SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Webová fitness aplikace

Jan Hadamčík

Obor:	18-20-M/01 INFORMAČNÍ TECHNOLOGIE se zaměřením na počítačové sítě a programování
Třída:	IT4
Školní rok:	2022/2023

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

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

V Opavě 31. 12. 2022

podpis autora práce

ANOTACE

Výsledný projekt je funkční webová aplikace, vytvořena pomocí Reactu, a sice frameworku Next.js, který umožní uživateli číst blogové příspěvky zabývající se tématy nutrice a životosprávy, které jsou tvořeny a uloženy pomocí webové headless CMS aplikace Hygraph. Uživateli se nabízí možnost procházet hlavní stránku zahrnující veškeré příspěvky řazené podle data zveřejnění, nebo obsah filtrovat pomocí kategorií. Detailní zobrazení příspěvku, které lze vidět po rozkliknutí některého z odkazů hlavní stránky, také obsahuje seznam relevantních článků, které jsou doporučeny podle kategorie právě zobrazeného příspěvku, ten je ze seznamu vyřezen pomocí klíče. Dále má pak čtenář možnost zanechat komentář, který bude po kontrole správcem buďto zveřejněn, upraven, nebo smazán. Součástí aplikace je podpora NextAuth.js, a sice poskytovatelů Google a Github, pomocí nichž se lze přihlásit. Aplikace má responzivní design a lze ji tedy prohlížet na všech typech zařízení.

Klíčová slova: webová aplikace, React, Next.js, redakční systémy, API

OBSAH

ÚVOD

Životospráva je téma, o které se do jisté míry zajímá asi každý. Ať už jde o diety, sporty nebo třeba meditace, každý se snažíme udržet naše tělo a mysl v chodu. Já osobně sportuji celý život a v oblasti nutrice se pohybuji už čtvrtým rokem. V posledním roce jsem dokonce začal skromnou osobní praxi v podobě výživového poradentství. Bohužel lidé jsou zmatení. V dnešní době koluje na internetu ohromné množství falešných informací a běžnou praktikou se stává vykořisťování lidí, snažících se zlepšit.

Mým cílem bylo vytvořit platformu, která by třeba v budoucnu mohla sloužit jako místo, na které s čistým svědomím odkážu potencionální zájemce o mou radu, případně jako shromaždiště informací, které mi přijdou zkrátka zajímavé. Aplikace by tak měla obsahovat domovskou stránku, která nám dá možnost shlédnout příspěvky přehledně seřazeny podle data vydání. Dále pak možnost příspěvky třídit a případně podle kategorií a případně i sdílet myšlenky s ostatními čtenáři v rámci komentářové sekce. Žijeme v moderním světě a tudíž je aplikace responzivní a připravena na mobilní zařízení a tablety.

Dokumentace se v první kapitole zabývá těmi nejzákladnějšími, pro projekt klíčovými pojmy. V následujících kapitolách jsou poté uvedeny a popsány využité technologie, vysvětleno, proč byly upřednostněny před konkurencí a případně i uvedeny a porovnány klíčové rozdíly. V závěru je pak vysvětlen princip fungování vybraných funkcí.

1 WEBOVÁ FITNESS APLIKACE

Jednou ze základních myšlenek webové aplikace je oddělení dat od zbytku aplikační logiky. Toho lze docílit mnoha způsoby. Tento projekt popisuje možnost rozdělení aplikace na dvě dílčí části, a sice prezenční vrstvu, také známou jako front-end, a datovou vrstvu, neboli backend. Dále pak popisuje technologie, které k tomu lze použít. Tématikou celého webu je fitness, ke kterému mám osobně blízko. Výsledný obsah je zobrazován formou blogu.

1.1 Frontend

Prezenční vrstva má na starost zobrazení uživatelských výstupů, tzv. UI (*User Interface*). To mohou být formuláře, které uživatel vyplní, nebo pouze text, který čte. Uživatelské rozhraní je řešeno prostřednictvím React.js, což je relativně nový JavaScript framework, který práci na vzhledu usnadnil díky šikovným *komponentů*. Komunikaci v rámci prezenční vrstvy poté zajišťuje Next.js.

1.2 Backend

Také známý jako *strana serveru*, backend je část webové stránky, se kterou se běžný uživatel nesetká. Dává nám přístup k datům a zajišťuje fungování veškerých frontendem nabízených funkcí. Zde jej obstarává Headless CMS aplikace Hygraph, která je s projektem spojena pomocí endpointu uloženého v *.env* souboru. Ke komunikaci mezi vrstvami je použit GraphQL. To všechno běží na adrese *localhost:3000* prostřednictvím *Node.js*, což je asynchronní běhové prostředí.

2 BACKEND: VYUŽITÉ TECHNOLOGIE

2.1 Headless CMS

Název napovídá, že se jedná o systém, který odděluje prezenční vrstvu, neboli hlavu, od těla, v tomto případě chápeme pojmem tělo repositář s daty. Aby bylo možné takový systém použít, musíme vybudovat webovou stránku, nebo aplikaci a připojit CMS systém pomocí API endpointu. Posléze z něj lze pomocí dotazů čerpat data. Na rozdíl od tradičních CMS systémů, úzce spjatých s jedním konkrétním frontendem, se tak Headless varianta vyznačuje možností data spravovat a následně prostřednictvím API rozesílat do prezenční vrstvy bezpočtu aplikací.

2.2 Hygraph

Hygraph je Headless CMS aplikace, která slouží pro správu dat. Dává nám možnost vytvářet modely, kterým lze posléze přidat atributy. Tyto atributy mohou být např. titulek, popis, obrázek atd. Mezi modely můžeme tvořit vztahy, a sice *1:1*, *1:N* a *M:N*. Díky funkci rolí můžete jako vlastník projektu kontrolovat, který uživatel má jaké pravomoci.

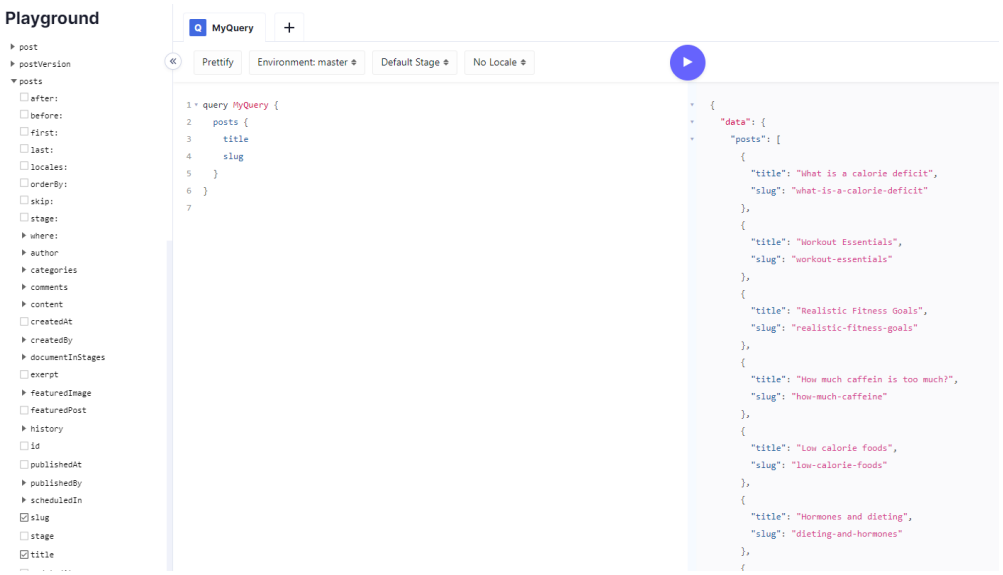
2.2.1 Role

Role určují vaše oprávnění v rámci projektu. Vlastník projektu má možnost uživatelské role měnit. Hygraph vám dává k dispozici 5 základních rolí, a sice:

- *Vlastník*, který má všechna oprávnění.
- *Admin* má možnost spravovat týmy a tvořit projekty.
- *Vývojář* může tvořit, měnit a mazat modely a vztahy mezi nimi.
- *Editor* je schopen smazat obsah.
- *Přispěvatel* může tvořit nový obsah.

2.2.2 API playground

Jedná se o funkci nabízející možnost vyzkoušet funkčnost API dotazů. Tyto dotazy lze napsat vlastnoručně anebo si pouze zaškrtnout, které data chci použít a playground mi jej vygeneruje.



Obrázek 2.1: Hygraph API playground ukázka

2.3 GraphQL

Původně vyvinut Facebookem, GraphQL je jazyk pro práci s daty v rámci databází a informačních systémů.Vyznačuje se použitím pouze jednoho endpointu. Mezi další přednosti patří funkce WYSIWYG *(What You See Is What You Get)*, která zajišťuje doručení pouze námi specifikovaných dat.

2.3.1 GraphQL vs. REST

Nelze jednoznačně určit lepší systém, neboť oba mají své plusy a mínusy. GraphQL je mladší, a jako takový se mohl poučit z nedostatků svého předchůdce. Tyto nedostatky zahrnují větší množství endpointů a chabou podporu mobilního vývoje, což GraphQL zvládl slušně. Na druhou stranu REST stále nabízí mnohem propracovanější systém odchytávání a zpracování chybových hlášení.

2.3.2 Přehled rozdílů GraphQL a REST API

GraphQL	REST API
Má pouze jeden endpoint.	Počet endpointů je spjat s počtem dotazů.
Dodá jen a pouze chtěná data.	Tendence doručit nadbytek/nedostatek dat.
Vrací pouze HTTP kód 200.	Vrací mnoho HTTP zpráv.
Jeden dotaz může pracovat s více objekty.	Každý objekt potřebuje vlastní dotaz.
Nový, zatím nepříliš implementovaný.	Velmi používaný a přijímaný.

2.3.3 Ukázka GraphQL dotazu

Podobně jako u práce s adresářem se i zde vnořuji do datové struktury. Nejprve otevřu objekt *__schema*, v něm *types*, ze kterých již vybírám konkrétní data.

Dotaz

```
{
  __schema {
    types {
      name
      description
    }
  }
}
```

Odpověď

```
{
  "data": {
    "__schema": {
      "types": [
        {
          "name": "String",
          "description": "…"
        }, …
```

2.4 Docker

Docker uzavírá procesy do tzv. kontejnerů, které zlehčují zprovoznění projektu na cizím zařízení a zároveň zvyšují bezpečnost. Fungují jako velmi malé virtuální stroje a nabízí nám tak možnost upravit jednu část aplikace, zatímco zbytek je dále funkční.

3 FRONTEND: VYUŽITÉ TECHNOLOGIE

3.1 React

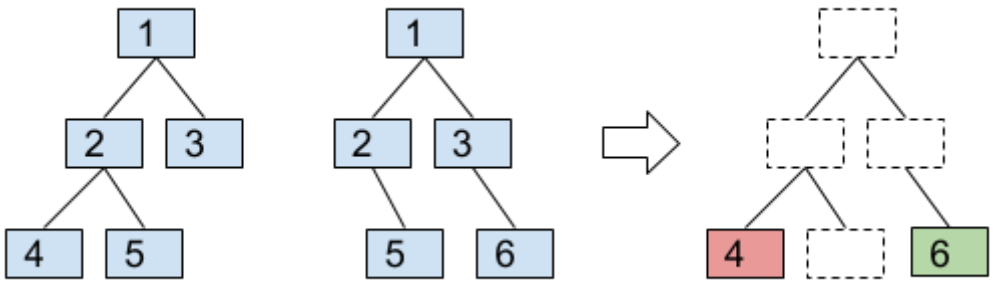
React je JavaScript framework zaměřující se na tvorbu uživatelského rozhraní. Vyznačuje se velmi efektivním kódem a vysokým výkonem. React se píše ve formátu JS, případně JSX, což je syntaktické rozšíření umožňující kombinovat HTML a JS v jediném souboru.

3.1.1 Next.js

Next.js je React framework zaměřující se na tvorbu webových aplikací. Nabízí širokou škálu funkcí jako např. routing, data fetching[1] a autentifikaci. Velmi vhodný pro projekt, který již má zakomponovaný React pro UI. Při vývoji byla použita verze Next.js 13.

3.1.2 Virtual Document Object Model (DOM)

DOM zachází se soubory XML a HTML jako se stromovou strukturou, složenou z objektů. Pokud v této struktuře dojde ke změně, varianta Real DOM obnoví veškeré objekty. Na druhou stranu varianta Virtual DOM nejprve porovná předchozí stav struktury s tím současným a poté obnoví pouze ty objekty, které byly pozměněny.



Obrázek 3.1: Ukázka fungování VDOM

[1] Získání a zobrazení dat.

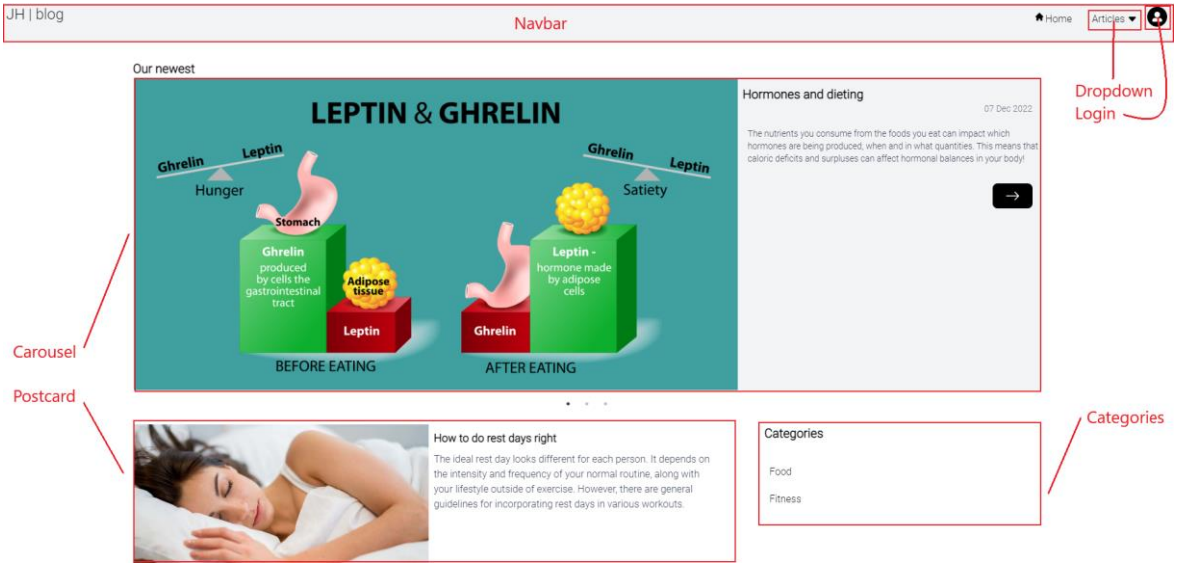
3.1.3 React Router

Knihovna, která se stará o pohyb uživatele mezi jednotlivými stránkami projektu. Ověří zdali uživatelem zapsaná URL adresa odpovídá některé z již uložených adres v router souboru, pokud ano, uživatel je přesměrován.

3.1.1 Komponenty

Komponenty jsou stavebními bloky React projektu. Vznikají poté co vytvořím a naplním kódem JS nebo JSX soubory, které následně importuji a vyvolám v hlavním JS souboru aplikace. Každý z těchto komponentů je znovu použitelný, lze v něm vyvolat jiný komponent a přijímá tzv. props. Perfektní příkladem pro komponent je např. header nebo footer.

```
const Footer = () => { //Pojmenování
  return(              //Komponent musí vždy obsahovat return
    <div>              //Kód, který už je překládám a vypsán na stránku
      Hello world!!
    </div>
  )
}
export default Footer //Komponenty vždy končí exportem
```



Obrázek 3.2: Ukázka komponentů

3.1.2 React Properties

Jedná se o proměnné, známé také jako Props ,které nám dávají možnost předávat data napříč komponenty.

```
//Deklaruji komponent PostCard, který bude pracovat s příspěvky. Tyto pří-
spěvky jsou zatím zastoupeny objektem post, který při vyvolání nahradím daty
const PostCard = ({post}) => {...
//Vyvolávám komponent PostCard a jako props „post“ nastavuji post.node
<PostCard post={post.node}  key={post.title}/>
```

3.1.3 React-icons

Jedná se o relativně malou knihovnu ikon, které lze v projektu vyvolat formou komponentu, což nám dává možnost je dále stylizovat a udržet tak jednotný design webu.

```
//Komponent BsFillCaretDownFill se zobrazí jako malá šipka, ml-1 nastaví
margin left na 1 stupeň a translate-y-1 ji posune na ose y o 1 stupeň dolů.
Articles <BsFillCaretDownFill className='ml-1 translate-y-1'/>
```

3.1 Tailwind

Tailwind je velmi flexibilní utility-first[2] CSS framework, který díky absenci přednastavených tříd, jako tomu je u Bootstrapu, nabízí možnost tvořit design více na míru. Aplikace je založena na verzi 3.2.

3.1.1 Ukázka Tailwind

```
//Md zajistí, že výška 80px a zrušení zaoblení se aplikuje pouze v případě,
že velikost zařízení je větší nebo rovna medium, tzn. tablety, notebook, PC.
//Hover funguje tak, jak čekáme, a sice pokud div zaregistruje kurzor, na-
staví barvu pozadí na světle šedou.
//Mb-12 nastaví margin bottom na 12px.
<div className='md:h-80 mb-12 rounded-lg md:rounded-none hover:bg-gray-100'>
```

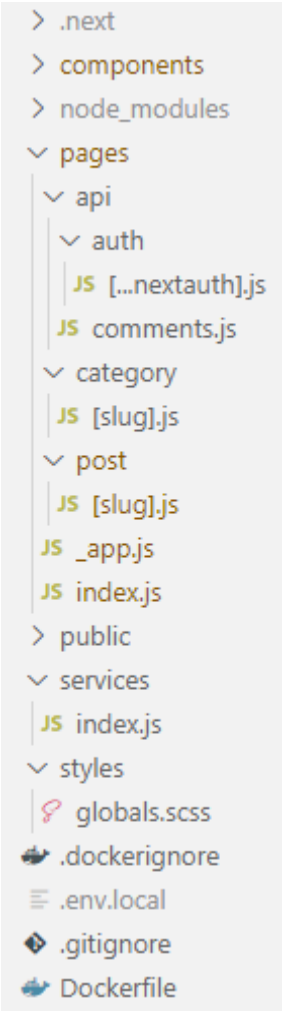
[2] Složen z mnoha nízko úrovňových grafických komponentů.

4 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

4.1 Tvorba Next.js projektu

K založení Next.js projektu je zapotřebí mít Node.js verzi 14.6.0 a výš. Po otevření našeho vývojového prostředí zapíšeme do konzole buď *npx create-next-app@latest*, nebo *yarn create next-app*. Posléze nainstalujeme react, next a react-dom, čehož docílíme pomocí příkazu *npm install react next react-dom* nebo *yarn add react next react-dom*. Teď už jen stačí pomocí *npm run dev* nebo *yarn dev* spustit aplikaci a navštívit *http://localhost:3000*

4.2 Adresářová struktura

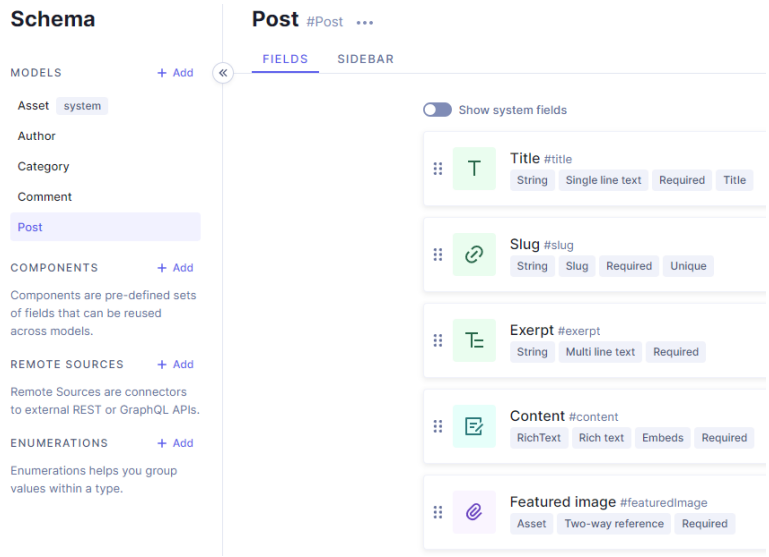


- Složka **components** obsahuje veškeré komponenty, jako například Carousel, Navbar, PostCard.
- **[…nextauth.js]** je soubor obsahující informace potřebné k autorizaci Next.js.
- **Comments.js** je náš endpoint pro odesílání dotazů.
- **[slug.js]** je proměnná, jejíž jméno je nahrazeno slugem konkrétního příspěvku, na který uživatel klikne.
- **_app.js** je hlavní soubor celého projektu.
- Do **index.js** zapisuji věci, které uživatel vidí po spuštění.
- **Services/index.js** obsahuje všechny GraphQL dotazy.

Obrázek 4.1:
Adresářová struktura

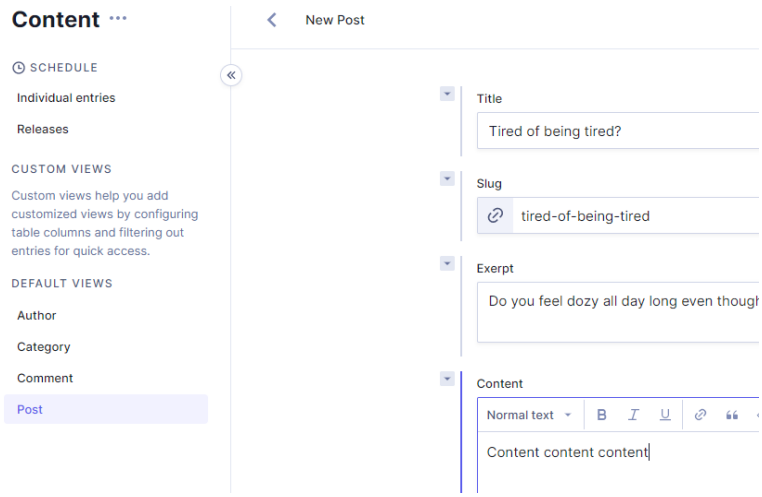
4.3 Správa dat

Data spravuji přes webovou aplikaci Hygraph. Zde vytvořím model, kterému dám určité atributy. Mezi modely lze podle potřeby vytvořit vztahy.



Obrázek 4.2: Ukázka Hygraph schématu

Pokud chci do takto nachystaného modelu přidat data, stačí jen přejít na sekci Content a zde už můžu snadno vidět, jaké informace jsou nutné a jaké dobrovolné.



Obrázek 4.3: Ukázka přidávání obsahu

Takto vyplněný obsah poté můžu buď uložit jako draft, anebo přímo publikovat. Záleží na uživatelské roli.

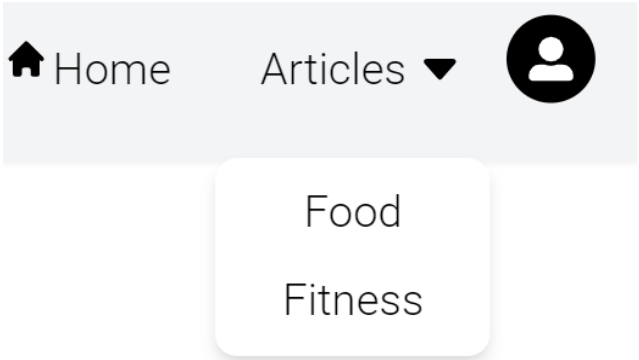
4.4 Práce s daty a jejich aplikace

4.4.1 Načtení dat

Po publikování obsahu autorem stačí jen data samotná vypsat do prezenční vrstvy. K tomu byl použit GraphQL. Pomocí metody GET můžu přesně specifikovat jaká data mě zajímají a vytvořit specifické dotazy pro specifické úkoly. Například komponent vypisující jméno kategorie nepotřebuje datum, kdy byla kategorie vytvořena. Napíšu proto dotaz, ve kterém si zažádám pouze jméno a slug.

```
Nejprve musíme napsat v services/index.js dotaz.
//Výpis jmen a slugů kategorií.
//React užívá asynchronní funkce, která dokáží řešit více dotazů najednou
// - nečekají na splnění předchozího, aby zahájily nový.
export const getCategories = async () => {
  const query = gql`
        query GetCategories{
            categories {
                name
                slug
            }
        }
    `
  const result = await request(graphqlAPI, query); //Čekání na odpověď
  return result.categories; //Návratová funkce
}

Tento dotaz poté vypíšeme v components/Categories.jsx
//Funkce zajišťující převedení dat v dotazu za pomocí useState
//Do 'categories' budou uložena data, která jsme získali GQL dotazem.
const Categories = () => {
  const [categories, setCategories] = useState([]);
  useEffect(() => {
    getCategories()
      .then((newCategories) => setCategories(newCategories))
  }, []);
//Mapování skrze pole categories. Výsledek bude výpis jmen kategorií.
      {categories.map((category) => (
        <Link key={category.slug} href={`/category/${category.slug}`}>
          <span>
            {category.name}
          </span>
        </Link>
      ))}
```

Takový komponent by nám tedy vypsal jména všech kategorií. Mapování skrze takové pole nám poté dává ideální příležitost pro vytvoření např. dropdown menu se jmény všech našich kategorií.



Obrázek 4.4: Výpis jmen kategorií

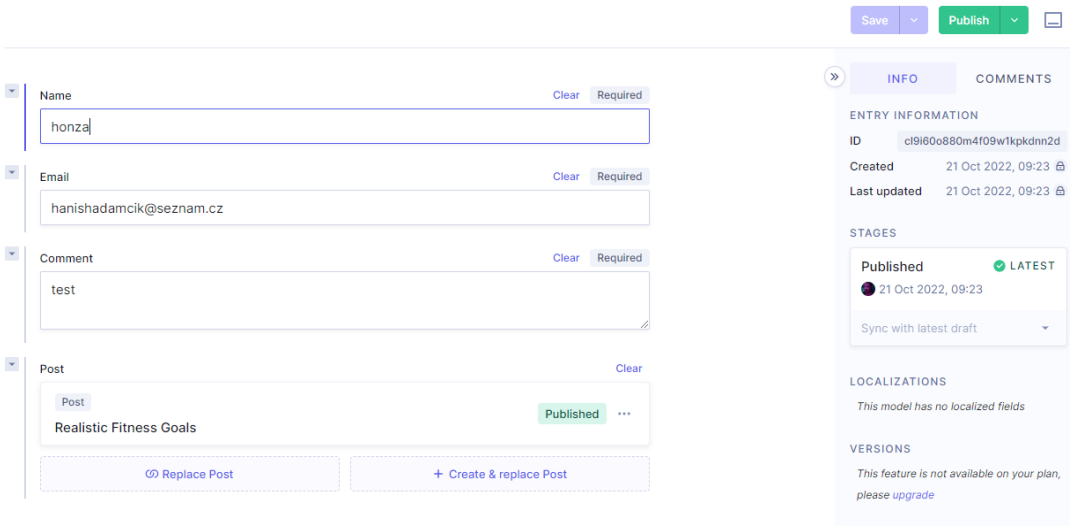
4.4.2 Odesílání dat

Odesílání dat je řešeno metodou POST. Pokud se uživatel rozhodne napsat komentář, vyplní email, uživatelské jméno a myšlenku, kterou chce sdělit. Tato data jsou posléze přes náš vlastní endpoint a za pomocí vygenerovaného tokenu zaslána do Hygraphu.

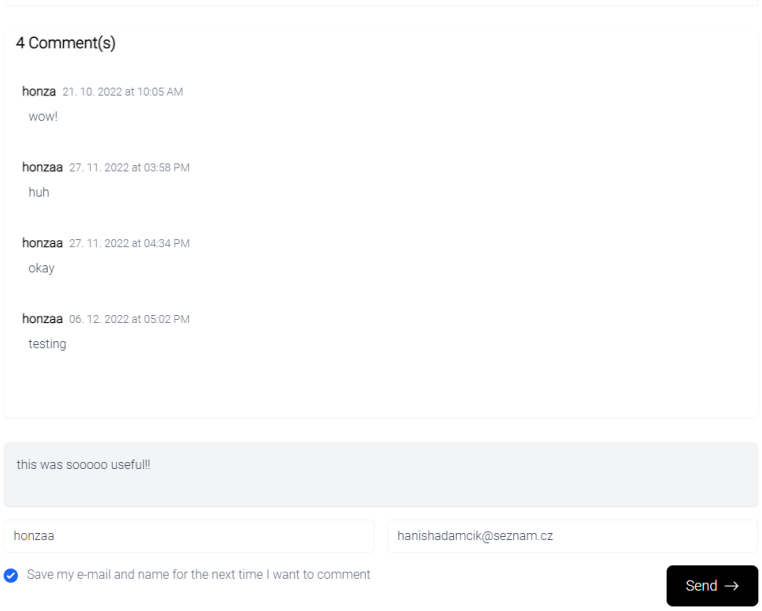
```
services/index.js
//Metodou POST odešle komentář ve formátu JSON do souboru comments.js,
//což je endpoint, přes který jsou data poslána do Hygraphu.
export const submitComment = async(obj) => {
  const result = await fetch('/api/comments', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(obj),
  });

  return result.json();
}
/api/comments.js
//Mutation query mi dává možnost vytvářet nová data
//String! Nastaví datový typ atributu
//Za pomoci slugu poté připojím komentář ke konkrétnímu příspěvku
  const query = gql`
    mutation CreateComment($name: String!, $email: String!, $comment:
String!, $slug: String!) {
```

```
      createComment(data: {name: $name, email: $email, comment: $comment,
post: {connect: {slug: $slug}}}) { id }
    }
  `
```

My poté máme možnost prostřednictvím Hygraphu rozhodnout, jestli se nám komentář líbí, nebo ne. V případě, že se rozhodneme komentář publikovat stane se viditelným pro běžného uživatele.



Obrázek 4.5: Náhled komentáře před zveřejněním



Obrázek 4.6: Výsledný komentář

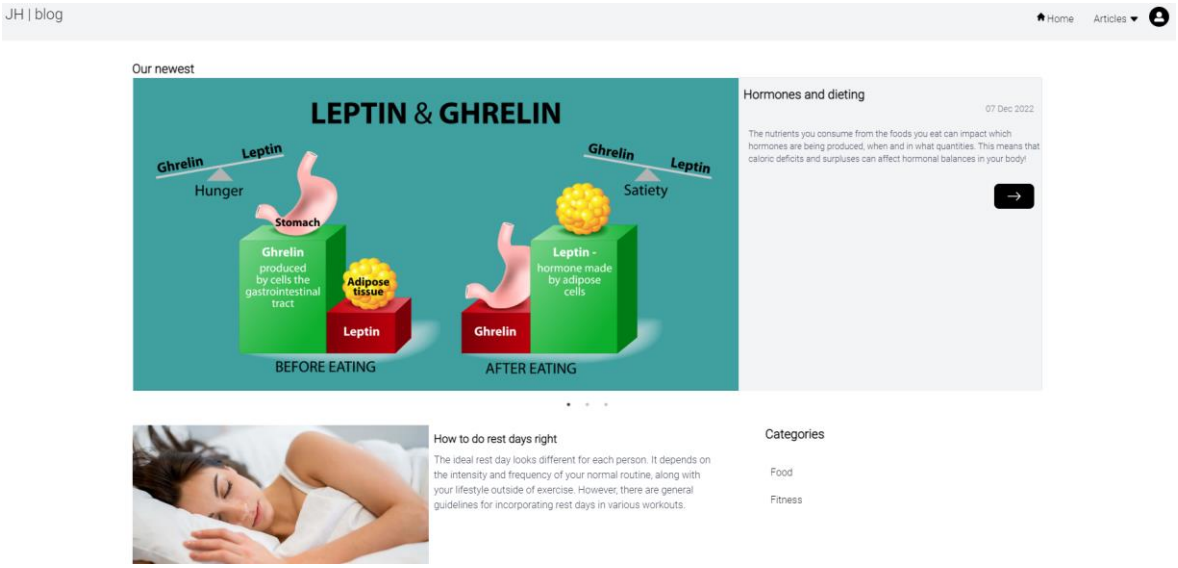
4.5 Dílčí prvky stránky

Součástí každé stránky je navbar, hlavička, která obsahuje jméno webu a odkazu na jednotlivé podstránky webu. Uživatel zde má možnost vybrat kategorii příspěvků, které ho zajímají a případně se přihlásit prostřednictvím NextAuth.js.

Obrázek 4.7: Navbar

4.5.1 Domovská stránka

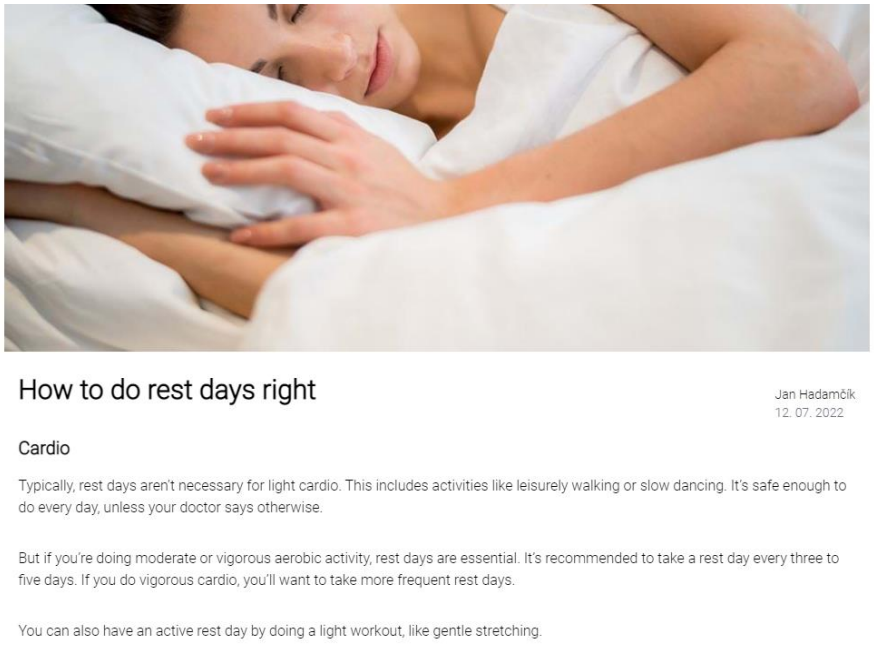
Stránka, kterou uživatel vidí bezprostředně po spuštění webu. Obsahuje posuvný seznam, což je komponent, s jehož pomocí lze vidět tři nejnovější příspěvky. Pod tímto seznamem jsou poté seřazeny ostatní starší příspěvky.



Obrázek 4.8: Domovská obrazovka

4.5.2 Detail příspěvku

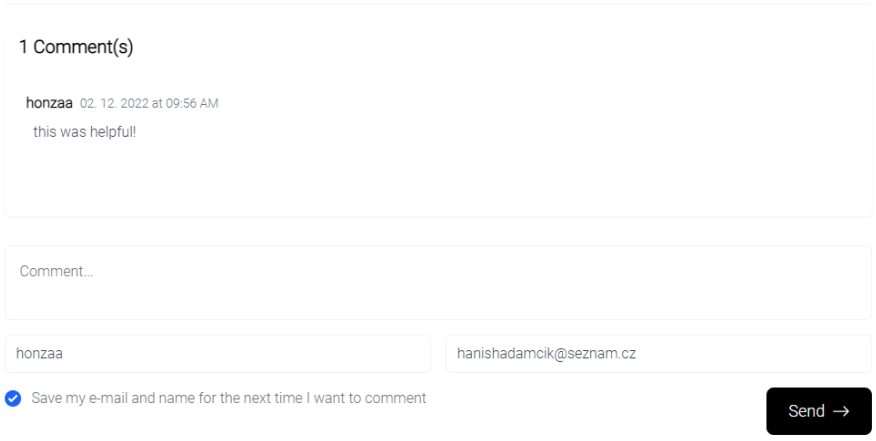
Detail příspěvku se zobrazí po rozkliknutí odkazu článku. Obsahuje obrázek, popis, jméno autora a kategorie. Samotný text článku je poté následován seznamem relevantních příspěvků, které jsou doporučovány na základě kategorie příspěvku právě čteného.



Obrázek 4.9: Detailní zobrazení příspěvku

4.5.3 Komentářová sekce

Obsahuje seznam komentářů, které jsou řazeny dle data přidání. Uživatel má možnost sdílet své názory po vyplnění jména, emailu a samotného textu myšlenky. Tento komentář je posléze zaslán na kontrolu, kterou provádí autor příspěvku prostřednictvím Hygraphu. Pokud autor shledá komentář nezávadným, má možnost ho zveřejnit, čímž se stane viditelným pro běžného uživatele.



Obrázek 4.10: Komentářová sekce

ZÁVĚR

Projekt vznikl jako záminka pro studium Reactu a jeho cílem bylo vytvořit jednoduchou webovou aplikaci, pomocí které bych mohl jakožto admin přidávat obsah formou blogových příspěvků. Tyto příspěvky by byly dodávány prostřednictvím Headless CMS aplikace Hygraph a v rámci komentářové sekce se mi povedlo komunikaci učinit oboustrannou a mimo vyvolávání obsah také posílat. Celou aplikaci jsem posléze zabalil pomocí Dockeru.

V budoucnu bych se k aplikaci rád vrátil a znova se ponořil do vývoje a zapracoval na nesplněných cílech mé práce, a sice správnou aplikaci NextAuth.js, který je v současném stavu spíše hračka než cokoliv jiného. Dále bych rád předělal design a bral více ohled na mobilní zařízení, jelikož ke konci vývoje mi bylo jasné, že prvky psané na začátku jsou krajně nešikovné.

Aplikaci lze nalézt na adrese *https://github.com/it1906/hadamcik_blog*

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] *Comparing Tailwind CSS to Bootstrap: Is it time to ditch UI kits?* [online]. [cit. 2023-01-02]. Dostupné z: https://blog.logrocket.com/comparing-tailwind-css-bootstrap-time-ditch-ui-kits/

[2] *Headless CMS explained in 5 Minutes* [online]. [cit. 2023-01-02]. Dostupné z: https://hygraph.com/academy/headless-cms

[3] *GraphCMS* [online]. [cit. 2023-01-02]. Dostupné z: https://www.getapp.com/website-ecommerce-software/a/graphcms/features/

[4] *What is Next.js?* [online]. [cit. 2023-01-02]. Dostupné z: https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs

[5] *The Best Guide to Know What Is React* [online]. [cit. 2023-01-02]. Dostupné z: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

[6] *Frontend vs Backend* [online]. [cit. 2023-01-02]. Dostupné z: https://www.geeksforgeeks.org/frontend-vs-backend/

[7] *Node.js Server-Side JavaScript – What is Node Used For?* [online]. [cit. 2023-01-02]. Dostupné z: https://www.freecodecamp.org/news/node-js-server-side-javascript-what-is-node-used-for/#:~:text=js%3F-,Node.,the%20browser%20%E2%80%94%20on%20the%20server.

[8] *Build and Deploy THE BEST Modern Blog App with React | GraphQL, NextJS, Tailwind CSS* [online]. In: . [cit. 2023-01-02]. Dostupné z: https://www.youtube.com/watch?v=HYv55DhgTuA&t=9987s&ab_channel=JavaScriptMastery

[9] *Add Auth to your React Project EASILY Tutorial* [online]. In: . [cit. 2023-01-02]. Dostupné z: https://www.youtube.com/watch?v=zkz2HuNR_gk&ab_channel=developedbyed

[10] *React Crash Course - Build A Full Recipe App Tutorial* [online]. In: . [cit. 2023-01-02]. Dostupné z: https://www.youtube.com/watch?v=xc4uOzlndAk&ab_channel=developedbyed

[11] *Vercel: Deployment* [online]. [cit. 2023-01-02]. Dostupné z: https://nextjs.org/docs/deployment

[12] *Tailwindcss: Documentation* [online]. [cit. 2023-01-02]. Dostupné z: https://tailwindcss.com/docs