Atividade Final (Opcional)

Pontuação da Projeto Final: 100 pontos

08/10/2016

Controle de Versão

Versão	Data	Descrição	Responsável
1.0	08/10/2016	Criação do documento	Ari

As soluções das questões, da mesma forma que o exercício anterior, deverão ser agrupadas em um diretório e depois encaminhadas para o GutHub. Todas as Questões devem ser nomeadas de acordo com o exercício anterior (ex.: q1-node1, q1-node2, q2-node2). Para uma melhor organização, se desejar, pode ainda agrupar os projetos dentro de um diretório com o nome da questão.

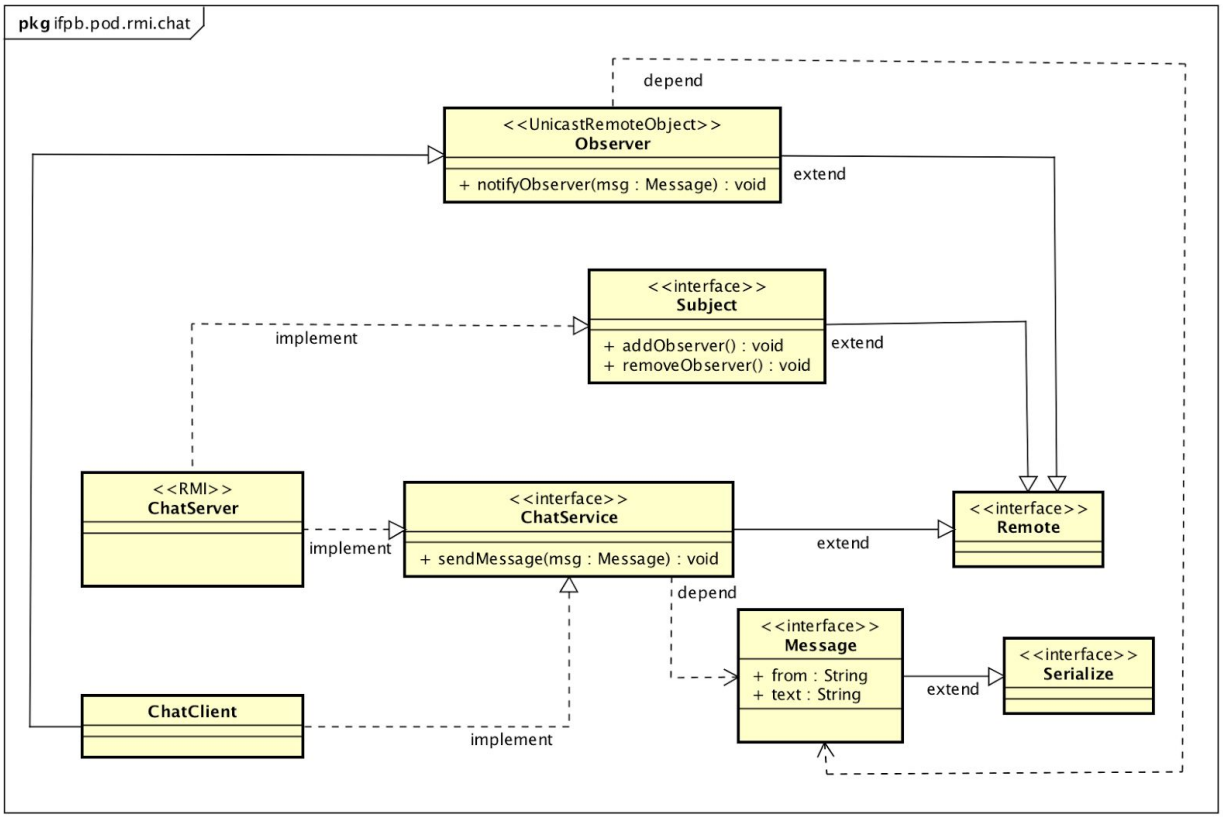
Observações:

- **Utilize apenas JDK 1.7.X**
- **Não é permitido utilizar frameworks para resolver qualquer questão**
- **Entregar até dia 11/10/2016**

1/3 - Fazendo um Observer com RMI/JRMP (20 pontos)

Desenvolva um chat utilizando o padrão observer com RMI. Este chat deverá seguir as especificações abaixo descritas.

- **Nome do pacote base**: ifpb.pod.rmi.chat
- **Quantidade de Projetos:** 3 (um para o servidor, um para o cliente e outro para os códigos compartilhados)



O código de execução do cliente deverá ser:

```
public class App {

  public static void main(String[] args) throws RemoteException, NotBoundException {
    //
    String uuid = UUID.randomUUID().toString();
    //
    Registry registry = LocateRegistry.getRegistry(10999);
    Subject subject = (Subject) registry.lookup("__ChatServer__");
    ChatService service = (ChatService) registry.lookup("__ChatServer__");
    //
    ChatClient client = new ChatClient(service);
    subject.registry(uuid, client);
    //
    Scanner scanner = new Scanner(System.in);
    while(true){
      //
      String text = scanner.nextLine();
      //
      Message message = new Message();
      message.setFrom(uuid);
      message.setText(text);
      //
      client.sendMessage(message);
    }

  }

}
```

O código de execução do servidor deverá ser:

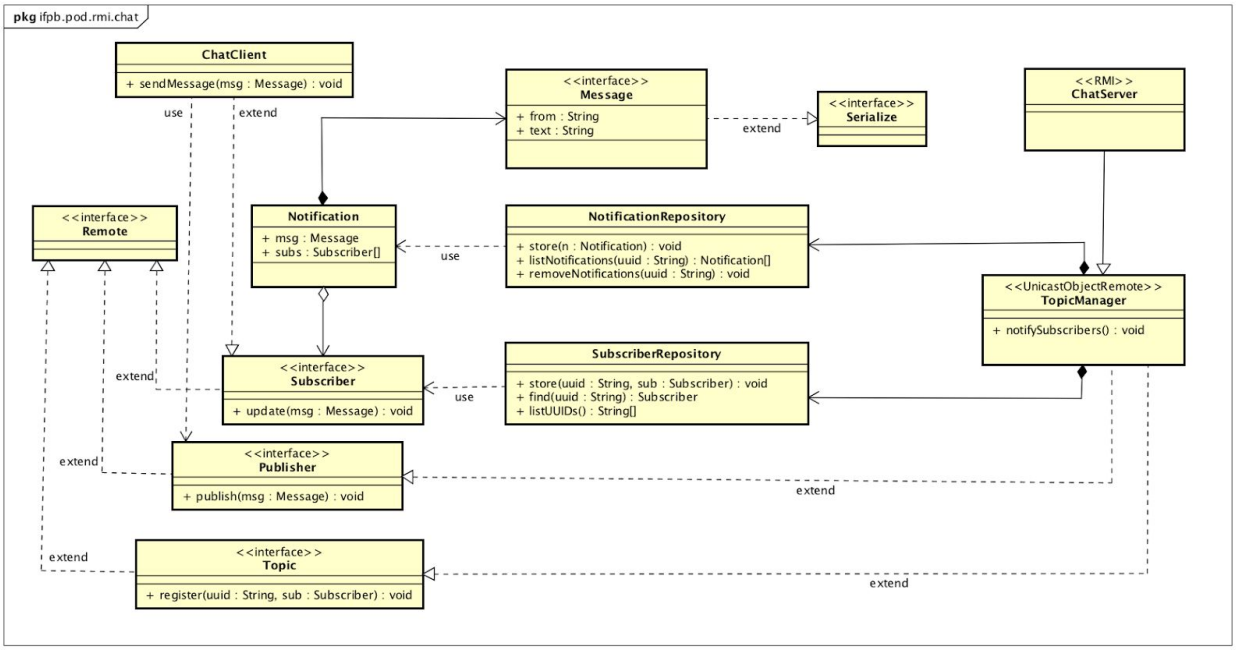
```
public class App {

  public static void main(String[] args) throws RemoteException, AlreadyBoundException {
    Registry registry = LocateRegistry.createRegistry(10999);
    registry.bind("__ChatServer__", new ChatServer());
  }

}
```

2/3 - Garantia de Entrega de Mensagem com RMI/JRMP (50 pontos)

Adotando o padrão Publisher/Subscriber, construa um chat que garanta a entrega da mensagem para todos os inscritos em um tópico chamado "chatgroup". Para tanto adote RMI/JRMP como teconologia para implementação das aplicações cliente-servidor.

Especificações:

- Utilize PostgreSQL como SGBD
- Utilize a arquitetura abaixo descrita
- Todo cliente deverá ser identificado pelo seu email
- A mensagem deverá ser armazenada por tempo ilimitado
- Os códigos de execução do cliente e do servidor derão seguir conforme indicado abaixo



Informações sobre a modelagem dos elementos da arquitetura:

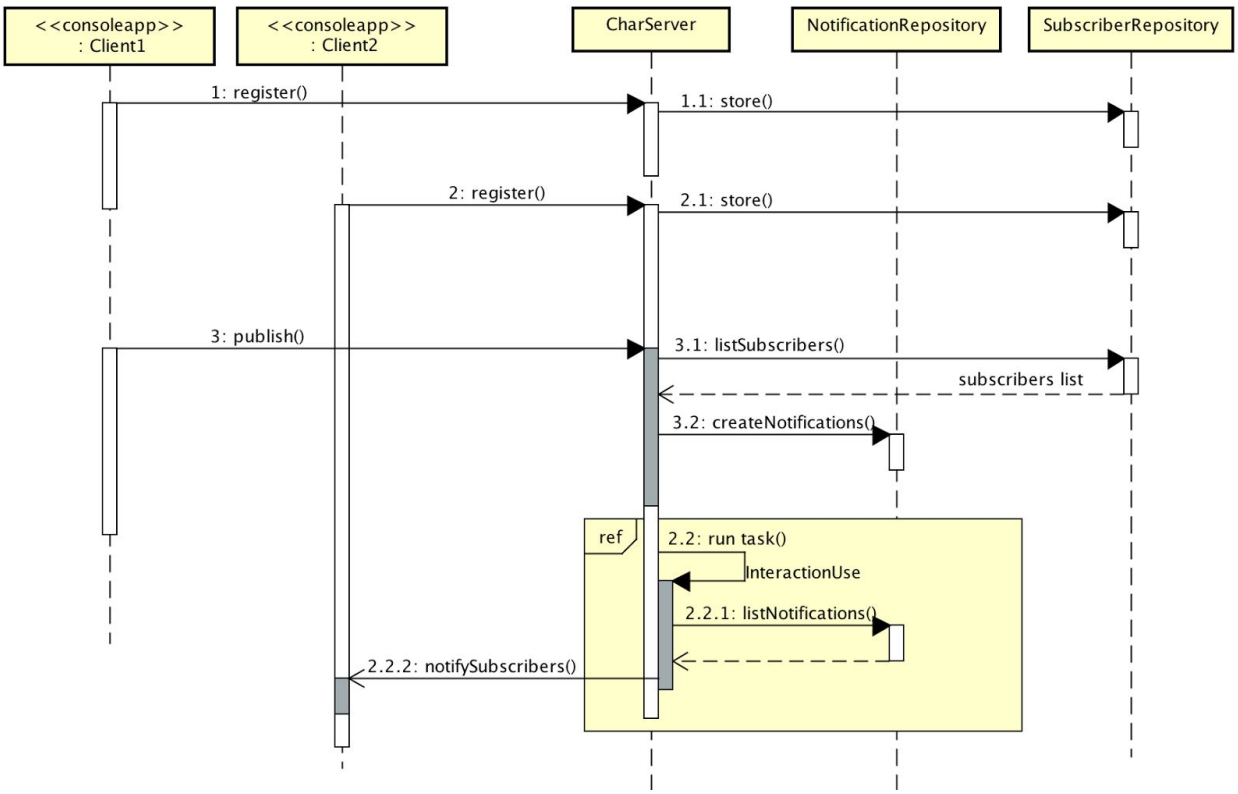
- Tanto **NotificationRepository** quanto **SubscriberRepository** deverão fazer uso de JDBC para amazenar e recuperar dados.
- **TopicManager** pode ser uma classe abstrata que é implementada por ChatServer, o qual deverá expor as interfaces as quais implementa **Publisher** e **Topic**.
- A notificação de um cliente (Subscriber) deverá ser feita a partir de uma rotina de atualização do lado do servidor, conforme exposto no código abaixo:

```
public class App {

  public static void main(String[] args) throws RemoteException, AlreadyBoundException {
    //
    final TopicManager manager = new TopicManagerImpl();
    //
    Registry registry = LocateRegistry.createRegistry(10999);
    registry.bind("__ChatServer__", manager);
    //
    Timer timer = new Timer();
    timer.schedule(new TimerTask() {
      @Override
      public void run() {
        try {
          manager.notifySubscribers();
        } catch (RemoteException e) {
          e.printStackTrace();
        }
      }
    }, 1000, 10000);//1s, 10s
  }

}
```

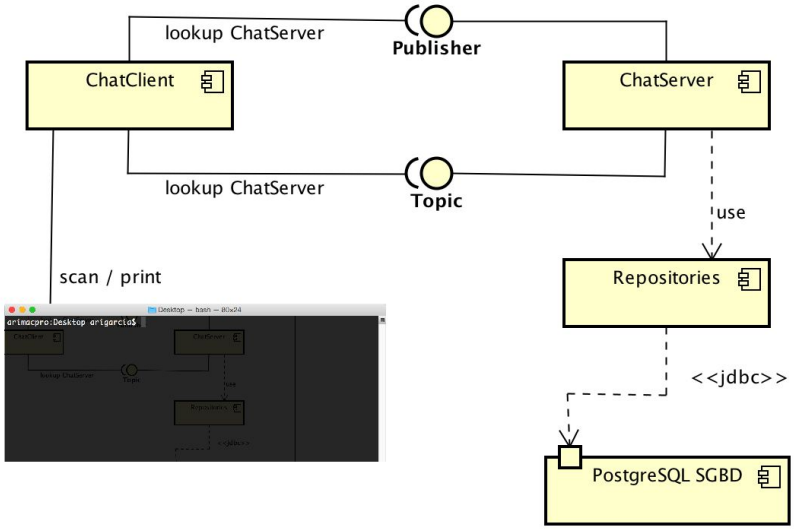
- O executor do lado do cliente deverá seguir o seguinte código:

```
public class App {

  public static void main(String[] args) throws RemoteException, NotBoundException {
    //
    String uuid = "<<seu email>>";
    //
    Registry registry = LocateRegistry.getRegistry(10999);
    Topic topic = (Topic) registry.lookup("__ChatServer__");
    Publisher publisher = (Publisher) registry.lookup("__ChatServer__");
    //
    ChatClientImpl client = new ChatClientImpl(publisher);
    topic.register(uuid, client);
    //
    Scanner scanner = new Scanner(System.in);
    while(true){
      //
      String text = scanner.nextLine();
      //
      Message message = new Message();
      message.setFrom(uuid);
      message.setText(text);
      //
      client.sendMessage(message);
    }

  }

}
```

- O fluxo de execução de um registro e um envio de mensagem para um cliente deverá ser o seguinte:



- Componentes da arquitetura:



3/3 - Garantia de Entrega de Mensagem com RMI/HTTP (30 pontos)

Implementa a aplicação servidora da questão 02 no Heroku e com base nisto altere o código do cliente para suportar tal alteração.

Especificações adicionais:

- Adote Jetty ou Tomcat como servidor embarcado no Heroku

Bom trabalho para todos!

Email para envio do link do github: aristofânio@hotmail.com