Exercício Programa 1 - Podquest

Prof. Guilherme Rey

guilherme.prey@sp.senac.br

- Data de entrega: 04 de abril de 2022, até as 23h59
- Este EP é individual
- O atraso na entrega resulta no novo limite máximo de nota. Caso você entregue um EP uma semana atrasado, pode tirar no máximo 8. Duas semanas atrasado, no máximo 6.
- Você deve entrar um único arquivo .txt no Blackboard. Neste arquivo, deve constar seu *nome completo*, *RA* e o **link para o repositório git público com sua solução**. O repositório deve conter um arquivo **README.md**, com a descrição do seu programa e como funciona a interação com ele (exemplos de comandos no menu, etc.)
- Não serão descontados pontos por possíveis vazamentos de memória, mas é legal se atentar a isso na construção. É o momento de treinar!
- Para corrigir seu EP, colocaremos o código dentro do replit e executaremos.
- A nota do EP será composta por:
  - 60% para a implementação de todas as funcionalidades obrigatórias
  - 20% pelo uso das estruturas de dados sugeridas
  - 10% para legibilidade do código e organização do exercício
  - 10% para a implementação das funcionalidades não obrigatórias

1 Podquest MVP

Nós nos juntamos para criar uma Startup, a Podquest, que terá uma solução de armazenamento de podcasts com gamification, ajudando as pessoas a aprender enquanto ficam motivadas. Para provar que sabemos o que estamos fazendo, precisamos criar um MVP (Minimum Viable Product) que tenha a capacidade de armazenar episódios de podcasts, simulando uma playlist efetivamente (note que será possível ter mais de um episódio de um mesmo podcast). Depois deste MVP vamos construir a parte do gamification, então o futuro da nossa Startup depende do seu código!

As *user stories* para o usuário, determinadas pelo nosso time de produto, são:

- Quero poder inserir um novo podcast na minha playlist, para depois poder verificar quais são os episódios de podcasts que tenho
- Quando inserir meu episódio, quero poder colocar algumas palavras-chave, pra poder buscar depois através delas
- Quero poder remover um episódio de podcast, caso eu não queira mais ouvi-lo
- Quero poder começar a tocar minha playlist e saber qual podcast está sendo reproduzido no momento, para poder verificar se quero ouvir aquele ou se passo para o próximo
- Quero poder escolher a opção "shuffle"que, ao passar para o próximo podcast, escolhe um episódio aleatório da minha playlist para tocá-lo
- Quero saber quais são e quantos são os podcasts (se tiver mais de um episódio do mesmo podcast, quero saber só o nome deste podcast) que possuo na minha playlist, para poder contar para os meus amigos

2 Especificações

Depois de obter as *user stories*, nos reunimos com nosso time e pensamos em algumas especificações técnicas, para podermos seguir com o desenvolvimento do MVP. Assim, definimos:

- Um episódio de podcast terá as seguintes informações:
  - Nome do Podcast, com no máximo 64 caracteres
  - PodcastId, um número inteiro que identifica aquele podcast
  - Nome do Episódio, com no máximo 64 caracteres
  - Um número inteiro que representará qual é o episódio
  - Uma lista com palavras-chave do episódio, cada palavra tendo no máximo 64 caracteres

Uma demonstração de um episódio que poderá ser inserido no nosso MVP é mostrada abaixo:

Figura 1: Demonstração de um episódio da playlist

Nosso time também definiu que nosso MVP pode assumir algumas premissas, desde que estejam avisadas no início do programa. Por exemplo, o MVP pode avisar no começo restrições como tamanho dos nomes, assim se o usuário tentar inserir algo fora da especificação, o problema é dele. É legal termos as validações, mas só faça se tiver tempo.

Nossa decisão de fazer o MVP utilizando a linguagem C, com um programa que funciona no console mesmo, veio top-down. O CTO é meio autoritário, mas não vamos entrar nesse mérito hoje, vamos só seguir...

3 Execução

Nosso MVP terá em sua execução os seguintes comandos obrigatórios dentro de um loop infinito:

1. **`adicionar`**: pede as informações do episódio e o insere no fim da playlist
2. **`remover`**: pede pelo id do podcast e o número do episódio para poder removê-lo da lista, avisando o usuário após a remoção ("Episódio XXX do podcast YYY removido!")
3. **`tocar`**: imprime o episódio atual ("Estou no {Episódio X} do Podcast {Y}...")
4. **`shuffle`**: ativa/desativa a opção shuffle a avisa ao usuário ("Opção shuffle ativada!", ou "Opção shuffle desativada!")
5. **`proximo`**: imprime o próximo episódio na lista. Se a opção shuffle estiver ativada, deve ser um episódio aleatório, diferente da ordem que foram inseridos (cuidado! o shuffle não pode tocar o mesmo episódio que estamos atualmente!)

Nossos possíveis investidores ficarão muito felizes se o MVP tiver as seguintes funções, não obrigatórias:

- **`adicionar`**: quando adicionarmos um episódio, o programa pode pedir algumas palavras chave. O usuário pode digitar quantas palavras vai inserir antes, não tem problema. Lembrando da especificação, cada palavra não ultrapassa o limite de 64 caracteres.
- **`relatorio`**: o programa imprime quais são os podcasts que existem na playlist, algo como "2 podcasts na sua playlist! Veja abaixo quais são:", com a lista de nomes, um por linha (lembra que é possível ter mais de um episódio do mesmo podcast?)

4 Informações importantes

Este EP deve ter, em algum lugar, o uso da estrutura de **Lista Ligada** que vimos em aula. Se você perceber que existe possibilidade de resolver o problema principal sem o uso dessa estrutura, não há problema, mas precisamos que exista o uso dessa estrutura em algum lugar da sua solução. O arquivo **leiame.txt** deve ser usado para sua explicação sobre a solução e é o espaço para qualquer amostra sobre o programa. Ali é o lugar para descrever o programa, dizer como utilizar os comandos, as limitações, etc. Happy Coding!