FACULDADES INTEGRADAS DE TAQUARA
CURSO DE SISTEMAS DE INFORMAÇÃO

SISTEMA DE APOIO Á DECISÃO PARA PLANEJAMENTO DE ESCALAS DE TRABALHO

RAMÃO ROBERTO CORSO

Taquara
2009

RAMÃO ROBERTO CORSO

SISTEMA DE APOIO À DECISÃO PARA PLANEJAMENTO DE ESCALAS DE TRABALHO

Trabalho de Conclusão apresentado ao curso de Sistemas de Informação das Faculdades Integradas de Taquara, como requisito parcial para obtenção de grau de Bacharel de Sistemas de Informação, sob orientação do Prof. Mestre em Ciência da Computação Eurico Jardim Antunes.

Taquara

2009

Dedico este trabalho àquelas pessoas que assim como eu, encaram a busca ao conhecimento como um desafio contínuo. Àquelas pessoas que após um dia estafante de trabalho, ainda encontram motivação para buscar nos estudos a qualificação pessoal. Àquelas pessoas que acreditam que o capital mais valioso de um ser humano é o intelectual.

AGRADECIMENTOS

Agradeço aos meus pais, Dirceu e Verani, por acreditarem que o estudo é fundamental na vida do ser humano, e mesmo no limite de seus esforços, sempre ofereceram as melhores condições possíveis para os seus filhos e netas. À minha irmã Dionéia, pelo companheirismo e força que sempre me deu, e pelo amor que tem pelas minhas filhas.

Agradeço aos meus colegas de trabalho, pela compreensão e ajuda que me deram quando solicitava uns minutos para estudar antes de uma prova. Aos professores do curso, pela paciência e pelo esforço com que dividiram seu conhecimento com os alunos, e aos meus colegas de aula, verdadeiros amigos que obtive ao longo do curso, sempre dispostos a ajudar nos momentos complicados e sempre incentivando nos momentos de dificuldade. Vocês tornaram estas noites na faculdade mais divertidas e prazerosas.

Um agradecimento muito especial a minha esposa Clausiana e as minhas filhas Natália e Isabela, que compreenderam a minha ausência nas noites de aula e a minha obrigatória dedicação aos trabalhos acadêmicos nos momentos em que eu estava em casa. Amo muito vocês, e prometo recompensá-las de agora em diante.

RESUMO

Mudanças econômicas, políticas, tecnológicas e sociais obrigam empresas e órgãos públicos, a além de oferecerem cada vez mais produtos ou serviços de qualidade, a reduzirem seus custos de produção, e dentre eles, o mais oneroso é o custo com pessoal. Uma das soluções entre as possibilidades existentes é a terceirização de alguns serviços. Desta forma, estabelece-se um crescimento ostensivo de empresas do terceiro setor, que prestam serviços nas mais diferentes áreas, e a um número grande de clientes, e a cada tipo de serviço executado em cada cliente, surge uma escala de pessoal. Sendo assim, surge um segmento importantíssimo, dentro das empresas do terceiro setor, que é a gerência das escalas de trabalho. O administrador das escalas necessita de um conhecimento tanto das necessidades de profissional para cada escala, quanto à disponibilidade para trabalho de cada prestador de serviço, bem como ter o cuidado de não ferir a questão legal/trabalhista do setor. Tendo como foco o problema anteriormente apresentado, este trabalho apresentará uma solução através da implementação de um sistema *web* para auxílio nas tomadas de decisões na montagem das escalas de trabalho com o objetivo de otimizar e reduzir os custos deste processo.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A concorrência cada vez mais acirrada entre as empresas, e a diminuição gradativa de recursos do estado, na esfera federal, estadual e municipal, obriga empresas e órgãos públicos a oferecerem cada vez mais produtos ou serviços de qualidade, reduzindo seus custos de produção. Dentre eles, o mais oneroso é o custo com pessoal.

Além do salário pago ao funcionário, as empresas arcam com custos dos encargos sociais e trabalhistas, como 13º salário, DSR[1], FGTS[2], e INSS[3], e, para obter um valor mais próximo da realidade, ainda deve-se acrescentar valores como o do Vale Transporte e médias para auxílio afastamento por doença ou acidente e indenização por aviso prévio. Segundo Zanluca (2007), uma empresa não optante pelo SIMPLES[4] e que paga seus salários baseados em valor/hora, tem um acréscimo de até 96,75% no valor pago por hora ao seu funcionário.

Já os órgãos públicos, em todas as esferas, foram obrigados a se adaptar a uma nova cultura gerencial na gestão de recursos públicos, através da Lei Complementar nº 101, chamada popularmente de Lei de Responsabilidade Fiscal (LRF). A LRF estabelece ações em que se previnam riscos e corrijam desvios capazes de afetar o equilíbrio das contas públicas. Algumas das resoluções da LRF, tratam de um dos maiores “gargalos” do dinheiro público, que são as despesas com pessoal.

> Art. 19. Para os fins do dispositivo do caput do art. 169 da Constituição, a despesa total com pessoal, em cada período de apuração e em cada ente da Federação, não poderá exceder os percentuais da receita corrente líquida, a seguir discriminados:
> I – União: 50% (cinquenta por cento).
> II – Estados: 60% (sessenta por cento).
> III – Municípios: 60% (sessenta por cento). (CASA CIVIL, 2000, p.5)

1 DSR: Descanso Semanal Remunerado
2 FGTS: Fundo de Garantia por Tempo de Serviço
3 INSS: Instituto Nacional do Seguro Social
4 SIMPLES: Sistema Integrado de Pagamento de Impostos e Contribuições das Microempresas e Empresas de Pequeno Porte

Estes fatores fizeram com que as empresas buscassem soluções para tornarem-se mais competitivas no mercado, e que os órgãos públicos se adequassem às novas resoluções fiscais, sem afetar a prestação de serviços essenciais.

Uma das melhores formas encontradas para estas adequações foi a terceirização de alguns serviços considerados de atividade meio. Porém, as empresas que terceirizam serviços encontram algumas dificuldades para selecionar profissionais em determinadas áreas onde existe maior concorrência de pessoal.

Analisando a metodologia de trabalho e as ferramentas utilizadas em uma destas empresas, encontrou-se uma solução através de um *software*, para o auxílio na seleção de pessoal adequado para prestação de mão de obra. Verificou-se também que no mercado não existe um *software* especializado para esta função.

Neste trabalho é apresentado o desenvolvimento de uma ferramenta que auxilie na busca e manutenção destes profissionais para escalas de trabalho.

Na seção 1 descreve-se a visão atual mercadológica, a contextualização do problema e a definição e justificativa de uma solução para o problema.

Na seção 2 apresenta-se a revisão bibliográfica contendo o cenário mercadológico, processos de desenvolvimento de *software*, metodologias de análise, arquiteturas de *software*, paradigmas de programação, tecnologias de modelagem e desenvolvimento de *software* e técnicas de testes.

A seção 3 contém todo o desenvolvimento da modelagem e da aplicação do sistema, justificando as metodologias e tecnologias utilizadas, e apresentando um plano de testes de *software*.

As conclusões obtidas no desenvolvimento do sistema são descritas na seção 4.

1.1 Problema

Além de prestar um serviço de qualidade, a empresa terceirizada deve garantir o fornecimento ininterrupto dos serviços de acordo com os períodos solicitados, ou seja, deve manter a integridade das suas escalas. É este um dos principais problemas a serem solucionados por estas empresas.

A terceirização de serviços de atividade meio[5], para que não gere vínculo, exige uma rotatividade dos prestadores de serviço por contratante, bem como o cuidado com o número de horas executadas semanalmente por trabalhador. Empresas que possuem um grande número de contratos e fornece diversas especialidades, possuem um grande número de escalas a serem geridas, e devem conter um grande quadro de profissionais para suprir as vagas de trabalho. Cada prestador de serviço pode trabalhar em diversos contratantes, e também oferecer seus serviços a diversas empresas de prestação de serviços, pois seus honorários correspondem somente a quantidade de horas trabalhadas.

Todos estes fatores criam grande transtorno ao administrador, pois ocorrem muitas “lacunas” nas escalas de trabalho, ocasionadas pela troca dos dias de trabalho entre os profissionais, pedidos de substituições nas escalas de trabalho, tanto por parte do prestador de serviço, quanto do contratante, e aditivos contratuais que aumentam a carga horária prestada de acordo com a necessidade. Além disso, o administrador deve estar atento a questão legal quanto à carga horária semanal.

No processo de análise visto mais à frente na sub-seção 3.2.1.2, verifica-se que os recursos utilizados pelos gestores não fornecem uma informação colaborativa que auxilie na gestão das escalas. Fica comprometida ao gestor a exigência de um conhecimento quase que pessoal das especialidades e horários de cada prestador de serviço, para que o processo de substituição seja mais rápido e menos oneroso.

1.2 Solução

A proposta apresentada é desenvolver um sistema *web* de apoio à decisão, aplicado a empresas de prestação de serviços. O *software* estará voltado à gestão de escalas e, de acordo com cada especialidade, disponibilizará as vagas de trabalho existentes, tanto para o próprio prestador de serviço, quanto para o usuário gestor das escalas. O sistema fornecerá informações ao administrador de escalas para tornar mais ágil o preenchimento e o fechamento mensal das escalas, agilizando assim a seleção de profissionais habilitados a cada vaga, diminuindo os

[5] Atividade meio: A atividade meio, na qual é permitida a terceirização, é aquela não representativa do objetivo da empresa, não fazendo parte portanto do processo produtivo e caracterizando um serviço necessário, mas não essencial.

riscos de que alguma lacuna não seja preenchida e controlando a carga horária semanal do prestador de serviço. O sistema também auxiliará o contratante de serviços a monitorar sua carga horária contratada, e agilizará o processo de informação de vagas abertas para os prestadores de serviço.

1.3 Justificativa

A ideia de implementar um sistema de apoio à decisão surgiu avaliando as necessidades visualizadas, e pesquisando na literatura a ferramenta que traria maior retorno.

Segundo Barbosa e Almeida (2002), os sistemas de apoio a decisão visam apoiar processos de decisão que possuam alguns problemas de estruturação. O sistema de apoio utiliza uma informação, que pode ser interna ou externa, trata-a, e informa qual é a melhor decisão a ser tomada, e o porquê.

Estes sistemas tem por arquitetura básica dados, modelo e usuário. Perante a tecnologia da informação, os dados tem proveniência interna e externa, e normalmente são armazenados em um Sistema Gerenciador de Banco de Dados (SGBD). Os modelos agem em conjunto com os dados para os mais variados tipos de análise. O sistema também deve fornecer interfaces amigáveis entre usuário e sistema.

As principais características de um Sistema de Apoio a Decisão são:

a) possibilidade de desenvolvimento rápido, com a participação ativa do usuário em todo o processo;
b) facilidade para incorporar novas ferramentas de apoio à decisão, novos aplicativos e novas informações;
c) “Flexibilidade na busca e manipulação de informações” (BURC, 1989 *apud* FALSARELLA E CHAVES, 2004 p. 4);
d) “Individualização e orientação para o usuário que toma as decisões [...]” (MITT, 1986 *apud* FALSARELLA E CHAVES, 2004 p. 4);
e) real pertinência ao processo de tomada de decisão;
f) “[...]facilidade para que o usuário o entenda, use e modifique de forma interativa” (AWAD, 1988 *apud* FALSARELLA E CHAVES, 2004 p. 4).

Esta ferramenta poderá abreviar consideravelmente o tempo em que o administrador de escalas leva para o suprimento de cada vaga aberta, além de

ajudar na manutenção do princípio legal da terceirização, e o mais importante, reduzir os custos operacionais do processo que são telefone, anúncios publicitários e pagamentos extras ou à vista para o fechamento destas escalas.

1.4 Objetivo geral

Desenvolver um sistema *web* de apoio à decisão, que auxilie o gestor de escalas no fechamento das vagas de trabalho em aberto, indicando o trabalhador mais apto a assumir cada vaga conforme sua especialidade e disponibilidade.

1.5 Objetivos específicos

Os objetivos específicos deste trabalho são:

a) criar uma base de dados que armazene dados usuais como nome, endereço, telefone, áreas de atuação ou especialidade e horários disponíveis para trabalho de todos os prestadores de serviço, como visto no item “b” da sub-seção 1.3;
b) criar uma base de dados para inserção e manutenção de todas as escalas de trabalho;
c) desenvolver uma *interface*[6] para que o prestador de serviço possa pessoalmente realizar seu cadastro completo (e mantê-lo atualizado), bem como visualizar as oportunidades disponíveis para trabalho, e que o contratante possa ter acesso às escalas, e ao número de horas contratadas, como sugere o ítem “c” da sub-seção 1.3;
d) desenvolver uma *interface* para o usuário administrador das escalas, com acesso total a todas as bases de dados, e que forneça todas as vagas abertas nas escalas, bem como um relatório com os possíveis candidatos a estas vagas, conforme sua disponibilidade de trabalho, conforme citam os ítem “d” e “e” da sub-seção 1.3;
e) implementar uma *interface* para impressões e alterações nas escalas, e que execute a contabilidade mensal do número de horas por contratante, especialidade e prestador de serviço;

[6] Interface: Uma ou mais ferramentas para o uso e movimentação de qualquer sistema de informações, seja ele material, seja ele virtual

f) implementar um processo de “mala-direta[7]” que informa através de *e-mail* as vagas em aberto aos médicos, e o interesse de determinada vaga ao gestor de escalas.

[7] Mala-Direta: É uma carta de promoção ou comunicação de uma empresa ou negócio para um grupo de pessoas.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta material literário sobre o cenário mercadológico das empresas prestadoras de serviços, sobre processos de desenvolvimento de *software*, metodologias de análise, arquiteturas e paradigmas de *software*, e também citará tecnologias utilizadas para análise e desenvolvimento de sistemas.

2.1 O cenário mercadológico

Nesta seção, descreve-se sobre o processo legal da terceirização, sobre empresas prestadores de serviços, o que são cooperativas e seus modelos. Também apresenta-se o método de trabalho da cooperativa de trabalho analisada para o desenvolvimento deste *software*.

2.1.1 Terceirização

Segundo o Ministério do Trabalho (2007), a terceirização é a relação criada entre uma empresa que presta serviços (prestadora), e outra que utiliza os serviços desta empresa (tomadora), e o empregado, vinculado à empresa prestadora de serviço. Assim, é contratada uma empresa de prestação de serviços, que intermedía a mão-de-obra e o tomador de serviços, excluindo a relação entre eles.

Porém, a terceirização só é possível de ser aplicada nas áreas da empresa definidas como atividade-meio. Segundo o SEBRAE (2007, p.4):

> Atividade meio é aquela que se presta a dar condições que uma empresa atinja seus objetivos sociais. Por exemplo: uma empresa que fabrica roupas (atividade-fim) necessita contratar uma outra empresa que lhe preste serviços de limpeza. A atividade de limpeza, no exemplo utilizado, se constitui em atividade-meio da confecção.

O Superior Tribunal de Justiça regulamenta a legalidade da contratação de empresas prestadoras de serviços:

> III - Não forma vínculo de emprego com o tomador a contratação de serviços de vigilância (Lei nº 7.102, de 20.06.1983) e de conservação e limpeza, bem como a de serviços especializados ligados à atividade-meio do tomador, desde que inexistente a pessoalidade e a subordinação direta. (TEIXEIRA, 2003, p. 3)

Não possuindo previsão legal, apenas jurisprudências e entendimentos doutrinários como o Enunciado 331 do Tribunal Superior do Trabalho, também deve estar ausente na terceirização o pressuposto da subordinação e da pessoalidade.

2.1.2 Empresas prestadoras de serviço

Segundo Fortes (2007), a prestação de serviços é toda espécie de atividade ou trabalho lícito, material ou imaterial, contratada mediante retribuição (CC, art.594), excluídas as relações de emprego e outros serviços regulados por legislação específica.

Estas atividades exercidas por profissionais liberais são diversificadas e heterogêneas, abrangendo tanto os serviços de tecnologia avançada ou ciência, com mão-de-obra qualificada, como consultoria técnica, quanto serviços executados por mão-de-obra com baixa qualificação, como serviços de vigilância e de limpeza.

O crescimento do número de empresas de mão-de-obra temporária, que prestam serviços à outras empresas, apresenta uma das maiores médias de pessoal empregado e, conforme o IBGE (2005), na sua última pesquisa realizada, o consumo intermediário da atividade responsável pela seleção, agenciamento e locação de mão-de-obra temporária ficou em 746,2% , percentual acima das demais atividades e da média do setor, que foi de 251,5%. Ainda conforme pesquisa do IBGE (2005), este segmento gerou R$46,6 bilhões de receita operacional líquida, representando 49,5% do total de R$94,1 bilhões estimados para prestação de serviços.

As vantagens das empresas que contratam de terceiros os serviços de atividade-meio, são:

a) não há vínculo empregatício com o prestador;
b) custo bem menor na contratação de uma empresa, em relação a contratação direta do trabalhador.

Em relação aos órgãos públicos, além do menor custo na contratação, pode-se observar que:

a) facilita a adequação do órgão as exigências da LRF[8], pois estes tipos de contrato não entram na folha de pagamento de pessoal.
b) a inexistência do vínculo empregatício entre tomador e prestador, inibe o relaxamento do trabalhador perante seu trabalho, causada pela estabilidade, princípio legal do regimento trabalhista do funcionário público. Assim, o tomador tem a liberdade de solicitar em qualquer momento a rescisão de contrato ou a substituição do prestador de serviço, caso este não o esteja executando adequadamente.

2.1.3 Cooperativas

Cooperativa é uma empresa que deve ser constituída e gerida democraticamente. O foco principal é o associado, e este deve ter total compreensão dos objetivos da cooperativa, seus direitos e suas responsabilidades enquanto sócio. A cooperativa é uma empresa, não uma entidade filantrópica[9].

> Conceito Teórico: A cooperativa é uma sociedade de pessoas de natureza civil, unidas pela cooperação e ajuda mútua, gerida de forma democrática e participativa, com objetivos econômicos e sociais comuns e cujos aspectos legais e doutrinários são distintos de outras sociedades. Fundamentam-se na economia solidária e se propõe a obter um desempenho eficiente através da qualidade e da confiabilidade dos serviços que presta a seus próprios associados e usuários.
> Conceito Legal: As cooperativas são sociedades de pessoas que naturalmente se obrigam a contribuir com bens e serviços para o exercício de uma atividade econômica, de proveito comum, sem objetivo de lucro (PERIUS, 1999, p.81 *apud* BERNARD, 2003, p. 17)

A Organização das Cooperativas Brasileiras (OCB) divide as cooperativas conforme a natureza de suas atividades, gerando 13(treze) segmentos:

a) cooperativas agropecuárias.
b) cooperativas de consumo;
c) cooperativas de crédito;
d) cooperativas educacionais;
e) cooperativas especiais;
f) cooperativas de habitação;

---

[8] LRF : Lei de Responsabilidade Fiscal
[9] Entidade Filantrópica: É uma entidade que presta serviços à sociedade, principalmente as pessoas mais carentes e que não possuem como finalidade a obtenção de lucro.

g) cooperativas de saúde;
h) cooperativas de serviço;
i) cooperativas de produção;
j) cooperativas de mineração;
k) cooperativas de turismo e lazer;
l) cooperativas de transporte de cargas e passageiros;
m) cooperativas de trabalho.

2.1.4 Cooperativas de trabalho

As cooperativas de trabalho tem sido uma alternativa ao desemprego, a insatisfação devido a baixa qualidade dos serviços oferecidos e aos baixos salários. Neste segmento de cooperativa, aos associados vislumbra-se a chance de obter sucesso num mercado altamente competitivo. Como resultado, na quase totalidade, produtos ou serviços oferecidos por estas associações tem maior qualidade, e até custo menor ao consumidor. Alguns exemplos de cooperativas desta natureza são: motoristas de táxi, promotores culturais, costureiras, médicos, dentistas, todos profissionais liberais. As palavras trabalho e cooperação são o princípio destas empresas, e muitas mantém fundos de assistência aos cooperados para ajuda financeira, no caso de que um cooperado fique impossibilitado de trabalhar, seja por doença ou acidente.

> Pelos dados, não há dúvidas de que as cooperativas de trabalho crescerão geométricamente nos próximos anos. Trata-se de um novo fenômeno frente a terceirização e o desemprego formal, pois procura corrigir as distorções de um sistema capitalista. A cooperativa de trabalho faz a verdadeira revolução que todo trabalhador deseja: ser um dia dono dos meios de produção, e isso as cooperativas devem perseguir desde o dia de seu surgimento. (PERIUS, 2001, p.210, *apud* BERNARD, 2003, p.35)

Como não existe um vínculo empregatício, pois os cooperados são os usuários e donos da empresa, as cooperativas não possuem alguns encargos sociais. Isso fez com pessoas de má índole vissem nestas cooperativas uma forma de se beneficiar. Com isso, muitos empregadores criaram cooperativas de fachada com o intuito exclusivo de não recolher os encargos sociais aos seus empregados. Esta prática é diariamente combatida pela OCB, que procura defender os princípios do cooperativismo.

2.2 Metodologia de desenvolvimento de *software*

A metodologia define os critérios escolhidos para uma pesquisa ou projeto. Em um projeto de desenvolvimento de *software*, uma metodologia escolhida fornecerá os métodos para planejar, projetar, analisar, executar e implantar o *software* a ser desenvolvido.

2.2.1 Processo de desenvolvimento de *software*

A utilização de um processo de *software* tem sido apontada como um fator primordial para o sucesso de empresas de desenvolvimento de *software*. Conforme Pressman (2002, p.24), um modelo de processo de *software* é uma estratégia de desenvolvimento que abrange camadas de processo, métodos e ferramentas utilizados para resolver problemas reais em instalações industriais. Este modelo deve ser escolhido avaliando a natureza do processo e da aplicação.

Todo o *software* é desenvolvido para solucionar um problema. Para isso, são listados quatro estágios distintos no processo, que são a situação atual encontrada, definição de um problema, desenvolvimento técnico e integração da solução.

2.2.1.1 Modelo em Cascata

Conforme cita Pressman (2006), o modelo em cascata sugere uma abordagem seqüencial e linear para o desenvolvimento de *software*, começando nas especificações de requisitos, progredindo ao longo do planejamento, modelagem, construção e implantação do sistema. Apesar de ser um dos modelos mais antigos de processos de *software*, este recebe algumas críticas, pois, dificilmente o cliente consegue explicitar diretamente todos os requisitos desejados, e este modelo dificulta interações ou modificações no andamento do processo. Entre os desenvolvedores também existem restrições, pois alguns membros da equipe de desenvolvimento ficam parados, até que os outros membros terminem as tarefas anteriores. O produto também só pode ser demonstrado ao final de todo o processo, e uma falha em algum requisito pode ser desastroso.

2.2.1.2 Modelo Incremental

"O modelo incremental combina elementos do modelo em cascata, aplicados de forma interativa" (PRESSMAN, 2006 p. 40). O modelo incremental aplica sequências lineares conforme o tempo passa. É interativo por natureza, e a cada sequência produz-se versões simplificadas do projeto, que depois serão integradas formando o produto final. A cada versão fica possível que o usuário avalie o produto.

2.2.1.3 Modelo RAD:

O RAD *(rapid application development),* é um modelo de processo de desenvolvimento de *software* incremental que enfatiza um ciclo de desenvolvimento extremamente curto. (PRESSMAN 2006, p.41). O modelo RAD tem sua concepção baseada em componentes, e é utilizado quando um projeto é restrito, sem ocorrências de alterações, e quando os requisitos são bem claros e objetivos.

Quando uma aplicação pode ser modularizada, desde que, observando-se as limitações de tempo, onde cada função deva ser completada em até 3 meses, esta é uma candidata ao modelo RAD.

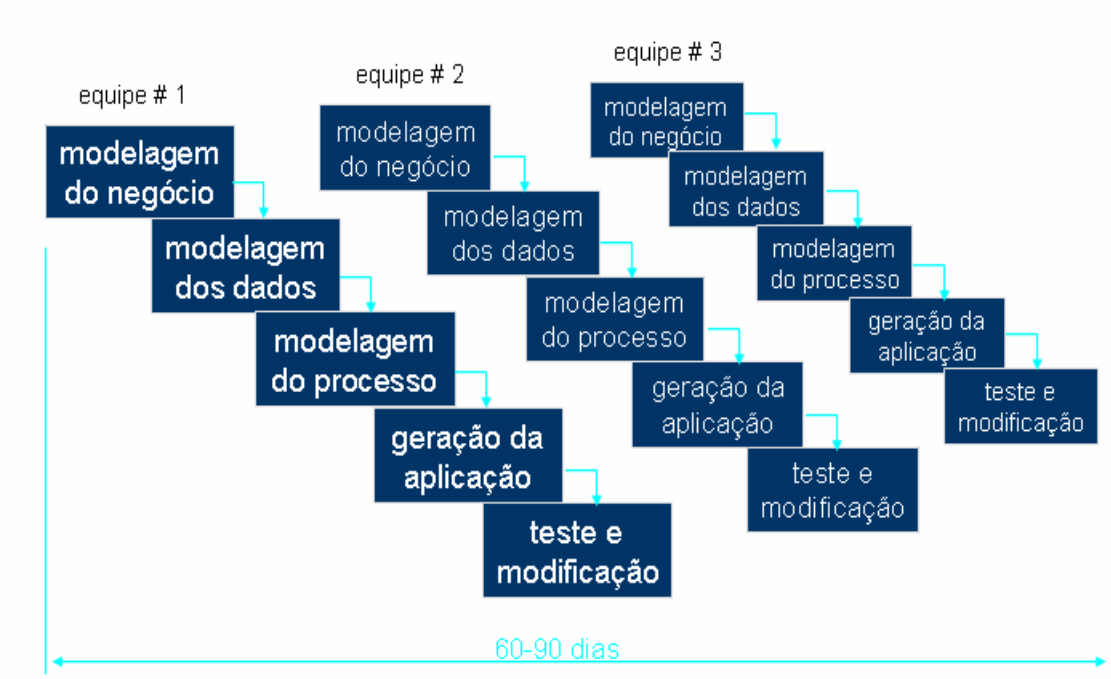


Figura 1 - Divisão de atividades no modelo RAD

Fonte: Ranzi, Silva e Ribeiro (2005, p.03).

O método RAD é mais apropriado em sistemas de informação do tipo *stand alone*[10], e é dividido em cinco fases na seguinte sequência:

a) modelagem de negócio: É utilizado para gerar informações do negócio, escopo, atores, requisitos, e define para onde vão todas estas informações.
b) modelagem de dados: Através do fluxo de informações obtidas na fase anterior, gera-se um grupo de objetos de dados, e nesta modelagem são definidas as características destes dados e seus relacionamentos.
c) modelagem do processo: É utilizada para implementar uma função do negócio, onde o processamento para adicionar, modificar ou descartar um objeto é descrito.
d) geração da aplicação: São definidas as técnicas e linguagens de programação que dão suporte para o reuso de componentes, ou criação de componentes reusáveis, e o código fonte é gerado.
e) teste e entrega: Teste de componentes e interfaces.

Algumas das vantagens de utilização do RAD são:

a) desenvolvimento e resultados (protótipos) mais rápidos;
b) ciclo de desenvolvimento extremamente curto (entre 60 e 90 dias);
c) criação e reutilização de componentes;
d) desenvolvimento é conduzido em um nível mais alto de abstração;
e) maior flexibilidade (desenvolvedores podem reprojetar praticamente a vontade);
f) envolvimento maior do usuário.

2.2.1.4 Modelos evolucionários.

Conforme Pressman (2006), os modelos evolucionários são iterativos e caracterizados de forma que o engenheiro de *software* consiga desenvolver cada vez mais versões mais complexas de *software*.

Dentre os modelos evolucionários, cita-se:

a) Modelos de prototipação: Este processo possibilita que o desenvolvedor crie um modelo do *software* que deva ser construído. Idealmente, o

[10] *Stand alone*: programas completamente autosuficientes, para seu funcionamento não necessitam de um software auxiliar, como um interpretador, sob o qual terão de ser executados.

modelo (protótipo) serve como um mecanismo para identificar os requisitos de *software*. Este modelo é apropriado para quando o cliente definiu um conjunto de objetivos gerais para o *software*, mas não identificou requisitos de entrada, processamento e saída com detalhes. Ainda que possam ocorrer problemas, a prototipação é um ciclo de vida eficiente. A chave é definirem-se as regras do jogo logo no começo, onde o cliente e o desenvolvedor devem concordar que o protótipo seja construído para servir como um mecanismo a fim de definir os requisitos.

b) Espiral: O modelo espiral é um modelo evolucionário de processo de *software* que combina interativa da prototipagem com os aspectos controlados e sistemáticos do modelo em cascata. (PRESSMAN, 2006 p.44). É atualmente, a abordagem mais realística para o desenvolvimento de *software* em grande escala, usando uma abordagem que capacita o desenvolvedor e o cliente entender e reagir aos riscos em cada etapa evolutiva, apesar de poder ser difícil convencer os clientes que uma abordagem "evolutiva" é controlável. O modelo espiral exige considerável experiência na determinação de riscos e depende dessa experiência para ter sucesso.

2.2.1.5 Modelos ágeis

A modelagem ágil prega que, desenvolvedores desvinculem-se do vício da produção de uma documentação perfeita, e prendam-se mais ao funcionamento do sistema e a satisfação das necessidades do negócio, com grande colaboração do cliente. (PRESSMAN, 2006, p.61). Alguns modelos ágeis de processo de *software* criados são a *Extreme Programming* (XP), a DSDM (Método de Desenvolvimento Dinâmico de Sistemas), o Scrum, a Crystal e o FDD (Desenvolvimento Guiado por Características).

2.3 Metodologia de análise de *software*

Identificar os problemas envolvidos nos processos, apresentar soluções e avaliar sua viabilidade é vital na análise de um sistema. Após a delimitação do

modelo de sistema a ser concebido e delimitar seu escopo, deve-se definir os requisitos do *software*.

Uma análise correta dos requisitos elucida as atribuições do *software*, e é através da lista de requisitos que são construídos os modelos de domínios de dados, como o *software* deverá funcionar e se comportar. Esta análise também tem por função auxiliar na montagem da interface, da arquitetura, e o nível de componentes do *software*, além de fornecer ao cliente e ao desenvolvedor, formas de avaliar a qualidade do *software* enquanto este é construído.

2.3.1 *O que é um requisito*

Segundo Kruchten (2003, p.132), “Definimos um requisito como uma condição ou capacidade para a qual um sistema tem que se conformar”.

As diversas propriedades de um sistema, sejam elas objetivos ou restrições estabelecidas em conjunto entre clientes e usuários, são os requisitos. Um conjunto de requisitos são os processos necessários implementados no *software* para que um usuário atinja um objetivo, ou que atenda as solicitações ou até restrições da organização, ou outros componentes do sistema.

Os requisitos de *software* são classificados em requisitos funcionais e não funcionais.

2.3.1.1 Requisitos funcionais

Requisitos funcionais descrevem as várias necessidades e funções que cliente e usuário definem como indispensáveis no *software*. Estes requisitos definem as operações que devem ser realizadas pelo sistema, sejam através de comandos do usuário, ou ativadas por eventos internos ou externos ao sistema.

Macoratti (2009, p.1), dá-se exemplos de requisitos funcionais num hipotético sistema:

> ‘ -o software deve possibilitar o cálculo dos gastos diários, semanais, mensais e anuais com pessoal.
> -o software deve emitir relatórios de compras a cada quinze dias
> -os usuários devem poder obter o número de aprovações, reprovações e trancamentos em todas as disciplinas por um determinado período de tempo. ’

Um requisito funcional deve determinar o que o *software* deve fazer, não importando como fazer.

2.3.1.2 Requisitos não funcionais

Os requisitos não funcionais são características implícitas que são esperadas em todos os *softwares* desenvolvidos profissionalmente, e definem propriedades de um *software* como a usuabilidade, confiabilidade, portabilidade, facilidade de manutenção, desempenho esperado, entre outros.

Novamente Macoratti (2009, p.1) cita exemplos hipotéticos de requisitos não funcionais:

> ‘ -a base de dados deve ser protegida para acesso apenas de usuários autorizados.
> -o tempo de resposta do sistema não deve ultrapassar 30 segundo.
> -o software deve ser operacionalizado no sistema Linux.
> -o tempo de desenvolvimento não deve ultrapassar seis meses. ‘

2.3.2 Técnicas para levantamentos de requisitos

Algumas técnicas foram criadas para levantamento e análise de requisitos, de acordo com o cenário encontrado. Dentre as técnicas mais utilizadas para levantamento de requisitos, são citadas:

2.3.2.1 Entrevistas

Uma entrevista de levantamento de informações é uma conversa direcionada com um propósito específico, que utiliza um formato pergunta-resposta. Os objetivos de uma entrevista são obter as opiniões do entrevistado que ajudem na descoberta dos problemas-chave a serem tratados, além de conhecer os sentimentos do entrevistado sobre o estado corrente do sistema, obtendo metas organizacionais e pessoais e levantando procedimentos informais. (KENDALL, 1992, *apud* FALBO 2006, p.8)

2.3.2.2 Questionários

O uso de questionários constitui uma técnica de levantamento de informações que permite ao engenheiro de *software* obter informações como postura, crenças, comportamento e características de várias pessoas afetadas pelo sistema (corrente ou proposto) (KENDALL, 1992, *apud* FALBO 2006, p.13).

2.3.2.3 Brainstorming

É uma ferramenta para geração de novas ideias, conceitos e soluções para qualquer assunto ou tópico num ambiente livre de críticas e de restrições a imaginação. É útil quando de deseja gerar em curto prazo uma grande quantidade de idéias sobre um assunto a ser resolvido, possíveis causas de um problema, abordagens a serem usadas, ou ações a serem tomadas.(SIQUEIRA, 2009, p.1).

2.3.2.4 FAST – Facilitated application specification techniques

A técnica FAST estimula a criação de uma equipe conjunta de clientes e desenvolvedores e segue diretrizes como encontros em locais neutros, regras para participação, agenda formal para cobrir os pontos importantes, e informal para encorajar fluxo de idéias, utilizar um moderador e criar mecanismos de definição. (MELO, 2007, p.17).

2.3.2.5 Técnicas de observação direta

Reis (2007, p. 52), assim definiu as técnicas de observação.

> Na utilização desta técnica, o engenheiro pode analisar o ambiente sem necessidade de interrupção do trabalho do especialista, proporcionando uma melhor conotação da complexidade do problema a ser resolvido, identificando as estratégias inconscientes, bem como as habilidades motoras e procedimentos automáticos.

O engenheiro entra no ambiente do processo, e observa as tarefas sendo executadas pelos especialistas, analisando as limitações que as restringem . Com esta técnica, é possível esclarecer como a tarefa está sendo executada no próprio

ambiente. Também é possível a coleta de documentos para esclarecimentos posteriores, sempre com aprovação da direção da empresa.

Antes da análise, o engenheiro deve identificar as áreas e usuários a serem observados, obter os nomes das pessoas chave do estudo e explicar a sua finalidade, sempre com a aprovação da direção da empresa.

Durante a observação, o engenheiro deve familializar-se com o ambiente e agrupamentos organizacionais existentes, visualizar os processos manuais e automatizados existentes, levantar amostras de documentos e gerar documentos escritos de cada processo específico observado e coletar informações quanto a frequência das tarefas e tempo médio de duração de cada tarefa.

Após a observação, as descobertas feitas são documentadas e revistas com as pessoas observadas e a direção da empresa.

2.3.2.6 Prototipação

A prototipação é uma técnica valiosa para se obter rapidamente informações específicas sobre requisitos de informação do usuário, verificando as reações iniciais do usuário em relação ao sistema, sugestões de refinamento para a eficácia nos requisitos, inovações propostas e definição de novas prioridades. (KENDALL, 1992, *apud* FALBO 2006, p.20).

2.4 Arquitetura do *software*

Conforme literatura, “[...] arquitetura de *software* define o que é o sistema em termos de componentes computacionais e os relacionamentos entre estes componentes.” (VAROTO, 2002, *apud* ROCHA, 2008, p.2). Estes relacionamentos geram uma abstração do sistema, suprimindo detalhes de componentes não afetando sua usuabilidade, auxiliando o gerenciamento da complexibilidade.

Dentre alguns modelos de arquiteturas pode-se citar:

2.4.1 P2P

É uma arquitetura de sistemas distribuídos caracterizada pela descentralização das funções na rede, onde cada nodo realiza tanto funções de servidor quanto de cliente. (SILVA, 2003, p.1).

2.4.2 Cliente-Servidor

Esta arquitetura é uma das mais adaptadas a sistemas *web*, dividindo-se em duas partes claramente diferenciadas, onde a primeira é a parte do servidor[11] e a segunda a de um conjunto de clientes. Normalmente o servidor é uma máquina bastante potente que atua como depósito de dados e funciona como um sistema gerenciador de banco de dados (SGBD). Por outro lado, os clientes costumam ser estações de trabalho que solicitam vários serviços ao servidor. Ambas as partes devem estar conectadas entre si mediante uma rede.

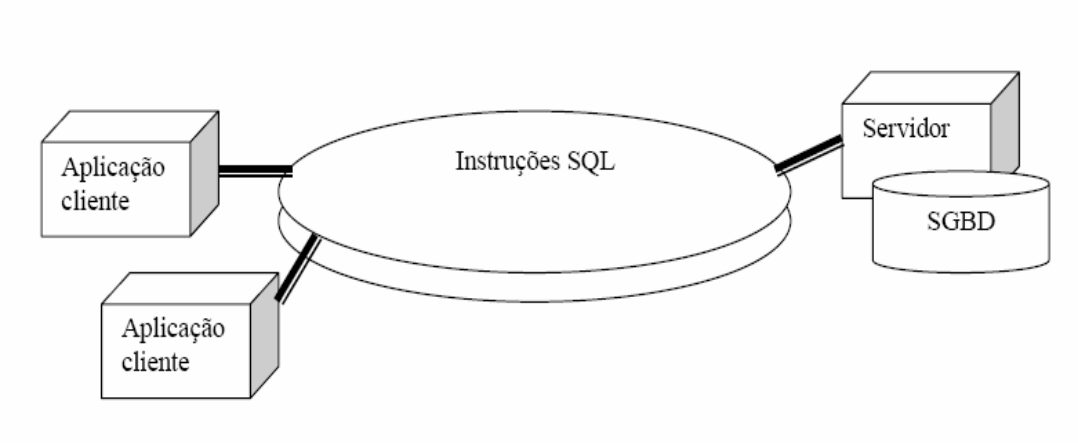


Figura 2 - Modelo Cliente-Servidor de base de dados
Fonte: Bragança (1999, p.05).

O funcionamento básico de uma arquitetura cliente-servidor é: O cliente emite um pedido de serviço específico para o servidor graças ao seu endereço IP[12]. O servidor recebe o pedido e responde à solicitação do endereço da máquina cliente.

[11] Servidor: é um sistema de computação que fornece serviços a uma rede de computadores.
[12] Endereço IP (*Internet Protocol*): um conjunto de números que representa o local de um determinado *equipamento* (normalmente computadores) em uma rede privada ou pública.

Vantagens da arquitetura:

a) mais segura, pois o número de pontos de entrada para acesso de dados é o mais importante;
b) o sistema possui uma rede evolutiva, possibilitando acrescentar ou reduzir clientes sem modificações que afetariam o funcionamento da rede;
c) a administração fica à nível de servidor;
d) utiliza-se uma base de dados centralizada, facilitando a gestão de recursos que são comuns a todos os clientes, evitando problemas de redundância ou contradição.

2.4.3 Arquitetura orientada a serviço

É um estilo de arquitetura de software cujo princípio fundamental preconiza que as funcionalidades implementadas pelas aplicações devem ser disponibilizadas na forma de serviços. Frequentemente estes serviços são organizados através de um "barramento de serviços" que disponibiliza *interfaces*, ou contratos, acessíveis através de *web services* ou outra forma de comunicação entre aplicações. (PUC, 2007, p.18).

2.5 Paradigmas[13]

Segundo Heidrich (2005), um paradigma impõe a forma como um desenvolvedor de programas analisa os dados, ou seja, determina a maneira como o mesmo enxerga o problema. Um paradigma define um modelo de análise, projeto e programação de um sistema de *software*.

Os paradigmas distinguem-se pelos modelos de análise e pelas técnicas de programação. Conforme o problema encontrado no mundo real, escolhe-se um paradigma para resolvê-lo de forma computacional.

Existem vários tipos de paradigmas de programação, porém, serão citados os tipos mais utilizados atualmente.

[13] Paradigmas: É a representação de um padrão a ser seguido

2.5.1 Programação imperativa

A programação imperativa define que a manipulação das variáveis é feita através de comandos, onde determinadas variáveis são lidas na entrada, manipuladas e salvas em outras variáveis de saída. Estes comandos são executados de forma sequencial.

O paradigma imperativo ainda é um dos mais usados, pela fácil compreensão e também pela facilidade em migrar para outros paradigmas, porém, com desvantagem, “Possui relacionamento indireto com a entrada/saída, resultando numa difícil legibilidade do programa, erros introduzidos na manutenção do código, ocasionados pela difícil legibilidade.” (HEIDRICH, 2005, p.1)

2.5.2 Programação estruturada

A programação estruturada define que todos os programas devem conter 3 (três) estruturas: sequência, decisão e iteração, orientando os programadores a utilizarem estruturas simples em seus programas.

Sequência implementa passos de programação necessários em qualquer programa, onde uma tarefa é executada após a outra. Seleção especifica a possibilidade de selecionar o fluxo de execução do processamento baseado em ocorrências lógicas, utilizando condições como *If* e *Switch*. Iteração permite a execução repetitiva de segmentos do programa, utilizando programas como o *While.*

A programação estruturada ainda é marcantemente influente, uma vez que grande parte das pessoas ainda aprendem programação através dela.

2.5.3 Programação Orientada a Objetos

A principal característica deste paradigma é a manipulação de objetos que interagem entre si. Este paradigma possui componentes básicos como classes, objetos e mensagens.

Os objetos do sistema são definidos através da implementação de um conjunto de classes. O comportamento e o estado dos objetos são definidos nas

classes através de métodos[14] e atributos[15], e o relacionamento entre eles é definido através de herança[16], encapsulamento[17], abstração[18], polimorfismo[19] e associação[20] de dados.

Este paradigma tem grande aceitação no mercado. A orientação a objetos possui todas as vantagens do paradigma imperativo, além da modularização[21] do programa e de uma grande facilidade de reusabilidade[22], podendo ser utilizados partes dele em outros programas. Também possui conceitos altamente sofisticados como polimorfismo, funções de alta ordem e avaliação sob demanda.

2.6 Tecnologias aplicadas no desenvolvimento

Neste capítulo são apresentadas tecnologias aplicadas para o desenvolvimento de um sistema *web*.

2.6.1 Linguagens de programação *web*

São linguagens de programação que permitem a codificação de sistemas que possam ser acessados e utilizados através do protocolo http.

2.6.1.1 HTML

Segundo a UFPA (2007, p. 3), HTML (*Hyper Text Markup Language),* significa Linguagem de Marcação de Hipertexto. É utilizada para criar arquivos (páginas), que podem ser visualizados na *world wide web* (www), e permite fazer ligações (*links*[23]), os quais possibilitam a navegação virtual.

---

[14] Métodos: Definem as habilidades de um objeto.
[15] Atributos: São as características de um objeto.
[16] Herança: É o mecanismo pelo qual uma classe (sub-classe) pode estender outra classe (super-classe), aproveitando seus comportamentos (métodos) e variáveis possíveis
[17] Encapsulamento: Separação de aspectos internos e externos de um objeto
[18] Abstração: É a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais.
[19] Poliformsmo: É a capacidade de um método poder ser implementado de diferentes formas, ou mesmo de realizar coisas diferentes.
[20] Associação: É o mecanismo pelo qual um objeto utiliza os recursos de outro.
[21] Modularização: o programa pode ser dividido em vários subprogramas.
[22] Reusabilidade: Reaproveitamento do código para outros métodos ou softwares.
[23] *Links* -palavra, texto, expressão ou imagem que permite o acesso imediato à outra parte de um mesmo, ou outro documento ou site

Por ser descritivo, o HTML permite a criação e transmissão de documentos que podem ser lidos em qualquer tipo de computador, desde que este contenha um navegador instalado (Internet Explorer, Mozilla, Netscape). Estes documentos podem conter qualquer tipo de informação, como vídeos, fotos, textos, áudio e até outros programas. Porém, por ser descritiva, conforme a formatação, nem sempre navegadores diferentes exibem a mesma apresentação em cada página. Caso um navegador não entenda um determinado comando, isto afetará minimamente a página, pois o comando é ignorado, e não será enviada mensagem de erro.

O HTML tem o papel de apenas informar ao navegador o que são, qual a localização e o tipo dos arquivos contidos nas páginas, deixando a critério de cada navegador a aparência dos documentos. As descrições dos elementos são feitas por intermédio de comandos chamados *tags*[24], que informam ao navegador como exibir o documento.

A popularização do HTML tornou-o quase um padrão de linguagem *web*, e é utilizado na quase totalidade de navegadores e editores entre outros.

2.6.1.2 PHP

Conforme a UNIBAN (2006, p. 4):

> PHP, que significa "*Hypertext Preprocessor*", é uma linguagem de programação (de *script*[25] *Open Source* de uso geral), interpretada, mais utilizada para desenvolvimento *web* e pode ser inserida dentro do código HTML, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e *links*.

Ainda segundo a UNIBAN (2006), o cliente recebe apenas o HTML puro em sua máquina, enquanto o código do PHP é executado no servidor. Desta forma, quando o cliente lida com senhas ou outras informações que necessitem de segurança, estas informações interagem com o banco de dados sem serem expostas. O objetivo principal do PHP é permitir que os desenvolvedores escrevam páginas dinâmicas e rápidas.

Para um melhor entendimento sobre o funcionamento do PHP, é necessário o esclarecimento das diferenças entre *Server Side Scipts* e *Client Side Scripts*:

---

[24] *Tags* - A *tag* é um trecho de código que é gerado pelo sistema, contendo o código identificador de cada afiliado

25 *Script* - seqüência de comandos inseridos em determinadas posições de uma página HTML

a) <u>*Server Side Scripts*</u>: Os *scripts server-side* são responsáveis pela criação de páginas em tempo real. Num mecanismo de busca, por exemplo, seria inviável manter um arquivo para cada consulta a ser realizada. O que existe é um modelo da página de resposta, que é mesclado com os dados no momento em que a página é requisitada.

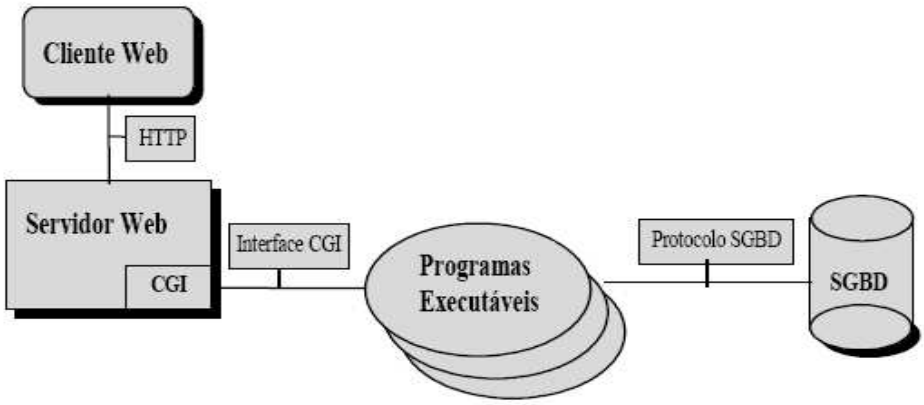


Figura 3 - *Server Side Scripts*
Fonte: Desenvolvimento Fácil (2007, p.3).

b) <u>*Client Side Scripts*</u>: São responsáveis pelas ações executadas no *browser*[26], sem contato com o servidor. Os exemplos mais comuns de aplicações client-side são imagens e textos que mudam com o passar do mouse.

Os *scripts client-side* são muito úteis para fazer validações de formulários sem utilizar processamento do servidor e sem provocar tráfego na rede. Outra utilização comum é na construção de interfaces dinâmicas e "leves".

[26] Browser - É um programa de computador que habilita seus usuários a interagirem com documentos virtuais da Internet

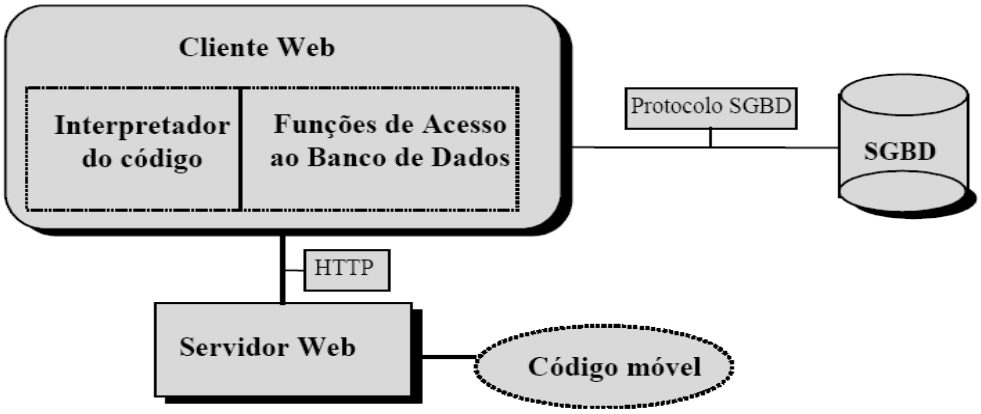


Figura 4 - *Client Server Side*
Fonte: Desenvolvimento Fácil (2007, p.3).

Avaliando as figuras 1 e 2, nota-se que a grande vantagem do *server side scripts*, caso do PHP, é que o lado cliente não tem acesso ao código da programação, pois este é interpretado diretamente no servidor, servindo ao cliente somente as respostas.

Além de ser gratuita, e possuir uma grande estabilidade e performance, a linguagem PHP será utilizada também por ser multi-plataforma, aceitando vários sistemas operacionais como o Windows, Unix e Linux, e permitindo a conexão direta com uma grande quantidade de Banco de Dados relacionais como: Oracle, Sybase, Informix, MySQL e outros disponibilizado através do ODBC. É suportado pela maioria dos servidores *web* que existem no mercado como o Apache, IIS e PWS.

2.6.1.3 JSP

JSP (*Java Server Pages*), é uma tecnologia utilizada para desenvolvimento de sistemas *web*, desenvolvida pela *Sun Microsystems*. É uma linguagem fundamentada na arquitetura SSI (*Server Side Includes*), que pode conter conteúdo estático e dinâmico. (SANTOS e JORGE, 2008). Os comandos dinâmicos são processados pelo servidor *web* antes de serem enviados ao usuário, que recebe os resultados destes comandos em formato HTML. Esta tecnologia permite ao programador *web* desenvolver aplicações com acesso a banco de dados, manipulações de arquivos texto e captura de informações através de formulários.

Por ser baseada na linguagem Java, tem a facilidade da portabilidade de plataformas. Porém, para desenvolver aplicações em JSP, é necessário um conhecimento sobre a especificação J2EE[27].

2.6.1.4 JavaScript

Conforme a Infowester (2007, p. 2) “JavaScript é uma linguagem para páginas da *web*, desenvolvida pela Netscape. Com essa linguagem, é possível adicionar recursos dinâmicos (*scripts*) às páginas HTML [...]”. É oferecido ao usuário a possibilidade de interação com a página, seja respondendo a questionários ou acessando qualquer conteúdo randômico[28]. O JavaScript permite a criação de várias aplicações para páginas *web*. A linguagem JavaScript atua inserida no meio do código HTML, e pode estar numa área determinada ou inserida em vários pontos da página.

Apesar do nome e sintaxe semelhante, JavaScript não tem relação com a linguagem Java. Java, é desenvolvida pela *Sun Microsystems*, é orientada a objetos e deve ser compilada para gerar um código intermediário independente de plataforma. Já o JavaScript é uma linguagem interpretada, ou seja, o código é incluído dentro do próprio arquivo HTML, e quando encontrado pelo navegador, é interpretado e depois executado.

2.6.2 Sistema Gerenciador de banco de dados

> Um Sistema Gerenciador de Banco de Dados, é o conjunto de programas (softwares) que compõe a camada responsável pelo armazenamento, e recuperação dos dados no Sistema de Informação. O objetivo principal é retirar da camada da Aplicação a responsabilidade dessas tarefas provendo um ambiente mais seguro, mais fácil de manter-se e mais confiável. A interface entre essas duas camadas é a uma linguagem padrão para consulta, manipulação e controle de acesso aos dados. Atualmente a linguagem mais utilizada para essa interface é o SQL - *Structured Query Language* (CDTC, 2006, p. 19).

[27] J2EE – Conjunto de padrões e especificações responsável por receber as requisições do cliente, entendê-las e direcioná-las aos responsáveis pelas respostas.
[28] Randômico – aleatório.

O SGBD disponibiliza uma interface para que os seus clientes possam incluir, alterar ou consultar dados. Em bancos de dados relacionais[29] a *interface* é constituída pelas APIs ou *drivers* do SGBD, que executam comandos na linguagem SQL. O SQL é uma linguagem normatizada para comunicação com a base de dados, que permite trabalhar com qualquer tipo de linguagem de programação, em combinação com qualquer tipo de base de dados.

2.6.2.1 Banco de dados MySQL

Segundo o Centro de Difusão de Tecnologia e Conhecimento (2006, p. 13)

> O Programa MySQL é um sistema de gerenciamento de banco de dados relacionais baseado em comandos SQL (*Structured Query Language* - Linguagem Estruturada para Pesquisas) que vem ganhando grande popularidade sendo atualmente um dos bancos de dados mais populares, com mais de 4 milhões de instalações.

Por ser otimizado para aplicações *web*, é amplamente utilizado na *internet*, sendo muito comum encontrar serviços de hospedagem de sites que oferecem o MySQL e a linguagem PHP, justamente porque ambos trabalham muito bem em conjunto.

O MySQL pode ser obtido gratuitamente através do site do desenvolvedor. Hoje seu desenvolvimento e manutenção empregam aproximadamente 70 profissionais no mundo inteiro, e mais de mil contribuem testando o *software*, integrando-o a outros produtos, e escrevendo a respeito do mesmo.

Algumas características do MySQL:

> - Alta compatibilidade com linguagens como PHP, Java, Python, C#, Ruby e C/C++;
> - Baixa exigência de processamento (em comparação como outros SGBD);
> - Recursos como transactions (transações), conectividade segura, indexação de campos de texto, replicação, etc;
> - Instruções em SQL, como indica o nome.(INFOWESTER, 2007, p.2)

[29] Bancos de dados relacionais - é um banco de dados que se baseia no princípio em que todos os dados estão guardados em tabelas, e estas estão vinculadas através de relações.

2.6.2.2 PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional (SGBDOR), baseado no Postgre Versão 4.2 desenvolvido pelo Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley. (OLIVEIRA, 2008, p.8).

O PostgreSQL suporta grande parte do padrão SQL:2003, além de oferecer funcionalidades como comandos complexos, chaves estrangeiras, gatilhos, visões, integridade transacional e controle de simultaneidade de várias versões. Além disso, fica permitido ao usuário estender suas funções adicionando novos tipos de dados, funções, operadores, funções de agregação, métodos de índice e linguagens procedurais.

Devido à sua licença liberal, o PostgreSQL pode ser utilizado, modificado e distribuído por qualquer pessoa para qualquer finalidade, seja privada, comercial ou acadêmica, livre de encargos.

2.6.2.3 SQL Server

O SQL Server é um sistema de gerenciamento de bancos de dados cliente/servidor de alto desempenho com alta integração com o Windows NT. Suas características são a integração com os serviços de *multithreading*[30], agendamento, monitor de desempenho, e log de eventos do Windows NT. Um usuário pode se conectar ao SQL Server com a mesma senha usada para a rede Windows NT. A sua replicação nativa permite disseminar informações para vários locais, reduzindo a dependência de um servidor único, e deixando a informação necessária mais próxima de quem realmente precisa dela. Sua arquitetura paralela, que executa as funções de banco de dados simultaneamente para diversos usuários e tira proveito de sistemas com múltiplos processadores. Um gerenciamento centralizado de todos os servidores através de uma arquitetura de gerenciamento distribuída, com uma interface visual de gerenciamento.

[30] *Multithreading* – Múltiplas linhas de execução dentro de um processo.

2.6.3 Servidores *web*

No momento em que qualquer *site* é acessado, existe um servidor por trás para disponibilizar todas as informações a que se tem acesso. Da mesma forma, quando se envia um e-mail, participa-se de um fórum ou faz-se uma compra *on-line*, existe um servidor ou um conjunto de servidores por trás.

Segundo a INFOWESTER (2007, p.2):

> [...] um servidor Web é um computador que processa solicitações HTTP (***H****yper-****T****ext* ***T****ransfer* ***P****rotocol*), o protocolo padrão da *Web*. Quando você usa um navegador de internet para acessar um site, este faz as solicitações devidas ao servidor *Web* do site através de HTTP e então recebe o conteúdo correspondente.

2.6.3.1 Apache

O Apache é o mais bem sucedido servidor *web* livre. O servidor é compatível com o protocolo HTTP versão 1.1. Suas funcionalidades são mantidas através de uma estrutura de módulos, permitindo inclusive que o usuário escreva seus próprios módulos, utilizando a API do software.

O Apache não executa somente o protocolo HTTP, mas também os protocolos HTTPS (O HTTP combinado com a camada de segurança SSL - *Secure Socket Layer*), o FTP (File *Transfer Protocol*), entre outros.

Graças ao Apache Server ser um *software* livre, pode-se utilizá-lo gratuitamente, o que faz com que seja o servidor mais usado no mundo. Algumas das características mais importantes para a utilização do Apache são:

> • Possui suporte a scripts cgi usando linguagens como Perl, PHP, Shell Script, ASP, etc.
> • Autenticação requerendo um nome de usuário e senha válidos para acesso a alguma página/sub-diretório/arquivo (suportando criptografia via Crypto e MD5).
> • Negociação de conteúdo, permitindo a exibição da página Web no idioma requisitado pelo Cliente Navegador.
> • Suporte a servidor Proxy ftp e http, com limite de acesso, caching (todas flexivelmente configuráveis).
> • Suporte a proxy e redirecionamentos baseados em URLs para endereços Internos.
> • Módulos DSO (Dynamic Shared Objects) permitem adicionar/remover funcionalidades e recursos sem necessidade de recompilação do programa.(CDTC, 2006, p.5)

2.6.3.2 IIS

O IIS (*Internet Information Services*) é um servidor *web* criado pela *Microsoft* para seus sistemas operacionais. Uma de suas características mais ressaltadas é a geração de páginas HTML dinâmicas, utilizando tecnologia proprietária ASP (*Active Server Pages*). Outra característica importante é a facilidade de utilização de outras tecnologias com adição de módulos de terceiros. Para utilizar essa ferramenta faz-se necessário adquirir licença de uso que para cada instalação ou versão é preciso de pagamento.

Totalmente integrado ao sistema operacional e dotado de uma *interface* administrativa 100% gráfica, o IIS é uma das melhores opções disponíveis para hospedagem de *websites*, *sites* FTP e grupos de notícias, bem como o desenvolvimento de aplicações.

2.6.4 Provedor de hospedagem

A hospedagem é um espaço na Internet para que qualquer pessoa possa acessar determinado site de qualquer lugar do mundo. Geralmente este espaço é fornecido por um provedor de hospedagem de sites. O provedor possui servidores conectados à internet e normalmente cobra uma taxa para utilização desse serviço, sendo que os valores cobrados variam dependendo dos tipos de ferramentas solicitadas pelo cliente, a quantidade de espaço em disco, taxa de transferência, quantidade de contas de e-mail, banco de dados, linguagens de programação, etc.

Cabe ao provedor realizar cópias de segurança (*backup*[31]) diárias, controlar infecções por vírus e ataques de *spammers*[32] ou *hackers*[33]**,** mantendo assim o maior tempo possível os sites no ar, e garantindo o sigilo das informações dos clientes. Todo o conteúdo relativo aos dados do sistema hospedado, suas atualizações, correções ou programações ficam a critério do cliente.

O sistema proposto neste projeto será hospedado em uma empresa que presta este tipo de serviço, para garantir a integridade e disponibilidade dos dados, e prover o maior tempo possível do sistema à disposição na *web*.

---

[31] *Backup* - cópia de dados de um dispositivo para o outro com o objectivo de posteriormente os recuperar

[32] *Sppamers* – pessoas que enviam mensagens não solicitadas em massa.

[33] *Hackers* - indivíduos que elaboram e modificam *software* e *hardware* de computadores.

2.6.5 *Design Patterns*

São padrões de projetos de *software* que descrevem soluções para problemas recorrentes no desenvolvimento de sistemas de *software* orientados a objetos. Segundo Oliveira (2004), os *design patterns* são altamente estruturados. Eles são documentados a partir de um modelo que identifica a informação necessária para entender o problema do *software* e a solução em termos de relacionamentos entre as classes e objetos necessários para implementar essa solução.

Na orientação a objetos, o termo *design pattern* é largamente utilizado para solucionar determinados problemas de projeto, descrevendo formas de comunicação e relacionamento entre classes e objetos. Isso traz maior flexibilidade, e principalmente, a reusabilidade de determinada solução em problemas semelhantes.

2.6.5.1 MVC (Model View Controller)

O MVC é um dos *design patterns*[34] mais conhecidos de todos, dividido em 3 (três) camadas, compostas pelas letras do seu nome.

a) *Model* (modelo) - Representa o domínio de negócios da aplicação, incluindo os métodos de acesso ao banco de dados. No sistema implementado, foi utilizado um active record para representar esta camada.
b) *View* (visualização) - Define a interface com o usuário, a organização dos campos e a sua distribuição na tela. A camada de visualização não exerce nenhum controle de fluxo, nem lógica de negócios.
c) *Controller* (controle) - Processa e responde a eventos, geralmente ações do usuário, e pode invocar alterações no Model. É lá que é feita a validação dos dados e também é onde os valores postos pelos usuários são filtrados.

A utilização do MVC trouxe algumas vantagens, como por exemplo, a reutilização de um mesmo objeto de modelo em visualizações diferentes.

[34] *Design patterns* – padrões de arquitetura de software.

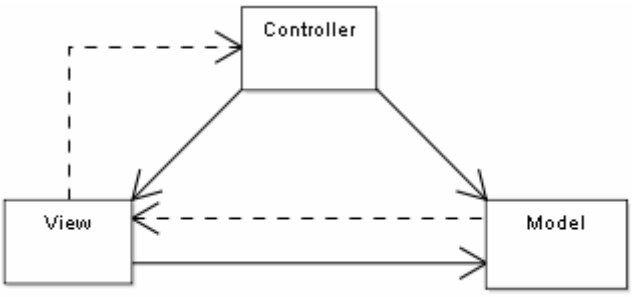


Figura 5 - Modelo MVC
Fonte: Dall'Oglio (2007, p. 478).

2.6.5.2 DAO

O *design pattern Data Access Object* (DAO) é parte essencial da arquitetura de aplicação, pois fornece uma técnica para separar a persistência de objetos e a lógica de acesso a dados de qualquer mecanismo de persistência específica ou API, trazendo benefícios à arquitetura. O *Data Access Object* oferece flexibilidade para alterar o mecanismo de um pedido de persistência ao longo do tempo sem a necessidade de re-engenharia lógica do aplicativo que interage com a camada de dados *Object Access*.

O DAO deve funcionar como um tradutor dos mundos. Suponha um banco relacional. O DAO deve saber buscar os dados do banco e converter em objetos para ser usado pela aplicação. Semelhantemente, deve saber como pegar os objetos, converter em instruções SQL e mandar para o banco de dados. É assim que um DAO trabalha. Devido à sua qualidade de tradutor, o DAO abstrai a origem e o modo de obtenção/gravação dos dados, de modo que o restante do sistema manipula os dados de forma transparente, sem se preocupar com o que acontece por trás dos panos. Isso ajuda muito em processos de migrações de fonte de dados e testes unitários. (CARNEIRO, 2009)

2.6.5.3 Query object

É uma estrutura de objetos que transforma-se em instruções SQL, criando um conjunto de classes que representam estas instruções, porém manipulando os dados através de critérios de seleção como *insert, delete* e *update*. (DALL'OGLIO, 2007).

Além de facilitar o uso do SQL dentro da aplicação, esta *design pattern* torna as expressões mais independentes dentro do sistema, permitindo a reusabilidade mesmo em tabelas ou bancos diferentes.

2.6.5.4 Composite Pattern

Na orientação a objetos, permitir que um objeto contenha outro objeto gerando relacionamentos complexos chama-se composição. (DALL'OGLIO, 2007). A *Composite Pattern* possibilita relações em todo o sistema através da hierarquia de objetos, tratando da mesma forma objetos individuais e objetos compostos.

2.6.5.5 Factory Pattern

A *Factory Pattern* possibilita ocultar detalhes da criação de objetos com características semelhantes. Oferece um ponto central onde serão instanciados os objetos, evitando assim que estas instâncias se espalhem pelo código, facilitando a manutenção do código. (DALL'OGLIO, 2007).

2.6.6 Mapeamento Objeto-Relacional

O mapeamento objeto-relacional é utilizado para separar o domínio de negócios da manipulação de dados via linguagem SQL. Os objetos trabalham como ponteiros para outros objetos, enquanto os bancos de dados tratam do relacionamento de suas estruturas por meio de chaves presentes nas tabelas, estabelecendo assim vínculos entre estas tabelas.

O sistema utilizou algumas técnicas para tratar da relação entre objetos e o modelo relacional.

2.6.6.1 Identify Field

É uma chave de identificação do registro para objetos do negócio da aplicação. É necessária porque, apesar de objetos não necessitarem de uma chave para se tornarem únicos, pois são armazenados em regiões diferentes da memória e referenciados por suas variáveis, em algum momento após serem instanciados,

haverá a necessidade armazenar novamente no banco de dados, substituindo o valor existente.

2.6.6.2 Lazy Initialization

Para evitarmos uma sobre-carga de dados em memória no momento de carregarmos os relacionamentos de um objeto, dependendo do ponto de partida, utiliza-se a *lazy initialization*, para que os objetos relacionados sejam instanciados somente quando forem necessários à aplicação. No PHP, utilizam-se métodos interceptadores para controlar as interações nos objetos. No sistema, o método *_get( )* é utilizado para tal.

2.6.6.3 Gateways

É uma interface utilizada para armazenar informações do modelo de negócios no banco de dados, que mantém um acesso transparente neste recurso. Existem algumas *gateway patterns* que são utilizadas para isso:

a) *Active Record*: Contém métodos de persistência comuns a todos os objetos da aplicação na base de dados, e também métodos do modelo de negócio, com operações para armazenar, remover e ler um objeto na base de dados.
b) *Repository*: É uma camada de aplicação que media a comunicação entre o banco de dados e os objetos do negócio, atuando como um gerenciador de coleções de objetos, evitando assim, que a cada diferente critério de seleção para retorno de um objeto, seja necessário um novo método.

2.6.7 Modelagem de dados

Conforme Ramos (2006, p.10), “[...] a UML é uma linguagem unificada que auxilia na modelagem de sistemas de informação desenvolvidos no paradigma orientado a objetos (OO)”.

A UML permite a modelagem de um grande número de aplicações, como sistemas *web*, sistemas de informação geográficos, sistemas concorrentes, o que a

diferencia de outros métodos de modelagem existentes, sendo independente dos tipos de ferramentas de modelagem.

Dentre outros, a UML providencia os seguintes tipos de diagramas:

a) diagramas de casos de uso;
b) diagramas de classes e de objetos;
c) diagramas de comportamento;
d) diagramas de estados;
e) diagramas de atividades;
f) diagramas de interação;
g) diagramas de arquitetura;
h) diagrama de componentes;
i) diagramas de instalação.

Para o paradigma orientado a objetos existem outras ferramentas, mas a mais popular e adotada no mercado é a UML.

Para outros paradigmas de análise de sistemas, como a análise estruturada ou análise essencial existem outras ferramentas e metodologias de projeto e documentação, que não cabem ao escopo deste trabalho.

2.7 Testes

A aplicação de testes é uma etapa importante na implementação de um *software*, e visa garantir a qualidade do sistema. A qualidade refere-se à ausência de defeitos, e principalmente, à aptidão do sistema aos objetivos propostos.

2.7.1 Tipos de testes

Existem muitos tipos de testes, cada um avaliando características e atributos individuais do *software*, que são aplicados ao longo do ciclo de vida do sistema. (KRUCHTEN, 2003). Os tipos de testes mais comuns são:

2.7.1.1 Teste de função

Testes destinados a validar as funções do objetivo do teste conforme o esperado, fornecendo os serviços, métodos ou casos de uso necessários. Esse

teste é implementado e executado em diferentes objetivos do teste, como unidades, unidades integradas, aplicativos e sistemas.

2.7.1.2 Teste de segurança

Testes destinados a garantir que o objetivo do teste e os dados (ou sistemas) possam ser acessados apenas por determinados atores. Esse teste é implementado e executado em vários objetos de teste.

2.7.1.3 Teste de volume

Teste destinado a verificar a capacidade do objetivo do teste de lidar com um grande volume de dados, como entrada e saída ou residente no banco de dados. O teste de volume abrange estratégias de teste, como, por exemplo, a entrada de dados do volume máximo de dados em cada campo ou a criação de consultas que retornem todo o conteúdo do banco de dados ou que tenham tantas restrições que nenhum dado seja retornado.

2.7.1.4 Teste de usabilidade

Testes que enfatizam fatores humanos, estética, consistência na interface do usuário, ajuda on-line e contextual, assistentes e agentes, documentação do usuário e material de treinamento.

2.7.1.5 Teste de integridade

Testes destinados a avaliar a robustez do objetivo do teste (resistência a falhas) e a compatibilidade técnica em relação a linguagem, sintaxe e utilização de recursos. Esse teste é implementado e executado em vários objetivos do teste, como unidades e unidades integradas.

2.7.1.6 Teste de estrutura

Utilizado para avaliar a adequação do objetivo do teste em relação a seu design e sua formação. Em geral, esse teste é realizado em aplicativos habilitados para a *web*, garantindo que todos os links estejam conectados, que o conteúdo apropriado seja exibido e que não haja conteúdo órfão.

2.7.1.7 Teste de stress

Tipo de teste de confiabilidade destinado a avaliar como o sistema responde em condições anormais. O stress no sistema pode abranger cargas de trabalho extremas, memória insuficiente, *hardware* e serviços indisponíveis ou recursos compartilhados limitados. Normalmente, esses testes são executados para compreender melhor como e em quais áreas o sistema será dividido, para que os planos de contingência e a manutenção de atualização possam ser planejados e orçados com bastante antecedência.

2.7.1.8 Teste de carga

Tipo de teste de desempenho usado para validar e avaliar a aceitabilidade dos limites operacionais de um sistema de acordo com cargas de trabalho variáveis, enquanto o sistema em teste permanece constante. Em algumas variáveis, a carga de trabalho permanece constante e a configuração do sistema em teste é que varia. Geralmente, as medições são tomadas com base na taxa de transferência de dados da carga de trabalho e no tempo de resposta da transação alinhado. As variações na carga de trabalho normalmente incluem a emulação das cargas médias e máximas que ocorrem dentro de tolerâncias operacionais normais.

2.7.1.9 Perfil de desempenho

Teste em que o perfil de andamento do objetivo do teste é monitorado (inclusive fluxo de execução, acesso a dados e chamadas de função e de sistema), a fim de identificar e lidar com gargalos de desempenho e processos ineficientes.

2.7.1.10 Teste de configuração

Destinado a garantir que o objetivo do teste funcione conforme o esperado em diferentes configurações de *hardware* e/ou *software*. Esse teste também pode ser implementado como um teste de desempenho do sistema.

2.7.1.11 Teste de instalação

Destinado a garantir que o objetivo do teste seja instalado conforme o esperado em diferentes configurações de *hardware* e/ou *software* e sob diferentes condições (como no caso de espaço insuficiente em disco ou interrupção de energia). Esse teste é implementado e executado em aplicativos e sistemas.

3 DESENVOLVIMENTO

Nesta seção são apresentadas as definições metodológicas para o desenvolvimento do *software*, opções de tecnologias de desenvolvimento, apresentação do levantamento de requisitos, da diagramação e resultados, estrutura do sistema e o plano de testes elaborado.

3.1 Processos de desenvolvimento de *software* escolhido

O processo de desenvolvimento de *software* escolhido para a implementação do sistema foi o RAD (*Rapid Aplication Development*), apresentado na sub-seção 2.2.1.3. A escolha foi baseada nos fundamentos do RAD, sendo um modelo baseado em componentes, aplicado em um desenvolvimento curto. Sendo o desenvolvedor um dos usuários do sistema, os requisitos do sistema tornaram-se claros e bem definidos. Outro quesito facilitador foi a modularização, graças ao acesso livre do analista na empresa e aos outros usuários do sistema proposto para apresentação dos resultados (módulos), e possíveis reparos nos requisitos. O sistema proposto também reaproveitou código já existente proveniente de um *framework*[35], e utilizou programas que facilitaram a instalação e configuração de ferramentas de codificação, outra característica apontada no RAD.

3.2 Definição da análise de *software*

Dentre as técnicas para levantamento de requisitos listadas na sub-seção 2.3.2, a opção escolhida foi a observação direta (sub-seção 2.3.2.5), pois o analista consegue avaliar não somente os requisitos em si, mas também o ambiente de trabalho sem interrupção das ações do usuários, podendo perceber alguns “vícios”, ou problemas não levantados ou não explicados em outras técnicas, por exemplo. Também fica facilitada ao analista a real verificação do que é necessário implementar, evitando desperdício de código. A seleção da técnica de observação

[35] *framework* – É uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

direta ficou facilitada devido ao desenvolvedor/analista ser usuário e assim, ter transito total a todos os setores da empresa analisada.

3.2.1 Efetuando a análise de requisitos

A técnica de observação direta foi efetuada na empresa Uniplus Cooperativa de Serviços Ltda, localizada na cidade de Taquara-RS.

3.2.1.1 Visão atual dos métodos de trabalho na Uniplus

Em análise realizada em uma empresa prestadora de serviços, e pesquisa feita em outras três empresas do mesmo ramo, verifica-se que o processo do fornecimento e controle dos serviços prestados é semelhante na quase totalidade das empresas, sejam elas empresas convencionais ou cooperativas de prestação de serviços.

A maioria das empresas é especializada em alguma área, seja limpeza, saúde, vigilância, e tem o mesmo método de trabalho. A Uniplus fornece serviços exclusivamente na área da saúde. O seu quadro de associados compõe-se de dentistas, fisioterapeutas, agentes de saúde, enfermeiros, técnicos de enfermagem, nutricionistas e médicos.

3.2.1.2 Observação direta

O tomador de serviços contrata determinado número de horas de trabalho por mês, definindo a especialidade solicitada, os turnos de trabalho e número de horas que necessita destes serviços por dia em cada local.

Cabe a empresa, planejar, criar e adequar-se as escalas de trabalho, para cada especialidade e tomador, buscando ou compondo do seu quadro de prestadores, os profissionais que possuam o horário disponível e o perfil mais adequado ao ambiente fornecido pelo contratante. Também fica a critério da empresa o controle das horas trabalhadas, bem como o fechamento do total mensal de horas, tanto para cobrança do contratante, quanto para o controle da folha de pagamento.

As ferramentas atuais utilizadas para o fechamento de escalas atualmente são:

a) planilhas do *Microsoft Excel* onde faz-se a geração das escalas e o cadastro de cada médico por especialidade (anexos A e B), que são impressas e expostas em um mural, onde sofrem alterações conforme a necessidade (anexo C);
b) quadro branco onde são listadas todas as vagas existentes nas escalas (anexo D);
c) para contato com os médicos são usados telefone, *e-mail* e MSN *Messenger*[36];
d) um *site* onde são inseridas manualmente as vagas nas escalas (anexo E).

O processo atual de fechamento de escalas invariavelmente ocorre nesta maneira:

a) ao final de cada mês, o administrador gera as escalas do mês seqüente, entra em contato com todos os médicos que são fixos por dia, confirmando suas datas, e conforme os contatos, anteriores ou durante o mês corrente, vai listando as vagas em aberto no quadro branco e inserindo-as no *site*.
b) semanalmente, são enviadas malas-diretas para todos os contatos médicos com e-mail cadastrado, oferecendo as vagas disponíveis.
c) conforme o retorno dos e-mails, as vagas fechadas são retiradas do quadro branco e do site.
d) para as vagas que não foram preenchidas, inicia-se contato telefônico com os médicos do cadastro no *Excel*, oferecendo-as. A escolha de qual profissional será contatado primeiramente, é feita exclusivamente baseada no conhecimento pessoal que cada administrador coleta quando entra em contato com os prestadores de serviço.
e) caso exista dificuldade no fechamento de uma vaga, e aproximando-se a data desta, a empresa oferece pagamento antecipado ou um aumento no valor/hora pago ao prestador de serviço.
f) fechando a vaga, esta é apagada do quadro branco e retirada do site.
g) no primeiro dia de cada mês, são recolhidas as folhas de horas de todos os prestadores de serviço em todos os contratantes, estas horas são

[36] MSN *Messenger*: *Software* que permite conversar online e em tempo real com outras pessoas.

somadas e produz-se um manifesto para cada contratante com o número de horas que cada prestador de serviço trabalhou no mês.

h) todo o processo inicia-se novamente.

Na conclusão da observação, elaborou-se um documento onde foram definidos os usuários do sistema e alguns aspectos referentes ao modo de trabalho, e quais tarefas são executadas pelos usuários do sistema (anexo F). Um documento prévio de requisitos (anexo G), também foi elaborado para aprovação da direção da empresa, e um fluxograma de ações foi gerado (anexo H).

3.2.1.3 Escopo

Conforme a dificuldade encontrada na busca de mão de obra qualificada, o aumento dos custos envolvidos na gestão e manutenção das escalas de trabalho, o crescimento na oferta de serviços terceirizados, e por fim, a inexistência de um *software* similar no mercado, busca-se desenvolver um sistema *web* de apoio à decisão para auxílio na gestão e manutenção das escalas de trabalho da área médica da empresa.

3.2.1.4 Requisitos levantados

Conforme o método de observação direta, e avaliação dos anexos A, B, C, D, E, F e G, foram levantados e listados os seguintes requisitos funcionais:

a) Cadastro de usuários: Permitir o cadastro com usuário, senha e permissões de acessos para pessoas que utilizarão o sistema (contratante, gestor de escalas e médico).
b) Cadastrar novo contrato: Permitir a inclusão e exclusão de novos contratos, inserindo as especialidades desejadas, o número total de horas prestadas por especialidade a cada mês e o valor/hora cobrado por especialidade.
c) Criar novas escalas: Permitir a criação, alteração e exclusão de novas escalas de trabalho, definidas por contratante, especialidade, período mensal e valor/hora.

d) Cadastar dados dos médicos: Ao gestor de escalas, bem como ao próprio médico, fica possível o cadastro dos dados pessoais e os horários disponíveis para trabalho.
e) Informar vagas existentes: O sistema deve informar ao gestor de escalas e ao médico cadastrado as vagas de trabalho disponíveis e as informações relevantes à esta vaga, e se o médico cadastrou seus horários disponíveis, deve fornecer as vagas conforme a disponibilidade de cada médico.
f) Listar candidatos a vaga: O sistema deve informar ao gestor de escalas os médicos que poderão assumir cada vaga conforme sua disponibilidade informada.
g) Reservar vaga: Sistema deve permitir que médico reserve a vaga a qual tenha interesse, e informe ao gestor de escalas através de e-mail o seu interesse.
h) Informar vaga disponível ao médico: Sistema deverá possibilitar ao gestor de escalas o envio de um e-mail com as vagas disponíveis conforme sua disponibilidade.
i) Inserir médico na vaga: Tendo o aceite do médico listado como possível candidato, sistema deve inserir diretamente o médico na vaga.
j) Alteração de valor/hora por vaga: Sistema deve permitir que seja modificado o valor pago por hora a cada vaga independente.
k) Consultar contratos: Sistema deve permitir ao contratante e ao gestor de escalas a visualização da quantidade de horas contratadas por especialidade por mês, e permitir a visualização da quantidade de horas já efetuadas
l) Exibir mensagens de restrição: Sistema deve fornecer mensagens ao gestor sobre excesso de horas semanais do mesmo médico por contratante, deve gerar mensagem quando vaga reservada por médico não corresponde à sua lista de horários disponíveis, e deve avisar ao gestor sobre um possível excesso de horas trabalhadas por especialidade em relação ao número de horas contratadas.
m) Gerar manifesto: Ao final de cada mês, sistema deve permitir que gestor de escalas realize o somatório de horas de cada médico por contratante, por especialidade e por local de trabalho.

n) Impressão de escalas e manifesto: Sistema deve permitir que o gestor de escalas e o contratante possam imprimir as escalas de trabalho.

3.2.1.5 Requisitos não funcionais

a) Segurança de dados: Sistema deve garantir a integridade dos dados, evitando acessos indevidos e restringindo aos nomes cadastrados na base de dados a escolha de cada usuário do sistema.
b) Oferecer interfaces amigáveis: Sistema deve oferecer uma interface amigável, principalmente na geração de escalas e na listagem de vagas abertas.
c) Confiabilidade: *Software* deve operar sem a ocorrência de falhas.
d) Robustez: Sistema deve continuar operando apesar de eventuais falhas de codificação não apontadas nos testes.
e) Disponibilidade do *software*: Sistema deve operar com qualquer versão de *hardware* ou sistema operacional.
f) Integridade dos dados: Sistema deve garantir que um dado só seja alterado após o término da execução de um processo, e caso este seja interrompido, o dado mantenha seu valor original.

3.3 Definição da arquitetura de *software*

A arquitetura definida para o sistema foi a cliente-servidor, vista na sub-seção 2.4.1.2, pois é uma das arquiteturas que mais adapta-se à sistemas *web*, e uma das que mais evoluem atualmente, onde todos os dados são centralizados em um servidor, que recebe as requisições dos clientes, as processa e devolve-as ao seu solicitante. Também foi escolhida por ser a arquitetura empregada no *framework* utilizada como base do desenvolvimento do *software.*

3.4 Definição do paradigma de programação

O paradigma de programação definido foi o orientado a objetos, por ser um dos paradigmas mais utilizados atualmente no mercado. Facilidades como modularidade, facilitando a manutenção, a reusabilidade, reaproveitando código,

pesaram na decisão, em vista de que a *template* utilizada como base foi toda desenvolvida orientada a objetos, e a ideia de que o sistema desenvolvido sirva como protótipo para sistema mais amplo.

3.5 Tecnologias escolhidas

As linguagens de programação utilizadas para a implementação do sistema foram o PHP (sub-seção 2.6.1.2), por ser uma linguagem onde é possível a programação orientada a objetos a partir da versão 4, também por suportar a maioria de protocolos de comunicação, e ter versões disponíveis a maioria absoluta de sistemas operacionais existentes. Existe documentação farta a respeito da linguagem, além de ser a linguagem utilizada no *framework* escolhido como base. O JavaScript (2.6.1.4) é uma linguagem auxiliar na interação em páginas *web*, e foi utilizado para acesso a alguns conteúdos aleatórios, como as chamadas aos arquivos de geração de PDF. O HTML (sub-seção 2.6.1.1) foi utilizado para a criação do *layout* da página.

O banco de dados utilizado foi o MySQL *Server* (sub-seção 2.6.2.1), por ser grátis para utilização, sendo um dos bancos mais populares atualmente e com enorme documentação para pesquisa. É um banco multi-tarefas e multi-usuários, e funciona em basicamente todos os sistemas operacionais. O MySQL possui desempenho e robustez satisfatórias para o sistema implementado.

O servidor *web* definido foi o Apache (sub-seção 2.6.3.1), por ser um dos mais bem sucedidos servidores de código livre, é compatível com o protocolo utilizado (http), e também é disponibilizado para a maioria dos sistemas operacionais.

A escolha do PHP, MySQLServer e o Apache também se deve a utilização do *WampServer*, um *software* utilizado para a instalação e configuração rápida destas 3 tecnologias, além de oferecer também *interfaces* para trabalho na base de dados, como o *PHPMyAdmin*.

Os *Design Patterns* utilizadas foram o MVC (sub-seção 2.6.5.1), utilizado para que as camadas de modelo, controle e visualização fiquem separadas, deixando um código mais limpo, e de fácil manutenção, além do reaproveitamento de código. O *Query Object* (seção 2.6.5.3) foi utilizado para que um conjunto de classes represente instruções SQL para manipulação de dados. O *Composite Pattern* (sub-

seção 2.6.5.4) foi utilizado para geração de condições de filtragem no *software*, e o *Factory Pattern* (sub-seção 2.6.5.5) foi utilizado para instanciar um objeto com as informações de conexões com o banco de dados, fazendo assim com que a aplicação desconheça o banco utilizado.

3.6 Modelagem e Implementação

A primeira etapa para da modelagem foi o levantamento dos casos de uso, seguido da elaboração do modelo relacional do banco de dados.

Seguindo o conceito do modelo de processo de desenvolvimento de *software* escolhido (RAD), dividiu-se o sistema em 4 módulos:

a) Módulo de *login*;
b) Módulo médico;
c) Módulo gestor;
d) Módulo contratante.

Cada módulo apresenta os modelos de negócio, modelos de dados e o resultado da geração da aplicação.

Para a construção do sistema, foram aplicadas as ferramentas e tecnologias mais indicadas à sistemas *web*. O Sistema também foi estruturado de forma a obter-se um código fonte mais claro, sem a inclusão de HTML e PHP numa mesma classe, por exemplo, e aproveitando-se o máximo de abstração e reaproveitamento de código, que a programação orientada a objetos oferece.

3.6.1 Casos de Uso

O caso de uso serve para exemplificar uma sequência de ações realizadas por um ou mais atores em um sistema, para obterem um resultado particular.

3.6.1.1 Atores

Os possíveis atores identificados foram:

a) gestor de escalas;
b) contratante;
c) médico.

3.6.1.2 Casos de uso verificados

Número	Caso de Uso
1	*Login*
2	Cadastrar médico
3	Editar médico
4	Visualizar plantões
5	Editar contratos
6	Editar vagas
7	Reservar vagas
8	Gerar manifesto
9	Imprimir manifesto
10	Imprimir escalas
11	Consultar contrato
12	Visualizar escalas

Quadro 1 - Casos de uso verificados

3.6.1.3 Diagrama de Casos de Uso

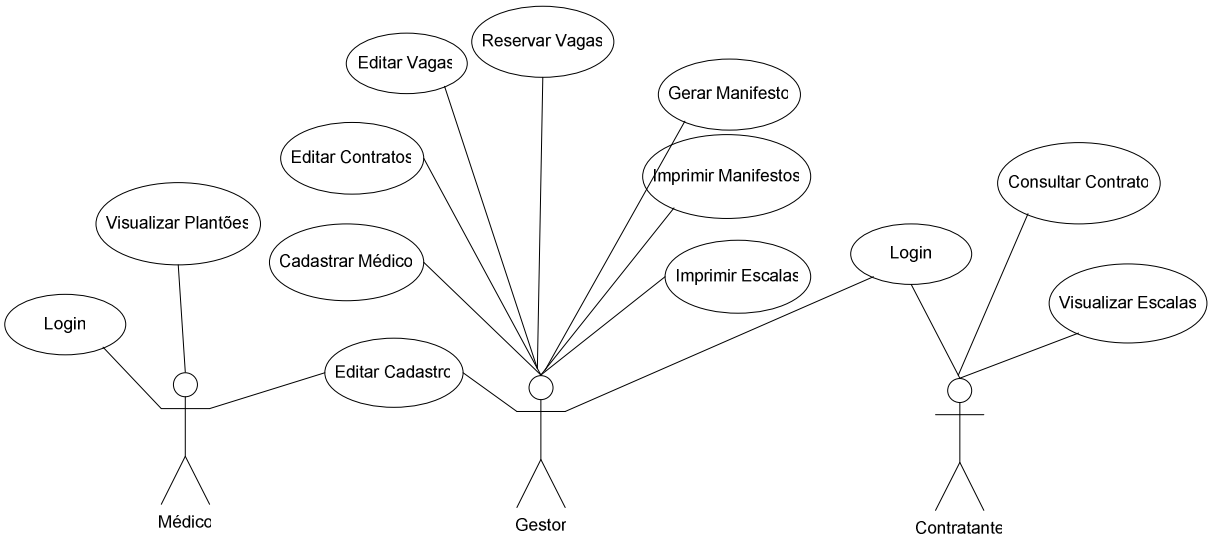


Figura 6 - Diagrama de Casos de uso

3.6.2 Modelo relacional

O modelo relacional é o modelo que descreve a base de dados do sistema apresentando, os relacionamentos e relações quantitativas entre suas entidades.

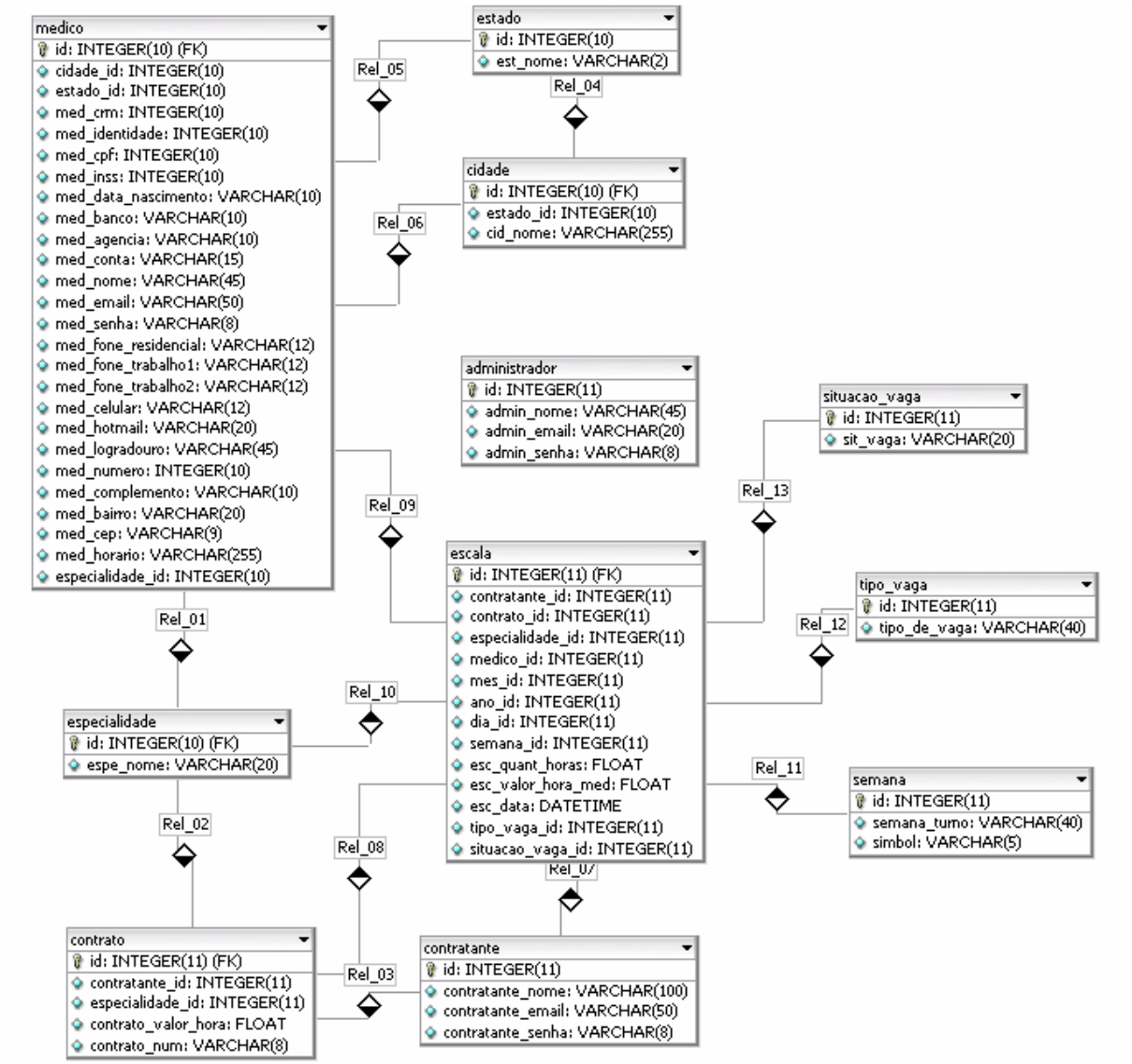


Figura 7 - Modelo Relacional do sistema

3.6.3 Módulo *Login*

O módulo de *login* trata da autenticação de todos os atores ao sistema, que é obrigatória. Neste módulo, os usuários tem a possibilidade de autenticar-se, recuperar senha perdida, ou quando o ator for o médico, a possibilidade de cadastrar-se no sistema.

3.6.3.1 Diagrama de classes

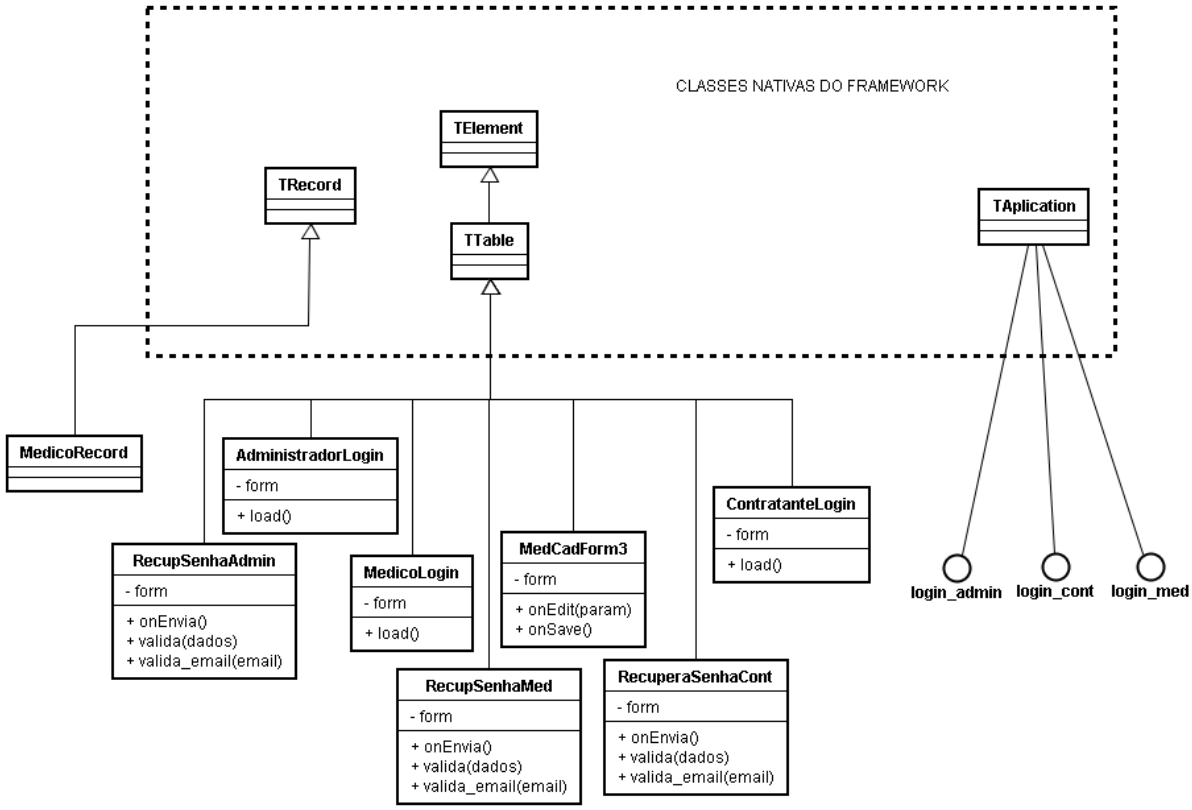


Figura 8 - Diagrama de classes do módulo login

3.6.3.2 Descrição dos casos de uso

a) *Login*

Nome: Login	**Versão 1.0**
Descritivo: Realizar a autenticação de todos os atores do sistema	
Atores envolvidos: Contratante; Médico; Gestor	
Cenário: 1. Todos os atores devem informar e-mail e senha para se autenticar no sistema. 2. O contratante e o gestor são cadastrados diretamente no sistema pelo seu administrador. 3. Caso o médico não seja usuário do sistema, deve cadastrar-se para então efetuar a autenticação. 4. Caso não lembrem da senha, ambos atores tem a possibilidade de à receber por e-mail. 5. Não é permitido o acesso a nenhum módulo do sistema sem que os atores efetuem a autenticação. 6. Cada ator trabalha em um módulo diferente, e só pode ter acesso a seu módulo correspondente.	
Dados: e-mail, senha	

Quadro 2 - Descrição do caso de uso 1

3.6.3.3 Diagramas de sequência

a) acessar

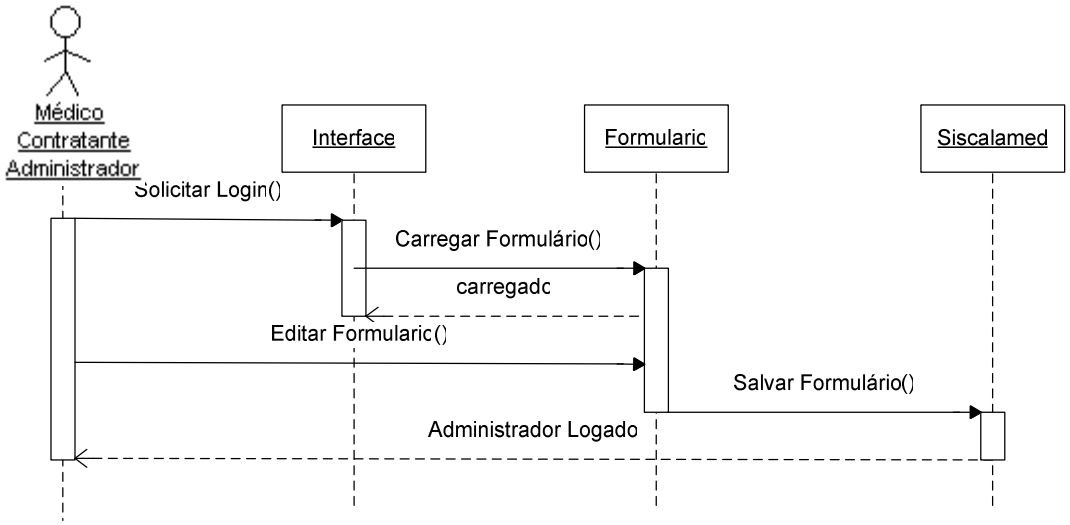


Figura 9 - Diagrama de sequência de autenticação

b) recuperar Senha

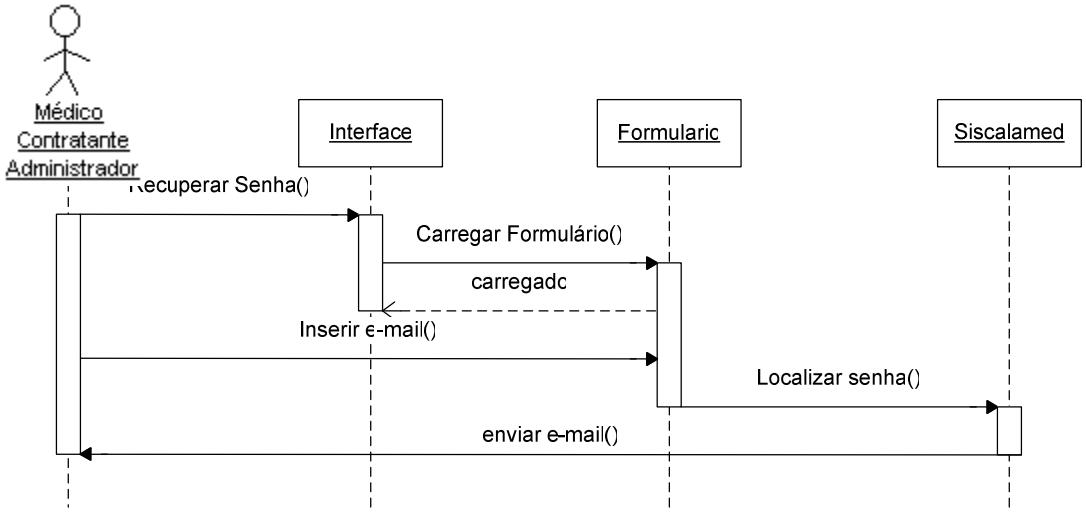


Figura 10 - Diagrama de sequência de recuperação de senha

c) cadastrar-se

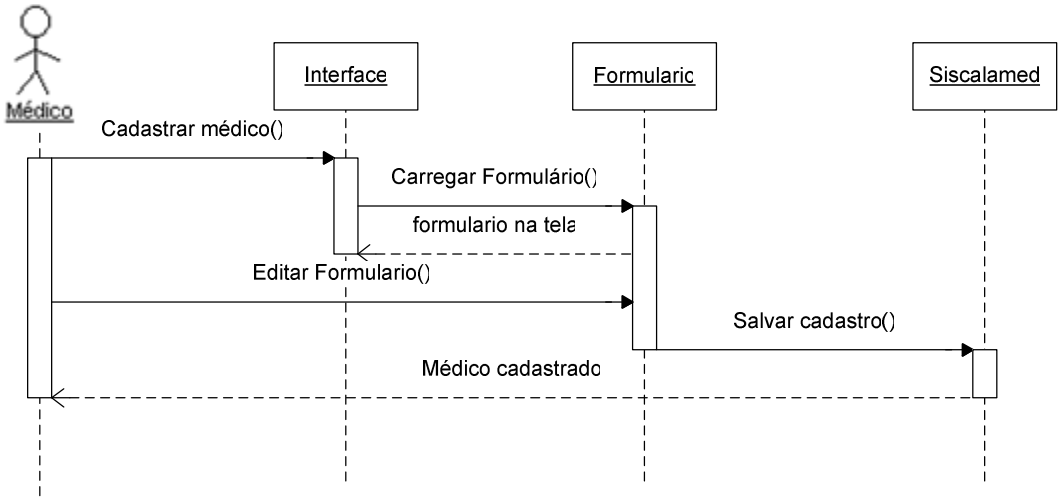


Figura 11 - Diagrama de sequência de cadastro

3.6.3.4 Implementação

a) acessar

A Figura 12 apresenta a interface de autenticação de todos os usuários do sistema, que devem inserir seu e-mail e sua senha.

Figura 12 - *Interface* de autenticação

b) recuperar senha

Na tela representada pela Figura 13, os usuários do sistema têm a opção de recuperar sua senha, recebendo-a através do e-mail cadastrado.

Figura 13 - *Interface* de recuperação de senha

c) cadastrar-se

Na tela representada pela Figura 14, caso o médico não esteja cadastrado no sistema, deverá efetuar o cadastro para que possa autenticar-se no sistema.

Figura 14 - *Interface* de cadastro

3.6.4 Módulo médico

O módulo médico permite que este usuário altere seus dados cadastrais, bem como visualize os plantões da sua especialidade, fazendo uma reserva e enviando um aviso ao gestor.

3.6.4.1 Digrama de classes

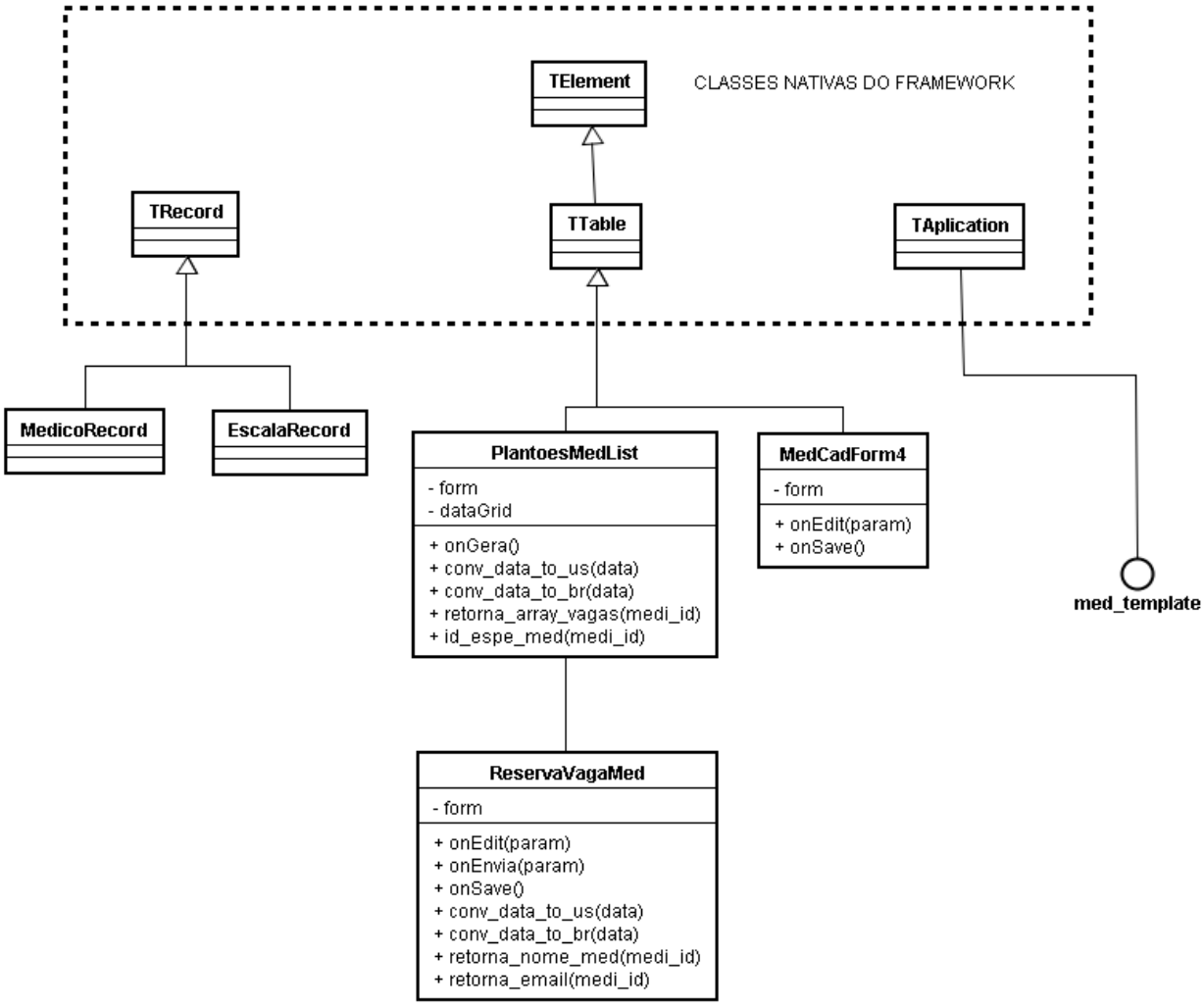


Figura 15 - Diagrama de Classes do módulo médico

3.6.4.2 Descrição dos casos de uso

a) editar Cadastro

Nome:	Editar Cadastro	**Versão 1.0**
Descritivo:	Editar dados cadastrados dos médicos	
Atores envolvidos:	Médico; Gestor	
Cenário: 1. O Médico deve estar autenticado, e só tem acesso aos seus dados cadastrados. 2. O sistema deve carregar todos os dados do referido médico para edição. 3. O sistema deve permitir que o gestor busque o médico pelo nome ou em uma lista. 4. Deve ser permitido que o gestor altere ou exclua um médico. 5. Os horários disponíveis só podem ser alterados em nenhum outro momento que não seja a edição do cadastro.		
Dados: Nome; crm; especialidade; logradouro; numero; complemento; bairro; cidade; cep; estado; tel_residencial; Tel_celular; Tel_trabalho; e-mail; hotmail; rg; cpf; inss; banco; agencia; conta, data_nasc; horários_disp		

Quadro 3 - Descrição do caso de uso de edição de cadastro

b) visualizar plantões

Nome:	Visualizar plantões	**Versão 1.0**
Descritivo:	Verificar os plantões que estão vagos	
Atores envolvidos:	Médico	
Cenário: 1. Ator deve informar o período em que deseja fazer a consulta. 2. Sistema deve gerar uma listagem com os plantões abertos no período, conforme a especialidade do ator. 3. Sistema deve permitir que ator selecione vaga na lista e a reserve. 4. Sistema deve enviar um e-mail ao gestor avisando da reserva da vaga. 5. Ator não tem possibilidade de alterar dados da vaga.		
Dados: Data inicial, Data final, Especialidade, Dia_semana, total_horas, Valor_hora, Tipo_vaga, situação_vaga, medico		

Quadro 4 - Descrição do caso de uso de visualização de plantões

3.6.4.3 Diagramas de Sequência

a) editar Cadastro

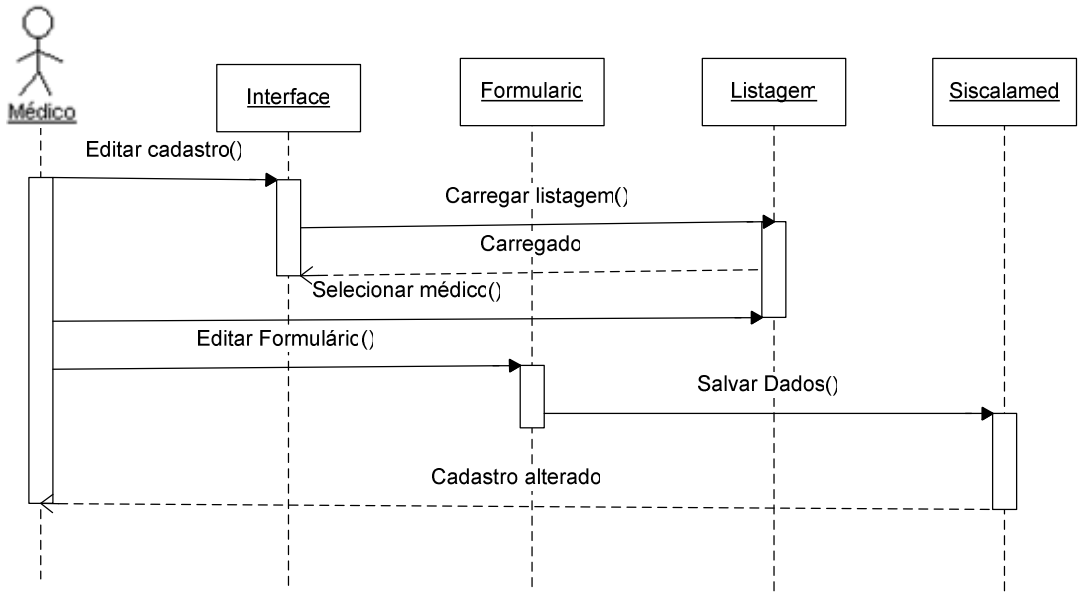


Figura 16 - Diagrama de sequência de edição de cadastro

b) visualizar plantões

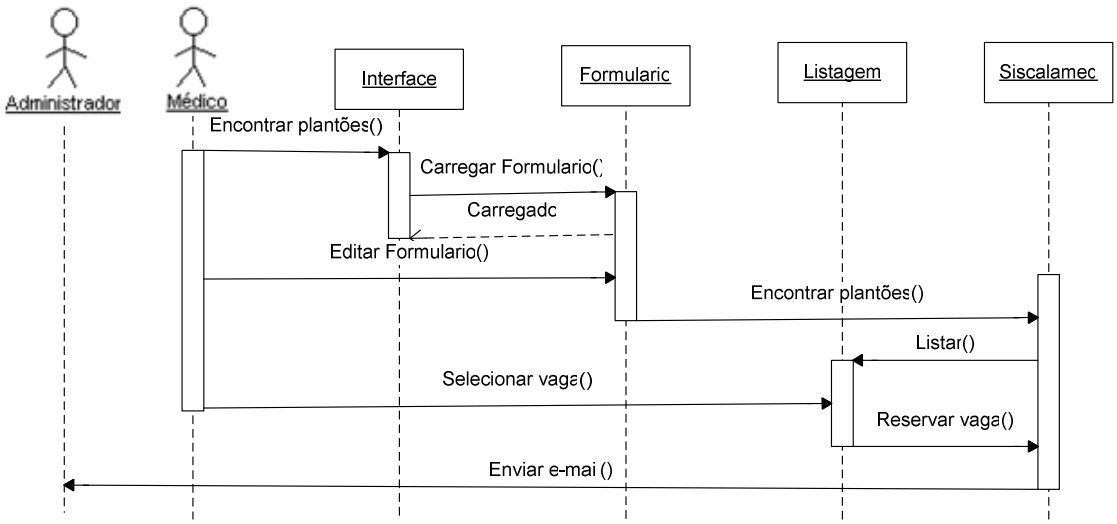


Figura 17- Diagrama de sequência de visualização e reserva de plantões

3.6.4.4 Implementação

a) editar Cadastro

Nas Figuras 18 e 19, o médico tem a opção de editar seus dados cadastrais.

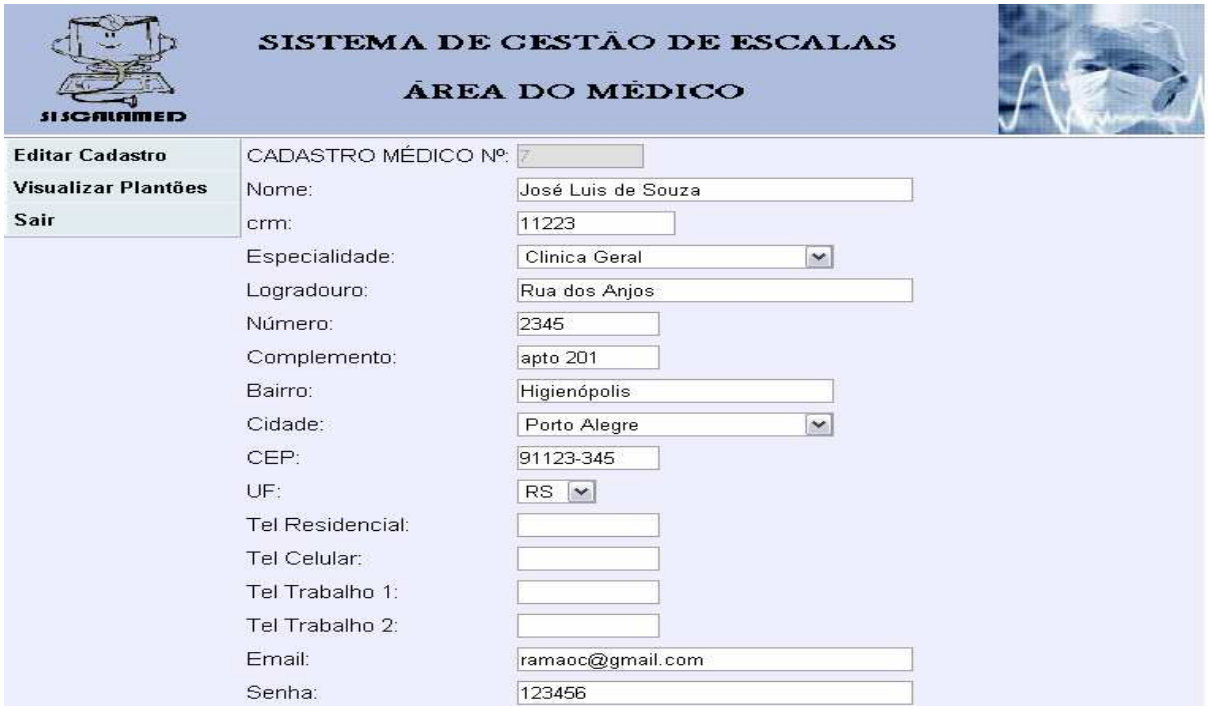


Figura 18 - *Interface* de edição de cadastro (parte 1)

Figura 19 - *Interface* de edição de cadastro (parte 2)

b) visualizar plantões

A Figura 20 representa a *interface* onde o médico lista os plantões que estão em aberto no sistema, conforme a sua disponibilidade de horários cadastrada.

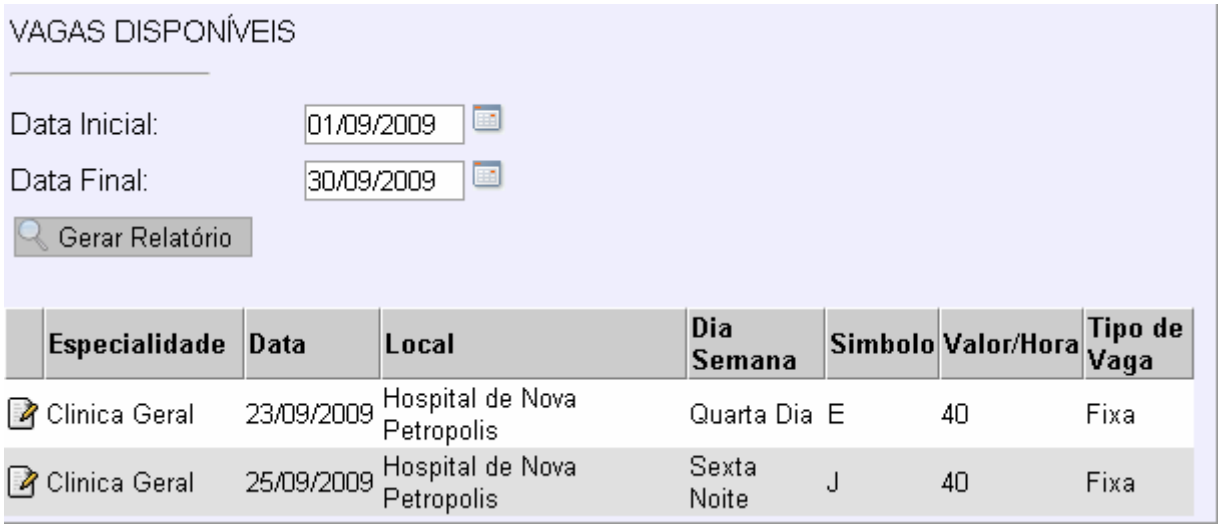


Figura 20 - *Interface* de visualização de plantões

Após a listagem das vagas em aberto, o médico tem a opção de efetuar uma reserva de uma determinada vaga, conforme apresenta a Figura 21.

Figura 21 - *Interface* de reserva de plantões

3.6.5 Módulo gestor

Neste módulo o gestor, que é o principal ator do sistema, executa todas as ações propostas no sistema para cadastro de médicos, gerência das escalas e informações quanto a contratos e contratantes.

3.6.5.1 Diagrama de Classes

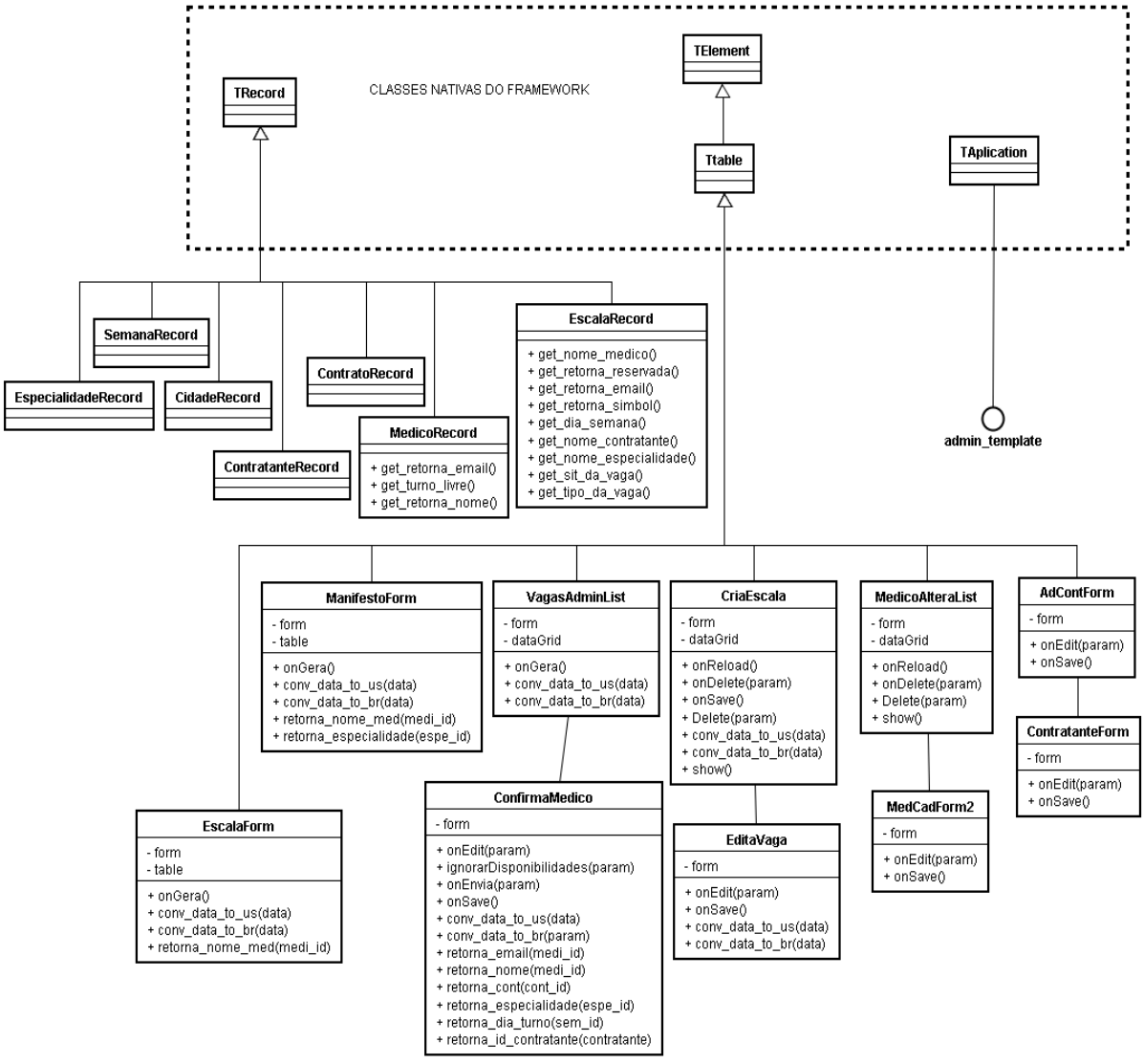


Figura 22 - Diagrama de classes do módulo gestor

3.6.5.2 Casos de uso

a) cadastar médico

Nome:	Cadastar Médico	**Versão 1.0**
Descritivo:	Gestor pode cadastrar novos médicos	
Atores envolvidos:	Médico	
Cenário: 1. Gestor deve estar autenticado. 2. Sistema deve carregar formulário de cadastro. 3. Gestor deve criar uma senha para o médico. 4. Gestor deve inserir os horários disponíveis do referido médico		
Dados: Nome; crm; especialidade; logradouro; numero; complemento; bairro; cidade; cep; estado; tel_residencial; tel_celular; tel_trabalho; e-mail; hotmail; rg; cpf; inss; banco; agencia; conta; data_nasc; horários_disp		

Quadro 5 - Descrição do caso de uso de cadastro médico

b) editar cadastro

A edição do cadastro médico está descrita no Quadro 3

c) editar contrato

Nome:	Editar Contrato	**Versão 1.0**
Descritivo:	Inserir novos contratantes e novos contratos	
Atores envolvidos:	Gestor	
Cenário: 1. O gestor tem a possibilidade de cadastrar novo contratante. 2. O gestor cadastra dados pertimentes a cada contrato. 3. Cada contrato corresponde a um único contratante. 4. Cada contratante pode ter mais de um contrato com mais de uma especialidade. 5. Cada especialidade contratada tem um limite de horas por mês. 6. Cada especialidade tem um valor/hora pago independente por contrato.		
Dados: Contratante; e-mail; senha; contrato; especialidade; num_contrato; valor_hora; total_horas		

Quadro 6 - Descrição do caso de uso de edição de contrato

d) editar vagas

Nome:	Editar Vagas	**Versão 1.0**
Descritivo:	Inserir novas vagas e editá-las	
Atores envolvidos:	Gestor	
Cenário: 1. Sistema deve permitir que ator crie novas escalas de trabalho. 2. As escalas devem estar dentro das especialidades contratadas e das limitações de horas/mês. 3. Cada vaga corresponderá a um turno de 12 horas. 4. Cada turno poderá ter um ou mais médicos. 5. Sistema deve gerar uma lista de vagas. 6. Cada vaga é rotulada como aberta, fechada ou reservada. 7. As vagas podem ser fixas ou substituições 8. A lista de vagas deve permitir que o ator edite ou delete determinada vaga.		
Dados: Contratante; contrato; especialidade; data; dia-semana; medico; total_horas; situação_vaga; tipo_vaga		

Quadro 7 - Descrição do caso de uso de criação e edição de vagas

e) reservar vagas

Nome:	Reservar Vagas	**Versão 1.0**
Descritivo:	Listar vagas em aberto e reservar vagas conforme médicos disponíveis.	
Atores envolvidos:	Gestor	
Cenário: 1. Ator solicita uma lista de vagas conforme período determinado. 2. sistema listará somente as vagas que não possuem médico cadastrado, ou a sua situação seja "aberta" ou "reservada". 3. As vagas são listadas pela ordem da data. 4. Deve-se permitir que o ator selecione a vaga que quer visualizar. 5. Sistema deve carregar os dados da vaga selecionada. 6. Sistema deve listar no campo "médico" somente aqueles que são da mesma especialidade da vaga e que possuem horários disponíveis compatíveis com a vaga. 7. Sistema deve permitir que o ator liste todos os médicos. 8. Sistema deve verificar se médico selecionado para vaga não extrapola número de horas semanais. 9. Sistema deve oferecer a opção de que o ator envie um e-mail ao médico selecionado informando da vaga editada.		
Dados: data_inicial; data_final; contratante; contrato; especialidade; data; dia-semana; medico; total_horas; situação_vaga; tipo_vaga		

Quadro 8 - Descrição do caso de uso de listagem e reserva de vagas

f) gerar Manifesto

Nome: Gerar Manifesto	**Versão 1.0**
Descritivo: Gerar um relatório que conste o somatório de horas executadas por mês para um determinado contratante.	
Atores envolvidos: Gestor	
Cenário: 1. Ator seleciona um contratante, e um período, para a geração do relatório. 2. Sistema deve filtrar todas os plantões para o determinado contratante, dentro do período executado, e que estejam rotulados como "fechado". 3. Sistema deve listar separadamente por especialidade contratada. 4. Sistema deve juntar o numero de horas e o valor pago se um mesmo médico executou mais de um plantão pela mesma especialidade dentro do período selecionado. 5. Sistema deve listar no relatório o medico, a especialidade, o numero total de horas e o valor pago. 6. Sistema deve informar o valor total a pagar.	
Dados: data_inicial; data_final; contratante; especialidade; num_horas; valor_hora; valor_pago; valor_total	

Quadro 9 - Descrição do caso de uso para gerar manifesto

g) imprimir manifestos

Nome: Imprimir Manifesto	**Versão 1.0**
Descritivo: Geração de PDF e Impressão	
Atores envolvidos: Gestor	
Cenário: 1. Após geração do manifesto, sistema deve oferecer a função de uma geração de um arquivo PDF para impressão. 2. Somente o gestor tem acesso a esta funcionalidade.	
Dados: manifesto_fpdf	

Quadro 10 - Descrição do caso de uso de impressão de manifesto

h) imprimir escalas

Nome: Consultar Contrato	**Versão 1.0**
Descritivo: Verificação do consumo de horas dentro de um período	
Atores envolvidos: Contratante	
Cenário: 1. Sistema deve permitir que contratante visualize o consumo de horas/mês por especialidade contratada. 2. Contratante define o período e a especialidade à consultar. 3. Sistema verifica as vagas com o rótulo "fechada" dentro do período e da especialidade solicitada. 4. Sistema gera relatório informando o a quantidade de horas/mês contratadas e o consumo dentro do período, totalizando o saldo.	
Dados: Data_inicial; data_final; especialidade; num_horas_cont; total_horas_cons; saldo_horas	

Quadro 11 - Descrição do caso de uso de impressão de escalas

3.6.5.3 Diagramas de Sequência

a) cadastrar médico

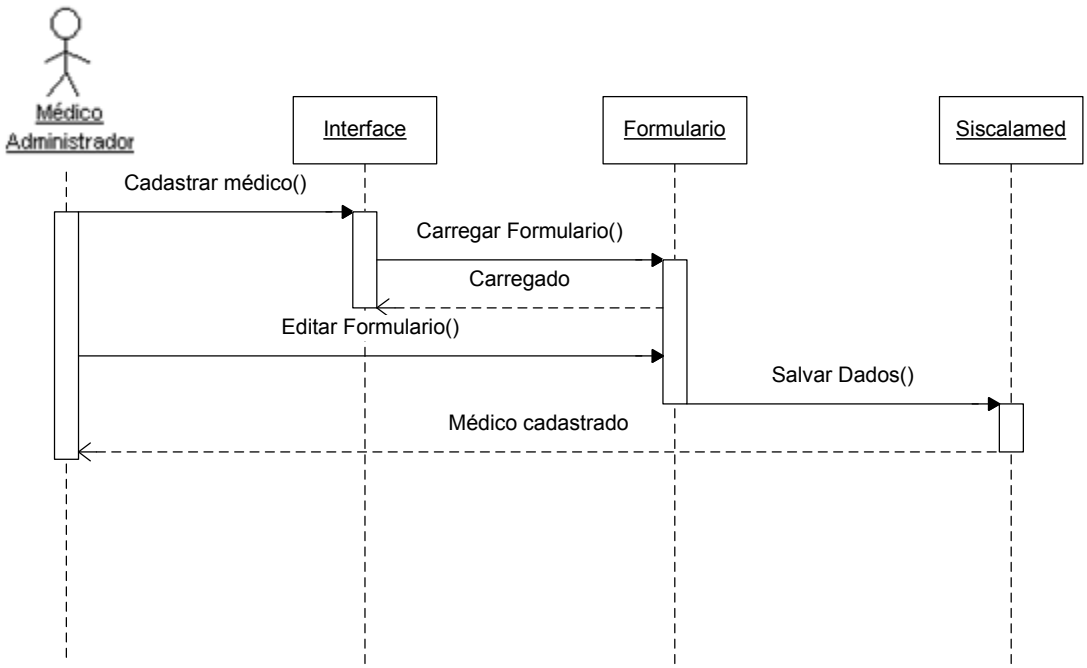


Figura 23 - Diagrama de sequência de cadastro médico

b) editar cadastro

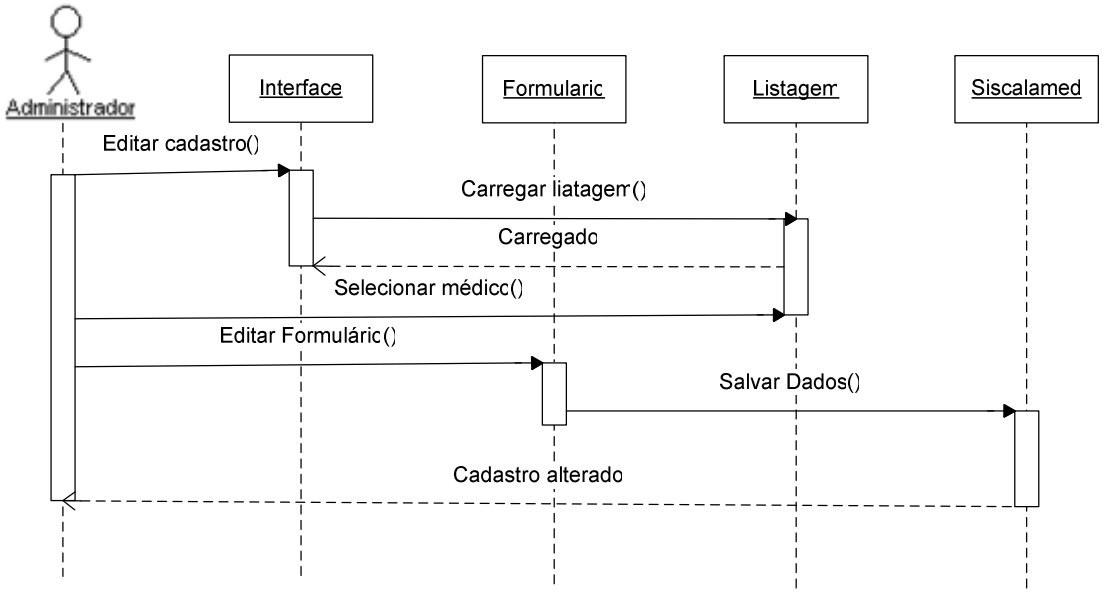


Figura 24 - Diagrama de sequência de edição de cadastro médico

c) editar contrato

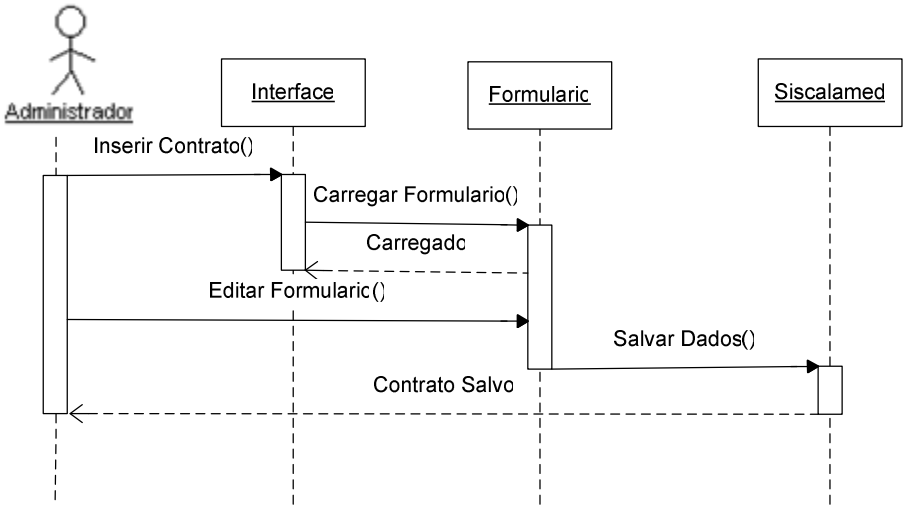


Figura 25 - Diagrama de sequência para inserção de novo contrato

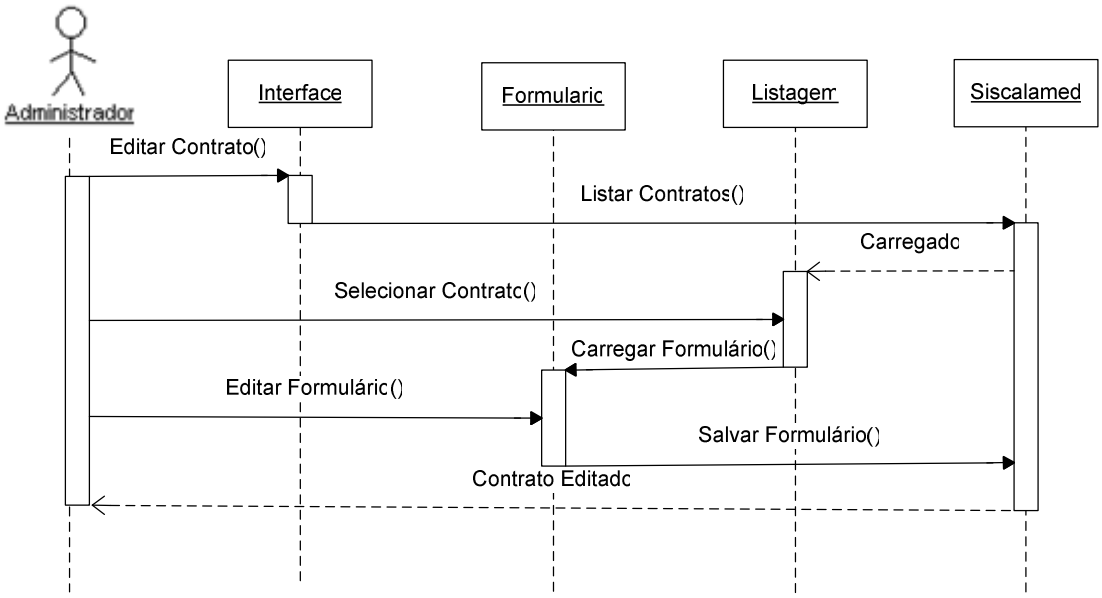


Figura 26 - Diagrama de sequência de edição de contrato

d) editar vagas

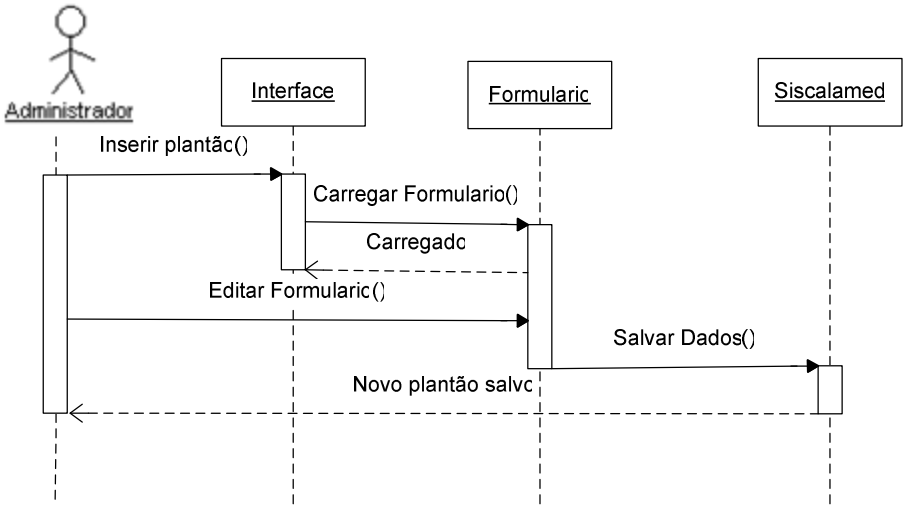


Figura 27 - Diagrama de sequência para inserir nova vaga

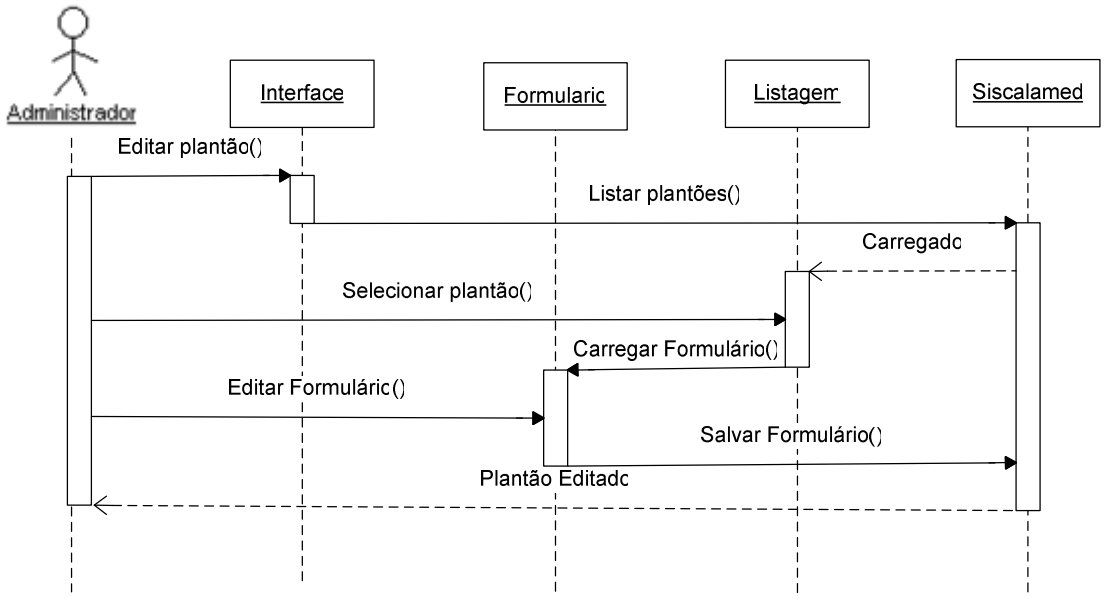


Figura 28 - Diagrama de sequência para edição das vagas

e) reservar vagas

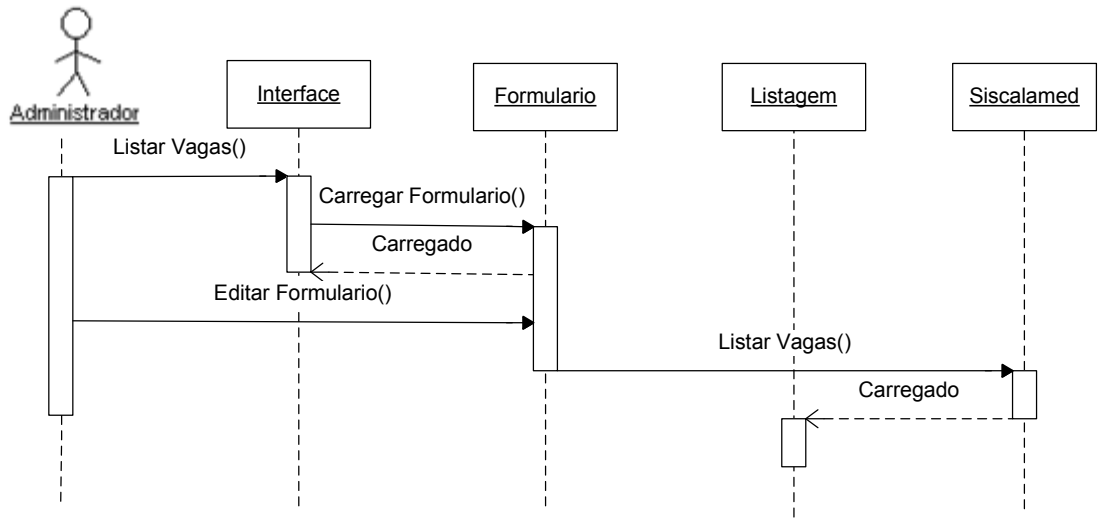


Figura 29 - Diagrama de sequência para listagem de vagas

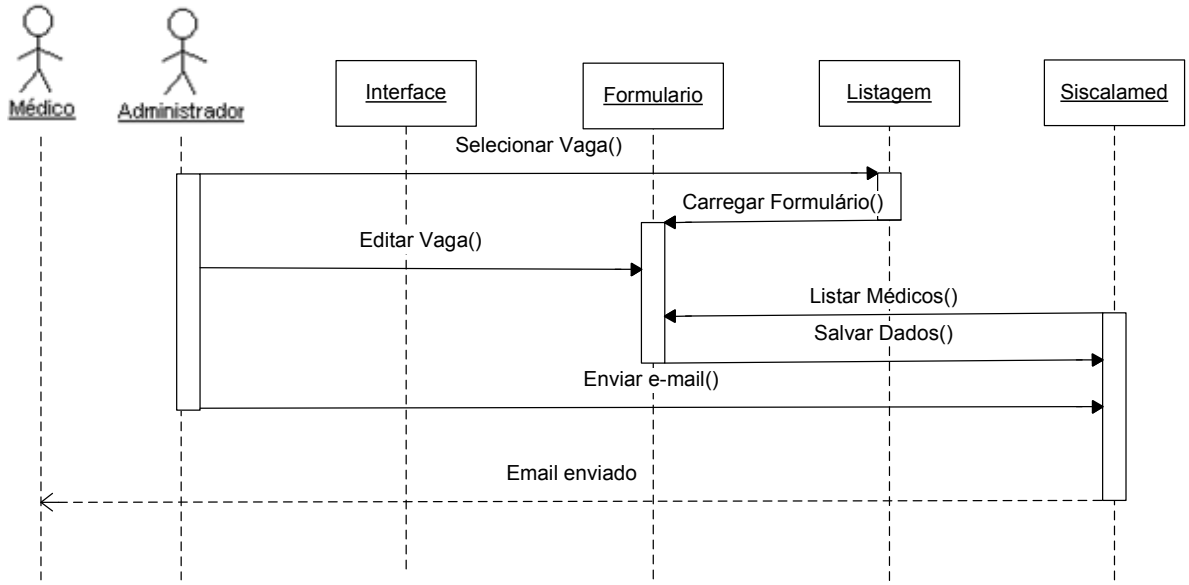


Figura 30 - Diagrama de sequência para edição e reserva de vagas

f) gerar manifesto e imprimir manifesto

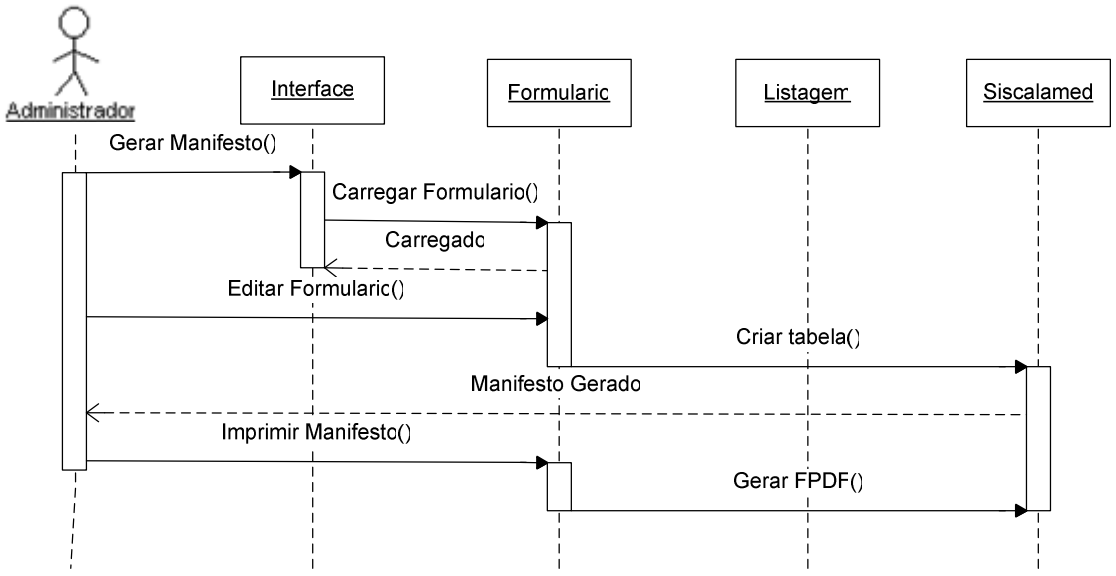


Figura 31 - Diagrama de sequência de geração e impressão de manifesto

g) imprimir escalas

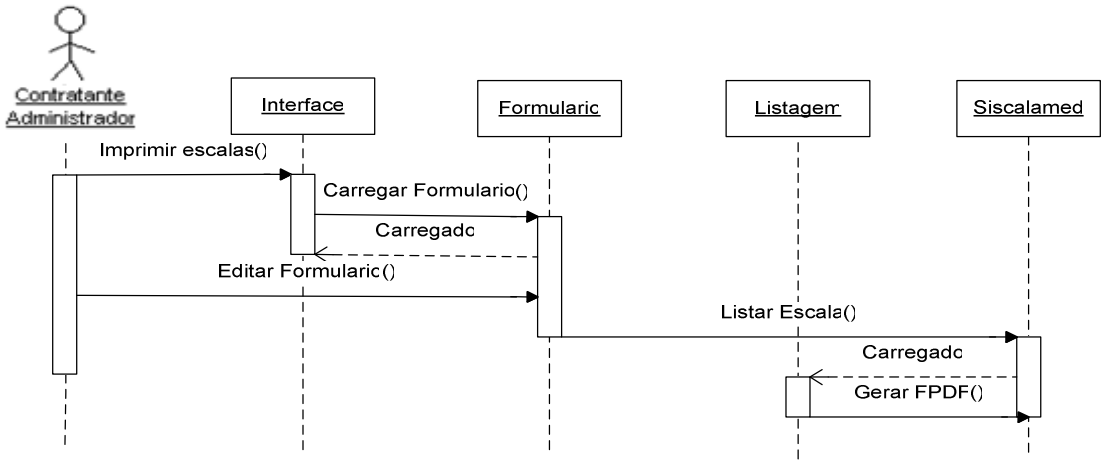


Figura 32 - Diagrama de sequência para impressão de escalas

3.6.5.4 Implementação

a) Cadastrar médico

O gestor tem a opção de efetuar cadastro de novos médicos, conforme a Figura 33.

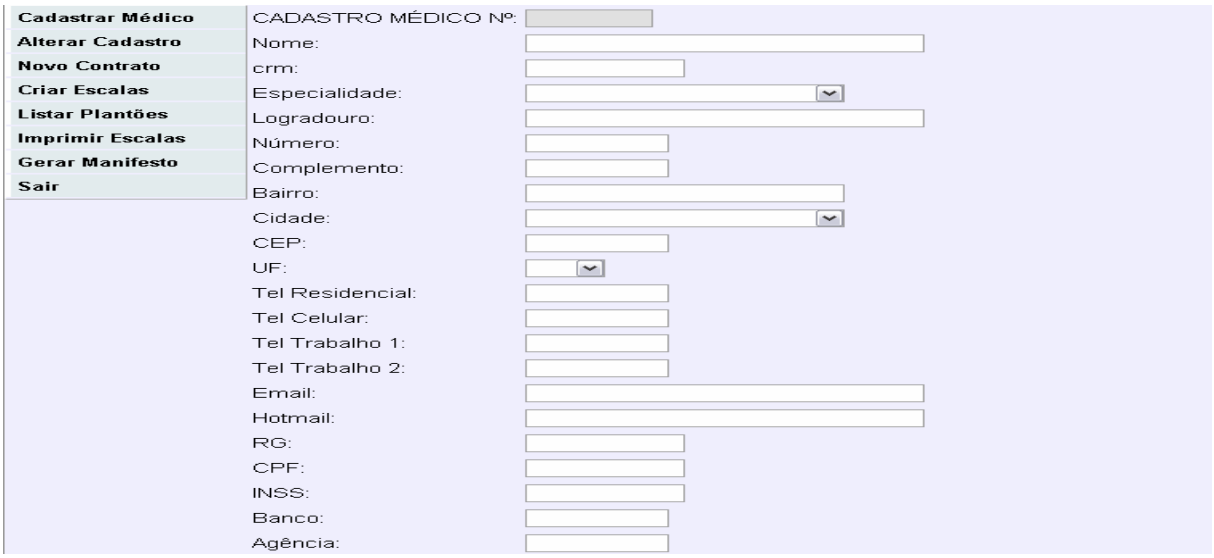


Figura 33 - *Interface* para cadastro médico

b) edição de cadastro

Na Figura 34, o gestor tem a possibilidade de selecionar em uma lista todos os médicos cadastrados, editar os dados cadastrais ou excluir o médico da lista.

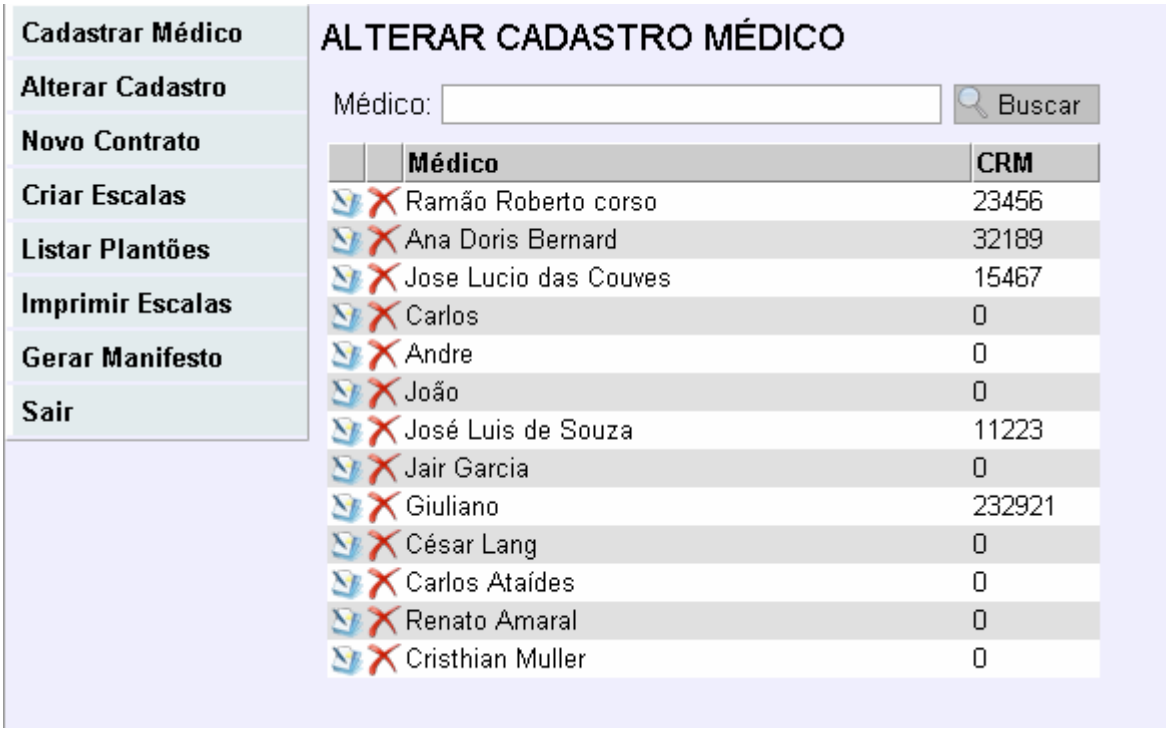


Figura 34 - *Interface* para seleção de cadastro para edição

A interface para edição do cadastro é semelhante às Figuras 18 e 19.

c) editar Contrato

Na Figura 35 o gestor tem a possibilidade de inserir um novo contrato, inserindo a especialidade, o número de horas contratadas e o valor/hora cobrado. Caso seja um novo contratante, o gestor pode adicioná-lo, como demonstrado na Figura 36.

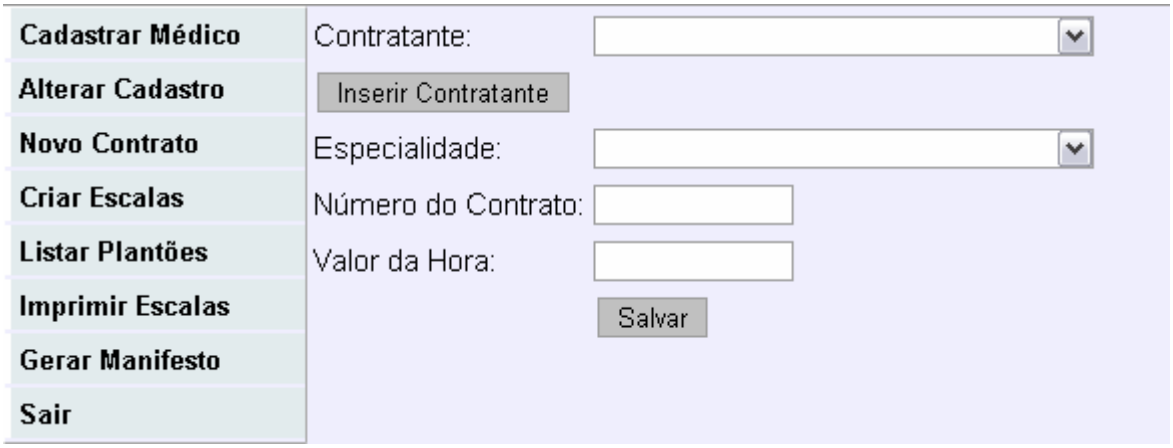


Figura 35 - *Interface* para inserção de novo contrato

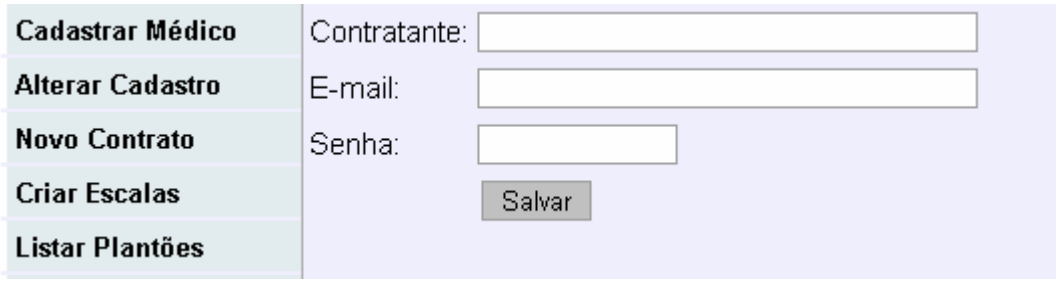


Figura 36 - *Interface* para inserção de novo contratante

d) editar vagas

A Figura 37 representa a *interface* onde o gestor cria as novas escalas, inserindo todas as vagas de trabalho de acordo com seus atributos. A medida que cada vaga é inserida, esta aparece em uma listagem.

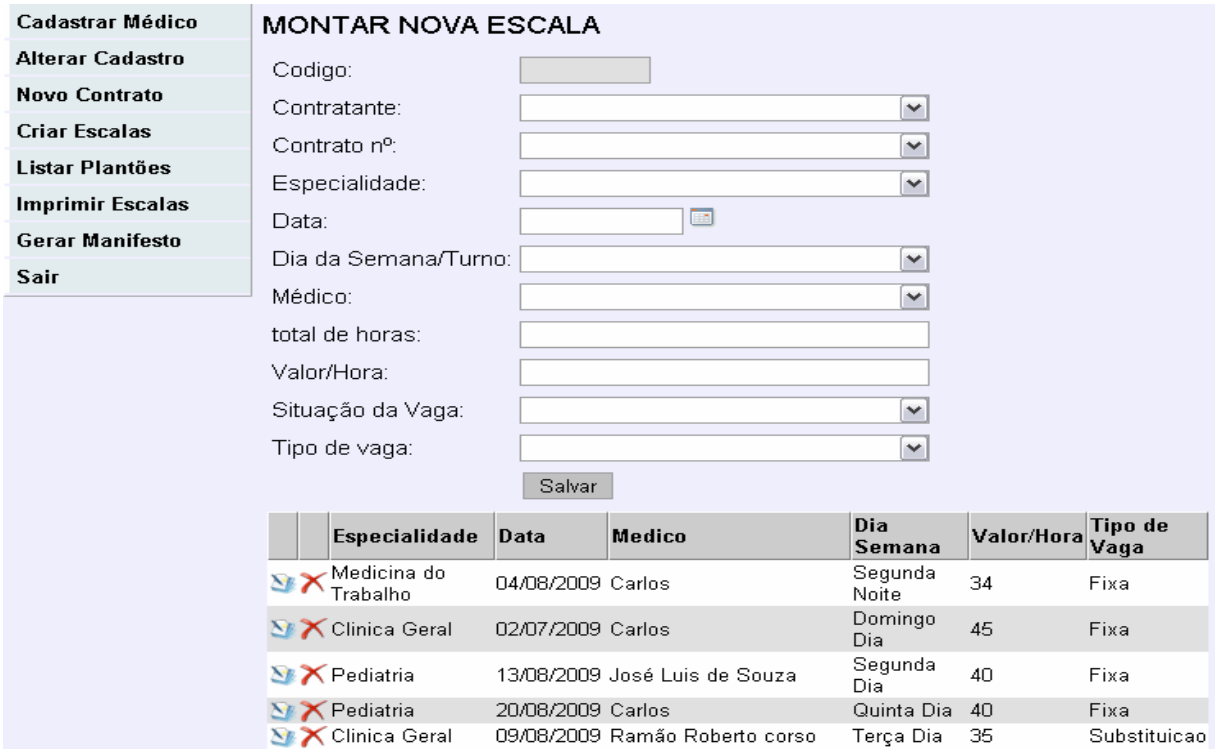


Figura 37 - *Interface* para criação de novas vagas

Se o gestor deseja editar uma determinada vaga, basta selecioná-la na listagem, e editá-la conforme alguma alteração em seus atributos, conforme a Figura 38.

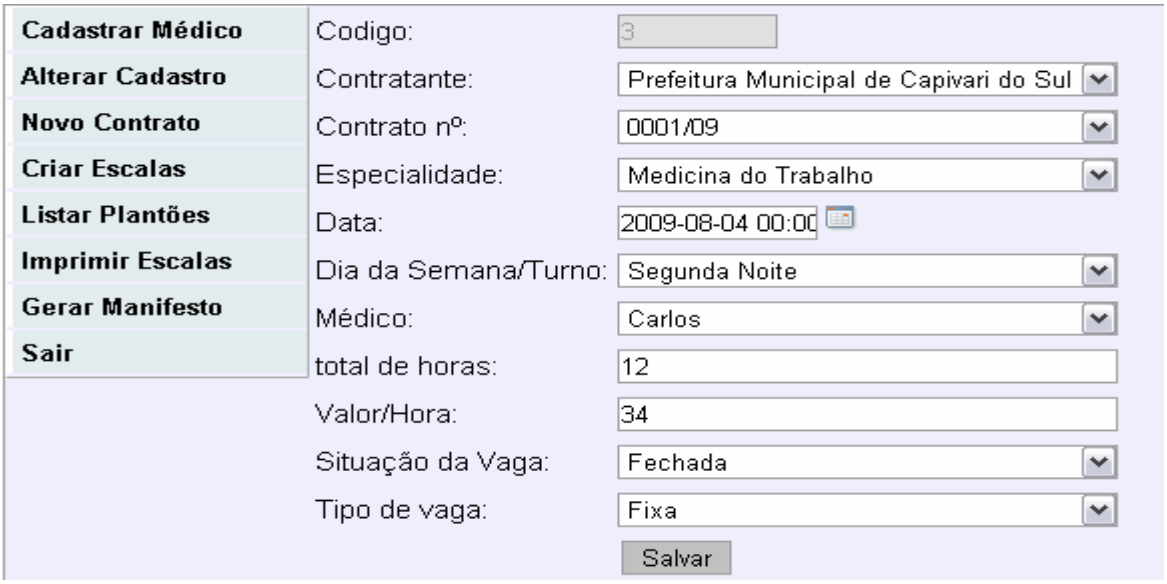


Figura 38 - *Interface* para edição das vagas

e) reservar vagas

O gestor tem a possibilidade de listar as vagas que estejam “reservadas” ou “abertas”, para efetuar a seleção dos médicos, conforme Figura 39.

Figura 39 - *Interface* para seleção de vagas abertas

O gestor seleciona a vaga que deseja reservar, e o sistema abrirá uma *interface* para edição da vaga (Figura 40), informando no campo “médico” somente aqueles com disponibilidades de horários compatíveis com a vaga. Antes de salvar, o sistema ainda avalia se o médico selecionado não superou um limite de horas/mês previamente definido. Caso o gestor deseje selecionar outro médico não apresentado na listagem, basta clicar em “ignorar disponibilidades”. O sistema ainda envia um *e-mail* ao médico selecionado, informando a vaga reservada.

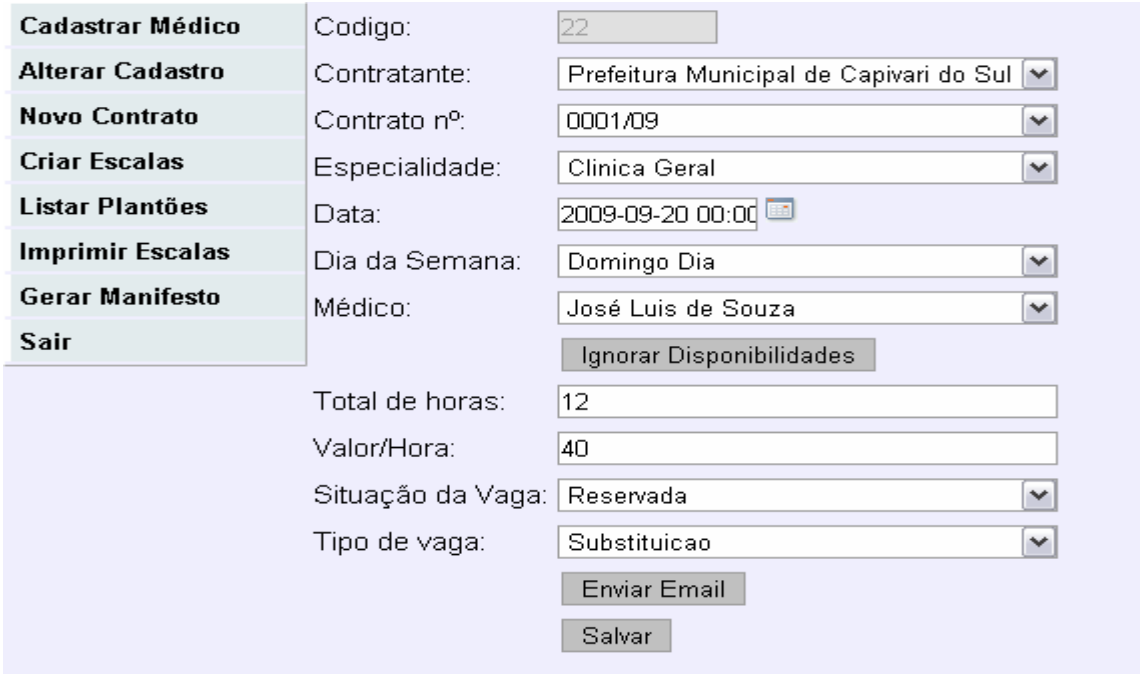


Figura 40 - *Interface* para seleção de médico e reserva de vaga

f) gerar manifesto

O sistema oferece a possibilidade ao gestor de efetuar o manifesto de horas consumidas por mês, selecionando o contratante e o período, conforme a Figura 41.

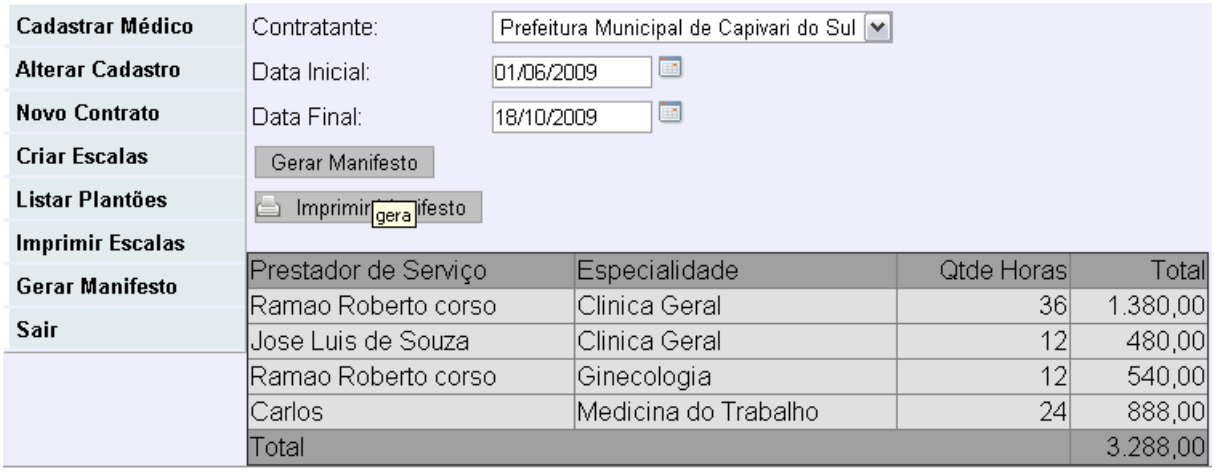
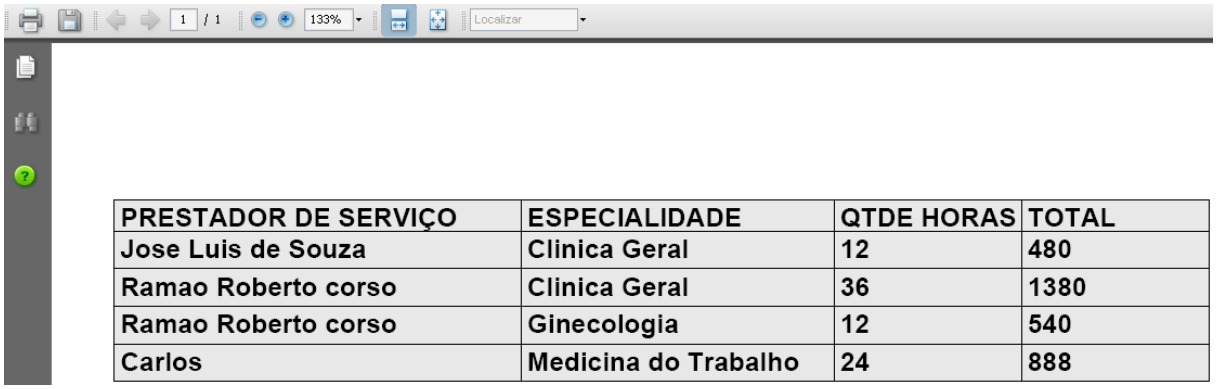


Figura 41 - *Interface* para geração de manifesto

g) imprimir manifesto

Caso o gestor tenha interesse, é possível a impressão deste manifesto, gerando um arquivo PDF (Figura 42).



PRESTADOR DE SERVIÇO	ESPECIALIDADE	QTDE HORAS	TOTAL
Jose Luis de Souza	Clinica Geral	12	480
Ramao Roberto corso	Clinica Geral	36	1380
Ramao Roberto corso	Ginecologia	12	540
Carlos	Medicina do Trabalho	24	888

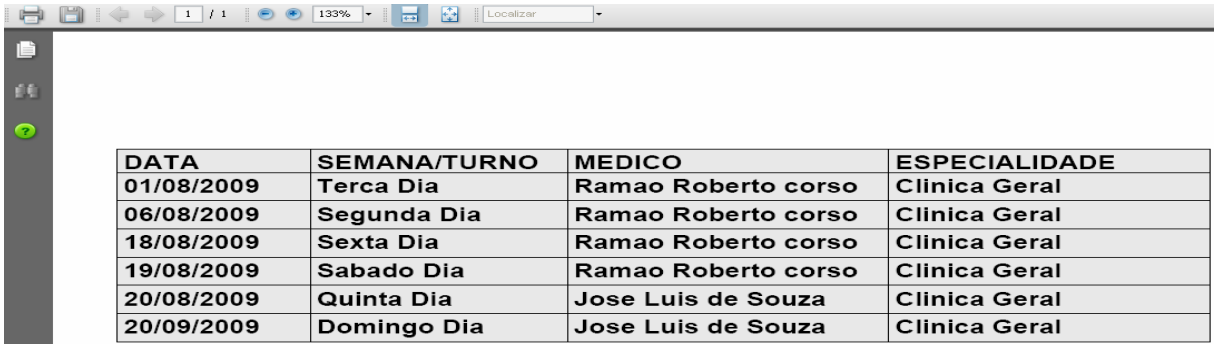
Figura 42 - Geração de PDF para impressão

h) imprimir escalas

O gestor tem a possibilidade de imprimir as escalas de trabalho, selecionando o período, o contratante e a especialidade selecionada. (Figura 43).

Figura 43 - *Interface* para geração de escalas

Para impressão, basta clicar na impressora existente na listagem, e um arquivo PDF será gerado (Figura 44).



DATA	SEMANA/TURNO	MEDICO	ESPECIALIDADE
01/08/2009	Terca Dia	Ramao Roberto corso	Clinica Geral
06/08/2009	Segunda Dia	Ramao Roberto corso	Clinica Geral
18/08/2009	Sexta Dia	Ramao Roberto corso	Clinica Geral
19/08/2009	Sabado Dia	Ramao Roberto corso	Clinica Geral
20/08/2009	Quinta Dia	Jose Luis de Souza	Clinica Geral
20/09/2009	Domingo Dia	Jose Luis de Souza	Clinica Geral

Figura 44 - Geração de PDF para impressão

3.6.6 Módulo contratante

Este módulo permite que o contratante de serviços consulte como está distribuído o seu consumo de horas contratadas e também que possa visualizar quem são os prestadores de serviço escalados para cada determinada especialidade.

3.6.6.1 Diagrama de classes

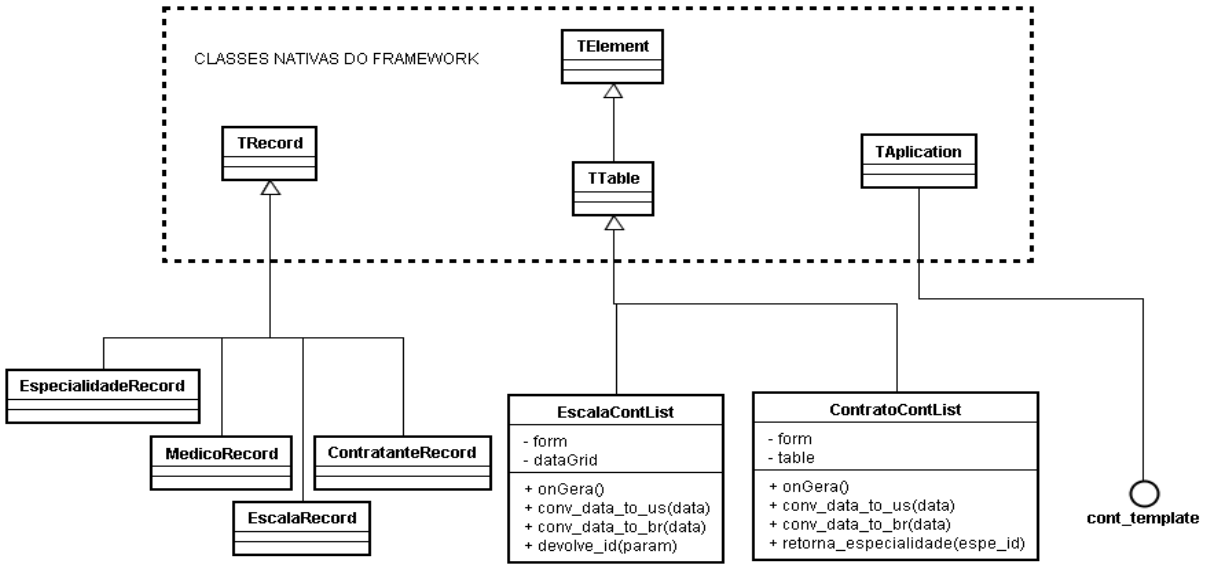


Figura 45 - Diagrama de classes do módulo contratante

3.6.6.2 Descrição dos casos de uso

a) consultar contrato

Nome: Consultar Contrato	**Versão 1.0**
Descritivo: Contratante pode verificar consumo de horas	
Atores envolvidos: Contratante	
Cenário: 1. Contratante tem acesso somente às especialidades por ele contratadas. 2. Contratante deve selecionar a especialidade e o período que deseja consultar. 3. Sistema deve gerar tabela informando o consumo total no período por especialidade. 4. Sistema deve informar o saldo de horas dentro do período, comparando com o número de horas em contrato.	
Dados: Contratante; especialidade; horas_cont; total_consumido; saldo	

Quadro 12 - Descrição do caso de uso de consulta a contrato

b) visualizar escalas

Nome: Visualizar Escalas	**Versão 1.0**
Descritivo: Visualizar médicos escalados para plantão	
Atores envolvidos: Contratante	
Cenário: 1. Sistema deve permitir que ator verifique quem são os médicos escalados para determinado período. 2. Contratante define período e especialidade a consultar. 3. Sistema seleciona as vagas dentro do determinado período, especialidade e contratante, e gera uma listagem. 4. Contratante não edita as vagas, somente visualiza.	
Dados: Data_inicial; data_final; especialidade; data; dia_semana; medico; turno	

Quadro 13 - Descrição do caso de uso de visualização de escalas

3.6.6.3 Diagramas de Sequência

a) consultar contrato

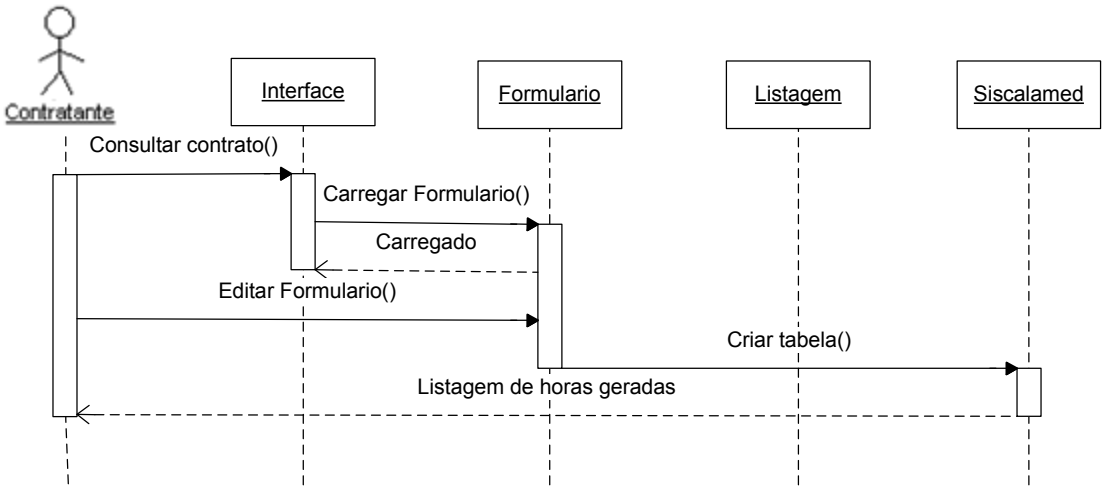


Figura 46 - Diagrama de sequência de consulta à contratos

b) visualizar escalas

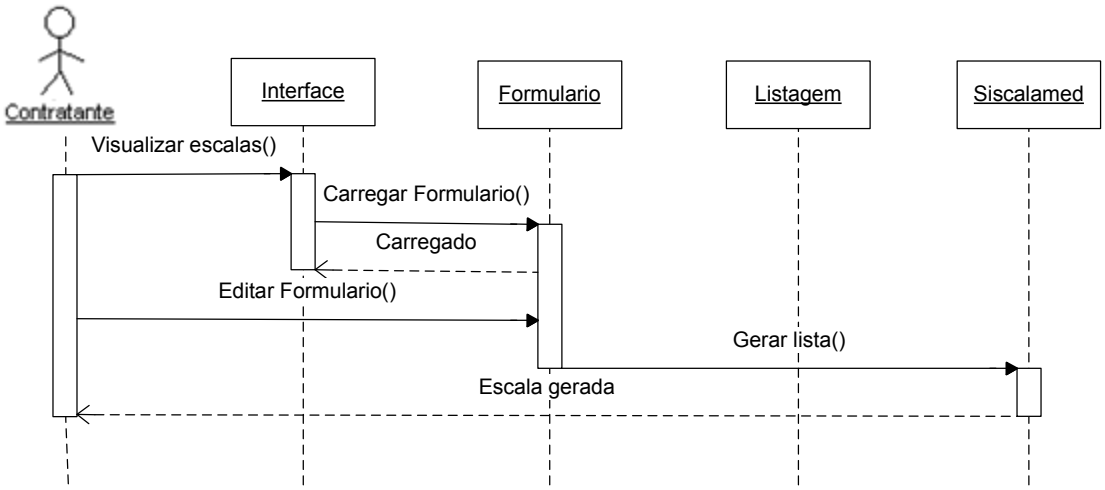


Figura 47 - Diagrama de sequência de visualização de escalas

3.6.6.4 Implementação

a) consultar contrato

O contratante tem a opção de efetuar uma consulta as suas especialidades contratadas, e verificar qual o consumo durante um determinado período, informando o saldo, conforme Figura 48.

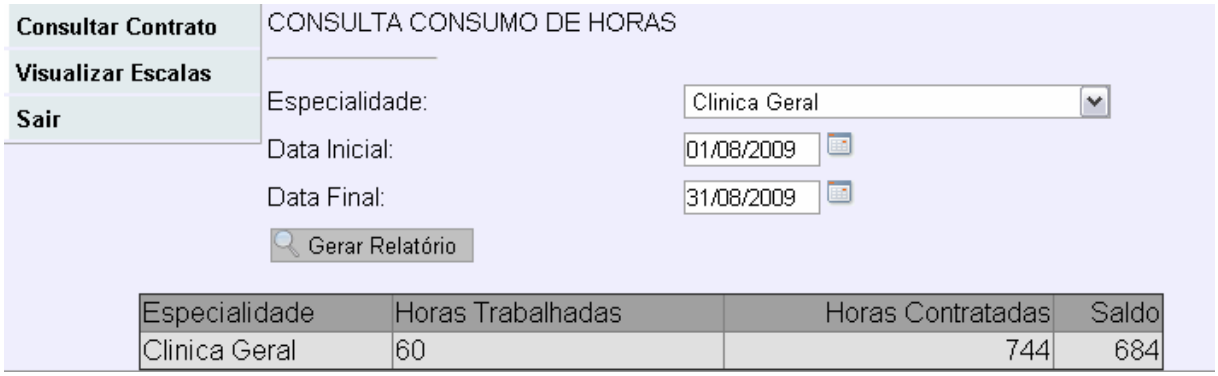


Figura 48 - *Interface* de consulta de contratos

b) visualizar escalas

O contratante pode acessar o sistema a qualquer momento, e visualizar quem serão os prestadores de serviços escalados para determinada vaga (Figura 49).

Figura 49 - *Interface* de visualização de escalas

3.7 Estrutura da aplicação

Para o desenvolvimento desta aplicação, utilizou-se um *framework* criado para fins educativos pela Adianti *Soluctions*. Sua literatura pode ser encontrada no livro PHP – Programando com Orientação a Objetos, de autoria de Pablo Dall'Oglio, e seus arquivos de código podem ser acessados pelo site da Editora. Este *framework* possui padrões de desenvolvimento e classes próprias para manipulações de dados, que foram adaptadas para uma melhor funcionalidade ao sistema proposto.

3.7.1 Manipulação de dados

Existem vários mecanismos de armazenamento de dados, e diferentes tecnologias e sistemas para acesso. Em um sistema complexo, dados podem estar armazenados em diferentes bancos de dados, os quais exigem métodos diferentes de acesso a seus serviços e também para a resolução dos problemas em questão.

O sistema desenvolvido apresentou soluções para acesso a diferentes bancos de dados, e também formas de comunicação e relacionamento entre classes e objetos.

3.7.1.1 PDO: PHP Data Objects

A PDO foi criada para a unificação do acesso a diferentes bancos de dados, unificando as chamadas de métodos, direcionando-os para as extensões correspondentes. O PDO cria uma API[37] limpa e consistente, na qual altera-se apenas a *string* [38]de conexão, para que seja possível conectar a bancos diferentes.

[37] API - Interface de Programação de Aplicativos é um conjunto de rotinas e padrões estabelecidos por um *software* para a utilização das suas funcionalidades por programas aplicativos que não querem envolver-se em detalhes da implementação do *software*, mas apenas usar seus serviços.

[38] *String* - Seqüência ordenada de caracteres (símbolos) escolhidos a partir de um conjunto pré-determinado.

3.7.1.2 Design Pattern

a) *Query Object* – No sistema, a classe *TSqlInstruction* representa uma forma abstrata de instruções enviadas ao banco de dados.

b) *Composite Pattern* – No sistema desenvolvido foi útil compondo diversas expressões de filtragem, através da classe *TCritéria*, adicionando objetos do tipo *TFilter*.

c) *Factory Pattern* – No sistema desenvolvido, a *factory* instancia um objeto PDO, na classe *TConection*, de acordo com as informações de conexão (*drivers*, usuários e senha). Também cria-se o arquivo *my_siscalamed.ini*, que contém as mesmas informações de conexão, o que desobriga a aplicação de conhecer qual banco de dados está sendo utilizado, e pode-se fazer qualquer alteração a qualquer momento, apenas modificando o arquivo ".ini".

3.7.1.3 Controle de transações

Transações garantem a ocorrência de propriedades como atomicidade, consistência, isolamento e durabilidade, no desenvolvimento de aplicações de negócio quando se processa ou armazena dados, independentemente do banco de dados, após sucessivas interações. Também permite que realize-se inúmeras operações no banco, aproveitando a mesma conexão. Para tornar a transação visível durante toda a aplicação, foi criada uma classe para manipular transações chamada *TTransaction.*

3.7.2 Mapeamento Objeto-Relacional

No mapeamento objeto-relacional foi empregado o uso de *gateways* (sub-seção 2.6.6.3), para uma manipulação mais transparente das informações do modelo de negócios contidas no banco.

a) *Active Record*: Na aplicação desenvolvida, a classe TRecord implementa este padrão.
b) *Repository*: No sistema desenvolvido, a classe TRepository implementa todos os métodos necessários para a manipulação das coleções de objetos.

3.7.3 Apresentação e controle

Para deixar o código “mais limpo”, ou menos confuso na leitura, foram criadas classes para exibir tabelas, painéis, janelas, textos, componentes de controle e caixas de diálogo na forma orientada a objetos, com o intuito de substituir as *tags*[39] presentes no código HTML, construindo uma página HTML por meio de objetos.

3.7.3.1 Componentes

a) *TElement* – Manipula as *tags* HTML;
b) *TStyle* – Manipula estilos css;
c) *TImage* – Classe para exibição de imagens;
d) *TParagraph* – Classe para exibir textos.

3.7.3.2 Contêineres

a) *TTable* – Classe que representa as tabelas;
b) *TPanel* – Cria o estilo do painel (altura, largura, borda, cor de fundo).

3.7.3.3 Page Controller

Controla o fluxo de execução, ou seja, é um objeto que interpreta uma requisição de uma determinada ação dentro do sistema e decide qual o método a ser executado. No sistema, são implementadas as seguintes classes de controle:

a) *TPage* - Encapsula todos os componentes visuais da página;
b) *TAction* – Transporta as ações de uma página para outra;
c) *TMessage* – Exibe informações ao usuário.

3.7.4 Formulários e Listagens

O sistema desenvolvido utiliza basicamente formulários e listagens. Os formulários são utilizados para entrada de dados na aplicação, e as listagens para

[39] *tags* - São rótulos usados para informar ao navegador como deve ser apresentado o *website*.

exibir os dados da aplicação. Estes componentes foram implementados na forma orientada a objetos.

3.7.4.1 Formulários

Um formulário é um conjunto de campos ajustados de forma agrupada, utilizado para que o usuário o preencha com informações úteis à aplicação. A classe que montará o formulário no sistema é a *TForm*, e a classe *TField* contém os métodos básicos de um campo do formulário, como alterar seu nome, valor ou tamanho.

As classes de cada elemento utilizados nos formulários do sistema foram descritas e apresentadas no anexo I:

a) *TEntry* – Representa os campos de entrada de dados;
b) *TPassword* – Representa os campos de senha;
c) *THidden* – Representa um campo escondido;
d) *TCombo* – Representa um lista de seleção;
e) *TCheckGroup* – Representa um grupo de campos de checagem.

3.7.4.2 Listagens

As listagens são utilizadas para apresentar um conjunto de registros de um banco de dados. No sistema, foi criada uma classe que cria listagens utilizando uma estrutura orientada a objetos, chamada *TDataGrid*. Esta classe contém colunas para ações na listagem, como editar, excluir, além da possibilidade de adicionar novas colunas.

3.7.5 Estrutura do sistema

Como o PHP ainda não suporta o conceito de pacotes, as classes são organizadas em conjunto, conforme seus objetivos, e salvas dentro de pastas no sistema de arquivos.

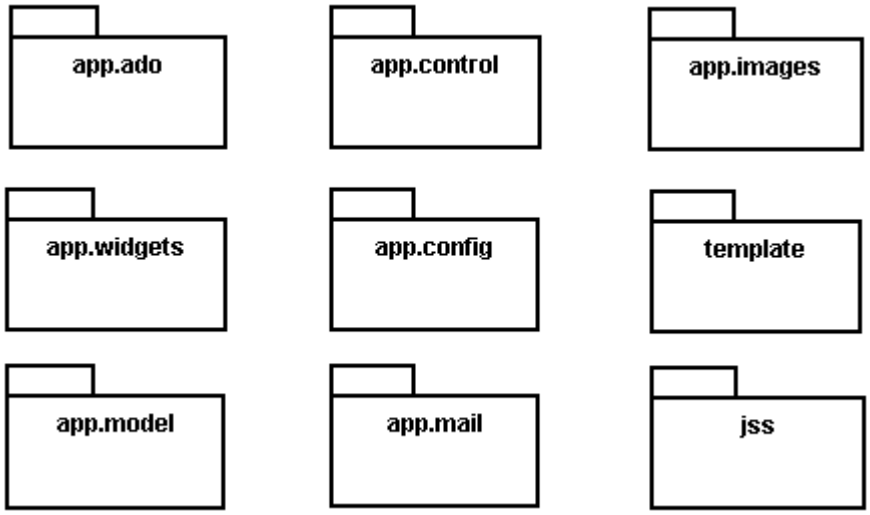


Figura 50: Organização das pastas de arquivos do sistema.

a) app.ado – Contém as classes de controle de transações;
b) app.widgets – Contém as classes de apresentação e controle;
c) app.model – Contém as classes de persistência;
d) app.control – Contém as classes de aplicação;
e) app.config – Contém as *strings* de conexão ao banco de dados;
f) app.mail – Contém arquivos e classe para utilização de correio eletrônico;
g) app.images – Contém as imagens utilizadas no sistema;
h) template – Contém os arquivos de código HTML;
i) jss – Contém a classe para ativação do calendário;

3.8 Testes

Para que o processo de testes avalie todos os requisitos do sistema, foi elaborado um plano de teste.

3.8.1 *Plano de teste*

O plano de teste descreve as estratégias usadas para a realização dos casos de teste, o tipo de teste que será realizado, a técnica aplicada e a descrição dos casos de teste, visando avaliar a qualidade dos componentes implementados e

verificar se a sua funcionalidade está correta, além de verificar se qualidades importantes e desejáveis foram alcançadas.

3.8.1.1 Escopo

Aplicar um plano de testes no sistema produzido, o Siscalamed, descrevendo estratégias para a realização de casos de teste que visem garantir a qualidade e a funcionalidade do *software* implementado.

3.8.1.2 Tipo de teste aplicado

Existem muitos tipos de testes, onde cada um focaliza um objetivo específico ou uma característica do sistema. Como o modelo de processo de *software* definido (RAD), enfatiza um ciclo de desenvolvimento curto, onde o desenvolvedor pode analisar e desenvolver o *software* em módulos bem definidos, optou-se apenas pelos testes de função (sub-seção 2.7.7.1), onde é verificado o funcionamento de cada caso de uso definido na análise. Testando a funcionalidade dos casos de uso, paralelamente serão avaliados outros tipos de testes, como os de integração, configuração e instalação, por exemplo.

3.8.1.3 Técnica da caixa-preta

Esta é uma técnica aplicada em todos os tipos ou fases de testes do sistema, e avalia o comportamento externo do *software*, isto é, não avaliando linha de código do sistema, apenas atribuindo dados na entrada, e comparando o resultado de saída a uma lista de resultados previamente esperados. Assim, são elaborados casos de teste que atribuem situações de entrada e o comportamento esperado do sistema. Quanto mais entradas são especificadas, mais eficientes serão os testes.

Apesar de este tipo de técnica ser insuficiente para um sistema de grande porte ou onde os riscos são muito grandes, no sistema proposto adapta-se perfeitamente, analisando os requisitos propostos na implementação.

3.8.1.4 Localização e aplicação dos casos de teste

Os casos de teste foram divididos nos 4 módulos do sistema, e são definidos pelos casos de uso descritos no ítem 3.6.1.2.

A aplicação dos casos de teste não utilizou uma ferramenta específica. Serão listadas os procedimentos para inserção dos dados para cada caso de teste, e os resultados esperados.

3.8.1.5 Definição dos casos de teste

Na definição dos casos de teste são descritas as condições, as entradas, as ações e os resultados esperados neste processo.

Cada caso de uso foi subdividido pelo módulo do sistema.

a) módulo *Login*

Objetivo: Testar os requisitos de *login* do sistema	
Dados de Teste: e-mail, senha	
Etapas:	**Resposta esperada:**
**** Acessar***	
1. Clicar em “Acessar”	Exibir formulário
2. Digitar e-mail	
3. Digitar a senha	
4. Clicar em “Acessar”	Direcionar para área do usuário
**** Recuperar Senha***	
1. Clicar em “Recuperar Senha”	Exibir formulário
2. Digitar e-mail	
3. Clicar em “Enviar”	Enviar senha via e-mail
**** Cadastrar-se***	
1. Clicar em “Cadastrar-se”	Abrir formulário
2. Editar formulário	
3. Clicar em “Salvar”	Dados armazenados com sucesso
	Enviar e-mail ao gestor
Testes:	**Resposta esperada:**
Inserir e-mail ou senha inválidos	Usuário ou senha inválidos
Inserir senha com menos de 6 dígitos	Senha inválida
Não inserir CRM	Campo CRM não pode ficar em branco
RG com caracteres que não sejam números	Inserir somente números no campo RG
CPF com caracteres que não sejam números	Inserir somente números no campo CPF
CPF maior ou menor que 11 dígitos	CPF inválido

Quadro 14 - Caso de teste de *login*

b) módulo médico

Objetivo: Testar o cadastro de dados médicos no sistema	
Dados de Teste: rg, cpf, e-mail, senha, crm	
Etapas:	**Resposta esperada:**
**** Editar Cadastro***	
1. Clicar em “Editar Cadastro”	Exibir formulário
2. Editar Dados	
3. Clicar em “Salvar”	Dados armazenados com sucesso
Testes:	**Resposta esperada:**
Inserir e-mail sem arroba	Email inválido
Inserir senha com menos de 6 dígitos	Senha inválida
Não inserir CRM	Campo CRM não pode ficar em branco
RG com caracteres que não sejam números	Inserir somente números no campo RG
CPF com caracteres que não sejam números	Inserir somente números no campo CPF
CPF maior ou menor que 11 dígitos	CPF inválido

Quadro 15 - Caso de teste de edição de cadastro

Objetivo: Testar a visualização e reserva de plantões	
Dados de Teste: data_inicial, data_final	
Etapas:	**Resposta esperada:**
**** Visualizar Vagas***	
1. Clicar em “Visualizar Vagas”	Exibir formulário
2. Entrar com data Inicial	
3. Entrar com data final	
4. Clicar em “Gerar Relatório”	Exibir relatório de plantões
	Exibir relatório em branco se não houver plantões que correspondam a consulta
**** Reservar vaga***	
1. Clicar em “Reservar Vaga”	Abrir formulário da vaga específica
2. Clicar em enviar e-mail	Enviar e-mail para o gestor
3. Clicar em “Reservar Vaga”	Vaga reservada com sucesso
Testes:	**Resposta esperada:**
Inserir período incorreto	Relatório vazio
Reservar vaga	“Situação da vaga” definida como “reservada”
	Impedida a edição

Quadro 16 - Caso de teste de visualização e reserva de vagas

c) módulo gestor

Objetivo: Testar a edição e cadastro médico	
Dados de Teste: rg, cpf, e-mail, senha, crm	
Etapas:	**Resposta esperada:**
**** Cadastrar-se***	
1. Clicar em “Cadastrar-se”	Abrir formulário
2. Editar formulário	
3. Clicar em “Salvar”	Dados armazenados com sucesso
****Alterar Cadastro***	
1. Clicar em “Alterar Cadastro”	Abrir formulário
2. Editar nome do médico no campo de pesquisa	
3. Clicar em Buscar	Listar médico no relatório
4. Clicar em “Editar Médico”	Abrir formulário com dados do médico correspondente
5. Editar cadastro	
6. Clicar em “Salvar”	Dados armazenados com sucesso
Testes:	**Resposta esperada:**
Inserir e-mail ou senha inválidos	Usuário ou senha inválidos
Inserir senha com menos de 6 dígitos	Senha inválida
Não inserir CRM	Campo CRM não pode ficar em branco
RG com caracteres que não sejam números	Inserir somente números no campo RG
CPF com caracteres que não sejam números	Inserir somente números no campo CPF
CPF maior ou menor que 11 dígitos	CPF inválido

Quadro 17 - Caso de teste de edição e cadastro médico

Objetivo: Testar de inserção de novo contratante e contrato	
Dados de Teste: contratante, e-mail, senha, numero_horas, valor_hora	
Etapas:	**Resposta esperada:**
**** Novo Contrato***	
1. Clicar em “Novo Contrato”	Abrir formulário
2. Selecionar campo “Contratante”	
3. Selecionar campo “Especialidade”	
4. Editar total horas e valor hora	
1. Clicar em “Salvar”	Dados armazenados com sucesso
**** Inserir Contratante***	
1. Clicar em “Novo Contrato”	Abrir Formulário
4. Clicar em “Inserir Contratante”	Abrir Formulário
5. Editar cadastro	
6. Clicar em “Salvar”	Dados armazenados com sucesso
Testes:	**Resposta esperada:**
Inserir e-mail ou senha inválidos	Usuário ou senha inválidos
Inserir senha com menos de 6 dígitos	Senha inválida
Não inserir valor hora	Dados inválidos
Não inserir número de horas	Dados inválidos

Quadro 18 - Caso de teste de inserção de novo contrato

Objetivo: Testar a criação e edição das escalas	
Dados de Teste: código, contratante, especialidade, data, dia_semana, total_horas, valor_hora, situação_vaga, tipo_vaga	
Etapas:	**Resposta esperada:**
**** Criar Escala***	
1. Clicar em “Criar Escalas”	Abrir formulário
2. Editar formulário	
3. Clicar em “Salvar”	Dados armazenados com sucesso
****Editar Vaga***	
1. Clicar em “Alterar Editar Vaga”	Abrir formulário com dados da vaga correspondente
2. Editar dados da vaga	
3. Clicar em “Salvar”	Dados armazenados com sucesso
Testes:	**Resposta esperada:**
Não inserir valor hora	Dados inválidos
Não inserir número de horas	Dados inválidos
Campo médico = vazio	
Situação da vaga = fechada	Dados inválidos
Campo médico = vazio	
Situação da vaga = reservada	Dados inválidos

Quadro 19 – Caso de teste de criação e edição de escalas

Objetivo: Testar a listagem de plantões em aberto e a seleção do médico apropriado	
Dados de Teste: data_inicial, data_final, código, contratante, especialidade, data, dia_semana, total_horas, valor_hora, situação_vaga, tipo_vaga	
Etapas:	**Resposta esperada:**
**** Listar plantões***	
1. Clicar em “Listar Plantões”	Abrir formulário
2. Editar data inicial e data final	
3. Clicar em “Gerar Relatório”	Gerar relatório de vagas com situação “reservada” ou “aberta”
****Editar Vaga***	
1. Clicar em “Editar Vaga”	Abrir formulário com dados da vaga correspondente
2. Editar dados da vaga	
4. Clicar em “Enviar Email”	Enviar e-mail informando dados da vaga para médico selecionado
6. Clicar em “Salvar”	Dados armazenados com sucesso
Testes:	**Resposta esperada:**
Não inserir valor hora	Dados inválidos
Não inserir número de horas	Dados inválidos
Campo médico = vazio	
Situação da vaga = fechada	Dados inválidos
Campo médico = vazio	
Situação da vaga = reservada	Dados inválidos
Clicando no ícone “Editar Vaga”	Retorna no campo somente nomes dos médicos com disponibilidade para a vaga
Clicando no botão “Ignorar Disponibilidade”	Carrega no campo todos os médicos com a especialidade da vaga

Quadro 20 – Caso de teste listagem e edição de vagas em aberto

Objetivo: Testar a listagem e impressão de escalas	
Dados de Teste: data_inicial, data_final, código, contratante, especialidade, data, dia_semana, total_horas, valor_hora, situação_vaga, tipo_vaga	
Etapas:	**Resposta esperada:**
**** Imprimir Escalas***	
1. Clicar em “Imprimir Escalas”	Abrir formulário
2. Editar formulário	
3. Clicar em “Gerar Relatório”	Gerar relatório de escalas referentes ao contratante e a especialidade selecionada
****Imprimir Escala***	
1. Clicar em qualquer dos ícones de Impressão no relatório	Gerar arquivo PDF com a escala de trabalho (necessário que a máquina cliente tenha instalado Adobe Reader)
2. Clicar no ícone “impressora” do relatório	Imprimir a escala
Testes:	**Resposta esperada:**
Não entrar com algum dos campos do relatório	Dados inválidos
Data inicial maior que data final	Período inválido

Quadro 21 – Caso de teste de impressão de escalas

Objetivo: Testar a geração e impressão do manifesto	
Dados de Teste: data_inicial, data_final, código, contratante, especialidade, data, dia_semana, total_horas, valor_hora, situação_vaga, tipo_vaga	
Etapas:	**Resposta esperada:**
**** Listar plantões***	
1. Clicar em “Gerar Manifesto”	Abrir formulário
2. Editar formulário	
3. Clicar em “Gerar Manifesto”	Gerar relatório consumo de horas num determinado período, por contratante, delimitando as especialidades e acumulando as referidas horas por médico, listando o valor total de horas e o valor a ser pago para cada médico
****Imprimir Manifesto***	
1. Clicar em “Imprimir Manifesto”	Gerar arquivo PDF com o manifesto selecionado (necessário que a máquina cliente tenha instalado Adobe Reader)
Testes:	**Resposta esperada:**
Não inserir algum dado no formulário	Dados inválidos
Data inicial maior que data final	Período inválido

Quadro 22 – Caso de teste de geração e impressão de manifesto

d) módulo contratante

Objetivo: Testar a consulta do consumo de horas pelo contratante	
Dados de Teste: data_inicial, data_final, especialidade	
Etapas:	**Resposta esperada:**
**** Consultar Contrato***	
1. Clicar em “Consultar Contrato”	Abrir formulário
2. Editar formulário	
3. Clicar em “Gerar Relatório”	Gerar relatório de consumo de horas num determinado período, por especialidade, apresentando o valor consumido, o valor contratado e o saldo.
Testes:	**Resposta esperada:**
Não inserir algum dado no formulário	Dados inválidos
Data inicial maior que data final	Período inválido
Selecionar especialidade não contratada	Especialidade não contratada no período

Quadro 23 – Caso de teste de consumo de horas

Objetivo: Testar a visualização das escalas de trabalho pelo contratante	
Dados de Teste: data_inicial, data_final, especialidade	
Etapas:	**Resposta esperada:**
**** Consultar Contrato***	
1. Clicar em “Visualizar Escala”	Abrir formulário
2. Editar formulário	
3. Clicar em “Gerar Relatório”	Gerar relatório das escalas de trabalho no determinado período e especialidade escolhidos pelo contratante
Testes:	**Resposta esperada:**
Não inserir algum dado no formulário	Dados inválidos
Data inicial maior que data final	Período inválido
Selecionar especialidade não contratada	Especialidade não contratada no período

Quadro 24 – Caso de teste de visualização de escalas

3.9 Integração

O *software* implementado é um sistema *web*, e momentaneamente não tem integração com algum ERP[40]. Num passo futuro, será projetada a integração com a folha de pagamento da empresa.

[40] ERP – Enterprise Resource Planning. Sistema de computador responsável por integrar todos os departamentos e funções de uma organização.

A integração entre as tecnologias da base de desenvolvimento do sistema ficou facilitada, graças a utilização da ferramenta Wampserver 2.0, que é uma plataforma de desenvolvimento *web* que instala e integra automaticamente o servidor Apache 2, o PHP 5, o banco de dados MySQL, e ainda o PhpMyAdmin, que facilita o gerenciamento do banco de dados.

4 CONCLUSÃO

Este trabalho apresentou o desenvolvimento e a implementação de um sistema *web*, como ferramenta auxiliar em empresas de terceirização de prestação de serviços, no caso específico, a mão de obra médica, para a elaboração e manutenção de escalas de trabalho.

Com este *software*, fica mais fácil ao gestor de escalas o controle dos horários disponíveis de cada prestador de serviço, e também a adequação de cada trabalhador às vagas disponíveis. Também propicia ao gestor um controle das horas contratadas, e a geração do manifesto mensal de horas trabalhadas pelos prestadores de serviço. Aos prestadores de serviço (médicos), ofereceu a possibilidade de efetuarem seu cadastro para a prestação de serviços, e a opção de visualizar uma vaga de trabalho conforme sua disponibilidade, atualizada automaticamente. Ao contratante, permitiu a visualização de suas escalas contratadas, e o consumo de horas/mês.

A escolha da metodologia foi fundamental, pois garantiu a execução de todas as etapas de análise e implementação sem desperdício de tempo e de código, graças a um levantamento de requisitos bem elaborado, e a facilidade que a metodologia RAD, aliada à programação orientada a objetos, permitem na questão de reaproveitamento de código e de um acompanhamento mais constante do usuário a cada requisito implementado. A escolha das tecnologias e ferramentas utilizadas também agilizaram todo o processo, graças a uma extensa documentação existente para consulta. A utilização de *design patterns* para a implementação também foram extremamente úteis, deixando o código mais claro e facilitando eventuais correções que foram necessárias.

O *software* auxiliará a empresa na facilidade da manutenção das escalas, refletindo em economia de mão de obra e custos em comunicação e divulgação. Também será um diferencial competitivo para a empresa na concorrência para novos clientes, porém, nada impede sua utilização em outras empresas de prestação de serviço médico, desde que estas integrem o modelo cooperativista.

A implementação deste sistema serviu também como ensaio e protótipo para um projeto maior, oferecendo um serviço *web* de fechamento de escalas direto aos contratantes de serviço.

5 APERFEIÇOAMENTOS FUTUROS NO SISTEMA

Alguns aperfeiçoamentos são projetados para uma melhor funcionalidade do sistema, e segurança de dados e acessos.

5.1 Melhorias na funcionalidade

Para que o processo de criação de escalas por parte do gestor torne-se ainda mais ágil, pode-se fazer necessária a implementação de uma função onde as vagas definidas como “fixas” sejam geradas automaticamente nos 3 meses seguintes à criação da vaga. Também é possível a criação de uma função, que altere as disponibilidades de um referido médico, quando este for escalado em uma vaga fixa, e esta vaga esteja com a situação “fechada”.

Pode-se fazer necessária a geração de um manifesto com o total de horas e o valor pago pelo contratante por especialidade. A informação dos telefones do referido médico no momento do fechamento de uma vaga também agilizaria este processo.

5.2 Melhorias na segurança

Em um ambiente cliente/servidor as atenções normalmente estão direcionadas para o servidor, onde residem as informações, portanto, tornando-se o foco de ameaças.

Para ter segurança em sessões em que são transmitidas informações confidenciais e secretas, como os formulários de *login*, faz-se necessário o emprego de criptografia de sessão SSL. Para garantir a segurança de senhas nas autenticações, será interessante implementar uma *Digest Autentication*, fazendo com que a senha não fica visível na rede. Utilizando este método, a comunicação entre as partes envolvidas contém um *checksum* incluindo o *username*, a senha, o método HTTP, e um autenticador único (geralmente um número randômico longo), além da URL requisitada.

Para evitar ataques através da incorporação de dados dinâmicos, deve-se implementar a permissão de tipos de dados, invalidando todos os tipos existentes, exceto os permitidos. Também deve-se implementar a validação de todos os dados de entradas nos formulários, garantindo sua integridade.

6 REFERÊNCIAS

BAIERLE, Anelise Weyrich. PLANTÃO TRABALHISTA ON-LINE. **Terceirização. O que é ?**. Disponível em: <http://www.plantaotrabalhista.floripa.com.br/terc.htm#oque>. Acesso em 06 out 2007.

BERNARD, Marcio Roberto. **O cooperativismo: Uma alternativa de trabalho e renda** – O caso Uniplus. Taquara: FACCAT, 2003.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: Guia do usuário**. 3ª ed. Rio de Janeiro, Campus, 2000.

BRAGANÇA, Alexandre. **Desenvolvimento Cliente-Servidor**. Cidade do Porto – Portugal: INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO, 1999.

CARNEIRO, Davi Luan. **Introdução ao Pattern DAO**. Disponível em < http://javafree.uol.com.br/artigo/871452/Introducao-ao-pattern-DAO.html> Acesso em 07 nov 2009.

CDTC. Centro de difusão de tecnologia e conhecimento. **Apache.** Disponível em: <http://www2.obid.senad.gov.br/cadastro/Cursos/Cursos%20CDTC/Apache.pdf>. Acesso em 17 out 2007.

CDTC. Centro de difusão de tecnologia e conhecimento. **MySQL**. Disponível em: <http://www2.obid.senad.gov.br/cadastro/Cursos/Cursos%20CDTC/MySQL.pdf>. Acesso em 17 out 2007.

CRIE SEU WEB SITE. **Tutorial PHP**. Disponível em: <http://www.crieseuwebsite.com/tutorial.php?linguagem=php&pagina=1>. Acesso em 02 nov 2007.

DALL'OGLIO, Pablo. **PHP – Programando com orientação a objetos**. 1ª ed. São Paulo: Novatec Editora, 2007.

DESENVOLVIMENTO FÁCIL. **Principais arquiteturas para aplicações web**. Disponível em: <http://www.desenvolvimentofacil.com.br/pasi/pas01.pdf>. Acesso em 24 nov 2007.

FALBO, Ricardo de Almeida. **Técnicas de Levantamento de Requisitos**. Disponível em < http://www.renatacampos.pro.br/Tecnicas%20de%20Levantamento%20de%20requisitos.pdf>. Acesso em 07 nov 2009.

FALSARELLA, Orandi Mina; CHAVES, Eduardo O. C. **Sistemas de Informação e Sistemas de apoio à decisão**. Disponível em: <http://www.chaves.com.br/TEXTSELF/COMPUT/sad.htm>. Acesso em 12 out 2007.

FORBECK, Vera. UNIBAN. Universidade Bandeirante de São Paulo. **PHP**. Disponível em:<http://www.apostilando.com/download.php?cod=2535&categoria=PHP>.Acesso em 15 nov 2007.

FORTES, José Carlos. FORTES ADVOGADOS ASSOCIADOS. **A prestação de serviços**. Disponível em: <http://www.fortesadvogados.com.br/artigos.view.php?id=533>. Acesso em 06 out 2007.

HEIDRICH, Leonardo. **Paradigmas de Programação** – Imperativo x Orientado a Objetos. São Leopoldo: UNISINOS, 2007.

IBGE. Instituto brasileiro de geografia e estatística. **Pesquisa anual de serviços – 2005.** Disponível em: <http://www.ibge.gov.br/home/estatistica/economia/comercioeservico/pas/pas2005/pas2005.pdf>. Acesso em 06 out 2007.

INFOWESTER. **Banco de dados MySQL e PostgreSQL.** Disponível em: <http://www.infowester.com/postgremysql.php>. Acesso em 13 out 2007.

_____. **Conhecendo o Servidor Apache (HTTP Server Project).** Disponível em: < http://www.infowester.com/servapach.php>. Acesso em 13 out 2007.

_____. **Linguagem JavaScript**. Disponível em: <http://www.infowester.com/lingjavascript.php>. Acesso em 13 out 2007.

INTERLEGIS. Comunidade virtual do poder legislativo**. Lei de responsabilidade fiscal comentada**. Disponível em: <http://www.interlegis.gov.br/fiscalizacao/20030512175516/view>. Acesso em: 06 out 2007.

KRUCHTEN, Philippe. **Introdução ao RUP. Rational Unified Process**. 1ª ed. Rio de Janeiro. Ciência Moderna Ltda, 2003.

MACORATTI, José Carlos. **A Gestão de Requisitos**. Disponível em http://imasters.uol.com.br/artigo/3860/des_de_software/a_gestao_de_requisitos/ . Acesso em 24 out 2009.

MELO, Lafayette B. **Análise e Projetos de Sistemas para Internet**. Disponível em <www.lafa.pro.br/aps/APS-3analisederequisitos.ppt> . Acesso em 07 nov 2009.

OLIVEIRA, Eric C.M. **Introdução a Design Patterns**. Disponível em <http://www.linhadecodigo.com.br/Artigo.aspx?id=345>. Acesso em 07 nov 2009.

OLIVEIRA, Halley Pacheco de. **Documentação do PostgreSQL 8.0.0.** Rio de Janeiro: The PostgreSQL Global Development Group, 2005.

PRESIDÊNCIA DA REPÚBLICA. Casa Civil. Sub-chefia para assuntos jurídicos. **Lei complementar nº 101, de 4 de maio de 2000**. Disponível em: <http://www.planalto.gov.br/ccivil_03/Leis/LCP/Lcp101.htm>. Acesso em 06 Out 2007.

PRESSMAN, Roger S. **Engenharia de Software**. 6ª Ed. São Paulo: McGraw_Hill do Brasil, 2006.

PUC Rio. **Arquitetura Orientada a Serviços**. Disponível em < http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0210486_04_cap_02.pdf> Aceso em 07 nov 2009.

RAMOS, Ricardo Argenton. **Treinamento Pratico em UML**. 1ª ed. São Paulo: Digerati Books, 2006.

RANZI, Augusto; SILVA, João Vitor Vilela; RIBEIRO, Leandro Telles de Souza. **Modelos de processo** – RAD. Rapid Application Development. Disponível em <www.cefet-to.org/~05203316/diretorio/tecnico/.../RAD.ppt > Acesso em 12 jun 2009.

REIS, Alessandro Cardoso dos. **Sistemas de Conhecimento**. Estudo de caso para modelagem de extensão universitária. Novo Hamburgo: FEEVALE, 2007.

RJHOS. Hospedagem de sites. **O que é um provedor de hospedagem de sites ?**. Disponível em <http://www.rjhost.com.br/faq/141/hospedagem-site/o-que-e-um-provedor-de-hospedagem-de-sites.html>. Acesso em 02 nov 2007.

ROCHA, Állex Motta Melo da. **Introdução à arquitetura de Software**. Belém Do Pará: CESUPA, 2008.

SANTOS, Roneclei Campos dos; JORGE, Eduardo Manoel de Freitas. **Java Server Pages – JSP**. JavaFree.org. 2008.

SCOTT, Kendall. **O processo unificado explicado**: UML. 1ª ed. Porto Alegre: Bookman, 2003.

SEBRAE-SP. Serviço de Apoio às Micro e Pequenas Empresas de São Paulo. **O que é uma atividade meio ?**. Disponível em: <tp://www.sebraesp.com.br/principal/melhorando%20seu%20neg%C3%B3cio/orienta%C3%A7%C3%B5es/marketing/atividade_meio.aspx>. Acesso em 06 out 2007.

SILVA, William Roger Salabert. **Introdução às Redes Peer-to-Peer (P2P)**. Disponível em < http://www.gta.ufrj.br/seminarios/semin2003_1/william/>. Acesso em 07 nov 2009.

SIQUEIRA, Jairo. **Ferramentas de Criatividade – Braimstoring**. Disponível em <http://criatividade.files.wordpress.com/2007/02/brainstorming.pdf>. Acesso em 07 nov 2009.

TEIXEIRA, Paulo Henrique. GUIA TRABALHISTA. **Cuidados na terceirização de atividades**. Disponível em: <ttp://www.guiatrabalhista.com.br/tematicas/perigosdeterceirizar.htm>. Acesso em 06 out 2007.

UFPA. Universidade federal do Pará. **A linguagem HTML**. Disponível em: <http://www2.ufpa.br/dicas/htm/htm-intr.htm>. Acesso em 16 out 2007.

ZANLUCA, Júlio Cezar. GUIA TRABALHISTA. **Cálculo de encargos sociais e trabalhistas**. Disponível em: <http://www.guiatrabalhista.com.br/tematicas/custostrabalhistas.htm>. Acesso em 06 out 2007.

ANEXOS

ANEXO A – Planilha contendo escalas de trabalho

UNIPLUS COOPERATIVA

ESCALA DE ATENDIMENTO EM PAROBÉ

Gestores Responsáveis: RAMÃO [redacted], FABIANA [redacted] e CRISTIANE [redacted]

MARÇO 2009

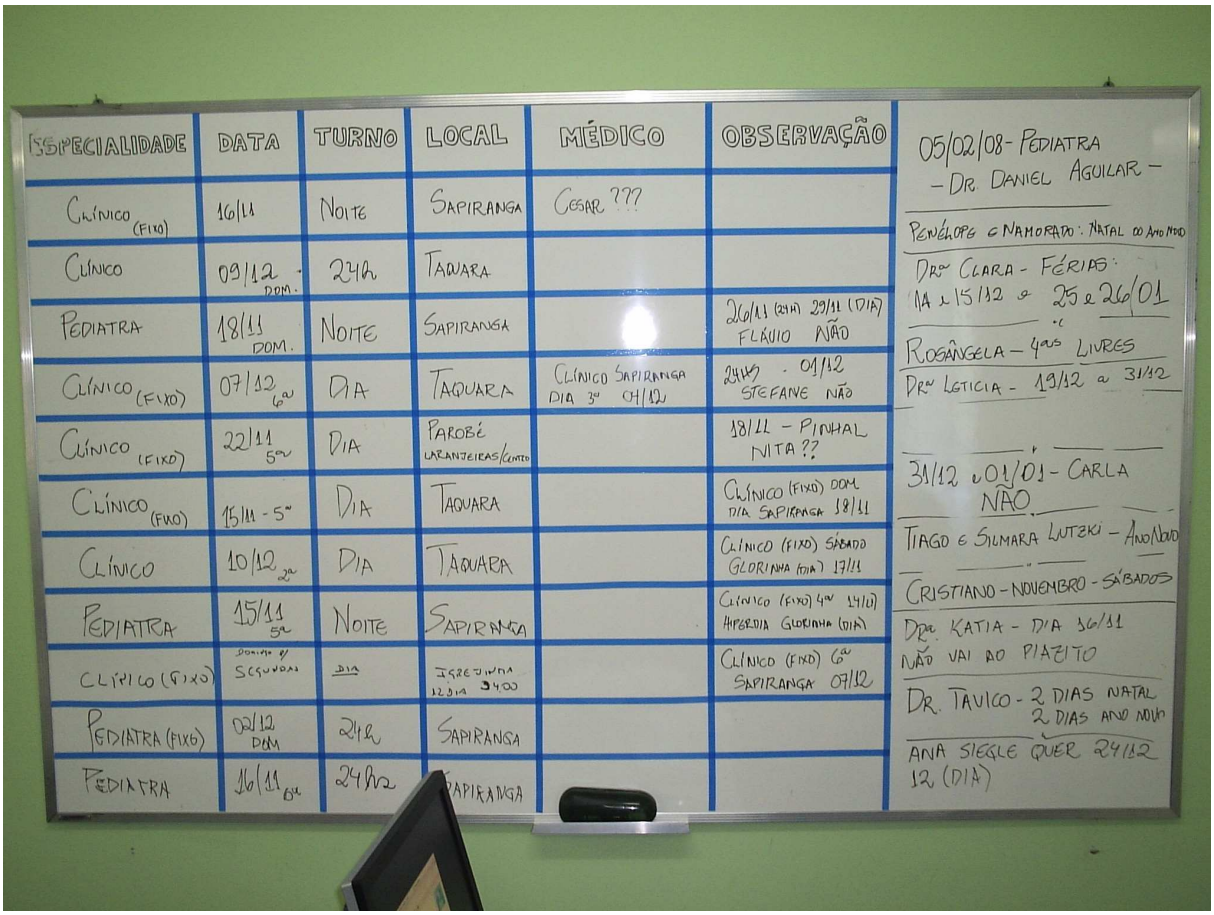
		CLINICA GERAL		PEDIATRIA	
		EQUIPE " 1" - DIURNO		EQUIPE " 2" - DIURNO	
		7 ÀS 19H	SUBSTITUIÇÃO		SUBSTITUIÇÃO
D	1	JULIO DIAS DANIELE TONDOLO MARTINS			
S	2	RUY NICE JAIR OLIVEIRA		DANIELE TONDOLO MARTINS	
T	3	DANIELE TONDOLO MARTINS TIAGO DAL BOSCO	SHANNA MELLO	PIMENTEL	
Q	4	DANIELE TONDOLO MARTINS JAIR OLIVEIRA		PIMENTEL	
Q	5	SHANNA PINTO NICE		PIMENTEL	
S	6	VERA LEITES NICE JANNE		PIMENTEL	
S	7	JULIO DIAS NICE			
D	8	JULIO DIAS DANIELE TONDOLO MARTINS			
S	9	RUY NICE JAIR OLIVEIRA		DANIELE TONDOLO MARTINS	
T	10	DANIELE TONDOLO MARTINS TIAGO DAL BOSCO	SHANNA MELLO	PIMENTEL	
Q	11	DANIELE TONDOLO MARTINS JAIR OLIVEIRA		PIMENTEL	
Q	12	SHANNA PINTO NICE		PIMENTEL	
S	13	VERA LEITES NICE		PIMENTEL	
S	14	JULIO DIAS PIMENTEL			
D	15	JULIO DIAS DANIELE TONDOLO MARTINS			
S	16	RUY NICE JAIR OLIVEIRA		DANIELE TONDOLO MARTINS	
T	17	DANIELE TONDOLO MARTINS MAICON OLIVEIRA	SHANNA MELLO	PIMENTEL	
Q	18	DANIELE TONDOLO MARTINS JAIR OLIVEIRA	WALTER FELAU (à tarde)	PIMENTEL	
Q	19	SHANNA PINTO NICE		PIMENTEL	
S	20	VERA LEITES NICE MAICON OLIVEIRA		PIMENTEL	
S	21	JULIO DIAS PIMENTEL			
D	22	JULIO DIAS DANIELE TONDOLO MARTINS			
S	23	RUY NICE JAIR OLIVEIRA		DANIELE TONDOLO MARTINS	
T	24	DANIELE TONDOLO MARTINS REGIS GOULART	SHANNA MELLO	PIMENTEL	
Q	25	DANIELE TONDOLO MARTINS JAIR OLIVEIRA		PIMENTEL	
Q	26	SHANNA PINTO NICE		PIMENTEL	
S	27	VERÔNICA NICE VERA LEITES		PIMENTEL	
S	28	JULIO DIAS PIMENTEL			
D	29	JULIO DIAS DANIELE TONDOLO MARTINS			
S	30	RUY NICE JAIR OLIVEIRA		DANIELE TONDOLO MARTINS	
T	31	DANIELE TONDOLO MARTINS	SHANNA MELLO	PIMENTEL	

ANEXO B – Trecho de planilha contendo os dados médicos

Nome	Telefone 1	Telefone 2	Telefone 3	Telefone 4	Categoria
Jorge, A. da Silva Moraes					Cardiologistas
José , Antônio Garcia Pinto					Cardiologistas
Júlio, Dias Peralta					Cardiologistas
Luiz, Alfredo Timenn					Cardiologistas
Milton, Simon Pires					Cardiologistas
Paulo, Henrique					Cardiologistas
Samuel					Cardiologistas
Yamile					Cardiologistas
Adair					Clínico Geral
Adolfo, Fernandes					Clínico Geral
Adriano, Felipe Gross Funck					Clínico Geral
Adriano, Ponsoni					Clínico Geral
Afrânio, Souza de Souza					**Clínico Geral**
Alessandra, Felicetti Pires					Clínico Geral
Alethea					Clínico Geral
Alex, Felipe Mullich					Clínico Geral
Alexandra, Pereira Bender					Clínico Geral
Alexandra , Tonel Schroder					Clínico Geral
Alexandre					**Clínico Geral**
Alexandre, Bertin					Clínico Geral
Alexandre, Bohrer Ritt					Clínico Geral
Alexandre, Corrêa Fernandes					Clínico Geral
Alexandre, Cury					Clínico Geral
Alexandre, de Campos Lubwig					**Clínico Geral**
Alexandre, Mitsuo Morita					Clínico Geral
Alexandre , Terra Fontes					Clínico Geral
Alice, Lisboa					Clínico Geral
Aline, Goldaz Szewczynski					Clínico Geral
Aline, Muller da Silva					Clínico Geral
Aline, Rosa					Clínico Geral
Almir					Clínico Geral
Alnete, Mazini Covalo					Clínico Geral
Amauri , Freitas					Clínico Geral
Ana, Carolina					Clínico Geral
Ana, Lucia					**Clínico Geral**
Ana, Lucia Dalmollin					Clínico Geral
Ana, Lucia Torres da Silva					Clínico Geral
Ana, Salazar					Clínico Geral
Ana, Simonete					Clínico Geral
Ana Carolina, Peçanha Antônio					Clínico Geral
Ana Francisca, Tronco Mazzotti					Clínico Geral
Ana Júlia, Shirochy					Clínico Geral
Ana Lusia, Siegle					Clínico Geral
Ana Paula, Moura Moreira					Clínico Geral
Anajara, Gazalle					Clínico Geral
Anderson, Dornelli da Silveira					**Clínico Geral**
Andre, de Aguilar Sá					Clínico Geral
Andre, Luis Lopes					Clínico Geral
André, Luis Ferreira da Silva					Clínico Geral
André , Zanette					Clínico Geral
André (UFRGS)					Clínico Geral
André Luis, Alvares Lourenço					Clínico Geral
Andrea					Clínico Geral
Andrea, D´Angelo					Clínico Geral
Andrea Simões, Martins da Silva					Clínico Geral
Andrei, Luis Ferreira da Silva					Clínico Geral
Andrei, Roberto da Silva					Clínico Geral
Andréia					Clínico Geral
Andréia, Gabriela da Silva Bueno					Clínico Geral
Andréia, Kist fernandes					Clínico Geral
Andréia, Mandatori					Clínico Geral
Andresa, Dulce Marks Bauer					Clínico Geral
Andrese, Gasparim					Clínico Geral
Anelise Kirsch					**Clínico Geral**

ANEXO C – Mural contendo as escalas de trabalho

ANEXO D – Quadro branco armazenando plantões em aberto



ANEXO E – Site informando plantões

ANEXO F – Relatório de análise de dados

DADOS ANALISADOS

Definição de pessoas:
Gestor de escalas
Gestor de contratos
Contratantes
Médicos

Outras definições:
Cada vaga aberta é considerada um plantão de 12horas de trabalho, divididas em turnos dia ou noite.
O pagamento de cada plantão é calculado por um valor/hora
Existem vagas fixas, onde o médico trabalha no mesmo local sempre num dia da semana, e vagas de substituição, onde existe a necessidade de um fechamento exporádico
Cada médico deve assinar o horário de entrada e horário de saída de cada plantão em cada local de prestação de serviço para o controle de efetividade.

Tempo de análise: 1 semana

Tarefas Avaliadas:

Tarefa	**Executantes**	**Descrição das tarefas**	**Tempode execusão por cada tarefa**	**Freq üênc ia**
Fechamento de contrato	Gestor de contratos + contratante	Contratante identifica as especialidades necessárias, o número de horas prestadas por mês por cada especialidade, e os locais de prestação de cada serviço. O gestor de contratos calcula o preço final/hora para cada especialidade.	2 horas	1 x mês
Confecção de escalas	Gestor de escalas	É criada uma escala por mês para cada contratante, e cada especialidade contratada, por local de prestação de serviço	30 minutos	1 x mês
Cadastro de médicos	Gestor de escalas	Médico ou gestor entram em contato, e gestor cadastra em uma planilha do Excel dados pessoais dos médicos	10 min	1 a 5 x por dia
Listagem de vagas nas escalas	Gestor de escalas	Além das vagas fixas em aberto, é comum que aconteça que médicos liguem e solicitem substituições, pelos mais variados motivos. Todas as vagas em aberto então são listadas em um quadro branco,	1 hora	1 x ao dia

		definindo a especialidade, a data, o dia da semana, o período de trabalho e o local de prestação do serviço. Após isso, estas vagas são manualmente inseridas em um site no mesmo modelo do quadro		
Oferecimento de vagas	Gestor de escalas	São inseridos nas vagas os médicos que tem seus períodos de maneira fixa. Então, com as vagas resultantes em aberto, cria-se uma mala-direta e oferta-se a todos os médicos com e-mail informado. Com as vagas ainda restantes em aberto, cada gestor seleciona uma vaga em que a data de execusão esteja mais próxima e entra em contato por meio de telefone ou msn com cada médico. A escolha de qual médico contatar fica a critério de cada gestor, conforme conhecimento pessoal da disponibilidade de cada médico. Em alguns casos, há necessidade de efetuar pagamentos antecipados, ou aumentar o valor/hora ofertado. O grau de dificuldade varia conforme a proximidade da data do plantão e também em datas comemorativas	De 10 minutos a 72 horas	1 a 5 x por dia conforme a dificulda de
Fechamento de vagas	Gestor de escalas	O médico que aceita o plantão é inserido na vaga da respectiva escala. Esta vaga é apagada do quadro branco, e depois é retirada manualmente do site	30 min	1 a 5 x ao dia conforme o fechamento
Procura de oportunidades	Médico	O médico entra em contato com o gestor de escalas, tendo acessado o site ou não, e solicita as vagas em aberto para determinado dia e turno. O gestor de escalas visualiza o quadro branco, e oferece a vaga que se encaixe conforme a solicitação. Se esta é aceita pelo médico, é executado o fechamento de vagas	15 min	1 a 5 x ao dia, conforme demanda
Somatório de horas	Gestor de escalas	São recolhidas todas as folhas de horas, separadas por contratante, e	1 a 2 horas por	1 x mês

		é feito o somatório de horas por médico em cada local de trabalho, comparando as datas descritas na folha de horas com as escalas de trabalho	contratante	
Geração de manifesto	Gestor de escalas	É produzido uma tabela no Excel onde listam-se todos os médicos que prestaram serviços em cada contrato com o seu total de horas e o valor pago ao médico, selecionados por especialidade e local de trabalho	1 hora	1 x mês

ANEXO G – Lista prévia de requisitos

Levantamento de requisitos funcionais:

Médicos:
Logar-se
Cadastrar seus dados e seus horários disponíveis
Verificar todas as vagas existentes
Verificar vagas conforme sua lista de horários disponíveis
Marcar interesse na vaga

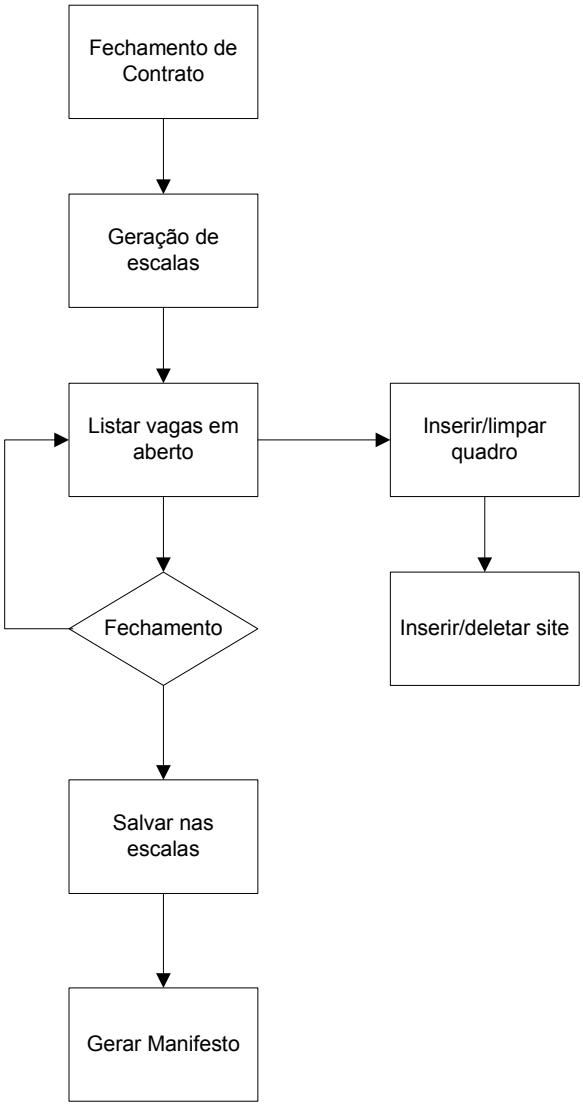
Contratante:
Logar-se
Cadastrar-se
Verificar numero de horas contratadas
Verificar quantidade de horas já efetuadas/mês/especialidade

Administrador:
Logar-se
Cadastrar médicos
Cadastrar horários disponíveis
Cadastrar novo contratante
Criar nova escala/período
Verificar todas as vagas abertas
Verificar médico disponível por vaga
Marcar possível médico por vaga
Inserir médico na vaga
Enviar e-mail geral de vagas
Contabilidade de horas mensal/ contratante/médico

Sistema:
Informar ao médico que seu interesse por vaga não corresponde a seu cadastro de horários disponíveis
Informar número de horas efetuadas semanalmente pelo médico no mesmo contratante no momento em que administrador marca possível médico
Enviar e-mail de interesse de médico por vaga
Enviar e-mail de vaga disponível a possível médico marcado
Avisar sobre possível excesso de horas trabalhadas conforme horas contratadas

Levantamento de Requisitos não-funcionais
Segurança de dados
Oferecer interfaces amigáveis
Confiabilidade:.
Robustez
Disponibilidade do software
Integridade dos dados

ANEXO H – Fluxograma de trabalho na Uniplus



ANEXO I – Representação das classes dos formulários