VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná informatika

MODULÁRNÍ REDAKČNÍ SYSTÉM PRO RESTAURACE

Bakalářská práce

Autor práce: Vojtěch Pelouch

Vedoucí práce: PaedDr. František Smrčka, Ph.D.

Jihlava 2026

VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	**Vojtěch Pelouch**
Studijní program:	Aplikovaná informatika
Garant studijního programu:	doc. Ing. Lenka Kuklišová Pavelková, Ph.D.

Název práce:	**Modulární redakční systém pro restaurace**
Vedoucí práce:	PaedDr. František Smrčka, Ph.D.
Cíl práce:	Cílem práce je navrhnout a implementovat redakční systém pro restaurace a další gastronomické podniky, jako jsou penziony, hotely či hospody. Systém bude umožňovat správu obsahu webových stránek a rozšíření funkcionality pomocí modulů. Práce se zaměří na analýzu existujících řešení, návrh architektury a vývoj systému s důrazem na přehlednost, jednoduchost správy a přizpůsobitelnost pro různé typy podniků v oblasti gastronomie.

Abstrakt

Práce se zabývá návrhem a implementací modulárního redakčního systému pro restaurace. Cílem bylo vytvořit webovou aplikaci, která umožňuje správu obsahu i provozních funkcí podniku bez nutnosti zásahu programátora. Systém je navržen jako multi-tenantní řešení umožňující správu více projektů a zahrnuje moduly pro správu menu, rezervací, online objednávek a uživatelských přístupů s využitím RBAC modelu. Aplikace byla implementována v jazyce PHP s využitím databáze MariaDB a knihovny Tailwind CSS pro tvorbu uživatelského rozhraní. Součástí práce je analýza existujících řešení, návrh architektury systému a jeho testování. Výsledkem je plně funkční webová aplikace s administrační i veřejnou částí, připravená pro reálné nasazení v menších gastronomických podnicích.

Klíčová slova

CMS; webová aplikace; restaurace; PHP; Tailwind CSS; MariaDB; administrace

Abstract

This thesis deals with the design and implementation of a modular content management system for restaurants. The aim was to create a web application that enables the management of both content and operational functions of a business without the need for a programmer. The system is designed as a multi-tenant solution enabling the management of multiple projects and includes modules for menu management, reservations, online orders, and user access management using the RBAC model. The application was implemented in PHP using a MariaDB database and the Tailwind CSS library for creating the user interface. The thesis includes an analysis of existing solutions, the design of the system architecture, and its testing. The result is a fully functional web application with both an administrative and a public part, ready for real deployment in small gastronomic businesses.

Keywords

CMS; web application; restaurant; PHP; Tailwind CSS; MariaDB; administration

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „**AZ**“).

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Podle § 47b zákona o vysokých školách souhlasím se zveřejněním své práce podle Směrnice pro vedení, vypracování a zveřejňování závěrečných prací na VŠPJ, a to bez ohledu na výsledek obhajoby.

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mě požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne 14. dubna 2026

…….

Podpis studenta

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce PaedDr. Františku Smrčkovi, Ph.D., za odborné vedení, cenné připomínky a konzultace při zpracování práce. Poděkování patří také rodině a blízkým za podporu během studia.

Obsah

Seznam obrázků

Seznam zkratek

API	Application Programming Interface
CMS	Content Management System
CRUD	Create, Read, Update, and Delete
CSRF	Cross-Site Request Forgery
ENISA	European Union Agency for Cybersecurity
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
JSON	JavaScript Object Notation
MVC	Model–View–Controller
NIST	National Institute of Standards and Technology
OWASP	Open Worldwide Application Security Project
PaaS	Platform as a Service
RBAC	Role-Based Access Control
REST	Representational State Transfer
SaaS	Software as a Service
SQL	Structured Query Language
SSL	Secure Sockets Layer
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XSS	Cross-Site Scripting

Úvod

Digitalizace představuje v současnosti významný faktor konkurenceschopnosti v oblasti gastronomických služeb. Zákazníci při výběru restaurací či ubytovacích zařízení stále častěji využívají online informace, jako je jídelní lístek, otevírací doba nebo možnost rezervace a online objednávky. Webová prezentace tak plní nejen marketingovou, ale i provozní a obchodní funkci.

Menší gastronomické podniky však často nemají k dispozici vhodný nástroj pro správu svého online obsahu. Obecné redakční systémy bývají pro jejich potřeby příliš komplexní, zatímco specializovaná řešení jsou často finančně náročná. V praxi tak vzniká potřeba jednoduchého a modulárního systému, který by odpovídal specifickým požadavkům těchto podniků.

Cílem bakalářské práce je analyzovat principy návrhu redakčních systémů, navrhnout vlastní modulární architekturu a implementovat funkční webovou aplikaci určenou pro gastronomické podniky. Důraz je kladen zejména na jednoduchost použití, přehlednost uživatelského rozhraní a možnost rozšiřování funkcionality prostřednictvím samostatných modulů.

Práce je rozdělena na teoretickou a praktickou část. Teoretická část se zaměřuje na základní principy návrhu systémů a analýzu dostupných řešení. Praktická část zahrnuje návrh architektury, implementaci systému a jeho následné testování a vyhodnocení.

1 Teoretická východiska

První kapitola vymezuje teoretické koncepty, na nichž je navržený systém založen. Zaměřuje se především na principy cloudových služeb, více-tenantní architektury, architektonických vzorů webových aplikací, návrhu databází a řízení přístupů.

1.1 Model Software as a Service (SaaS)

Software as a Service (SaaS) je servisní model cloud computingu, při němž jsou aplikace poskytovány jako služba prostřednictvím sítě. Uživatel přistupuje k aplikaci vzdáleně, aniž by spravoval infrastrukturu, operační systém nebo samotný software. SaaS je v rámci cloud computingu vymezen jako samostatný model vedle IaaS a PaaS a představuje nejvyšší úroveň abstrakce z pohledu zákazníka (Mell & Grance, 2011).

1.1.1 Charakteristika SaaS modelu

Podstatou SaaS je oddělení odpovědností mezi poskytovatelem a uživatelem. Poskytovatel zajišťuje provoz, údržbu a aktualizace aplikace i související infrastruktury, zatímco uživatel využívá aplikaci prostřednictvím standardního rozhraní, typicky webového prohlížeče. Zákazník nemá kontrolu nad síťovou infrastrukturou, servery ani úložištěm dat, ale může konfigurovat vybraná nastavení aplikace podle svých potřeb (Mell & Grance, 2011).

Obr. 1: Princip fungování modelu SaaS
Zdroj: IONOS (2023)

1.1.2 Výhody a omezení SaaS řešení

Mezi hlavní výhody SaaS řešení patří snížení počátečních investic do IT infrastruktury a přesun nákladů do provozní roviny. Organizace nemusí budovat vlastní serverové zázemí ani zajišťovat

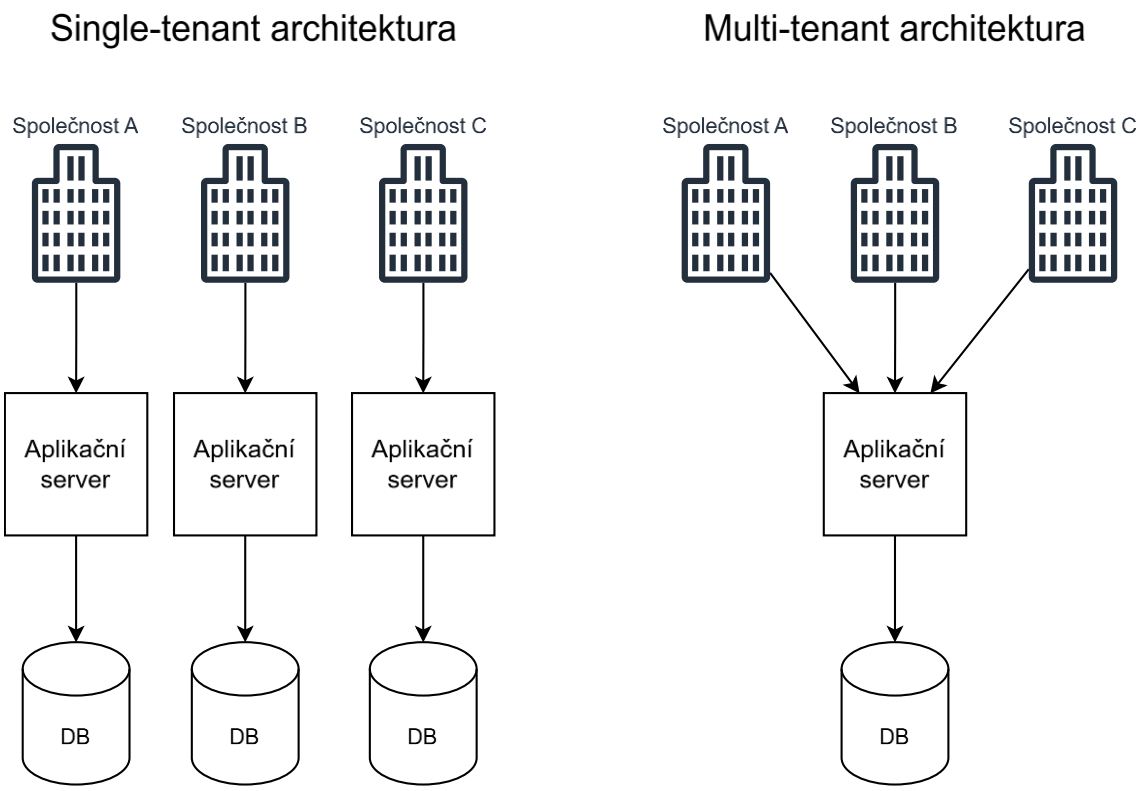
jeho správu, což umožňuje rychlejší zavedení systému a lepší škálovatelnost podle aktuální potřeby. Cloudové služby rovněž podporují dostupnost aplikací z různých zařízení a lokalit (ENISA, 2009).

Další přínos spočívá v centralizované správě systému. Aktualizace, bezpečnostní opravy i technická údržba jsou realizovány poskytovatelem služby, čímž se snižuje administrativní zatížení zákazníka. Model zároveň umožňuje efektivnější využití výpočetních zdrojů díky sdílení infrastruktury mezi více uživateli (ENISA, 2009).

Na druhé straně SaaS přináší i určitá omezení. Jedním z hlavních rizik je závislost na poskytovateli služby a dostupnosti internetového připojení. Výpadky infrastruktury nebo ukončení služby mohou mít přímý dopad na provoz organizace. ENISA dále upozorňuje na otázky bezpečnosti dat, ochrany soukromí a právní nejistoty v případě přeshraničního ukládání dat (ENISA, 2009).

1.2 Více-tenantní architektura

Více-tenantní architektura představuje způsob návrhu softwaru, při kterém jedna instance aplikace obsluhuje více zákazníků (tenantů), přičemž jejich data jsou logicky oddělena. Přístup je typický pro SaaS řešení, kde sdílení infrastruktury umožňuje efektivnější využití výpočetních zdrojů a snížení provozních nákladů. Každý tenant využívá stejnou aplikační logiku, ale pracuje se svými vlastními daty a konfigurací (Bezemer & Zaidman, 2010).



Obr. 2: Porovnání single-tenant a multi-tenant architektury v modelu SaaS

Zdroj: vlastní zpracování

1.2.1 Přístupy k oddělení dat tenantů

Oddělení dat tenantů lze realizovat několika způsoby. Základní variantou je sdílená databáze se sdílenými tabulkami, kde jsou data jednotlivých tenantů rozlišena pomocí identifikátoru (např. tenant_id). Přístup maximalizuje efektivitu využití databázových zdrojů, ale klade vyšší nároky na správné řízení přístupů a bezpečnost.

Další možností je sdílená databáze s oddělenými schématy pro jednotlivé tenanty. Model poskytuje vyšší úroveň izolace, přičemž stále umožňuje využívat jednu databázovou instanci. Nejvyšší úroveň oddělení představuje varianta samostatné databáze pro každého tenanta, která minimalizuje riziko vzájemného ovlivnění dat, avšak zvyšuje nároky na správu a infrastrukturu (Bezemer & Zaidman, 2010).

1.2.2 Výhody a rizika jednotlivých přístupů

Sdílené databázové řešení se společnými tabulkami přináší nejvyšší efektivitu a nejnižší provozní náklady, avšak vyžaduje důslednou implementaci bezpečnostních mechanismů, aby nedošlo k narušení izolace dat. Jakékoli pochybení v aplikační logice může vést k úniku informací mezi tenanty.

Naopak oddělené databáze poskytují vyšší úroveň bezpečnosti a jednodušší splnění regulatorních požadavků, avšak zvyšují komplexitu správy systému a náklady na provoz. Volba konkrétního přístupu proto závisí na požadavcích na škálovatelnost, bezpečnost a administrativní náročnost daného řešení (Bezemer & Zaidman, 2010).

1.3 Architektonické vzory webových aplikací

Volba architektury systému představuje zásadní rozhodnutí při návrhu softwarového řešení. Architektura vymezuje základní strukturu systému, definuje jeho hlavní komponenty a určuje vztahy mezi nimi. Slouží jako koncepční rámec, který propojuje specifikaci požadavků s následnou implementací. Vhodně zvolená architektura přispívá k přehlednosti systému, jeho udržitelnosti a možnosti dalšího rozvoje (Sommerville, 2016, kap. 6).

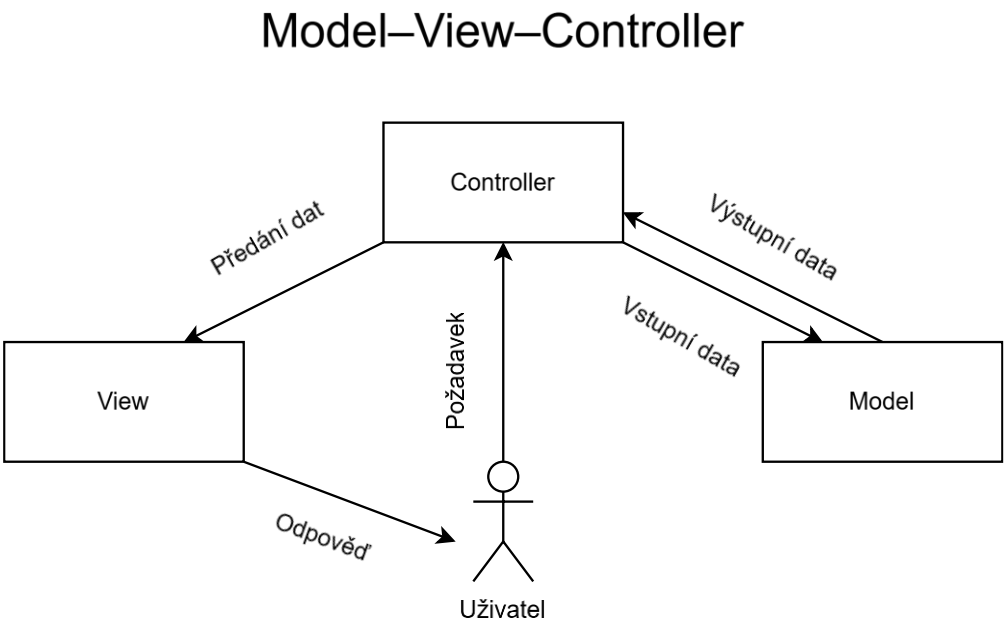
1.3.1 Monolitická architektura

Monolitická architektura označuje řešení, ve kterém je aplikace realizována jako jeden celek bez výrazného oddělení jednotlivých vrstev. Prezentační část, aplikační logika i práce s daty jsou často úzce propojeny, což umožňuje rychlou implementaci a jednoduché nasazení zejména u menších projektů. S rostoucí složitostí systému však dochází ke zhoršení přehlednosti kódu, obtížnější údržbě a omezeným možnostem rozšiřování, protože změny v jedné části aplikace mohou ovlivnit její ostatní části (Hartinger, 2020a).

1.3.2 Model–View–Controller

Model–View–Controller (MVC) je architektonický vzor založený na oddělení tří základních částí systému. Model zajišťuje práci s daty a obsahuje aplikační logiku, View slouží k prezentaci dat uživateli a Controller zprostředkovává komunikaci mezi uživatelským vstupem a aplikační logikou. Tím dochází k oddělení odpovědností mezi jednotlivými komponentami systému.

Oddělení aplikační logiky od prezentační vrstvy přispívá ke zvýšení přehlednosti systému a usnadňuje jeho údržbu i rozšiřování. Každá část může být vyvíjena a upravována relativně nezávisle na ostatních (Hartinger, 2020b).



Obr. 3: Princip architektury MVC
Zdroj: vlastní zpracování dle Verma (2023)

1.3.3 Vícevrstvá architektura

Vícevrstvá architektura představuje přístup, při kterém je systém rozdělen do několika logických vrstev. Typicky se jedná o prezentační vrstvu, aplikační vrstvu a datovou vrstvu. Každá vrstva má jasně vymezenou odpovědnost a komunikuje pouze s bezprostředně sousední vrstvou.

Model umožňuje oddělit zpracování dat od jejich prezentace a podporuje modularitu systému. Díky jasně definovaným hranicím mezi vrstvami je možné jednotlivé části systému upravovat nebo nahrazovat bez zásadního zásahu do ostatních komponent (Hartinger, 2020c).

1.3.4 Porovnání architektonických přístupů

Monolitická, vícevrstvá architektura a architektura MVC se liší zejména mírou oddělení jednotlivých částí systému a způsobem jejich organizace. Monolitické řešení integruje všechny části aplikace do jednoho celku, zatímco vícevrstvá architektura rozděluje systém do samostatných logických vrstev. MVC dále zpřesňuje strukturu prezentační části tím, že odděluje zpracování dat, uživatelské rozhraní a řízení toku požadavků.

V praxi mohou být jednotlivé principy kombinovány, například implementací MVC v rámci prezentační vrstvy vícevrstvé architektury. Takové řešení podporuje přehlednost, testovatelnost a dlouhodobou udržitelnost systému (Sommerville, 2016, kap. 4).

1.4 Návrh databázových systémů

Návrh databázového systému představuje klíčovou součást vývoje informačního systému, protože určuje způsob organizace, ukládání a správy dat. Správně navržená databáze musí zajistit konzistenci dat, minimalizovat redundanci a podporovat efektivní zpracování dotazů. Databázový návrh obvykle zahrnuje logický model dat, definici klíčů a vztahů mezi entitami (Silberschatz, Korth a Sudarshan, 2019, kap. 1).

1.4.1 Relační databázový model

Relační databázový model organizuje data do relací, které jsou reprezentovány tabulkami. Každá relace se skládá z atributů (sloupců) a n-tic (řádků), přičemž každá n-tice představuje konkrétní záznam. Základním principem relačního modelu je práce s primárními klíči, které jednoznačně identifikují jednotlivé záznamy, a s cizími klíči, které umožňují vytvářet vztahy mezi tabulkami (Silberschatz, Korth a Sudarshan, 2019, kap. 2).

Relační model je založen na formálních základech relační algebry a umožňuje manipulaci s daty prostřednictvím strukturovaného dotazovacího jazyka. Díky oddělení logické struktury od fyzického uložení dat podporuje nezávislost aplikace na konkrétní implementaci databázového systému (Silberschatz, Korth a Sudarshan, 2019, kap. 2).

1.4.2 Normalizace dat

Normalizace představuje systematický proces úpravy struktury databáze s cílem odstranit redundanci a zabránit vzniku anomálií při vkládání, aktualizaci nebo mazání dat. Proces je založen na analýze funkčních závislostí mezi atributy a je formalizován prostřednictvím jednotlivých normalizačních forem (Silberschatz, Korth a Sudarshan, 2019, kap. 7).

Dodržení třetí normální formy (3NF) je v praxi často považováno za dostatečné pro zajištění konzistence dat a omezení redundance. Vyšší normalizační formy dále zpřísňují podmínky závislostí mezi atributy, avšak mohou vést ke zvýšení složitosti databázové struktury (Silberschatz, Korth a Sudarshan, 2019, kap. 7).

1.4.3 Integrita a vazby mezi entitami

Integrita databáze zajišťuje správnost a konzistenci uložených dat. Entitní integrita vyžaduje, aby každý záznam byl jednoznačně identifikován primárním klíčem, zatímco referenční integrita zabezpečuje, že vztahy mezi tabulkami definované prostřednictvím cizích klíčů odkazují pouze na existující záznamy (Silberschatz, Korth a Sudarshan, 2019, kap. 2 a 4).

Vazby mezi entitami umožňují modelovat vztahy typu jeden-ku-jednomu, jeden-ku-mnoha nebo mnoho-ku-mnoha. Správná definice těchto vazeb je nezbytná pro zachování logické konzistence dat a správné fungování aplikační logiky nad databází (Silberschatz, Korth a Sudarshan, 2019, kap. 2 a 4).

1.5 Řízení přístupů v informačních systémech

Řízení přístupů představuje soubor mechanismů, jejichž cílem je zajistit, aby uživatelé systému měli přístup pouze k těm zdrojům a operacím, ke kterým jsou oprávněni. Správně nastavený model řízení přístupů přispívá k ochraně dat, prevenci neoprávněných zásahů a omezení bezpečnostních rizik. V informačních systémech je řízení přístupů realizováno prostřednictvím definice oprávnění, rolí a pravidel, která určují, kdo může provádět konkrétní operace (Sandhu, Ferraiolo a Kuhn, 2000).

1.5.1 Role-based access control (RBAC)

Role-Based Access Control (RBAC) je model řízení přístupů, který přiřazuje oprávnění nikoli přímo uživatelům, ale rolím. Uživatel je následně přiřazen k jedné nebo více rolím, které určují jeho oprávnění v systému. Přístup zjednodušuje správu oprávnění, zejména v rozsáhlejších systémech s větším počtem uživatelů (Sandhu, Ferraiolo a Kuhn, 2000).

Model RBAC rozlišuje základní prvky: uživatele, role, oprávnění a relace mezi nimi. Standardizovaný model NIST dále zavádí hierarchii rolí a omezení, která umožňují jemnější kontrolu přístupových práv. RBAC je dnes považován za jeden z nejrozšířenějších modelů řízení přístupů v podnikových informačních systémech (Sandhu, Ferraiolo a Kuhn, 2000).

1.5.2 Princip nejmenších oprávnění

Princip nejmenších oprávnění (principle of least privilege) vychází z požadavku, aby uživatelé i procesy měli pouze taková oprávnění, která jsou nezbytná pro vykonávání jejich úkolů. Cílem je minimalizovat potenciální škody v případě chyby, zneužití nebo bezpečnostního incidentu.

Uplatnění tohoto principu v rámci modelu RBAC znamená pečlivé definování rolí a přiřazování pouze nezbytných oprávnění. Tím se omezuje riziko neoprávněného přístupu k citlivým datům a snižuje se pravděpodobnost eskalace oprávnění v systému (Sandhu, Ferraiolo a Kuhn, 2000).

1.6 Bezpečnost webových aplikací

Bezpečnost webových aplikací představuje soubor opatření a mechanismů, jejichž cílem je ochrana aplikace a jejích dat před neoprávněným přístupem, manipulací nebo zneužitím. Webové aplikace jsou vystaveny širokému spektru hrozeb, které mohou vést k úniku citlivých informací, narušení integrity dat nebo omezení dostupnosti systému. Z tohoto důvodu je nezbytné implementovat odpovídající bezpečnostní mechanismy již ve fázi návrhu systému (OWASP, 2025).

1.6.1 Autentizace a autorizace

Autentizace představuje proces ověřování identity uživatele, zatímco autorizace určuje, k jakým zdrojům a operacím má daný uživatel přístup. Správná implementace těchto mechanismů je klíčová pro ochranu citlivých dat a omezení neoprávněného přístupu.

OWASP uvádí, že nedostatečné řízení autentizace a autorizace patří mezi nejzávažnější bezpečnostní rizika webových aplikací. Chyby v těchto mechanismech mohou vést k převzetí účtů, obcházení přístupových kontrol nebo neoprávněné manipulaci s daty (OWASP, 2025).

1.6.2 Ochrana proti běžným útokům

Webové aplikace jsou ohroženy řadou útoků, mezi které patří zejména SQL injection, cross-site scripting (XSS) nebo porušení řízení přístupu. Tyto útoky mohou být způsobeny nedostatečnou validací vstupních dat, chybami v implementaci autentizačních mechanismů nebo nesprávnou konfigurací systému.

OWASP identifikuje jako nejkritičtější rizika například injection útoky, porušení řízení přístupu nebo kryptografické chyby. Implementace bezpečnostních opatření, jako je validace vstupů, použití parametrizovaných dotazů, bezpečné ukládání hesel a správné řízení relací, významně snižuje riziko těchto útoků (OWASP, 2025).

1.7 Integrace externích služeb

Integrace externích služeb představuje propojení informačního systému s dalšími softwarovými systémy prostřednictvím standardizovaných komunikačních rozhraní. V moderních webových aplikacích se jedná zejména o napojení na platební brány, e-mailové služby, účetní systémy nebo analytické nástroje, které umožňují rozšířit funkcionalitu aplikace bez nutnosti implementace těchto mechanismů přímo v jejím jádru (Fielding, 2000).

API (Application Programming Interface) slouží jako rozhraní pro komunikaci mezi jednotlivými systémy. V prostředí webových aplikací je běžně využíván princip REST, který je založen na standardních HTTP metodách a bez stavové komunikaci (Fielding, 2000).

2 Analýza problému a požadavků

Cílem kapitoly je porovnat vybraná řešení pro správu webových stránek gastronomických podniků, identifikovat jejich omezení a na tomto základě stanovit požadavky na navrhovaný systém. Analýza se zaměřuje na systémy WordPress, UpMenu a Choice.

2.1 Analýza současných řešení

Redakční systémy (CMS) představují základní nástroj pro tvorbu a správu webových stránek bez nutnosti znalosti programování. Umožňují uživatelům jednoduše přidávat a upravovat obsah, spravovat strukturu webu a měnit jeho vzhled. Díky tomu se staly široce rozšířeným řešením nejen mezi vývojáři, ale i mezi běžnými uživateli (Teaganne a Downie, 2025).

V gastronomickém prostředí mají CMS specifický význam. Web restaurace dnes slouží nejen k prezentaci nabídky a otevírací doby, ale také k online komunikaci se zákazníky – od rezervací stolů až po objednávky jídla s doručením. Pandemie COVID-19 a následná digitalizace gastro sektoru významně zvýšily poptávku po jednoduchých online řešeních, která zvládne obsluhovat i provozovatel bez technických znalostí (Bedřich, 2020).

Pro účely této práce byly vybrány tři zástupné systémy, které reprezentují odlišné přístupy k tvorbě webů. WordPress představuje univerzální open-source redakční systém s vysokou mírou přizpůsobitelnosti. UpMenu je specializovaná SaaS platforma zaměřená na restaurace a online objednávky. Choice představuje české řešení pro digitalizaci gastronomických podniků. Hodnoceny byly zejména funkční možnosti, uživatelská přívětivost, technická náročnost, lokalizace a ekonomická dostupnost.

2.2 Přehled dostupných systémů

Podkapitola se zaměřuje na detailní rozbor vybraných systémů z hlediska jejich funkcionality, uživatelské přívětivosti, technické náročnosti a ekonomické dostupnosti. Analýza je provedena z pohledu provozovatele menšího gastronomického podniku.

2.2.1 WordPress

WordPress je open-source redakční systém, který patří mezi nejpoužívanější systémy na světě a díky své otevřenosti a rozsáhlé komunitě umožňuje vytvářet i webové prezentace pro gastronomické podniky. Jeho hlavní výhodou je vysoká přizpůsobitelnost a široká nabídka pluginů, například pro správu menu, rezervací nebo online prodej. Nevýhodou je vyšší technická náročnost správy, závislost na pluginech třetích stran a riziko nekompatibility mezi rozšířeními. Základní provozní náklady jsou sice nízké a odvíjejí se především od hostingu a domény, avšak při využití placených doplňků nebo externí správy se mohou výrazně zvýšit. Z pohledu této práce je WordPress vhodný spíše pro technicky zkušenější uživatele než pro běžného provozovatele menšího gastronomického podniku (Gupta, 2025; WordPress, 2025; Wedos, 2025).

2.2.2 UpMenu

UpMenu je mezinárodní SaaS platforma zaměřená výhradně na gastronomický sektor. Nabízí tvorbu webových stránek, správu menu, online objednávky, rezervace, marketingové nástroje i další doplňkové funkce v rámci jednoho systému. Jeho hlavní výhodou je jednoduché ovládání, rychlé nasazení a skutečnost, že uživatel nemusí řešit hosting ani technickou správu. Nevýhodou je uzavřenost systému, omezené možnosti hlubších úprav, administrace dostupná pouze v angličtině a také pravidelné měsíční poplatky, které mohou být pro menší podniky zatěžující, přestože zároveň umožňují rozložit náklady v čase a eliminují potřebu počáteční investice. Z pohledu této práce jde o funkčně silné řešení, které však není plně přizpůsobené lokálním podmínkám (UpMenu, 2025).

2.2.3 Choice

Choice je česká SaaS platforma zaměřená na digitalizaci gastronomických podniků. Umožňuje správu online menu, objednávek, rezervací, plateb i marketingových funkcí a podporuje také integrace s rozvozovými a pokladními systémy. Výhodou je lokalizace pro české prostředí, jednoduché používání a sjednocení více funkcí do jednoho systému. Omezením je uzavřená struktura, menší možnost individuálního přizpůsobení webu a pravidelné měsíční náklady. Choice je proto vhodné zejména pro podniky, které hledají rychlé a jednoduše použitelné řešení, ale nevyžadují větší míru technických možností (StartupJobs, 2025; Choice, 2025).

2.2.4 Souhrnné porovnání systémů

Tab. 1: Souhrnné porovnání vybraných systémů

Systém	**Typ řešení**	**Hlavní výhody**	**Hlavní nevýhody**	**Vhodnost pro cílového uživatele**
WordPress	open-source CMS	vysoká přizpůsobitelnost, široká komunita, nízké základní náklady	vyšší technická náročnost, závislost na pluginech, riziko nekompatibility	spíše nižší
UpMenu	SaaS	jednoduché ovládání, široká funkcionalita, bez nutnosti hostingu	uzavřenost systému, vyšší cena, anglická administrace	spíše nižší kvůli absenci lokalizace
Choice	SaaS	česká lokalizace, jednoduché použití, integrace pro gastro	omezené přizpůsobení, uzavřenost, měsíční poplatky	vysoká, s omezeními

Zdroj: vlastní zpracování

Z porovnání vyplývá, že analyzované systémy sice nabízejí širokou funkcionalitu, avšak často jsou omezeny uzavřeností, vyššími náklady nebo omezenými možnostmi přizpůsobení konkrétním požadavkům podniku. Tyto skutečnosti představují motivaci pro návrh vlastního řešení, které by tyto nedostatky eliminovalo.

2.3 Komparativní vyhodnocení analyzovaných systémů

Z provedené analýzy vyplývá, že WordPress nabízí nejvyšší míru přizpůsobitelnosti, avšak za cenu vyšší technické náročnosti a závislosti na externích rozšířeních. UpMenu a Choice naopak poskytují uživatelsky jednodušší a provozně pohodlnější řešení, ale současně omezují možnosti individuálních úprav a zvyšují závislost na poskytovateli služby. Pro návrh vlastního systému je proto vhodné spojit jednoduchost ovládání typickou pro SaaS platformy s možností přizpůsobení funkcionality, vzhledu a integrací, a zároveň zajistit lokalizaci pro české prostředí.

2.4 Charakteristika cílového uživatele

Cílový uživatel informačního systému představuje osobu nebo skupinu osob, které systém využívají k plnění svých pracovních úkolů. Při návrhu systému je důležité identifikovat jednotlivé skupiny uživatelů, jejich role a očekávání, protože tyto faktory ovlivňují požadavky na funkčnost, přístupová práva i způsob ovládání systému. Uživatelé jsou v rámci softwarového inženýrství považováni za klíčové stakeholdery, jejichž potřeby musí být zohledněny již ve fázi analýzy požadavků (Sommerville, 2016, kap. 4).

2.4.1 Typologie uživatelů a jejich potřeby

Informační systémy obvykle rozlišují uživatele podle jejich role a úrovně odpovědnosti. Mezi základní skupiny patří koncoví uživatelé, administrátoři systému a manažeři.

Koncoví uživatelé pracují se systémem při každodenních operacích a kladou důraz na jednoduché a přehledné ovládání, rychlou odezvu a minimalizaci chyb při zadávání dat. Administrátoři jsou odpovědní za správu systému, konfiguraci oprávnění a dohled nad jeho provozem, proto vyžadují zejména kontrolu nad přístupovými právy a možnost nastavení systému. Manažeři využívají systém především pro získávání přehledných výstupů a agregovaných dat, která podporují rozhodování (Sommerville, 2016).

2.5 Funkční požadavky systému

Kapitola shrnuje požadavky na navrhovaný redakční systém určený pro gastronomické podniky. Požadavky vycházejí z předchozí analýzy existujících řešení a reflektují jejich silné i slabé stránky. Cílem je definovat funkce a vlastnosti systému tak, aby byl jednoduchý, stabilní a zároveň dostatečně přizpůsobitelný.

Požadavky na systém nebyly stanoveny pouze na základě teoretické analýzy, ale vycházejí také z reálných zkušeností z provozu gastronomických zařízení. Tyto zkušenosti reflektují praktické potřeby uživatelů při každodenní práci se správou menu, rezervací a objednávek.

Identifikované požadavky tak kombinují poznatky z analýzy existujících řešení s reálnými problémy z praxe, jako je složitost správy obsahu, omezená přizpůsobitelnost nebo nedostatečná přehlednost systémů.

Pro stanovení priorit požadavků byla využita metoda MoSCoW, která rozděluje požadavky na nezbytné (Must have), doporučené (Should have) a volitelné (Could have) (Agile Business Consortium, [s. d.]).

2.5.1 Funkční požadavky

Funkční požadavky určují, co má systém konkrétně dělat a jaké činnosti musí umožnit uživatelům. Vymezují jednotlivé funkce, služby a procesy, které jsou nezbytné k naplnění účelu systému. Požadavky popisují chování systému v různých situacích a specifikují reakce na uživatelské vstupy. V praxi tak zahrnují například správu obsahu, objednávek či rezervací (GeeksforGeeks, 2024).

Must have (nezbytné funkce)

Funkce, bez nichž systém nemůže plnit svůj základní účel.

- Správa obsahu webu – vytváření, úprava a mazání textových sekcí, fotografií a nabídek.
- Správa jídelního lístku – přidávání, editace a kategorizace položek menu.
- Vizuální editor stránek – nástroj pro úpravu vzhledu bez nutnosti programování.
- Tvorba a správa rezervací – možnost spravovat rezervace stolů.
- Online objednávky – přijímání a správa objednávek pro rozvoz nebo osobní odběr.
- Základní uživatelská autentizace – přihlášení majitele či zaměstnanců restaurace do administračního rozhraní.

Should have (doporučené funkce)

Funkce, které výrazně zvyšují komfort a efektivitu používání systému.

- Vícejazyčnost – možnost vytvářet a spravovat obsah alespoň ve dvou jazycích.
- Základní analytika – přehled návštěvnosti webu, počtu objednávek nebo rezervací.
- Galerie – možnost nahrávání a správy fotografií interiéru, pokrmů nebo akcí.
- Export a import dat – export objednávek nebo rezervací do souboru CSV/Excel.
- Automatické potvrzení rezervace e-mailem – zasílání notifikací zákazníkům.

Could have (volitelné funkce)

Doplňkové funkce, které nejsou nezbytné, ale mohou systém dále rozšířit.

- Integrace platební brány pro online platby.
- Napojení na externí rozvozové platformy (např. Wolt, Bolt Food).
- Napojení na pokladní systémy (např. Dotykačka).
- Správa více provozoven – možnost vytvářet více profilů v rámci jednoho účtu.
- Napojení na Google Maps – synchronizace otevírací doby a recenzí.

2.5.2 Nefunkční požadavky

Nefunkční požadavky se zaměřují na vlastnosti a kvalitu fungování systému, nikoli na jeho konkrétní funkce. Definují podmínky, za kterých systém vykonává své funkce, například požadavky na rychlost, bezpečnost, dostupnost, použitelnost nebo rozšiřitelnost. Aspekty mají zásadní vliv na uživatelský komfort a dlouhodobou udržitelnost systému (TechTarget, 2023).

Must have (nezbytné vlastnosti)

Základní požadavky, které musí být splněny, aby byl systém funkční a použitelný v praxi.

- Přehledné a intuitivní uživatelské rozhraní – ovládání musí být snadné i pro uživatele bez technických znalostí.

- Plná lokalizace do češtiny – všechna rozhraní, systémové zprávy i nápovědy musejí být v českém jazyce.
- Responzivní design – správné zobrazení a plná funkčnost na mobilních zařízeních i PC.
- Stabilita a spolehlivost – systém musí zvládat běžný provoz bez výpadků nebo ztráty dat.
- Bezpečnost dat – šifrované přenosy, zabezpečené přihlášení a oddělení uživatelských oprávnění.

Should have (doporučené vlastnosti)

Požadavky, které přispívají ke komfortu a efektivitě používání systému.

- Možnost budoucího rozšíření – architektura systému by měla umožnit doplňování nových modulů a funkcí.
- Rychlá odezva a optimalizace výkonu – krátká doba načítání stránek a minimální zátěž serveru.
- Jednotný vizuální styl – sjednocené ovládací prvky a grafika napříč všemi moduly.
- Kompatibilita s běžnými prohlížeči – zajištění správného zobrazení v moderních verzích Chrome, Firefox a Safari.
- Základní uživatelská podpora – dostupná dokumentace nebo integrovaná nápověda v rozhraní.

Could have (volitelné vlastnosti)

Doplňkové požadavky, které nejsou nutné, ale zvyšují uživatelský komfort.

- Tmavý a světlý režim rozhraní – volba vzhledu podle preferencí uživatele.
- Základní offline režim – načtení posledních známých dat i bez připojení k internetu.
- Personalizace rozhraní – možnost upravovat barvy nebo uspořádání panelů v administraci.
- Notifikace v reálném čase – upozornění na nové objednávky nebo rezervace.
- Export logů a diagnostických dat – pro snadnější řešení technických problémů.

2.5.3 Uživatelské role a oprávnění

Správa uživatelských oprávnění by měla umožňovat definici přístupových práv k jednotlivým částem systému tak, aby uživatelé měli přístup pouze k funkcím odpovídajícím jejich roli. V systému je proto vhodné uvažovat řízení přístupu založené na rolích, které umožňuje přiřazovat oprávnění ke konkrétním funkcím a sekcím administrace.

Systém by měl podporovat rozdělení administrace do logických celků, například obsah webu, menu nebo nastavení, a umožnit řízení přístupů na úrovni těchto částí. Uživatelé by tak mohli mít oprávnění pouze ke čtení dat nebo i k jejich úpravě v závislosti na své roli.

Přístup přispívá ke zvýšení bezpečnosti a zároveň umožňuje přizpůsobení systému potřebám konkrétního podniku. Uvedený princip odpovídá modelu Role-Based Access Control (RBAC), který je běžně využíván v informačních systémech (IBM, [s. d.]).

3 Návrh architektury systému

Na základě provedené analýzy požadavků a komparace existujících řešení byla navržena architektura systému odpovídající potřebám modulárního redakčního systému pro gastronomické podniky. Cílem návrhu je zajistit přehlednost struktury, možnost budoucího rozšiřování, oddělení aplikační logiky od prezentační vrstvy a efektivní řízení uživatelských oprávnění.

3.1 Zvolená architektura systému

Pro realizaci navrhovaného CMS byla zvolena architektura Model–View–Controller (MVC), která umožňuje jasné oddělení práce s daty, aplikační logiky a prezentační vrstvy. Toto řešení podporuje modularitu systému, snadnou údržbu a možnost postupného rozšiřování funkcionality bez zásahu do jádra aplikace.

V rámci návrhu bylo současně zachováno vícevrstvé členění systému na:

- prezentační vrstvu,
- aplikační logiku,
- datovou vrstvu.

3.1.1 Přehled používaných architektonických přístupů v PHP

Při návrhu systému byly zvažovány tři základní přístupy: monolitická architektura, vícevrstvá architektura a architektura Model–View–Controller (MVC).

Monolitická architektura představuje kompaktní řešení, ve kterém jsou jednotlivé části aplikace úzce provázány. Přístup je vhodný pro menší projekty, avšak s rostoucí komplexitou dochází ke zhoršení udržitelnosti a rozšiřitelnosti systému.

Vícevrstvá architektura odděluje prezentační, aplikační a datovou vrstvu, čímž zvyšuje přehlednost a modularitu systému. Sama o sobě však neřeší řízení toku požadavků mezi uživatelem a aplikací.

Architektura MVC rozšiřuje vícevrstvý přístup o řízení toku požadavků prostřednictvím kontrolerů. Umožňuje jasné oddělení odpovědností a podporuje dlouhodobou udržitelnost systému.

3.1.2 Porovnání architektur z hlediska potřeb CMS systému

Při výběru architektury byly hodnoceny následující faktory:

- výkon,
- modularita a možnost budoucího rozšíření,
- přehlednost a čitelnost kódu,
- oddělení logiky od šablon,
- možnost implementace detailního řízení oprávnění,
- dlouhodobá udržitelnost systému.

Na základě těchto kritérií byla zvolena architektura MVC, která nejlépe odpovídá požadavkům definovaným v kapitole 2. Přístup umožňuje přidávat nové moduly bez zásahu do jádra systému, odděluje prezentační část od aplikační logiky a podporuje přehlednou strukturu projektu.

Monolitické řešení bylo vyhodnoceno jako nevhodné z důvodu nízké modularity a obtížné rozšiřitelnosti. Vícevrstvá architektura sice přináší oddělení vrstev, avšak neposkytuje tak efektivní řízení toku požadavků jako MVC.

Tab. 2: Porovnání architektur z pohledu potřeb CMS

Architektura	**Monolitická**	**Vícevrstvá**	**MVC**
Výkon	vysoký u malých systémů, klesá s rostoucí složitostí	stabilní, ale závislý na počtu vrstev	vyvážený, vhodný i pro větší systémy
Srozumitelnost kódu	nízká, kód bývá provázaný	dobrá, logika oddělena po vrstvách	vysoká, jasné rozdělení
Rozšiřitelnost	slabá, změna může ovlivnit celý systém	dobrá, vrstvy lze doplňovat	velmi dobrá, snadné přidávání modulů
Údržba a testování	obtížné kvůli provázanosti	jednodušší díky vrstvení	snadné, každá část testovatelná samostatně
Oddělení logiky od šablon	slabé	částečné	výborné
Správa práv a uživatelů	často pevně integrovaná	možná na aplikační úrovni	snadno řešitelná na aplikační úrovni
Vhodnost pro CMS	nízká	dobrá	**nejvhodnější**

Zdroj: vlastní zpracování

3.1.3 Uplatnění architektury MVC v navrženém systému

Navržený systém je strukturován podle principů MVC do tří základních částí:

Model

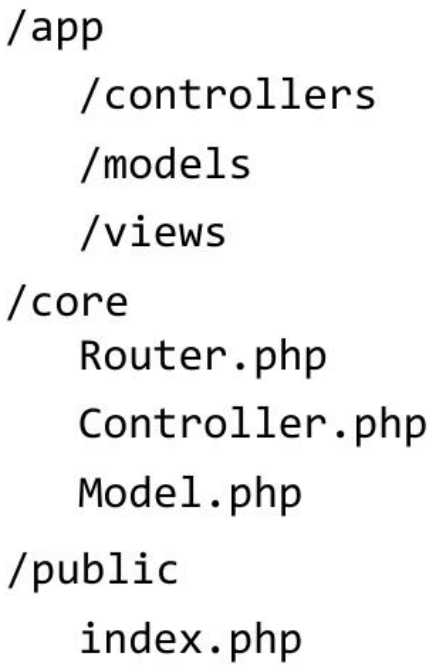
Obsahuje třídy zodpovědné za komunikaci s databází a práci s entitami systému (např. stránka, uživatel, rezervace, objednávka). Každý model implementuje základní operace typu CRUD a zapouzdřuje práci s databázovou vrstvou.

View

Zahrnuje prezentační šablony odpovědné za výstup dat uživateli. Šablony obsahují minimální množství aplikační logiky a pracují výhradně s daty předanými z kontroleru.

Controller

Zajišťuje zpracování požadavků uživatele, volání příslušných modelů a předání dat prezentační vrstvě. Každý controller reprezentuje konkrétní funkční oblast systému (např. správa stránek, administrace, rezervace).



Obr. 4: Adresářová struktura MVC architektury
Zdroj: vlastní zpracování

Směrování požadavků

Požadavky veřejné části aplikace procházejí centrálním vstupním bodem, který zajišťuje inicializaci aplikace a předání řízení směrovací logice. Směrovací logika na základě URL adresy určuje odpovídající controller a metodu.

Použitý přístup umožňuje centralizované řízení aplikace, oddělení směrovací logiky a snadné rozšiřování systému o nové funkční části.

Tok zpracování požadavku

Proces zpracování požadavku probíhá v následujících krocích:

1. Uživatel odešle požadavek prostřednictvím webového rozhraní.
2. Router identifikuje odpovídající controller a metodu.
3. Controller načte nebo upraví data prostřednictvím modelu.
4. Data jsou předána prezentační vrstvě.
5. Výsledný HTML výstup je odeslán uživateli.

Takto navržený mechanismus zajišťuje jasné oddělení odpovědností a podporuje budoucí rozšiřování systému bez zásahu do existujících částí aplikace.

3.2 Návrh více-tenantního řešení

Více-tenantní řešení v systému GastroHub je navrženo tak, aby jedna nasazená instance aplikace obsluhovala více zákazníků (tenantů) s logickým oddělením dat. V praxi je tenant v systému reprezentován záznamem v tabulce projects a identifikován primárně podle domény požadavku (HTTP_HOST), která se porovnává s hodnotou projects.domain. Zvolený přístup odpovídá modelu „pooled“, v němž více tenantů sdílí jednu aplikaci a společnou datovou vrstvu, zatímco oddělení dat je zajištěno návrhem databáze a aplikační logikou.

3.2.1 Zvažované varianty

V oblasti více-tenantních SaaS systémů se běžně uvažují tři základní přístupy:

1. Sdílená databáze a sdílené schéma – všechny projekty využívají stejné tabulky, data jsou oddělena pomocí identifikátoru tenanta.
2. Sdílená databáze s oddělenými schématy – každý tenant má vlastní schéma v rámci jedné databáze.
3. Samostatná databáze pro každého tenanta – každý tenant má vlastní databázovou instanci.

Sdílené schéma představuje provozně nejefektivnější řešení, avšak vyžaduje důslednou kontrolu izolace dat. Oddělená schémata zvyšují míru izolace, ale komplikují správu a migrace. Varianta databáze pro každého tenanta poskytuje nejvyšší úroveň izolace, zároveň však výrazně zvyšuje provozní náročnost.

3.2.2 Zdůvodnění zvoleného přístupu

V systému GastroHub byl zvolen model sdílené databáze a sdílených tabulek s oddělením dat pomocí identifikátoru project_id. Identifikace tenanta na veřejné části aplikace probíhá na základě domény, která je mapována na odpovídající záznam v tabulce projects. V administraci je aktivní projekt určován samostatně, což umožňuje správu více projektů v rámci jednoho rozhraní.

Hlavním důvodem volby je provozní jednoduchost a efektivita vývoje. Jednotná kódová základna, společné databázové migrace a centralizovaná správa nasazení umožňují rychlejší rozšiřování systému a snížení provozních nákladů. Modulární funkcionalita je řízena na úrovni projektu pomocí konfiguračních mechanismů.

Zvolený model klade zvýšené nároky na správnou implementaci izolace dat. Každý dotaz nad tenantově závislými tabulkami musí zahrnovat filtr podle project_id a kontext tenanta musí být jednoznačně definován v rámci celého požadavku. Nedodržení těchto principů může vést k neoprávněnému přístupu k datům jiného tenanta.

3.3 Návrh datového modelu

Datový model systému GastroHub je navržen jako relační databázový model implementovaný v systému MariaDB. Návrh vychází z požadavků na modularitu, rozšiřitelnost a podporu více projektů v rámci jedné databázové instance.

Centrální entitou modelu je tabulka projects, která reprezentuje jednotlivé zákazníky systému. Ostatní části databáze jsou na tuto entitu navázány pomocí atributu project_id, což umožňuje logické oddělení dat jednotlivých provozoven.

Datový model kombinuje projektově oddělená data, globální systémové entity a modulární části odpovídající hlavním funkčním oblastem systému, jako jsou menu, objednávky, rezervace nebo analytika. Návrh je veden s důrazem na přehlednost struktury, omezení redundance a možnost dalšího rozšiřování systému.

3.3.1 Hlavní skupiny entit

Databázový model lze rozdělit do několika základních skupin:

- Projekty a konfigurace – evidence projektů a jejich nastavení, včetně aktivace jednotlivých modulů.
- Uživatelé a oprávnění – správa administrátorů, rolí a oprávnění na základě RBAC modelu.
- Provozní moduly – data související s menu, objednávkami, rezervacemi a případně hotelovým provozem.
- Obsah a marketing – správa novinek, událostí a dalších obsahových prvků.
- Analytická data – agregované statistiky návštěvnosti a provozu.

Struktura modelu reflektuje modulární charakter systému, kdy jednotlivé části mohou být aktivovány nebo deaktivovány podle potřeb konkrétního projektu.

3.3.2 Návrh vztahů mezi entitami

Datový model využívá především vztahy typu 1:N a N:M.

Vztahy 1:N se uplatňují zejména mezi projektem a jeho daty (např. objednávky, rezervace, analytika) a mezi nadřazenými a podřízenými entitami (např. kategorie menu a položky menu, objednávky a jejich položky).

Vztahy N:M jsou realizovány pomocí propojovacích tabulek, například mezi rolemi a oprávněními nebo mezi rezervacemi a stoly.

Referenční integrita je zajištěna pomocí cizích klíčů u klíčových vazeb, zatímco u vybraných částí systému je vazba řešena na aplikační úrovni s ohledem na flexibilitu návrhu.

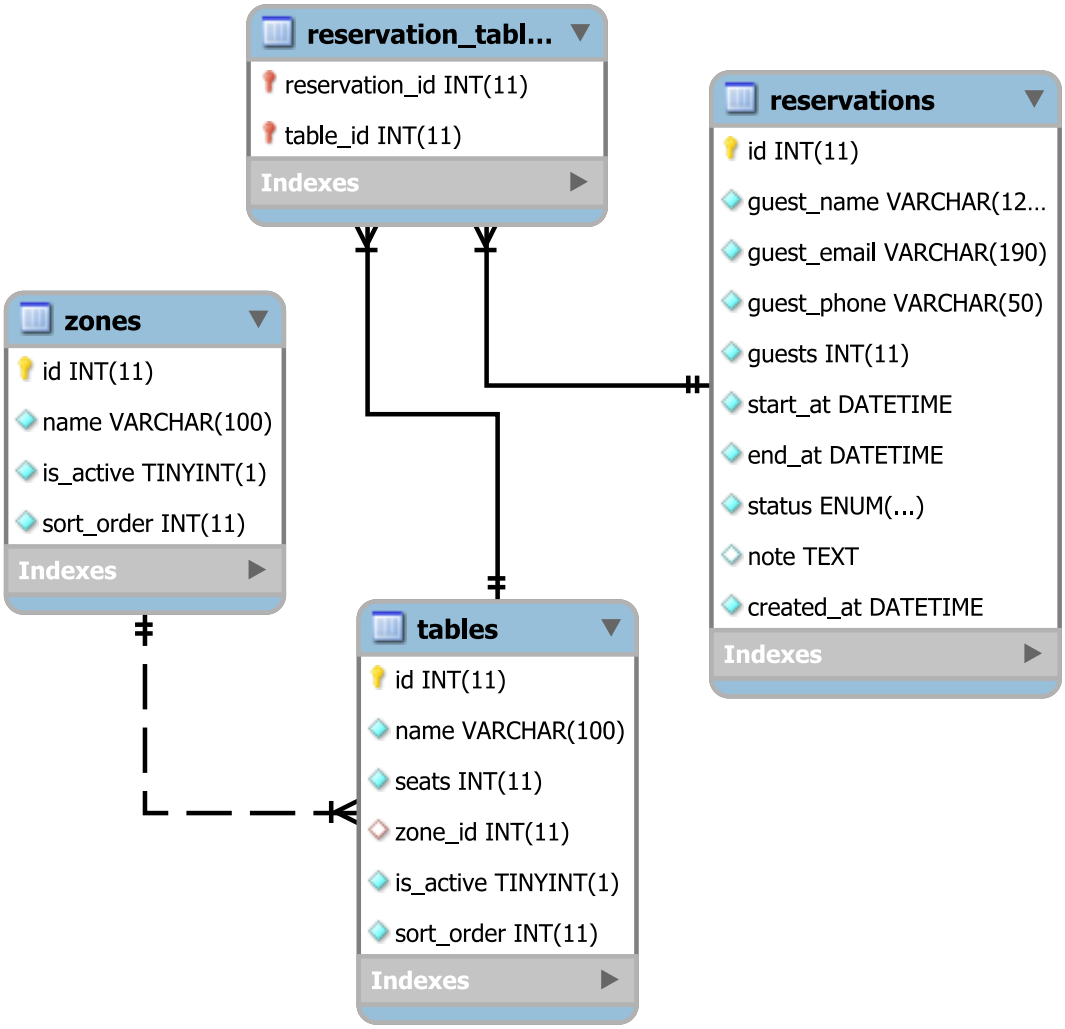
3.3.3 Logický model databáze

Datový model je navržen s důrazem na oddělení hlavních entit a omezení redundance dat. Samostatně jsou vedeny například objednávky a jejich položky, kategorie menu a položky menu nebo role a oprávnění.

Primární klíče jsou realizovány jako jednoznačné identifikátory. Vazby mezi entitami jsou implementovány pomocí cizích klíčů nebo logických vazeb na aplikační úrovni.

Podpora více projektů je zajištěna prostřednictvím centrální entity projects a atributu project_id, který je přítomen u vybraných částí databáze. Některé entity, například systém oprávnění, jsou sdílené napříč projekty.

Takto navržený model umožňuje provoz více zákazníků v rámci jedné databáze při zachování logického oddělení dat a současně podporuje modulární rozšiřování systému.



Obr. 5: Zjednodušený datový model rezervačního modulu systému GastroHub

Zdroj: vlastní zpracování

Kompletní datový model systému je uveden v příloze 1.

3.4 Návrh modulární struktury systému

Modulární struktura systému GastroHub je navržena tak, aby umožňovala postupné rozšiřování funkcionality bez nutnosti zásahů do jádra aplikace. Jednotlivé části systému jsou odděleny podle funkčních domén a mohou být vyvíjeny nezávisle.

Modularita je založena na třech hlavních principech:

- oddělení aplikační logiky podle domén v rámci architektury MVC,
- oddělení administrační a veřejné části aplikace,
- řízení dostupnosti funkcí na úrovni projektu pomocí feature toggles.

Aplikační struktura je rozdělena na jádro systému, veřejnou část a administrační rozhraní. Jádro zajišťuje základní infrastrukturu (routing, konfigurace, základní vrstvy aplikace), zatímco

jednotlivé moduly jsou implementovány jako samostatné doménové celky napříč vrstvami systému.

3.4.1 Princip oddělení modulů

Každý modul je realizován jako soubor spolupracujících částí napříč jednotlivými vrstvami aplikace – modelovou, řídicí a prezentační. Součástí modulu je také administrační rozhraní a případné sdílené služby.

Takové rozdělení umožňuje rozšiřovat funkcionalitu bez zásahu do nesouvisejících částí systému a zároveň zachovává konzistentní strukturu napříč všemi moduly.

Administrace je oddělena od veřejné části aplikace a využívá vlastní vstupní bod, autentizaci a routing. Oddělení zvyšuje bezpečnost a přehlednost kódu, protože logika správy systému není promíchána s veřejnými funkcemi.

Dostupnost jednotlivých modulů je řízena pomocí feature toggles na úrovni projektu. Aktivní funkce jsou určeny konfigurací v databázi a následně ovlivňují jak aplikační logiku, tak uživatelské rozhraní.

Vedle databázového modelu systém využívá i souborový přístup pro vybrané obsahové části. Statický nebo méně strukturovaný obsah je uložen ve formátu JSON, zatímco transakční data zůstávají v relační databázi. Hybridní přístup snižuje složitost a velikost databáze.

3.4.2 Možnosti rozšíření systému

Navržená architektura umožňuje rozšiřování dvěma základními způsoby: přidáním nového modulu nebo rozšířením stávající funkcionality.

Při přidání nového modulu dochází k doplnění datového modelu, aplikační logiky a administračního rozhraní. Modul je zároveň integrován do systému řízení funkcí, což umožňuje jeho aktivaci pouze pro vybrané projekty.

Rozšíření stávajících modulů probíhá úpravou existujících entit a logiky bez nutnosti zásahu do ostatních částí systému.

Struktura aplikace je navržena tak, aby bylo možné systém nasadit na standardní PHP hosting bez specifických požadavků. To snižuje provozní náklady a zároveň umožňuje škálování prostřednictvím více projektů v rámci jedné instance.

3.5 Návrh řízení přístupů

Řízení přístupů v systému GastroHub je navrženo jako víceúrovňové a odděluje administrátorskou část a zákaznickou část aplikace.

Oddělení je realizováno nejen logicky, ale i technicky, protože administrátorské a zákaznické účty využívají samostatnou autentizaci, oddělené session a vlastní modely. Tím je minimalizováno riziko eskalace oprávnění mezi jednotlivými částmi systému.

Architektura řízení přístupů je založena na kombinaci autentizace pomocí session, role-based access control (RBAC), systému detailních oprávnění a ochranných bezpečnostních mechanismů.

3.5.1 Role a oprávnění

Administrátorská část

Administrace využívá model RBAC, kde jsou oprávnění přiřazována rolím a ty následně jednotlivým uživatelům.

Speciální roli představuje systémový vlastník, který má plný přístup bez omezení. Mechanismus usnadňuje správu systému, ale zároveň představuje bezpečnostní riziko v případě kompromitace účtu.

Hierarchická oprávnění

Oprávnění jsou definována pomocí strukturovaných klíčů, které umožňují hierarchické uspořádání. Přístup k nadřazené úrovni automaticky zahrnuje i přístup k podřízeným částem.

Použitý přístup snižuje počet nutných vazeb a zjednodušuje rozšiřování systému o nové funkcionality.

Automatická správa oprávnění

Oprávnění jsou generována automaticky na základě struktury administrace. Tím je zajištěna konzistence mezi uživatelským rozhraním a bezpečnostní logikou a zároveň se snižuje riziko chyb při vývoji nových částí systému.

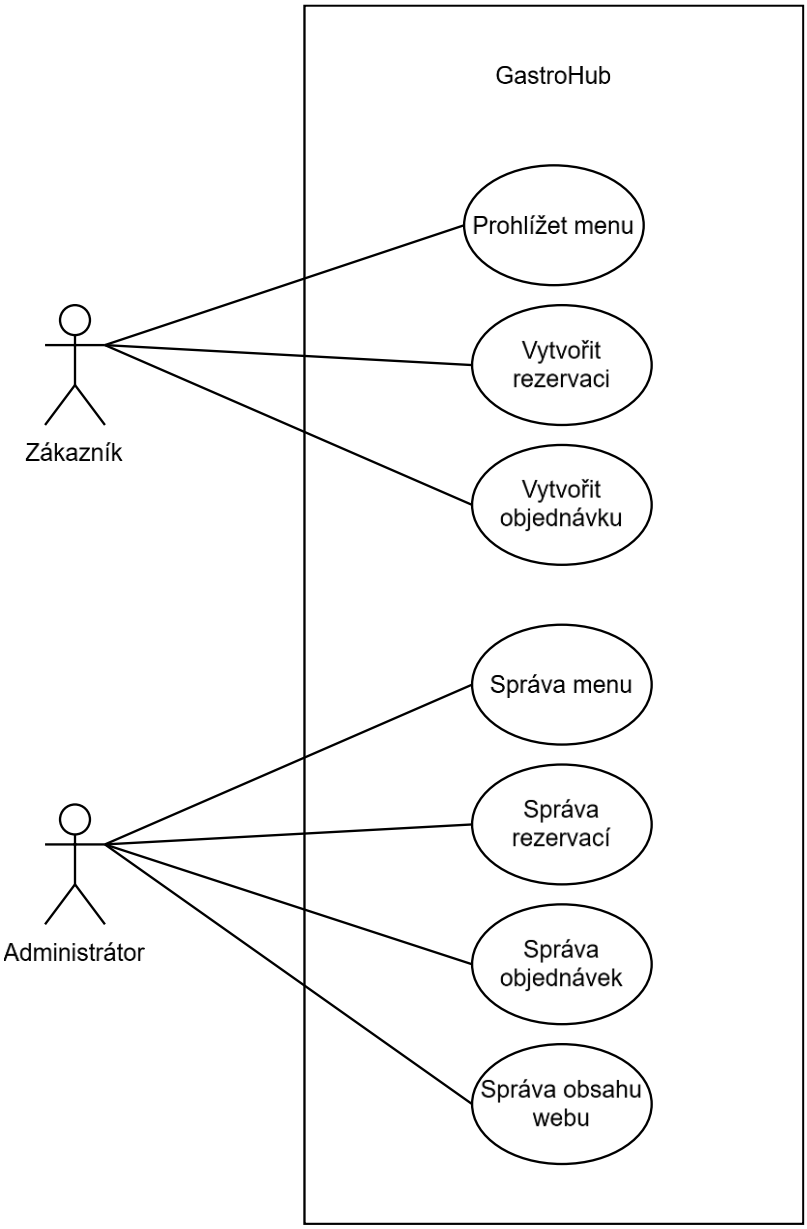
Autorizace přístupu

Přístup ke chráněným částem systému je ověřován na serverové straně. Každý požadavek prochází kontrolou přihlášení a ověřením odpovídajících oprávnění.

Zákaznická část

Zákaznické účty jsou zcela oddělené od administrace a slouží pouze pro přístup k uživatelským datům, jako jsou objednávky, rezervace nebo profil.

Oddělení těchto dvou domén výrazně snižuje riziko neoprávněného přístupu a zvyšuje celkovou bezpečnost systému.



Obr. 6: Zjednodušený use case diagram systému GastroHub

Zdroj: vlastní zpracování

Diagram znázorňuje základní aktéry systému, přičemž správa oprávnění je řešena pomocí RBAC modelu umožňujícího definici libovolného počtu rolí.

3.5.2 Bezpečnostní principy návrhu

Návrh respektuje základní bezpečnostní principy:

- Oddělení odpovědností – samostatná autentizace pro administrátory a zákazníky.
- Princip nejmenších oprávnění – uživatel má pouze nezbytný rozsah přístupů.
- Ochrana proti CSRF – validace požadavků pomocí tokenů.
- Bezpečná správa session – změna identifikátoru relace při přihlášení.
- Server-side autorizace – kontrola přístupu probíhá výhradně na backendu.

3.5.3 Shrnutí návrhu

Řízení přístupů v systému GastroHub je založeno na RBAC modelu rozšířeném o hierarchická oprávnění a doplněném o základní bezpečnostní mechanismy.

Návrh představuje kompromis mezi bezpečností, jednoduchostí správy a možností dalšího rozšiřování systému. Architektura umožňuje přidávání nových modulů bez narušení existující logiky a zachovává konzistentní řízení přístupů napříč projekty.

Další rozšíření může zahrnovat například vícefaktorovou autentizaci, auditní logování nebo pokročilejší omezení přístupu.

3.6 Návrh integrace externích služeb

Integrace externích služeb v systému GastroHub je navržena jako volitelná a konfigurovatelná. Systém je funkční i bez aktivních integrací a jednotlivé služby lze zapínat nebo vypínat podle potřeb konkrétního projektu.

Konfigurace je řešena pomocí proměnných prostředí (.env), což umožňuje oddělení citlivých údajů od aplikačního kódu a usnadňuje nasazení v různých prostředích.

3.6.1 Platební brána

Zvolený přístup

Pro zpracování plateb byla zvolena integrace prostřednictvím služby Stripe, konkrétně využití Stripe Checkout.

Hlavním důvodem volby je minimalizace bezpečnostních rizik a implementační složitosti. Platební údaje nejsou zpracovávány v rámci aplikace, ale přímo na straně poskytovatele služby, což výrazně snižuje požadavky na zabezpečení systému.

Z pohledu architektury jde o vhodný kompromis mezi rychlostí implementace a bezpečností. Alternativní přístup založený na přímé integraci plateb by poskytoval větší flexibilitu, ale za cenu výrazně vyšší složitosti.

Tok platby

Proces platby je navržen jako přesměrování uživatele na externí platební stránku. Aplikace nejprve vytvoří platební relaci a následně uživatele přesměruje na platební rozhraní služby Stripe.

Po dokončení platby je uživatel vrácen zpět do aplikace, kde probíhá ověření výsledku transakce na serverové straně. Rezervace je označena jako zaplacená pouze v případě, že externí služba potvrdí úspěšné dokončení platby.

Postup eliminuje riziko falešného potvrzení platby na základě manipulace s klientskou částí aplikace.

Degradace funkčnosti

Integrace je navržena jako volitelná. V případě absence konfiguračních údajů systém umožňuje pokračovat v režimu bez online plateb.

Zhodnocení návrhu

Zvolený přístup využívající Stripe Checkout výrazně snižuje bezpečnostní rizika, protože aplikace nepřichází do kontaktu s platebními údaji. Zároveň umožňuje rychlou implementaci bez nutnosti řešit komplexní platební logiku na straně systému.

Nevýhodou tohoto řešení je závislost na přesměrování uživatele zpět do aplikace po dokončení platby. V případě, že se uživatel nevrátí, může dojít k dočasné nekonzistenci stavu rezervace.

Pro produkční nasazení by proto bylo vhodné doplnit webhook mechanismus, který umožní potvrzení platby nezávisle na chování uživatele.

3.7 Diskuse alternativ řešení

Návrh architektury systému GastroHub vychází z kombinace MVC přístupu, vícevrstvého členění a více-tenantního modelu se sdílenou databází. Následující část uvádí hlavní alternativní přístupy a zdůvodňuje volbu zvoleného řešení s ohledem na rozsah systému a způsob jeho nasazení.

3.7.1 Alternativní architektonické přístupy

Použití PHP frameworku

Alternativou k vlastní implementaci bylo využití frameworku, například Laravel nebo Symfony. Tyto nástroje poskytují standardizovanou architekturu, hotové komponenty a rozsáhlý ekosystém, což umožňuje rychlejší vývoj a usnadňuje údržbu aplikace.

Nevýhodou je vyšší míra abstrakce a režie, která nemusí být opodstatněná u menších systémů. Vzhledem k rozsahu navrhované aplikace a cílovému nasazení nebylo využití robustního frameworku nezbytné a vedlo by ke zvýšení komplexity bez výrazného přínosu.

Zvolený přístup umožňuje jednodušší strukturu aplikace a přímou kontrolu nad jednotlivými vrstvami systému.

Mikroservisní architektura

Další alternativou bylo rozdělení systému do samostatných služeb. Přístup umožňuje nezávislé škálování a nasazování jednotlivých částí systému a je vhodný pro rozsáhlé aplikace s vysokou zátěží.

Nevýhodou je výrazné zvýšení složitosti, zejména nutnost řešit komunikaci mezi službami, správu infrastruktury a konzistenci dat. Pro systém daného rozsahu by takové řešení představovalo nepřiměřenou komplexitu bez odpovídajícího přínosu.

3.7.2 Omezení navrženého řešení

Navržená architektura přináší několik omezení, která je nutné zohlednit při dalším rozvoji systému.

Modulární monolit

Systém je implementován jako modulární monolit. To znamená, že při růstu zátěže je nutné škálovat aplikaci jako celek, nikoli jednotlivé části samostatně.

Absence asynchronního zpracování

Systém aktuálně nevyužívá asynchronní zpracování ani fronty úloh. Operace jako potvrzení plateb nebo odesílání notifikací probíhají synchronně, což může být limitující při vyšší zátěži.

Bezpečnostní omezení

Základní bezpečnostní mechanismy jsou implementovány, avšak některé pokročilé prvky chybí, například vícefaktorová autentizace, auditní logování nebo ochrana proti útokům typu brute-force.

Škálovatelnost databázové vrstvy

Sdílený databázový model vyžaduje důslednou kontrolu přístupu k datům. Chyba v implementaci může vést k narušení izolace mezi projekty.

Pro dlouhodobý rozvoj by bylo vhodné doplnit automatizované kontroly tenant kontextu nebo další ochranné mechanismy.

4 Implementace navrženého řešení

Implementační kapitola popisuje praktickou realizaci navržené architektury systému GastroHub. Zaměřuje se na použité technologie, strukturu aplikace a způsob, jakým byly do výsledného řešení promítnuty principy uvedené v kapitole 3, zejména architektura MVC, více-tenantní provoz, modularita a řízení přístupů. Cílem kapitoly je doložit, že navržené řešení bylo převedeno do funkční webové aplikace použitelné pro reálný provoz menších gastronomických podniků.

4.1 Použité technologie

Implementace systému GastroHub je realizována jako webová aplikace bez využití plnohodnotného PHP frameworku. Zvolený přístup umožňuje zachovat jednoduchou strukturu projektu, přímou kontrolu nad aplikační logikou a snadné nasazení na běžném PHP hostingu. Použité technologie pokrývají serverovou logiku, ukládání relačních dat, generování PDF výstupů, odesílání e-mailů i tvorbu uživatelského rozhraní veřejné a administrační části systému.

4.1.1 Serverová část

Serverová část je implementována v jazyce PHP 8. Projekt využívá Composer pro automatické načítání tříd podle standardu PSR-4, což zjednodušuje organizaci kódu a eliminuje nutnost ručního includování souborů.

Pro doplňkové funkce jsou využity externí knihovny spravované přes Composer, konkrétně knihovna dompdf pro generování PDF dokumentů a knihovna phpmailer pro odesílání e-mailů prostřednictvím SMTP.

Konfigurace prostředí je řešena pomocí .env souborů načítaných v aplikační konfiguraci, což umožňuje oddělit citlivé údaje od zdrojového kódu a usnadňuje nasazení v různých prostředích.

Součástí serverové části je také integrace platební brány Stripe Checkout, která je využita pro realizaci online plateb v rámci hotelového modulu.

4.1.2 Databáze

Datová vrstva je realizována pomocí relační databáze MariaDB. Přístup k databázi je implementován prostřednictvím rozhraní PDO, které zajišťuje jednotnou práci s databází a podporu připravených dotazů.

Konfigurace databázového připojení je nastavena s důrazem na bezpečnost a konzistentní chování aplikace, zejména využitím výjimek při chybách a preferováním prepared statements, což přispívá k ochraně proti SQL injection.

Databáze je navržena pro více-tenantní provoz a modulární rozšiřování systému. Jednotlivé části aplikace využívají samostatné tabulky a projektově závislá data jsou oddělena pomocí identifikátoru projektu.

Z hlediska implementace je databázová vrstva udržována záměrně jednoduchá. Místo ORM je použit přímý přístup přes modely a SQL dotazy, což umožňuje lepší kontrolu nad databázovou logikou a odpovídá cíli práce zachovat transparentnost návrhu.

4.1.3 Frontendové technologie

Frontendová vrstva je založena na kombinaci serverově renderovaných šablon a utility-first CSS frameworku Tailwind CSS. Projekt používá Tailwind ve verzi 4.1.11 a kompilace probíhá přes Node toolchain (skripty v package.json):

- dev: build CSS v režimu watch (pro vývoj),
- build: minifikovaný výstup pro produkci.

Vstupní Tailwind soubor je veden v public/assets/css/tailwind.css a výsledný CSS bundle je generován do public/assets/css/main.css, který je následně používán ve veřejné části i v administraci. Použití Tailwind CSS udržuje UI konzistentní napříč moduly a současně umožňuje rychlý návrh rozhraní bez nutnosti psát rozsáhlé vlastní CSS.

JavaScriptová část je řešena minimalisticky jako několik menších skriptů ve public/assets/script/ (např. logika modálních oken, chování hlavičky, cookies banner). Záměrem je ponechat UI co nejvíce serverově řízené a JS používat pouze tam, kde vylepšuje použitelnost rozhraní.

4.2 Struktura aplikace

Aplikace GastroHub je rozdělena do několika hlavních adresářů podle jejich role:

- core/ – infrastrukturní jádro (router, základní třídy, konfigurace),
- app/ – aplikační logika veřejné části,
- admin/ – administrační rozhraní,
- public/ – vstupní bod aplikace,
- vendor/ – externí knihovny spravované pomocí Composeru.

Adresář public/ obsahuje vstupní skripty index.php pro veřejnou část a admin.php pro administraci. Veřejné požadavky jsou zpracovány přes centrální router, zatímco administrace využívá samostatný vstupní bod.

Adresář app/ obsahuje controllery, modely a view šablony veřejné části. GET požadavky jsou obsluhovány controllery, zatímco zpracování POST požadavků je odděleno do samostatné aplikační vrstvy, která se zaměřuje především na validaci dat a změnu stavu aplikace.

Adresář admin/ obsahuje samostatnou strukturu pro správu systému, včetně stránek a skriptů.

4.3 Implementace databázového modelu

Databázová vrstva systému GastroHub je realizována relační databází MariaDB. Návrh vychází z principů popsaných v kapitole 3.3 a je postaven na centrální entitě projects, která reprezentuje jednotlivé tenanty. Všechna projektově závislá data jsou na tuto entitu navázána pomocí identifikátoru project_id, což umožňuje více-tenantní provoz v rámci jedné databáze.

Z hlediska implementace jsou u většiny tabulek použity primární klíče typu INT s automatickým inkrementem a časová razítka (created_at, updated_at) pro sledování změn. Stavové informace jsou reprezentovány buď pomocí předdefinovaných hodnot (např. status rezervace), nebo textových atributů v případech, kde je vyžadována větší flexibilita.

Důležitým principem je ukládání „snapshotů“ u transakčních dat (např. položek objednávek), které zajišťuje konzistenci historických záznamů i při změně souvisejících dat v systému.

Vazby mezi tabulkami jsou realizovány pomocí cizích klíčů a aplikační logiky. U klíčových vztahů je využito pravidlo CASCADE, zatímco v případech, kde je vhodné zachovat data i po odstranění související entity, je použito SET NULL.

4.4 Realizovaná aplikace a její funkce

Výsledkem implementace je webová aplikace umožňující tvorbu a správu webových prezentací gastronomických podniků včetně jejich provozních funkcí, jako jsou menu, rezervace nebo online objednávky. Pro účely demonstrace byla vytvořena ukázková instance v podobě fiktivní italské restaurace Bella Vista, na níž jsou jednotlivé části systému prezentovány v realistickém kontextu.

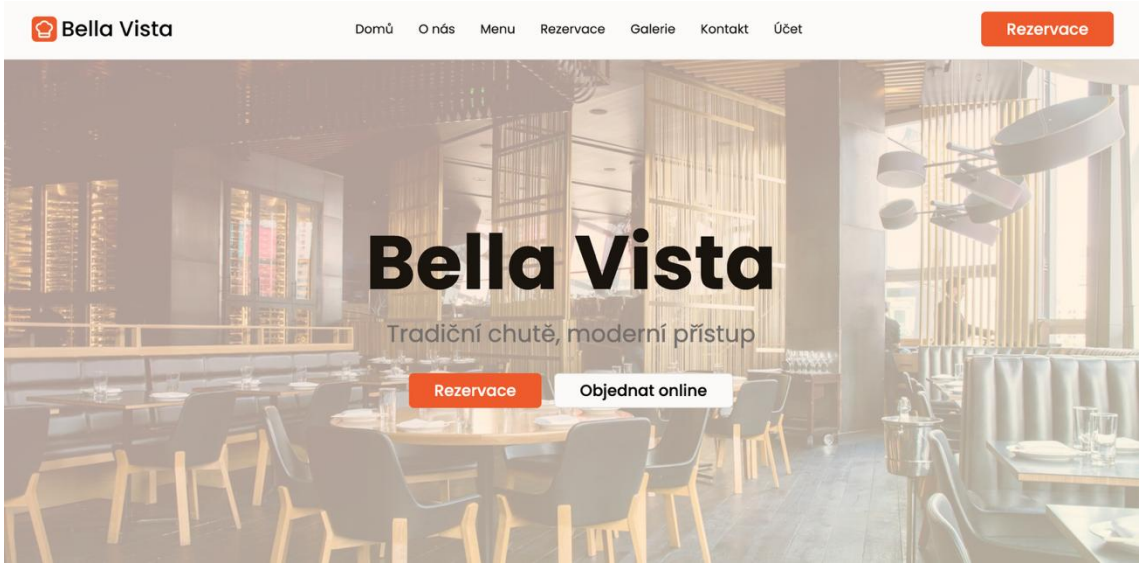
Aplikace se skládá ze dvou hlavních částí:

- veřejné části (front-office), určené zákazníkům podniku,
- administrační části (back-office), určené provozovateli.

4.4.1 Veřejná část webu

Veřejná část aplikace reprezentuje kompletní webovou prezentaci restaurace. Struktura stránky je navržena jako jednostránková prezentace kombinující marketingový obsah s interaktivními prvky podporujícími přímou akci uživatele.

Úvodní sekce obsahuje název restaurace, stručný popis a hlavní výzvy k akci ve formě tlačítek pro rezervaci a online objednávku. Navigace umožňuje rychlý přesun na jednotlivé části stránky.



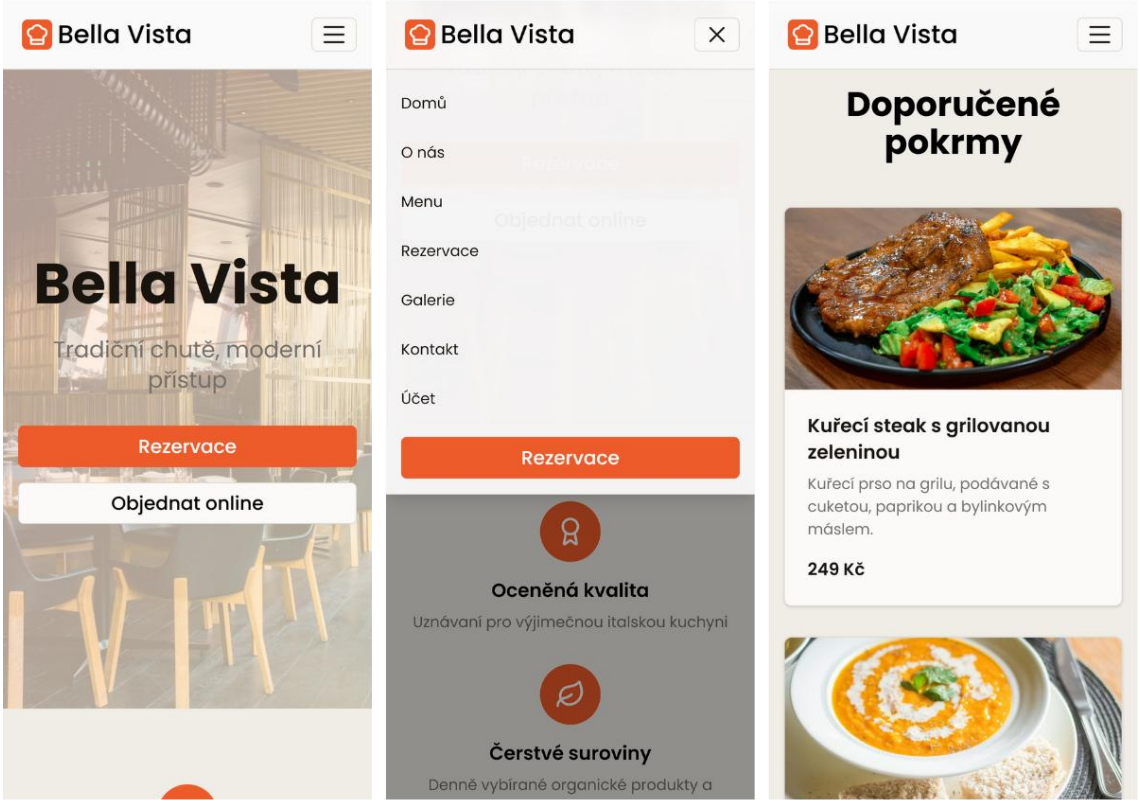
Obr. 7: Úvodní sekce veřejné části webu

Zdroj: vlastní zpracování

Bezprostředně následuje blok prezentující hlavní benefity podniku, který slouží k rychlému sdělení hodnoty pro návštěvníka.

Sekce „O nás“ kombinuje textový popis restaurace s fotografií, která pomáhá návštěvníkovi lépe si představit prostředí podniku.

Sekce doporučených pokrmů zobrazuje výběr tří jídel včetně obrázku, popisu a ceny. Cílem je rychle prezentovat klíčovou nabídku bez nutnosti procházet celé menu.



Obr. 8: Mobilní zobrazení aplikace – úvodní sekce, navigace a doporučené pokrmy
Zdroj: vlastní zpracování

Kompletní menu je generováno dynamicky na základě dat vytvořených v administraci. Položky jsou rozděleny do kategorií a obsahují název, popis a cenu. Součástí sekce je také možnost zobrazení menu ve formátu PDF. PDF může být buď nahráno přímo v administraci, nebo je automaticky generováno ze strukturovaných dat v případě, že vlastní soubor není dostupný nebo není aktivní.

Součástí stránky je blok podporující online objednávky, který obsahuje stručný popis služby, výhody a odkaz na objednávkový proces.

Rezervační formulář umožňuje zadání termínu, počtu osob a kontaktních údajů. V případě, že je podnik označen jako uzavřený, není možné rezervaci provést, což zajišťuje konzistentní chování systému vůči reálnému provozu. Po odeslání rezervace dochází k uložení dat, odeslání potvrzovacího e-mailu zákazníkovi a vytvoření notifikace v administraci.

Součástí prezentace je také sekce ubytování, která je napojena na hotelový modul. Zobrazené pokoje jsou načítány z databáze a každý obsahuje detail s možností rezervace. Po odeslání rezervace je uživatel přesměrován na platební bránu Stripe, která je v rámci demonstrace provozována v testovacím režimu. Po úspěšném dokončení platby je rezervace potvrzena a systém odešle potvrzovací e-mail zákazníkovi i notifikaci do administrace.

Další obsahové sekce zahrnují nadcházející události, novinky, galerii a reference zákazníků. Prvky rozšiřují prezentaci o doplňující informace a vizuální obsah, který pomáhá návštěvníkovi při rozhodování.

Stránka je zakončena kontaktní sekcí obsahující adresu, otevírací dobu, mapu a kontaktní údaje. Patička obsahuje doplňující navigaci a základní informace o provozovateli.

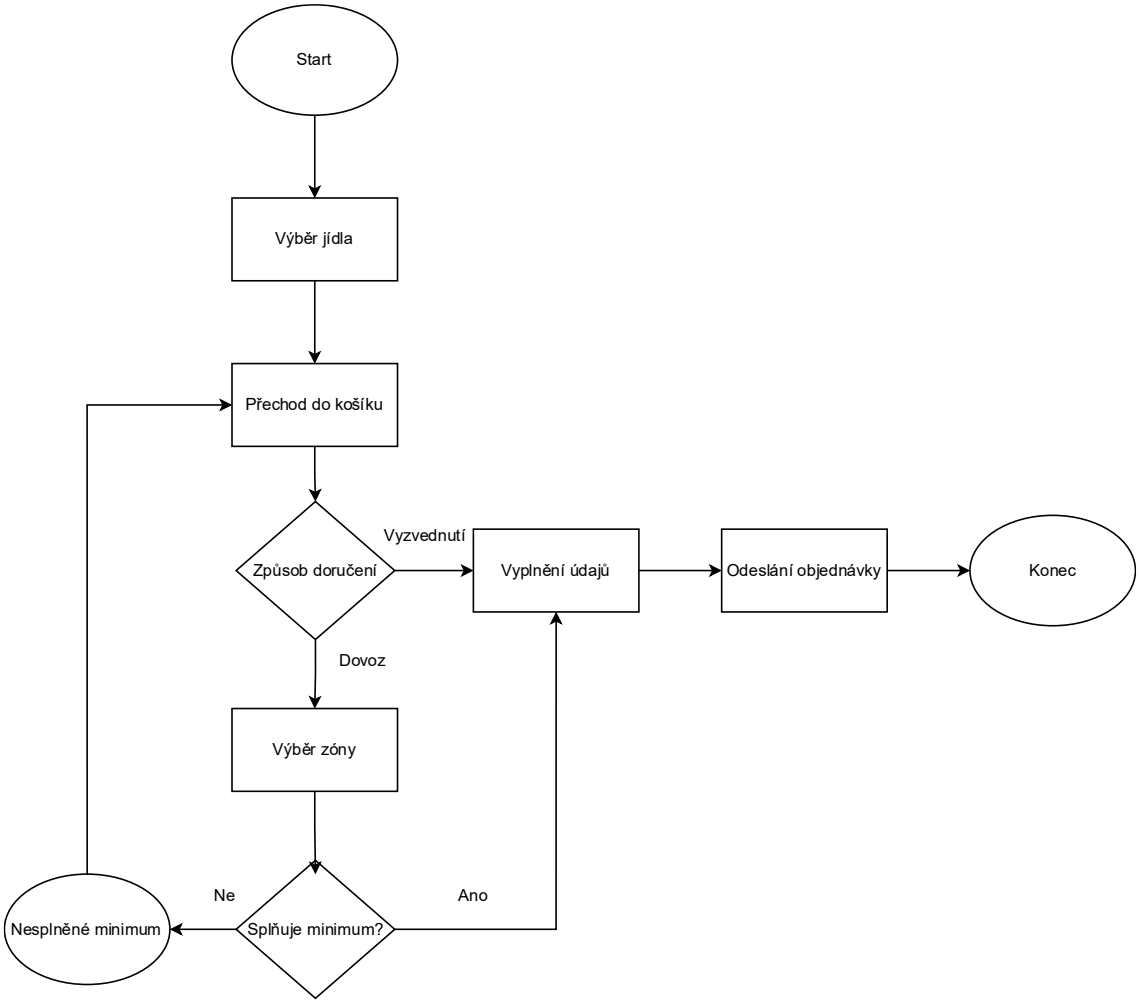
4.4.2 Scénář online objednávky

Součástí aplikace je samostatný objednávkový proces, který je oddělen od hlavní stránky a představuje vícekrokovou interakci uživatele se systémem.

Proces objednávky probíhá v několika navazujících krocích:

1. Uživatel vybírá položky ze seznamu dostupných jídel a jejich přidáním průběžně vytváří obsah košíku.
2. V košíku kontroluje souhrn objednávky a může položky upravit nebo odstranit.
3. Následně volí způsob odběru, tedy osobní vyzvednutí nebo dovoz.
4. Při doručení zadává adresu a kontaktní údaje. Systém současně validuje minimální hodnotu objednávky a cenu dopravy podle zvolené zóny.
5. Po odeslání dochází k validaci vstupních dat, uložení objednávky do databáze, odeslání potvrzovacího e-mailu zákazníkovi a zobrazení objednávky v administraci.

Objednávkový proces je navržen jako lineární tok bez zbytečných přechodů mezi stránkami, což snižuje složitost interakce a zrychluje dokončení objednávky.



Obr. 9: Diagram znázorňující průběh objednávkového procesu

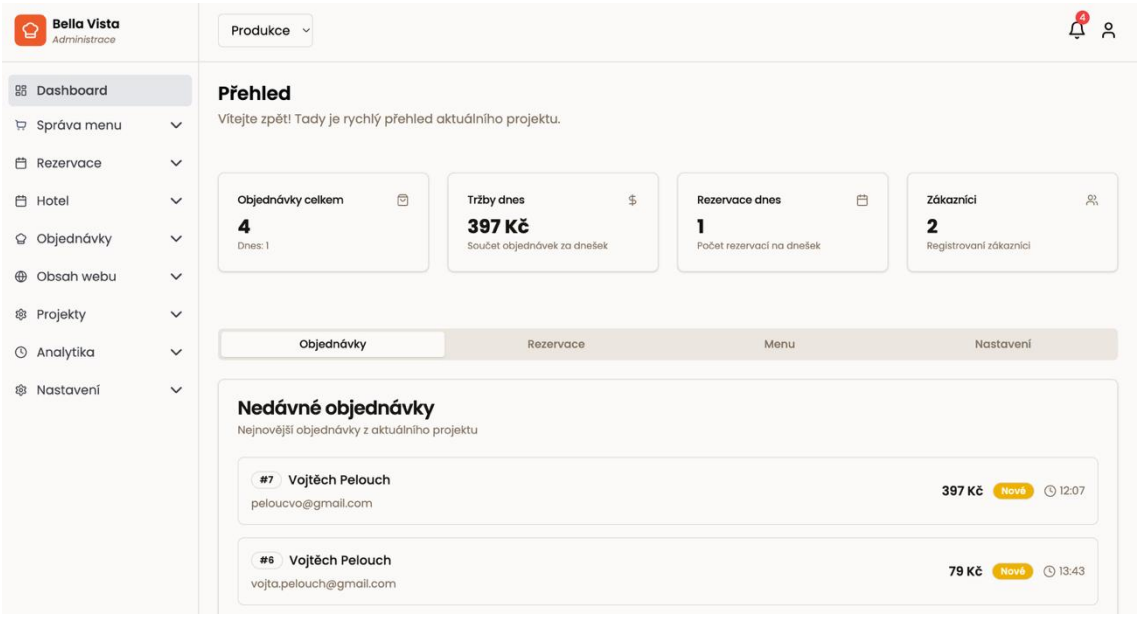
Zdroj: vlastní zpracování

4.4.3 Administrace systému

Administrace systému slouží k řízení obsahu webové prezentace i provozních funkcí podniku. Je oddělena od veřejné části aplikace a přístupná pouze autorizovaným uživatelům. Rozhraní je navrženo s důrazem na přehlednost a seskupuje jednotlivé funkce do logických sekcí v levém navigačním menu. Součástí administrace je také uživatelský návod dostupný pod ikonou uživatele. Návod obsahuje popis jednotlivých funkcí systému a slouží k usnadnění orientace v administraci, zejména pro nové uživatele.

Dashboard

Úvodní stránku administrace tvoří dashboard, který poskytuje základní přehled o stavu projektu. Slouží jako výchozí rozcestník a zobrazuje souhrn důležitých informací, například nové objednávky, rezervace nebo další provozní údaje, které vyžadují pozornost obsluhy.



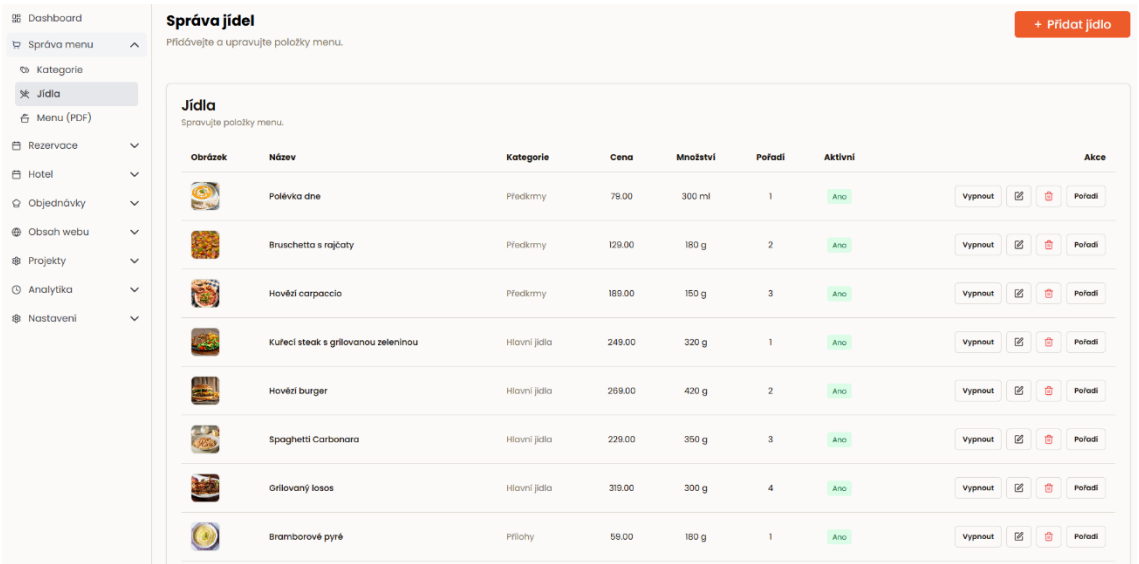
Obr. 10: Celkový přehled administrace

Zdroj: vlastní zpracování

Správa menu

Sekce Správa menu slouží k tvorbě a úpravě jídelního lístku. Obsahuje správu kategorií, jednotlivých jídel a také práci s PDF verzí menu. Uživatel může vytvářet strukturu jídelního lístku, upravovat položky a zajistit jejich následné zobrazení ve veřejné části webu. Součástí je i možnost nahrání vlastního PDF souboru, případně využití automaticky generovaného menu.

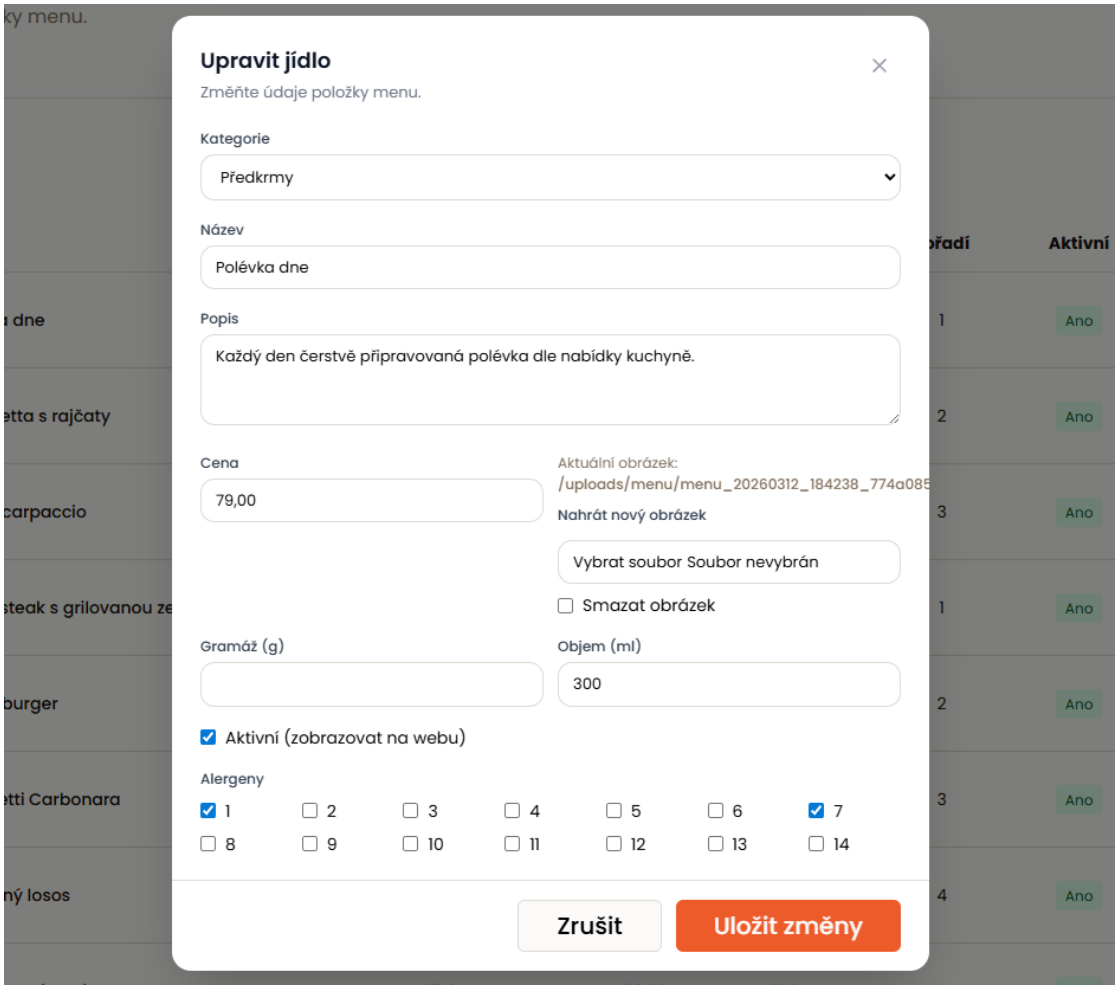
Přehled jídel je zobrazen v tabulkové formě, která umožňuje rychlou orientaci v položkách a jejich základních parametrech.



Obr. 11: Přehled položek menu v administraci systému

Zdroj: vlastní zpracování

Jednotlivé položky lze dále upravovat prostřednictvím modálního formuláře.



Obr. 12: Formulář pro úpravu položky menu v administraci systému
Zdroj: vlastní zpracování

Rezervace

Sekce Rezervace umožňuje správu rezervací stolů a současně nastavení stolů v provozu. Přehled rezervací slouží ke kontrole přijatých požadavků zákazníků a jejich dalšímu zpracování, zatímco nastavení stolů představuje konfigurační část rezervačního modulu.

Hotel

Sekce Hotel rozšiřuje aplikaci o podporu ubytovacích služeb. Umožňuje správu pokojů, evidenci hostů, přehled rezervací pokojů a sledování obsazenosti prostřednictvím kalendářového zobrazení. Díky tomu může být systém použit nejen pro restaurace, ale i pro podniky kombinující gastronomii a ubytování.

Objednávky

Sekce Objednávky slouží ke správě online objednávek a souvisejících údajů. Zahrnuje přehled objednávek, evidenci zákazníků a správu dovozových zón. Právě dovozové zóny představují důležitou část objednávkového procesu, protože umožňují definovat cenu dopravy a minimální hodnotu objednávky pro jednotlivé oblasti rozvozu.

Obsah webu

Sekce Obsah webu představuje obsahovou část administrace. Umožňuje správu stránek, novinek, událostí a médií. Prostřednictvím této sekce lze upravovat textový i obrazový obsah veřejné části webu a průběžně aktualizovat prezentaci podniku bez nutnosti zásahu do zdrojového kódu.

Projekty

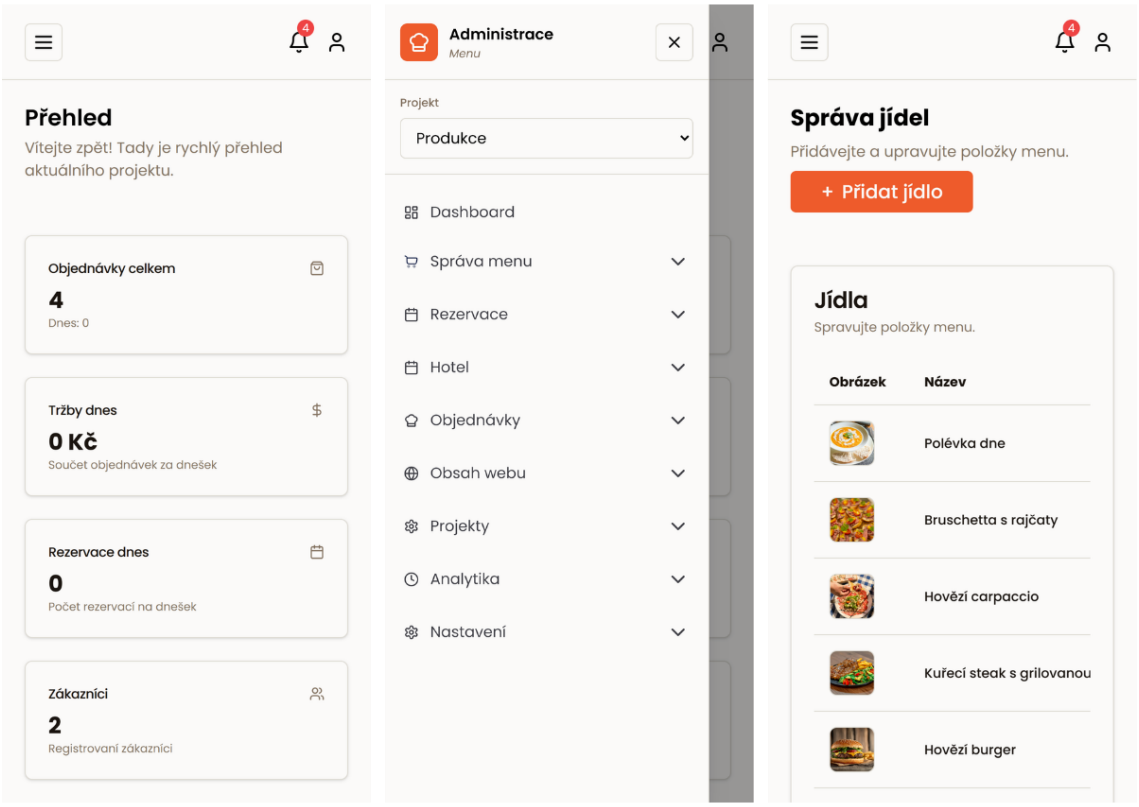
Sekce Projekty slouží ke správě více projektů v rámci jedné instalace systému. Uživatel zde může spravovat jednotlivé projekty a nastavovat dostupné funkce pro každý z nich samostatně. Část podporuje více projektový provoz a umožňuje přizpůsobit aplikaci různým typům gastronomických podniků.

Analytika

Sekce Analytika poskytuje přehled o provozu webu a základních statistických údajích. Umožňuje nastavení integrace Google Analytics a současně aktivaci cookies lišty, která zajišťuje informování návštěvníků webu o používání analytických nástrojů. Analytická část tak rozšiřuje správu systému o sledování návštěvnosti a chování uživatelů.

Nastavení

Sekce Nastavení soustřeďuje konfigurační části systému. Obsahuje všeobecné nastavení projektu, otevírací dobu, blokace, správu právních podmínek a uživatelské přístupy. Zvláštní význam má správa uživatelů a oprávnění, která umožňuje řídit dostupnost jednotlivých částí administrace podle role konkrétního uživatele.



Obr. 13: Mobilní zobrazení administrace – dashboard, navigace a správa jídel

Zdroj: vlastní zpracování

4.5 Implementace řízení přístupů

Implementace řízení přístupů v systému GastroHub přímo navazuje na návrh popsaný v kapitole 3.5. V praxi je bezpečnostní model realizován kombinací autentizace založené na serverové relaci (session), role-based access control (RBAC), hierarchických oprávnění a důsledné serverové autorizace jednotlivých operací. Systém současně odděluje administrátorskou část (back-office) od zákaznické části (front-office), a to jak logicky, tak technicky.

Toto oddělení představuje klíčový bezpečnostní prvek, protože minimalizuje riziko eskalace oprávnění mezi jednotlivými doménami systému. Administrátorské účty a zákaznické účty využívají rozdílné session proměnné, rozdílné autentizační mechanismy i oddělené modely práce s daty.

4.5.1 Autentizace uživatelů

Autentizace administrátorských uživatelů je realizována prostřednictvím databázové tabulky administrátorů a serverové relace. Po ověření přihlašovacích údajů jsou identifikační údaje uživatele uloženy do session, která reprezentuje jeho přihlášený stav. Při úspěšném přihlášení dochází k regeneraci identifikátoru relace, čímž je eliminováno riziko útoku typu session fixation.

Po přihlášení jsou z databáze načtena oprávnění příslušející roli daného uživatele a jsou uložena do relace. Tím je zajištěno, že autorizace dalších požadavků probíhá bez nutnosti opakovaných databázových dotazů na každé stránce.

Každá chráněná stránka administrace nejprve ověřuje, zda je uživatel autentizován. Pokud není, je přesměrován na přihlašovací stránku. Mechanismus zajišťuje, že administrace není přístupná nepřihlášeným uživatelům.

Zákaznická část systému využívá samostatný autentizační mechanismus. Zákaznické účty nejsou propojeny s administrátorskými účty a používají oddělenou session proměnnou. Toto řešení zabraňuje tomu, aby kompromitace zákaznického účtu umožnila přístup do administrace. Oddělení těchto dvou domén odpovídá principu oddělení odpovědností (Separation of Concerns).

4.5.2 Autorizace operací

Autorizace operací v administraci navazuje na implementovaný RBAC model a určuje způsob vyhodnocení oprávnění při zpracování požadavků. Každý administrátor má přiřazenou roli, která definuje množinu dostupných oprávnění.

Oprávnění jsou reprezentována textovými klíči (např. „orders“, „settings.mail“, „news.edit“), což umožňuje jemnozrnnou kontrolu přístupu k jednotlivým částem administrace. Systém podporuje hierarchickou strukturu oprávnění založenou na tečkové notaci. Pokud má uživatel oprávnění k nadřazenému klíči, například „settings“, automaticky získává přístup i ke všem podřízeným sekcím, například „settings.mail“ nebo „settings.legal“. Princip usnadňuje rozšiřování systému o nové moduly bez nutnosti zásahu do existujících rolí.

Každá chráněná operace je autorizována na serverové straně. Nestačí tedy skrýt položku v navigaci administrace. Přístup je vždy ověřen při zpracování požadavku. V případě nedostatečných oprávnění systém vrací chybový stav HTTP 403. Tím je zabráněno obcházení autorizace manipulací s klientskou částí aplikace.

Speciálním případem je uživatel označený jako systémový vlastník, který má přístup ke všem částem administrace bez ohledu na přiřazená oprávnění. Přístup zjednodušuje správu systému, zároveň však představuje bezpečnostní riziko. Kompromitace takového účtu by znamenala plnou kontrolu nad aplikací. V produkčním prostředí je proto vhodné účet chránit dodatečnými mechanismy, například dvoufaktorovou autentizací.

4.5.3 Ochrana proti CSRF a bezpečnost relací

Systém implementuje ochranu proti útokům typu Cross-Site Request Forgery (CSRF). Každý formulář obsahuje jedinečný token uložený v session, který je při odeslání požadavku validován. Token je generován kryptograficky bezpečným způsobem a porovnání probíhá funkcí odolnou vůči timing útokům.

Důsledná práce se session (včetně regenerace identifikátoru relace po přihlášení) dále snižuje riziko zneužití relace útočníkem.

4.5.4 Další možnosti ochrany

Další možné rozšíření by mohlo zahrnovat implementaci dvoufaktorové autentizace administrátorů, auditní log změn oprávnění nebo ochranu proti opakovaným neúspěšným pokusům o přihlášení. V aktuálním rozsahu však implementace odpovídá požadavkům definovaným v analytické i návrhové části práce.

4.6 Souhrn implementace a její omezení

Implementace systému GastroHub naplňuje architektonické principy definované v kapitole 3 a demonstruje jejich praktickou realizaci bez využití plnohodnotného frameworku. Výsledné řešení lze charakterizovat jako modulární monolit s více-tenantní podporou, explicitně implementovaným MVC vzorem a role-based řízením přístupů.

Zvolený přístup přináší několik zásadních výhod. Především umožňuje plnou kontrolu nad tokem aplikace, transparentní práci s databází a vysokou míru přenositelnosti mezi běžnými PHP hostingy. Absence závislosti na konkrétním frameworku snižuje technologickou komplexitu a eliminuje vendor lock-in. Modulární struktura současně umožňuje postupné rozšiřování funkcionality bez zásadních zásahů do jádra aplikace.

Na druhé straně je nutné reflektovat i omezení navrženého řešení. Vlastní implementace MVC jádra znamená vyšší odpovědnost za bezpečnost, správu chyb a dlouhodobou udržitelnost kódu. Oproti zavedeným frameworkům zde chybí vestavěné mechanismy jako automatická ochrana proti CSRF ve všech vrstvách, middleware architektura, centralizované logování.

Podobně více-tenantní řešení založené na sdílené databázi s identifikátorem projektu představuje kompromis mezi jednoduchostí a izolací dat. Pro malé a střední projekty je model

dostatečný, avšak při výrazném škálování by bylo vhodné zvážit oddělené databázové instance nebo schémata.

Platební modul využívající Stripe Checkout minimalizuje bezpečnostní zátěž aplikace, nicméně aktuální implementace je závislá na návratu uživatele z platební brány. Pro produkční prostředí by bylo vhodné doplnit asynchronní webhook mechanismus, který by finální potvrzení platby prováděl nezávisle na chování klienta.

Celkově lze implementaci hodnotit jako funkční a architektonicky konzistentní řešení odpovídající rozsahu bakalářského projektu. Systém pokrývá hlavní provozní i obsahové scénáře gastronomického podniku, je modulární, rozšiřitelný a přenositelný na běžné hostingové prostředí. Identifikovaná omezení nepředstavují zásadní nedostatky aktuální implementace, ale spíše oblasti, které by bylo vhodné dále rozvíjet při nasazení ve větším měřítku.

5 Testování a vyhodnocení

Cílem testování bylo ověřit, zda implementovaný systém GastroHub splňuje funkční a nefunkční požadavky stanovené v analytické a návrhové části práce. Testování bylo zaměřeno na správnou funkčnost jednotlivých modulů, práci s databází, řízení přístupů v administraci, oddělení dat v rámci více-tenantní architektury a základní výkonnostní parametry aplikace. Testování probíhalo průběžně během vývoje a následně i po nasazení aplikace na server.

Ověření bylo provedeno především manuálně formou simulace běžných scénářů používání systému. Testovány byly jak funkce dostupné ve veřejné části webu, tak operace prováděné v administraci. Součástí testování bylo také zaznamenání identifikovaných chyb a jejich následné odstranění.

5.1 Ověření funkčních požadavků

Cílem funkčního testování bylo ověřit, zda systém správně realizuje hlavní scénáře používání definované v analytické části práce. Ověření se soustředilo zejména na správu obsahu, práci s menu, vytváření rezervací, správu objednávek, administraci uživatelů a oddělení dat jednotlivých projektů.

Tab. 3: Přehled testovacích scénářů funkčního testování

Oblast	**Testovací scénář**	**Očekávaný výsledek**	**Skutečný výsledek**
Rezervace	odeslání formuláře	rezervace se uloží a zobrazí v administraci	splněno
Oprávnění	přístup bez role	HTTP 403 / zamítnutí	splněno
Multi-tenant	přepnutí projektu	zobrazí se jen data projektu	splněno
Platba	návrat ze Stripe success	ověření session a změna stavu	splněno

Zdroj: vlastní zpracování

5.1.1 Testování hlavních funkcí systému

V rámci testování byly ověřeny hlavní moduly aplikace. U správy obsahu byla kontrolována možnost vytvářet, upravovat a zobrazovat obsahové stránky a jejich následná prezentace ve veřejné části webu. U modulu menu bylo testováno zakládání kategorií a položek, jejich úprava a správné zobrazení návštěvníkům webu. Výsledky testování potvrdily, že změny provedené v administraci se správně ukládají a promítají do veřejné části aplikace.

Rezervační modul byl ověřen vytvořením rezervace prostřednictvím formuláře ve veřejné části webu a následnou kontrolou uložených dat v administraci. Po odeslání formuláře byla rezervace správně zapsána do databáze a zobrazena v přehledu rezervací, kde bylo možné měnit její stav a dále ji spravovat. Obdobným způsobem byla testována také správa objednávek. Bylo ověřeno vytvoření objednávky, její uložení a změna stavu v administraci.

Součástí testování byla i správa uživatelů a řízení přístupů. Bylo ověřeno, že uživatel bez odpovídajícího oprávnění nemá přístup k chráněným částem administrace. Dále byla testována práce s více projekty v rámci jedné instalace systému. Ověření potvrdilo, že data jednotlivých projektů zůstávají oddělena a v administraci nedochází k jejich vzájemnému promíchání.

Na základě provedených scénářů lze konstatovat, že hlavní moduly systému fungují ve vzájemné návaznosti správně a odpovídají požadované funkcionalitě.

5.1.2 Ověření správnosti datových operací

Vedle ověření uživatelských scénářů byla testována také správnost datových operací nad relační databází. Kontrola byla zaměřena na operace vytváření, načítání, úprav a mazání záznamů v administraci a na správné propsání těchto změn do uživatelského rozhraní.

Bylo ověřeno, že při založení nového záznamu dojde k jeho korektnímu uložení do databáze a následnému zobrazení v administraci. Při úpravách existujících záznamů se změny ihned projevily v rozhraní aplikace. Testováno bylo i odstranění záznamů, při němž došlo ke smazání dat a jejich současnému zmizení z příslušných přehledů.

Zvláštní pozornost byla věnována vazbám mezi entitami. Ověřováno bylo zejména správné přiřazení objednávek a rezervací ke konkrétním projektům a ukládání údajů zákazníků k příslušným záznamům. Testování potvrdilo, že datové operace probíhají konzistentně a že mezi jednotlivými částmi systému nedochází k narušení vazeb.

5.1.3 Identifikované chyby během testování

Důležitým výstupem testování nebylo pouze potvrzení funkčnosti, ale také odhalení konkrétních implementačních chyb. Jedním z nalezených problémů bylo nesprávné generování dynamického titulku stránky, který se v některých případech nezobrazoval podle aktuálního obsahu. Příčinou bylo nesprávné předávání dat do layoutové šablony. Chyba byla odstraněna úpravou logiky v kontroleru.

Další chyba se týkala načítání obsahu u uživatelsky vytvořených stránek. Původní implementace byla vázána na strukturu definovanou pouze v souboru home.json, takže nově vytvořené stránky neměly vlastní definici sekcí. Úprava spočívala v rozšíření mechanismu načítání konfigurace tak, aby systém pracoval i s dalšími stránkami.

Popsaná část testování ukázala, že manuální ověřování reálných scénářů je přínosné nejen pro kontrolu funkčnosti, ale i pro odhalení chyb, které nejsou na úrovni jednotlivých komponent na první pohled patrné.

5.2 Ověření nefunkčních požadavků

Vedle funkčních požadavků byly ověřovány také vybrané nefunkční vlastnosti systému, zejména bezpečnost a výkon aplikace. Vzhledem k charakteru navrženého řešení nebylo cílem provést zátěžové nebo penetrační testování v plném rozsahu, ale ověřit základní provozní vlastnosti systému při běžném použití.

5.2.1 Bezpečnostní aspekty

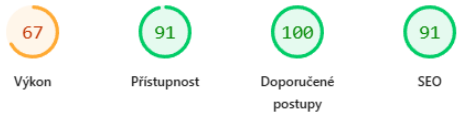
Bezpečnostní testování bylo zaměřeno především na přístup do administrace, řízení oprávnění a validaci vstupních dat ve formulářích. Bylo ověřeno, že administrátorské rozhraní je dostupné pouze přihlášeným uživatelům a že přístup k jednotlivým částem administrace je omezen podle přiřazených rolí a oprávnění. Uživatel bez potřebného oprávnění nemohl provést nepovolenou operaci ani vstoupit do chráněné sekce.

Ve veřejné části systému byla kontrolována validace formulářových vstupů, například správnost e-mailové adresy a přítomnost povinných údajů při vytváření rezervací nebo objednávek. Tím bylo ověřeno, že aplikace základním způsobem omezuje vkládání neplatných dat.

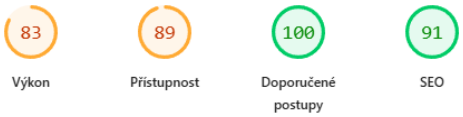
Důležitou oblastí bylo také ověření oddělení dat mezi projekty. Každý záznam je přiřazen ke konkrétnímu projektu pomocí identifikátoru projektu a testování potvrdilo, že administrace zobrazuje pouze data odpovídající aktivnímu projektu. Základní mechanismy autentizace, autorizace a oddělení dat tak byly ověřeny jako funkční.

5.2.2 Výkonnostní omezení

Výkon aplikace byl hodnocen především z pohledu rychlosti načítání stránek a efektivity práce se statickými zdroji. Pro orientační měření byl použit nástroj Google PageSpeed Insights. První měření vykázalo na mobilních zařízeních skóre 67 pro výkon, 91 pro přístupnost, 100 pro dodržení osvědčených postupů a 91 pro SEO. Na desktopových zařízeních bylo dosaženo skóre 83 pro výkon a 89 pro přístupnost, přičemž ostatní sledované metriky dosahovaly shodných hodnot jako u mobilní verze.



Obr. 14: Výsledky analýzy výkonu webové aplikace – mobilní zařízení
Zdroj: vlastní zpracování podle nástroje Google PageSpeed Insights



Obr. 15: Výsledky analýzy výkonu webové aplikace – desktop
Zdroj: vlastní zpracování podle nástroje Google PageSpeed Insights

Na základě těchto výsledků byly provedeny dílčí optimalizace, zejména komprese obrázků, přesun fontů přímo do projektu a minifikace CSS souborů. Upravena byla také struktura HTML dokumentu, aby odpovídala doporučením pro správnou hierarchii nadpisů. Po těchto úpravách došlo ke zlepšení rychlosti načítání a celkového výkonu aplikace.

Výsledky ukazují, že výkon aplikace je pro běžné použití v menších gastronomických provozech dostatečný. Při větším rozsahu nasazení by však bylo vhodné doplnit další optimalizační mechanismy, například cache nebo distribuci statických souborů prostřednictvím CDN.

5.3 Identifikace limitů systému

Testování současně umožnilo identifikovat limity vyplývající z navržené architektury a aktuálního způsobu nasazení systému. Tyto limity nepředstavují kritické chyby, ale ukazují oblasti, které by bylo nutné řešit při rozsáhlejším provozu nebo dalším vývoji aplikace.

5.3.1 Technická omezení

Jedním z omezení je absence automatizovaných testů a nástrojů pro průběžné monitorování provozu. Testování probíhalo převážně manuálně, což je pro rozsah bakalářské práce dostačující, ale při větším nasazení by bylo vhodné doplnit automatizované testovací scénáře a monitorovací nástroje.

Dalším omezením je práce se statickými soubory ukládanými přímo na souborový systém serveru. Přístup je jednoduchý a funkční pro běžné nasazení, ale při provozu na více serverech by bylo nutné řešit synchronizaci souborů nebo využít externí úložiště. Omezení představuje také absence asynchronního zpracování u některých operací, což může být problematické při vyšší zátěži nebo při zpracování časově náročnějších úloh.

5.3.2 Možnosti škálování

Systém je navržen jako modulární monolit, což přináší výhodu jednoduché struktury a snazšího nasazení, avšak při růstu počtu uživatelů nebo klientských projektů může být omezením nutnost škálovat aplikaci jako celek. Obdobně více-tenantní řešení založené na sdílené databázi představuje efektivní přístup pro menší a střední provozy, ale při výrazném rozšíření systému klade vyšší nároky na kontrolu izolace dat a správu databázové vrstvy.

Určitým omezením je také platební modul využívající Stripe Checkout, jehož současná implementace je závislá na návratu uživatele z platební brány zpět do aplikace. Pro produkční prostředí by bylo vhodné doplnit webhook mechanismus, který by potvrzení platby prováděl nezávisle na chování klienta.

5.4 Možnosti dalšího rozvoje

Na základě výsledků testování lze jako hlavní směry dalšího rozvoje označit zejména doplnění automatizovaných testů, rozšíření monitoringu provozu a zavedení pokročilejších mechanismů pro výkon a bezpečnost. Přínosné by bylo také rozšíření práce se statickými soubory o externí úložiště, doplnění asynchronního zpracování u vybraných operací a úprava platebního modulu o webhook potvrzení plateb.

Z pohledu aplikační logiky lze dále rozšiřovat jednotlivé moduly systému podle potřeb konkrétních provozů. Architektura aplikace je pro takový rozvoj připravena, nicméně při rozšíření do většího měřítka by bylo nutné doplnit i odpovídající technická opatření na úrovni infrastruktury a provozu.

Závěr

Cílem bakalářské práce bylo navrhnout a implementovat redakční systém pro gastronomické podniky, který umožňuje správu webového obsahu a současně poskytuje modulární architekturu pro postupné rozšiřování funkcionality. Stanovený cíl byl naplněn návrhem a realizací systému GastroHub.

V rámci práce vznikla plně funkční webová aplikace, která pokrývá klíčové provozní scénáře gastronomických podniků, včetně správy webových stránek, jídelního lístku, rezervací, online objednávek a hotelových rezervací. Součástí řešení je administrační rozhraní s řízením přístupových práv založeným na principu Role-Based Access Control a podpora více projektů v rámci jedné aplikace.

Navržená architektura vychází z vícevrstvého členění a vzoru Model–View–Controller. Důležitým prvkem je modulární struktura systému, která umožňuje zapínání a vypínání jednotlivých funkcí podle potřeb konkrétního provozu a usnadňuje další rozšiřování systému.

Implementované řešení je připraveno pro reálné nasazení v prostředí malých a středních gastronomických podniků, kde poskytuje ucelený nástroj pro správu webové prezentace i provozních procesů bez nutnosti využití externích systémů.

Pro nasazení ve větším měřítku nebo v prostředí s vyšší zátěží by bylo vhodné doplnit některé technické prvky, například asynchronní zpracování nebo rozšířené bezpečnostní mechanismy.

Další rozvoj systému může směřovat k rozšíření funkcionality a optimalizaci vybraných částí aplikace. Výsledkem práce je ucelené a funkční řešení, které může sloužit jako základ pro produkční nasazení i další vývoj.

Seznam použité literatury

BEDŘICH, Václav. *Rekordní růst, digitalizace gastra i změna nákupního chování. Pohled NetDirectu na dopady covidu na českou e-commerce. CzechCrunch* [online]. 2020-11-11 [cit. 2025-12-18]. Dostupné z: https://cc.cz/rekordni-rust-digitalizace-gastra-i-zmena-nakupniho-chovani-pohled-netdirectu-na-dopady-covidu-na-ceskou-e-commerce/

BEZEMER, Cor-Paul a Andy ZAIDMAN. *Cloud computing: benefits, risks and recommendations for information security* [online]. European Union Agency for Network and Information Security, 2009 [cit. 2026-03-29]. Dostupné z: https://www.enisa.europa.eu/sites/default/files/all_files/ENISA%20-%20Cloud%20Computing%20-%20final.pdf

FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures* [online]. University of California, 2000 [cit. 2026-03-29]. Dostupné z: https://roy.gbiv.com/pubs/dissertation/fielding_dissertation.pdf

Functional and Non Functional Requirements. GeeksforGeeks [online]. 2020 [cit. 2025-11-02]. Dostupné z: https://www.geeksforgeeks.org/software-engineering/functional-vs-non-functional-requirements/

GUPTA, Shivani. *What is WordPress?* GeeksforGeeks [online]. 2025-08-21 [cit. 2025-12-18]. Dostupné z: https://www.geeksforgeeks.org/wordpress/what-is-wordpress/

HARTINGER, David. *Monolitická a dvouvrstvá architektura.* ITnetwork [online]. 2020a [cit. 2025-11-03]. Dostupné z: https://www.itnetwork.cz/navrh/architektury-a-dependency-injection/monoliticka-a-douvrstva-architektura

HARTINGER, David. MVC architektura. ITnetwork [online]. 2020b [cit. 2025-11-03]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor

HARTINGER, David. *Třívrstvá architektura a další vícevrstvé architektury.* ITnetwork [online]. 2020c [cit. 2025-11-03]. Dostupné z: https://www.itnetwork.cz/navrh/architektury-a-dependency-injection/trivrstva-architektura-a-dalsi-vicevrstve-architektury

Chapter 10: MoSCoW Prioritisation. Agile Business Consortium [online]. [s. d.] [cit. 2025-11-02]. Dostupné z: https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html

Choice [online]. 2020 [cit. 2025-12-18]. Dostupné z: https://choiceqr.com/cs/

Choice QR s.r.o. StartupJobs [online]. [cit. 2025-10-31]. Dostupné z: https://www.startupjobs.cz/startup/choice-qr-s-r-o-1

MELL, Peter a Timothy GRANCE. *The NIST Definition of Cloud Computing* [online]. NIST Special Publication 800-145. Gaithersburg, MD, 2011 [cit. 2026-03-29]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Nejoblíbenější webhosting. WEDOS [online]. 2025-12-08 [cit. 2025-12-18]. Dostupné z: https://vedos.cz/webhosting/

OWASP Top Ten Web Application Security Risks. OWASP [online]. 2025 [cit. 2026-02-25]. Dostupné z: https://owasp.org/www-project-top-ten/

Plugins. WordPress [online]. 2025 [cit. 2025-12-18]. Dostupné z: https://wordpress.org/plugins/

SANDHU, Ravi, David FERRAIOLO a Richard KUHN. *The NIST Model for Role-Based Access Control: Towards a Unified Standard.* 2000 [cit. 2026-03-29]. Dostupné z: https://www.nist.gov/publications/nist-model-role-based-access-control-towards-unified-standard

SILBERSCHATZ, Abraham, Henry F. KORTH a S. SUDARSHAN. *Database System Concepts. 7th ed.* McGraw-Hill, 2019. ISBN 9780078022159.

SOMMERVILLE, Ian. *Software engineering. 10th ed.* Harlow: Pearson, 2016. ISBN 978-0-13-394303-0.

TEAGANNE, Finn a Amanda DOWNIE. *What is a content management system (CMS)?* IBM [online]. 2025-11-17 [cit. 2025-12-18]. Dostupné z: https://www.ibm.com/think/topics/content-management-system

Upmenu [online]. 2008 [cit. 2025-10-31]. Dostupné z: https://www.upmenu.com/

VERMA, Anushka. *Benefit of using MVC.* GeeksforGeeks [online]. 2023-04-15 [cit. 2026-03-23]. Dostupné z: https://www.geeksforgeeks.org/software-engineering/benefit-of-using-mvc/

What is RBAC (Role-Based Access Control)? IBM Think [online]. [s. d.] [cit. 2025-11-02]. Dostupné z: https://www.ibm.com/think/topics/rbac

What is SaaS (Software as a Service)? IONOS [online]. 2023-03-20 [cit. 2026-03-23]. Dostupné z: https://www.ionos.co.uk/digitalguide/server/know-how/an-overview-of-saas-software-as-a-service/

Přílohy

Příloha 1 – Kompletní datový model