INSTITUTO SUPERIOR DE TECNOLOGIAS AVANÇADAS

Relatório Projeto Final

Team NeverMind

Tomás Fernandes

Pablo Rodrigues

Cadeira de Projeto

Gonçalo Feliciano

Índice

1. Introdução

Esta ideia surge como uma solução a uma necessidade de um grupo de amigos. Surge como resposta quase imediata a um sonho: ser '*streamers*'. Este projeto resolve a parte da gestão desse mesmo sonho, criar uma '*app*' que consiga avisar aos '*followers*' de novos eventos e streams. Acreditamos que esta forma de trabalhar é inovadora porque ninguém no mundo dos '*streams*' deu-se ao trabalho de criar uma aplicação que fizesse esta própria gestão. Por outra parte, este projeto era desafiante, tínhamos que criar uma aplicação que recebesse dados doutra plataforma como a Twitch.

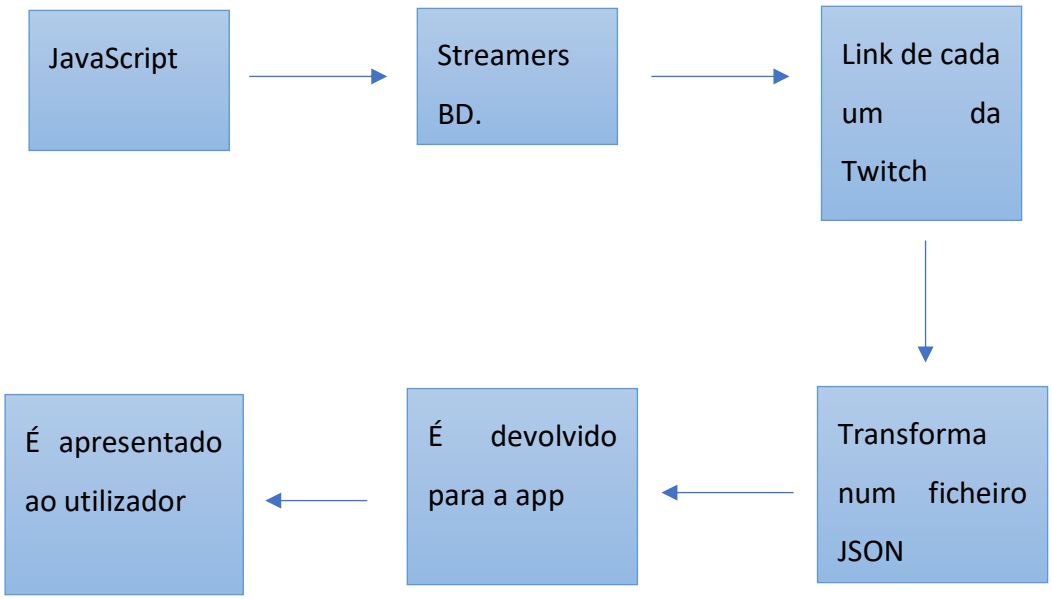
Mesmo que este projeto seja já um produto final, ainda queremos continuar a adicionar novas funcionalidades, como por exemplo, ter suporte para outras plataformas como a Youtube e, ou, outras redes sociais e criar um feed conjunto que recolha todos os dados de esta equipa numa só aplicação. Este é o nosso objetivo a longo prazo e estamos entusiasmados por consegui-lo.

Como já tínhamos dito, a finalidade deste projeto será a gestão das redes sociais e dos streams da equipa de jogadores chamada "*Team NeverMind*", tanto da parte dos seguidores como da parte das pessoas que conformam a equipa. Na parte dos seguidores, a aplicação fornecerá dados sobre '*streamings*' que estejam a acontecer ou que virão a acontecer. Por outro lado, na parte dos jogadores, vão poder dar a conhecer aos seus followers noticias e outras publicações feitas pelos próprios.

2. Desenvolvimento BackEnd: Server Side.

O primeiro passo para a realização deste projeto foi criar uma base dados em MySQL para guardar dados dos utilizadores, dados dos streamers e as notícias do feed. Esta base de dados esta num servidor, indexado a um domínio nosso privado. Depois, para ligar a base de dados com a nossa aplicação, foi criado um API em PHP que faz o export das tabelas da base de dados em ficheiro JSON para a fácil leitura da aplicação. Este tipo de ficheiros é usado para resolver os problemas de compatibilidade entre diferentes linguagens de programação. Em Java não existe nenhuma forma nativa de recolher os dados de MySQL.

O nosso ponto a seguir depois de conseguir ler o ficheiro JSON, foi obter os dados da API da Twitch, necessários para a obtenção da informação continuamente atualizada do estado dos jogadores desta equipa. Esta API foi lida a partir de JavaScript, pela facilidade de ler ciclos de repetição. Explicamos este processo com um esquema simples:



3. Desenvolvimento da aplicação: Core

Foi necessário dividir o desenvolvimento da aplicação para uma melhor organização do projeto, para isto seguimos a framework MVC (Modal View Controller). Nesta parte, temos todo o relacionado com as classes de processamento da nossa aplicação. Onde existem classes para a comunicação entre o servidor e a aplicação como por exemplo o '*Get Streamers*', '*Http Handler*' e '*PostOnServer*' e também temos a classe '*PasswordEncrypter*' que usamos para encriptar a password dos utilizadores e assim ter uma melhor segurança. Sem dúvida um *must* de qualquer projeto minimamente sério.

Na parte da comunicação entre o servidor e a aplicação, cada classe tem uma função diferente, mas ao mesmo tempo relacionada. A classe '*Get Streamers*' é uma classe assíncrona que recebe os dados da '*Live Stream*' dos streamers. O seu funcionamento está explicado no ponto anterior.

A classe '*Http Handler*' encarrega-se da comunicação no nível raiz a partir do protocolo Http, esta classe e fundamental para o funcionamento das classes '*Get Streamers*' e '*PostOnServer*'. Esta classe utiliza um URL como forma de comunicação e converte os dados para a posterior leitura.

A última das classes que se encarrega da comunicação é a classe '*PostOnServer*', que tal como a classe '*GetStreamers',* é assíncrona, já que usa outra thread para processar a informação. Isto é necessário para que a thread principal não fique a processar a informação e por tanto não ficar "entupida". Desta forma conseguimos obter uma maior fluidez na aplicação assim como não obter erros desnecessários e indesejados. Esta classe envia informação a partir de um URL ao servidor.

Para finalizar este ponto temos a classe '*PasswordEncrypter*', que como o nome indica serve para encriptar as palavras-passe dos utilizadores da aplicação. O protocolo usado para encriptar a palavra-passe é AES (Advanced Encryption Standard) também conhecido por "Rain Doll", é um dos algoritmos mais populares usados em criptografia simétrica.

4. Desenvolvimento da aplicação: Layouts

A estrutura da aplicação é formada por 3 layouts (classes) simples: a classe '*Main'*, a classe '*Feed'* e a classe '*Settings'*. No aspeto do design aplicacional tentamos ser o mais simples possíveis usando o '*Material Design'* da Google com o objetivo final de que a nossa aplicação fosse o mais *friendly-user* que conseguíssemos. Nesse sentido seguimos todos os parâmetros que a própria Google fornece na sua documentação, para evitar um sistema de layouts relacional confuso e assim ter uma aplicação limpa.

Seguindo todos estes parâmetros, o menu escolhido para a nossa aplicação, foi o *Navigation View*, já desenhado e fornecido pela API da Google e feito a medida por nós. Este menu exibe uma *side bar* com todos os layouts que temos dentro da aplicação e só se pode aceder a ele desde o nosso layout principal: '*Main'*. Isto foi assim escolhido para dar sempre a entender qual é que é o nosso layout principal.

No nosso layout principal, '*Main'*, decidimos apresentar informação geral sobre os streamers da equipa: se o streamer está online ou não, o que é que esta a jogar nesse preciso momento entre outras coisas. No layout do Feed apresentamos as notícias tais como futuros streamings, publicações… etc.

Por último temos o layout das opções, '*Settings'*, no qual temos acesso a duas opções: poupar dados móveis e ativar o modo oscuro

a. Extra: Bot Team NeverMind

Como extra deste projeto, decidimos fazer um bot para o Discord, já que o Discord é uma plataforma *Online Talking* muito utilizada pelos jogadores do mundo inteiro devido a sua *Interface* leve e a sua grande capacidade para oferecer muitas possibilidades aos seus utilizadores, entre elas: os bots. O nosso bot serve para avisar os outros Streamers quando um Streamer da equipa fica *online* no Discord. Isto facilita muito o aspeto de conseguir que os Streamers joguem juntos.

5. Conclusões

Para finalizar este relatório, concluímos que este trabalho foi extremamente enriquecedor.

Não só pelo projeto em si e a sua finalidade, que foi o que nos entusiasmou e motivou para a realização do mesmo, mas também por tudo o que nos aprendemos ao fazê-lo, desde ligar um servidor a uma aplicação, até usar uma API de outra plataforma para recolher os dados de forma continua da mesma.

Queremos destacar o quanto foi gratificante para nós a finalização deste projeto e o poder partilhá-lo de forma académica como também para esta equipa de streamers.

6. Agradecimentos

Queríamos agradecer em primeiro lugar, ao professor Gonçalo Feliciano, que nos orientou ao longo deste tempo em que fomos desenvolvendo a aplicação e nos prestou uma ajuda incalculável, que foi fundamental para a realização deste trabalho.

E em último lugar, não nos poderíamos esquecer da equipa '*Team NeverMind*' e queremos agradecer-lhes por dar um feedback continuo e orientar-nos sempre até o objetivo final do projeto.