Projeto:
Rede de Relacionamentos Jackut

Introdução

Este documento descreve o projeto que será desenvolvido na disciplina. O projeto é baseado no material do Prof. Jacque Sauvé da Universidade Federal de Campina Grande.

Este semestre, desenvolveremos uma rede de relacionamentos, o **Jackut**. Nesta disciplina, a análise é dada pronta para vocês, através desse documento (leiam com bastante atenção) e, principalmente, através dos artefatos de análise executáveis representados pelos testes de aceitação automáticos que serão fornecidos. Um dos objetivos desse projeto é fazer com que vocês consigam construir um sistema que passe 100% dos testes de aceitação. Se conseguirem, significa que vocês atingiram o objetivo mais importante no desenvolvimento de software de qualidade: atender aos requisitos do cliente.

Os testes de aceitação acessam apenas a lógica do negócio, não a interface com o usuário. Em um sistema qualquer, interfaces bonitas e fáceis de usar são uma característica muito importante. Entretanto, para facilitar a vida de vocês, só implementaremos a lógica de negócio do sistema, sem implementação alguma de interface com o usuário. Também é uma regra importante separar a interface com o usuário da lógica de negócio e fazer com que seu projeto não tenha interface com o usuário é uma forma de você sacar esse ponto arquitetural importante.

Vocês terão que projetar e implementar o sistema de uma forma incremental. Haverá alguns milestones e, em cada um, novas funcionalidades serão adicionadas ao sistema. As funcionalidades estão descritas dentro de User Stories. Uma User Story consiste de duas partes: 1) uma descrição informal do que se deseja em termos de funcionalidade e das interações que o sistema deve suportar; 2) um conjunto formal de testes de aceitação que, se executarem direitinho, comprovam que a funcionalidade foi implementada como desejada. Em cada um dos milestones, haverá um conjunto de user stories que deverá ser implementado. Antes de entregar um milestone, sempre leia as recomendações e a listagem do que deve ser entregue.

Descrição do Projeto (em poucas palavras)

O Jackut é um sistema que mantém uma rede de relacionamentos, nos moldes de uma série de outras que há na internet hoje em dia (Orkut, Friendster, etc.). Ele é particularmente inspirado no Orkut (www.orkut.com).

A funcionalidade desejada do Jackut está descrita em User Stories (Use Cases informais). Cada Story descreve uma interação que o sistema deve suportar. As User Stories receberam prioridades (pelo cliente interessado) e devem ser implementadas de acordo com tais prioridades. Em geral o jackut deve ser capaz de:

- Manter um cadastro de informações dos usuários (perfis, álbuns, etc.)
- Manter uma série de informações de relacionamentos entre os usuários (agrupamentos em comunidades, redes de amizade, de fãs, listas de paqueras, etc.)
- Manter o fluxo de mensagens entre os usuários do sistema.

As Users Stories

As User Stories (plural de User Story) levantadas inicialmente para o sistema estão mostradas abaixo. User Stories são uma forma de expressar requisitos funcionais desejados para o sistema (o que o sistema deve fazer). Observe que, no mundo real, o cliente poderá mudar de idéia com respeito a esses requisitos funcionais ao longo do tempo. É até possível que o professor (danado que é) altere os requisitos no meio do semestre. As User Stories foram priorizadas pelo cliente conforme podem ver na descrição dos milestones.

US	Título	Breve Descrição
1	Criação de conta	Permita a um usuário criar uma conta no Jackut . Deve ser fornecido um login, uma senha e um nome com o qual o usuário será conhecido na rede.
2	Criação/Edição de perfil	Permita a um usuário cadastrado do Jackut criar/editar atributos de seu perfil. Ele deve poder modificar qualquer atributo do perfil ou preencher um atributo inexistente.
3	Adição de amigos	Permita a um usuário cadastrado do Jackut adicionar outro usuário como amigo, o que faz o sistema enviar-lhe um convite. O relacionamento só é efetivado quando o outro usuário o adicionar de volta.
4	Envio de recados	Permita a um usuário cadastrado do Jackut enviar um recado a qualquer outro usuário cadastrado.
5	Criação de comunidades	Permita a um usuário cadastrado do Jackut criar uma comunidade. Deve ser fornecido um nome e uma descrição. O usuário passa a ser o dono da comunidade e é o responsável por gerenciar os membros.
6	Adição de comunidades	Permita a um usuário cadastrado do Jackut se adicionar a uma comunidade.

7	Envio de mensagens a comunidades	Permita a um usuário cadastrado do Jackut enviar uma mensagem a uma comunidade. Todos os usuários da comunidade a recebem.
8	Criação de novos relacionamentos	Permita a um usuário cadastrado do Jackut estabelecer outros tipos de relacionamentos dentro da rede, além de amizade; novos tipos incluem relação de fã-ídolo, paqueras e inimizades. Cada uma tem regras específicas, explicitadas nos testes de aceitação.
9	Remoção de conta	Permita a um usuário encerrar sua conta no Jackut. Todas as suas informações devem sumir do sistema: relacionamentos, mensagens enviadas, perfil.

Testes de Aceitação

A validação das User Stories implementadas pelos alunos serão avaliadas por um conjunto de testes de aceitação pré definidos. Os testes serão disponibilizados pelo professor à medida que os milestones avançarem.

Os testes serão implementados em uma linguagem de script que é lida pela biblioteca Easy Accept. Esta biblioteca será fornecida pelo professor.

Para poder rodar os testes, vocês precisarão criar uma Façade que acessa a lógica de negócio do seu sistema segundo uma linguagem de script que foi criada para os testes de aceitação.

Linguagem de Script do Easy Accept

Para acessar a business logic de seu programa e poder realizar os testes de aceitação, você precisará implementar uma linguagem de script que consiste nos comandos abaixo. Esses são os comandos usados nos testes de aceitação (us1.txt, us2.txt, etc.). Cada um deles deve corresponder a um método na sua façade.

- zerarSistema
    - Apaga todos os dados mantidos no sistema.
- criarUsuario login=<String> senha=<String> nome=<String>
    - Cria um usuário com os dados da conta fornecidos.
- abrirSessao login=<String> senha=<String>
    - Abre uma sessão para um usuário com o login e a senha fornecidos, e retorna uma id para esta sessão.
- getAtributoUsuario login=<String> atributo=<String>
    - Retorna o valor do atributo de um usuário, armazenado em seu perfil
- editarPerfil id=<String> atributo=<String> valor=<String>

    - Modifica o valor de um atributo do perfil de um usuário para o valor especificado. Uma sessão válida (identificada por id) deve estar aberta para o usuário cujo perfil se quer editar.
- adicionarAmigo id=<String> amigo=<String>
    - Adiciona um amigo ao usuário aberto na sessão especificada através de id.
- ehAmigo login=<String> amigo=<String>
    - Retorna true se os dois usuários são amigos.
- getAmigos login=<String>
    - Retorna a lista de amigos do usuário especificado (codificado em uma String)
- enviarRecado id=<String> destinatario=<String> mensagem=<String>
    - Envia o recado especificado ao destinatário especificado. Uma sessão válida (identificada por id) deve estar aberta para o usuário que deseja enviar o recado.
- lerRecado id=<String>
    - Retorna o primeiro recado da fila de recados do usuário com a sessão aberta identificada por id.
- encerrarSistema
    - Grava o cadastro em arquivo e encerra o programa. Atingir o final de um script (final de arquivo) é equivalente a encontrar este comando.

Milestones

A seguir serão apresentados os milestones que deverão ser entregues.

Primeiro milestone

Implemente as user stories 1 a 4.

Recomendações para a entrega:

Coisas que devem ser entregues:

- Código-fonte documentado
- Javadoc do código-fonte (rode o javadoc antes de mandar)
- Um relatório descrevendo seu design e explicando suas escolhas para o mesmo. Inclua um diagrama de classes para facilitar seu texto do relatório.

Lembre dos seguintes pontos ao entregar o resultado:

- Entregue um único arquivo zip (ou rar). O nome do arquivo zip deve ser "milestone1-joao-maria.zipx" (ou.rarx) e assim por diante (identificando o milestone e o nome dos membros da equipe). Use a terminação zipx (ou rarx) e não zip (rar) para driblar o gmail que não entrega mails com attachments contendo arquivos executáveis (ele olha dentro de attachments zip/rar). Pode mandar mail para hozano@ic.ufal.br.
- Ao extrair do arquivo zip(rar), tudo deve cair no lugar certo para ajudar minha tarefa de testar e verificar seu trabalho.

- Posso extrair do arquivo em qualquer diretório que eu quiser na minha máquina.
- Use apenas nomes de arquivos relativos, pois nomes absolutos que poderão existir na sua máquina poderão não existir na minha máquina.
- Lembre que vou brincar de testar seu sistema com meus próprios testes adicionais, hehehe...
- Para ter certeza que tudo funcione adequadamente, teste os passos acima numa máquina diferente daquela onde você desenvolveu o software.
- Se seu projeto não compilar, há penalidade equivalente a 4 dias de atraso (isto é, 20%). Portanto, teste bem e em várias máquinas!
- Estarei testando seu software com o JDK 1.5.0 ou superior. Não peça para que eu use uma versão velha de Java.
- Não topo editar build.xml ou arquivos .bats para acertar o classpath ou qualquer outra coisa.
- Entregue um relatório (chamado relatorio-milestone1.doc - ou txt, ou html, ou pdf, ...) descrevendo seu design e explicando suas escolhas para o mesmo. Inclua um diagrama de classes para facilitar seu texto do relatório.
- O projeto será avaliado de acordo com os seguintes pesos:
    - Compilação (0% - se não compilar, não aceito o projeto)
    - Execução dos testes de aceitação (50%)
    - Qualidade da documentação (20%)
    - Qualidade do design (20%)
    - Qualidade do relatório (10%)
- Eis as instruções de como seu projeto será corrigido:
    - Compilação: O programa deve compilar. Se não compilar, os alunos (e o professor) devem ser avisados imediatamente do fato. A correção desse projeto pára. O projeto receberá uma penalidade, mesmo se for entregue depois para corrigir o problema.
    - Testes de aceitação: Observe se todos os testes de aceitação executam. Para a user story 1, os testes estão em us1.txt e assim por diante. Retire 1 ponto (sobre 10) para cada testes de aceitação que não funciona. No milestone 1, apenas us1, us2, us3 são testadas. A partir do milestone 2, os testes das user stories de milestones anteriores devem ser retestadas.
    - Relatório: leia o relatório e dê uma nota de 0 a 10 sobre a qualidade.
    - Documentação: verifique a qualidade do javadoc. Deve-se ter uma descrição de cada pacote, cada classe, cada método, cada parâmetro, cada valor de retorno e cada exceção. Não precisa verificar todo o código. Basta fazer um scan rápido. Dê nota de 0 a 10.
    - Design: Avalie a qualidade do design examinando o código e/ou o relatório (diagrama de classes). Verifique as entidades principais e seus métodos. As entidades principais fazem sentido? Estão desempenhando responsabilidades adequadas (seus atributos e métodos fazem sentido)? os relacionamentos criados fazem sentido? Dê nota de 0 a 10.