SŠPU STŘEDNÍ ŠKOLA PRŮMYSLOVÁ A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Rezervační systém HealthSync

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 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

Výběr závěrečného projektu byl pro mě velice náročný, Hledaj jsem takévý 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 danné lekci. Hlavním účelem aplikace je spíše část administrátorská, možnost úpravy profilu klenta, zobrazení příhlášených 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 podobu. Probereme si použité technologie,budoucí cíle a možnosti rozšíření.

1 REZERVAČNÍ SYSTÉM A PROBLEMATIKA

1.1 CO JE REZERVAČNÍ SYSTÉM

Rezervační systémje 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 umožňuje 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 rozšiřitelnost

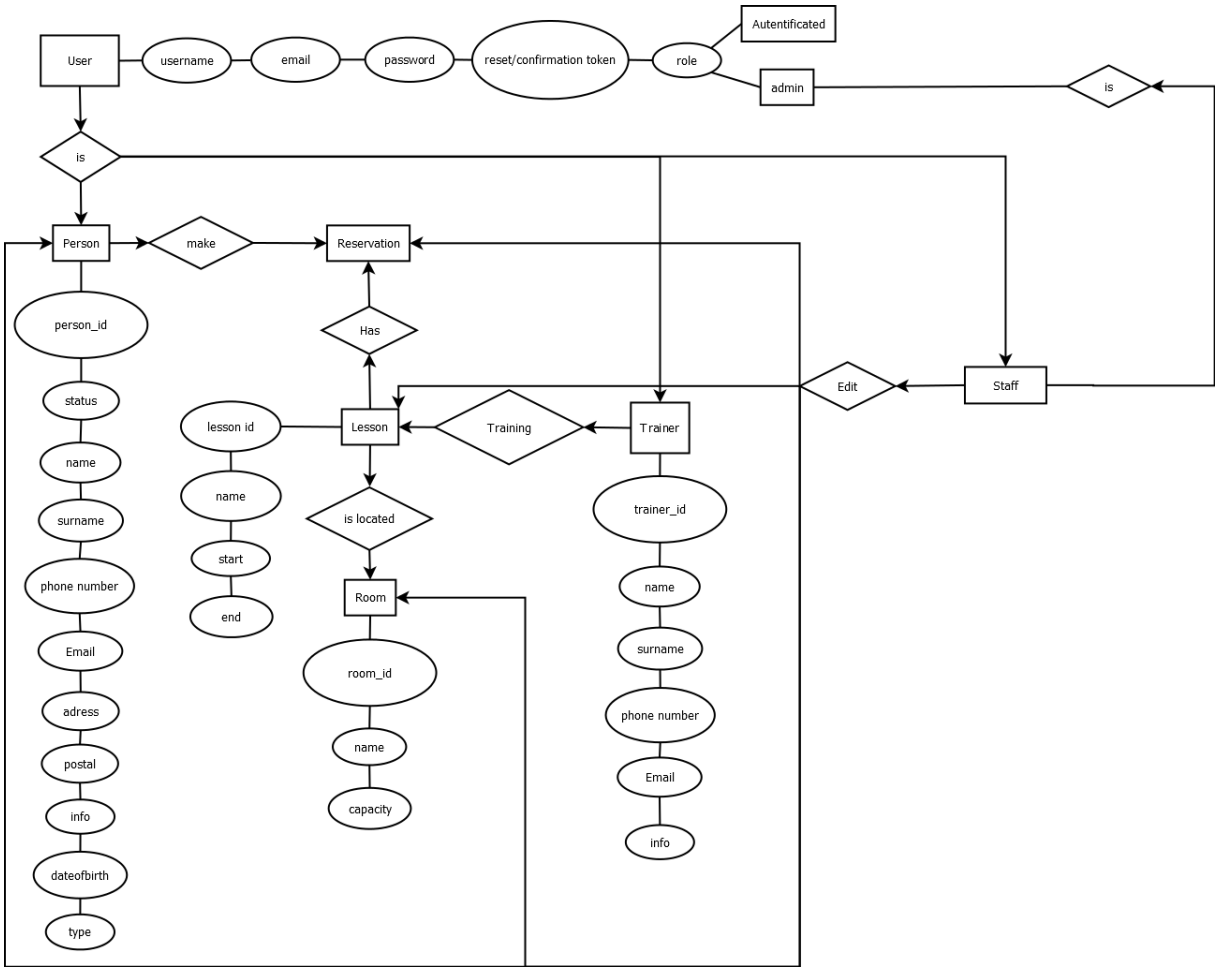
Databázový model využívá 1:N a M:N vztahy, které umožňují propojení různých entit podle potřeb systému (např. jedna lekce může mít více rezervací, jedna osoba může být rezervována na více lekcí).

1.2.4 Flexibilita a rozšiřitelnost

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.



1.3 ZKUŠENOSTI S TVORBOU DATABÁZÍ

Prbolematika databázových aplikací pro mě není probblémem. Jelikož tento projekt výchází 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

ZÁVĚR

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

A.0.1 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:

```python
# Python code here
def hello_world():
  print("Hello, world!")
```

Kód A.1: Ukázka Python kódu

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

Kód A.2: Ukázka JS kódu

```javascript
/* 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 = ''
```

```
13        this.isActive = false
14        this.lastLogin = ''  // ISO 8601 formatted timestamp.
15        this.lastPWChange = ''  // ISO 8601 formatted timestamp.
16      }
17    }
18
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 A.3: ES6 (ECMAScript-2015) Listing

A.1 Pokročilejší tipy, které se mohou hodit