Stredná priemyselná škola elektrotechnická  
Hálova 16, 851 01 Bratislava

**AUTOBAHN.SK - BACKEND WEBOVEJ APLIKÁCIE**

KOMPLEXNÁ ODBORNÁ MATURITNÁ PRÁCA

Bratislava, 2024 4.D Martin Mojžiš

Stredná priemyselná škola elektrotechnická  
Hálova 16, 851 01 Bratislava

**AUTOBAHN.SK - BACKEND WEBOVEJ APLIKÁCIE**

KOMPLEXNÁ ODBORNÁ MATURITNÁ PRÁCA

Štúdijný odbor: 2573M programovanie digitálnych technológií  
Konzultant: Ing. Dominik Zatkalík, PhD.

Bratislava, 2024 4.D Martin Mojžiš

<SEM VOLZITE ZADANIE, PODPISANE>

**Čestné vyhlásenie**

Ja, dolupodpísaný, študent 4. D triedy Strednej priemyselnej školy elektrotechnickej Hálova 16, 851 01 Bratislava, týmto vyhlasujem, že som túto prácu vyhotovil sám, s použitím uvedenej literatúry a podľa rád môjho konzultanta.

.........................................................

V Bratislave, <dd. mm. rrrr> Martin Mojžiš

**Poďakovanie**

Rád by som sa touto cestou poďakoval svojmu <školiteľovi> za prístup a odborné rady. Tiež by som sa rád poďakoval <spoločnosti> za finančnú podporu pri realizácii praktickej časti mojej práce. PODĽA VÁŠHO UVÁŽENIA

**Abstrakt:**

**Kľúčové slová:**

**Abstract:**

**Keywords:**

**Obsah**

[ÚVOD 6](#_Toc187269982)

[1 Teoretické východiská 7](#_Toc187269983)

[1.1 Štruktúra webových aplikácií 7](#_Toc187269984)

[1.1.1 Frontend – Užívateľská vrstva 7](#_Toc187269985)

[1.1.2 Backend – Logická vrstva 8](#_Toc187269986)

[1.1.3 Databáza – Úložisko dát 8](#_Toc187269987)

[1.1.4 API – Prepojenie vrstiev 8](#_Toc187269988)

[1.2 Vývojový plán webovej aplikácie 9](#_Toc187269989)

[1.2.1 Brainstorming nápadov 9](#_Toc187269990)

[1.2.2 Tvorba funkčnej špecifikácie 9](#_Toc187269991)

[1.2.3 Tvorba wireframes 10](#_Toc187269992)

[1.2.4 Plánovanie roboty 10](#_Toc187269993)

[1.3 Analýza backendových technológií vhodných na tvorbu webových aplikácií 10](#_Toc187269994)

[1.3.1 ASP .NET (C#) 10](#_Toc187269995)

[1.3.2 Spring Boot (Java) 11](#_Toc187269996)

[1.3.3 Django (Python) 11](#_Toc187269997)

[1.3.4 Ruby on Rails (Ruby) 11](#_Toc187269998)

[1.3.5 Express (Node.js) 12](#_Toc187269999)

[1.3.6 Laravel (PHP) 12](#_Toc187270000)

[1.4 Backendové technológie použité pri tvorbe webovej aplikácie 12](#_Toc187270001)

[2 Návrh riešenia 13](#_Toc187270002)

[2.1 Vytvorte backendovú časť webovej aplikácie 13](#_Toc187270003)

[2.2 Popíšte a znázornite technickú architektúru backendovej časti webovej aplikácie 13](#_Toc187270004)

[2.3 Navrhnite a vytvorte databázu webovej aplikácie 14](#_Toc187270005)

[2.4 Zverejnite webovú aplikáciu na doménu a popíšte postup jej nasadenia na server 14](#_Toc187270006)

[2.5 Popíšte podnikateľský plán webovej aplikácie 14](#_Toc187270007)

[2.6 Popíšte marketingový plán a stratégiu webovej aplikácie 14](#_Toc187270008)

[3 Záver 15](#_Toc187270009)

[Zoznam použitej literatúry 16](#_Toc187270010)

[Prílohy <volitelne> 7](#_Toc187270011)

[Príloha A – Zdrojový kód I](#_Toc187270012)

**Zoznam skratiek, značiek a symbolov**

API - Application Programming Interface

WWW - World Wide Web

IP - Internet Protocol

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

CMS - Content Management System

MVP - Minimum Viable Product

IoT - Internet of Things

JSON - JavaScript Object Notation

AWS – Amazon Web Services

**Zoznam tabuliek, grafov a ilustrácií**

<Zoznam skratiek, značiek a symbolov>

# ÚVOD

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.

Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy.

# Teoretické východiská

Webová aplikácia sa najčastejšie chápe ako HTML stránka, ktorá sa zobrazuje používateľom vo webových prehliadačoch na zariadeniach pripojených k internetu. Tieto stránky sú uložené na vzdialených serveroch, ktoré sú prostredníctvom internetovej siete prepojené s konečnými užívateľmi. Každý server, rovnako ako akékoľvek iné zariadenie v sieti, má svoju vlastnú IP adresu. Väčšinou sú však tieto servery identifikované prostredníctvom domén (napríklad google.com) alebo subdomén (napríklad docs.google.com).

V bežnej reči sa však pod pojmom webová stránka často myslí celý systém, ktorý zabezpečuje jej fungovanie. Vizuálna časť, ktorú vidí používateľ, je len malou časťou celého systému. Na pozadí prebieha väčšinou komplexná logika, ktorá pracuje s dátami, ktoré užívatelia buď poskytujú, alebo ich na stránke potrebujú. Okrem toho existujú aj jednoduchšie prezentačné stránky, ktoré slúžia na prezentáciu organizácie alebo produktu. Takéto stránky sú zvyčajne statické, pretože ich obsah sa často nemení.

Na druhej strane sú dynamické webové stránky, ktoré svoj obsah získavajú z backendu. Tento backend spracúva dáta a odosiela ich na vykreslenie. Typickými príkladmi dynamických stránok sú internetové obchody alebo stránky novín. V oboch prípadoch je základom stránky obsah, ktorý spravujú administrátori. Na to, aby mohli obsah ľahko upravovať, sa dnes často používajú systémy na správu obsahu (CMS). Administrátori tak môžu prostredníctvom formulárov jednoducho vytvárať blogové príspevky, produkty a ďalší obsah. Navyše majú možnosť nastaviť titulné obrázky či iné detaily.

## Štruktúra webových aplikácií

Webové aplikácie sú komplexné systémy, ktoré pozostávajú z viacerých prepojených častí. Ich štruktúra je navrhnutá tak, aby umožnila efektívnu komunikáciu medzi jednotlivými komponentmi a zabezpečila požadovanú funkcionalitu. Táto štruktúra sa zvyčajne skladá z troch hlavných vrstiev: frontend, backend a databáza.

### Frontend – Užívateľská vrstva

Frontend je časť aplikácie, ktorú vidí a s ktorou interaguje užívateľ. Zodpovedá za vizuálne prvky, ako sú rozloženie, farby, texty, tlačidlá a formuláre. Táto vrstva je zvyčajne naprogramovaná pomocou technológií ako HTML, CSS a JavaScript. Moderné frontend frameworky ako React, Vue.js alebo Angular umožňujú vývojárom vytvárať rýchle, responzívne a dynamické používateľské rozhrania.

Frontend komunikuje s backendom prostredníctvom API (Application Programming Interface), aby získal dáta alebo poslal požiadavky. Napríklad po kliknutí na tlačidlo na odoslanie formulára sa tieto údaje odosielajú na backend, kde sú ďalej spracované.

### Backend – Logická vrstva

Backend je motorom každej webovej aplikácie. Ide o časť, ktorá spracúva požiadavky od frontendovej vrstvy, vykonáva zložité výpočty alebo logiku, a zabezpečuje komunikáciu s databázou. Backendové systémy sú programované v rôznych jazykoch, ako sú PHP, Python, Node.js, C# alebo Java.

Na efektívnu organizáciu backendového kódu sa často využívajú frameworky, ako napríklad Laravel, Django, Express.js, Spring alebo .NET Core. Tieto nástroje pomáhajú vývojárom implementovať funkcionalitu rýchlejšie a s menším rizikom chýb.

Backend tiež zabezpečuje bezpečnosť aplikácie, autentifikáciu užívateľov a ochranu dát. Napríklad pri prihlasovaní do aplikácie backend overí zadané údaje a vygeneruje token, ktorý užívateľovi umožní prístup k ďalším funkcionalitám.

### Databáza – Úložisko dát

Databáza je miesto, kde sú uložené všetky dáta potrebné pre fungovanie aplikácie. Môže ísť o údaje o užívateľoch, produkty, transakcie alebo akýkoľvek iný obsah, ktorý aplikácia spravuje. Najčastejšie sa používajú dva druhy databáz:

1. Relačné databázy – ako napríklad MySQL, PostgreSQL alebo Microsoft SQL Server, ktoré organizujú dáta do tabuliek a umožňujú ich prepojenie pomocou vzťahov.
2. NoSQL databázy – ako MongoDB alebo Cassandra, ktoré sú flexibilnejšie a používajú sa najmä pri práci s veľkým množstvom nestruktúrovaných dát.

Databáza komunikuje s backendom, ktorý od nej žiada dáta alebo do nej zapisuje nové informácie.

### API – Prepojenie vrstiev

Komunikácia medzi frontendom, backendom a databázou prebieha pomocou API. API definuje pravidlá, ako si jednotlivé vrstvy môžu vymieňať dáta. Najčastejšie sa používajú REST API alebo GraphQL, ktoré umožňujú rýchlu a efektívnu výmenu informácií medzi systémami.

## Vývojový plán webovej aplikácie

Pri vývoji webovej aplikácie je dôležité venovať dostatok času plánovaniu, ktoré zabezpečí hladký priebeh celého procesu. Kľúčovým krokom je rozdelenie celkového plánu na jednotlivé fázy, ktoré umožnia postupný vývoj aplikácie, minimalizujú riziko nejasností a problémov počas implementácie. Tieto fázy zahŕňajú brainstorming nápadov, tvorbu funkčnej špecifikácie, vytvorenie wireframes a podrobný plán práce.

### Brainstorming nápadov

Prvým krokom vo vývoji aplikácie je brainstorming, počas ktorého sa zameriavame na zhromažďovanie a analyzovanie nápadov. Ide o veľmi dôležitú fázu, kde je potrebné zamyslieť sa nad cieľom aplikácie, jej používateľmi, požiadavkami a funkciami. Zameriavame sa na všetky možnosti a snažíme sa nájsť najefektívnejšie riešenia. Môžeme využiť rôzne techniky ako diskusie, analýzu konkurencie alebo spätnú väzbu od potenciálnych používateľov, aby sme pochopili ich potreby a očakávania.

Tento proces nám pomáha definovať hlavné ciele aplikácie a určiť, aké funkcie a vlastnosti by mala mať, aby bola úspešná. Na základe týchto nápadov a postrehov vytvoríme zoznam funkcií, ktoré sa následne stanú základom pre ďalšie fázy vývoja.

### Tvorba funkčnej špecifikácie

Po úspešnom brainstormingu je ďalším krokom vytvorenie funkčnej špecifikácie, ktorá podrobne popisuje, čo aplikácia musí robiť a ako bude vyzerať jej funkcionalita. Funkčná špecifikácia sa zvyčajne delí do niekoľkých častí:

1. Základná funkcionalita – určenie, čo všetko aplikácia musí umožniť. To môže zahŕňať registráciu užívateľov, správu účtov, zobrazenie produktov alebo spracovanie platieb.
2. Rozdelenie na stránky a sekcie – určenie, ako bude aplikácia organizovaná, aké stránky alebo sekcie budú potrebné a aké funkcie budú priradené k jednotlivým stránkam.
3. Prioritizácia funkcií – stanovenie, ktoré funkcie sú nevyhnutné a ktoré môžu byť implementované neskôr. Tento krok pomáha sústrediť sa na kľúčové vlastnosti aplikácie a minimalizovať riziko zdržania projektu.

Funkčná špecifikácia je nevyhnutná na zabezpečenie, že všetky požiadavky sú jasne definované ešte pred začiatkom vývoja a umožňuje tímu lepšie pochopiť, čo je potrebné na dosiahnutie úspechu.

### Tvorba wireframes

Ďalšou dôležitou fázou je vytvorenie wireframes – vizuálnych náhľadov alebo skíc, ktoré zobrazujú rozloženie jednotlivých stránok aplikácie. Wireframes slúžia ako základný nástroj pre dizajnérov a vývojárov, aby pochopili, ako bude aplikácia vyzerať a ako budú usporiadané jej jednotlivé prvky.

Wireframes zvyčajne nezahŕňajú detailný dizajn, ale ukazujú základnú štruktúru stránky – kde budú umiestnené tlačidlá, navigačné menu, obrázky a texty. Vytvorením wireframes sa zabezpečí, že všetky dôležité prvky budú na správnych miestach a že používateľská skúsenosť bude intuitívna.

### Plánovanie roboty

Posledným krokom pred samotným začiatkom vývoja je plánovanie roboty. Tento krok zahŕňa rozdelenie celého vývojového procesu na menšie úlohy a stanovenie termínov, kedy by mali byť jednotlivé časti aplikácie hotové. Je dôležité, aby vývojový tím mal jasný časový harmonogram a vedel, ktoré úlohy sú najurgentnejšie.

Plánovanie roboty tiež zahŕňa identifikáciu potrebných technológií, nástrojov a rôznych frameworkov, ktoré budú použité pri vývoji. Každý člen tímu by mal mať presne definovanú úlohu a vedieť, na čo sa má počas vývoja sústrediť.

## Analýza backendových technológií vhodných na tvorbu webových aplikácií

Pri vývoji webových aplikácií je výber správnej backendovej technológie kľúčový pre zabezpečenie efektivity, rozšíriteľnosti a udržateľnosti celého systému. Ako som už spomínal, backend zodpovedá za spracovanie logiky aplikácie, komunikáciu s databázami a poskytovanie dát pre frontend. V tomto texte sa zameriame na niektoré populárne backendové frameworky, ktoré sú vhodné pre tvorbu webových aplikácií, pričom pri ich výbere som sa zameral aj možnosti CMS nadstavieb, ktoré môžu výrazne uľahčiť správu obsahu.

### ASP .NET (C#)

ASP .NET je silný a flexibilný framework od Microsoftu, ktorý sa často používa na vývoj webových aplikácií s využitím jazyka C#. Tento framework je veľmi populárny v prostredí podnikových aplikácií a umožňuje vývoj aplikácií s vysokým výkonom a bezpečnosťou. Pre vývojárov, ktorí preferujú C# a .NET ekosystém, je ASP .NET skvelou voľbou.

Pre ASP .NET existujú rôzne CMS nadstavby, ktoré umožňujú jednoduchú správu obsahu bez potreby rozsiahleho vlastného vývoja. Umbraco CMS je jedným z najpopulárnejších CMS riešení pre tento framework, ktorý ponúka užívateľsky prívetivé rozhranie a flexibilitu pri správe obsahu. Ďalšími možnosťami sú Butter CMS a Strapi CMS, ktoré poskytujú headless (bez vlastného alebo frontendu, ktorý by zobrazoval obsah) CMS riešenia s API na správu obsahu.

### Spring Boot (Java)

Spring Boot je moderný framework pre jazyk Java, ktorý zjednodušuje vývoj robustných a škálovateľných webových aplikácií. Spring Boot je obľúbený medzi vývojármi Java, pretože ponúka množstvo nástrojov na rýchle vytváranie aplikácií a ich následné nasadenie. Tento framework je veľmi silný, pokiaľ ide o prácu s mikroservisami a enterprise aplikáciami.

Pre tento framework existujú viaceré CMS riešenia, ktoré umožňujú jednoduchú integráciu do aplikácie. Crafter CMS je flexibilné a výkonné CMS, ktoré je navrhnuté špeciálne pre Java ekosystém. Rovnako ako pri ASP .NET, aj tu sú k dispozícii ďalšie headless CMS riešenia, ako napríklad Butter CMS a Strapi CMS, ktoré poskytujú jednoduchú správu obsahu cez API.

### Django (Python)

Django je robustný webový framework pre Python, ktorý je známy svojím plne vybaveným ekosystémom. Django poskytuje množstvo vstavaných funkcií, ako je autentifikácia, správa databáz alebo admin rozhranie, čo výrazne urýchľuje vývoj webových aplikácií. Tento framework je ideálny pre projekty, kde je potrebná vysoká úroveň organizácie a bezpečnosti.

V prípade Django existuje viacero kvalitných CMS riešení, medzi ktoré patrí Django CMS a Wagtail, ktoré ponúkajú pokročilú správu obsahu s rôznymi rozšíriteľnými možnosťami. Okrem toho sú tu aj možnosti ako Butter CMS a Strapi CMS, ktoré sú platformovo nezávislé a umožňujú flexibilné API riešenia na správu obsahu.

### Ruby on Rails (Ruby)

Ruby on Rails je veľmi obľúbený framework pre jazyk Ruby, ktorý sa vyznačuje vysokou produktivitou a konvenciami (množstvo vstavaných funkcií), ktoré uľahčujú vývoj aplikácií. Rails je ideálny pre rýchly vývoj MVP aplikácií a startupy, ktoré potrebujú rýchlo uviesť aplikáciu do prevádzky.

Pre Ruby on Rails existujú CMS riešenia ako Alchemy CMS a Refinery CMS, ktoré sa výborne integrujú s týmto frameworkom. Tieto CMS ponúkajú flexibilitu pri správe obsahu a jednoduché rozšírenie funkcionality. Rovnako ako pri iných frameworkoch, aj pre Rails sú dostupné headless CMS ako Butter CMS a Strapi CMS, ktoré sa ľahko implementujú do aplikácie cez API.

### Express (Node.js)

Express je minimalistický framework pre Node.js, ktorý sa často používa pri tvorbe moderných webových aplikácií a mikroservisov. Express poskytuje flexibilitu a rýchlosť pri vývoji, čo ho robí ideálnym pre startupy a projekty, ktoré vyžadujú rýchle iterácie a zmeny.

Pre Express existuje niekoľko CMS riešení, ako napríklad Payload CMS a Keystone, ktoré sú plne prispôsobiteľné a ľahko integrované s aplikáciami postavenými na Express. Pre headless CMS riešenia sa môžu použiť aj Butter CMS a Strapi CMS, ktoré umožňujú efektívnu správu obsahu cez API.

### Laravel (PHP)

Laravel je veľmi populárny PHP framework, ktorý je známy svojou elegantnosťou a jednoduchosťou. Laravel poskytuje množstvo nástrojov na správu databáz, autentifikáciu a vytváranie RESTful API, čo ho robí ideálnym pre tvorbu komplexných webových aplikácií.

Pre Laravel existujú kvalitné CMS riešenia ako October CMS a Winter CMS, ktoré umožňujú jednoduchú správu obsahu a sú veľmi flexibilné, čo je ideálne pre rôzne typy webových aplikácií. Tieto CMS sú špecifické pre Laravel, ale pre weby, ktoré potrebujú API-first prístup, môžu byť použité aj Butter CMS a Strapi CMS.

## Typy databáz a ich využitie pri vývoji webových aplikácií

Pri vývoji webových aplikácií je výber vhodnej databázy kľúčovým rozhodnutím, ktoré môže ovplyvniť výkon, škálovateľnosť a flexibilitu aplikácie. Databázy sa delia do rôznych typov na základe spôsobu ukladania a spracovania dát. Dve hlavné kategórie databáz sú relačné databázy a nerelačné databázy. Každý typ má svoje výhody a nevýhody, a preto je dôležité vybrať správny typ podľa konkrétnych požiadaviek aplikácie.

### Relačné databázy (SQL)

Relačné databázy sú najbežnejším typom databáz, ktoré udržiavajú dáta v štruktúre tabuliek, pričom medzi jednotlivými tabuľkami sú definované vzťahy (relácie). Dajú sa veľmi efektívne používať v aplikáciách, ktoré vyžadujú silnú integritu dát a komplexné dotazy.

Príklady relačných databáz:

1. MySQL – veľmi populárna a rýchla relačná databáza, vhodná pre širokú škálu aplikácií.
2. PostgreSQL – pokročilá open-source databáza s robustnými funkciami, ideálna pre aplikácie s veľkými dátami alebo zložitými transakciami.
3. Microsoft SQL Server – relačná databáza od Microsoftu, obľúbená v podnikovom prostredí.
4. Oracle Database – silná databáza, ktorá sa využíva v náročných podnikových aplikáciách.

Relačné databázy sú ideálne pre aplikácie, ktoré potrebujú transakcie, zložité dotazy, spoľahlivosť a štruktúrované dáta (napríklad e-shopy, bankové systémy alebo aplikácie spravujúce inventár).

### Nerelačné databázy (NoSQL)

Nerelačné databázy, alebo NoSQL databázy, sú flexibilnejšie a umožňujú ukladanie dát v rôznych formátoch, ako sú dokumenty, kľúč-hodnota páry, grafy alebo stĺpcové modely. Tieto databázy sú vhodné pre aplikácie, ktoré potrebujú rýchlo spracovávať veľké množstvo nestruktúrovaných dát alebo dát s dynamickými schémami.

Príklady nerelačných databáz:

1. MongoDB – populárna dokumentová databáza, ktorá ukladá dáta vo formáte JSON. Vhodná pre aplikácie, ktoré musia spracovávať veľké objemy dát s meniacimi sa štruktúrami (napríklad sociálne siete, analytické nástroje).
2. Cassandra – distribuovaná stĺpcová databáza, ideálna pre aplikácie, ktoré potrebujú vysokú dostupnosť a horizontálnu škálovateľnosť (napríklad sledovanie logov v reálnom čase).
3. Redis – databáza kľúč-hodnota, ktorá je extrémne rýchla a často sa používa na cacheovanie alebo správu relácií (napríklad v systémoch, ktoré potrebujú rýchle čítanie/zápis dát).
4. CouchDB – databáza dokumentov, ktorá umožňuje flexibilné ukladanie dát a vysokú dostupnosť.

Nerelačné databázy sú vhodné pre aplikácie, ktoré potrebujú flexibilitu pri ukladaní rôznych druhov dát (napríklad mobilné aplikácie, real-time aplikácie, IoT aplikácie).

## Backendové technológie použité pri tvorbe webovej aplikácie

Pri tvorbe backendovej časti mojej webovej aplikácie som sa rozhodol pre technológie, ktoré sú robustné, flexibilné a overené v praxi. Jedným z kľúčových rozhodnutí bolo vybrať si OctoberCMS, postavený na frameworku Laravel (PHP), ktorý mi poskytol silný základ pre tvorbu aplikácie. Mám s ním viac ako 3 ročné skúsenosti, čo bol jeden z rozhodujúcich faktorov pri jeho výbere. Okrem toho, aplikácia využíva MySQL ako databázu a rôzne ďalšie technológie na ukladanie dát, logovanie a správu súborov, vrátane AWS S3 na správu obrázkov a BugSnag na analytiku a logovanie chýb.

### OctoberCMS: Ideálny základ pre flexibilnú a modulárnu aplikáciu

OctoberCMS je open-source CMS postavený na veľmi populárnom PHP frameworku Laravel. Jeho hlavnou výhodou je modularita a prehľadná štruktúra kódu, vďaka ktorej je vývoj aplikácie jednoduchší a efektívnejší. Tento systém je založený na pluginovej architektúre, čo znamená, že je možné pridávať rôzne moduly a funkčnosti podľa potreby bez zásahu do základného kódu. To je ideálne pre aplikácie, ktoré sa môžu v priebehu času rozširovať alebo potrebujú pravidelnú úpravu, pretože umožňuje rýchle prispôsobenie a modifikáciu aplikácie.

Jednou z ďalších veľkých výhod OctoberCMS je integrovaný CMS systém, ktorý umožňuje správu obsahu priamo z administratívneho rozhrania. Tento systém je veľmi užívateľsky prívetivý a umožňuje rýchle nastavenie stránok, spravovanie obsahu a ich publikovanie bez potreby zásahu do kódu. Pre webové aplikácie, ktoré potrebujú ľahkú správu obsahu bez zbytočného zložitého backendového vývoja, je OctoberCMS výborným riešením.

Vďaka tomu, že je postavený na Laravel frameworku, ktorý je jedným z najpopulárnejších a najefektívnejších PHP frameworkov, môžem využívať množstvo existujúcich knižníc a nástrojov, ktoré Laravel ponúka, pomocou Composer Package Manager. Rýchly vývoj, podpora pre RESTful API, automatické generovanie migračných skriptov pre databázu, jednoduchá autentifikácia a ďalšie vlastnosti robia Laravel aj v kombinácii s OctoberCMS skvelou voľbou pre vytváranie moderných webových aplikácií.

### MySQL: Spoľahlivá relačná databáza

Pre správu dát v mojej webovej aplikácii som sa rozhodol pre MySQL, čo je jedna z najrozšírenejších a najspoľahlivejších relačných databáz. MySQL je známa svojou rýchlosťou, škálovateľnosťou a vysokou dostupnosťou, čo ju robí ideálnou voľbou pre aplikácie s vysokými požiadavkami na výkon a spracovanie veľkého objemu dát. S MySQL mám skúsenosti, ktoré mi umožňujú efektívne optimalizovať dotazy, navrhovať komplexné schémy databáz a implementovať transakcie s dôrazom na integritu dát.

Relačné databázy, ako je MySQL, sú ideálne pre aplikácie, ktoré vyžadujú komplexné dotazy, transakcie a vzťahy medzi dátami. Pre aplikáciu, ktorá pracuje s rôznymi dátovými entitami a požaduje vysokú spoľahlivosť a transakčnú integritu, je MySQL perfektnou voľbou. S jej podporou pre ACID vlastnosti (Atomicity, Consistency, Isolation, Durability) môžem byť istý, že dáta v aplikácii sú vždy v súlade a správne spravované.

### MySQL: Spoľahlivá relačná databáza

Na ukladanie médií, ako sú obrázky a videá, využívam AWS S3 (Simple Storage Service), ktorý poskytuje spoľahlivé, škálovateľné a bezpečné ukladanie súborov v cloudovom prostredí. AWS S3 je vynikajúca voľba pre aplikácie, ktoré potrebujú ukladať veľké množstvo súborov s vysokými požiadavkami na dostupnosť a škálovateľnosť. S S3 môžem jednoducho spravovať a pristupovať k súborom cez API, čo mi umožňuje flexibilne prispôsobovať ukladanie médií podľa potreby aplikácie.

Integrácia AWS S3 do aplikácie je veľmi jednoduchá a podporuje nielen ukladanie, ale aj správu prístupových práv, verziovanie súborov a šifrovanie dát. Vďaka tomu môžem bezpečne a efektívne spravovať súbory bez toho, aby som musel spravovať vlastné servery na ukladanie dát.

### BugSnag: Analytika a logovanie chýb

Aby sme mali prehľad o výkone aplikácie a mohli efektívne riešiť chyby, využívame BugSnag. BugSnag je služba na logovanie chýb a analytiku, ktorá poskytuje real-time monitoring a podrobný prehľad o chybách, ktoré sa v aplikácii vyskytnú. Pomáha nám rýchlo identifikovať a opraviť problémy, ktoré by mohli ovplyvniť používateľov, a to ešte predtým, ako sa stanú vážnymi problémami.

BugSnag umožňuje sledovať chybové hlásenia v aplikácii, detailne zobrazuje kontext chyby, vrátane informácií o prostredí, používateľoch a ďalších faktoroch. Tieto informácie nám pomáhajú analyzovať príčiny problémov a rýchlo ich vyriešiť, čo zvyšuje kvalitu aplikácie a zlepšuje používateľskú skúsenosť.

# Návrh riešenia

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem.

In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede. Suspendisse dapibus lorem pellentesque magna. Integer nulla. Donec blandit feugiat ligula. Donec hendrerit, felis et imperdiet euismod, purus ipsum pretium metus, in lacinia nulla nisl eget sapien.

Donec ut est in lectus consequat consequat. Etiam eget dui. Aliquam erat volutpat. Sed at lorem in nunc porta tristique. Proin nec augue. Quisque aliquam tempor magna. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nunc ac magna. Maecenas odio dolor, vulputate vel, auctor ac, accumsan id, felis. Pellentesque cursus sagittis felis.

## Vytvorte backendovú časť webovej aplikácie

In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede. Suspendisse dapibus lorem pellentesque magna. Integer nulla. Donec blandit feugiat ligula. Donec hendrerit, felis et imperdiet euismod, purus ipsum pretium metus, in lacinia nulla nisl eget sapien.

## Popíšte a znázornite technickú architektúru backendovej časti webovej aplikácie

In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede. Suspendisse dapibus lorem pellentesque magna. Integer nulla. Donec blandit feugiat ligula. Donec hendrerit, felis et imperdiet euismod, purus ipsum pretium metus, in lacinia nulla nisl eget sapien.

## Navrhnite a vytvorte databázu webovej aplikácie

In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede. Suspendisse dapibus lorem pellentesque magna. Integer nulla. Donec blandit feugiat ligula. Donec hendrerit, felis et imperdiet euismod, purus ipsum pretium metus, in lacinia nulla nisl eget sapien.

## Zverejnite webovú aplikáciu na doménu a popíšte postup jej nasadenia na server

In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede. Suspendisse dapibus lorem pellentesque magna. Integer nulla. Donec blandit feugiat ligula. Donec hendrerit, felis et imperdiet euismod, purus ipsum pretium metus, in lacinia nulla nisl eget sapien.

## Popíšte podnikateľský plán webovej aplikácie

In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede. Suspendisse dapibus lorem pellentesque magna. Integer nulla. Donec blandit feugiat ligula. Donec hendrerit, felis et imperdiet euismod, purus ipsum pretium metus, in lacinia nulla nisl eget sapien.

## Popíšte marketingový plán a stratégiu webovej aplikácie

In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede. Suspendisse dapibus lorem pellentesque magna. Integer nulla. Donec blandit feugiat ligula. Donec hendrerit, felis et imperdiet euismod, purus ipsum pretium metus, in lacinia nulla nisl eget sapien.

# Záver

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem.

In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede. Suspendisse dapibus lorem pellentesque magna. Integer nulla. Donec blandit feugiat ligula. Donec hendrerit, felis et imperdiet euismod, purus ipsum pretium metus, in lacinia nulla nisl eget sapien.

Donec ut est in lectus consequat consequat. Etiam eget dui. Aliquam erat volutpat. Sed at lorem in nunc porta tristique. Proin nec augue. Quisque aliquam tempor magna. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nunc ac magna. Maecenas odio dolor, vulputate vel, auctor ac, accumsan id, felis. Pellentesque cursus sagittis felis.

# Zoznam použitej literatúry

<TU CITOVAT PODLA NORMY STN ISO 690 >

[1]

[2]

[3]

[4]

[5]

# Prílohy <volitelne>

## Príloha A – Zdrojový kód

**príloha B - Fotodokumentácia**