Rodando o projeto

É preciso ter o Python 3 instalado, o qual pode ser baixado no link a seguir.

https://www.python.org/downloads/

Uma vez que o Python esteja instalado na máquina e acessível no *path* do sistema, abra três terminais e em todos eles acesse a pasta *src* do projeto. Feito isso, siga os passos para iniciar cada uma das aplicações.

Para carregar o server:
$ python init.py server <host> <port>

Para carregar o publisher:
$ python init.py publisher <host do server> <porta do server>

Para carregar o subscriber:
$ python init.py subscriber <host do server> <porta do server>

O servidor precisa de uma porta e de um host para expor a API. Este mesmo host e esta mesma porta deverão ser informados aos clientes, pois eles precisam saber onde encontrar o servidor.

Para iniciar a comunicação on-line entre o subscriber e o server, o subscriber precisará expor uma API, então também necessitará de um host e de uma porta (que, obviamente, não deve ser a mesma porta usada pelo servidor). Isso será explicado em mais detalhes.

Arquitetura

Os aplicativos clientes foram projetados para não ficarem acoplados ao servidor. Dessa forma é possível criar novas aplicações que consomem a API exposta pelo servidor sem impactar nos clientes atuais.

Servidor

O servidor provê recursos através de uma API exposta na rede utilizando a tecnologia RPC. Essa API foi projetada para ser *stateless*, ou seja, o servidor não guarda uma "sessão" para os usuários logados. A ideia é ser similar a uma API REST-ful.

O servidor possui duas camadas:

- Camada de dados (arquivo *server/DataBase.py*): Essa camada contém tudo que é relacionado ao banco de dados (conexão, queries etc). Não há regras de negócio nessa camada.
- Camada de negócio (arquivo *server/Server.py*): Essa camada contém a API que é exposta na rede. Todas as regras de negócio estão aqui.

Publisher

O cliente publisher (arquivo *client/Publisher.py*) é uma aplicação de linha de comando que exibe um menu ao usuário para ele navegar entre as opções. Internamente a aplicação representa uma máquina de estados, sendo que cada ação do usuário é uma transição de um estado para outro.

A finalidade dessa aplicação é criar assuntos e postagens. Sempre que um publisher cria algo, esse algo é enviado ao servidor que, por sua vez, valida as informações e armazena em disco se tudo estiver correto, ou lança uma exceção se algo estiver errado.

Subscriber

A estrutura do cliente subscriber (arquivo *client/Subscriber.py*) é muito similar à estrutura do publisher. A finalidade do subscriber é visualizar as informações criadas pelo publisher.

A única diferença estrutural entre o subscriber e o publisher é que o subscriber possui comunicação on-line com o servidor. O projeto requer que o subscriber veja os posts criados sem que o usuário precise manualmente atualizar a página. Para atender este requisito foi criada a comunicação on-line. Quando o usuário ativa a comunicação on-line no subscriber, a aplicação expõe um método na rede via RPC e informa sua localização ao servidor. Assim sendo, o servidor pode mandar novos posts ao subscriber automaticamente.

Entendendo a comunicação on-line

A comunicação on-line será explicada detalhadamente pois é a parte mais complexa do projeto.

Lado Subscriber (arquivo client/Subscriber.py)

Na linha 78 do subscriber é declarado o método *_watch_posts*, que solicita ao usuário o host e a porta a serem utilizados na comunicação on-line. Estas duas informações são enviadas ao servidor. Depois disso, a função *notify_new_post* é exposta na rede. Esta função recebe um post por parâmetro e printa ele na tela. Quando o usuário clica Ctrl+C

para encerrar a comunicação on-line, o bloco *try* é finalizado e o servidor é informado de que aquele subscriber já não está mais online.

Lado Servidor (arquivo server/Server.py)

Nas linhas 131 e 134 do servidor estão os métodos para informar que um subscriber ficou on-line e saiu do modo on-line, respectivamente.

O método *add_post*, declarado na linha 72, recebe e armazena os posts gerados pelo publisher. Neste mesmo método há uma chamada para o *_dispatch_post*, que é o responsável por sinalizar a todos os subscribers on-line que um novo post foi criado.

O método *_dispatch_post* é declarado na linha 14. Ele faz uma iteração por todos os subscribers on-line e envia o post para todos que estejam subscritos no assunto do post. Caso um subscriber não seja encontrado, ele é excluído da lista de on-line.