Universidade Federal de Viçosa – *Campus* Florestal

CCF 355 - Sistemas Distribuídos

Trabalho Prático III

Álbum de Figurinhas

Professora: : Thais Regina de Moura Braga Silva

Guilherme Aguiar Milanez - 3496

Lucas Cunha Barbosa - 3493

Florestal, MG

2020

1. Introdução

O objetivo deste trabalho consiste na implementação do programa do álbum de figurinha implementado com suporte para invocação remota de objetos. Para tal, foi implementado um banco de dados em MySQL para suportar todos os dados do sistema, um módulo para o servidor contendo a conexão com o banco de dados e um objeto que integra todas as funcionalidades do sistema, e um cliente que por meio da invocação remota de objetos, tem acesso às operações do sistema.

As seções a seguir descrevem alguns detalhes acerca das operações citadas acima, enfatizando escolhas de projeto, limitações e recursos.

1) Tecnologias Utilizadas
    a) MySQL
    b) Python
    c) pyro
    d) os (Python biblioteca)
    e) subprocess (Python biblioteca)
    f) multiprocessing (Python biblioteca)
    g) queue (Python biblioteca)
    h) signal (Python biblioteca)
    i) MySQLdb (Python biblioteca)
    j) pyngrok (Python biblioteca)
    k) json (Python biblioteca)
    l) sys (Python biblioteca)

Primeiramente é necessário instalar o pyro4 com o seguinte comando:

pip install pyro4

Para executar o projeto execute:

1) /Server/ server.py
2) copie o URI
3) /Client/ client.py
4) cole o URI
5) Use o sistema.

2. RMI

O conceito de invocação remota de métodos consiste na implementação de instanciações dos métodos de um objeto de maneira remota, em diferentes clientes. Enquanto que na versão do sistema com implementação em socket, todos os clientes tinham que implementar seus métodos e apenas acessar informações no servidor, na implementação com RMI os clientes possuem apenas uma interface de acesso aos métodos de um objeto instanciado no servidor. Tornando o sistema mais eficiente e com menos redundância.

Optou-se por implementar um único objeto em que são abrigados basicamente todas as operações do sistema, não apenas as concernentes ao álbum de figurinhas. Neste objeto todo o processo de execução do sistema é implementado, como: login, registro, acesso a informações e demais operações.

Na antiga implementação, com sockets, havia sido implementada uma API no servidor que, a partir de comandos em string, identificava qual requisição deveria ser executada. Na atual implementação a API foi refeita de na forma de um objeto em que cada método é um tipo de comando útil para o colecionador que a partir das escolhas feitas no front-end (client) são instanciadas executando uma operação.

Para utilizar a tecnologia de RMI utilizou-se neste trabalho o Pyro4 [1], biblioteca do python adaptada para suportar invocação remota de métodos.

2.1 Comunicação RMI

- Servidor - Implementação do objeto que será compartilhado entre os objetos e acessado por meio de comunicação TCP/IP:

  O comando @Pyro4.behavior recebe como parâmetro o formato de acesso que será permitido no servidor. Ao denotar "sessions", garante-se que mais de um cliente poderá acessar os serviços simultaneamente.

  @Pyro4.expose é responsável por tornar os métodos do objeto acessíveis para o cliente. Optou-se por conferir essa característica para todo o objeto e não unicamente para métodos individuais.

```
@Pyro4.behavior(instance_mode="session")
@Pyro4.expose
```

Ao instanciar um daemon e registrar um novo objeto garante-se a possibilidade do cliente acessar os métodos.

```
daemon = Pyro4.Daemon()                # Faz um Pyro daemon
uri = daemon.register(AlbumConnection)   # Registra uma instancia de objeto como  um Pyro object
```

- Cliente - Acesso aos métodos do objeto implementado no servidor:

```
if __name__ == '__main__':
    uri = input("What is the Pyro uri of the greeting object? ").strip()
    greeting_maker = Pyro4.Proxy(uri)         # obtem um Pyro proxy para o objeto
    init(greeting_maker)
```

3) Banco de Dados

Utilizou-se um banco de dados relacional (MySQL) para armazenar todas as informações usadas no sistema. Tabelas construídas:

1) Collector ⇒ Tabela usada para armazenar os dados referentes aos colecionadores. Tendo os seguintes atributos:
    a) id
    b) nome
    c) email
    d) password
    e) coins
    f) IdAlbum
2) Inventory_cards ⇒ Tabela usada para armazenar os dados referentes ao inventário dos colecionadores. Tendo os seguintes atributos:
    a) id
    b) idcard
    c) quantity
    d) idCollector

3) Card ⇒ Tabela usada para armazenar os dados referentes às cartas. Tendo os seguintes atributos:
    a) id
    b) name
    c) description
    d) picture
4) Album ⇒ Tabela usada para armazenar os dados referentes ao álbum do colecionador. Tendo os seguintes atributos:
    a) id
    b) QuantityCards
    c) QuantityStickeredCards
5) Collection_card ⇒ Tabela usada para armazenar as cartas que estão relacionadas ao álbum do jogador. Ou seja, armazenar as cartas coladas. Tendo os seguintes atributos:
    a) id
    b) idAlbum
    c) idCard
6) Store_cards ⇒ Tabela usada para armazenar os dados referentes às cartas compradas na loja. Tendo os seguintes atributos:
    a) id
    b) price
    c) name
    d) category
    e) icCollector
    f) idCards
7) Exchange ⇒ Tabela usada para armazenar os dados referentes às trocas de cartas entre os usuários. Tendo os seguintes atributos:
    a) id
    b) IdCollectorOwner
    c) IdCollecotorTarget
    d) idCard
    e) idCardReceived

4) Operações Disponíveis

Um usuário pode logar ou solicitar um cadastro no programa como primeira ação dentro do sistema. Após ser autenticado, em sua tela, será exibido um menu com todas as opções de ações disponíveis no programa. Tais como, visualizar perfil, visualizar o álbum, visualizar o inventário ou realizar alguma compra de figurinhas.

Cada uma das operações recebe como passagem de parâmetro o cliente já em execução em uma das threads. Ao ser acionada a funcionalidade envia uma mensagem para o servidor como primeira requisição. Essa mensagem é utilizada para que seja possível identificar no servidor qual informação será coletada do banco de dados e posteriormente retornada para a interface com o usuário.

Cada uma das operações implementadas pela API, realiza consultas no banco de dados retornando uma lista de dicionário com as coletas feitas no banco de dados. Essa lista então é novamente decodificada na interface do usuário para que, depois de ser tratada, o usuário possa analisar informações relevantes de acordo com a operação solicitada.

5) Recompensas

O usuário só pode comprar novas figurinhas com dinheiro em sua conta. Pensando nisso, projetou-se um sistema capaz de combinar a necessidade da movimentação de dinheiro e a usabilidade. O sistema utiliza um esquema de gamificação para recompensar o usuário da seguinte maneira: quanto mais tempo logado o usuário obtiver mais recompensas, em dinheiro, o mesmo terá. No banco de dados o usuário terá anotado o tempo de logIn e o tempo de logOut.

6) Troca

Para o sistema de troca há uma tabela especial no banco de dados usada para manter informações sobre as trocas em trânsito, negadas e aceitas. Foi implementado o seguinte esquema de troca:

1) Usuário solicita a troca ;
2) É solicitado ao usuário que escolha uma das seguintes opções:

a) Iniciar nova troca
b) Monitorar troca

3) Para iniciar uma nova troca:
    a) Usuário escolhe o id da carta que deseja disponibilizar para a transação e escolhe o id da carta que deseja receber;
    b) Usuário seleciona o id do usuário que deseja realizar a troca
    c) A troca é criada
    d) Basta que o usuário solicitado aprove ou desaprove a troca
4) Para monitorar uma nova troca:
    a) O sistema exibe os id de todas as trocas ativas
    b) O usuário seleciona um dos ids
    c) O usuário terá duas opções
        i) Caso o usuário seja o solicitante, poderá tanto cancelar a proposta de troca quanto manter.
        ii) Caso o usuário seja o solicitado, poderá tanto aprovar quanto negar a troca.

A cada passo, mensagens são trocadas entre os processos clientes e o processo servidor, todas as opções selecionadas durante a troca são enviadas para o servidor pois, majoritariamente, cada passo realiza um tipo de consulta diferente no banco de dados. Ao final o usuário terá ciência do estado final de sua solicitação.

7) Referências

[1] https://pyro4.readthedocs.io/en/stable/