Universidade Federal de Campina Grande
Programação II/Laboratório de programação II
Centro de Engenharia Elétrica e Informática
Docente: Francisco de Oliveira Neto
Discente: Gerson Sales

Relatório: Rede social +Pop

Caso de uso 1/2: cadastrar usuários, atualizar informações, login e logout.

Usuário é uma entidade que representa um utilizador do +Pop. É a partir dessa entidade que ocorrerá a maioria dos acontecimentos no sistema. Porém, nesse ponto as características do usuário são insuficientes para definir sua real importância. Sua criação se dará a partir de quatro propriedades principais: nome, e-mail, senha e data de nascimento, também é possível a inserção não obrigatória de uma foto de perfil(**Overload**). O usuário é o único responsável por armazenar estados sobre si mesmo, mantendo assim a proteção da informação. Para garantir sua integridade o usuário se utiliza de uma entidade responsável por validar e formatar suas informações, fazendo com que um padrão de criação seja obrigatoriamente seguido.

Essa entidade, ***UsuarioFormat***, segue o padrão de projeto **Singleton**, ou seja, apenas uma única instancia dela estará presente em todo o projeto. A criação de novos usuários no sistema acontece de forma cada vez mais crescente, mas isso não implica no surgimento de novos "formatadores" já que apenas uma instancia do ***UsuarioFormat*** existirá. Isso é possível porque a entidade possui apenas comportamento que pode ser utilizado de forma genérica para quaisquer usuários, sendo assim não seria coerente ter varias instancias obsoletas espalhadas, isso ajuda a não sobrecarregar o sistema e a otimizar processamento.

O +Pop conta também com um ***Banco de Usuários***, que é responsável por: realizar o cadastramento e/ou remoção de usuários do sistema, atualizar informações de usuários e classificá-los de acordo com a respectiva quantidade de POP's. Essa entidade é a única capaz de realizar alterações diretas nas informações cadastrais de um usuário. Assumindo tais responsabilidades o *Banco de Usuários* provê algumas características relacionadas ao **GRASP**:

Creator: A responsabilidade de criação de um usuário está fortemente atrelada ao *Banco de Usuários* já que é nele em que será mantido todos os usuários cadastrados do sistema.

Coesão: As responsabilidades, comportamentos e características do *Banco de Usuários* estão fortemente relacionadas e altamente focadas no usuário.

Baixo acoplamento: A dependência entre entidades foi reduzida já que o *Banco de Usuários* faz com que a alteração de um *Usuário* gere um um impacto quase nulo em outras entidades.

Expert: O *Banco de Usuários* não possui toda informação necessária para a manipulação de usuários, porém tem o conhecimento suficiente para lhe pedir as informações de que precisa.

Caso de uso 3: postar mensagem no mural.

O +Pop conta com um sistema de postagens, o *Usuário* poderá compartilhar suas opiniões, musicas prediletas, imagens favoritas e marcar tudo isso com suas hashtags, para que tudo isso seja possível foi desenvolvido uma entidade Postagem.

A Postagem precisa de dois atributos para ser criada: o conteúdo e a data, esse conteúdo se transformara posteriormente em algo que seja "***postável***", ou seja, um **Áudio**, **Imagem**, **Mensagem** e/ou **Hashtag**. Mas a partir desse ponto a postagem se utiliza de "ferramentas" para auxiliá-la em sua formação. Pensando nisso foi desenvolvido a entidade ***FabricaPostavel*** que se responsabiliza por fabricar qualquer tipo que seja considerado postável. Todo postável possui um comportamento que se encarrega de informar as condições a serem seguidas para sua criação, a *FabricaPostavel* utiliza esse comportamento para que de forma dinâmica possa criar entidades *postáveis*.

Mesmo que futuramente outras entidades sejam desenvolvidas, contanto que obedeçam esse padrão de comportamento a *FabricaPostavel* não precisará mudar seu algoritmo de busca, será necessário apenas informar que essa nova entidade passou a existir. Com pode ser percebido, há uma diminuição significante no acoplamento, ou seja, a dependência entre as entidades *Postagem*, *FabricaPostavel* e *Postável* é quase nula.

O usuário não possui relacionamento direto com a *postagem*, o responsável por essa intermediação é a entidade **MuralDeUsuario** que se responsabiliza por armazenar todas as postagens feitas pelo usuário. Essa entidade tem a função de diminuir o acoplamento entre a *postagem* e o *usuário*.

Esse esquema de postagem ofereceu um algumas contribuições para elaboração de um bom *design*:

Creator: A *FabricaPostavel* tem o comportamento e as informações necessárias para a criação de algo *postável*, fazendo assim com que nenhuma outra entidade interfira ou intervenha nesse processo. Enquanto que o *MuralDeUsuario* é responsável por criar as postagens já que é no mesmo que serão armazenadas todas elas.

Coesão: Os comportamentos e características da *postagem* estão diretamente relacionadas à algo postável e suas mídias, da mesma forma que as propriedades de uma postagem estão ligadas ao *MuralDeUsuario*.

Baixo acoplamento: A dependência entre essas quatro entidades foi reduzia, já que que uma assumiu a responsabilidade e o motivo de sua existência. Não existe dependência direta entre elas, existe apenas um acordo de que determinado padrão será seguido.

Expert: A postagem só precisa se apresentar como uma postagem, ou seja, não possui conhecimento necessário para a manipulação de algo que seja postável, entregando assim essa responsabilidade a *FabricaPostavel* que realiza o processo de produção, pois conhece o necessário para o manuseio comportamental de algo postável.

Override: A sobrescrita de métodos é de uma importância fundamental para o processo de criação de algo postável, pois é a partir da maneira com que cada entidade deseja ser criada que a *FabriaPostavel* consegue identificar cada tipo distinto e tratá-los de forma individual.

Interfaces: A postagem e a *FabricaPostavel* reconhecem apenas algo que assuma a forma de algo *postável*, garantindo assim um bom funcionamento de toda o mecanismo.

Caso de uso 4: Adicionar e remover amigos.

Para manter o relacionamento entre usuários o mais próximo possível, foi implementado um sistema para que qualquer *usuário* possa adicionar e/ou remover um amigo. Cada usuário possui um relacionamento *amigável*, possibilitando-o a possuir uma "***Lista de amigos***" essa lista é responsável por fazer a comunicação *amigável* entre todos os usuários, ou seja, nela estarão contidos todos os amigos de um *usuário* e todos os convites que ainda estão pendentes. Cada vez que um *usuário* tenta adicionar ou remover um amigo a entidade ***Notificações*** se encarregará de comunicar ao amigo sobre tal acontecimento.

Como o sistema presa por **segurança**, nenhum *usuário* terá acesso direto à informação de outro *usuário*, pois cada usuário assume uma forma de algo "*Amigável*" e a lista de amigos na verdade é uma lista de amigáveis e não de usuários de fato, isso garante que nenhum usuário poderá manipular a informação de seu novo amigo.

O relacionamento entre usuários contribuiu positivamente em alguns aspectos para o desenvolvimento de um bom *design*, foram eles:

Interfaces/Polimorfismo (proteção da informação): O *usuário* é protegido de outros usuários, pois a relação direta entre eles não existe, o que existe são intermediários que mantêm um comunicação segura entre ambos.

Coesão: As responsabilidades da lista de amigos estão unicamente ligadas ao "amigável", ou seja, tudo que se diga necessário para firmar a amizade entre dois usuários é dito pela lista de amigos, fazendo com que sua existência seja crucial para o perfeito funcionamento do sistema.

Expert: A lista de amigos não tem acesso à todas as informações de um usuário, quando precisa de algo que não possua ela pede ao usuário e só então decide o que fazer com tal informação.

Caso de uso 5/6: Popularidade, mudança no tipo de usuário.

A popularidade de cada usuário é o resultado da combinação entre suas postagens e suas amizades, pois tudo que um usuário posta será avaliado de forma positiva ou negativa por seu amigos. A pontuação arrecadada pelo usuário influenciará em seu o tipo, ou seja, à medida em que essa pontuação cresce ou decresce o comportamento do usuário é alterado.

Essa mudança acontece de forma dinâmica, fazendo com que os atributos dos usuários não sejam afetados(apenas seus comportamentos), essa mudança oferece ao sistema maior dinamicidade pois possui vários usuários que se encarregam de suas próprias modificações comportamentais, evitando assim o sobrecarregamento por ter de manter todas essas atualizações sob sua responsabilidade.

O principais características que influenciaram em um bom design foram:

Polimorfismo: A mudança de comportamento do *usuário* está fortemente atrelada ao fato de alguns de seus estados poderem assumir diferentes formas de acordo com cada ambiente.

Strategy: Padrão de projeto fundamental para a realização da troca dinâmica de comportamento de um usuário. Utilizando esse padrão o sistema manteve-se com um baixo acoplamento, pois o processo de atualização do tipo de um usuário não necessita de nenhuma contribuição externa.

Coesão: Foi atribuído ao usuário a o controle total sobre sua mudança de comportamento em relação à seu tipo, pois é nele em que se encontram as principais informações para que essa mudança ocorra.

Acoplamento: A troca dinâmica de comportamento faz com que o usuário seja facilmente gerenciado por entidades externas que não precisam realizar nenhuma alteração em seu estado para se adequar ao novo comportamento do usuário .

Caso de uso 7: Ranking de usuários e trending topics.

O +Pop oferece ao usuário um sistema de ***Ranking***, responsável por classificar todos os usuários cadastrados de acordo com suas respectivas quantidades de POP's. A entidade responsável por esse gerenciamento é o *Banco de usuários*, já que o mesmo tem o conhecimento sobre todos os usuários cadastrados no sistema. O *Banco de Usuário* consegue expor os três usuários mais populares e o três menos populares. Esse processo ocorre de forma dinâmica, pois no momento do cadastro o usuário já é considerado classificável e está sujeito a entrar no ranking, mas só será exibida a classificação quando for feita a requisição.

É possível realizar também a classificação de *Hashtags***,** informar ao usuário qual a *hashtag* que mais presente no sistema. O incumbido de realizar essa identificação é o **Banco de Hashtag**, toda e qualquer postagem feita no sistema é de certa forma monitorada pelo *Banco de Hashtag*, isto é, as hashtags são capturadas e enviadas ao banco logo no momento de sua criação. O *Banco de Hashtag* é atualizado automaticamente, basta somente uma "nova" hashtag surgir no sistema, isso faz com que toda a execução do programa flua de forma dinâmica e consistente, isso é possível porque existe algo totalmente capacitado a conhecer, manipular e acessar as informações de uma hashtag, ou seja, um especialista.

As principais características do processo de classificação que contribuem com um bom design:

Coesão: A responsabilidades incumbidas à *Banco de Usuário e Banco de Hashtag* existem porque ambas conhecem as informações necessárias para fazer o que fazem. Cada uma é especialista naquilo que faz, têm seus estados e comportamentos altamente focados em seus objetos manipulados respectivamente.

Expert: Ao necessitar de alguma informação que não esta em seu alcance, o *Banco de Usuário e Banco de Hashtag* pedem à seus objetos compostos tal informação, a partir desse ponto seŕa decidido o que fazer com essa informação: armazená-la ou não.

Caso de uso 8: Feed de notícias.

O +Pop possui um **Feed** onde os usuários poderão ver as mais recentes postagens de seus amigos de acordo com seus respectivos tipos. O *Feed* precisa ter acesso à Lista de amigos e as postagens desses amigos. Infelizmente isso gerou um forte acoplamento, pois o *Feed* tem conhecimento sobre uma lista de amigos e as postagens.

Na tentativa de melhorar essa relação foi implementado ao invés do *Feed* um **Relacionamento Amigável** que faz a intermediação entre usuário e a lista de amigos. Esse *Relacionamento amigável* é quem gerencia as tarefas de adicionar/remover amigos e também armazena as postagens de cada amigo.

Contribuições do Relacionamento amigável para o design:

Acoplamento: essa nova funcionalidade trouxe ao sistema um grande acoplamento, pois um usuário tem um relacionamento amigável que tem uma lista de amigos e postagens, e usuário tem um mural de usuário de que contém postagens. Existem muitas classes relacionadas para cumprir um único objetivo.]

Coesão: cada classe tem seus objetivos focados e realizam apenas aquilo que está capacitado a fazer. Ḿantendo sempre suas responsabilidades principais em primeiro plano.

Expert: o acoplamento foi gerado pelo fato de que os objetos tem acesso uns aos outros, porém essas informação que transitam pelo sistema estao protegidas

por seus detentores, ou seja, sempre que um objeto precisa da informação de outro objeto é necessário pedir e só então decidir o que fazer com tal informação pois existem informação que alguns objetos não têm a competência de manipular.

Caso de uso 9: Exportar e importar Posts por arquivos.

Por questões de segurança o +Pop permite que os usuários salvem todas as suas postagens. Para que isso fosse possível foi necessário implementar uma encapsulamento responsável por administrar todo tipo de informação que se refere à escrita e leitura de arquivos. Esse encapsulamento, **GerenciadorES** é utilizado apenas pela **Façade** do sistema, que sempre que lhe é solicitado para gravar algum tipo de informação passa a responsabilidade para o *GerenciadorES.*

influência no design:

Coesão: o *GerenciadorES* é totalmente capacitado para realizar suas atividades, e está altamente focado em tal atividade.

Expert: não é responsabilidade da *Façade* manipular esse tipo de informação, então ela entrega a responsabilidade para quem está realmente capacitado.

Caso de uso 10: persistência em arquivos.

Para mantar todas as informações salvas o +Pop conta com um sistema de persistência em arquivos. Tudo que foi realizado durante a execução do programa deverá e será armazenado no sistema. Como no passo anterior, o responsável por realizar tal atividade é o *GerenciadorES* já que o mesmo tem o conhecimento necessário para realizar esse tipo de operação.