jako reakci na potřebu více dynamického obsahu, a tím i větších webových stránek, které by nebylo možné statickým renderováním efektivně provozovat.

Pro tuto techniku Osmani a Miller (Osmani & Miller, 2019) definují potencionálně nízký čas FCP a nižší TBT, na druhou stranu definují riziko vyššího TTFB. Jako důvod uvádí fakt, že k vykreslení dochází na serveru v čase požadavku, což na jednu stranu zajistí doručení již vykreslené stránky klientovi, který ji musí pouze zobrazit, tudíž je docíleno nízkého času FCP. Na druhou stranu pozdrží vykreslování na straně serveru odesílání samotné stránky, což vede k vyššímu TTFB. Techniku ilustruje následujícím obrázkem, imitujícím výkonnostní výstup ve vývojářských nástrojích:

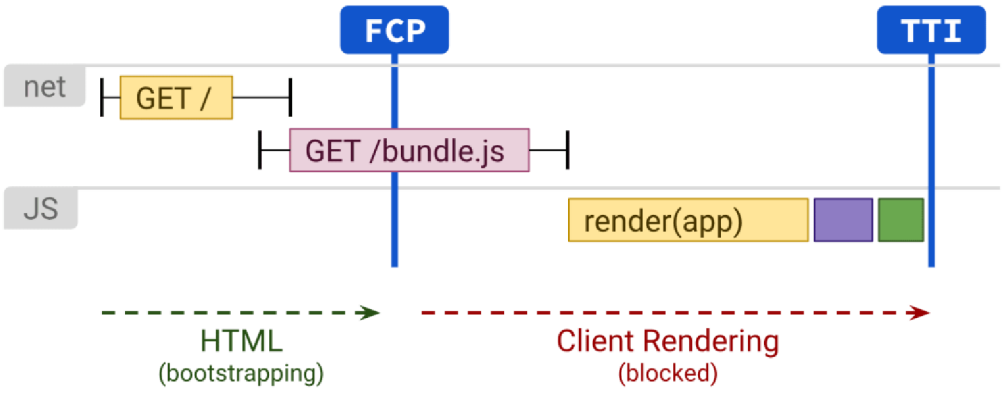


Obr. 2: Grafická ilustrace SSR, zdroj (Osmani & Miller, 2019)

4.3 Client-Side Rendering (CSR)

Ibsen (Ibsen, 2021) říká: *„Client-side rendered websites do everything on their own“*. Konkrétnějšími slovy vývojářů Chrome (Osmani & Miller, 2019) to znamená vykreslování stránek přímo v prohlížeči, za pomocí jazyka JavaScript. Oba zdroje detailně popisují, že prohlížeč tedy vykonává všechnu logiku, routování (směrování) a získávání dat z API (Application Programming Interface). K vykonání všech zmíněných aktivit potřebuje aplikace spoustu dat, která musí být klientovi přenesena. Což jak hovoří vývojáři Chrome (Osmani & Miller, 2019), může mít negativní dopad na načítání stránky, a to především v případě, že je požadována na slabším internetovém připojení či méně výkonném zařízení. Naylor (Naylor, 2023) připisuje nárůst popularity této techniky ruku v ruce se zvýšenou adopcí jazyka JavaScript v prohlížečích, tato i nadále rostla s příchodem moderních front-end frameworků, jako např. Angular, React nebo Vue.

Techniku znovu popisují Osmani a Miller (Osmani & Miller, 2019) a jako její největší slabinu, definují riziko vysoké hodnoty TBT. Celestial Systems (Team Product Engineering, nedatováno) déle na základě zasílání de facto prázdného HTML souboru předpokládají velmi nízké TTFB a průměrné FCP. Osmani a Miller (Osmani & Miller, 2019) techniku ilustrují následujícím obrázkem, imitujícím výkonnostní výstup ve vývojářských nástrojích:



Obr. 3: Grafická ilustrace CSR, zdroj (Osmani & Miller, 2019)

4.4 Static Site Generation (SSG)

SSG je technika založená na kombinaci statického vykreslování a vykreslení na straně serveru, na základě Ibsen (Ibsen, 2021) také zvaná „*Static-Side Rendering*“. G. Cocca (Cocca, 2023) techniku popisuje jako generaci HTML, CSS a JS v předstihu, ve fázi zvané build (přel. z angličtiny: stavba) místo na serveru nebo klientu při vytvoření požadavku. Dle Ibsen (Ibsen, 2021) bychom také mohli říci, že stránky jsou vykresleny *„Ahead Of Time“* (AOT). De facto je tedy stránka předem vygenerována jako HTML značení na základě SSR, a poté dále využívána jako SR. Naylor (Naylor, 2023) uvádí vzestup techniky na počátku 21. století s příchodem nástrojů jako Jekyll.

Z výkonnostního hlediska by měla technika dosahovat parametrů zmíněných v kapitole věnující se „*Static Rendering*“.

4.5 Incremental Static Regeneration (ISR)

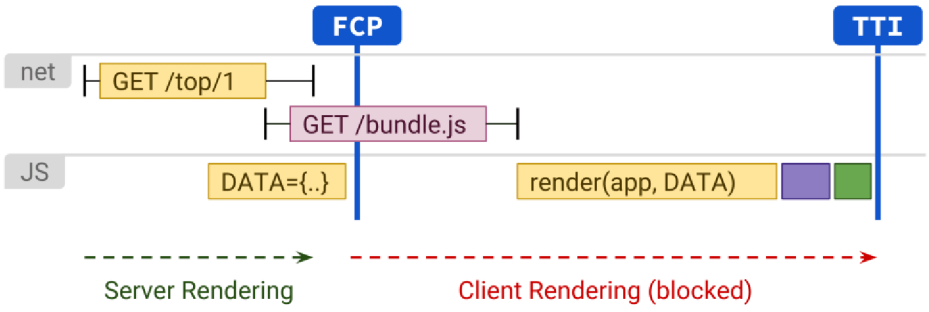
V tradičním slova smyslu se dle G. Cocca (Cocca, 2023) jedná o kombinaci SSR a SSG. Autor uvádí následující cyklus událostí počínající SSG generací stránek, kde každý takový vygenerovaný soubor disponuje souborem metadat obsahujícím údaje o expiraci obsahu. Dle Naylor (Naylor, 2023) i Cocca (Cocca, 2023) je tento statický soubor odesílán jako odpověď, dokud nedojde k jeho expiraci, kdy je tento vygenerován „*JIT*“ a následně uložen pro další odesílání, dokud nedojde znovu k jeho expiraci. Naylor (Naylor, 2023) techniku uvádí jako proprietární pro framework Next. Tato byla dále přebrána např. frameworkem Nuxt 3 (Rendering Modes, nedatováno). Na základě dokumentací zmíněných frameworků (Incremental Static Regeneration (ISR), nedatováno) (Rendering Modes, nedatováno) je expiraci možné definovat jako číslo vyznačující počet sekund, po kterých je třeba obsah obnovit. Nebo ještě více experimentálním způsobem, představeným týmem Next (Incremental Static Regeneration (ISR), nedatováno) zvaným On-Demand Revalidation. V popsaném případě lze expiraci „*manuálně*“ vyvolat požadavkem na speciální API bod, který tuto expiraci provede. Lze tedy říct, že k vykreslování a ukládání dochází pouze za předpokladu, že stránka zatím vygenerována nebyla či vypršela její platnost. Platnost stránky je obnovena při každém jejím vykreslení.

Z výkonnostního hlediska by měla technika dosahovat parametrů zmíněných v kapitole věnující se „*Server-Side Rendering*“ či „*Static Site Generation*“ v závislosti na platnosti a existenci stránky.

4.6 Universal Rendering (UR)

Celestial Systems (Team Product Engineering, nedatováno) považují UR za kombinaci SSR a CSR, k dosažení nejlepšího z obou technik. Vývojáři Chrome ve svém článku (Osmani & Miller, 2019) vyjadřují návaznost těchto technik a proces zvaný „*hydratace*“ či „*rehydratace*“, který dle dokumentace frameworku Nuxt 3 (Rendering Modes, nedatováno) slouží k obnovení interaktivity webové stránky ze statického HTML značení poskytnutého serverem. Chrome vývojáři (Osmani & Miller, 2019) hovoří o tom, že plné načtení stránky, či její obnovení je zpracováno technikou SSR, načež je do značení výsledné HTML stránky zabudována sekce obsahující data, se kterými SSR pracovalo a identicky jako v CSR přiložen JS vstupní soubor daného frameworku, který se postará o hydrataci, a tím převezme kontrolu nad aplikací technika CSR. Jak uvádí J. Josef (Josef & Malý, 2016) ve své diplomové práci, tato technika by měla mít pozitivní vliv na prvotní načtení stránky, které by mělo být rychlejší než při použití samotného CSR.

I poslední technika je popsána vývojáři Chrome (Osmani & Miller, 2019). Uvádí, že při správném použití může mít za efekt snížení FCP vůči původní CSR technice. Upozorňují, že stále hrozí vysoké riziko neoptimálního TBT, načež hodnota TTFB se mění v závislosti na použitém vykreslování na serveru, jelikož je možné CSR tímto způsobem kombinovat s libovolnou dříve zmíněnou technikou mimo zmíněnou CSR. Techniku v tomto případě Osmani a Miller ilustrují následujícím stylem:



Obr. 4: Grafická ilustrace UR, zdroj (Osmani & Miller, 2019)

4.7 Vykreslování a SEO

Forma vykreslování stránky nemá, jak uvádí vývojář Lapinski (Lapinski, 2022), dopad pouze na výkonností metriky. Jak, nebo lépe řečeno kde je obsah stránky vykreslen, je zásadní pro tzv. „search engine optimization“ (vol. přel. z angličtiny: optimalizace pro vyhledávače), neboli SEO. Autor v blogu popisuje vývoj webových technologií na čemž demonstruje důvod proč, jak uvádí „bots like HTML“ (vol. přel. z angličtiny: crawleři mají rádi HTML). Odpověď na tuto otázku je prostá, od počátku byly navrženi pro práci s kompletním HTML a potřeba změny byla zaznamenána až s příchodem, či širším rozšířením SPA, jenž pracuje s de facto prázdným HTML souborem. Vyhledávače postupně začaly budovat vestavěnou podporu SPA pro vyhledavače, ale jak uvádí výzkum od Unless (Which search engines index Javascript content?, nedatováno) z roku 2018 testující schopnosti práce vybraných vyhledávačů s JavaScriptem, výsledky nebyly příliš zdárné. S přehledem nejlépe si vedl vyhledávač Google, jenž splnil všechna testovaná očekávání, na druhou stranu vyhledávače jako Bing či Baidu nezískaly ani jeden jediný bod. I přes možné změny k dnešnímu dni se nejedná o naprosto spolehlivou metodu pro SEO. Jak uvádí Google v článku (Understand the JavaScript SEO basics, 2024) popisujícím korelaci SEO a JavaScriptu, je vhodné splnit mnoho pravidel, aby bylo zajištěno co nejlepších výsledků indexování. Současně jak uvádí Lipinski (Lapinski, 2022) k vykreslování dochází až s odezvou, což může mít za následek neaktuálnost indexovaných dat. Další výhodou přístupu, kdy k vykreslení stránky nedochází na straně klienta, je udržení základní funkcionality stránky i bez použití Javascriptu na straně klienta, a to právě z důvodu, že obdrží již vygenerované značení pro vykreslení, je ale nutné dodat, že taková stránka neprojde fází hydratace a poskytuje tedy pouze minimální interaktivitu. Takováto stránka může např. umožňovat přechod mezi stránkami, není to ale pravidlem.

Možnými řešeními překlenutí nedostatků SPA z hlediska SEO jsou techniky UR a „*Dynamic Rendering*“, jenž ale Google v článku jemu věnovaném (Dynamic rendering as a workaround, 2024) již označuje jako dočasné řešení.

5 Představení frameworků

Existuje mnoho frameworků, které vyvstanou na mysli při návrhu SPA aplikace. Nejčastěji zmiňovanými jsou React, Angular a Vue.js, mimo to jsou v článku pro Medium (Vitarag, 2023) pojednávajícím o osmi nejlepších SPA frameworcích dále zmíněny Ember.js, Svelte, Aurelia, Backbone.js a Mithril. Tato práce pojednává o jednom z prvních třech zmíněných frameworků a to konkrétně Vue.js.

5.1 Vue.js

Na základě úvodní sekce oficiální dokumentace Vue.js (Introduction, nedatováno) se jedná o JavaScript framework sloužící k tvorbě uživatelského rozhraní postavený nad standardním HTML, CSS a JS umožňující tvorbu jednoduchých i komplexních rozhraní. Autoři Yi a Li vyvozují v konferenčním sborníku (Yi & Li, 2021) využití modelu Model-View-ViewModel (MVVM), jenž dle E. Gallardo (Gallardo, 2023) slouží k oddělení aplikační logiky a uživatelského rozhraní, stejně tak jako lépe známá architektura MVC. Gallardo (Gallardo, 2023) dále uvádí, že díky nahrazení C (Controller) za VM (ViewModel) byla umožněna obousměrná komunikace. Časté otázky na stránkách Vue.js (Frequently Asked Questions, nedatováno) poté o Vue.js hovoří jako o nezávislém, komunitou řízeném projektu, který byl vytvořen roku 2014 bývalým vývojářem společnosti Google jménem Evan You. Sekce také uvádí velice dobré výkonnostní metriky v porovnání s dalšími SPA frameworky a tyto podkládá důkazy.

5.2 Quasar.js

V dokumentaci frameworku Quasar.js (Why Quasar?, nedatováno) lze vyčíst, že se jedná o open-source řešení postavené na Vue.js, poskytované pod licencí MIT. Sekce dále uvádí jednoduché motto tohoto frameworku, jenž zní následovně *„write code once and simultaneously deploy it as a website, a Mobile App and/or an Electron App“.* Softwarový vývojář J. Bhatt ve svém článku na platformě Medium (Bhatt, 2020) uvádí jako hlavní výhodu právě zmíněnou možnost vývoje pro více platforem v jednom. Déle Bhatt (Bhatt, 2020) vyzdvihává vestavěnou knihovnu UI (User Interface), velmi dobrou strukturu aplikace a v neposlední řadě snadnou znuvupoužitelnost kódu. První verze byla uvedena v roce 2019 a framework je aktivní do dnes (2024) viz (Quasar Framework Roadmap, nedatováno).

5.3 Nuxt 3

Domovská stránka Nuxt 3 (The Intuitive Vue Framework, nedatováno) framework definuje jako open source řešení umožňující intuitivní vývoj s cílem vytvořit výkonnou full-stack aplikaci připravenou k okamžitému produkčnímu nasazení. Mezi silné stránky frameworku, patří na základě domovské stránky Nuxt 3 (The Intuitive Vue Framework, nedatováno) a blogu L. Mausera (Mauser, 2023) systém hybridního routování, pokročilé mechanismy pro získávání dat ze serveru, jednoduchá správa stavu (state management) a automatické importy. Druhou stranou mince je jak Mauser (Mauser, 2023) posléze hovoří velice striktní přístup frameworku, a právě zmíněné pokročilé mechanismy. Kde kombinace těchto může při vývoji způsobit neznalému vývojáři hodně zmatení a potíží. Byl vyvinut v reakci na framework Next.js, a to první verzí v roce 2018 jak uvádí (Onyenma, 2023).

6 Vývoj demonstračních SPA aplikací

V minulých kapitolách byl čtenář obeznámen s teoretickým skeletem této práce. Mimo jiné byl nastíněn koncept architektury SPA, nyní budou předvedeny kroky k vypracování série aplikací postavených na této architektuře. Všechny zdrojové kódy vytvořené v rámci práce jsou veřejně dostupné na GitLab repositáři https://gitlab.com/jirka124/bachelor-thesis, stejně tak lze v repositáři pro reprodukci najít testovací databáze a výsledná data výzkumu. Data jsou k dispozici v plné verzi popisující porovnání výsledků každé stránky modelové aplikace s korespondujícími stránkami ostatních aplikací využívajících jiné frameworky a ve verzi zkrácené se zaměřením na porovnání celkových výsledků jednotlivých frameworků v rámci modelové aplikace jako celku. V neposlední řadě je v repositáři obsažen návod k použití a odkazy na demonstrační SPA aplikace.

6.1 Instalace prerekvizit

Prvním a nezbytným krokem při tvorbě moderních SPA s využitím frameworku Vue.js je instalace potřebných nástrojů a aplikací.

6.1.1 Node.js

Na základě informací poskytnutých v sekci „*Learn*“ oficiálních stránek Node.js (Node.js, nedatováno) lze technologii popsat následovně. Node.js je tzv. „*runtime environment*“, neboli běhové prostředí umožňující běh aplikací napsaných ve skriptovacím jazyce JavaScript, a to za pomoci „*V8 engine*“ od společnosti Google. Jedná se o open source řešení spravované komunitou, dostupné zdarma uživatelům většiny operačních systémů, dále jen OS (Operating System). Instalační možnosti jsou striktně vázány na používaný OS.

- První možností je instalace instalačního balíčku přímo z oficiálních stránek Node.js (https://nodejs.org/en/download), je dostupná pro všechny mainstreamové platformy (Windows, Linux, Mac).
- Druhou možností je instalace skrz tzv. „*package manager tool*“, instalace typicky probíhá skrze CLI (Command-Line Interface) přes vybraného správce balíčků. Tento typ instalace je dostupný pouze pro Linux a Mac.
- Třetí, doporučený styl instalace a v tomto případě také správy Node.js je tzv. „*nvm*“ (Linux, Mac) či „*nvm-windows*“ (Windows). Tento umožňuje

jednoduchou instalaci jakékoli verze Node.js a přepínání mezi již nainstalovanými verzemi.

Instalaci lze ověřit v CLI (cmd, terminal, …) a to použitím příkazu „*node -v*“.

```
C:\Users\BlackSpace>node -v
v20.9.0
```

Příklad 1: Ověření instalace Node.js

6.1.2 Node Package Manager (NPM)

Jedná se o defaultního a nejznámějšího správce balíčků pro Node.js, po důkladném zvážení lze použít i jiné správce balíčků jako „*yarn*“, či novější „*pnpm*“ (nutno postupovat s opatrností). V průběhu této práce byl využit „*npm*“, který je automaticky nainstalován po boku Node.js a jehož přítomnost lze ověřit použitím příkazu „*npm -v*“.

```
C:\Users\BlackSpace>npm -v
10.2.1
```

Příklad 2: Ověření instalace NPM

6.1.3 MySQL

Dokumentace MySQL (Oracle, nedatováno) uvádí MySQL jako jeden z nejznámějších a nejoblíbenějších SQL (Structured Query Language) databázových systémů. Připisuje vývoj a údržbu systému společnosti Oracle, která tento poskytuje jako open source vydávaný pod licencí „*GNU GPL*“. V neposlední řadě definuje systém jako relační, rychlý a dobře škálovatelný s podporou klient/server architektury. V případě potřeby lze zaměnit za jiný databázový systém. MySQL lze instalovat samostatně (např. na produkční server), nebo jako součást tzv. „*LAMP*“ řešení jako je „*XAMPP*“ (https://www.apachefriends.org/download.html), který mimo databázový systém MySQL poskytuje také grafický systém pro správu databází zvaný „*PhpMyAdmin*“ a další. XAMPP je dostupný na všech mainstreamových platformách (Windows, Linux, Mac).

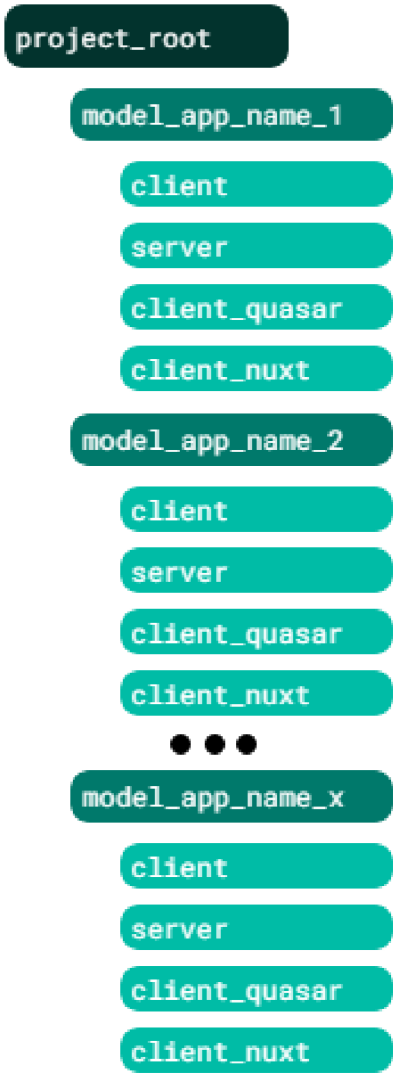
6.2 Inicializace modelových projektů

Předchozí sekce se zabývala instalací technologií potřebných pro vytvoření a běh modelových situací, v této části práce bude nastíněno schéma tvorby těchto aplikací. Jak již bylo zmíněno, bude se jednat o Vue.js aplikace. V této práci bylo využito Vue.js

s technologií Vite, původně speciálně vytvořenou pro tvorbu Vue.js aplikací samotným tvůrcem Vue.js, jedná se o doporučený způsob vytváření projektů, propagovaný v oficiální dokumentaci Vue.js (Quick Start | Vue.js, nedatováno). V této práci byl upřednostněn před Vue CLI, postaveném na technologii Webpack.

6.2.1 Struktura modelových projektů

Každý z modelových projektů byl implementován se třemi variantami na pohled identické klientské aplikace. Jedná se o SPA vytvořenou za pomocí Vue.js a dvě UA tvořené za pomocí frameworků postavených na Vue.js, konkrétně pak frameworků Quasar.js a Nuxt 3. Všechny ze zmíněných aplikací získávají data z univerzálního serveru poskytujícího API pro získání potřebných dat, s výjimkou první ze zmíněných aplikací postavené na Vue.js, která ze serveru současně získává statický obsah aplikace ve formě uživatelského prostředí. V rámci práce bylo dosaženo následující adresářové stromové strukturu.



„*project root*“ je adresář obsahující zdrojové kódy všech modelových situací např. „*bachelor_thesis*“.

„*model_app_name_x*“ je adresář konkrétní modelové situace. Aplikace budou následovat jednotnou strukturu jenž byla načrtnuta na Obr. 5. Kde každá z klientských aplikací disponuje verzí Vue.js (client), Quasar.js (client_quasar) a Nuxt 3 (client_nuxt) obsluhovanou univerzálním serverem (server) pro všechny verze.

Obr. 5: Základní struktura modelových projektů, zdroj: autor.

6.2.2 Inicializace klientské aplikace varianta Vue.js

V této části bude pro každou z modelových situací předveden postup vytvoření Vue.js verze aplikace (client). Postup je následující a opakoval se pro každou jednu z aplikací:

- ✓ V CLI systému byla provedena navigace do kořenového adresáře jedné z modelových aplikací (model_app_name_x).

```
cd C:\cesta\bachelor_thesis\model_app_name_x
```

Příklad 3: Navigace do modelového adresáře

- ✓ Byl spuštěn příkaz k vytvoření Vue.js Vite projektu na základě oficiální dokumentace Vue.js.

```
npm create vue@latest
```

Příklad 4: Vytvoření Vue.js projektu s Vite

- ✓ Byla vyplněna pole průvodcem tvorbou aplikace.

```
√ Project name: ... client
√ Add TypeScript? ... No / Yes
√ Add JSX Support? ... No / Yes
√ Add Vue Router for Single Page Application development? ... No / Yes
√ Add Pinia for state management? ... No / Yes
√ Add Vitest for Unit Testing? ... No / Yes
√ Add an End-to-End Testing Solution? » No
√ Add ESLint for code quality? ... No / Yes
√ Add Prettier for code formatting? ... No / Yes
```

Příklad 5: Parametry tvorby Vue.js Vite projektu

- ✓ V CLI systému byla provedena navigace do kořenového adresáře klientské verze Vue.js (client).

```
cd client
```

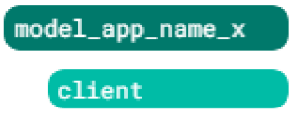
Příklad 6: Navigace do Vue.js verze modelové aplikace

✓ V CLI byl spuštěn příkaz k instalaci závislostí projektu

```
npm install
```

Příklad 7: Instalace závislostí za pomoci npm

Po provedení popsaného postupu byla vytvořena kostra klientské aplikace verze Vue.js (client) pro každou z modelových situací.



Obr. 6: Dosažení struktury client, zdroj: autor.

6.2.3 Inicializace Node.js serveru

V předchozí sekci byly položeny základy pro tvorbu klientské aplikace modelových situací. Tato sekce bude věnována vytváření kostry serverové aplikace (server), která jak již bylo vytyčeno, funguje jako API pro všechny ze zmíněných aplikací a jako poskytovatel uživatelského prostředí pro klientskou aplikace verze Vue.js. Postup se stejně jako v předchozí sekci opakoval pro všechny modelové situace:

✓ V CLI systému byla provedena navigace do kořenového adresáře jedné z modelových aplikací (model_app_name_x). Viz Příklad 3: Navigace do modelového adresáře.

✓ Byl vytvořen kořenový adresář pro serverovou aplikaci (server).

```
mkdir server
```

Příklad 8: Vytvoření kořenového adresáře server

✓ V CLI systému byla provedena navigace do kořenového adresáře serverové aplikace (server).

```
cd server
```

Příklad 9: Navigace do adresáře server modelové aplikace

✓ V CLI systému byl spuštěn příkaz pro vytvoření Node.js projektu.

```
npm init
```

Příklad 10: Vytvoření Node.js projektu

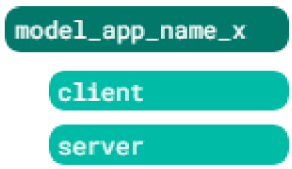
✓ Byla vyplněna pole průvodcem tvorbou aplikace.

```
package name: (server)
version: (1.0.0)
description: Simple Node.js server for model situation
entry point: (index.js)
test command:
git repository:
keywords:
author: Jiří Žák
license: (ISC) MIT
```

Příklad 11: Parametry tvorby Node.js aplikace

✓ V CLI byl spuštěn příkaz k instalaci závislostí projektu. Viz Příklad 7: Instalace závislostí za pomoci npm

Po provedení popsaných kroků byla vytvořena kostra serverové aplikace (server) pro každou z modelových situací.



Obr. 7: Dosažení struktury server, zdroj: autor.

6.2.4 Návrh klientské aplikace

Výzkum byl proveden na pěti, v praxi běžných, typech aplikací. Konkrétně pak budou implementovány stránky e-shopu, informační / prezentační stránky, interní aplikace, diskusní fórum a v neposlední řadě jednoduchá sociální síť. Předmětem této části práce je porovnání vývoje a výsledků u daných frameworků, nikoliv předložit ukázkové řešení vývoje zmíněných aplikací. Dokumentace vývoje těchto aplikací tedy proběhne pouze v abstraktní rovině. Nutno brát v zřetel, že se jedná o primitivní imitaci aplikací, které lze nalézt v reálném světě, a že jejich přesná podoba se může lišit kus od kusu. Dále je nutné podotknout, že budou v práci z důvodu evaluace upřednostněny

experimentální techniky vykreslování i přes jejich jinak možnou nevhodnost. Především pak bude kladen důraz na co nejvyšší míru znovu použitelnosti vykreslených stránek, pokud je to tedy možné, bude před SSR upřednostněna jakákoliv jiná vhodná vykreslovací technika umožňující její uložení pro další identické požadavky.

Pro získávání dat budou z organizačních důvodů využita následující diskrétní pravidla:

- Všechna data, která jsou trvalá nebo předmětem SEO jsou získávána a vykreslována na straně serveru, či před samotným nasazením na server.
- Všechna ostatní data, především data specifická ku uživateli jsou předmětem získání a vykreslení na straně klienta. Jednoduše všechna data, která nemusí či nesmí být indexována vyhledávači.

Návrh aplikace E-shop

První zkoumanou aplikací je e-shop se zaměřením na prodej levitujících květináčů s názvem „*Plant Levitate*“. Obsahuje stránky nutné k představení organizace, zprostředkování prodeje a primitivní redakční systém pro správu produktů. Disponuje následující navigační strukturou.

- **/**
    - Funkce: skromné seznámení s prodejcem a upoutání pozornosti.
    - SSR (SSG): konstantní obsah.
    - CSR: výčet oblíbených produktů.
- **/stock**
    - Funkce: seznam všech prodejcem nabízených produktů s možností hledat, řadit a stránkovat.
    - SSR (SSG): konstantní obsah stránky.
    - CSR: jednotlivé nabízené produkty.
- **/product/:x**
    - Funkce: prezentovat detaily konkrétního produktu, umožnit jeho hodnocení a přidání do košíku.
    - SSR (ISR): detailní informace o produktu.
    - CSR: hodnocení a recenze produktu, doplněné o pár podobných produktů.

- **/about**
    - Funkce: prezentovat základní informace o lokaci a kontaktu na organizaci. (vynecháno z důvodu podobnosti s hlavní stranou)
    - SSR (SSG): konstantní obsah.
- **/admin**
    - Funkce: jednoduchý redakční systém pro správu produktů e-shopu.
    - CSR: seznam nabízených produktů.
- **/admin/login**
    - Funkce: umožnění autentizace jako správce.

Návrh aplikace informační stránka

Druhou zkoumanou aplikací je informační stránka sloužící čistě k webové prezentaci abstraktní politické strany „*The Unity Vanguard*“. Obsahuje pouze stránku nutnou k představení organizace. Disponuje následující navigační strukturou.

- **/**
    - Funkce: prezentovat všechny potřebné informace o organizaci. Misi, tým, atd…
    - SSR (SSG): konstantní obsah.

Návrh aplikace interní systém

Třetí zkoumanou aplikací je interní systém pro generickou evidenci docházky nazývaný „*ATS*“. Obsahuje nástroje nutné ke správě docházky uzavřené za autentizační branou. Disponuje následující navigační strukturou.

- **/auth/login**
    - Funkce: umožnit přihlášení uživatele do systému.
    - SSR (SSG): konstantní obsah.
- **Ostatní**
    - Funkce: různorodá práce uvnitř systému za autentizační bránou.
    - CSR: různorodý obsah.

Návrh aplikace diskusní fórum

Čtvrtou zkoumanou aplikací je obecně zaměřené diskusní fórum s názvem „*Debate Mingle*“. Obsahuje stránky nutné k představení organizace a vykonávání potřeb fóra. Disponuje následující navigační strukturou.

- **/home-guest**
    - Funkce: představení debatního fóra a upoutání pozornosti.
    - SSR (SSG): konstantní obsah.
    - CSR: oblíbené diskuse.
- **/search-guest**
    - Funkce: zobrazení výsledků hledání na základě hledaného výrazu.
    - SSR (SSG): konstantní obsah.
    - CSR: diskuse odpovídající hledanému výrazu.
- **/ask-guest**
    - Funkce: umožnění tvorby nových debat.
    - SSR (SSG): konstantní obsah.
- **/discuss-guest/:x**
    - Funkce: zobrazení informací o debatě a jejího celého obsahu.
    - SSR (ISR): informace o diskusi a příspěvky v diskusi.

Návrh aplikace sociální síť

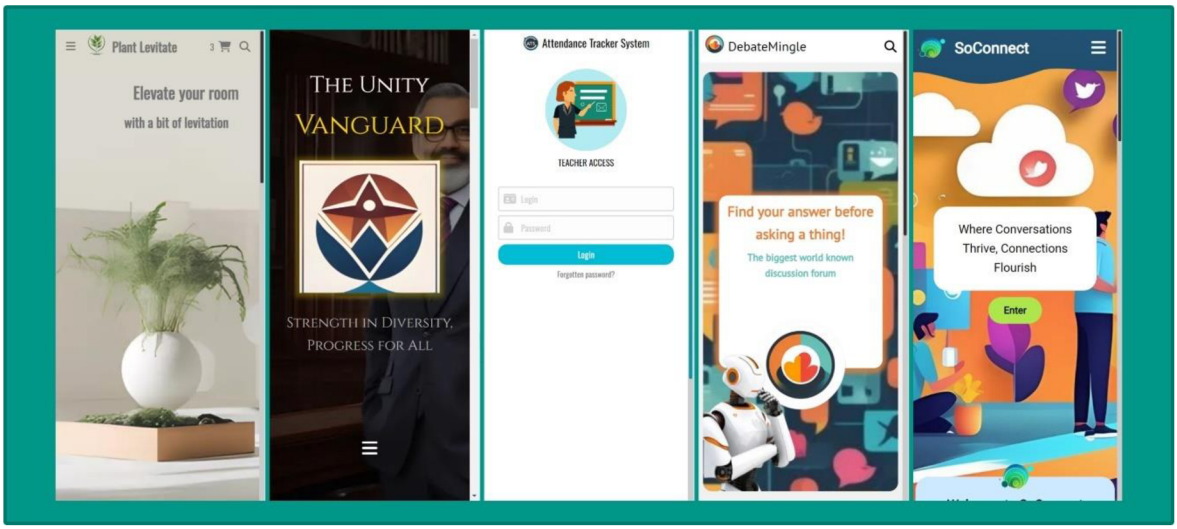
Pátou a poslední zkoumanou aplikací je sociální síť „*SoConnect*“. Obsahuje stránky nutné k představení organizace a potřebami sociálního média. Disponuje následující navigační strukturou.

- **/**
    - Funkce: představení média a navnadění zákazníka.
    - SSR (SSG): konstantní obsah.
- **/app/feed**
    - Funkce:
        - Přihlášený uživatel: zobrazení nových pro uživatele relevantních příspěvků.
        - Nepřihlášený uživatel: de facto pouze možnost přihlášení.
    - SSR (SSG): konstantní obsah.
    - CSR:

        - Přihlášený uživatel: nejnovější příspěvky relevantní pro uživatele.
- **/app/c-post**
    - Funkce:
        - Přihlášený uživatel: umožnění tvorby nových příspěvků.
        - Nepřihlášený uživatel: de facto pouze možnost přihlášení.
    - SSR (SSG): konstantní obsah.
- **/app/user/:x**
    - Funkce:
        - Přihlášený uživatel, sám: zobrazení a úprava vlastního profilu, seznam příspěvků a přátel. Možnost se odhlásit.
        - Přihlášený uživatel, přítel: zobrazení profilu uživatele, jeho příspěvků a přátel. Možnost odebrání z přátel.
        - Přihlášený uživatel, neznámý: zobrazení základních údajů uživatele a možnost přidání do přátel.
        - Nepřihlášený uživatel: zobrazení základních údajů uživatele a de facto možnost se přihlásit.
    - SSR (ISR): množina či podmnožina informací o uživateli.
    - CSR:
        - Přihlášený uživatel, sám: seznam příspěvků a přátel.
        - Přihlášený uživatel, přítel: seznam příspěvků a přátel.
- **/app/login**
    - Funkce: umožnění přihlášení do systému.
    - SSR (SSG): konstantní obsah.
- **/app/signup**
    - Funkce: umožnění registrace do systému.
    - SSR (SSG): konstantní obsah.

6.2.5 Výsledná podoba aplikací

Výsledkem je pět aplikací operujících nad architekturou SPA s různou podobou, funkcionalitou a účelem. Tyto by měly představovat pro UA frameworky set problémů, jimž je nutno čelit a jejichž řešení bude nejen provedeno, ale současně také testováno.



Obr. 8 Náhled výsledných SPA, zdroj: autor.

7 Implementace On-Demand ISR

Dříve zmíněná a popsaná technika vykreslení On-Demand ISR, která byla současně několikrát zmíněna v návrhu aplikací, představuje z teoretického hlediska v podstatě dokonalé přemostění funkcionality mezi SSG a SSR, její využití v praxi, avšak jaksi pokulhává. Hlavním, nikoliv ale jediným problémem, jenž je nutno vyřešit je určení stránek, na kterých byl obsah využit. Jelikož jak uvádí T. Marshall v blogu pro headless CMS (Content Management System) Kontent.ai (Marshall, 2022), obsah nemusí a pravděpodobně není využit pouze na jedné známé stránce. Situace se ještě přiostří za předpokladu, že provázání obsahu není implicitně známé a je třeba ho „*JIT*“ zjistit, jak uvádí v blogu (Lange, 2022) D. Lange. Nejhorším možným scénářem je náhodné provázání obsahu, které již nelze zpětně získat, a to např. použitím funkce RAND v MySQL.

7.1 Možnosti implementace On-Demand ISR

V této práci budou vytyčeny tři možné způsoby implementace na základě automatizovanosti daného řešení.

- **Plně manuální**: seznam stránek pro obnovení je manuálně vytvořen předem pro každou datovou položku. Tento způsob byl nastíněn jak v blogu T. Marshalla (Marshall, 2022), tak v blogu D. Langea (Lange, 2022).
- **Poloautomatický**: seznam stránek je vytvořen na základě předem manuálně definovaných pravidel pro každou datovou položku. Tento způsob definuje de facto funkční a efektivní řešení a byl nastíněn v blogu D. Langea (Lange, 2022).
- **Plně automatický**: seznam stránek je sestavován inkrementálně, či naopak dekrementálně na základě sledování provozu aplikace. Tento řeší problém manuální implementace pravidel, ale již nepředstavuje příliš efektivní řešení. Tento způsob bude v práci implementován jako tzv. ODAI.

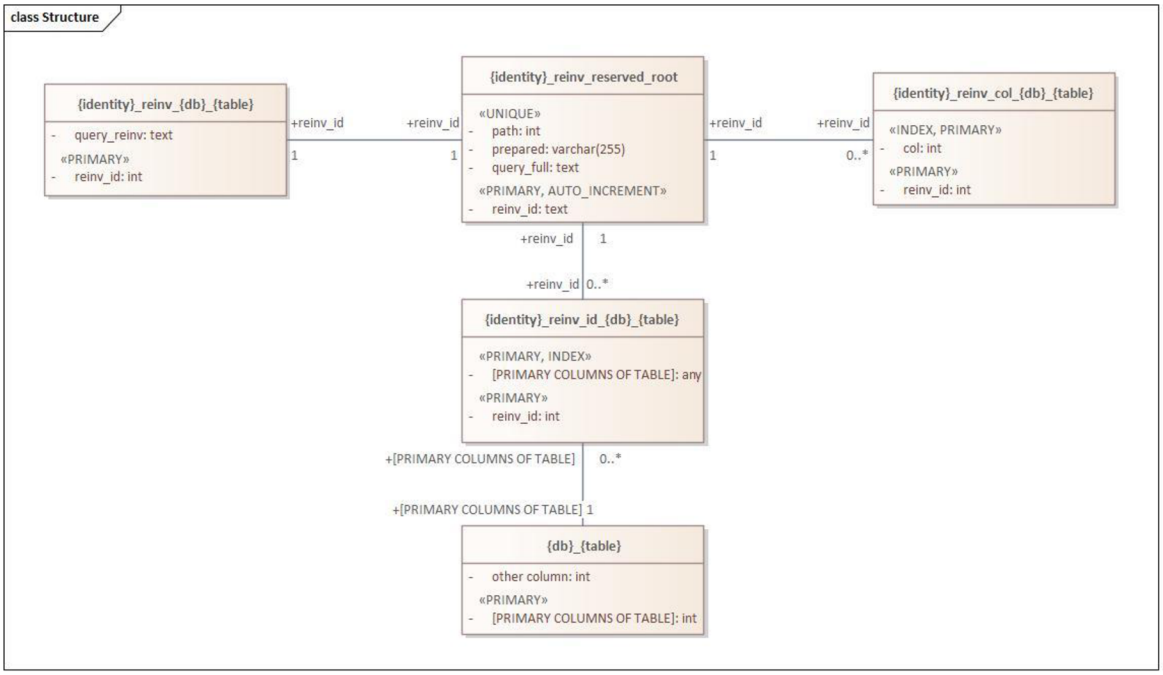
7.2 ODAI (On-Demand Action Interceptor)

Název byl odvozen na základě účelu návrhového vzoru a způsobu jakým pracuje. Ve zkratce se jedná o návrhový vzor sloužící k zapouzdření CRUD operací nad vybraným datovým zdrojem (v této práci databází MySQL) sloužící k správě datových provázaností a reakcí na jejich změny.

7.3 Teoretický návrh návrhového vzoru

Návrhový vzor je postaven na principu sledování provozu aplikace. Každý požadavek zpracovávaný technikou On-Demand ISR je speciálně označen, a následně předán ODAI k dalšímu zpracování. ODAI si za pomocí těchto požadavků aktualizuje databázi, či jinou datovou strukturu obsahující informace o využití dat na jednotlivých On-Demand ISR stránkách.

7.3.1 Ukázková struktura ODAI databáze



Obr. 9 Struktura databáze ODAI, zdroj: autor.

Předešlý diagram tříd znázorňuje šablony všech datových struktur, které ODAI interně využívá k svému provozu a jejich základní vlastnosti. Názvy jednotlivých tříd obsahují následující proměnné části uzavřené znaky „*{}*“:

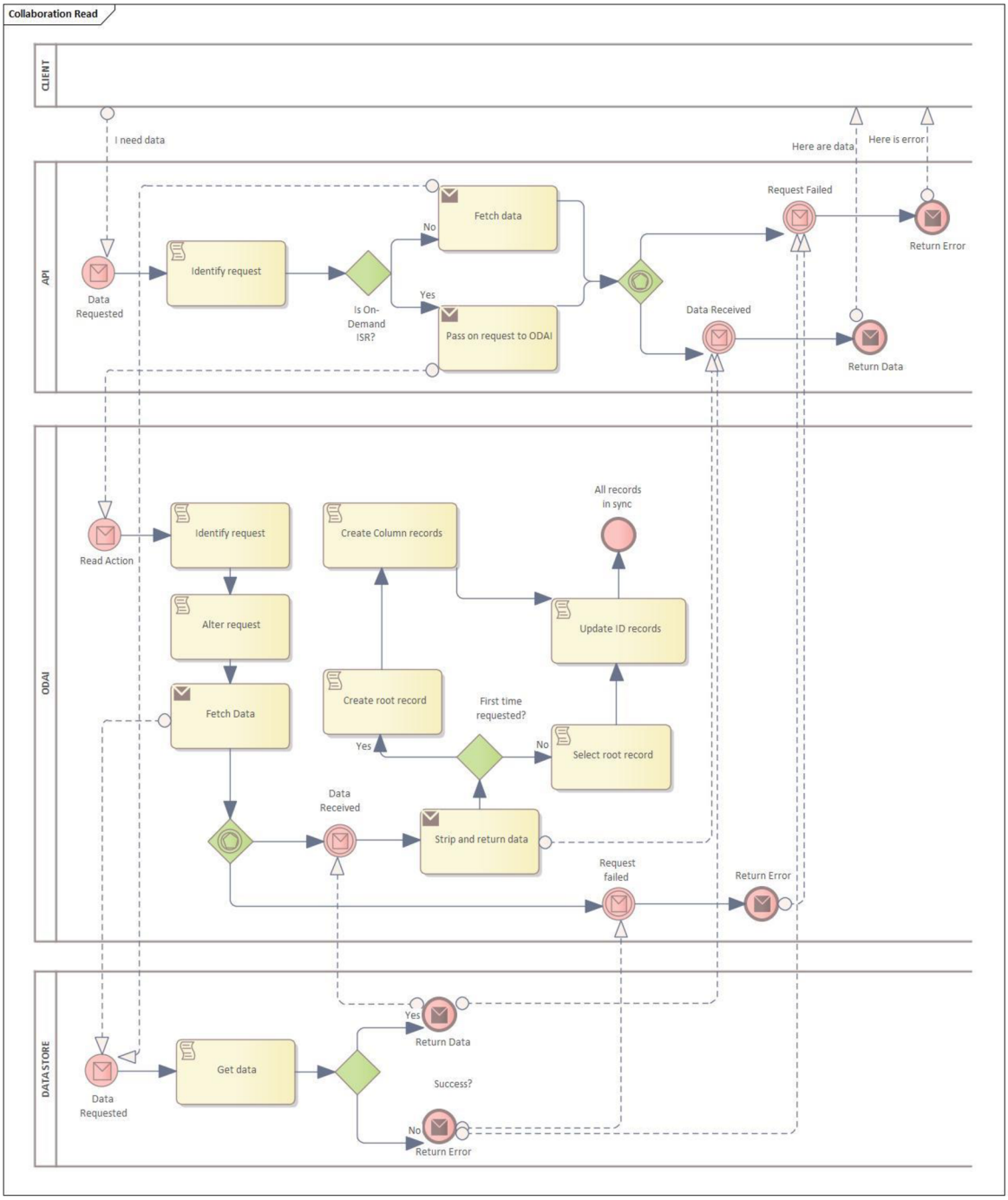
- {identity}: jedinečný identifikátor aplikace v rámci celého ekosystému.
- {db}: jedinečný název databáze v rámci dané aplikace, odvíjí se od názvu databáze, o které eviduje data.
- {table}: jedinečný identifikátor tabulky v rámci dané databáze, odvíjí se od názvu tabulky, o které eviduje data.

Jednotlivé datové struktury jsou pak:

- {identity}_reinv_reserved_root
  - Výskyt: jednou pro každou z aplikací.
  - Funkce: slouží k jedinečné identifikaci požadavku.
  - Atributy:
    - reinv_id… jedinečný identifikátor požadavku.
    - path… URL cesta, pro kterou byl požadavek vykonán.
    - prepared… pole argumentů pro požadavek.
    - query_full… originální podoba požadavku.
- {identity}_reinv_{db}_{table}
  - Výskyt: jednou pro každou z tabulek.
  - Funkce: obsahuje pro každou z dotčených tabulek zjednodušený požadavek pro ověření dotčenosti tabulky.
  - Atributy:
    - reinv_id… jedinečný identifikátor požadavku.
    - query_reinv… požadavek k ověření dotčenosti požadavkem.
- {identity}_reinv_id_{db}_{table}
  - Výskyt: jednou pro každou z tabulek.
  - Funkce: slouží k dokumentaci všech záznamů dotčených požadavkem.
  - Atributy:
    - reinv_id… jedinečný identifikátor požadavku.
    - [všechny primární klíče k identifikaci záznamu]
- {identity}_reinv_col_{db}_{table}
  - Výskyt: jednou pro každou z tabulek.
  - Funkce: slouží k dokumentaci všech sloupců dotčených požadavkem.
  - Atributy:
    - reinv_id… jedinečný identifikátor požadavku.
    - col… název daného sloupce.

7.3.2 Akce číst

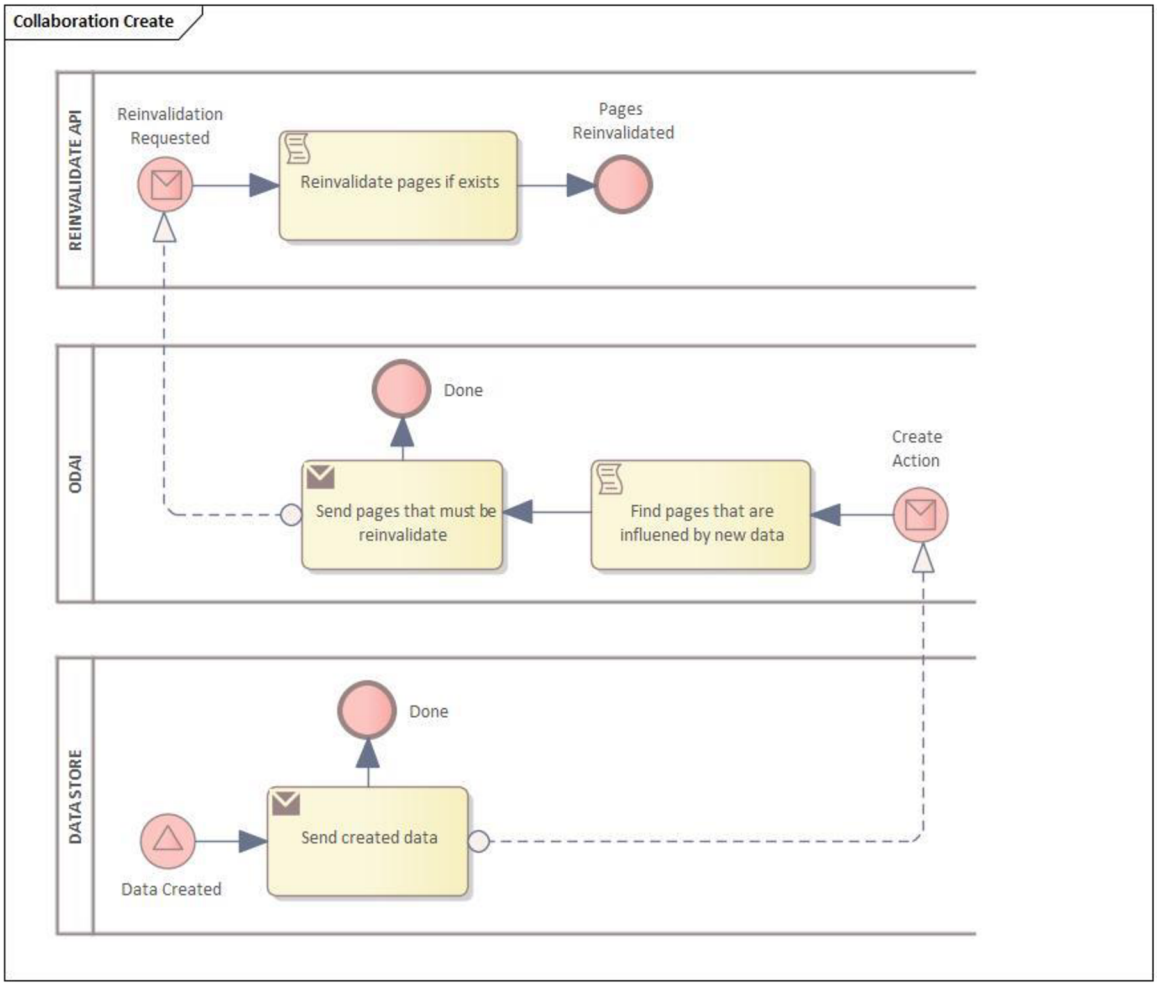
Průběh akce čtení je znázorněn v následujícím BPMN (Business Process Model and Notation) diagramu popisujícím její vznik a průběh. Akce vede na sestavení reinvalidačních záznamů.



Obr. 10 ODAI akce READ, zdroj: autor.

7.3.3 Akce vytvořit

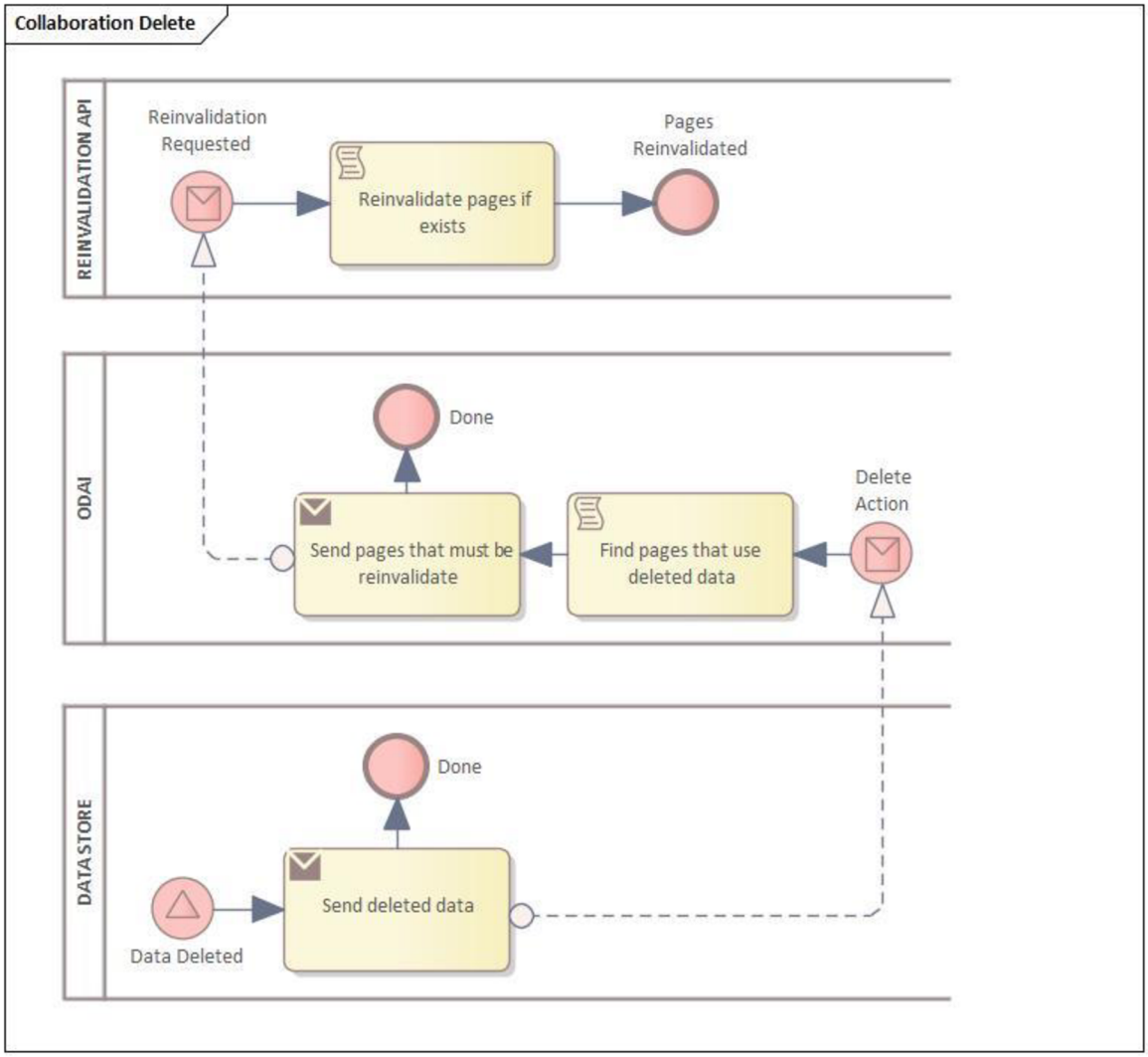
Průběh akce tvorby je znázorněn v následujícím BPMN diagramu popisujícím její vznik a průběh. Akce vede na samotnou reinvalidaci stránek.



Obr. 11 ODAI akce CREATE, zdroj: autor.

7.3.4 Akce smazat

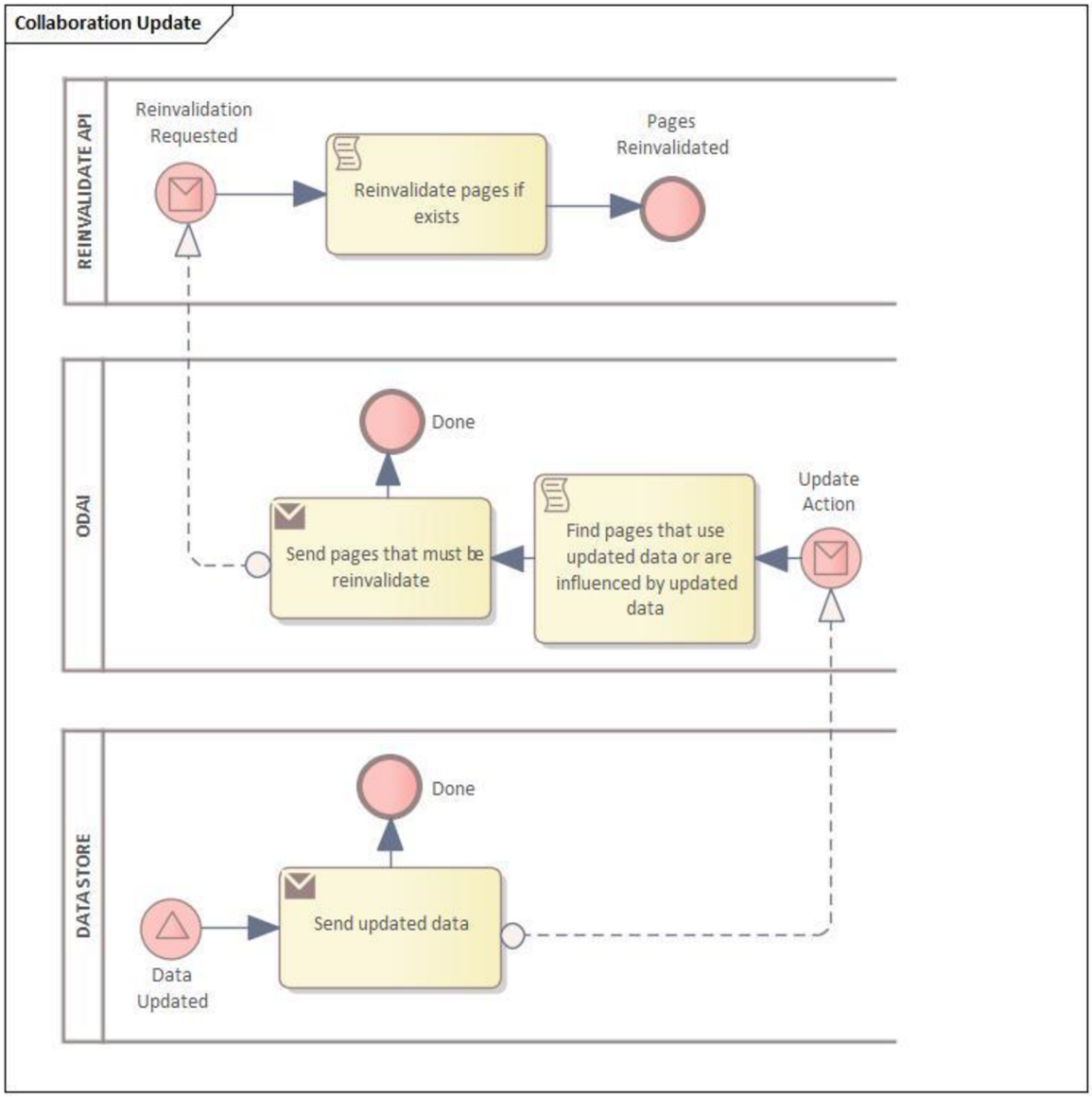
Průběh akce mazání je znázorněn v následujícím BPMN diagramu popisujícím její vznik a průběh. Akce vede na samotnou reinvalidaci stránek.



Obr. 12 ODAI akce DELETE, zdroj: autor.

7.3.5 Akce upravit

Průběh akce úprav je znázorněna v následujícím BPMN diagramu popisujícím její vznik a průběh. Akce vede na samotnou reinvalidaci stránek.



Obr. 13 ODAI akce UPDATE, zdroj: autor.

7.4 Praktická implementace návrhu v práci

Z praktického hlediska bylo nutno vyřešit extrakci informací z požadavků a umožnit sledování změn v datovém úložišti. V práci byl použit databázový systém MySQL, který implicitně nenabízí, ani možnost jednoduché extrakce těchto informací, ani sledování změn provedených na datech.

7.4.1 Extrakce informací z požadavků

Informace byly z jednotlivých MySQL dotazů získávány pomocí npm modulu „*node-sql-parser*“, který umožňuje celý MySQL dotaz konvertovat na JS objekt obsahující strukturované informace o dotazu, případně provést pomocí JS úpravy a následně konvertovat zpět na MySQL dotaz.

7.4.2 Reakce na datové změny

Sledování změn provedených na datech bylo docíleno pomocí npm modulu „*zongji*“ pracujícího na bázi replikace. Modul vystupuje v roli repliky, a je díky tomu schopen upozornit na změny provedené jak na datech, tak i na samotné struktuře databáze.

7.5 Shrnutí On-Demand ISR

Jak uvádí T. Marshall (Marshall, 2022), techniku lze využít u relativně triviálních stránek, a to jak velikostí, tak komplexností. Prozatím není známa efektivní implementace automatizující proces reinvalidace a manuální implementace není dobře škálovatelná. Technika je teoreticky efektivně využitelná, pokud je nasazena s rozumem. Špatné použití může ale mít za následek nepředpověditelné chování aplikace.

8 Migrace aplikací na UA

Proces migrace aplikace je na základě stránek IBM (What is application migration?, nedatováno) proces, při němž dochází k přesunu aplikace z jednoho výpočetního prostředí do jiného. V případě migrace na framework jsou k dispozici dvě cesty, které jsou zmíněny v blogu Oprea (Two strategies for migrating an existing application to a new framework, 2018). Jsou to možnosti „*inkrementální migrace*“, popisující postupné přeměňování částí aplikace na nový framework a současně v něm tvorbu nových. Nebo způsob druhý, popisující znovu vytvoření takové aplikace najednou s použitím pouze nového frameworku. V tomto případě dává smysl pouze druhý ze zmíněných způsobů, který bude dále prezentován pro každý jeden ze zkoumaných frameworků.

8.1 Migrace na Quasar.js

Migrace bylo v této práci docíleno vytvořením nové Quasar.js aplikace a nahrazením jejího obsahu za obsah definovaný klientskou aplikací Vue.js. K inicializaci byl proveden postup navržený v oficiální dokumentaci frameworku Quasar.js v sekci Quasar CLI (Quasar CLI, nedatováno) sestávající z následujících kroků:

- ✓ Byla provedena globální instalace Quasar CLI.

```
npm i -g @quasar/cli
```

Příklad 12 Globální instalace Quasar CLI

- ✓ Byl spuštěn příkaz k vytvoření Quasar.js projektu na základě oficiální dokumentace Quasar.js.

```
npm init quasar
```

Příklad 13 Příkaz pro inicializaci Quasar.js projektu

✓ Byla vyplněna pole průvodce tvorbou Quasar.js projektu.

What would you like to build? » **App with Quasar CLI, let's go!**
√ Project folder: ... **client_quasar**
√ Pick Quasar version: » **Quasar v2 (Vue 3 | latest and greatest)**
√ Pick script type: » **Javascript**
√ Pick Quasar App CLI variant: » **Quasar App CLI with Vite**
√ Package name: ... **client_quasar**
√ Project product name: (must start with letter if building mobile apps) ... **Client Quasar**
√ Project description: ... **Quasar version of app**
√ Author: ... **Jiří Žák**
√ Pick your CSS preprocessor: » **None (the others will still be available)**
√ Check the features needed for your project: » **ESLint, State Management (Pinia), Axios**
√ Pick an ESLint preset: » **Prettier**

Příklad 14 Ukázkové vyplnění průvodce tvorbou projektu

Po vytvoření Quasar.js projektu byl proveden import příslušných zdrojových kódů z klientské aplikace Vue.js. Pro zkoumané aplikace neobsahují hierarchie projektů velké rozdíly, avšak hierarchie se může pro jiné projekty diametrálně lišit, a tím i rozdíly při migraci. Manuálně pak bylo třeba přidat podporu písma od Google a knihovnu ikon Font Awesome, implementovat modul „*dotenv*“ a „*dotenv-expand*“ pro získání ENV proměnných ze souborové struktury stejné jako u Vue.js Vite verze. Quasar.js z technických důvodů neumožnuje definici „*main.js*“ (vstupního souboru Vue.js aplikace), bylo tedy třeba dané konfigurace přesunout do tzv. „boot“ struktur. V neposlední řadě bylo třeba provést několik změn v konfiguračním souboru frameworku „*quasar.config.js*“:

```
preFetch: true, // umožní získání dat před vykreslením
boot: ["general", "axios"], // definice vlastních boot souborů
css: ["main.css"], // definice globálního stylování
extras: ["fontawesome-v6"], // podpora fontawesome
build: {
  env: (async () => {
    // přečtení ENV ze souboru
    let myEnv = {};
    const dotenv = require("dotenv");
    const dotenvExpand = require("dotenv-expand");

    myEnv = dotenv.config({
      path: path.join(
        __dirname,
        `.env.${ctx.prod ? "production" : "development"}.local`
      ),
    });
    dotenvExpand.expand(myEnv);
    return myEnv;
  })(),
  alias: {
    // odstranění základních stylů definovaných frameworkem
    "quasar/dist/quasar.css": path.resolve(__dirname,
"./src/css/quasar.css"),
  },
  vueRouterMode: "history", // změna na history mód u Routeru
  extendViteConf(viteConf) {
    // definice původních Vite konfigurací
    viteConf.envPrefix = "PLANT_LEVIT_";
    Object.assign(viteConf.resolve.alias, {
      "@": path.join(__dirname, "./src"),
    });
  },
},
devServer: {
  port: 5173, // (volitelně) změna portu
},
framework: {
  plugins: ["Cookies"], // povolit plugin pro práci s cookies (pouze pokud
využívá cookies)
},
```

Příklad 15: Základní konfigurační soubor frameworku Quasar.js

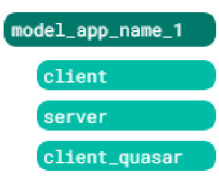
V této chvíli byla aplikace plně funkční v módu SPA a jediné co bylo třeba udělat pro získání funkcionality vykreslování na serveru, bylo přidání SSR módu a přesun získávání dat potřebných při vykreslení na serveru do dříve povoleného bloku „*preFetch*“, umožňujícího získání dat před vykreslením. V případě, že aplikace využívá

cookies, pak také integraci modulu umožňujícího práci s Cookies i v serverovém kontextu, který lze získat integrací Quasar pluginu „*Cookies*“.

Po dokončení dříve zmíněných akcí aplikace operovala plně v módu SSR s přechodem na CSR (tedy technicky UR), zkoumané aplikace ale vyžadovaly i podporu jiných vykreslovacích technik. Framework Quasar.js tzv. „*hybridní vykreslování*“ implicitně nepodporuje, proto bylo vytvořeno a zkoumáno vlastní řešení.

Předmětem zkoumání byla možnost a obtížnost implementace jednoduchého hybridního vykreslování, nikoli jeho samotný návrh. V rámci implementace bylo třeba vyřešit otázky, jako definici technik u jednotlivých stránek a vytvořit funkcionalitu jednotlivých technik. Mechanismus využívá princip umožňující ukládání statického obsahu a operuje nad vestavěným SSR módem daného frameworku. V rámci obtížnosti lze implementaci ohodnotit jako poměrně obtížnou, a to i přes to, že implementace technik jako jsou ISR a SSR vyžadovala minimální úsilí díky modularitě tzv. „middleware“ (vol. přel. z angličtiny: prostředník) souborů, umožňujících definovat vlastní funkcionalitu serveru v podobě „*express handlers*“ (vol. přel. z angličtiny: obslužných rutin express). Značně obtížnější byla implementace technik SSG, a především pak CSR, kde SSG vyžaduje vykonání práce před samotným během serveru, možnost jejího uložení a znovuvyužití při dalším běhu serveru. Technika CSR naopak vyžaduje vygenerování souboru „*index.html*“, obsahujícího základní kostru aplikace a odkazujícího na zdrojový kód frameworku, jenž se poté postará o zbytek. V rámci funkčnosti se jedná o řešení funkční, nikoli ale ideální. Prvním problémem je, že se nejedná o příliš modulární řešení a druhým jsou omezené schopnosti vývojáře, které mohou vést k tvorbě nestabilního či neefektivního řešení. Na druhou stranu se jedná o řešení, nad kterým má vývojář plnou kontrolu, je to tedy věc případu.

Po provedení popsaných kroků byla vytvořena klientská aplikace ve frameworku Quasar.js pro každou z modelových situací.



Obr. 14: Dosažení struktury client_quasar, zdroj: autor.

8.2 Migrace na Nuxt 3

Migrace bylo v této práci docíleno vytvořením nové Nuxt 3 aplikace a nahrazením jejího obsahu za obsah definovaný klientskou aplikací Vue.js. K inicializaci byl proveden postup navržený v oficiální dokumentaci frameworku Nuxt 3 v sekci instalace (Installation, nedatováno), sestávající z následujících kroků:

- ✓ Byl proveden příkaz pro inicializaci Nuxt 3 projektu prostřednictvím npx.

```
npx nuxi@latest init client_nuxt
```

Příklad 16 Příkaz pro inicializaci Nuxt 3 projektu

- ✓ Byl vybrán npm jako správce balíčků Nuxt 3 projektu.

```
Which package manager would you like to use? npm
```

Příklad 17 Výběr npm jako správce balíčků Nuxt 3 projektu

Po vytvoření Nuxt 3 projektu byl proveden import příslušných zdrojových kódů z klientské aplikace obdobným stylem jako v předchozí sekci týkající se frameworku Quasar.js, mimo fakt, že framework implicitně upouští od struktury „*/src*“. Lze se nové struktuře přizpůsobit, či využít silnou modularitu frameworku a definovat strukturu „*/src*“ v konfiguračním souboru projektu, viz příklad níže:

```
export default defineNuxtConfig({
  srcDir: "src/", // definice adresáře pro zdrojový kód aplikace
});
```

Příklad 18 Definice adresáře zdrojového kódu Nuxt 3 aplikace

Framework v základním nastavení poskytuje definici stránek na základě souborové struktury, ve které je cesta stránky definována její hierarchií v adresářích, či jejím názvem. Každá ze stránek disponuje automaticky odvozeným názvem, který bylo třeba v určitých případech v rámci modelových situací nutno předefinovat. Současně bylo v určitých případech třeba manuálně doplnit alias nebo přesměrování, a to v konfiguračním souboru, viz příklad níže:

```
export default defineNuxtConfig({
  hooks: {
    "pages:extend"(pages) {
      // např. předefinování názvu „product-productId“ na „product“
      // např. vytvoření aliasu pro „schedule“
      // např. vytvoření redirectu pro „/“
      const route = pages.map((r) => {
        if (r.name === "product-productId") r.name = "product";

        if (r.name === "schedule") r.alias = "/teacher";
      });
      pages.push({ path: "/", redirect: "/teacher" });
    },
  },
});
```

Příklad 19 Předefinování implicitních definic stránek Nuxt 3 aplikace

V rámci souborové navigace bylo třeba v neposlední řadě vyřešit nahrazení původních komponent „*RouterView,*“ na frameworku specifické komponenty „*NuxtPage*“ a „*NuxtLayout*“ a vykonat s tím spojené změny v souborech „*layouts*“.

Jedním z dalších potřebných kroků byla instalace a integrace „data store“ (vol. přeloženo z angličtiny: datový obchod) Pinia použitého ve většině modelových aplikací. Instalace proběhla ve dvou jednoduchých krocích stávajících z instalace modulů viz následující příkaz:

```
npm install pinia @pinia/nuxt
```

Příklad 20 Instalace modulu Pinia pro Nuxt 3

A následně integrace modulu v rámci konfiguračního souboru:

```
export default defineNuxtConfig({
  modules: ["@pinia/nuxt"], // integrace modulu Pinia
});
```

Příklad 21 Integrace modulu Pinia do Nuxt 3 aplikace

Následovalo zapojení globálního stylování, knihovny Font Awesome a písma Google, to vše např. znovu skrze konfigurační soubor:

```
export default defineNuxtConfig({
  css: ["~/assets/main.css"], // zapojení souboru s globálním stylem
  app: {
    head: {
      link: [
        {
          // vyžádání Google fontu Oswald
          rel: "stylesheet",
          href: "https://fonts.googleapis.com/css?family=Oswald",
        },
      ],
      script: [
        {
          // vyžádání ikon Fontawesome
          src: "https://kit.fontawesome.com/e84002e394.js",
          crossorigin: "anonymous",
          defer: true,
        },
      ],
    },
  },
});
```

Příklad 22 Zapojení globálního stylu, Google písma a ikon Fontawesome v Nuxt 3 aplikaci

Framework Nuxt 3 stejně jako Quasar.js neumožňuje definici souboru „*main.js*“, a bylo tedy nutné dané konfigurace přesunout do tzv. „*plugin*“ struktur. V rámci udržení integrity s klientskou aplikací Vue.js byl implementován modul „*dotenv*“ a „*dotenv-expand*“ pro získání ENV proměnných ze souborové struktury, a rozšířena konfigurace Vite v konfiguračním souboru:

```
export default defineNuxtConfig({
  hooks: {
    "vite:extendConfig"(config, { isClient, isServer }) {
      // příklad nastavení vlastního prefixu pro Vite ENV proměnné
      config.envPrefix = "PLANT_LEVIT_";
    },
  },
});
```

Příklad 23 Nastavení ENV prefixu pro Vite proměnné pro Nuxt 3 aplikaci

Pracovat s cookies v serverovém kontextu lze za pomoci vestavěného tzv. „*composable*“ bez nutnosti jakýchkoliv konfigurací. Dalším krokem byl přesun

získávání dat potřebných pro vykreslení na serveru do „*setup*“ bloku s využitím speciálního „*composable*“ pro získávání dat zvaného „*useAsyncData*“. Posledním nutným krokem před zapojením hybridního vykreslování byl přesun logiky implementované v tzv. „*Navigation Guards*“ Vue Routeru do struktury „*middleware*“ a v určitých případech i „*server middleware*“. Nejdůležitějším a posledním krokem migrace na framework bylo nastavení hybridního vykreslování. Na rozdíl od předešlého frameworku Quasar.js je Nuxt 3 vybaven velice solidní sadou vykreslovacích technik, které lze v aplikaci využít za cenu minimální potřeby konfigurace. V současné době disponuje následujícími vestavěnými vykreslovacími technikami: „CSR, SSR, SSG, ISR, SWR“ načež implicitně operuje v SSR (UR) módu a případné změny lze provést v konfiguračním souboru:

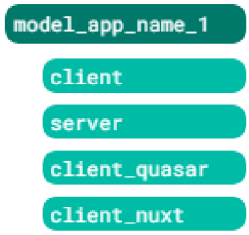
```
export default defineNuxtConfig({
  // příklad definice pravidel pro jednotlivé stránky aplikace
  routeRules: {
    "/": {
      prerender: true, // SSG
    },
    "/about": {
      prerender: true, // SSG
    },
    "/stock": {
      prerender: true, // SSG
    },
    "/product/**": {
      isr: true, // ISR
      cache: {
        base: "db",
        name: "invalidable",
      },
    },
    "/admin/**": {
      ssr: false, // CSR
    },
  },
});
```

Příklad 24 Příklad definice pravidel vykreslování u Nuxt 3 aplikace

Migrace byla dokončena implementací On-Demand ISR na základě příspěvku uživatele „*deniskoronets*“ na diskusním fóru Stack Overflow (deniskoronets, 2023). V rámci hybridního vykreslování nebylo dosaženo všech požadovaných funkcionalit, nepodařilo se implementovat SSG pracující s „*Query*“ neboli „*Search*“ parametry URL a ani možnost definovat komplexní logiku ukládání na straně serveru, jenž by umožnila

ukládat více stavové stránky jako bylo prezentováno v rámci modelové situace sociální sítě.

Po provedení popsaných kroků byla vytvořena klientská aplikace ve frameworku Nuxt 3 pro každou z modelových situací.



Obr. 15: Dosažení struktury client_nuxt, zdroj: autor.

9 Měření webových stránek

Po dokončení migrace všech modelových situací na příslušné UA byly analyzovány rozdíly v rámci SEO a použitelnosti stránek s deaktivovaným JavaScriptem u prohlížeče. Následně bylo u jednotlivých aplikací provedeno měření zaměřené na výkonnostní hodnoty jednotlivých verzí testovaných aplikací. Pro měření byl vybrán nástroj od Google zvaný „*Lighthouse*“, dokumentace na stránkách Chrome for Developers (Overview | Lighthouse, 2016) nástroj popisuje jako automatizovaný, open-source nástroj pro růst kvality webových stránek. Dle komerční platformy Shopify (Shopify Staff, 2023) nástroj napomáhá vývojářům, profesionálům v oblasti SEO a v neposlední řadě majitelům stránek auditovat webové stránky a to v oblastech výkonu, přístupnosti, SEO a dalších. Nástroj lze využít více způsoby, v tomto případě byl z důvodu docílení vysoké míry automatizace zvolen NPM modul „*lighthouse*“. Modul pro svou práci vyžaduje otevřený „*Chromium*“ prohlížeč, ve kterém je poté schopen simulovat měření pro danou stránku. K vytvoření a práci s prohlížečem byl zvolen NPM modul „*puppeteer*“, jenž poskytuje sadu operací nad testovací verzí prohlížeče Chrome. Operace lze využít například k simulaci přihlášení do aplikace.

V rámci měření je každá definovaná stránka aplikace podrobena dvanácti vybraným auditům služby Lighthouse, mimo jiné zahrnujících pět základních metrik, ze kterých nástroj defaultně vychází při hodnocení výkonu stránky. Měření probíhá sekvenčně v několika daných cyklech, ze kterých je následně agregován výsledek měření ve formě kvartilů Q1, Q2 a Q3 pro dosažení vyšší přesnosti výsledků. Popsaný test je proveden pro všechny vybrané stránky modelové aplikace, a to odděleně pro jednotlivé frameworky.

Druhou částí měření je zpracování dříve popsaných výsledků k vzájemnému porovnání výsledků jednotlivých frameworků. Pro dříve získaná data je poté vyhodnocen průměr. Na základě, něhož je následně pro výsledky jednotlivých frameworků vyhodnocena procentuální odchylka pro jednodušší porovnání výsledků. Na základě těchto dat již lze porovnat efektivitu frameworků u každé stránky zvlášť, bohužel tato data nejsou z důvodu velkého množství výsledků dobře prezentovatelná, a proto byla tato znovu průměrována, a to na základě jednotlivých frameworků. Výsledné porovnání poskytuje validní informace pro základní porovnání, ale ztrácí velké množství informací.

Měření bylo provedeno na výkonném počítači průměrně ohodnoceném „*bechmarkIndexem*“ ~3434. Simulace omezení CPU byla tedy na základě doporučení v souboru throttling.md umístěném v GitHub repositáři Lighthouse (GoogleChrome / Lighthouse, 2023) nastavena na 12.2x a to na základě následující tabulky popisující benchmarkIndex pro zařízení jednotlivých kategorií:

-	High-End Desktop	Low-End Desktop	High-End Mobile	Mid-Tier Mobile	Low-End Mobile
Example Device	16" Macbook Pro	Intel NUC i3	Samsung S10	Moto G4	Samsung Galaxy J2
Lighthouse BenchmarkIndex	1500-2000	1000-1500	800-1200	125-800	<125
Octane 2.0	30000-45000	20000-35000	15000-25000	2000-20000	<2000
Speedometer 2.0	90-200	50-90	20-50	10-20	<10
JavaScript Execution of a News Site	2-4s	4-8s	4-8s	8-20s	20-40s

Obr. 16: Tabulka popisující Lh BenchmarkIndex, zdroj: (GoogleChrome / Lighthouse, 2023).

Ostatní nastavení byla ponechána implicitní, tedy simulace na mobilním zařízení se síťovým omezením „*Slow 4G*“. V rámci aplikací bylo minimalizováno množství externích závislostí pro dosažení co nejrelevantnějších výsledků. Písma od Google a ikony Font Awesome, byla tedy pro měření zavedena jako lokální závislost.

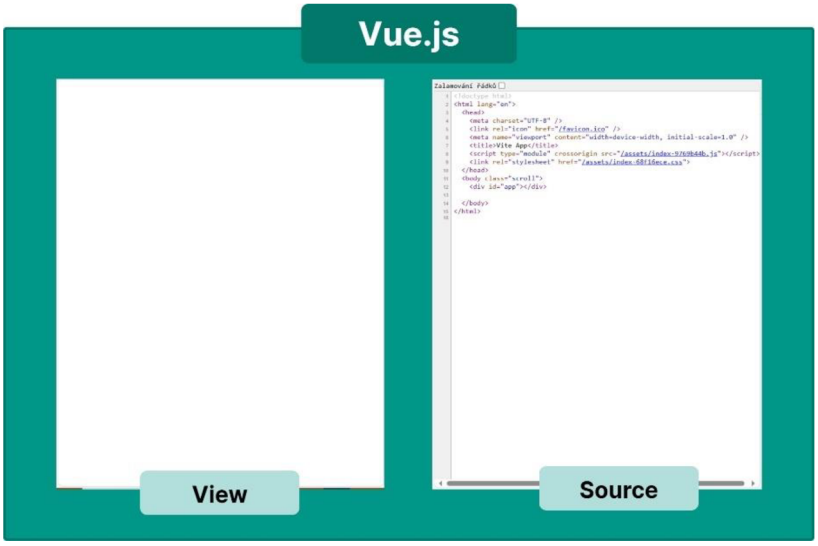
10 Porovnání v měřitelné hladině

Následující sekce práce poskytuje všechny dříve zmíněné výsledky analýzy a měření. Analýza SEO a použitelnosti bez zapnutého JavaSriptu byla provedena pouze pro aplikaci e-shopu, výsledek zkoumaného jevu by byl pro všechny zkoumané aplikace podobný. Výkonnostní data jsou prezentována jako tabulky k přehlednému porovnání výkonnosti jednotlivých frameworků v testovaných modelových situacích. Všechna poskytnutá data jsou jednoznačná a lze z nich vyvodit objektivní závěr, je ale třeba mít na paměti, že se jedná pouze o testovací aplikace, které nemusí nutně prezentovat efektivitu obdobných aplikací v reálném nasazení. Současně se jedná o tzv. „*Lab*“ data sbírána v poměrně uzavřeném prostředí, je tedy nutné k výsledkům přistupovat s obezřetností. V reálném nasazení může hrát roli využití cachování, zatížení serveru, CDN, DNS a podobně, jiné důvody jsou uvedeny např. v blogu DebugBear (Why Does Lighthouse Lab Data Not Match Field Data?, 2022).

Každá sada výsledků byla zpracována ve dvou rovinách. V první bylo pracováno se všemi naměřenými daty a všem datům byla přiřazena stejná důležitost. Druhý prezentovaný výsledek je inspirován oficiální funkcionalitou Google Lighthouse, ve které hraje roli pouze pět základních metrik (ovlivněných ostatními metrikami) s předem daným ohodnocením pro každou z metrik. První výsledek obsahuje náhled většiny známých metrik, druhý obsahuje pouze ty na základě Lighthouse nejrelevantnější.

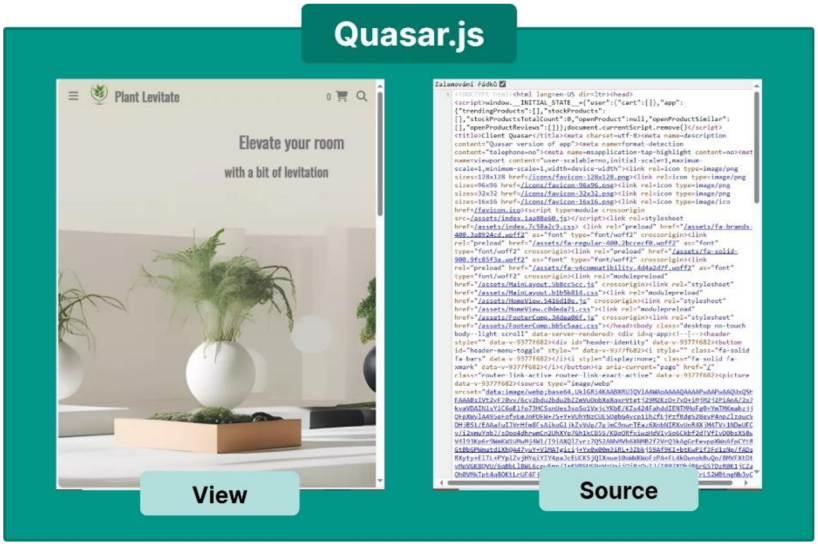
10.1 Analýza SEO a vypnutého JS

Analýza byla provedena v prohlížeči „*Microsoft Edge*“ s globálním zákazem JavaSriptu. V rámci analýzy byly přezkoumány vzhled a interaktivita stránka z pohledu uživatele, který buďto nedisponuje aktivním JavaScriptem, nebo naopak disponuje verzí JS jenž je příliš stará a nepodporuje syntaxi využitou v kritických částech aplikace. Druhou částí zmíněné analýzy je SEO. Toto je zkoumáno na základě obsahu HTML souboru vráceného serverem v rámci počáteční odpovědi.



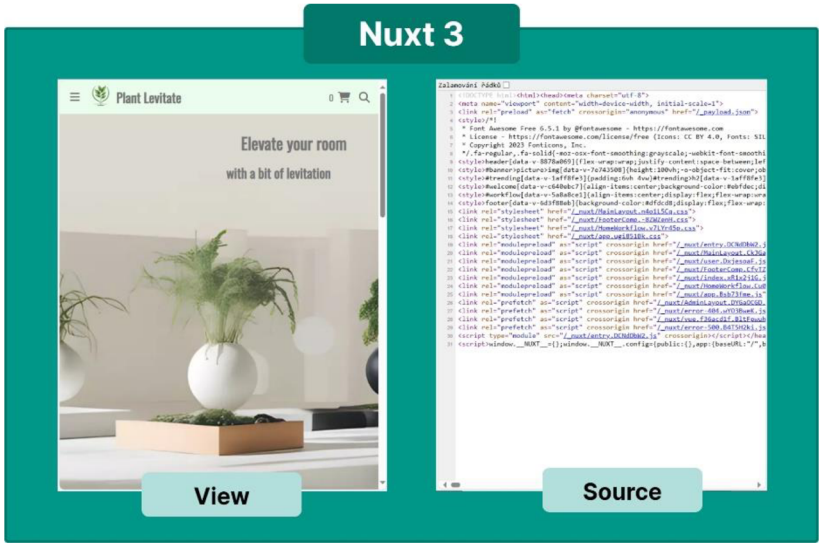
Obr. 17: Analýza SEO a NoJS v rámci Vue.js, zdroj: autor.

Na levé straně ilustrace značené „*View*“, je možné vidět obsah stránky, jenž vidí při navigaci na Vue.js stránku uživatel bez JS. Na stránce není vidět žádný obsah, jelikož ten měl být načten právě za pomoci JS. Tuto nepříjemnou situaci lze do jisté míry kompenzovat za použití „*noscript*“ tagu v HTML, který se zobrazí právě ve chvíli, kdy uživatel nemá povolené JS. Toto ale neplatí v případě, že je JS uživatele příliš staré, kde v takovém případě zůstává stránka stále prázdná. Na pravé straně lze vidět počáteční odpověď serveru značenou jako „*Source*“, obsahující pouze HTML kostru s dvěma odkazy na stažení globálního stylu aplikace a vstupního souboru SPA k převzetí kontroly.



Obr. 18: Analýza SEO a NoJS v rámci Quasar.js, zdroj: autor.

V případě aplikace postavené na frameworku Quasar.js využívající UR lze již vidět na levé straně popisující pohled uživatele bez JS standardně vypadající stránku, jak již bylo řečeno v kapitole 4.7 aplikace reaguje na základní požadavky jako jsou navigace (za předpokladu, že nejsou implementovány přes JS), ale neobsahuje žádnou jinou původní interaktivitu. V pravé části je vidět již v tomto případě velice zdlouhavý zdrojový kód aplikace obsahující jak odkazy na zdroje důležité pro běh aplikace, ale také obsah stránky důležitý pro indexování stránky.



Obr. 19: Analýza SEO a NoJS v rámci Nuxt 3, zdroj: autor.

Více méně stejný výsledek je možné pozorovat pro aplikaci postavenou na frameworku Nuxt 3 využívajícím UR. Pro pohled uživatele bez JS platí funkcionalita jako u aplikace postavené na frameworku Quasar.js. O něco větší rozdíl lze pozorovat na pravé straně, znázorňující počáteční odpověď serveru, na níž si lze i přes dodání prakticky stejné aplikace povšimnout velkého množství rozdílů.

10.2 Pravidla prezentace dat

V rámci prezentace jsou výsledky zaokrouhleny zpravidla na jedno desetinné místo, případně dle potřeby. Pro audity nástroje Lighthouse byly zavedeny výrazy dostupné v sekci příloh, konkrétně jako Tabulka 3 (resp. příloha č. 1).

10.3Data naměřená v rámci výzkumu

Všechna výsledná data měření jsou dostupná v sekci příloh této práce, konkrétně se jedná o Tabulka 4 až Tabulka 13 (resp. příloha č. 2). Konkrétnější datová sada je k dispozici v dříve zmíněném GitLab repositáři.

11 Porovnání v neměřitelné hladině

Tato část práce prezentuje základní porovnání frameworků v teoretické a převážně neměřitelné hladině, z níž nelze vyvodit jednoznačný závěr, jedná se o výčet vlastností, které mohou být vnímány subjektivně a z toho důvodu je nelze v této práci ohodnotit. Kapitola se opírá především o data poskytnutá v již dříve zmíněných oficiálních dokumentacích frameworků Vue.js (Introduction, nedatováno), Quasar.js (Quasar CLI, nedatováno) a Nuxt 3 (The Intuitive Vue Framework, nedatováno). Z důvodu předejití TL; DR budou tyto zdroje citovány substitucí jako dokumentace Vue.js, dokumentace Quasar.js a dokumentace Nuxt 3 bez opakování citační reference.

11.1Podpora uživatele

Práce v jakémkoliv nástroji poskytovaném třetí stranou se neobejde bez nutnosti podpory jeho uživatelů. I přes to, že všechny zkoumané frameworky disponují rozsáhlou dokumentací se najdou případy, kdy dokumentace neposkytuje odpověď na danou otázku, či je třeba zkontaktovat vývojáře z důvodů chyb, či nedostatků daného nástroje. Na základě dokumentací lze říct, že všechny zkoumané frameworky nabízejí spojení skrze populární platformu Discord. Dále diskusní fórum na platformě GitHub a v neposlední řadě i blog obsahující další zdroje nad rámec samotné dokumentace.

11.2 Životnost frameworku

Jedním z nejdůležitějších faktorů, které je třeba zvážit při výběru frameworku je jaké má vyhlídky pro budoucnost a v případě frameworku postaveného výhradně na jiném frameworku toto platí dvojnásob. Pro každý z porovnávaných frameworků je tedy uvedena informace vypovídající o oblíbenosti frameworku, a to na základě počtu „*hvězd*“ u GitHub repositáře, číslo hovořící o reálném využití frameworku a to v tomto případě na základě počtu stažení daného modulu z NPM registrů za poslední týden. Jako poslední je uveden údaj nepřímo vypovídající o finanční stabilitě frameworku, uskutečněný analýzou sponzorů projektu. Všechny tyto informace jsou datovány ke dni 10.3.2024. Data o oblíbenosti byla čerpána z jednotlivých GitHub repositářů (vuejs / vue, 2023), (vuejs / core, 2024), (quasarframework / quasar, 2024), (nuxt / nuxt, 2024). Nápodobně byla získávána i data o reálném využití z jednotlivých NPM registrů (vue, nedatováno), (quasar, nedatováno), (nuxt, nedatováno). V neposlední řadě, data, o finanční stabilitě byla extrahována z jednotlivých GitHub sponzorských stránek (You,

nedatováno), (Stoenescu, nedatováno), (Nuxt, nedatováno) a mimo to z již zmíněných GitHub repositářů a dokumentací Quasar.js a Nuxt 3.

	Vue.js	Quasar.js	Nuxt 3
Oblíbenost (hvězdy)	207K + 43.7K	25K	51K
Reálné využití (stažení)	4 832 278	126 216	628 074
Finanční stabilita (bez jednotky)	Appwrite, Storyblok, Ionic, Open Collective, Sentry a mnoho dalších + 227 GitHub uživatelů	Digital Ocean, Dream Monkey, Campus Cloud, Letsbutterfly a další + 164 GitHub uživatelů	Netlify, Vercel, AWS, Strapi, Open Collective a další + 107 GitHub uživatelů

Tabulka 1: Porovnání životnosti frameworků, zdroj: vlastní.

11.3 Cíl a zaměření

Vybrané frameworky se společně protínají v tvorbě webových aplikací, ale jak již bylo nastíněno v kapitole 5, každý z frameworků definuje svůj primární cíl jinak. Samotné Vue.js se zaměřuje na tvorbu SPA, Nuxt 3 je veden jako nadstavba Vue.js k vylepšení možností při tvorbě webových stránek, a to především podporou různých vykreslovacích technik. Na druhou stranu je tu Quasar.js poskytující nadstavbu Vue.js k vylepšení možností při tvorbě nejen aplikací a webových aplikací, ale také rozšíření prohlížeče. Funkcionalita a také architektura frameworků přímo vychází z jejich implicitního zaměření.

11.4 Rozšiřitelnost a ekosystém

Dalším důležitým aspektem frameworku je přizpůsobivost vypovídající o možné rozšiřitelnosti či nahraditelnosti funkcionality frameworku, což dále podporuje růst tzv. „ekosystému“. Na základě dokumentací lze říci, že tuto podmínku splňují všechny zkoumané frameworky. U Vue.js je rozšiřitelnost dosažena především za pomoci tzv. pluginů. O rozšiřitelnosti dále hovoří fakt, že další zmíněné frameworky jsou na Vue.js postavené. Quasar.js umožňuje rozšíření např. za pomocí tzv. „*Boot*“ a „*Middleware*“ souborů a ekosystém podporuje skrze tzv. „*App Extensions*“. Nuxt 3 stejně tak jako

Quasar.js využívá např. tzv. „*Middleware*“ soubory, které doplňují např. pluginy, načež ekosystém je podpořen tzv. „*moduly*“.

12 Shrnutí a diskuse výsledků

V rámci měření byla získána sada výsledků popisující výpočetní efektivitu jednotlivých implementovaných a částečně předvedených řešení, na vybraných modelových situacích, na základě vybraných metrik. Výsledky naměřené jsou v této práci členěny do dvou základních skupin. První skupinou jsou všechna získaná data. Lze z nich vyvodit jednoznačný výstup, ale je třeba uvážit další aspekty, které mohou na metriku mít vliv při reálném nasazení a následně vyhodnotit, jaký bude v realitě mít naměřený výsledek význam. Druhá skupina výsledků je inspirována, nebo lépe řečeno přímo vychází z fungování nástroje Lighthouse. Těmto je třeba přikládat zvláštní význam vzhledem k faktu, že přímo jsou, či z nich vychází metriky, které vyhledávač Google využívá při výpočtu celkové SEO úrovně a mají tak silný význam nejen pro spokojenost opravdových zákazníků, ale také pro získání zákazníků nových. Toto je pro většinu aplikací zásadní a význam takových metrik tedy nemusí být zpochybňován. Na druhou stranu i ostatní metriky mohou potencionálně být nemálo důležité.

Pokud by měla být krátce zodpovězena otázka, zdali je architektura UA z výkonnostního hlediska přínosná, poté by bylo možné odměnit vítězství v rámci modelových aplikací s hodnocením dle Lighthouse následujícím způsobem:

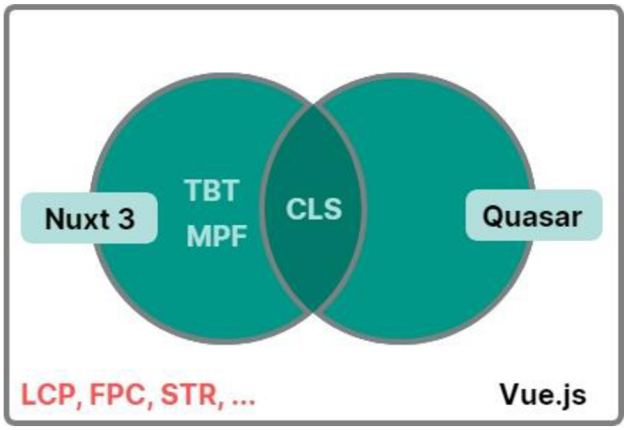
- První místo: 3 body.
- Druhé místo: 2 body.
- Třetí místo: 1 bod.

Výstupem bude v takovém případě tabulka bezpochyby vypovídající o přínosu UA architektury. Je možné si povšimnou, že naprostého vítězství SPA dosáhla pouze u interní aplikace, jenž dle kapitoly 6.2.4 představuje de facto SPA vzhledem k tomu, že UR je využito pouze na jedné z desíti stránek, kde zbytek pracuje plně na CSR vykreslování. Na druhou stranu, jak ukazuje Tabulka 9, bylo i zde vítězství velice těsné, a to především pro UA tvořenou frameworkem Quasar.js, z čehož lze usuzovat, že u zkoumaných frameworků nedochází přes navýšení efektivity při použití UR ani k výraznému snížení použitelnosti v rámci čistého CSR. Tím bylo prokázáno tvrzení v práci J. Josefa (Josef & Malý, 2016) hovořící o možném doručení lepšího uživatelského zážitku při prvotním načtení stránky.

Aplikace	Nuxt 3	Quasar.js	Vue.js
E-shop	3b	2b	1b
Diskusní fórum	3b	2b	1b
Interní aplikace	1b	2b	3b
Prezentační web	2b	3b	1b
Sociální síť	3b	1b	2b
Celkem	**12b**	**10b**	**8b**

Tabulka 2: Sumarizace výsledků Lighthouse měření, zdroj: vlastní.

K vyvození konkrétnějších výstupů práce je třeba zabřednout do analýzy kompletních výsledků, z nichž lze poměrně spolehlivě extrahovat následující údaje. Modelové UA postavené na frameworku Nuxt 3 trumfovaly základní Vue.js aplikace v metrikách TBT, MPF a CLS. Pro framework Quasar.js poté platí ze zmíněných pouze metrika CLS. Dominance v rámci jednotlivých metrik je také znázorněna následujícím Vennovým diagramem:



Obr. 20: Vennův diagram výsledných metrik, zdroj: autor.

Jak naznačuje graf, oba ze zmíněných frameworků přinesly zlepšení CLS. Snížení bylo vykonáno extrakcí všech souborů CSS do vrchní části HTML, konkrétně do sekce „*head*“. V rámci synchronní nátury těchto souborů jsou tyto vnímány prohlížeči jako tzv. „*render-blocking*“ zdroje, tedy zdroje bránící vykreslení stránky. Popsaný postup při správné implementaci zaručuje, že podoba stránky se již po prvním vykreslení nebude muset bez uživatelské interakce měnit. Z pohledu uživatele je tak podpořena konzistentní orientace na webu. V realitě k této změně může stále dojít např. v závislosti na vykonání asynchronního skriptu, či CSS. Vylepšení v rámci metriky CLS, jde v závislosti na použité technice ruku v ruce se snížením efektivity u metrik

vykreslení, jako FCP, či v určitých případech i LCP. Důvodem je samozřejmě dříve zmíněné blokování vykreslení, které má v tomto případě pozitivní efekt na metriku CLS. Na druhou stranu ale výrazně brzdí vykreslení časem stažení a aplikace CSS. Velice zajímavá je také konkrétní implementace zmíněné extrakce stylů u frameworků. Framework Nuxt 3 totiž vloží CSS do HTML značení dvěma způsoby. Zaprvé vytvoří interní styl pro každý jeden CSS soubor použitý na stránce a zadruhé tyto interní styly následují externí synchronní požadavky na již jednou interně vložené styly komponent, což zbytečně zdržuje vykreslení stránky. Přičemž by tyto mohly být vynechány a ponechány v kompetenci vstupního souboru Vue.js, nebo případně vynechány z procesu extrakce do interního stylu. Framework Quasar.js v opačném případě nevymýšlí komplikovaná nefunkční řešení, ale bohužel v tomto ohledu nevymýšlí řešení žádná. Externí CSS soubory se v rámci zdrojového kódu aplikace vyskytují všude v hlavičce a nejsou propagovány do její vrchní části. Bohužel tím dochází k tomu, že je před soubory CSS blokujícími vykreslení staženo potencionálně několik dalších zdrojů, při čemž každý může výrazně pozdržet první vykreslení stránky. Žádná ze zmíněných optimalizací není možná pro výchozí Vue.js SPA, jelikož tyto používají de facto prázdný HTML soubor odesílaný ve stejné formě všem stránkám aplikace, kde není předem známo, které komponenty bude aplikace využívat (mimo ty naprosto globální). Na základě toho byla v podstatě potvrzena hypotéza Osmaniho a Millera (Osmani & Miller, 2019) hovořící o snížení FCP při využití architektury UR, jelikož v případě, kdy by nebylo CSS dané stránky vyžadováno před jejím počátečním vykreslením, tak by klientovi stačilo pouze interpretovat HTML vrácený serverem a nebylo by nutné načtení vstupního souboru SPA k jejímu vygenerování. Na druhou stranu taková implementace by uživateli zajisté prezentovala tzv. „*FOUC*“ hovořící o krátké prezentaci nenastylovaného obsahu uživateli dříve, než se tento styl podaří stáhnout a použít, což by vedlo k velmi špatnému uživatelskému zážitku.

Rychlost prvního načtení stránky by dále mohla být volitelně podpořena technikou zvanou „*critical CSS*“ (vol. přel. z angličtiny: kritické CSS). Ta hovoří o extrakci stylů potřebných pro vykreslení pouze obsahu, jenž uživatel uvidí při prvním pohledu po navigaci na stránku a asynchronní načtení zbytku stylování, jenž by vedlo k dalšímu snížení času prvního vykreslení stránky. Technika má samozřejmě i své problémové části, jako např. „*jaký obsah bude prezentován uživateli*“ a „*co se stane při skrolování*“, ale při pečlivém nasazení může být velice prospěšná. V současné době se objevují i další

techniky, které by mohly dále navyšovat efektivitu UA jako např. technika „*Streaming server-side rendering*“ umožňující přenášet počáteční HTML odpověď serveru po malých částech, díky čemuž první části stránky dorazí k uživateli rychleji a ten je tak schopen na nich začít dříve pracovat. Dalšími technikami, které by bylo dle Osmaniho a Millera (Osmani & Miller, 2019) možné použít ke zlepšení výkonnosti stránky, jsou „*partial*“ či „*lazy*“ hydratace vedoucí na možné snížení v rámci TBT a TTI.

Mimo všechny zmíněné výkonnostní aspekty nelze opomenout ani jednoznačné výhody v rámci SEO a podpory klientů bez JavaScriptu. Jelikož, jak bylo nastíněno v kapitole 10.1 UA, poskytuje obsah stránky v určité míře i pro „*web crawlery*“ a uživatele bez JS, což se plně shoduje s očekáváními danými poznatky v kapitole 4.7.

13 Závěry a doporučení

V rámci práce byly objasněny základní pojmy, na kterých práce stojí a popsány techniky, jenž je možné při implementaci UA použít, a které mají vliv nejen na efektivní vykreslení stránky, ale také SEO a použitelnost bez JS. Dále byly představeny všechny frameworky, na nichž je postaven empirický výzkum zabývající se využitím UA v rámci kontextu SPA frameworku Vue.js. Samotný výzkum se zabýval vývojem několika běžných demonstračních SPA, pro které byly předvedeny základní kroky migrace na UA využívající frameworky Quasar.js a Nuxt 3. Na všech vytvořených aplikacích bylo následně provedeno výkonnostní měření, jenž odhalilo přínos, či pokles výkonnosti architektury UA v porovnání s architekturou SPA. V neposlední řadě byly sepsány vybrané z dalších aspektů, jenž je dobré vzít v úvahu před výběrem řešení. Mimo zmíněné byla také samostatná kapitola věnována experimentální vykreslovací technice On-Demand ISR mající pro UA potencionálně obrovský přínos, ale současně vyžadující velké množství implementací. Výsledky empirického výzkumu nelze plně generalizovat pro ostatní SPA a UA. Konkrétní implementace a výkonnost se mohou lišit v závislosti na vybraném frameworku, jehož výkonnost bude dále odvozena na základě složitosti a kvalitě dané aplikace. Současně i přesto, že bylo měření provedeno v konstantních a doporučených podmínkách, tak stále plně nevypovídá o výkonnosti na různých zařízeních. Pro zjištění relevantnějších výsledků by bylo třeba výzkum provádět v uzavřeném produkčním nasazení, jehož by bylo složité docílit. Dále by bylo dosažení rozdílných výsledků možné při využití protokolu HTTP/2 (Hypertext Transfer Protocol) namísto využitého protokolu HTTP/1.1 a měřením s odlišnými parametry. V každém případě lze využití UA v praxi jednoznačně doporučit pro aplikace profitující z lepšího SEO a podpory bez JS. Z výkonnostního hlediska, jak se ukázalo, mohou být UA prospěšné i naopak. Bylo zjištěno, že UA nepochybně má potencionál pro zlepšení výkonnostní stránky webu pro výčet metrik daných nástrojem Lighthouse. Současně ale bylo zjištěno, že přínos v těchto metrikách může být vykoupen horším výsledkem v metrikách ostatních. Na základě toho tedy nelze definovat jednoznačné doporučení pro praxi, je nutné zvážit klíčové priority plánovaného projektu a na základě těchto učinit rozhodnutí. V každém případě je při výběru UA frameworku třeba současně s vhodností frameworku vyhodnotit také jeho životnost, vzhledem k tomu, že horší framework je stále lepší než žádný.

14 Seznam použité literatury

admin. (nedatováno). *A Guide to Isomorphic (Universal) JavaScript | Orion eSolutions*. Retrieved 20. 1 2024, from Orion. https://orionesolutions.com/a-guide-to-isomorphic-universal-javascript/

AirbnbEng. (11. 11 2013). *Isomorphic JavaScript: The Future of Web Apps*. Retrieved 20. 1 2024, from Medium. https://medium.com/airbnb-engineering/isomorphic-javascript-the-future-of-web-apps-10882b7a2ebc#.4nyzv6jea

Altynpara, E., & Ropalo, Y. (05. 10 2023). *Single-page Application vs. Multi-page Application: What to Choose*. Retrieved 20. 1 2024, from Cleveroad Inc. - Web and App development company. https://www.cleveroad.com/blog/single-page-application-vs-multi-page-application/

Bhatt, J. (22. 4 2020). *Quasar Framework: Tutorial part 1 — Introduction*. Retrieved 27. 1 2024, from Medium. https://medium.com/@bhattjaldhi27/quasar-framework-tutorial-part-1-introduction-8628f7bf91e0

Cocca, G. (6. 3 2023). *Rendering Patterns for Web Apps – Server-Side, Client-Side, and SSG Explained*. Retrieved 23. 1 2024, from freeCodeCamp. https://www.freecodecamp.org/news/rendering-patterns/#static-websites

deniskoronets. (27. 10 2023). *Can you invalidate SWR/static routes depending on user actions using Hybrid Rendering in Nuxt 3?* Retrieved 29. 2 2024, from Stackoverflow. https://stackoverflow.com/questions/75849646/can-you-invalidate-swr-static-routes-depending-on-user-actions-using-hybrid-rend

Dynamic rendering as a workaround. (7. 2 2024). Google Search Central. https://developers.google.com/search/docs/crawling-indexing/javascript/dynamic-rendering

Frequently Asked Questions. (nedatováno). Retrieved 26. 1 2024, from Vue.js. https://vuejs.org/about/faq.html

Gallardo, E. G. (9. 1 2023). *What Is MVVM Architecture?* Retrieved 26. 1 2024, from Built in. https://builtin.com/software-engineering-perspectives/mvvm-architecture

GoogleChrome / Lighthouse. (19. 4 2023). GitHub. https://github.com/GoogleChrome/lighthouse/commit/2b8caf21a3e9d03633b52e43e35e97291ae6553d

Gordon, E. K. (nedatováno). *1 Introduction to Isomorphic Web Application Architecture · Isomorphic Web Applications: Universal Development with React*. (Manning) Retrieved 20. 1 2024, from Manning.

Hengeveld, G. (3. 9 2015). *Isomorphism vs Universal JavaScript*. Retrieved 20. 1 2024, from Medium. https://medium.com/@ghengeveld/isomorphism-vs-universal-javascript-4b47fb481beb

IBM. (4. 3 2021). *What is Ajax?* Retrieved 20. 1 2024, from IBM.

Ibsen, M. (7. 4 2021). *3 Ways of Rendering on The Web*. Retrieved 23. 1 2024, from Medium. https://medium.com/compendium/3-ways-of-rendering-on-the-web-4363864c859e

Incremental Static Regeneration (ISR). (nedatováno). Retrieved 24. 1 2024, from Next.js. https://nextjs.org/docs/pages/building-your-application/data-fetching/incremental-static-regeneration

Installation. (nedatováno). Retrieved 29. 2 2024, from Nuxt. https://nuxt.com/docs/getting-started/installation

Introduction. (nedatováno). Retrieved 26. 1 2024, from Vue.js. https://vuejs.org/guide/introduction.html

Isomorphic react. (nedatováno). Retrieved 20. 1 2024, from OhMyCrawl. https://www.ohmycrawl.com/react/isomorphic/

Josef, J., & Malý, F. (7. 9 2016). *Principy a vývoj isomorfních webových aplikací.* https://doi.org/http://evskp.uhk.cz/eM4589

Lange, D. (9 2022). *On demand ISR with Next.JS and Sanity*. Retrieved 28. 1 2024, from Finiam. https://blog.finiam.com/blog/on-demand-isr-with-next-js-and-sanity

Lapinski, N. (9. 9 2022). *How You Render Can Affect Your SEO (CSR vs SSR vs Dynamic Rendering)*. Retrieved 15. 3 2024, from Medium. https://medium.com/@natelapinski/how-you-render-can-affect-your-seo-csr-vs-ssr-vs-dynamic-815a91dea894

Mamgain, D. (29. 11 2022). *Single-page application vs Multi-page application: What Is Better for Your Project?* Retrieved 10. 1 2024, from LinkedIn. https://www.linkedin.com/pulse/single-page-application-vs-multi-page-what-better-your-

Marshall, T. (3. 8 2022). *How to improve cache efficiency and reduce costs with Next.js on-demand ISR*. Retrieved 28. 1 2024, from Kontent.ai. https://kontent.ai/blog/leveraging-next-js-on-demand-isr/

Mauser, L. (7. 12 2023). *Is Nuxt 3 Really Production Ready?* Retrieved 27. 1 2024, from Dev. https://dev.to/nuxt-wimadev/is-nuxt-3-really-production-ready-1gik

Naylor, S. A. (13. 4 2023). *The Acronyms of Rendering on the Web*. Retrieved 23. 1 2024, from Netlify. https://www.netlify.com/blog/the-acronyms-of-rendering/

Neoteric. (2. 12 2016). *Single-page application vs. multiple-page application*. Retrieved 20. 1 2024, from Medium. https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

Node.js. (nedatováno). *Introduction to Node.js*. Retrieved 30. 10 2023, from Node.js. https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

NR, J. (10. 5 2015). *What is an isomorphic application?* Retrieved 20. 1 2024, from Lullabot.

nuxt. (nedatováno). Retrieved 10. 3 2024, from npm. https://www.npmjs.com/package/nuxt

nuxt / nuxt. (22. 2 2024). Retrieved 10. 3 2024, from GitHub. https://github.com/nuxt/nuxt

Nuxt. (nedatováno). *https://github.com/sponsors/nuxt*. Retrieved 10. 3 2024, from GitHub. https://github.com/sponsors/nuxt

Onyenma, M. (5. 9 2023). *Nuxt*. Retrieved 27. 1 2024, from Bejamas. https://bejamas.io/discovery/static-site-generators/nuxtjs/

Oracle. (nedatováno). *1.2.1 What is MySQL?*, 8.0. Retrieved 30. 10 2023, from MySQL. https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

Osmani, A., & Miller, J. (2. 6 2019). *Rendering on the Web*. Retrieved 24. 1 2024, from web.dev. https://web.dev/articles/rendering-on-the-web

Overview | Lighthouse. (27. 9 2016). Retrieved 10. 3 2024, from Chrome for Developers. https://developer.chrome.com/docs/lighthouse/overview

quasar. (nedatováno). Retrieved 10. 3 2024, from npm. https://www.npmjs.com/package/quasar

Quasar CLI. (nedatováno). Retrieved 6. 2 2024, from Quasar. https://quasar.dev/start/quasar-cli

Quasar Framework Roadmap. (nedatováno). Retrieved 27. 1 2024, from Github. https://github.com/quasarframework/quasar/blob/dev/ROADMAP.md

quasarframework / quasar. (8. 3 2024). Retrieved 10. 3 2024, from GitHub. https://github.com/quasarframework/quasar

Quick Start | Vue.js. (nedatováno). Retrieved 23. 10 2023, from Vue.js. https://vuejs.org/guide/quick-start.html#creating-a-vue-application

Ram, P. (19. 10 2021). *Server Side Rendering (SSR) vs. Client Side Rendering (CSR) vs. Pre-Rendering using Static Site Generators (SSG) and client-side hydration.* Retrieved 24. 1 2024, from Medium. https://medium.com/@prashantramnyc/server-side-rendering-ssr-vs-client-side-rendering-csr-vs-pre-rendering-using-static-site-89f2d05182ef

Rendering Modes. (nedatováno). Retrieved 20. 1 2024, from Nuxt. https://nuxt.com/docs/guide/concepts/rendering

Shopify Staff. (21. 9 2023). *How Google Lighthouse Helps Optimize Website Performance (2023)*. Retrieved 10. 3 2024, from Shopify. https://www.shopify.com/blog/google-lighthouse

Stoenescu, R. (nedatováno). *Become a sponsor to Razvan Stoenescu*. Retrieved 10. 3 2024, from GitHub. https://github.com/sponsors/rstoenescu

Team Product Engineering. (nedatováno). *Navigating Web Efficiency with 4 Modern Web Rendering Strategies*. Retrieved 24. 1 2024, from Celestial Systems. https://www.celestialsys.com/blogs/navigating-web-efficiency-with-4-modern-web-rendering-strategies

The Intuitive Vue Framework. (nedatováno). Retrieved 27. 1 2024, from Nuxt. https://nuxt.com/

Two strategies for migrating an existing application to a new framework. (29. 1 2018). Retrieved 6. 2 2024, from Oprea. https://oprea.rocks/blog/two-strategies-for-migrating-an-existing-application-to-a-new-framework.html

Understand the JavaScript SEO basics. (7. 2 2024). Google Search Central. https://developers.google.com/search/docs/crawling-indexing/javascript/javascript-seo-basics#how-googlebot-processes-javascript

Veverka, R. (nedatováno). *Lekce 11 - AJAX v JavaScriptu - Základní dotazy*. Retrieved 20. 1 2024, from Itnetwork. https://www.itnetwork.cz/javascript/oop/ajax-v-javascriptu-zakladni-dotazy

Vitarag, D. (4. 10 2023). *8 Best Single-Page Application Frameworks For Web App Development*. Retrieved 26. 1 2024, from Medium. https://medium.com/@vitarag/8-best-single-page-application-frameworks-for-web-app-development-eb81b035c545

vue. (nedatováno). Retrieved 10. 3 2024, from npm. https://www.npmjs.com/package/vue

vuejs / core. (28. 2 2024). Retrieved 10. 3 2024, from GitHub. https://github.com/vuejs/core

vuejs / vue. (24. 10 2023). Retrieved 10. 3 2024, from GitHub. https://github.com/vuejs/vue

What is application migration? (nedatováno). Retrieved 6. 2 2024, from IBM. https://www.ibm.com/topics/application-migration

Which search engines index Javascript content? (nedatováno). Retrieved 15. 3 2024, from Unless. https://unless.com/en/science/javascript-indexing/

Why Does Lighthouse Lab Data Not Match Field Data? (12. 12 2022). Retrieved 11. 3 2024, from DebugBear. https://www.debugbear.com/blog/lighthouse-lab-data-not-matching-field-data

Why Quasar? (nedatováno). Retrieved 27. 1 2024, from Quasar. https://quasar.dev/introduction-to-quasar

Yi, Y., & Li, Z. (2021). Design and Implementation of Music Web Application based on Vue and Spring Boot. New York: Association for Computing Machinery.

You, E. (nedatováno). *Become a sponsor to Evan You*. Retrieved 10. 3 2024, from GitHub. https://github.com/sponsors/yyx990803

15 Přílohy

Příloha č. 1

Příloha č. 2

first-contentful-paint	FCP
largest-contentful-paint	LCP
first-meaningful-paint	FMP
speed-index	SI
total-blocking-time	TBT
max-potential-fid	MPF
cumulative-layout-shift	CLS
server-response-time	SRT
interactive	I
total-byte-weight	TBW
network-requests	NR
mainthread-work-breakdown	MWB

Tabulka 3: Vysvětlivka zkratek Lighthouse auditů, zdroj: vlastní.

Audit	Nuxt 3	Quasar.js	Vue.js	Průměr
FCP Q1 (ms)	2914.6 (9.1%)	2816.4 (5.5%)	2280.8 (-14.6%)	2670.6
FCP Q2 (ms)	3181.8 (9.9%)	3101.4 (7.1%)	2404.7 (-17.0%)	2896.0
FCP Q3 (ms)	3711.9 (14.0%)	3213.5 (-1.3%)	2843.8 (-12.7%)	3256.4
LCP Q1 (ms)	4241.5 (-4.3%)	4906.5 (10.7%)	4149.0 (-6.4%)	4432.3
LCP Q2 (ms)	4366.5 (-5.2%)	5219.2 (13.3%)	4238.7 (-8.0%)	4608.1
LCP Q3 (ms)	4818.8 (-2.6%)	5420.3 (9.6%)	4596.8 (-7.0%)	4945.3
FMP Q1 (ms)	3054.1 (5.0%)	2976.0 (2.3%)	2697.3 (-7.3%)	2909.1
FMP Q2 (ms)	3322.2 (7.4%)	3181.3 (2.9%)	2774.4 (-10.3%)	3092.6
FMP Q3 (ms)	3739.4 (12.0%)	3213.5 (-3.8%)	3066.6 (-8.2%)	3339.8
SI Q1 (ms)	2918.2 (7.3%)	2842.9 (4.5%)	2399.0 (-11.8%)	2720.1
SI Q2 (ms)	3190.0 (9.1%)	3101.4 (6.1%)	2477.9 (-15.2%)	2923.1
SI Q3 (ms)	3711.9 (13.2%)	3213.5 (-2.0%)	2908.0 (-11.3%)	3277.8
TBT Q1 (ms)	22.6 (-56.6%)	73.9 (42.2%)	59.4 (14.4%)	52.0
TBT Q2 (ms)	30.7 (-47.6%)	82.8 (41.4%)	62.2 (6.2%)	58.6
TBT Q3 (ms)	37.4 (-43.6%)	95.7 (44.3%)	65.9 (-0.7%)	66.3
MPF Q1 (ms)	77.8 (-26.5%)	127.5 (20.4%)	112.4 (6.1%)	105.9
MPF Q2 (ms)	93.4 (-19.7%)	131.3 (12.9%)	124.3 (6.9%)	116.3
MPF Q3 (ms)	97.7 (-20.3%)	141.0 (15.0%)	129.1 (5.3%)	122.6
CLS Q1 (-)	0.0500 (-59.6%)	0.0070 (-94.3%)	0.3140 (153.9%)	0.1237
CLS Q2 (-)	0.0504 (-59.5%)	0.0080 (-93.6%)	0.3151 (153.1%)	0.1245
CLS Q3 (-)	0.0505 (-59.5%)	0.0081 (-93.5%)	0.3156 (153.1%)	0.1247
SRT Q1 (ms)	6.0 (26.8%)	6.8 (43.6%)	1.4 (-70.4%)	4.7
SRT Q2 (ms)	6.2 (11.4%)	8.9 (59.5%)	1.6 (-70.9%)	5.6
SRT Q3 (ms)	6.9 (-1.7%)	12.3 (75.3%)	1.8 (-73.6%)	7.0
I Q1 (ms)	3820.6 (-3.4%)	4501.3 (13.8%)	3544.9 (-10.4%)	3955.6
I Q2 (ms)	4043.8 (-0.9%)	4622.9 (13.3%)	3570.4 (-12.5%)	4079.0
I Q3 (ms)	4113.7 (-0.8%)	4658.7 (12.3%)	3672.4 (-11.5%)	4148.3
TBW Q1 (byte)	642842.1 (-7.3%)	792456.9 (14.3%)	644256.1 (-7.1%)	693185.0
TBW Q2 (byte)	642842.1 (-7.3%)	792456.9 (14.3%)	644256.1 (-7.1%)	693185.0
TBW Q3 (byte)	642842.1 (-7.3%)	792456.9 (14.3%)	644256.1 (-7.1%)	693185.0
NR Q1 (-)	25.5 (12.1%)	23.1 (1.6%)	19.6 (-13.7%)	22.7
NR Q2 (-)	25.5 (12.1%)	23.1 (1.6%)	19.6 (-13.7%)	22.8
NR Q3 (-)	25.5 (12.1%)	23.1 (1.6%)	19.6 (-13.7%)	22.8
MWB Q1 (ms)	1096.1 (7.2%)	1019.9 (-0.3%)	951.4 (-6.9%)	1022.5
MWB Q2 (ms)	1115.8 (7.5%)	1036.7 (-0.2%)	962.6 (-7.3%)	1038.3
MWB Q3 (ms)	1131.8 (7.1%)	1057.9 (0.1%)	981.9 (-7.1%)	1057.2
Celkem Q1 (%)	-90.1	64.3	25.8	0.0
Celkem Q2 (%)	-82.9	78.6	4.2	0.0
Celkem Q3 (%)	-77.5	72.0	5.5	0.0
Celkem (%)	-250.5	214.9	35.6	0.0

Tabulka 4: Výsledky úplného výkonnostního měření pro E-shop, zdroj: vlastní.

Audit	Nuxt 3	Quasar.js	Vue.js
FCP Q3 (ms)	14.0% * 0.1	-1.3% * 0.1	-12.7% * 0.1
LCP Q3 (ms)	-2.6% * 0.25	9.6% * 0.25	-7.0% * 0.25
TBT Q3 (ms)	-43.6% * 0.3	44.3% * 0.3	-0.7% * 0.3
CLS Q3 (-)	-59.5% * 0.25	-93.5% * 0.25	153.1% * 0.25
SI Q3 (ms)	13.2% * 0.1	-2.0% * 0.1	-11.3% * 0.1
Celkem Q3	-25.8849	-8.015	33.915

Tabulka 5: Výsledky Lighthouse výkonnostního měření pro E-shop, zdroj: vlastní.

Audit	Nuxt 3	Quasar.js	Vue.js	Průměr
FCP Q1 (ms)	3034.8 (14.1%)	2712.5 (2.0%)	2233.0 (-16.1%)	2660.1
FCP Q2 (ms)	3833.9 (26.6%)	2719.1 (-10.2%)	2533.1 (-16.4%)	3028.7
FCP Q3 (ms)	4012.9 (27.3%)	2725.0 (-13.5%)	2717.9 (-13.8%)	3151.9
LCP Q1 (ms)	3574.8 (-11.0%)	4643.2 (15.6%)	3831.4 (-4.6%)	4016.5
LCP Q2 (ms)	4413.1 (1.2%)	4653.6 (6.7%)	4017.2 (-7.9%)	4361.3
LCP Q3 (ms)	4614.8 (3.7%)	4661.0 (4.8%)	4069.7 (-8.5%)	4448.5
FMP Q1 (ms)	3096.5 (14.2%)	2712.5 (0.0%)	2326.2 (-14.2%)	2711.8
FMP Q2 (ms)	3915.6 (26.8%)	2719.1 (-11.9%)	2626.3 (-14.9%)	3087.0
FMP Q3 (ms)	4102.9 (27.7%)	2725.0 (-15.2%)	2811.6 (-12.5%)	3213.2
SI Q1 (ms)	3034.8 (11.7%)	2712.5 (-0.1%)	2400.8 (-11.6%)	2716.0
SI Q2 (ms)	3833.9 (25.7%)	2719.1 (-10.8%)	2594.2 (-14.9%)	3049.1
SI Q3 (ms)	4012.9 (26.5%)	2725.0 (-14.1%)	2778.6 (-12.4%)	3172.2
TBT Q1 (ms)	18.2 (-72.4%)	77.6 (18.1%)	101.4 (54.3%)	65.7
TBT Q2 (ms)	21.0 (-72.3%)	81.8 (8.1%)	124.1 (64.2%)	75.6
TBT Q3 (ms)	36.6 (-56.5%)	85.7 (1.8%)	130.3 (54.7%)	84.2
MPF Q1 (ms)	66.5 (-37.4%)	121.8 (14.7%)	130.3 (22.7%)	106.2
MPF Q2 (ms)	68.5 (-37.6%)	125.1 (13.9%)	135.8 (23.7%)	109.8
MPF Q3 (ms)	73.0 (-36.0%)	128.4 (12.8%)	140.3 (23.2%)	113.9
CLS Q1 (-)	0.0008 (-92.0%)	0.0008 (-92.0%)	0.0284 (184.0%)	0.0100
CLS Q2 (-)	0.0008 (-92.0%)	0.0008 (-92.0%)	0.0284 (184.0%)	0.0100
CLS Q3 (-)	0.0008 (-92.0%)	0.0008 (-92.0%)	0.0284 (184.0%)	0.0100
SRT Q1 (ms)	2.2 (-50.1%)	9.8 (124.0%)	1.1 (-73.9%)	4.4
SRT Q2 (ms)	2.5 (-47.6%)	10.2 (116.5%)	1.5 (-68.9%)	4.7
SRT Q3 (ms)	2.8 (-45.2%)	10.8 (109.5%)	1.8 (-64.4%)	5.2
I Q1 (ms)	3805.9 (-2.4%)	4153.1 (6.5%)	3735.3 (-4.2%)	3898.1
I Q2 (ms)	4235.4 (4.2%)	4162.1 (2.3%)	3802.2 (-6.5%)	4066.6
I Q3 (ms)	4402.0 (6.1%)	4168.4 (0.5%)	3876.5 (-6.6%)	4149.0
TBW Q1 (byte)	667373.3 (3.4%)	722668.6 (11.9%)	546653.6 (-15.3%)	645565.2
TBW Q2 (byte)	667467.2 (3.4%)	722668.6 (11.9%)	546653.6 (-15.3%)	645596.5

TBW Q3 (byte)	667566.1 (3.4%)	722668.6 (11.9%)	546653.6 (-15.3%)	645629.4
NR Q1 (-)	23.6 (24.2%)	19.2 (1.1%)	14.2 (-25.3%)	19.0
NR Q2 (-)	23.6 (24.2%)	19.2 (1.1%)	14.2 (-25.3%)	19.0
NR Q3 (-)	23.6 (24.2%)	19.2 (1.1%)	14.2 (-25.3%)	19.0
MWB Q1 (ms)	1112.3 (15.4%)	917.7 (-4.8%)	861.5 (-10.6%)	963.8
MWB Q2 (ms)	1135.3 (15.5%)	933.1 (-5.1%)	879.9 (-10.5%)	982.7
MWB Q3 (ms)	1159.3 (16.1%)	946.1 (-5.3%)	891.5 (-10.8%)	999.0
Celkem Q1 (%)	-182.2	97.0	85.3	0.0
Celkem Q2 (%)	-121.9	30.6	91.2	0.0
Celkem Q3 (%)	-94.7	2.2	92.4	0.0
Celkem (%)	-398.7	129.8	268.9	0.0

Tabulka 6: Výsledky úplného výkonnostního měření pro Diskusní fórum, zdroj: vlastní.

Audit	Nuxt 3	Quasar.js	Vue.js
FCP Q3 (ms)	27.3% * 0.1	-13.5% * 0.1	-13.8% * 0.1
LCP Q3 (ms)	3.7% * 0.25	4.8% * 0.25	-8.5% * 0.25
TBT Q3 (ms)	-56.5% * 0.3	1.8% * 0.3	54.7% * 0.3
CLS Q3 (-)	-92.0% * 0.25	-92.0% * 0.25	184.0% * 0.25
SI Q3 (ms)	26.5% * 0.1	-14.1% * 0.1	-12.4% * 0.1
Celkem Q3	-33.645	-24.02	57.665

Tabulka 7: Výsledky Lighthouse výkonnostního měření pro Diskusní fórum, zdroj: vlastní.

Audit	Nuxt 3	Quasar.js	Vue.js	Průměr
FCP Q1 (ms)	2048.8 (-5.3%)	2297.0 (6.2%)	2144.0 (-0.9%)	2163.2
FCP Q2 (ms)	2155.0 (-7.6%)	2524.1 (8.2%)	2317.4 (-0.6%)	2332.2
FCP Q3 (ms)	2415.7 (-6.2%)	2942.5 (14.3%)	2365.7 (-8.1%)	2574.6
LCP Q1 (ms)	4178.8 (9.6%)	3729.2 (-2.2%)	3530.4 (-7.4%)	3812.8
LCP Q2 (ms)	4268.0 (7.8%)	3910.7 (-1.3%)	3703.7 (-6.5%)	3960.8
LCP Q3 (ms)	4481.8 (7.4%)	4277.0 (2.5%)	3755.8 (-10.0%)	4171.5
FMP Q1 (ms)	2359.9 (-4.4%)	2552.3 (3.4%)	2494.2 (1.0%)	2468.8
FMP Q2 (ms)	2466.9 (-6.7%)	2779.4 (5.1%)	2686.4 (1.6%)	2644.3
FMP Q3 (ms)	2691.7 (-5.5%)	3121.4 (9.6%)	2734.6 (-4.0%)	2849.2
SI Q1 (ms)	2052.3 (-8.7%)	2297.0 (2.2%)	2394.2 (6.5%)	2247.8
SI Q2 (ms)	2173.2 (-9.8%)	2524.1 (4.7%)	2533.3 (5.1%)	2410.2
SI Q3 (ms)	2415.7 (-11.4%)	2942.5 (7.9%)	2823.3 (3.5%)	2727.1
TBT Q1 (ms)	65.4 (47.8%)	34.0 (-23.1%)	33.3 (-24.6%)	44.2
TBT Q2 (ms)	72.3 (47.2%)	35.5 (-27.7%)	39.5 (-19.5%)	49.1
TBT Q3 (ms)	83.1 (46.8%)	40.9 (-27.8%)	45.8 (-19.1%)	56.6

MPF Q1 (ms)	105.0 (0.3%)	114.4 (9.3%)	94.6 (-9.6%)	104.7
MPF Q2 (ms)	109.1 (-2.4%)	126.5 (13.1%)	99.9 (-10.7%)	111.8
MPF Q3 (ms)	111.4 (-6.3%)	133.1 (11.9%)	112.2 (-5.6%)	118.9
CLS Q1 (-)	0.0605 (0.9%)	0.0582 (-2.9%)	0.0612 (2.0%)	0.0600
CLS Q2 (-)	0.0624 (1.7%)	0.0592 (-3.6%)	0.0625 (1.9%)	0.0614
CLS Q3 (-)	0.0626 (1.6%)	0.0595 (-3.5%)	0.0627 (1.8%)	0.0616
SRT Q1 (ms)	8.2 (11.3%)	11.9 (62.9%)	1.9 (-74.2%)	7.3
SRT Q2 (ms)	15.5 (1.8%)	28.2 (84.6%)	2.1 (-86.4%)	15.3
SRT Q3 (ms)	26.9 (20.0%)	38.1 (69.8%)	2.3 (-89.8%)	22.4
I Q1 (ms)	3716.8 (0.7%)	3592.8 (-2.7%)	3768.6 (2.1%)	3692.7
I Q2 (ms)	3733.6 (-0.3%)	3645.0 (-2.6%)	3853.4 (2.9%)	3744.0
I Q3 (ms)	3894.0 (0.5%)	3762.5 (-2.9%)	3969.9 (2.4%)	3875.4
TBW Q1 (byte)	622196.7 (7.1%)	583208.7 (0.4%)	537147.8 (-7.5%)	580851.1
TBW Q2 (byte)	622196.7 (7.1%)	583208.7 (0.4%)	537147.8 (-7.5%)	580851.1
TBW Q3 (byte)	622196.7 (7.1%)	583208.7 (0.4%)	537147.8 (-7.5%)	580851.1
NR Q1 (-)	24.5 (21.7%)	18.8 (-6.6%)	17.1 (-15.1%)	20.1
NR Q2 (-)	24.5 (21.7%)	18.8 (-6.6%)	17.1 (-15.1%)	20.1
NR Q3 (-)	24.5 (21.7%)	18.8 (-6.6%)	17.1 (-15.1%)	20.1
MWB Q1 (ms)	1040.8 (7.5%)	959.9 (-0.8%)	902.9 (-6.7%)	967.9
MWB Q2 (ms)	1060.4 (8.0%)	971.1 (-1.1%)	913.3 (-7.0%)	981.6
MWB Q3 (ms)	1076.6 (8.2%)	984.8 (-1.0%)	924.2 (-7.1%)	995.2
Celkem Q1 (%)	88.5	46.0	-134.5	0.0
Celkem Q2 (%)	68.4	73.3	-141.7	0.0
Celkem Q3 (%)	84.0	74.6	-158.6	0.0
Celkem (%)	240.9	193.8	-434.7	0.0

Tabulka 8: Výsledky úplného výkonnostního měření pro Interní aplikaci, zdroj: vlastní.

Audit	Nuxt 3	Quasar.js	Vue.js
FCP Q3 (ms)	-6.2% * 0.1	14.3% * 0.1	-8.1% * 0.1
LCP Q3 (ms)	7.4% * 0.25	2.5% * 0.25	-10.0% * 0.25
TBT Q3 (ms)	46.8% * 0.3	-27.8% * 0.3	-19.1% * 0.3
CLS Q3 (-)	1.6% * 0.25	-3.5% * 0.25	1.8% * 0.25
SI Q3 (ms)	-11.4% * 0.1	7.9% * 0.1	3.5% * 0.1
Celkem Q3	14.53	-6.37	-8.2400

Tabulka 9: Výsledky Lighthouse výkonnostního měření pro Interní aplikaci, zdroj: vlastní.

Audit	Nuxt 3	Quasar.js	Vue.js	Průměr
FCP Q1 (ms)	1657.4 (1.4%)	1892.5 (15.8%)	1354.6 (-17.1%)	1634.8
FCP Q2 (ms)	1958.5 (9.7%)	1894.2 (6.1%)	1505.3 (-15.7%)	1786.0
FCP Q3 (ms)	2185.6 (10.3%)	1896.2 (-4.3%)	1861.3 (-6.0%)	1981.0
LCP Q1 (ms)	2790.9 (-12.6%)	4007.5 (25.5%)	2782.4 (-12.9%)	3193.6
LCP Q2 (ms)	3614.8 (4.2%)	4010.7 (15.6%)	2783.3 (-19.8%)	3469.6
LCP Q3 (ms)	3617.7 (4.2%)	4014.5 (15.6%)	2788.5 (-19.7%)	3473.6
FMP Q1 (ms)	1657.4 (1.4%)	1892.5 (15.8%)	1354.6 (-17.1%)	1634.8
FMP Q2 (ms)	1958.5 (9.7%)	1894.2 (6.1%)	1505.3 (-15.7%)	1786.0
FMP Q3 (ms)	2185.6 (10.3%)	1896.2 (-4.3%)	1861.3 (-6.0%)	1981.0
SI Q1 (ms)	1657.4 (1.4%)	1892.5 (15.8%)	1354.6 (-17.1%)	1634.8
SI Q2 (ms)	1958.5 (9.7%)	1894.2 (6.1%)	1505.3 (-15.7%)	1786.0
SI Q3 (ms)	2185.6 (10.3%)	1896.2 (-4.3%)	1861.3 (-6.0%)	1981.0
TBT Q1 (ms)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0
TBT Q2 (ms)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0
TBT Q3 (ms)	0.0 (-100.0%)	0.0 (-100.0%)	45.6 (200.0%)	15.2
MPF Q1 (ms)	16.0 (0.0%)	16.0 (0.0%)	16.0 (0.0%)	16.0
MPF Q2 (ms)	16.0 (0.0%)	16.0 (0.0%)	16.0 (0.0%)	16.0
MPF Q3 (ms)	16.0 (-72.3%)	16.0 (-72.3%)	141.3 (144.6%)	57.8
CLS Q1 (-)	0.0063 (0.0%)	0.0063 (0.0%)	0.0063 (0.0%)	0.0063
CLS Q2 (-)	0.0063 (0.0%)	0.0063 (0.0%)	0.0063 (0.0%)	0.0063
CLS Q3 (-)	0.0063 (0.0%)	0.0063 (0.0%)	0.0063 (0.0%)	0.0063
SRT Q1 (ms)	0.9 (-21.4%)	1.2 (13.3%)	1.2 (8.2%)	1.1
SRT Q2 (ms)	1.2 (-6.7%)	1.4 (4.7%)	1.4 (1.9%)	1.3
SRT Q3 (ms)	1.5 (-20.6%)	2.2 (15.5%)	2.0 (5.0%)	1.9
I Q1 (ms)	1657.4 (1.4%)	1892.5 (15.7%)	1355.8 (-17.1%)	1635.2
I Q2 (ms)	1958.5 (6.7%)	1894.2 (3.2%)	1655.0 (-9.9%)	1835.9
I Q3 (ms)	2185.6 (5.1%)	1896.2 (-8.8%)	2158.3 (3.8%)	2080.0
TBW Q1 (byte)	566401.0 (-0.5%)	680072.0 (19.5%)	460686.0 (-19.0%)	569053.0
TBW Q2 (byte)	566401.0 (-0.5%)	680072.0 (19.5%)	460686.0 (-19.0%)	569053.0
TBW Q3 (byte)	566401.0 (-0.5%)	680072.0 (19.5%)	460686.0 (-19.0%)	569053.0
NR Q1 (-)	15.0 (21.6%)	13.0 (5.4%)	9.0 (-27.0%)	12.3
NR Q2 (-)	15.0 (21.6%)	13.0 (5.4%)	9.0 (-27.0%)	12.3
NR Q3 (-)	15.0 (21.6%)	13.0 (5.4%)	9.0 (-27.0%)	12.3
MWB Q1 (ms)	1169.4 (8.5%)	1060.5 (-1.6%)	1003.7 (-6.9%)	1077.9
MWB Q2 (ms)	1189.4 (8.3%)	1088.4 (-0.9%)	1016.8 (-7.4%)	1098.2
MWB Q3 (ms)	1203.9 (7.3%)	1106.9 (-1.4%)	1056.3 (-5.9%)	1122.3
Celkem Q1 (%)	1.1	125.1	-126.2	0.0
Celkem Q2 (%)	62.6	65.7	-128.3	0.0

Celkem Q3 (%)	-124.2	-139.3	263.6	0.0
Celkem (%)	-60.5	51.4	9.1	0.0

Tabulka 10: Výsledky úplného výkonnostního měření pro Prezentační stránky, zdroj: vlastní.

Audit	Nuxt 3	Quasar.js	Vue.js
FCP Q3 (ms)	10.3% * 0.1	-4.3% * 0.1	-6.0% * 0.1
LCP Q3 (ms)	4.2% * 0.25	15.6% * 0.25	-19.7% * 0.25
TBT Q3 (ms)	-100.0% * 0.3	-100.0% * 0.3	200.0% * 0.3
CLS Q3 (-)	0.0% * 0.25	0.0% * 0.25	0.0% * 0.25
SI Q3 (ms)	10.3% * 0.1	-4.3% * 0.1	-6.0% * 0.1
Celkem Q3	-26.89	-26.96	53.875

Tabulka 11: Výsledky Lighthouse výkonnostního měření pro Prezentační stránky, zdroj: vlastní.

Audit	Nuxt 3	Quasar.js	Vue.js	Průměr
FCP Q1 (ms)	2662.8 (5.2%)	2615.4 (3.3%)	2316.0 (-8.5%)	2531.4
FCP Q2 (ms)	2957.9 (11.1%)	2633.1 (-1.1%)	2394.5 (-10.0%)	2661.8
FCP Q3 (ms)	3195.0 (11.7%)	2648.6 (-7.4%)	2739.7 (-4.2%)	2861.1
LCP Q1 (ms)	3714.6 (-12.2%)	4948.2 (16.9%)	4036.2 (-4.6%)	4233.0
LCP Q2 (ms)	4149.3 (-8.1%)	5215.4 (15.5%)	4182.5 (-7.4%)	4515.7
LCP Q3 (ms)	4412.1 (-7.3%)	5425.4 (14.0%)	4437.4 (-6.7%)	4758.3
FMP Q1 (ms)	2893.1 (7.4%)	2628.8 (-2.4%)	2557.5 (-5.0%)	2693.1
FMP Q2 (ms)	3189.7 (12.8%)	2646.0 (-6.5%)	2650.1 (-6.3%)	2828.6
FMP Q3 (ms)	3284.0 (10.2%)	2663.2 (-10.6%)	2990.5 (0.4%)	2979.2
SI Q1 (ms)	2663.0 (-0.5%)	2615.4 (-2.2%)	2748.5 (2.7%)	2675.6
SI Q2 (ms)	2959.0 (5.0%)	2633.1 (-6.6%)	2864.1 (1.6%)	2818.7
SI Q3 (ms)	3195.0 (7.6%)	2648.6 (-10.8%)	3067.5 (3.3%)	2970.4
TBT Q1 (ms)	8.9 (-74.6%)	53.4 (52.0%)	43.1 (22.6%)	35.1
TBT Q2 (ms)	10.8 (-74.3%)	69.6 (65.9%)	45.5 (8.4%)	41.9
TBT Q3 (ms)	13.5 (-72.8%)	86.3 (73.8%)	49.1 (-1.0%)	49.7
MPF Q1 (ms)	63.5 (-30.3%)	123.4 (35.4%)	86.5 (-5.1%)	91.2
MPF Q2 (ms)	65.3 (-34.9%)	137.3 (37.0%)	98.1 (-2.1%)	100.2
MPF Q3 (ms)	69.2 (-35.1%)	148.3 (39.2%)	102.1 (-4.1%)	106.5
CLS Q1 (-)	0.0252 (35.5%)	0.0155 (-16.7%)	0.0151 (-18.8%)	0.0186
CLS Q2 (-)	0.0252 (35.4%)	0.0155 (-16.8%)	0.0151 (-18.6%)	0.0186
CLS Q3 (-)	0.0252 (34.6%)	0.0158 (-15.6%)	0.0151 (-19.1%)	0.0187
SRT Q1 (ms)	17.9 (22.1%)	24.4 (66.8%)	1.6 (-88.9%)	14.7
SRT Q2 (ms)	18.4 (8.6%)	30.6 (80.8%)	1.8 (-89.4%)	16.9
SRT Q3 (ms)	19.9 (-9.6%)	44.1 (100.5%)	2.0 (-90.8%)	22.0

I Q1 (ms)	3758.5 (-0.4%)	3756.2 (-0.5%)	3807.2 (0.9%)	3774.0
I Q2 (ms)	3911.5 (1.6%)	3797.6 (-1.3%)	3837.4 (-0.3%)	3848.8
I Q3 (ms)	3942.2 (1.3%)	3811.8 (-2.0%)	3916.5 (0.7%)	3890.2
TBW Q1 (byte)	593997.1 (1.4%)	667899.8 (14.0%)	495785.8 (-15.4%)	585894.2
TBW Q2 (byte)	593997.1 (1.4%)	667899.8 (14.0%)	495785.8 (-15.4%)	585894.2
TBW Q3 (byte)	593997.1 (1.4%)	667899.8 (14.0%)	495785.8 (-15.4%)	585894.2
NR Q1 (-)	26.7 (8.8%)	25.7 (4.8%)	21.2 (-13.6%)	24.5
NR Q2 (-)	26.7 (8.8%)	25.7 (4.8%)	21.2 (-13.6%)	24.5
NR Q3 (-)	26.7 (8.9%)	25.7 (4.7%)	21.2 (-13.6%)	24.5
MWB Q1 (ms)	1010.4 (10.3%)	875.3 (-4.4%)	862.3 (-5.9%)	916.0
MWB Q2 (ms)	1022.9 (10.1%)	890.5 (-4.2%)	874.7 (-5.9%)	929.4
MWB Q3 (ms)	1038.4 (9.7%)	906.6 (-4.2%)	894.2 (-5.5%)	946.4
Celkem Q1 (%)	-27.3	166.9	-139.7	0.0
Celkem Q2 (%)	-22.5	181.5	-159.0	0.0
Celkem Q3 (%)	-39.3	195.6	-156.2	0.0
Celkem (%)	-89.1	544.0	-454.9	0.0

Tabulka 12: Výsledky úplného výkonnostního měření pro Sociální síť, zdroj: vlastní.

Audit	Nuxt 3	Quasar.js	Vue.js
FCP Q3 (ms)	11.7% * 0.1	-7.4% * 0.1	-4.2% * 0.1
LCP Q3 (ms)	-7.3% * 0.25	14.0% * 0.25	-6.7% * 0.25
TBT Q3 (ms)	-72.8% * 0.3	73.8% * 0.3	-1.0% * 0.3
CLS Q3 (-)	34.6% * 0.25	-15.6% * 0.25	-19.1% * 0.25
SI Q3 (ms)	7.6% * 0.1	-10.8% * 0.1	3.3% * 0.1
Celkem Q3	-13.085	19.92	-6.84

Tabulka 13: Výsledky Lighthouse výkonnostního měření pro Sociální síť, zdroj: vlastní.

16 Zadání práce z IS (eVŠKP)

Zadání bakalářské práce

Autor: **Jiří Žák**

Studium: I2100309

Studijní program: B1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Název bakalářské práce: **Možnosti využití frameworků pro tvorbu UA (Universal App) v modelových situacích**

Název bakalářské práce AJ: Exploring the Applications of Universal App (UA) Frameworks in Model Scenarios

Cíl, metody, literatura, předpoklady:

Cílem práce je vysvětlit základní principy architektur webových aplikací jako SPA (Single Page Application), MPA (Multi Page Application) a UA (Universal App).

Představit "vykreslovací" techniky v rámci architektur (CSR, SSR, SSG, ...), představit jeden z frameworků pro tvorbu SPA, konkrétně Vue.js, a také frameworky pro tvorbu UA vycházející z Vue.js.

Cílem praktické části je demonstrovat a následně porovnat využitelnost daných frameworků v modelových situacích.

Osnova:

1. Úvod

2. Cíl a metodika práce

3. Architektury webových stránek

4. Vykreslení webové stránky

5. Představení frameworků

6. Vývoj demonstračních SPA aplikací

7. Implementace On-Demand ISR

8. Migrace aplikací na UA

9. Měření webových stránek

10. Porovnání v měřitelné hladině

11. Porovnání v neměřitelné hladině

12. Závěr

Yi, Y., & Li, Z. (2021). Design and Implementation of Music Web Application based on Vue and Spring Boot. New York: Association for Computing Machinery.

Naylor, S. A. (13. 4 2023). *The Acronyms of Rendering on the Web*. Retrieved 23. 1 2024, from Netlify. https://www.netlify.com/blog/the-acronyms-of-rendering/

Osmani, A., & Miller, J. (2. 6 2019). *Rendering on the Web*. Retrieved 24. 1 2024, from web.dev. https://web.dev/articles/rendering-on-the-web

Josef, J., & Malý, F. (7. 9 2016). *Principy a vývoj isomorfních webových aplikací*. https://doi.org/http://evskp.uhk.cz/eM4589

AirbnbEng. (11. 11 2013). *Isomorphic JavaScript: The Future of Web Apps*. Retrieved 20. 1 2024, from Medium. https://medium.com/airbnb-engineering/isomorphic-javascript-the-future-of-web-apps-10882b7a2ebc#.4nyzv6jea

Cocca, G. (6. 3 2023). *Rendering Patterns for Web Apps – Server-Side, Client-Side, and SSG Explained*. Retrieved 23. 1 2024, from freeCodeCamp. https://www.freecodecamp.org/news/rendering-patterns/#static-websites

Zadávající pracoviště: Katedra informačních technologií, Fakulta informatiky a managementu

Vedoucí práce: Mgr. Hana Rohrová

Datum zadání závěrečné práce: 15.10.2021