Chris Yealy Daniel Muckerman Ryan Hammett

Fall 2015 Senior Design Project

TWICCIAN

A TWITCH CLIENT FOR LINUX

Twiccian is a native app written for Linux to allow the user to watch Twitch.tv streams without the use of Adobe Flash or a web browser. Twitch is a very common platform for streaming games, speedruns and more recently general creativity, but suffers from the fact that the web player and chat are built on Flash. It is well known that this has an impact on the battery life, security, and system usage of many computers, and this is the problem Twiccian tackles.

Abstract

Twiccian is a native app written for Linux to allow the user to watch Twitch.tv streams without the use of Adobe Flash or a web browser. Twitch is a very common platform for streaming games, speed runs[1] and more recently general creativity, but suffers from the fact that the web player and chat are built on Flash. It is well known that this has an impact on the battery life, security, and system usage of many computers, and this is the problem Twiccian tackles.

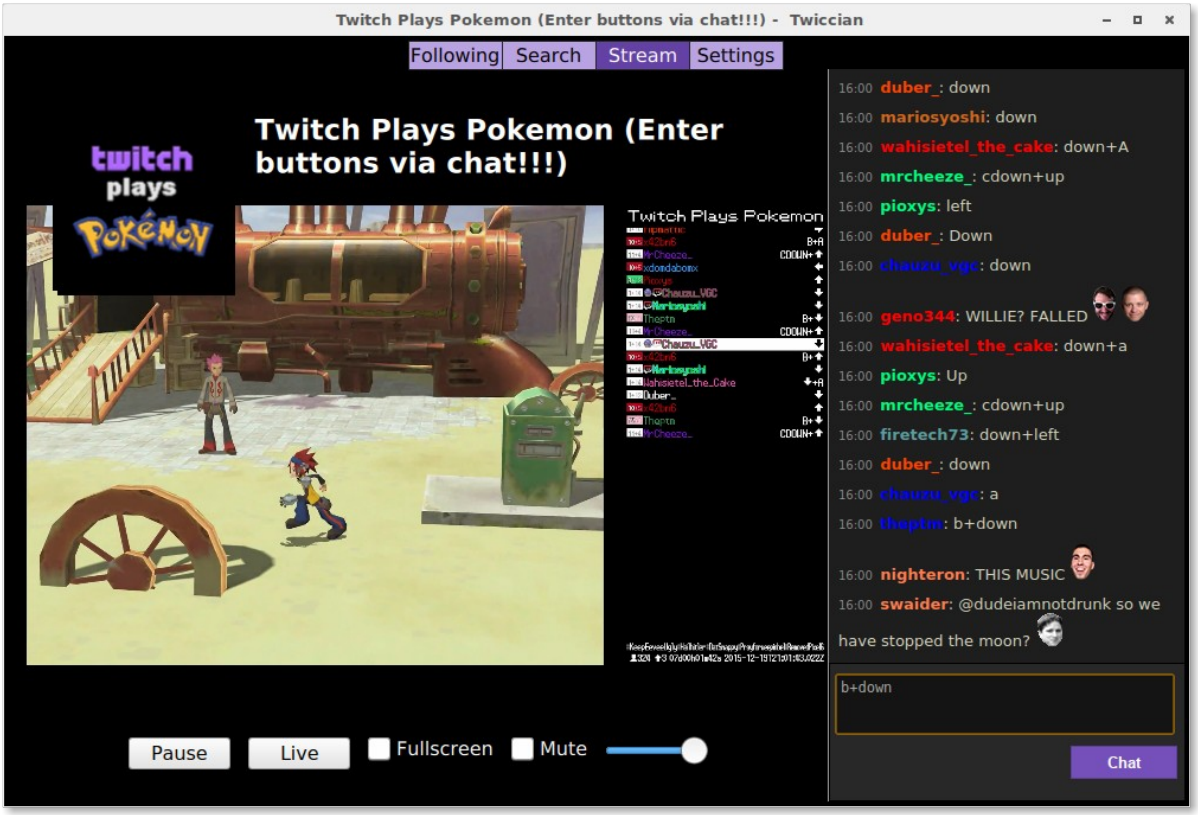

Figure 1: Twiccian viewing the twitchplayspokemon channel

[1] Beating a game to credits or completion as fast as possible.

1. Introduction

Twitch.tv[2] is a popular website with 45 million viewers a month.[3] It allows gamers, tournaments, competitions, and artists to live stream themselves and what they do for viewers to watch. There is a chat interface alongside the stream for viewers to interact with each other and the streamer. Twitch allows all of its users to access this over IRC if they would like.[4] It also provides many ways for viewers to watch streams as they have mobile applications for iOS and Android, applications for video games consoles, and the aforementioned web interface.

Until recently Twitch used Flash for their web application which is undesirable due to Flash having so many security vulnerabilities. Alongside these vulnerabilities are the strain on system resources which negatively impact battery life on laptops. During this project Twitch was in the process of switching to an HTML5 interface to address some of the issues mentioned above. Currently they only have HTML5 overlay, player controls, and chat interface as the video rendering and underlying chat framework are still done with Flash.[5] [6] This may not be an issue for much longer, as Adobe themselves have stated that they are ending Flash as it is known today.[7] It will be renamed and re-purposed into an animation suite, leaving sites like Twitch to find other alternatives to Flash for their website.

Twiccian was written to provide a native desktop experience, avoid Flash for increased performance, and to include the best features of several extensions available to users for the web interface (see Figure 1). As there are

[2] http://www.twitch.tv/
[3] "How Twitch Hooked 45 Million Viewers To Watch 13 Billion Minutes Of Gaming A Month", Fast Company.
[4] Internet Relay Chat, a text-based communication protocol.
[5] "Video Player Controls Now in HTML!", The Official Twitch Blog.
[6] "HTML5 Chat Is Live!", The Official Twitch Blog.
[7] "Adobe Flash Is Dead in Name Only", Wired.com.

no plans to continue this project, there is no intention to do much more than a few more bug fixes before making the final package to the Arch User Repository.[8] From some testing, this application can only be built under an Arch based Linux distribution. The initial goal was to have a cross-platform POSIX application, but certain libraries have made this much more difficult to port to other systems.

2. Background

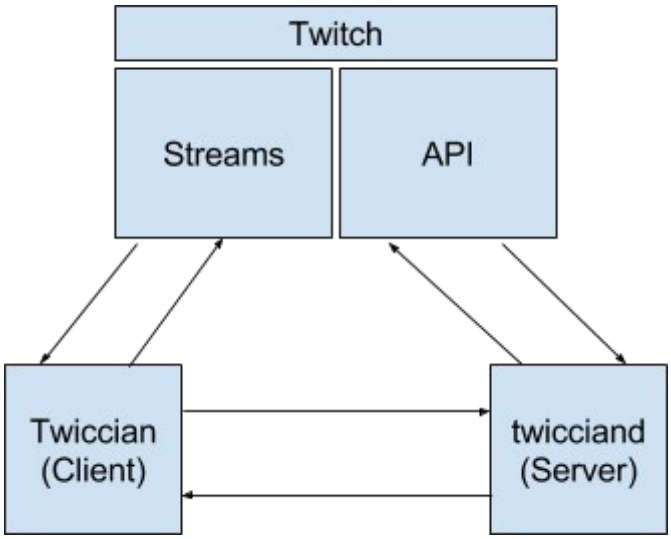

Figure 2: Diagram of Twiccian's architecture

Twiccian is not the first application of its kind, but it is the first that aimed to become a fully featured desktop application. A similar application, gnome-twitch, also provided a native Twitch experience for Linux.[9] However, gnome-twitch only allows for a small subset of the Twitch experience, such as searching and a basic stream viewer. Twiccian aims to provide a richer user experience, with support for logging into Twitch to see if the people the user

[8] https://aur.archlinux.org/packages/twiccian/
[9] Vincent, GNOME Twitch.

follow are live, include chat integration, and much more leading up to a more browser-free experience.

2.1. Application Framework

In order to create a native experience for Twitch.tv, Twiccian uses Qt[10], a cross-platform application framework. Twiccian is written in QML, a markup language that is part of Qt Quick, which is a UI creation kit for use with the Qt framework. The QML syntax is JavaScript based, and is designed to be seamlessly integrated and extended by C++.[11] This is the primary and suggested programming language to use with QML, but a variety of other languages are also available and in development.[12] C++ was chosen due to the lack of documentation for ruby-qml, which was the ideal language of choice.[13]

2.2. Video Rendering

Since Qt's media library is neither easy to work with nor widely used outside of certain applications, Twiccian uses the mpv[14] library in order to play video streams from Twitch. These streams are played in a native container for mpv to better read the stream URL. This was chosen because it supports an extremely wide variety of video, audio codecs, and file types by default, making it an ideal library for playing Twitch streams (see Figure 3). This was the first major feature implemented and has proven to be quite reliable in its processing.

[10] http://qt.io
[11] "Qt QML 5.5", Qt Documentation.
[12] "Using Qt with Alternative Programming Languages - Part 1", ICS.
[13] Ikegami, Ryohei, ruby-qml.
[14] http://mpv.io

Figure 4: Screenshot of the Fullscreen View with chat hidden

2.3. Obtaining Video Stream URLs

Since Twitch does not expose the video stream URLs in their API, Twiccian uses the youtube-dl library in order to fetch the video stream URL from a streamer's Twitch page.[15] This converts any live stream into a playable URL for the application and mpv to use as it exposes the raw video link of that which is played. From there, mpv directly streams the video stream from Twitch (see Figure 2).

2.4. Chat

Twiccian contains a daemon written in Golang for all of its chat functions (see Figure 2). The daemon, named twicciand, is designed to run in the background and connect to Twitch's chat via its IRC bridge.

[15] Garcia, Ricardo, youtube-dl.

The chat is exposed to the client using WebSockets and is implemented using HTML, CSS and JavaScript on the client-side. This provides a cross-platform solution based on WebSocket interaction for the chat.[16]

As for connecting to IRC, twicciand uses several Golang libraries: walle/cfg, gorilla/websocket, sorcix/irc and gorilla/handlers. The sorcix/irc library allows Go programs to connect to IRC and interact with it in an event-driven manner, treating messages as Go objects.[17] The gorilla/websocket and gorilla/handlers are used to allow the daemon to open and debug WebSocket connections, which is vital for connecting to the chat's client-side.[18] [19] This is also used for logging to ensure everything is working as intended. The walle/cfg library is used to store the username and OAuth token once they have gone through the authentication flow.[20] This is so they can be pulled to connect to chat, since Twitch's IRC bridge only allows authenticated users to connect.

2.5. Platform Support

Twiccian was developed for Linux desktops, but due to Qt's cross-platform nature, Twiccian can theoretically be ported to any POSIX compatible platform. Since mpv and youtube-dl are also cross-platform across POSIX compatible platforms, the app can be ported to a number of non-Windows machines. Twiccian has been tested on Arch Linux (under AntergOS) but is unable to compile on Debian/Ubuntu

[16] "Notice of Upcoming Changes: Websockets and Secure Connections", Twitch Developer Forums.
[17] Demuzere, Vic, Go irc package.
[18] Gorilla web toolkit, Gorilla WebSocket.
[19] Gorilla web toolkit, gorilla/handlers.
[20] Wallgren, Fredrik, cfg.

and OSX. The main issue that was found was the lack of developer packages for mpv outside of Arch Linux. Several Qt and rapidjson libraries are also deprecated on systems such as Linux Mint, making the porting of this application much more difficult.

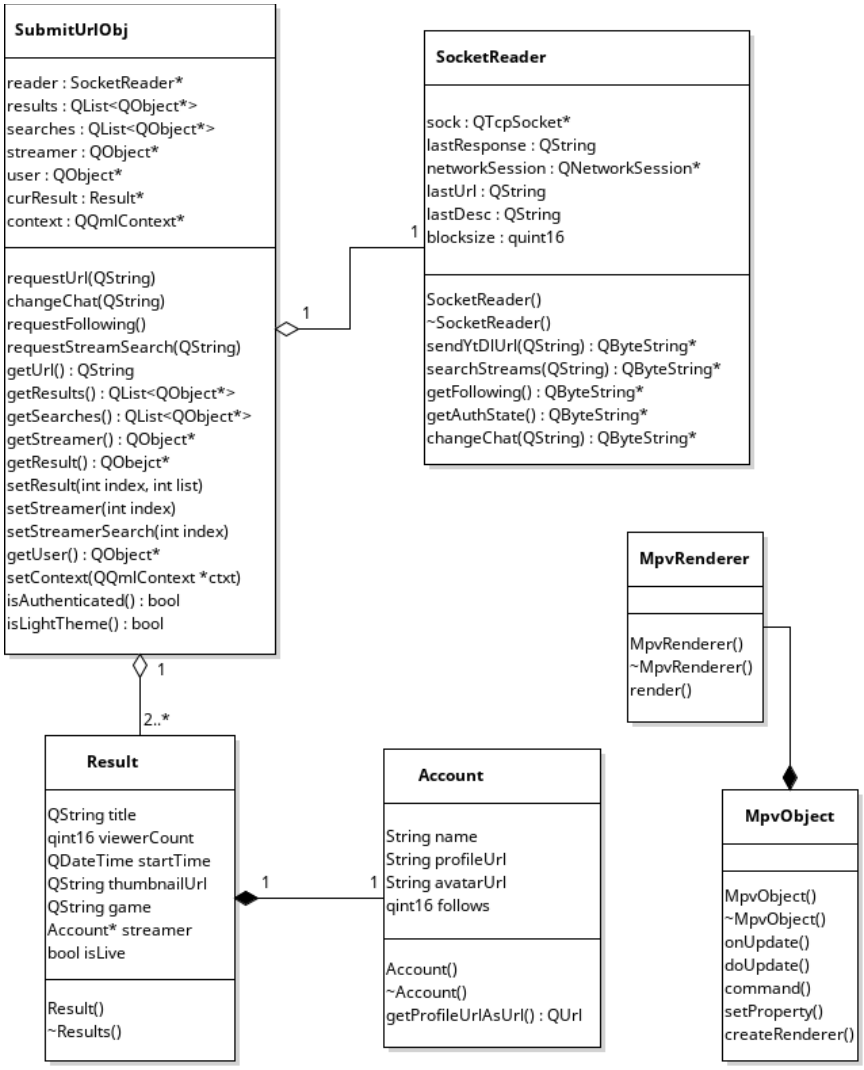

Figure 4: Complete UML diagram of the Twiccian client

3. Design

The client relies heavily on communication with the daemon to function, since most all of the networking calls are offloaded to the daemon

(see Figure 4). Thus, SubmitUrlObj is directly accessible from the QML, and heavily utilized by the application as a whole, being responsible for wrapping calls made to the daemon with SocketReader, as well as exposing data the app needs at any given moment (see Figure 5).

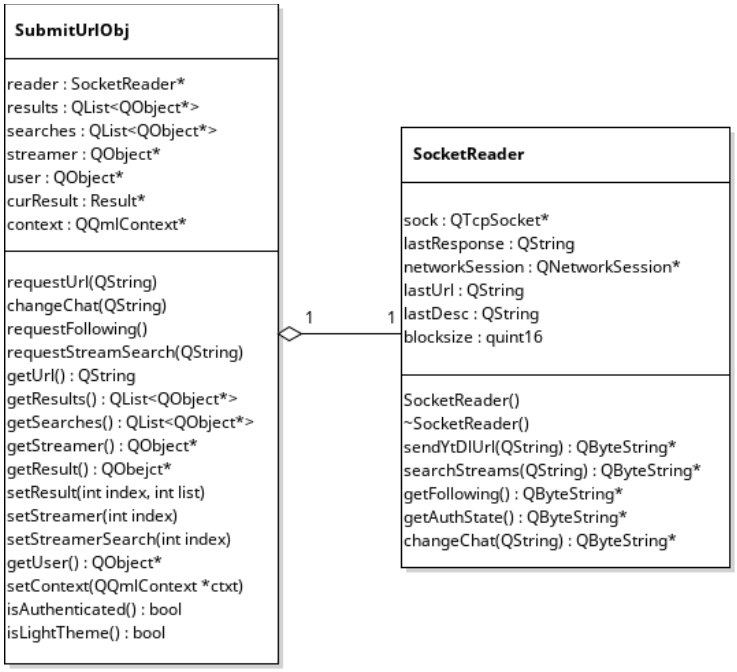

Figure 5: UML cutout of Twiccian's networking classes

Result and Account act as wrappers for data received from the Twitch API, making it easier for us to structure items like the following and search views (see Figure 4). Account is a simple wrapper for some useful and important information associated with every Twitch account. Result is used to represent both live streams the user is following and that they have searched for, since they both require basically the exact same information. An Account object is part of each Result, because it's important to be able to associate a streamer's account with whatever they're streaming at any given point in time. MpvRenderer and MpvObject are classes to wrap mpv's functionality such that

it can be easily accessed from within the application's code, allowing things like issuing commands to the mpv instance from a QML button (see Figure 4).

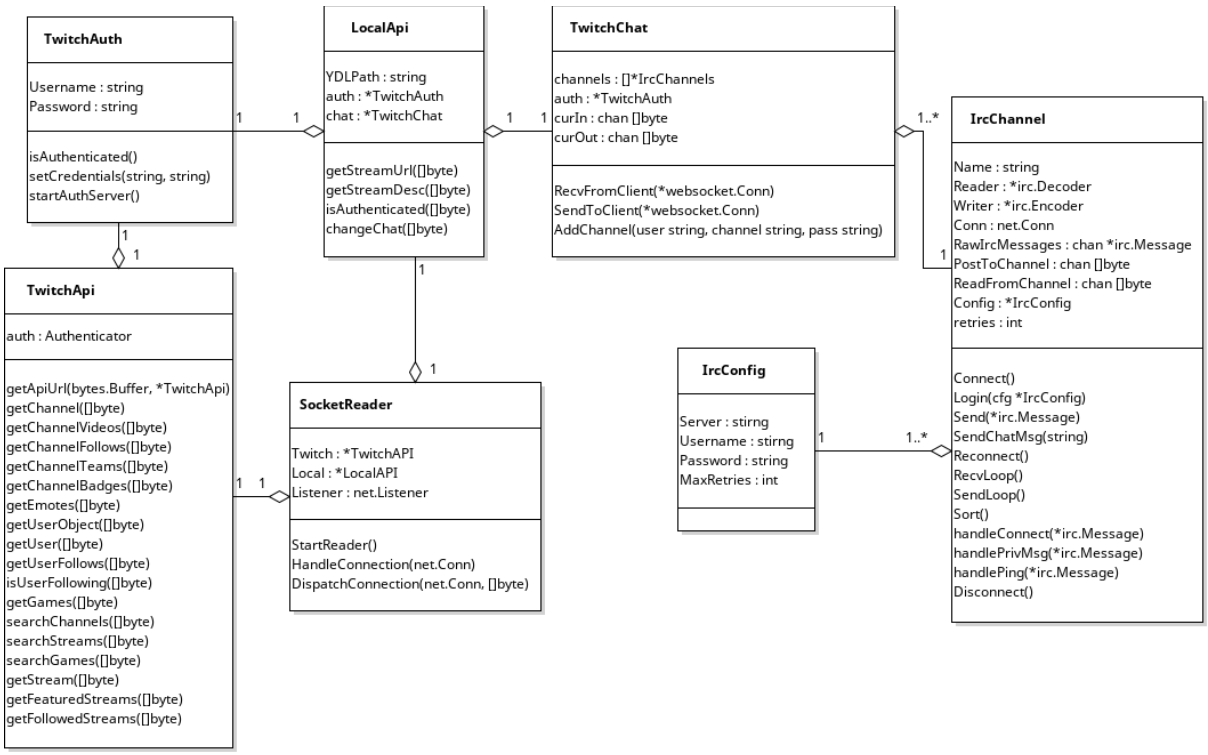

Figure 6: The complete UML diagram for the daemon, twicciand

The daemon was created to offload work from the client, and because the original design required a component to run in the background to notify the user when a streamer they follow went online (see Figure 6). The daemon subsumed responsibility for communicating with Twitch's API and interfacing with the local system. The SocketReader interface is responsible for receiving messages from the client, interpreting them, then sending the result back to the client. It uses both the TwitchApi and LocalApi objects obtain the necessary data, then serializes it as JSON before sending the data back to the client (see Figure 7).

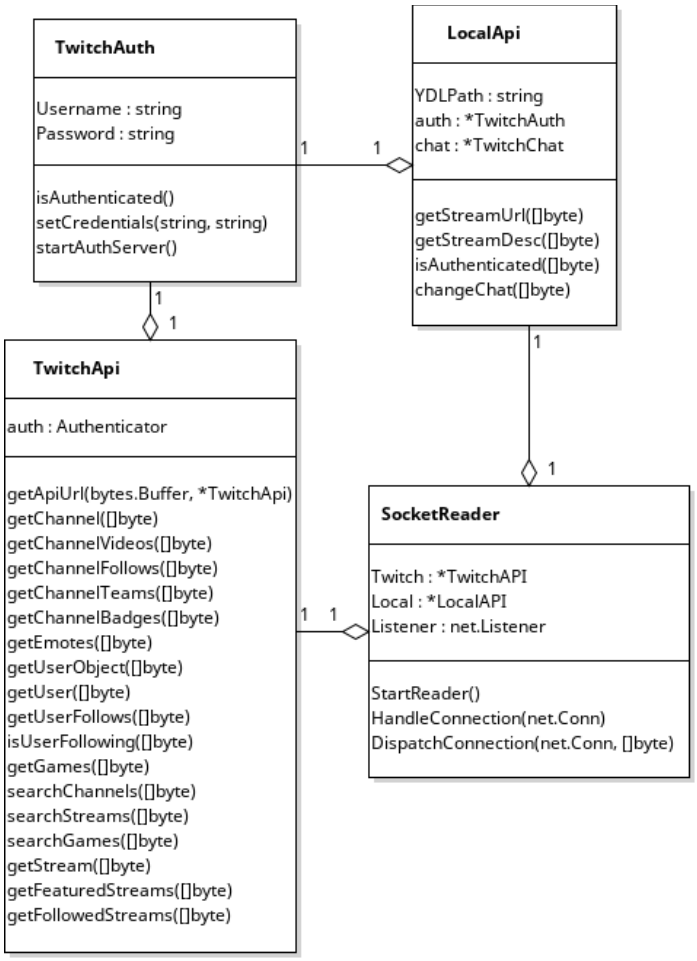

Figure 7: UML cutout of the API interface section of twicciand

The TwitchChat interface provides a straightforward way to create and maintain IRC connections to Twitch's IRC chat back end. TwitchChat sends and receives messages from the client over websockets, so the messages can be styled and rendered in HTML on the client side. TwitchChat creates IrcChannels, which are than thin wrappers around the Go IRC library twicciand uses. Each IrcChannel contains a pointer to an IrcConfig object, which store the credentials for the IRC session and could potentially store more (see Figure 8).

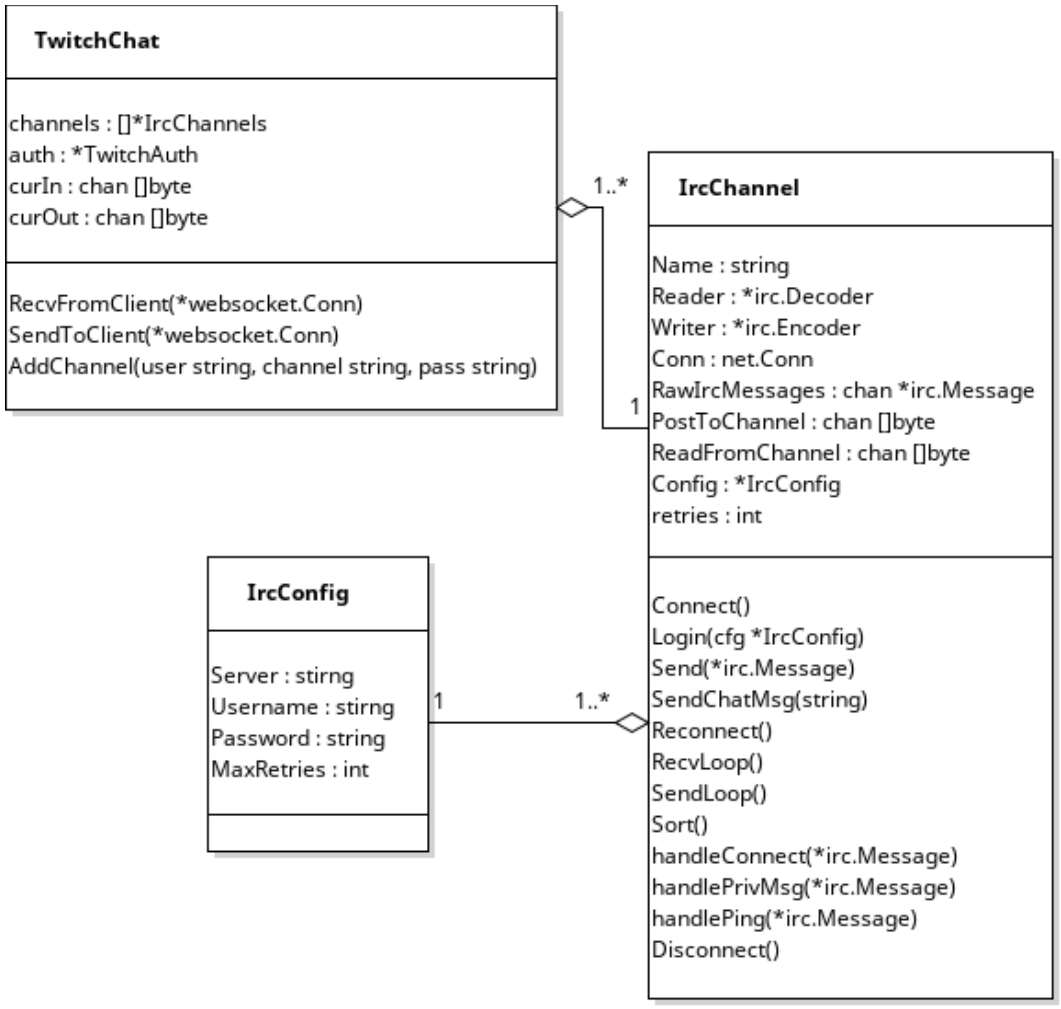

Figure 8: UML cutout of the chat section of twicciand

Finally, the TwitchAuth interface is responsible for storing the credentials to interact with the Twitch API and Twitch chat. It also creates an authentication server to receive an OAuth token from Twitch. For desktop applications Twitch implements the OAuth Implicit Authorization Flow, so as not to require the client secret which is not secure to distribute to an end user.[21] [22]

[21] "RFC 6749 - The OAuth 2.0 Authorization Framework", IEFT Tools.
[22] "Justintv/Twitch-API", GitHub.

The appearance of Twiccian is based off of the themes used on Twitch and Better Twitch TV (a browser extension).[23] Twitch's brand colors are used prominently throughout the application in order to help create a consistent style alongside Twitch's official apps.[24] The chat theme colors, however, are directly based on those of Better Twitch TV's dark and light chat themes.

4. Implementation

4.1. Team Communication

There were a variety of programs available to tackle this project, but the primary source of communication was Slack.[25] This was the easiest way to communicate to everyone on the team with multi-platform support. Slack also has apps and integration's that allow for GitHub and Google Drive support within Slack. Files were then easier to share and GitHub issues were quickly shared in the Slack as well.

4.2. Organization

Google Drive was used to house any and all files related to the project that were not necessary to stick in the repositories. Trello was initially setup to be used for keeping track of any to-do's and management of the project. This was later dropped for GitHub Issues and proper communication between members. Git, hosted through GitHub, was used for versioning and several branches were made over the course of the project to facilitate development of features without breaking the master branch.

[23] https://nightdev.com/betterttv/
[24] http://www.twitch.tv/p/brandassets
[25] http://slack.com

4.3. Workspace

All of the team members used AntergOS (using i3, Gnome 3, and Cinnamon) as the operating system environment for this project.[26] Being an Arch based Linux distribution, it was simple to obtain the many upstream packages needed to build this project. Some of the Qt libraries came pre-installed as well, allowing for a nicer environment to work with. The suggested IDE for Qt development is Qt Creator, which also includes some of the smaller Qt applications like Qt Designer.[27] It provides project support, configuration options, and a debugger which benefited the team greatly. However, it wasn't without its downsides. It was quite buggy, most of the team could not use the debugger, and the Designer refused to load if certain logic code was present in the project. Neither of these reported bugs were completely fixed before the end of the project.

The Go code was written using Vim and Emacs while it was built and deployed using the standard Go toolchain. The standard Go test was used for unit testing, but Go Convey was also used to provide automated testing of the daemon.[28]

Tmux was used to simplify work in the terminal.[29] For example, it allowed the team to run the daemon in the background while continuing to develop using the same terminal window. Qmake was used to compile the Qt application as Qt has its own Meta Object

[26] https://antergos.com/
[27] http://www.qt.io/ide/
[28] SmartyStreets, GoConvey.
[29] https://tmux.github.io/

System, and requires the developer to generate C++ code before the application can be compiled.

4.4. Application Features

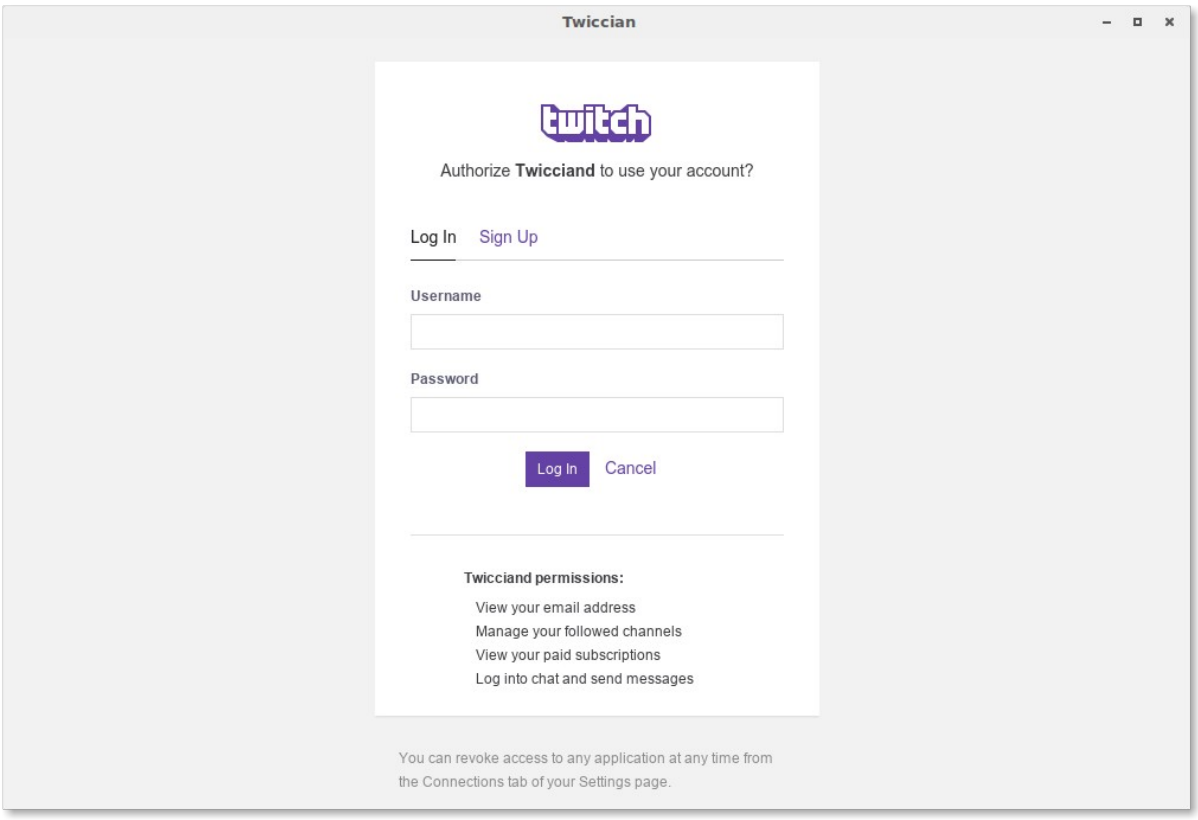

Figure 9: Screenshot of the initial Login View

Twiccian allows the user to login using the Twitch API to view their following and the chat (see Figure 9). This gives the user access to the main functions of Twitch, alongside a basic searching view. Once the user has logged into the application, it keeps the user logged in until the Qt cache or config file are removed.

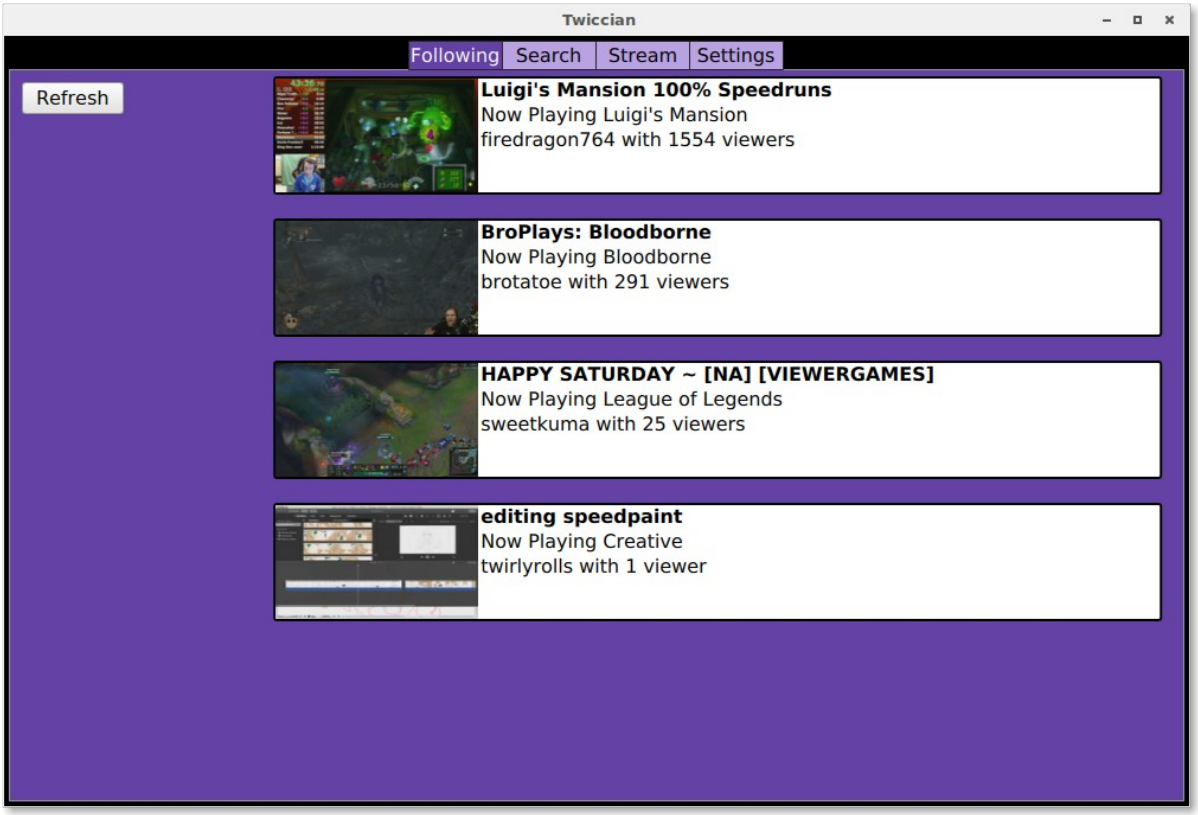

Figure 10: Screenshot of the Following View

The following view allows the user to see the streams from all the users they follow that are currently live, much like the Following page on Twitch's website (see Figure 10). It will show a recent snapshot of their stream, their title, the game they are playing, as well as the number of viewers they are viewing. There is also a [Refresh] button to update the following view with more accurate and up-to-date information. The search view allows for the user to search for a term among *every* live stream on Twitch (see Figure 11). The layout and formatting is inline with the following view, showing the user the same amount of information of another user. The search term is checked

against streamer names, stream titles and games, and returns the first 10 results.

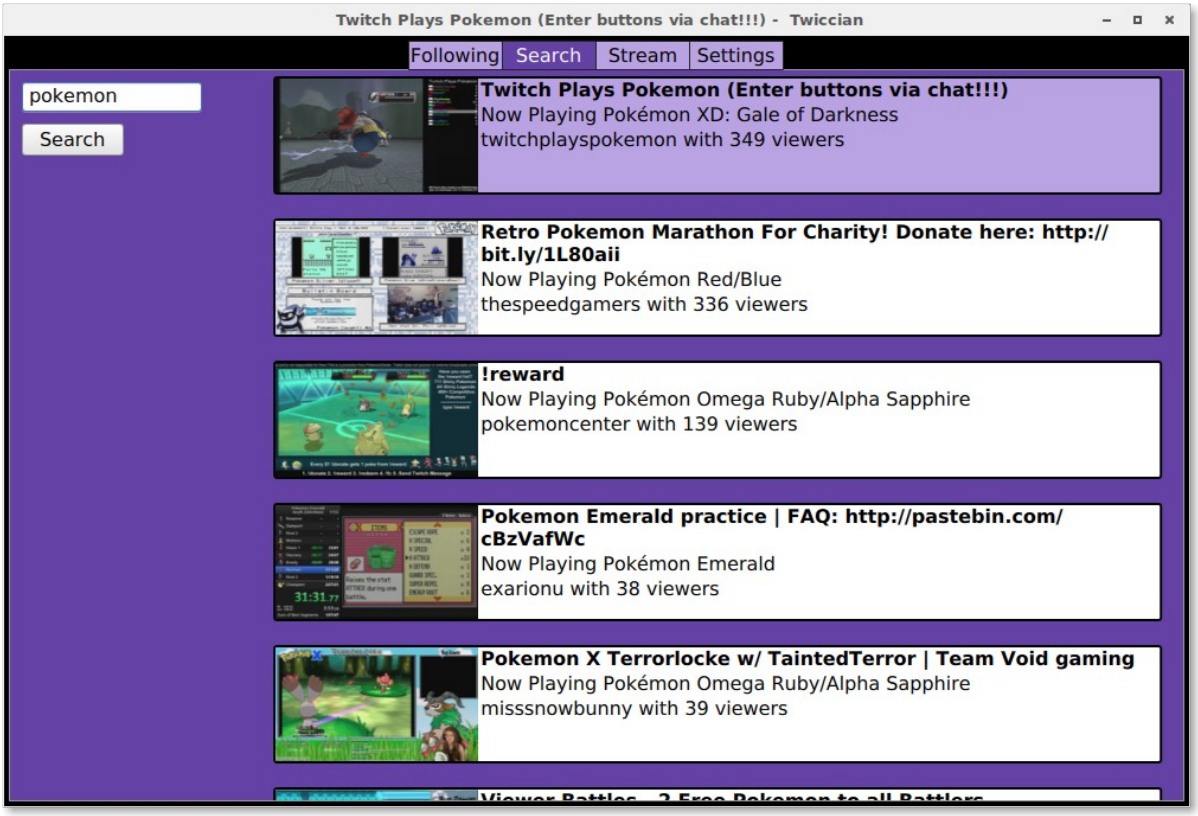

Figure 11: Screenshot of the Search View

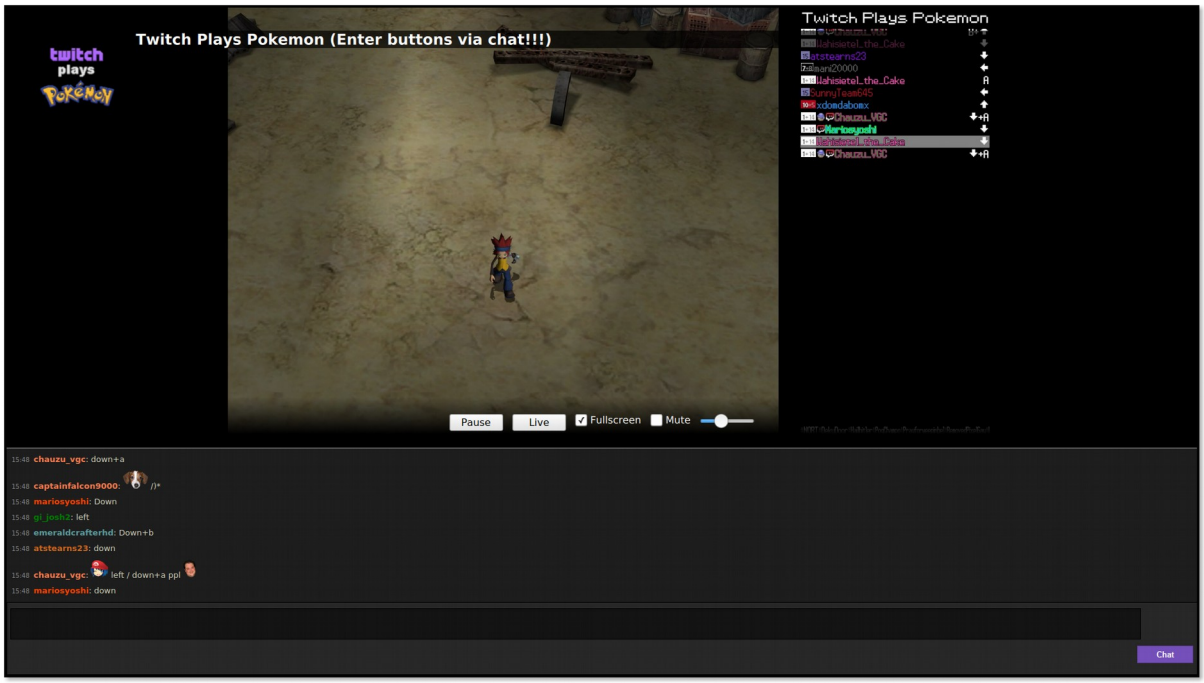

Figure 12: Screenshot of the Fullscreen View with chat

The stream view is the heart of the application, and is where the actual video player and chat views reside (see Figure 1, Figure 12). The video player has controls to pause the stream and jump to the current live point, as well as control the volume. There is also a toggle allowing the user to take the stream view fullscreen, which hides the tabs and window chrome when the app takes over the screen (see Figure 12). Overlaid over the video is the streamer's avatar and the title of the current stream. The avatar also links to the streamer's profile page, which opens an external browser due to issues with Qt's web engine. The chat view has a light and dark theme, which can be toggled from the settings view (see Figure 13), and also renders Twitch emotes, much like the official Twitch chat (see Figure 1, Figure 12). The chat can be fully collapsed by dragging the left or top edge in both the windowed

and fullscreen states of the stream view, respectively. This allows the video to take up the whole window for a more theatrical viewing experience (see Figure 3) .

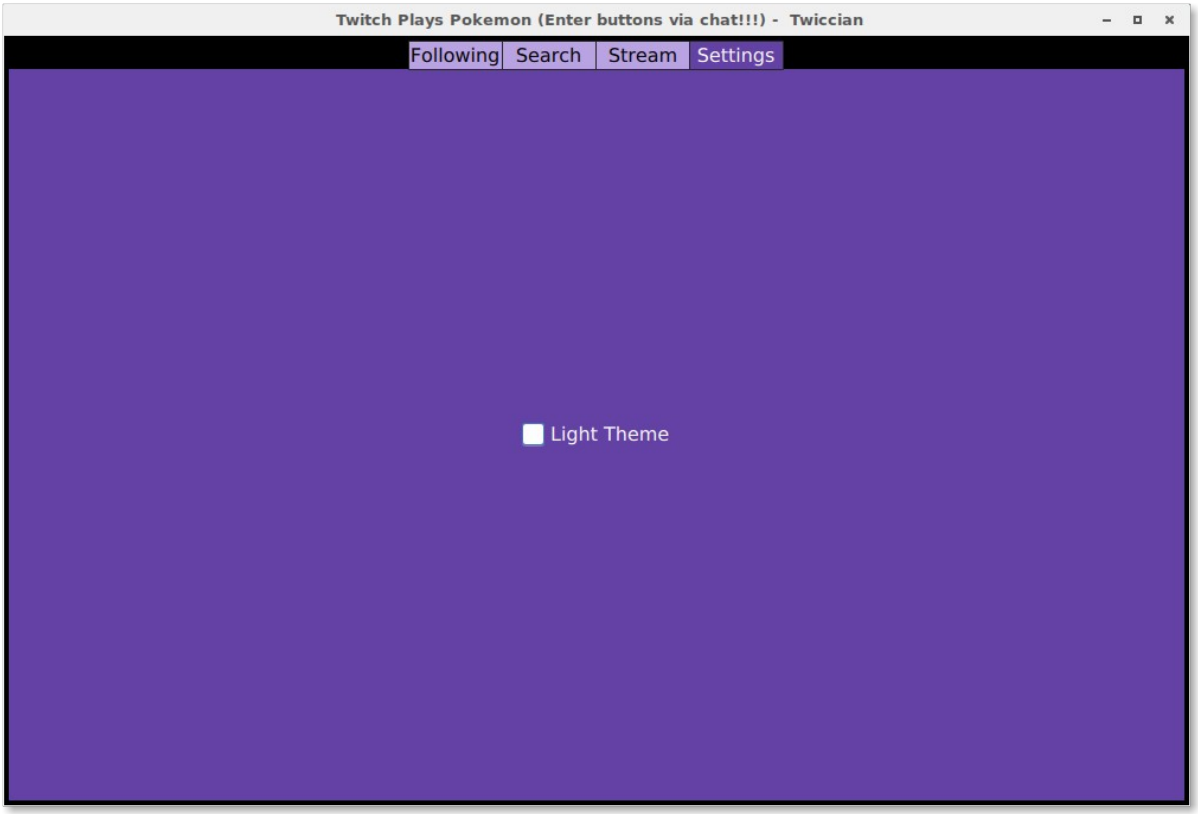

Figure 13: Screenshot of the Settings View

The settings view is powered using Qt's QSettings which allows the application to store and persist user settings.[30] This helps set the basis for future local settings, such as emoticon toggle and default window size (see Figure 13) .

[30] "QSettings Class | Qt Core 5.5", Qt Documentation.

5. Future Work

There are many features which did not make it into the final version of Twiccian due to time constraints. There are also major design flaws which make the application hard to extend or improve. One of the major changes that would be made would be using Ruby instead of C++ and GTK instead of Qt 5. Ruby or any other higher level language would have been much easier to use in developing the app. Part of the motivation for the Client-Server architecture arose from the desire to write the interface to Twitch's API in a language with an easy to use, straightforward standard library. With a more feature rich standard library, the server component may have been unnecessary by the end.

QML was chosen because it made it simple to layout the application, but made it very difficult to add any logic to the view. As an example, allowing a button to re-size the window would cause Qt Designer to decline previews due to logical statements within the button. The application can still be built with such logic involved. It also made it very difficult to render a model from C++, where it was needed to execute the networking calls. Trying to pass that model into the QML design complicated the applications architecture much more than it should have been.

Twiccian's original design encompassed other features of Twitch, such as browsing and watching past broadcasts and highlights. Due to time constraints, the final working system only implemented searching for streams and browsing the list of streamers. Other worthwhile features to add include following streamers from the search view, incorporating a native profile view for both the logged in user and the current streamer, and making the application more customizable by adding more settings.

Although this application may not be supported in the future, this final implementation is a great basis for any future work on a native desktop client for Twitch on a POSIX machine. Many of the features that have been implemented here allow for many of the missed stretch goals to become a reality. That, and with how Twitch has been growing and developing its features there is and may always be room for Twiccian to grow. Having an application with more features to implement is not always a bad thing, but this is a nice stopping point.

References

"Adobe Flash Is Dead in Name Only." *Wired.com*. Conde Nast Digital, 1 Dec. 2015. Web.

Demuzere, Vic, Go irc package, (2015), GitHub repository, https://github.com/sorcix/irc.

Garcia, Ricardo, youtube-dl, (2015), GitHub repository, https://github.com/rg3/youtube-dl/.

Gorilla web toolkit, gorilla/handlers, (2015), GitHub repository, https://github.com/gorilla/handlers.

Gorilla web toolkit, Gorilla WebSocket, (2015), GitHub repository, https://github.com/gorilla/websocket.

"How Twitch Hooked 45 Million Viewers To Watch 13 Billion Minutes Of Gaming A Month." *Fast Company*. 11 Mar. 2014. Web.

"HTML5 Chat Is Live!." *The Official Twitch Blog*. 30 June 2015. Web.

Ikegami, Ryohei, ruby-qml, (2015), GitHub repository, https://github.com/seanchas116/ruby-qml.

"Justintv/Twitch-API." *GitHub*. 31 Aug. 2015. Web.

"Notice of Upcoming Changes: Websockets and Secure Connections." *Twitch Developer Forums*. 20 May 2015. Web.

"Qt QML 5.5." *Qt Documentation*. 2015. Web.

"QSettings Class | Qt Core 5.5." *Qt Documentation*. 2015. Web.

"RFC 6749 - The OAuth 2.0 Authorization Framework." *IEFT Tools*. Oct. 2012. Web.

SmartyStreets, GoConvey, (2015), GitHub repository, https://github.com/smartystreets/goconvey.

"Using Qt with Alternative Programming Languages - Part 1." *ICS*. 19 Aug. 2015. Web.

"Video Player Controls Now in HTML!." *The Official Twitch Blog*. 22 July 2015. Web.

Vincent, GNOME Twitch, (2015), GitHub repository, https://github.com/Ippytraxx/gnome-twitch.

Wallgren, Fredrik, cfg, (2015), GitHub repository, https://github.com/walle/cfg.