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

Abstrakt

Sem napíšeš svůj abstrakt.

Slouží jako pomoc čtenáři rychle se zorientovat v dané práci.

“Redukovaný text, který charakterizuje obsah dokumentu bez rozlišování autorství abstraktu, bez doplňkových informací, bez vlastní interpretace a hodnocení dokumentu (tj. nikoliv "v práci velmi dobře hodnotím podle mne zajímavý systém...", ale "práce hodnotí systém..."). Základními vlastnostmi anotace jsou výstižnost, přehlednost, jasnost, stručnost, přesnost, objektivnost a čtivost. Anotace je formulována v přirozeném jazyce – obvykle ve větách. Anotace může používat textových formulací z referovaného dokumentu, ale jako celek je formulován nově.“

Délka cca 100 – 250 slov

Klíčová slova

Šablona, LaTeX, závěrečná práce, dokumentace, …

Abstract

Write your abstract here! Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Keywords

Template, LaTeX, High school proffessional activity, …

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.

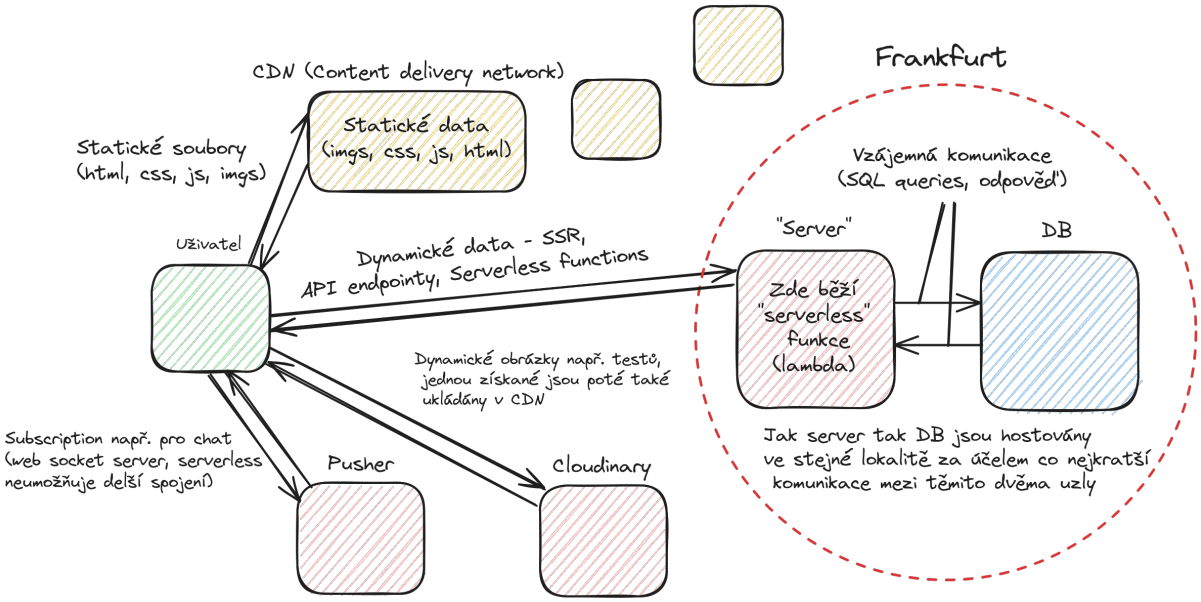
Má aplikace by krom již zmíněné funkcionality pro tvorbu testů a kvízů měla do jisté míry umožňovat prvky sociálních sítí jako třeba skupiny, komunitní místo kde by se mimo jiné zobrazovaly testy ostatních uživatelů. Hlavní myšlenkou bylo vytvořit nejen aplikaci jako takovou ale také využít moderní technologie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

Možná tady doplnit co je v dokumentaci

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, 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.

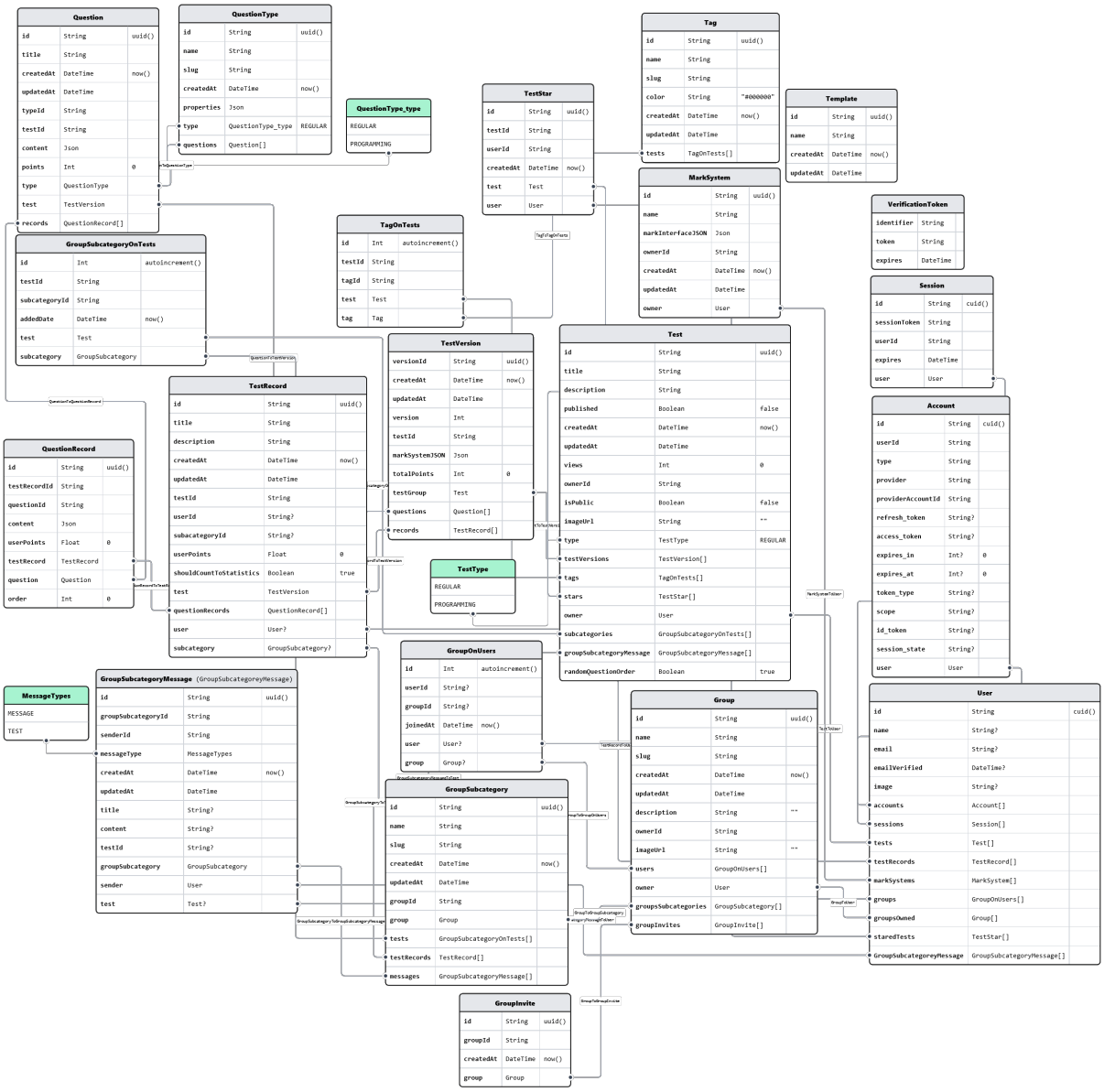
3 Cesta k řešení a její implementace

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

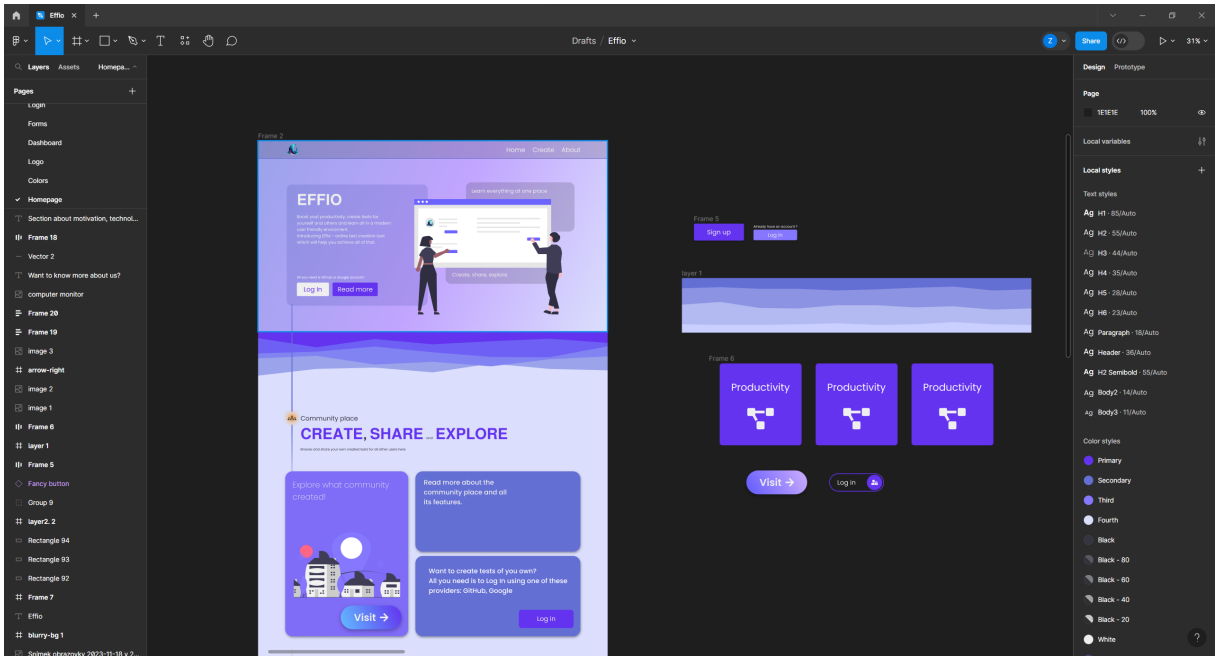
Dalším krokem bylo vytvořit databázový model, vystavený obrázek nezachycuje model počáteční ale ten, ke kterému postupem práce Effio dospělo přidáváním nových funkcionalit, model byl také několikrát částečně přepracován.



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.

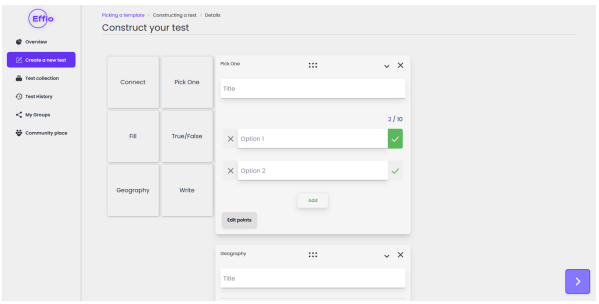
3.4.1 Tvorba testu

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

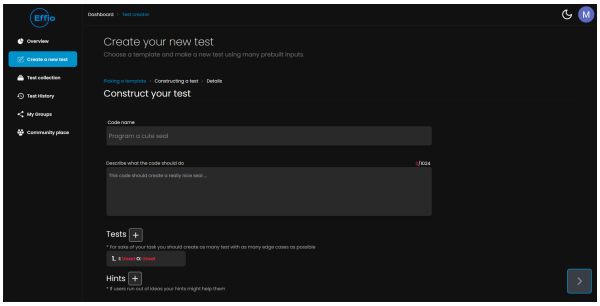
- Kvízový - po výběru šablony, kde si uživatel může zvolit i import z GIFT formátu, se uživatel dostane do tvorby testu samotného, vybírat si aktuálně může z 6 typů otázek: *Pick One*, *True/False*, *Connect*, *Write*, *Fill* a *Geography*, otázkám lze svévolně měnit pořadí, přidávat komentáře k odpovědím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

3.4.2 Vyplňování kvízu

Vytvořený test si poté může kdokoliv s přístupem k němu vyplnit (testy jsou základně dostupné 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*.

3.4.3 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.

3.4.4 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.

3.4.5 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.

3.4.6 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

3.4.7 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

Zvýraznění textu

V LaTeXu existuje několik způsobů, jak zvýraznit text. Můžeme použít tučné písmo, kurzívu nebo podtržení.

```
\textbf{tučné písmo}, \textit{kurzíva}, \underline{podtržený text}
```

Seznamy a výčty

Seznamy jsou užitečné pro strukturování informací a jejich uspořádání do čitelné formy. LaTeX podporuje nečíslované, číslované a popisné seznamy.

```
\begin{itemize}
\item Nečíslovaný seznam
\end{itemize}

\begin{enumerate}
\item Číslovaný seznam
\end{enumerate}

\begin{description}
\item[Popisek] Popisný seznam
\end{description}
```

3.4.8 Křížové odkazy a poznámky pod čarou

Křížové odkazy a poznámky pod čarou jsou důležité pro odkazování na jiné části dokumentu a poskytování dodatečných informací.

Křížové odkazy

Pomocí křížových odkazů můžeme odkazovat na jiné sekce, obrázky nebo tabulky v dokumentu.

```
\label{sec:nazev_sekce}
Odkaz na sekci \ref{sec:nazev_sekce}.
```

Poznámky pod čarou

Poznámky pod čarou poskytují dodatečné informace bez přerušení toku hlavního textu.

```
Text s poznámkou pod čarou.\footnote{Text poznámky pod čarou.}
```

3.5 Matematické vzorce a symboly

Tato část poskytuje přehled o vkládání matematických vzorců a symbolů do dokumentů v LaTeXu, což je nezbytné pro tvorbu akademických a vědeckých textů.

3.5.1 Základní matematické prostředí

LaTeX nabízí několik prostředí pro práci s matematikou, včetně "math" pro základní matematické výrazy a "displaymath" pro samostatné rovnice.

```
$z = x + y$ % Inline matematika
\begin{displaymath}
z = x + y
\end{displaymath}
```

3.5.2 Rovnice a symboly

Matematické rovnice a symboly jsou základem mnoha vědeckých dokumentů, a LaTeX poskytuje širokou škálu nástrojů pro jejich efektivní použití.

Vložení jednoduché rovnice

Pro vložení jednoduché rovnice můžeme použít prostředí "equation" nebo "align" pro více rovnic s zarovnáním.

```
\begin{equation}
E = mc^2
\end{equation}
```

Pokročilé matematické výrazy

Pro složitější matematické výrazy, jako jsou integrály, sumy nebo frakce, LaTeX nabízí rozsáhlé možnosti.

```
\begin{equation}
\int_0^\infty e^{-x} \, dx
\end{equation}
```

3.6 Práce s obrázky a tabulkami

Tato kapitola je zaměřena na vkládání a formátování obrázků a tabulek v LaTeXu, což jsou klíčové dovednosti pro vytváření vizuálně atraktivních a informativních dokumentů.

3.6.1 Vkládání obrázků

Vkládání obrázků do dokumentů LaTeXu umožňuje autorům přidávat vizuální prvky, které podporují a doplňují textový obsah.

Formáty obrázků

LaTeX podporuje různé formáty obrázků, včetně populárních formátů jako JPEG, PNG a PDF. Výběr správného formátu je důležitý pro kvalitu a velikost souboru.

```
\includegraphics[width=0.5\textwidth]{obrazek.jpg}
```

Pozicování obrázků

Správné pozicování obrázků je klíčové pro zachování čitelnosti a estetiky dokumentu. LaTeX nabízí několik možností, jak ovlivnit umístění obrázků v textu.

```
\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{obrazek.jpg}
\caption{Popisek obrázku}
\label{fig:obrazek}
\end{figure}
```

3.6.2 Vytváření tabulek

Tabulky jsou nezbytné pro organizované a efektivní prezentování dat. LaTeX umožňuje vytváření jak jednoduchých, tak složitých tabulek.

Základní tabulky

Pro vytváření základních tabulek lze využít prostředí "tabular". Jednoduchá tabulka může být vytvořena bez složitých formátovacích nástrojů.

```
\begin{tabular}{|c|c|c|}
\hline
A & B & C \\
\hline
1 & 2 & 3 \\
\hline
\end{tabular}
```

Pokročilé tabulky

Pro složitější tabulky, jako jsou tabulky s více řádky nebo sloupci, lze použít pokročilé formátovací možnosti, jako jsou sloučené buňky a speciální zarovnání.

```
\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
\multirow{2}{*}{A} & B1 & C1 \\
\cline{2-3}
& B2 & C2 \\
\hline
\end{tabular}
\caption{Pokročilá tabulka}
\label{tab:pokrocila_tabulka}
\end{table}
```

3.7 BIBLIOGRAFIE A CITACE

Tato kapitola poskytuje podrobný návod na vytváření bibliografie a správné citování zdrojů v LaTeXu, což jsou nezbytné dovednosti pro akademické psaní a publikování.

3.7.1 Vytváření bibliografie

LaTeX umožňuje efektivní správu bibliografických záznamů a jejich automatické formátování. Tento proces zahrnuje několik kroků od definování zdrojů po jejich začlenění do dokumentu.

3.7.2 Citování zdrojů

Správné citování zdrojů je klíčové pro akademickou integritu a umožňuje čtenářům dohledat zmiňované informace. V LaTeXu je možné citovat zdroje jednoduše pomocí příkazu `\cite`.

```
Jak bylo zmíněno v \cite{nazev}, ...
```

4 TIPY K PSANÍ

Jak už jsem psal výše LaTeXje dosti komplexní systém, který umožňuje psát velmi rozsáhlé text. Jeho autor Donald Knuth ho stvořil, aby mohl vydat jeho učebnici *The Art of Computer Programming* a dodnes se je využíván pro sazbu skript, učebnic, článků či závěrečných prací. V této kapitole najdeš ukázky různých funkcí a balíčků LaTeXu od těch nejzákladnějších až po složitější. Neznamená to nutně, že všechny musíš použít, ale když potřebuješ pomoct, tak je dobré mít oporu.

Pokud s LaTeXem úplně začínáš tak ti můžu doporučit přiručku *Ne příliš stručný úvod do systému LaTeX2e* [2]. Případně spoustu užitečných informací nalezneš na Wikibooks [3]. Pokud narazíš na nějaký problém googli. Na internetu je spoustu fór, kde pravděpodobně už někdo podobný problém řešil. Asi nejvíce otho najdeš na stránce *TeX - LaTeX Stackexchange* [4].

4.1 ZÁKLADY: TEXT, OBRÁZKY, TABULKY A CITACE

Psaní v LaTeXu není žádná věda, stačí psát normálně do zdrojového souboru. Pokud bys chtěl psát obrážky či číslovaný seznam, pak můžeš použít prostředí `itemize` či `enumerate`. Často je důležité používat nezlomitelnou mezeru. Tu uděláš pomocí ~ (tildy). Pokud budeš chtít psát uvozovky použij příkaz `uv`, pomocí něj se ti vytvoří uvozovky podle příslušného jazyka. V česku tedy ve formátu 99 66. Použití příkazu najdeš níže v textu.

Občas je zapotřebí LaTeXu pomoct při rozdělování slov. To se udělá snadno vložením symbolů `\-` mezi jednotlivé slabiky.

4.1.1 Tabulky

U tabulek platí to stejné co u obrázků. Zarovnávají se na střed a nechávají se „plavat“ v textu. Tabulka narozdíl od textu, má popisek nahoře. U tabulky 4.1 je použit balíček `booktabs`, pomocí kterého je celá tabulka naformátovaná.

Seznam jak obrázků tak tabulek je pak vytvořen pomocí příkazů `listoftables` a `listoffigures` na konci práce před literaturou.

Tabulka 4.1: Tato tabulka slouží jako ukázka toho, jak mohou tabulky vypadat.

záhlaví	této	tabulky
obsah	tabulky	už
není	oddělený	čarami

4.1.2 Obrázky

U obrázků je dobré používat vektorové formáty, pokud to jde. LaTeXse nejvíc kamarádí s formátem PDF. Do známého PDFka lze z jiných vektorových formátů (ať už SVG či ESP) obrázky přenést snadno pomocí grafických programů, jako je třeba Inkscape. LaTeXsi rozhodně poradí i s tradičními formáty PNG a JPG, avšak tyto obrázky mohou zabírat více prostoru a při tisku se může projevit nižší rozlišení obrázků. Pokud chceš používat tyto obrázky, rozhodně měj na paměti, aby měli rozlišení alespoň 250 indálně 330 ppi.

Obrázky se vkládají do prostředí `figure`, při úpravě šířky je možné krom tradičních jednotek jako cm nebo mm použít také jako jednotku šířku stránky `textwidth` to se hodí zejména když chceš mít více podobrázků.

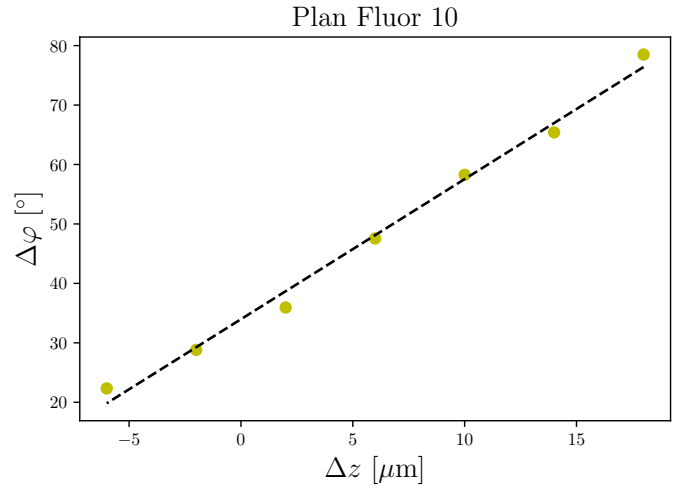
U každého obrázku je důležité aby měl popisek, `caption`. Do popisku napiš, co na obrázku je, případně nějaký další popis, tak aby čtenář následně neměl sebemenší pochybnost. U obrázků co nejsou tvoje nezapomeň an citaci. Jinak by to totiž znamenalo, že jsi obrázek dělal ty sám, což není etické přivlastňovat si cizí díla. Popisek obrázku je věta, proto musí vždy končit tečkou.

Obrázek 4.1: Logo SŠPU Opava [5].

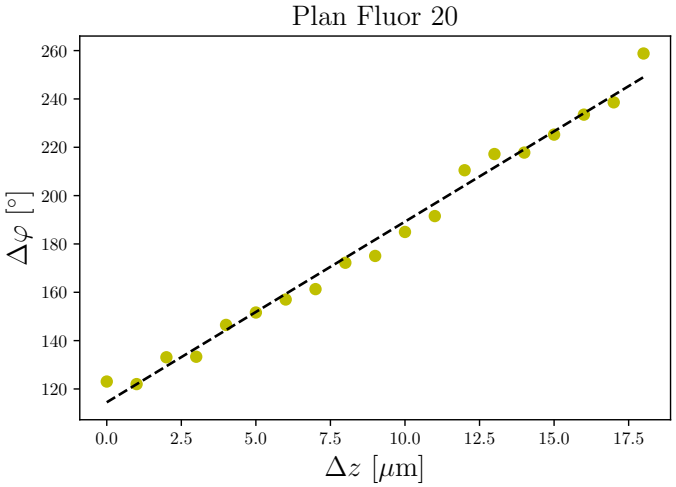
Když chceš odkazovat na obrázek, stačí pak už jen napsat příkaz `ref` a do závorek napsat označení obrázku. Třeba logo SOČky, můžeš vidět na obrázku 4.1 [**?**].

Pokud bys měl více podobrázků přichází do hry balíček `subcaption`. Pomocí něj lze vysázet i podobrázky. U podobrázků se popisek píše pouze jeden, dolů. Je v tomto případě vhodné použít navíc hranaté závorky, do nichž se napíše kratší popisek, který se následně ukáže v seznamu obrázků.

Všimni si, že obrázky jsou naschvál široké. Je to proto, aby byly dobře čitelné. Také si



(a)



(b)

Obrázek 4.2: Graf závislosti rotace DH PSF $\Delta\varphi$ na defokusaci objektivu Δz, (a) při použití objektivu Plan Fluor 10, (b) při použití objektivu Plan Fluor 20. Měřená data (žluté body) jsou lineárně proloženy (přerušovaná přímka).

všimni popisku grafů. Ačkoli nejspíš netušíš co je to DH PSF či defokusace objektivu mělo by ti být jasné, že je důležité přesně graf popsat. To znamená co je na vodorovné ose, co je na svislé ose. V jakých jednotkách veličiny jsou. Které body co znamenají, která křivka má jaký význam. Napsat samotné „$\Delta\varphi$“ je málo, vždy raději připoměň, co daná značka znamená.

4.1.3 Literatura

V LaTeXu lze dělat seznam literatury dvěma způsoby. V této šabloně jsem použil ten, kdy se seznam literatury píše přímo do práce. Pro jeho vygenerování doporučuji použít některý z generátarů, jako jsou například Citace PRO [6]. Pomocí citací lze vygenerovat přímo dokument,

který se pak už jen překopíruje do textu a člověk nemusí nic zvýrazňovat. Dále lze využít Bibtex, který rozhodně do budoucna hodlám zaimplementovat do šablony, avšak jeho použití nemusí být tak přátelské k začátečníkům.

Pokud bys chtěl odkazovat na vícero zdrojů stačí je napsat vedle sebe oddělené čárkou [2, 6, 7]. Případně můžu odkaz na konkrétní stránku dát do hranatých závorek, viz [7, str. 1]

4.1.4 Programový kód

Pro vložení programového kódu do dokumentu LaTeX s možností zvýraznění syntaxe můžete použít balíček `listings`. Tento balíček nabízí široké možnosti pro formátování kódu, včetně zvýraznění syntaxe pro různé programovací jazyky.

Nejprve je třeba do preambule LaTeX dokumentu přidat `usepackagelistings` a nastavit příslušné parametry. Příklad nastavení pro jazyk Python by mohl vypadat takto:

```
// JavaScript code here
function helloWorld() {
  console.log("Hello, world!");
}
```

Kód 4.1: Ukázka JS kódu

```
/* eslint-env es6 */
/* eslint-disable no-unused-vars */

import Axios from 'axios'
import { BASE_URL } from './utils/api'
import { getAPIToken } from './utils/helpers'

export default class User {
  constructor () {
    this.id = null
    this.username = null
    this.email = ''
    this.isActive = false
    this.lastLogin = ''  // ISO 8601 formatted timestamp.
    this.lastPWChange = ''  // ISO 8601 formatted timestamp.
  }
}

```

```
19  const getUserProfile = async (id) => {
20    let user = new User()
21    await Axios.get(
22    `${BASE_URL}/users/${id}`,
23    {
24      headers: {
25        'Authorization': `Token ${getAPIToken()}`,
26      }
27    }
28    ).then{response => {
29        // ...
30      }).catch(error => {
31        // ...
32      })
33    }
```

Kód 4.2: ES6 (ECMAScript-2015) Listing

4.2 POKROČILEJŠÍ TIPY, KTERÉ SE MOHOU HODIT

4.2.1 Rovnice

Sazba matematiky je věda sama o sobě. Ačkoli Word prošel obrovskou změnou a je v tomto mnohem lepší, tak LaTeXje pro to přímo (ještě jsem neviděl matematika, co by používal Word). Spolu s balíčky `amsmath` a `amsfonts` snad neexistuje nic, co by se používalo a LaTeXby to nezvládl. Ať už jde o základní věci jako řecká písmenka – $\alpha, \beta, \gamma, \ldots$ – integrály – $\int_{l_i}^{l_f} \tau \mathrm{d}l$ – až třeba po speciální písmena – $\mathscr{F} : \mathbb{R}^n \to \mathbb{R}^m$. Pro případ, že bys potřeboval nějaké speciální integrály, je tu balíček `esint`, pomocí něj můžeš napsat třeba

$$\oiint_{S(V)} \vec{E} \cdot \mathrm{d}\vec{S} = \iiint_V \left(\vec{\nabla} \cdot \vec{E}\right) \mathrm{d}V.$$

Jak můžeš vidět tak rovnice lze psát jednak do textu a nebo pokud se jedná o nějakou důležitou nebo rozsáhlejší rovnici tak na samostatný řádek. Pokud je rovnice opravdu důležitá, tak je vhodné ji také číslovat. Pak se na ni můžeš dále odkazovat v textu.

$$\vec{F} = m\vec{a} \tag{4.1}$$

... Například podle druhého Newtonova zákona, rovnice (4.1) ... Zároveň je vždy nutné vysvětlit co která veličina znamená. V tomto případě bych napsal, že v druhém Newtonově zákoně vektor síly $\vec{F}$ odpovídá součinu hmotnosti tělesa m a jeho zrychlení $\vec{a}$.

Věřím, že se sazbou matematiky ti pomůže tvůj školitel, případně mi můžeš napsat (mail je v úvodu). Jednotlivé funkcionality spolu se seznamem znaků nalezneš jednak v Ne příliš stručném úvodu [2] nebo na Wikibooks v sekcích *Mathematics* a *Advanced mathematics* [3].

5 KDYŽ DOKONČUJI PRÁCI

Každou práci je dobré zkontrolovat, aby v ní nebyly pravopisné chyby, nebyla těžkopádně napsaná – byla čtivá – a neobsahovala žádný typografický nedostatek. Proto, když práci sepíšeš, nech ji chvilku odležet, třeba týden. Pak si ji po sobě znovu přečti. Hned uvidíš, kolik věcí bys napsal jinak případně kde tě bije do očí jaká chyba. Dej práci přečíst také svému školiteli a případně češtináři. Zajistíš tak, že bude obsahovat méně chyb.

Pak můžeš práci vytisknout a hurá do soutěže.

ZÁVĚR

Věřím, že jsem ti spolu se šablonou poskytl několik tipů, jak napsat práci. Ať už jde o úplné začátky s LaTeXem. Či ukázku toho, co vše s ním zvládneš. Pokud bys měl k šabloně libovolné dotazy, rouhodně se na mě obrať. LaTeXtvé práci dodá určitou krásu, tak doufám, že ti dodá sebevědomí a uspěješ při souteži. A i kdyby ne vzpomeň si, kolik ses toho musel naučit a hned uvidíš o jaký kus ses posunul.

LITERATURA

[1] DOKULIL Jakub. *Šablona pro psaní SOČ v programu LaTeX* [Online]. Brno, 2020 [cit. 2020-08-24]. Dostupné z: https://github.com/Kubiczek36/SOC_sablona

[2] OETIKER, Tobias, Hubert PARTL, Irene HYNA, Elisabeth SCHEGL, Michal KOČER a Pavel SÝKORA. *Ne příliš stručný úvod do systému LaTeX2e* [online]. 1998 [cit. 2020-08-24]. Dostupné z: https://www.jaroska.cz/elearning/informatika/typografie/lshort2e-cz.pdf

[3] *Wikibooks: LaTeX* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-08-24]. Dostupné z: https://en.wikibooks.org/wiki/LaTeX

[4] *TeX - LaTeX Stack Exchange* [online]. Stack Exchange, 2020 [cit. 2020-09-01]. Dostupné z: https://tex.stackexchange.com

[5] *Střední škola průmyslová a umělecká Opava* [online]. [cit. 2023-11-11]. Dostupné z: https://www.sspu-opava.cz

[6] *Citace PRO* [online]. Citace.com, 2020 [cit. 2020-08-31]. Dostupné z: https://www.citacepro.com

[7] BORN, Max a Emil WOLF. *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. 7th (expanded) edition. Reprinted wirth corrections 2002. 15th printing 2019. Cambridge: Cambridge University Press, 2019. ISBN 978-0-521-64222-4.

Seznam obrázků

Seznam tabulek

PŘÍLOHA A SPOT DIAGRAMY A DALŠÍ