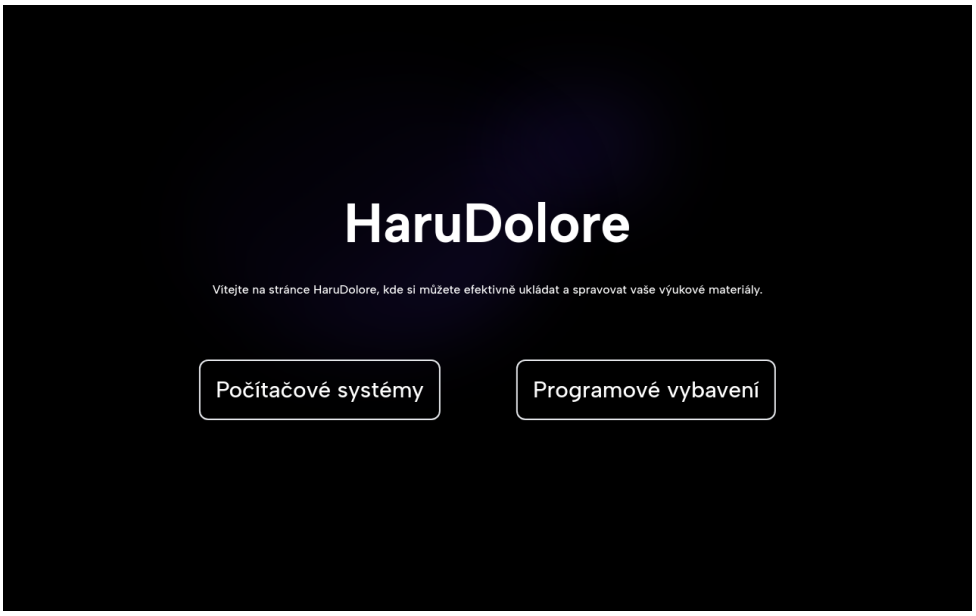
SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

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. Aplikace umožňuje přihlášení uživatele přes Email, 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 matetirálu 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. 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

Úvod

1 BACKEND

1.1 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

1.1.1 Optimalizace obrázků

Komponent Next.js Image rozšiřuje HTML element <img> 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.

1.1.2 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.

```
import { Albert_Sans } from "next/font/google";

const albert = Albert_Sans({
  subsets: ["latin"],
  weight: ["200", "500", "700", "800", "900"],
});
```

Kód 1.1: Ukázka layout.js

1.1.3 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.

```
export async function generateStaticParams() {
  const client = getClient();
  const { data } = await client.query({ query: getSubject });
  const result = data.subjects.data.map((subject) => {
    return subject.attributes.questions.data.map((question) => {
      return {
        subjectSlug: subject.attributes.Slug,
        questionSlug: question.attributes.Slug,
      };
    });
  });
  return result.flat();
}
```

Kód 1.2: Ukázka funkce generateStaticParams v page.js

1.1.4 <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.

```
import Link from "next/link";

  <Link
  key={question.slug}
  href={
    "/subjects/" + params.subjectSlug + "/" + question.slug
  }
  >
  {order}. {question.name}
  </Link>
```

Kód 1.3: Ukázka next/link v layout.js

1.1.5 React server komponenty

Server komponenty představují způsob spouštění React kmponentů 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.

1.2 Komunikace pomocí GraphQL

GraphQL je dotazovací jazyk pro rozhraní API a běhové prostředí pro plnění těchto dotazů s existujícími daty. GraphQL poskytuje úplný a srozumitelný popis dat v rozhraní API a dává klientům možnost žádat přesně to, co potřebují. Usnadňuje vývoj rozhraní API v čase a umožňuje používat výkonné vývojářské nástroje.

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

```
query {
  categories {
    data {
      id
      attributes {
        Name
      }
    }
  }
}
```

Kód 1.4: Ukázka query v Content.js

1.3 Databázový model

Obrázek 1.1: Ukázka databázového modelu

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

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>.

```
import { ClerkProvider } from "@clerk/nextjs";

export default function RootLayout({ children }) {
  return (
    <ClerkProvider appearance={{ baseTheme: dark }}>
      <html lang="en">
        <body className={albert.className}>
          <ApolloWrapper>{children}</ApolloWrapper>
        </body>
      </html>
    </ClerkProvider>
  );
}
```

Kód 1.5: 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.

```
import { authMiddleware, redirectToSignIn } from "@clerk/nextjs";

export const config = {
  matcher: ["/((?!.+\\.[\\w]+$|_next).*)", "/", "/(api|trpc)(.*)"],
};
```

Kód 1.6: 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/>

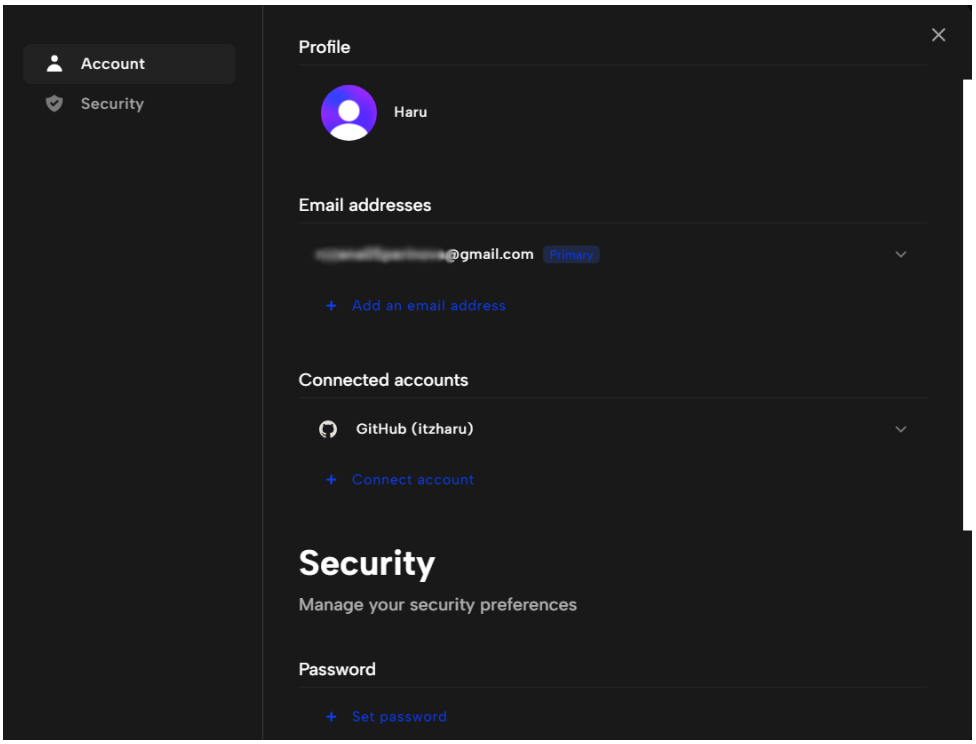
```
<button>
<SignedIn>
  <UserButton />
</SignedIn>
<SignedOut>
<SignInButton />
  </SignedOut>
</button>
```

Kód 1.7: 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 1.2: Ukázka správy účtu

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

```
import { dark } from "@clerk/themes";

export default function RootLayout({ children }) {
  return (
  <ClerkProvider appearance={{ baseTheme: dark }}>
    <html lang="en">
      <body className={albert.className}>
        <ApolloWrapper>{children}</ApolloWrapper>
      </body>
    </html>
  </ClerkProvider>
  );
}
```

Kód 1.8: 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().

```
return (
  <div>
    {data.subjects.data.map((subject) => {
      return (
        <div key={subject.slug}>
          <h3>
            {subject.attributes.Name}
          </h3>
        </div>
      );
    })}
  </div>
);
```

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 byl využit komponent funkce.

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.

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

Kód 2.2: Ukázka useState() v Collapsible.jsx

2.2 Tailwind CSS

Tailwind CSS je populární CSS framework, který poskytuje sadu předdefinovaných utilit a tříd pro stylování webových stránek a aplikací. Díky Tailwind CSS se nemusí vytvářet vlastní CSS styl a rovnou do elementů se píší 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 nastylovat 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 2.3: Ukázka useState() v Collapsible.jsx

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

```
import createUploadLink from "apollo-upload-client/createUploadLink.mjs";
import {
  NextSSRInMemoryCache,
  NextSSRApolloClient,
} from "@apollo/experimental-nextjs-app-support/ssr";
import { registerApolloClient } from
 "@apollo/experimental-nextjs-app-support/rsc";

let apolloClient;

export const { getClient } = registerApolloClient(() => {
  const client = new NextSSRApolloClient({
    cache: new NextSSRInMemoryCache(),
    link: createUploadLink({
      uri: "https://renowned-gift-126140aec8.strapiapp.com/graphql",
    }),
  });

  apolloClient = client;
  return client;
});

```

Kód 2.4: Ukázka inicializace Apollo Clientu v client.js

Posledním krokem bylo obalit aplikaci wrapperem <ApolloProvider>.

```
<html lang="en">
  <body className={albert.className}>
    <ApolloWrapper>{children}</ApolloWrapper>
  </body>
</html>
```

Kód 2.5: Ukázka komponentu <ApolloProvider> v client.js

3 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.

3.1 STRAPI

Původně jsem zvažovala možnost využití Hygraph, [] ale po následné konzultaci s panem učitelem jsme došli k závěru využití Strapi a později dockerizování aplikace.

3.2 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 využila filtraci a třízení. Plugin měl také k dispozici playground, který jsem hodně využívala na dotazy předtím než jsem je aplikovala v kódu.

```
1  const uploadFile = gql`
2    mutation ($file: Upload!) {
3      upload(file: $file) {
4        data {
5          id
6        }
7      }
8    }
9  `;
```

Kód 3.1: Ukázka mutace v Content.js

3.3 STRAPI CLOUD

4 Funkce aplikace

4.1 Nahrávání souboru

4.2 Mazání souboru

ZÁVĚR

Závěr

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