UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

GÍLSON MAEKAWA KANASHIRO

SISTEMA DE TI PARA GESTÃO CONDOMINIAL

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA - PR
2011

GÍLSON MAEKAWA KANASHIRO

SISTEMA DE TI PARA GESTÃO CONDOMINIAL

Monografia de Especialização apresentada ao Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de “Especialista em Tecnologia Java”

Orientador: Prof. Paulo Roberto Bueno.

CURITIBA - PR

2011

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria do Campus Curitiba
Gerência de Pesquisa e Pós-graduação
Departamento Acadêmico de Informática
Curso de Especialização em Tecnologia Java

TERMO DE APROVAÇÃO

Sistema de TI para Gestão Condominial

Esta monografia foi apresentada às 16 h 20 min, do dia 21 de setembro de 2011, como requisito parcial para a obtenção do título de Especialista em Tecnologia Java – Departamento Acadêmico de Informática – Universidade Tecnológica Federal do Paraná. O candidato apresentou o trabalho para a Banca Examinadora composta pelos professores abaixo assinados. Após a deliberação, a Banca Examinadora considerou o trabalho APROVADO.

Prof. MSc. Marcelo Mikosz Gonçalves
Membro
(UTFPR)

Prof. Dr. Gustavo Alberto Giménez Lugo
Membro
(UTFPR)

Prof. MSc. Paulo Roberto Bueno
Orientador
(UTFPR)

Visto da Coordenação:

Prof. Dr. João Alberto Fabro
Coordenador
Curso de Especialização em Tecnologia Java

DEDICATÓRIA

À minha esposa, Graciele.

À nossa menininha recém-chegada, bênção de felicidade e de ternura em nossas vidas: Sayuri, filha amada.

À memória de Maíza, irmã dedicada e exemplo de compreensão e carinho.

AGRADECIMENTOS

É pouco o tempo em que pudemos compartilhar o convívio entre novos colegas, novos professores.

Pouco não é o termo adequado. É daqueles raros tempos vividos intensamente, imersos em afazeres, muitos afazeres; são tempos aventureiros, em que se vive o desafio de fazer, aprender e conhecer. São momentos inefáveis como aqueles em que um mero velejador começa a aprender com grandes navegadores.

Meus sinceros votos de apreço e consideração a todos os colegas, em especial ao Leonardo Ravazzi, parceiro de muitos trabalhos de fim de semana; Fernando, que nos passou inúmeras dicas para domar o Eclipse, Gláucia e Rebeca, meninas bem humoradas com quem demos boas gargalhadas; Najib El Alam, com quem tentamos negociar algumas piadas, mas acabamos nos perdendo em boas risadas. Cintia, obrigado! Graças a você muita coisa funcionou na hora certa.

Aos professores, agradeço o esforço didático para ensinar em noites de quem já teve um dia cheio no trabalho; principalmente, por responder dúvidas e trocar ideias valiosas. Em especial, agradeço aos professores Alexandre Reis Graeml, Celso Antonio Alves Kaestner, Cesar Augusto Tacla, Leandro Batista de Almeida e Robson Ribeiro Linhares. Com esses mestres e doutores pude assimilar conceitos bastante sutis em suas áreas do conhecimento.

Ao meu orientador, professor Paulo Roberto Bueno, o meu grande agradecimento por contribuir, reforçando ideias e conceitos.

As palavras não são suficientes para expressar o meu profundo agradecimento à minha esposa, cujo apoio ajudou a me manter firme em meus propósitos, apesar das intensas mudanças profissionais por que passamos. Comecei a monografia em Curitiba e a termino em Londrina. Graciele, esposa amada, com certeza teremos histórias para contar à nossa pequena Sayuri.

RESUMO

KANASHIRO, Gílson M. Sistema de TI para gestão condominial. 2011. Monografia (Especialização em Tecnologia Java) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2011.

Esse trabalho apresenta uma abordagem do problema de coordenação das atividades sob o ponto de vista de um programa de gerenciamento de condomínios. Da experiência do autor, ex-síndico de condomínio, focou-se na visão daquele que gerencia, buscando a praticidade e a flexibilidade. Aborda aspectos da gestão condominial e de seus problemas. Traz como resultado uma ferramenta gerencial simplificada, desenvolvida para auxiliar o acompanhamento de atividades.

Palavras-chave:
Gestão condominial, coordenação de atividades, acompanhamento de atividades.

ABSTRACT

KANASHIRO, Gílson M. IT System to condominium management. 2011. Monografia (Especialização em Tecnologia Java) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2011.

This work presents a boarding of the question of coordination of activities under the point of view of a condominium management software. Of the author's experience, ex-condominium syndic, it focus on who manages, aiming flexibility and practice. It presents aspects of the condominium management. As results, research brings a simplified management tool that was developed to help on activities tracking.

Palavras-chave:
Condominium management, activities coordination, activities tracking.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A gestão de empresas é uma área da Administração que possui conhecimentos e técnicas consolidadas.

Particularmente, a gestão de condomínios habitacionais possui alguns diferenciais que a torna um caso *sui generis*. Pode-se observar que, em geral, um condomínio não é uma organização destinada ao lucro; mas, ao mesmo tempo, é um sistema gerencial, cujos processos financeiros, contábeis e fiscais devem estar sob rigoroso controle.

No sistema gerencial de um condomínio, em termos executivos ou operacionais, a figura central é o Síndico, cuja função é administrar o patrimônio comum e representar os interesses coletivos dispostos em Assembleia Geral, conforme estabelecem o artigo 1347 da Lei Federal nº 10.406/02 (Código Civil) e a Lei Federal nº 4591/64 (Convenção Condominial).

O assunto do presente trabalho é um sistema de tecnologia da informação que auxilie na coordenação de atividades e se desenvolva com foco nas necessidades do síndico do condomínio, de forma a facilitar e a agilizar a rotina de seu trabalho.

1.1. Problema

Os condomínios residenciais são sistemas com processos gerenciais dos mais simples até os que alcançam a complexidade de um grande sistema corporativo. Invariavelmente, utilizam-se *softwares* para auxiliar a gestão condominial.

Para serem competitivos, os *softwares* vem agregando o maior número de funcionalidades possível. É evidente que isso, além de tornar o produto final caro, descaracteriza-o para a utilização centrada em uma única pessoa: o síndico.

Dessa forma, em função da realidade atual, diversas funcionalidades não serão utilizadas (ou o seriam de forma diferente) pelo síndico. Um exemplo disso é a emissão de boletos. Para o síndico, essa funcionalidade é secundária, já que tal serviço pode ser terceirizado para uma administradora de condomínios. A atenção do síndico converge para o status da arrecadação e na situação de inadimplências. O síndico precisa dessas informações principalmente para o planejamento de longo prazo.

Não se deseja, com a argumentação acima, minimizar a importância e a adequação das diversas funcionalidades em programas já existentes no mercado. As observações acima são

evidências de que as atividades condominiais já estão veementemente distantes das atividades de uma administradora de condomínios.

Sob a expressão administração de condomínios ou gestão condominial, encontram-se principalmente os programas com foco nas pessoas jurídicas, administradoras de vários condomínios. Pouco se vê em termos de ferramentas de tecnologia da informação centradas nos síndicos.

Uma das atividades sujeitas a um maior desgaste é o gerenciamento de projetos, em particular, no acompanhamento desses projetos. Vários deles são administrados concomitantemente. O esquecimento de datas importantes ou a ausência da visão do todo tendem a turvar a percepção do síndico a tal ponto que ao final não é possível inferir se determinado projeto alcançou seus objetivos.

1.2. Justificativa

No que tange à administração condominial:

> "(…) identificam-se três modalidades de gestão: autogestão, co-gestão e terceirização. Na autogestão, o síndico, eleito na forma da Lei Federal nº 10.406/02, assume total responsabilidade e administra o condomínio. Na co-gestão, o síndico recorre a uma assessoria administrativa, ou seja, partilha suas tarefas com empresas especializadas na administração de condomínios, mas sem transferências das responsabilidades legais. Já a terceirização, uma das tendências atuais do mercado imobiliário, trata-se da eleição da própria administradora de condomínios como síndico, esta, assim, assumindo total responsabilidade, administrando e representando legalmente o condomínio." (Farber e Segreti, 2006, p. 3 e 4)

Farber e Segreti (2006, p. 12) afirmam ainda que 56% dos condomínios de São Paulo compartilham sua administração com empresas especializadas (co-gestão). Logo, os outros 44% representam a autogestão ou a terceirização.

Embora a pesquisa tenha-se realizado na cidade de São Paulo, a realidade urbana em muitas cidades, incluindo Curitiba, vem contribuindo para o surgimento de condomínios e de empresas administradoras de condomínio (Rede Globo – Jornalismo, PEGN, 2011).

Exceto pela terceirização, as outras duas modalidades exigem a decisão do síndico. Nesse caso, existe uma característica preponderante na maioria dos condomínios residenciais: os síndicos não detem conhecimento gerencial suficiente para o mister.

É natural, portanto, que haja uma carência em conceitos de gestão, tais como: identificar os problemas e as necessidades, mobilizar equipes de trabalho, desenvolver e controlar o andamento de soluções financeiramente viáveis.

Diante dessa dificuldade, o síndico se vê assoberbado pelas demandas, em um ciclo vicioso de urgências e soluções precipitadas. Assim, há poucos instrumentos para apoiar decisões estratégicas, como o orçamento anual.

É evidente que o conhecimento gerencial se faz imprescindível. No entanto, a falta de uma estrutura gerencial mínima tende a se agravar com a carência de ferramentas computacionais para facilitar a atividade do síndico.

A Tabela 1 abaixo apresenta programas tradicionais do mercado, focando suas características e os diferenciais de mercado. Neste último quesito, é possível identificar aqueles que se posicionam para empresas ou para administradores (síndicos).

Tabela 1: Programas de gestão condominial: características principais e diferenciais

	Programa	**Características principais e diferenciais**
1	Base Condomínio	Controle dos contratos de administração com os condomínios. Controle de contas a pagar e a receber. Integração com bancos. Cálculo de rateio e emissão de boleto. Relatórios diversos. Acesso via *web* opcional.
2	BRCondomínio	Controle de contas a pagar e a receber. Integração com todos os bancos. Emissão de boletos. Folha de pagamento. Personalização do sistema. Relatórios diversos. Repasse de pagamentos efetuados a menor ou a maior para a próxima mensalidade. Configuração de acesso multinível (administradora, contador, síndico, conselho fiscal, condôminos e funcionários).
3	Condomínio	Controle de contas a pagar e a receber. Cadastros de diversos condomínios, blocos e suas unidades. Previsões de receitas e despesas. Cálculo de rateio. Desenvolvido para administrar o próprio ou diversos condomínios. Atende às necessidades de controle das rotinas diárias. Curso multimídia para aprendizado do programa para administradores e funcionários.
4	Condomínio21	Há três versões para se adaptar ao tamanho do condomínio. Controle de contas a pagar e a receber. Integração com bancos. Emissão de boletos. Folha de pagamento. Personalização do sistema. Relatórios diversos. Consultas a informações via *web*. Adaptável para utilização de administradores de perfis variados, na gestão de condomínios de pequeno, médio ou grande porte.
5	Condor	Controle de contas a pagar e a receber. Cobranças de taxas. Integração com banco. Emissão de boletos. Personalização do sistema. Relatórios diversos. Controle de inadimplências. Acesso via *web*. Há uma versão gratuita.
6	Immobile	Controle de contas a pagar e a receber, de lançamentos fixos, indexados, cobranças de taxas. Integração com banco. Emissão de boletos. Personalização do sistema. Relatórios diversos. Interface para imobiliárias. Integração com outros sistemas da empresa relacionados a armazenamento seguro e serviços de contabilidade e recursos humanos.

Fonte: Arquivos de pesquisas pessoais do autor.

Com base nas características observadas e também nos diferenciais apresentados em material publicitário, pode-se claramente definir os programas focados nas atividades rotineiras do síndico.

Essa observação em relação ao diferencial se baseia em material propagandístico. No entanto, é preciso observar que também o síndico ou a respectiva Assembleia Geral farão a comparação dessas características através do material disponibilizado em alguma mídia publicitária.

Na Tabela 2, resumem-se os programas analisados, a empresa desenvolvedora e o seu público-alvo. De acordo com o comparativo, verifica-se uma forte tendência de varejo voltado ao mercado corporativo, em detrimento da pessoa física do síndico.

Tabela 2: Principais programas de gestão condominial e seu público-alvo

	Programa	**Empresa**	**Público-alvo**
1	Base Condomínio	Base Software	Empresa
2	BRCondomínio	BRCondomínio	Empresa
3	Condomínio	Best Software	Empresa/Síndico
4	Condomínio21	Group Software	Empresa/Síndico
5	Condor	Superlogica	Empresa
6	Immobile	Aterdata Software	Empresa

Fonte: Arquivos de pesquisas pessoais do autor.

Como se viu em Farber e Segreti (2006, p. 4), embora haja uma tendência à terceirização, a autogestão e a co-gestão são as duas modalidades de gestão condominial dominantes. Em ambas, verifica-se que o processo decisório passa pelas mãos de uma única pessoa: o síndico.

Dessa forma, esse trabalho se focaliza no estudo de uma abordagem do problema de coordenação das atividades sob o ponto de vista de ferramentas que possam ajudar o síndico

1.3. Objetivos

Os objetivos gerais desse trabalho se voltaram para o desenvolvimento de uma ferramenta computacional, através da qual apresenta-se uma abordagem do problema de coordenação de atividades.

Como objetivos específicos, os esforços se concentraram no desenvolvimento das seguintes funcionalidades:

- Cadastro de usuários: controle do acesso ao sistema.
- Cadastro de atividades: acompanhamento e coordenação das atividades.
- Agenda: acompanhamento e coordenação das atividades.

Algumas premissas fundamentaram os trabalhos. São elas:

- Fazer ferramentas computacionais mais amigáveis.
- Considerar que síndicos tem pouco tempo para decidir.
- Ter a possibilidade de colaboração, independente de horários e reuniões presenciais.

1.4. Metodologia

O presente trabalho se compõe basicamente de estudos, testes e aplicação de várias tecnologias, integrando-as em um produto, destinado ao gerenciamento de projetos. Conforme Silva e Menezes (2005), na essência, esse trabalho se caracteriza eminentemente como uma pesquisa aplicada.

A pesquisa bibliográfica foi o principal instrumento para identificar a situação mercadológica dos *softwares* para gestão condominial, de modo que se viu uma oportunidade de trabalho específico na área.

Testes foram realizados para aplicar as tecnologias e os resultados foram sendo refinados etapa por etapa à medida que se adquiria maior desenvoltura na codificação.

O capitulo 1 foi uma apresentação do trabalho, buscando esclarecer seus objetivos, os problemas envolvidos e seu conceito geral.

No capítulo 2, abordam-se os aspectos teóricos que envolvem as atividades sobre as quais um síndico se debruça no cotidiano. A seguir, passa-se a abordar os aspectos técnicos que permeiam o presente trabalho: *frameworks* Java, Hibernate, padrões de projeto DAO e MVC, a especificação JSF e MySQL.

No capítulo 3, demonstra-se o desenvolvimento da projeto à medida que a pesquisa evoluía. Busca-se apresentar os principais conceitos envolvidos na solução, a observação que deu origem ao trabalho, às restrições técnicas. O relatório foi elaborado sem divagar em torno dos códigos Java. Ênfase é dada na explicação da funcionalidade implementada.

Os capítulos 4 e 5 apresentam considerações finais e as conclusões propiciadas pelo trabalho.

2. COORDENAÇÃO DE ATIVIDADES CONDOMINIAIS: TEORIA E FUNDAMENTOS

Vários conceitos comuns à gestão de empresas podem ser aplicados na administração de um condomínio. Trata-se de serviço privado com características de serviço público, já que o síndico não age deliberadamente e deve prestar conta de suas ações aos condôminos.

No texto a seguir, a exposição teórica se divide em duas áreas do conhecimento muito diferentes: o Direito e a Informática.

Inicialmente, alguns conceitos do Direito Civil serão expostos para evidenciar como as atividades do síndico se revestem de especificidades muito sérias: os aspectos legais.

Na sequencia, serão abordados os conceitos teóricos de algumas tecnologias utilizadas. Brevemente, as principais tecnologias se referem a: *Java Server Faces* (JSF) e Hibernate (ambos, *frameworks* Java); MySQL, um banco de dados relacional e MVC, o *design pattern* (padrão de projeto) que mais se sobressai.

Obviamente, na prática, o projeto se utiliza de várias outras tecnologias, não menos importantes. Entretanto, a prolixidade levaria a abordar assuntos tais como XHTML, XML, *Webdesign*, os próprios conceitos de orientação a objetos, Engenharia de *Software* e Gestão de Projetos, dentre outros assuntos que fogem aos objetivos do trabalho.

No projeto implementado, entretanto, convem citar o emprego das seguintes tecnologias e programas: Facelets, Spring Security, JSF (PrimeFaces), Apache Commons, Apache Tomcat, Eclipse, MySQL Workbench e LibreOffice.

Além das tecnologias acima, convem citar também a experiência do autor como síndico do condomínio em que residia. Esse fato simplificou a abordagem no desenvolvimento, já que síndico e programador, cliente e desenvolvedor, são a mesma pessoa. O projeto do programa se deu por refinamentos sucessivos. Assim, as idas e vindas, comuns a essa técnica, foi bastante abreviada.

2.1 Aspectos legais da gestão condominial

Toda organização possui uma estrutura administrativa e, no caso de um condomínio, essa estrutura é fixada em lei.

Ao contrário da gestão empresarial, que visa ao lucro, a gestão condominial visa ao controle, ou seja, o síndico deve prestar conta de suas ações aos condôminos.

Na estrutura organizacional de um condomínio, existem basicamente três instâncias: o síndico, o conselho fiscal e a assembleia geral.

Ao contrário do que muitos podem crer, a eleição do síndico, bem como a composição de um conselho fiscal e de uma assembleia geral, são obrigações instituídas pela Lei Federal nº 10.406/02 (novo Código Civil Brasileiro) e pela Lei Federal nº 4591/64 (Convenção Condominial).

Em particular, o Código Civil estabelece os deveres do síndico:

> Art. 1.348. Compete ao síndico:
> I - convocar a assembleia dos condôminos;
> II - representar, ativa e passivamente, o condomínio, praticando, em juízo ou fora dele, os atos necessários à defesa dos interesses comuns;
> III - dar imediato conhecimento à assembleia da existência de procedimento judicial ou administrativo, de interesse do condomínio;
> IV - cumprir e fazer cumprir a convenção, o regimento interno e as determinações da assembleia;
> V - diligenciar a conservação e a guarda das partes comuns e zelar pela prestação dos serviços que interessem aos possuidores;
> VI - elaborar o orçamento da receita e da despesa relativa a cada ano;
> VII - cobrar dos condôminos as suas contribuições, bem como impor e cobrar as multas devidas;
> VIII - prestar contas à assembleia, anualmente e quando exigidas;
> IX - realizar o seguro da edificação.

É importante observar que a competência estabelecida no inciso V demanda muito em termos de gerenciamento da rotina de um condomínio.

O dia a dia determina um fluxo de caixa e uma certa periodicidade para alguns tipos de serviço. Tem-se, assim, no inciso VI uma competência que somente se consubstanciará ao fim do exercício anual, acompanhando diariamente os lançamentos a débito das contas condominiais.

O artigo 1356 do Código Civil institui o Conselho Fiscal. Sua função é examinar a contabilidade do condomínio, que é de responsabilidade do síndico. Esse conselho não é de instituição obrigatória e, de fato, existem condomínios que não contam com essa instância. Entretanto, existindo, seus membros compartilham parte das responsabilidades administrativas do condomínio.

A assembleia geral (artigo 1334 do Código Civil), no rigor do termo, é a denominação das reuniões em que ocorrem as decisões, as prestações de contas e a aprovação de planos condominiais. Apesar disso, por extensão, também se refere à coletividade, isto é, à máxima instância do condomínio, composta por todos os proprietários das unidades condominiais.

Além da estrutura de um condomínio acima descrita, convem destacar alguns diplomas legais pertinentes.

É preciso observar que a legislação brasileira obedece a um princípio hierárquico, denominado ordenamento jurídico, que tem na Constituição Federal o seu elemento originário ou constituinte. A partir da Carta Magna, várias normas gerais tratam dos mais diversos campos do Direito.

O Código Civil é norma geral que dispõe sobre direitos e obrigações de ordem privada concernentes às pessoas, aos bens e suas relações. Em seu Capítulo VI, conceitua o condomínio geral, isto é, dá a conformação em sentido mais amplo em matéria de condomínios.

A convenção condominial e o regulamento interno são normas regulamentadoras, ou seja, são centradas na forma em que se materializará ou se instrumentalizará o Direito Civil instituído. Farber e Segreti (2006, p. 6) definem bem ambos os conceitos:

- A convenção condominial é uma autêntica lei interna da comunidade, destinada a resguardar, em proveito de todos, o patrimônio condominial e a moralidade ambiente, num sistema de normas que, mais rigorosamente do que as decorrentes do direito de vizinhança, objetiva garantir a todos os ocupantes das unidades autônomas o sossego, a tranquilidade e a segurança.
- O regulamento interno se preocupa com a rotina diária do condômino, com o seu funcionamento. Considerando que a lei não define sua abrangência, o regulamento interno poderá conter regras meramente procedimentais, mas também ir além, regulando o comportamento exigível dos condôminos, moradores e funcionários.

2.2 A gestão condominial

Como se observou, o novo Código Civil Brasileiro impõe ao síndico deveres. Em razão das responsabilidades assumidas, que recaem sobre uma única pessoa, é recomendável e necessário um suporte para alguns serviços administrativos, tais como, os que se referem aos incisos VI a IX.

Nesse contexto, quanto à administração condominial:

> "(…) identificam-se três modalidades de gestão: autogestão, co-gestão e terceirização. Na autogestão, o síndico, eleito na forma da Lei Federal nº 10.406/02, assume total responsabilidade e administra o condomínio. Na co-gestão, o síndico recorre a uma assessoria administrativa, ou seja, partilha suas tarefas com empresas especializadas na administração de condomínios, mas sem transferências das

> responsabilidades legais. Já a terceirização, uma das tendências atuais do mercado imobiliário, trata-se da eleição da própria administradora de condomínios como síndico, esta, assim, assumindo total responsabilidade, administrando e representando legalmente o condomínio." (Farber e Segreti, 2006, p. 3 e 4)

Exceto pela terceirização, as outras duas modalidades exigem a tomada de decisão pelo síndico. Nesse caso, existe uma característica preponderante na maioria dos condomínios residenciais: os síndicos não detem conhecimento gerencial suficiente para o mister.

Essa realidade é observada na pesquisa de Farber e Segreti (2006, p. 5), quando afirmam que apenas 39% dos síndicos contam com maior experiência na gestão de condomínios.

É natural, portanto, que haja uma carência em conceitos de gestão, tais como: identificar os problemas e as necessidades, mobilizar equipes de trabalho, desenvolver e controlar o andamento de soluções financeiramente viáveis.

De todos os conceitos, aquele sobre o qual o síndico deve-se debruçar é na identificação dos problemas e das necessidades, ou melhor, em suas causas.

É relativamente fácil definir indicadores de desempenho para acompanhar processos em um condomínio. Por exemplo: inadimplências (expressas em R$ e em percentual), atrasos nos pagamentos de fornecedores, atrasos no pagamento de serviços de concessionárias (água, luz, telefonia), faltas e atrasos por parte dos funcionários.

Diametralmente oposta à facilidade de estabelecer indicadores, a dificuldade em se identificar as causas dos problemas passa pela chamada análise de processos.

Na Tabela 3, adaptaram-se as etapas que Araujo (2001, p. 66) sugere para a análise de processos. As etapas são genéricas o suficiente para serem aplicadas em quaisquer tipos de processos gerenciais.

É preciso observar que os processos gerenciais são compostos eminentemente por etapas previsíveis, monitoráveis por indicadores de desempenho, por mais que seja difícil identificá-los.

Assim, investigar quem pichou o muro do condomínio não é um processo gerencial, pois os resultados da investigação são imprevisíveis. Já a cobrança das taxas condominiais é um processo gerencial, que pode ser causa das inadimplências e que pode ser monitorada por indicadores e influenciada por ações.

Tabela 3: A análise de processos em etapas

Etapa	Descrição
1. Escolha	É importante priorizar o estudo e escolher processos que realmente causem impactos significativos. Todo processo gerencial possui inicio, meio e, no fim, um produto.
2. Representação	Entender como um processo se desenvolve passo a passo. Busca-se retratar em fluxogramas o processo, deixando as mudanças para a etapa seguinte.
3. Análise	Ao representar o processo, frequentemente já se vislumbram melhorias. Essa fase é imprescindível para compreender como um processo deve ou deveria funcionar, modificando-o.
4. Documentação	Com as melhorias idealizadas, documenta-se esse método com fluxogramas e com explicações pertinentes.

Fonte: Arquivos de pesquisas pessoais do autor.

De acordo com a legislação e também por questões práticas, é frequente que os condomínios possuam um Conselho Consultivo Fiscal, que além das funções fiscais, exerce uma função mais participativa, tornando-se uma representação de moradores e de suas demandas.

Por fim, é importante frisar que a gestão condominial, ao contrário da empresarial, não visa ao lucro. Suas características lembram mais o fim público, ou seja, controlar os processos administrativos para prestar contas de forma transparente.

2.3 Tecnologias Java, banco de dados e padrões de projeto

Pode-se dizer que a linguagem Java se tornou o cerne multifacetado de várias frentes em desenvolvimento. Em seus primórdios, a linguagem Java se diferenciava ao incorporar inerentemente as grandes vantagens do paradigma da orientação a objetos.

Nos anos que se seguiram, diversas frentes de pesquisa das Ciências da Computação evoluíram exponencialmente; grandes empresas fomentaram organizações, que promoveram partido por essa evolução e várias tecnologias nasceram a partir do que, inicialmente, era apenas uma linguagem orientada a objeto.

As linguagens de programação são uma espécie de linguagem formal[1]. Conforme Gersting (2000, p. 427), as linguagens formais foram empregadas na definição das linguagens de alto nível e correspondem aos conjuntos (símbolos) aceitos por diversos tipos de autômatos (elementos matematicamente definidos, que "entendem" os símbolos aceitos).

1 Para uma abordagem mais detalhada, vide (Gersting, 2000, Capítulo 8).

Em seus dias atuais, Java transcendeu o conceito de linguagem. Tornou-se a base formal sobre a qual se desenvolveram ferramentas consagradas. Multiplicaram-se as soluções com base em Java, desenvolvidas pelas organizações mais influentes do planeta. Popularizou-se o uso da linguagem Java para vê-la presente em carros, computadores pessoais, celulares e caixas eletrônicos de bancos.

Essa gama de aplicações não deu apenas o poder da flexibilidade à comunidade Java. Deu a ela também a imprescindível capacidade de se integrar a outras tecnologias. Como será visto mais adiante, o exemplo histórico disso, foi o advento do Hibernate, como solução de integração dos conceitos de orientação a objetos e o de mapeamento objeto relacional dos Sistemas Gerenciadores de Banco de Dados (SGBD).

2.3.1 Java Server Faces

Conforme Geary e Horstmann (2010, p. 3), atualmente, pode-se escolher entre vários *frameworks* para desenvolver a interface com o usuário de uma aplicação *web*. Java Server Faces (JSF) é uma especificação de *framework* baseado em componentes.

A base em componentização permite que o desenvolvedor pense em seu projeto em um nível de abstração de nível mais alto. Exemplificando, ao exibir uma tabela com linhas e colunas, não se gera iterativamente o código HTML; ao invés disso, adiciona-se um componente tabela à página, que é gerada e atualizada por mecanismos específicos.

Podem-se utilizar conjuntos personalizados de componentes ou os de terceiros. Além disso, é possível utilizar um ambiente de desenvolvimento visual do tipo arraste e solte componentes em um formulário.

A especificação JSF possui as seguintes partes:

- Um conjunto padrão de componentes de interface do usuário;
- Um modelo de programação orientado a eventos;
- Um modelo de componentes que permite o desenvolvimento de novos componentes.

Um *framework* baseado em JSF não é apenas uma ferramenta *web* baseada em componentes. Trata-se de uma camada de visualização no padrão da especificação *Java Enterprise Edition* (JEE). Atualmente, vários *frameworks* atendem as especificações JSF, tais como MyFaces, IceFaces, PrimeFaces e Mojarra.

Convem citar também que o padrão JSF é definido por uma comunidade chamada *Java Community Process* (JCP), que é composta por empresas tais como Oracle, Novell, Siemens, Hewlett-Packard, Apache, Fujitsu, IBM, Macromedia, dentre outros. Por essa razão, torna-se evidente por que essa especificação é de fato um padrão mundialmente aceito.

No presente trabalho, utilizou-se os *frameworks* Mojarra e PrimeFaces para compor a camada de visualização.

2.3.2 Hibernate

Sobre o desenvolvimento do *framework* Hibernate, Fisher e Murphy (2010, p. 3) apresentam uma abordagem histórica tão interessante que os acompanham as linhas abaixo.

O Hibernate foi criado pela empresa JBoss Incorporated (hoje, pertencente à Red Hat) e seu código fonte foi desenvolvido em Java. Foi um dos primeiros *frameworks* de mapeamento objeto relacional (ou *object relational mapping* – ORM), que forneceu uma solução corporativa viável para a persistência de dados.

Nessa época, o Hibernate era uma solução proposta para o problema que havia entre o modelo relacional de tabelas e o modelo orientado a objetos (OO) do Java. A título de exemplo, um dos problemas era que as bases de dados relacionais não implementavam conceitos do paradigma da orientação a objetos, tais como polimorfismo, encapsulamento e acessibilidade. Outro dos grandes problemas, era a noção de igualdade, que é drasticamente diferente para Java e para SQL.

Inicialmente, o Hibernate fazia a ponte entre os paradigmas ORM e OO através de arquivos XML. Após a incorporação à empresa Red Hat, sua implementação passou por mudanças e os relacionamentos passaram a ser através de *Annotations* (ou anotações, metadados do Java, implementados na versão JDK 5.0).

Atualmente, as especificações Hibernate se aproximam das especificações do *Enterprise Java Beans* (EJB), que é um dos principais componentes do *Java Enterprise Edition* (JEE).

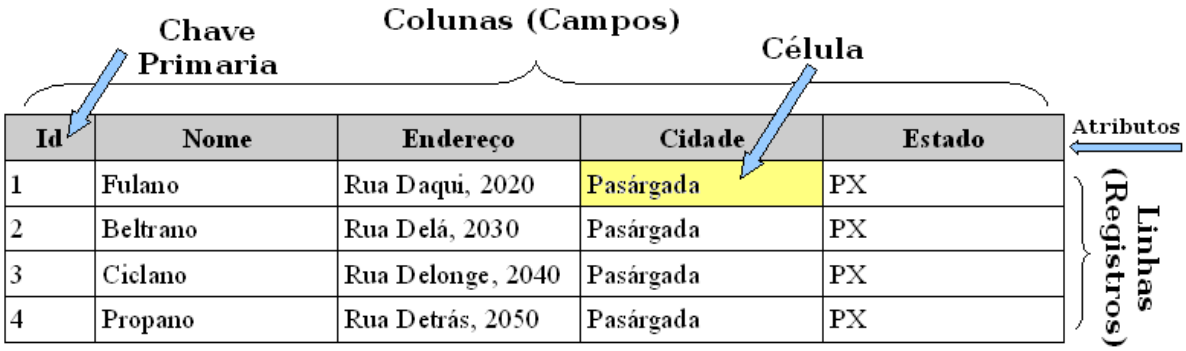
No presente trabalho, utiliza-se o Hibernate como ferramenta para implementar o paradigma *object relational mapping* (ORM), relacionando e mapeando, através das *annotations*, os objetos Java ao sistema de gerenciamento de banco de dados.

2.3.3 MySQL

O Sistema de Gerenciamento de Banco de Dados (SGBD) é um programa que implementa operações que visam, em última análise, à persistência de dados. Dentre vários artigos e livros a respeito do assunto, escolheu-se adaptar os estudos apresentados por Murach e Harris (2010, p. 93).

Em 1970, o Dr. Edgar Frank Codd apresentou o modelo teórico dos bancos de dados relacionais, que suplantava os problemas de recuperação e redundância de dados. Atualmente, o modelo de bancos de dados relacionais consiste de tabelas que, tipicamente, representam uma entidade do mundo real, como uma lista de produtos ou um cadastro de clientes.

Como mostra a Figura 1, essas tabelas contem colunas (ou campos) e linhas (ou registros). Cada coluna representa um atributo de uma entidade. Já as linhas, um conjunto de valores para uma determinada entidade. A interseção entre uma linha e uma coluna é chamada célula e armazena um único valor, correspondente a um atributo da entidade.



Id	Nome	Endereço	Cidade	Estado
1	Fulano	Rua Daqui, 2020	Pasárgada	PX
2	Beltrano	Rua Delá, 2030	Pasárgada	PX
3	Ciclano	Rua Delonge, 2040	Pasárgada	PX
4	Propano	Rua Detrás, 2050	Pasárgada	PX

Figura 1: Representação de tabela de cadastro de clientes em um banco de dados relacional.
Fonte: arquivos do autor.

Como se verifica na Figura 1, uma chave primária é tipicamente uma coluna, cujas células contem valores que identificam univocamente uma entidade. Mas, para ser chave primária, um campo preferencialmente deve possuir outra característica também muito importante: não deve ser editável.

Além das chaves primárias, os sistemas de gerenciamento de banco de dados permitem a definição de uma ou mais chaves não primárias. Essas chaves não primárias, em MySQL, são chamadas chaves únicas. Tais como as primárias, essas chaves únicas não se repetem, mas são editáveis. Um exemplo disso são os *e-mails*, campos editáveis, utilizados no cadastramento de clientes.

Um dos grandes diferenciais do modelo de banco de dados relacionais, em relação a um simples armazenamento de dados, é que as tabelas podem se relacionar umas com as outras.

As relações são feitas através da chave primária em uma tabela e da chave estrangeira em outra tabela. Empiricamente, a chave estrangeira é uma coluna de uma determinada tabela, que contem a chave primária de outra tabela.

Pode-se observar esse fato no esquemático da Figura 2 abaixo. A tabela Cliente possui uma chave primária chamada IdCliente. A tabela Produtos também possui a sua chave primária, IdProdutos. No entanto, para se configurar a relação um-para-muitos (*one-to-many*) – em que um cliente pode ter vários produtos relacionados a ele – a tabela Produtos deve conter a chave estrangeira Cliente_IdCliente (nome este no formato Tabela-Chave Primária). É importante observar que em Indexes, constam PRIMARY (referência à chave primária da tabela) e/ou fk_Produtos_Cliente (referência à chave estrangeira).

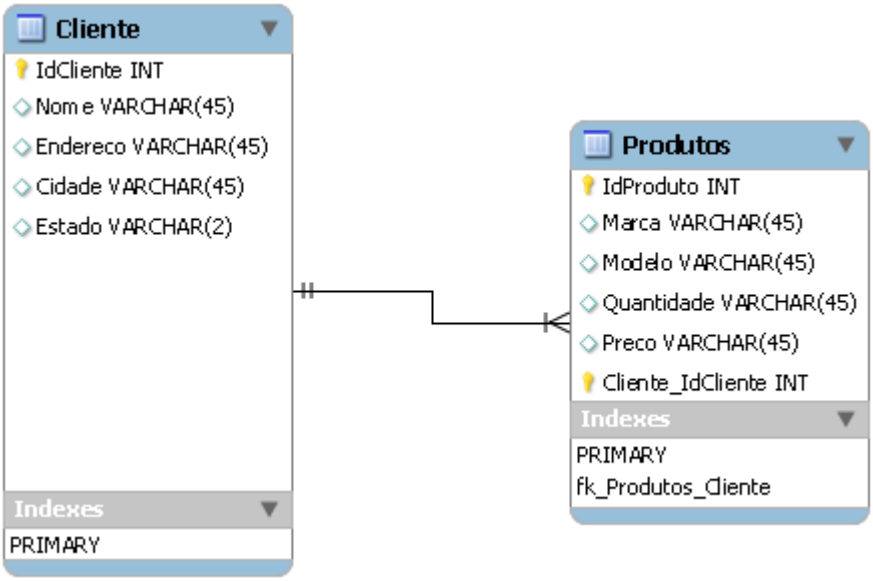


Figura 2: Esquemático demonstrando o relacionamento entre tabelas
Fonte: arquivos do autor.

A partir do MySQL 5.0, passou-se a dar suporte à integridade referencial e ao processamento de transações. Assim, havendo integridade referencial, não seria possível apagar uma linha que estivesse relacionada a uma outra tabela. Havendo processamento de transações, se houvesse uma alteração de valores nas tabelas de investimentos pré-fixados e nas de investimentos pós-fixados, a tabela extrato de conta somente seria atualizada somente se todos os processamentos forem concluídos com sucesso.

Com essa evolução, o MySQL tornou-se uma alternativa para substituir produtos comerciais como Oracle Database e Microsoft SQL Server.

A Tabela 4 mostra algumas das características do banco de dados MySQL, que o tornaram tão popular.

Tabela 4: Características e vantagens do MySQL.

MySQL é:
Viável. MySQL é gratuito para muitas aplicações e não é caro para outras.
Rápido. MySQL é um dos bancos de dados relacionais mais rápidos atualmente disponíveis.
Fácil. Comparado a outros sistemas de gerenciamento de banco de dados, MySQL é fácil de instalar e de usar.
Portável. MySQL executa na maioria dos sistemas operacionais modernos, incluindo Windows, Unix, Solaris e OS/2.
MySQL proporciona:
Suporte a SQL. MySQL dá suporte à linguagem padrão, que trata dados junto ao banco de dados relacional.
Suporte a múltiplos clientes. MySQL permite acesso de múltiplos clientes de interfaces diversas e de linguagens de programação diferentes, incluindo Java, PHP, Perl, Python e C.
Segurança. MySQL permite acesso somente a usuários autorizados.
Integridade referencial. MySQL evita alterações em dados de uma tabela que estejam relacionados a uma outra tabela.
Processamento de transações. MySQL somente permite o processamento de transações íntegras, isto é, se, dentre vários procedimentos, algum deles der errado, a transação é abortada, preservando os dados e as tabelas envolvidas.
Descrição:
MySQL é um sistema de gerenciamento de banco de dados de código aberto e é incluído nos pacotes de muitos provedores de serviços de *internet*.
MySQL foi desenvolvido pela empresa MySQL AB, que foi comprada pela Sun Microsystems. Esta por sua vez, foi comprada, juntamente com todos os seus produtos, pela Oracle Corporation.
Integridade referencial se refere à capacidade de manter intactos os relacionamentos entre as tabelas. Ou seja, não é possível apagar uma linha de uma tabela que contem uma chave primária referenciada como chave estrangeira em uma outra tabela.
Processamento de transações se refere à capacidade de processar como um única transação uma série de declarações SQL inter-relacionadas. Nesse caso, se uma das declarações falha, o processamento que foi executado pelas outras declarações pode ser desfeita, de forma que o banco de dados se mantem íntegro.

Fonte: Adaptado de Murach e Harris (2010, p. 107).

Nesse trabalho, o MySQL foi utilizado como sistema de gerenciamento de banco de dados. Adicionalmente, a Oracle também oferece uma ferramenta gráfica que proporciona uma grande praticidade no projeto do banco de dados: o MySQL Workbench. A Figura 2 demonstra um exemplo de sua utilização. As tabelas, as propriedades e os relacionamentos são rápida e graficamente configuráveis através de uma interface bastante intuitiva.

2.3.4 O padrão de projeto *Model – View – Controller* (MVC)

Gamma et al. (1995, p 2) resgatam a inspiração que deu origem aos padrões de projeto no campo das Ciências da Computação ao citar Christopher Alexander, Matemático, Arquiteto e Urbanista: "Cada padrão descreve um problema que ocorre uma vez e novamente outra vez em nosso ambiente e, então, delineia-se o cerne de uma solução para aquele problema, de tal maneira que se pode usar essa solução milhões de vezes, sem jamais fazer a mesma coisa duas vezes."

Apesar do contexto de Alexander ter sido a Arquitetura, seu trabalho inspirou os padrões de projetos, também conhecidos por *design patterns*. Segundo Gamma et al. (1995, p. 3), podem-se distinguir quatro elementos essenciais:

- O nome do padrão, normalmente uma ou duas palavras, é um artifício usado para designar um problema de projeto, suas soluções e consequências.
- O problema descreve quando se deve aplicar um padrão; explica o problema e seu contexto.
- A solução descreve os elementos que delinearam o projeto, seus relacionamentos, responsabilidades e colaborações.
- As consequências são os resultados e os dilemas enfrentados ao aplicar os padrões.

O advento dos padrões de projeto teve grande repercussão na manutenção dos sistemas. Em certo momento, passou-se a falar em separação de responsabilidades e em baixo acoplamento.

Um dos princípios do padrão MVC é a separação de responsabilidades. Assim, cada uma das três camadas tem responsabilidades distintas e bem definidas. Havendo essa separação, define-se que as camadas mantem um baixo nível de dependência umas com as outras. Isso permite que uma camada seja alterada sem afetar a outra.

Pode-se dizer que MVC é um padrão composto, isto é, compõe-se da integração de alguns dos padrões conhecidos. Apesar de ser uma composição de padrões de projeto, o MVC é considerado na literatura, ele mesmo, um padrão de projeto. Entretanto, de acordo com Freeman et al. (2004, p. 528), a estrutura do MVC se torna mais clara sob a óptica dos vários padrões que o compõem.

No MVC, identificam-se claramente os padrões: estratégia, observador e composição.

No padrão de projeto de estratégia, quando o usuário interage, a camada de visualização delega ao controlador a decisão do que fazer. Em uma classe controladora,

existem métodos que reagem às ações do usuário. Esse padrão é representado esquematicamente na Figura 3.

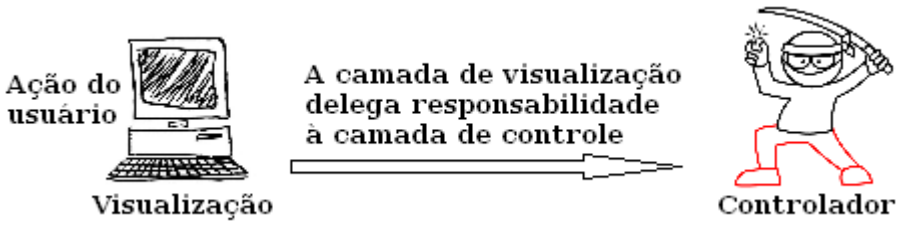


Figura 3: Esquemático - Padrão de projeto da estratégia
Fonte: arquivos do autor.

Em função da separação de responsabilidades, a camada de visualização apenas se preocupa com a apresentação da interface com o usuário. A camada de controle se preocupa em tratar as entradas do usuário com ações do modelo (regras de negócio).

No padrão de projeto observador, toda nova classe de visualização, que precisa ser notificada sobre alterações no estado do modelo, deve ser registrada como observador (vide Figura 4).

Uma vez registrados, os observadores serão notificados pela classe da camada de modelo sobre qualquer ação do usuário que altere seu estado. Dentre as classes observadoras podem existir classes da camada de visualização, bem como, classes da camada de controle.

É importante observar que a classe de modelo não tem dependências em relação às classes das camadas de visualização e de controle.

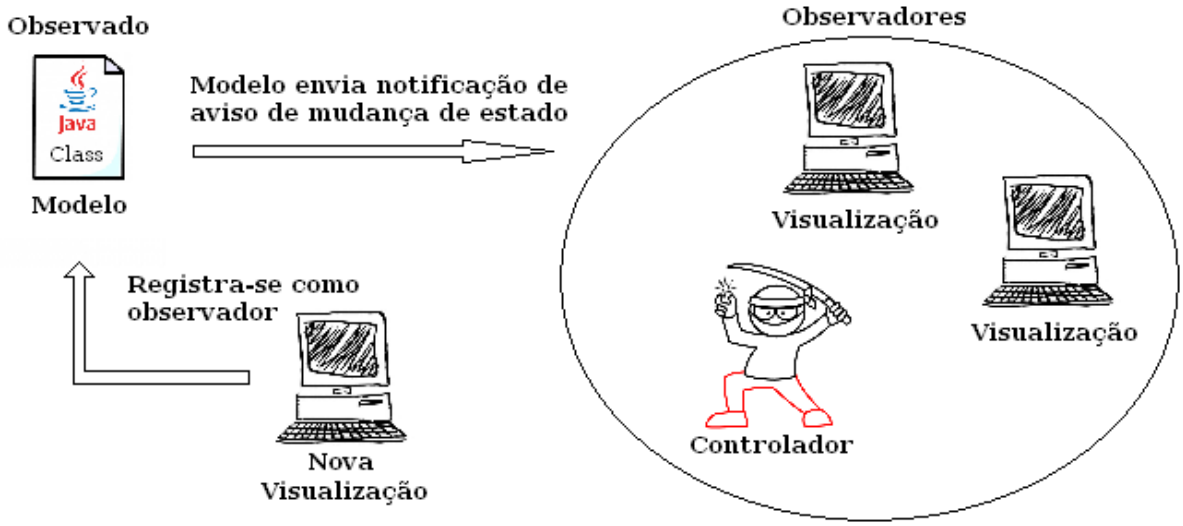


Figura 4: Esquemático - Padrão de projeto observador
Fonte: arquivos do autor.

No padrão de projeto de composição, a camada de visualização é uma composição de componentes, que são classes que implementam a interface gráfica com o usuário, conhecida também como *Graphical User Interface* (GUI).

Um componente no topo de uma camada de visualização contem outros componentes que, por sua vez, conterão outros e assim sucessivamente até se chegar a componentes unitários. Isto é, explodindo-se as camadas de um padrão de composição, verifica-se que o todo é composto por pequenas partes. Essa ideia é representada esquematicamente pela Figura 5.

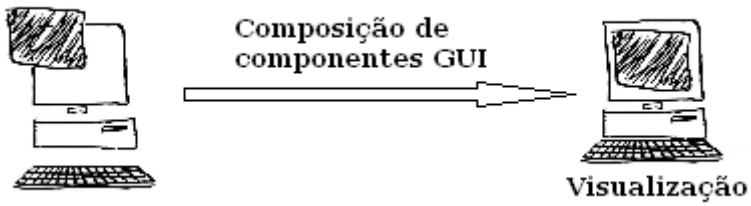


Figura 5: Esquemático - Padrão de projeto da composição
Fonte: arquivos do autor.

Por fim, reiterando-se: a fim de facilitar a manutenção dos sistemas, frequentemente se trabalha com a separação de responsabilidades em camadas. Segundo Luckow e Melo (2010), pode-se comparar o modelo de camadas de responsabilidades com o padrão MVC. A Figura 6 mostra esse comparativo, em que os retângulos azuis representam as camadas de acesso a dados (DAO), de regras de negócio (RN) e de apresentação. Em cinza, agruparam-se os retângulos correspondentes ao modelo (*model*) à visualização (*view*) e ao controlador (*controller*).

Figura 6: Comparativo entre o padrão MVC e o modelo de separação de responsabilidades.
Fonte: Adaptado de Luckow e Melo (2010, p. 181).

3. METODOLOGIA

Em pequenos condomínios, há uma tendência a uma gestão mais participativa, no sentido de que existe naturalmente uma propensão à aproximação do síndico e dos condôminos.

Quando as relações são relativamente prósperas, o síndico consegue converter as reivindicações em projetos. Porém, às vezes, isso só não é implementável pelo fato de que é difícil controlar projetos participativos. Além disso, o pequeno condomínio pode ser proporcionalmente mais eficiente que um condomínio grande e bem estruturado, haja vista que ganha em agilidade e flexibilidade.

A instituição de comissões seria solução, mas leva à dificuldade de comunicação, pois não é possível fazer reuniões constantemente entre seus membros e o síndico. A dificuldade passa a ser a coordenação das atividades descentralizadas.

O objetivo geral é o desenvolvimento de uma ferramenta de TI que auxilie o síndico na coordenação e no acompanhamento de atividades.

Considerou-se que, embora de pequeno porte, o condomínio possui recursos financeiros para a co-gestão (Farber e Segreti, 2006, p. 3), ou seja, as atividades são de responsabilidade do síndico, mas este é assessorado por uma empresa especializada em administração de condomínios.

Quanto às tecnologias Java, ainda que existam diversas ferramentas, evitou-se propositalmente discussões e comparativos acerca de desempenho. Como o foco é a integração de tecnologias e sua aplicação, as escolhas se deram simplesmente pela popularidade ou pela conveniência.

3.1. Desenvolvimento do projeto

Primordialmente, era preciso acompanhar as atividades do condomínio para coordená-las. Dessa forma, a partir dessas necessidades primordiais, algumas funcionalidades naturalmente se desdobraram.

Focando-se nesses objetivos específicos, concentrou-se o desenvolvimento nas seguintes funcionalidades:

- Cadastro de usuários: controle do acesso ao sistema.
- Cadastro de atividades: acompanhamento e coordenação das atividades.

- Agenda: acompanhamento e coordenação das atividades.

Em termos de implementação é importante mencionar o aspecto da reutilização de código, do qual se beneficiou o desenvolvimento exposto no presente capítulo.

Particularmente, os dois primeiros objetivos específicos citados acima, que se referem a cadastros de usuários e de atividades, possuem características análogas, em termos de desenvolvimento.

A fim de orientar e formalmente definir o que está sendo desenvolvido, a partir de um diagrama de caso de uso, conforme os padrões UML, elaborou-se um diagrama de classes. Essa elaboração é subjetiva, por se tratar de uma atividade de síntese, ou seja, aquela que depende da criatividade do desenvolvedor.

Nos tópicos seguintes, serão abordados com mais detalhes os conceitos acima.

3.1.1. Cadastro de usuários

Os sistemas gerenciadores envolvem abstrações de diversos entes do mundo real, tais como administradores, bancos de dados, cadastros, dentre outros.

É uma prática comum cadastrar usuários para legitimá-los à utilização de um sistema gerenciador. A razão é simples: segurança. Um sistema de gestão condominial aberto ficaria exposto ao democratizar informações financeiras ou particulares do condomínio. É frequentemente indesejável essa exposição.

Assim, acessar o sistema é um caso de uso que faz parte da administração do cadastro de usuários. Essa é uma forma de restringir o acesso ao sistema gerencial do condomínio. E, da mesma forma, fazem parte também os casos de inserção, edição, consulta e exclusão de outros usuários.

A Figura 7 demonstra o caso de uso relacionado à administração do cadastro de usuários. No caso de uso Administrar Cadastro, os atores são o Síndico e o Sistema Gerenciador de Banco de Dados, o SGBD. Esses atores estão associados ao caso de uso e essa associação é representada por linhas cheias.

A representação da Figura 7 permite a interpretação de que uma pessoa (habilitada) faça uso do sistema gerencial para administrar um cadastro. Um dos usos possíveis será, por exemplo, a inserção de um novo usuário no sistema gerenciador de banco de dados.

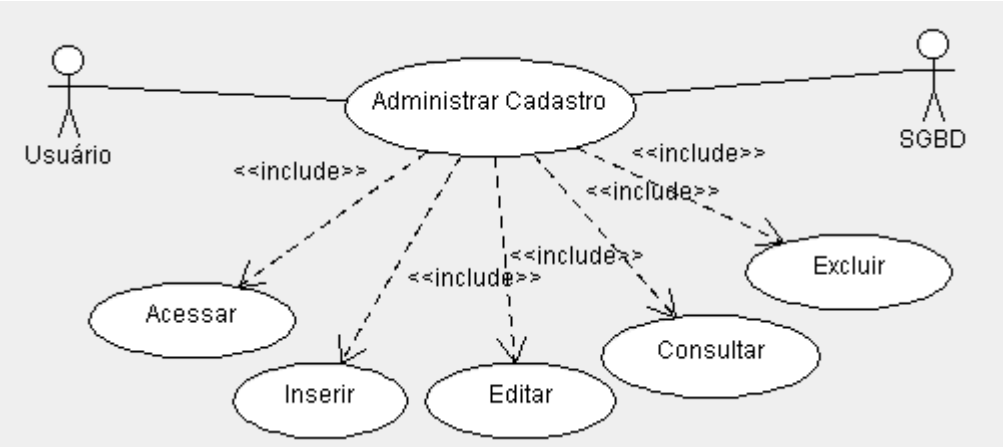


Figura 7: Diagrama de caso de uso Administrar Cadastro, demonstrando representação de atores e casos de uso e associações.
Fonte: Arquivos pessoais do autor.

Como se pode visualizar, o caso de uso Administrar Cadastro inclui outros cinco casos: Inserir, Editar, Consultar, Excluir e Acessar. Este último restringe o acesso ao sistema, ao exigir *login* e senha. Os cinco casos se incluem como parte de Administrar Cadastro e a relação de inclusão (<<include>>) é representada por linhas tracejadas.

Já se fez alusão ao fato de que a elaboração do diagrama de classes é um processo criativo do desenvolvedor. Mas, existe uma abordagem a partir da qual se começa a refinar o modelo.

A abordagem consiste em:

- Atribuir uma classe de interface para cada ator.
- Criar uma classe de controle e instanciar classes para cada caso de uso.
- Criar classes de entidade para os casos de uso.

Seguindo a abordagem acima, na Figura 8 abaixo, representa-se um esboço do diagrama de classes, conforme o padrão UML.

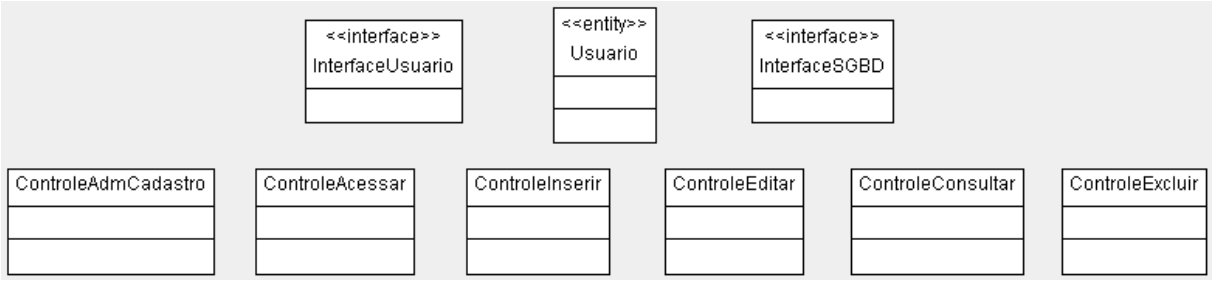


Figura 8: Diagrama de classes do caso de uso Administrar Cadastro, demonstrando as classes criadas.
Fonte: Arquivos pessoais do autor.

Na sequência, convem lembrar que o padrão de projeto adotado é o MVC. Aliado ao padrão, separou-se as responsabilidades por camadas, de modo que a Tabela 5 auxilia a compreensão do método de desenvolvimento abaixo descrito.

Tabela 5: Visualização esquemática do padrão MVC e separação de responsabilidades

		Responsabilidades		
		Acesso a dados	**Regras de negócio**	**Apresentação**
Camadas	***Model***	UsuarioDAO UsuarioDAOHibernate DAOFactory	UsuarioRN	-
		Usuario (POJO)		
	View	-	-	JSP ou XHTML
	Controller	-	-	*Bean*

Fonte: Arquivos pessoais do autor.

A camada de apresentação (vide Figura 6, p. 28) foi idealizada utilizando os recursos do PrimeFaces, uma implementação do JSF (vide 2.3.1 Java Server Faces, p. 21) e a camada de acesso a dados, os recursos do Hibernate (vide 2.3.2 Hibernate, p. 22).

Serão explicadas como se deu a síntese do diagrama de classes final, representado na Figura 9. Além disso, com base na Figura 7, apenas por questão de ordenamento do texto, a análise se dará da esquerda para a direita, isto é, do Usuário para o SGBD.

A interface entre o usuário e o sistema gerencial se dá através de classes *bean* e páginas XHTML. Representou-se essa interface de apresentação como uma anotação no diagrama de classes UML. No modelo JSF, as chamadas classes *backing bean* (ou, simplesmente, *bean*) controlam a comunicação com o XHTML - a camada de apresentação do sistema (vide Figura 6, p. 28).

Feitas as considerações acima, uma classe chamada UsuarioBean controlará a comunicação entre a camada de apresentação e a camada do modelo.

Se todas as propriedades dos usuários estiverem definidas na classe UsuarioBean, será necessário o envio dessa classe para gravação em banco de dados, o que desvirtua o conceito MVC. Nesse caso, um objeto da camada de controle está avançando sobre a camada de modelo (vide Tabela 5). Ou, visto por outro ângulo, um objeto cuja responsabilidade é controlar a camada de apresentação também detem parte da responsabilidade pelo acesso a dados.

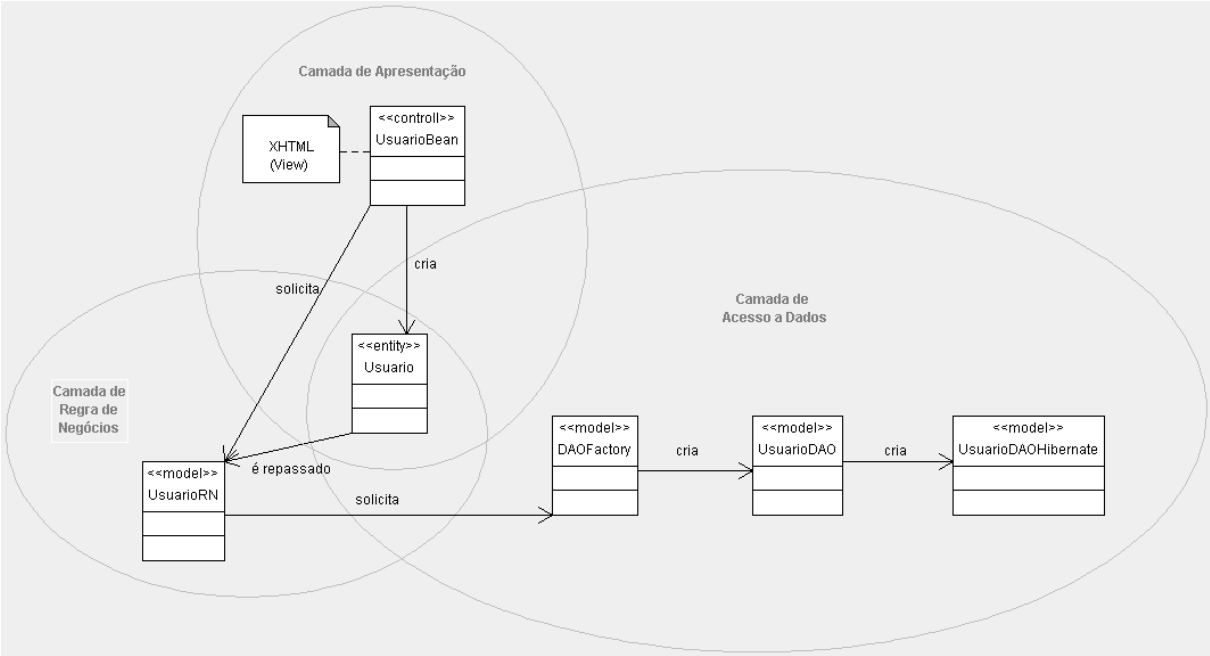


Figura 9: Diagrama de classes do caso de uso Administrar Usuário, demonstrando o padrão MVC e a separação de responsabilidades por camadas.
Fonte: Arquivos pessoais do autor.

Para solucionar o problema, cria-se a chamada classe POJO, *Plain Old Java Object*, ou seja, uma classe simples, contendo apenas dados, que trafegará livremente por todas as camadas como um mensageiro (vide Figura 9).

Dessa forma, a classe UsuarioBean contem como uma de suas propriedades uma classe POJO, que chamaremos Usuario. Além disso, convém ressaltar que as classes POJO são abstrações de tabelas de um banco de dados, o que representa a operacionalização do mapeamento objeto-relacional.

Comparando-se a Figura 8 e a Figura 9, observa-se que temos a camada de apresentação desenvolvida. O que era uma classe InterfaceUsuario se tornou XHTML e as classes UsuarioBean e Usuario.

Prosseguindo no desenvolvimento, passaremos pela camada de regra de negócios. Essa responsabilidade representa o cerne da modelagem, pois é nesta camada que ocorrerão as decisões. Na Figura 8, regras de negócios[2] foram definidas conforme os possíveis casos de uso do sistema. Ao Administrar Cadastro, o usuário poderá se deparar com situações como Inserir, Editar, Consultar, Excluir e Acessar.

2 Classes da camada de controle do padrão MVC possuem definição um tanto diferente dos casos de uso, que, convertidos em diagramas de classes, podem ser definidos em UML com o estereótipo de controlador (*control* ou *controller*). Buscou-se contornar tal fato usando a expressão “regra de negócios”.

Analisando-se tais situações, evidencia-se ser mais coerente que as regras de negócio sejam representadas em uma única classe. Por exemplo, seria incoerente fazer uma instanciação para inserir um usuário no cadastro e outra instanciação para editar o cadastro desse mesmo usuário.

Dessa forma, as ações de Inserir, Editar, Consultar, Excluir e Acessar foram definidas em uma única classe: UsuarioRN. Nela, estão definidos quais métodos serão chamados, quando isso acontecerá, como serão tratados os erros, bem como outras regras.

Em uma classe de regra de negócio, deve-se definir:

- Uma propriedade DAO (UsuarioDAO), correspondente à regra de negócio que se deseja definir.
- Uma instanciação dessa propriedade, usando DAOFactory no construtor.

Uma vez tendo recebido dados da classe Usuario, a classe UsuarioRN solicita à classe DAOFactory que gere as classes UsuarioDAO e UsuarioDAOHibernate. Esta possui o código necessário para acessar e tratar dados de um banco, através da plataforma Hibernate. Aquela possui apenas as assinaturas dos métodos e os atributos necessários para o repasse de dados. Essa descrição corresponde à implementação do padrão DAO, que separa nitidamente as responsabilidades de acesso a dados das de regras de negócio.

Com isso, verifica-se o desenvolvimento completo do diagrama de classes sugerido na Figura 9, a partir do diagrama inicial da Figura 8.

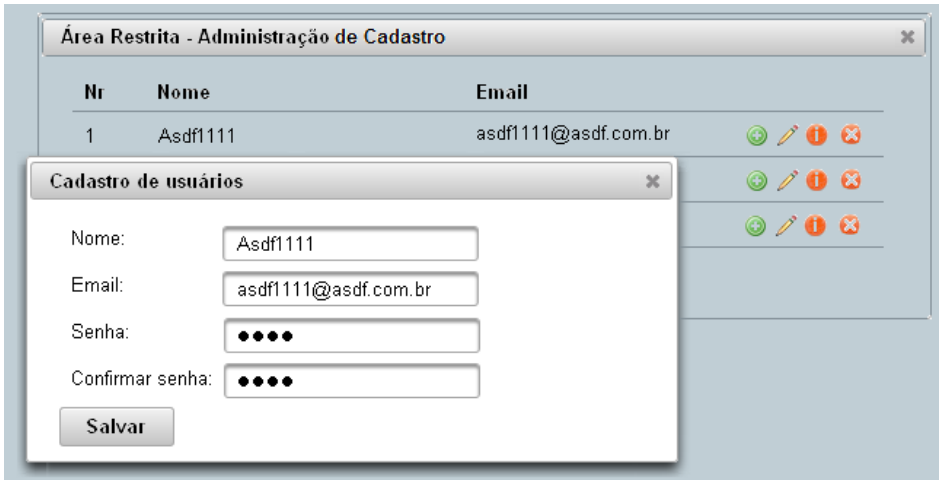


Figura 10: Edição de dados na área de administração do cadastro de usuários
Fonte: arquivos pessoais do autor.

A Figura 10 ilustra a página XHTML controlada pela classe UsuarioBean. Outros detalhes da implementação podem ser encontrados no APÊNDICE.

3.1.2. Cadastro das atividades

Do ponto de vista analítico, cadastrar usuários como parte de uma atividade é o mesmo que criar grupos específicos de usuários.

Observa-se que, no caso típico, um usuário participará de um grupo responsável por uma atividade. O grupo de usuários assim formado recebe um nome sugestivo e passa a ter componentes.

A reutilização de código é uma máxima já largamente difundida entre os desenvolvedores. Nessa proposta de trabalho, também se falou em reuso da metodologia de desenvolvimento, sobre a qual discorrerão as linhas seguintes.

O caso de uso Administrar Atividade é essencialmente análoga ao caso Administrar Usuário. A diferença é que existe um caso de uso adicional, referente a uma associação entre um usuário já cadastrado e uma atividade.

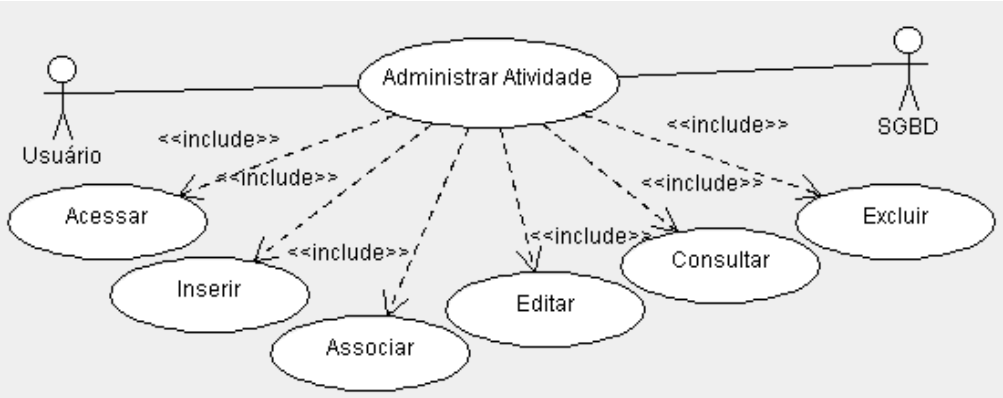


Figura 11: Diagrama de caso de uso Administrar Atividade
Fonte: arquivos pessoais do autor.

Associar um usuário a uma atividade é relativamente simples, quando se instancia um objeto Usuario na própria classe AtividadeBean. Dessa forma, todos os dados relativos a um usuário acompanharão a atividade relacionada.

Da mesma forma, se todas as propriedades das atividades estiverem definidas na classe AtividadeBean, será necessário o envio dessa classe para gravação em banco de dados, o que desvirtua o conceito MVC. Nesse caso, um objeto da camada de controle está avançando sobre a camada de modelo. Ou, visto por outro ângulo, um objeto cuja responsabilidade é controlar a camada de apresentação também detem parte da responsabilidade pelo acesso a dados.

Reutilizando-se artifício anterior, cria-se uma classe POJO, contendo apenas dados, que trafegará livremente por todas as camadas como um mensageiro (vide Figura 12).

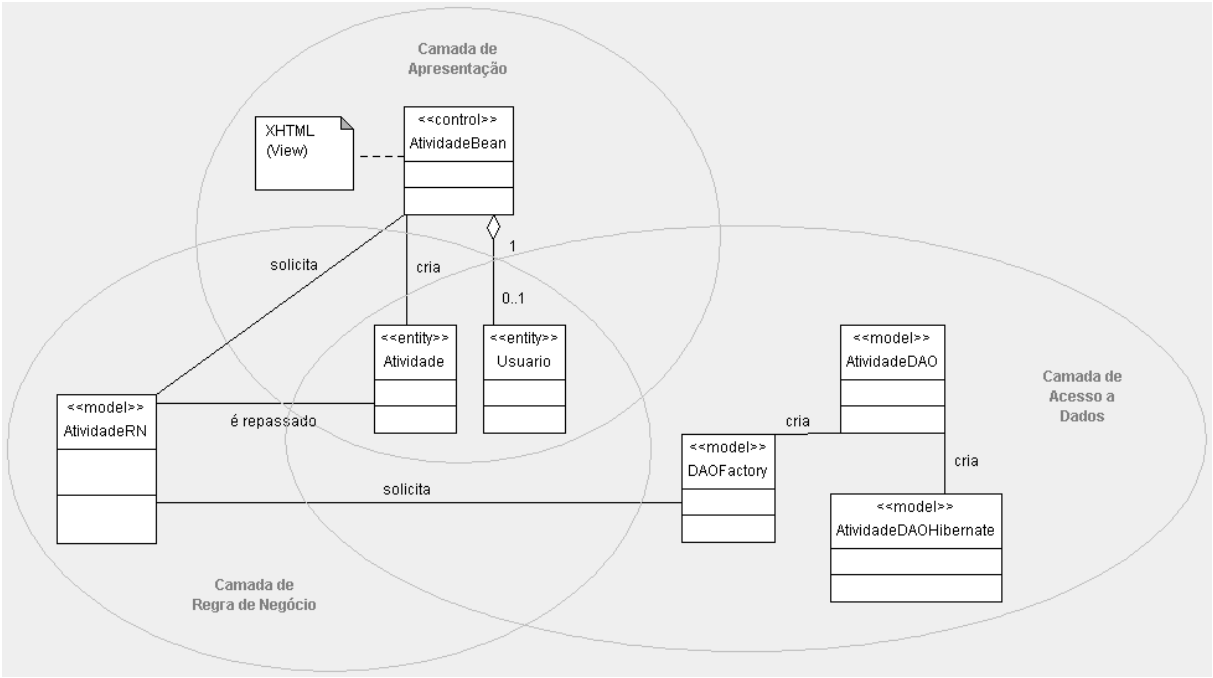


Figura 12: Diagrama de classes do caso de uso Administrar Atividade, demonstrando o padrão MVC e a separação de responsabilidades por camadas.
Fonte: Arquivos pessoais do autor.

Também no presente caso, evidencia-se ser mais coerente que as regras de negócio sejam representadas em uma única classe. Por exemplo, seria incoerente fazer uma instanciação para inserir uma atividade e outra instanciação para edição dessa mesma atividade.

Dessa forma, as ações de Inserir, Associar, Editar, Consultar, Excluir e Acessar foram definidas em uma única classe: AtividadeRN. Nela, estão definidos quais métodos serão chamados, quando isso acontecerá, como serão tratados os erros, bem como outras regras.

Nessa classe de regra de negócio, deve-se definir:

- Uma propriedade DAO (AtividadeDAO), correspondente à nova regra de negócio definida.
- Uma instanciação dessa propriedade, usando DAOFactory no construtor.

Uma vez tendo recebido dados da classe Atividade, a classe AtividadeRN solicita à classe DAOFactory que gere as classes AtividadeDAO e AtividadeDAOHibernate. Esta possui o código necessário para acessar e tratar dados de um banco, através da plataforma Hibernate. Aquela possui apenas as assinaturas dos métodos e os atributos necessários para o

repasse de dados. Essa descrição corresponde à implementação do padrão DAO, que separa nitidamente as responsabilidades de acesso a dados das de regras de negócio.

Observa-se que a classe DAOFactory está sendo reutilizada. Assim, é preciso modificá-la para acrescentar o seguinte método:

```
public static AtividadeDAO criarAtividadeDAO() {
        AtividadeDAOHibernate atividadeDAO = new AtividadeDAOHibernate();
        atividadeDAO.setSession(HibernateUtil.getSessionFactory().getCurrentSession());
        return atividadeDAO;
}
```

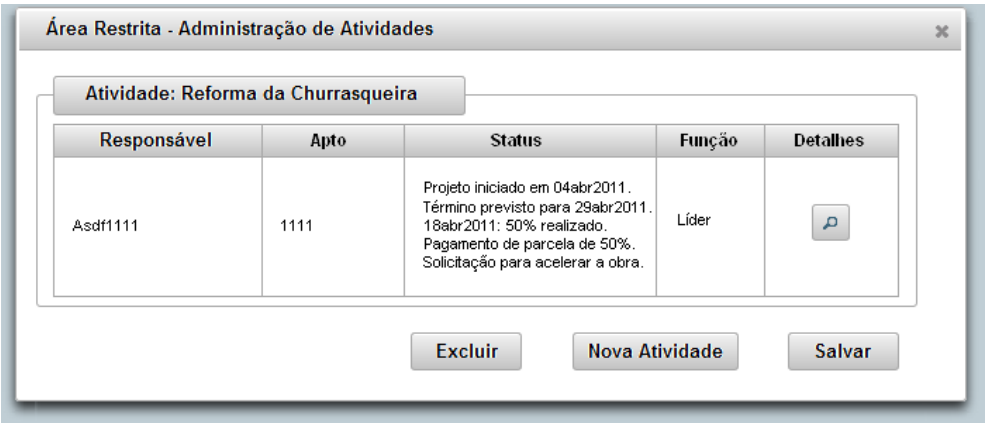


Figura 13: Área de acesso restrito para administração das atividades.
Fonte: arquivos pessoais do autor.

Com isso, verifica-se o desenvolvimento completo do diagrama de classes sugerido na Figura 12, a partir do diagrama inicial da Figura 11.

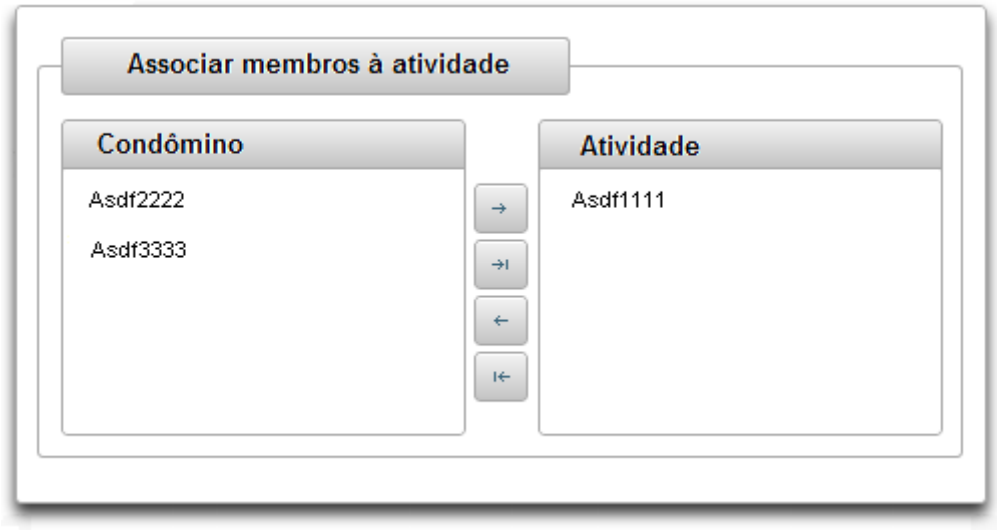


Figura 14: Componente picklist: utilização para associar um usuário à respectiva atividade. O primeiro da lista Atividade é sempre o líder do grupo.
Fonte: arquivos pessoais do autor.

A Figura 13 ilustra a página XHTML controlada pela classe AtividadeBean. Apesar da apresentação diferente, em relação à página de administração de usuários, observa-se que as

mesmas funcionalidades estão presentes. Um clique no componente DataTable da Figura 13, ativa a edição dos responsáveis por uma atividade. São ativados os componentes *picklist* e *overlay panel* do *framework* PrimeFaces (Figura 14). Outros detalhes da implementação podem ser encontrados no APÊNDICE.

3.1.3. Agenda

Essa última ferramenta foi implementada a partir da necessidade de proporcionar ao gestor a visão sistêmica, isto é, em uma única interface, ser possível observar os principais aspectos das atividades em curso no condomínio.

Segundo pesquisas do *Project Management Institute - PMI*, no Brasil, as maiores causas de insucessos nos projetos estão por conta de: descumprimento de prazos, problemas de comunicação e dificuldades relacionadas ao escopo do projeto.

Para auxiliar na definição das informações disponibilizadas ao gestor do condomínio, elegeram-se aqueles três mais incidentes aspectos (vide Figura 15). Sem grande perda de generalidade, permitiu-se considerar que, por exemplo, atrasos em atividades remetem a atrasos no projeto.

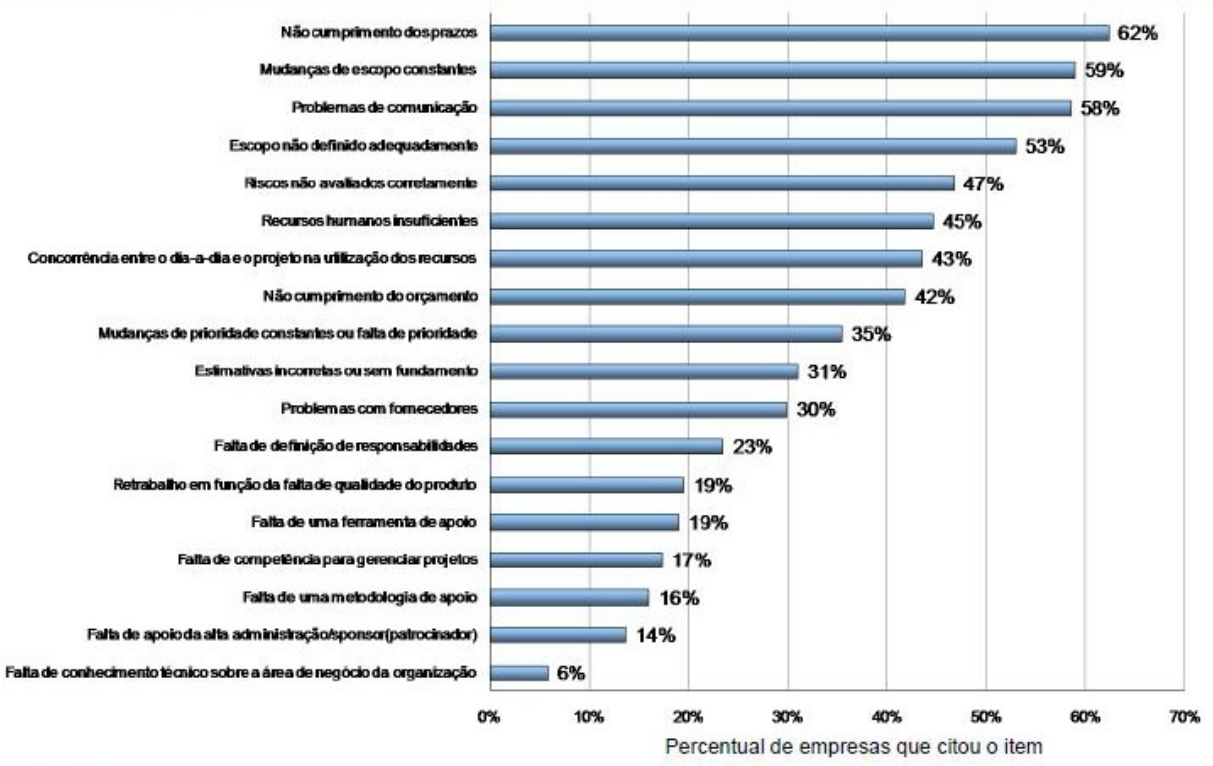


Figura 15: Causas de falhas em projetos, segundo pesquisa do PMI no Brasil, em 373 empresas.
Fonte: PMI

Com relação aos prazos, uma agenda seria uma boa forma de controlá-los. Para o gestor, uma visão geral bastante prática é a mensal, pois, como as obrigações e os recebíveis se dão nessa janela de tempo, o melhor é trabalhar dessa forma.

Porém, como na agenda real, no mesmo espaço, não é possível incluir toda a informação (início/término das atividades, responsáveis, atividades e observações gerais). Seria conveniente disponibilizar dados gerais e, caso necessário, detalhes à parte.

Com relação à comunicação, a disponibilidade de uma agenda comum facilita compartilhar as informações, contribuindo para o bom relacionamento. A agenda, nesse ponto, torna-se um canal centralizado de comunicação entre gestores, facilitando a coordenação das atividades.

O escopo de projeto, ou seja, a abrangência a que faz menção a metodologia PMI pode ser aplicada no âmbito das atividades. Na Agenda, não se modificam as atividades, apenas se visualiza, de modo que seu escopo não se altera; para modificá-lo, deve-se alterar o cadastro da própria atividade.

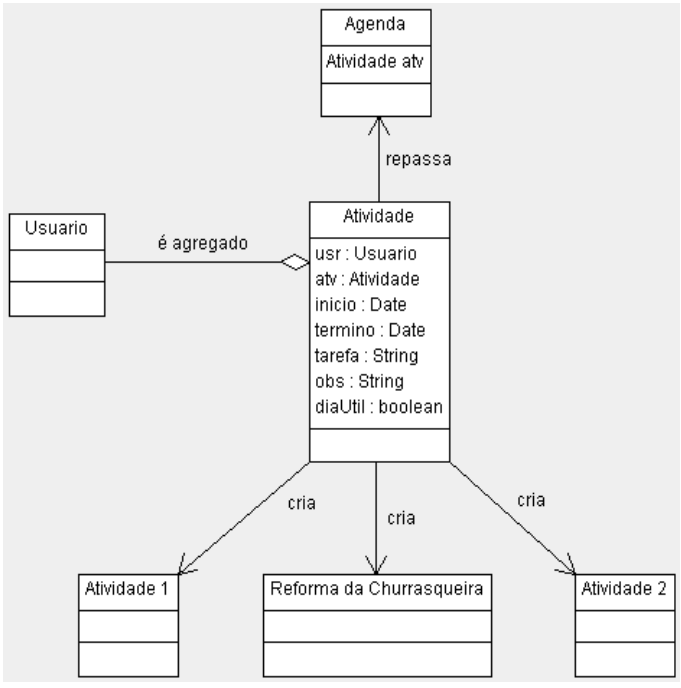


Figura 16: Classe Agenda como fachada de apresentação de várias informações.
Fonte: arquivos pessoais do autor.

Analisando-se outro aspecto, verifica-se que as informações desejadas já se encontram gravadas em banco de dados. Ao se cadastrar as atividades e as informações que lhes foram acrescidas no decorrer do tempo, todas as informações foram armazenadas nos bancos de

dados. Assim, apenas é necessário uma classe de fachada que reúna tais informações e as apresente. Esse é um padrão de projeto conhecido como *Façade* ou Fachada.

A Figura 16 omite propositalmente as interações com as classes de banco de dados, a fim de simplificar a visualização do padrão fachada. A vantagem está no repasse das dependências da classe Agenda para a classe Atividade.

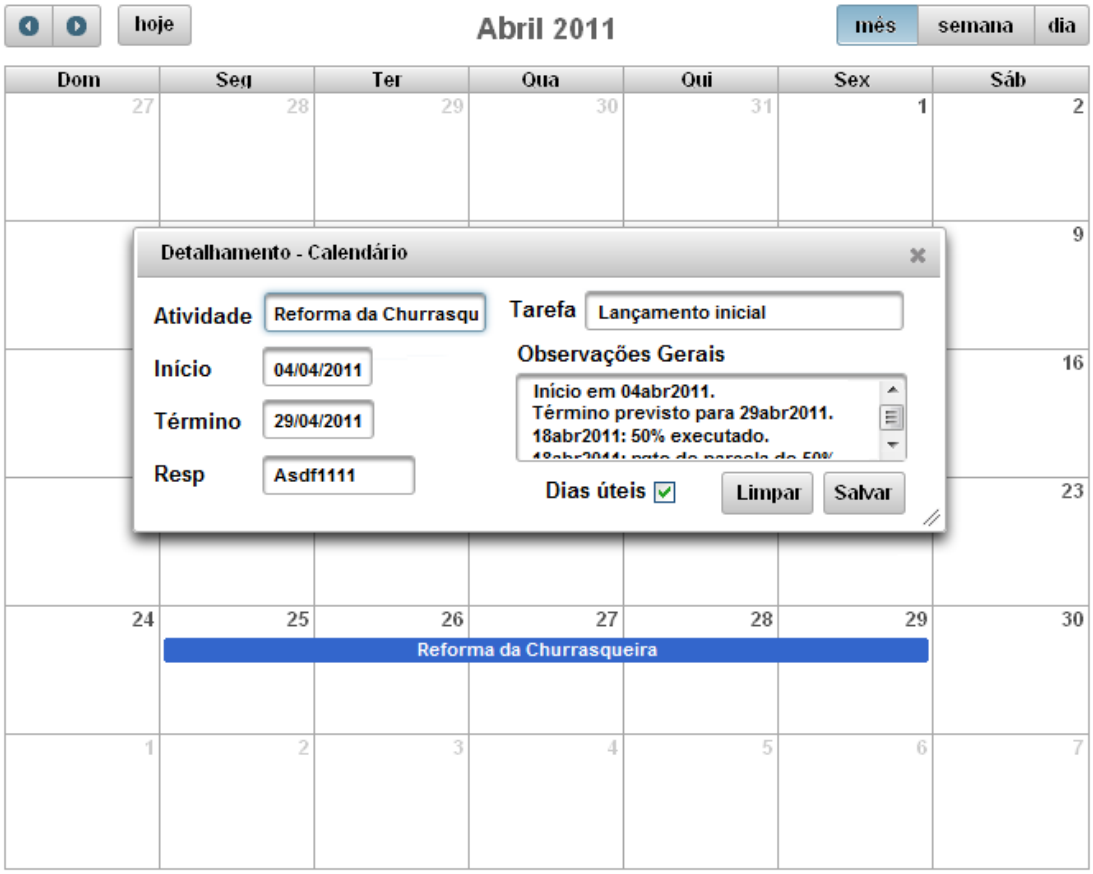


Figura 17: Agenda de acompanhamento dos projetos: detalhe dos apontamentos.
Fonte: arquivos pessoais do autor.

Conforme estabelece o modelo JSF, uma classe AgendaBean é responsável pelo controle da camada de apresentação. Essa classe possui o desenvolvimento bastante similar ao da classe AtividadeBean (vide Figura 12).

A página XHTML controlada por AgendaBean é aquela apresentada na Figura 17, em que se destaca o fato de que os botões Limpar e Salvar se destinam ao campo Observações Gerais.

4. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Houve uma pesquisa realizada entre o Conselho Consultivo e o Síndico, na qual se levantaram as possíveis causas dos problemas administrativos do condomínio. Mencionou-se que uma das grandes dificuldades no gerenciamento das atividades condominiais seria a coordenação entre elas. A Tabela 6 demonstra alguns dos principais problemas levantados.

Tabela 6: Levantamento dos problemas, principais causas e retrabalho.

Processos	Problemas	Causas	Retrabalho
Pagamento	Pagamento de juros de mora	Datas de pagamentos esquecidas.	Ligações à Administradora do Condomínio. Em média, 5 min/ligação, em horário comercial.
Compra	Atraso de fornecimento	Falta de acompanhamento de entrega.	Atualizar orçamento ou mudar de fornecedor. Em média, 7 dias úteis.
Cobrança	Falta de isonomia	As grandes inadimplências eram as mais acompanhadas.	Em média, 4 meses para regularização de cada inadimplente.
Comunicação	Descontinuidade de ações	Anotações do Síndico que não eram do conhecimento de mais ninguém.	Retrabalho de ação já executada. Em média 3 dias.
Serviços de manutenção	Falta de acompanhamento	Inexistência de escala de revezamento de inspeções.	Contratação de serviços complementares. Em média, 5 dias para contratar e 2 dias para corrigir.

Fonte: arquivos pessoais do autor.

As obrigações relativas ao acompanhamento e à coordenação entre as diversas atividades acabam sendo o foco da rotina do Síndico. Os processos de pagamento, cobrança, comunicação são executados diretamente pelo Síndico. O processo de compras tem seus orçamentos feitos por membros do Conselho Consultivo. É deste também a responsabilidade pelo acompanhamento das obras e dos serviços, mas a autorização destes é do Síndico, o que não o exime de acompanhar todas as atividades do Condomínio.

Devido ao volume de processos para coordenar, o programa trouxe uma vantagem operacional ou um ganho de escala relativo. Diminuindo-se a necessidade de reuniões constantes, bem como mantendo dados disponíveis a todos os envolvidos, o programa desenvolvido veio a facilitar a tomada de decisão.

Considerando as funcionalidades implementadas, o sistema de TI apresentou recursos que permitem uma grande economia de tempo, como a agenda e o cadastro de atividades. A

Tabela 7 demonstra as soluções propostas com o apoio do *software* e os resultados obtidos frente aos problemas levantados na tabela anterior.

Tabela 7: Principais soluções propostas e resultado em 6 meses de monitoramento.

Processos	Problemas (Causas)	Soluções	Resultados
Pagamento	Pagamento de juros de mora. (Datas de pagamentos esquecidas).	Agendamento no sistema.	Não houve esquecimentos. Desnecessidade de ligações à Administradora por esse motivo.
Compra	Atraso de fornecimento (Falta de acompanhamento de entrega)	Agendamento de datas limites antes da entrega a fim de propiciar troca de fornecedor, se for o caso.	Em média, 7 dias úteis por evento economizado. Furo do fornecedor contornado em uma situação, devido à antecipação pelo acompanhamento.
Cobrança	Falta de isonomia (As grandes inadimplências eram as mais acompanhadas)	Cobrança efetivada de forma agendada. Acompanhamento semanal.	Em média, 2,5 meses para regularização de cada inadimplente.
Comunicação	Descontinuidade de ações (Anotações do Síndico que não eram do conhecimento de mais ninguém)	Estabelecimento de regra: anotações devem constar em campo observações na Agenda.	Inexistência de fato reincidente no período monitorado
Serviços de manutenção	Falta de acompanhamento (Inexistência de escala de revezamento de inspeções.).	Após contato com o grupo, com base na Agenda, os próprios responsáveis passaram a estabelecer os revezamentos. Passou-se a postar o resultado das inspeções no campo observações da Agenda.	Os contratos passaram a ser cobrados mais efetivamente.

Fonte: arquivos pessoais do autor.

A existência de dados, prontamente acessíveis e beneficamente claros, tornam o sistema bastante prático. Nesse ínterim, verificou-se uma importante constatação: a curva de aprendizagem de um sistema de TI que agrega conceitos de gestão é menor que a mesma curva focada na aprendizagem das técnicas de gestão condominial.

Empiricamente, em quatro meses (vide Figura 18), observou-se que os esforços de nivelamento do grupo, em termos de assimilação do sistema de TI, desempenharam um papel quase cinco vezes mais relevante do que a mesma assimilação de conteúdo versando sobre gestão de projetos ou matérias correlatas.

A escala da Figura 18 tomou por base a quantidade de aplicações dos conceitos gerenciais, ou aplicados através do sistema de TI, ou aplicados através do conhecimento de gestão.

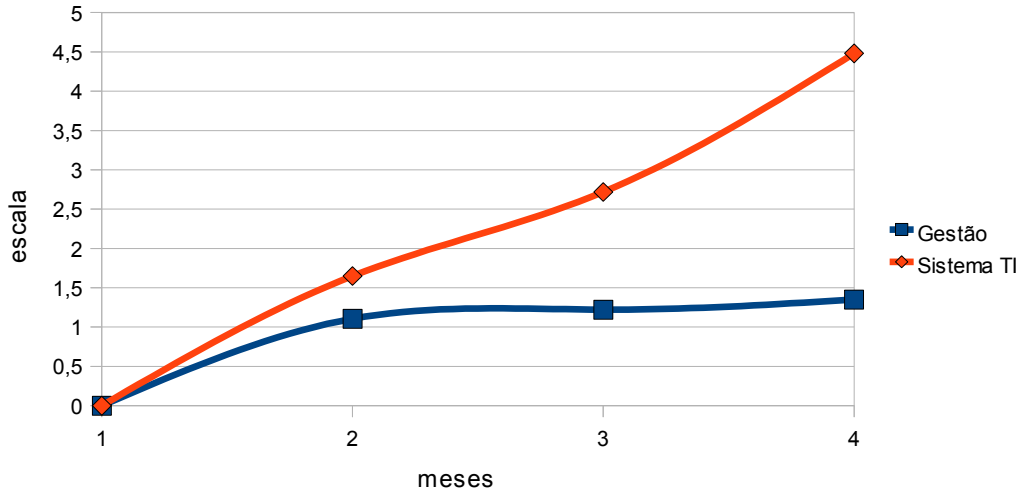


Figura 18: Curva de aprendizagem comparativa entre o sistema de TI e as técnicas de gestão. Fonte: arquivos pessoais do autor.

O uso do sistema de TI para gestão condominial foi rotulado por dois participantes como intuitivo, com o que se quis dizer que a utilização do programa dispensa treinamentos demorados e exaustivos. Muitas das vezes, bastaram algumas orientações rápidas, além da exposição inicial, que se deu em uma reunião de uma hora e meia.

5. CONSIDERAÇÕES FINAIS

A maioria dos sistemas condominiais são direcionados para as empresas. Tal fato levou ao empreendimento desse projeto, com um sistema de TI com foco nas atividades rotineiras de um síndico.

Após sua codificação, o sistema de TI para a gestão condominial apresentou uma grande tendência a corroborar sua importância e sua funcionalidade. Aspectos importantes embasaram essa observação: facilitou a coordenação dos grupos envolvidos em projetos, apresentou economia de tempo, curva de aprendizagem reduzida, retorno financeiro por redução do retrabalho.

Em parte, na fase de desenvolvimento e também na fase de testes, percebeu-se que o programa pode ser conformado para a utilização em outras atividades, que exijam a coordenação das ações de vários participantes, em formato simples e direto, possivelmente, agregando tecnologia, como e-mails e SMS.

Poder-se-ia argumentar que é por essa razão que não existem sistemas de TI específicos para as atividades do síndico: suas atividades já seriam alvo de programas de gestão administrativa.

Entretanto, uma das premissas que motivou o presente trabalho continua tendo um valor preponderante: as atividades condominiais são muito específicas, conformadas por aspectos legais e muito restringidas financeiramente.

É fato a carência no mercado, quando o assunto é sistemas de TI para o gerenciamento de atividades sujeitas à legislação federal, estadual ou municipal. Exemplos concretos, além da gestão condominial, é a gestão contábil. Existem muitos *softwares* contábeis, mas é muito raro encontrar um *software* para gestão administrativa da contabilidade.

Por fim, cabe ressaltar que uma das principais observações, se não a mais importante, é o fato de que é muito mais rápido ensinar a operar um *software* que agrega vários conceitos complexos do que ensinar os próprios conceitos complexos. Certos conhecimentos demandam muito mais tempo para serem ensinados, de modo que, para a prática do dia a dia, a curva de aprendizagem não é viável.

6. REFERÊNCIAS

Alterdata Software. **Folder de Automação Imobiliária (Immobile)**. Disponível em: <http://www.alterdata.com.br/imagens/folders/Alterdata_IMMOBILE_2009_web.pdf>. Acesso em: 13 jun 2011, 10:30:00.

Araujo, Luis César G. de, **Organização, sistemas e métodos e as modernas ferramentas de gestão organizacional: arquitetura, benchmarking, empowerment, gestão pela qualidade total, reengenharia**. 1. ed. - São Paulo: Editora Atlas, 2001. 311 p.

Base Software. **Página eletrônica do programa Base Condomínio**. Disponível em: <http://www.basesoft.com.br/base-condominio/?gclid=CM6C5N7xsqkCFQrt7QodxEk__w>. Acesso em: 13 jun 2011, 10:30:00.

Best Software. **Página eletrônica do programa Condomínio**. Disponível em: <http://www.bestsoftware.com.br/?DeptoId=276&ProdId=2114>. Acesso em: 13 jun 2011, 10:30:00.

BRCondomínio. **Página eletrônica do programa BRCondomínio**. Disponível em: <http://www.brcondominio.com.br/recursos.cfm>. Acesso em: 13 jun 2011, 10:30:00.

Farber, João Carlos e Segreti, João Bosco , **Análise da adequação das informações econômico-financeiras para a tomada de decisão nos condomínios residenciais da cidade de São Paulo através de uma pesquisa empírica**, 6º Congresso de Controladoria e Contabilidade – USP, 2006.

Fisher, Paul T. and Murphy, Brian D., **Spring Persistence with Hibernate**, 1. ed. - New York: Apress, 2009, 350 p.

Freeman, Eric et al., **Head First Design Patterns**, 1. ed. - Sebastopol: O'Reilly, 2004, 688 p.

Gamma, Erich et al., **Design Patterns: Elements of Reusable Object-Oriented Software**, 1. ed. - EUA: Addison-Wesley, c1995, 395 p.

Geary, David & Horstmann, Cay, **Core JavaServer Faces**, 3. ed. - New Jersey: Prentice Hall, 2010, 672 p.

Gersting, Judith L., **Fundamentos matemáticos para a Ciência da Computação**, 3. ed. - Rio de Janeiro: LTC, 1995, 538 p.

Group Software. **Página eletrônica do programa Condomínio21**. Disponível em: <http://www. condominio21.com.br/produtos/visao.aspx?id=7>. Acesso em: 13 jun 2011, 10:30:00.

Murach, Joel and Harris, Ray, **Murach's PHP and MySQL**, 1. ed. - Fresno: Mike Murach & Associates, Inc., 2010, 840 p.

Project Management Institute, Chapters Brasileiros. **Estudo de Benchmarking em Gerenciamento de Projetos, 2008.** Disponível em: <pmi-rio.ning.com/page/benchmarking-1>. Acesso em: 13 jun 2011, 14:00:00.

Rede Globo – Jornalismo, **PEGN: Administração de condomínio cresce com alta da venda imobiliária**. Disponível em: <http://redeglobo.globo.com/novidades/jornalismo/noticia/2011/04/pegn-administracao-de-condominio-cresce-com-alta-da-venda-imobiliaria.html>. Acesso em: 13 jun 2011, 17:30:00.

Silva, E. L. da e Menezes, E. M., **Metodologia da Pesquisa e Elaboração de Dissertação**. 4. ed. rev. atual. - Florianópolis: UFSC, 2005. 138 p.

Superlogica. **Página eletrônica do programa Condor**. Disponível em: <http://superlogica.com/conheca/>. Acesso em: 13 jun 2011, 10:30:00.

7. APÊNDICE

No texto a seguir, alguns comentários foram acrescentados para detalhamentos técnicos sobre o desenvolvimento. Apesar da intenção, o excessivo detalhamento acabaria deixando o texto longo e pouco explicativo. Visou-se, portanto, o meio termo.

7.1. Cadastro de usuários

No Eclipse, criou-se uma classe *UsuarioBean.java*, que é acessada dinamicamente por páginas *web*. Para viabilizar esse acesso, utilizou-se o mapeamento via *annotations*, abstendo-se, portanto, do uso do arquivo de configuração *faces-config.xml*.

A seguir, criou-se uma página que contem: um formulário de cadastro (*usuario.xhtml,* vide Figura 10), uma que exibe os usuários já cadastrados (*mostraUsuario.xhtml*) e outra que faz o papel de página inicial de cadastro de usuários (*indexCadastro.xhtml*).

A chamada navegação implícita simplifica o mapeamento de navegação entre as páginas. Isso quer dizer, por exemplo, que ao clicar no *link* salvar da página *usuario.xhtml*, o método correspondente na classe *UsuarioBean.java* retornará uma *string mostraUsuario*, que será interpretada como o nome da próxima página a ser acessada. A navegação implícita tem comportamento dependente de página de origem e de destino. A Tabela 8 resume os acessos resultantes em função da configuração origem e destino.

Tabela 8: Navegação implícita. Acesso resultante, em função da origem e do destino.

Origem	Destino	Acesso resultante
/publico/usuario.xhtml	mostraUsuario	/publico/mostraUsuario.xhtml
/publico/usuario.xhtml	/mostraUsuario	/mostraUsuario.xhtml
/publico/usuario.xhtml	/restrito/pets	/restrito/pets.xhtml
/publico/usuario.xhtml	perfil.pdf	/publico/perfil.pdf
/publico/usuario.xhtml	/modelos/perfil.pdf	/modelos/perfil.pdf

Fonte: Adaptado de Luckow e Melo (2010, p. 100).

Visando aumentar o desacoplamento da camada de persistência, criaram-se duas classes que representam o padrão de projetos *Data Access Object – DAO*. Essas duas classes são instanciadas a partir de uma classe *DAOFactory.java* e sua descrição é feita na sequência.

A primeira classe de interface, *UsuarioDAO.java*, contem somente as assinaturas dos métodos correspondentes às ações de carregar, listar, atualizar, salvar, excluir e buscar por *login*. Essa classe contem apenas as chamadas das funcionalidades utilizadas no acesso ao banco de dados.

```java
public interface UsuarioDAO {

    public void salvar(Usuario usuario);

    public void atualizar(Usuario usuario);

    public void excluir(Usuario usuario);

    public Usuario carregar(Integer codigo);

    public Usuario buscarPorLogin(String login);

    public List<Usuario> listar();
}
```

Figura 19: Trecho de código correspondente à classe de interface UsuarioDAO.java.
Fonte: arquivos pessoais do autor.

Uma segunda classe (de interface) é implementada através da classe *UsuarioDAOHibernate.java*, cujo nome sugestivamente faz lembrar o *framework* Hibernate. Essa classe implementa o acesso ao banco de dados propriamente dito.

Poderia ser modificado para qualquer outro SGBD, sem grandes problemas, o que evidencia o baixo acoplamento do padrão DAO.

```java
public class UsuarioDAOHibernate implements UsuarioDAO {

    private Session session;

    public void setSession(Session session) {

    public void salvar(Usuario usuario) {

    public void atualizar(Usuario usuario) {

    public void excluir(Usuario usuario) {

    public Usuario carregar(Integer codigo) {

    public Usuario buscarPorLogin(String login) {
        String hql = "select u from Usuario u where u.login = :login";
        Query consulta = this.session.createQuery(hql);
        consulta.setString("login", login);

        //TODO mostrar primeiramente com o list e
        //depois apresentar o uniqueResult
        return (Usuario) consulta.uniqueResult();
    }

    @SuppressWarnings("unchecked")
    public List<Usuario> listar() {
        return this.session.createCriteria(Usuario.class).list();
    }
}
```

Figura 20: Trecho de código correspondente à classe UsuarioDAOHibernate.java.
Fonte: arquivos pessoais do autor.

A camada de regra de negócio é implementada pela classe *UsuarioRN.java* (vide Figura 21). Definem-se regras relacionadas ao funcionamento ou ao comportamento do

software propriamente dito. A título de exemplo, ao excluir um usuário do cadastro, deve-se fazê-lo considerando que a exclusão se estende aos projetos dos quais participava. Isso pode ser visualizado no método excluir da classe *UsuarioRN.java*.

```
public class UsuarioRN {

    private UsuarioDAO  usuarioDAO;

    public UsuarioRN() {

    public Usuario carregar(Integer codigo) {

    public Usuario buscarPorLogin(String login) {

    public void salvar(Usuario usuario) {

    public void excluir(Usuario usuario) {

        ProjetoRN prjRN = new ProjetoRN();
        prjRN.excluir(usuario);

        this.usuarioDAO.excluir(usuario);
    }

    public List<Usuario> listar() {
```

Figura 21: Trecho de código correspondente à classe UsuarioRN.java
Fonte: arquivos pessoais do autor.

Um recurso de atualização assíncrona chamado *Partial Page Rendering* (PPR) do PrimeFaces foi utilizado. É um dos mais frequentes nas páginas *web*. As premissas básicas para sua utilização são:

- Evitar ações de mouse para exibição de dados.
- Evitar tráfego intenso de dados redundantes.

Para implementar o recurso PPR do PrimeFaces, utilizam-se identificadores, tais como:

<h:form id = "edicao"> ... </h:form>
<h:form id = "exibicao"> ... </h:form>

E, em um botão de comando, pode-se usar um elemento como:

<p:commandButton update="edicao, exibicao" value="Salvar" action="#{projetoBean.salvar}"/>

As *tags* iniciadas por "p:" indicam a utilização de recursos AJAX no PrimeFaces. No exemplo acima, existem dois formulários, em que: inserem-se dados (área que se denominou "edicao") e em que, ao clicar o botão Salvar, mostram-se os dados gravados (área que se denominou "exibicao"). O atributo "update" determina quais formulários (áreas) serão recarregados e redesenhados na tela.

A Figura 10 demonstra a funcionalidade de edição, quando ao clicar no ícone lápis, a tela de edição aparece em primeiro plano. Os dados correspondentes àquele usuário são exibidos, permitindo a edição de conteúdo. Ao clicar em Salvar, o método dispara os métodos DAO que se encarregarão de gravarem em banco os dados atualizados.

7.2. Cadastro de atividades

A Figura 13 e a Figura 14 são telas de edição dos dados referentes a uma atividade e suas respectivas tarefas. Dessa forma, reutilizou-se de forma análoga todas as técnicas explanadas para o caso de uso referente ao cadastramento de usuários.

É somente através dessa ferramenta que o administrador poderá alocar uma pessoa a determinado projeto, como se pode verificar na Figura 14. Trata-se de uma regra de negócio, que se deve ao fato do síndico ser a única pessoa que poderá autorizar a participação de outros condôminos em algum tipo de projeto condominial.

Essa funcionalidade foi implementada com *Picklist* e *Overlay Panel* do *framework* PrimeFaces.

7.3. Agenda

A lupa da Figura 14 remete à agenda e seus dados servem para que as pessoas possam se coordenar e serem coordenadas. Dessa forma, estão visíveis a todos os usuários envolvidos com uma determinada atividade.

Cada atividade é inserida na agenda, como ponto de apoio visual para os apontamentos que serão reflexo do acompanhamento das atividades tanto pelo Síndico quanto pelos demais responsáveis.

A Agenda foi implementada com o componente *Scheduler* do *framework* PrimeFaces.