O Mackenzie Pub

Prof. Diogo Soares'

1 Introdução

O dono de um pub gostaria de um sistema com a seguinte funcionalidade: os clientes acessam um *link* através de um QR Code e vão para uma *home page* na qual podem inserir *links* de músicas do spotify. Finalmente, as músicas são adicionadas para fila de músicas que está tocando no bar em cada noite, tornando assim a playlist de cada noite colaborativa e totalmente feita pelos usuários. 🎵🎶

Uma breve esquematização do que é requisitado neste trabalho prático é ilustrado na Figura 1.

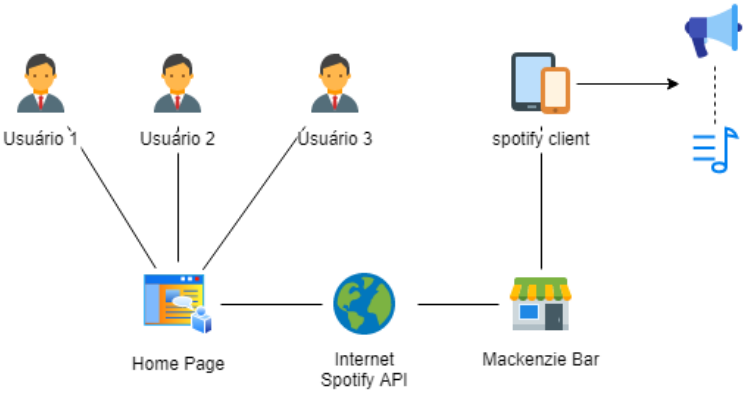

Figure 1: Esquema de funcionamento do sistema.

Cada usuário vai acessar uma home page que contenha um campo para adicionar um link de uma música do spotify (utilizando a funcionalidade de compartilhar música). Um exemplo simples pode ser visto na Figura 2.

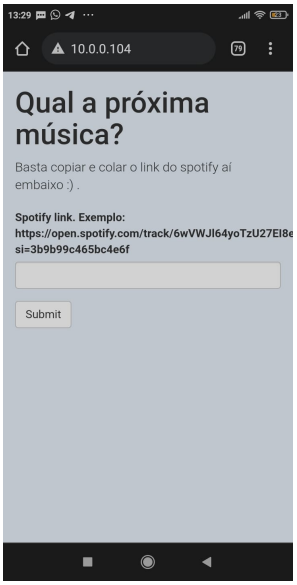

Figure 2: Exemplo de acesso à home page usando celular para a página em deploy no servidor.

O sistema de som vai ficar ligado a um computador ou dispositivo móvel com acesso a um cliente spotify já autorizado na aplicação (vejam a documentação da api para entender sobre a autorização para o usuário). Esse cliente não precisa efetivamente acessar a aplicação no frontend/backend, mas ele deve estar em um computador com acesso à Internet.

Desse modo, este trabalho visa o entendimento os princípios básicos de uma aplicação em um contexto de redes de computadores, tais como DNS, camada de aplicação, integração entre múltiplas aplicações, endereçamento IP, camada de transporte, sockets, entre outros.

2 Tarefa

O trabalho prático consiste na implementação das seguintes etapas

- **Frontend**: Um grupo de pessoas deve ser responsável pela implementação da tela que será acessível via url pelos clientes do local. A tela precisa ser informativa e possuir um campo de entrada para uma requisição HTTP POST do *link* da música do spotify que o cliente quer que vá para a fila de músicas;
- **Backend**: Um grupo de pessoas deve ser responsável pela implementação do backend da aplicação que vai ser responsável por receber os dados do front, se comunicar com a API do Spotify[2], validar e tratar os dados da entrada (de acordo com os *challenges* na Seção 3), além de atualizar a playlist do cliente spotify logado na caixinha de som. O deploy da aplicação pode ser feito em qualquer servidor em nuvem da preferência da equipe.
- **DNS**: Um grupo deve ser responsável pela configuração do servidor DNS, isto é, pela inserção dos registros A e/ou AAAA (se houver) no serviço DNS para redirecionamento. A url utilizada neste trabalho será: https://mackeziebar.xyz/. Utilizem a ferramenta dig para verificação do correto *status* do endereçamento DNS, tanto para o registro NS (nameserver), quanto para os registros A/AAAA que devem apontar para o IP da aplicação frontend em nuvem. Aqui, recomenda-se o uso da ferramenta cloudflare[1], no entanto, a escolha é aberta.
- **Relatório**: Um grupo de pessoas deve ser responsável por descrever detalhadamente cada passo usado na solução, descrição das tecnologias utilizadas, descrição da rede, fotos do experimento, vídeos da execução e como foi conduzido o experimento de teste.

- **Casos de teste**: Um grupo deve ser responsável por testar múltiplos casos e assegurar o bom funcionamento da aplicação e descrever os possíveis limites nos quais a aplicação funciona, seja por limite de usuários, limites de acesso, o que for considerado pertinente.

3 *Challenges*

Os *challenges* consistem em tarefas extras que devem ser feitas, seja na implementação ou na condução do teste experimental. Resolvam se possíveis cada um, descrevendo os passos necessários para cada *challenge*, equipe responsável e limites do que pode ser resolvido no *challenge*.

Cada *challenge* pode ser resolvido por um ou múltiplas equipes. Considere que atividades envolvendo wireshark devem ser feitas pelas pessoas do grupo de relatório e de casos de teste:

1. Como desenvolvedor da aplicação frontend, é possível modificar o protoloco de transporte da aplicação entre TCP e UDP. Qual a diferença prática entre eles? E na implementação muda muito? Qual o mais adequado?
2. Teste a aplicação frontend no wireshark, defina a troca de pacotes. Qual o fluxo de comunicação entre cliente e aplicação? Quais requisições HTTP são realizadas desde o acesso até o envio do *link* da música?
3. Verificar a possibilidade de uma solução local, isto é, apenas usuário conectados ao roteador do pub podem adicionar músicas. Perceba que nessa solução, o usuário vai acessar uma aplicação em deploy em um servidor local, no mesmo estabelecimento.
4. Teste no wireshark a comunicação do backend com o serviço de API do spotify, qual o protocolo utilizado e como é a troca de mensagens?
5. Quais as possíveis falhas de segurança e vulnerabilidades dessa aplicação? Dada a implementação final do trabalho.
6. A partir do endereço IP do cliente pedindo a música, evitar que o mesmo cliente peça mais que 5 música num intervalo menor que 10 minutos. É possível? Discuta e resolva.
7. limpe a playlist currente todo ”dia”. É possível? Quais as perspectivas da solução ideal.

4 Exemplo de Execução e Saída

Um código de prova de conceito foi feito em python + flask. Isso não significa que a solução do trabalho deva seguir esse padrão. Link: Código do protótipo.

Um vídeo de execução do exemplo pode ser visto em: Teste de experimento.

Referências

[1] Cloudflare. *Cloudflare - learning about DNS A record*. 2021. URL: https://www.cloudflare.com/pt-br/learning/dns/dns-records/dns-a-record/ (visited on 10/01/2021).

[2] Spotify. *Documentation Spotify API*. 2021. URL: https://developer.spotify.com/documentation/web-api/ (visited on 10/01/2021).