

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Rezervační systém HealthSync



Autor: Ondřej Němčík
Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování
Třída: IT4
Školní rok: 2024/25

Poděkování

Rád bych poděkoval panu učiteli Mgr. Marku Lučnému a Ing. Petru Grussmannovi za pomoc s projektem a poskytnutí rad a poznatků.

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. 2025

.....
Podpis autora

Abstrakt

Výsledkem projektu je funkční rezervační systém pro rezervaci lekcí. Stránka zahrnuje registraci uživatelů pomocí OAuth přes Google, také standartní přihlášení přes uživatelské jméno a heslo. Po prvním přihlášení je uživatel vyzván doplnit své údaje jako jsou věk, telefonní číslo a adresa místa bydliště. Uživatelé se přihlašují na lekce pomocí kalendáře zobrazujícího aktuální týden. Mohou vidět historii ale i nadcházející lekce a také si aktualizovat své osobní údaje. Primární je však pohled administrátora, který má možnost spravovat veškeré uživatelské údaje, plus možnost vidět extra údaje jako je typ permanentky atd. Dále možnost vytvářet treninky, místnosti popřípadě manuální rezervace klienta na určitou lekci. Všechna tato data jsou pomocí backendu Strapi ukládána do databáze.

Klíčová slova

rezervační systém, uživatelské účty, závěrečná práce, OAuth, Strapi, databáze, backend ...

Obsah

Úvod	6
1 Rezervační systém a problematika	7
1.1 Co je rezervační systém	7
1.1.1 Výhody používání rezervačních systémů	7
1.2 Architektura databázového modelu	8
1.2.1 Relační databáze	8
1.2.2 Normalizace	8
1.2.3 Podpora více relací, Flexibilita a rozšiřitelnost	8
1.2.4 Flexibilita a rozšiřitelnost	8
1.2.5 Diagram	8
1.3 Zkušenosti s tvorbou databází	9
2 Využité technologie	10
2.1 Frontend: Uživatelské rozhraní	10
2.1.1 Next.js	10
2.1.2 CSS Modules	10
2.2 Backend: Správa dat a autentizace	10
2.2.1 Strapi CMS	10
2.2.2 Node.js	10
2.3 Databáze	11
2.3.1 SQLite	11
2.4 Autentizace a autorizace	11
2.4.1 OAuth 2.0 a JWT	11
2.5 Další technologie	11
2.5.1 Axios - Fetch API	11
2.5.2 JWT Decode	11
3 Způsoby řešení a použité postupy	12
3.1 Backend	12
3.1.1 Správa dat	12
3.1.2 API pro komunikaci	13
3.1.3 Autentizace a autorizace	14
3.2 Frontend	15
3.2.1 Přihlašovací a registrační systém	15

3.2.2	Hlavní stránka	17
3.2.3	Administrace	19
4	Výsledky a zhodnocení	21
4.0.1	Splněné cíle	21
4.0.2	Budoucí rozvoj	21
4.0.3	Zhodnocení	22

ÚVOD

Výběr závěrečného projektu byl pro mě velice náročný. Hledal jsem takový projekt, kde bych si mohl rozšířit obzory a naučit se něco nového. Zároveň uplatnit znalosti webů a webových aplikací.

Dlouho jsme se v práci potýkali s omezenou možností zrychlit rezervaci treninků a tak ulehčit recepčním práci i zdraví. V ten moment jsem se rozhodl právě pro toto téma mého závěrečného projektu.

Mým cílem bylo vytvořit webovou aplikaci se zobrazením lekcí dostupných pro naše klienty. Celkově přehledný a jednoduchý rezervační systém. Aplikace obsahuje přihlášení i registraci pomocí google, popřípadě klasickou možnost přihlášení. Lekce se zobrazují v kalendáři, kde má možnost uživatel si zobrazit detailní informace o dané lekci. Hlavním účelem aplikace je spíše část administrátorská, možnost úpravy profilu klenta, zobrazení přihlášených uživatelů na lekci, úprava a vytváření lekcí, místností. To celé v přehledném jednoduchém responzivním designu.

V dokumentaci budeme postupně řešit postup vytváření celé aplikace až po současnou dobu. Probereme si použité technologie, budoucí cíle a možnosti rozšíření.

1 REZERVAČNÍ SYSTÉM A PROBLEMATIKA

1.1 CO JE REZERVAČNÍ SYSTÉM

Rezervační systém je druh informačního systému, jehož primárním účelem je přesně evidovat rezervace a dostupnost libovolných komodit v reálném čase. Komoditou v naše případě je lekce, kterou si uživatel zarezervuje. Rezervační systém poskytuje informace o vytížení komodity a zabraňuje přečerpání kapacit. Hlavním nástrojem je rezervační formulář. Historicky vznikly v letecké a železniční dopravě a následně našly uplatnění v ubytovacích a dalších službách cestovního ruchu a později pronikly i do ostatních oborů.

1.1.1 Výhody používání rezervačních systémů

Hlavní výhodou rezervačního systému je jeho schopnost zefektivnit správu času. Umožňuje uživatelům rezervovat služby nebo zdroje bez nutnosti manuálního zásahu zaměstnanců. Mezi další výhody patří:

- Eliminace překryvů: Systém automaticky kontroluje dostupnost a zabraňuje dvojím rezervacím, což minimalizuje chyby.
- Dostupnost 24/7: Rezervace lze provádět kdykoliv, aniž by bylo nutné kontaktovat pracovníky
- Přehledné rozhraní: Uživatelé mohou snadno vidět dostupné termíny, stav rezervace nebo historii svých rezervací.

1.2 ARCHITEKTURA DATABÁZOVÉHO MODELU

Databázový model systému HealthSync je navržen s cílem efektivně ukládat, spravovat a dotazovat data spojená s rezervačním systémem. Architektura modelu kombinuje normalizovaný relační design a robustní strukturu vztahů, která umožnuje jednoduché rozšiřování a flexibilitu při správě dat. Tento model podporuje všechny klíčové funkce systému a zajišťuje datovou konzistenci a integritu.

1.2.1 Relační databáze

Databáze je postavena na relačním modelu, což znamená, že data jsou uspořádána v tabulkách (entitách) s jasně definovanými sloupci (atributy). Každá tabulka reprezentuje specifickou entitu, např. Lekce, Osoby nebo Rezervace.

1.2.2 Normalizace

Návrh databázového modelu dodržuje principy normalizace, čímž se minimalizuje redundance dat a zvyšuje efektivita úložiště. Klíčové tabulky obsahují primární klíče, které jsou používány k propojení tabulek prostřednictvím cizích klíčů.

1.2.3 Podpora více relací, Flexibilita a rozšířitelnost

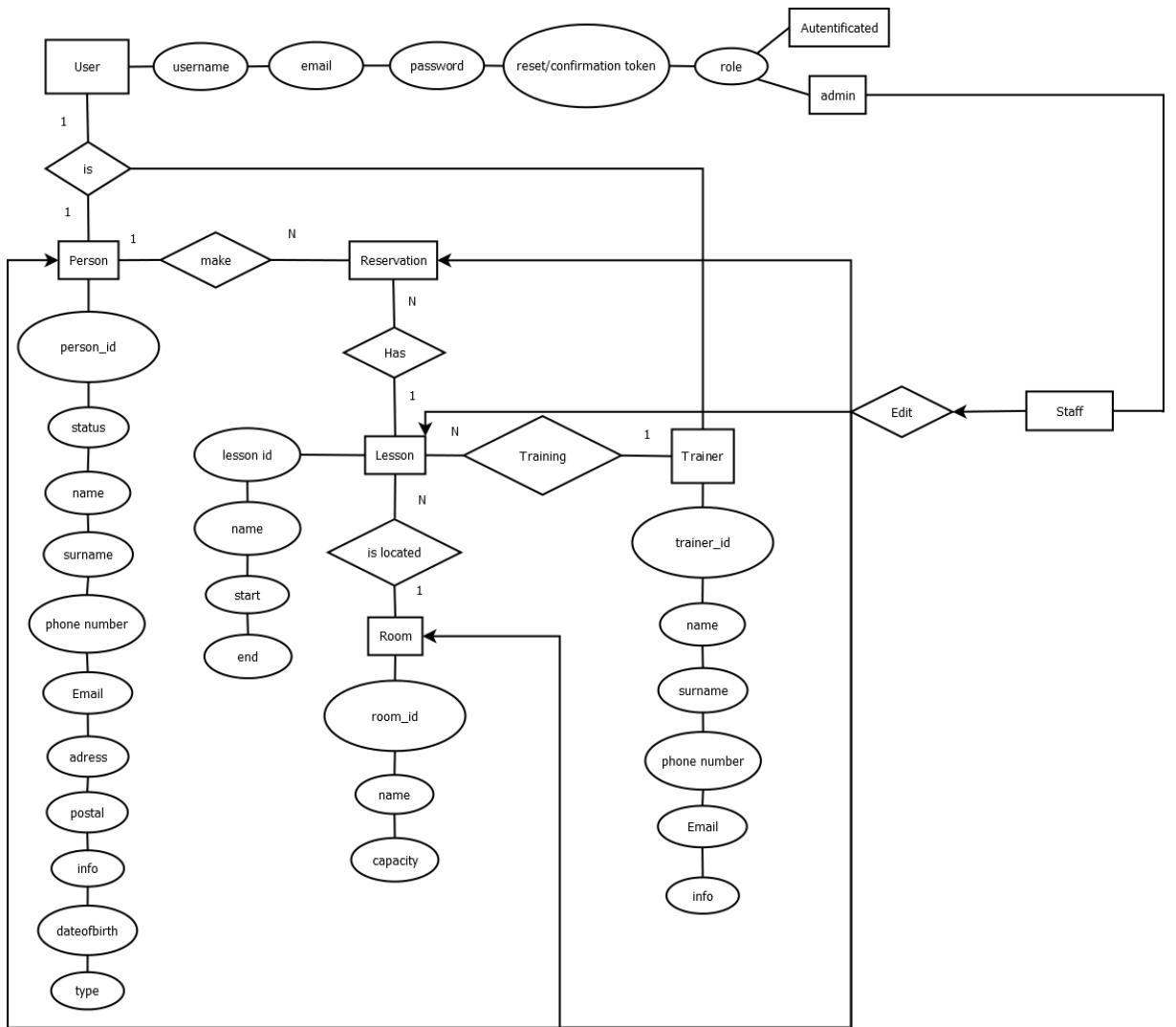
Databázový model využívá 1:N a M:N vztahy, které umožňují propojení různých entit podle potřeb systému (např. jedna lekce může mít více rezervací, jedna osoba může být rezervována na více lekcí).

1.2.4 Flexibilita a rozšířitelnost

Model je navržen tak, aby mohl být snadno rozšířen o další tabulky, atributy nebo vztahy bez zásadní změny stávající struktury.

1.2.5 Diagram

Diagram představuje propojení jednotlivých datových modelů, vzájemné vztahy a celkový pohled na strukturu databáze.



Obrázek 1.1: ER diagram

1.3 ZKUŠENOSTI S TVORBOU DATABÁZÍ

Prbolemataka databázových aplikací pro mě není probblémem. Jelikož tento projekt vychází z aplikace v Djangu ze třetího ročníku. Dále jsem vycházel ze zkušeností se správou databáze MySQL ve webovém rozhraní phpMyAdmin.

2 VYUŽITÉ TECHNOLOGIE

V projektu HealthSync byly využity následující technologie, které zajišťují funkčnost systému, jeho škálovatelnost, bezpečnost a uživatelskou přívětivost:

2.1 FRONTEND: UŽIVATELSKÉ ROZHRANÍ

2.1.1 Next.js

Next.js rozšířil možnosti Reactu o server-side rendering (SSR) a generování statických stránek, což vedlo ke zlepšení výkonu aplikace a SEO. Integrované API routy umožnily jednodušší implementaci komunikace s backendem.

2.1.2 CSS Modules

CSS Modules byly použity k modularizaci a izolaci stylů, což zamezilo konfliktům mezi různými částmi aplikace.

2.2 BACKEND: SPRÁVA DAT A AUTENTIZACE

2.2.1 Strapi CMS

Strapi byl nasazen jako hlavní backendový systém díky své jednoduchosti, možnosti rychlé konfigurace a nativní podpoře REST API i GraphQL. Zajišťuje správu entit, jako jsou uživatelé, lekce, rezervace a místnosti. Funkce autentizace a oprávnění umožnily snadnou správu přístupu k datům.

2.2.2 Node.js

Základ backendové logiky a runtime prostředí, které umožňuje škálovatelnost a efektivní zpracování požadavků.

2.3 DATABÁZE

2.3.1 SQLite

SQLite je lehká a snadno použitelná databáze, která nevyžaduje konfiguraci serveru. Je vhodná pro projekty s nižšími nároky na paralelní zpracování dat a pro aplikace, kde je kladen důraz na rychlou a bezproblémovou správu databázových souborů.

2.4 AUTENTIZACE A AUTORIZACE

2.4.1 OAuth 2.0 a JWT

K zajištění bezpečné autentizace byl použit OAuth 2.0 pro přihlašování pomocí externích služeb (např. Google). JSON Web Tokens (JWT) slouží k uchování uživatelské relace a k autorizaci přístupů k chráněným zdrojům.

2.5 DALŠÍ TECHNOLOGIE

2.5.1 Axios - Fetch API

Axios a nativní Fetch API byly využity pro komunikaci mezi frontendem a backendem. Tyto knihovny umožnily snadnou práci s REST API.

2.5.2 JWT Decode

Pro dekódování a ověřování JWT tokenů byl použit balíček jwt-decode, což usnadnilo správu uživatelských relací.

3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

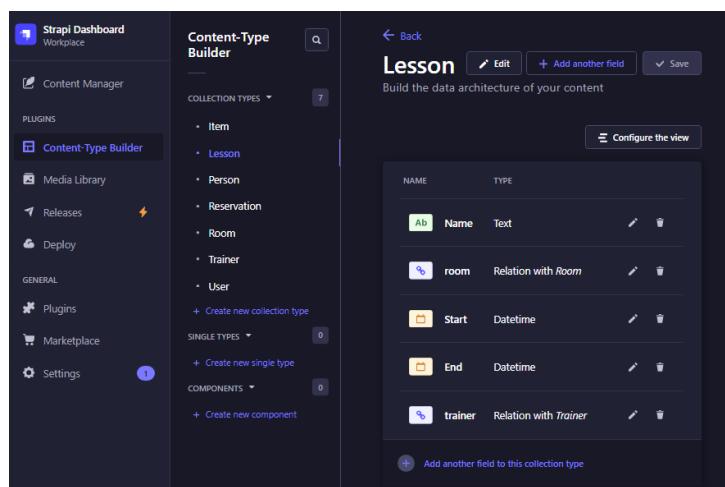
3.1 BACKEND

Pro backend aplikace jsem použil Node.js společně s frameworkm Strapi, což je headless CMS, který umožnuje snadné vytváření REST API a spravování obsahu. Backend jsem navrhl tak, aby zajišťoval následující funkce:

3.1.1 Správa dat

Next.js rozšířil možnosti Reactu o server-side rendering (SSR) a generování statických stránek, což vedlo ke zlepšení výkonu aplikace a SEO.

- Vytvoření entit pro lekce, uživatele, místo a rezervace.
- Každá entita obsahuje relevantní atributy, například:Lekce: název, čas zahájení, čas ukončení, místo, trenér.



Obrázek 3.1: Struktura modelu

3.1.2 API pro komunikaci

Implementace REST API endpointů umožňujících CRUD operace (vytvoření, čtení, úprava, mazání). Endpoints pro správu lekcí, rezervací a uživatelů, například: /api/lessons pro manipulaci s lekcemi. Pomocí parametrů "?populate=trainer" například vypíše navazující data o trenérovi.

```
1  "data": [
2      {
3          "id": 5,
4          "attributes": {
5              "Name": "lesson 2",
6              "locale": "cs-CZ",
7              "Start": "2024-09-21T18:00:00.000Z",
8              "End": "2024-09-21T19:00:00.000Z",
9              "trainer": {
10                  "data": {
11                      "id": 2,
12                      "attributes": {
13                          "name": "trainer",
14                          "surname": "1",
15                          "locale": "cs-CZ",
16                          "email": "trainer2@test.test",
17                          "info": [
18                              {
19                                  "type": "paragraph",
20                                  "children": [
21                                      {
22                                          "type": "text",
23                                          "text": "geraegyčgyshszSGyhzger"
24                                      }
25                                  ]
26                              }
27                          ],
28                          "telephone": "+420 132 789 654"
29                      }
30                  }
31              },
32              "room": {
```

```

33     "data": {
34         "id": 1,
35         "attributes": {
36             "Name": "Funkční zóna",
37             "Capacity": 30,
38         }
39     }
40 }
41

```

Kód 3.1: Ukázka API

3.1.3 Autentizace a autorizace

Použití JWT (JSON Web Token) pro zabezpečení API. Uživatelé mohou být registrováni a přihlašováni buď přes Google OAuth, nebo pomocí tradičního přihlašovacího mechanismu. Ochrana endpointů podle rolí (administrátor má přístup k širším funkcím než klient).

```

1 // ukladání jwt tokenu pro přístup k api requestům
2 try {
3     const res = await
4         fetch(`#${process.env.NEXT_PUBLIC_API_URL}/api/auth/local`,
5     {
6         method: 'POST',
7         headers: {
8             'Content-Type': 'application/json',
9         },
10        body: JSON.stringify({
11            identifier, // email nebo username
12            password,
13        }),
14    });
15    const data = await res.json();
16
17    if (res.ok) {
18        // Ukládání JWT, userId po přihlášení
19        localStorage.setItem('jwt', data.jwt);
20        localStorage.setItem('userId', data.user.id);

```

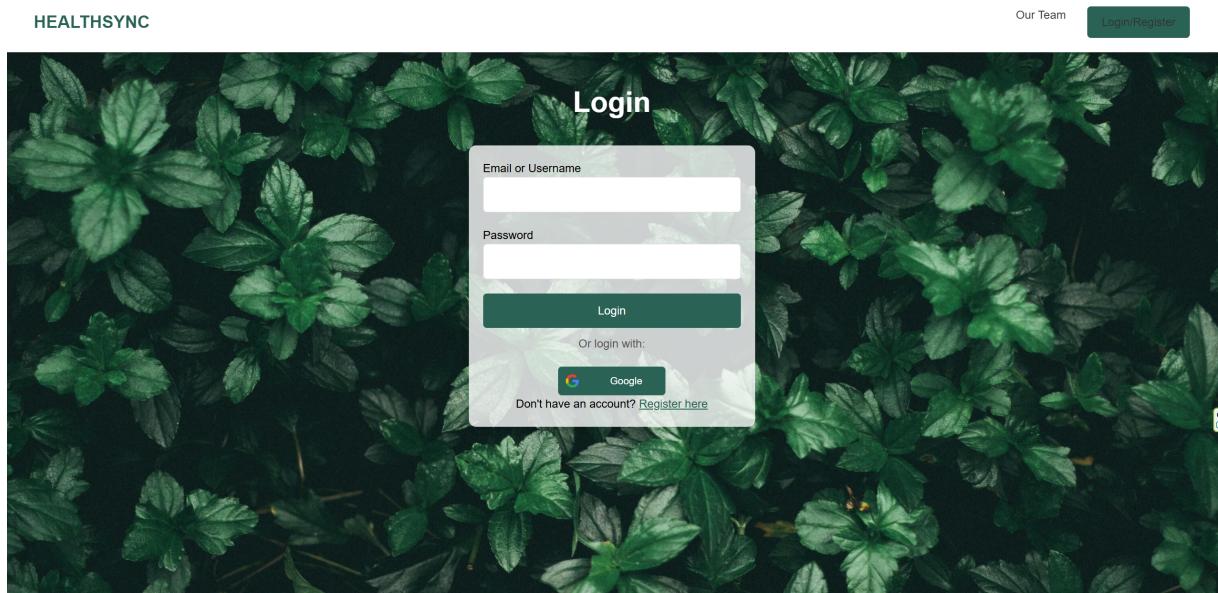
Kód 3.2: JWT přístup

3.2 FRONTEND

Frontend byl postaven s použitím Next.js, moderního frameworku pro React.js, který umožňuje server-side rendering a optimalizovaný výkon.

3.2.1 Přihlašovací a registrační systém

Uživatelé se mohou registrovat a přihlašovat pomocí Google OAuth nebo e-mailu. Po přihlášení získávají uživatelé přístup k odpovídajícím funkcím podle svého typu (klient nebo administrátor).



Obrázek 3.2: Přihlašovácí stránka

Přihlášení klasickým způsobem bylo řešeno přímo pomocí Strapi, kterému byly předána pouze data od uživatele při registraci. Při přihlášení přes OAuth nastal problém. Zde muselo být dořešeno přihlášení přes OAuth, Strapi v základu automaticky nevytvoří uživatele, pouze si vezme jeho data z Googlu. Bylo nutné dořešit funkci pro vytvoření uživatele nebo přihlášení uživatele pokud již existuje. Díky tomu bylo dosáhнуto jednoduššího přihlášení/registrace pomocí jednoho tlačítka.

```
1 useEffect(() => {
2     const params = new URLSearchParams(window.location.search);
3     const oauthToken = params.get('id_token'); // Zachycení OAuth tokenu z URL
4
5     if (!oauthToken) {
6         setError('Missing ID Token');
7         setLoading(false);
```

```

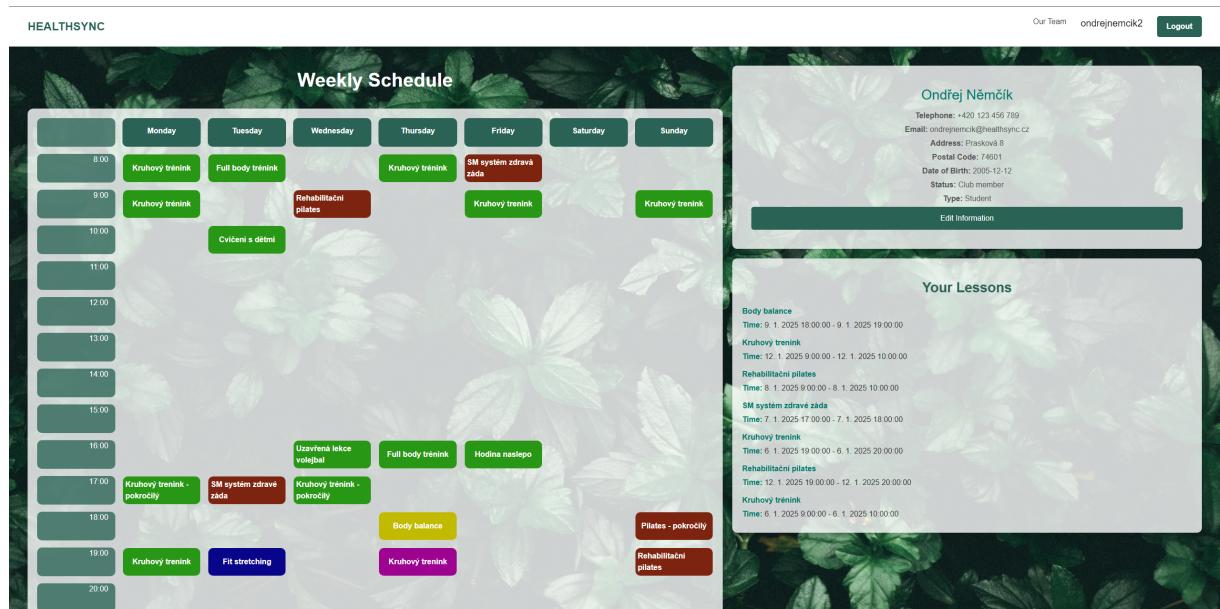
8     return;
9 }
10
11 try {
12     // Decodování JWT tokenu pri prístupe k uživatelským datám
13     const decoded = JSON.parse(atob(oauthToken.split('.')[1]));
14     const userEmail = decoded.email;
15
16     if (!userEmail) {
17         setError('Email not found in the token');
18         setLoading(false);
19         return;
20     }
21
22     // Step 1: Kontrola pokud uživatel existuje podľa emailu
23     fetch(` ${process.env.NEXT_PUBLIC_API_URL}/api/users?filters[email]=${userEmail}`)
24         .then((response) => {
25             if (!response.ok) {
26                 throw new Error(`Failed to fetch user data, Status: ${response.status}`);
27             }
28             return response.json();
29         })
30         .then((data) => {
31             // Pokud existuje
32             if (data && data.length > 0) {
33                 console.log('User found:', data[0]);
34                 // Step 2: Zaloguje sa za nalezeného užívateľa v: /api/auth/local
35                 loginUser(data[0].email, data[0].id);
36             } else {
37                 // Pokud neexistuje, Vytvoří sa užívateľ.
38                 createUser(userEmail);
39             }
40             setLoading(false);
41         })
42         .catch((err) => {
43             setError(`Error fetching user data: ${err.message}`);
44             console.error('Error fetching user data:', err);
45             setLoading(false);
46         });
47     } catch (err) {
48         setError('Failed to decode ID Token');
49         console.error('Error decoding token', err);
50         setLoading(false);
51     }
52 }, [router]);

```

Kód 3.3: Ukázka řešení oauth loginu

3.2.2 Hlavní stránka

Zde byl vytvořen kalendář lekcí co nejjednodušejí. Kalendář zobrazuje aktuální týden, v nedělní hodinách se automaticky zobrazí týden následující. Každá lekce má detailní stránku, kde jsou uvedeny informace o čase, místnosti, trenérovi a seznamu účastníků. Klienti mohou jedním kliknutím rezervovat místo na lekci. Dále mohou na hlavní stránce sledovat následující rezervované lekce a detail profilu (v budoucnu stav peněženky/permanentky).



Obrázek 3.3: Hlavní stránka

```

1 const Calendar = ({ lessons, onLessonClick }) => {
2   const daysOfWeek = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
3     'Saturday', 'Sunday'];
4   const timeSlots = Array.from({ length: 15 }, (_, i) => `${i + 8}:00`);
5   //období od 8:00 do 22:00
6
7
8   return (
9     <div className={styles.calendar}>
10       <div className={styles.header}>
11         <div className={styles.timeCell}></div>
12         {daysOfWeek.map(day => (
13           <div key={day} className={styles.dayCell}>{day}</div>
14         )));
15       </div>
16       {timeSlots.map((time) => (
17         <div key={time} className={styles.timeRow}>
18           <div className={styles.timeCell}>{time}</div>
19           {daysOfWeek.map((day) => {
20             const lessonForCell = lessons.find((lesson) => {

```

```

21     const start = new Date(lesson.attributes.Start);
22     const end = new Date(lesson.attributes.End);
23     const lessonDay = start.toLocaleDateString('en-US', { weekday: 'long' });
24
25     // Kontrola jestli lekce patří do daného dne
26     if (lessonDay !== day) return false;
27
28     // Kalkulace počátku a konce lekce
29     const startHour = start.getHours() - 1;
30     const endHour = end.getHours() - 1;
31     console.log(`Lesson: ${lesson.attributes.Name}, Day: ${lessonDay},
32     Time: ${startHour}-${endHour}`);
33     // Kontrola jestli hodina nepřetéká do další buňky
34     return (
35         (startHour === parseInt(time) && start.getMinutes() > 0) ||
36         (endHour === parseInt(time) && end.getMinutes() === 0) ||
37         (startHour < parseInt(time) && endHour > parseInt(time))
38     );
39 );
40
41 return (
42     <div key={day} className={`${styles.lessonCell}
43     ${lessonForCell ? styles.hasLesson : ''}`}
44     {lessonForCell && (
45         <div
46             className={styles.lessonContent}
47             onClick={() => onLessonClick(lessonForCell)} // Click handler
48         >
49             {lessonForCell.attributes.Name}
50             </div>
51         )})
52     </div>
53 );
54 )
55 </div>
56 ))
57 </div>
58 );
59 };

```

Kód 3.4: Ukázka řešení kalendáře

3.2.3 Administrace

Administrace byla klíčovým prvkem celého systému i celého mého projektu. Po přihlášení má administrátor možnost vytvářet, mazat a upravovat lekce, místnosti a uživatelské profily. Jedná se o přehlednou administraci, která zobrazuje všechny klíčové údaje na jednom místě s možností hledání či filtrování pomocí měsíců nebo týdnů. Na druhém obrázku je možnost vidět konkrétní uživatelé přihlášené na lekci, manuálně je vyhledat, přidat nebo rezervaci zrušit.

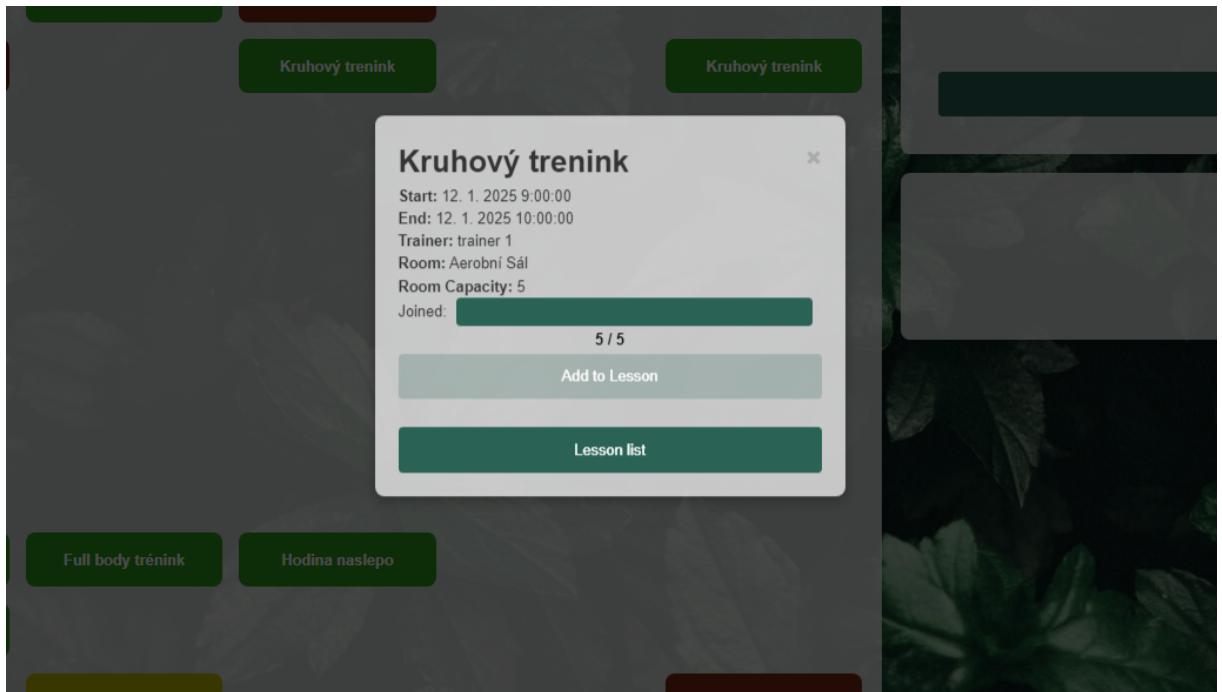
Lessons				
Search by lesson name:				
Select Month	Select Week	Create Lesson		
Kruhový trénink 1. 2025 08:00 - 6. 2025 10:00	6. 1. 2025 09:00 - 6. 1. 2025 10:00	Aerobní Sál	5	Details Delete
Kruhový trénink 1. 2025 10:00	6. 1. 2025 09:00 - 6. 1. 2025 10:00	Funkční zóna	30	Details Delete
Kruhový trenink - pokročilý 6. 1. 2025 17:00 - 6. 1. 2025 18:00	6. 1. 2025 19:00 - 6. 1. 2025 20:00	Funkční zóna	30	Details Delete
Kruhový trenink 6. 1. 2025 19:00 - 6. 1. 2025 20:00	6. 1. 2025 09:00 - 6. 1. 2025 10:00	Aerobní Sál	5	Details Delete
Full body trénink 7. 1. 2025 08:00 - 7. 1. 2025 09:00	7. 1. 2025 10:00 - 7. 1. 2025 11:00	Aerobní Sál	5	Details Delete
SM systém zdravé žády 7. 1. 2025 17:00 - 7. 1. 2025 18:00	7. 1. 2025 19:00 - 7. 1. 2025 20:00	Funkční zóna	30	Details Delete
Fit stretching 7. 1. 2025 19:00 - 7. 1. 2025 20:00	7. 1. 2025 10:00 - 8. 1. 2025 10:00	Aerobní Sál	5	Details Delete
Rehabilitační pilates 8. 1. 2025 16:00 - 8. 1. 2025 17:00	8. 1. 2025 18:00 - 8. 1. 2025 19:00	Plavecký bazén	35	Details Delete
Uzavřená lekce volejbal 8. 1. 2025 16:00 - 8. 1. 2025 17:00	8. 1. 2025 18:00 - 8. 1. 2025 19:00	Venkovní Místě	35	Details Delete
Kruhový trénink - pokročilý 8. 1. 2025 17:00 - 8. 1. 2025 18:00	8. 1. 2025 19:00 - 8. 1. 2025 20:00	Aerobní Sál	5	Details Delete
Kruhový trénink 9. 1. 2025 08:00 - 9. 1. 2025 09:00	9. 1. 2025 10:00 - 9. 1. 2025 11:00	Aerobní Sál	5	Details Delete

Obrázek 3.4: Seznam lekcí

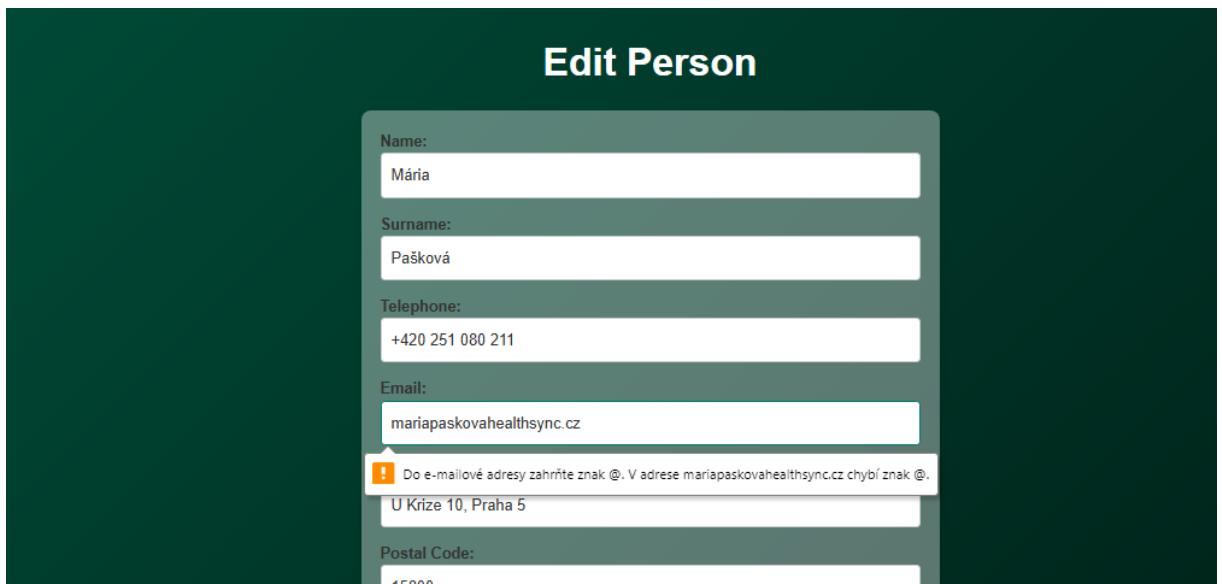
Rehabilitační pilates				
Start: 8. 1. 2025 09:00:00	End: 8. 1. 2025 10:00:00	Trainer: trainer 2	Room: Plavecký bazén	
Add Person				
Search by name:				
People Reserved				
Ondřej Němcík	Delete			
Barbora Čiháková	Delete			
Dušan Jirák	Delete			
Daniel Blažek	Delete			
Stanislav Bica	Delete			

Obrázek 3.5: Seznam rezervovací na lekci

Úpráva dat je ošetřena aby nebylo možno zadávat neplatná data u emailu, tel. čísla ve špatném tvaru, psč. Bylo ošetřeno také překročení maximální kapacity u rezervace lekce.



Obrázek 3.6: Ošetření



Obrázek 3.7: Maximální počet rezervací

4 VÝSLEDKY A ZHODNOCENÍ

4.0.1 Splněné cíle

Na začátku projektu byly definovány jasné cíle, jejichž dosažení mělo zajistit funkční a uživatelsky přívětivý rezervační systém.

- Vytvoření webové aplikace s uživatelským rozhraním pro klienty i administrátory.
- Implementace přihlášení uživatelů pomocí OAuth 2.0 (Google) a klasické registrace.
- Zobrazení dostupných lekcí v přehledném kalendáři s možností detailního náhledu.
- Administrátorský přístup umožňující správu uživatelů, lekcí a místností.
- Použití responsivního designu pro snadné ovládání na mobilních zařízeních.
- Vytvořit pevný základ pro možné budoucí rozšíření.

4.0.2 Budoucí rozvoj

Ačkoli je aplikace v současné podobě plně funkční, existují další možnosti rozšíření, které by mohly zlepšit uživatelskou zkušenosť i technickou stránku systému. Mezi nejdůležitější plánované kroky pro budoucí rozvoj patří:

- Implementace notifikací: Přidání e-mailových a push notifikací pro upozornění na rezervace a změny v rozvrhu.
- Platby online: Integrace platební brány pro snadné platby přímo v aplikaci.
- Statistiky a reporty: Vytvoření analytických nástrojů pro sledování obsazenosti lekcí a preferencí uživatelů.
- Víceúrovňová správa uživatelů: Zavedení dalších rolí, například trenérů, s přístupem pouze k jejich vlastním lekcím.
- Eshop: Nákup permanentek dobíjení kreditu atd.

- Změna použité technologie na backend (Strapi je naprosto jednoduchý systém pro učení, ale bohužel k nasazení do realného provozu je příliš pomalý a chybí zde pokročilejší funkce pro vývoj backendu.)
- Implementace do webových stránek: Posledním krokem by byl redesign celých webových stránek a implementace rezervačního systému
- Navázání systému: navázání na odbavovací systém, později i vytvoření celého interního odbavovacího/pokladního systému navázaného na web a rezervační systém.

4.0.3 Zhodnocení

Projekt přinesl mnoho výzev, ale také příležitostí k rozšíření znalostí a dovedností. Hlavním přínosem projektu je vytvoření plně funkčního systému, který zjednodušuje proces rezervací a přináší větší komfort uživatelům i administrátorům.

Na druhou stranu projekt odhalil několik oblastí pro zlepšení, například potřebu hlubší optimalizace backendu a rozšíření API o pokročilejší funkcionality. Tyto nedostatky jsou však plánovány k odstranění v budoucích verzích aplikace.

Projekt lze hodnotit jako úspěšný, protože splnil definované cíle a přinesl užitečný nástroj. Navíc položil pevné základy pro budoucí rozvoj a další vylepšení.

ZÁVĚR

Vytvoření tohoto rezervačního systému bylo náročným, ale zároveň velmi přínosným procesem. Cílem projektu bylo navrhnout a implementovat moderní webovou aplikaci, která zjednoduší proces rezervací a přinese větší pohodlí jak uživatelům, tak administrátorům. Projekt nejenže splnil všechny stanovené cíle, ale poskytl také příležitost pro rozšíření odborných znalostí a dovedností v oblasti vývoje webových aplikací.

Aplikace v aktuální podobě nabízí intuitivní uživatelské rozhraní, efektivní správu rezervací a přístupné administrátorské funkce. Díky integraci OAuth 2.0 pro přihlášení a využití moderních technologií, jako je React.js a Strapi.js, bylo dosaženo vysoké úrovni uživatelského komfortu a technické spolehlivosti.

Z praktického hlediska přináší aplikace mnoho výhod. Umožňuje rychle a snadno rezervoват lekce, spravovat uživatelské účty a plánovat události. Z administrátorského pohledu systém snižuje pracovní zátěž a zajišťuje přehlednost.

Během realizace projektu se ukázalo, že některé oblasti vyžadují další rozvoj. Patří mezi ně například rozšíření o pokročilé analytické funkce, integrace notifikací nebo podpora více jazyků. Tyto kroky však představují další fáze, na které je aplikace technicky připravena.

Zdrojový kód projektu:(<https://github.com/Endru88/zaverecny-projekt>)

SEZNAM POUŽITÝCH ZDROJŮ

- [1] STRAPI *Strapi Documentation* [Online]. 2024. Dostupné z: <https://docs.strapi.io/dev-docs/intro>
- [2] NEXTJS *NextJs Documentation* [Online]. 2024. Dostupné z: <https://nextjs.org/docs>
- [3] NODEJS *NodeJs Documentation* [Online]. 2024. Dostupné z: <https://nodejs.org/docs/latest/api/>
- [4] YOUTUBE TUTORIALY *Strapi tutorials* [Online]. 2024. Dostupné z: <https://www.youtube.com/@Strapi>
- [5] EXAMPLE *Strapi+NextJs Example* [Online]. 2024. Dostupné z: <https://github.com/buraste/strapi-nextjs-docker-boilerplate>