DELTA – Střední škola informatiky a ekonomie, s.r.o.

Ke Kamenci 151, Pardubice

Implementace backendové části aplikace pro sestavování newsletterů

Ungvári, Erik

4.B

Informační technologie 18-20-M/01

2025/2026

Jméno a příjmení: Erik Ungvári
Pro školní rok: 2025/2026
Třída: 4. B
Obor: Informační technologie 18-20-M/01
Téma práce: Vývoj backendového rozhraní pro aplikaci pro sestavování newsletterů s napojením na e-shopy
Vedoucí práce: Jan Čech
Cíl projektu:
Cílem tohoto projektu je vyvinout backendovou část aplikace pro efektivní sestavování newsletterů, která bude zahrnovat systém pro registraci uživatelských účtů, napojení na produktová data e-shopů našich klientů a integraci těchto dat do aplikace pro sestavení newsletterů. Backend bude postaven na Nette frameworku a bude navržen tak, aby aplikace směřovala do modelu SaaS (Software as a Service).
Specifikace projektu:

Analýza a návrh:

- Provedení analýzy požadavků na backendovou část aplikace.
- Výběr vhodné technologie pro implementaci backendu aplikace, s využitím Nette frameworku.
- Návrh architektury aplikace, která bude umožňovat snadnou registraci uživatelských účtů a připojení k e-shopům.
- Návrh modelu pro správu uživatelských účtů s ohledem na budoucí expanze do SaaS modelu.

Implementace uživatelského rozhraní pro registraci účtu:

- Vytvoření systému pro registraci, přihlášení a správu uživatelských účtů.
- Implementace bezpečné autentizace a správy přístupových práv.
- Implementace rozhraní pro správu účtů, včetně možnosti správy e-shopů a přiřazování konkrétních produktových dat do newsletterů.
- Zajištění zabezpečení aplikace proti neoprávněnému přístupu a zneužití dat.

Napojení na e-shopy a integrace produktových dat:

- Analýza možností napojení na produktová data e-shopů našich klientů.
- Výběr vhodné technologie pro integraci produktových dat (API, XML feed, CSV import).

- Implementace systému pro získávání a ukládání produktových dat do aplikace.
- Napojení na API nebo jiné systémy klientů pro automatickou aktualizaci produktových dat v aplikaci.

SaaS model a škálovatelnost:

- Příprava backendu aplikace pro přechod do modelu SaaS (Software as a Service).
- Zajištění toho, že aplikace bude schopna efektivně obsluhovat více klientů, přičemž každý klient bude mít vlastní data a vlastní konfiguraci.
- Navržení systému pro správu různých úrovní přístupu uživatelů podle jejich role (např. administrátor, editor).

Testování a ladění:

- Testování správnosti a výkonnosti registrace účtů a správy uživatelských dat.
- Testování napojení na e-shopy a pravidelných aktualizací produktových dat.
- Testování bezpečnosti aplikace a správnosti implementace autentizace.

Požadované výstupy:

- Funkční backendová část aplikace pro správu uživatelských účtů a napojení na e-shopy.
- Implementovaný systém pro registraci a správu účtů, včetně autentizace a přiřazování e-shopů.
- Napojení na produktová data e-shopů a implementace jejich integrace do aplikace pro tvorbu newsletterů.
- Připravenost backendu pro rozšíření aplikace do modelu SaaS.
- Uživatelská a vývojářská dokumentace.
- Prezentace projektu a případně demonstrační video ukazující klíčové funkce backendu aplikace.

Hodnocení:

Projekt bude hodnocen na základě následujících kritérií:

- Kvalita a funkčnost implementace backendu.
- Efektivita a správnost napojení na produktová data e-shopů.
- Schopnost backendu zvládnout různé klienty v SaaS modelu.

- Úroveň dokumentace a prezentace projektu.
- Kvalita a bezpečnost implementace uživatelských účtů.

Stručný časový harmonogram (s daty a konkretizovanými úkoly):

- **Září:** Analýza požadavků, návrh backendové struktury aplikace a technologie pro napojení na e-shopy.
- **Říjen:** Implementace systému pro registraci a správu uživatelských účtů.
- **Listopad:** Implementace napojení na produktová data e-shopů a integrace dat do aplikace.
- **Prosinec:** Příprava aplikace na rozšíření do SaaS modelu.
- **Leden:** Testování a ladění backendu, integrace s frontendem.
- **Únor - březen:** Finalizace projektu, práce na dokumentaci a příprava na prezentaci.

Prohlašuji, že jsem maturitní projekt vypracoval(a) samostatně, výhradně s použitím uvedené literatury.

V Pardubicích 30. 3. 2023

(vlastnoruční podpis)

Upřímně děkuji panu Janu Čechovi za odborné vedení při zpracovávání maturitního projektu, za konzultace ohledně architektury systému a metodickou podporu. Dále děkuji panu Dominiku Šlechtovi za konzultace v oblasti bezpečnosti API komunikace a šifrování. Zvláštní poděkování patří Janu Kocandovi, který zpracoval frontendovou část projektu jako svou vlastní maturitní práci a umožnil tak realizaci kompletní funkční aplikace.

Resumé

Tato maturitní práce se zabývá návrhem a implementací backendového rozhraní pro webovou aplikaci určenou k sestavování newsletterů s napojením na e-shopové platformy. Hlavním cílem je automatizace procesu vkládání produktových dat do newsletterů, který je při ručním zpracování časově náročný a náchylný k chybám.

Systém je postaven na PHP frameworku Nette 3.2 s využitím ORM knihovny Nextras 5 a databáze MariaDB. Komunikace s externím e-shopem probíhá prostřednictvím zabezpečeného REST API s autentizací založenou na algoritmu HMAC-SHA256. Součástí řešení je Composer balíček eshop-api-exposer, který je instalován na stranu e-shopu a vystavuje produktové API včetně supplementů (doplňkových údajů) a obrázků.

Práce zahrnuje návrh databázového modelu, UML diagramy (Use Case, ER, diagram tříd, sekvenční diagram), kompletní dokumentaci API endpointů a popis implementace jednotlivých komponent systému. Backend je navržen s ohledem na budoucí rozšíření do modelu SaaS (Software as a Service).

Klíčová slova

Informační systém, REST API, newsletter, e-shop, databázový model, UML, HMAC autentizace, Nette framework, Nextras ORM, Composer balíček

Abstract

This thesis focuses on the design and implementation of a backend interface for a web application used to compose newsletters integrated with e-commerce platforms. The system is built on the PHP Nette framework using Nextras ORM and a MariaDB database. Communication with e-shops is handled through a secured REST API with HMAC-SHA256 authentication. The solution includes a Composer package (eshop-api-exposer) installed on the e-shop side to expose product data including supplements and images. The backend is designed with future SaaS scalability in mind.

Keywords

Information system, REST API, newsletter, e-commerce, database model, UML, HMAC authentication, Nette framework, Nextras ORM, Composer package

1 Úvod

E-mailový marketing patří mezi nejefektivnější nástroje online komunikace. Podle průzkumů generuje každý dolar investovaný do e-mailového marketingu průměrně 36 dolarů návratnosti. [1] Pro firmy provozující e-shopy je pravidelné odesílání newsletterů klíčovým kanálem pro udržení kontaktu se zákazníky, propagaci produktů a zvyšování obratu.

Problémem však zůstává integrace produktových dat z e-shopu do newsletteru. Ruční kopírování názvů, cen a obrázků je časově náročné, náchylné k chybám a neefektivní. Existující nástroje jako Mailchimp či SmartEmailing nabízí integrace s vybranými platformami, avšak pro menší české e-shopy postavené na vlastních řešeních často chybí přímá podpora. Navíc tyto nástroje neumožňují pracovat se specifickými datovými strukturami jednotlivých e-shopů, jako jsou supplementy (doplňkové údaje u produktů).

Cílem této maturitní práce je navrhnout a implementovat backendové rozhraní pro aplikaci pro sestavování newsletterů s přímou integrací na produktová data e-shopů. Práce zahrnuje návrh databázového modelu, implementaci REST API pro frontendovou aplikaci, zabezpečenou komunikaci s externím e-shopem pomocí HMAC-SHA256 autentizace a vytvoření Composer balíčku eshop-api-exposer, který je instalován na stranu e-shopu.

Backend je postaven na Nette frameworku [2] s ORM Nextras [3] a databází MariaDB. Komunikace s e-shopem probíhá přes REST API s HMAC-SHA256 autentizací. [4] Frontendová část aplikace je předmětem samostatné maturitní práce Jana Kocandy. Důraz je kladen na bezpečnost, modulární strukturu a připravenost pro SaaS model.

2 Teoretická část

2.1 Informační systémy a architektura klient-server

Informační systém je soubor softwarových, hardwarových a organizačních prostředků sloužících ke zpracování, ukládání a přenosu informací. [5] Moderní webové informační systémy jsou často navrhovány jako distribuované aplikace, kde jednotlivé části systému komunikují prostřednictvím síťového rozhraní.

V praxi se využívá architektura klient-server, kde klient (typicky webový prohlížeč) komunikuje se serverem prostřednictvím HTTP protokolu. Server zpracovává požadavky, přistupuje k databázi a vrací data ve formátu JSON. Distribuoáná

architektura umožňuje oddělení zodpovědností – frontendová vrstva se stará o prezentaci, backendová o obchodní logiku a data. Tím je umožněn nezávislý vývoj obou částí, což bylo v případě tohoto projektu využito – backendové a frontendové rozhraní jsou vyvíjeny odděleně s definovaným API kontraktem.

2.2 REST API

REST (Representational State Transfer) je architektonický styl používaný při návrhu webových služeb. [6] REST API využívá standardní HTTP metody pro práci se zdroji (resources): GET pro získání, POST pro vytvoření, PUT pro aktualizaci a DELETE pro odstranění záznamu. Komunikace je bezstavová (stateless) – každý požadavek obsahuje veškeré informace potřebné k jeho zpracování.

Výhodou REST API je jednoduchost, univerzální kompatibilita a možnost využití libovolným klientem schopným HTTP komunikace. V tomto projektu je REST API využito dvojím způsobem: interně mezi frontendovou aplikací a backendem, a externě mezi backendem sestavovače a API e-shopu.

2.3 HMAC autentizace

Při komunikaci mezi dvěma systémy je nutné zajistit, že požadavek pochází od autorizovaného klienta a nebyl po cestě modifikován. K tomuto účelu slouží HMAC (Hash-based Message Authentication Code) – mechanismus pro ověření integrity a autenticity zpráv na principu sdíleného tajného klíče. [4]

Proces autentizace probíhá následovně: klient vytvoří časové razítko (timestamp), zkombinuje ho s API klíčem a pomocí sdíleného secretu vypočítá kryptografický podpis algoritmem SHA-256. Podpis je odeslán spolu s požadavkem v HTTP hlavičkách. Server přijatý podpis ověří konstantním časovým porovnáním (funkce hash_equals), což zamezuje timing attackům. Timestamp je omezen na maximálně 5 minut od aktuálního času serveru jako ochrana proti replay attackům.

2.4 Volba technologií

Při volbě technologií pro tento projekt hrál zásadní roli praktický kontext. Všechny použité nástroje jsou standardní součástí technologického stacku firmy, ve které autor působí. Veškeré e-shopy a webové aplikace firmy jsou postaveny na téže kombinaci Nette, Nextras ORM a MariaDB. Tím bylo možné využít zkušenosti z reálných komerčních projektů a současně zajistit bezproblémovou integraci výsledné aplikace do stávající firemní infrastruktury.

2.4.1 Nette framework

Pro backendovou část byl zvolen Nette framework [2] – český open-source PHP framework, který je ve firmě využíván pro veškeré projekty. Oproti alternativám jako Laravel nebo Symfony nabízí Nette několik výhod zásadních pro daný typ projektů. Automatická ochrana proti XSS a CSRF útokům je zabudována přímo do šablonovacího systému Latte a formulářového frameworku, což výrazně snižuje riziko bezpečnostních zranitelností. Dependency injection kontejner umožňuje čistou architekturu bez skrytých závislostí. V porovnání se Symfony je Nette kompaktnější, což se pozitivně projevuje na odezvě API endpointů.

Podstatným důvodem pro volbu Nette je rovněž kompatibilita s balíčkem eshop-api-exposer, který se registruje jako Nette DI extension a využívá Nette presentery pro zpracování API požadavků. Použití stejného frameworku na obou stranách (sestavovač i e-shop) umožňuje sdílet konvence a zjednodušuje údržbu.

2.4.2 Nextras ORM

Jako ORM knihovna byla zvolena Nextras ORM [3], která je navržena specificky pro Nette ekosystém. Oproti Doctrine ORM, používané především v Symfony komunitě, je Nextras ORM výrazně jednodušší na konfiguraci a lépe se integruje s Nette DI kontejnerem. Relace mezi entitami se definují anotacemi přímo v třídě entity ({m:1}, {1:m}, {m:m}), což odstraňuje potřebu externích konfiguračních souborů.

S touto knihovnou existují ve firmě bohaté zkušenosti z desítek projektů, včetně znalosti jejích omezení – například nutnost explicitního mapování sloupců v Mapper třídě, pokud automatická konvence camelCase → snake_case nepracuje správně. Tato znalost se ukázala jako klíčová při řešení problémů s mapováním entity Product.

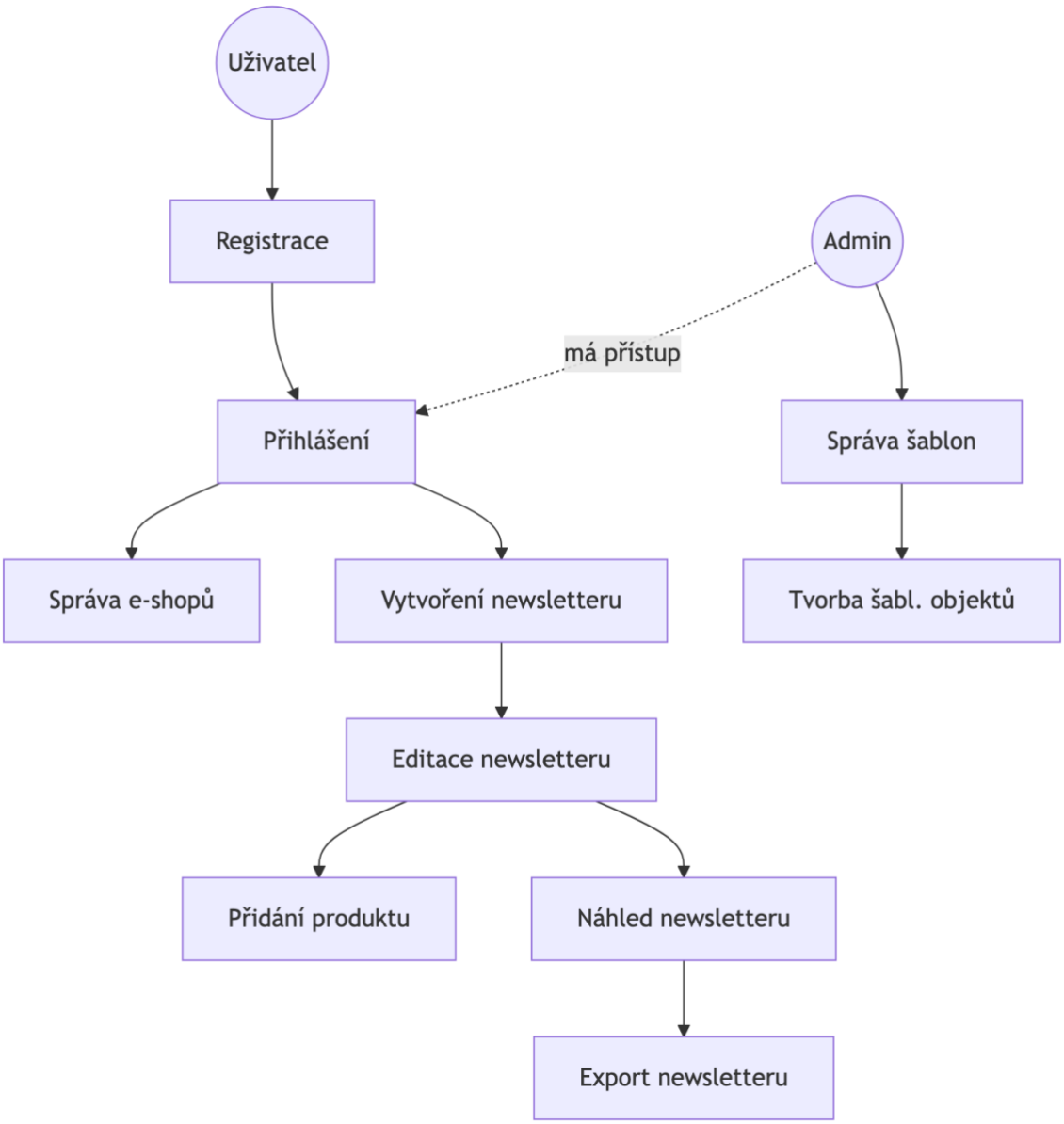
2.4.3 MariaDB

Jako databázový systém je použita MariaDB – open-source fork MySQL, který nabízí plnou kompatibilitu s MySQL a současně lepší výkon u složitějších dotazů. Ve firmě je MariaDB standardním databázovým systémem pro všechny projekty včetně e-shopů, jejichž databáze jsou přímo čteny balíčkem eshop-api-exposer.

3 Návrh systému

3.1 Use Case diagram

Diagram případů užití (obrázek 1) zachycuje funkčnost systému z pohledu dvou aktérů. Běžný uživatel se může zaregistrovat, přihlásit, spravovat své e-shopy (přidávání, generování API klíčů), vytvářet a editovat newslettery, přidávat produkty z e-shopu do jednotlivých bloků, zobrazit náhled a exportovat výsledek do HTML. Administrátor dědí všechny schopnosti uživatele a navíc může spravovat šablony a vytvářet šablonové objekty.

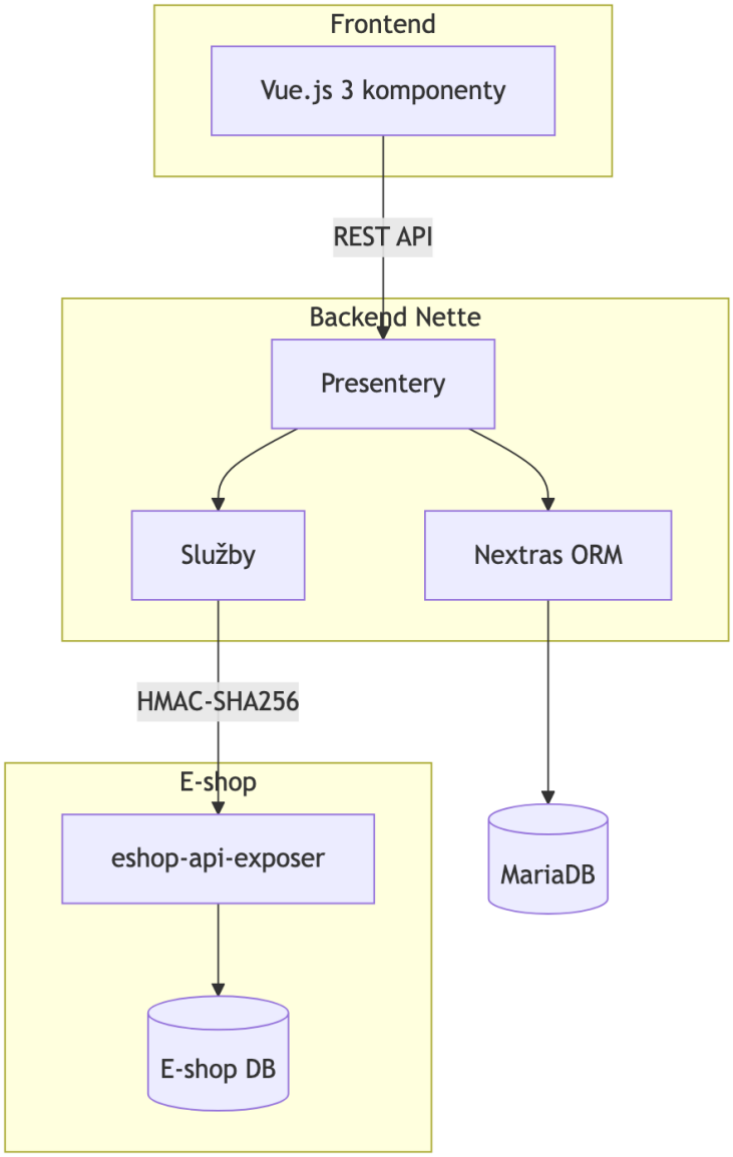


Obrázek 1: Use Case diagram systému

3.2 Architektura systému

Systém je navržen jako třívrstvá aplikace (obrázek 2). Frontendová vrstva (Vue.js 3 SPA) běží v prohlížeči uživatele a komunikuje s backendem přes interní REST API. Backendová vrstva (Nette PHP, Nextras ORM, MariaDB) zpracovává požadavky, spravuje data a komunikuje s externím e-shopem. Třetí vrstvou je externí API e-shopu poskytované balíčkem eshop-api-exposer.

Komunikace mezi backendem a e-shopem je zabezpečena HMAC-SHA256 podpisem. API klíč a secret jsou uloženy šifrovaně v databázi sestavovače (libsodium). Při každém požadavku je secret dešifrován, vytvořen podpis a odeslán v HTTP hlavičkách.

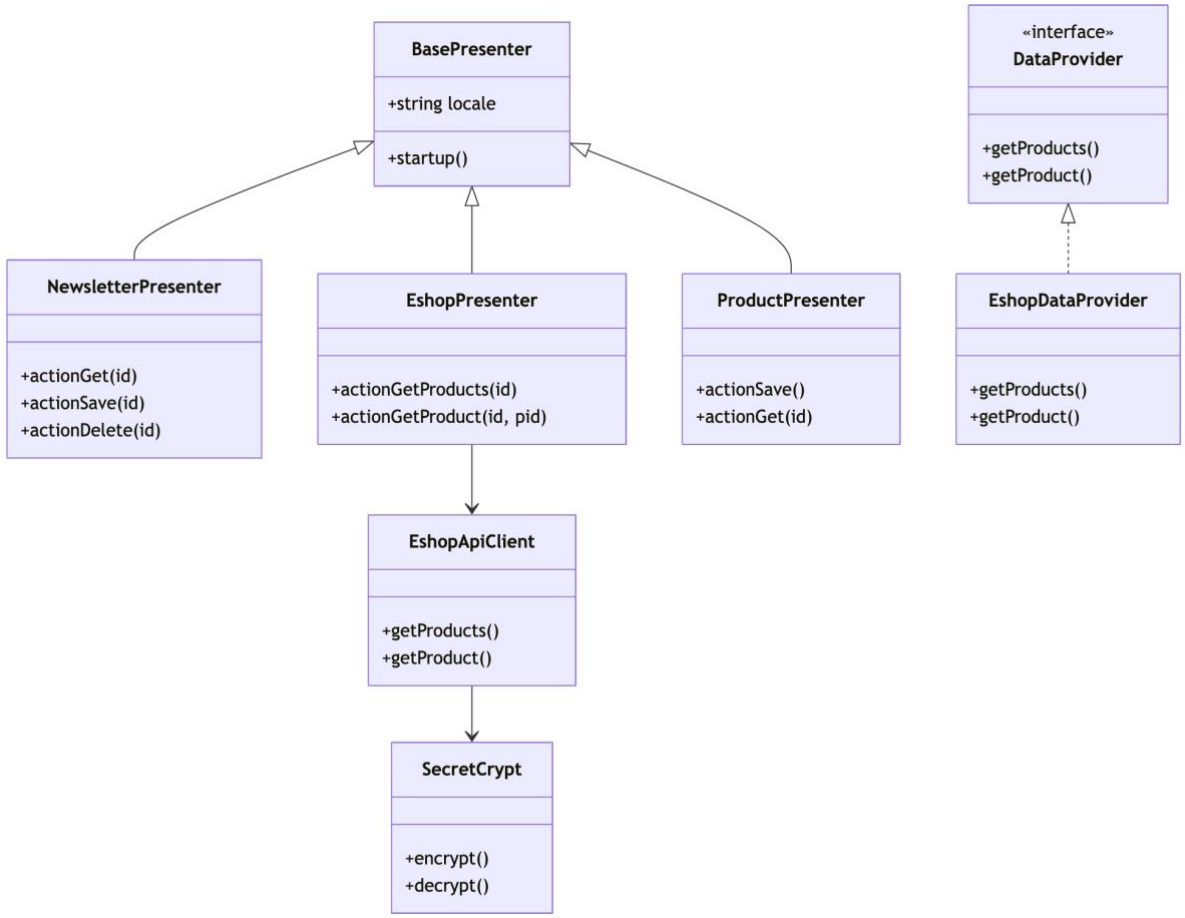


Obrázek 2: Diagram architektury systému

3.3 Diagram tříd

Diagram tříd (obrázek 3) zobrazuje hlavní třídy backendu sestavovače a e-shopového balíčku. Na straně sestavovače všechny presentery (NewsletterPresenter, EshopPresenter, ProductPresenter) dědí z BasePresenter, který zajišťuje autentizaci, lokalizaci a kontrolu oprávnění. EshopPresenter využívá EshopApiClient pro komunikaci s externím API a SecretCrypt pro dešifrování API secretu.

Na straně e-shopu je definováno rozhraní DataProvider s metodami getProducts() a getProduct(). Toto rozhraní implementuje třída EshopDataProvider, která čte data přímo z databáze e-shopu. Pro testovací účely existuje i DefaultDataProvider vracející prázdná data.

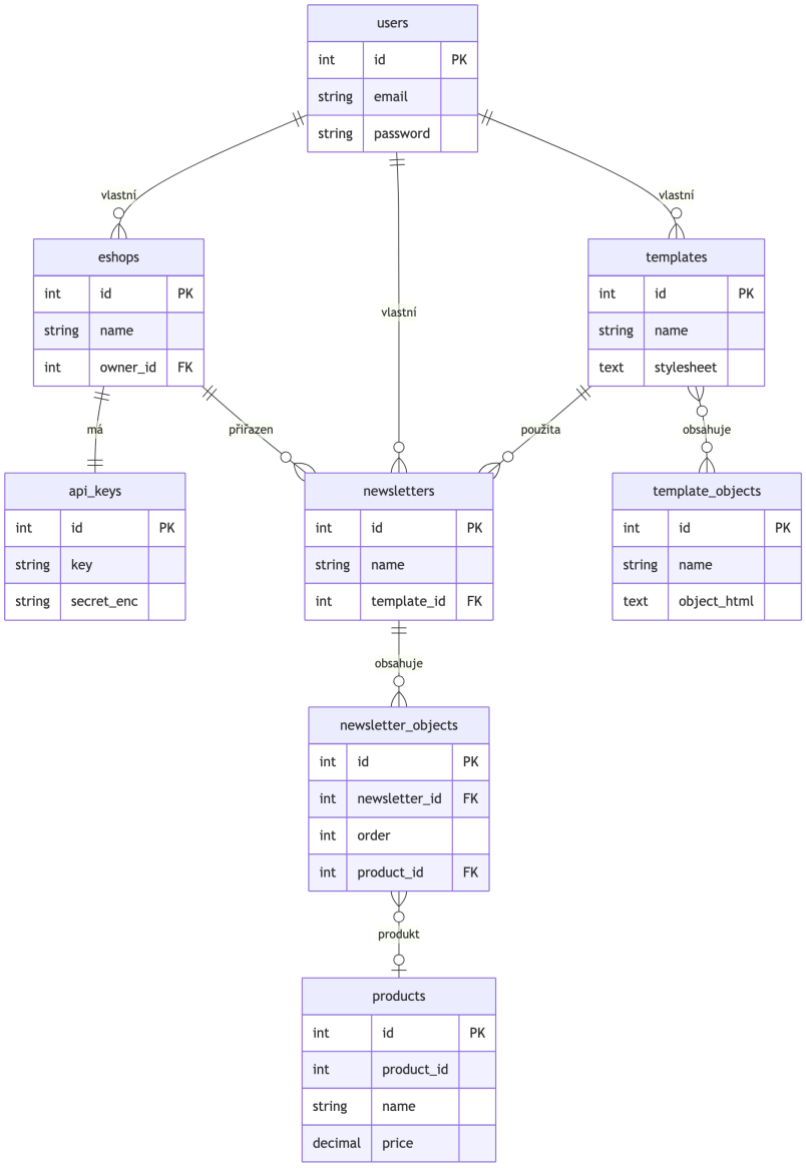


Obrázek 3: Diagram tříd

3.4 ER diagram databáze

Entitně-relační diagram (obrázek 4) zachycuje strukturu databáze sestavovače newsletterů. Centrální entitou je tabulka newsletters, která je propojena s uživatelem (owner), šablonou (template) a volitelně s e-shopem (eshop). Každý newsletter obsahuje kolekci objektů (newsletter_objects), které představují jednotlivé bloky obsahu. Každý objekt může mít přiřazený produkt z lokální tabulky products.

Tabulka users obsahuje uživatelské účty s rozlišením rolí. Tabulka eshops je propojena 1:1 s api_keys, kde jsou uloženy šifrované API přístupové údaje. Šablony (templates) obsahují CSS stylesheet a jsou propojeny s šablonovými objekty (template_objects), které definují HTML strukturu jednotlivých bloků.



Obrázek 4: ER diagram databáze

4 Metodika a vlastní řešení

4.1 Postup práce

Vývoj probíhal v několika fázích v souladu s harmonogramem uvedeným v zadání. V září a říjnu byla provedena analýza požadavků, návrh databázového modelu a implementace systému pro registraci a správu uživatelských účtů. V listopadu bylo realizováno napojení na produktová data e-shopů – návrh a implementace balíčku eshop-api-exposer. V prosinci byl backend připraven pro SaaS model (validace vlastnictví dat, podpora více e-shopů). V lednu proběhlo testování a integrace s frontendovou částí. V únoru a březnu byla finalizována dokumentace a opraveny nalezené chyby.

Zdrojové kódy byly průběžně verzovány v Git repozitáři. Vývoj probíhal v Docker prostředí pro zajištění konzistentního běhového prostředí nezávislého na lokálním nastavení vývojáře.

4.2 Architektura backendu

Backend je implementován v PHP 8.2 s frameworkem Nette 3.2 a ORM Nextras 5. Architektura sleduje vzor MVP (Model–View–Presenter), kde každý funkční celek má vlastní presenter zpracovávající HTTP požadavky. Všechny presentery dědí z BasePresenter, který zajišťuje:

- Ověření přihlášení – nepřihlášení uživatelé jsou přesměrováni na přihlašovací stránku
- Lokalizaci – persistentní parametr locale umožňující přepínat mezi češtinou a angličtinou (ve vývoji)
- Kontrolu rolí – systém rozlišuje role Admin a User s různými oprávněními
- CSRF ochranu formulářů – automatické přidání tokenu prostřednictvím FormFactory

Pro API endpointy (akce vracející JSON) je v metodě startup() každého presenteru selektivně vypnuto autoCanonicalize chování Nette. Bez tohoto vypnutí by Nette přesměrovávalo PUT a POST požadavky 302 redirectem za účelem zahrnutí persistentních parametrů, což by vedlo ke ztrátě JSON těla požadavku.

4.2.1 Autentizace uživatelů

Pro autentizaci byla vytvořena vlastní implementace rozhraní Nette\Security\Authenticator. Při přihlášení je e-mailová adresa normalizována na malá písmena a heslo je ověřováno vůči bcrypt hashi uloženému v databázi. Po úspěšném

ověření je vytvořen objekt SimpleIdentity obsahující ID uživatele a jeho roli. Platnost session je nastavena na 4 hodiny.

Při obnovení session (metoda wakeupIdentity) jsou data uživatele znovu načtena z databáze. Tím je zajištěno, že změna role administrátorem se projeví okamžitě, aniž by se uživatel musel znovu přihlašovat. Tato technika je důležitá zejména v kontextu SaaS modelu, kde správce může měnit oprávnění uživatelů za provozu.

4.2.2 Šifrování API klíčů

API secret každého e-shopu je šifrován pomocí knihovny libsodium (algoritmus XChaCha20-Poly1305). [7] Při šifrování je generován náhodný 24bajtový nonce, který je připojen před šifrovaný text. Výsledek je kódován v base64 a uložen do sloupce api_secret_encrypted.

Při vytvoření nového e-shopu je automaticky generován API klíč ve formátu eshop_[32 hexadecimálních znaků] a secret o délce 64 hexadecimálních znaků. Secret je uživateli zobrazen pouze jednou při vytvoření – po zavrření dialogu je v databázi uložen pouze v šifrované podobě a nelze ho již získat.

4.2.3 Správa newsletterů

Při vytvoření newsletteru uživatel zvolí název, šablonu a volitelně e-shop. Systémem jsou automaticky zkopírovány objekty (bloky) z vybrané šablony. Každý objekt má obsah (content), pořadí (order) a volitelně odkaz na produkt. Obsah jednoho objektu může být rozdělen oddělovačem ";" pro více editovatelných oblastí v rámci jednoho bloku.

Ukládání přijímá JSON payload s aktualizovanými daty existujících objektů a s novými objekty. Serverem jsou zpracovány změny, vytvořeny nové objekty a vráceno mapování dočasných ID (generovaných frontendem) na reálná databázová ID. Při smazání objektu je automaticky smazán i přiřazený produkt, čímž je zabráněno vzniku orphan záznamů.

4.2.4 Integrace produktů z e-shopu

Produkty jsou načítány z externího API e-shopu a ukládány do lokální databáze. Při ukládání je aplikován filtr názvu – pokud je k produktu přiřazen supplement Nadpis_H1 (ID 26 v databázi e-shopu), je jeho hodnota použita jako zobrazovaný název. Tím je marketingovým týmům umožněno přizpůsobit názvy produktů pro newsletter bez nutnosti měnit data v e-shopu.

Všechny API endpointy ověřují vlastnictví zdrojů – při přístupu k newsletteru, e-shopu či produktu je kontrolováno, zda přistupující uživatel je vlastníkem daného záznamu. V opačném případě je vrácen HTTP 403 Forbidden (ochrana proti IDOR útokům).

4.3 Databázový model

Databáze běží na MariaDB s ORM knihovnou Nextras. Každá tabulka má v aplikaci odpovídající trojici tříd: Entity (definice sloupců a relací v anotacích), Repository (dotazy a ukládání) a Mapper (mapování názvů sloupců na vlastnosti entity). Následující tabulky popisují strukturu jednotlivých databázových tabulek.

Tab. 1: users – uživatelské účty

Sloupec	Typ	Popis
id	int, PK	Primární klíč, auto-increment
email	varchar(255)	Unikátní e-mailová adresa
password	varchar(255)	Heslo hashované algoritmem bcrypt
role	varchar(50)	Role uživatele: Admin nebo User
date_created	datetime	Datum registrace

Tab. 2: eshops – připojené e-shopy

Sloupec	Typ	Popis
id	int, PK	Primární klíč
name	varchar(255)	Název e-shopu
owner_id	int, FK	Cizí klíč na users. Relace 1:N.
created_at	datetime	Datum vytvoření

Tab. 3: api_keys – API přístupové údaje (relace 1:1 s eshops)

Sloupec	Typ	Popis
id	int, PK	Primární klíč
key	varchar(255)	Veřejný API klíč (formát eshop_[32hex])
api_secret_encrypted	text	Šifrovaný secret (libsodium, base64)
api_url	varchar(255)	Základní URL API e-shopu
eshop_id	int, FK	Cizí klíč na eshops
expiration_date	datetime, null	Datum vypršení klíče

Tab. 4: newsletters – newslettery

Sloupec	Typ	Popis
id	int, PK	Primární klíč
name	varchar(255)	Název newsletteru
description	text, null	Popis newsletteru
template_id	int, FK	Cizí klíč na templates
eshop_id	int, FK, null	Cizí klíč na eshops (volitelné)
owner_id	int, FK	Cizí klíč na users
created_at	datetime	Datum vytvoření
updated_at	datetime	Datum poslední změny

Tab. 5: newsletter_objects – bloky v newsletteru

Sloupec	Typ	Popis
id	int, PK	Primární klíč
newsletter_id	int, FK	Cizí klíč na newsletters (kaskádní mazání)
template_object_id	int	ID šablonového objektu
name	varchar(255)	Název bloku
content	text	Obsah bloku (oddělovač ;;; pro více oblastí)
order	int	Pořadí v newsletteru
product_id	int, FK, null	Cizí klíč na products (volitelné)

Tab. 6: templates – šablony

Sloupec	Typ	Popis
id	int, PK	Primární klíč
name	varchar(255)	Název šablony
eshop_id	int, null	Přiřazený e-shop (volitelné)
owner_id	int, FK	Cizí klíč na users
stylesheet	text	CSS styly šablony
created_at	datetime	Datum vytvoření

Tab. 7: template_objects – šablonové HTML bloky

Sloupec	Typ	Popis
id	int, PK	Primární klíč
name	varchar(255)	Název bloku
object_html	text	HTML s placeholdery: {{content}}, {{product_name}}, {{product_price}}, {{product_img}}, {{product_link}}

Tab. 8: products – lokálně uložené produkty

Sloupec	Typ	Popis
id	int, PK	Primární klíč
product_id	int	Externí ID produktu v e-shopu
lang	int	Jazyk produktu
eshop_id	int	ID původního e-shopu
name	varchar(255)	Název (po aplikaci filtru Nadpis_H1)
price	decimal	Cena s DPH
stock	varchar(255), null	Stav skladu
img	varchar(255), null	Cesta k obrázku
url	varchar(255), null	URL produktu

4.4 Composer balíček eshop-api-exposer

Balíček czechgroup/eshop-api-exposer běží na straně e-shopu a poskytuje zabezpečené REST API pro přístup k produktovým datům. Registruje se jako Nette DI extension (EshopApiExtension), čímž je automaticky zapojen do DI kontejneru e-shopu. Balíček se skládá z následujících komponent:

4.4.1 ApiPresenter

Zpracovává HTTP požadavky na endpointech /api/eshop/products (seznam všech produktů) a /api/eshop/products/{id} (detail jednoho produktu). Před zpracováním každého požadavku je volán RequestAuthenticator pro ověření HMAC podpisu. Podporován je volitelný parametr locale pro vícejazyčnou podporu.

4.4.2 RequestAuthenticator

Ověřuje HMAC-SHA256 podpisy příchozích požadavků. Z HTTP hlaviček jsou extrahovány hodnoty X-Api-Key, X-Timestamp a X-Signature. Podpis je porovnáván konstantním časovým porovnáním (hash_equals). Timestamp musí být v rozmezí 5 minut od aktuálního času serveru – starší požadavky jsou automaticky odmítnuty jako ochrana proti replay attackům.

4.4.3 DataProvider a EshopDataProvider

Rozhraní DataProvider definuje dvě metody: getProducts(?string $locale) pro seznam produktů a getProduct(int $productId, ?string $locale) pro detail jednoho produktu. Třída EshopDataProvider implementuje toto rozhraní a čte data přímo z databáze e-shopu. Pro každý produkt jsou načteny:

- Základní údaje z tabulky product (název, cena, sklad, kód)
- Obrázky z tabulky settings_image (řazeny podle pozice)
- Supplementy z pivot tabulky product_x_supplement (klíčový Nadpis_H1, ID 26)
- Cena s DPH (vypočtena jako price * 1.21)

Pro testovací účely existuje DefaultDataProvider vracející prázdná data bez přístupu k databázi.

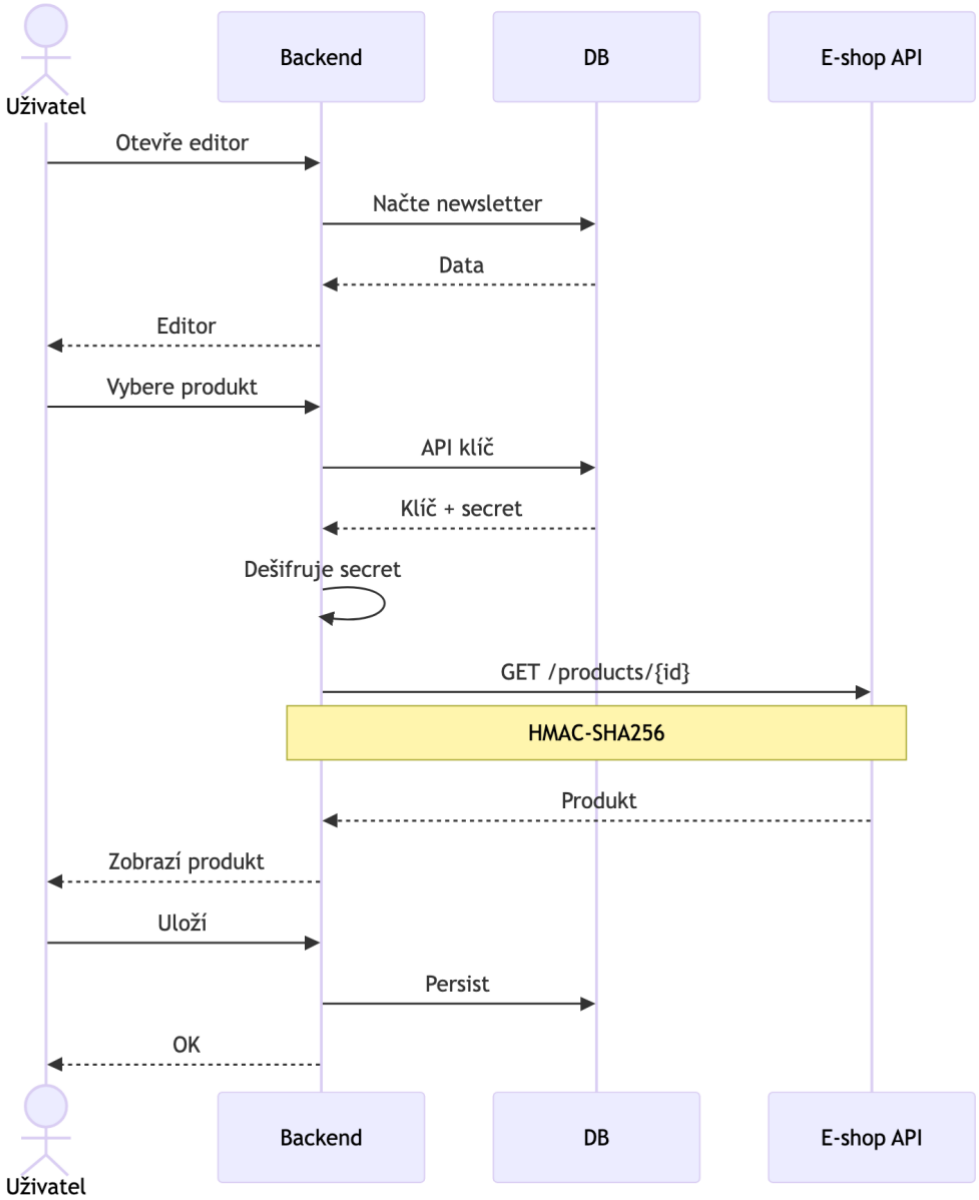
4.4.4 Klientská strana – EshopApiClient

Na straně sestavovače je komunikace s API e-shopu zajištěna třídou EshopApiClient využívající knihovnu GuzzleHttp. [8] Při každém požadavku je vytvořeno časové razítko, dešifrován API secret pomocí SecretCrypt (libsodium), vypočítán HMAC-SHA256 podpis

a požadavek odeslán s příslušnými hlavičkami. Odpověď je dekódována z JSON a vrácena presenteru.

4.5 Sekvenční diagram API komunikace

Obrázek 5 zachycuje kompletní průběh načtení produktu z e-shopu do newsletteru. Uživatel otevře editor newsletteru, backend načte data z databáze. Uživatel vybere produkt – backend dešifruje API secret, vytvoří HMAC podpis a zavolá externí API. E-shop ověří podpis, načte data z databáze včetně supplementů a obrázků a vrátí je. Backend uloží produkt lokálně a zobrazí uživateli.



Obrázek 5: Sekvenční diagram API komunikace

4.6 Dokumentace API endpointů

Všechny API endpointy vyžadují autentizaci uživatele (Nette session) a validují vlastnictví zdrojů. Endpointy jsou definovány v RouterFactory pomocí Nette Route s explicitním mapováním HTTP metod.

4.6.1 E-shop endpointy

- GET /api/eshop/{id} – detail e-shopu (název, URL). Ověřuje vlastnictví.
- GET /api/eshop/{id}/products – seznam produktů z externího API e-shopu (HMAC autentizace).
- GET /api/eshop/{id}/products/{pid} – detail produktu vč. supplementů a obrázků.
- GET /api/eshop/{id}/url – základní URL e-shopu pro sestavení odkazů na produkty.

4.6.2 Newsletter endpointy

- GET /api/newsletter/{id} – detail newsletteru (toArray s relacemi serializovanými jako ID).
- GET /api/newsletter/{id}/get-objects – objekty s HTML šablonou, content, order a product_id.
- POST /api/newsletter/{id}/save – uložení. JSON payload: updatedData (změny existujících objektů) + newObjects (nové objekty). Vrací newObjectMapping.
- DELETE /api/newsletter/{id}/delete – smazání newsletteru s kaskádním smazáním objektů.

4.6.3 Newsletter objekt endpointy

- POST /api/newsletter-object/create – vytvoření objektu (template_object_id, name, content, order).
- DELETE /api/newsletter-object/delete/{id} – smazání objektu + automatické smazání přiřazeného produktu.
- POST /api/newsletter-object/{id}/update-product – přiřazení produktu (JSON: productId). Starý produkt je automaticky smazán.

4.6.4 Produkt a šablona endpointy

- PUT /api/product/save – uloží produkt z e-shopu do lokální DB. JSON: productData, eshopId, save (bool), newsletterObjectId.

- GET /api/product/{id} – detail lokálně uloženého produktu (id, productId, name, dphPrice, stock, img, url, eshopId).
- GET /api/template/{id} – detail šablony vč. stylesheet.
- GET /api/template/{id}/objects – šablonové objekty s HTML.
- POST /api/template/create-object – vytvoření nového šablonového objektu (vyžaduje roli Admin).

4.7 Popis aplikace z pohledu uživatele

Aplikace je určena pro marketingové pracovníky firem provozujících e-shopy. Po registraci a přihlášení se uživatel ocitne na stránce se seznamem svých newsletterů. Odtud může vytvořit nový newsletter volbou názvu, šablony a e-shopu, nebo otevřít existující k editaci.

V editoru newsletteru jsou zobrazeny jednotlivé bloky (objekty) zkopírované ze šablony. Uživatel může přímo editovat texty, měnit obrázky, přidávat odkazy a vkládat produkty z připojeného e-shopu. Produkty se vyhledají podle ID a automaticky se doplní název (včetně supplementu Nadpis_H1 pokud existuje), cena s DPH a obrázek. Bloky je možné přesouvat, přidávat nové z šablony a mazat. Veškeré změny se uloží tlačítkem Uložit.

Před odesláním je k dispozici náhled, který zobrazuje newsletter přesně tak, jak bude vypadat v e-mailovém klientu. Výsledek lze exportovat do HTML souboru s inline CSS styly.

Správa e-shopů umožňuje připojit libovolný e-shop postavený na Nette frameworku s nainstalovaným balíčkem eshop-api-exposer. Při připojení se automaticky vygeneruje API klíč a secret, který se zadá do konfigurace e-shopu. Secret je zobrazen pouze jednou a nelze ho později získat.

Administrátor má navíc možnost spravovat šablony – vytvářet nové šablonové objekty s HTML strukturou a CSS styly, které pak běžní uživatelé využívají při tvorbě newsletterů.

5 Výsledky

Výsledkem práce je funkční backendové rozhraní pro aplikaci pro sestavování newsletterů s napojením na e-shopy. Implementovány byly následující funkčnosti:

- Systém pro registraci a přihlášení uživatelů s rozlišením rolí Admin a User
- Správa e-shopů s automatickým generováním API klíčů a HMAC-SHA256 autentizací
- REST API pro tvorbu a správu newsletterů ze šablon
- Integrace produktových dat z externích e-shopů včetně supplementů (Nadpis_H1)
- Lokální ukládání produktů s automatickým čištěním orphan záznamů
- Zabezpečení všech API endpointů validací vlastnictví dat (IDOR ochrana)
- Šifrování API secretů pomocí libsodium

Součástí řešení je Composer balíček eshop-api-exposer, který je nezávislý na konkrétním e-shopu a je možné ho nasadit na libovolný projekt postavený na Nette frameworku. Backend je navržen s ohledem na budoucí rozšíření do modelu SaaS – data uživatelů jsou oddělena validací vlastnictví, systém podporuje více e-shopů na jednoho uživatele a lokalizaci rozhraní.

6 Diskuse

Při vývoji systému bylo nutné čelit několika technickým výzvám, které ovlivnily výslednou podobu aplikace.

Zásadní otázkou byla volba způsobu komunikace s e-shopem. Zvažovány byly tři přístupy: přímý přístup do databáze e-shopu, XML produktový feed a REST API. Přímý přístup do databáze byl zamítnut z bezpečnostních důvodů – vyžadoval by sdílení databázových přístupových údajů s externí aplikací. XML feed by neumožňoval dotazování na konkrétní produkty v reálném čase a vyžadoval by pravidelné generování. REST API s HMAC-SHA256 autentizací bylo zvoleno jako nejlepší kompromis mezi bezpečností, flexibilitou a jednoduchostí nasazení.

Další výzvou bylo řešení autoCanonicalize chování Nette frameworku. Ve výchozím nastavení Nette přesměrovává požadavky 302 redirectem za účelem zahrnutí persistentních parametrů (typicky locale). U PUT a POST požadavků s JSON tělem to vedlo ke ztrátě těla požadavku. Problém byl vyřešen selektivním vypnutím

autoCanonicalize pro API akce v metodě startup() každého presenteru s využitím strtolower() pro porovnání názvů akcí (Nette interně používá PascalCase).

Třetím problémem bylo mapování vlastností entit v Nextras ORM. Automatická konvence převodu camelCase na snake_case v některých případech selhávala – například u vlastnosti eshopId entity Product, kde ORM nevytvořilo správné mapování na sloupec eshop_id. Řešením bylo přidání explicitních mapování v Mapper třídě pomocí metody setMapping(). Tato zkušenost potvrdila výhodu firemní znalosti daného ORM.

Při porovnání zvoleného řešení s existujícími nástroji pro tvorbu newsletterů (Mailchimp, SmartEmailing) je hlavní výhodou přímá integrace s databází e-shopu včetně supplementů, což komerční řešení nenabízí. Nevýhodou je aktuální omezení na Nette-based e-shopy, které je však možné v budoucnu řešit implementací dalších DataProvider tříd pro jiné platformy.

Závěr

Cílem této maturitní práce bylo navrhnout a implementovat backendové rozhraní pro aplikaci pro sestavování newsletterů s napojením na e-shopy. Tento cíl byl splněn – vznikl funkční systém postavený na Nette frameworku, který umožňuje kompletní workflow od registrace uživatele přes připojení e-shopu až po sestavení a export newsletteru.

Klíčovým přínosem práce je návrh zabezpečené API komunikace pomocí HMAC-SHA256, která umožňuje bezpečný přenos produktových dat mezi e-shopem a sestavovačem. Dále pak modulární architektura založená na Composer balíčku eshop-api-exposer, který je možné nasadit na libovolný Nette-based e-shop bez nutnosti zásahů do kódu e-shopu.

Možná rozšíření projektu zahrnují: implementaci plánovaného odesílání newsletterů přímo z aplikace, rozšíření balíčku o další formáty integrace (XML feed, CSV import), přidání statistik odesílání, implementaci A/B testování obsahu a vytvoření DataProvider implementací pro další e-shopové platformy (WooCommerce, Shopify).

Použitá literatura

[1] Litmus. The ROI of Email Marketing [online]. 2005-2026 [cit. 2026-03-20]. Dostupné z: https://www.litmus.com/blog/infographic-the-roi-of-email-marketing

[2] Nette Foundation. Nette Framework Documentation [online]. 2008, 2026 [cit. 2026-03-20]. Dostupné z: https://doc.nette.org/cs/

[3] Nextras. Nextras ORM Documentation [online]. 2013, 2026 [cit. 2026-03-20]. Dostupné z: https://nextras.org/orm/docs/5.0/

[4] IETF. RFC 2104: HMAC: Keyed-Hashing for Message Authentication [online]. 1997 [cit. 2026-03-20]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc2104

[5] Ing. Radim Dolák, Ph.D. Informační systémy ve veřejné správě [online]. 2021 [cit. 2026-03-20]. Dostupné z: https://is.slu.cz/el/opf/leto2021/INMBPISV/um/prednasky/ISVS-01-Uvod_do_teorie_informacnich_systemu.pdf

[6] Mozilla Developer Network. HTTP request methods [online]. 1998–2026 [cit. 2026-03-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

[7] PHP Group. Sodium – PHP Manual [online]. 2001-2026 [cit. 2026-03-20]. Dostupné z: https://www.php.net/manual/en/book.sodium.php

[8] GuzzleHttp. Guzzle Documentation [online]. 2015 [cit. 2026-03-20]. Dostupné z: https://docs.guzzlephp.org/en/stable/

Seznam obrázků