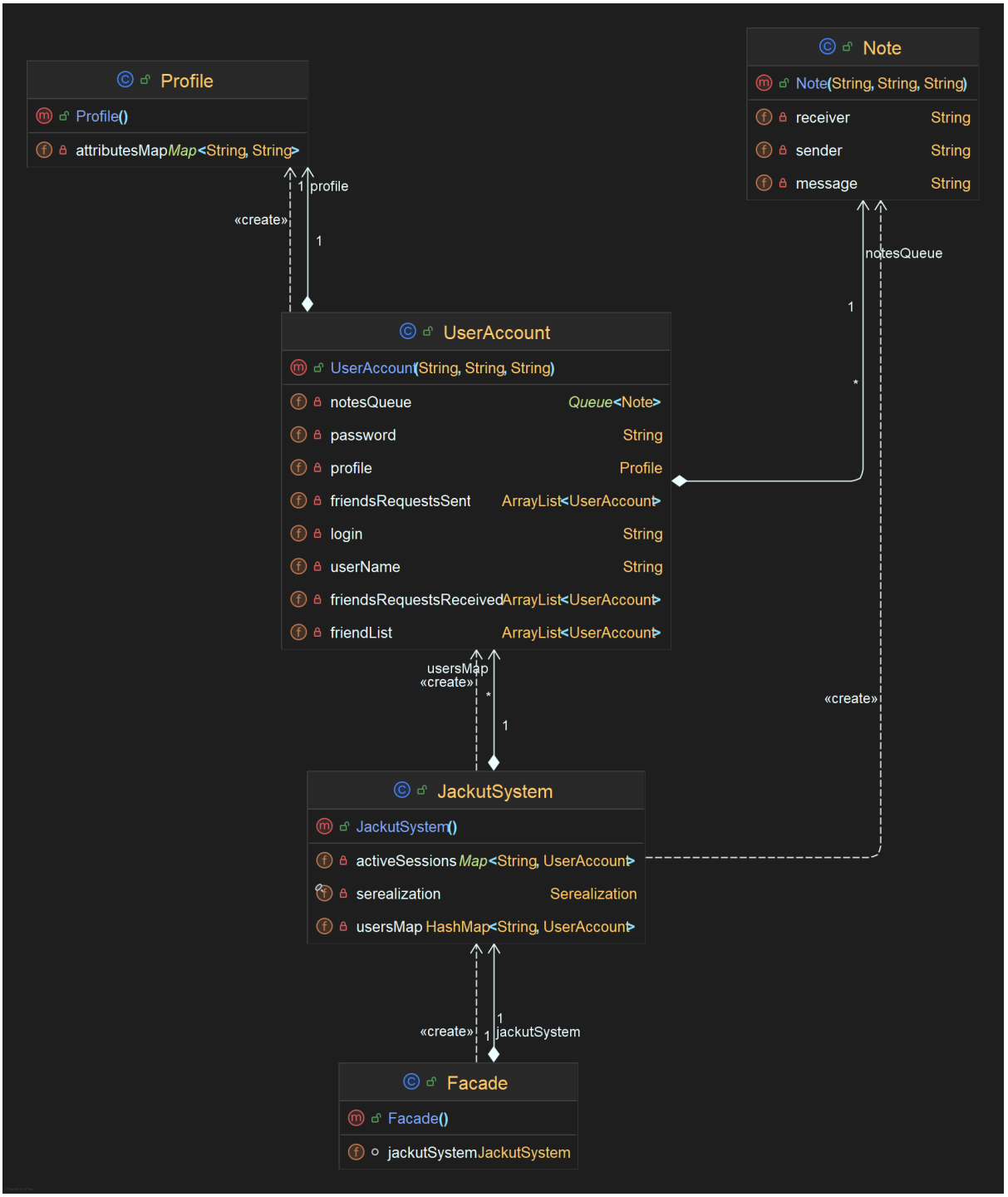
# Relatório - Jackut

Aluno: Jhordan Borges Almeida de Lacerda

Diagrama UML:



# No Jackut criei 5 classes principais:

- **Facade:**

  Facade é a classe que irá fazer a comunicação com o easyAccept, contendo todos os métodos que o easyAccept irá testar. Os métodos da classe Facade apenas chamam os métodos das outras classes que irão executar aquela ação, livrando ela da implementação de qualquer regra de negócio.

- **JackutSystem:**

  *JackutSystem* é o coração da aplicação, é onde todas as regras de negócio estão e onde as sessões de usuários logados e a lista de usuários cadastrados são gerenciadas. Para essas variáveis citadas anteriormente, todas elas são armazenadas em HashMaps, pois utilizo seus valores chaves como *login* e *id* para localizarmos determinada *UserAccount*.

  Pode-se destacar o construtor dessa classe que faz a persistência de dados acontecer, uma vez que sempre que o sistema é desligado precisamos recuperar os dados antes salvos e ela faz isso através da função *readData*.

  Podemos destacar também o método *openSession*, responsável por atribuir a um usuário que foi devidamente logado, o seu id de sessão.

- **UserAccount:**

  A classe *UserAccount* é responsável por guardar todas as informações relacionadas a um usuário, como: login, senha, nome de usuário, seus atributos de perfil, lista de amigos, lista de requisições enviadas e recebidas de amizade e a fila de recados.

  Vale dar um destaque para o **Profile** que é instanciado dentro da classe e é responsável por guardar quaisquer atributos dados ao usuário dessa conta.

- **Profile:**

  A classe Profile é utilizada pelo UserAccount, e é responsável por guardar seus atributos, que, pela estrutura solicitada, podem ter qualquer chave e valor, logo utilizei um *HashMap* para armazenar tais chaves e valores. Portanto, foi criada uma classe para gerenciar essa relação entre os atributos e o usuário.

- **Note:**

  A classe Note(recado) é utilizada pela classe UserAccount para representar recados enviados e armazenados em uma fila, onde cada recado possui um remetente, um destinatário e o conteúdo do recado.

  Eu acabei escolhendo a estrutura de dados de fila pois, a partir do momento em que percebi que o easyAccept adiciona e lê recados de forma, que, o primeiro adicionado é o primeiro a ser lido, o que caracteriza o padrão FIFO. Portanto escolhi usar a Queue para armazenar os recados do usuário.

## Classes utilitárias (utils):

As classes utilitárias que estão na pasta utils foram criadas para tirar certas responsabilidades da classe *JackutSystem*, como formatar o print de um Arraylist e realizar a serialização e desserialização.

A classe *UtilsString* serve para formatar um Arraylist para que a saída seja do tipo {valor,valor}, onde várias strings são colocadas dentro de chaves e separadas por vírgulas sem espaços entre eles

A classe *Serealization* é responsável por fazer a serialização e desserialização utilizando os métodos: serealizeObject que recebe um HashMap das contas a serem serializadas e um nome de arquivo para nomear o arquivo criado e o deserealizeObject que recebe apenas o nome do arquivo como entrada e desserializa o arquivo trazendo de volta os dados.

## Exceções:

As exceções criadas nesse projeto são bem simples. São herdadas da classe RuntimeException e realização a exceção que lhe é atribuída.