Pós-graduação MIT em Engenharia de Software

GCREDIRECT

Marcelo Rezende Módolo
Tomás de Aquino Tinoco Botelho, Érico Corrêa Torres

Rio, Janeiro 2012

DEDICATÓRIA

Quando comecei a pós-graduação no Instituto Infnet fiquei feliz em saber que teria contato com professores que estavam inseridos no mercado (em minha formação na instituição anterior, cerca de setenta e cinco por cento dos mestres estava fora do mercado). Todos os professores e mestres que conheci durante o curso contribuíram para minha formação e conhecimento.

Agradeço a todos com carinho.

Cabe aqui um agradecimento especial aos mestres Tomás de Aquino Tinoco Botelho e Érico Corrêa Torres pela constante luta pela melhora da qualidade das aulas e por estarem sempre junto aos alunos.

"Se eu vi mais longe, foi por estar de pé sobre ombros de gigantes."

– Sir Isaac Newton

RESUMO

O sistema aqui desenvolvido visa gerenciar a criação e manutenção de redirecionamentos em sítios Web.

O objetivo deste trabalho é exemplificar a construção de um software servindo como base para que outros estudantes possam se beneficiar das soluções encontradas no decorrer do mesmo.

Desenvolver um software com qualidade não é uma tarefa trivial. Durante os quase vinte e quatro meses da pós-graduação em engenharia de software passamos por várias disciplinas e analisamos várias metodologias modernas que devem levar a um modelo de desenvolvimento com qualidade e com artefatos entregues dentro dos prazos estabelecidos. Aprendemos a analisar os riscos, estimar a complexidade, usar metodologias orientadas a objetos, mas, principalmente entendemos que não existe uma "bala de prata" capaz de resolver todos os problemas e desafios do desenvolvimento de software com qualidade e nos curtos prazos que o mercado atual exige.

Desenvolver um software que atenda aos requisitos exige uma grande competência por parte da equipe envolvida. A qualidade não deve ser medida ao final, ela deve ser parte intrínseca do processo. Os riscos devem ser levantados e avaliados logo no início para verificar o custo e a viabilidade do sistema. O conhecimento adquirido com a construção do sistema deve dar origem a uma cultura que permita facilitar a construção de novos sistemas, conforme explicado nas próximas seções

Palavras-Chave: Desenvolvimento, Risco, Cultura, Conhecimento, Qualidade.

ABSTRACT

The system here designed aims to manage the creation and maintenance of redirects in websites.

The goal of this paper is to set an example of software construction to other students so they can take benefit from the solutions found throughout it.

To build up a software with quality is not a trivial job. During - almost - 24 months of software engineering post-graduation course, we have been through several subjects and we analyzed several modern methods that must lead to a qualitative development model and with artifacts delivered on time. We learned how to analyze risks, assess its complexity, to use methodologies directed to objects, but, above all, we understood that there is no “silver bullet” capable of solving all the problems and challenges of developing a software with quality and short terms that the prevailing market requires.

To develop a software that is up to its requirements demand great ability from the team engaged in it. Quality cannot be measured at the end, but it must be an inherent part of the process. Risks must be surveyed and evaluated at the beginning to ascertain the costs and viability of the system. The knowledge acquired with the system’s construction must originate a culture that makes easier the making of new systems, as explained on the next sections.

Keywords: Development, Risk, Culture, Knowledge, Quality.

SUMÁRIO

ÍNDICE DE TABELAS

ÍNDICE DE ILUSTRAÇÕES

1 INTRODUÇÃO

"Diga-me e eu me esquecerei. Ensine-me e eu me lembrarei. Envolva-me e eu aprenderei."

– Benjamin Franklin

O crescimento da Web tornou imperativo a presença dos canais de televisão paga em sítios onde é possível consultar a programação, visualizar resumos, opinar, e até participar de jogos e outras atividades. É comum para o visitante de um sítio Web armazenar os endereços de páginas que mais o agradem. Como o conteúdo é altamente dinâmico e a publicação de matérias e de novas páginas é diária, esses endereços armazenados acabam por se tornar inválidos causando um desconforto.

Para evitar que um endereço armazenado pelo visitante se torne desatualizado em relação ao novo conteúdo, os sítios passaram a usar uma técnica chamada de redirecionamento. Um redirecionamento funciona como um apelido onde o conteúdo novo pode ser acessado a partir de seu antigo endereço. Passou-se então a criar redirecionamentos para todo conteúdo que fosse de relevância para o sítio (endereços de programas com muita audiência, sítios de clubes de futebol, jogos e etc.), assim o número de redirecionamentos cresceu tornando a manutenção dos mesmos um problema (duplicação, não atualização, destinos não existentes, autorização).

A demanda para a manutenção desses redirecionamentos culminou no desenvolvimento de um sistema aqui chamado de GCRedirect.

Hoje os sítios hospedados com a Globosat (Sportv, Multisow, GNT, etc.) usam os redirecionamentos para tornar a navegação confortável para os visitantes evitando que os endereços armazenados por eles tornem-se inválidos.

No texto que segue, alguns termos foram mantidos em Inglês para facilitar o entendimento e não por desrespeito à nossa língua (é mais fácil comunicar o sentido de *frameworks* do que o de arcabouços).

1.1 TEMA

Não existe maneira fácil para desenvolver um sistema, deve-se buscar sempre a maneira correta, a que melhor se aplica, com o menor custo e que permita a entrega de um produto com qualidade, usabilidade e manutenibilidade. A escolha das ferramentas (linguagem, plataforma, bibliotecas e etc.) que vão auxiliar no desenvolvimento de um sistema é parte crucial do sucesso.

Para o desenvolvimento do sistema proposto no prazo estipulado, optou-se pelo uso do Framework Spring que é um dos mais populares para o desenvolvimento Java. O Framework Spring permite criar sistemas com alta performance e facilmente testáveis.

Uma das maiores dificuldades com sistemas voltados para a Web é a segurança. Autenticação e autorização devem ser verificadas e validadas durante a análise de risco para evitar surpresas nas fases posteriores. O caso de uso para autenticação do sistema - existe alguma controvérsia (ver *"Login" is not a Use Case* na referência bibliografia) em considerar autenticação e autorização como um verdadeiro caso de uso já que na verdade um usuário "não quer" autenticar-se em um sistema, o que ele deseja realmente é o uso de uma funcionalidade específica que se torna acessível após o processo de autenticação/autorização -, apesar de ser o menor, foi o que causou maior dificuldade (e impacto no prazo) pois deveria atender regras específicas de negócio que invalidavam a maioria das configurações prontas encontradas em diversas bibliotecas. Foi necessário uma customização específica do processo como um todo para atender aos requisitos de segurança e registro do sistema desenvolvido.

1.2 PROBLEMA

Como entregar um sistema com qualidade, dentro do prazo estabelecido e com o menor custo possível? Os estudos e pesquisas mostram os seguintes aspectos:

- Cerca de um quarto dos projetos de desenvolvimento de grandes sistemas é cancelado antes da conclusão;
- em média, o tempo de desenvolvimento é bem maior do que o estimado;
- três quartos dos grandes sistemas não são usados ou não funcionam como planejado;
- a manutenção e reutilização de software são, em geral, extremamente difíceis e custosas;
- às vezes é mais vantajoso desenvolver um novo sistema do que modificar, estender ou usar a funcionalidade de um já existente.

Trabalhando com suporte ao desenvolvimento e implantação de sistemas observamos uma grande dificuldade na manutenção da documentação e na padronização dos sistemas que entregamos. Não é criado um conhecimento comum, uma cultura, que permita aos novos sistemas o benefício do conhecimento adquirido com outras soluções. É como se inventássemos a roda a cada vez que um carro fosse construído!

1.3 OBJETIVOS

Desenvolver um sistema que atenda aos requisitos do cliente com qualidade e no prazo estabelecido.

Ilustrar o processo de desenvolvimento com o uso de tecnologias atuais e que permitam facilitar a construção e garantir a qualidade do produto final.

Possibilitar a outros um guia das tecnologias e ferramentas usadas para alcançar os objetivos apresentados.

Demonstrar que o que nos foi ensinado ao longo de nossa pós-graduação pode e deve ser aplicado com sucesso em nossas carreiras como engenheiros de software.

1.4 Relevância do Trabalho

Na introdução à engenharia de software um dos assuntos abordados foi a comparação dos artefatos gerados pelas diferentes "engenharias". É comum a vários produtos de engenharia uma tolerância ou margem de conformidade com o que seria o ideal. Por exemplo, ao adquirir uma casa esperamos que as paredes estejam no prumo e no esquadro mas não existem paredes perfeitas e diferenças aceitáveis são comuns. Um motor ao ser construído usa peças que atendem a uma margem de tolerância com relação a folgas. Mas e o software? Qual a margem de tolerância que podemos (ou devemos) aceitar? Ele deve ser sempre preciso ou apenas na maioria das vezes? É muito difícil mensurar o que é intangível!

Ao desenvolver um sistema, passamos por alguns problemas comuns:

- A escolha da linguagem;
- que ferramentas usar;
- como documentar;
- o que documentar;
- metodologias;
- prazos.

O leque de opções disponíveis acaba muitas vezes por confundir aqueles que estão iniciando na carreira de engenheiros de software. Mesmo com a facilidade de se encontrar material de apoio e tutoriais na Web ainda assim fica a lacuna de como juntar as diversas tecnologias para obter um produto funcional. Um importante objetivo do trabalho apresentado é ser útil como um guia para aqueles que necessitam desenvolver um sistema e ainda não tem uma cultura de desenvolvimento formada.

O trabalho apresentado propõe-se a resolver a demanda gerada pelo departamento responsável pelo conteúdo dos sítios hospedados na Web e o departamento de infra-estrutura de tecnologia da informação.

O que é um redirecionamento? Um redirecionamento é um recurso disponibilizado pelo serviço que entrega o conteúdo de um sítio e que permite que mais de um caminho lógico (por caminho lógico entende-se um *Uniforn Resource Locator*, em português Localizador Padrão de Recursos ou URL) aponte em um endereço físico (no caso, o recurso localizado no sistema de arquivos do servidor). Por exemplo, a URL "*sportv/bem-amigos*" na verdade endereça o recurso */var/www/sportv/conteúdo/bem-amigos-01*. O redirecionamento permite também criar nomes mais apropriados ou com significado mais amigável para o visitante.

2 FUNDAMENTAÇÃO TEÓRICA

Segundo Martin Fowler (autor, palestrante, consultor de desenvolvimento de software. É um dos pioneiros da tecnologia orientada a objetos, refatoração, padrões, metodologias ágeis, modelagem de domínio, UML e XP) o desenvolvimento de software atualmente exige que o programador seja poliglota, ou seja, não basta apenas conhecer uma linguagem. É necessário saber falar mais de uma linguagem para que seja possível encontrar a melhor solução para o problema em questão. Além disso, uma metodologia ágil deve permitir a entrega de artefatos intermediários que possibilitem validar os requisitos e garantir que o desenvolvedor esteja no caminho certo. Não podemos esquecer que a refatoração é um processo muito importante para alcançar a qualidade e permitir um código manutenível.

"Codificar como documentação!" Ainda citando o autor Martin Fowler o desenvolvimento ágil implica que a codificação é parte central no desenvolvimento de um produto, e com isso, o código passa a ser a fonte primária de documentação.

Entregar pequenas partes de um software que sejam funcionais exige que mais tarde possamos integrar e testar cada parte garantindo que mudanças nos requisitos não causem problemas nos artefatos entregues anteriormente. A única maneirar de garantir que isso aconteça é desenvolver com base em testes que possam ser aplicados e executados com facilidade. O desenvolvimento dirigido por testes é uma técnica que se baseia em um ciclo curto de repetições onde o desenvolvedor escreve um caso de teste automatizado que define uma melhoria ou funcionalidade levantada nos requisitos. O que tem causado uma certa confusão, e alguma discussão também, é que não saímos escrevendo testes sem antes definir com clareza os contratos. Kent Beck (engenheiro de software, autor de diversos livros, e criador da Extreme Programming e Test Driven Development Software) autor do livro Test-Driven Development: By Example. Addison-Wesley. Winner of the Jolt Productivity Award. (ISBN 978-0321146533) mostra como desenvolver um software que pode ser testado e quais são os requisitos para que esses testes possam ser realizados com sucesso. Não adianta tentar desenvolver software orientado a testes se não tiver um forte conhecimento em padrões e orientação a objetos. O TDD exige que possamos exprimir as funcionalidades com base em contratos e a partir desses contratos criar os testes.

Um fator importante para a comunicação entre a equipe que desenvolve um sistema e o cliente é a manutenção de uma linguagem comum onde os termos e regras do negócio possam ser identificadas com facilidade. O desenvolvimento orientado ao domínio do problema ou DDD teorizado pelo autor do livro Domain Driven Design Eric Evans baseia-se na premissa de manter o foco na lógica do domínio e que domínios complexos devem se basear em modelos.

Não podemos deixar de citar os padrões de projeto. Não é possível desenvolver com qualidade sem seguir os padrões estabelecidos de arquitetura e boas práticas. Um padrão é uma maneira testada e que pode ser reutilizada com sucesso. O livro Design Patterns: Elements of Reusable Object-Oriented Software, publicado em 1995, dos autores Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides é uma obra fundamental que deve ser lida e aplicada.

Cada uma das metodologias e técnicas citadas acima estão interligadas e não podem ser aplicadas de forma independente. Vejamos por exemplo o desenvolvimento orientado a testes ou TDD:

- Para usar o TDD é necessário fatorar o sistema em contratos (ou interfaces) que reflitam os requisitos;
- interfaces bem definidas devem expressar regras de negócio em um formato que possa ser entendido e validado por todos os envolvidos;
- para criar boas interfaces temos que ter uma linguagem comum ou "*Ubiquitous Language*" conforme postulada pela Domain-Driven Design Community;
- sem um forte conhecimento em padrões de projeto é praticamente impossível criar boas interfaces e sistema com baixo acoplamento;
- para criar sistemas que exijam o mínimo de documentação é necessário fazer uso de convenções e de uma nomenclatura padrão baseada em uma cultura estabelecida.

O desenvolvimento de software não é um processo isolado, é o somatório de muitas disciplinas que culminam em um conhecimento comum (uma cultura) que pode e deve ser reutilizado.

3 METODOLOGIA E DESENVOLVIMENTO

3.1 METODOLOGIA

O sistema em desenvolvimento foi concebido como solução para a demanda de criação e manutenção dos redirecionamentos para os sítios hospedados. O sistema deve também ilustrar de forma satisfatória as dificuldades comuns no desenvolvimento de um sistema real onde segurança e desempenho devem estar presentes.

O processo de criação de um redirecionamento para um sítio segue os passos a seguir:

- O editor responsável pela criação de conteúdo para um sítio envia por e-mail uma solicitação para criação de um redirecionamento;
- a solicitação chega ao departamento de infra-estrutura onde um dos analistas verifica junto a coordenação do sítio se o editor pode pedir o redirecionamento;
- em caso afirmativo, o analista cria o redirecionamento e informa ao editor;
- caso tudo esteja correto o processo termina.

Para a remoção de um redirecionamento o processo é muito similar ao anterior sendo que nesse caso o redirecionamento é removido.

A partir do exposto podemos observar uma grande fragilidade no processo. O analista de infra-estrutura não sabe se o editor está autorizado a fazer o pedido, sendo necessário validar com o coordenador do sítio a legitimidade do mesmo. Os pedidos são feitos por e-mail, tornando difícil o registro.

Um dos requisitos para o sistema proposto é que a manutenção dos redirecionamentos e o controle das permissões (quem está autorizado) seja de inteira responsabilidade do departamento diretamente responsável pelo sítio, retirando essa tarefa do departamento de infra-estrutura de tecnologia da informação.

Outro requisito importante é a capacidade do sistema criticar as entradas e evitar erros, como destinos não existentes e a criação de redirecionamentos já existentes.

Toda operação relacionada à criação ou remoção de um redirecionamento deve ser rastreada em registro para posterior auditoria.

O levantamento dos requisitos foi baseado em entrevistas com os editores dos sítios e os seus respectivos coordenadores . Além disso, os e-mails (na Ilustração 1 temos um exemplo de e-mail enviado) trocados entre os departamentos e arquivados ao longo de quase

dois anos foram cadastrados por ordem de envio, e a partir dos mesmos criou-se uma tabela (ver Tabela 1) com os campos data, sítio, editor, caminho de origem, caminho de destino e observação. O campo observação foi usado para identificar os redirecionamentos que não funcionaram e a razão do problema.

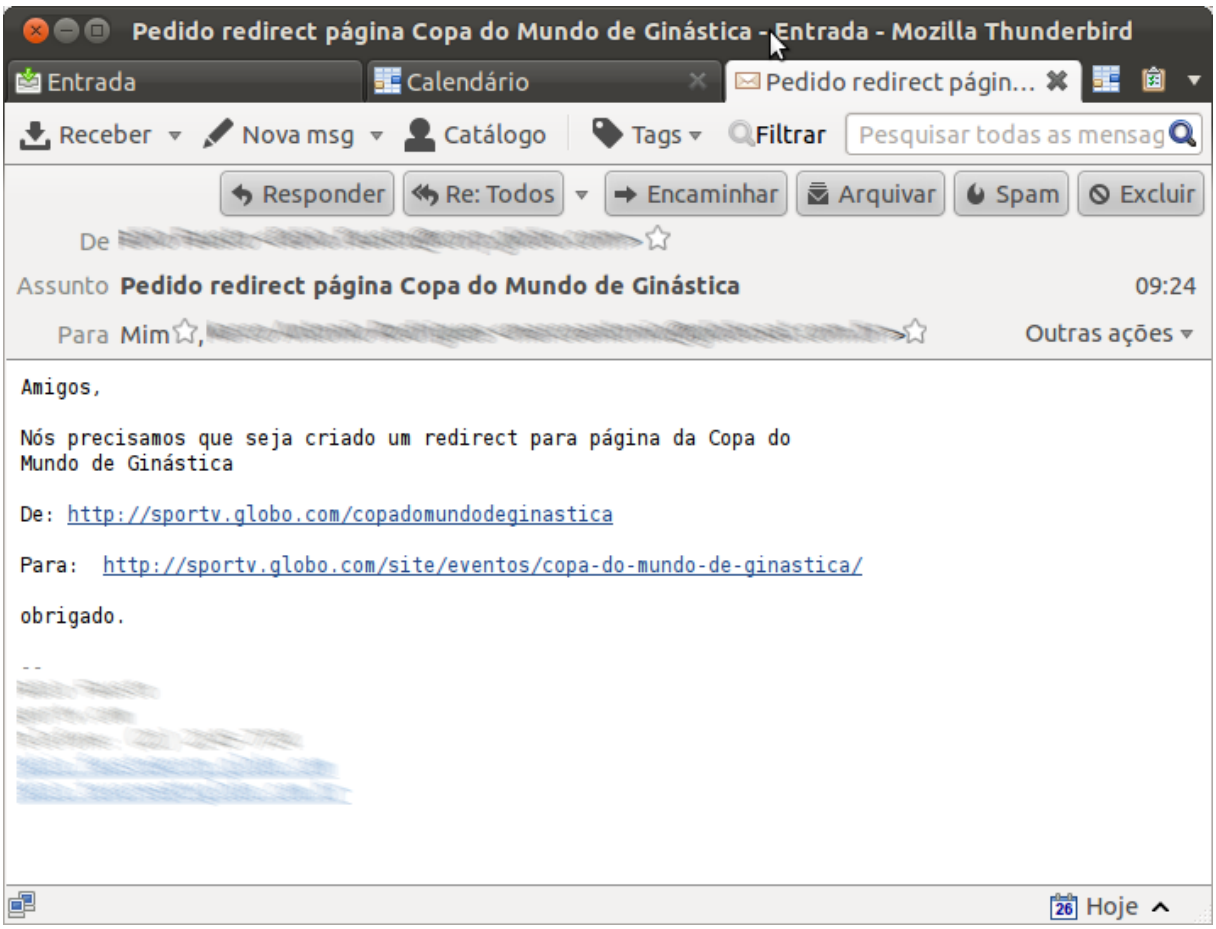

ILUSTRAÇÃO 1: E-MAIL COM PEDIDO DE REDIRECIONAMENTO

A partir do e-mail recebido o conteúdo que indica o redirecionamento é editado para refletir o formato dos arquivos de configuração:

```
De: http://sportv.globo.com/copadomundodeginastica

Para:  http://sportv.globo.com/site/eventos/copa-do-mundo-de-ginastica/

Redirect permanent /copadomundodeginastica
http://sportv.globo.com/site/eventos/copa-do-mundo-de-ginastica/
```

A Tabela 1 (meramente ilustrativa) foi criada com base no conteúdo dos e-mails recebidos.

DATA	SÍTIO	EDITOR	ORIGEM	DESTINO	OBSERVAÇÃO
01/03/2010	SPO	Carlos Eduardo	/bons-amigos	/bons-amigos-03	OK
...					

TABELA 1: MAPEAMENTO DOS PEDIDOS

A partir das tabelas e dos requisitos obtidos, um esboço lógico para o banco de dados foi criado, a relação dos redirecionamentos existentes foi obtido diretamente dos arquivos de configuração e foram usados como entrada para preenchimento das tabelas no banco de dados.

Os arquivos de configuração são formatados com um redirecionamento por linha sendo cada linha dividida em quatro campos:

- Palavra-chave - *Redirect*;
- palavra-chave - *permanent*;
- origem;
- destino.

A seguir o arquivo de configurações para um dos sítios:

Redirect permanent /Sportv/2009/cartolafc/0,,17026,00.html	http://sportv.globo.com/cartola-fc
Redirect permanent /Sportv/2009/blogs/0,,17086,00.html	http://sportv.globo.com/Blogs
Redirect permanent /Sportv/2009/atletismo/0,,17045,00.html	http://sportv.globo.com/atletismo
Redirect permanent /Sportv/2009/basquete/0,,17017,00.html	http://sportv.globo.com/basquete
Redirect permanent /Sportv/2009/esportesaquaticos/0,,17044,00.html	http://sportv.globo.com/esportesaquaticos
Redirect permanent /Sportv/2009/esportemotor/0,,17018,00.html	http://sportv.globo.com/esportemotor
Redirect permanent /Sportv/2009/futebol/0,,17020,00.html	http://sportv.globo.com/futebol
Redirect permanent /Sportv/2009/futsal/0,,17021,00.html	http://sportv.globo.com/futsal

Cada sítio tem o seu arquivo correspondente e todos os redirecionamentos são do tipo permanente. Foi necessário também criar um processo para exportar os redirecionamentos antigos para um formato compatível com as tabelas criadas.

Durante o levantamento dos riscos alguns pontos foram destacados:

- O formato do arquivo de configuração é crítico e qualquer erro pode comprometer o serviço de publicação;
- os arquivos de configuração não devem ser apagados, eles devem ser renomeados adicionando uma estampa de data e hora ao nome;

- o acesso ao sistema será feito exclusivamente pela rede interna através de VPN dedicada;
- o usuário de um sítio não pode ter acesso ao conteúdo dos outros sítios.

Em relação aos registros de auditoria ficou acertado que:

- Todas as operações devem ficar registradas em arquivos de auditoria;
- os redirecionamentos não serão removidos das tabelas, serão identificados como removidos e não serão usados para gerar novos arquivos;
- os usuários também não serão removidos ao invés disso serão marcados como inativos e não poderão mais se autenticar no sistema.

Determinou-se também que a documentação seria baseada na UML com os diagramas essenciais para entendimento e manutenção do sistema.

Foi acertado que o código fonte será de domínio público conforme a licença LGPL e que o material obtido e usado para o TCC não fere de forma alguma a propriedade intelectual da empresa onde os dados foram obtidos.

3.1 CONTEXTUALIZAÇÃO DO CASO

O redirecionamento surgiu como resposta a um problema comum em sítios dinâmicos e com conteúdo jornalístico. As matérias criadas mudam de nome; os programas são movidos de uma localização para outra; alguns produtos ganham destaque e passam a ser comentados na mídia usando apelidos (o Flamengo é chamado de *mengão*, o Fluminense de *nense*, Marília Gabriela Entrevista de *gabi,* etc...) tudo isso é possível de ser resolvido com um simples redirecionamento que funciona como uma tabela de busca onde o apelido é pesquisado e o destino associado a ele passa a ser retornado.

Com o aumento da audiência dos sítios, a manutenção desses redirecionamentos começou a se tornar ineficiente. Os arquivos onde os redirecionamentos são criados tornaram-se extensos e desorganizados. Os pedidos para serem atendidos necessitavam de autorização causando atrasos e desconforto aos editores. Cada plantonista responsável por um sítio obrigava a necessidade de um analista para poder criar e verificar o redirecionamento pedido.

Assim, questionou-se a possibilidade de desenvolvimento interno de um sistema que atendesse a essa demanda de forma satisfatória e com segurança. A partir desse ponto as entrevistas foram feitas entre os departamentos envolvidos (o que foi fácil devido ao bom relacionamento que já existia e pela necessidade de acabar com a burocracia que cercava a autorização para a efetivação dos pedidos) o material de apoio coletado e os primeiros protótipos apresentados.

3.2 Resultados Obtidos

Ao final da análise dos riscos e do levantamento inicial dos requisitos um conjunto extenso de artefatos ficou disponível para permitir a criação dos casos de uso e protótipos que serão usados para validar os requisitos.

Usou-se pequenas histórias como narrativa para cada caso de uso. Os e-mails arquivados ao longo de quase dois anos de pedidos foram suficientes para criar bastante documentação de apoio. Os arquivos de configuração estavam bem documentados, inclusive indicando os redirecionamentos que não funcionaram e a razão do problema.

Foi criado um formulário para que cada sítio retornasse os nomes dos editores que estariam autorizados a criar redirecionamentos e quais as informações que deveriam constar para um editor no sistema (nome, e-mail, telefone).

O sistema proposto não é de grande complexidade mas apresenta requisitos interessantes de segurança e manipulação de arquivos que podem com certeza ser úteis para outros sistemas.

Acredita-se que para construir bons sistemas é necessário antes estabelecer uma cultura de boas práticas e conhecimento. Esse tópico será utilizado para ilustrar as maiores dificuldades e as soluções encontradas ao longo do caminho.

3.2.1 Plataforma

Escolher a plataforma para o desenvolvimento de um sistema é uma decisão crítica e que normalmente vai impactar no sucesso ou fracasso do sistema.

Hoje em relação ao mercado, a escolha recai em duas tecnologias: Java (Oracle) e .NET (Microsoft), ambas oferecem muitos recursos e uma enorme quantidade de bibliotecas e frameworks à disposição (alguns inclusive gratuitos). A escolha feita, neste caso, e explicado a seguir, foi a plataforma Java.

Segundo o sítio TIOBE www.tiobe.com a linguagem de programação Java é a mais usada (ver Ilustração 2).

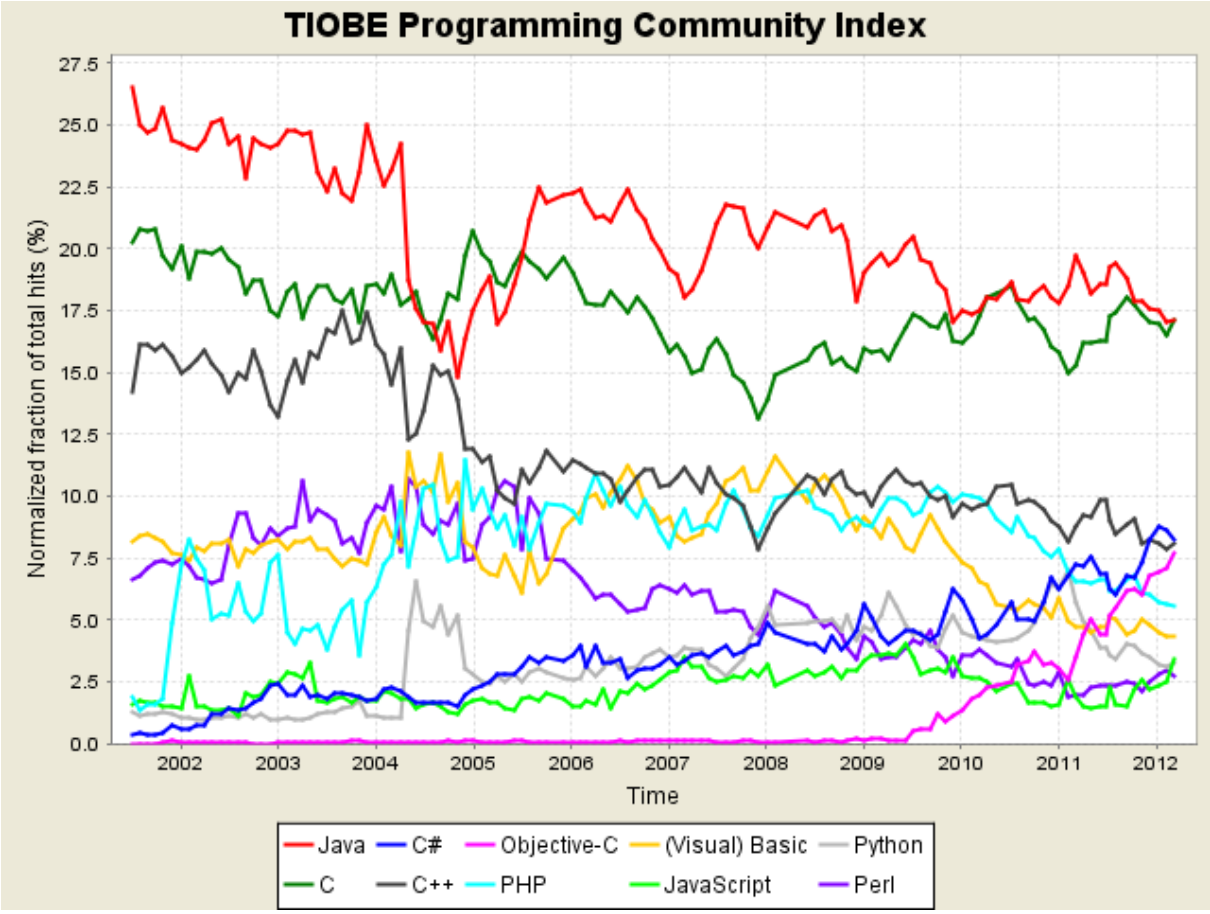

ILUSTRAÇÃO 2: TIOBE LINGUAGENS DE PROGRAMAÇÃO

Como plataforma para aplicações que exigem escalabilidade e segurança o Java se destaca e oferece um conjunto de recursos bastante consolidado e reconhecido pelo mercado.

Em abril de 2009 a Oracle ofereceu US$ 7,4 bilhões pela Sun Microsystems. Após o fracasso das negociações com a IBM, finalmente a proposta foi aceita pela Sun. O reflexo da compra pode ser observado no gráfico onde é claro uma tendência de declínio (somada ao fato dos grandes investimentos por parte da Microsoft em seu Framework .NET). Seguiu-se um período de incertezas sobre o destino da linguagem. Os investimentos em novas funcionalidades continuariam? A saída do criador da linguagem Java James Gosling poderia prejudicar a evolução da linguagem? O tão esperado Java 7 seria realmente lançado? Depois de quase dois anos de incertezas finalmente é possível observar o comprometimento da Oracle com o futuro do Java. As últimas boas notícias vêm do esforço para tornar o Java Community Process (JCP) novamente efetivo e com capacidade de decisão.

O lançamento do Java 7 com muitas melhorias e maior integração com linguagens dinâmicas mostrou que a linguagem continua em evolução. Na verdade, o Java deixou de ser

uma linguagem para se tornar uma plataforma. A máquina virtual Java permite que outras linguagens (inclusive linguagens dinâmicas como Python e Ruby) rodem sobre a JVM beneficiando-se do extenso conjunto de bibliotecas existentes.

Mas se o Java é tão eficiente como plataforma por que existem tantas reclamações quanto à complexidade e quantidade de código necessário para desenvolver aplicações em comparação com outras linguagens? A linguagem Java nasceu fortemente baseada em padrões, o JCP é formado por engenheiros que seguiram os padrões de projeto na criação das bibliotecas e utilitários. Esses padrões apesar de tornarem a escrita de código mais verborrágica (em decorrência disso algumas brincadeiras surgiram... "Eu tinha um problema, tentei resolver em Java e agora tenho um *IProblema* e um *ProblemaImpl*" fazendo alusão ao uso exagerado dos padrões de construção) permitem por outro lado uma extensibilidade e robustez inexistente em outras linguagens.

A maior parte da insatisfação surgiu com o lançamento do Entreprise JavaBeans (EJB). O EJB é um componente gerenciado pelo servidor de aplicações e com foco na construção de aplicações corporativas de arquitetura modular, robustas e escaláveis. A especificação para os EJB foi originalmente desenvolvida pela IBM em 1997 e depois adotada pela Sun Microsystems como versões EJB 1.0 e 1.1 em 1999. Através do Java Community Process (JCP) vêm sendo constantemente melhorada JSR 19 (EJB 2.0), JSR 153 (EJB 2.1), JSR 220 (EJB 3.0) e JSR 318 (EJB 3.1).

Acreditamos que caiba aqui uma melhor explicação dos EJB e seu campo de aplicação, já que é comum encontrar dúvidas do tipo "Por que não posso usar o Apache Tomcat (um servidor Java Servlet) que é muito mais leve do que o Redhat JBoss (um servidor de aplicações java) para a minha aplicação distribuída?"

A especificação EJB traz para o desenvolvedor uma solução padrão para implementação da "lógica de negócio" em um servidor de aplicações, resolvendo com isso problemas comuns como persistência, integridade transacional e segurança (que acabam sendo repetidas pelos programadores em cada aplicação corporativa). Assim o desenvolvedor pode focar no problema sem se preocupar com a infra-estrutura necessária para a solução (veremos que isso não é tão simples assim).

Um servidor de aplicações Java que seja compatível com a especificação EJB deve prover entre outras coisas:

- Processamento de transações;
- integração com o serviço de persistência disponibilizado pela Java Persistence API (JPA);
- controle de concorrência;

- eventos através do Java Message Service (JMS);
- naming and directory services JNDI;
- segurança (JCE e JAAS);
- chamada de métodos remotos (RMI-IIOP);
- métodos da camada de negócios expostos como Web Services;
- chamada de métodos de forma assíncrona;
- serviços de agendamento;
- capacidade de extensão através de componentes adicionais.

Além disso a especificação define o papel do servidor de aplicação (JBoss, WebLogic, GlassFish e etc...) e do próprio EJB. A especificação define também como um EJB é "entregue" a um servidor de aplicação e como este último desempacota e expões as regras de negócio contidas no pacote.

Com todos os benefícios citados acima, por que então surgiram tantas alternativas à especificação? Logo no início, após adoção por grande parte da comunidade (principalmente grandes companhias) os problemas começaram a surgir:

- Alguns desenvolvedores começaram a sentir que a API oferecida pelo padrão EJB era por demais complexa (muito mais do que deveria ser);
- uma grande quantidade de exceções checadas (uma das premissas do framework Spring é diminuir ao máximo as exceções checadas) que dificultavam a criação e manutenção do código;
- obrigatoriedade de estender algumas classes e implementar outras tantas interfaces (contrariamente ao que hoje chamamos de Plain Old Java Objects POJO);
- problemas de performance - a especificação original só permitia o protocolo CORBA - com chamadas de métodos remotos mesmo que a aplicação não fizesse uso dos mesmos. Isso foi melhorado mais tarde através das interfaces locais;
- curva de aprendizado acentuada;

A especificação JSR 318 (EJB 3.1) resolve se não todos, uma grande parte dos problemas relatados, e frameworks como Hibernate e Spring completam a lacuna, permitindo ao desenvolvedor o foco na aplicação.

Programar de forma correta e eficaz em Java não é fácil, exige um forte embasamento em orientação a objeto, o domínio da sintaxe, estrutura semântica e conhecimento das bibliotecas disponíveis (não é necessário conhecer todas as bibliotecas mas algumas como coleções e IO são indispensáveis para a construção de sistemas robustos). É muito comum encontrar problemas de origem conceitual em programadores que não têm a base de conhecimento necessária. A escolha do framework correto pode auxiliar muito na construção e entrega do software. A seguir falaremos do framework Spring que permite uma grande produtividade e facilidade de criação de casos de teste.

3.2.2 Framework web – Spring

O Spring foi o framework escolhido para desenvolver o sistema apresentado. Ele é o framework de código aberto mais usado por desenvolvedores Web e que permite a criação de aplicações com alta qualidade e curto prazo de desenvolvimento.

Fundamentado no conceito de convenção ao invés de configuração (COC) provê um modelo consistente de programação e configuração fortemente baseado em padrões. Diferente da tradicional plataforma Java EE, o Spring oferece um amplo leque de opções para a criação de aplicações Web ricas e para o consumo das mesmas através de um sistema leve e facilmente configurável.

Os principais componentes para o desenvolvimento de aplicações com o Spring são:

- Spring para aplicações corporativas – Web services, transações, segurança, chamadas de métodos remotos, serviço de mensagens, acesso a dados, programação orientada a aspectos (AOP) e muito mais. O framework Spring é fortemente baseado no conceito de Inversão de Controle IoC (Martin Fowler) algumas vezes chamado também de injeção de dependência. O IoC se traduz em componentes que podem ser facilmente configurados (de forma centralizada) e unidos uns aos outros sem comprometer o acoplamento, resultando em código que é mais portável, reutilizável, testável e manutenível.

- Spring para aplicações Web ricas – Criação de aplicações com interface rica e grande usabilidade. Gerenciamento de fluxo e navegação em alto nível permitindo a criação de aplicações mais interativas.

O sistema desenvolvido exemplifica e documenta o uso e configuração do Framework Spring e sua integração ao servidor de aplicações open source JBoss.

3.2.3 Casos de teste

Não custa nada ser repetitivo, desenvolver qualquer aplicação sem criar os testes de unidade e integração termina em falta de qualidade e um código que dificilmente será refatorado.

Qualquer nova funcionalidade acrescentada ou removida por causa de mudanças nos requisitos (eles mudam sempre...) deve ser testada de forma unitária e como um todo. Como fazer isso se não criamos os testes?

Mas desenvolver código orientado a testes não é simplesmente sair escrevendo testes e a partir das falhas implementar métodos e classes. Escrever código orientado a testes impõe um sistema modularizado com interfaces bem definidas e um framework que permita a injeção de código de teste.

Existem muitas ferramentas disponíveis para essa tarefa, mas o destaque fica para o JUnit como framework de testes e o EasyMock para criação de objetos sob demanda. Eles se integram bem ao framework Spring, facilitando ainda mais a automação dos testes.

3.2.4 Testes de Banco de Dados

É muito importante obter uma massa de dados iniciais que corrobore a estrutura criada e possa validar os relacionamentos e regras de negócio:

- Validar os índices únicos;
- validar as chaves candidatas e primárias;
- validar os relacionamentos estrangeiros;
- validar o tamanho máximo e mínimo dos campos;
- validar os campos que não podem ser nulos;
- validar os valores de auto-preenchimento;
- validar os cascateamentos;
- verificar a criação de índices que possam melhorar a performance das consultas.

Verificar se alguma informação foi perdida após a normalização, e tentar criar consultas que exibam os dados de forma a satisfazer os requisitos funcionais. No projeto apresentado um arquivo de configuração será construído a partir de valores obtidos em várias tabelas no banco de dados. É importante criar consultas que possam obter os valores com facilidade e no formato desejado. O arquivo de redirecionamentos apresenta quatro campos:

```
REDIRECT    PERMANENT        /URL_ANTIGA        HTTP://URL_NOVA
```

A consulta a seguir permite obter os valores no formato exigido pelo servidor Web.

```
SELECT
  'Redirect' AS C1,
  CASE WHEN r.status=301 THEN 'permanent'
  END AS C2,
  '/' || r.url_antiga AS C3,
  'http://' || s.url_sitio || '/' || r.url_nova AS C4
FROM
  public.redirecionamento r,
  public.sitio s
WHERE
  s.id = r.id_sitio
  AND s.sigla = 'GNT'
ORDER BY r.url_antiga;
```

3.2.5 PROTOTIPAÇÃO

Para auxiliar na criação dos casos de uso, fizeram-se protótipos que permitiam discutir e exemplificar as propriedades de cada objeto e até o seu comportamento. Não existem muitas ferramentas baratas ou gratuitas para prototipação Como alternativa, podemos usar uma ferramenta de desenho vetorial como o Inkscape ou o LibreOffice.

O sistema GCRedirect foi prototificado com um complemento do do navegador Web Firefox chamado Pencil e que permite de forma fácil criar e organizar as várias telas de sistema (as telas podem inclusive conter dados que vão facilitar a validação do modelo). Os protótipos são geralmente anexados aos casos de uso para ilustrar e documentar os requisitos.

3.2.6 UML

Em sala muito se falou sobre documentação e desenvolvimento ágil. Ficou claro que documentar sem razão é um castigo, mas documentar nada não é plausível. Faz-se necessário encontrar o equilíbrio correto e gerar os artefatos necessários ao entendimento, validação e manutenção do sistema.

A UML 1.4 apresenta oito tipos diferente de diagramas:

- Diagrama de Classe;
- diagrama de Caso de Uso;
- diagrama de Atividade;
- diagrama de Máquina de Estado;
- diagrama de Sequência;
- diagrama de Colaboração;

- diagrama de Componente;
- diagrama de Implantação.

Na UML 2.0 esse número cresceu para treze (alguns foram modificados):

- Diagrama de Atividade;
- diagrama de Classe;
- diagrama de Comunicação;
- diagrama de Componentes
- diagrama de Estrutura Composta;
- diagrama de Implantação;
- diagrama de Interação;
- diagrama de Objeto;
- diagrama de Pacote;
- diagrama de Sequência;
- diagrama de Máquina de Estado;
- diagrama de Tempo;
- diagrama de Caso de Uso.

Acreditamos que um sistema deva ser acompanhado dos seguintes artefatos:

- Casos de Uso – Um caso de uso documenta requisitos. Além do diagrama fornecido pela UML deve vir acompanhado do texto que ilustra a história e iteração do usuário com o sistema. Pode ser enriquecido com cenários alternativos e protótipos;
- Diagramas de Classe – Não existe sistema bem feito sem o diagrama de classes! É o mapa do sistema, não deve ser poluído (por isso a importância do uso de pacotes). É possível criar mais de um diagrama de classes usando diferentes níveis de abstração;
- Diagramas de Atividade – Alguns sistemas mais complexos se beneficiam de um diagrama de atividades (principalmente aqueles que fazem uso de operações automatizadas e em lotes);

- Diagramas de Sequência - Explica o caso de uso e relaciona às classes do domínio. Nem todos os casos de uso necessitam de um diagrama de sequência, porém ao documentar uma API, esse é um diagrama obrigatório;

- Diagrama de Implantação – Obrigatório para que a infra-estrutura possa configurar e instalar o sistema. Auxilia também na identificação de problemas em produção.

Mesmo não estando relacionado com a UML vale lembrar que o diagrama de entidade e relacionamento é muito importante para a equipe responsável pela criação e manutenção das bases de dados necessárias ao sistema (dificilmente em produção um desenvolvedor irá criar as bases por conta própria). O Mini-Mundo e o documento de Análise de Riscos complementam a documentação e ajudam a entender o propósito do sistema.

3.2.7 Automatizando a Construção - Maven

O processo de criação de um sistema demanda uma série de passos, além da preparação do ambiente de forma apropriada. De todas as ferramentas experimentadas, uma se destacou: Maven. A palavra tem sua origem na língua Yiddish que é uma fusão de vários dialetos germânicos e significa acumulador de conhecimento. O Maven é geralmente considerado por muitos como uma ferramenta de construção (na verdade é muito mais do que isso).

Maven trata da aplicação de padrões para conseguir uma infra-estrutura com visibilidade, reusabilidade, manutenibilidade e compreensão.

Sem essas características é muito improvável que vários indivíduos possam trabalhar em conjunto no desenvolvimento de um sistema. Como é possível sem visibilidade que alguém possa saber o que outra pessoa está fazendo e com isso reutilizar o seu trabalho?

Ao criar um projeto usando os recursos do Maven, automaticamente somos presenteados com uma estrutura de pastas padrão (inclusive para os casos de teste) e configuração para documentação do projeto na forma de um sítio Web.

As várias bibliotecas das quais um projeto depende podem ser especificadas como artefatos que serão automaticamente baixados e anexados ao projeto, respeitando as versões definidas na configuração.

O sistema desenvolvido exemplifica o uso do Maven em integração com a IDE Eclipse

3.2.8 Ambiente de Desenvolvimento Integrado

É possível desenvolver uma aplicação Java usando um simples editor de textos, mas quando se trata de um time de desenvolvimento, algumas coisas tornam-se mandatórias:

- Versionamento – Não é possível trabalhar com uma equipe de desenvolvimento sem a possibilidade de versionar o código. A melhor ferramenta disponível hoje é gratuita e se chama Git;
- Refatoração – Uma boa IDE permite que a refatoração seja feita quase automaticamente;
- Comparação – Deve ser possível através de uma ferramenta visual comparar mudanças no código entre diferentes versões;
- Consistência na formatação – Deve ser possível configurar uma IDE para que modelos sejam aplicados durante a criação do código (os modelos devem ser consistentes e usados por todos na equipe);
- Depuração – Mesmo com casos de testes algumas vezes a depuração é obrigatória;
- Testes – Integração com frameworks de teste;
- Documentação – Possibilidade de gerar a documentação automaticamente;
- Dicas de código e auto-completar – Não é obrigatória mas aumenta a produtividade;

No desenvolvimento do sistema escolhemos a IDE Eclipse por ser uma ótima ferramenta e praticamente um consenso no mercado.

3.2.9 Segurança – Autenticação e Autorização

O JavaTM Authentication and Authorization Service (JAAS) foi introduzido como um pacote opcional a partir da versão 1.3. Ele é usado com dois propósitos:

- Autenticação de usuários – Verificação das credencias;
- Autorização de usuários – Verificação se o usuário autenticado está também autorizado para realizar determinada operação.

Não foi usado diretamente o JAAS no desenvolvimento do sistema, optou-se pelo uso do framework Spring Security que facilita muito a configuração da segurança em uma aplicação Web. O Spring Security permite a configuração de segurança independente do servidor de aplicação utilizado. Assim é possível por exemplo rodar a mesma aplicação no JBoss ou GlassFish sem mudanças significativas.

Exemplos de configuração para segurança estão disponíveis e comentados no sistema desenvolvido servindo como base para outras aplicações Web.

3.2.10 Open Source

Defende-se a utilização de padrões abertos pois acredita-se que contribuem para o enriquecimento do conhecimento da comunidade de desenvolvimento. Por exemplo, ter acesso ao código usado no framework Spring possibilitou um fácil entendimento e guiou a construção e a extensão das classes de forma integrada ao modelo apresentado.

3.2.11 Registrando as operações do sistema

Um importante recurso das linguagens modernas é a capacidade de gerar registros detalhados das operações que estão sendo realizadas. O que antes era feito de forma manual pelo programador (os arquivos eram criados, abertos, o conteúdo adicionado e finalmente fechados) pode agora ser feito e ajustado inclusive após a entrega do sistema.

As instruções de registro foram removidas do código e transferidas para arquivos de configuração onde podem ser ajustadas em tempo de execução. O framework de registro mais usado no mundo Java é o Log4J, ele permite um controle preciso de como os registros serão feitos dividindo os mesmos em seis níveis diferentes:

1. Palavra-chave - *trace* (menos importante);
2. palavra-chave - *debug*;
3. palavra-chave - *info*;
4. palavra-chave - *warn*;
5. palavra-chave - *error*;
6. palavra-chave - *fatal* (mais importante)

Conhecer e saber aplicar esses níveis em um sistema real é vital para rastrear erros em produção e registrar mudanças com fins de auditoria. É comum encontrar sistemas em produção que registram tudo em uma única categoria, tornando a depuração impossível e gerando arquivos gigantescos que acabam por lotar o sistema de arquivos.

A categorização dos registros em níveis requer também o conhecimento e uso apropriado de um mecanismo da linguagem Java: os pacotes. Um pacote é um espaço de nomes que agrupa e organiza as funcionalidades de uma ou mais classes. As classes em um pacote devem ter algum tipo de *afinidade.* Os pacotes, quando bem projetados, melhoram a organização do sistema.

É comum iniciar o nome do pacote com um identificador único, normalmente o registro da empresa na Web em forma reversa (infnet.edu.br torna-se br.edu.infnet) acrescentando-se o nome do sistema. A partir desse ponto, sufixos específicos que agrupam funcionalidades são acrescentados (*.security*, *.view*, *.model*, *.controller*, *.service*, *.util*, etc...). É boa prática manter esses sufixos em Inglês, já que permitem indicar padrões organizacionais comuns.

Com os pacotes e categorias planejados e definidos, fica fácil registrar todas as operações realizadas em uma categoria/pacote. A seguir, um exemplo de como os registros de auditoria para o sistema serão criados:

```
<!-- Cria-se um appender que será referenciado pelo seu "name" -->
<appender name="gcredirect.security" class="org.apache.log4j.FileAppender">
	<!-- Nome do arquivo de saída -->
	<param name="File" value="/gcredirect.security.log"/>
	<param name="Append" value="false"/>
	<layout class="org.apache.log4j.PatternLayout">
		<param name="ConversionPattern" value="%d [%t] %p - %m%n"/>
	</layout>
</appender>

<!-- A categoria aponta para um pacote no sistema (pode ser uma classe
também) -->
<category name="br.com.globosat.gcredirect.security">
	<!-- Apenas o nível INFO-->
	<priority value="INFO" />
	<!-- Associamos ao appender criado -->
	<appender-ref ref="gcredirect.security" />
</category>
```

O mais importante é que as configurações podem ser alteradas em tempo de execução sem necessidade de recompilação.

Para o sistema desenvolvido, os registros de mudança como criação e remoção ficaram com a categoria *info*. Durante o ciclo de desenvolvimento informações detalhadas eram exibidas alterando-se simplesmente o nível de saída para *debug*.

É sempre bom lembrar que uma aplicação que é executada junto com outras em um servidor de aplicações como o JBoss tem seu nível de registro configurado pelo servidor, e o mesmo deve ser acordado com a infra-estrutura de suporte (dificilmente é permitido o nível *debug* em um ambiente de produção, devido à sobrecarga no sistema de arquivos).

3.2.12 Sistema Operacional

O sistema operacional é um importante fator no sucesso de um projeto. É fundamental saber qual o ambiente alvo antes de iniciar a construção de um sistema (esse conhecimento deve estar documentado na análise de riscos). Escolheu-se o Ubuntu - uma distribuição baseado no Debian - como SO para o sistema em desenvolvimento.

3.2.13 Documentação

É possível confiar em um GPS cujos mapas não são atualizados? Será que as ruas mudaram de mão? É mais ou menos assim que funciona uma documentação não atualizada. Se for para não manter atualizada, é melhor não documentar. Quando se faz manutenção em código não documentado, nos obrigamos a investigar e descobrir o que cada função faz no

sistema. É demorado, mas não se corre o risco de assumir que determinada parte do código vai afetar o restante após uma modificação.

Se o código apresenta documentação defasada e acabamos por confiar no mesmo os resultados serão imprevisíveis. Como manter então a documentação sincronizada com os artefatos? Citando novamente o autor Martin Fowler, gerando código auto documentado:

- Classes, métodos e funções com nomes retirados do domínio da aplicação;
- nomear as variáveis e propriedades de forma clara, dar preferência à criação das variáveis no ponto onde serão usadas;
- evitar o acoplamento (grau de dependência entre dois artefatos);
- um método ou função deve executar uma única tarefa. Evitar métodos que façam mais do que o seu nome sugere e que acabam tendo que ser comentados para explicar o seu comportamento incomum;
- evitar usar classes como repositório de funções estáticas (transforma-se programação orientada a objetos em programação procedural em nível de classes);
- criar classes específicas que reflitam as regras do negócio;
- criar uma hierarquia de exceções que traduzam as quebras de contrato de forma clara;
- usar assertivas nas partes críticas do código. Assertivas promovem um fácil entendimento do código e ilustram o que o programador esperava no ponto onde foram inseridas;
- documentar apenas o que não é óbvio.

É claro que existem exceções. Uma *Application programming interface* (API) por exemplo, deve ter seus métodos, classes e espaços de nome muito bem documentados. Mas diferente de um sistema, uma API costuma ter um contrato muito rígido que não pode ser quebrado sem perda de retro-compatibilidade.

3.2.14 Servidor de aplicação

Existem vários servidores de aplicação Java EE homologados:

- GlassFish (GlassFish Community);
- WebSphere Application Server (IBM);
- JBoss AS (RedHat);

- WebLogic Server (Oracle Corporation BEA Systems).

O servidor de aplicação usado como referência para os padrões Java EE é o GlassFish, mas no mercado, a crescente utilização do JBoss AS é notável (o futuro do WebLogic e WebSphere tem sido inclusive alvo de muitas especulações).

No JBoss sete muita coisa mudou, principalmente na modularidade e na configuração dos diversos componentes que constituem um servidor de aplicação. O servidor se torna disponível em uma máquina modesta em apenas três segundos, lembrando que a versão mais usada até hoje, JBoss 4.2.3, chega a levar até dois minutos para disponibilizar as mesmas funcionalidades.

Recomenda-se a leitura do manual disponível no próprio sítio da RedHat que exemplifica a configuração das fontes de dados e dos registros de depuração e auditoria. Novamente, o sistema desenvolvido contém exemplos de fontes de dados e configuração dos registros para o JBoss sete.

3.2.15 A Importância do Diagrama de Entidade Relacionamento

É comum o argumento de que com o advento dos frameworks de mapeamento objeto-relacional não seja mais necessário a preocupação com a criação dos diagramas de entidade relacionamento e dos dicionários de dados associados. Isso tem levado a criação de sistemas que uma vez em produção começam a apresentar problemas de performance.

O DER permite uma visão dos relacionamentos de alto-nível e documenta decisões importantes do projeto. Com o DER e junto a um DBA podemos discutir melhores formas de obter os dados e talvez até reestruturar os relacionamentos.

Ao normalizar as tabelas, é importante também verificar o quanto essa normalização vai custar em termos de performance. Muitas vezes é melhor não normalizar (com cuidado é claro) para evitar um número exagerado de junções que podem diminuir muito a performance. Observando o DER podemos também verificar redundâncias e valores do tipo *point-in-time* que não devem ser normalizados.

3.2.16 Consistência

É importante manter a consistência. Convencionar a forma como classes, métodos e propriedades serão nomeados ajuda a estabelecer uma codificação e documentação que pode ser reconhecida e transmitida para os novos membros da equipe.

Detalhes como espaçamento e tabulações são importantes para estabelecer um aspecto visual agradável e facilitar a manutenção do código. Depois de estabelecido, o padrão deve ser respeitado por todos (o padrão escolhido deve ser agradável visualmente e confortável ao uso).

Todos na equipe devem ter acesso a modelos que facilitem a criação de novos tipos, por isso recomenda-se o uso de uma IDE que possibilite a criação prévia desses modelos.

4 CONCLUSÃO

A construção de um sistema (por mais simples que o mesmo seja) envolve muitas disciplinas. Começamos com um problema e vamos passo a passo desenhando uma solução que atenda plenamente os requisitos levantados.

O mais difícil é identificar os detalhes que permeiam os requisitos e que muitas vezes vão impactar na performance e na usabilidade do produto final. Identificar esse requisitos não funcionais com eficiência é o resultado de um execício contínuo.

É necessário humildade para reconhecer que escrever código com qualidade só vem com o tempo, o aprendizado não é instantâneo e deve ser sempre atualizado e enriquecido.

Temos visto um grande número de projetos escritos em Java que fracassaram por problemas de desempenho e gerenciamento de memória. Se a máquina virtual Java aloca e desaloca memória automaticamente, como podemos ter problemas de memória? Isso não era para acontecer apenas com linguagens como C ou C++? A máquina virtual não faz milagres (por mais que os engenheiros estejam melhorando e otimizando o Garbage Collector (GC)) e para evitar esse tipo de problemas é muito importante conhecer conceitos básicos da linguagem muitas vezes menosprezados! Quantos desenvolvedores se preocupam em conhecer o mecanismo de encaixotamento, entender como laços são otimizados e como evitar problemas de referências cíclicas em coleções usando WeakReference? A linguagem Java encontra-se em constante evolução. Com as novas APIs do Java EE 6 como CDI, é possível que no futuro as aplicações necessitem menos de frameworks de terceiros para resolver os problemas comuns em sistemas corporativos.

Acreditamos na importância de criar-se uma cultura onde o conhecimento acumulado é mantido e documentado no intuito de melhorar constantemente a qualidade. Hoje dispomos de ferramentas de qualidade muitas delas gratuitas e abertas; resta aprender como usá-las de forma eficiente.

Espera-se que o sistema aqui desenvolvido possa ajudar na maneira de identificar as várias partes componentes de um sistema real e como usar os artefatos disponíveis para juntar tudo em um produto funcional.

Procurou-se documentar o código e os arquivos de configuração em um formato que torne fácil aplicar partes do sistema em outros que estejam sendo construídos.

REFERÊNCIAS BIBLIOGRÁFICAS

SPOLSKY, Joel. The Design of Software, "Login" is not a Use Case. Disponível em: <http://discuss.joelonsoftware.com/default.asp?design.4.7340.13>. Acessado em: 5 jan. 2012, 13:15:11

FOWLER, Martin. Is Design Dead? Disponível em: <http://martinfowler.com/articles/designDead.html>. Acessado em: 5 jan. 2012, 13:45:23

FOWLER, Martin. Inversion of Control Containers and the Dependency Injection pattern. Disponível em: <http://martinfowler.com/articles/injection.html>. Acessado em: 6 jan. 2012, 17:50:00

FOWLER, Martin. Continuous Integration. Disponível em: <http://martinfowler.com/articles/continuousIntegration.html>. Acessado em: 6 jan. 2012, 18:05:32

FOWLER, Martin. Mocks Aren't Stubs. Disponível em: <http://martinfowler.com/articles/mocksArentStubs.html>. Acessado em: 2 fev. 2012, 08:20:00

FOWLER, Martin. CodeAsDocumentation. Disponível em: <http://martinfowler.com/bliki/CodeAsDocumentation.html>. Acessado em: 6 mar. 2012, 09:16:05

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph ; VLISSIDES, John. Design Patterns: Elements of Reusable Object-Oriented Software. 1 ed. USA. Addison-Wesley. 1994.

FOWLER, Martin. UML Essencial: Um breve guia para a linguagem-padrão. 3 ed. Porto Alegre. Bookman. 2005.

METSKER, Steven John. Padrões de Projeto em Java. 1 ed. Porto Alegre. Bookman. 2004.

EVANS, Eric. Domain-Driven Design: Tackling Complexity in the Heart of Software. 1 ed. Boston, MA. Addison-Wesley. 2004.

BECK, Kent. Test-Driven Development By Example. 1 ed. Boston, MA . Addison-Wesley. 2003.

ANEXOS

MINI-MUNDO

Mantendo redirecionamentos paras os sites da Globosat Programadora LTDA.

EXPLICANDO OS REDIRECIONAMENTOS

Para um servidor Web[1] um redirecionamento é constituído de uma diretiva Redirect, que mapeia uma URL[2] não mais existente para uma nova, indicando ao cliente (o navegador Web) para recarregar o recurso a partir de uma nova localização. Um servidor WEB suporta quatro tipos de redirecionamentos:

- Permanent (301) – Indica que o recurso foi movido de forma permanente;
- Temp (302) – Padrão;
- Seeother (303) – Indica que o recurso foi substituído;
- Gone (410) – Indica que o recurso foi removido permanentemente.

Os redirecionamentos que serão descritos nas próximas seções serão todos do tipo permanent (301).

Quais os benefícios no uso dos redirecionamentos?

Ao acessar a página do programa "Bem, Amigos!" do Sportv decidimos por armazenar a mesma usando gerenciador de favoritos do navegador Web, essa página será armazenada com a URL "sportv.globo.com/site/programas/bem-amigos/". Se esse URL mudar por alguma razão, ao tentarmos acessar uma segunda vez usando a URL salva, recebemos o erro 404 (esse código de retorno indica que o recurso não foi encontrado). Com um redirecionamento podemos indicar o novo endereço para o navegador evitando assim, o desagradável erro 404.

Outra razão comum para o uso dos redirecionamentos é a substituição de uma URL longa por outra mais amigável.

Alguns exemplos de redirecionamentos para o site do Sportv:

```
Redirect permanent /Sportv/2009/atletismo/0,,17045,00.html
       http://sportv.globo.com/atletismo
```

1 O sistema em questão usa como servidor Web o software da Fundação Apache HTTPD.

2 Um recurso, é uma página, imagem, áudio, vídeo, etc. Ou seja qualquer coisa que possa ser recuperada através de uma URL.

```
Redirect permanent /Sportv/2009/basquete/0,,17017,00.html
       http://sportv.globo.com/basquete

Redirect permanent /timao
       http://sportv.globo.com/videos/corinthians
```

CENÁRIO ATUAL

Hoje os redirecionamentos são pedidos para a área de infraestrutura de tecnologia de informação através de e-mails. Pode-se explicar o processo da seguinte forma simplificada:

- Um dos editores responsáveis pelo conteúdo do site envia um e-mail com o pedido de redirecionamento para um dos analistas no departamento de infraestrutura de TI;
- Ao receber o e-mail, o analista entra em contato (telefone ou e-mail) com o coordenador do site para verificar se o editor tem permissão para pedir o redirecionamento;
- Uma vez aprovado pelo coordenador do site, o redirecionamento informado no e-mail é copiado e modificado para obedecer ao formato do arquivo de configuração dos servidores Web;
- O analista conecta-se à máquina remotamente e faz uma cópia do atual arquivo de redirecionamentos;
- O analista altera o arquivo para incluir o novo redirecionamento;
- O analista recarrega os serviços WEB para que o redirecionamento tenha efeito;
- O analista responde ao e-mail de pedido informando que o redirecionamento foi feito;
- Os e-mails trocados são armazenados para o caso de uma auditoria.

Durante esse processo, alguns erros podem ocorrer: erro na escrita do redirecionamento; o destino do redirecionamento não existe; o redirecionamento já foi feito por outro analista; já existe um outro redirecionamento com a mesma origem; redirecionamento circular (origem aponta para origem causando um processo sem fim na URL).

Também é comum o pedido para remover ou alterar um redirecionamento onde os passos são semelhantes ao caso de inclusão.

O SISTEMA PROPOSTO - GCREDIRECT

Desejamos construir um sistema que permita aos editores manter os redirecionamentos sem a necessidade de intervenção de um analista de infraestrutura de TI. O sistema deve permitir que os editores sejam mantidos e que as devidas permissões possam ser aplicadas por um administrador.

O sistema deve impedir a entrada de redirecionamentos inválidos e mal formatados, assim como verificar se a URL final existe. O sistema não deve permitir redirecionamentos duplicados e circulares.

Por motivos de auditoria, um redirecionamento ao ser removido muda de status mas continua a existir no sistema (um redirecionamento para ser alterado, deve antes ser removido).

Todas as operações feitas por um editor devem ser auditadas com o tipo de operação, data e hora em que foi efetuada.

O sistema deve gerar um arquivo de redirecionamento no formato adequado ao servidor WEB e com os redirecionamentos em ordem alfabética.

O sistema deve permitir ao editor a consulta dos redirecionamentos em ordem alfabética pelo alias ou pela URL. Deve ser possível também listar os redirecionamentos removidos.

Com esse sistema desejamos transferir a responsabilidade de criação dos redirecionamentos para os editores autorizados, que por conhecerem o negócio podem fazer críticas mais seguras sobre a validade e sintaxe de um redirecionamento.

Requisitos Funcionais

- Autenticação e autorização – O sistema deve garantir que apenas os editores cadastrados possam criar ou remover redirecionamentos;
- Um editor só pode ver a lista de redirecionamentos dos canais associados a ele;
- Manter usuários – O sistema deve permitir a criação e manutenção dos usuários (editores);
- Remover um redirecionamento – O sistema deve permitir ao editor remover um redirecionamento (o status do redirecionamento muda para removido);
- Criar um redirecionamento - O sistema deve permitir ao editor criar um novo redirecionamento;
- Não criar redirecionamentos duplicados – O sistema deve verificar durante a criação se não existe um redirecionamento anterior com o mesmo alias;
- Não criar redirecionamentos circulares – O sistema deve verificar durante a criação se não existe um destino com o mesmo alias a ser criado evitando um redirecionamento circular;
- Listar os redirecionamentos em ordem alfabética – O sistema deve permitir a listagem dos redirecionamentos em uso por ordem alfabética do alias ou da URL;
- Auditar a criação dos redirecionamentos – O sistema deve auditar a criação dos redirecionamentos (nome do editor, data, hora e operação);
- Auditar a remoção dos redirecionamentos – O sistema deve auditar a criação dos redirecionamentos (nome do editor, data, hora e operação);
- Criar os arquivos de redirecionamentos dos servidores WEB – O sistema deve criar o arquivo de redirecionamentos em ordem alfabética de alias e no formato correto para o servidor WEB;
- Listar os redirecionamentos removidos em ordem alfabética – O sistema deve permitir a listagem dos redirecionamentos em uso por ordem alfabética do alias ou da URL;

REQUISITOS NÃO FUNCIONAIS

- Não permitir a mudança do nome de autenticação do usuário editor – O nome completo, e-mail e ramal podem ser alterados pelo administrador mas o nome usado para autenticação deve ser mantido para efeito de auditoria;
- O usuário administrador será criado manualmente e não pode ser removido (a senha pode ser alterada posteriormente);
- O sistema deve gerar LOGS de todas as operações relacionadas ao armazenamento na categoria INFO. Esses LOGS devem ser direcionados para uma arquivo;
- O sistema deve gerar o backup do arquivo de redirecionamentos antes de efetuar qualquer alteração;
- O formato para nome do arquivo de backup deve ser na forma: **`SIGLA_SÍTIO-REDIRECT-DIA-MES-ANO-HORA-MINUTO-SEGUNDO.backup`**;
- Ao recarregar as configurações informar o sucesso ou falha na leitura do arquivo de redirecionamentos;
- Os redirecionamentos devem ser armazenados em um banco de dados relacional e a partir do mesmo o arquivo final será criado;
- O acesso ao sistema será pela rede interna no formato Web;
- A autenticação será feita através de um banco de dados relacionais;

ANÁLISE DE RISCO

Através da análise de risco podemos descobrir se a construção do software é viável ou não. O sistema para manutenção dos redirecionamentos não está sendo projetado com objetivo em lucro e sim em resolver uma demanda para o departamento de infraestrutura de TI. Mesmo assim deve-se levar em conta os custos com horas de trabalho da equipe que vai levantar os requisitos e desenvolver o sistema em questão.

Como o sistema roda em um ambiente remoto, devemos levar em conta a segurança, a conectividade e os possíveis atrasos de comunicação. Assim é importante pensar na possibilidade de recuperação de falhas e como comunicar esses incidentes para os envolvidos. Os pontos críticos identificados são:

- Executar comandos remotos através de um túnel SSH;
- Liberação dos acessos entre a Globo.com e a Globosat (através de uma VPN);
- A operação de criação dos arquivos de configuração dos servidores Web não pode deixar os mesmos corrompidos;
- Notificação de erros ocorridos;
- Recuperação das falhas.

Esboço de Telas Auxiliares

Tela Pós-autenticação para o Administrador

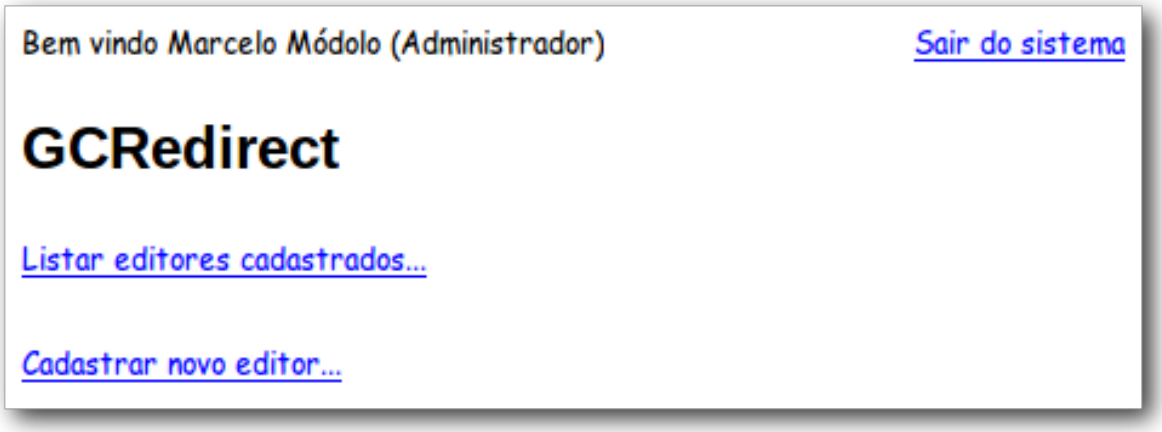

Ilustração 3: Esboço de Tela: Administrador Pós-autenticação

CASOS DE USO

VISÃO GERAL (LÓGICA) UML

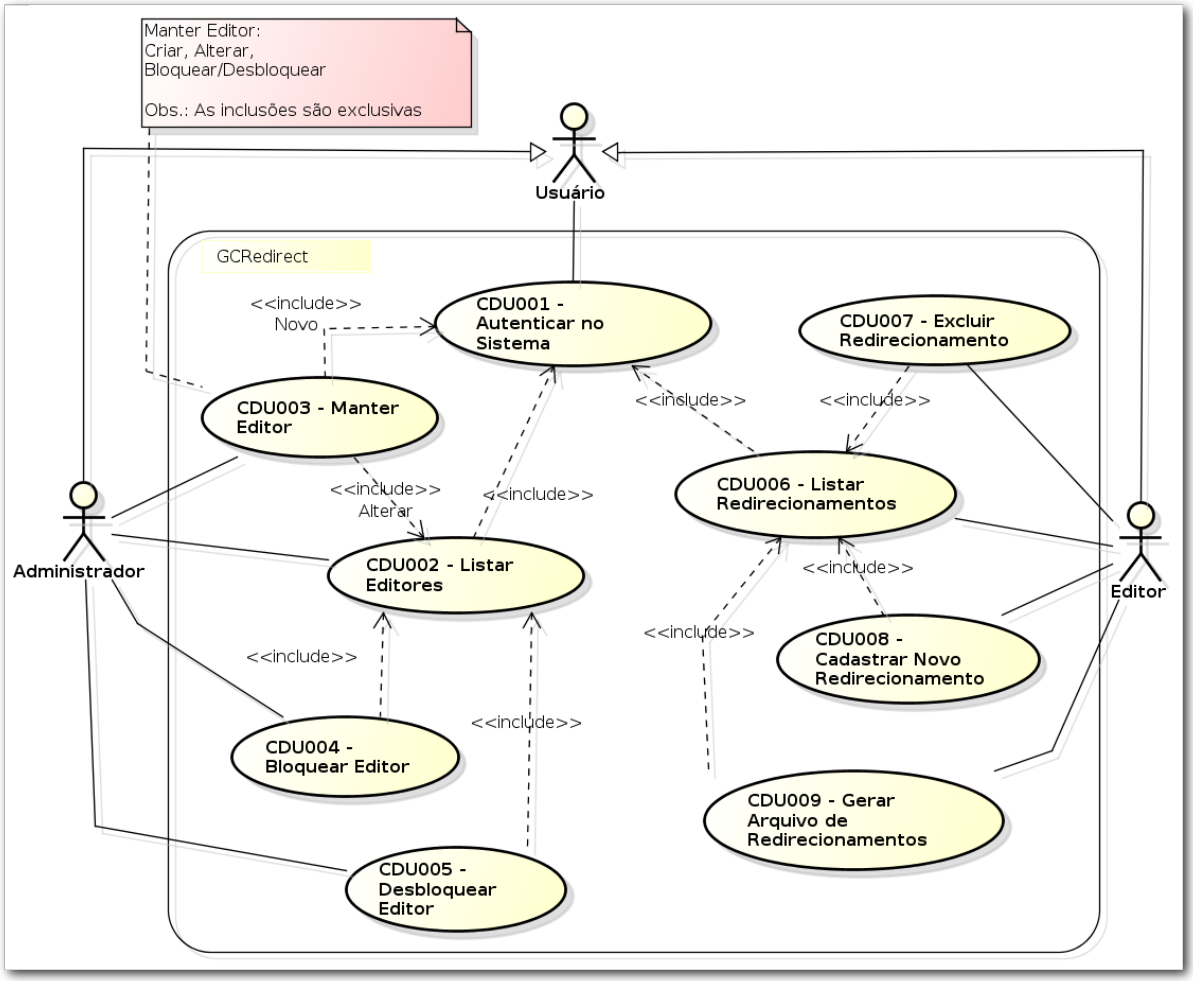

ILUSTRAÇÃO 4: VISÃO GERAL (LÓGICA) UML

O Caso de uso “CDU003 – Manter Editor” agrupa as metas:

- Criar um novo editor;
- Alterar as propriedades de um editor previamente cadastrado;
- Bloquear/desbloquear um editor;

Esse caso de uso pode ser “alcançado” após autenticação no sistema para criação de novos editores ou pela exibição de editores para alterar as propriedades de um editor, por isso as duas inclusões.

CDU001 – AUTENTICAR NO SISTEMA

Especificação de Caso de Uso – Tipo 1

Versão 1.0

Última atualização: 04/03/2012

Histórico de revisão do documento

Data	Versão	Responsável	Descrição da alteração
04/01/2012	**1.0**	Marcelo Rezende Módolo	Criação

1. Descrição breve

O caso de uso "Autenticar no sistema" é iniciado quando o editor ou o administrador desejam acesso ao sistema.

2. Atores

Editor ou Administrador, aqui chamados de usuários.

3. Condições Prévias (ou pré-condições)

Nenhuma

4. Fluxo de Eventos

4.1 Fluxo Básico

1. Usuário acessa a URL do sistema
2. Sistema exibe a página de autenticação
3. Usuário informa login, senha e confirma;
4. Sistema verifica que o usuário está autenticado e autorizado
5. Sistema registra a autenticação
6. Sistema exibe a página inicial

4.2 Fluxos Alternativos

3a. Usuário deixa de informar usuário, senha ou ambos

1. Sistema rejeita a entrada e exibe o erro

4a. Sistema verifica que o usuário não está autenticado

1. Sistema informa que o usuário ou senhas são inválidos

4b. Sistema verifica que o usuário não está autorizado

1. Sistema informa que o usuário não está autorizado

5. Pós-Condições

Usuário autenticado, autorizado e registro de auditoria criado

6. Requisitos Especiais

Todo editor deve estar associado a um ou mais canais e seu status deve ser ativo para que a autorização seja permitida.

7. Esboço de Protótipo e Regras de Interface

7.1 Esboço de Protótipo

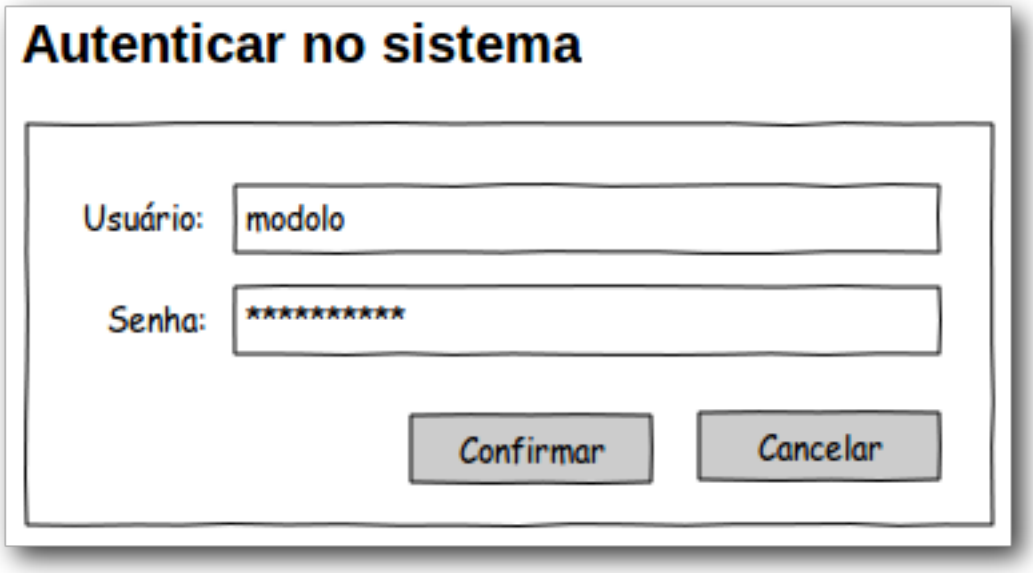

ILUSTRAÇÃO 5: ESBOÇO DE TELA: AUTENTICAR NO SISTEMA

7.2 Regras de Interface

- Nome de usuário: mínimo de 5 e máximo de 12 caracteres;
- Senha: mínimo de 8 e máximo de 16 caracteres;

- Número de tentativas: Como o sistema não tem acesso externo, não existe limite para o número de entradas inválidas.

CDU002 – LISTAR EDITORES

Especificação de Caso de Uso – Tipo 1

Versão 1.0

Última atualização: 04/03/2012

Histórico de revisão do documento

Data	Versão	Responsável	Descrição da alteração
10/01/2012	**1.0**	Marcelo Rezende Módolo	Criação
12/01/2012	**1.1**	Marcelo Rezende Módolo	Atualização

1. Descrição breve

O caso de uso "Listar Editores" é iniciado quando o administrador deseja exibir a lista de editores previamente cadastrados no sistema (Listar editores cadastrados no sistema).

2. Atores

Administrador.

3. Condições Prévias (ou pré-condições)

O Administrador já se autenticou no sistema e escolheu a opção "Listar editores cadastrados no sistema...".

4. Fluxo de Eventos

4.1 Fluxo Básico

1. Administrador escolhe "Listar editores cadastrados no sistema..."

2. Sistema exibe a página com a lista de editores cadastrados no sistema, as informações exibidas são: nome, e-mail e telefone.

4.2 Fluxos Alternativos

Nenhum

5. Pós-Condições

Nenhuma

6. Requisitos Especiais

Nenhum

7. Esboço de Protótipo e Regras de Interface

7.1 Esboço de Protótipo

ILUSTRAÇÃO 6: ESBOÇO DE TELA: LISTAR EDITORES

7.2 Regras de Interface

- Permitir sair do sistema;
- Exibir as propriedades:
  - Nome;
  - E-mail;
  - Telefone
- Permitir as ações:
  - Bloquear (bloqueia o acesso do editor ao sistema) caso o editor esteja desbloqueado;
  - Desbloquear (permite acesso ao sistema) caso o editor esteja bloqueado;
  - Alterar (permite alterar as propriedades do editor);

- Criar um novo editor (permite adicionar um novo editor ao sistema)

CDU003 – MANTER EDITOR

Especificação de Caso de Uso – Tipo 1

Versão 1.0

Última atualização: 04/03/2012

Histórico de revisão do documento

Data	Versão	Responsável	Descrição da alteração
12/01/2012	**1.0**	Marcelo Rezende Módolo	Criação
12/01/2012	**1.1**	Marcelo Rezende Módolo	Atualização
13/01/2012	**1.2**	Marcelo Rezende Módolo	Atualização

1. Descrição breve

O caso de uso "Manter editor" é iniciado quando o administrador deseja cadastrar um novo editor no sistema ou alterar as propriedades de um editor previamente cadastrado.

2. Atores

Administrador.

3. Condições Prévias (ou pré-condições)

O Administrador já se autenticou no sistema e escolheu uma das opões:

- Cadastrar novo editor...;
- Alterar... (a partir do caso de uso CDU002 – Exibir Editores)

4. Fluxo de Eventos

4.1 Fluxo Básico

1. Administrador escolhe "Cadastrar novo editor..."
2. Sistema exibe a página com as propriedades para o editor:
    - Nome completo (obrigatório);
    - E-mail (obrigatório);
    - Telefone (obrigatório);

- Nome de usuário (obrigatório);
- Senha (obrigatório);
- Confirmar senha (obrigatório);
- Lista de sítios (GNT, Multishow, etc...);
- Editor bloqueado

3. O Administrador informa as propriedades e confirma;

4. O sistema valida a entrada e grava as informações do editor;

5. O sistema informe que o editor foi atualizado com sucesso

4.2 Fluxos Alternativos

1a. Administrador escolhe “Alterar...”;

1b. O sistema exibe a página com as propriedades do editor preenchidas

4a. O sistema verifica que as informações não são válidas;

4b. O sistema informa ao administrador que as informações não estão corretas

5. Pós-Condições

Informações do editor salvas com sucesso.

6. Requisitos Especiais

Nenhum

7. Esboço de Protótipo e Regras de Interface

7.1 Esboço de Protótipo

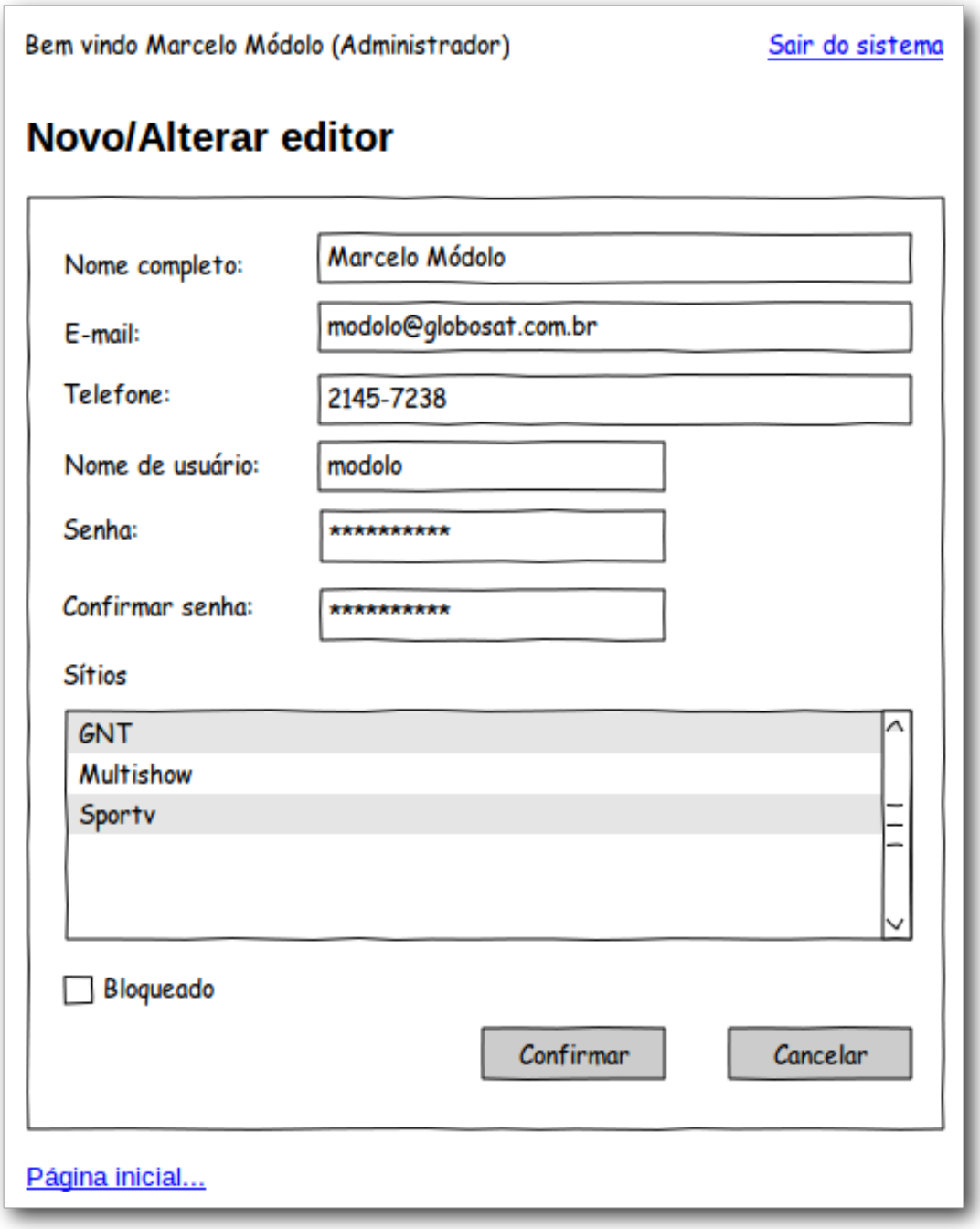

ILUSTRAÇÃO 7: ESBOÇO DE TELA: MANTER EDITOR

7.2 Regras de Interface

- Nome completo, e-mail, telefone, nome de usuário, senha e confirmação de senha são obrigatórios;
- Nome de usuário: mínimo de 5 e máximo de 12 caracteres;

- Senha: mínimo de 8 e máximo de 16 caracteres;
- Telefone e e-mail são validados quanto ao ser formato;
- É possível associar um editor a zero ou mais sítios;
- O editor pode ser bloqueado, o que impede que possa se autenticar no sistema.

CDU004 – BLOQUEAR EDITOR

Especificação de Caso de Uso – Tipo 1

Versão 1.0

Última atualização: 04/03/2012

Histórico de revisão do documento

Data	Versão	Responsável	Descrição da alteração
13/01/2012	**1.0**	Marcelo Rezende Módolo	Criação

1. Descrição breve

O caso de uso "Bloquear Editor" é iniciado quando o administrador deseja bloquear o acesso de um editor ao sistema.

2. Atores

Administrador.

3. Condições Prévias (ou pré-condições)

O Administrador já se autenticou no sistema e escolheu a opção "Bloquear" do caso de uso CDU002 – Exibir Editores.

4. Fluxo de Eventos

4.1 Fluxo Básico

1. Administrador escolhe a opção "Bloquear" do caso de uso CDU002 – Exibir Editores;

2. O sistema altera a situação do editor para bloqueado.

3. O sistema informa que o editor foi bloqueado.

4.2 Fluxos Alternativos

Nenhum

5. Pós-Condições

Nenhuma

6. Requisitos Especiais

Nenhum

7. Esboço de Protótipo e Regras de Interface

Ver o esboço de tela para o caso de uso CDU003 – Manter Editor.

7.2 Regras de Interface

A opção “Bloquear” só é exibida quando o editor estiver desbloqueado.

CDU005 – DESBLOQUEAR EDITOR

Especificação de Caso de Uso – Tipo 1

Versão 1.0

Última atualização: 04/03/2012

Histórico de revisão do documento

Data	Versão	Responsável	Descrição da alteração
13/01/2012	**1.0**	Marcelo Rezende Módolo	Criação

1. Descrição breve

O caso de uso "Desbloquear Editor" é iniciado quando o administrador deseja permitir o acesso de um editor ao sistema.

2. Atores

Administrador.

3. Condições Prévias (ou pré-condições)

O Administrador já se autenticou no sistema e escolheu a opção "Desbloquear" do caso de uso CDU002 – Exibir Editores.

4. Fluxo de Eventos

4.1 Fluxo Básico

1. Administrador escolhe a opção "Desbloquear" do caso de uso CDU002 – Exibir Editores;

2. O sistema altera a situação do editor para desbloqueado.

3. O sistema informa que o editor foi desbloqueado.

4.2 Fluxos Alternativos

Nenhum

5. Pós-Condições

Nenhuma

6. Requisitos Especiais

Nenhum

7. Esboço de Protótipo e Regras de Interface

Ver o esboço de tela para o caso de uso CDU003 – Manter Editor.

7.2 Regras de Interface

A opção “Desbloquear” só é exibida quando o editor estiver bloqueado.

CDU006 – LISTAR REDIRECIONAMENTOS

Especificação de Caso de Uso – Tipo 1

Versão 1.0

Última atualização: 04/03/2012

Histórico de revisão do documento

Data	Versão	Responsável	Descrição da alteração
09/02/2012	**1.0**	Marcelo Rezende Módolo	Criação

1. Descrição breve

O caso de uso "Listar Redirecionamentos" é iniciado quando o editor deseja exibir os redirecionamentos para um sítio.

2. Atores

Editor.

3. Condições Prévias (ou pré-condições)

O Editor se autenticou no sistema.

4. Fluxo de Eventos

4.1 Fluxo Básico

1. O sistema exibe os sítios disponíveis para o editor;

2. O Editor seleciona o sítio para o qual deseja exibir os redirecionamentos;

2. O sistema obtêm a lista de redirecionamentos para o sítio selecionado;

3. O sistema exibe todos os redirecionamentos para o sítio selecionado.

4.2 Fluxos Alternativos

3a. Não existem redirecionamentos para o sítio selecionado;

3b. O sistema informa que não existem redirecionamentos a exibir.

5. Pós-Condições

Nenhuma

6. Requisitos Especiais

Nenhum

7. Esboço de Protótipo e Regras de Interface

ILUSTRAÇÃO 8: ESBOÇO DE TELA: LISTAR REDIRECIONAMENTOS

7.2 Regras de Interface

Ordenação:

- Permitir ordenar a listagem de redirecionamentos pelo alias (“De”);
- Permitir ordenar a listagem de redirecionamentos pelo destino (“Para”);

Ações:

- Permitir excluir um redirecionamento;
- Permitir criar um novo redirecionamento (**Opção disponível apenas quando o sítio estiver selecionado**);
- Permitir gerar os arquivos de redirecionamento (**Opção disponível apenas quando o sítio estiver selecionado**);

- Permitir sair do sistema.

CDU007 – EXCLUIR UM REDIRECIONAMENTO

Especificação de Caso de Uso – Tipo 1

Versão 1.0

Última atualização: 04/03/2012

Histórico de revisão do documento

Data	Versão	Responsável	Descrição da alteração
20/02/2012	**1.0**	Marcelo Rezende Módolo	Criação

1. Descrição breve

O caso de uso "Excluir um Redirecionamento" é iniciado quando o editor deseja excluir um redirecionamento para um sítio.

2. Atores

Editor.

3. Condições Prévias (ou pré-condições)

O Editor selecionou um dos sítios disponíveis (caso de uso CDU006 Listar Redirecionamentos).

4. Fluxo de Eventos

4.1 Fluxo Básico

1. O Editor seleciona a opção excluir para o redirecionamento;

2. O sistema altera o status do redirecionamento para excluído;

3. O sistema registra as informações de auditoria (nome do editor e hora da exclusão)

4.2 Fluxos Alternativos

Nenhum

5. Pós-Condições

Nenhuma

6. Requisitos Especiais

Nenhum

7. Esboço de Protótipo e Regras de Interface

Ver esboço de tela Listar Redirecionamentos.

7.2 Regras de Interface

Nenhuma

CDU008 – CADASTRAR UM NOVO REDIRECIONAMENTO

Especificação de Caso de Uso – Tipo 1

Versão 1.0

Última atualização: 04/03/2012

Histórico de revisão do documento

Data	Versão	Responsável	Descrição da alteração
21/02/2012	**1.0**	Marcelo Rezende Módolo	Criação
22/02/2012	**1.1**	Marcelo Rezende Módolo	Atualização

1. Descrição breve

O caso de uso "Cadastrar um Novo Redirecionamento" é iniciado quando o editor deseja criar um novo redirecionamento para um sítio.

2. Atores

Editor.

3. Condições Prévias (ou pré-condições)

O Editor selecionou um dos sítios disponíveis (caso de uso CDU006 Listar Redirecionamentos).

4. Fluxo de Eventos

4.1 Fluxo Básico

1. O Editor informa origem, destino e confirma;

2. O sistema valida as entradas;

3. O sistema verifica que o redirecionamento não existe;

4. O sistema grava o novo redirecionamento;

5. O sistema registra as informações de auditória (nome do editor e hora da criação)

4.2 Fluxos Alternativos

2a. O sistema verifica que os dados informados não são válidos;

2b. O sistema informa o erro.

3a. O sistema verifica que já existe um redirecionamento com a mesma origem;

3b. O sistema informa o erro.

5. Pós-Condições

Nenhuma

6. Requisitos Especiais

Nenhum

7. Esboço de Protótipo e Regras de Interface

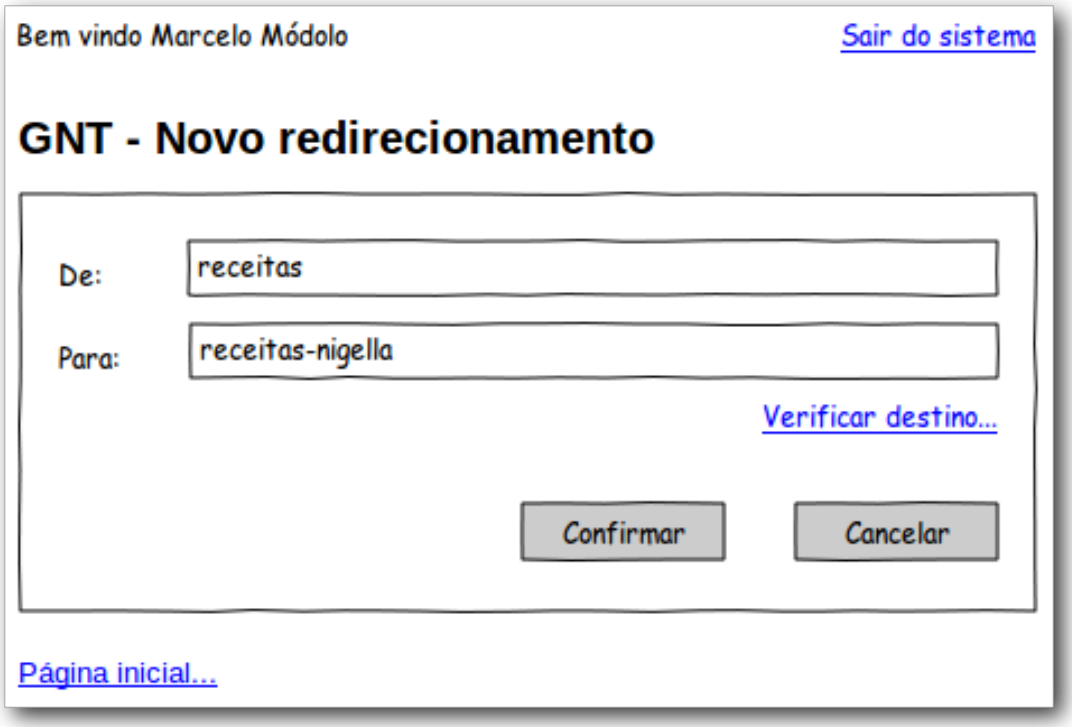

ILUSTRAÇÃO 9: ESBOÇO DE TELA: CADASTRAR UM NOVO REDIRECIONAMENTO

7.2 Regras de Interface

- Origem (“De”) não pode conter caracteres especiais;
- Destino (“Para”) não pode conter caracteres especiais;

Ações:

- Permitir sair do sistema;
- Permitir ir para a página inicial;
- Permitir verificar se o destino existe (a URL de destino formada existe no servidor);

CDU008 – GERAR ARQUIVO DE REDIRECIONAMENTO

Especificação de Caso de Uso – Tipo 1

Versão 1.0

Última atualização: 04/03/2012

Histórico de revisão do documento

Data	Versão	Responsável	Descrição da alteração
01/03/2012	**1.0**	Marcelo Rezende Módolo	Criação
05/03/2012	**1.1**	Marcelo Rezende Módolo	Modificação

1. Descrição breve

O caso de uso "Gerar Arquivo de Redirecionamento" é iniciado quando o editor deseja gerar o arquivo de redirecionamentos para um sítio.

2. Atores

Editor.

3. Condições Prévias (ou pré-condições)

O Editor selecionou um dos sítios disponíveis (caso de uso CDU006 Listar Redirecionamentos).

4. Fluxo de Eventos

4.1 Fluxo Básico

1. O Editor seleciona a opção Gerar Arquivo de Redirecionamento;

2. O sistema gera o arquivo com todos os redirecionamentos para o sítio selecionado;

3. O sistema renomeia o arquivo de redirecionamentos anterior;

4. O sistema grava o novo arquivo de redirecionamentos.

4.2 Fluxos Alternativos

3a. O sistema não consegue renomear o arquivo anterior;

3b. O sistema informa o erro.

4a. O sistema não consegue gravar o novo arquivo de redirecionamento;

4b. O sistema informa o erro.

5. Pós-Condições

Nenhuma

6. Requisitos Especiais

Nenhum

7. Esboço de Protótipo e Regras de Interface

Ver esboço de tela Listar Redirecionamentos.

7.2 Regras de Interface

Nenhuma

DIAGRAMA DE ENTIDADE RELACIONAMENTO

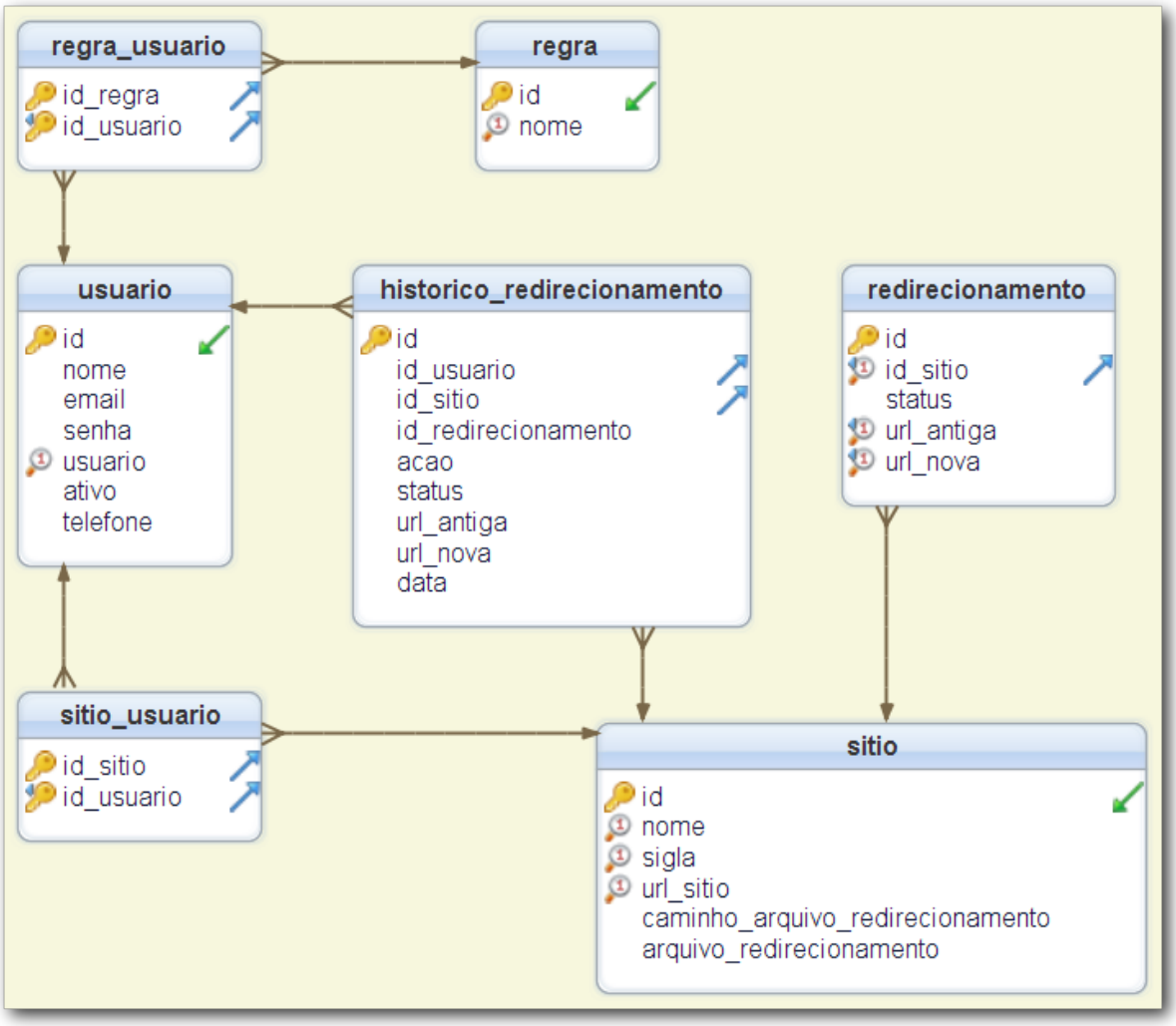

ILUSTRAÇÃO 10: DIAGRAMA DE ENTIDADE RELACIONAMENTO

Todas as ações efetuadas em um redirecionamento serão auditadas na tabela "histórico_redirecionamento", o sistema proposto apresenta dois tipos de ações:

- CRIADO – Quando um redirecionamento é criado;
- EXCLUÍDO – Quando um redirecionamento é excluído.

A tabela "histórico_redirecionamento" é propositalmente não normalizada pois seus valores são do tipo "*point-in-time*".

Tabela regra_usuario		
Um usuário pode ter mais de uma regra no sistema		
id_regra	int4 NOT NULL	Chave estrangeira para regra
id_usuario	int4 NOT NULL	Chave estrangeira para usuário
Índices		
regra_usuario_pkey primary key		(id_regra, id_usuario)
Chave estrangeira		
regra_usuario_id_regra_fkey		(id_regra) ref regra (id)
regra_usuario_id_usuario_fkey		(id_usuario) ref usuario (id)

Tabela regra Lista as regras ('roles')		
id	serial NOT NULL	Chave primária
nome	varchar(30) NOT NULL	Nome da regra
Índices		
regra_pkey primary key	(id)	

Tabela sitio_usuario		
Um usuário deve estar associado a um sítio para obter acesso ao sistema		
id_sitio	int4 NOT NULL	Chave estrangeira para o sítio
id_usuario	int4 NOT NULL	Chave estrangeira para o usuário
Índices		
canal_usuario_pkey	primary key	(id_sitio, id_usuario)
Chave estrangeira		
canal_usuario_id_canal_fkey		(id_sitio) ref sitio (id)
canal_usuario_id_usuario_fkey		(id_usuario) ref usuario (id)

Tabela sitio		
Informações para um sítio web		
id	serial NOT NULL	Chave primária
nome	varchar(150) NOT NULL	Nome do sítio
sigla	varchar(150) NOT NULL	Sigla do sítio
url_sitio	varchar(150) NOT NULL	URL
caminho_arquivo_redirecionamento	varchar(250) NOT NULL	
		Caminho para o arquivo de redirecionamentos
arquivo_redirecionamento	varchar(150) NOT NULL	
		Nome do arquivo de redirecionamentos
Chave estrangeira		
canal_pkey primary key	(id)	
canal_nome_key unique	(nome)	
canal_sigla_key unique	(sigla)	
canal_url_sitio_key unique	(url_sitio)	

Tabela redirecionamento		
Redirecionamentos para um sítio web		
id	serial NOT NULL	Chave primária
id_sitio	int4 NOT NULL	Chave estrangeira
status	int2 NOT NULL DEFO 301	Permanente
url_antiga	varchar(150) NOT NULL	URL antiga (origem)
url_nova	varchar(150) NOT NULL	URL nova (destino)
Índices		
pk_redirecionamento primary key	(id)	
idx_redirecionamento	(id_sitio)	
idx_redirecionamento_0 unique	(id_sitio, url_antiga, url_nova)	
Chave estrangeira		
fk_redirecionamento_canal	(id_sitio) ref sitio (id)	

Tabela historico_redirecionamento Todas as operações relacionadas a um redirecionamento são arquivadas		
id	serial NOT NULL	Chave primária
id_usuario	int4 NOT NULL	Chave estrangeira para usuário
id_sitio	int4 NOT NULL	Chave estrangeira para o sítio
id_redirecionamento	int4 NOT NULL	Identificado único do redirecionamento
acao	varchar(150) NOT NULL	Tipo de ação executada (CRIADO, EXCLUIDO)
status	int2 NOT NULL	Permanente
url_antiga	varchar(150) NOT NULL	URL antiga (origem)
url_nova	varchar(150) NOT NULL	URL nova (destino)
data	timestamp NOT NULL DEFO current_timestamp	Data e hora da operação realizada
Índices		
pk_historico_redirecionamento primary key	(id)	
idx_historico_redirecionamento	(id_usuario)	
idx_historico_redirecionamento_0	(id_sitio)	
Chave estrangeira		
fk_historico_redirecionamento_canal	(id_sitio) ref sitio (id)	
fk_historico_redirecionamento	(id_usuario) ref usuario (id)	

Tabela usuario		
Usuários que podem obter acesso ao sistema. Para acessar o sistema o usuário deve estar ativo e associado a um ou mais sítios		
id	serial NOT NULL	Chave primária
nome	varchar(150) NOT NULL	Nome completo do usuário
email	varchar(150) NOT NULL	E-mail do usuário
senha	varchar(150) NOT NULL	Senha
usuario	varchar(150) NOT NULL	Nome usado como autenticação no sistema
ativo	bool NOT NULL DEFO true	Indica se o usuário está ativo ou não. Usuários não podem ser excluídos do sistema, apenas desabilitados.
telefone	varchar(150) NOT NULL	Telefone para contato
Índices		
usuario_pkey primary key	(id)	
usuario_usuario_key unique	(usuario)	

DIAGRAMA DE CLASSES

PACOTE MODELO

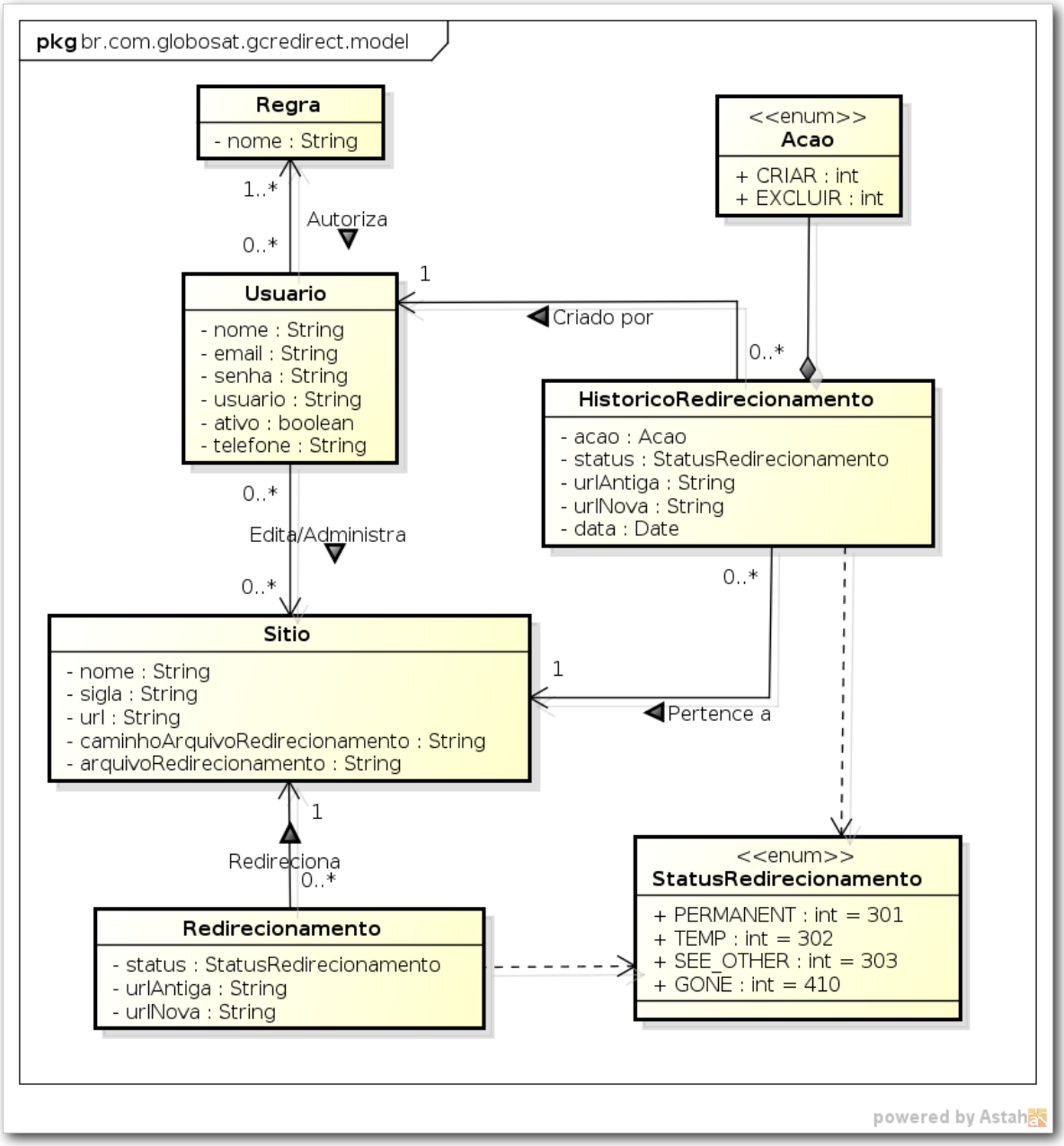

ILUSTRAÇÃO 11: PACOTE MODELO

O modelo de domínio abstrai as entidades do negócio e faz a ligação com com as tabelas no diagrama de entidade relacionamento.

DICIONÁRIO DE OBJETOS – MODELO DO DOMÍNIO

Regra		
Regras para autorização no sistema		
nome	String	O nome da regra (ROLE_ADMIN, ROLE_EDITOR)

Usuario		
Usuário do sistema		
nome	String	O nome completo do usuário
email	String	E-mail usado para contato
senha	String	A senha não pode ser salva em texto claro, usado HASH MD5
usuario	String	Nome usado para autenticação
ativo	boolean	Indica se o usuário pode se autenticar no sistema
telefone	String	Telefone usado para contato

Sitio		
Um sítio Web		
nome	String	O nome do sítio (Multishow, GNT, Sportv)
sigla	String	Sigla (MSW, GNT, SPO)
url	String	A URL para acesso ao sítio (gnt.globo.com, sportv.globo.com)
caminhoArquivoRedirecionamento	String	Caminho para o local onde o arquivo de redirecionamentos é salvo
arquivoRedirecionamento	String	Nome do arquivo usado para salvar os redirecionamentos

Redirecionamento Um novo caminho para uma URL antiga		
status	enum	O status do redirecionamento: • 301 – permanent; • 302 – temp; • 303 – seeother; • 410 – gone.
urlAntiga	String	O caminho antigo
urlNova	String	O novo caminho

HistoricoRedirecionamento Ações auditadas		
acao	enum	O tipo de ação: • CRIAR – Quando um redirecionamento é criado; • EXCLUIR - Quando um redirecionamento é excluído.
status	enum	O status do redirecionamento: • 301 – permanent; • 302 – temp; • 303 – seeother; • 410 – gone.
urlAntiga	String	O caminho antigo
urlNova	String	O novo caminho
data	Date	A data em que a ação foi executada

PACOTE SERVIÇOS

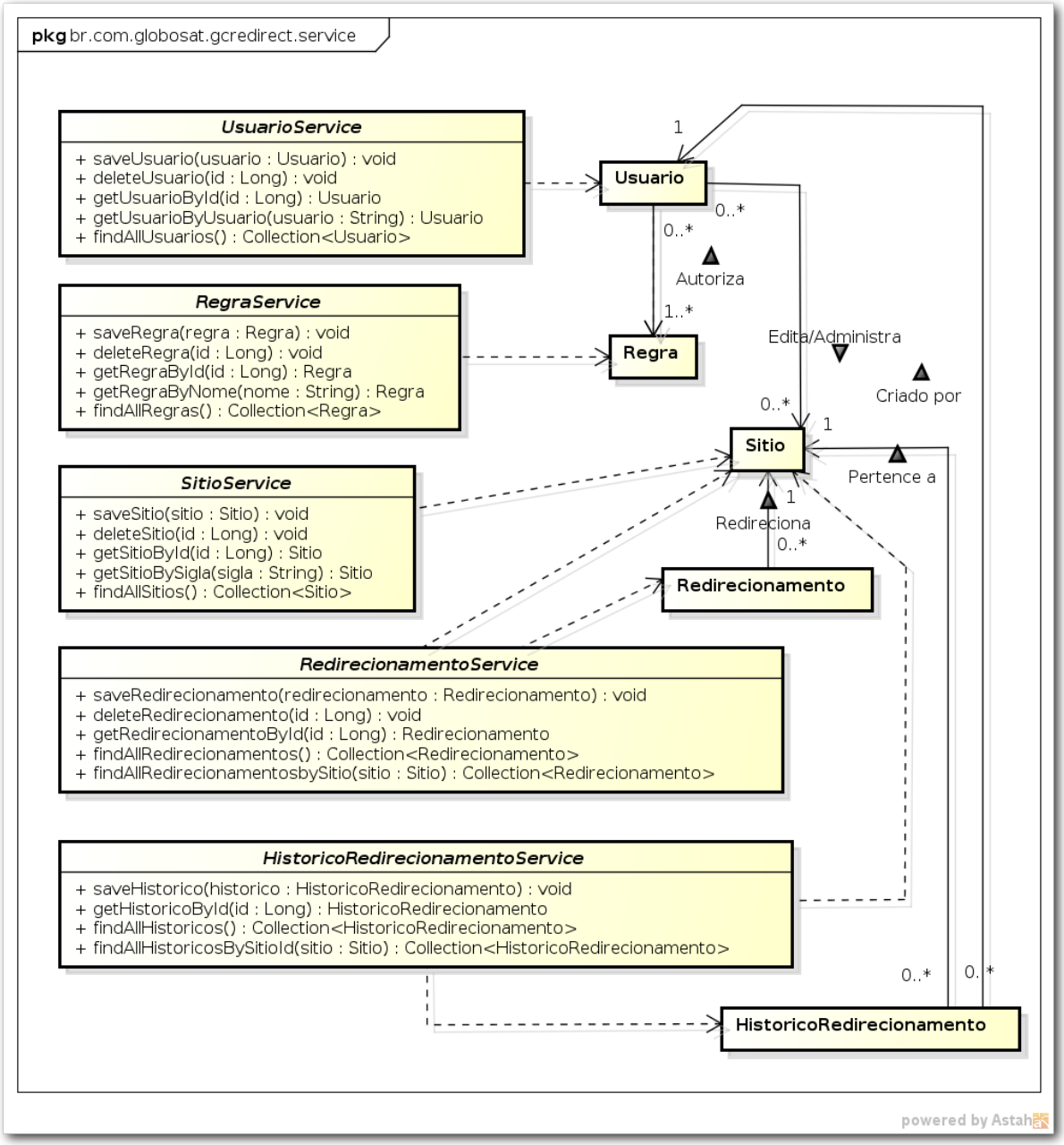

ILUSTRAÇÃO 12: PACOTE SERVIÇOS

O pacote serviços agrupa responsabilidades que não estão diretamente relacionas ao modelo mas que fazem uso desses objetos. O Serviço (também chamado por alguns de repositório) é responsável por recuperar, salvar, excluir e todas as operações relacionadas ao domínio do problema.

PACOTE SEGURANÇA

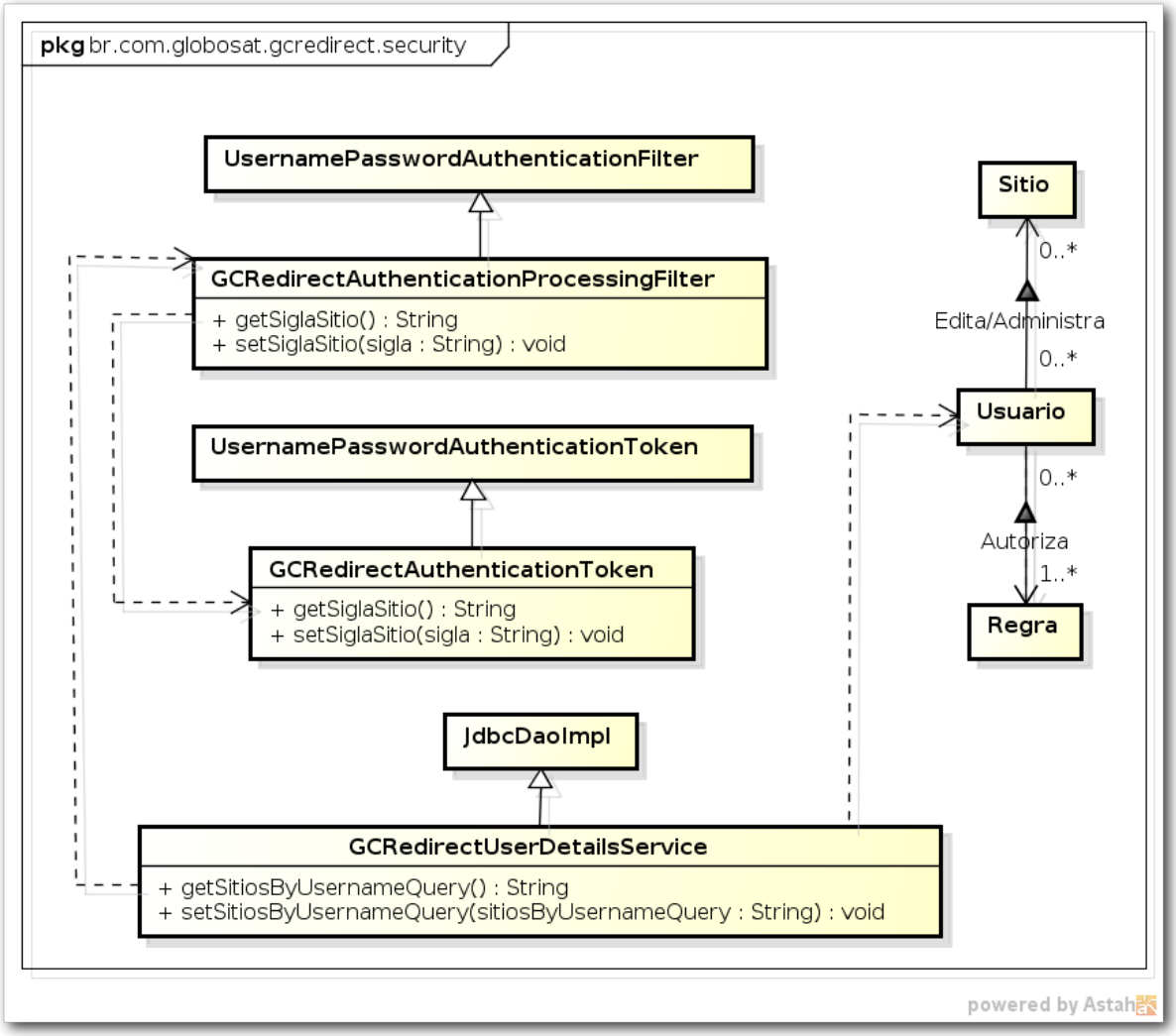

ILUSTRAÇÃO 13: PACOTE SEGURANÇA

O pacote Segurança agrupa as operações relacionas a autenticação e autorização dos usuários. As classes nesse pacote estendem o framework Spring adicionando as funcionalidades necessárias para garantir as regras do negócio.

DIAGRAMA DE MÁQUINA DE ESTADO

USUÁRIO

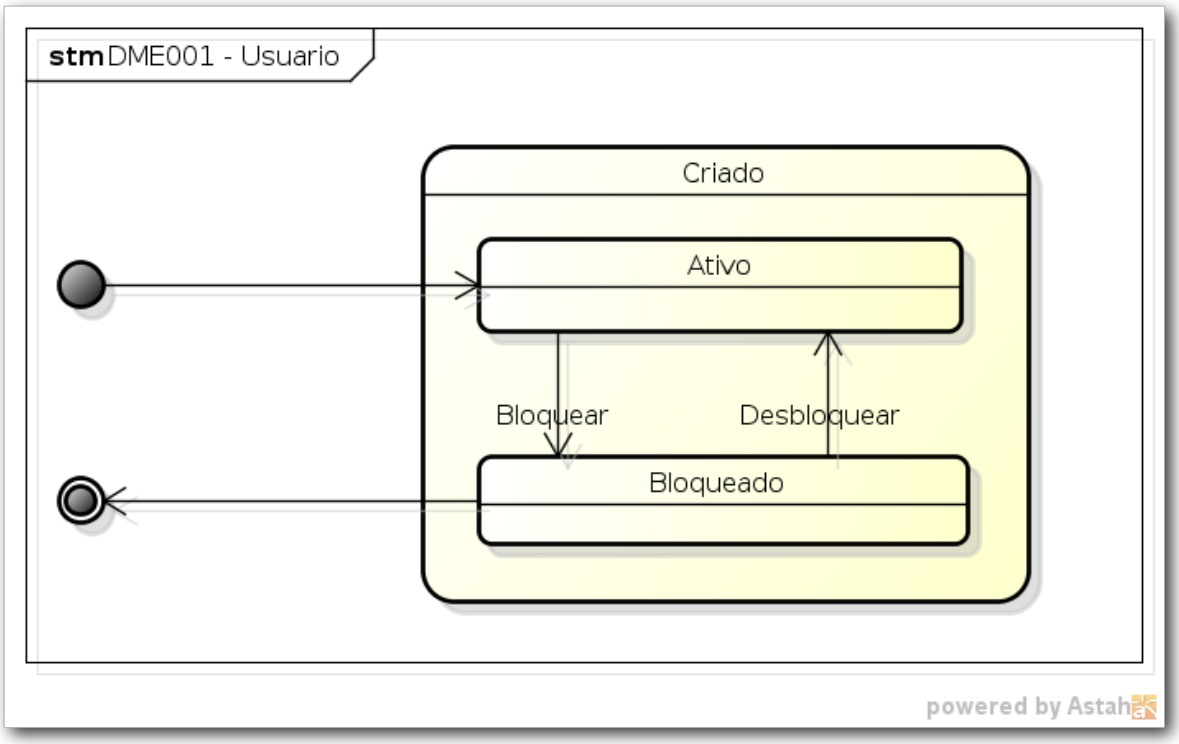

ILUSTRAÇÃO 14: DIAGRAMA DE MÁQUINA DE ESTADO - USUÁRIO

Os usuários no sistema (Administrados/Editores) após sua criação não podem ser excluídos, apenas mudam o status de Ativo para Bloqueado ou de Bloqueado para Ativo.

REDIRECIONAMENTO

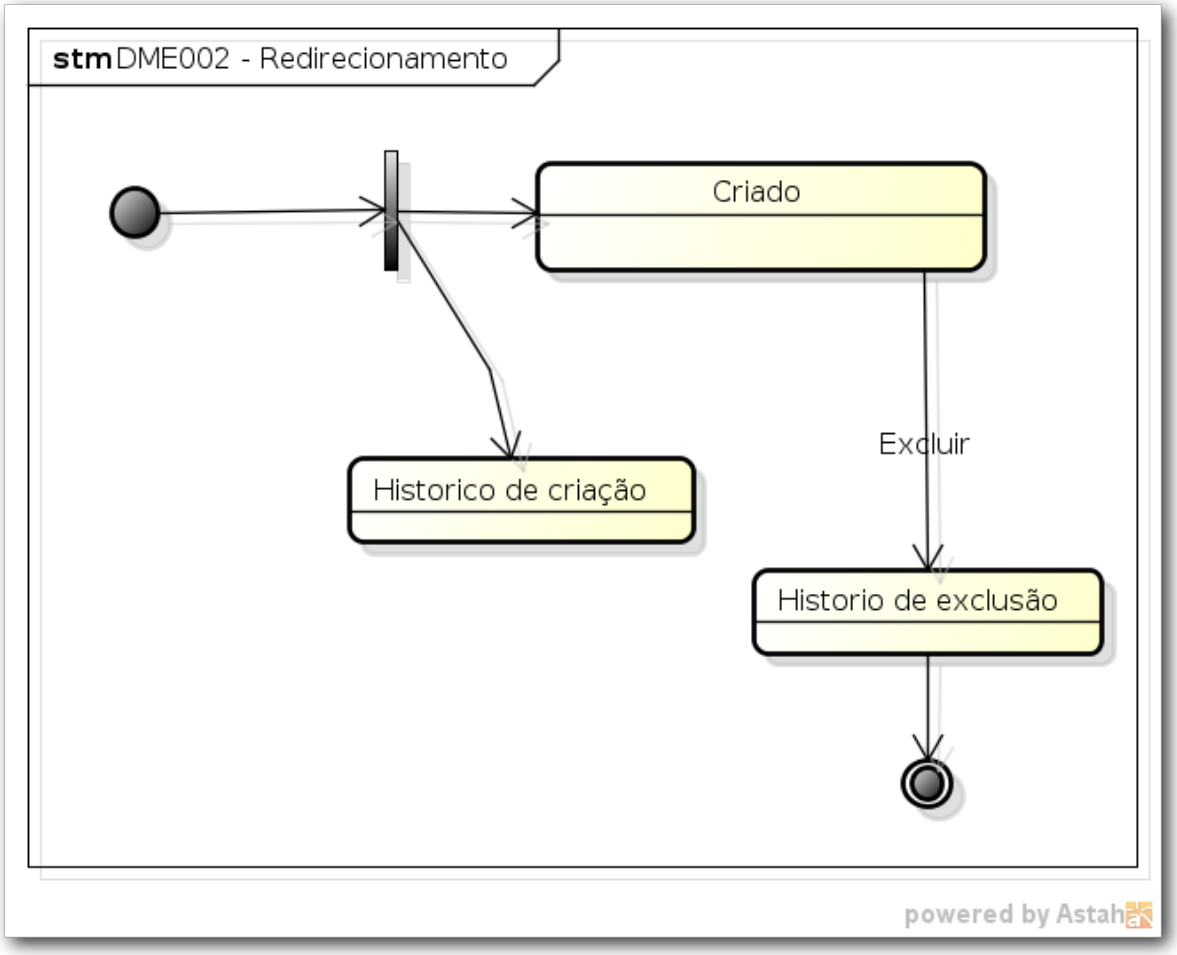

ILUSTRAÇÃO 15: DIAGRAMA DE MÁQUINA DE ESTADO - REDIRECIONAMENTO

Todas as ações (o sistema na versão 1.0 tem dois tipos de ações: CRIAR e EXCLUIR) sobre um redirecionamento serão auditadas. Ao ser criado um redirecionamento ao mesmo tempo o evento é auditado. Quando excluído, o evento é registrado no histórico.

DIAGRAMAS DE SEQUÊNCIA

Os diagramas de sequência criados são todos de alto-nível, documentando os contratos (interfaces) do sistema independente dos frameworks utilizados.

DS001 – LISTAR EDITORES

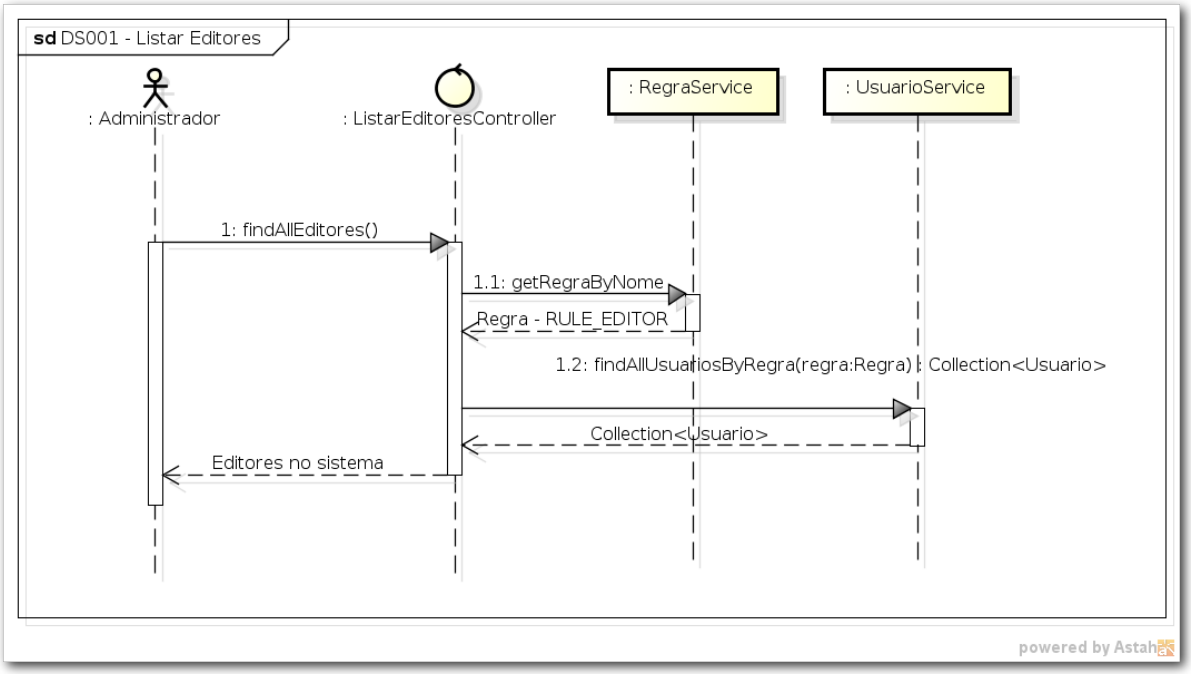

ILUSTRAÇÃO 16: DS001 – LISTAR EDITORES

DS002 – BLOQUEAR EDITOR

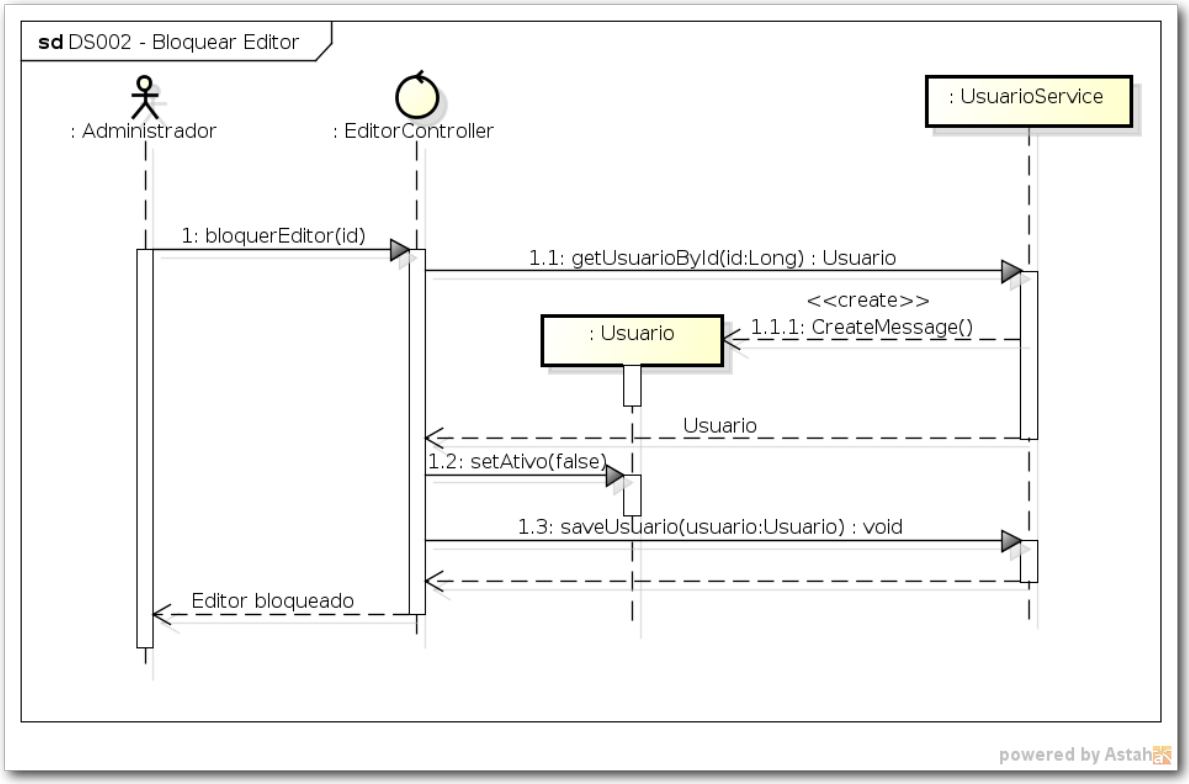

ILUSTRAÇÃO 17: DS002 – BLOQUEAR EDITOR

DS003 – DESBLOQUEAR EDITOR

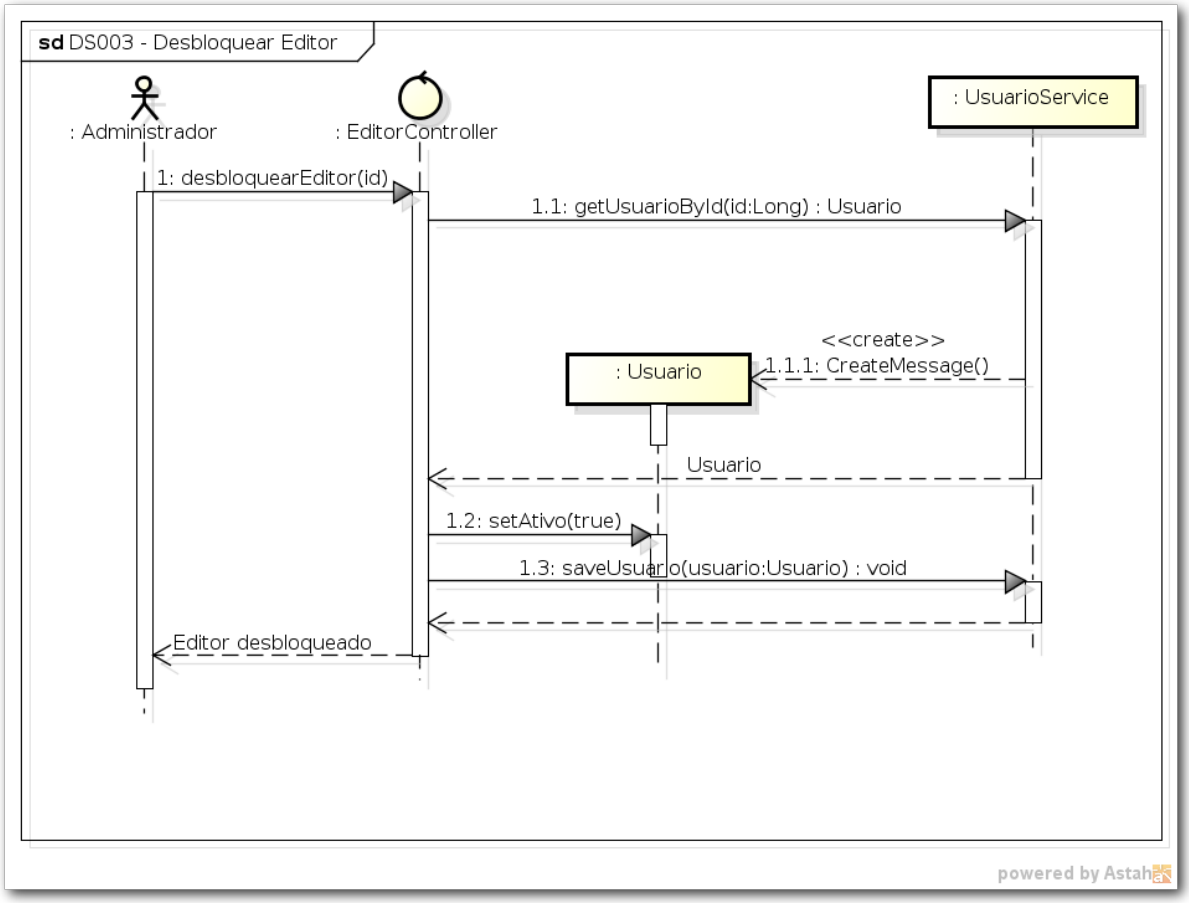

ILUSTRAÇÃO 18: DS003 – DESBLOQUEAR EDITOR

DS004 – MANTER EDITOR – CRIANDO UM NOVO

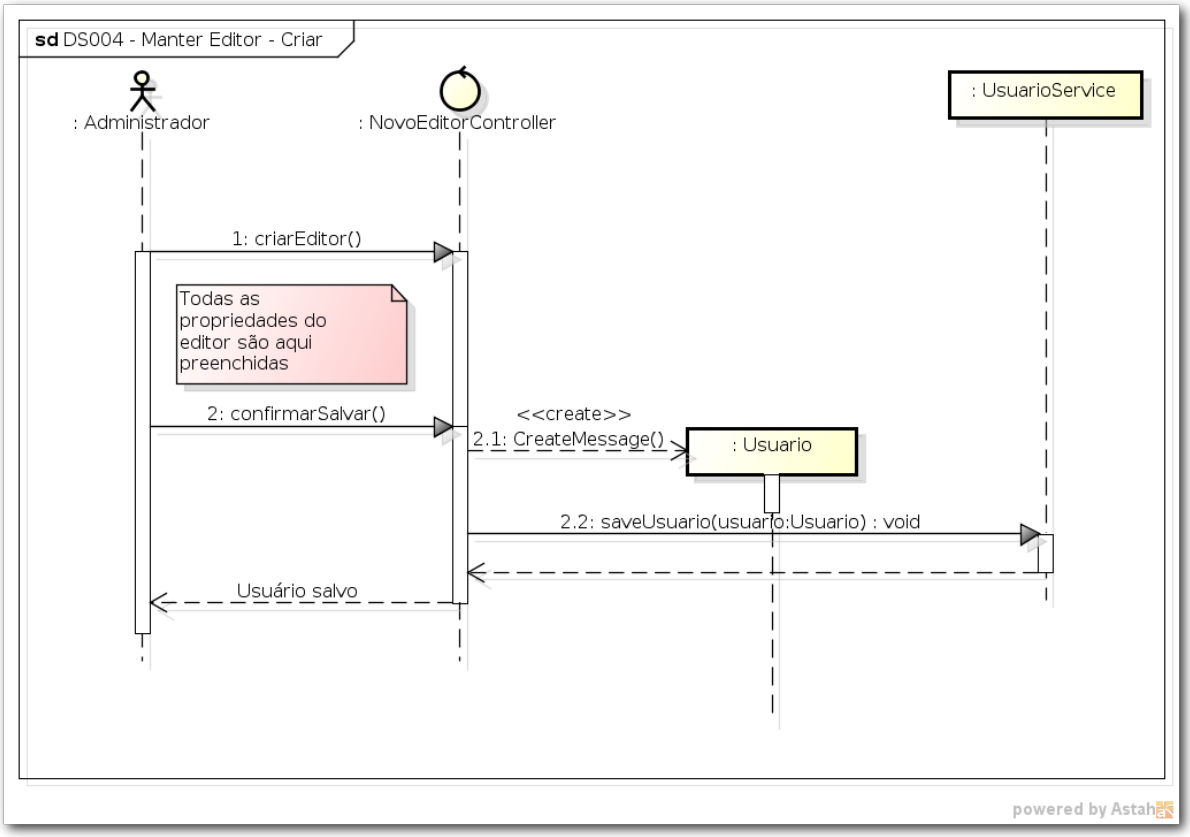

ILUSTRAÇÃO 19: DS004 – MANTER EDITOR – CRIANDO UM NOVO

DS005 – MANTER EDITOR – ATUALIZANDO

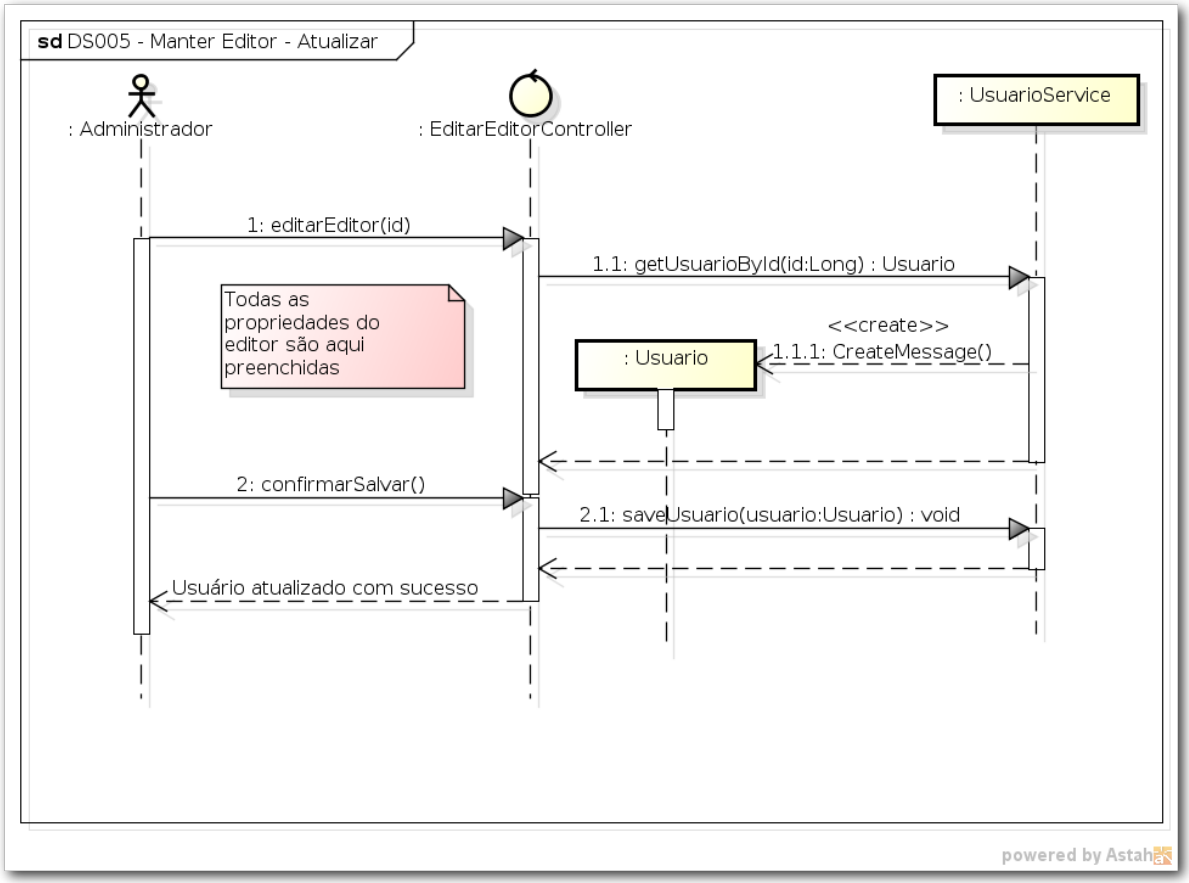

ILUSTRAÇÃO 20: DS005 – MANTER EDITOR – ATUALIZANDO

DS006 – LISTAR REDIRECIONAMENTOS

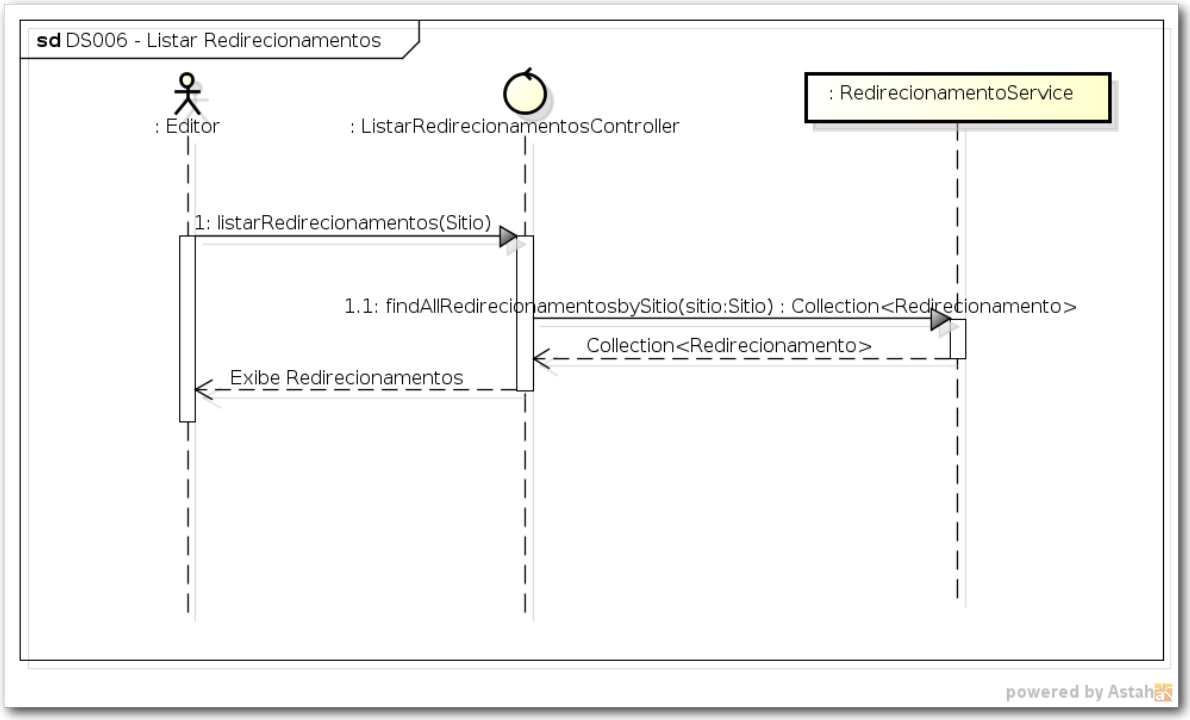

ILUSTRAÇÃO 21: DS006 – LISTAR REDIRECIONAMENTOS

DS007 – EXCLUIR REDIRECIONAMENTOS

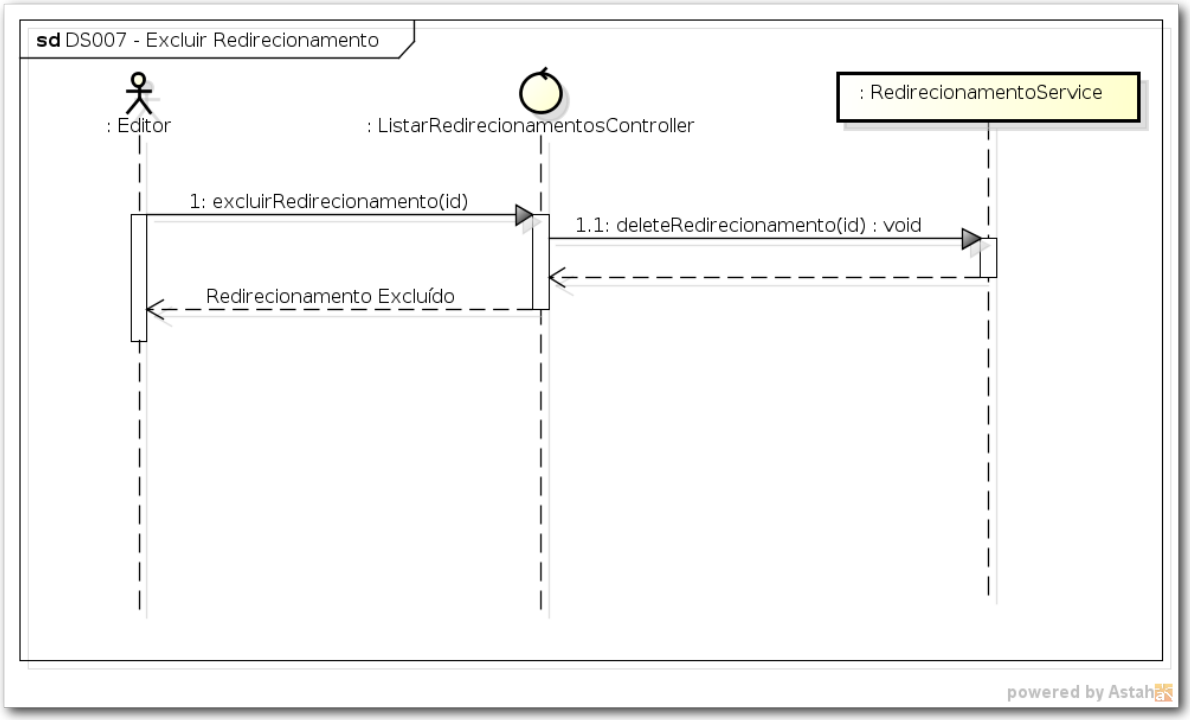

ILUSTRAÇÃO 22: DS007 – EXCLUIR REDIRECIONAMENTOS

DS008 – CADASTRAR NOVO REDIRECIONAMENTO

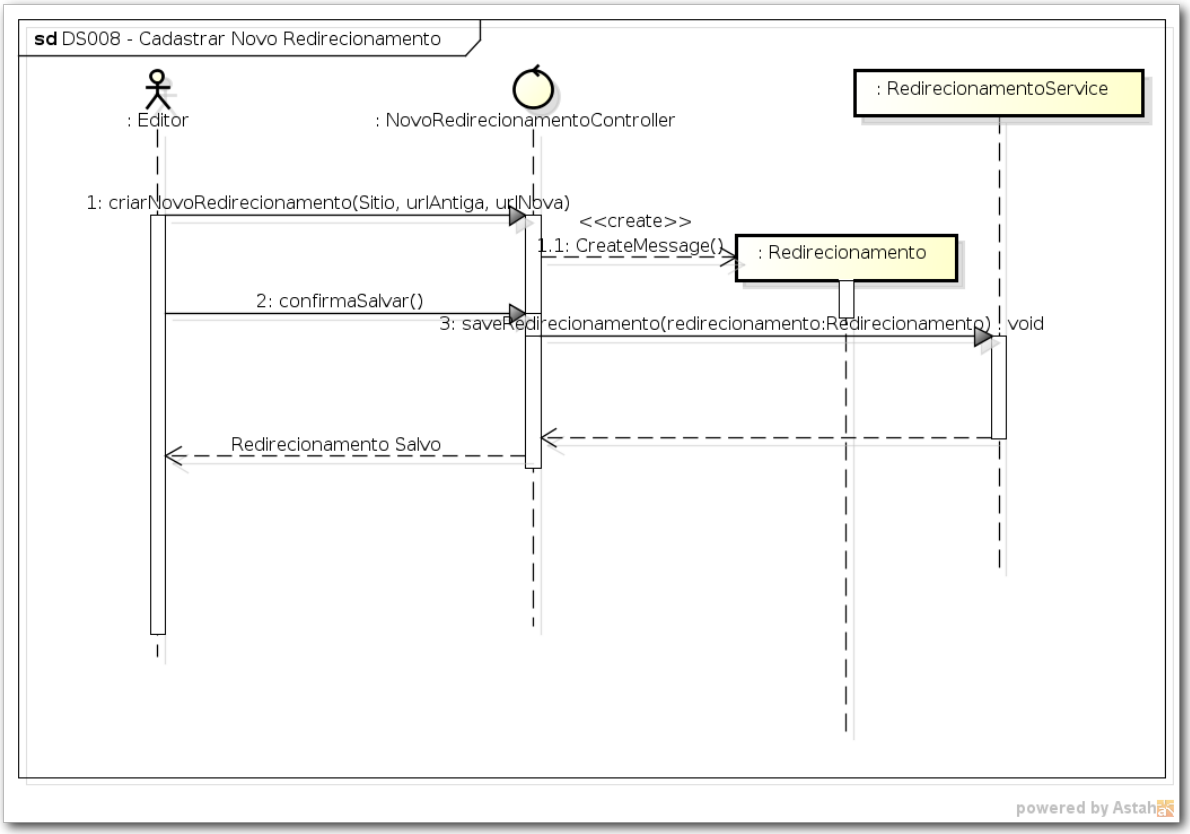

ILUSTRAÇÃO 23: DS008 – CADASTRAR NOVO REDIRECIONAMENTO

DS009 – CADASTRAR NOVO REDIRECIONAMENTO

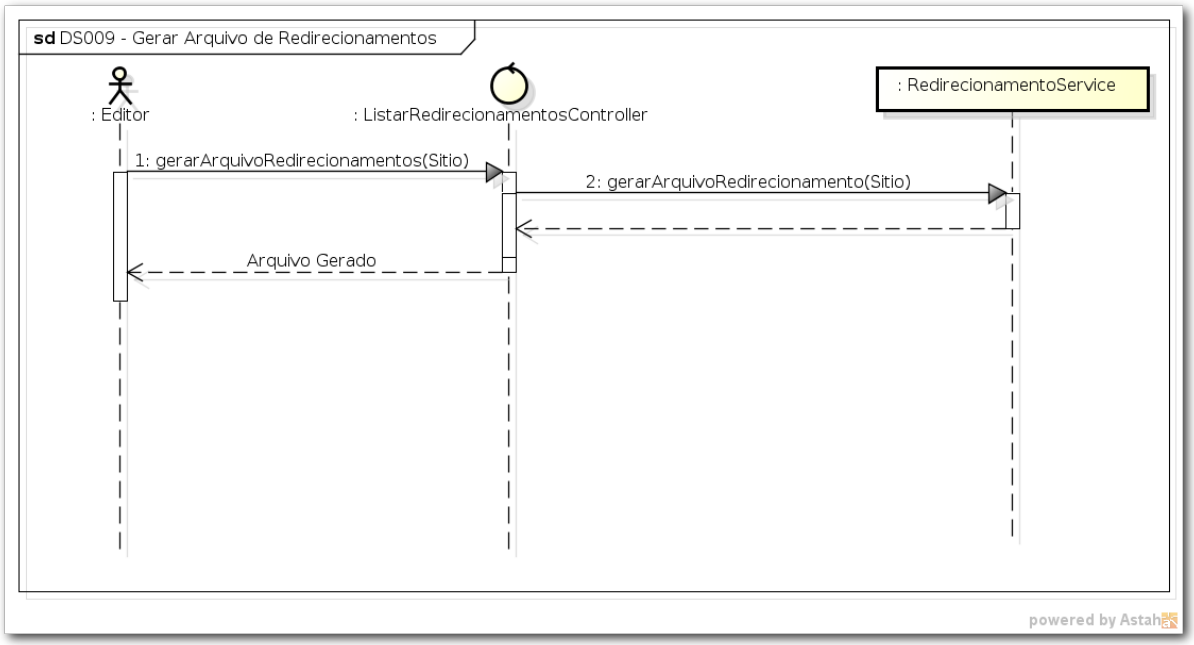

ILUSTRAÇÃO 24: DS009 – CADASTRAR NOVO REDIRECIONAMENTO

DIAGRAMA DE INSTALAÇÃO

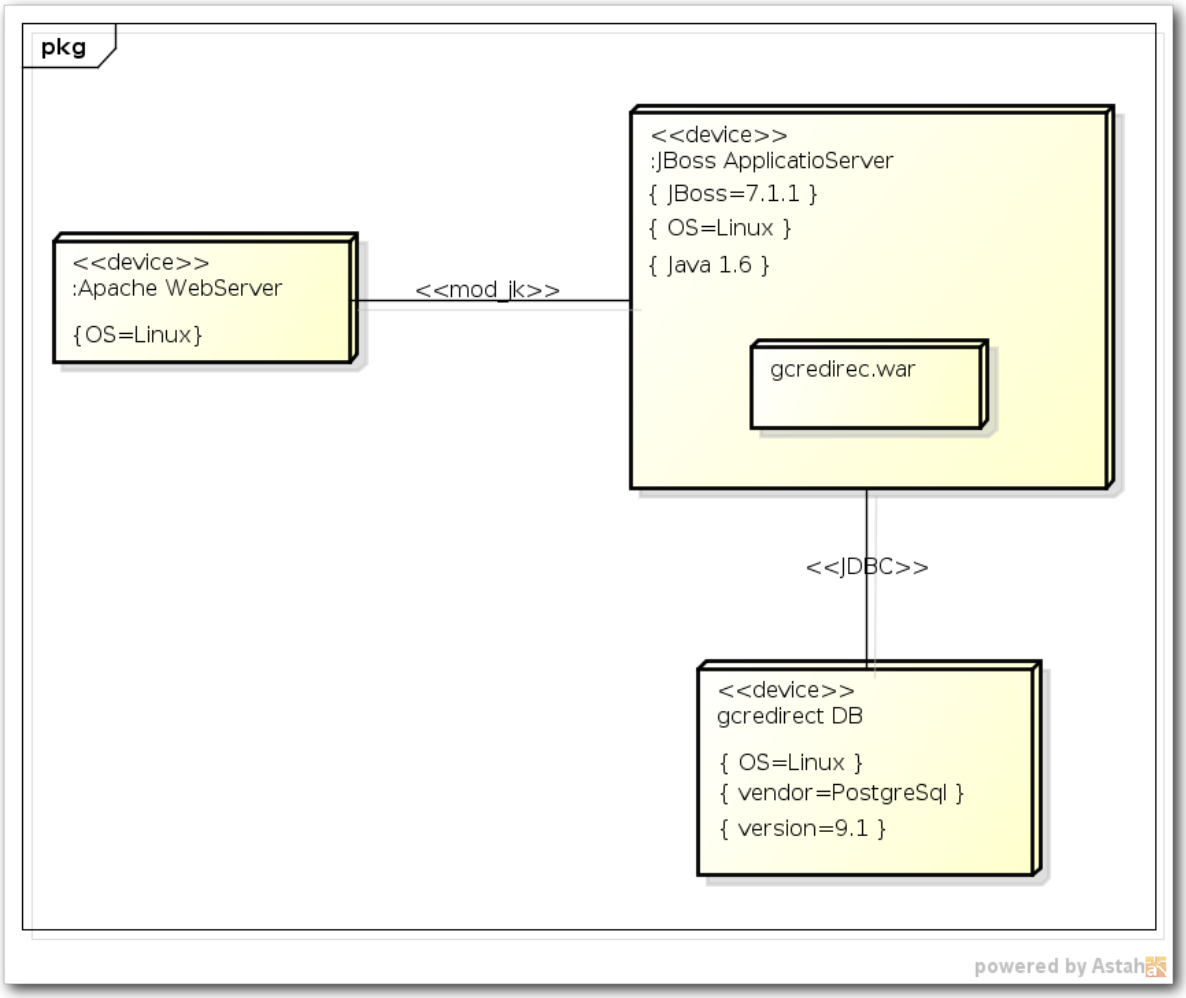

ILUSTRAÇÃO 25: DD001 - DIAGRAMA DE INSTALAÇÃO

MANUAL DE INSTALAÇÃO

Todos os softwares necessários estão contidos no DVD que acompanha o sistema nas versões usadas para o desenvolvimento. O sistema operacional escolhido e recomendado é o Linux sendo possível usar qualquer distribuição moderna (CentOS, Ubunto, RedHat).

SERVIDOR WEB

A comunicação entre o servidor de aplicações Java e o servidor web Apache será feita através do protocolo AJP usando o módulo Apache JK (MOD_JK). Essa configuração é feita pelo departamento de infra-estrutura de TI e transparente para a aplicação.

Uma vez configurado, será possível acessar a URL da aplicação pela porta 80. Para fins de teste e homologação é possível acessar a aplicação diretamente pelo servidor TOMCAT interno ao JBoss.

SDK JAVA

É necessário a instalação do SDK Java 1.6

SERVIDOR DE APLICAÇÃO JBOSS

O sistema foi desenvolvido e testado para o servidor JBoss 7.1.1.

SERVIDOR DE BANCO DE DADOS

O SGBD usado foi o PostgreSQL versão 9.1, no DVD estão os SCRIPTS DDL para criação dos objetos. Antes aplicar os arquivos de DDL é necessário criar um SCHEMA com o nome GCREDIRECT e um usuário para posterior configuração do DATASOURCE.

FONTE DE DADOS

A configuração de uma nova fonte de dados no JBoss 7.1.1 mudou bastante, no DVD temos um exemplo de configuração para o DATASOURCE e o DRIVER JDBC.

Software de Apoio

Durante o desenvolvimento e documentação de um sistema é muito importante escolher as ferramentas corretas e que facilitem o trabalho. Escolhemos usar apenas ferramentas gratuitas e de preferência de código aberto.

LibreOffice

O LibreOffice é uma suíte de edição muito completa e que vem evoluindo e amadurecendo com o passar dos anos. Toda a documentação foi escrita usando a versão 3.5.1 que pode ser obtida em <http://www.libreoffice.org/>.

IDE Eclipse

Para facilitar o desenvolvimento foi usada a IDE Eclipse <http://eclipse.org/>.

Diagramas UML

Os diagramas UML foram criados com a ferramenta ASTAH COMMUNITY que apesar de não ser de código aberto, é multiplataforma e disponibiliza uma versão sem custos <http://astah.net/editions/community>.

Diagrama de Entidade Relacionamento

Existem poucas ferramentas gratuitas para o desenho do DER (pelo menos compatíveis com o PostgreSQL) usamos o DbSchema que permite a engenharia reversa do banco para os diagramas. A versão usada foi a pessoal com algumas funcionalidades desabilitadas <http://www.dbschema.com/>.

Marcelo Rezende Módolo, 13 de abril de 2012