Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

# BAKALÁŘSKÁ PRÁCE

Webová aplikace pro zasílání pravidelných notifikací

2023

Pavel Kryl

Vedoucí práce: Mgr. Jiří Valůšek

Studijní obor: Aplikovaná informatika, kombinovaná forma

**Bibliografické údaje**

| | |
|---|---|
| Autor: | Pavel Kryl |
| Název práce: | Webová aplikace pro zasílání pravidelných notifikací |
| Typ práce: | bakalářská práce |
| Pracoviště: | Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci |
| Rok obhajoby: | 2023 |
| Studijní obor: | Aplikovaná informatika, kombinovaná forma |
| Vedoucí práce: | Mgr. Jiří Valůšek |
| Počet stran: | 52 |
| Přílohy: | elektronická data v úložišti katedry informatiky |
| Jazyk práce: | český |

**Bibliographic info**

| | |
|---|---|
| Author: | Pavel Kryl |
| Title: | Web application for sending regular notification |
| Thesis type: | bachelor thesis |
| Department: | Department of Computer Science, Faculty of Science, Palacký University Olomouc |
| Year of defense: | 2023 |
| Study field: | Applied Computer Science, combined form |
| Supervisor: | Mgr. Jiří Valůšek |
| Page count: | 52 |
| Supplements: | electronic data in the storage of department of computer science |
| Thesis language: | Czech |

## Anotace

*Tato práce se zaměřuje na návrh notifikačního systému, který je navržen tak, aby pravidelně zasílal uživatelům notifikace obohacené o data získaná z API různých služeb. Systém je především určen pro potřeby IT profesionálů, kteří často pracují s API a potřebují pravidelně získávat informace z dat různých zdrojů. Systém je navíc navržen tak, aby umožnil uživatelům přizpůsobit si čas a způsob zaslání notifikace podle svých potřeb a preferencí.*

## Synopsis

*This thesis focuses on the design of a notification system, which is designed to regularly send users notifications enriched with data obtained from the APIs of various services. The system is primarily intended for the needs of IT professionals who often work with APIs and need to regularly obtain information from data from various sources. Additionally, the system is designed to allow users to customize the time and method of sending notifications according to their needs and preferences.*



**Klíčová slova:** notifikace; API; autorizace; autentizace

**Keywords:** notification; API; authorization; authentication

Děkuji vedoucímu práce Mgr. Jiřímu Valůškovi za odborné vedení, trpělivost, cenné rady a připomínky, které mi během zpracovávání této bakalářské práce poskytl. Především děkuji rodině, přátelům, kolegům a hlavně své přítelkyni za podporu při studiu.

*Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.*

datum odevzdání práce podpis autora

# Obsah

# Seznam obrázků



# Seznam tabulek

# 1 Úvod

V současné éře digitalizace se setkáváme s neustálým nárůstem dat a informací, které nám jsou k dispozici. Tato data jsou často rozprostřena napříč různými zdroji a to může způsobit značnou časovou náročnost při hledání potřebných informací.

Během mé praxe jsem se opakovaně setkával s potřebou jak zákazníků, tak vývojářů po notifikačním systému. Tyto požadavky byly klíčovou motivací pro vývoj takového řešení. Přesto jsem se při návrhu tohoto systému rozhodl prioritně zaměřit na potřeby IT profesionálů. Důvodem je, že právě oni jsou často ti, kdo pracují s API a zpracovávají data z různých zdrojů. Pro běžné uživatele by taková práce mohla být příliš komplikovaná, a proto jsem usoudil, že je lepší je od těchto technických detailů odstínit. S tímto závěrem jsem se snažil vytvořit systém, který by byl technicky flexibilní a umožňoval IT profesionálům efektivně vytvářet notifikace obohacené o data získaná z různých API.

Výslednou myšlenkou řešení bylo vytvořit webovou aplikaci, která bude pravidelně zasílat vybraným subjektům notifikace na základě informací získaných z API různých služeb. Tento softwarový systém je navržen tak, aby umožnil uživatelům přidat přístup k API různých služeb, spravovat dotazy na API, definovat zpracování a reprezentaci výsledných notifikací. Navíc systém poskytuje možnost nastavit čas a způsob zaslání notifikace, což umožní uživatelům přizpůsobit si systém podle svých potřeb a preferencí.

Součástí řešení budou také předdefinované šablony, které budou demonstrovat použití systému. Tyto šablony nejen ukážou, jak lze systém využít, ale také poskytnou uživatelům výchozí bod pro vytváření vlastních konfigurací.

# 2 Webová API

API neboli aplikační programové rozhraní (Application Programming Interface) je soubor definovaných pravidel a protokolů, které umožňují komunikaci a interakci mezi softwarovými aplikacemi. API slouží jako rozhraní, které umožňuje jedné aplikaci využívat funkcí a služeb jiné aplikace bez nutnosti znát detaily její interní implementace.

V této bakalářské práci se zaměříme na webová API. Webová API poskytují rozhraní pro komunikaci a interakci mezi webovými aplikacemi a službami (podrobnější informace o webových API lze například najít na stránkách [1, 2]).

Tato API jsou často založena na HTTP protokolu a umožňují webovým službám poskytovat data a funkce prostřednictvím síťového rozhraní. Existuje několik různých typů webových API, ale v této práci se zaměříme na typ webových API zvaný REST (Representational State Transfer).

## 2.1 REST API

REST API je založeno na architektuře klient-server a ke komunikaci využívá standardní HTTP metody [3]. Co ho však odlišuje od jiných typů webových API, jsou jeho klíčové principy [4, 3], které jsou popsány níže:

- **Bezstavovost (Statelessness)** – Tento princip znamená, že každý požadavek od klienta na server je samostatný a musí obsahovat všechny informace potřebné k jeho zpracování. Server nesmí uchovávat žádné informace o stavu mezi jednotlivými požadavky, což znamená, že neexistuje žádná „relace“ (session) mezi klientem a serverem mimo jednotlivé požadavky.

- **Využití vyrovnávací paměti (Cacheability)** – Odpovědi na požadavky mohou být označeny jako vhodné pro uložení do vyrovnávací paměti nebo nevhodné. Pokud je odpověď vhodná pro uložení do vyrovnávací paměti, klient ji může uložit a využít pro opakované dotazy. Tím se může výrazně zlepšit rychlost a efektivita komunikace mezi klientem a serverem.

- **Jednotné rozhraní (Uniform Interface)** – REST API poskytuje konzistentní a standardizované rozhraní pro komunikaci mezi klientem a serverem. To zjednodušuje interakci a umožňuje klientům snadno předvídat, jaké požadavky mohou odeslat a jaké odpovědi mohou očekávat.

- **Vrstvený systém (Layered System)** – Tento princip umožňuje oddělení různých částí systému do samostatných vrstev. Klient nemusí mít povědomí o tom, jestli komunikuje přímo se serverem nebo s nějakou meziúrovní. Díky tomu je možné vytvářet složité systémy, které jsou pro klienta stále jednoduché k použití.

### 2.1.1 CRUD operace

REST API využívá běžné HTTP metody, které odpovídají základním operacím, které můžeme provádět s daty. Tyto operace se často označují jako CRUD[1] [5].

V kontextu REST API tyto operace odpovídají následujícím standardním HTTP metodám pro manipulaci s daty:

- **POST (Create)** – Odeslání nových dat na server. Klient může například poslat POST požadavek na server pro vytvoření nového uživatele.
- **GET (Read)** – Získání dat ze serveru. Klient může například poslat GET požadavek na server pro získání seznamu všech uživatelů.
- **PUT (Update)** – Aktualizace existujících dat na serveru. Klient může například poslat PUT požadavek na server pro aktualizaci informací o existujícím uživateli.
- **DELETE (Delete)** – Odstranění dat ze serveru. Klient může například poslat DELETE požadavek na server pro odstranění existujícího uživatele.

Každý z těchto požadavků vrátí HTTP stavový kód, který informuje klienta o výsledku požadavku.

### 2.1.2 HTTP stavové kódy

Když klient posílá požadavek na server prostřednictvím REST API, server odpovídá s HTTP stavovým kódem (viz [6]). Tyto kódy informují klienta o výsledku jeho požadavku. Některé běžně používané HTTP stavové kódy v REST API jsou:

- **200 OK** – Požadavek byl úspěšně zpracován.
- **201 Created** – Požadavek byl úspěšně zpracován a byla vytvořena nová data.
- **400 Bad Request** – Server nerozumí požadavku, protože je nesprávně formulován.
- **401 Unauthorized** – Klient nemá oprávnění pro provedení požadované operace.
- **404 Not Found** – Server nemohl najít požadovaný zdroj.
- **500 Internal Server Error** – Na serveru došlo k chybě při zpracování požadavku.

Podrobný přehled a vysvětlení všech HTTP stavových kódů lze nalézt na webové stránce [6].

[1] zkratka pro vytvoření (Create), čtení (Read), aktualizaci (Update) a smazání (Delete)

## 2.2 Formáty odpovědí API

API může vracet data v různých formátech, záleží na konkrétním API a jeho nastavení. Nejběžnější formáty, které se používají k přenosu dat z webového serveru na klienta, jsou CSV, XML a JSON. Každý z těchto formátů má své výhody a nevýhody. V další části kapitoly si jednotlivé formáty popíšeme.

### 2.2.1 CSV (Comma Separated Values)

CSV je formát, který je v podstatě seznam prvků oddělených čárkami. Zde si uvedeme příklad, jak můžou vypadat data ve formátu CSV:

```
Diana,Petr,Karel,Eva
```

Tento formát je nejkompaktnější ze všech tří formátů. Obecně platí, že CSV formáty jsou zhruba poloviční velikosti oproti formátům XML a JSON (podle článku [7]). Toto je hlavní výhoda CSV formátu. Nicméně CSV formát má také své nevýhody. Jde o nejméně univerzální z probíraných tří formátů, protože vyžaduje vytvoření vlastního parseru[2] pro převod CSV dat do nativní datové struktury. Pokud se datová struktura změní, musí se také upravit nebo dokonce přepracovat parser, což je spojeno s dodatečnými náklady.

Další nevýhodou je, že formát CSV nepodporuje datové hierarchie, takže v případě, kdy potřebujeme zasílat data obsahující více úrovní detailů (například objekt s vlastnostmi), musíme vytvořit složitější parser.

### 2.2.2 XML (Extensible Markup Language)

XML je formát, který byl vytvořen pro lepší reprezentaci datových formátů s hierarchickou strukturou. XML plně podporuje hierarchické datové struktury a je velmi vhodný pro přenos dat se složitou hierarchickou strukturou. Navíc je XML formát velmi čitelný pro lidi. Zde si uvedeme příklad, jak můžou vypadat data ve formátu XML:

```
<osoby>
  <osoba>
    <jmeno>Diana</jmeno>
    <vek>26</vek>
  </osoba>
  <osoba>
    <jmeno>Petr</jmeno>
    <vek>32</vek>
  </osoba>
</osoby>
```

[2]parser – nástroj nebo program, který převádí vstupní data (například text) do vhodné struktury pro další zpracování

Jelikož byl XML první standardní hierarchický datový formát, většina API má vestavěné funkce pro automatické převádění dat ve formátu XML do nativních datových struktur, jako jsou objekty. Velkou nevýhodou XML formátu je jeho velikost, XML formát je zhruba třikrát větší než CSV (podle článku [7]).

### 2.2.3 JSON (JavaScript Object Notation)

Formát JSON byl vytvořen jako alternativa k XML. JSON reprezentuje hierarchická data s použitím čárek, složených závorek a hranatých závorek. Tento datový formát podporuje hierarchická data, zatímco je menší než XML. Zde si uvedeme příklad, jak můžou vypadat data ve formátu JSON:

```
{
  "osoby": [
    {
      "jmeno": "Diana",
      "vek": 26
    },
    {
      "jmeno": "Petr",
      "vek": 32
    }
  ]
}
```

Jak název napovídá, byl také vytvořen pro snadnější parsování dat do nativních JavaScript objektů, což je velmi užitečné pro webové aplikace. JSON je nejlepší z obou světů s ohledem na CSV a XML. Je jednoduchý a kompaktní jako CSV a navíc podporuje hierarchická data jako XML. Na rozdíl od XML jsou formáty JSON pouze zhruba dvakrát větší než CSV formáty (podle článku [7]). JSON se stal velmi rozšířeným datovým formátem.

Přestože byl JSON formát původně vytvořen pro JavaScript, jeho využití se rozšířilo do mnoha dalších programovacích jazyků, jako jsou Python, Java, C#, Ruby a další.

Lze obecně říci (podle článku [7]), že v současné době pro většinu API webových služeb, které vyžadují rychlý a efektivní přenos dat, je JSON považován za nejvhodnější formát pro výměnu dat. Je kompaktní a univerzální. CSV by měl být použit v případech, kdy je potřeba odesílat velmi velké objemy dat.

## 2.3 Příklad použití API

Představme si, že máme webovou aplikaci, která potřebuje zobrazit aktuální počasí. Místo toho, aby naše aplikace sama shromažďovala, ukládala a zpracovávala data o počasí, může využít API poskytované třetí stranou, například webovou službu pro předpověď počasí, jako je OpenWeatherMap[3] [8].

Naše aplikace by poslala HTTP požadavek na API webové služby pro předpověď počasí, specifikující informace, které potřebuje (například aktuální teplotu a stav oblohy pro určitou lokalitu).

API webové služby pro předpověď počasí by zpracovalo náš požadavek a vrátilo odpověď ve formátu JSON, který naše aplikace může zpracovat a zobrazit uživateli. Tato odpověď by mohla obsahovat informace, jako je teplota, vlhkost, rychlost větru, popis počasí atd.

Tímto způsobem API umožňuje naší aplikaci využívat služeb a dat, které sama nevlastní ani nezpracovává. Díky API tak můžeme poskytnout uživatelům naší aplikace přesné a aktuální informace o počasí bez nutnosti vlastního shromažďování a zpracování těchto dat.

## 2.4 Autentizace a autorizace

Autentizace a autorizace jsou dva základní pojmy v oblasti zabezpečení API [9], které hrají klíčovou roli v ochraně dat a služeb.

**Autentizace** se zabývá ověřováním identity uživatele nebo aplikace, která se pokouší získat přístup k API. Cílem autentizace je ověřit, zda osoba či aplikace je skutečně tím, za koho se vydává, což je zásadní krok k zabránění neoprávněného přístupu. Toto ověření identity žadatele o přístup k API se zajišťuje za pomocí různých metod a to například uživatelským jménem a heslem, tokeny nebo za pomocí digitálních certifikátů.

**Autorizace** je proces, který určuje, jaké operace a zdroje jsou přístupné pro daného uživatele či pro danou aplikaci. Autorizace často využívá role a oprávnění, které jsou jednotlivým subjektům přiřazovány.

V následující části kapitoly uvedeme několik příkladů druhů autentizací:

- základní autentizace (Basic Authentication),
- tokenová autentizace (Bearer Authentication[4]),
- JWT (JSON Web Token),
- OAuth,
- OAuth 2.0.

[3] https://openweathermap.org/api
[4] případně také Token Authentication

### 2.4.1 Základní autentizace

Základní autentizace je jedním z nejjednodušších způsobů autentizace, kde se uživatel nebo aplikace přihlašuje pomocí svého uživatelského jména a hesla jako důkaz o své identitě. Tyto údaje jsou přenášeny v rámci HTTP hlavičky, často jsou zakódovány pomocí kódování Base64.

Důležitá informace je, že tato metoda autentizace neposkytuje žádné šifrování a proto je důležité ji používat v kombinaci se zabezpečeným spojením, jako je například HTTPS [10].

### 2.4.2 Tokenová autentizace

Tokenová autentizace [11] je metoda autentizace, kde se uživatel nebo aplikace přihlašuje za pomocí bezpečnostního tokenu. Token je zpravidla vygenerován serverem po úspěšném přihlášení uživatele nebo aplikace a tento token se následně používá pro další komunikaci. Token slouží jako důkaz identity, bývá časově omezen a také může být platný jen pro určité operace. Tokeny jsou obvykle přenášeny v HTTP hlavičce.

### 2.4.3 JWT

Jedná se o standard pro vytváření bezpečných tokenů, které slouží jak k autentizaci, tak i k autorizaci. Struktura JWT se skládá ze tří částí a to z hlavičky („Header“), dat („Payload“) a podpisu („Signature“). Každá část je zakódována ve formátu Base64Url a oddělena tečkami. Podrobnější informace o JWT lze najít v RFC[5] dokumentaci s číslem 7519 [12].

### 2.4.4 OAuth

Jedná se o protokol pro autentizaci a autorizaci službám třetích stran, který umožňuje uživatelům sdílet své informace nebo umožnit omezený přístup službě třetí strany k jejich účtu na jiné službě. Například uživatel může povolit aplikaci přístup k jeho účtu na Google nebo Facebook bez nutnosti sdílet své uživatelské jméno a heslo. OAuth je definován v RFC 5849 [13], který je nyní zastaralý a odkazuje na novější verzi protokolu, konkrétně na OAuth 2.0, který přináší řadu vylepšení.

### 2.4.5 OAuth 2.0

OAuth 2.0 je moderní verze protokolu OAuth. Myšlenka OAuth 2.0 je založena na poskytnutí tokenu, který může aplikace využít k přístupu k zabezpečeným zdrojům uživatele na cílové službě. Tento token je vydán autorizačním serverem na základě souhlasu uživatele a může být omezen na konkrétní operace, které aplikace může provádět.

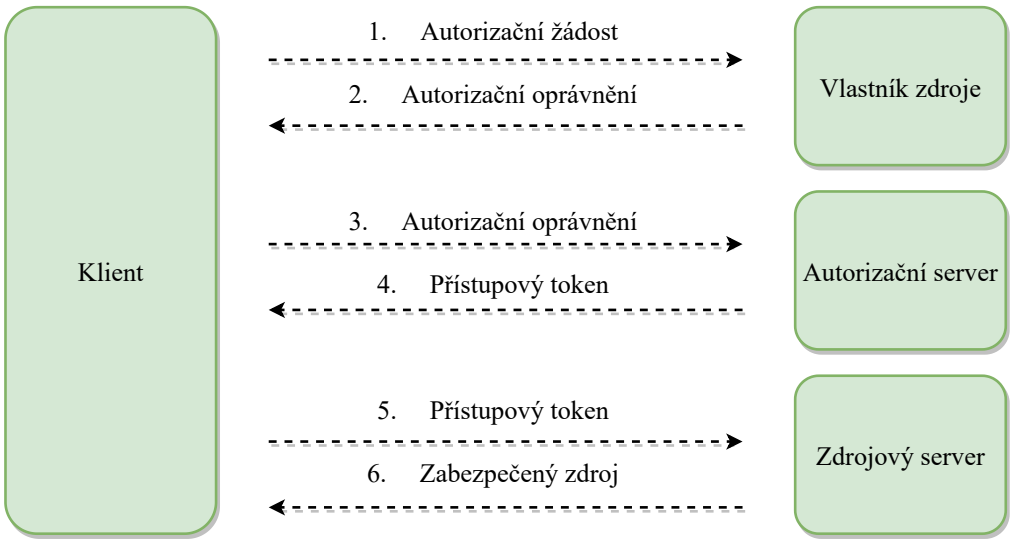
[5] Request for Comments

### Abstraktní tok protokolu

Zde si vysvětlíme abstraktní tok protokolu OAuth 2.0. Pojem „abstraktní tok protokolu“ je série kroků, které popisují, jak se data přesouvají a jaké procesy se v rámci daného protokolu odehrávají.

OAuth 2.0 definuje čtyři role, které figurují v abstraktním toku protokolu a které mezi sebou interagují (viz [14] kapitola 1.1):

- **Vlastník zdroje (Resource Owner)** – Tato entita má možnost povolit přístup k určitému zabezpečenému zdroji. Pokud je tato entita člověk, nazýváme ho koncovým uživatelem.
- **Zdrojový server (Resource Server)** – Tento server, který uchovává zabezpečené zdroje, je schopen přijímat a odpovídat na požadavky na tyto zdroje. Tyto požadavky jsou chráněny za pomocí přístupových tokenů[6].
- **Klient (Client)** – Jedná se o aplikaci, která chce přistupovat k zabezpečeným zdrojům za vlastníka těchto zdrojů a má k tomu jeho svolení. Termín „klient“ nespecifikuje, jak je aplikace realizována – může běžet na serveru, na osobním počítači nebo na jiném zařízení.
- **Autorizační server (Authorization Server)** – Jedná se o server, který po úspěšném ověření identity vlastníka zdroje a poté, co získá jeho svolení, vydává přístupové tokeny aplikaci (klientovi).




Obrázek 1: Abstraktní tok protokolu OAuth 2.0.

[6] Access tokens

Teď můžeme popsat, jak je protokolový tok definován v RFC 6749 (viz [14] kapitola 1.2):

1. Aplikace (klient) žádá o svolení k přístupu ke zdroji od jeho vlastníka. Tato žádost může být adresována přímo vlastníkovi nebo, což je častější, přes autorizační server, který zprostředkovává tento proces.

2. Aplikace (klient) dostane autorizační oprávnění[7], což je jakési „povolení“ od vlastníka zdroje. Toto povolení může být získáno čtyřmi způsoby, které jsou definovány v této specifikaci (RFC 6749 [14]) nebo pomocí rozšířeného způsobu. Jaký způsob bude použit, záleží na tom, jakým způsobem aplikace žádá o autorizaci a jaké způsoby žádostí o autorizaci autorizační server podporuje.

3. Aplikace (klient) se obrátí na autorizační server s žádostí o přístupový token a prokáže se již získaným autorizačním oprávněním.

4. Autorizační server ověří totožnost aplikace (klienta) a zkontroluje platnost autorizačního oprávnění. Pokud je vše v pořádku, server vrátí přístupový token jako odpověď na žádost aplikace (klienta).

5. Aplikace (klient) se obrátí na zdrojový server s požadavkem na zabezpečený zdroj a prokáže svou totožnost za pomocí přístupového tokenu.

6. Zdrojový server zkontroluje platnost přístupového tokenu a pokud je token platný, zpracuje požadavek aplikace (klienta).

**Způsob udělení autorizačního oprávnění**[8]
Způsob udělení autorizačního oprávnění v OAuth 2.0 definuje metody, jakým způsobem aplikace (klient) získává autorizační oprávnění, které je poté využito k získání přístupového tokenu. Existují čtyři standardní způsoby udělení autorizačního oprávnění definované v OAuth 2.0 specifikaci (viz [14] kapitola 4):

- **Autorizační kód (Authorization Code)** – Tento způsob je vhodný pro webové aplikace, které dokážou bezpečně skrýt své přihlašovací údaje. Uživatel je přesměrován na stránku (autorizační server), kde povolí přístup aplikaci k jeho datům. Po povolení je uživatel přesměrován zpět na aplikaci spolu s autorizačním kódem, který aplikace využije k získání přístupového tokenu.

- **Implicitní (Implicit)** – Tento způsob je určen pro aplikace, které nemohou bezpečně uchovat své přihlašovací údaje, jako jsou aplikace běžící přímo v prohlížeči nebo mobilní aplikace. Uživatel je přesměrován na stránku (autorizační server), kde povolí přístup aplikaci k jeho datům.

[7] Authorization grant
[8] také známé jako grant types

Po povolení je uživatel přesměrován zpět do aplikace spolu s přístupovým tokenem. Implicitní způsob udělení autorizačního oprávnění je v dnešní době zastaralý a místo něho je vhodné používat PKCE (Proof Key for Code Exchange) [15]. PKCE byl původně navržen pro bezpečné použití OAuth 2.0 pro nativní aplikace[9], ale nyní se doporučuje pro všechny typy OAuth klientů včetně webových aplikací.

- **Přihlašovací údaje vlastníka zdroje (Resource Owner Password Credentials)**[10] – Tento způsob je aktuálně již nepoužívaný v OAuth 2.1, dokonce již není zahrnut. Způsob se používal v situaci, kdy uživatel plně důvěřoval aplikaci, když byla aplikace nainstalována na jeho zařízení. Uživatel předal své přihlašovací údaje aplikaci, která následně tyto údaje použila k získání přístupového tokenu. Tento typ oprávnění je méně bezpečný, protože vyžaduje, aby uživatel poskytl aplikaci své přihlašovací údaje.

- **Přihlašovací údaje klienta (Client Credentials)** – Tento způsob je vhodný, když aplikace potřebuje přístup k vlastním zdrojům na zdrojovém serveru. Aplikace využije své přihlašovací údaje k získání přístupového tokenu.

**Přístupové tokeny** jsou klíče, které umožňují přístup k zabezpečeným zdrojům. Jsou to řetězce, které klientovi udělují určitou autorizaci. Tyto tokeny určují specifický rozsah a dobu přístupu, kterou vlastník zdroje schválil (viz [14] kapitola 1.4). Přístupové tokeny obvykle po určité době vyprší a poté je třeba zažádat o nový.

### Obnovení tokenu[11]

V rámci OAuth 2.0 protokolu existuje koncept obnovování tokenu, který umožňuje aplikacím obnovit přístupový token bez nutnosti opětovného žádání o autorizaci od uživatele.

Obnovovací token je vydán spolu s přístupovým tokenem a může být použit k získání nového přístupového tokenu, když stávající token vyprší nebo je z nějakého důvodu zrušen. Tento proces je velmi užitečný u aplikací, které potřebují udržovat dlouhodobý přístup k zabezpečeným zdrojům (viz [14] kapitola 1.5).

Důležitá poznámka k bezpečnosti: Při použití obnovovacího tokenu je důležité zajistit jeho bezpečnost, protože jakákoliv aplikace, která má tento token, může získat přístupový token a tím i přístup k zabezpečeným zdrojům. To znamená, že by obnovovací token měl být uchováván bezpečně stejně jako přihlašovací údaje uživatele.

[9] Nativní aplikace jsou aplikace, které jsou nainstalovány přímo na zařízení, jako jsou mobilní telefony nebo počítače a jsou specificky navrženy a optimalizovány pro tato zařízení.

[10] také známé jako Password grant type

[11] Refresh Token

# 3 Dokumentace

Tato kapitola se zaměřuje na technickou dokumentaci notifikačního systému, který byl vyvinut v rámci této bakalářské práce. Dokumentace je určena pro vývojáře a uživatele, kteří se chtějí dozvědět detailněji o tomto systému.

Cílem této dokumentace je poskytnout detailnější přehled o technologiích použitých při vývoji systému, struktuře systému, zabezpečení, testování a na závěr se podíváme na různé způsoby zasílání notifikací, které systém podporuje.

## 3.1 Použité technologie

Při vývoji notifikačního systému bylo využito několik technologií, které byly vybrány na základě mých osobních preferencí a schopností a také na základě širokého použití v komunitě vývojářů[12].

### 3.1.1 .NET 6

Hlavní technologií, na které je notifikační systém postaven, je platforma .NET 6 [16], což je nejnovější verze open-source[13] vývojové platformy od společnosti Microsoft (v době vývoje notifikačního systému). .NET 6 byl zvolen pro svou univerzálnost, podporu moderních aplikací a vysokou úroveň abstrakce, která zjednodušuje vývoj.

### 3.1.2 C#

Jazyk C# [17] byl zvolen jako hlavní programovací jazyk pro tento projekt. C# je moderní, objektově orientovaný jazyk, který byl navržen tak, aby byl srozumitelný a efektivní. C# má silnou podporu v .NET platformě a je široce používán pro vývoj webových aplikací.

### 3.1.3 MSSQL

Pro správu a ukládání dat byla zvolena databáze Microsoft SQL Server (MSSQL). MSSQL je robustní, výkonný a spolehlivý systém pro správu databází.

### 3.1.4 JavaScript, HTML a CSS

Pro vývoj frontend klientské části systému byly použity technologie JavaScript [22, 23], HTML [18, 19] a CSS [20, 21]. Tyto technologie jsou základem pro vývoj webových aplikací a jsou podporovány všemi moderními webovými prohlížeči.

[12] Což znamená větší množství dostupných informací a podpory.

[13] S otevřeným zdrojovým kódem. Software, jehož zdrojový kód je veřejně dostupný a může být volně používán, upravován a šířen komunitou vývojářů.

## 3.2 Struktura systému

Klíčovým aspektem jakéhokoliv softwarového řešení je jeho struktura. Struktura systému nejen definuje, jak jsou jednotlivé části systému organizovány a jak spolu interagují, ale také ovlivňuje jeho udržovatelnost a rozšiřitelnost.

V této podkapitole se podíváme na strukturu notifikačního systému, který je rozdělen do šesti projektů. Každý z těchto projektů vykonává svoji specifickou roli v rámci celého systému.

Tyto projekty jsou:

- `ReportApp.ApiSender`,
- `ReportApp.Client`,
- `ReportApp.Data`,
- `ReportApp.DbStructure`,
- `ReportApp.Dispatcher`,
- `ReportApp.Models`.

### 3.2.1 ReportApp.ApiSender

`ReportApp.ApiSender` je zodpovědný za komunikaci s externími API, zpracování získaných dat z externích API, upravování těla zprávy notifikace o získaná data a následné odesílání notifikací.

### 3.2.2 ReportApp.Client

`ReportApp.Client` je klientská část systému[14]. Umožňuje uživatelům spravovat svůj účet, spravovat kontaktní informace příjemců, spravovat API různých služeb, spravovat dotazy na tyto API. Dále tato část systému umožňuje uživatelům spravovat šablony notifikací, které slouží k definování zpracování a reprezentaci notifikace. Nakonec zde může uživatel vytvářet a upravovat notifikace podle svých preferencí a potřeb (kterou šablonu má notifikace použít, čas a způsob odesílání).

### 3.2.3 ReportApp.Data

Projekt `ReportApp.Data` aktuálně slouží jako databázový kontext pro Entity Framework. Myšlenkou je, že v budoucnu bude tento projekt zodpovědný za přímou interakci s databází (select, insert a update databáze). Půjde tedy o jakousi DAO (Data Access Object). Mohlo by to zlepšit udržovatelnost a rozšiřitelnost systému. Dále by to zefektivnilo jednotkové testování.

### 3.2.4 ReportApp.DbStructure

Projekt `ReportApp.DbStructure` obsahuje informace o databázové struktuře notifikačního systému a slouží k vytvoření databázové struktury v databázi pro notifikační systém, obsahuje tedy tabulky a inicializační skripty pro testování. Inicializační skripty není nutné spouštět pro správnou funkčnost systému.

### 3.2.5 ReportApp.Dispatcher

Projekt `ReportApp.Dispatcher` je služba, která běží neustále na pozadí a která v pravidelných intervalech prohledává databázi, kde vyhledává notifikace, které se mají odeslat. Notifikace k odesílání zasílá na `ReportApp.ApiSender` k dalšímu zpracování.

### 3.2.6 ReportApp.Models

Projekt `ReportApp.Models` jsou modely systému, které se používají ve třech projektech a to v: `ReportApp.ApiSender`, `ReportApp.Client` a `ReportApp.Dispatcher`. Modely jsou strukturovány podobně jako tabulky v databázové struktuře, takže sémantika jednotlivých tříd a vlastností je stejná tak, jak si to popíšeme v kapitole „Struktura databáze“ 3.3.

[14] Jinak formulováno: uživatelské rozhraní systému (UI).

## 3.3 Struktura databáze

Systém využívá MSSQL[15] jako relační databázový systém. V této kapitole si popíšeme databázovou strukturu, se kterou náš notifikační systém pracuje. Nejdříve si představíme přehled názvů tabulek včetně jejich stručného popisu. Poté se zaměříme na detailní představení jednotlivých tabulek.

Databázová struktura vypadá následovně:

- `Notification` – Tabulka 1 ukládá informace týkající se jednotlivých notifikací.
- `NotificationRecipient` – Vazební tabulka 2 ukládá vztahy mezi notifikacemi a vybranými příjemci.
- `PeriodicityWeekly` – Tabulka 3 obsahuje plánovací informace pro odesílání notifikací.
- `Recipient` – Tabulka 4 uchovává přidané subjekty, kterým se má notifikace zasílat.
- `Template` – Tabulka 5 uchovává šablony notifikace.
- `TemplateApi` – Tabulka 6 uchovává šablony API.
- `TemplateApiAuthApiKey` – Tabulka 7 je určena pro ukládání API klíčů, které se využívají pro autentizaci přístupu k API.
- `TemplateApiAuthOAuth2` – Tabulka 8 slouží k ukládání informací o OAuth 2.0 autentizaci.
- `TemplateApiRequest` – Tabulka 9 ukládá informace o konkrétních dotazech na API.
- `TemplateSelectedApiRequest` – Vazební tabulka 10 ukládá vybrané konkrétní dotazy na API v šabloně notifikace.
- `User` – Tabulka 11 ukládá informace o uživateli.

[15] Microsoft SQL Server

| Atribut | Datový typ | Popis |
|---|---|---|
| Id | INT | Identifikátor notifikace (primární klíč) |
| UserOwnerId | INT | Identifikátor uživatele vlastníka (cizí klíč) |
| TemplateId | INT | Identifikátor vybrané šablony (cizí klíč) |
| NotificationName | NVARCHAR(128) | Název notifikace |
| DispatchHour | INT | Hodina odeslání |
| DispatchMinute | INT | Minuta odeslání |
| Inserted | DATETIME2 | Datum a čas vložení záznamu |
| Updated | DATETIME2 | Datum a čas poslední aktualizace záznamu |

Tabulka 1: Tabulka Notification

| Atribut | Datový typ | Popis |
|---|---|---|
| Id | INT | Identifikátor vazby notifikace a příjemce (primární klíč) |
| RecipientId | INT | Identifikátor příjemce (cizí klíč) |
| NotificationId | INT | Identifikátor notifikace (cizí klíč) |
| Inserted | DATETIME2 | Datum a čas vložení záznamu |
| Updated | DATETIME2 | Datum a čas poslední aktualizace záznamu |

Tabulka 2: Tabulka NotificationRecipient

| Atribut | Datový typ | Popis |
|---|---|---|
| Id | INT | Identifikátor týdenní periodicity (primární klíč) |
| NotificationId | INT | Identifikátor notifikace (cizí klíč) |
| DayOfWeek | INT | Den v týdnu |
| Inserted | DATETIME2 | Datum a čas vložení záznamu |
| Updated | DATETIME2 | Datum a čas poslední aktualizace záznamu |

Tabulka 3: Tabulka PeriodicityWeekly

| Atribut | Datový typ | Popis |
|---|---|---|
| Id | INT | Identifikátor příjemce (primární klíč) |
| UserOwnerId | INT | Identifikátor uživatele vlastníka (cizí klíč) |
| RecipientName | NVARCHAR(64) | Jméno příjemce |
| Email | NVARCHAR(64) | Email příjemce |
| SlackWebhookUrl | NVARCHAR(256) | URL webhooku pro Slack |
| DiscordChannelId | DECIMAL(20, 0) | ID kanálu na Discordu |
| TelegramChatId | BIGINT | ID chatu na Telegramu |
| Inserted | DATETIME2 | Datum a čas vložení záznamu |
| Updated | DATETIME2 | Datum a čas poslední aktualizace záznamu |

Tabulka 4: Tabulka Recipient

| Atribut | Datový typ | Popis |
|---|---|---|
| Id | INT | Identifikátor šablony (primární klíč) |
| UserOwnerId | INT | Identifikátor uživatele vlastníka (cizí klíč) |
| TemplateName | NVARCHAR(128) | Název šablony |
| MessageSubject | NVARCHAR(78) | Předmět zprávy |
| MessageBody | NVARCHAR(MAX) | Tělo zprávy |
| Inserted | DATETIME2 | Datum a čas vložení záznamu |
| Updated | DATETIME2 | Datum a čas poslední aktualizace záznamu |

Tabulka 5: Tabulka Template

| Atribut | Datový typ | Popis |
|---|---|---|
| Id | INT | Identifikátor API šablony (primární klíč) |
| UserOwnerId | INT | Identifikátor vlastníka uživatele (cizí klíč) |
| ApiName | NVARCHAR(64) | Název API |
| TemplateApiAuthApiKeyId | INT | Identifikátor autentizace pomocí API klíče (cizí klíč) |
| TemplateApiAuthOAuth2Id | INT | Identifikátor autentizace pomocí OAuth2 (cizí klíč) |
| Inserted | DATETIME2 | Datum a čas vložení záznamu |
| Updated | DATETIME2 | Datum a čas poslední aktualizace záznamu |

Tabulka 6: Tabulka TemplateApi

| Atribut | Datový typ | Popis |
|---|---|---|
| Id | INT | Identifikátor autentizace pomocí API klíče (primární klíč) |
| ApiKey | BINARY(128) | API klíč |
| PublicKey | BINARY(16) | Veřejný klíč |
| Inserted | DATETIME2 | Datum a čas vložení záznamu |
| Updated | DATETIME2 | Datum a čas poslední aktualizace záznamu |

Tabulka 7: Tabulka TemplateApiAuthApiKey

| Atribut | Datový typ | Popis |
|---|---|---|
| Id | INT | Identifikátor autentizace pomocí OAuth2 (primární klíč) |
| AuthorizeUrl | NVARCHAR(2048) | URL autorizace |
| AccessTokenUrl | NVARCHAR(2048) | URL pro získání přístupového tokenu |
| GrantType | NVARCHAR(32) | Způsob udělení autorizačního oprávnění |
| ClientId | NVARCHAR(64) | ID klienta |
| ClientSecret | BINARY(64) | Tajný klíč klienta |
| TokenType | NVARCHAR(32) | Typ tokenu |
| AccessToken | BINARY(2048) | Přístupový token |
| RefreshToken | BINARY(512) | Token pro obnovení přístupu |
| PublicKey | BINARY(16) | Veřejný klíč |
| Inserted | DATETIME2 | Datum a čas vložení záznamu |
| Updated | DATETIME2 | Datum a čas poslední aktualizace záznamu |

Tabulka 8: Tabulka TemplateApiAuthOAuth2

| Atribut | Datový typ | Popis |
|---|---|---|
| Id | INT | Identifikátor požadavku na API šablonu (primární klíč) |
| TemplateApiId | INT | Identifikátor API šablony (cizí klíč) |
| ApiRequestName | NVARCHAR(64) | Název požadavku na API |
| UrlQuery | NVARCHAR(2048) | URL dotazu |
| Inserted | DATETIME2 | Datum a čas vložení záznamu |
| Updated | DATETIME2 | Datum a čas poslední aktualizace záznamu |

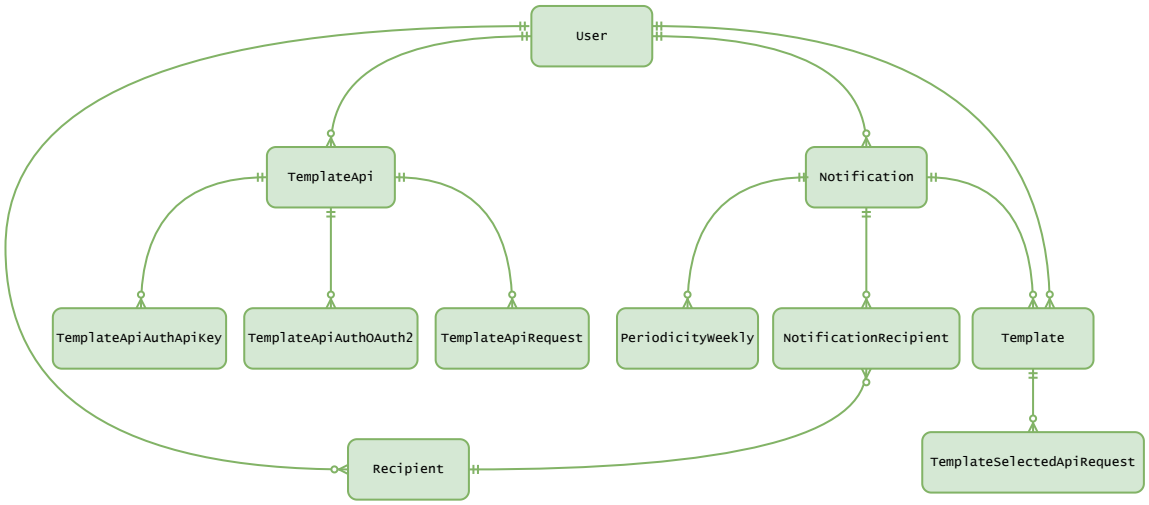
Tabulka 9: Tabulka TemplateApiRequest

| Atribut | Datový typ | Popis |
|---|---|---|
| Id | INT | Identifikátor vybraného požadavku na API šablonu (primární klíč) |
| TemplateId | INT | Identifikátor šablony (cizí klíč) |
| TemplateApiRequestId | INT | Identifikátor požadavku na API šablonu (cizí klíč) |
| Inserted | DATETIME2 | Datum a čas vložení záznamu |
| Updated | DATETIME2 | Datum a čas poslední aktualizace záznamu |

Tabulka 10: Tabulka TemplateSelectedApiRequest

| Atribut | Datový typ | Popis |
|---|---|---|
| Id | INT | Identifikátor uživatele (primární klíč) |
| UserEmail | NVARCHAR(128) | Email uživatele |
| Password | BINARY(32) | Hashované heslo |
| Salt | BINARY(16) | Salt pro heslo |
| Inserted | DATETIME2 | Datum a čas vložení záznamu |
| Updated | DATETIME2 | Datum a čas poslední aktualizace záznamu |

Tabulka 11: Tabulka User




Obrázek 2: ER diagram databázové struktury.

## 3.4 Zabezpečení

V našem notifikačním systému je velmi důležité zabezpečení citlivých údajů, jako jsou hesla, tokeny a client secret. K zajištění tohoto zabezpečení využíváme technologii šifrování známou jako AES[16] a funkci PBKDF2[17] pro hašování hesel.

### 3.4.1 Použití AES pro šifrování a dešifrování

AES [24, 25] je široce uznávaný a používaný standard pro šifrování dat. V naší aplikaci je klíč pro AES uložen v konfiguračním souboru a je načten do naší třídy `SecureHash` při její inicializaci. Tento klíč je pak použit pro šifrování a dešifrování dat.

### 3.4.2 Použití PBKDF2 pro hašování hesel

Pro hašování hesel používáme funkci PBKDF2 [26, 27]. Tato funkce vytváří bezpečný haš hesla. V případě, kdy je třeba ověřit heslo, použijeme stejnou funkci

[16] Advanced Encryption Standard
[17] Password-Based Key Derivation Function 2

PBKDF2 na heslo poskytnuté uživatelem a porovnáme výsledný haš s uloženým hašem.

### 3.4.3 Generování a použití soli

Sůl je náhodně generovaný bajtový řetězec, který je přidán k heslu před jeho hašováním [28]. Použití soli zvyšuje bezpečnost zahašovaných hesel tím, že ztěžuje útoky pomocí předem vypočítaných hašovacích tabulek[18]. V naší aplikaci je sůl generována nově pro každé heslo, pokud již neexistuje.

Za pomocí těchto popsaných technik jsou všechny citlivé údaje v naší aplikaci bezpečně šifrovány a dešifrovány.

## 3.5 Testování

Klientskou část systému jsem testoval sám manuálně. Toto testování zahrnovalo interakci s uživatelským rozhraním, čímž jsem ověřoval, zda jsou všechny funkce přístupné a fungují správně.

Testování backend části systému spočívalo hlavně v testování různých API služeb, aby se ověřilo, že systém správně komunikuje s těmito službami a správně zpracovává data. Toto testování zahrnovalo také ověřování, že systém správně reaguje na různé typy vstupů.

Hlavní výzvou při testování bylo velké množství různých API služeb, se kterými systém může komunikovat. Služby mají často své vlastní specifické vlastnosti, což komplikovalo proces testování. Byla snaha provést co nejvíce testů, aby se zajistilo, že systém je co nejvíce robustní a odolný vůči různým situacím.

## 3.6 Způsoby zasílaní notifikací

Řešení umožňuje zasílaní notifikací několika způsoby a to za pomocí emailu, Discord a Slack.

### 3.6.1 Email

SMTP[19] je internetový standard pro elektronickou poštu (email), který se používá pro spolehlivý a efektivní přenos emailových zpráv mezi serverem odesílatele a serverem příjemce. SMTP je definován v RFC 5321 [30]. Tato kapitola vysvětluje, jak se nastavuje SMTP pro odesílání emailů.

Pro posílání emailů za pomocí protokolu SMTP je potřeba mít správně nastaveno několik parametrů. Jsou to tyto parametry:

- adresa SMTP serveru,
- port,

[18] Tzv. Rainbow Table Attack [29]
[19] Simple Mail Transfer Protocol

- přihlašovací údaje.

**Port**
SMTP server běží na specifickém síťovém portu. Standardní port pro SMTP bývá port s číslem 25 (viz [30] sekce 4.5.4.2). Číslo portu může být odlišné, proto je lepší si číslo portu ověřit.

**Přihlašovací údaje:**

- uživatelské jméno,
- heslo.

Tyto údaje slouží k ověření identity odesílatele. Obvykle jsou přihlašovací údaje pro SMTP nastavení stejné jako při registraci či přihlášení do emailového klienta poskytovatele emailové služby, jako je například Seznam nebo Google. Pro přesnější informace o nastavení SMTP serveru je ale vždy lepší, když prostudujeme dokumentaci konkrétního poskytovatele emailové služby, jelikož se může stát, že se postup přihlášení liší.

**Konfigurace zasílání za pomocí emailu**
Konfigurace parametrů pro správnou funkcionalitu odesílání emailů za pomocí SMTP provádíme v souboru „appsettings.json“ v objektu s názvem „EmailSetting“ v projektu „ReportApp.ApiSender“.

```
"EmailSetting": {
    "EmailSender": "example@example.cz",
    "Password": "password",
    "Host": "smtp.example.cz",
    "Port": "25"
}
```

Sémantika parametrů objektu *EmailSetting*:

- *EmailSender* je uživatelské jméno a zároveň slouží jako email odesílatele, který se bude příjemci emailu zobrazovat jako odesílatel.
- *Password* je heslo.
- *Host* je adresa SMTP serveru.
- *Port* je číslo portu.

### 3.6.2 Discord

Discord je online komunikační platforma, která slouží k organizaci komunit a komunikaci mezi uživateli. V této kapitole budou vysvětleny základní kroky pro nastavení bota a umožnění zasílání notifikací na vybraný server nebo kanál prostřednictvím bota.

**Postup přidání bota na Discord kanál:**

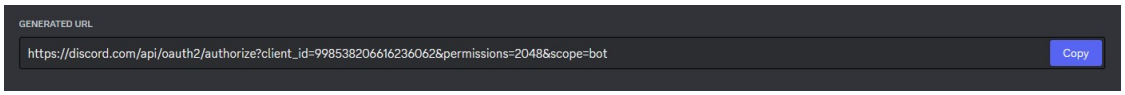
1. Registrace aplikace.
2. Vytvoření a nastavení bota.
3. Oprávnění bota.
4. Generování odkazu pro přidání bota.
5. Přidání bota na kanál.

**Registrace aplikace:** Nejdříve je nutné se zaregistrovat jako vývojář na oficiálních stránkách Discord. Po registraci je třeba přejít do sekce pro vývojáře a vytvořit novou aplikaci v sekci Aplikace[20]. Po registraci a vytvoření aplikace v sekci OAuth2 najdeme „Client ID“, což je identifikátor, který jednoznačně identifikuje naši aplikaci.

**Vytvoření a nastavení bota:** Na vývojářském portálu Discord v sekci Bot lze nastavit bota. Prostřednictvím bota může vaše aplikace interagovat s Discord serverem nebo s konkrétním kanálem. V této sekci také získáme token bota, který je pro nás velice důležitý pro správnou funkcionalitu zasílání notifikace na Discord (v pozdější části kapitoly si to vysvětlíme).

**Oprávnění bota:** Před tím, než bota přidáme na server nebo kanál, lze nastavit, jaká oprávnění bot na serveru bude mít. Discord poskytuje různé oprávnění, jako je čtení zpráv, posílání zpráv, správa serveru atd. Pro naše potřeby stačí, když umožníme, aby bot mohl zasílat textové zprávy na server.

**Generování odkazu pro přidání bota:** Po nastavení oprávnění můžete vygenerovat speciální odkaz, který umožní přidat bota na server (kanál). Tento odkaz obsahuje informace o aplikaci, botovi a udělených oprávněních. Odkaz je generován na stránce „OAuth2“ v sekci „URL Generátor“ a je vygenerován podle zvolených oprávnění. Jak může vypadat vygenerovaný odkaz můžete vidět na obrázku 3.




Obrázek 3: Generování odkazu pro přidání bota.

[20] https://discord.com/developers/applications

Parametry ve vygenerovaném odkazu (viz příklad na obrázku 3) jsou:

- *client_id* – jde o identifikátor, který jednoznačně identifikuje aplikaci. Discord používá tento parametr k identifikaci aplikace, kterou chcete autorizovat.

- *permissions* – jde o oprávnění. V Discord jsou oprávnění reprezentována pomocí číselných hodnot. Každé oprávnění má svoji unikátní hodnotu. Když potřebujeme omezit přístup botu nebo uživatelům k určitým funkcím či akcím v Discordu, používáme kombinace těchto hodnot. Kombinování oprávnění funguje jednoduše a to tak, že se sečtou hodnoty odpovídajících oprávnění. Například pokud chcete udělit oprávnění ke čtení zpráv (Read Messages) a posílání zpráv (Send Messages), součet hodnot těchto dvou oprávnění je 1024 (Read Messages) + 2048 (Send Messages) = 3072. Pro naše potřeby postačí, když udělíme botu oprávnění posílání zpráv.

- *scope* – parametr specifikuje rozsah autorizace, který naše aplikace žádá. Hodnota „bot“ v parametru *scope* v tomto příkladu indikuje, že aplikace žádá o autorizaci a přístup k Discordu serveru pro svého bota.

**Přidání bota na kanál**
Vygenerovaný odkaz v předchozím kroku můžeme zkopírovat a vložit odkaz do webového prohlížeče. Otevře se stránka, která nám umožní vybrat server nebo kanál, do kterého chceme bota přidat. Potvrzením akce bude bot přidán na vybraný server nebo kanál.

**Konfigurace zasílání za pomocí Discordu**
Pro správnou funkcionalitu zasílání notifikace na Discord je navíc třeba upravit konfiguraci v souboru „appsettings.json“ v objektu s názvem *DiscordSetting* v projektu *ReportApp.ApiSender*.

```
"DiscordSetting": {
    "Token": "TokenBot",
}
```

Sémantika parametrů objektu *DiscordSetting*:

- *Token* – jde o token bota. Token najdeme v sekci nastavení bota na portálu Discord pro vývojáře viz část této kapitoly 3.6.2 v sekci „Vytvoření a nastavení bota“.

### 3.6.3 Slack

Slack je online komunikační platforma, která je často využívána v obchodních a vývojářských týmech pro efektivní spolupráci. Umožňuje vytváření týmů a kanálů pro různé účely a podporuje přímé zprávy. Slack také umožňuje integraci s různými nástroji.

Slack prostřednictvím webhooků umožňuje automatizované zasílání zpráv a právě této funkcionality náš notifikační systém využívá. Postup pro získání URL webhooku je popsán v podkapitole 4.3.2 uživatelské dokumentace a také v oficiální Slack dokumentaci [32]. Více informací o službě Slack se můžete dozvědět v oficiální Slack dokumentaci [31].

# 4 Uživatelská dokumentace

Tato kapitola je určena pro uživatele systému spravujícího notifikace. Cílem je poskytnout jasné a srozumitelné instrukce pro efektivní používání systému. Každá sekce obsahuje podrobné kroky a instrukce, které vás provedou danými procesy.

## 4.1 Vytvoření účtu a přihlášení

K přístupu do systému spravujícího notifikace je třeba si nejprve vytvořit účet. Z bezpečnostních důvodů je doporučeno zvolit si dlouhé heslo obsahující různé symboly, což výrazně ztíží jakékoliv pokusy o jeho prolomení.

Vytvoření účtu a následné přihlášení provedete následujícím způsobem:

1. Na úvodní přihlašovací stránce (viz obrázek 4) kliknutím na tlačítko *Založit nový účet*, přejděte na stránku „Vytvoření nového účtu“.

2. Na stránce „Vytvoření nového účtu“ se zaregistrujete vyplněním všech polí v registračním formuláři a kliknutím na tlačítko *Vytvořit účet*. Minimální počet symbolů hesla je 6 znaků. V případě neúspěchu Vám systém důvod neúspěchu oznámí hláškou.

3. Pokud registrace proběhla úspěšně, tak Vás systém přesměruje na úvodní stránku a úspěch Vám oznámí hláškou.

4. Po úspěšném založení účtu se již můžeme přihlásit do systému pod přihlašovacími údaji, které jste zadali při registraci, vyplněním polí a kliknutím na tlačítko *Přihlásit*.

## 4.2 Změna hesla účtu

Pokud potřebujete změnit heslo účtu, je to možné na stránce s názvem „Změna hesla“. Postup je následující:

1. Kliknutím na *Upravit účet* v hlavním menu přejděte na stránku „Změna hesla“.

2. Vyplněním vašeho aktuálního hesla v poli „Staré heslo“, zadáním nového hesla v poli „Nové heslo“, zadáním nového hesla pro kontrolu v poli „Nové heslo kontrola“ a následným kliknutím na tlačítko „Upravit účet“ si změníte heslo účtu.

3. Při úspěšné změně hesla Vás systém přesměruje na úvodní stránku a úspěch Vám sdělí hláškou.

4. Na úvodní stránce se již můžete přihlásit za pomocí nového hesla.


Obrázek 4: Úvodní přihlašovací stránka.

## 4.3 Adresář příjemců

Na této stránce si můžete spravovat příjemce, kterým chcete odesílat notifikace. Příjemci přidaní na této stránce budou následně dostupní k výběru na stránce „Detail notifikace“.

Na stránku „Adresář příjemců“ (viz obrázek 5) se dostanete kliknutím na *Příjemci* v hlavním menu. Na stránce nalezneme panel se záložkami, kde lze přepínat mezi příjemci podle způsobu zasílání. Aktuální vybranou záložku poznáte tak, že je název záložky barevně zvýrazněn.

V každé záložce nalezneme pole, kde se vyplňují názvy a kontaktní údaje příjemců. Pod sekcí přidávání příjemců naleznete výpis (seznam) Vámi přidaných příjemců pro zvolený způsob zasílání. Jednotlivé položky příjemců lze mazat ve výpisu příjemců za pomocí výzvy k akci *smazat*.

Systém umí zasílat notifikace za pomocí služeb email, Slack a Discord.

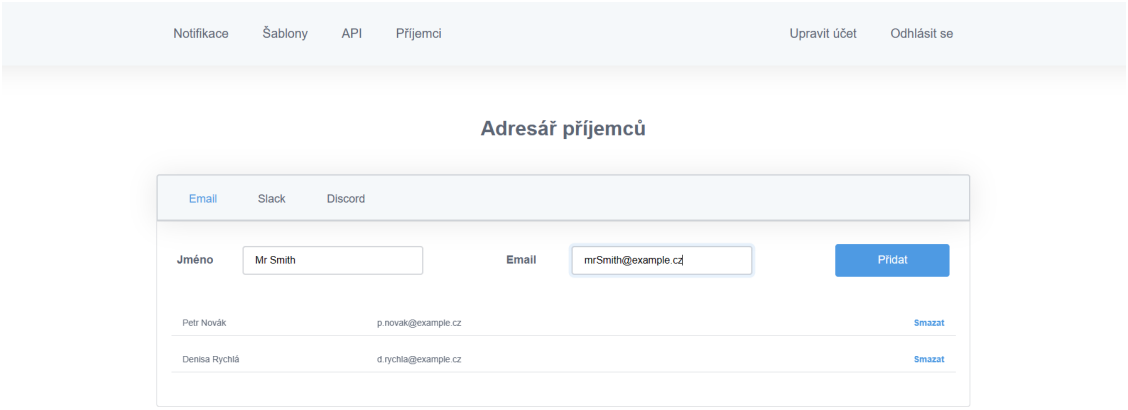
### 4.3.1 Email

Zde jako kontaktní údaj slouží emailová adresa. Postup přidání a případné smazání emailového příjemce je následující:

1. Kliknutím na záložku *Email* na panelu si zobrazíte sekci emailových příjemců.

2. Po vyplnění polí „Jméno“ a „Email“, což je emailová adresa, si nový kontakt uložíte kliknutím na tlačítko *Přidat*.

3. Pokud chcete smazat existujícího emailového příjemce, najděte ho v seznamu a klikněte na výzvu k akci *Smazat*, která je na pravé straně řádku.




Obrázek 5: Stránka adresář příjemců záložka *Email*.

### 4.3.2 Slack

Zde jako kontaktní údaj slouží „URL webhooku“, což je webová adresa, kterou aplikace nebo služba využívá k přijímání oznámení a dat z externích zdrojů. Postup přidání a případné smazání Slack příjemce je následující:

1. Kliknutím na záložku *Slack* na panelu si zobrazíte sekci příjemců služby Slack.

2. Po vyplnění polí „Název“ a „URL webhook“ si nového příjemce uložíte kliknutím na tlačítko *Přidat*.

3. Pokud chcete smazat existujícího Slack příjemce, najděte ho v seznamu a klikněte na výzvu k akci *Smazat*, která je na pravé straně řádku.

**Postup, jak získat Slack webhook:**

1. Přihlaste se do svého Slack workspace.
2. Klikněte na následující odkaz nebo ho zkopírujte do prohlížeče: https://my.slack.com/services/new/incoming-webhook/.
3. V sekci „Post to Channel“ vyberte kanál, do kterého chcete odesílat notifikace v rozevíracím seznamu a poté klikněte na tlačítko *Add Incoming WebHooks integration*.
4. Zobrazí se Vám červeně zvýrazněné pole s názvem *Webhook URL*. Zkopírujte „Webhook URL“ z pole. Toto URL použijte jako kontaktní údaj pro Slack příjemce notifikací.

Poznámka k bezpečnosti: URL webhooku je tajné a kdokoli, kdo ho má, může odesílat zprávy do vašeho kanálu.

### 4.3.3 Discord

Pro zasílání notifikací přes Discord je jako kontaktní údaj potřeba ID kanálu, což je identifikátor kanálu služby Discord. Níže je popsán postup, jak získat ID kanálu a jak přidat Discord příjemce notifikací.

Pro správnou funkcionalitu zasílání notifikací na službu Discord je také potřeba pozvat bota na kanál, kam potřebujeme notifikace zasílat. Bot slouží jako prostředník, který nám zasílané notifikace na Discord kanálu vypíše.

**Postup přidání a případné smazání příjemce Discord je:**

1. Kliknutím na záložku *Discord* na panelu si zobrazíte sekci příjemců služby Discord.
2. Po vyplnění polí „Název“ a „Kanál ID“ si nového příjemce uložíte kliknutím na tlačítko *Přidat*.
3. Pokud chcete smazat existujícího Discord příjemce, najděte ho v seznamu a klikněte na výzvu k akci *Smazat*, která je na pravé straně řádku.

**Postup pozvání bota na kanál je:**

1. V záložce *Discord* na panelu ve spodní části seznamu příjemců služby Discord nalezneme úplně napravo pod seznamem přidaných Discord příjemců výzvu k akci *Pozvat bota na kanál*.
2. Kliknutím na výzvu v akci *Pozvat bota na kanál* se otevře v prohlížeči nová stránka v záložce.
3. V záložce se přihlásíte pod svým Discord účtem.

4. Následně v dalším kroku vyberete server, kde chcete zobrazovat zasílané notifikace, v rozevíracím seznamu „ADD TO SERVER“ a kliknutím na tlačítko *Continue* přejděte k dalšímu kroku.

5. V dalším kroku nechte zaškrtávátko „Send Messages“ zaškrtnuté a kliknutím na tlačítko *Authorize* povolíte botu na zvoleném serveru zasílat notifikace.

**Postup získání ID Discord kanálu je:**

1. Otevřete aplikaci Discord a přihlaste se do svého účtu.

2. Přejděte na server a kanál, pro který chcete získat ID.

3. Klikněte pravým tlačítkem myši na název kanálu a vyberte možnost *Kopírovat ID serveru*. Tato možnost je dostupná pouze, pokud máte povolený režim pro vývojáře.

Pokud nemáte povolený režim pro vývojáře, můžete ho povolit následováním těchto kroků:

1. Klikněte na ikonu ozubeného kola v levém dolním rohu aplikace Discord k otevření nastavení uživatele.

2. V levém panelu v sekci „Nastavení aplikace“ vyberte *Pokročilé*.

3. V sekci „Pokročilé“ zaškrtněte „Vývojářský režim“.

4. Poté můžete získat ID kanálu kliknutím pravým tlačítkem myši na název kanálu a výběrem možnosti *Kopírovat ID serveru*.

Toto ID kanálu můžete nyní použít jako kontaktní údaj pro Discord příjemce notifikací. Pokud možnost „Vývojářský režim“ nevidíte, je to možná způsobeno tím, že nemáte pro zvolený server dostatečné oprávnění.

## 4.4 Správa API

Notifikační systém umožňuje obohatit zasílané notifikace o data získaná z API různých služeb. Aby to fungovalo, je potřeba mít v systému přidaná API, na která chceme dotazy směřovat.

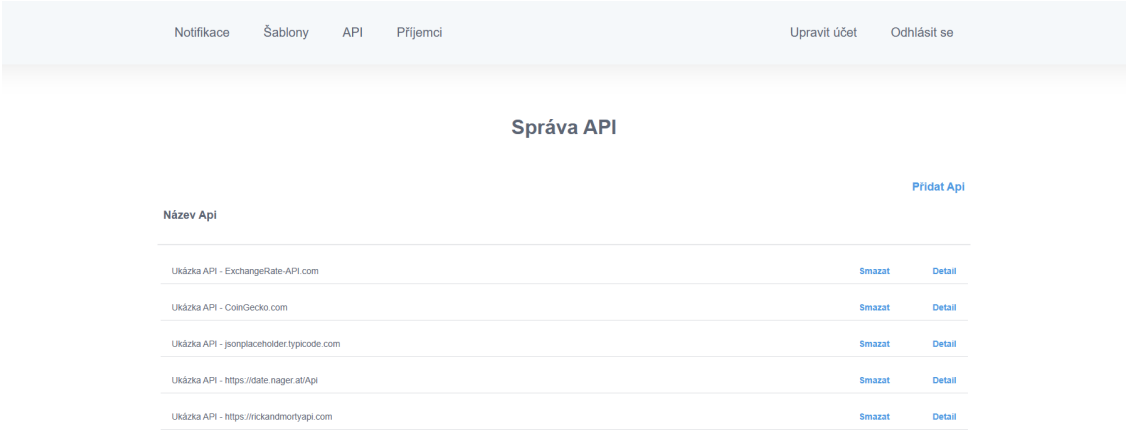
### 4.4.1 Přidání API

Stránka „Detail šablony API“ (viz obrazek 7) slouží jak pro přidávání nového API, tak pro modifikaci existujícího API. Postup přidání nového API je následující:

1. Kliknutím na *API* v hlavním menu, které naleznete v hlavičce, přejděte na stránku „Správa API“ (viz obrazek 6).

2. Kliknutím na výzvu k akci *Přidat Api* si zobrazíte stránku „Detail šablony API“.

3. Na stránce „Detail šablony API“ po vyplnění pole „Název API“, nastavení způsobu autorizace API (viz 4.4.2 detailně) a přidání dotazů na API (viz 4.4.3 detailně) přidáte nové API kliknutím na tlačítko *Uložit změny*.




Obrázek 6: Stránka správa API.

### 4.4.2 Nastavení způsobu autorizace API

Notifikační systém funguje s API, které mají tyto způsoby autorizace API – bez autorizace, autorizace za pomocí API klíče, OAuth 2.0 – Authorization Code a OAuth 2.0 – Client Credential. Postup je následující:

1. Na stránce „Detail šablony API“ v sekci „Autorizace API“ vyberte v rozbalovacím seznamu způsob autorizace API.

2. Pokud zvolíte způsob autorizace API klíč, vyplňte pole „Api klíč“.

3. Pokud zvolíte způsob autorizace OAuth 2.0 – Authorization Code, vyplňte pole „Auth URL“, „Access Token URL“, „Client ID“ a „Client Secret“. Poté můžete získat přístupový token kliknutím na tlačítko *Get Access Token*, čímž se Vám otevře okno, kde potvrdíte autorizaci[21].

[21] postup může být různý, záleží na API, případné informace najděte na stránkách provozovatele API

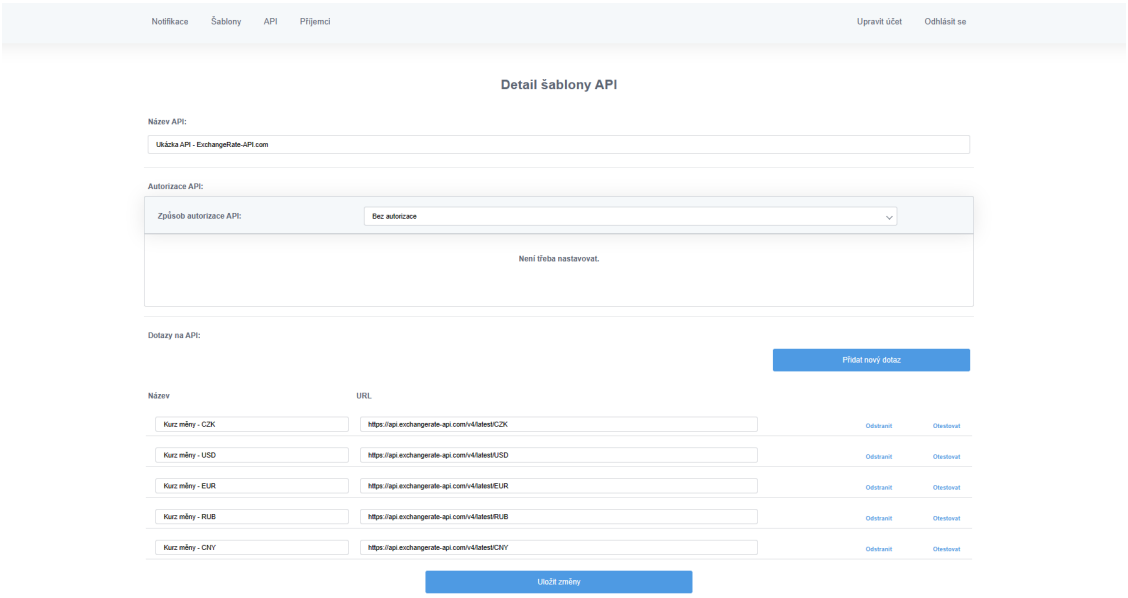
4. Pokud zvolíte způsob autorizace OAuth 2.0 – Client Credential, vyplňte pole „Access Token URL“, „Client ID“ a „Client Secret“. Poté můžete získat přístupový token kliknutím na tlačítko *Get Access Token*.

5. Změny uložíte kliknutím na tlačítko *Uložit změny* dole na stránce.

**Další podrobnější informace k nastavení způsobu autorizace API**
API klíč je tajný identifikátor, který je používán pro autentizaci požadavků na API. Tento klíč by měl být chráněn a nikdy by neměl být sdílen veřejně.

Client ID je veřejný identifikátor pro aplikace, zatímco Client Secret je tajný a je známý pouze aplikaci a autorizačnímu serveru. Je důležité, aby byl Client Secret chráněn a nikdy nebyl sdílen veřejně. Pro více informací o Client ID a Client Secret můžete navštívit tuto stránku [33].

Při uložení změn kliknutím na tlačítko *Uložit změny* se uloží aktuálně vybraný způsob autorizace API v rozbalovacím seznamu a tento způsob autorizace se bude brát jako zvolený způsob autorizace API.




Obrázek 7: Stránka detail šablony API.

### 4.4.3 Přidávání a správa dotazů na API

K získání dat z API je potřeba mít definované dotazy, které budou na API směřovány. Tyto dotazy můžete přidat na stránce „Detail šablony API“ v sekci „Dotazy na API“. Důležitým předpokladem je, že všechny dotazy jsou dotazy typu HTTP metody GET.

Postup správy dotazů na API je následující:

1. Na stránce „Detail šablony API“ v sekci „Dotazy na API“ kliknutím na tlačítko *Přidat nový dotaz* přidáte nový řádek v seznamu dotazů.

2. V nově přidaném řádku vyplňte pole „Název“ a „URL“, kde pole „URL“ je URL dotazu na API.

3. Kliknutím na výzvu k akci *Otestovat* můžete vyzkoušet, zda API na dotaz vrátí kladnou odpověď. Zobrazí se hláška s výsledkem.

4. Kliknutím na výzvu k akci *Odstranit* můžete smazat dotaz.

### 4.4.4 Úprava a mazání API

Postup úpravy a mazání API je následující:

1. Kliknutím na *API* v hlavním menu, které naleznete v hlavičce, přejděte na stránku „Správa API“.

2. Kliknutím na výzvu k akci *Smazat* v seznamu API smažete tento řádek (toto API).

3. Kliknutím na výzvu k akci *Detail* v seznamu API si zobrazíte stránku „Detail šablony API“, kde můžete vybrané API modifikovat (viz 4.4.1).

## 4.5 Šablony a šablonovací jazyk

Šablony hrají klíčovou roli v notifikačním systému, jelikož slouží pro formátování textu notifikace. V předchozí kapitole jsme se zaměřili na přidávání API a vytváření dotazů na tato API. V této kapitole se podíváme na to, jak můžeme tyto dotazy využít k obohacení textu notifikace o data získaná z API. K tomu nám poslouží specifický šablonovací jazyk, který umožňuje integraci těchto dat přímo do textu notifikace.

### 4.5.1 Přidání šablony

Stránka „Detail šablony“ (viz obrazek 8) slouží jak pro přidávání nové šablony, tak pro modifikaci existující šablony. Postup přidání nové šablony je následující:

1. Kliknutím na *Šablony* v hlavním menu, které naleznete v hlavičce, přejděte na stránku „Správa šablon“.

2. Kliknutím na výzvu k akci *Nová šablona* si zobrazíte stránku „Detail šablony“.

3. Na stránce „Detail šablony“ vyplníte pole „Název šablony“.

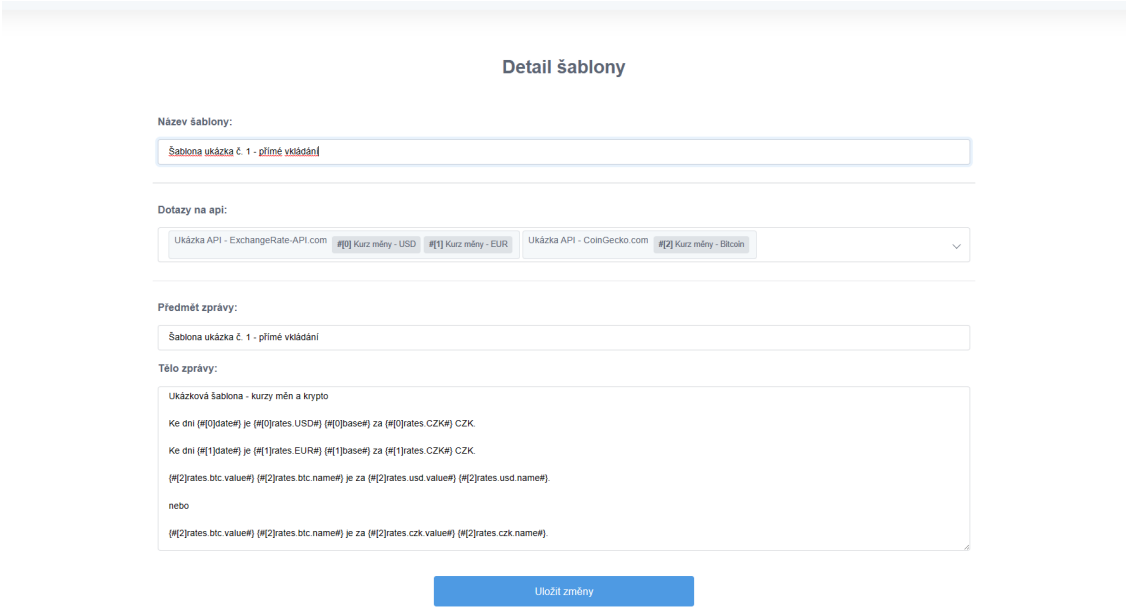
4. Následně v sekci „Dotazy na api“ vyberte dotazy na API, které chcete pro tuto šablonu použít. Dotazy vybírejte zaškrtáváním dotazů v rozbalovacím menu.

5. Vyplňte pole „Předmět zprávy“, které bude sloužit jako název zprávy zobrazovaný jako předmět emailu.

6. V dalším kroku vyplňte pole „Tělo zprávy“, které specifikuje text notifikace. Tento text bude zasílán v notifikacích využívajících tuto šablonu.

7. Po dokončení všech úprav klikněte na tlačítko *Uložit změny* pro uložení šablony.

Jak spravovat API a přidávat dotazy na API je popsáno v kapitole 4.4. Vybrané dotazy mají před názvem uvedené číslo v hranatých závorkách. Toto číslo je číslo zdroje dat (viz 4.5.4 „Zdroj dat“). Jak formátovat „Tělo zprávy“ je popsáno v kapitole „Šablonovací jazyk“ 4.5.4.




Obrázek 8: Stránka detail šablony.

### 4.5.2 Úprava a mazání šablony

Pro úpravu nebo smazání šablony postupujte následovně:

1. Kliknutím na *Šablony* v hlavním menu, které naleznete v hlavičce, přejděte na stránku „Správa šablon“.

2. Pro smazání šablony najděte v seznamu šablon tu, kterou chcete smazat a klikněte na tlačítko *Smazat*.

3. Pro úpravu šablony najděte v seznamu šablon tu, kterou chcete upravit a klikněte na tlačítko *Detail*. Tím se dostanete na stránku „Detail šablony“, kde můžete provést potřebné úpravy (viz 4.5.1).

### 4.5.3 Přidání ukázkových šablon

Pokud jste smazali ukázkové šablony, které se vytvořily při registraci a potřebujete je obnovit, tak pro jejich obnovu můžete použít následující postup:

1. Kliknutím na *Šablony* v hlavním menu přejděte na stránku „Správa šablon“.
2. Pro přidání ukázkových šablon klikněte na výzvu k akci „Přidat ukázkové šablony“.
3. Po úspěšném přidání ukázkových šablon se Vám zobrazí hláška. Kromě ukázkových šablon se Vám přidají i API, které jsou pro tyto šablony potřeba.

### 4.5.4 Šablonovací jazyk

Šablonovací jazyk je základním nástrojem pro vytváření dynamických textů notifikací. Je to jednoduchý jazyk, který se skládá z několika základních symbolů a konstrukcí.

- **Vkládání jednoduchých dat** – `{#[i]#key}`. Tato konstrukce se používá pro vkládání jednoduchých dat z odpovědi na API dotaz. Symbol `i` označuje zdroj dat (číslo dotazu na API) a `key` je klíč, pod kterým jsou data uložena v odpovědi na API dotaz. Například pokud máme odpověď na API dotaz ve formátu JSON:

  ```
  {
      "id": 1,
      "mesto": "Olomouc"
  }
  ```

  Pak `{#[0]mesto#}` vloží do textu notifikace hodnotu „Olomouc“.

- **Vkládání vnořených dat** – `{#[i]key.key#}`. Tato konstrukce se používá pro vkládání vnořených dat z odpovědi na API dotaz. Například pokud máme odpověď na API dotaz ve formátu JSON:

  ```
  {
      "id": 1,
      "pocasi": {
          "teplota": 26
      }
  }
  ```

Pak `{#[0]pocasi.teplota#}` vloží do textu notifikace hodnotu „26“.

- **Iterace přes pole dat** – `{%for item in [i]%}...{%endfor%}`. Tato konstrukce se používá pro iteraci přes pole dat z odpovědi na API dotaz. Symbol `i` označuje zdroj dat (číslo dotazu na API). V těle cyklu můžete použít symbol `{%item.key%}` pro vkládání dat z aktuálního prvku v iteraci. Například pokud máme odpověď na API dotaz ve formátu JSON:

```
[
    {
        "id": 1,
        "mesto": "Olomouc",
    },
    {
        "id": 2,
        "mesto": "Praha",
    }
]
```

  Pak `{%for item in [0]%}{%item.mesto%}{%endfor%}` vloží do textu notifikace hodnoty „Olomouc“ a „Praha“. Pole může být i vnořené v nějakém klíči, pak můžeme použít konstrukci pro vkládání vnořených dat `{%for item in [i]key%}`.

Důležité konstrukce šablonovacího jazyka a jejich význam:

- **Značky pro přímé vkládání dat** – `{#` a `#}`. Tyto značky označují místo, kde se vloží data z odpovědi na dotaz na API. Můžeme je nazvat otevíracími a uzavíracími značkami.
- **Zdroj dat** – `[i]`. Tato konstrukce, kde `i` je číslo, označuje zdroj dat. Číslo udává, z jakého dotazu na API budou data vložena mezi otevíracími a uzavíracími značkami.
- **Klíč dat** – `key`. Tato konstrukce reprezentuje klíč, který určuje, která hodnota z odpovědi na API dotaz se má vložit mezi otevíracími a uzavíracími značkami.
- **Cykly pro iteraci přes pole dat** – `{%for item in [i]%}` a `{%endfor%}`. Tato konstrukce označuje začátek a konec cyklu, který se používá pro iteraci přes pole dat získaných z API. Symbol `item` reprezentuje aktuální prvek v iteraci. Pokud chceme iterovat pole, které je uvnitř jiného vnějšího pole, je to možné za pomocí konstrukce `{%for item in [i]key[j]%}`.

Tyto základní konstrukce umožňují vytvářet šablony pro notifikace, které se dynamicky přizpůsobují datům získaným z API.

Poznámka: V aplikaci jsou k dispozici ukázkové šablony s využitím různých konstrukcí.

### 4.5.5 Příklady použití šablonovacího jazyka

**Základní příklad**
Představme si, že máme odpověď z API ve formátu JSON:

```
{
    "id": 1,
    "mesto": "Olomouc",
    "datum": "2023-06-01",
    "pocasi": {
        "teplota": 26,
        "stav": "slunečno",
    }
}
```

Chceme vytvořit notifikaci, která bude informovat o počasí ve formátu:

„Hezký den, v městě X je dnes teplota Y a je Z.“

V šabloně to zapíšeme takto:

```
Hezký den,
v {#[0]mesto#} je dnes teplota
{#[0]pocasi.teplota#} a je {#[0]pocasi.stav#}.
```

Výsledný text notifikace pro tuto konkrétní odpověď bude:

„Hezký den, v městě Olomouc je dnes teplota 26 a je slunečno.“

**Příklad se dvěma dotazy na API**
Máme odpovědi na dva dotazy na API (dva zdroje dat) a chceme vytvořit notifikaci, která bude informovat o počasí ve dvou městech.

Představme si, že máme odpověď na první dotaz na API (zdroj dat 0) ve formátu JSON:

```
{
    "id": 1,
    "mesto": "Olomouc",
    "datum": "2023-06-01",
    "pocasi": {
        "teplota": 26,
        "stav": "slunečno",
    }
}
```

a odpověď na druhý dotaz na API (zdroj dat 1) ve formátu JSON:

```
{
    "id": 2,
    "mesto": "Praha",
    "datum": "2023-06-01",
    "pocasi": {
        "teplota": 50,
        "stav": "deštivo",
    }
}
```

Chceme vytvořit notifikaci, která bude ve formátu:

„Hezký den, v městě X_0 je dnes teplota Y_0 a je Z_0 a v městě X_1 je dnes teplota Y_1 a je Z_1“.

V šabloně to zapíšeme takto:

```
Hezký den,
v {#[0]mesto#} je dnes teplota
{#[0]pocasi.teplota#} a je {#[0]pocasi.stav#} a
v {#[1]mesto#} je dnes teplota
{#[1]pocasi.teplota#} a je {#[1]pocasi.stav#}.
```

Výsledný text notifikace pro tyto dotazy bude:

„Hezký den, v městě Olomouc je dnes teplota 26 a je slunečno a v městě Praha je dnes teplota 50 a je deštivo.“

**Příklad odpovědi na dotaz na API, která obsahuje pole**
Máme odpověď z API, kde se nachází pole a chceme vytvořit notifikaci, která bude informovat o počasí ve všech městech v poli. Provedeme to za pomocí cyklu (iterace přes pole dat).

Představme si, že máme odpověď z API ve formátu JSON, kde se nachází pole:

```
[
    {
        "id": 1,
        "mesto": "Olomouc",
        "datum": "2023-06-01",
        "pocasi": {
            "teplota": 26,
            "stav": "slunečno"
        }
    },
    {
        "id": 2,
        "mesto": "Praha",
        "datum": "2023-06-01",
        "pocasi": {
            "teplota": 50,
            "stav": "deštivo"
        }
    }
]
```

Chceme vytvořit notifikaci, která bude informovat o počasí ve formátu:

„Hezký den, v městě X_i je dnes teplota Y_i a je Z_i.“

s tím, ať se vypíšou všechna města v poli. V šabloně to zapíšeme takto:

```
Hezký den,
{%for item in [0]%}
v městě {%item.mesto%}
je dnes teplota {%item.pocasi.teplota%}
a je {%item.pocasi.stav%};
{%endfor%}.
```

Výsledný text notifikace s tímto dotazem bude:

„Hezký den, v městě Olomouc je dnes teplota 26 a je slunečno; v městě Praha je dnes teplota 50 a je deštivo;.“

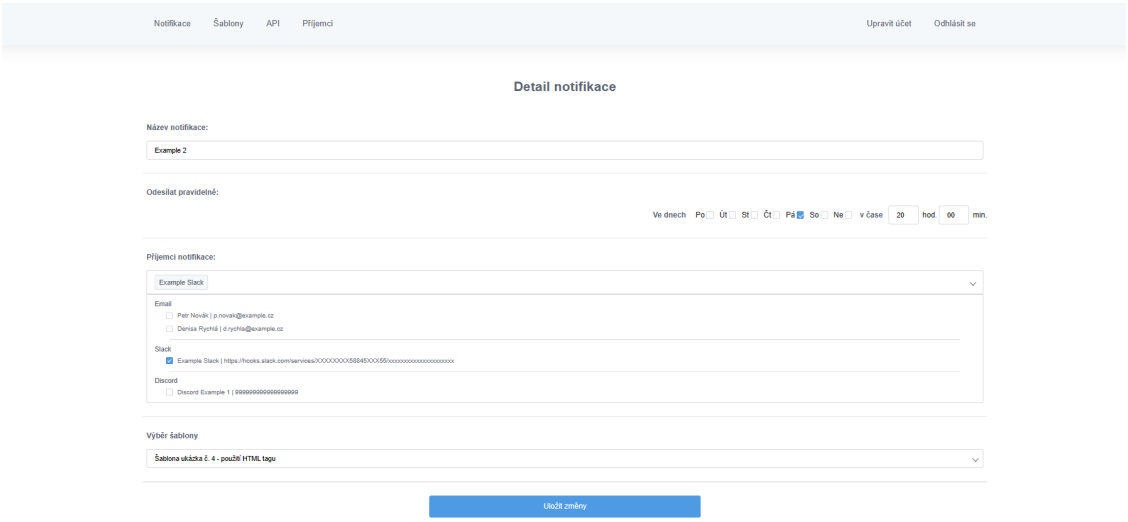
Na příkladech jsme si ukázali způsoby, jak lze využít šablonovací jazyk pro dynamické vytváření textu notifikací na základě dat získaných z API. Je možné i používat vnořené cykly. Má to však své omezení a to takové, že služby, které odesílají notifikace, mají určitý limit na počet znaků ve zprávě. Pokud je text zprávy příliš dlouhý, může dojít k tomu, že notifikace nebude odeslána.

## 4.6 Notifikace

Notifikace jsou jádrem našeho systému. Tato kapitola se zaměřuje na to, jak notifikace vytvářet a spravovat. Nyní se podíváme na to, jak je vytvářet.

### 4.6.1 Přidání Notifikací

Stránka „Detail notifikace“ (viz obrázek 9) slouží jak pro přidávání nové notifikace, tak pro modifikaci existující notifikace. Postup přidání nové notifikace je následující:




Obrázek 9: Stránka detail notifikace.

1. Kliknutím na *Notifikace* v hlavním menu přejděte na stránku „Správa notifikací“.
2. Kliknutím na výzvu k akci *Nová notifikace* si zobrazíte stránku „Detail notifikace“.
3. Na stránce „Detail notifikace“ vyplníte pole „Název notifikace“.
4. Dále na stránce v sekci „Odesílat pravidelně“ zaškrtněte ve kterých dnech a v jakých časech si přejete notifikaci dostávat.
5. Následně v sekci „Příjemci notifikace“ vyberte kterým příjemcům notifikace zasílat. Příjemce vybírejte zaškrtáváním příjemců v rozbalovacím menu.
6. V dalším kroku vyberte šablonu ve výběru „Výběr šablony“. Tato vybraná šablona určuje, jakou zprávu bude notifikace odesílat.
7. Po dokončení všech úprav klikněte na tlačítko *Uložit změny* pro uložení notifikace.

### 4.6.2 Úprava a mazání notifikací

Pro úpravu nebo smazání notifikací postupujte následovně:

1. Kliknutím na *Notifikace* v hlavním menu přejděte na stránku „Správa notifikací“.

2. Pro smazání notifikace najděte v seznamu notifikací tu, kterou chcete smazat a klikněte na výzvu k akci *Smazat*.

3. Pro úpravu notifikace najděte v seznamu notifikací tu, kterou chcete upravit a klikněte na výzvu k akci *Detail*. Tím se dostanete na stránku „Detail notifikace“, kde můžete provést potřebné úpravy (viz 4.6.1).

# Závěr

V rámci této bakalářské práce byl navržen notifikační systém, který je schopen pravidelně zasílat notifikace uživatelům několika způsoby. Informace pro tyto notifikace mohou být získávány prostřednictvím API z různých služeb.

Uživatelé mají možnost přidávat přístup k API, spravovat dotazy na API a nastavovat zpracování a reprezentaci výsledků dotazů za pomocí šablon.

Jako součást řešení byly nakonec vytvořeny ukázkové šablony. Tyto šablony slouží jako výchozí bod pro uživatele při tvorbě vlastních šablon a poskytují jasné ukázky toho, jak lze systém využít.

Hlavní přínos pro uživatele, kteří potřebují pravidelně získávat informace z různých API služeb, vidím v tom, že díky tomuto systému mohou uživatelé efektivně spravovat a automatizovat proces získávání a zpracování dat, což jim šetří čas. Díky možnosti přizpůsobení notifikací prostřednictvím šablon a šablonovacího jazyka mohou uživatelé získat přesně takové informace z API, které potřebují.

Nicméně existuje prostor pro vylepšení. V budoucnu by mohla být zlepšena uživatelská přívětivost při vytváření šablon a rozšířena podpora formátování textu v šablonách o bohatý text. To by mohlo zahrnovat možnosti, jako je tučné písmo, kurzíva, podtržení, seznamy, tabulky a další. Tyto vylepšení by mohly notifikační systém ještě více přizpůsobit potřebám uživatelů a zvýšit jeho užitečnost.

Notifikační systém funguje a splňuje všechny požadavky, které byly v úvodu tvorby zadány.

# Conclusions

As part of this bachelor's thesis, a notification system was designed that is capable of regularly sending notifications to users in several ways. Information for these notifications can be obtained through APIs from various services.

Users have the option to add API access, manage API queries, and set up processing and representation of query results using templates.

As part of the solution, example templates were eventually created. These templates serve as a starting point for users when creating their own templates and provide clear examples of how the system can be used.

The main benefit for users who need to regularly obtain information from various API services, I see in the fact that thanks to this system, users can effectively manage and automate the process of obtaining and processing data, which saves them time. Thanks to the possibility of customizing notifications through templates and a templating language, users can get exactly the information from the API they need.

However, there is room for improvement. In the future, user-friendliness could be improved when creating templates and support for text formatting in templates could be expanded to rich text. This could include options such as bold, italics, underlining, lists, tables, and more. These improvements could further customize the notification system to the needs of users and increase its usefulness.

The notification system works and meets all the requirements that were set at the beginning of the creation.

# A Obsah elektronických dat

**bin/**
: Kompletní adresářová struktura se soubory notifikačního systému připravených ke zkopírování na webový server. Adresářová struktura je komprimovaná do zip archivu. Archiv také obsahuje inicializační skript databáze pro vytvoření databázové struktury notifikačního systému.

**src/**
: Kompletní zdrojové soubory notifikačního systému potřebné pro bezproblémové vytvoření spustitelných verzí komponent systému pro zkopírování na webový server.

**text/**
: Text práce ve formátu PDF a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu.

**install/**
: Instalátory aplikací MS SQL Server, SQL Server Management Studio (SSMS) a ASP.NET Core Runtime – Hosting Bundle potřebných pro provoz webové aplikace a API.

**README.txt**
: Instrukce pro zprovoznění notifikačního systému.

# Literatura


[1] Postman, What is an API? [online]. [cit. 2023-07-22]. Dostupné z: https://www.postman.com/what-is-an-api/

[2] Geeks for Geeks, What is an API? [online]. [cit. 2023-07-22]. Dostupné z: https://www.geeksforgeeks.org/what-is-an-api/

[3] Geeks for Geeks, REST API Introduction [online]. [cit. 2023-07-22]. Dostupné z: https://www.geeksforgeeks.org/rest-api-introduction/

[4] IBM, What is a REST API? [online]. [cit. 2023-07-22]. Dostupné z: https://www.ibm.com/topics/rest-apis

[5] Wikipedia, Create, read, update and delete [online]. [cit. 2023-07-22]. Dostupné z: https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

[6] Mozilla Developer Network, HTTP response status codes [online]. [cit. 2023-07-22]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

[7] Digital Hospital, CSV vs XML vs JSON - Which is the Best Response Data Format? [online]. [cit. 2023-07-22]. Dostupné z: https://digitalhospital.com.sg/csv-vs-xml-vs-json-which-is-the-best-response-data-format/

[8] OpenWeatherMap, OpenWeatherMap API [online]. [cit. 2023-07-26]. Dostupné z: https://openweathermap.org/api

[9] Microsoft, Authentication and Authorization in ASP.NET Web API [online]. [cit. 2023-06-12]. Dostupné z: https://learn.microsoft.com/en-us/aspnet/web-api/overview/security/authentication-and-authorization-in-aspnet-web-api

[10] Reschke, J., HTTP Authentication: Basic and Digest Access Authentication. IETF, September 2015. Dostupné z: https://datatracker.ietf.org/doc/html/rfc7617

[11] Jones, M., Hardt, D., The OAuth 2.0 Authorization Framework: Bearer Token Usage. IETF, October 2012. Dostupné z: https://datatracker.ietf.org/doc/html/rfc6750

[12] Jones, M., Bradley, J., & Sakimura, N., JSON Web Token (JWT) [online]. RFC 7519, 2015. [cit. 2023-06-12]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc7519

[13] Hammer-Lahav, E. (Ed.), The OAuth 1.0 Protocol [online]. RFC 5849, 2010. [cit. 2023-06-12]. Dostupné z: https://www.rfc-editor.org/rfc/rfc5849

[14] Hardt, D. (Ed.), The OAuth 2.0 Authorization Framework [online]. RFC 6749, 2012. [cit. 2023-06-12]. Dostupné z: https://www.rfc-editor.org/rfc/rfc6749.html

[15] OAuth Community, OAuth 2.0 [online]. [cit. 2023-07-14]. Dostupné z: https://oauth.net/2/

[16] Microsoft, .NET Documentation [online]. [cit. 2023-06-30]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/

[17] Microsoft, C# Guide [online]. [cit. 2023-06-30]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/

[18] W3Schools, HTML Tutorial [online]. [cit. 2023-06-30]. Dostupné z: https://www.w3schools.com/html/

[19] Mozilla, Learn HTML [online]. [cit. 2023-06-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/HTML

[20] W3Schools, CSS Tutorial [online]. [cit. 2023-06-30]. Dostupné z: https://www.w3schools.com/css/default.asp

[21] Mozilla, Learn CSS [online]. [cit. 2023-06-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS

[22] W3Schools, JavaScript Tutorial [online]. [cit. 2023-06-30]. Dostupné z: https://www.w3schools.com/js/default.asp

[23] Mozilla, Learn JavaScript [online]. [cit. 2023-06-30]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript

[24] Advanced Encryption Standard (AES) [online]. [cit. 2023-06-12]. Dostupné z: https://www.nist.gov/publications/advanced-encryption-standard-aes

[25] Aes Class (System.Security.Cryptography) | Microsoft Learn [online]. [cit. 2023-06-12]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.aes

[26] Vettivel, N., PBKDF2 Hashing Algorithm [online]. Medium, 2023. [cit. 2023-06-12]. Dostupné z: https://nishothanvettivel.medium.com/pbkdf2-hashing-algorithm-8fd8b7e5d8a0

[27] Node.js crypto.pbkdf2Sync() Method [online]. [cit. 2023-06-12]. Dostupné z: https://nodejs.org/api/crypto.html#crypto_crypto_pbkdf2sync_password_salt_iterations_keylen_digest

[28] Kaliski, B., Password-Based Cryptography Specification Version 2.0 [online]. RFC 2898, 2000. [cit. 2023-06-12]. Dostupné z: https://www.rfc-editor.org/rfc/rfc2898.txt

[29] Understanding Rainbow Table Attack [online]. GeeksforGeeks, 2023. [cit. 2023-06-12]. Dostupné z: https://www.geeksforgeeks.org/understanding-rainbow-table-attack/

[30] Klensin, J., Simple Mail Transfer Protocol. IETF, October 2008. Dostupné z: https://www.rfc-editor.org/rfc/rfc5321

[31] Slack, Slack API Documentation [online]. Slack, 2023 [cit. 2023-07-14]. Dostupné z: https://api.slack.com/docs

[32] Slack, Sending messages using Incoming Webhooks [online]. Slack, 2023 [cit. 2023-07-14]. Dostupné z: https://api.slack.com/messaging/webhooks

[33] OAuth 2.0 Simplified, The Client ID and Secret [online]. OAuth 2.0 Simplified, 2023 [cit. 2023-07-24]. Dostupné z: https://www.oauth.com/oauth2-servers/client-registration/client-id-secret/

[34] Nagel, Ch., Professional C# 7 and .NET Core 2.0. 7th edition. Wrox, 2018.

[35] Price, Mark J., C# 7.1 and .NET Core 2.0 – Modern Cross-Platform Development. 3rd Revised edition. Packt Publishing, 2017.

[36] Freeman, A., Pro ASP.NET Core MVC 2. 7th edition. Apress, 2017.