Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Pomocná webová aplikace pro platformy Twitch a YouTube Live

2023

Petr Špirka

Vedoucí práce:
Mgr. Radek Janoštík, Ph.D.

Studijní program: Informatika,
Specializace: Programování a vývoj software

Bibliografické údaje

Autor:	Petr Špirka
Název práce:	Pomocná webová aplikace pro platformy Twitch a YouTube Live
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2023
Studijní program:	Informatika, Specializace: Programování a vývoj software
Vedoucí práce:	Mgr. Radek Janoštík, Ph.D.
Počet stran:	49
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Petr Špirka
Title:	Supporting web application for platforms Twitch and YouTube Live
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2023
Study program:	Computer Science, Specialization: Programming and Software Development
Supervisor:	Mgr. Radek Janoštík, Ph.D.
Page count:	49
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Webová aplikace, která je určena k poskytování služeb pro platformy YouTube Live a Twitch. Konkrétně umožňuje tvůrcům audiovizuálních záznamů na těchto platformách vytvářet a integrovat různé widgety do jejich streamů. Aplikace také poskytuje nástroje určené k podpoře zapojení diváků, především pak systém odměn pro ty, kteří se aktivně zapojují do diskuze v chatu daného streamu.

Synopsis

Web application designed for providing services for platforms YouTube Live and Twitch. Specifically enabling content creators of audiovisual streams on the platforms to create and integrate different widgets into their streams. The application also provides tools used for increased engagement of viewers in the aforementioned streams, namely a reward system for viewers who actively engage in chat of the streams.

Klíčová slova: webová aplikace; C#; ASP.NET; Twitch; YouTube Live

Keywords: web application; C#; ASP.NET; Twitch; YouTube Live

Děkuji panu Mgr. Radku Janoštíkovi, Ph.D., za jeho ochotu a pomoc při vedení této bakalářské práce.

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

Seznam zdrojových kódů

1 Úvod

Tato bakalářská práce se zaměřuje na tvorbu interaktivních prvků pro platformy Twitch a YouTube Live (dále jen widgetů), jež se používají pro vysílání živého audiovizuálního záznamu (dále jen streamu). Na těchto platformách se typicky rozlišují dva (základní) druhy uživatelů:

- Streamer – je uživatel, který nahrává zmíněný audiovizuální záznam na tyto platformy.
- Divák – je uživatel, který sleduje audiovizuální záznam vysílaný streamerem.

1.1 Proč jsou streamovací platformy populární?

Nejdříve je důležité si vyjasnit, proč jsou vůbec streamovací platformy populární, a až poté je možné vysvětlit, proč je žádoucí používat interaktivní prvky, jako jsou ty, které nabízí tato bakalářská práce. Důvodů je hned několik:

- Esporty – jsou velice podobné konvenčním sportům s tím rozdílem, že se jejich obsah vztahuje na kompetitivní hraní videoher [1]. Tento obsah je většinou vysílán právě přes streamovací platformy, jako jsou Twitch a YouTube Live.
- Hra na vyšší úrovni – velké množství diváků sleduje streamery, kteří jsou v dané hře lepší než oni sami. Z tohoto pohledu se diváci mohou ze stylu hraní streamera něco přiučit. Do této kategorie spadají například „speedruny“, což jsou výzvy, ve kterých se snaží hráči dokončit hru co nejrychleji.
- Sledování obsahu, který divák nemá k dispozici – často se stává, že streameři dostávají exkluzivní přístup ke closed beta verzím her za účelem propagace. Toto je výhodné jak pro vývojáře her, tak pro streamery. Vývojáři potenciálně zvýší povědomí o své hře a získají více kupujících, zatímco streameři získají pozornost diváků, kteří chtějí záznam z těchto her vidět. V některých případech vývojáři navíc poskytují divákům těchto streamerů možnost získat beta verze těchto her. (např. vývojáři hry VALORANT [2] – Riot Games [3][4]). Diváci také mohou využít streamů k rozhodnutí, zda by si chtěli hru koupit.
- Parasociální vztah – jde o vztah, ve kterém si jedna osoba představuje, že je ve vztahu s osobou druhou, ta je ovšem buď fiktivní, nebo o této osobě vůbec neví [5]. V kontextu streamovacích platforem je tento fenomén o dost rozsáhlejší, a to kvůli možnosti streamera v reálném čase s diváky komunikovat. Kvůli tomuto rozdílu se v praxi někdy setkáváme s označením těchto vztahů jako vztahy „jednoho a půl“ [6].

- Pocit komunity – jelikož streamovací platformy (většinou) poskytují rozhraní pro chat, dochází ke vzniku komunit kolem streamerů [7], což vede k většímu zapojení diváků.
- Sledování kvůli streamerovi – divákům se líbí osobnost streamera nebo způsob, jakým streamuje.

1.2 Proč je dobré používat interaktivní prvky?

Použití interaktivních prvků má hned několik výhod, které se u jednotlivých streamerů liší a jež jsou částečně závislé na oblíbenosti streamera (například přispívání nebude pro diváky tak atraktivní u streamerů s 2000 diváky oproti streamerům s 20 diváky).

- Streamer si všímá diváků, kteří přispívají – pokud divák přispěje určitou finanční částkou, streamer mu většinou poděkuje. Toho si všimne nejen přispívající divák (někdy to může vést k parasociálním vztahům), ale i ostatní, což může představovat incentivu „ukázat se“.
- Odměny – diváci, kteří se aktivně zapojují do streamů, mohou získávat body, jež lze uplatnit na odměny. Ty mohou donutit streamera hrát hru s postihem (např. na 5 minut bude muset používat pouze jednu ruku) nebo u některých her s integrací přímo ovlivnit průběh hry (např. Dead Cells [8]).
- Cíle – streameři mohou vytyčit tzv. cíle. Idea je taková, že po celkovém přispění dané peněžní sumy něco provedou (např. si pořídí novou kameru, zahrají si hru s diváky apod.).

Tato bakalářská práce se zaměřuje na zvýšení interakce mezi streamerem a divákem poskytnutím několika různých widgetů, které se zobrazují na vysílaném streamu. Aplikace byla vytvořena primárně pro aplikaci OBS Studio, ale je možné ji použít s libovolnou jinou streamovací aplikací, která podporuje webové zdroje (tedy umožňuje zobrazit webovou stránku s JavaScriptem jako součást záznamu).

2 Použité jazyky a technologie

V aplikaci je použito hned několik technologií. Hlavními jazyky v aplikaci jsou C# (sekce 2.4) a HTML (sekce 2.1). Velice důležitými částmi aplikace jsou také technologie ASP.NET (sekce 2.5, poskytuje samotný webový server, na kterém aplikace běží a funkcionalitu kolem něj) a Bootstrap (sekce 2.10, toolkit pro frontend, který podstatně zjednodušuje návrh stránek). Některé knihovny v této kapitole nebudou zmíněny (např. knihovna jscolor, která se používá pro volbu RGBA barev).

2.1 HTML

HTML (HyperText Markup Language) je jazyk sloužící k popsání struktury (sémantiky) webových stránek. Stránka je popisována z pohledu jednotlivých prvků (elementů), které se na ní nacházejí. Je nutné poznamenat, že jazyk HTML by téměř nikdy neměl popisovat vzhled stránek. K tomuto účelu slouží jazyk CSS (sekce 2.2).

2.2 CSS

CSS (Cascading Style Sheets) je jazyk, který se většinou používá v kombinaci s HTML k popisu stylu jednotlivých prvků stránky, které jsou stanoveny jazykem HTML. CSS podporuje formu dědičnosti, jež funguje tak, že potomci elementu v HTML dědí vlastnosti jejich rodičů. Jazyk také podporuje kaskádování (proto jméno **Cascading** Style Sheets). Kaskádování slouží pro rozhodnutí mezi tím, které styly jsou uplatněny v případě, že dojde ke konfliktu. Funguje tak, že se pravidla s nízkou konkrétností (např. pravidlo pro všechny elementy) postupně přepisují pravidly s vyšší konkrétností (např. pravidlo upravující styl všech elementů, které mají danou třídu). Pro příklad kaskádování a dědičnosti viz zdrojový kód 1 a obrázek 1.

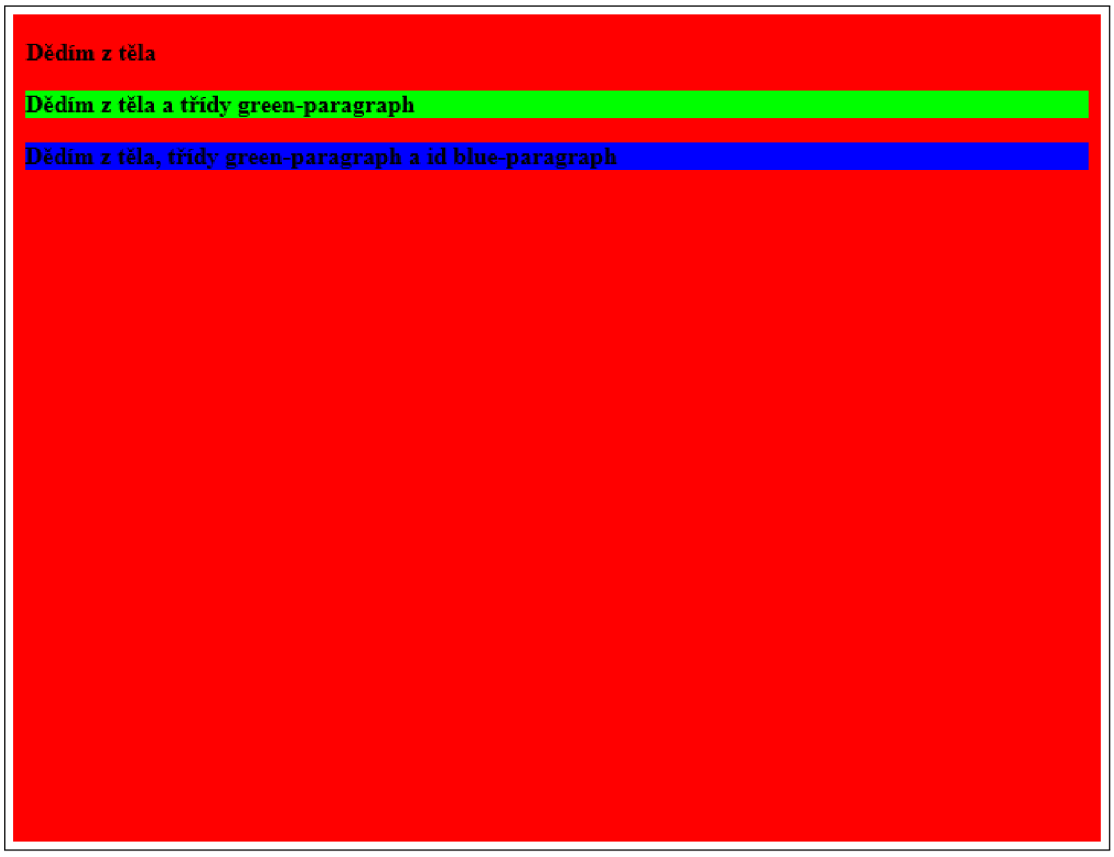
```
<!DOCTYPE html>
<html>
    <head>
        <style>
            /* Tento styl bude aplikován na tělo a všechny jeho
               potomky */
            body {
                background-color: #FF0000;
                font-weight: bold;
            }

            /* Tento styl bude aplikován na elementy s třídou green-
               paragraph */
            .green-paragraph {
                background-color: #00FF00;
            }

            /* Tento styl bude aplikován na elementy s id blue-
               paragraph */
            #blue-paragraph {
                background-color: #0000FF;
            }
        </style>
    </head>
    <body>
        <p>Dědím z těla</p>
        <p class="green-paragraph">Dědím z těla a třídy green-
           paragraph</p>
        <p id="blue-paragraph" class="green-paragraph">Dědím z těla,
            třídy green-paragraph a id blue-paragraph</p>
```

```
26      </body>
27  </html>
```

Zdrojový kód 1: Ukázka dědičnosti a kaskádování v jazyce CSS



Obrázek 1: Stránka generovaná zdrojovým kódem 1

2.3 JavaScript

JavaScript je jazyk sloužící primárně k přidání funkcionality a logiky webovým stránkám, ale dá se použít i jako skriptovací jazyk pro aplikace mimo prohlížeč (např. runtime Node.js [9]). Při použití v prohlížeči má JavaScript k dispozici API zvané *DOM* (Document Object Model). Tímto rozhraním můžeme nejen přistupovat k elementům definovaným jazykem HTML, ale také je modifikovat spolu s jejich styly definovanými pomocí CSS. Pro změnu elementů se používá objekt *document*, který nám umožňuje přistupovat k stromové struktuře celého HTML dokumentu. Pro změnu stylu stačí získat element, jenž chceme změnit, a upravit jeho vlastnost *style*. Ve zdrojovém kódu 2 a obrázku 2 lze vidět ukázku použití DOM API.

```
<!DOCTYPE html>
<html>
    <body>
        <p style="background-color:#00FF00;" id="my-paragraph">Moje
           barva pozadí bude změněna na červenou</p>
        <script>
            //Získání elementu podle id
            const el = document.getElementById("my-paragraph");

            //Nastavení stylu elementu
            el.style.backgroundColor = "#FF0000";
        </script>
    </body>
</html>
```

Zdrojový kód 2: Ukázka použití JavaScriptu s DOM API

Obrázek 2: Stránka generovaná zdrojovým kódem 2

2.4 C#

C# je objektově orientovaný jazyk vytvořený společností Microsoft. Původně byl navržen pouze pro běh na systému Microsoft Windows pomocí frameworku *.NET Framework*, ale později se rozšířil i na jiné operační systémy jako macOS a Linux (původně verzí *.NET Core*, dnes je známá jako *.NET*).

2.5 ASP.NET

ASP.NET je framework sloužící k vývoji webových aplikací pomocí jazyka C#. Poskytuje mnoho pokročilých funkcí jako například stránky *Razor Pages*, které používají kombinaci jazyků HTML a C#, jež umožňuje přímý přístup k C# kódu z HTML pomocí znaku @. ASP.NET také podporuje Angular, React a Vue, které

mohou nahradit Razor Pages jako frontend aplikace. Mimo to ASP.NET poskytuje funkcionalitu jako je například middleware sloužící k úpravě požadavků nebo možnost jednoduchého zakomponování autentizace.

2.6 OAuth

OAuth je standard sloužící ke sdílení dat o uživateli mezi webovými aplikacemi. Protokol funguje tak, že aplikace, která žádá o přístup k datům uživatele, jej přesměruje na speciální stránku aplikace, jež data poskytuje, Aplikace požadující data má v rámci protokolu zvolen parametr známý jako Redirect URI a jsou jí přiřazeny Client ID a Client Secret. Uživatel se zde přihlásí a je přesměrován na stránku aplikace danou parametrem Redirect URI s tím, že URI obsahuje parametr s kódem, jenž aplikace pošle na autentifikační server, který ji vrátí Access Token (popř. Refresh Token, kterým se dají získávat Access Tokeny). Access Token potom slouží jako autentifikace pro přístup k datům uživatele. Mezi výhody tohoto přístupu patří například možnost invalidovat Access Tokeny (pokud by došlo k úniku dat) nebo ovládat, k čemu mají různé aplikace přístup.

2.7 IdentityServer

IdentityServer je framework poskytující systém pro autentizaci a autorizaci uživatelů. Je vyvíjen společností Duende Software [10], která umožňuje volné použití IdentityServeru v případě interního použití ve firmách nebo open source projektech. Výhodou IdentityServeru je jednoduchá integrace do existujících aplikací a snadné používání (IdentityServer si sám dokáže vytvořit potřebné modely a tabulky v databázi).

2.8 Entity Framework Core

Entity Framework Core (dále jen EF Core) je ORM (Object Relation Mapper) od společnosti Microsoft poskytující jednoduchou interakci s databázemi bez nutnosti znalosti SQL či používání funkcionality jedné specifické databáze. EF Core umožňuje jednoduchou změnu modelů pomocí *migrací*, které provádějí přechod z jedné verze dat na jinou. Má také rozsáhlou podporu databází, s nimiž může pracovat (viz dokumentace Microsoftu) [11].

2.9 SignalR

SignalR je knihovna od Microsoftu umožňující komunikaci klienta se serverem v reálném čase. Knihovna funguje na principu RPC (Remote Procedure Call), kdy server dokáže zavolat danou metodu na klientovi [12]. Navíc existuje možnost posílání dat z klienta na server a možnost ovládat jednotlivé klienty odděleně. V aplikaci je SignalR použit k vynucení obnovení stránky, pokud dojde ke změně obsahu (například při změně textu v marquee) nebo při vyvolání události, jež ovlivňuje widgety, v aplikaci.

2.10 Bootstrap

Bootstrap je toolkit usnadňující tvorbu webových stránek. Poskytuje velké množství komponent (jako jsou například modály) a styly, které zjednodušují práci s CSS, přičemž ho často zcela nahrazují. Bootstrap byl vytvořen jako mobile-first framework, což znamená, že je kladen důraz na vývoj pro mobilní zařízení. K tomuto účelu má Bootstrap hned několik možností, například body zlomu, kterými lze definovat chování aplikace, pokud je stránka na malé obrazovce. CSS podporuje stejnou funkcionalitu pomocí tzv. *media query*, ale bootstrap značně usnadňuje použití díky předdefinovaným třídám bez nutnosti jakéhokoliv zásahu do CSS. Pro tvorbu aplikace byl použit Bootstrap 5.3, který přidává primárně možnost použití motivů v aplikaci.

3 Frontend aplikace

Aplikace používá technologii zvanou *Razor Pages*. Ta umožňuje při návrhu stránek v jazyce HTML přímo zakomponovat kód jazyka C#. V ASP.NET je běžně používána architektura MVC s tím, že jednotlivé části této architektury (tedy modely, zobrazení a kontrolery) se nacházejí v samostatných souborech. Razor Pages se dá v této architektuře chápat jako druh zobrazení, ale s tím rozdílem, že kontrolery mohou být automaticky generovány bez jakékoliv specifikace (viz direktiva *@page* zdrojového kódu 3 vysvětlená v sekci 3.1).

```
@page
@model RazorPagesContacts.Pages.Customers.IndexModel
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers

<h1>Contacts home page</h1>
<form method="post">
    <table class="table">
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th></th>
            </tr>
        </thead>
        <tbody>
        @if (Model.Customers != null)
        {
            foreach (var contact in Model.Customers)
            {
                <tr>
                    <td> @contact.Id </td>
                    <td>@contact.Name</td>
                    <td>
                        <!-- <snippet_Edit> -->
                        <a asp-page="./Edit" asp-route-id="@contact.
                            Id">Edit</a> |
                        <!-- </snippet_Edit> -->
```

```
27                          <!-- <snippet_Delete> -->
28                          <button type="submit" asp-page-handler="
                                delete" asp-route-id="@contact.Id">
                                delete</button>
29                          <!-- </snippet_Delete> -->
30                      </td>
31                  </tr>
32              }
33          }
34          </tbody>
35      </table>
36      <a asp-page="Create">Create New</a>
37  </form>
```

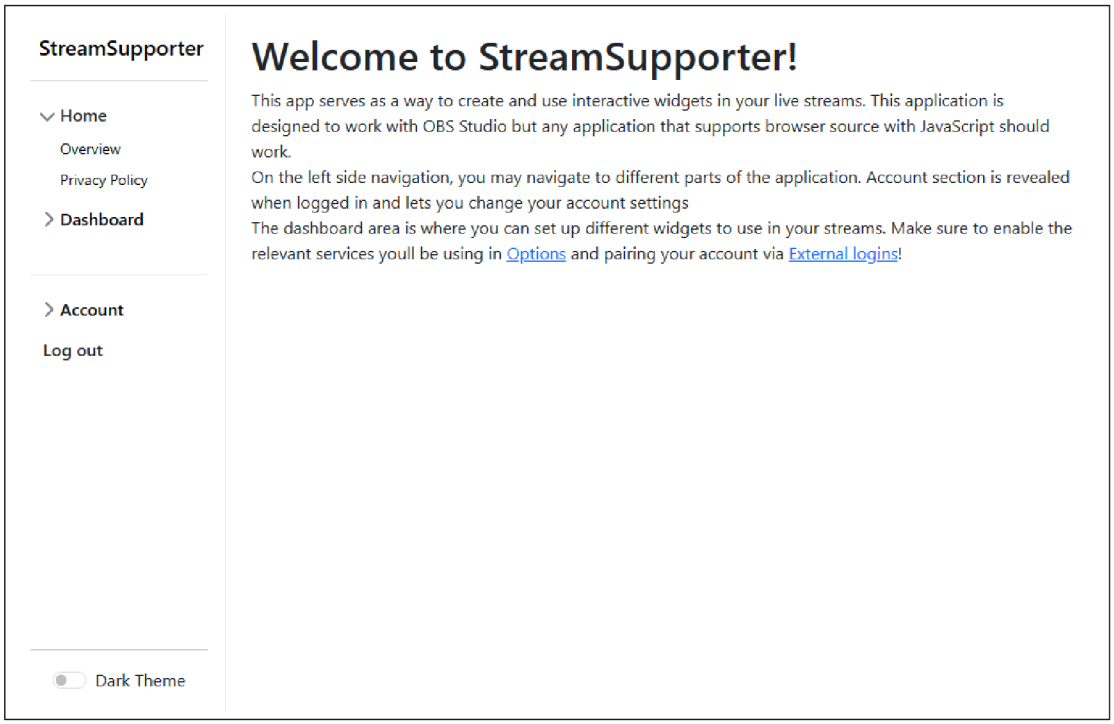
Zdrojový kód 3: Ukázkový kód ze stránek Microsoftu [13]

3.1 Vysvětlení funkcionality Razor Pages

V ukázce kódu 3 je možné vidět zakomponování C# kódu přímo do kódu HTML. Na začátku máme vždy specifikaci modelu (pokud stránka nějaký model používá) a import potřebných knihoven. Direktiva *@page* nám říká, že ke stránce se dá přistoupit přímo pomocí odkazu bez použití kontroleru (pokud tedy *@page* chybí, je potřeba použít kontroler či jinou metodu pro přístup ke stránce, jako je *@partial*). Na řádku 16 je možné vidět rozvětvení v případě, že vlastnost Customers našeho modelu stránky je různá od hodnoty null. Tímto způsobem ošetříme možnost reference na null objekt. Ve větvi, kde vlastnost není null, pak můžeme vidět použití foreach (řádek 18) pro vypsání všech prvků uložených ve vlastnosti Customers.

3.2 Hlavní stránka

Hlavní stránka aplikace slouží pouze jako rozcestník do jiných *oblastí* aplikace. Mimo funkcionality jako vstupní bod aplikace nebude dále použita. Ukázka této stránky se nachází na obrázku 3.



Obrázek 3: Hlavní stránka aplikace

3.3 Uživatelská nástěnka

Tato oblast slouží pro konfiguraci jednotlivých widgetů aplikace. Všechny stránky v této oblasti požadují přihlášení. Kromě zmíněné konfigurace se zde také nachází nastavení služeb, nastavení odměn a nástěnka, na níž jsou notifikace a nepotvrzené odměny.

3.3.1 Nástěnka

Nástěnka obsahuje všechny notifikace a nepotvrzené odměny daného uživatele. Ten má možnost na stránce jednotlivé notifikace smazat a přijmout nebo odmítnout jednotlivé odměny (příklad nástěnky s notifikacemi a odměnami lze vidět na obrázku 4). Notifikace a odměny jsou implementovány pomocí partial zobrazení, které umožňuje větší modulárnost (viz zdrojové kódy 4 a 5). Samotný partial funguje tak, že je vyhledána stránka podle atributu *name*. Poté, co je stránka nalezena, jsou jí data předána atributem *model* (data musí odpovídat třídě, kterou stránka definuje pomocí direktivy model). Tato stránka je potom vložena místo direktivy *partial*.

```
...
<div class="d-flex flex-column">
    @foreach(var notification in Model.Notifications!)
    {
        <partial name="_NotificationPartial" model="notification" />
    }
```

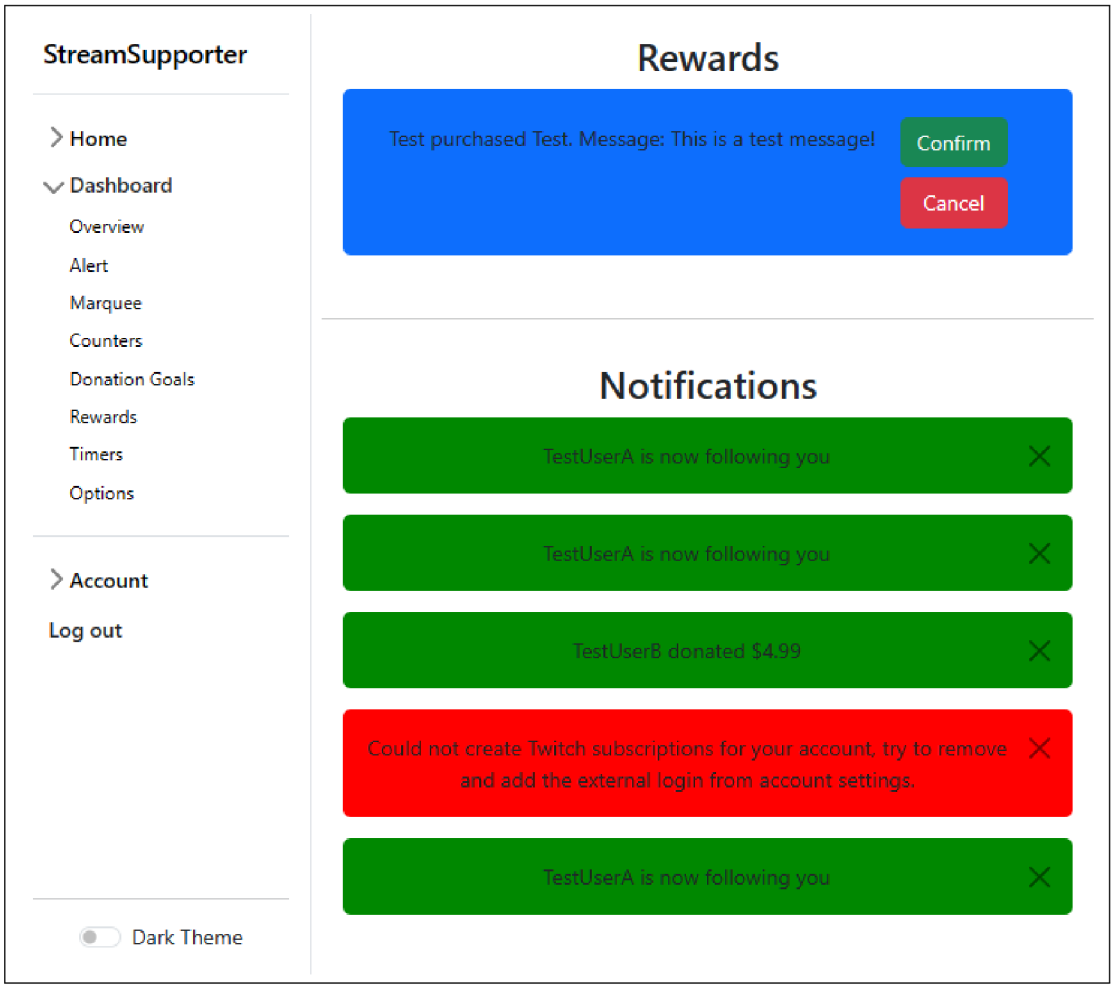
```
</div>
... (kód zkrácen)
```

Zdrojový kód 4: Část kódu nástěnky zobrazující notifikace

```
@using NewStreamSupporter.Data;
@model NewStreamSupporter.Data.NotificationModel
@{
}

<div style="background-color:@Model.BackgroundColor" class="alert
    alert-dismissible fade show flex-row d-flex" role="alert">
     <div class="flex-grow-1">
         <p class="m-0">@Model.Text</p>
     </div>
     <form method="post">
         <input type="hidden" value="@Model.Id" name="id" />
         <input type="hidden" value="@nameof(NotificationModel)" name
             ="type" />
         <button type="submit" class="btn-close" data-bs-dismiss="
             alert"></button>
     </form>
</div>
```

Zdrojový kód 5: Zdrojový kód notifikací



Obrázek 4: Nástěnka aplikace

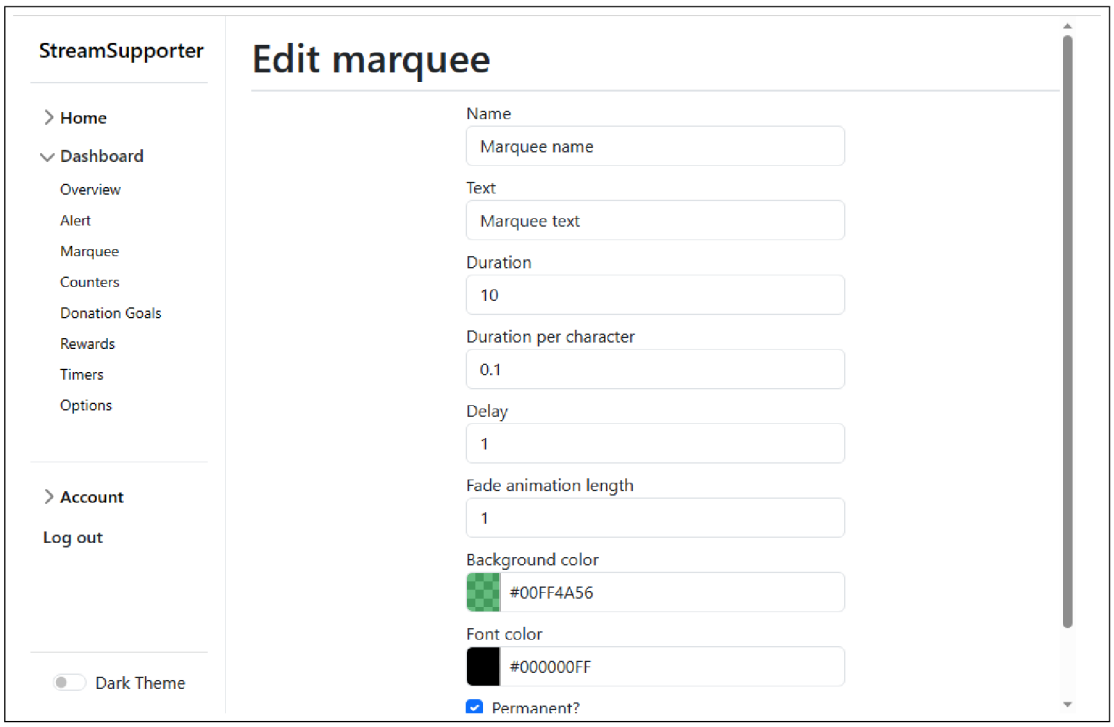
3.3.2 Nastavení jednotlivých widgetů

Stránky pro nastavení widgetů byly vygenerovány funkcionalitou zvanou *scaffolding*. Ta podle vstupních parametrů (v tomto případě hlavně modelem jednotlivých widgetů) vygeneruje stránky umožňující uživatelům práci s tímto modelem. Scaffolding ale není perfektní, a proto je potřeba několika úprav u všech stránek widgetů. Hlavním příkladem je validace uživatelů, protože základní vzor, který scaffolding používá, neobsahuje kód pro IdentityServer. Mezi další velké změny patří úprava UI kódu (použití prvků knihovny jscolor namísto textového pole, odstranění nadbytečných vlastností z formulářů apod.). Po přechodu na stránku nastavení widgetu se zobrazí výpis všech již vytvořených widgetů tohoto typu. U každého z nich se také nachází následující možnosti:

- Edit – umožňuje změnu widgetu.
- Test widget – spustí všechny aktivní instance daného widgetu. Změny provedené touto možností jsou pouze lokální a neovlivní widget v databázi.

- Details – zobrazí detaily o widgetu. Umožňuje přechod na stránky *Edit* a *Delete*.
- Open widget – otevře widget v novém okně. Odtud je možné jej poté přesunout do aplikace OBS Studio.
- Delete – zobrazí stránku umožňující smazání widgetu.

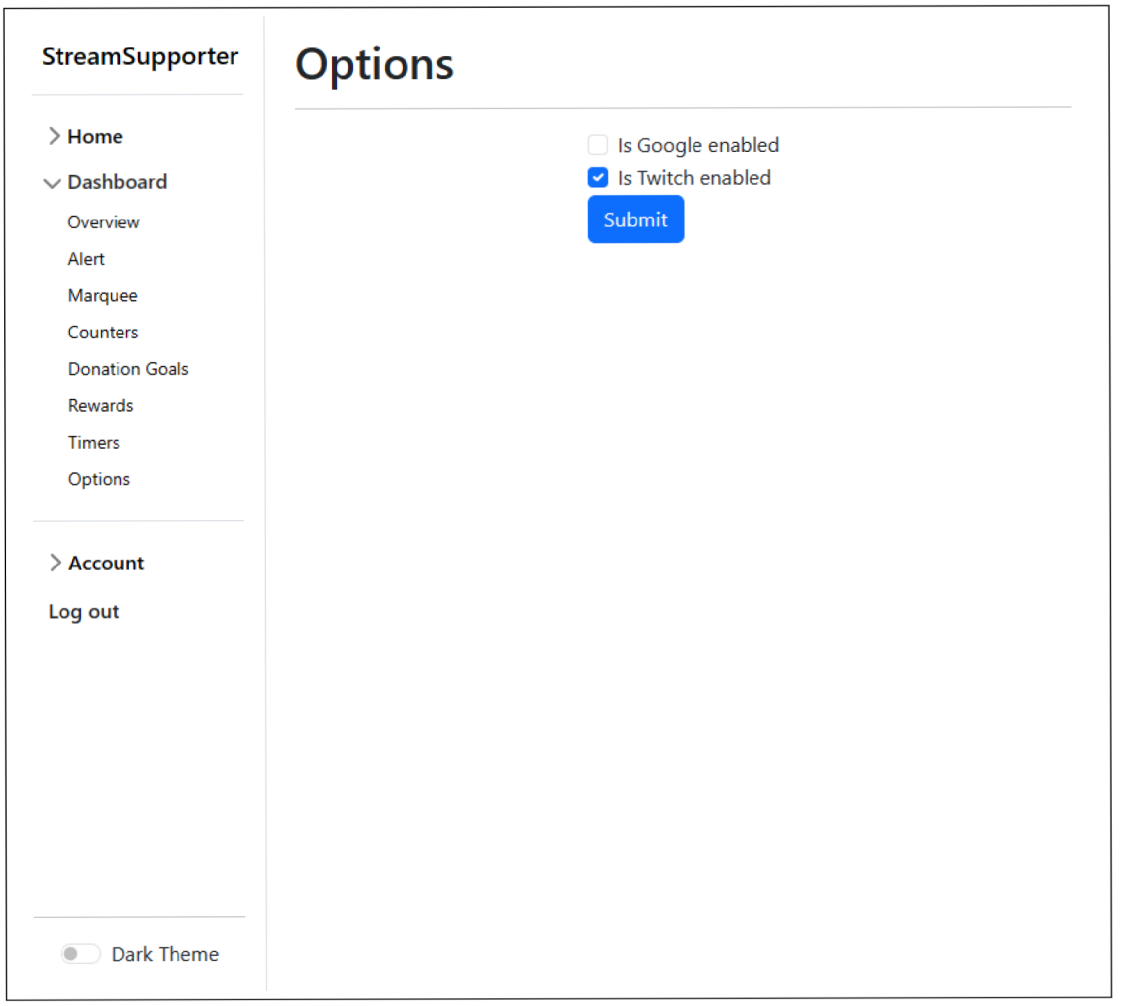
Tato stránka také obsahuje odkaz na vytvoření nového widgetu. Všechny tyto stránky byly vygenerovány obecným *scaffoldingem CRUD* (vyjma možností *Test widget* a *Open widget*). Jedinou výjimkou je stránka nastavení upozornění, která byla také vygenerována touto metodou, ale pouze jako stránka pro editaci. Příklad stránky pro editaci marquee lze vidět na obrázku 5. Pro informace o jednotlivých widgetech aplikace viz sekce 5.



Obrázek 5: Ukázka stránky pro editaci marquee

3.3.3 Nastavení služeb

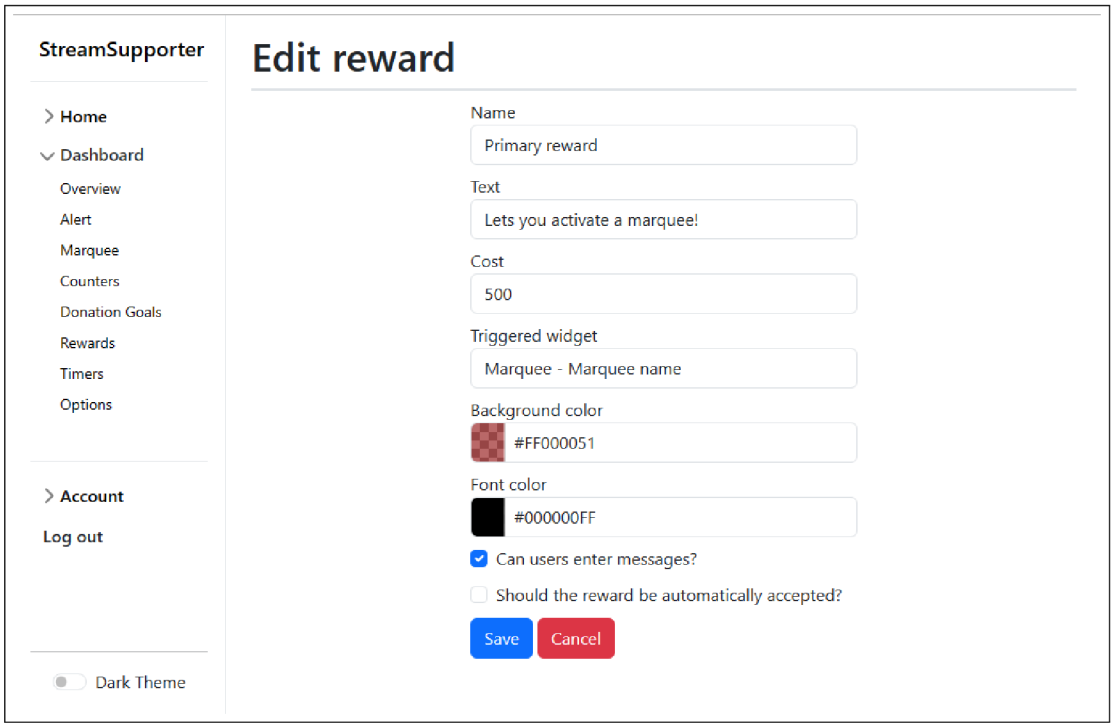
Nastavení služeb umožňuje uživatelům aplikace zvolit, které služby chtějí a nechtějí používat. Ukázku stránky lze vidět v obrázku 6.



Obrázek 6: Ukázka stránky pro nastavení služeb

3.3.4 Nastavení odměn

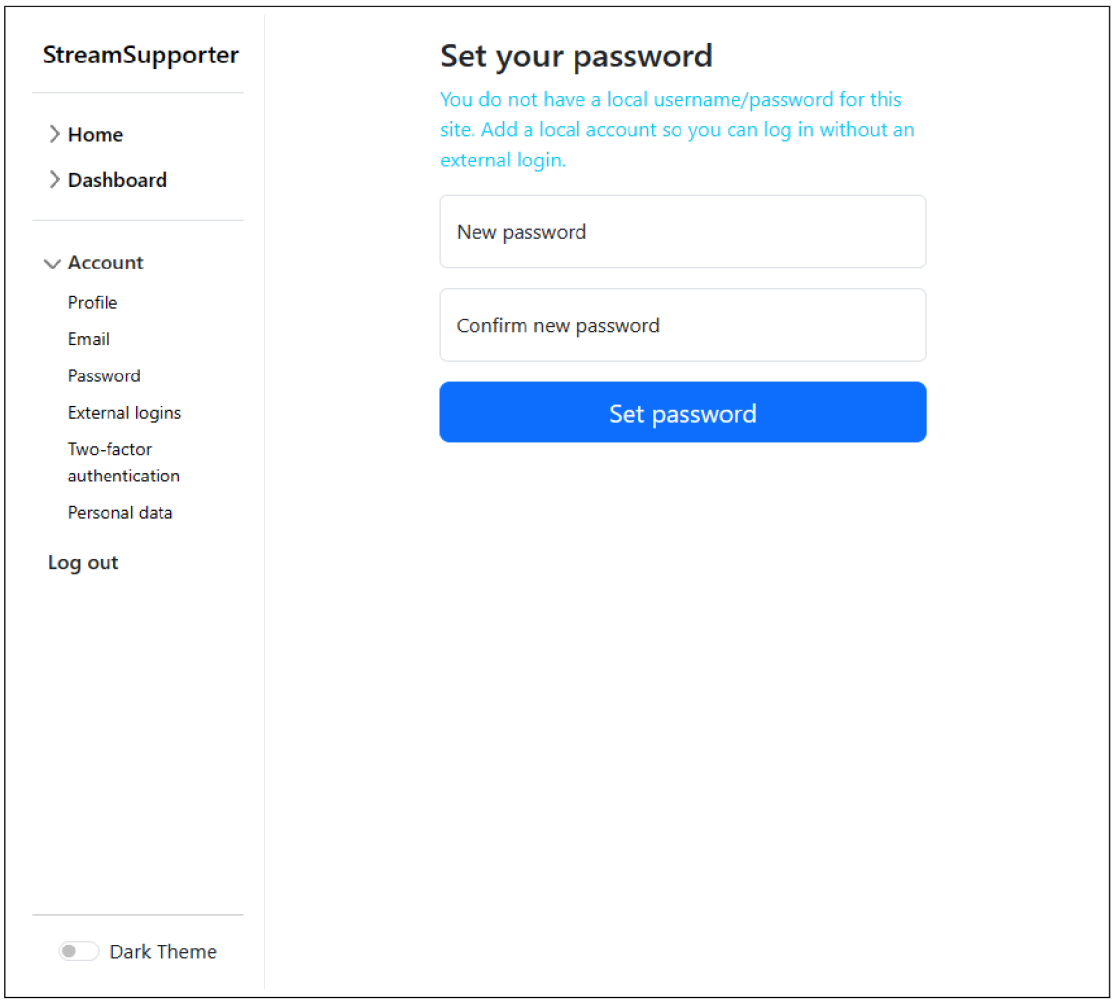
Nastavení odměn je poslední implementovanou stránkou v rámci oblasti uživatelské nástěnky. Slouží k nastavení jednotlivých odměn, které si mohou diváci koupit. Stránka opět používá stránky CRUD generované scaffoldingem (obrázek 7). Rozdílem oproti nastavení widgetů (popsané v sekci 3.3.2) jsou chybějící akce *Test widget* a *Open widget*. Možnost *Open widget* byla přesunuta vedle možnosti vytvoření nové odměny a otevírá stránku se všemi odměnami (akce byla také přejmenována na *Open reward page*). Možnost *Test widget* není pro odměny implementována.



Obrázek 7: Ukázka stránky pro editaci odměn

3.4 Nastavení uživatele

Oblast pro nastavení uživatele je z velké části ponechána původní – tedy tak, jak je vygenerována aplikací Visual Studio. Jedinou větší změnou je použití vlastního rozložení. Pro jednoduchost zde nebudou informace o IdentityServeru uvedeny. Ukázka oblasti pro nastavení uživatele se nachází na obrázku 8.



Obrázek 8: Ukázka stránky pro editaci uživatele

4 Backend aplikace

Backend aplikace používá architekturu podobnou SOA (Service-oriented architecture). Jednotlivé části backendu, jako jsou třídy, které slouží k získávání informací z podporovaných platforem, jsou chápány jako služby, jež poskytují nějakou funkcionalitu. Služby pak mohou používat jiné služby a také je možné služby použít v kontrolerech a zobrazeních. Tyto služby lze získat prostřednictvím návrhového vzoru zvaného *Dependency Injection*. Ten funguje v C# na principu vkládání závislostí do konstruktoru. Nejprve jsou služby, které budou poskytovány, vloženy do kontejneru. Po vložení všech poskytovaných služeb se z kontejneru sestaví kolekce služeb. Každá služba je asociována s typem, jenž je buď rozhraním, nebo třídou. Tato kolekce se dá použít k vytváření nových tříd tím, že se zjistí typy, které má vytvářená třída v konstruktoru. Podle těchto typů je třída vytvořena konstruktorem se službami, jež jsou uložené v této kolekci. V následujících sekcích budou nejdůležitější služby aplikace popsány.

4.1 Služby pro jednotlivé platformy

Každá platforma v aplikaci má hned několik služeb, které umožňují práci s nimi. Všechny platformy mají definovanou třídu dědící z abstraktní třídy *BaseListenerService*, která provádí tyto akce:

1. Registraci nových posluchačů (posluchač je chápán jako uživatel, pro kterého přichází události vztahující se na jeho stream).

2. Rušení existujících posluchačů (je potřebná, pokud uživatel smaže svůj účet nebo pokud deaktivuje danou službu v nastavení služeb).

3. Vyvolávání událostí informujících o stavu streamu (nový příspěvek, stream skončil, nový sledující atd.).

V rámci služeb se pracuje s několika událostmi jako jsou příspěvky a sledující. Jako příspěvky jsou chápány události, které vyvolává divák tím, že přispěje peněžní sumu streamerovi. Na platformě Twitch tomuto odpovídá systém odběrů a bitů. Na platformě YouTube to jsou pak systémy *Superchat* a *Super Sticker*. Sledující jsou uživatele, kteří na platformě Twitch zvolili možnost *Sledovat*. Na platformě YouTube ekvivalentní funkcionalita chybí.

4.1.1 Služby pro platformu Twitch

Pro platformu Twitch existuje hned několik služeb. Základní službou z pohledu vývojáře je třída *TwitchListenerService*. Ta rozšiřuje abstraktní třídu *BaseListenerService* (pro více informací o této třídě viz sekce 4.1). Pro svou funkcionalitu používá několik dalších tříd:

1. Třída *TwitchChatClient* umožňuje připojení do chatu daného streamu. To je důležité pro získávání uživatelů, kteří se aktivně zapojují do diskuze.

2. Třída *TwitchEventSubManager* spravuje EventSub odběrů platformy Twitch. Ty jsou použity pro většinu událostí, které přicházejí z platformy Twitch.

3. Třída *TwitchEventSubWebhookReceiver* přijímá události platformy Twitch a zajišťuje jejich převod na vnitřní události aplikace.

Služby pro platformu Twitch fungují primárně na principu *Publish-subscribe*, kdy aplikace odebírá události (chápáno jako *subscribe*), které ji zajímají, a platforma Twitch posílá notifikace (chápáno jako *publish*) formou HTTP POST požadavků na adresu zvolenou při vytvoření odběru. Pro interakci s chatem streamu je použita technologie IRC (Internet Relay Chat). Do ní se aplikace přihlásí a čte zprávy ostatních uživatelů.

4.1.2 Služby pro platformu YouTube Live

Obdobně jako pro platformu Twitch (sekce 4.1.1), tak i pro platformu YouTube Live existuje několik služeb, které se starají o správný běh aplikace na této platformě. I zde existuje základní třída *YouTubeListenerService*, která dědí z abstraktní třídy *BaseListenerService*. Způsob získávání dat je ale rozdílný. Zatímco Twitch má API pro posílání událostí pomocí HTTP POST požadavků a IRC, YouTube Live žádné takové API nemá. Služby YouTube Live tedy fungují na principu opakovaného dotazování na standardní API. Podle intervalu daném v konfiguračním souboru bude docházet k opakovaným požadavkům. K tomuto účelu slouží následující třídy:

1. Třída *YouTubeProviderService* slouží k vyvolávání požadavků na YouTube API a ke správě uživatelských Access Tokenů.

2. Třída *YouTubePollingService* umožňuje samotné vyvolávání opakovaných dotazů na YouTube API za použití zmíněné služby YouTubeProviderService.

4.1.3 Služba pro správu bodů

Aplikace používá systém bodů, na jehož základě jsou odměňováni diváci, kteří přispějí streamerovi formou peněžní sumy nebo prostřednictvím aktivního zapojení do chatu. Ke správě těchto bodů slouží třída *CurrencyService*. Aplikace vnitřně rozlišuje dva druhy účtů z pohledu bodů – vlastněné účtem aplikace a nevlastněné účtem aplikace. Tento systém umožňuje sbírat body i bez existujícího účtu v aplikaci. Služba *CurrencyService* se tedy pokusí najít, zda existuje uživatel, který má propojený účet pod ID, které aplikace dostane od platformy. Pokud ne, je buď použit existující, nebo vytvořen nový záznam o bodech nevlastněných účtem aplikace, v němž je uloženo ID diváka z platformy, na které body nasbíral. Pokud si následně uživatel vytvoří účet, jsou body nevlastněné účtem aplikace převedeny na body vlastněné účtem aplikace a přiřazeny tomuto nově vytvořenému účtu.

4.1.4 Služby DispatcherHub a DispatcherHubStateService

V aplikaci je často potřeba vynutit obnovení widgetů (např. pokud dojde ke změně nastavení daného widgetu). K tomuto účelu slouží třídy *DispatcherHub* a *DispatcherHubStateService*. Ty používají API poskytované knihovnou SignalR (sekce 2.9) a fungují tak, že každý widget se připojí k instanci třídy *DispatcherHub* pod speciálním endpointem (defaultně se nachází na adrese */Api/Dispatcher*). Po připojení jsou uchována jeho data (o jaký widget se jedná, jeho ID, ID uživatele a ID SignalR klienta) ve třídě *DispatcherHubStateService*. Tato třída potom dokáže převést kombinaci typu, ID a ID uživatele na ID SignalR klienta a naopak. Při požadavku na obnovení nebo poslání dat danému widgetu jsou pak podle jeho parametrů vyhledáni všichni SignalR klienti a je jim poslán daný požadavek. Opačné mapování ID SignalR klienta na parametry widgetu umožňuje smazat reference na klienty, kteří se odpojili.

4.1.5 Jiné služby

Kromě těchto hlavních služeb aplikace obsahuje i následující služby:

1. Služba *SMTPMailSender* umožňuje posílat potvrzovací e-maily a e-maily pro obnovení hesla.
2. Služba *RewardManagerService* zajišťuje správu odměn.
3. Služba *NotificationService* je určena ke správě notifikací.
4. Služba *ListenerStartupService* umožňuje spuštění a propojení služeb.
5. Služba *LocalFileStore* umožňuje ukládání dat do souborů.

4.2 Kontroler pro zobrazování jednotlivých widgetů

Jednotlivé widgety aplikace jsou dostupné pomocí odkazu. Kvůli omezení prohlížeče v OBS Studiu je nutné, aby nebyla použita autentizace, ale zároveň aby neměl přístup k jednotlivým widgetům každý uživatel. Řešením je požadovat jak uživatelské ID, tak ID widgetu. Tento přístup nezaručuje úplnou bezpečnost, ale ID jsou poměrně dlouhé a zkoušení všech různých možností uživatelského ID a ID widgetu by zabralo poměrně dlouhou dobu. Navíc každý widget má svůj vlastní endpoint, čímž se omezí počet widgetů, které může útočník z jednoho endpointu získat. V kódu 6 lze vidět příklad endpointu pro získávání widgetů typu marquee.

```
1 ...
2 public async Task<IActionResult> Marquee(string uid, string id)
3 {
4     var user = await _context.Users.Where(user => user.Id == uid).
          Include(user => user.OwnedMarquees).FirstOrDefaultAsync();
5     var marquee = user?.OwnedMarquees.Where(marquee => marquee.Id ==
          id).FirstOrDefault();
```

```
    if (marquee == null)
    {
        return NotFound();
    }
    return View("MarqueeView", marquee);
}
... (kód zkrácen)
```

Zdrojový kód 6: Ukázkový zdrojový kód kontroleru pro získávání widgetů

5 Widgety aplikace a systém odměn

Aplikace poskytuje hned několik widgetů, které mají zvýšit zapojení diváků. Každý widget má definované standardní API (vnitřně označováno jako Widget API), kterým se dá widget otestovat, aniž by bylo nutné spustit samotný stream. Toto API je u každého widgetu popsáno, ale všechny widgety mají stejnou funkci *reload*, která obnovuje stránky widgetu. Toto API je možné spustit z JavaScriptové konzole prohlížeče, ve kterém se widgety nachází. Alternativně je možné widgety testovat použitím možnosti *Test widget* u daného widgetu. Všechny widgety mají také následující vlastnosti:

- Name – libovolné jméno widgetu.
- Background color – barva pozadí daného widgetu.
- Font color – barva písma widgetu.
- Text – text, který widget zobrazuje.

V některých případech dané widgety mění, jak se daná vlastnost chová. V tomto případě bude u daného widgetu popsáno, co přesně daná vlastnost dělá.

5.1 Marquee

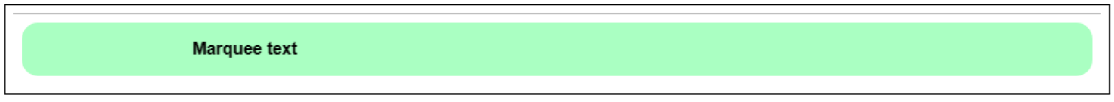
Marquee představuje „přejíždějící text“. Jedná se o widget, který může být spuštěn pomocí odměn a ukazuje danou zprávu, jež pomalu přejede přes celý widget. Text nastavený ve vlastnosti *Text* je použit, pokud je tento widget spuštěn z odměny bez textového pole, nebo pokud se nachází v permanentním režimu. V permanentním režimu používá marquee pouze vlastnost *Text* a spuštění pomocí odměny na něj nemá žádný efekt. Pokud je marquee, který není v permanentním režimu, spuštěn odměnou s textovým polem, ukáže marquee zprávu danou uživatelem, nebo výchozí text, pokud uživatel nezadá do textového pole odměny žádný text. Ukázka marquee se nachází na obrázku 9.

5.1.1 Dodatečné vlastnosti

- Duration – základní doba v sekundách, po jakou marquee zůstane.
- Duration per character – počet sekund, o který je zvýšena vlastnost *Duration* za každý znak (tedy pokud Duration = 3, Duration per character = 0,2 a vstupem je slovo „Test“, výsledná hodnota Duration je $3 + [4 \times 0.2]$).
- Fade animation length – doba animace zmizení a objevení. Nemá vliv na marquee, pokud se nachází v permanentním režimu a vlastnost *Delay* je rovna 0.
- Delay – nemá vliv na marquee, pokud se nenachází v permanentním režimu. Pokud se v permanentním režimu nachází, tato vlastnost ovládá dobu mezi iteracemi jeho spuštění.
- Permanent? – zda má marquee opakovaně zobrazovat text daný vlastností *Text*.

5.1.2 Widget API

- *trigger(text)* – spustí widget a zobrazí text daný parametrem této funkce. Pokud je marquee permanentní, nestane se nic.
- *trigger()* – spustí widget s tím, že se jako text zobrazí vlastnost *Text* widgetu. Pokud je marquee permanentní, nestane se nic.



Obrázek 9: Ukázka marquee

5.2 Upozornění

Upozornění je widget, který se zobrazí poté, když nastane libovolná nakonfigurovaná událost. Může jí být zakoupení odměny, nový peněžní příspěvek nebo nový sledující kanálu. Upozornění se při nakonfigurované události zobrazí po dobu danou vlastností *Duration* (tato doba se může mírně lišit kvůli animacím zobrazení a zmizení) se zprávou vztahující se k druhu události. Pokud je upozornění spuštěno odměnou bez textového pole nebo s prázdným textovým polem, zobrazí se informace o tom, která odměna byla koupena a kým. Pokud je upozornění spuštěno odměnou s textovým polem, které není prázdné, upozornění ukáže zprávu v tomto textovém poli. Při zobrazování widgetu bude také přehráno audio. Pokud není použita vlastnost *Choose custom audio*, je přehrán výchozí zvuk aplikace. Pokud je do této vlastnosti nahrán platný audio soubor a upozornění je poté uloženo, dojde k přehrání tohoto audia. Audio bude spuštěno nejvýše po dobu zobrazení widgetu. Ukázka upozornění se nachází na obrázku 10.

5.2.1 Dodatečné vlastnosti

- Name – na rozdíl od ostatních widgetů má uživatel jenom jedno upozornění, a proto není vlastnost *Name* použita.
- Text – jelikož upozornění se spouští jako reakce na jiné události, text je přijat jako parametr z těchto událostí a vlastnost tedy není použita
- Do follows trigger alerts? – zda má být spuštěn tento widget, pokud streamer dostane nového sledujícího. Tato možnost není podporována na platformě YouTube Live.
- Do donations trigger alerts? – zda má být spuštěn tento widget, pokud streamerovi uživatel přispěje peněžní částku.
- Choose custom audio – umožňuje uživateli zvolit vlastní audio, které bude přehráno při spuštění upozornění.
- Duration – doba, po kterou má být upozornění zobrazeno, pokud dojde k jeho spuštění.

5.2.2 Widget API

- *trigger(text)* – spustí upozornění a zobrazí text daný argumentem *text*.

Obrázek 10: Ukázka upozornění

5.3 Cíl příspěvků

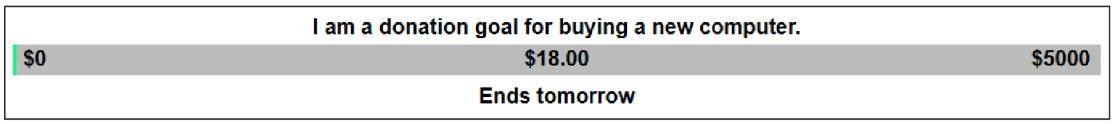
Cíl příspěvků slouží jako počítadlo příspěvků od diváků s tím, že je zde navíc zahrnut cíl, ke kterému se příspěvky přičítají. Při vyvolání události příspěvku je v cíli příspěvku ukázáno, kdo přispěl a jakou částku, přičemž se zvětší poměr vyplněné části ukazatele průběhu. Tento ukazatel nebude svou výplň zvětšovat, jestliže již byl dosažen cíl daný vlastností *Target amount*. I přes to se ale bude zvyšovat hodnota *Current amount*. Tato hodnota má ovšem limit, nad který se nezvýší, i když dojde k zobrazení nového příspěvku. Ukázka cíle příspěvků se nachází na obrázku 11.

5.3.1 Dodatečné vlastnosti

- Background color – přejmenována na *Progress bar color*. Místo změny pozadí widgetu dojde ke změně barvy náplně cíle.
- Target amount – celkové množství peněz, které je potřeba ke splnění cíle.
- Current amount – momentální množství peněz.
- Expiry – do kdy je daný cíl platný. Příspěvky jsou počítány i po překročení platnosti.

5.3.2 Widget API

- *trigger(name, amount)* – zvýší počítadlo o množství *amount* a zobrazí poděkování uživateli dané parametrem *name*.



Obrázek 11: Ukázka cíle příspěvků

5.4 Počítadlo

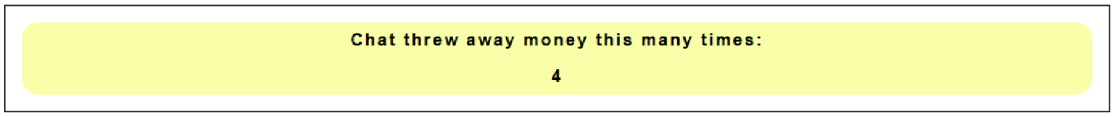
Počítadlo je widget, který umožňuje zobrazit aktuální hodnotu počítadla a popis, čeho se hodnota týká. Tato hodnota je definována vlastností *Value* a popis vlastností *Text*. Je možné jej navýšit koupením odměny nebo příspěvkem, pokud je tato možnost aktivována v nastavení počítadla. Počítadlo má, podobně jako cíl příspěvků, limit, nad který se hodnota počítadla nezvýší. Textové pole nemá v případě odměny na tento widget žádný vliv. Ukázka počítadla se nachází na obrázku 12.

5.4.1 Dodatečné vlastnosti

- Value – aktuální hodnota počítadla.
- Should donations increase value? – zda má počítadlo navyšovat svou hodnotu, pokud je vyvolána událost příspěvku.

5.4.2 Widget API

- *trigger()* zvýší počítadlo o 1.



Obrázek 12: Ukázka počítadla

5.5 Časovač

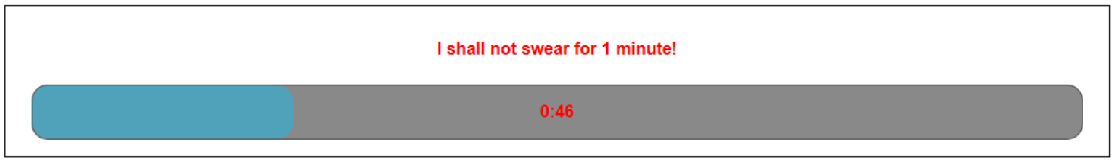
Časovač umožňuje uživatelům po zakoupení odměny spustit časovač s daným textem. Ten se hodí například pro odpočet času výzvy. Skládá se z ukazatele průběhu a popisu. Ukazatel průběhu se naplňuje relativně k uběhnutému času – na začátku bude ukazatel prázdný a po uběhnutí času definovaného vlastností *Timer length* bude plný. Ukázku časovače lze vidět na obrázku 13.

5.5.1 Dodatečné vlastnosti

- Background color – přejmenována na *Progress bar color*. Místo změny pozadí widgetu dojde ke změně barvy náplně časovače.
- Timer length – celková délka časovače v sekundách.

5.5.2 Widget API

- *trigger()* – Spustí časovač s textem daným vlastností *Text* a dobou trvání rovnou vlastnosti *Timer Length*.



Obrázek 13: Ukázka časovače

5.6 Systém bodů a odměn

Aplikace podporuje virtuální systém bodů, které mohou diváci sbírat aktivním zapojením do diskuse v chatu streamu nebo přispíváním peněžních částek streamerovi. Pro tuto funkcionalitu je potřeba, aby streamer zapnul platformu, na které by mělo být možné sbírat body, v nastavení služeb. Pokud je aktivní na daném kanálu stream a uživatel v něm napíše zprávu, je provedena procedura přidání bodů (buď okamžitě, nebo po uběhnutí intervalu daném v konfiguračním souboru). Ta první zkontroluje, zda daný divák v době dané konfiguračním souborem již neposlal zprávu (toto omezení je zavedeno pro omezení spamu – pokud by se každá zpráva počítala do bodů diváka, psaní rychlých zpráv by bylo odměněno vyšším ziskem bodů), Pokud již divák zprávu v tomto intervalu poslal, bude jeho zpráva přeskočena. Pokud zprávu v tomto intervalu neposlal, budou mu přidány body na jeho účet přímo (pokud má v aplikaci účet a ten je spárován s danou platformou, na které se stream nachází) nebo se uloží do dočasného účtu příslušného ID na dané platformě. Takto je možné sbírat body, i když uživatel nemá v aplikaci účet. Pokud by jej poté v budoucnu vytvořil, body budou podle ID spárovány s jeho novým účtem. Obdobně aplikace funguje pro příspěvky (není zde omezené, jak rychle mohou být příspěvky posílány).

Stránky pro nastavení odměn se velice podobají stránkám nastavení jednotlivých widgetů s tím rozdílem, že všechny odměny se nachází na jedné stránce pod uživatelským ID, pro které byly odměny vytvořeny. U jednotlivých odměn tedy neexistuje možnost je zobrazit. Místo toho se odkaz na stránku s odměnami nachází u nadpisu *Rewards*. Mimo to odměny obsahují následující možnosti:

- Cost – počet bodů, které jsou uživateli odečteny po zakoupení této odměny.
- Can users enter messages? – umožňuje uživatelům vložit k odměně zprávu. Pro více informací o podpoře zpráv viz sekce 5.7.
- Triggered widget – widget, který by měl být spuštěn, pokud je tato odměna zakoupena. Specifické chování jednotlivých widgetů viz sekce 5.7.
- Should the reward be automatically accepted? – umožňuje automatické přijetí této odměny, aniž by bylo nutné ji manuálně potvrzovat na nástěnce uživatele.

5.7 Podpora odměn

Některé widgety mohou být spuštěny systémem odměn. Různé widgety se v tomto případě chovají jinak. Toto chování je popsáno v tabulce 14.

Název widgetu	Spuštění bez textového pole	Spuštění s textovým polem
Marquee	Zobrazí text definovaný vlastností text	Zobrazí text daný textovým polem
Upozornění	Zobrazí zprávu o tom, že si uživatel zakoupil odměnu	Zobrazí text daný textovým polem
Cíl příspěvků	Není definováno	Není definováno
Počítadlo	Zvýší počítadlo o 1	Zvýší počítadlo o 1
Časovač	Spustí časovač podle jeho vlastností	Spustí časovač podle jeho vlastností

Plně podporováno
Částečně podporováno
Nepodporováno

Obrázek 14: Tabulka vysvětlující funkcionalitu různých widgetů pro odměny

6 Uživatelská příručka

Aplikace je rozdělená na několik různých *oblastí*, přičemž každá z nich je určena k jinému účelu. Základní oblastí je domovská oblast, ve které se uživatel nachází po prvním navštívení stránky. Zde se nachází domovská stránka, stránky sloužící k informování o ochraně osobních údajů a nakonec stránky s odměnami. Mimo domovské oblasti se v aplikaci také nachází oblast pro správu účtu a oblast uživatelské nástěnky. V této sekci budou všechny tyto oblasti popsány.

6.1 Navigace

Aplikace používá k navigaci dvouúrovňovou postranní navigaci. Ta umožňuje přesun na libovolnou stránku (vyjma stránek pro odměny, widgety a specifické stránky pro úpravu jednotlivých widgetů) kategorizovaných do oblastí. První úroveň navigace představují již zmíněné oblasti – tedy *Home*, *Dashboard* a *Account*. Dále se zde nachází přímý odkaz *Log out* umožňující odhlášení. Výjimkou je situace, kdy uživatel není přihlášen. V tomto případě jsou *Account* a *Log out* nahrazeny možnostmi *Register* a *Login*. Samotné oblasti budou popsány v následujících sekcích. Navigační menu také obsahuje přepínač, který ovládá temný motiv aplikace.

6.2 Domovská oblast

V domovské oblasti se nachází pouze tři stránky – domovská stránka, stránka s ochranou osobních údajů a stránka s odměnami streamerů.

Na domovské stránce se nachází pouze základní informace k aplikaci. Primárně slouží k přivítání uživatelů po otevření odkazu aplikace. Jsou zde také

přesměrováni uživatelé, kteří se odhlásí ze svého účtu.

Stránka s ochranou osobních údajů primárně slouží pro informování uživatelů o tom, jak jsou jejich data používána a jak je s nimi nakládáno.

Poslední stránkou je stránka s odměnami streamerů. Pro přístup k této stránce je potřeba mít k dispozici odkaz stránky daného streamera. Na stránce se nachází jak odměny streamera (pokud nějaké má), tak i počet bodů uživatele, pokud je jeho účet spárován. Pro více informací o tom, jak funguje systém odměn v rámci párování, viz sekce 5.6. Pokud má streamer definovány odměny, každá odměna obsahuje jméno, popis a tlačítko s cenou. Pokud má uživatel dostatek bodů, aplikace mu umožní stisknout tlačítko s cenou, čímž se otevře dialog zakoupení odměny. V tomto dialogu se nachází jak shrnutí toho, co uživatel kupuje, tak textové pole, pokud má daná odměna toto pole povoleno.

6.3 Oblast správy uživatelského účtu

Tato oblast slouží pro správu informací o účtech uživatelů aplikace. Jako základní stránky sem patří přihlášení a registrace. Tyto stránky jsou přístupné pouze tehdy, pokud uživatel není přihlášen. Dá se k nim dostat pomocí odkazů *Login* a *Register* ze všech stránek aplikace (vyjma widgetů). Stránka pro registraci obsahuje formulář k vytvoření účtu, ve kterém je potřeba vyplnit uživatelské jméno, e-mail a heslo. Po stisknutí tlačítka *Register* dojde k vytvoření nového účtu v aplikaci, kdy buď dojde k poslání e-mailu na zadaný uživatelský účet, který je třeba potvrdit, nebo k zobrazení odkazu pro potvrzení, pokud aplikace nemá aktivovanou validní e-mailovou službu. Na stránce pro přihlášení formulář obsahuje pouze uživatelské jméno a heslo. Stisknutím tlačítka *Login* se uživatel přihlásí příslušným uživatelským jménem a heslem. Pokud jsou přihlašovací údaje špatné, zobrazí se chyba upozorňující uživatele na tuto skutečnost. Obě stránky navíc obsahují tlačítka pro externí přihlášení pomocí platformy Twitch a Google (platforma Google je v tomto případě chápána jako platforma YouTube Live). Po kliknutí na odkaz a úspěšné autentizaci se buď uživatel přihlásí k účtu, který je spárován s tímto Twitch / YouTube Live účtem, nebo dojde k vytvoření nového účtu, kdy je uživatel požádán o to, aby si zvolil uživatelské jméno a e-mail. Pokud by při externím přihlášení uživateli trvala autorizace delší dobu, aplikace po dokončení vrátí chybu „Failed getting information from external login“, kdy je potřeba opětovné spárování účtu.

Po přihlášení se uživateli odemkne přístup do oblasti správy uživatelského účtu a uživatelské nástěnky. V oblasti uživatelského účtu existuje hned několik dalších stránek:

- Profile – stránka umožní uživateli změnit jeho uživatelské jméno. Po zvolení *Save* dojde k pokusu o změnu uživatelského jména. Pokud aplikace selže, objeví se chybová hláška zmiňující důvod selhání. V některých případech může dojít k chybě „Unexpected error when trying to set username“. tato chyba oznamuje interní selhání IdentityServeru a měla by být nahlášena vývojáři.

- Email – stránka email umožňuje uživateli změnit jeho e-mailovou adresu v rámci aplikace. Po volbě nové adresy a zvolení možnosti *Change email* je na ní odeslán e-mail, který potvrdí novou adresu.

- Password – stránka password umožňuje uživateli změnit heslo v rámci aplikace (popř. vytvořit nové heslo, pokud byl účet vytvořen pomocí externího přihlášení). Po vepsání nového hesla (popř. starého hesla, pokud si uživatel již heslo v minulosti zvolil) a zvolení možnosti *Set password* se aplikace pokusí nastavit nové heslo. Pokud dojde k chybě, je uživatel o této skutečnosti informován zprávou na této stránce.

- External logins – stránka external logins umožňuje uživatele spárovat (a popřípadě rušit párování) s externími platformami. Pokud má uživatel účet spárovaný s externí platformou, bude její jmeno v sekci *Registered logins*, kde je možné párování odstranit. Párování lze odstranit pouze pokud by tím nedošlo ke smazání účtu (tedy pokud uživatel nemá zvolené v aplikaci heslo, je potřeba mít spárovaný alespoň jeden účet a aplikace mu nedovolí jej odpárovat). V sekci *Add another service to log in* se nachází služby, se kterými si uživatel může spárovat účet. Sekce mohou být aplikací vynechány, jestliže jsou prázdné.

- Two-factor authentication – tato stránka slouží pro aktivaci dvoufázové autentizace (dále jen 2FA). Ta umožňuje vyšší bezpečnost účtu při standardním přihlášení (nedá se použít při externím přihlášení). Funguje tak, že při přihlášení heslem a e-mailem je také potřeba zadat kód z aplikace poskytující tuto autentifikaci. Pokud uživatel nemá momentálně nastavenou 2FA, najde na této stránce pouze možnost *Add authenticator app*. Tato možnost ukáže uživateli postup, jakým spárovat 2FA aplikaci. Po dokončení všech kroků na této stránce se uživateli aktivuje 2FA a zobrazí se kódy pro obnovení účtu, pokud by uživatel ztratil přístup k jeho autentifikační aplikaci. Stránka bude obsahovat nové možnosti:

  - *Disable 2FA*, která po potvrzení deaktivuje 2FA tomuto účtu.
  - *Reset recovery codes*, jež po potvrzení vygeneruje nové kódy pro obnovení účtu.
  - *Set up authenticator app*, která umožní opětovné nastavení autentifikační aplikace.
  - *Reset authenticator app*, která resetuje klíč autentifikační aplikace a umožní opětovné přenastavení aplikace (oproti předchozí volbě zneplatní všechny stávající autentifikační aplikace).

- Personal data – stránka personal data umožňuje uživateli stáhnout nebo smazat všechny informace o něm, které má aplikace uložené v databázi. Volbou *Download* dojde ke stažení těchto dat, zatímco volbou *Delete* dojde ke smazání všech dat uživatele.

6.4 Uživatelská nástěnka

Tato oblast je určena pro správu funkcionality aplikace. Uživatelé zde mohou spravovat své widgety a vypínat nebo zapínat jednotlivé platformy, pro které by měla aplikace zjišťovat stav.

6.4.1 Nástěnka

Nástěnka informuje streamera o aktuálním dění v aplikaci. Stránka má dvě sekce – notifikace a odměny. Do notifikací jsou aplikací přidávány zprávy vztahující se ke stavu účtu (např. zneplatněné přihlášení jedné z platforem). U každé notifikace se nachází zpráva a tlačítko umožňující notifikaci odstanit. U odměn se nachází informace o tom, kdo si koupil jakou odměnu a popřípadě text této odměny, pokud má odměna povoleno textové pole. U odměn se nachází dvě možnosti. První možností je odměnu přijmout, čímž dojde ke spuštění widgetu daného odměnou se zprávou (pokud je textové pole aktivní). Pokud odměna nemá spuštěný widget definovaný, nestane se nic. Druhou možností je odměnu odmítnout, čímž dojde k vrácení bodů uživateli, který tuto odměnu zakoupil.

6.4.2 Stránky pro správu widgetů

Všechny widgety aplikace mají stránku, která umožňuje jejich správu. Ke všem těmto widgetům se dá přistoupit z navigačního menu, přičemž všechny podporují stejné operace (vyjma upozornění, jelikož každý uživatel má právě jedno upozornění, a odměn, které se chovají odlišně, viz dále popsáno níže). Pro libovolný widget aplikace existuje výpis, ke kterému se dá dostat pomocí postranního navigačního menu. Zde je možné vidět všechny uživatelem vytvořené widgety. Na stránce výpisu se vždy nachází možnost *Create new*, která přesune uživatele na stránku, kde je možné vytvořit příslušný widget. Na této stránce je potřeba vyplnit všechny vlastnosti pro tento widget (pro více informací o těchto vlastnostech viz sekce 5). Dále má uživatel možnost pokusit se vytvořit tento widget stisknutím tlačítka *Create*. Pokud se v libovolném poli nachází neplatné údaje, aplikace uživatele na tuto skutečnost upozorní. Samotný výpis widgetů je řešen formou tabulky, ve které se nachází uživatelem vytvořené widgety, které jsou kategorizovány podle jména a obsahují možnosti pro tento widget. Zde má uživatel u každého widgetu pět možností:

- Edit – otevře stránku, která umožňuje uživateli upravit tento widget. Stránka je vizuálně stejná jako v případě stránky pro vytváření nových widgetů. Jediným rozdílem je to, že hodnoty jsou již předvyplněny.
- Details – otevře stránku s detaily o widgetu. Stránka je totožná se stránkou pro úpravu widgetů s tím rozdílem, že pole není možné upravovat.
- Test widget – spustí všechny aktivní instance widgetů s testovacími daty. Změny widgetu provedené touto možností jsou pouze lokální a budou ztraceny po opětovném načtení stránky widgetu.

- Open widget – otevře nové okno, ve kterém se nachází tento widget. Uživatel by měl zkopírovat odkaz této stránky a použít jej v aplikaci OBS Studio či jiné alternativě této aplikace. Uživatel také může widget na této stránce testovat (viz sekce 5).
- Delete – otevře okno, ve kterém je možné smazat daný widget. Tato operace se nedá vrátit.

V případě stránky pro správu upozornění pak aplikace imituje chování stránky pro úpravu, tedy neexistuje zde stránka výpisu, smazání ani detailů. Pro testování a zobrazení tohoto widgetu se nachází odkazy v dolní části této stránky. Jinak se také chová stránka pro editaci odměn. Zatímco stránky *Edit*, *Details* a *Delete* se chovají stejně, možnost *Open widget* ve výpisu odměn chybí a nachází se místo toho vedle možnosti *Create new reward* (všechny odměny se nachází na jedné stránce). Ve výpisu odměn také úplně chybí možnost *Test widget*. Pro více informací o widgetech viz sekce 5.

6.4.3 Nastavení

Na stránce pro nastavení má uživatel možnost aktivovat nebo deaktivovat jednotlivé služby, se kterými je jeho účet spárován. Stisknutím tlačítka *Submit* se změny uloží.

7 Použití aplikace

V této kapitole bude popsáno, jak aplikaci zprovoznit a nasadit.

7.1 Sestavení a spuštění aplikace

Aplikace byla vytvořena pro platformu .NET, což znamená, že se dá spustit jak na operačním systému Windows, tak na systémech macOS a Linux. Pro systémy macOS a Windows je doporučeno použít vývojové prostředí poskytované Microsoftem nazvané Visual Studio. Pro systémy založené na Linuxu je možné použít příručku na stránkách Microsoftu[14].

Před spuštěním aplikace je nutné upravit konfigurační soubor aplikace – appsettings.json. V tomto souboru se nachází mimo jiné nastavení Kestrel serveru, na němž aplikace poběží, a nastavení autentizace. Do tohoto souboru je potřeba dodat data z jednotlivých poskytovatelů autentizací (v nastavení označeny jako Client ID a Client Secret). Tato data se dají sehnat ze stránek poskytovatelů:

- Twitch[15]: po přihlášení/vytvoření účtu je potřeba zvolit možnost *Your Console* a zde vytvořit novou aplikaci tlačítkem *Create your application*. Zde se po vypsání hodnot jako Redirect URI (měla by být stejná jako adresa aplikace s */signin-twitch* na konci) a jména objeví Client ID a Client Secret, které je potřeba dosadit do konfiguračního souboru aplikace. Aby

byla zajištěna správná funkcionalita, je navíc potřeba vytvořit klíč pro interakci s IRC. K tomu je potřeba použít stránku Twitch Chat Password Generator[16]. Po propojení Twitch účtu s aplikací je dále nutné vložit vygenerovaný token a uživatelské jméno spárovaného účtu do konfiguračního souboru aplikace. Posledním krokem je volba klíče pro komunikaci s platformou Twitch a adresu, na kterou by měl Twitch posílat EventSub notifikace. Klíč by měl být netriviální pro ochranu validity přijímaných notifikací.

- YouTube[17]: po přihlášení/vytvoření účtu je potřeba vytvořit nový projekt vybráním možnosti *Select a project* v levém horním rohu stránky a poté možnost *Vytvořit projekt*. Po vytvoření a zvolení projektu je potřeba přejít do sekce *APIs and Services*, podsekce *Library*, kde najdeme a přidáme *YouTube Data API v3*. Poté přejdeme na sekci *APIs and Services* a podsekci *OAuth consent screen*. Zde je potřeba přidat uživatele tlačítkem *Add User*, ale pouze tehdy, pokud aplikace není spuštěna oficiálně pomocí tlačítka *Publish*. Po přidání uživatelů, kteří budou aplikaci využívat, stačí uložit změny do seznamu testovacích uživatelů a přejít do sekce *Credentials*, kde zvolíme *Create Credentials* a *OAuth client ID*. V nabídce zvolíme typ aplikace jako *Web Application*, přidáme URI naší aplikace jako Redirect URI s tím, že za ní přidáme */signin-google* a zvolíme *Create*. Poté nám pouze stačí zkopírovat právě vytvořená data do konfiguračního souboru aplikace.

Dále je také potřeba získat FreecurrencyAPI[18] klíč, jež slouží pro převod měn na platformě Google. Existují placené možnosti, ale také možnost využívat pomalejší a omezenější API zdarma (omezeno na 5000 požadavků za měsíc a pomalejšími aktualizacemi hodnot měn). API klíč získáme po registraci[19] v záložce *Dashboard*, Potom už pouze stačí klíč v sekci *Default Key* zkopírovat do konfiguračního souboru.

7.2 Vytvoření účtu a widgetů v samotné aplikaci

Po spuštění aplikace (popř. použití již hostované verze) je potřeba vytvořit uživatelský účet. K tomu stačí přejít na stránky aplikace (defaultně adresa https://localhost). Je možné, že prohlížeč ukáže varování o neplatném SSL certifikátu – v prohlížeči Chrome stačí zvolit tlačítko detail a poté *pokračovat na stránku*. Je nutno poznamenat, že bez platného certifikátu SSL, popřípadě pokud je aplikace spuštěna na jiném portu než 443, může dojít k omezení funkcionality aplikace, jelikož jsou tyto parametry vyžadovány platformou Twitch (konkrétně je potřeba, aby notifikace z platformy Twitch byly posílány na port 443. Toho se dá dosáhnout i bez toho, aby samotná aplikace běžela na portu 443, a sice použitím tunelovací služby, jako je ngrok[20]).

Po zobrazení hlavní stránky stačí zvolit tlačítko registrace v postranní navigaci. Po zvolení je nutné buď přímo vytvořit nový účet pomocí registračního formuláře, nebo použít externí přihlášení pomocí platformy Twitch/Google. (Google

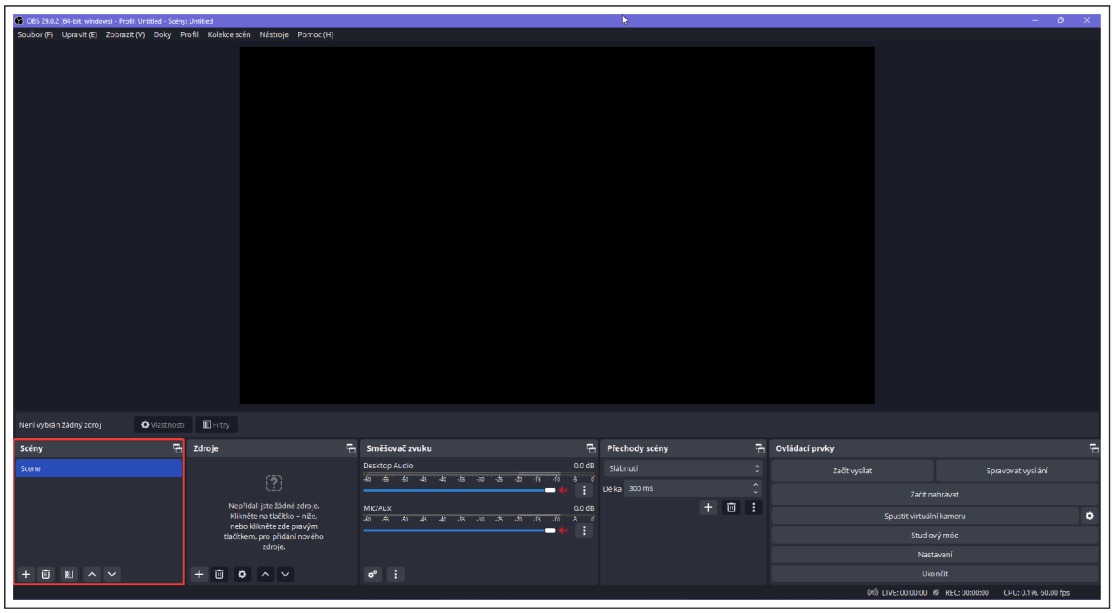
v tomto případě odpovídá platformě YouTube Live, jelikož platforma používá pro autentizaci OAuth systém Google). Po přihlášení se vrátíme na hlavní stránku aplikace a zvolíme možnost *Dashboard* v postranní navigaci. Zde je poté možné nastavit jednotlivé widgety, které by měly být poté přidány do aplikace OBS. Získat odkaz widgetu je možné kliknutím na tlačítko *Show widget* po vytvoření libovolného widgetu.

7.3 Zobrazení a použití v aplikaci OBS Studio

Všechny widgety poskytované aplikací jsou v konečném důsledku kódy jazyka HTML. Pro jejich zobrazení tedy stačí libovolný moderní prohlížeč, ve kterém stránku widgetu otevřeme.

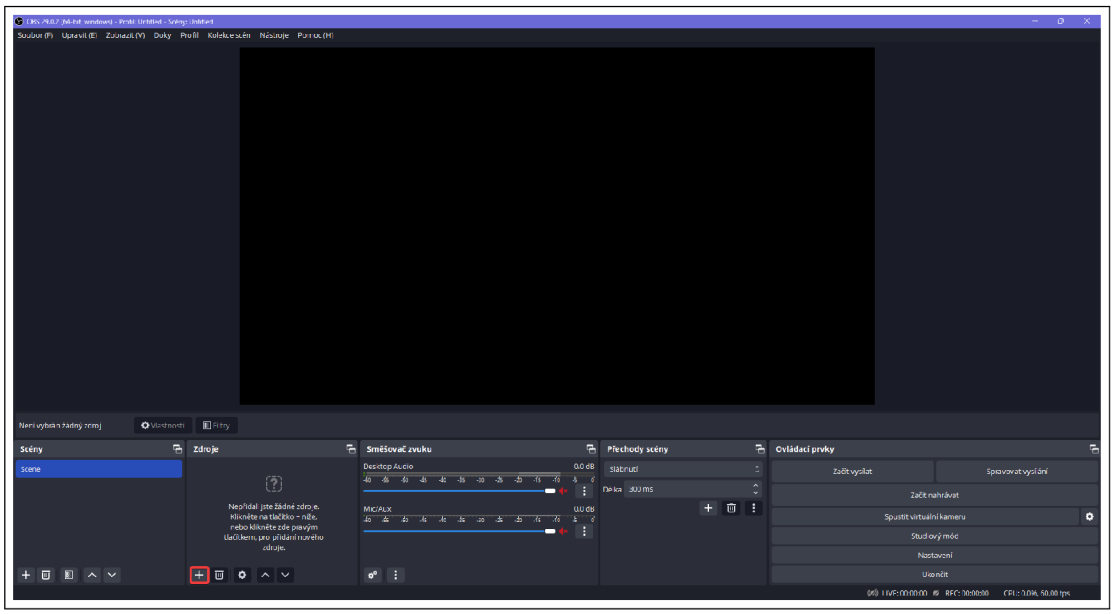
V aplikaci OBS Studio existuje zdroj zvaný *Prohlížeč*, který toto zobrazení umožňuje. K přidání tento postup:

1. Vytvoříme novou scénu v dolním panelu hlavní stránky OBS, pokud již neexistuje (obrázek 15).



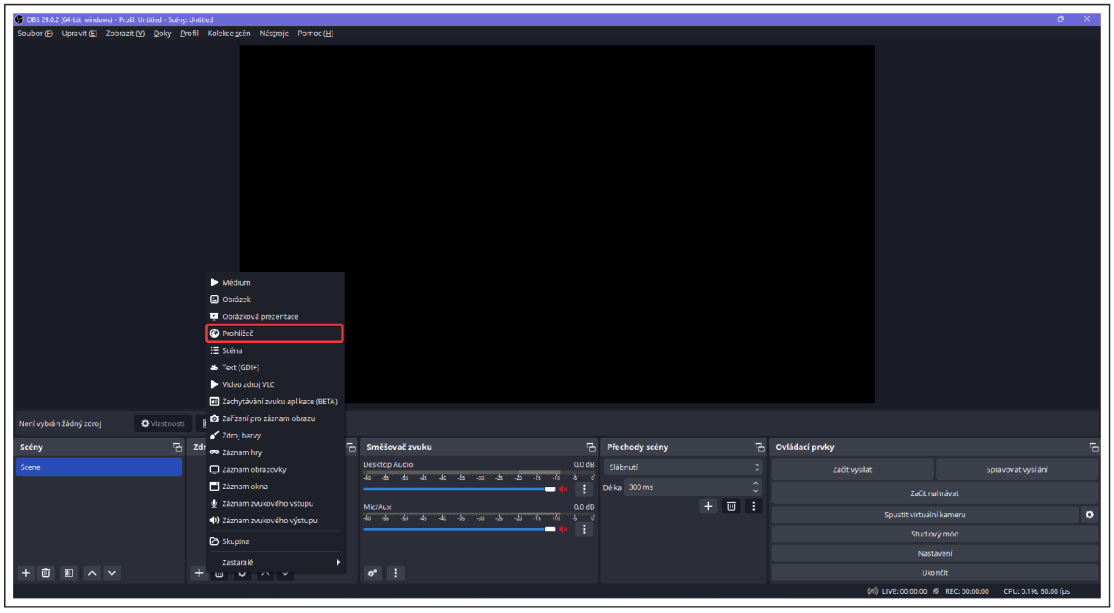
Obrázek 15: Přidání scény

2. V sekci *Zdroje* zvolíme tlačítko přidat (obrázek 16).



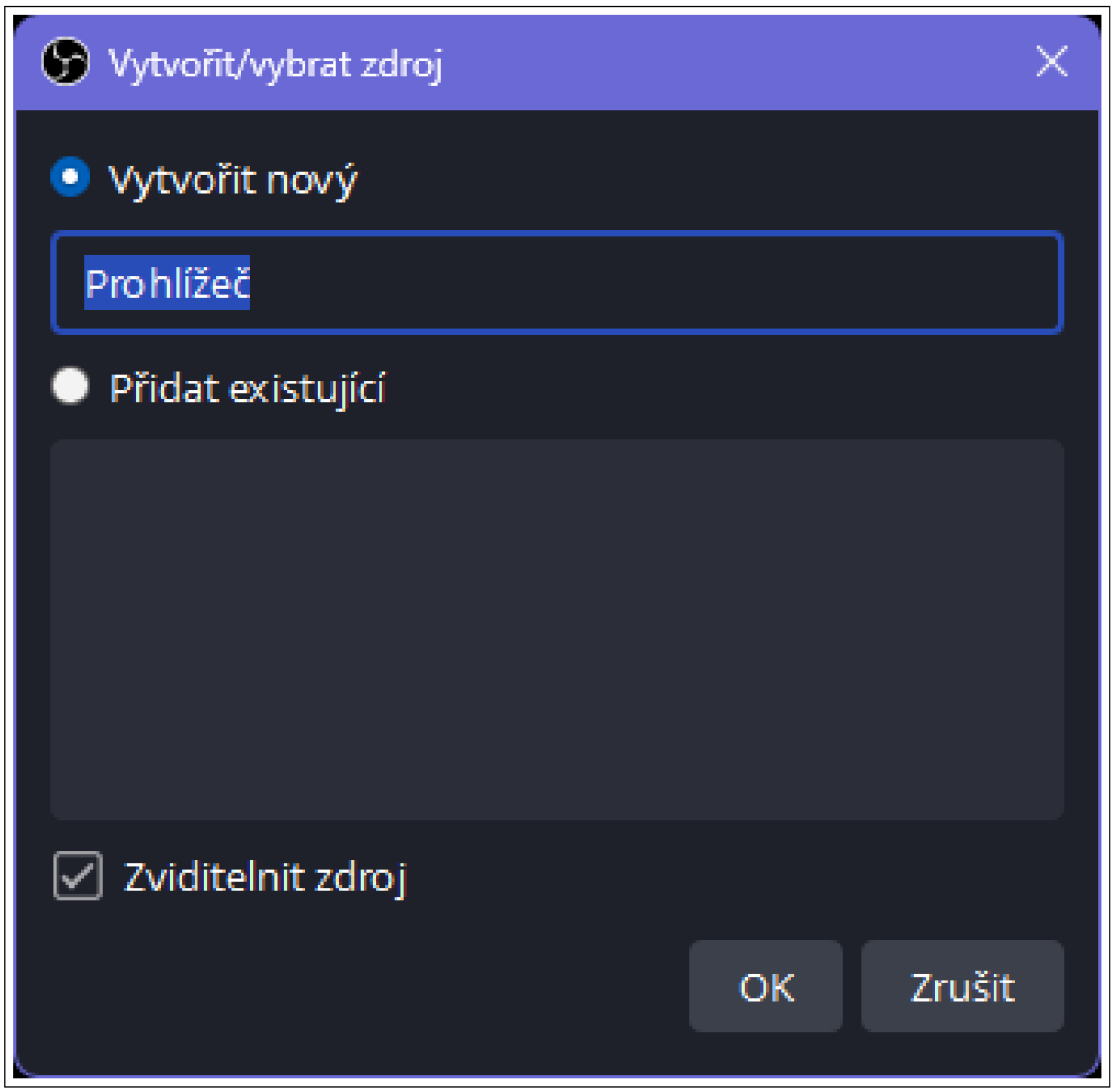
Obrázek 16: Vytvoření zdroje

3. Ze seznamu dostupných zdrojů zvolíme *Prohlížeč* (obrázek 17).



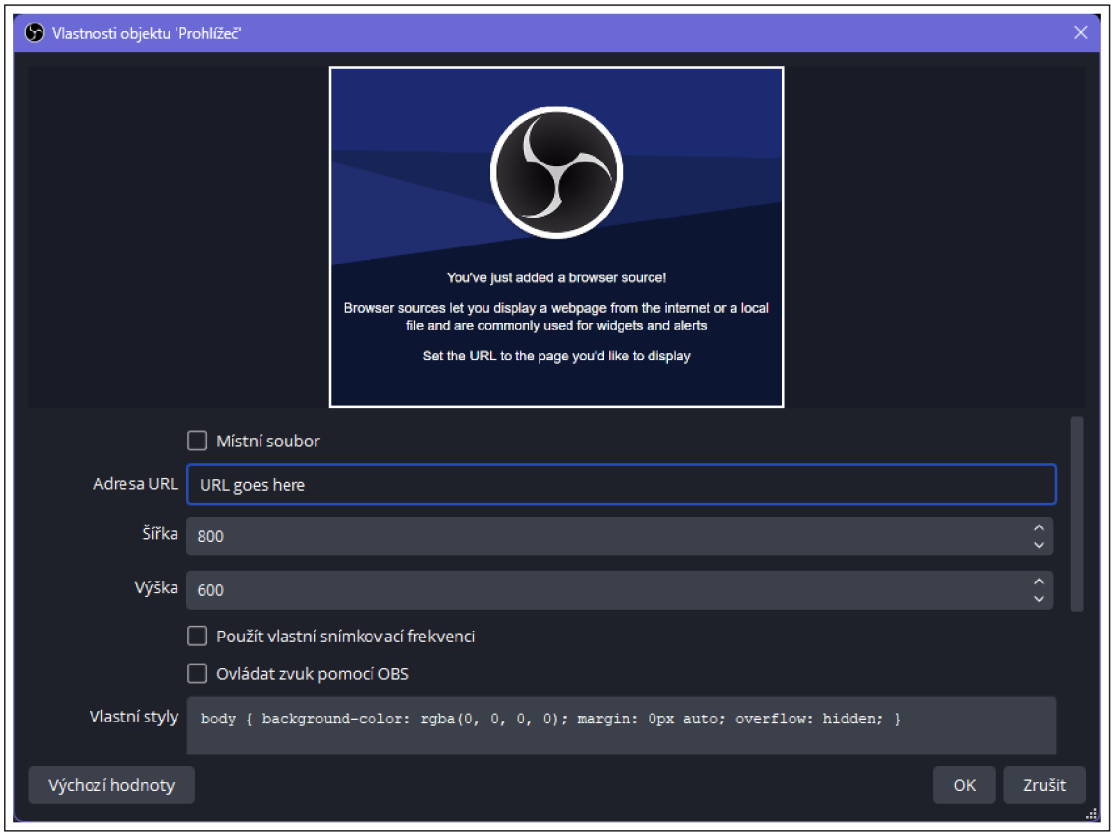
Obrázek 17: Vybrání zdroje

4. Zdroj poté pojmenujeme (obrázek 18).



Obrázek 18: Pojmenování prohlížeče

5. Zvolíme si rozlišení zdroje, jako adresu zadáme adresu našeho widgetu a nastavíme CSS kód, pokud je potřeba přepsat některou část widgetu (např. přepis fontu u elementu *, výšku řádku, maximální délku...) (obrázek 19).



Obrázek 19: Vložení URL do prohlížeče

8 Kompatibilita a rozšiřitelnost aplikace

Jak použité technologie, tak zvolená architektura ovlivňují kompatibilitu a možnost dalšího rozšíření aplikace. Ty budou v této sekci dále rozvinuty.

8.1 Kompatibilita

Aplikace se skládá ze dvou částí – první je část, ve které uživatelé mění své nastavení, vytváří widgety apod., zatímco druhou část představují samotné widgety. Obě části mají různá omezení na kompatibilitu.

Část pro správu uživatelských dat používá pro styly primárně technologii Bootstrap (viz sekce 2.10). Bootstrap poskytuje informace o tom, které prohlížeče a která zařízení jsou podporovány [21] a uživatelé aplikace by se měli ujistit, že jejich prohlížeč použitou verzi Bootstrapu (5.3) podporuje. Mezi další technologie, které omezují kompatibilitu, patří JQuery. Tato technologie také definuje, které prohlížeče jsou a nejsou podporovány [22]. Obecně je ale pravděpodobné, že tato aplikace bude fungovat korektně na většině aktuálních verzí prohlížečů stavěných na projektu *Chromium* (zde patří např. Google Chrome, Microsoft Edge, Opera).

Část, která obstarává widgety, je o něco benevolentnější s podporovanými prohlížeči, jelikož widgety musí správně fungovat v poměrně omezeném prohlížeči aplikace OBS Studio. Minimálně je nutná podpora ECMAScript 2017 a podpora jQuery. U některých widgetů je také použita CSS vlastnost *filter*, která nemusí být některými prohlížeči podporována.[23]

8.2 Rozšiřitelnost

Aplikace byla navržena tak, aby ji bylo možné jednoduše rozšířit. Jelikož aplikace používá principy *Inversion of Control* a *Dependency injection*, je velice jednoduché do aplikace přidat podporu testování. Aplikace má také připravené komponenty a částečně hotový backend pro přidání podpory více jazyků. Aplikace je o něco málo hůře rozšiřitelná, co se týče nových služeb. Po vytvoření všech tříd pro novou službu (viz implementace existujících služeb) je potřeba je přidat do služby *ListenerStartupService* (viz sekce 4.1.5).

Pro podporu získávání bodů je také potřeba rozšířit službu pro správu bodů (viz sekce 4.1.3). Aplikace je také poměrně rozšiřitelná co se týče widgetů, neboť byly předvytvořeny abstraktní třídy, poskytující základní vlastnosti, ze kterých by měly widgety dědit. K vytvoření samotných stránek pro widgety stačí použít existující widgety jako šablony a pro vytvoření základní struktury nových widgetů je doporučeno použít šablony aplikace Visual Studio. Je ovšem doporučeno těmto šablonám věnovat zvláštní pozornost, protože neobsahují žádnou ochranu v rámci přístupu uživatelů (defaultně každý uživatel vidí i widgety všech ostatních uživatelů a může je měnit). Opět je doporučeno použít již existující widgety jako referenci. Nové typy widgetu je také potřeba přidat do třídy *ValidClientFilter*, která ovládá validní widgety pro synchronizaci s aplikací. Zde stačí přidat

novou větev pro ověření, zda widget existuje (opět lze jako referenci použít již existující větve v této třídě). Nakonec je třeba přidat samotný widget, přičemž je doporučeno použít stejný systém, kterým je definováno existující Widget API (viz sekce 5). Pro tento účel je v projektu zahrnut JavaScriptový skript (component.js), který umožňuje vytvořit nového klienta SignalR připojit ho ke službě *DispatcherHub*.

Závěr

Aplikace by měla streamerům poskytovat možnost jednoduše začlenit interaktivní prvky do jejich vysílání a divákům pak příležitost aktivně se do těchto vysílání zapojit. Tuto funkcionalitu aplikace poskytuje a umožňuje jednoduchou rozšiřitelnost poskytovaných prvků a další funkce. Zprovoznění samotné aplikace není zcela triviální kvůli omezením stanovených streamovacími platformami, pro které byla aplikace vytvořena. Mezi tato omezení patří například SSL certifikát, doménová adresa, API klíče pro jednotlivé platformy atd.

Tento proces se ale netýká samotných streamerů ani diváků, ale pouze administrátorů a vývojářů, kteří chtějí aplikaci nasadit. I přes toto omezení jsem přesvědčen o tom, že aplikace disponuje funkcionalitou, kterou by měla mít. Nabízí relativně snadné použití pro koncové uživatele, a přestože ji administrátoři a vývojáři mohou ze svého hlediska hodnotit jako složitější, jistě ocení její efektivitu.

Conclusions

The application should allow streamers an easy addition of interactive elements to their streams and their viewers to then actively engage in those streams. This functionality is provided by the app and allows for further extension both in widgets and provided functionality. The application itself isn't trivial to run due to the restrictions by the streaming platforms for which this app has been created. Among these limitations are SSL certificates, domain names, API keys for individual platforms etc.

This process, however, doesn't involve individual streamers nor their viewers, but only the administrators and developers of this application. Even with the aforementioned limitations, I believe that the application provides all functionality it should. It further allows easy use by end users and even though the developers might from their point of view rate the app as more complex, they will also surely acknowledge it's efficiency.

A Obsah elektronických dat

Všechny soubory vztahující se k této práci se nachází v systému katedry.

`doc/`
: Adresář obsahující zdrojové kódy systému TeX umožňující kompilaci tohoto textu.

`src/`
: Adresář obsahující zdrojový kód aplikace. Jsou zahrnuty také soubory projektu aplikace Visual Studio pro jednodušší kompilaci.

`README.txt`
: Textový soubor obsahující návod pro spuštění a otestování aplikace.

`dist/`
: Adresář obsahující zkompilovanou verzi aplikace.

Literatura

[1] *Esports definition; meaning.* [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://www.dictionary.com/browse/esports⟩.

[2] *VALORANT, Riot games' competitive 5V5 character-based tactical shooter.* [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://playvalorant.com/⟩.

[3] *Riot games. developer of league of legends, valorant, teamfight tactics, legends of Runeterra, and wild rift. creators of arcane. home of lol and Valorant Esports.* [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://www.riotgames.com/⟩.

[4] *All valorant streams now drop closed beta access.* [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://playvalorant.com/en-us/news/announcements/all-valorant-streams-now-drop-closed-beta-access/⟩.

[5] Hameed, Alya. *Parasocial relationship* [online]. 2021 [cit. 2023-5-7]. Dostupný z: ⟨https://www.dictionary.com/e/tech-science/parasocial-relationship/⟩.

[6] Kowert, Rachel; Daniel, Emory. The one-and-a-half sided parasocial relationship: The curious case of live streaming. *Computers in Human Behavior Reports.* 2021, roč. 4, s. 100150. Dostupný také z: ⟨https://www.sciencedirect.com/science/article/pii/S2451958821000981⟩. ISSN 2451-9588.

[7] Hamilton, William A; Garretson, Oliver; Kerne, Andruid. Streaming on twitch: fostering participatory communities of play within live mixed media. In. *Proceedings of the SIGCHI conference on human factors in computing systems.* 2014, s. 1315–1324.

[8] *Dead cells' twitch integration is great.* [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://venturebeat.com/pc-gaming/dead-cells-twitch-integration-is-great/⟩.

[9] *Node.js.* [online]. [cit. 2023-5-30]. Dostupný z: ⟨https://nodejs.org/en⟩.

[10] *Duende Software.* [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://duendesoftware.com/⟩.

[11] *Overview of entity Framework Core - EF Core.* [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://learn.microsoft.com/en-us/ef/core/⟩.

[12] *Overview of ASP.NET core SIGNALR.* [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?WT.mc_id=dotnet-35129-website&view=aspnetcore-7.0⟩.

[13] *Úvod do Razor Pages v ASP.NET Core.* [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://learn.microsoft.com/cs-cz/aspnet/core/razor-pages/?view=aspnetcore-7.0&tabs=visual-studio#the-home-page⟩.

[14] *Hostování ASP.NET Core v Linuxu s Nginx*. [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://learn.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-nginx?view=aspnetcore-7.0&tabs=linux-ubuntu⟩.

[15] *Twitch Developers*. [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://dev.twitch.tv/console⟩.

[16] *Twitch Chat Password Generator*. [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://twitchapps.com/tmi/⟩.

[17] *Google Cloud console*. [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://console.cloud.google.com⟩.

[18] *Free Currency Conversion API | Freecurrencyapi.com*. [online]. [cit. 2023-7-28]. Dostupný z: ⟨https://freecurrencyapi.com/⟩.

[19] *Register | FreecurrencyAPI*. [online]. [cit. 2023-7-28]. Dostupný z: ⟨https://app.freecurrencyapi.com/register⟩.

[20] *ngrok*. [online]. [cit. 2023-5-7]. Dostupný z: ⟨https://ngrok.com/⟩.

[21] *Browsers and Devices · Bootstrap v5.3*. [online]. [cit. 2023-7-11]. Dostupný z: ⟨https://getbootstrap.com/docs/5.3/getting-started/browsers-devices/⟩.

[22] *Browser Support | jQuery*. [online]. [cit. 2023-7-13]. Dostupný z: ⟨https://jquery.com/browser-support/⟩.

[23] *filter - CSS: Cascading Style Sheets | MDN*. [online]. [cit. 2023-7-14]. Dostupný z: ⟨https://developer.mozilla.org/en-US/docs/Web/CSS/filter#browser_compatibility⟩.