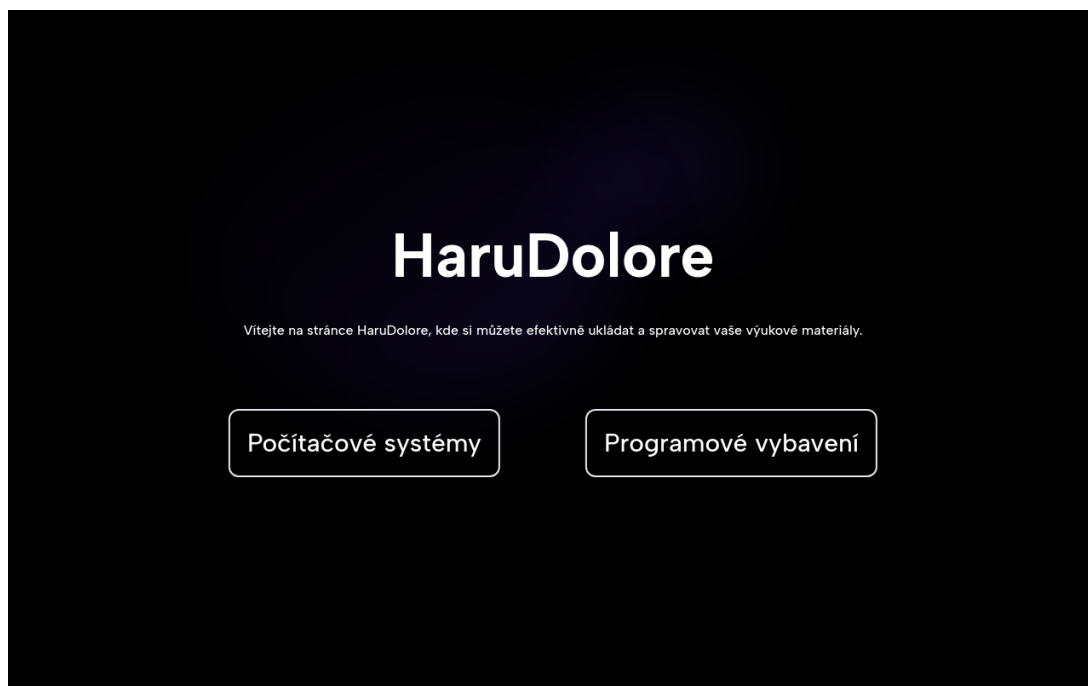


ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

HaruDolore



Autor: Mai Anh Perinová
Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování
Třída: IT4
Školní rok: 2023/24

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

Výsledkem projektu je funkční webová aplikace pro ukládání souborů k určité maturitní otázce. Využila jsem Strapi jako backend a na frontend React Next.js. Komunikaci mezi něma jsem zajistila pomocí GraphQL. Aplikace umožňuje přihlášení uživatele přes E-mail, Google, GitHub a Microsoft. Stránka obsahuje maturitní otázky, které jsou rozděleny do určitých kategorií a jsou přehledně zobrazeny. Uživatel si vybírá otázku ke které pak přidá výukový soubor. Soubory se ukládají ke každé otázce zvlášť a jsou k dispozici ke stažení pro přihlášeného uživatele. Každý uživatel vidí pouze své uložené soubory. Při nahrávání výukového materiálu musí uživatel soubor zařadit do kategorie, která se pak zobrazuje jako štítek vedle nahraného souboru. Vedle štítku se pak nachází tlačítko, které umožňuje smazání souboru.

Klíčová slova

webová stránka, databáze, uživatelské účty, soubory ...

Abstract

The result of the project is a functional web application for storing files for a specific graduation question. I used Strapi for the backend and React Next.js for the frontend. I ensured communication between them using GraphQL. The application allows user login via Email, Google, GitHub and Microsoft. The site contains graduation questions that are divided into specific categories and are clearly displayed. The user chooses a question to which they then add a educational file. The files are stored for each question separately and are available for download to the logged-in user. Each user can only see their own saved files. When uploading a educational material, the user must categorize the file, which is then displayed as a label next to the uploaded file. Next to the label, there is a button allowing the user to delete the file.

Keywords

website, database, user accounts, files ...

Obsah

Úvod	2
1 Backend	3
1.1 Headless CMS	3
1.1.1 Strapi	3
1.1.2 Strapi GraphQL	4
1.1.3 Strapi Cloud	6
1.2 Komunikace pomocí GraphQL	6
1.3 Databázový model	7
1.4 Vercel	7
1.5 Autorizace a Autentizace	7
1.5.1 Clerk	8
1.5.2 Základní operace s uživatelským účtem	9
2 Frontend	11
2.1 React	11
2.1.1 React JSX	11
2.1.2 React komponent	12
2.1.3 React stav	12
2.2 Next.js 13	12
2.2.1 Optimalizace obrázků	13
2.2.2 Optimalizace fontů	13
2.2.3 Funkce generateStaticParams	14
2.2.4 <Link> komponent	14
2.2.5 React server komponenty	15
2.3 Apollo Client	15
2.3.1 Propojení frontendu s backendem	15
2.4 Tailwind CSS	16
3 Funkce aplikace	17
3.1 Přihlášení	17
3.2 Nahrávání souboru	17
3.3 Stahování souboru	18
3.4 Mazání souboru	19

ÚVOD

Mým záměrem bylo si zkusit vytvořit webovou aplikaci v React Next.js na téma maturitní okruhy. Chtěla jsem docílit toho, aby všechny výukové materiály, týkající se maturity, byly na jednom místě a také se přehledně zobrazovaly. K tomu jsem chtěla přidat uživatelské účty, aby každý měl prostor si ukládat své výukové materiály a kdykoli je měl k dispozici ke stáhnutí. Měla jsem zájem se seznámit s moderními technologiemi React a Next.js, ale tyhle technologie by mi nestačily a musela jsem k tomu přidat backend. Původně jsem chtěla využít headless CMS Hygraph, který má self cloud hosting a nemusela bych řešit cloudové hostování svých dat. Ale po konzultaci se svým učitelem jsme došli ke závěru, že by aplikace obsahovala moc dat, tak by bylo lepší využít Strapi a následně aplikaci dockerizovat. Strapi ale nabízelo plugin Strapi cloud, který zajistil cloud hosting mých dat, a tak jsem se k dockerizování aplikaci nedostala. U komunikace frontendu s backendem jsem využila dotazovací jazyk GraphQL. Narozdíl od REST API, které přenáší pevně stanovená data bez ohledu na potřeby klienta, GraphQL umožňuje klientům získávat přesně ta data, která potřebují, a to v jediném požadavku.

K řešení jsem využila technologie Strapi, Vercel, Clerk, React, Next.js, Apollo Client a Tailwind CSS.

1 BACKEND

1.1 HEADLESS CMS

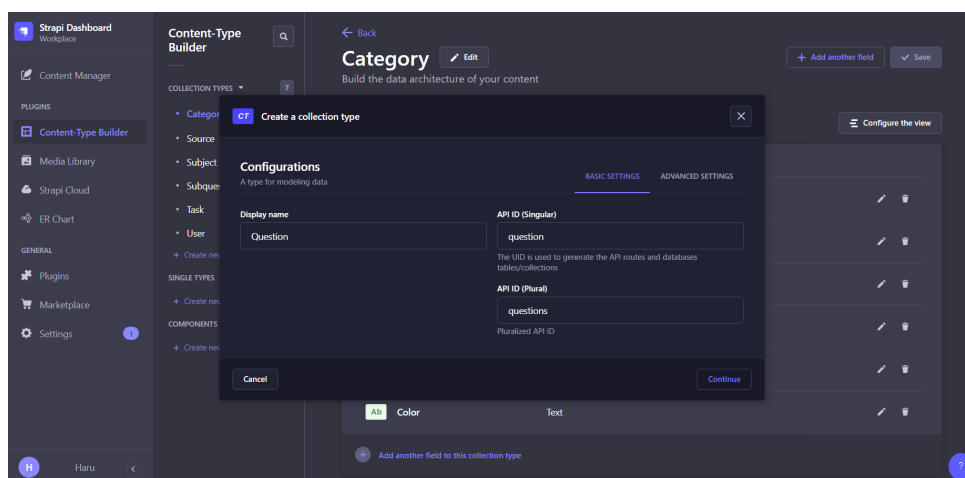
Pro vývoj své webové aplikace jsem se rozhodla využít Headless CMS, protože je to jednoduchý způsob jak ukládat a spravovat své data na backendu, a to bez nutnosti vytváření vlastního backendu se serverem. Headless CMS umožňuje oddělit správu obsahu (backend) od prezentace obsahu (frontend), což přináší flexibilitu ve výběru frontendových technologií a zjednodušuje celkový vývoj aplikace.

1.1.1 Strapi

Původně jsem zvažovala možnost využití Hygraph, který využívá self cloud hosting, ale po následné konzultaci s panem učitelem jsme došli k závěru využití Strapi a později dockerizování aplikace.

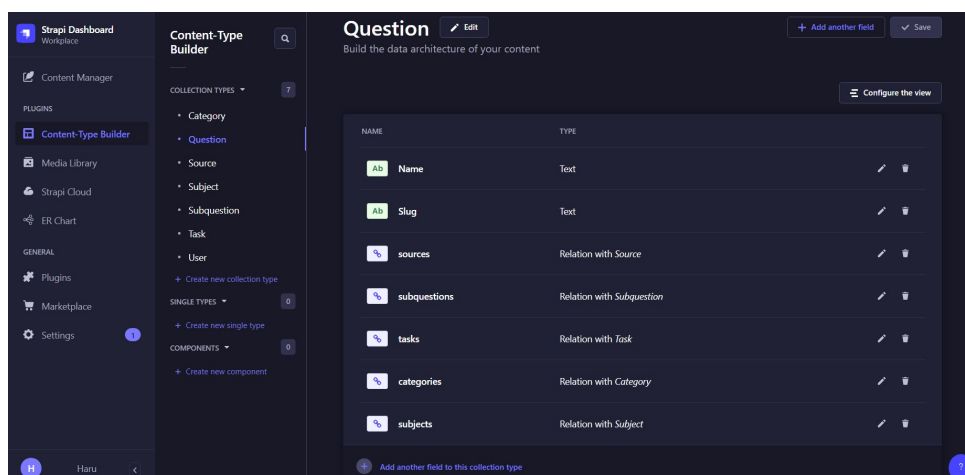
Tvorba modelu

Strapi umožňuje lehkou a přehlednou tvorbu modelu. V Strapi dashbord kliknutím na 'Create new collection type' jsem vytvořila model nazvaný 'Question'.



Obrázek 1.1: Ukázka vytváření modelu ve Strapi

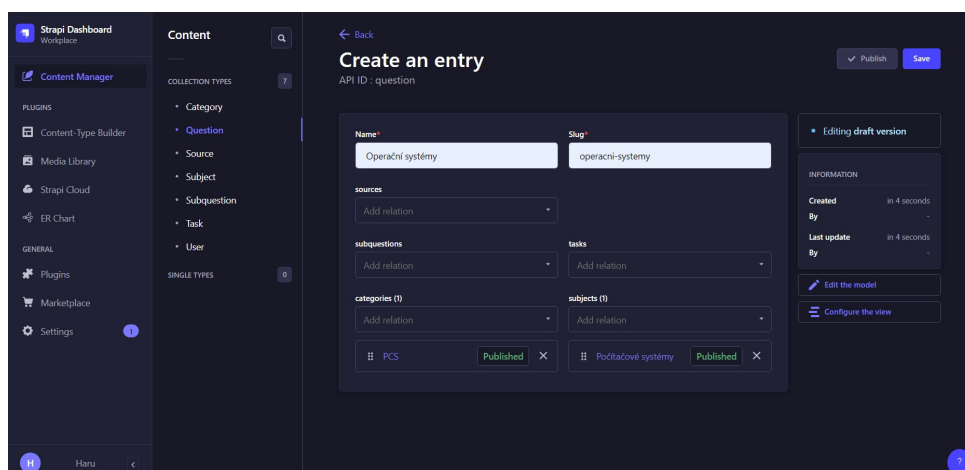
Poté jsem přidala jednotlivá pole pomocí tlačítka 'Add another field'. Také jsem propojila model Question s ostatními modely pomocí relací.



Obrázek 1.2: Ukázka přidávání pole do modelu

Vkládání dat

Data jsem pak vkládala tlačítkem 'Create new entry', kde jsem vyplnila následující pole a uložila tlačítkem 'Save'



Obrázek 1.3: Ukázka vkládání dat do modelu

1.1.2 Strapi GraphQL

GraphQL API umožňuje provádět dotazy a mutace pro interakci s typy obsahu prostřednictvím GraphQL pluginu v Strapi. Výsledky lze filtrovat, třídit a stránkovat. Ve své aplikaci jsem vy-

užila filtraci a třídění. Plugin měl také k dispozici playground, který jsem často využívala k testování dotazů předtím, než jsem je použila v kódu.

```
1  const createNewSource = gql`
2  mutation (
3    $name: String!
4    $userId: String!
5    $file: ID!
6    $categoriesId: [ID]!
7    $questionId: ID
8  ) {
9    createSource(
10     data: {
11       Name: $name
12       Userid: $userId
13       publishedAt: "2007-12-03T10:15:30Z"
14       categories: $categoriesId
15       question: $questionId
16       Files: $file
17     }
18   ) {
19     data {
20       id
21     }
22   }
23 }
24 `;
25
26 const uploadFile = gql`
27 mutation ($file: Upload!) {
28   upload(file: $file) {
29     data {
30       id
31     }
32   }
33 }
34 `;
```

Kód 1.1: Ukázka mutace v Content.js

1.1.3 Strapi Cloud

Strapi Cloud je platforma pro správu obsahu postavená na Strapi, předním open-source headless CMS. Jedná se o plně spravovanou hostingovou platformu, která umožňuje nasadit Strapi aplikace během několika minut. Je to placená platforma, ale využila jsem jejich zkušební dobu na 14 dní. Díky Strapi cloud jsem nemusela aplikaci dockerovat.

1.2 KOMUNIKACE POMOCÍ GRAPHQL

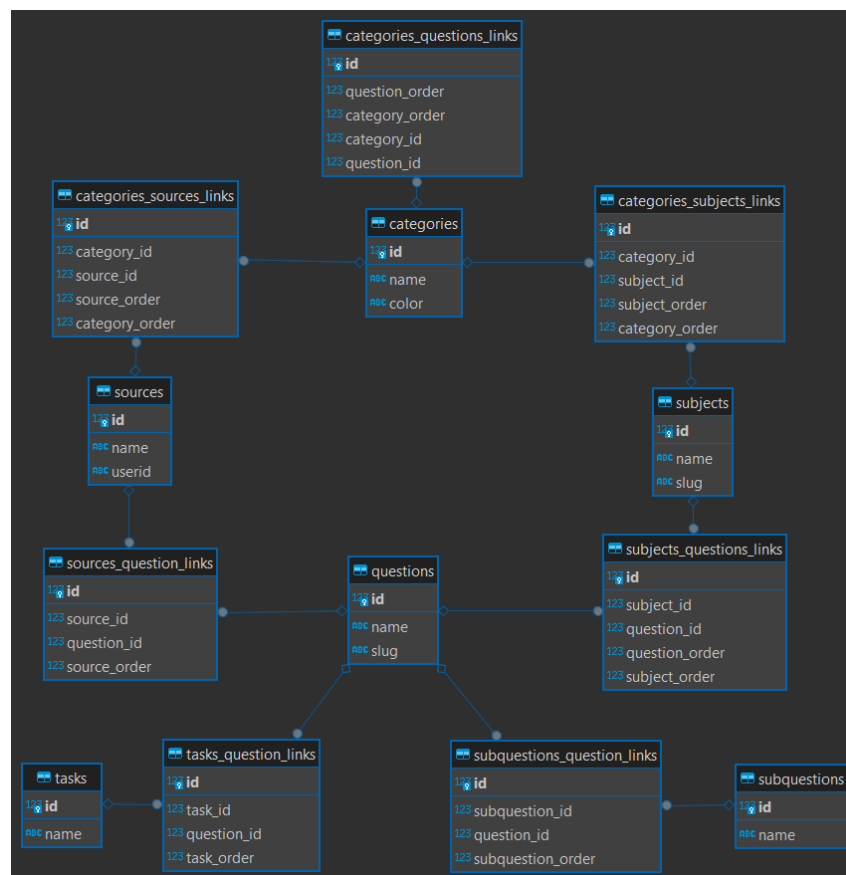
GraphQL je dotazovací jazyk pro rozhraní API a běhové prostředí pro plnění těchto dotazů s existujícími daty. Pro komunikaci mezi frontendem a backendem jsem využila GraphQL hlavně z důvodu, že GraphQL umožňuje klientům získat přesně ty data, které potřebují, což může vést k menšímu množství přenášených dat narozdíl od REST API, které poskytuje pevně definované endpointy s předem stanovenými daty, což může znamenat, že klienti musí získat více dat, než skutečně potřebují.

GraphQL dotaz je využíván k získání informací z API, které implementuje server postavený na Strapi. Na frontendové straně jsem pak využila Apollo Client k odeslání requestu.

```
1  query {  
2    categories {  
3      data {  
4        id  
5        attributes {  
6          Name  
7        }  
8      }  
9    }  
10 }
```

Kód 1.2: Ukázka query v Content.js

1.3 DATABÁZOVÝ MODEL



Obrázek 1.4: Ukázka databázového modelu

Základem modelu jsou otázky - questions, na které navazují předměty - subjects (počítačové systémy, programové vybavení), kategorie - categories (PCS, HW, PVY, PRP), podotázky - subquestions, praktické úkoly - tasks, zdroje - sources

1.4 VERCEL

Vercel je cloudová platforma pro nasazení a správu webových aplikací. Rozhodla jsem se využít tuto technologii, protože poskytuje jednoduchý způsob, jak nasadit statické a dynamické webové stránky, serverless funkcionalitu a další projekty spojené s vývojem webových a serverless aplikací.

1.5 AUTORIZACE A AUTENTIZACE

1.5.1 Clerk

Clerk je autentizační platforma umožňující přihlašování pomocí hesel, poskytovatelů sociálních identit, jednorázových přístupových kódů e-mailem nebo SMS, vícefaktorového ověřování a základní správy uživatelů.

Přidání ověřování a správy uživatelů do aplikace

Clerk se nainstaloval pomocí příkazu 'npm install @clerk/nextjs'. Poté se v souboru '.env.local' vložil KEY kód. Aplikace se následně zabalila do <ClerkProvider>.

```
1  import { ClerkProvider } from "@clerk/nextjs";
2
3  export default function RootLayout({ children }) {
4    return (
5      <ClerkProvider appearance={{ baseTheme: dark }}>
6        <html lang="en">
7          <body className={albert.className}>
8            <ApolloWrapper>{children}</ApolloWrapper>
9          </body>
10         </html>
11       </ClerkProvider>
12     );
13  }
```

Kód 1.3: Ukázka <ClerkProvider> v layout.js

Vyžadování ověřování pro přístup k aplikaci je řešeno v souboru 'middleware.ts'. Tímto způsobem je zabezpečena ochrana před neautorizovaným přístupem na tyto cesty, přičemž uživatelé, kteří nejsou ověřeni, budou přesměrováni na stránku pro přihlášení nebo na hlavní stránku.

```
1  import { authMiddleware, redirectToSignIn } from "@clerk/nextjs";
2
3  export const config = {
4    matcher: ["/(?!.+\\.\\. [\\w]+$_next).*", "/", "/(api|trpc)(.*)"],
5  };
```

Kód 1.4: Ukázka kódu v middleware.ts

Posledním krokem bylo vložit tlačítko `<UserButton/>`. Zde je `<UserButton/>` použit tak, že když je uživatel přihlášen, zobrazí se mu `<UserButton/>` a v opačném případě kdy je uživatel odhlášen, zobrazí se mu `<SignInButton/>`

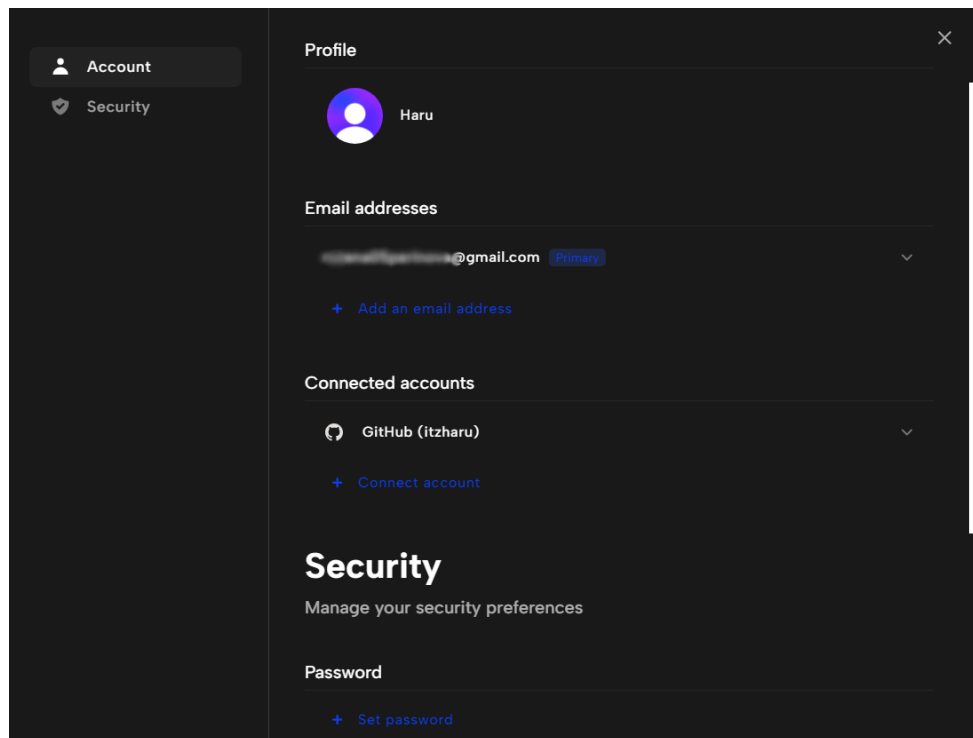
```
1 <button>
2 <SignedIn>
3   <UserButton />
4 </SignedIn>
5 <SignedOut>
6 <SignInButton />
7   </SignedOut>
8 </button>
```

Kód 1.5: Ukázka kódu v `middleware.ts`

1.5.2 Základní operace s uživatelským účtem

Clerk poskytuje řadu základních operací s uživatelským účtem, které umožňují správu uživatelských účtů v aplikaci. Mezi tyto základní operace patří:

- Registrace
- Přihlášení
- Odhlášení
- Obnova hesla
- Správa účtů
- Ověření emailu



Obrázek 1.5: Ukázka správy účtu

Clerk má předem vestavěnou správu účtů a nabízí řadu stylů, které se mohou aplikovat. V aplikaci jsem použila tmavé téma.

```
1 import { dark } from "@clerk/themes";
2
3 export default function RootLayout({ children }) {
4   return (
5     <ClerkProvider appearance={{ baseTheme: dark }}>
6       <html lang="en">
7         <body className={albert.className}>
8           <ApolloWrapper>{children}</ApolloWrapper>
9         </body>
10      </html>
11    </ClerkProvider>
12  );
13 }
```

Kód 1.6: Ukázka Clerk dark theme v layout.js

2 FRONTEND

2.1 REACT

React je knihovna pro webová a nativní uživatelská rozhraní. Uživatelská rozhraní se skládají z jednotlivých částí nazývanými komponenty, které jsou napsané v jazyce JavaScript. React umožňuje snadno vytvářet interaktivní uživatelská rozhraní.

2.1.1 React JSX

JSX je zkratka pro JavaScript XML a umožňuje zapisovat prvky HTML v jazyce JavaScript a umisťovat je do DOM (Document Object Model) bez metod createElement() a/nebo appendChild().

```
1  return (  
2    <div>  
3      {data.subjects.data.map((subject) => {  
4        return (  
5          <div key={subject.slug}>  
6            <h3>  
7              {subject.attributes.Name}  
8            </h3>  
9          </div>  
10         );  
11       })}  
12     </div>  
13  );
```

Kód 2.1: Ukázka využití JSX v page.js

2.1.2 React komponent

Komponenty jsou nezávislé a opakovaně použitelné části kódu. Slouží ke stejnému účelu jako funkce JavaScriptu, ale pracují samostatně a vracejí HTML. Komponenty existují ve dvou typech, komponenty třídy a komponenty funkce. V aplikaci jsem využila jeden komponent funkce nazvaný 'Collapsible.jsx' a na stránce mi umožňuje v sidebaru rozkliknout kategorii, ve které se nachází otázky a zároveň zpátky kategorii zavřít.

2.1.3 React stav

React komponenty mají vestavěný stavový objekt. Do objektu stavu se ukládají hodnoty vlastností, které patří komponentě. Když se stavový objekt změní, komponenta se znovu vykreslí. Využila jsem React stav ve své komponentě 'Collapsible.jsx'. Používá se zde jednoduchý přepínací mechanismus, který umožňuje změnu stavu z otevřené kategorie na zavřenou po každém kliknutí. Tento mechanismus je následně implementován na stránce v bočním panelu a poskytuje funkcionalitu rozklikávání a zavírání kategorií.

```
1  import { useState } from "react";
2
3  const [isOpen, setIsOpen] = useState(true);
4
5  function toggleCollapsible() {
6    setIsOpen((old) => !old);
7  }>
```

Kód 2.2: Ukázka useState() v Collapsible.jsx

2.2 NEXT.JS 13

Next.js je framework pro React určený k tvorbě plně vybavených webových aplikací. K vytváření uživatelských rozhraní se používají React komponenty a Next.js se využívá pro další funkce a optimalizace.

Využité výhody z Next.js:

- Vykreslování na straně serveru (SSR)
- Optimalizace SEO
- Optimalizace obrázků

- Optimalizace fontů
- funkce `generateStaticParams`
- `<Link>` komponent
- Server komponenty

2.2.1 Optimalizace obrázků

Komponent `Next.js Image` rozšiřuje HTML element `` o funkce pro automatickou optimalizaci obrázků:

- Optimalizace velikosti: Automatické zobrazování obrázků ve správné velikosti pro každé zařízení pomocí moderních formátů, jako jsou WebP a AVIF.
- Vizuální stabilita: Automatická prevence posunu rozvržení při načítání obrázků.
- Rychlejší načítání stránek: Obrázky se načítají pouze ve chvíli, kdy vstoupí do zobrazovacího prostoru pomocí nativního líného načítání prohlížeče, s volitelnými zástupnými znaky rozmazání.
- Flexibilita prostředků: Možnost změny velikosti obrázků na vyžádání, a to i pro obrázky uložené na vzdálených serverech.

2.2.2 Optimalizace fontů

Automaticky optimalizuje font (včetně vlastních fontů) a odstraňuje externí síťové požadavky, čímž zlepšuje soukromí a výkon.

`'next/font'` obsahuje vestavěný automatický selfhosting pro jakýkoli soubor fontů. Webové fonty se optimálně načítají bez jakéhokoli posunu v rozložení díky podkladovému CSS.

```

1  import { Albert_Sans } from "next/font/google";
2
3  const albert = Albert_Sans({
4    subsets: ["latin"],
5    weight: ["200", "500", "700", "800", "900"],
6  });

```

Kód 2.3: Ukázka `layout.js`

2.2.3 Funkce generateStaticParams

Funkce generateStaticParams je zde využita ke generování slugů pro subject a question na základě dat získaných z databáze. Využívá objekt klienta k provedení dotazu na subject a následně mapuje výsledek, aby se vytvořil seznam objektů obsahujících slugy.

Výsledkem této funkce je ploché pole obsahující slugy, které se využívají při generování statických stránek a navigaci v aplikaci.

```
1 export async function generateStaticParams() {  
2   const client = getClient();  
3   const { data } = await client.query({ query: getSubject });  
4   const result = data.subjects.data.map((subject) => {  
5     return subject.attributes.questions.data.map((question) => {  
6       return {  
7         subjectSlug: subject.attributes.Slug,  
8         questionSlug: question.attributes.Slug,  
9       };  
10    });  
11  });  
12  return result.flat();  
13 }
```

Kód 2.4: Ukázka funkce generateStaticParams v page.js

2.2.4 <Link> komponent

<Link> je komponent v Reactu, který rozšiřuje HTML prvek <a> a umožňuje přednačítání a navigaci mezi routami na straně klienta. Je to primární způsob navigace mezi trasami v Next.js.

```
1 import Link from "next/link";  
2  
3 <Link  
4   key={question.slug}  
5   href={"/subjects/" + params.subjectSlug + "/" + question.slug }  
6   >  
7   {order}. {question.name}  
8 </Link>
```

Kód 2.5: Ukázka next/link v layout.js

2.2.5 React server komponenty

Server komponenty představují způsob spouštění React komponentů na straně serveru a následné odeslání vykresleného výstupu do Reactu na straně klienta. To znamená, že se nemusí odesílat všechny závislosti, na kterých projekt závisí, což vede k rychlejšímu počátečnímu načtení stránky.

Next.js 13 přináší řadu změn ve způsobu vytváření a vykreslování komponent, včetně zavedení RSC. App Router je nový směrovací systém, který je postaven nad RSC a poskytuje podporu vnořených tras, rozvržení, stavů načítání, zpracování chyb a dalších funkcí.

2.3 APOLLO CLIENT

Apollo Client je komplexní knihovna pro správu stavu v jazyce JavaScript, která umožňuje spravovat místní i vzdálená data pomocí jazyka GraphQL. Pro propojení frontendu s backendem jsem se rozhodla využít Apollo Client pro jeho popularitu a taky protože většina tutoriálu na propojení Strapi GraphQL s React Next.js nabízela Apollo Client.

2.3.1 Propojení frontendu s backendem

Začala jsem instalací závislostí 'npm install @apollo/client graphql'. V souboru client.js jsem inicializovala Apollo Client pomocí uvedeného kódu.

```
1 import createUploadLink from "apollo-upload-client/createUploadLink.mjs";
2 import {
3   NextSSRInMemoryCache,
4   NextSSRApolloClient,
5 } from "@apollo/experimental-nextjs-app-support/ssr";
6 import { registerApolloClient } from
7   "@apollo/experimental-nextjs-app-support/rsc";
8
9 let apolloClient;
10
11 export const { getClient } = registerApolloClient(() => {
12   const client = new NextSSRApolloClient({
13     cache: new NextSSRInMemoryCache(),
14     link: createUploadLink({
15       uri: "https://renowned-gift-126140aec8.strapiapp.com/graphql",
16     }),
17   });
```

```

17   });
18
19   apolloClient = client;
20   return client;
21 });
22

```

Kód 2.6: Ukázka inicializace Apollo Clientu v client.js

Posledním krokem bylo obalit aplikaci wrapperem `<ApolloProvider>`.

```

1   <html lang="en">
2     <body className={albert.className}>
3       <ApolloWrapper>{children}</ApolloWrapper>
4     </body>
5   </html>

```

Kód 2.7: Ukázka komponentu `<ApolloProvider>` v client.js

2.4 TAILWIND CSS

Tailwind CSS je populární CSS framework, který poskytuje sadu předdefinovaných utilit a tříd pro stylování webových stránek a aplikací. Díky Tailwind CSS jsem nemusela vytvářet vlastní CSS styl a rovnou do elementů jsem psala styly pomocí `'className'`. Vybrala jsem si Tailwind CSS jako svůj framework, protože místo předdefinovaných komponentů či stylů umožňuje pracovat přímo s jednotlivými vlastnostmi stylu a díky tomu jsem si mohla stránku nastýlovat přesně podle sebe.

```

1   <div
2     className="relative py-1 px-2 w-fit h-full text-white rounded-l-sm"
3     style={{ backgroundColor: tag.Color }}
4   >
5     {tag.Name}
6   </div>

```

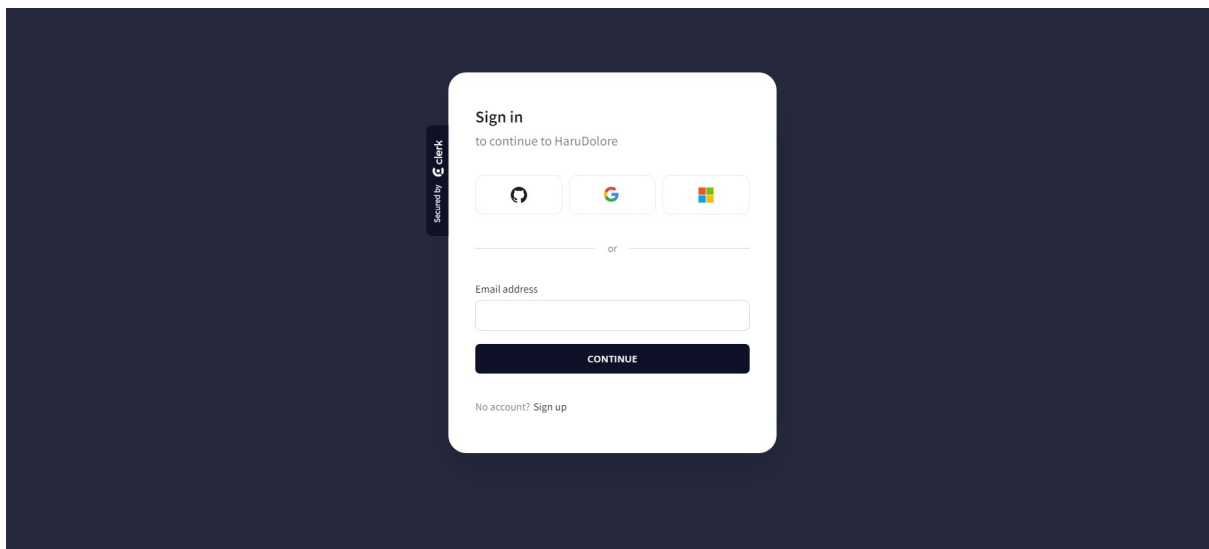
Kód 2.8: Ukázka `useState()` v Collapsible.jsx

3 FUNKCE APLIKACE

Aplikace nemá moc funkcí, jelikož jsem začala aplikaci vyvíjet pozdě a za tu dobu mě potkalo hodně komplikací. Jedna z hlavních funkcí která aplikace poskytuje je nahrávání souboru a stahování již nahraných souborů. Poslední funkcí je smazání nahraných souborů.

3.1 PŘIHLÁŠENÍ

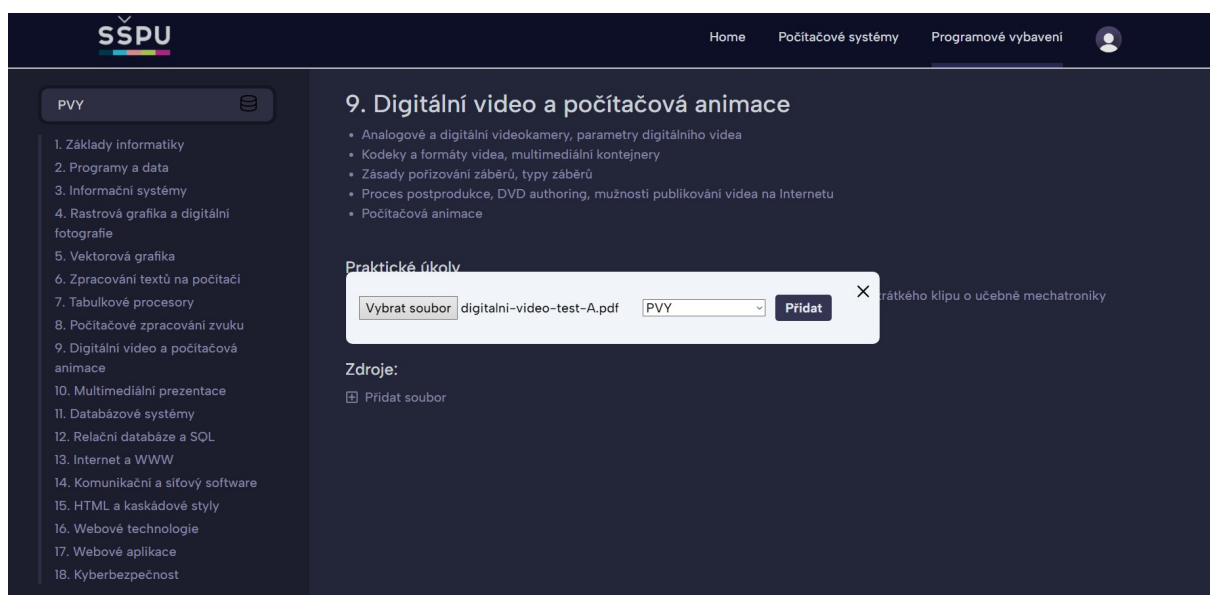
Přihlášení jsem zajistila pomocí Clerk, které nabízí přihlášení přes Github, Google, Microsoft a e-mail. Clerk má také k dispozici běžné zaregistrování uživatelů.



Obrázek 3.1: Ukázka přihlášení

3.2 NAHRÁVÁNÍ SOUBORU

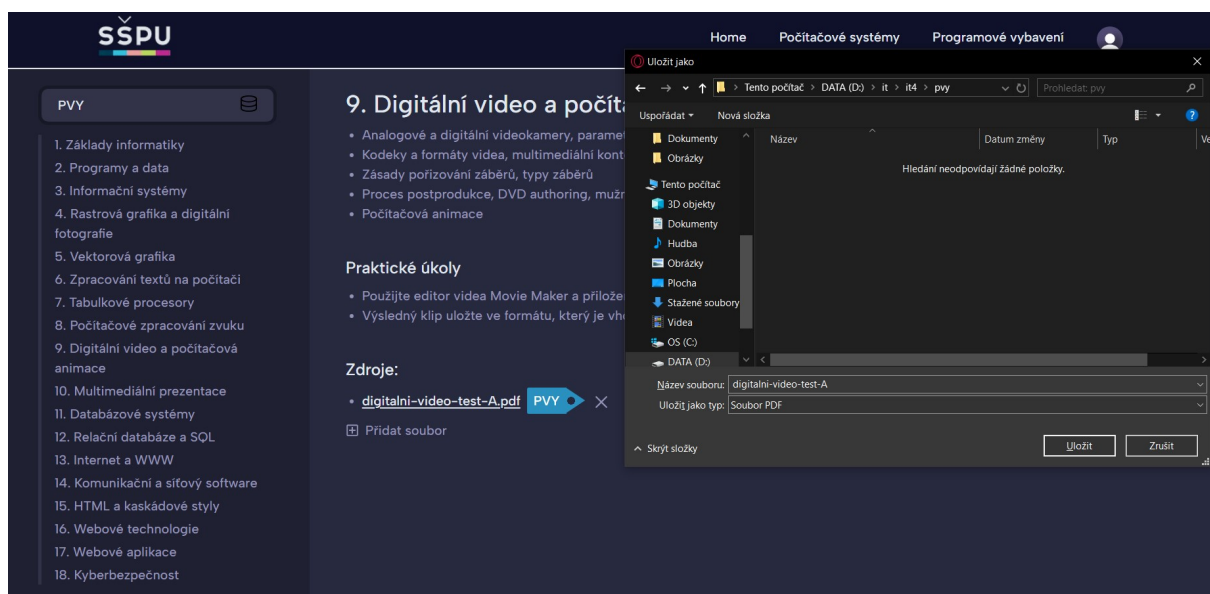
Pro nahrání souboru uživatel klikne na tlačítko 'přidat soubor'. Vyskočí formulář, kde si vybere výukový soubor společně s kategorií, do které se výukový materiál řadí. Poté stačí potvrdit tlačítkem 'přidat' a soubor se uloží. Uložené soubory se zobrazují společně se štítkem vybrané kategorie. Uživatel vidí pouze své uložené materiály.



Obrázek 3.2: Ukázka nahrávání souboru

3.3 STAHOVÁNÍ SOUBORU

Uživatel si může nahrané soubor jednoduše stáhnout kliknutím na jejich název.



Obrázek 3.3: Ukázka stahování souboru

3.4 MAZÁNÍ SOUBORU

Poslední funkce kterou aplikace umožňuje je mazání souboru. Uživatel jednoduše může mazat své soubory kliknutím na ikonku křížku, která se nachází hned vedle štítku.

ZÁVĚR

Cílem projektu bylo vytvořit webovou aplikaci s maturitním obsahem a s možností ukládání výukových materiálů. Backend je řešený pomocí Strapi s GraphQL, Vercel a frontend je řešen využitím React Next.js 13, Apollo Client a Tailwind CSS. Uživatelský systém je zajištěn pomocí Clerku. Cíla databáze běží na Strapi cloud, kde je využívána zkušební verze na 14dní.

Myslím si, že základní cíle projektu jsem splnila, i když v aplikaci chybí dost funkcionalit, které jsem plánovala přidat. Kvůli komplikacím, na které jsem narazila při vyvíjení aplikace, se mi bohužel nepodařilo přidat další funkce. Už mi nezbyl čas a potřebovala jsem dokončit hlavní funkce stránky, abych mohla stránku nasadit na Vercel. Do budoucna bych chtěla v aplikaci přidat další funkce a také pořešit responzivitu stránky, aby byla dostupná i pro uživatele na mobilu.

Aplikace je zálohována na GitHub adresách: <https://github.com/ItzHaru/frontend>

<https://github.com/ItzHaru/stranka> (starý repozitář)

<https://github.com/ItzHaru/backend>

SEZNAM POUŽITÝCH ZDROJŮ

- [1] Next.js by Vercel - The React Framework [Online]. [cit. 2023-01-12]. Dostupné z: <https://nextjs.org>
- [2] GraphQL | A query language for your API [Online]. [cit. 2023-01-12]. Dostupné z: <https://graphql.org>
- [3] Clerk | Authentication and User Management [Online]. [cit. 2023-01-12]. Dostupné z: <https://clerk.com>
- [4] W3schools Online Web Tutorials [Online]. [cit. 2023-01-13]. Dostupné z: <https://www.w3schools.com>
- [5] Tailwind CSS - Rapidly build modern websites without ever ... [Online]. [cit. 2023-01-13]. Dostupné z: <https://tailwindcss.com>
- [6] Vercel: Build and deploy the best Web experiences with The ... [Online]. [cit. 2023-01-13]. Dostupné z: <https://vercel.com>
- [7] Introduction to Apollo Client [Online]. [cit. 2023-01-14]. Dostupné z: <https://www.apollographql.com/docs/react/>
- [8] React Server Components in Next.js 13 [Online]. [cit. 2023-01-14]. Dostupné z: <https://blog.logrocket.com/react-server-components-next-js-13/>
- [9] Strapi - Open source Node.js Headless CMS [Online]. [cit. 2023-01-14]. Dostupné z: <https://strapi.io>

Seznam obrázků

1.1	Ukázka vytváření modelu ve Strapi	3
1.2	Ukázka přidávání pole do modelu	4
1.3	Ukázka vkládání dat do modelu	4

1.4	Ukázka databázového modelu	7
1.5	Ukázka správy účtu	10
3.1	Ukázka přihlášení	17
3.2	Ukázka nahrávání souboru	18
3.3	Ukázka stahování souboru	18