SŠPU STŘEDNÍ ŠKOLA PRŮMYSLOVÁ A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Effio - webová aplikace pro vytváření testů

Autor: Matěj Kotrba

Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování

Třída: IT4

Školní rok: 2023/24

Poděkování

Rád bych poděkoval Mgr. Markovi Lučnému za poskytuntí konzultace ohledně tohoto projektu.

Prohlášení

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým a prezentačním účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 1. 1. 2024

. . . . . .

Podpis autora

Anotace

Výsledkem projektu je funkční webová aplikace pro vytváření a vyplňování testů, které se skládají z různých možností otázek včetně programovací. Aplikace zahrnuje přihlášení přes Google a GitHub. Uživatel vytváří jednotlivé testy výběrem šablony, nebo importem z GIFT formátu, otázek, komentářů a následně upravuje detaily testu jako jméno, popis, obrázek, upravitelný známkovací systém, zařazení do skupin a tagy. Hotový test může sám zkusit z vlastní kolekce testů a nebo z komunitního centra, kde se nacházejí komunitou vytvořené testy, po vyplnění testu se uživateli objeví výsledky a známka. Krom výroby a vyplňování testů aplikace obsahuje také skupiny, kde mohou mezi sebou uživatelé komunikovat pomocí chatu a majitel do ní může sdílet testy, na dříve vyplněné testy se může podívat v sekci testové historie. Přehled o aktivitě uživatele si může prohlédnout v dashboardu prostřednictvím vizuálních grafů. Dříve vytvořené testy se dají v části kolekce editovat, mazat a také exportovat do GIFT formátu v textovém souboru pro použití například v Moodlu. Aplikace disponuje zcela responsivním designem se světlým a tmavým režimem.

Klíčová slova

webová aplikace, databáze, responsivní design, účty, grafy, tvorba testů, barevné režimy

Obsah

Úvod

V dnešní době se běžně využívají webové aplikace, které umožňují vytváření testů/kvízů, které poté jiní uživatelé vyplňují. Prostředí těchto aplikací jsou však často nepřehledné a vytváření testů či kvízů je úporné. S touto myšlenkou jsem se rozhodl vytvořit aplikaci, která by kombinovala možnosti jiných aplikací s přehledným moderním zobrazením a dalšími užitečnými prvky.

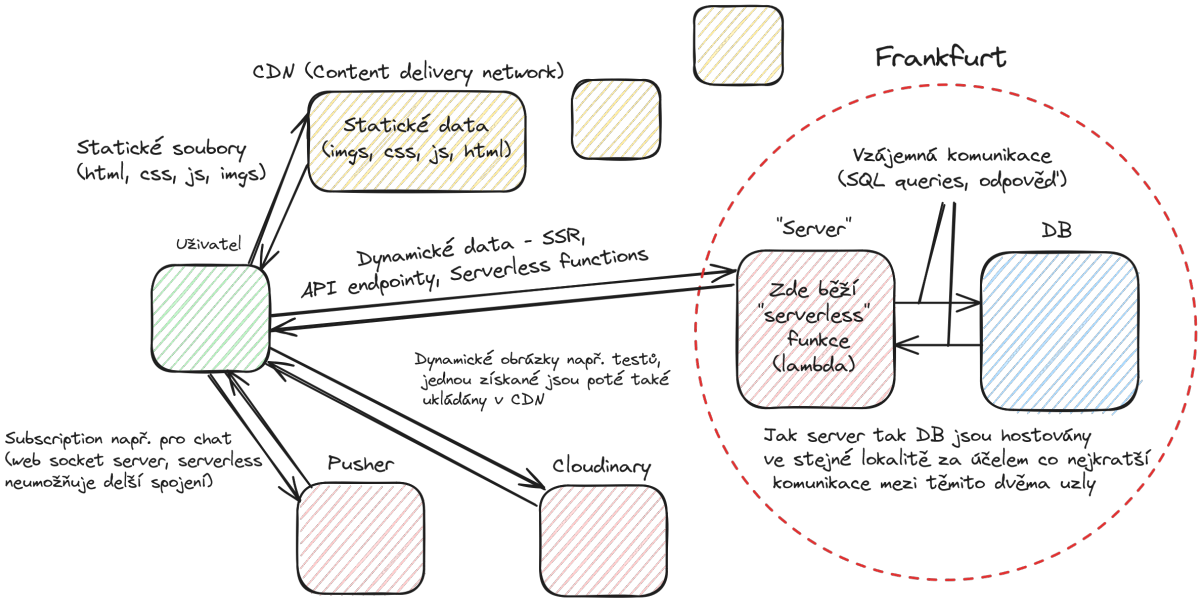
Má aplikace by krom již zmíněné funkcionality pro tvorbu testů a kvízů měla do jisté míry umožňovat prvky sociálních sítí jako třeba skupiny, komunitní místo kde by se mimo jiné zobrazovaly testy ostatních uživatelů. Hlavní myšlenkou bylo vytvořit nejen aplikaci jako takovou ale také využít moderní technologie a postupy, neboli vytvořit ji „typesafe“, bez potřeby vlastního serveru za pomocí cloudové technologie „serverless“ a plně responzivní pro uživatele na kterémkoli zařízení.

V dokumentaci jsou popsané využité technologie, postupy a jednotlivé funkcionality celé aplikace. První část popisuje architekturu a přístup k řešení, ve druhé části popisuji využité technologie a jejich účel v aplikaci, ve třetí části se potom zmiňuji o různých možnostech, které Effio nabízí. Nakonec se poohlížím na dosažené cíle a možné vylepšení.

1 Architektura a koncepty aplikace

1.1 Architektura

Vzhledem k tomu, že záměrem bylo vytvořit aplikaci, která bude pro veškeré „backendové“ úlohy využívat cloud, tak je architektura relativně složité téma, proto se pokusím stručně popsat základní proces vykreslení stránky. Uživatel získává statické soubory ze „CDN“, dynamicky vytvořené stránky pro první zobrazení („first render“) jsou generovány na instancích „serverless funkcí“ třetí strany, každá instance se vytváří podle potřeby na reálném serveru poskytovatele, není dedikovaná a ani nezůstává aktivní po delší dobu. Server je dále využíván pro potřeby klienta, které si on sám nemůže obstarat především z bezpečnostních důvodů (DB queries, secret env variables...). Databáze vytváří spojení se serverem a ten získává potřebná data, viz. obrázek dole 1.1.



Obrázek 1.1: Jednoduchý přehled architektury Effia.

1.1.1 Koncept „single page application“ a tradičního web serveru

V minulé sekci jsem zmínil pojem web server, ten generuje stránku, která je poslána uživateli, ta se poté generuje znovu vždy když je uživatel přesměrován na jinout stránku, tento koncept má spoustu nevýhod jako například nemožnost jednoduše uchovávat data mezi těmito přechody. „Single page application“ je vlastně JavaScript framework, který se o překreslování obrazu a přesměrování stará sám za pomocí JavaScriptu bez nutnosti dotazovat se serveru o novou stránku či data. Tento přístup bohužel také není perfektní, nutnost stahovat veškerý kód nebo velice chabý SEO rating.

1.1.2 Metaframework

Oba zmíněné koncepty přinášejí své výhody ale s ním také nevýhody, Metaframework je technologie, která kombinuje výhody obou konceptů a přináší za mě nejlepší variantu jak vytvářet webovout aplikaci. Disponuje možností běhu kódu na serveru, přesměrováním na klientské části a dalšími výhodami. Takových technologií existuje celá řada, jako například populární NextJS, já si pro svůj projekt zvolil SvelteKit.

1.1.3 Cloud hosting

Jak již bylo zmíněno v úvodu tak tato aplikace by se měla obejít bez vlastního serveru, nejde ale jenom o databázi ale také například o ukládání obrázků nebo hostování aplikace jako takové, to je prováděno přes „Cloud hostingy“ jako Vercel pro hostování stránky jako takové, zároveň ale řeší i rozesílání statických dat do CDN a poskytuje serverless lambda funkce, Planetscale pro hostování mojí MySQL databáze, Cloudinary, který slouží jako „bucket“ pro obrázky a také jejich možnost editace přes url parametry, nebo Pusher, který slouží jako web socket server například pro chat.

1.2 TYPESAFETY

Webové aplikace standartně využívají JavaScript, ten ale obnáší signifikantní nevýhodu v podobě nemožnosti „otypovat“ kód, to způsobuje obtíž orientovat se v kódu, velké množství produkčních chyb a také spoustu času stráveného pochopením dříve napsaného kódu. Pro Effio jsem se tedy rozhodl využít moderní technologie a vytvořit tak téměř plně „typesafe“ (otypovanou) aplikaci. TypeScript v Effiu nahrazuje JavaScript, ten do tohoto jazyka přináší typy. To ale nestačí, API endpointy, stejně jako databázové dotazy stále nemůžou být otypované a proto jsem připojil také knihovny tRPC a Prisma.

2 VYUŽITÉ TECHNOLOGIE

2.1 SVELTEKIT

Svelte je open source JavaScriptový framework vyvíjený od roku týmem Riche Harrise, jeho hlavní výhodou je rychlost ale také intuitivita, protože jazyk se snaží vypadat jako JavaScript, zatímco rozšiřuje jeho možnosti, díky tomu se za posledních let těší rostoucí popularitě webobých vývojářů. Na rozdíl od ostatních webových frameworků (například React, Angular, Solid nebo Qwik) Svelte disponuje vlastním jazykem, který se zapisuje do `.svelte` souboru, ten se následně kompiluje do vysoce efektivního JavaScriptu.

SvelteKit je metaframework postavený na Svelte. Jeho hlavní výhody se skládají z:

- Rychlost - Svelte vytváří velice rychlou aplikaci, v kombinaci s Vitem (bundler) se ale také spojuje s velice rychlým build timem a fast refreshy. To ale není jediné místo kde se rychlost projevuje, za pomocí SvelteKitu se aplikace vyvíjí velmi rychle díky velice malému množství „boilerplate“ kódu.
- Flexibilita - Aplikace často potřebuje různé typy vykreslování stránek, SvelteKit dovoluje jednoduše nakonfigurovat jednotlivé stránky či cesty pro specifické způsoby jako SPA, SSR, SSG nebo MPA.
- Přehlednost - SvelteKit využívá „file based routing“, tedy cesty aplikace jsou generovány podle složek, které vývojář vytvoří, soubory se poté vždy jmenují stejně, `+page.svelte` pro stránku, `+layout.svelte` pro layout apod. Díky tomu je vždy jasné pro co specifický souboru slouží, přehlednosti také přidává, již u Sveltu zmíněná podobnost s JavaScriptem.

2.2 TYPESCRIPT

Typescript se dá považovat jako nadstavba JavaScriptu, poskytuje ale jednu výraznou výhodu - typy, díky nim je možno mnohem snadněji dohledávat chyby, vracet se k dříve napsanému kódu a celkově mnohem zlepší „developer experience“ při vytváření aplikace. Sám o sobě pomůže s otypováním jednotlivých částí kódu, neporadí si však například s API endpointy nebo databázovými dotazy, které se poté musí otypovat ručně, což je ale velice špatný způsob.

2.3 tRPC

V minulé sekci jsem se zmínil o problémech s otypováním API endpointů, tRPC (Typescript Remote Procedure Call) tento problém řeší tím, že vytváří dynamické typy pro jednotlivé endpointy, podle toho jak si je sami nadefinujeme.

2.4 Prisma

Prisma slouží jako ORM (Object–relational mapping), to znamená pomocí JavaScriptu získávat data z databáze bez přímého použití jazyka SQL, Prisma se skládá z klientské části, která pomocí protokolu založeném na JSON komunikuje se serverovou částí, tam je následně uskutečněn SQL dotaz a odpověď je poslána klientovi. Také řeší již zmíněný problém s otypováním těchto dotazů, pro Prismu je totiž nutné vytvořit `schema.prisma` soubor kde se definuje model databáze, ten poté můžeme pomocí Prisma CLI nahrávat do databáze ale také vytvářet dynamické typy, které poté využijeme jak v dotazech tak v aplikaci pro data, která dostaneme zpět.

2.5 Zod

Zod je validační knihovna jako např. Yup. Jeho výhodou je avšak možnost využít jeho validační schémata jako typy a také samotná validace funguje jako „type guard“ (kontroluje a nastavuje typy u vložené proměnné). Jeho další výhodou je například jeho nízká velikost.

2.6 Tailwind CSS

Tailwind CSS je „CSS utility library“, to znamená, že narozdíl od frameworků jako je třeba Bootstrap nebo Material UI neposkytuje celé předpřipravené komponenty ale připravené CSS classy, které se aplikují na HTML elementy, jeho výhodou je naprostá kontrola nad chováním stylů, které se aplikují, přehlednost a rychlost se kterou se dá styly vytvářet.

2.7 Auth.js

Auth.js je knihovna sloužící pro autentifikaci a autorizaci, poskytuje možnost „session based“, to je použito v Effiu, a JWT autentifikace. Dále knihovna podporuje OAuth s mnohými providery, v tomto projektu je využit GitHub a Google s jednoduchou možností přidat další. Výhodou knihovny je, že data si vývojář spravuje sám, neboli jsou ukládána do jeho vlastní databáze v

podobě tabulek (které si také může sám upravit): Account, Session, User a Verification Token, které poskytují naprostou kontrolu nad ověřením uživatelů.

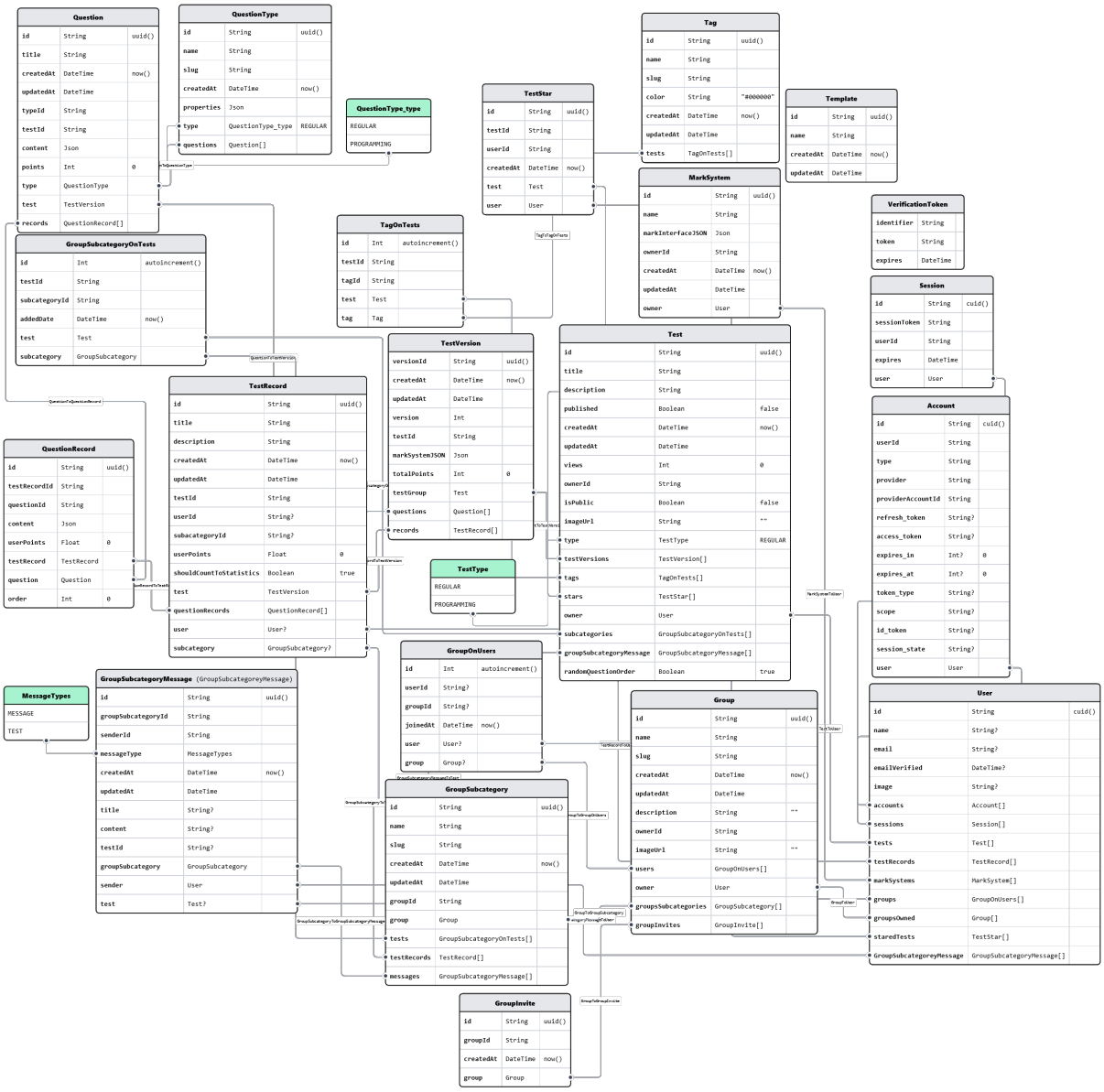
3 Kroky k řešení, funkcionality aplikace a implementace

3.1 Založení a konfigurace projektu

Prvním krokem bylo založení projektu a stažení potřebných knihoven technologií, které jsem plánoval využít. Kombinace mnou vybraných technologií nebyla kompletně konvenční a proto jsem se musel v některých případech obrátit na komunitou vytvořené adaptéry, příkladem je například knihovna `trpc-sveltekit`, která propojuje SvelteKit a tRPC, které je primárně navrženo buď jako samostatný server a nebo jako implementace do Next.js.

3.2 MODEL DATABÁZE

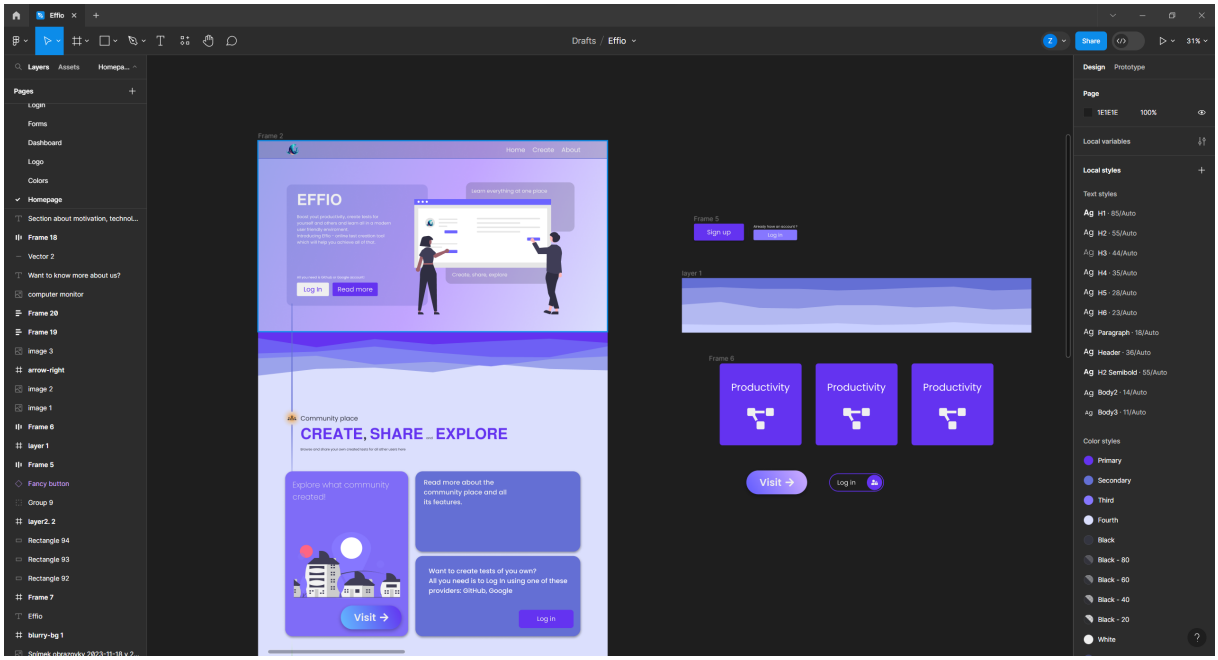
Dalším krokem bylo vytvořit databázový model, vystavený obrázek nezachycuje model počáteční ale ten, ke kterému postupem práce Effio dospělo přidáváním nových funkcionalit, model byl také několikrát částečně přepracován.



Obrázek 3.1: Databázový model Effia. Vytvořeno pomocí Prismaliser

3.3 Design

Jednou z hlavních myšlenek bylo vytvořit pohledem přívětivou aplikaci, proto se návrh designu stál klíčovou částí pro stylově propracovanější prvky stránky. Pro tvorbu designu, stejně jako vytváření a úpravu potřebných obrázků jsem využil aplikaci Figma.



Obrázek 3.2: Prvotní návrh domovské stránky.

3.4 Testy a jejich vlastnosti

Jednou z esenciálních funkcionalit Effia je možnost vytvořit test, k tomuto existuje mnoho různých postupů, nad danou problematikou jsem se zprvu zamýšlel a až poté napsal funkční nástroj pro jejich vytváření ale to stejně nakonec nezabránilo následné nutnosti přepsat téměř celou funkcionalitu při vytváření testu.

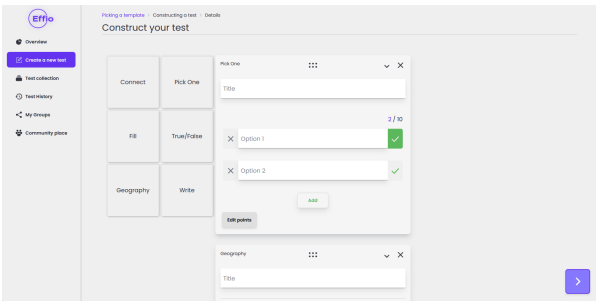
3.4.1 Tvorba testu

Jako první si uživatel zvolí mezi kvízovým a programovacím testem, u obou si poté vybere šablonu.

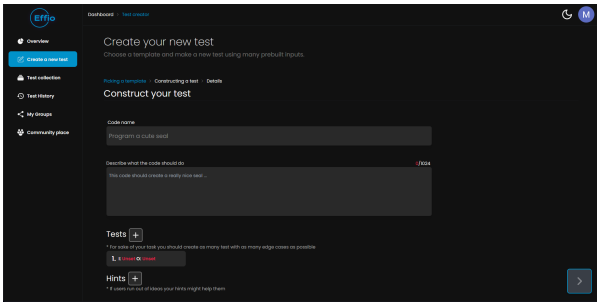
- Kvízový - po výběru šablony, kde si uživatel může zvolit i import z GIFT formátu, se uživatel dostane do tvorby testu samotného, vybírat si aktuálně může z 6 typů otázek: *Pick One*, *True/False*, *Connect*, *Write*, *Fill* a *Geography*, otázkám lze svévolně měnit pořadí, přidávat komentáře k odpovědím a upravovat počet získaných bodů.

- Programovací - po výběru šablony se uživatel dostane do tvorby testu programovacího, kde ho pojmenuje problém, popíše co má uživatel řešit, nadefinuje kontrolní vstupy a očekávané výstupy, poté může zanechat nápovědy

Po dokončení těchto úprav se uživatel dostane do konečných úprav testu, což činí jméno, popisek a obrázek testu, volitelné zařazení do skupin, tagy, rozhodne se jestli využít známkovací systém, který si může sám upravit, zvolí si zdali náhodně třídit otázky a následně tvorbu ukončí a rozhodne se zdali test uložit jako návrh nebo ho publikovat.



Obrázek 3.3: Kvízový test



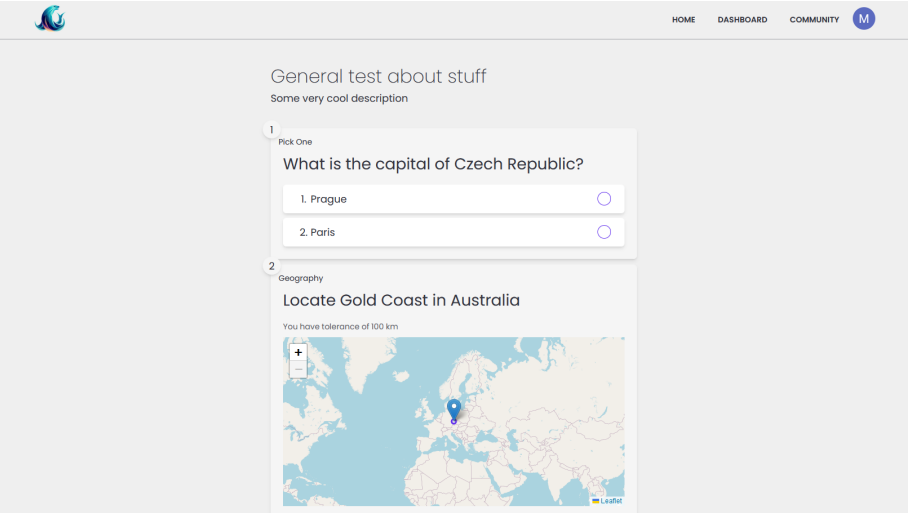
Obrázek 3.4: Programovací test

3.5 Vyplňování testu

3.5.1 Vyplňování kvízu

Vytvořený test si poté může kdokoliv s přístupem k němu vyplnit (testy jsou základně dostupně pro všechny, po úpravě mohou být zveřejněny pouze pro členy skupin).

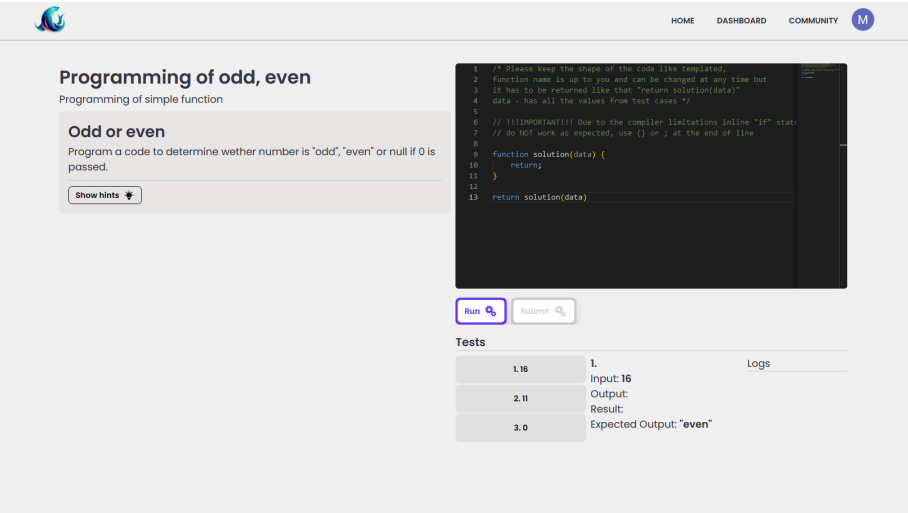
Otázky se náhodně seřadí a uživatel je vyplňuje, zamíchané jsou také odpovědi určitých typů otázek. Po vyplnění všech uživatel test odevzdá, zkontroluje se a vrátí mu správné odpovědi, počet bodů, známku co získal a pokud je uživatel přihlášený tak se záznam a vyplnění uloží do databáze, uživatel si ho poté může zpětně zobrazit v sekci *Test history*.



Obrázek 3.5: Obrázek kvízu.

3.5.2 Plnění programovacího testu

Uživatel dostane popis toho co by měl kód umět, sadu testů, které mají otestovat funkcionalitu kódu a popřípadě nějaké nápovědy. Programovací test obsahuje vlastní editor do kterého uživatel píše, pro kontrolu testu můžeme použít tlačítko *„Run“* a pokud testy prochází tak je test možné řešení odevzdat.

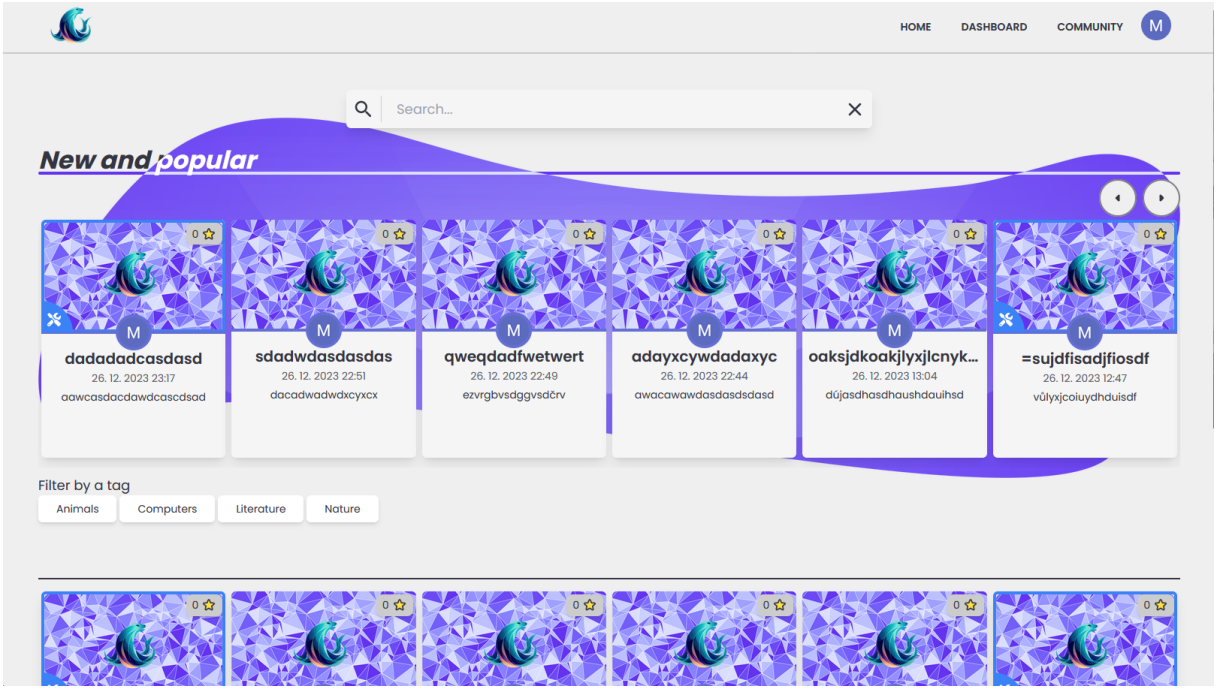


Obrázek 3.6: Obrázek programovací úlohy.

3.6 Zobrazování testů

3.6.1 Komunitní místo

Na této stránce může uživatel najít nové a populární testy včetně všech testů, které existují. Testy jsou zobrazovány postupně podle principu „infinite scrolling“, na což využívám JavaScript API Intersection Observer abych zjistil kdy uživatel dosáhl posledního prvku a vyžádal jsem si tak další, implementované je také vyhledávací pole, které filtruje zobrazované testy, další možností je filtrace pomocí tagů. Vizuálně jsou také rozlišeny kvízy od programovacích testů. Testy se také dají ohodnotit hvězdičkou, jejich aplikování využívá principu „optimistic update“, to znamená, že po přidání hvězdičky ji uživatel okamžitě vidí přidanou zatímco se ověřuje jeho oprávnění a vytváří záznam hvězdičky v databázi, v případě neúspěchu se poté uživateli sama hvězdička opět odebere.



Obrázek 3.7: Komunitní místo.

3.6.2 Kolekce testů

Zde si uživatel může zobrazit jím vytvořené testy, aplikovaná je stejná funkcionalita vyhledávacího pole a „infinite scrollingu“. Každý test má ale také další možnosti, a to úpravu, export a smazání.

- Úprava - uživatel se přesune na stránku úprav, tam může celý test přepracovat.
- Export - vytvoří z testu textový soubor ve formátu GIFT se všemi otázkami daného testu, které jsou podporovány Moodlem
- Delete - smazání testu z databáze

3.7 Skupiny

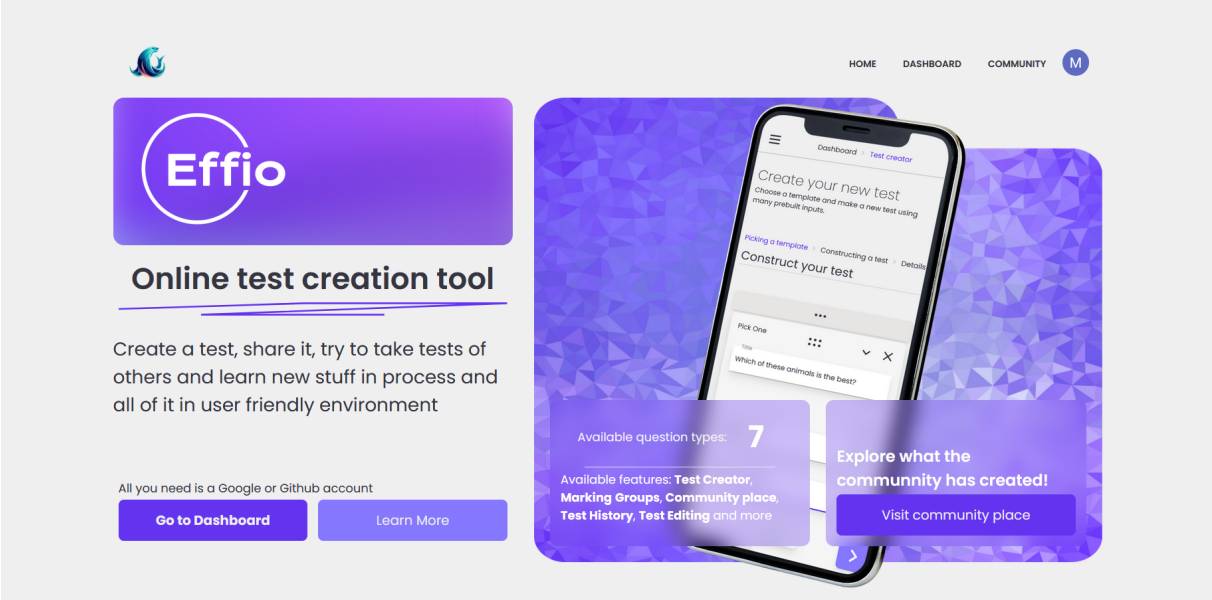
Každý přihlášený uživatel si může vytvořit vlastní skupinu, do které se můžou pomocí generovaného kódu připojit ostatní uživatelé. Skupina obsahuje kanály, ve kterých je možné psát textové zprávy, taky zde můžeme nají přidané testy, vlastník si poté může procházet grafy výsledku členů skupiny.

3.8 Světlý a tmavý režim

S ohledem na uživatele co preferují tmavý režim jsem se také rozhodl pro tvorbu tmavého režimu, ten je možné vidět na obrázku 3.4, oba tyto režimy vyžadovali vlastní paletu barev a byly mnohokrát přepracovány aby k sobě jednotlivé barvy co nejlépe pasovaly.

3.9 Domovská stránka

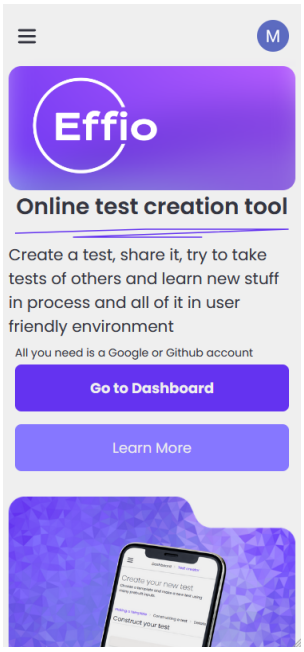
Domovská stránka slouží jako místo pro seznámení návštěvníka s výhodami Effia, rychlá navigace mezi jednotlivými stránkami ale také jako místo nejpečlivěji vytvářeného designu a efektů aby na uživateli zanechala dojem kvalitní aplikace.



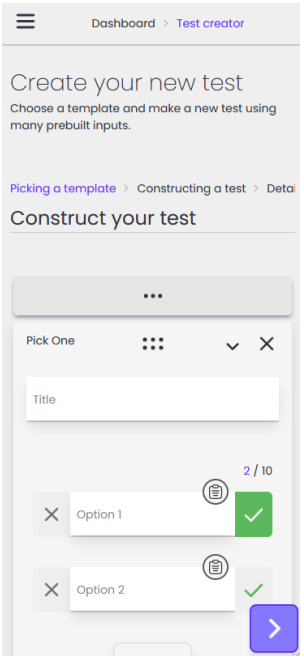
Obrázek 3.8: Domovská stránka Effia.

3.10 Responsivita

Celá aplikace je uzpůsobená jak pro počítače tak pro mobilní zařízení. Responsivita není vůbec lehká práce, mně ale pomohl Tailwind, ve kterém se CSS media query dělají snadněji společně s moderními CSS containers, které umožňují responsivní breakpointy odvozovat ne jen od velikosti stránky ale od rozměrů rodičovských elementů.



Obrázek 3.9: Domovská stránka na mobilním zařízení



Obrázek 3.10: Generátor testů na mobilním zařízení

4 VÝSLEDKY PRÁCE

4.1 FUNKCE APLIKACE

Po příchodu na stránku se uživateli zobrazí domovská stránka, zde může prozkoumat výhody aplikace nebo se přihlásit, to ho přesune na přihlašovací stránku, na ni se uživatel do aplikace může přihlásit pomocí Google nebo GitHub účtu. Po přihlášení je uživatel přesunut na dashboard, kde může vidět rychlou navigaci na další části nebo se podívat na souhrn jeho aktivity v grafech. První možností je vytvořit si nový test, tato možnost je detailně probraná zde 3.4.1. Po vytvoření testu je přesunut do kolekce jeho testů, kde může své testy procházet a upravovat, více popsáno zde 3.6.2. Test si poté může uživatel zkusit vyplnit 3.5, buď přímo ze své kolekce nebo z komunitního centra 3.6.1, kde může najít také testy jiných. Po dokončení testu se dozví výsledky, ty může poté najít zpětně v sekci *Test history*, kde jsou uspořádané v tabulce.

4.2 SPLNĚNÉ A NESPLNĚNÉ CÍLE

Cíle byly rozdělené jak do rozsahu aplikace tak do kvality implementace jednotlivých prvků, co se týče rozsahu tak ten jsem v určitých místech významně předčil, ne všechny body původních cílů jsou ale aktuálně plně dosaženy.

Hlavním cílem bylo vytvořit rychlou, cloudovou aplikaci pro vytváření a sdílení testů využívající moderních „techstack“, v tomto ohledu jsem cíl kompletně splnil, aplikace je v těchto bodech plně funkční a můj původní plán využití technologií se během vývoje ještě významně rozrostl.

Za úspěch považuji také grafickou část aplikace, která za mě tvoří minimalistický moderní vzhled.

Hlavní neúspěch nebo spíše nedodělanost vidím ve skupinách a přizpůsobení možností testů pro ně, původně jsem zamýšlel vytvořit skupiny jako místo pro sdílení materiálů s více zajímavými možnostmi pro vlastníka, aby sloužili jako funkce vhodná pro výuku. Cíle nejsou zdaleka nerealistické ale těchto cílů jsem nebyl schopen dosáhnou z nedostatku času, jako vylepšení by ale za mně bylo velice přínosné.

ZÁVĚR

Cílem projektu bylo vytvořit webovou aplikaci pro vytváření a vyplňování testů. Aplikace je postavená na frameworku SvelteKit, psaný v jazyce TypeScript. Přihlašování stojí na knihovně Auth.js, MySQL databáze je hostovaná službou Planetscale, jako ORM je použita Prisma. Frontend řeší framework Svelte s CSS utility knihovnou Tailwind, pro validaci využívá Zod.

Základem aplikace je generátor testů, kde uživatel využívá předpřipravené typy otázek. Testy se následně dají vyplnit v rámci komunity nebo vlastníkovi kolekce. Nepřihlášený uživatel může testy pouze vyplňovat, přihlásit se uživatel může pomocí Google nebo GitHub účtu. Uživatelé také mají k dispozici skupiny kde je chat a také možnosti sdílet testy, test historii k zobrazení dříve vyplněných testů a přehled nedávné aktivity v podobě grafů. Využité technologie činí aplikaci lehce škálovatelnou a velice výkonnou. Aplikace je téměř zcela funkční a použitelná na všech zařízeních díky responzivnímu designu.

Aplikace je zálohovaná na GitHubu na adrese `https://github.com/matej-kotrba/effio`. Je také volně přístupná na adrese `https://effio.vercel.app/`

LITERATURA

[1] Svelte [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://svelte.dev/

[2] SvelteKit [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://kit.svelte.dev/

[3] tRPC [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://trpc.io/

[4] Prisma [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://www.prisma.io/

[5] Zod [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://zod.dev/

[6] Auth.js [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://authjs.dev/

[7] Tailwind CSS [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://tailwindcss.com/

[8] tRPC-SvelteKit [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://icflorescu.github.io/trpc-sveltekit/

[9] ChatGPT [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://chat.openai.com/

[10] Stackoverflow [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://stackoverflow.com/

[11] Joy of Code. Youtube kanál. Https://www.youtube.com/ [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://www.youtube.com/@JoyofCodeDev

[12] Huntabyte. Youtube kanál. Https://www.youtube.com/ [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://www.youtube.com/@Huntabyte

[13] BROWNE, Theo. Youtube kanál. Https://www.youtube.com/ [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://www.youtube.com/@t3dotgg

[14] POWELL, Kevin. Youtube kanál. Https://www.youtube.com/ [online]. 2023 [cit. 2023-12-27]. Dostupné z: https://www.youtube.com/@KevinPowell

Seznam obrázků