UNISA - UNIVERSIDADE DE SANTO AMARO
SISTEMAS DE INFORMAÇÃO

FELIPE FERREIRA DE BRITO SANTOS
MARCOS CESAR BRITO MEIRA

GESTÃO INFORMATIZADA DE CONSULTAS E EXAMES MÉDICOS

São Paulo

2012

FELIPE FERREIRA DE BRITO SANTOS
MARCOS CESAR BRITO MEIRA

GESTÃO INFORMATIZADA DE CONSULTAS E EXAMES MÉDICOS

Trabalho de Conclusão de Curso apresentado para obtenção do título de Bacharel em Sistemas de Informação da Universidade de Santo Amaro, sob orientação do Prof. Dr. Eugênio Akihiro Nassu.

São Paulo

2012

FELIPE FERREIRA DE BRITO SANTOS
MARCOS CESAR BRITO MEIRA

GESTÃO INFORMATIZADA DE CONSULTAS E EXAMES MÉDICOS

Trabalho de Conclusão de Curso apresentado para obtenção do título de Bacharel em Sistemas de Informação da Universidade de Santo Amaro, sob orientação do Prof. Dr. Eugênio Akihiro Nassu.

Data da aprovação: _____/_____/_____

BANCA EXAMINADORA

Professor Dr. Eugenio Akihiro Nassu
Universidade de Santo Amaro

Professor Itsche Baran
Universidade de Santo Amaro

Professor Marlom Alves Konrath
Universidade de Santo Amaro

CONCEITO FINAL: _____

RESUMO

Uma necessidade inspira uma melhoria. Um problema inspira uma solução. E assim acontece com o assunto tratado no presente trabalho. No segmento de atenção à saúde, especificamente na informatização do setor de regulação, existe atualmente uma solução que tem por objetivo administrar informações geradas e necessárias para o registro de agendamento de consultas e exames em unidades básicas de saúde. A solução utilizada atualmente abrange o registro, consulta e gerenciamento destas informações inseridas, porém de maneira descentralizada e carente de uma visão gerencial e abrangente. Tal sistema não possui informações centralizadas, pois cada unidade básica de saúde possui seu próprio banco de dados. E assim, nasce o problema: por cada unidade possuir seu próprio banco de dados, as informações ficam fragmentadas e o acesso gerencial que possa prover uma visão totalizadora da informação inserida é extremamente dificultado e tem de ser feito manualmente. Dessa forma surge a ideia para este trabalho de conclusão de curso: o desenvolvimento de uma solução que atue como um software integrado a um banco de dados único que proporcione aos usuários acesso e informações altamente gerenciáveis. Através de uma interface semelhante à existente atualmente (para minimizar o impacto de uma futura implantação) e um banco de dados que permita acessos simultâneos, a aplicação desenvolvida tem o objetivo de solucionar os problemas de fragmentação de informação e dificuldade de gerenciamento dos registros de exames e consultas médicas. Tudo deverá estar centralizado, para que a equipe responsável pelo gerenciamento dessas informações, que hoje depende de relatórios enviados separadamente de cada unidade para realizar a consolidação e verificação de informações, tenha seus processos amplamente melhorados, emitindo diretamente do sistema relatórios consolidados ou unitários desta base única.

Palavras chave: regulação, unidade básica de saúde, especialidade, exame, gestão, banco de dados, sistema, relatório.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Definição

Sistemas são essenciais na vida das pessoas. No trabalho, na escola, em casa, em todo lugar sempre há algum dispositivo executando algum sistema. Em todas as áreas de trabalho, lidar com sistemas é fundamental, pois sem eles muitos processos teriam sua execução demorada e dificultada. Na área da saúde este cenário não é diferente. Neste trabalho aborda um sistema voltado à saúde, especificamente o desenvolvimento de um sistema que tem como objetivo a melhoria de processo e gestão após a análise de uma situação que envolve os sistemas já utilizados para a finalidade de registrar consultas e exames no setor de regulação das unidades básicas de saúde.

Há uma solução informatizada disponibilizada pela Prefeitura de São Paulo que abrange todas as necessidades da regulação das unidades no ponto de vista operacional, pois possui diversos módulos que são necessários ao setor. Porém, do ponto de vista gerencial a ferramenta é insuficiente, pois não possui relatórios satisfatórios e assim os gestores ficam impossibilitados de desenvolver estratégias a partir das informações do sistema. Para suprir tal necessidade, há outra ferramenta informatizada que deve ser utilizada em conjunto com o sistema da prefeitura.

Tal solução informatizada que existe hoje consegue suprir parcialmente a necessidade dos gestores de unidades e do serviço de regulação. O sistema utilizado possui relatórios essenciais para estratégias dos gestores, porém uma das características deste sistema é ser descentralizado. Isto significa que cada unidade básica de saúde possui seu próprio banco de dados, sendo assim, caso o gestor do serviço de regulação necessite de informações consolidadas de todas as unidades, ele terá de solicitar a cada unidade relatórios individuais e consolidar manualmente as informações em uma planilha.

1.2 Objetivo

O objetivo do presente trabalho foi o desenvolvimento de um sistema capaz de registrar agendamentos de exames e consultas no setor de regulação das unidades básicas de saúde. O sistema deve abranger todas as funcionalidades existentes no sistema utilizado atualmente no setor de regulação e agregar também diversas melhorias com a finalidade de auxiliar os gestores do serviço de regulação a ter uma visão gerencial das informações contidas no sistema (por meio de um único banco de dados que integrará todas as unidades necessárias). Tal visão gerencial será possível por meio de relatórios que incluirão informações das regulações de todas as unidades que forem cadastradas no sistema.

1.3 Organização do trabalho

O presente trabalho esta organizado em cinco capítulos e um apêndice.

No capítulo 2 são apresentados os temas relacionados à referência bibliográfica do trabalho. O capítulo abrange teorias e metodologias de engenharia de sistemas, engenharia de requisitos e programação orientada a objetos, demonstrando técnicas utilizadas e estudadas para o desenvolvimento do trabalho. Este capítulo também aborda a modelagem de software, especificamente a modelagem UML, exemplificando sua utilização com explicações dos diagramas utilizados e exemplos dos mesmos em forma de figuras. Por fim, neste capítulo, é apresentado o assunto de banco de dados, com foco no modelo entidade-relacionamento e nos sistemas de gerenciamento de banco de dados (SGBD).

No capítulo 3 são apresentados conceitos abordados pelo assunto tratado no trabalho. Materiais e métodos utilizados para o desenvolvimento do trabalho, materiais estes que tratam dos fundamentos do Sistema Único de Saúde (SUS), as UBS (Unidades de Saúde Básica) que funcionam no modelo de PSF (Programa Saúde da Família). Também são apresentados conceitos e regras referentes ao setor de regulação das UBS e uma breve explanação dos sistemas utilizados no setor para informatização do serviço.

No capítulo 4 são apresentados os resultados obtidos com o trabalho, sendo, neste caso, o sistema que foi desenvolvido, assim como as suas funcionalidades e as vantagens e desvantagens do sistema inclusive em comparativo com o sistema já utilizado atualmente no setor de regulação.

No capítulo 5 são apresentadas as conclusões do trabalho e as ideias de futuras melhorias pensadas para o contexto do sistema desenvolvido.

Adicionalmente há um apêndice com a documentação do sistema desenvolvido que abrange diagramas e documentos gerais e também um glossário com alguns termos abordados no presente trabalho.

2 REVISÃO BIBLIOGRÁFICA

2.1 Engenharia de sistemas

Como escreveu (PRESSMAN, 2010, p. 1) “... a invenção de uma tecnologia pode ter efeitos profundos e inesperados em outras tecnologias aparentemente não relacionadas, em empresas comerciais, nas pessoas e até na cultura como um todo...".

O *software* é um programa criado por desenvolvedores, mantido por empresas e utilizado por todos que de alguma forma tem qualquer ínfimo contato com tipos variados de aparelhos eletrônicos para fins que em sua grande maioria que permitam facilitar, automatizar e sem dúvida melhorar o ambiente ao que esta relacionado.

Os sistemas têm importância marcante e irrefutável em todos os âmbitos da sociedade e tal importância cresce a cada dia, através da criação de novas soluções e a substituição e aprimoramento de soluções e métodos que podem ser julgados como ultrapassados.

Software, por definição e uso é uma área extremamente abrangente e pode ser destrinchada como um universo próprio dentro do universo da computação.

Sendo assim, nada mais lógico que subdividir o "universo" do *software* em sete categorias abrangentes, como explicado por Pressman (2010) e elucidado resumidamente abaixo:

- ***Software* de sistemas:** são *softwares* que realizam interação diretamente com o *hardware*. Podem ser citados os compiladores, componentes de S.O e *drivers*.

- ***Software* de aplicação:** são os *softwares* comuns em empresas, que tratam de uma necessidade de negócio específica, utilizando dados técnicos ou comerciais.

Nesta categoria enquadra-se nosso trabalho que como explicado anteriormente, trata-se do desenvolvimento de um *software* de aplicação, voltado ao uso organizacional, baseado em dados, requisitos e especificações próprias do negócio a que esta relacionado, no nosso caso, o registro de consultas médicas e exames.

- ***Software* científico e de engenharia:** Esta caracterização de *software* engloba sistemas que sem dúvida alguma são mais complexos que outros, não pelo seu desenvolvimento, mas sim pelo seu nível crítico e necessidade de alto conhecimento técnico para o desenvolvimento de suas regras e cálculos que muitas vezes envolvem alto processamento. Podemos citar como exemplos os sistemas utilizados em astronomia e previsão climática.

- ***Software* embarcado:** é o *software* utilizado em aparelhos eletrônicos comuns, em que muitas vezes não é possível perceber estar lidando com o *software*, porém ele está lá, rodando e mantendo o funcionamento do aparelho em questão. Para elucidar este tipo de *software*, é possível citar os sistemas utilizados em aparelho de micro-ondas e televisores.

- ***Software* para linha de produtos:** é o *software* desenvolvido para uma finalidade muito específica e que pode ser usado por diversos clientes diferentes. Sistemas que utilizam planilhas, processamento de texto e de imagens são exemplos que *software* desta categoria.

- **Aplicações para *web*:** iniciados sendo apenas arquivos interligados por hipertexto, hoje são *softwares* complexos, muito bem desenvolvidos e com diversas funcionalidades indispensáveis a qualquer tipo de negócio. Com a evolução da internet, os *softwares* para *web* também sofrem evolução constante e crescimento sem igual. Podem ser descritos como aplicações para web os *softwares* de gestão de banco de dados, entretenimento e multimídia online, entre outros.

- ***Software* de Inteligência Artificial:** são utilizados para resolver situações complexas, em que algoritmos que não utilizam exclusivamente números são empregados. São fortemente baseados em conhecimento específico da área em que será utilizado, funcionando por padrões pré-definidos. Alguns exemplos encontrados comumente são os sistemas de reconhecimento de voz e reconhecimento de imagens.

Dando continuidade, um tema de interessante comentário no presente trabalho diz respeito ao conceito de *software* legado e as maneiras de tratá-lo. Como dito

anteriormente, este trabalho baseia-se na reformulação, atualização e melhoria de um *software* já utilizado há alguns anos no ambiente das UBS.

Sendo assim, podemos caracterizá-lo como um *software* legado, pois de acordo com (WARD E BENNET, 1995, pg. 1) o "*software* legado pode ser informalmente definido como aquele que executa tarefas úteis para a organização, mas que foi desenvolvido utilizando-se técnicas atualmente consideradas obsoletas".

A motivação de realizar este trabalho deriva principalmente da possibilidade de melhoria a partir de um *software* legado, que é extremamente necessário para o negócio ao qual esta relacionado e que apesar de obsoleto não pode ter seu funcionamento simplesmente interrompido.

O *software* legado em muitas vezes apresenta uma estrutura já instável para a atualidade, técnicas obsoletas e até dificuldade para melhorias e alterações, e tais problemas podem ser explicados por três sentidos:

- **Portabilidade X disponibilidade:** incompatibilidade com sistemas operacionais mais recentes é um problema recorrente nos *softwares* legados. No caso do sistema legado abordado neste trabalho, por ter sido desenvolvido em *Access* (parte integrante do pacote *Office* da *Microsoft*) é excessivamente dependente do pacote em que foi desenvolvido e apresenta diversos problemas de incompatibilidade quando utilizado com um pacote *Office* de release anterior ou superior. A adaptação torna-se muito complicada e força os desenvolvedores a recriar o sistema no pacote *Office* apropriado e atualizado para o uso 100% funcional;

- **Boas práticas X manutenção:** a manutenção de um *software* legado é extremamente complicada, pois em grande parte dos casos a equipe que o desenvolveu não esta mais na empresa, a documentação é praticamente inexistente. No caso do sistema abordado neste trabalho, a manutenção do mesmo é extremamente complicada, pois por ter sido desenvolvido em *Access* não é possível realizar alterações e atualizações modulares, sendo necessário

assim desenvolver uma nova versão do "zero", alteração que inclusive afeta o banco de dados (também desenvolvido no *Microsoft Access*);

- **Usabilidade:** a interface do *software* deve ser intuitiva, que possibilite ao usuário facilidade e tranquilidade para lidar com a ferramenta, além de possibilidade de evolução em sua utilização. Nos *softwares* legados isto não acontece, pois é extremamente complicado adaptar a interface de um sistema antigo para padrões atuais.

Manter um *software* legado pode ser caro para a organização e complicado para o suporte da área de T.I, porém há motivos para que as organizações mantenham os *softwares* legados.

Primeiramente, é cabível citar o custo de manter o sistema. As organizações veem nos *softwares* legados a possibilidade de ter seu negócio em andamento, o funcionamento do software em estado contínuo e a não necessidade de gastar com uma nova solução, pois a antiga ainda funciona. Porém, é neste ponto que as organizações erram, pois pode sair muito mais caro (tanto monetariamente quanto em relação às informações) manter um *software* antigo, pois caso o mesmo pare de funcionar é possível que ocorra perda de informações e de valor para a organização.

Outro motivo é o receio de migrar para novas soluções que não seriam bem aceitas pelos usuários e consequentemente não "iriam para frente". É preciso arriscar para acertar, sendo assim, é necessário que as organizações aceitem novas ideias, inovações, evolução. O presente trabalho tem também o intuito de ir contra esse receio e promover a inovação, mantendo a aparência que é amigável ao usuário ao mesmo tempo em que implementa diversas novas funcionalidades.

O *software* legado pode ser caracterizado como um risco para a organização, com grandes chances de tornar-se um problema. Sendo assim, para eliminar ou mitificar esse risco, é extremamente indicado migrar o *software* legado para um atualizado.

Para desenvolver o novo *software*, a decisão foi de basear-se na interface de usuário do *software* legado utilizado atualmente, para que caso ocorra a utilização em

campo do mesmo, os usuários não o rejeitem devido a preferirem a interface antiga que já conhecem há tantos anos e estão acostumados.

Por tratar-se de um *software* obsoleto e que necessita de reestruturação em sua arquitetura, a sugestão é a de migração total, pois é necessário que a implantação da nova solução desenvolvida envolva aplicação e banco de dados.

Por fim, é importante citar o assunto referente aos "Sete Princípios do Desenvolvimento de *Software*" de David Hooker. Descritos por Hooker (1996) são princípios chave para desenvolver um *software* da melhor maneira possível. Tais princípios são elencados abaixo como:

- **1º Princípio - "A razão pelo qual tudo existe":** um sistema existe para entregar algum valor ao seu usuário, seja no âmbito financeiro, organizacional ou de melhoria de processo. Sendo assim, também existe uma razão para cada funcionalidade que compõe o *software*. Por isso é importante que ao desenvolver pergunte-se se a funcionalidade imaginada acrescentará valor no sistema, pois caso não acrescente, não deve ser feita. Somente deve ser acrescentado ao sistema o que agregará valor quando implementado, por tal motivo este princípio se aplica.

- **2º Princípio - "KISS (*Keep It Simple, Stupid*!)":** neste princípio o importante e manter o *software* simples. Quanto mais complexo é, mais difícil seu entendimento e consequentemente mais complicado será o suporte, correções e alterações provenientes de erros e sugestões de melhoria. O *software* simples tem menor probabilidade de erros e é mantido mais facilmente. Simples, neste caso, não se trata de incompleto e esteticamente ruim, mas sim de funcionalidades bem elaboradas e discutidas para serem aplicadas da melhor maneira possível.

- **3º Princípio - "*Maintain the Vision*":** para que um *software* seja bem desenvolvido e tenha sucesso em sua utilização é importante manter a visão do projeto. Manter a visão significa, neste caso, conceber uma ideia e firmar-se

nela. A ideia não precisa ser mantida fielmente durante todo o projeto, porém é importante que a integridade e a visão de arquitetura e negócio sejam mantidas.

- **4º Princípio - "*What you produce they will consume*":** o *software* desenvolvido será utilizado por alguém. Porém, mais que utilizar, pessoas vão dar suporte, documentar e possivelmente alterar este *software*, sendo assim, ao construir o *software* é importante realizar tudo de maneira clara, para quem for lidar com isso entenda do que se trata. É interessante inclusive para o crescimento e melhoria do sistema que todos que realizam algum tipo de interação com ele o entendam.

- **5º Princípio - "*Build for today. Design for tomorrow*":** ao desenvolver o *software* é importante deixa-lo adequado e apto a mudanças que ocorrem com o tempo, sejam elas em relação a plataforma utilizada, *hardware* e outros fatores externos, assim a vida útil do sistema se estenderá consideravelmente. É importante, no projeto e desenvolvimento do *software* a equipe perguntar-se “o que acontece se...”, de maneira a entender previamente situações futuras e preparar a estrutura do *software* a receber as alterações e tornar-se reutilizável.

- **6º Princípio - "*Plan ahead for reuse*":** é importante planejar, estruturar e desenvolver um sistema visando a sua reutilização. A reutilização, como acontece em *softwares* desenvolvidos com tecnologia de linguagem orientada a objetos, é muito interessante porque possibilita que um sistema que já é utilizado tenha novas implementações de forma simples, economizando tempo e esforço da equipe responsável.

- **7º Princípio - "*Think*!":** o princípio mais simples e incrivelmente um dos que são mais ignorados: pense! Os melhores resultados são obtidos a partir de ideias bem pensadas e principalmente bem planejadas. Se a ação é bem pensada e ainda assim falha, o erro torna-se aprendizado para que em uma situação posterior seja utilizado.

2.1.1 Requisitos

Uma tarefa primordial para que o desenvolvimento de um sistema seja realizado da maneira correta e apropriada é realizar a engenharia de requisitos. E realizar a engenharia de requisitos não é tarefa fácil. Clientes confusos, usuários que divergem em suas opiniões e em muitas vezes comunicação escassa podem dificultar.

A engenharia de requisitos ajuda os desenvolvedores a compreender como construir o sistema, sendo realizada por eles em conjunto com pessoas que possuem o conhecimento de negócio relacionado com o sistema a ser desenvolvido. É extremamente importante atender os requisitos especificados no processo de engenharia de requisitos, pois caso o sistema desenvolvido não siga o que foi definido previamente, estará fadado a ter de ser refeito, alterado ou até mesmo não ser aceito antes mesmo de sua implantação.

A engenharia de requisitos, como citada, é parte fundamental na engenharia de sistemas. Sendo uma atividade complexa e bem completa no desenvolvimento, Pressman (2010) especifica a engenharia de requisitos em funções. São sete etapas, que descrevem a engenharia de requisitos com a finalidade de proporcionar melhor entendimento e desenvolvimento:

- **Concepção:** corresponde à etapa de concepção inicial do sistema a ser desenvolvido, com o objetivo de entender as necessidades do cliente, as ideias dos envolvidos e os problemas em geral que envolvem o projeto. É necessário nesta fase o entendimento entre cliente e desenvolvedor por meio de comunicação clara, aberta e explicativa a respeitos do que o cliente deseja que o *software* a ser desenvolvido seja e faça.

- **Levantamento:** a fase de levantamento abrange os questionamentos que devem ser feitos ao cliente sobre o *software*. Questionamentos estes que podem ser simples ou complexos. Para citar alguns exemplos é possível utilizar perguntas como: qual o objetivo do software? Como ele vai se enquadrar nas necessidades do negócio? Como será utilizado?

Porém o levantamento de requisitos pode não ser tarefa tão fácil assim, principalmente devido aos problemas relacionados ao levantamento de requisitos. Tais problemas foram elucidados por (CHRISTEL E KANG, 1992, p. 7, 8, 9, 10, 11 e 12) e ainda continuam utilizáveis, como segue:

Problemas de escopo: "o limite do sistema é mal definido ou o cliente/usuário especifica detalhes técnicos desnecessários que podem confundir, em vez de esclarecer, os objetivos globais do sistema".

Problemas de entendimento: "os clientes/usuários não estão completamente certos do que é necessário, tem pouca compreensão das capacidades e limitações de seu ambiente computacional, não tem pleno entendimento do domínio do problema, têm dificuldade de informar as necessidades aos desenvolvedores, omitem informações por serem 'óbvias', especificam requisitos conflitantes ou especificam requisitos que são ambíguos ou impossíveis de testar".

Problemas de volatilidade: "os requisitos mudam ao longo do tempo".

Segundo Zultner (1992) os requisitos podem ser agrupados em três tipos:

Requisitos normais: são os requisitos que estão relacionados aos objetivos e metas do sistema. Se tais requisitos estiverem inclusos no software, o cliente estará satisfeito.

Requisitos esperados: são requisitos que caso faltem no *software* causarão insatisfação do cliente. Em grande parte são implícitos, porque o cliente os considera óbvios, porém fundamentais.

Requisitos excitantes: são os requisitos que vão além das expectativas do cliente. Caso consigam ser implementados corretamente são muito satisfatórios, porém se não são entregue como combinado, isto pode gerar desconforto na relação com o cliente.

- **Elaboração:** nesta etapa as informações obtidas são organizadas, filtradas e são acrescidos alguns documentos de modelagem, como os cenários de interação de usuário com o sistema e modelagem de relação de informações no sistema. Nesta fase da engenharia de requisitos, são usados diagramas UML. Como

resultado da elaboração, a equipe tem modelos que possibilitam a análise das informações, funcionalidades e comportamento do sistema.

- **Negociação:** após a fase de elaboração, com as documentações e requisitos definidos, pode ocorrer de certos requisitos definidos e elaborados não estarem de acordo com a necessidade do cliente, com os recursos ou até mesmo apresentem conflito entre si. A fase de negociação deve ser utilizada para regularizar e corrigir tais problemas encontrados nos requisitos.

- **Especificação:** fase correspondente a apresentação dos documentos e informações obtidas em forma de textos, gráficos, documentos de modelagem, cenários ou protótipos. Os documentos apresentados na fase de especificação são os responsáveis por nortear o desenvolvimento do sistema, pois com as especificações de requisitos definidos, é possível que a equipe conheça suas possibilidades e restrições futuras.

- **Validação:** a etapa de validação é utilizada pela equipe envolvida no desenvolvimento e pelo cliente para revisar a documentação apresentada na fase anterior, para realizar correções, melhorias e esclarecimentos.

- **Gestão de Requisitos:** durante o projeto é necessário controlar a mudança nos requisitos, assim como o cumprimento dos que já foram previamente estabelecidos. E neste cenário que a etapa de gestão de requisitos é utilizada. Para controlar e identificar tais requisitos**,** são usadas cinco tipos de tabelas para rastrear requisitos, como segue: características (relação de requisitos com características do sistema); fontes (identifica o responsável do requisito); dependência (relação entre requisitos); subsistemas (requisitos por subsistemas); interface (relação dos requisitos com interfaces). Exemplo de tabela de rastreamento de requisitos pode ser visualizado na figura 1:

Figura 1 - Tabela de rastreamento de requisitos

Fonte: Jenny (2012)

Indiscutivelmente, em todas as sete etapas descritas na engenharia de requisitos, uma necessidade abrange de modo geral todas e se falha, diversos problemas podem ocorrer: a comunicação.

O desenvolvimento de um sistema depende de três partes: o usuário, o cliente e o desenvolvedor. A necessidade de um cliente deve ser bem explicada, para que os desenvolvedores criem uma ferramenta que supra esta necessidade sendo utilizada pelos usuários. Sendo assim, a comunicação entre estas partes deve ser coordenada e bem realizada.

Porém, em muitos casos a comunicação clara dos requisitos é difícil, pois o cliente e os usuários não tem conhecimento acerca do desenvolvimento do *software* e os desenvolvedores não tem conhecimento do negócio e da finalidade do *software*.

Para superar esta dificuldade é importante que as durante as fases de concepção e levantamento a comunicação seja bem elaborada e que a equipe responsável coordene muito bem as reuniões para definição de requisitos de requisitos, assim como os documentos que registrem tais requisitos.

Os requisitos podem ser divididos, além dos três tipos definidos anteriormente, também em outros grupos, especificados por Pressman (2010) como segue:

- **Requisitos de Usuário:** definidos pela norma ISO/IEC 14143-1 (medição funcional de tamanho) em requisitos funcionais de usuário e não funcionais de

usuário. São requisitos feitos para o cliente, com as possibilidades do sistema, serviços e restrições. Devem ser escritos em linguagem que todos entendam, tanto desenvolvedores quanto cliente.

Os requisitos funcionais de usuário descrevem o que o sistema deve fazer, e somente os requisitos funcionais são utilizados para parametrizar a quantidade de funções do sistema.

Os requisitos não funcionais de usuário não são definidos especificamente pela ISO/IEC 14143-1, porém podem ser exemplificados como requisitos que não fazem parte do sistema, em que as funcionalidades principais não dependem dos mesmos. Os requisitos não funcionais também envolvem particularidades da organização, políticas internas de privacidade e padrões de qualidade aplicáveis.

- **Requisitos de Sistema:** é o documento que descreve de maneira detalhada das funcionalidades do sistema e suas restrições. É a formalização do que deve ser realizado no projeto do *software*. São os fundamentos do sistema e também podem integrar o contrato como meio de formalizar o que foi acordado em relação ao *software* a ser entregue.

- **Requisitos do Domínio:** são requisitos do domínio da aplicação e demonstram as características da mesma. Podem ser requisitos novos funcionais ou restrições a requisitos funcionais previamente definidos. São em grande parte obrigatórios, pois caso o sistema não os atenda, suas funcionalidades serão insatisfatórias. Podem apresentar problemas em relação a compreensão dos requisitos, devido a linguagem utilizada e requisitos implícitos por serem considerados óbvios por especialistas e que acabam desconhecidos por outros não especialistas.

- **Requisitos de Interface:** definem a interoperabilidade entre o sistema desenvolvido e outros demais sistemas. Utilizando notações formais é possível

especificar interfaces de procedimentos, estruturas de dados e representação de dados que vão interagir com outro sistema como requisito do mesmo.

2.1.2 Programação orientada a objetos

A programação orientada a objetos não é um conceito novo, mas sim um conceito que existe há muitas décadas e foi aprimorado com o passar do tempo. A programação orientada a objeto tem em uma de suas maiores vantagens a facilidade na manutenção do código: é possível alterar uma classe isoladamente, sem afetar o restante do código que esta funcionando normalmente como pode ocorrer nas linguagens estruturais e procedurais.

A programação orientada a objetos tem alguns conceitos principais referentes a si, dos quais é interessante comentar: classe, atributo, método, objeto, herança e polimorfismo. De acordo com artigos da MSDN (2012) tais conceitos podem ser descritos da seguinte maneira:

- Classe é o modo abstrato, generalizado, de definir um tipo de objeto na programação orientada a objetos. A classe possui atributos (que definem o que ela é) e métodos (que definem o que ela faz). A classe é o conceito do objeto, como se fosse sem vida, apenas descrição e funções do que o objeto, quando instanciado será e fará.
- Os atributos são as características da classe, são as variáveis que estão presentes nos objetos instanciados de cada classe. Uma classe pode ter quantos tipos de atributos for necessário, e os objetos dessa classe podem ter os atributos preenchidos de maneira diferente em cada instância, respeitando o tipo de atributo definido na classe.
- O método é a ação que uma classe pode realizar, representado por um bloco de instruções. Os métodos são relacionados à classe e responsáveis por manipular a instância no objeto ao qual estão relacionados.
- O objeto representa uma entidade real, e que dentro do sistema representa a classe em sua execução, uma instância. Um objeto pode executar funções (métodos) de acordo com o necessário e estipulado.

- A herança, dentro do conceito de programação orientada a objetos, é a capacidade que uma classe tem de estender, reutilizar e modificar as características de outra com as suas próprias.
- Polimorfismo é explicado sob dois diferentes aspectos. O primeiro aspecto diz respeito à mudança do objeto em relação ao momento de sua execução, sendo assim, seu tipo pode se diferenciar entre objetos da mesma classe. O outro aspecto diz respeito à ação do objeto, isto é, a variação dos métodos a serem executados.

É interessante citar a brevemente evolução das linguagens de programação até chegar à que utilizaremos em nosso trabalho: o C#.

- *SIMULA* 67 (1967): primeira linguagem de programação orientada a objetos. Já possuía funções muito utilizadas e conhecidas hoje como classes, objetos, instâncias, herança e polimorfismo.
- *Smalltalk* (1972): nesta linguagem tudo é tratado como objeto: classes, métodos, blocos de código, entre outros. Não há tipos primitivos e os tipos básicos como string, números e caracteres são implementados como classes. Os objetos são organizados de forma hierárquica, facilitando a criação de novos objetos com características de outro já existente.
- *Ada* (1982): linguagem estruturada e estática, orientada a objetos e de alto nível, originada no *Pascal*. Seu nome deriva-se da primeira mulher da programação *Ada Lovelace*. Foi criada por uma equipe contratada pelo departamento de defesa dos EUA com o intuito de substituir as muitas linguagens que eles utilizavam. A versão mais recente da *ADA* data de 2005.
- *Eiffel* (1986): linguagem orientada a objeto que utiliza teorias do teórico francês da orientação a objeto *Bertrand Meyer*. Apresenta diversas inovações, porém também características de linguagens antigas como o *Simula* 67 e a *ADA*.
- *Modula* 3 (1986): sucessora da *Modula* 2, não foi muito utilizada em indústrias de software, porém auxiliou no desenvolvimento do *Java*, C# e *Python* por exemplo. Tem como destaques as características de simplicidade e segurança. Foi

acrescida com o tempo de tratamento de exceções, orientação a objeto, encapsulamento de código, entre outras.

- *Ruby* (1995): linguagem orientada a objetos, que combina recursos do *Perl* e do *Smalltalk*. Foi originada no Japão por *Yukihiro Matsumoto*. Possui sistema de dados dinâmicos e gerenciamento de memória automática, com aspectos similares ao *Python*, por exemplo.
- *Java* (1995): linguagem orientada a objetos criada na *Sun Microsystems*. Diferencia-se de linguagens convencionais, pois é compilada em *bytecode* e executada por uma *VM*. É a linguagem orientada a objetos mais difundida entre todas e esta presente em muitos sistemas e *softwares* disponíveis hoje, inclusive no *Android*.

- C# (2000): linguagem orientada a objetos criada pela *Microsoft*, sendo parte integrante da *plataforma .NET*. É baseada em C++ e *Java*, possui compilador próprio desenvolvido e por ter sido desenvolvida do zero, funciona sem problemas na plataforma .*NET*. É a linguagem de programação que utilizaremos no desenvolvimento do sistema abordado por nosso trabalho, e por tal motivo aprofundaremos a explicação desta linguagem de programação.

De acordo com o Portal Educação (2008), a linguagem de programação C# foi criada por diversos desenvolvedores, porém é atribuída principalmente a *Anders Hejlsberg*, um engenheiro da *Microsoft*, que trabalhando em outras empresas e já havia criado o *Turbo Pascal* e o *Delphi*.

De acordo com artigos da MSDN (2012) a implementação do C# não inclui bibliotecas de classes e funções, mas sim o *framework .NET* para obter as classes e funções. As classes com funções similares são agrupadas por *namespaces*. Um nível de organização superior também é provido pelo conceito de montador, com arquivos ligados que podem conter diversos *namespaces* ou objetos.

2.2 Modelagem

A modelagem de um *software* parte da necessidade de determinar com maior precisão e detalhamento o que um sistema será e fará antes mesmo de ser desenvolvido, com base nos requisitos levantados com o cliente.

Utilizando a modelagem é possível prever como o projeto será, sua dificuldade de desenvolvimento, custo, tempo de criação, entre outros fatores. Quanto maior o projeto do *software*, maior a quantidade de requisitos e consequentemente mais importante será realizar uma modelagem completa e bem pensada. Todo sistema deve ter a etapa de modelagem prévia ao desenvolvimento, e durante seu uso também devem ter revisões, pois as necessidades se modificam com o tempo, sugestões de melhoria, mudanças de mercado, novas políticas internas, mudança de leis que afetam diretamente o sistema, dentre outros motivos.

Para facilitar manutenção, suporte, correções e melhorias são extremamente importantes que a documentação seja muito bem detalhada, de maneira a documentar o sistema e auxiliar todos os envolvidos a entendê-lo.

Utilizar modelagem resulta em criar modelos de software, que como descrito por (GUEDES, 2011, p. 21), o modelo de software “captura uma visão de um sistema físico, é uma abstração do sistema com certo propósito, como descrever aspectos estruturais ou comportamentais do software”, determinando assim o que existirá no sistema e que cada um poderá fazer ao utilizá-lo.

2.2.1 Modelagem UML

A UML (*Unified Modeling Language*) é utilizada, com base no que foi escrito por Guedes (2011) para modelar *softwares* baseados em orientação a objetos. Consiste numa linguagem visual, composta de diagramas que se tornou padrão internacional para realizar a modelagem de *softwares*.

Os diagramas UML tem como objetivo auxiliar os engenheiros de *software* a definir as características do sistema a ser desenvolvido, como requisitos, comportamento, estrutura lógica, dinâmica de processos e necessidades antes do desenvolvimento do sistema.

De acordo com Guedes (2011), a UML nasceu da junção de três métodos de modelagem: *Booch*, *OMT* e *OOSE*. Estes três métodos eram até os anos 90 os mais populares para modelagem entre os profissionais de desenvolvimento.

Inicialmente foi realizada a união do método de *Booch* com o OMT no fim de 1995, que como resultado criou o Método Unificado. Esta nova metodologia incorporou o OOSE e assim, com as três metodologias unidas a UML foi criada em 1996. Em 1997 a UML foi adotada pela OMG (*Object Management Group*) como uma linguagem padrão de modelagem de *software*. Em 2005 foi lançada a versão 2.0.

A linguagem UML tem uma grande quantidade de diagramas, e o objetivo de possuir um portfólio tão extenso de diagramas é fornecer a equipe desenvolvedora, diversas visões do sistema a ser desenvolvido, realizando analise e modelagem, para que cada diagrama complemente outro.

Cada diagrama tem um objetivo específico e analisa o sistema a ser desenvolvido sob uma visão diferente. Baseado em Guedes (2011) é possível dizer que há diagramas que abordam a modelagem de software de maneira mais generalizada, apresentando uma visão externa do sistema do sistema, como acontece com o diagrama de caso de uso. Entretanto, há outros diagramas que abordam a modelagem do sistema de maneira mais profunda ou técnica, como explicado posteriormente.

No desenvolvimento de um sistema é importante utilizar diversos tipos de diagrama, pois isso propicia ao desenvolvedor uma visão completa do que o sistema deverá ser, diminui a possibilidade de erros e problemas de desenvolvimento.

2.2.2 Diagramas UML

A linguagem UML, como dito previamente, possui uma quantidade muito grande de diagramas, elencados por Guedes (2007) como diagramas de: casos de uso, classes, objetos, estrutura composta, sequencia, comunicação, máquina de estados atividade, interação geral, componentes, implantação, pacotes e tempo. Porém o foco de explicação neste trabalho será em apenas alguns destes diagramas, que são de grande importância para o desenvolvimento da monografia.

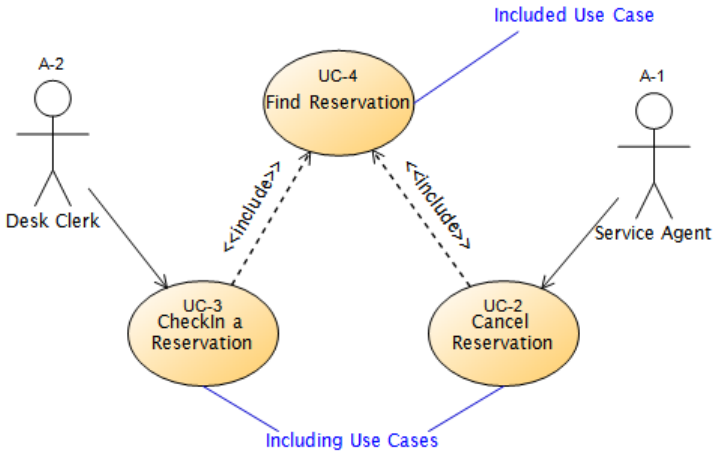
- **Diagramas de Casos de Uso**

O diagrama de caso de uso é o mais difundido e utilizado para analisar requisitos e ações que usuários podem fazer no sistema. Segundo (GUEDES, 2007, p. 15) o diagrama de caso de uso “apresenta uma linguagem simples e de fácil compreensão para que os usuários possam ter uma ideia geral de como o sistema irá se comportar”, e para demonstrar esse comportamento o diagrama utiliza dois aspectos principais: os atores e os serviços.

Os atores, como exemplificado por (GUEDES, 2007, p. 15) podem ser “usuários, outros *softwares* que interajam com o sistema ou até mesmo algum hardware especial”, sendo assim, é possível definir o ator como uma entidade que interage de alguma forma com o sistema, utilizando suas funcionalidades, neste caso os serviços.

A demonstração de interação no caso de uso é indicada através da associação entre um ator e uma ação, interconectados por uma linha. Também é possível especificar generalização e especificação, por meio da conexão entre casos de uso relacionados de maneira generalizada. Também é possível especificar no caso de uso uma situação que envolva mais de um caso de uso, a inclusão. O relacionamento de inclusão, como explicado por (GUEDES, 2007, p. 42) os “relacionamentos de inclusão indicam uma obrigatoriedade, ou seja, quando um determinado caso de uso possui um relacionamento de inclusão com outro, a execução do primeiro obriga também a execução do segundo”, sendo assim os casos de uso são interligados e dependentes entre si.Para exemplificar o diagrama de caso de uso, é demonstrado o exemplo na figura 2:

Figura 2 - Exemplo de caso de uso



Fonte: Terski (2005)

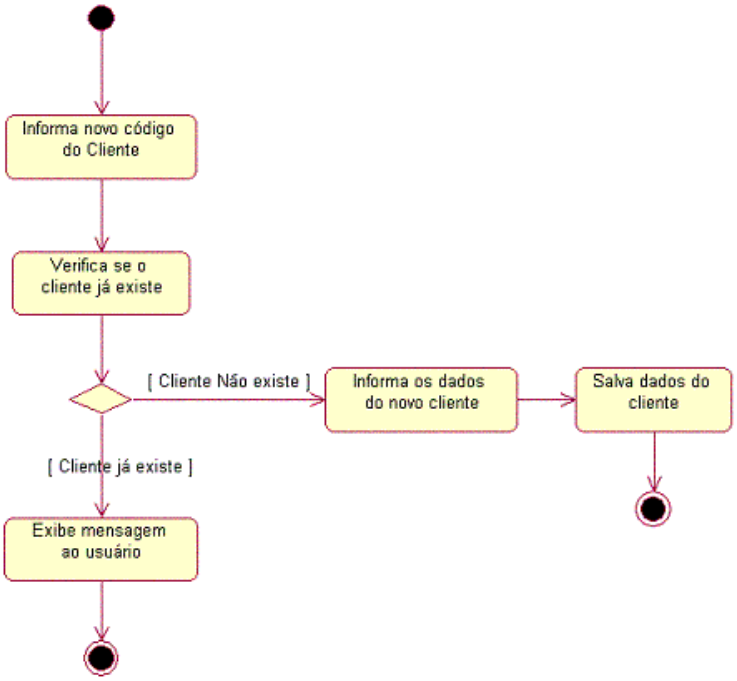
- **Diagramas de Atividades**

O diagrama de atividades pode ser considerado um diagrama relativamente novo no grupo de diagramas da linguagem UML, pois antes da versão 2.0 do UML ele fazia parte do diagrama de gráfico de estados.

O objetivo deste diagrama consiste, como explicado por (GUEDES, 2007, p. 22) "em descrever os passos a serem percorridos para a conclusão de uma atividade específica, muitas vezes representada por um método com certo grau de complexidade, podendo, no entanto, modelar um processo completo". Sendo assim, o diagrama de atividades tem a funcionalidade que antes era atribuída ao diagrama de fluxo de dados, por exemplo, com a diferença que a representação é realizada em relação ao objeto que realiza uma atividade.

Sua representação é, de acordo com Pressman (2010) baseada em retângulos arredondados que descrevem uma função em particular do sistema, setas que demonstram o fluxo no sistema, losangos que representam decisão e linhas horizontais para representar atividades paralelas. Tais elementos podem ser visualizados no exemplo de diagrama de atividades na figura 3:

Figura 3 - Exemplo de diagrama de atividades



Fonte: Macoratti (2010)

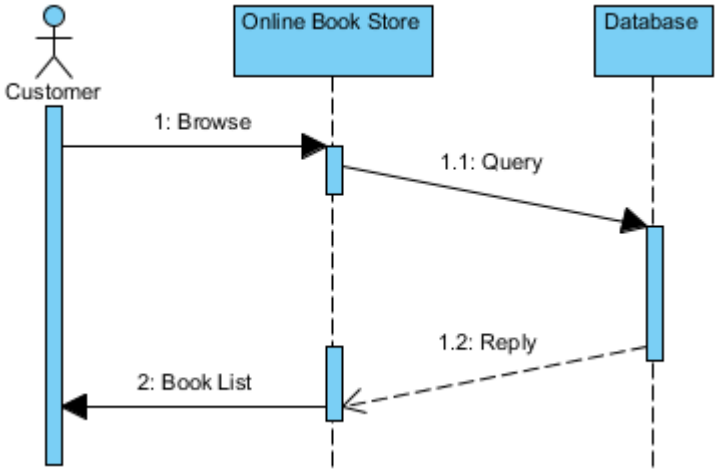
- **Diagrama de Sequência:**

O diagrama de sequência, de acordo com Guedes (2007) é voltado a representar a ordem temporal que os objetos se comunicam em um processo específico, com base no diagrama de caso de uso e de classes. Desta maneira é possível além de determinar “quem faz o que” também é possível especificar o “quando”.

A comunicação entre os objetos demonstrados neste diagrama acontece por meio de mensagens que tem a finalidade de iniciar os métodos relacionados com estes objetos, assim, demonstrando claramente como deve ocorrer as ações e sua ordem de ocorrência no sistema.

Para representação deste diagrama, Pressman (2010) indica o uso de setas para representar eventos derivados dos casos de uso, o tempo é demonstrado de maneira vertical de cima para baixo e são usados retângulos verticais para representar o processamento da mensagem. Na figura 4 há um exemplo deste tipo de diagrama da linguagem UML:

Figura 4 - Exemplo de diagrama de sequência



Fonte: Yeung (2010)

- **Diagrama de Classes:**

Quando um desenvolvedor e sua equipe lidam com um projeto de software que será realizado com programação orientada a objetos, o diagrama da linguagem UML que não pode de maneira alguma faltar é o diagrama de classes, devido a sua importância em todo o projeto.

Como definido por (GUEDES, 2007, p. 17) o diagrama de classe “define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos possuídos por cada classe, além e estabelecer como as classes se relacionam”.

O diagrama de classes tem diversas particularidades em si, desde a forma como é organizado até os elementos que utiliza em sua confecção. É importante citar algumas destas particularidades que foram descritas por Guedes (2007) como partes integrantes do diagrama de classes:

- Associação: é o relacionamento entre as classes, para troca de informações e execução de métodos em comum. A associação pode ser representada por diversos tipos de vínculos diferentes, sempre por setas conectando as classes.
- Associação reflexiva: relacionamento que ocorre quando há algum tipo de associação de um objeto de certa classe com outro objeto que pertença à mesma classe.

- Associação N-ária: também chamadas de associações ternárias, são associações que envolvem quantidade maior que duas classes. É representado por um losango, que conecta as classes.
- Agregação: associação em que são demonstradas informações de um objeto-todo (que necessita de informações de outra classe) com o objeto-parte que o completa. Para simbolizar esta associação é usado um losango na extremidade da linha que liga as classes relacionadas, posicionando o losango na ponta da linha que esta próxima do objeto-todo.
- Composição: tipo de associação derivada da agregação, com o objetivo de demonstrar que o vínculo entre o objeto-todo e o objeto-parte, explicitando que os objetos-parte informados só podem estar relacionados a um objeto-todo. Também é representado por um losango na extremidade da linha que conecta no objeto-todo, porém neste caso o losango é preenchido.
- Especialização / Generalização: associação que tem a função de identificar as classes gerais (mãe) e especializadas (filhas), demonstrando herança entre as classes.
- Dependência: o relacionamento de dependência demonstra que uma classe depende de alguma maneira de outra. Isto é ilustrado pela conexão de uma linha tracejada ligando as duas classes, e uma seta apontando para qual classe a outra depende.
- Multiplicidade: indica, em um relacionamento, a quantidade mínima e máxima de objetos que relacionam na associação, com o número referente a quantidade especificado nas extremidades da linha que os conectam. Na tabela 1 são demonstrados os tipos de multiplicidade:

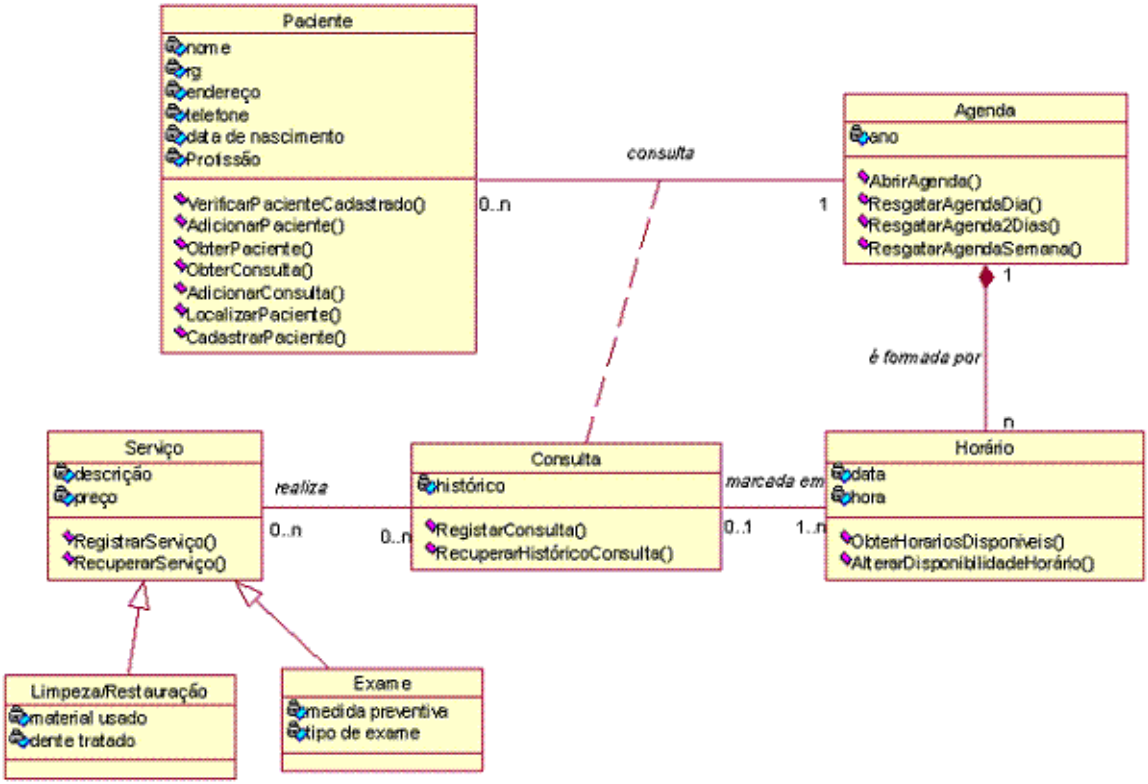
Tabela 1 - Tipos de multiplicidade

Multiplicidade	Significado
0..1	No mínimo zero e no máximo um. Não há obrigatoriedade de relacionamento entre os objetos, porém caso ocorra associação, apenas uma instância se relacionará com as instâncias da outra classe.
1	Um e somente um. Um objeto da classe especificada deve se relacionar com o objeto de outra classe.
0..*	No mínimo zero e no máximo muitos. Demonstra que pode não haver nenhum objeto executando o relacionamento, e pode se relacionar com diversos outros objetos.
*	Muitos. Demonstra que vários objetos de uma classe específica podem estar no relacionamento.
1..*	No mínimo um e no máximo muitos. No mínimo um objeto deve estar envolvido, mas pode haver vários.

Fonte: Elaborado pelo autor

Na figura 5 há um exemplo de diagrama de classe utilizando os conceitos demonstrados:

Figura 5 - Exemplo de diagrama de classes



Fonte: Macoratti (2005)

2.3 Banco de Dados

De acordo com (ELMASRI e NAVATHE, 2005, p. 3) um banco de dados é "uma coleção de dados relacionados. Com dados, queremos dizer fatos conhecidos que podem ser registrados e possuem significado implícito". Sendo assim, o banco de dados é o responsável por armazenar qualquer informação necessária e utilizada por um sistema. É parte integrante e importante dos sistemas e possibilita ao usuário que as informações necessárias possam ser inseridas e manipuladas de acordo com regras e funcionalidades estabelecidas.

Para descrever de maneira menos genérica, Elmasri e Navathe (2005) determinam algumas características para banco de dados, sendo eles: representação de aspecto do mundo real, alteração baseada em alterações reais, agrupamento lógico e coerente de dados e desenvolvimento com finalidade específica. Sendo assim, um banco de dados deve armazenar que forma organizada informações que existem fora do sistema e que sofrem mudanças de acordo com o fato ao qual é relacionado no mundo real se altera.

Tratando de banco de dados, é importante citar que o banco de dados de modelo mais utilizado atualmente e que também foi utilizado neste trabalho é o banco de dados relacional, que de acordo com definição de (SILBERCHATZ; KORTH E SUDARSHAN, 2006, p. 4) é o banco de dados "baseado no modelo relacional e usa um conjunto de tabelas para representar os dados e as relações entre elas". Para utilização do banco de dados é utilizado um SGBD (sistema gerenciador de banco de dados), estrutura de tabelas e manipulação de dados por meio de DDL (linguagem de definição de dados) e DML (linguagem de manipulação de dados), o conhecido SQL.

O SGBD é o sistema responsável por gerenciar o acesso ao banco de dados e prover o armazenamento, alteração e manutenção dos dados nele armazenados. Um SGBD utiliza, para acessar os dados do banco, dois métodos de acesso: a DML e a DDL.

A DML é a linguagem para definição do banco, sua estrutura, dados, tipos e informações que o mesmo possuirá. Por definição de (SILBERCHATZ; KORTH E SUDARSHAN, 2006, p. 6) a DML "permite aos usuários acessar ou manipular dados conforme são organizados pelo modelo de dados apropriado". Esses acessos de usuário podem ser consulta de dados armazenados, inserção de novos dados, alteração de dados e exclusão de dados armazenados. Por estas características uma sigla remete facilmente a memória: o SQL. A DML pode ser procedural ou declarativa, de acordo com a necessidade de acesso e a finalidade da manipulação dos dados.

Outra linguagem integrante dos SGBDs é a DDL, linguagem de definição de dados, que de acordo com Silberschatz; Korth e Sudarshan (2006) é a linguagem utilizada especificar o esquema do banco de dados e propriedades adicionais de dados. Ainda de acordo com os autores anteriormente citados, a linguagem DDL possui um

tipo especial de tratamento, que especifica estrutura de armazenamento e acesso através do SGBD, como regras e restrições para acesso e modificação de dados armazenados. Tais regras e restrições são definidas por Silberschatz; Korth e Sudarshan (2006) em algumas características principais:

- Restrições de domínio: valores associados com atributos. Os atributos são definidos previamente, visando restringir a o tipo de dado que será inserido no banco em uma localização específica.
- Integralidade referencial: um conjunto de atributos é relacionado com outro conjunto de atributos, gerando assim uma referência entre eles. A modificação dos dados que possuem integralidade referencial deve ser realizada com cautela, pois se violada, será rejeitada.
- Assertivas: são regras que delimitam as modificações no banco de dados, uma condição que deve acontecer. Se valida, alterações futuras nos dados só serão feitas se a assertiva não for violada.
- Autorização: diferencia o tipo de acesso aos dados para diferentes usuários. Os usuários podem ter permissão apenas para visualizar dados, inserir dados, alterar dados, excluir dados ou uma combinação de algumas ou todas estas permissões.

2.3.1 Modelo Entidade Relacionamento

De acordo com Elmasri e Navathe (2011) o modelo relacional é uma representação dos relacionamentos existentes no banco de dados, isto é, os diferentes valores de diferentes tabelas são interligados de alguma forma e possuem relação entre si para diferentes objetivos.

A partir do modelo relacional, é possível entender que as informações ficam armazenadas no banco de dados em tabelas bidimensionais, de nome único no banco de dados, que possuem colunas (atributos) e linhas (registros ou tuplas), sendo que em cada coluna há apenas um tipo de atributo, neste caso o domínio do dado.

No modelo relacional, dentre algumas regras que se aplicam as tabelas, uma interessante e importante a ser citada é em relação às chaves de registro. De acordo com Silberschatz; Korth e Sudarshan (2006) as chaves são o método capaz de identificar os registros num banco de dados unicamente em sua tabela. As chaves podem ser primárias (*primary key*) ou estrangeiras (*foreign key*).

A escolha da chave de uma tabela começa com o levantamento da chave candidata, que possivelmente pode se tornar uma chave primária. A chave primária é o atributo do registro que representa valor único e que pode ser utilizado para identificar o registro, servindo de organizador para a tabela. A chave estrangeira é o atributo que existe nos registros de uma tabela e é utilizado para relacioná-la com outra tabela em que este atributo encontra-se em forma de chave primária. A conexão entre a chave primária de uma tabela e chave estrangeira corresponde a si em outra tabela cria uma ligação lógica de tabelas, sendo assim um relacionamento entre tabelas.

O diagrama relacional é a forma gráfica simplificada de representar a relação entre duas tabelas. Para sua confecção são utilizados símbolos que representam aspectos do banco de dados como especificado por Silberschatz; Korth e Sudarshan (2006): a entidade (tabela) é representada por um retângulo, os atributos são representados por elipses, os losangos representam o relacionamento entre as entidades e as linhas, que conectam as entidades com o relacionamento que as unem. As entidades correspondem a objetos que existem na vida real, sendo que os atributos são as características que tais objetos possuem.

Para demonstrar o relacionamento entre as entidades é necessário utilizar uma notação que indica a restrição e quantidade mínima e máxima de entidades para realizar tal relacionamento, a cardinalidade. Para explicar brevemente tais relacionamentos, é interessante demonstrar o seguinte:

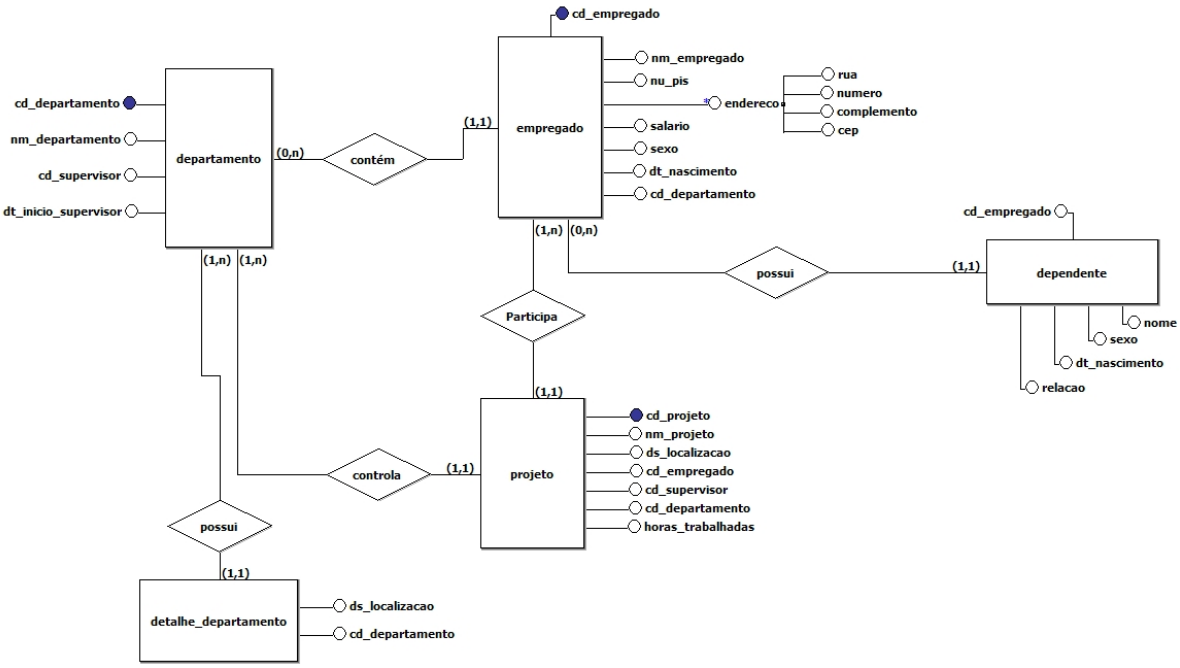
- Relacionamento 1 para 1 (um para um): apenas um registro de cada lado do relacionamento podem se relacionar entre si, como por exemplo, duas chaves primárias;
- Relacionamento 1 para N (um para muitos): o registro em uma tabela pode se relacionar com muitos outros registros na outra extremidade do relacionamento, por meio de chave estrangeira;

- Relacionamento N para N (muitos para muitos): relacionamento em que pode haver vários registros de uma tabela com outros vários registros correspondentes em outra tabela.

Com base nestas notações de cardinalidade, é possível fazer o uso da notação de par de cardinalidade, com a finalidade de especificar com maior detalhamento o relacionamento das entidades. Utilizando a notação (min, max) para determinar os valores do relacionamento.

Os relacionamentos entre entidades também podem apresentar maior complexidade, utilizando generalização e agregação. A generalização é utilizada para especificar uma superentidade e as entidades relacionadas a ela, representando que os atributos presentes na superentidade estão presentes nas entidades a si relacionadas, porém cada entidade tem suas características próprias. A demonstração da generalização é realizada por meio de um trapézio. A agregação une duas entidades para a criação de uma "entidade" com maior quantidade de atributos e maior complexidade. São formadas por entidades primitivas e agregadas. Estas características formam o modelo entidade-relacionamento, visto na figura 6:

Figura 6 - Exemplo de modelo entidade-relacionamento



Fonte: Santos (2010)

2.3.2 SGBD

O SGBD, por definição de (SILBERSCHATZ; KORTH e SUDARSHAN ,2006, p. 4) "um sistema de gerenciamento de banco de dados é uma coleção de dados inter-relacionados e um conjunto de programas para acessar esses dados". Esta coleção de dados que os autores citam corresponde ao banco de dados, responsável por armazenar as informações inseridas. Sendo assim o SGBD é o programa responsável por gerenciar este banco de dados, tornar suas informações disponíveis e acessíveis de forma eficiente.

Um sistema utilizado para gerenciar um banco de dados é utilizado para realizar todo tipo de alteração e implementação necessária no banco, e para realizar tais ações, o SGBD utiliza de linguagens de banco de dados: a DML e a DDL.

Por definição de Elmasri e Navathe (2011) o SQL é uma linguagem que abrange diversos aspectos do banco de dados, como instruções para definição de dados, atualizações e consultas. Sendo assim é possível manipular o banco de dados tanto de maneira estrutural quanto de maneira lógica no que diz respeito aos dados. Esta manipulação é realizada utilizando-se dois tipos de linguagens: a DDL e a DML.

Os comandos DDL são usados para gerenciar o banco, suas tabelas e informações estruturais. Já os comandos DML são os que podem manipular as informações contidas no banco de dados. Cada linguagem é voltada especificamente para um tipo de usuário, sendo que os usuários que fazem uso da DDL em geral são os administradores do banco de dados (DBA). Os usuários que comumente fazem uso da DML, ou ao menos de seu resultado, são os usuários finais do sistema, que manipulam as informações contidas no banco por meio de expressões pré-definidas pelos DBAs.

Com base nestas duas linguagens, é possível subdividir os comandos em grupos, de acordo com sua finalidade. Consulta de dados (*SELECT*), alteração de dados (*INSERT*, *UPDATE* e *DELETE*) e manipulação de dados (*CREATE*, *DROP* e *ALTER*).

A utilização de um SGBD para gerenciar um banco de dados tem diversas vantagens, seja elas estruturais ou funcionais. Sendo assim, é interessante citar brevemente algumas dessas vantagens com base no que foi especificado por Elmasri e Navathe (2011):

- **Controle de redundância**: trata a redundância no armazenamento de arquivos e registros, através da atualização lógica dos registros, para que os mesmos não se dupliquem na atualização. Controlar a redundância também apresenta vantagem em relação ao espaço de armazenamento, pois o mesmo não sofre desperdício. Com esta funcionalidade, o banco de dados torna-se consistente, pois se torna impossível que registros que correspondem ao mesmo dado atualizem-se de forma diferente.

- **Restrição de acesso:** o SGBD pode restringir o acesso aos dados armazenados no banco de dados de acordo com as configurações de permissão necessárias. É possível dar permissões específicas para os usuários, permitindo-os apenas visualizar informações específicas, inserir novos dados e manipular os dados existentes no banco.

- **Backup e recuperação de dados:** uma funcionalidade importante de um SGBD é a possibilidade de recuperar os dados mantidos no banco caso ocorra alguma falha de hardware. Por mecanismos de recuperação, o SGBD impede que o registro seja corrompido caso ocorra alguma falha durante uma atualização. Dessa maneira o *Backup* e *Restore* são duas funcionalidades fundamentais de um SGBD.

- **Restrição de integridade:** o SGBD fornece a capacidade de configurar e impor restrições para integridade dos dados presentes no banco. A restrição de integridade é utilizada para garantir segurança na alteração de registros e pode envolver regras de definem tipo de dados, restrição de chave e constraints.

- **Múltiplos usuários:** o SGBD deve permitir acessos simultâneos ao banco de dados, porém deve evitar que ocorra acesso simultâneo para alteração de um mesmo registro e que isto não afete outros usuários. Esta característica é presente em um SGBD multiusuário, que de acordo com Elmasri e Navathe (2011) deve permitir que vários usuários realizem acesso ao mesmo tempo ao banco de dados, por meio de controle de concorrência, para que em uma transação um registro fique reservado e não possa ser alterado por um usuário quando já estiver selecionado por outro.

Há diversos SGBDs disponíveis no mercado, voltados a diferentes públicos e até mesmo concorrentes de mercado. *Softwares* comerciais, gratuitos e de grande utilização. Dentre os principais é possível citar o *Oracle*, *SQLServer*, *FireBird* e o *MySQL*. No desenvolvimento do presente trabalho utilizaremos o *MySQL* como SGBD do banco de dados de nosso sistema.

De acordo com o Portal Infowester (2008), dentre as vantagens de utilizar o MySQL, é possível elencar as seguintes:

- Disponibilidade para diversos sistemas operacionais diferentes;
- Gratuito sob licença GPL;

- Ótima compatibilidade com C#;
- Baixa exigência de hardware;
- Conexão SSL;

O *MySQL* é voltado para agilidade e é utilizado em sistemas que não demandam complexidade no tratamento dos dados. Por não exigir recursos avançados de hardware, ser gratuito, processamento ágil e compatibilidade com C#, este SGBD escolhido.

3 MATERIAIS E MÉTODOS

3.1 Sistema Único de Saúde – SUS

O SUS (Sistema Único de Saúde) é um projeto de âmbito nacional que tem como objetivo prover o acesso à saúde para todo cidadão brasileiro. Foi criado na promulgação da Constituição Federal de 1988 e regulamentado pelas leis nº 8080/90 e nº 8142/90, que correspondem a leis relacionadas a saúde, que tornam obrigatório o atendimento público a qualquer cidadão gratuitamente. Foi um grande avanço da saúde no país, pois até aquela época o atendimento de saúde era dividido em três categorias, que de acordo com o Ministério da Saúde, caracterizavam: atendimentos pagos e serviços privados, atendimento gratuito em saúde pública se tivesse carteira assinada e os cidadãos que não tinham qualquer direito a atendimento.

Após a implantação do modelo de atendimento SUS, de acordo com informações do Governo Federal, a quantidade de cidadãos que são atendidos aumentou expressivamente, passando de 30 para 190 milhões, sendo que destes, 80% possuem somente o SUS como acesso a saúde. Com a unificação da saúde pública, a responsabilidade de gestão deixou de ser somente do governo federal e passou a ser também do governo estadual e municipal por todo o país.

O SUS tem diversos projetos de reconhecimento nacional, dentre eles o SAMU (Serviço de Atendimento Móvel de Urgência), atenção a saúde da mulher, saúde do trabalhador, atenção básica e saúde da família, entre muitos outros. Para abrigar tantos serviços diversificados, a infraestrutura do SUS conta com unidades básicas de saúde, hospitais, ambulatórios de especialidades, pronto socorro e diversos outros estabelecimentos, que disponibilizam aos cidadãos a possibilidade de realizar consultas, exames, internação e tratamentos gratuitamente.

No presente trabalho, o foco foi no serviço de atenção básica a saúde, especificamente no serviço provido pelas regulações das UBSF (Unidades Básicas de Saúde da Família) no agendamento de consultas e exames médicos dos pacientes. Esta modalidade de serviço de Unidade Básica de Saúde teve início em 1991, com a implantação do programa de agentes comunitários de saúde, responsáveis por visitar

as residências na área de abrangência das unidades. Em 1994 foi criado o Programa Saúde da Família, para atender cidadãos de acordo com o local em que residem. Em 2001 foi aprovada a NOAS (Norma Operacional de Assistência à Saúde), que regulamenta e define a forma como ocorre a regionalização e descentralização dos serviços de saúde.

3.1.1 Programa Saúde da Família

O programa saúde da família (PSF) é um serviço do governo de orientação assistencial, realizado por meio da implantação de equipes multiprofissionais nas unidades. Cada equipe é responsável por acompanhar certo número de famílias, localizada em uma área específica.

O PSF surgiu como um novo modo de lidar com a saúde pública, focando na família para atenção à saúde, seja para tratamento e para prevenção. O PSF pode ser entendido como um modelo de atenção, que de acordo com Campos (1997) é um conceito que estabelece a conexão a abordagem técnica e a política. Pode ter influência clínica, sociocultural e organizacional.

Começou em 1991 quando foi criado o PACS (Programa de Agentes Comunitários de Saúde) com o objetivo de auxiliar na redução da mortalidade infantil e materna nas regiões Norte e Nordeste, em áreas pobres e com tendência a problemas relacionados à saúde. Após perceber que o programa foi bem aceito e obteve resultados interessantes, o enfoque do programa deixou de ser o indivíduo e passou a ser a família.

De acordo com Rosa e Labate (2005) mesmo sendo concebido como um programa, o PSF não é especificado como tal, pois não é diretamente relacionado com os programas convencionais do Ministério da Saúde. É na verdade uma estratégia de acompanhamento, focando a família com o objetivo de prevenir doenças e problemas relacionados à saúde. Por definição de (ROSA e LABATE, 2005, p. 4), o PSF é: “um modelo de atenção que pressupõe o reconhecimento de saúde como um direito de cidadania, expresso na melhoria das condições de vida; no que toca a área de saúde,

essa melhoria deve ser traduzida em serviços mais resolutivos, integrais e principalmente humanizados".

As unidades de PSF são organizadas em equipes. Cada equipe possui agentes comunitários de saúde (responsáveis pelas visitas a domicílio), auxiliares de enfermagem (responsáveis pelo atendimento inicial e básico aos membros das famílias) e médicos (responsáveis por prover consultas, receitas e encaminhamentos de especialidades para os pacientes). Sendo assim, com a equipe definida, cada uma fica responsável por uma área de abrangência, que corresponde às famílias que moram em determinadas ruas que são atendidas por esta equipe. Tal divisão de área de abrangência é realizada através da territorialização, que corresponde à divisão do território ao redor da unidade em equipes, para melhor organização e atendimento das famílias.

Após a explicação básica da fundamentação das unidades baseadas em PSF, é importante citar como funciona o fluxo de informações nas unidades para chegar até o foco de nosso trabalho, a regulação.

Primeiramente, uma família é cadastrada na unidade pelo seu respectivo agente comunitário de saúde (ACS). Por meio deste cadastro os membros da família podem ser atendidos na unidade, passar em consultas, participar de programas e receber medicação gratuitamente. A partir do cadastro da família na unidade, já é possível que quando necessário, os membros da família solicitem ao seu ACS a marcação de uma consulta médica. A consulta médica do membro da família é realizada com o médico da equipe correspondente a que a família está cadastrada e que será responsável pelo acompanhamento desta família e sua evolução clínica. O médico da família pode encaminhar o paciente para realizar exames caso necessário, assim como tem a função de encaminhar o paciente para médicos especialistas se o caso necessitar.

Com posse do encaminhamento para realizar exames ou consulta de especialidade, o paciente é orientado a passar na regulação da unidade, setor responsável pelo agendamento dos exames e especialidades, é que é o setor foco deste trabalho e consequentemente o utilizador do sistema que realiza o registro destes exames e encaminhamentos para especialidades.

3.1.2 Regulação

A instituição de regulações e complexos reguladores teve início com a aprovação da NOAS (Norma Operacional de Assistência à Saúde) 01/2001 e 01/2002, que foi deliberada com o intuito de reduzir a fragmentação na gestão da saúde pública e melhorar sua estruturação. Em conjunto com o Pacto de Gestão, a saúde foi delimitada por competência e complexidade de serviço, estabelecendo fluxos adequados e a referência e contra-referência.

O complexo regulador assistencial, por definição da Portaria/SAS/MS n.º 356 (2000) “compreende a concepção que institui ao poder público o desenvolvimento de sua capacidade sistemática em responder às demandas de saúde em seus diferentes níveis e etapas do processo de assistência". Sendo assim, a responsabilidade de um complexo regulador é lidar com a demanda de necessidade da população em relação à saúde, no que diz respeito aos procedimentos e consultas passíveis de regulação.

A central de regulação define quais são as especialidades e procedimentos (consultas e exames) que serão regulados, isto é, que terão vaga disponibilizada para pacientes da rede básica de saúde. As unidades solicitantes são as próprias unidades de saúde, responsáveis por solicitar a central de regulação às vagas para agendar seus encaminhamentos de consultas e exames. As unidades executantes são os estabelecimentos de saúde, sendo eles públicos ou privados, que possuem atendimento conveniado com o SUS a qual o paciente é encaminhado após o agendamento de seu encaminhamento.

Dentre os tipos de centrais de regulação definidas pela SMS (Secretaria Municipal de Saúde), que definem assistência de urgência, hospitalar, entre outras, o foco no presente trabalho é a Central de Regulação de Consultas e Exames. Esta central é responsável com regular a disponibilidade e acesso dos pacientes das unidades de saúde a exames por meio de SADT (Serviço Auxiliar de Diagnóstico e Terapia) e encaminhamentos para consultas especializadas por meio de ficha Referência / Contra-Referência.

A regulação, como concebida pela NOAS e aplicada às unidades de saúde pode ser definida como uma maneira de proporcionar ao paciente, de forma organizada

acesso à saúde, consultas especializadas e SADT (serviço de apoio diagnóstico e terapêutico), assim atendendo sua necessidade em relação à saúde.

A regulação funciona por meio de uma política de oferta de recursos, disponibilizando consultas e exames de acordo com a determinação do complexo regulador. O setor de regulação tem a responsabilidade de receber solicitações de exames e consultas, ordená-los por prioridade, agendar a solicitação, emitir confirmação de agendamento, comunicar o paciente a respeito do agendamento, emitir e analisar relatórios, assim como desenvolver e implementar estratégias que contribuam com a redução de absenteísmo.

O setor de regulação possui documentos norteadores, que auxiliam e normatizam a organização de agendamentos. Manuais e protocolos estão disponíveis para auxiliar os funcionários do setor ao agendamento correto das solicitações, entregues em forma de encaminhamento devidamente preenchidas pelo médico respectivo ao qual o paciente foi atendido. O agendamento é responsabilidade da unidade de saúde e assim que há vaga disponível para agendar o encaminhamento, é necessário comunicar o paciente a respeito para saber de sua disponibilidade.

3.2 Sistemas utilizados na regulação

O serviço de regulação das unidades de saúde, abordando somente o cenário da cidade de São Paulo, tem sua informatização determinada pela Prefeitura de São Paulo, como informado no Diário Oficial da Cidade de São Paulo (2012) que por meio da divulgação da ATA da 2497ª sessão divulgou algumas determinações acerca da informatização das regulações de unidades básicas de saúde.

- Em todas as UBS é obrigatório o lançamento dos encaminhamentos no sistema de gestão SIGA saúde.
- Registro individualizado por meio de número do CNS (Cartão Nacional de Saúde) desde o primeiro agendamento de demanda no sistema SIGA Saúde.

- A fila de espera para realização de exames e consultas deve ser informatizadas e estar incluída no sistema SIGA Saúde, de acordo com a necessidade das unidades solicitantes.

O sistema SIGA Saúde é uma ferramenta disponibilizada pela Prefeitura de São Paulo e de uso obrigatório em toda a rede de saúde de São Paulo , como determinado pela Portaria nº 887/SMS publicada no Diário Oficial Municipal de 30/06/2007 ao que diz respeito à unidades de saúde públicas que tem gestão municipal. O uso do sistema é regularizado por normas da SMS e abrange todos os serviços de saúde disponíveis no município de São Paulo.

Dentre os serviços que o SIGA Saúde abrange está incluída a regulação, assunto tratado neste trabalho. O sistema tem que por obrigatoriedade ser utilizado pela regulação, devido a sua integração com todas as agendas locais e reguladas do município para registro de encaminhamentos. Do ponto de vista operacional, o sistema SIGA saúde é uma ferramenta completa para o setor de regulação, pois possui módulos de agenda local, agenda regulada, fila de espera, registro de atendimento e demanda reprimida (encaminhamentos pendentes de agendamento) de consultas e exames.

Porém, apesar de ser um sistema completo do ponto de vista operacional ao que diz respeito à regulação das unidades básicas de saúde, o sistema SIGA Saúde é insuficiente ao ponto de vista gerencial do serviço de regulação. O sistema não possui relatórios que atendam completamente as necessidades estabelecidas para o serviço, sendo assim os gerentes tanto de unidades quanto os gerentes regionais responsáveis pela regulação não tem como obter as informações que necessitam apenas com a utilização do SIGA Saúde. Informações estas que são de essencial importância e que são exigidas pelas gerências regionais para acompanhamento, controle e melhoria do serviço prestado a população. São exigidos, de acordo com o Plano de Reestruturação do Setor de Regulação nas Unidades, por exemplo:

- Relatório mensal do setor;
- Lista de espera por “Especialidade” e “SADT”, do período do dia 20 do mês anterior ao dia 21 do mês corrente;

- Demanda reprimida de consultas e exames;

Ao gestor da unidade são determinadas algumas competências relacionadas as informações necessárias, como análise da lista de espera, implantação de diretrizes de atendimento, garantia de funcionamento da regulação, discussão com órgãos reguladores a respeito de vagas e desenvolvimento e implantação de estratégias que auxiliem na redução do absenteísmo.

Para o gestor conseguir desempenhar este papel e ter as informações necessárias para aplicar suas competências é necessário que as unidades de saúde tenham uma ferramenta que possibilite gerar relatórios gerenciais que possuam tais informações.

Nas unidades básicas de saúde da região Cidade Ademar é utilizado um sistema voltado ao serviço da regulação. Este sistema chama-se NCR (Novo Controle de Referência) e foi desenvolvido com o intuito de auxiliar os gestores a organizar o fluxo de trabalho do setor de regulação e criar uma alternativa com relatórios que não é atendida pelo SIGA Saúde da prefeitura.

O NCR é um sistema desenvolvido em *Microsoft Access* (tanto banco de dados quanto aplicação) e é utilizado diariamente pelas unidades. A utilização do NCR não exclui a utilização do SIGA Saúde, inclusive pois é regulamentada e obrigatória, porém é necessário que o a utilização do NCR seja mantida, pois o mesmo possui todos os relatórios necessários para os gestores das unidades e para os gestores do serviço. O funcionário do setor de regulação deve incluir a ficha de referência ou SADT no sistema NCR e no SIGA Saúde, sendo assim, é possível ter um acompanhamento em tempo real dos encaminhamentos da unidade assim como a norma estabelecida pela SMS esta sendo seguida. Quando há alguma alteração no status do encaminhamento, seja por qualquer motivo (contato com paciente, agendamento confirmado, cancelamento e falta) o registro no NCR também deve ser alterado, mantendo-se assim sempre atualizado.

Porém, o sistema NCR utilizado tem alguns problemas e situações que poderiam ser melhoradas, em acréscimo com os problemas já citados referentes a software legado:

- O sistema é instalado localmente nas unidades, sendo assim, além da aplicação o banco de dados também fica localmente em uma máquina na unidade. Caso esta máquina apresente problemas, o serviço fica dependente exclusivamente de *backup* para restaurar as informações, porém há sempre a possibilidade de perda de informações;

- Por ser desenvolvido em *Microsoft Access*, o funcionamento do sistema está condicionado ao pacote *Office* instalado na máquina, sendo assim, caso o *Office* for de versão diferente a que o sistema foi desenvolvido (2007), o mesmo apresentará problemas de funcionamento;

- Cada unidade tem seu próprio sistema e seu próprio banco de dados, caracterizando assim extrema fragmentação do serviço. O gestor do serviço tem de solicitar a cada unidade os relatórios atualizados mensalmente e após isso consolidar manualmente as informações recebidas para assim gerar as estatísticas e valores de demanda reprimida e absenteísmo de todas as unidades sob sua gestão.

Com o estudo e visualização de tais problemas e a necessidade do sistema NCR para os processos de trabalho do setor de regulação, nasceu a ideia do presente trabalho, e seu desenvolvimento e aplicação serão explicados com maiores detalhes no capítulo 4.

4 RESULTADOS

4.1 Sistema desenvolvido: Zephyros

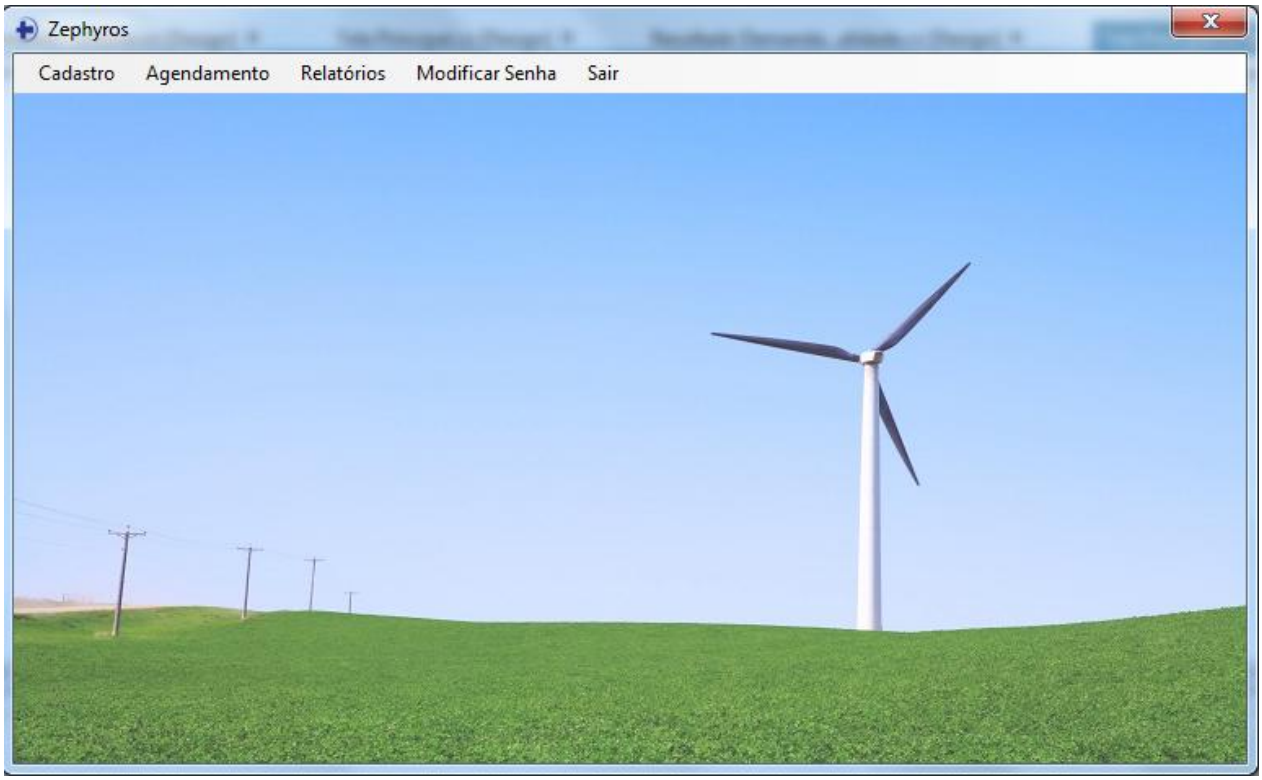
Como resultado do presente trabalho, foi desenvolvido o Zephyros, um sistema que abrange a gestão de agendamento de consultas médicas (especialidades) e exames no setor de regulação das Unidades Básicas de Saúde do Programa Saúde da Família, com o objetivo de entregar melhoria de processo, segurança e facilidade na informatização do setor de regulação em relação ao sistema utilizado atualmente (NCR).

O Zephyros possui todas as opções, telas e funcionalidades existentes no sistema utilizado atualmente, porém foi acrescentado diversas novas funcionalidades que abrangem desde melhorias na consistência das informações inseridas no sistema até novas opções de cadastramento e relatórios.

O desenvolvimento do Zephyros foi realizado por meio do *Microsoft Visual Studio*, utilizando *Windows Forms* para criação do sistema em conjunto com a linguagem de programação orientada a objetos C#. Para o banco de dados que mantém armazenadas as informações inseridas no sistema foi utilizado o SGBD *MySQL*.

A figura 7 demonstra a tela inicial do sistema Zephyros, com os menus que podem ser acessados:

Figura 7 - Tela inicial Zephyros



Fonte: Elaborado pelo autor.

4.1.1 Funcionalidades

Para explicar as funcionalidades do Zephyros, foi realizado um detalhamento do sistema e demonstrado cada tela e usa função e importância. São tópicos que abordam as telas de: *login*, modificação de senha, cadastro de paciente, cadastro de exame, cadastro de especialidade, cadastro de profissional, agendamento de exame, agendamento de especialidades e relatórios.

- ***Login***

A tela de *login* contempla os campos de usuário e senha e realiza as validações necessárias ao acesso do usuário, caso o mesmo preencha incorretamente as informações.

Em acréscimo às funcionalidades comuns, há a possibilidade do usuário alterar a senha de acesso ao sistema caso for seu primeiro acesso. A alteração de senha esta condicionada à validação do sistema, pois o usuário não pode inserir como nova senha a mesma utilizada como atual.

Uma funcionalidade nova desenvolvida na tela de *login* é o filtro de acesso e localidade por *login*. O filtro de acesso é referente a possibilidade do usuário somente acessar as telas permitidas ao seu tipo de usuário, sendo assim é possível criar diferentes perfis de acesso para usuários com necessidades e permissões de acesso diferentes. A localidade por *login* funciona como uma configuração que permite que todas as ações que o usuário realize no sistema (inserção, consulta, exclusão, geração de relatórios) fiquem atreladas somente a unidade que o mesmo foi cadastrado. Exceto nos casos de usuários com acesso de gestor e administrador do sistema, que podem ter seus *logins* atrelados a todas as unidades. A criação e configuração de usuários, assim como a explicação detalhada das funcionalidades relacionadas ao filtro de acesso e localidade por *login* são apresentadas posteriormente na seção "Cadastro de Usuários".

A figura 8 demonstra a tela de login do sistema Zephyros:

Figura 8 - Tela de login

Fonte: Elaborado pelo autor.

- **Modificação de senha**

A tela de modificação de senha é uma implementação que visa proporcionar ao usuário a possibilidade de alterar sua senha assim que julgar necessário. Para realizar a alteração é preciso que o usuário insira as informações do seu *login,* a senha atual e a nova senha desejada. É importante que tais informações estejam corretamente

preenchidas, pois caso o usuário preencha incorretamente algum dos campos, não será permitida a alteração de senha.

A figura 9 demonstra a tela de modificação de senha do sistema Zephyros:

Figura 9 - Tela de modificação de senha

Fonte: Elaborado pelo autor.

- **Cadastro de paciente**

Na tela de cadastro de paciente foram mantidos todos os campos que existem no software legado ao qual o presente trabalho tem como inspiração, porém foram realizadas diversas melhorias relacionadas à validação de campos e informações.

Primeiramente é importante citar que há campos obrigatórios que devem ser preenchidos pelo usuário para realizar o cadastro do paciente. Tais campos são identificados por um asterisco ao fim do texto que o identifica.

Além do preenchimento dos campos obrigatórios, o cadastro do paciente também é condicionado a validação de alguns campos por critérios pré-definidos. O campo de data nascimento não pode ser preenchido com uma data posterior a data atual. Os campos RG, CPF e CNS não são chaves no cadastro de paciente (a chave utilizada é a matricula), porém tais campos não podem ser duplicados com diferentes cadastros de pacientes, independente da unidade em que o paciente está cadastrado. Para tal objetivo, o sistema realiza uma comparação das informações contidas nos campos do sistema com as informações presentes no banco de dados. Caso houver algum paciente já incluso no sistema com alguma destas informações, o usuário é

notificado e o cadastro não é salvo até que o usuário preencha as informações corretamente e tente salvar novamente.

Ao salvar o cadastro do paciente, é exibido um *pop-up* com o número da matrícula gerada do paciente. Caso for necessário atualizar posteriormente o cadastro do paciente incluso no sistema, é necessário realizar apenas uma pesquisa na tela de cadastro de paciente utilizando o número de CNS como parâmetro.

A figura 10 corresponde à tela de cadastro de paciente do sistema Zephyros:

Figura 10 - Tela de cadastro de paciente

Fonte: Elaborado pelo autor.

- **Cadastro de exame e de especialidade**

As telas de cadastro de especialidade e exame contemplam além da possibilidade de simplesmente incluir uma especialidade ou exame no sistema ou alterá-los, é possível também atrelar tal especialidade e exame a duas informações fundamentais: profissional permitido e sexo permitido.

O campo profissional permitido define qual profissional poderá encaminhar a especialidade cadastrada ou solicitar um exame. De acordo com regras de negócio pré-definidas, foi realizada uma carga de cadastro de especialidades e exames no sistema com suas respectivas informações de profissionais permitidos, divididos entre médico,

dentista e enfermeiro. Tal configuração é necessária para o funcionamento correto da tela de agendamento de especialidade e de exame, para que possa ser realizado filtro na lista de especialidade e exame evitando assim erro por parte do usuário ao agendar um encaminhamento de especialidade ou solicitação de exame não permitido a um profissional.

A informação de sexo permitido é importante, assim como o profissional permitido, para filtrar as especialidades e exames que podem ser selecionadas na tela de agendamento de especialidade e de agendamento de exame, por meio de um comparativo entre o sexo do paciente e o sexo permitido do cadastro da especialidade e do exame.

As figuras 11 e 12 correspondem às telas de cadastro de exames e cadastro de especialidade respectivamente, do sistema Zephyros:

Figura 11 - Tela de cadastro de exames

Fonte: Elaborado pelo autor.

Figura 12 - Tela de cadastro de especialidade

Fonte: Elaborado pelo autor.

- **Cadastro de profissional**

A tela de cadastro de profissional faz parte do conjunto de funcionalidades do sistema utilizada para configuração do sistema e carga inicial. O cadastro de profissional deve ser realizado previamente à utilização do sistema, pois os profissionais cadastrados nesta tela são selecionados nas telas de agendamento de especialidades e de exames.

Todos os campos nesta tela são de preenchimento obrigatório, porém é importante ressaltar a importância do campo número do conselho e função. O campo número do conselho tem grande importância na tela de cadastro de profissional, pois é utilizado como parâmetro para busca de profissionais quando for necessário editar o cadastro do profissional.

O campo função deve ser essencialmente preenchido corretamente, pois a seleção de exames e de especialidades em suas respectivas telas de agendamento depende da função preenchida no cadastro do profissional para exibir corretamente os resultados de sua lista. Esta funcionalidade será explicada melhor posteriormente na seção correspondente à "Agendamento de exames e agendamento de especialidades".

A figura 13 corresponde à tela de cadastro de profissional do sistema Zephyros:

Figura 13 - Tela de cadastro de profissional

Fonte: Elaborado pelo autor.

- **Cadastro de usuário**

Na tela de cadastro de usuários é possível criar três tipos de usuários de acesso ao sistema: usuário, gerente e administrador. Além da diferenciação de tipo de acesso, também é preciso configurar a qual unidade o usuário pertence.

A finalidade de criar diferentes perfis de acesso ao sistema é para assegurar que os usuários acessem somente as funcionalidades do sistema a que lhe são permitidos. Sendo assim, caso um usuário esteja cadastrado como usuário comum seu acesso será restrito às funcionalidades comuns e operacionais, sendo elas: cadastro de paciente, agendamento de exames, agendamento de especialidades, alteração senha e geração de relatórios. Um usuário cadastrado com perfil de gerente (gestor do serviço de regulação de todas as unidades) tem acesso ao cadastro de especialidades, cadastro de exames, cadastro de profissionais, alteração de senha e geração de relatórios. O usuário com perfil de administrador (neste caso administrador do sistema) tem acesso a todas as funcionalidades do sistema.

A configuração de unidade no cadastro de usuário é uma das funcionalidades mais importantes no desenvolvimento do sistema apresentado neste trabalho. Um dos maiores problemas enfrentados pelos gestores do serviço de regulação das unidades que utilizam o sistema NCR, como explicado no capítulo 3, é a descentralização do banco de dados, onde cada unidade possui sua própria base para armazenar informações da regulação. No sistema desenvolvido no presente trabalho, tal problema foi sanado, pois a base de dados é centralizada e pode ser configurada com diversas unidades que compartilham o mesmo banco de dados. E tal funcionalidade é possível devido à configuração de unidade no cadastro de usuário.

É possível cadastrar um usuário no sistema e atrelá-lo somente a unidade ao qual ele pertence, sendo assim, ao acessar o sistema, toda informação inserida, alterada ou consultada no sistema possui um identificador referente a unidade que o usuário pertence. Tal identificador é utilizado pelo sistema desde o *login* até o momento em que o usuário sair do sistema. Usuários comuns podem pertencer a somente uma unidade, porém usuários de gestores e administradores de sistemas podem estar configurados em quantas unidades for necessário. Sendo assim, o gestor pode gerar relatórios que possuam informações de todas as unidades cadastradas no sistema, facilitando o processo de análise de informações e melhorando consideravelmente o processo de trabalho estratégico do gestor, que deixará de perder tempo solicitando relatórios a todas as unidades e consolidando-os manualmente, para focar em planos estratégicos de melhoria no serviço.

A figura 14 demonstra a tela de cadastro de usuários do sistema Zephyros:

Figura 14 - Tela de cadastro de usuários

Fonte: Elaborado pelo autor.

- **Agendamento de exames e especialidades**

Duas funcionalidades essenciais no Zephyros são o agendamento de consultas com médicos especialistas (especialidades) e o agendamento de exames. Ambas funcionalidades de agendamento estão divididas em telas diferentes, acessíveis por meio do menu de agendamentos, porém devido a grande semelhança entre ambas, serão especificadas no mesmo tópico no presente trabalho. A diferença entre as duas telas esta somente na lista de exames (na tela de agendamento de exames) e na lista de especialidades (na lista de agendamento de especialidades).

Nestas telas o usuário comum tem a possibilidade de realizar o agendamento de exames e especialidades para os pacientes cadastrados. Caso trate-se de um novo agendamento, o usuário deve clicar em "Nova Consulta" para que o campo de pesquisa utilize como parâmetro de busca o número do CNS do cadastro do paciente. Caso a necessidade do usuário for atualizar as informações de um agendamento já existente

no sistema, ao clicar em "Consulta Existente" o campo de pesquisa tem seu parâmetro alterado para o código do agendamento de exame.
Ambas telas foram desenvolvidas com base em diversas regras de negócio baseadas em parâmetro de processo do setor de regulação e definições do SUS.

Uma regra de negócio aplicada ao sistema nestas telas é em relação ao processo do setor de regulação. A primeira ação realizada pelo usuário é agendar o encaminhamento de exame ou especialidade, sendo assim a data do pedido é obrigatória e não pode ser superior a data atual. Sem esta data não é possível realizar o agendamento da especialidade ou exame. Após isto, quando o agendamento for atualizado, o usuário preencherá a data de contato com o paciente, que não poderá ser maior que a data da marcação (pois ele deve consultar o paciente antes e saber de sua disponibilidade antes de confirmar o agendamento) e por fim preencher a data da marcação. Para tais campos não apresentarem conflitos entre si foram criadas diversas regras de validação no sistema que impedem que ocorra erro de preenchimento do usuário, alertando-o em diversas situações com mensagens.

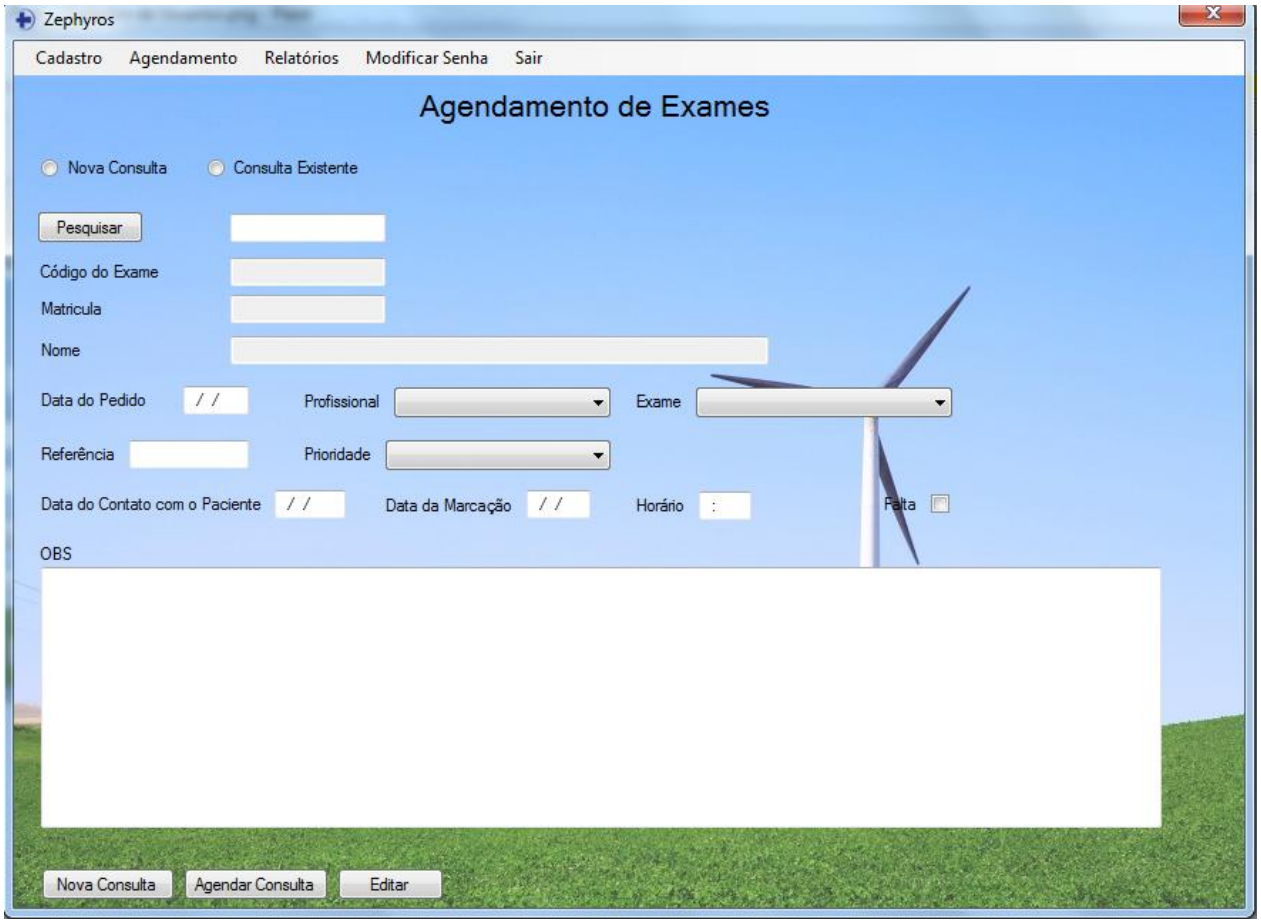
Outra particularidade importante e inovadora no Zephyros em comparação com o sistema em qual o presente trabalho foi inspirado é a possibilidade de filtrar os exames e especialidades por sexo do paciente e função dos profissionais. Ao selecionar um paciente são exibidos na lista somente exames ou especialidades que possam ser agendados para o sexo do paciente selecionado ou para ambos os sexos. A lista de exames e especialidades também é filtrada por profissional, sendo assim, quando o usuário selecionar o nome do profissional o sistema filtra a lista de acordo com a função do profissional configurada na tela de cadastro de profissional. Tais filtros da lista de exames e de especialidades foram definidos com base em regras de negócio que especificam quais exames e especialidades cada profissional esta autorizado a solicitar.

Por fim, é importante citar a funcionalidade em relação ao campo de "Falta" no sistema. Este campo é de grande importância estratégica para os gestores do serviço de regulação, pois permite saber estatisticamente o absenteísmo de pacientes nas consultas e exames agendados. Ao salvar um agendamento com a falta marcada, o usuário está informando ao sistema que tal agendamento está finalizado e não sofrerá

mais alterações, sendo assim o sistema "encerra" o agendamento, bloqueando-o para alterações futuras e o deixa disponível somente para consulta do usuário.

A figura 15 exibe a tela de agendamento de exames do sistema Zephyros:

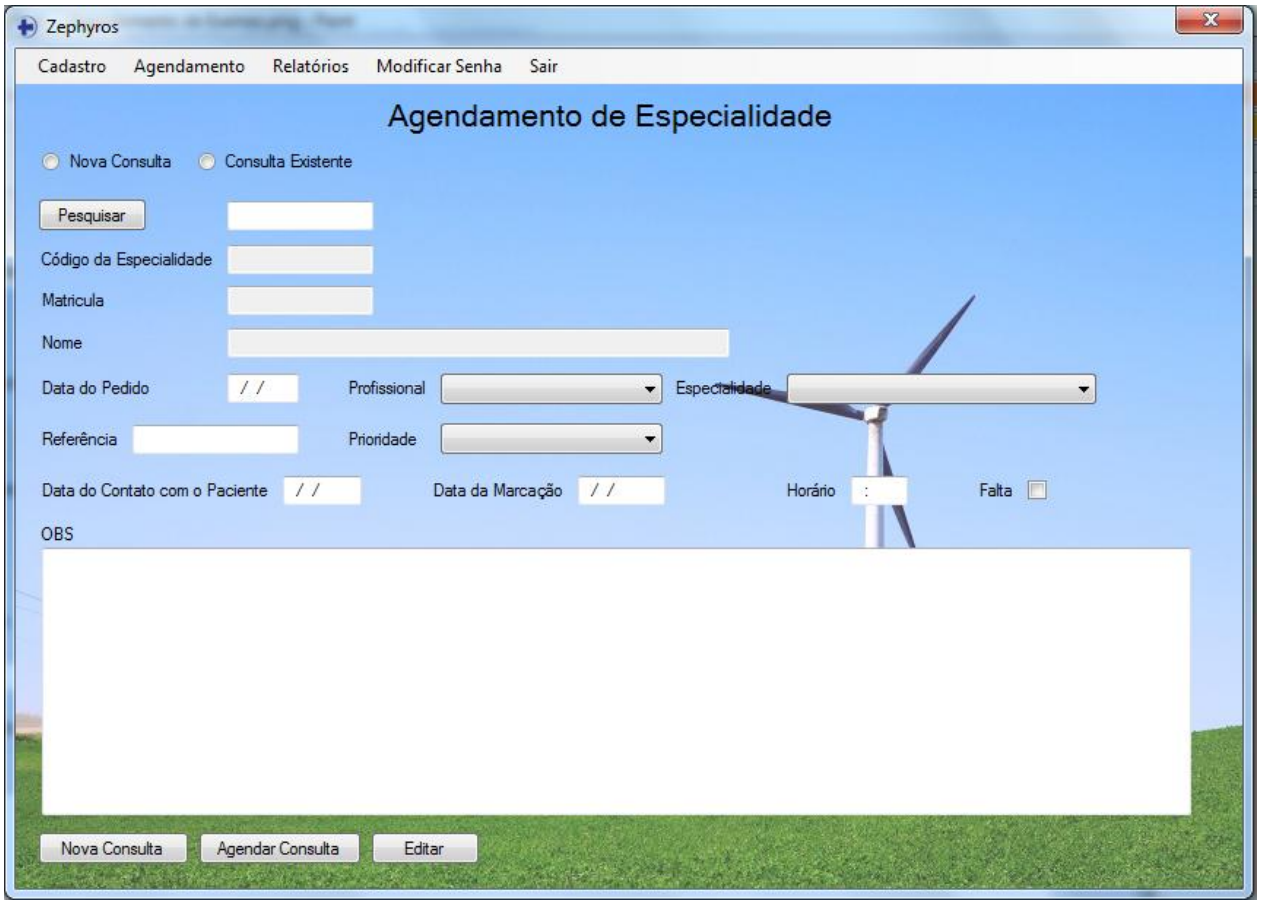
Figura 15 - Tela de agendamento de exames



Fonte: Elaborado pelo autor.

A figura 16 exibe a tela de agendamento de especialidade do sistema Zephyros:

Figura 16 - Tela de agendamento de especialidade



Fonte: Elaborado pelo autor.

- **Relatórios**

Dentre as funcionalidades do sistema, uma das mais importantes é a de gerar relatórios. Os relatórios emitidos pelo sistema são baseados na informação inserida e tem finalidade de auxiliar usuários comuns e gestores a controlar as informações existentes e criar estratégias a partir destas informações. Os relatórios do sistema são divididos em relatórios de exames e relatórios de especialidades.

Para emitir o relatório o sistema disponibiliza ao usuário a possibilidade de definir parâmetros de pesquisa, para assim conseguir adequar o resultado do relatório com sua necessidade. Tais parâmetros são: unidade, exame, especialidade, data do pedido e prioridade. É importante salientar que a emissão dos relatórios está condicionada ao

login do usuários, sendo assim, caso o utilizador do sistema for um usuário comum, somente poderá emitir relatórios da sua própria unidade. Porém, caso o usuário do sistema for um gestor ou o administrador do sistema, os relatórios poderão ser emitidos sem limitação de unidades, sendo possível selecionar todas as necessárias. Ao emitir o relatório, o usuário pode imprimir.

O sistema abrange todos os relatórios já existentes no sistema utilizado atualmente no setor de regulação das unidades, porém também há diversos relatórios adicionais no sistema que atendem às necessidades atuais dos gestores do setor de regulação:

- Exames agendados;
- Exames solicitados;
- Exames solicitados por profissional;
- Demanda reprimida por exame;
- Absenteísmo total por exame;
- Absenteísmo individual por exame;
- Absenteísmo total de exames por unidade;
- Especialidades agendadas;
- Especialidades solicitadas;
- Especialidades solicitadas por profissional;
- Demanda reprimida por especialidade;
- Absenteísmo total por especialidade;
- Absenteísmo individual por especialidade;
- Absenteísmo total de especialidades por unidade.

A figura 17 exibe um exemplo de tela de geração de relatório:

Figura 17 - Exemplo de tela de geração de relatório

Fonte: Elaborado pelo autor.

4.1.2 Vantagens e desvantagens

O Zephyros, assim como qualquer sistema, apresenta vantagens e desvantagens. Para mensurar tais vantagens e desvantagens, foi realizada uma análise em comparação com o sistema utilizado atualmente no setor de regulação das Unidades Básicas de Saúde, o NCR. Sendo assim, é importante exemplificar as conclusões de tal análise, divididas nos tópicos de vantagens e desvantagens.

Vantagens

- **Banco de dados único:** o maior problema existente no sistema utilizado atualmente no setor de regulação (NCR) é a fragmentação do banco de dados, situação que faz com que cada UBS possua seu próprio banco de dados, dificultando assim além da extração de relatórios, também a manutenção do sistema. O Zephyros possui um único banco de dados, que unifica todas

unidades e centraliza as informações inseridas no sistema, facilitando assim a extração de relatórios e manutenções quando necessário.

- **Independência do pacote *Office*:** o sistema NCR, desenvolvido em *Microsoft Access*, depende do pacote *Office* para funcionar e além disso depende que o pacote *Office* seja da mesma versão da qual o sistema foi desenvolvido, pois caso contrário o mesmo pode apresentar problemas de execução. O Zephyros não possui essa dependência, pois foi desenvolvido em *Windows Forms,* voltado à utilização em qualquer máquina que possua como sistema operacional o *Microsoft Windows.*

- **Validações de campos:** uma adição importante de funcionalidade no Zephyros em relação ao NCR é a validação de diversos campos. Foram desenvolvidas validações em campos de documentos (que não podem ser duplicados), de datas (de acordo com regras de negócio) e de exames e especialidades (de acordo com regras de negócio e definições gerais do SUS). Sendo assim, com tais validações aplicadas, as informações contidas no banco de dados tornam-se muito mais confiáveis e corretas, pois as validações evitam que ocorra erro por parte do usuário.

- **Perfis de acesso:** funcionalidade inexistente no sistema utilizado atualmente no setor de regulação (NCR), foi implementada no Zephyros com o objetivo de especificar com precisão "quem pode acessar o que", dividindo os usuário em: usuários comuns, gestores e administradores do sistema.

- **Relatórios:** o Zephyros possui todos os relatórios existentes no sistema NCR e também abrange outros relatórios importantes e estratégicos que não existem no NCR. Vantagem também associada aos relatórios é a possibilidade de gerar relatórios gerenciais com informações de todas as unidades existentes no sistema, devido a centralização de informação em um banco de dados único.

- **Configurável:** o Zephyros possibilita ao usuário a capacidade de configurar o sistema de acordo com a necessidade e as normas estabelecidas pelo SUS, sendo assim, sempre que necessário é possível alterar o cadastro de exames, especialidades e usuários no sistema. Caso houver a necessidade de alterações no sistema, acréscimo de novas funcionalidades ou de novas telas, as alterações são facilitadas devido ao desenvolvimento do sistema ter sido realizado a partir de uma linguagem de programação orientada a objeto.

- **Segurança dos dados**: com a utilização do sistema NCR, cada unidade tem seu próprio banco de dados, que fica armazenado localmente em um computador na unidade, sendo assim qualquer problema que comprometa o disco rígido do computador que possui o sistema também comprometerá os dados salvos no banco do sistema. O Zephyros possui a proposta de armazenar o banco de dados em um servidor que possua rotinas de *backup* além de estratégias de redundância e recuperação de dados, para evitar qualquer tipo de perda de informações.

Desvantagem

- **Dependência de internet:** uma desvantagem identificada no Zephyros é a necessidade do uso de internet para sua utilização. O sistema (desenvolvido em *Windows Forms*) funciona localmente sem a necessidade de internet, porém o planejamento de implantação futura do sistema possui uma abordagem de alocação da aplicação em um servidor para acesso via conexão remota (*Remote Desktop)* para que as unidades possam acessá-lo, sendo assim, caso houver falha no link de internet da unidade o sistema ficará inacessível. Isto não acontece atualmente, pois o sistema NCR possui aplicação e banco de dados alocados localmente em cada unidade.

5 CONCLUSÕES

Neste trabalho foi apresentado os fundamentos e assuntos principais relacionados a área de sistemas de informação e saúde pública que tem influência direta no desenvolvimento do trabalho. Com base nos conteúdos abordados e na experiência da utilização de sistemas voltados a informatização do setor de regulação das unidades básicas de saúde, foi desenvolvido um sistema que possui todas as funcionalidades do sistema já utilizado atualmente no setor de regulação (NCR), porém acrescenta diversas novas funcionalidades e melhorias de processo ao setor.

O sistema desenvolvido Zephyros cumpre seu objetivo, pois dispõe de banco de dados único, que centraliza as informações inseridas e facilita com que os gestores do serviço de regulação das UBS consigam gerar relatórios com facilidade e abrangência de todas as unidades inclusas no sistema.

A implementação em campo do sistema é um plano futuro a ser realizado, para que o setor de regulação possa migrar do sistema atual para o sistema desenvolvido neste trabalho sem impacto nas rotinas de trabalho e processos internos da unidade. Esta implementação depende da preparação de ambiente, testes em servidor, alocação de espaço para aplicação e banco de dados no servidor, disponibilização de acesso *remote desktop* multiusuário e resolução de diversas questões burocráticas.

É tranquilamente possível a utilização do sistema em sua versão atual, porém foi verificada que há a necessidade de melhorias futuras como adição de novos relatórios, maior detalhamento do cadastro de usuários e profissionais e desenvolvimento de um método que possibilite a desativação de profissionais sem que isto afete os agendamentos relacionados a ele (por tal motivo, na versão atual não há a opção de exclusão de profissionais). Uma ideia para implementação futura é o desenvolvimento de uma versão que funcione via *web,* com aplicação e banco de dados hospedados em servidores seguros e que possibilite ao usuário acessar o sistema via *browser.*

Em geral, é possível concluir que o sistema atende a expectativa definida e excede em diversos aspectos a ideia concebida inicialmente com base no sistema utilizado atualmente no setor de regulação das unidades, tendo apresentado como

resultado final uma ferramenta eficiente para a gestão informatizada das consultas médicas e exames no setor de regulação.

REFERÊNCIAS

ALECRIM. Emerson. **Banco de dados MySQL e PostgreSQL**. 2008. Disponível em: <http://www.infowester.com/postgremysql.php>. Acesso em 15 out. 2012.

CAMPOS, Carlos. **Requisitos do usuário**. 2010. Disponível em: <http://carloscamposinfo.com/cjec/?p=217>. Acesso em 06 out. 2012.

CRUZ, Fundação Oswaldo. **A história do SUS**. 2008. Disponível em: <http://157.86.8.13/sus20anos/index.php?option=com_content&view=article&id=54:a-historia-do-sus&catid=31:geral>. Acesso em 15 out. 2012.

DINIZ, Samuel. **O problema do software legado.** 2010. Disponível em: <http://www.infoblogs.com.br/view.action?contentId=205651>. Acesso em 03 out. 2012.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. 6. ed. São Paulo: Pearson, 2011.

FERNANDES, Horácio. **Evolução das Linguagens de Programação**. 2009. Disponível em:<http://www.dcc.ufam.edu.br/horacio/images/stories/Disciplinas/PP/PP-Aula3EvolucaoLinguagens.pdf>. Acesso em 06 out. 2012.

GUEDES, Gilleanes T. A. **UML 2 – Guia Prático**. 1. ed. São Paulo: Novatec, 2007.

GUEDES, Gilleanes T. A. **UML 2 – Uma Abordagem Prática**. 2. ed. São Paulo: Novatec, 2011.

HOOKER. David. **Seven Principles Of Software Development**. 1996. Disponível em: <http://c2.com/cgi/wiki?SevenPrinciplesOfSoftwareDevelopment>. Acesso em 16 out. 2012.

JENNY, Juliana. **Gestão de Requisitos**. 2012. Disponível em: <http://julianakolb.files.wordpress.com/2012/04/tabela.jpg>. Acesso em 19 out. 2012.

MACORATTI, José Carlos. **Padrões de Projeto - Os 7 princípios básicos do desenvolvimento de software**. 2011. Disponível em: <http://www.macoratti.net/11/05/sd_prnc1.htm>. Acesso em 03 out. 2012.

MACORATTI, José Carlos. **UML – Diagrama de Classes e objetos**. 2005. Disponível em: <http://www.macoratti.net/net_uml6.gif>. Acesso em 19 out. 2012.
MENDES, Silas. **DML, DDL?!? O que é isso?.** 2010. Disponível em: <http://silasmendes.com/dba/dml-ddl-o-que-e-isso/>. Acesso em 14 out. 2012.

MACORATTI, José Carlos. **UML – Modelagem de projeto OOP completo em VB.NET**. 2010 Disponível em: < http://www.macoratti.net/vbn_ouc2.gif>. Acesso em 19 out. 2012.

MICROSOFT. **Atributos (guia de programação do C#).** 2005. Disponível em: <http://msdn.microsoft.com/pt-br/library/z0w1kczw(v=vs.80).aspx>. Acesso em 15 out. 2012.

MICROSOFT. **Class.** 2003. Disponível em: <http://msdn.microsoft.com/enus/library/0b0thckt(v=vs.71).aspx>. Acesso em 15 out. 2012.

MICROSOFT. **Introdução à linguagem C# e o .NET Framework**. 2012. Disponível em: <http://msdn.microsoft.com/library/vstudio/z1zx9t92>. Acesso em 15 out. 2012.

MICROSOFT. **Métodos (guia de programação do C#).** 2012. Disponível em: <http://msdn.microsoft.com/pt-br/library/ms173114.aspx>. Acesso em 15 out. 2012.

MICROSOFT. **Polimorfismo (guia de programação do C#)**. 2012. Disponível em: <http://msdn.microsoft.com/pt-br/library/ms173152.aspx>. Acesso em 15 out. 2012.

PAULISTA, Universidade Estadual. **Requisitos de Software**. 2005. Disponível em: <http://www.dcce.ibilce.unesp.br/~ines/cursos/eng_soft/aula04.pdf>. Acesso em 06 out. 2012.

PAULO, Diário Oficial da Cidade de São. **ATA da 2.497ª sessão (ordinária).** 2012. Disponível: <http://www.tcm.sp.gov.br/atas/OrdExp/2497.pdf>. Acesso em 21 out. 2012.

PORTAL, Educação**. História e características da linguagem C#.** 2008. Disponível em: <http://www.portaleducacao.com.br/informatica/artigos/6137/historia-caracteristicas-da-linguagem-c>.Acesso em 06 out. 2012.

PRESSMAN, Roger S. **Engenharia de Software**. 6. ed. Porto Alegre: AMGH, 2010.

ROSA, Walisete de Almeida Godinho, LABATE, Renata Curi. **Programa Saúde da Família: A construção de um novo modelo de assistência**. 2005. Disponível em: <http://www.scielo.br/pdf/rlae/v13n6/v13n6a16.pdf>. Acesso em 18 out. 2012.

SANTOS, Thiago. **M.E.R. – Modelo de Entidade e Relacionamento**. 2010. Disponível em: <http://sql-tts.blogspot.com/2012/04/mer.html >. Acesso em 20 out. 2012.

SAÚDE, Ministério da. **Atendimento - SUS**. 2012. Disponível em: <http://www.brasil.gov.br/sobre/saude/atendimento>. Acesso em 15 out. 2012.

SAÚDE, Ministério da. **Diretrizes para a Implantação de Complexos Reguladores**.1. ed. Brasília: MS, 2006.

SAÚDE, Portal da**. Programa Saúde da Família**. 2012. Disponível em: <http://portal.saude.gov.br/portal/saude/cidadao/area.cfm?id_area=149>. Acesso em 18 out. 2012.

SAÚDE, Portal da**. Sistema Único de Saúde**. 2012. Disponível em: <http://portal.saude.gov.br/portal/saude/visualizar_texto.cfm?idtxt=24627>. Acesso em: 15 out. 2012.

SAÚDE, Secretaria Municipal da**. Portaria 2566/2011 - SMS.G**. Disponível em: <http://extranet.saude.prefeitura.sp.gov.br/areas/crsleste/regulacao/diretrizes/Portaria%20SMS-G%20No%202566-2011.pdf>. Acesso em 21 out. 2012.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 5. ed. Rio de Janeiro: Elsevier, 2006.

SUL, Coordenadoria Regional de Saúde Sul**. Reestruturação do Setor de Regulação nas Unidades da CRSSUL**. São Paulo: 2010.

TERSKI, Matt. **Master the Use Case Include Relationship**. 2005. Disponível em:<http://blog.casecomplete.com/image.axd?picture=WindowsLiveWriter/MastertheUseCaseIncludeRelationship_EABC/IncludeUseCaseDiagram_3.png>. Acesso em 19 out. 2012.

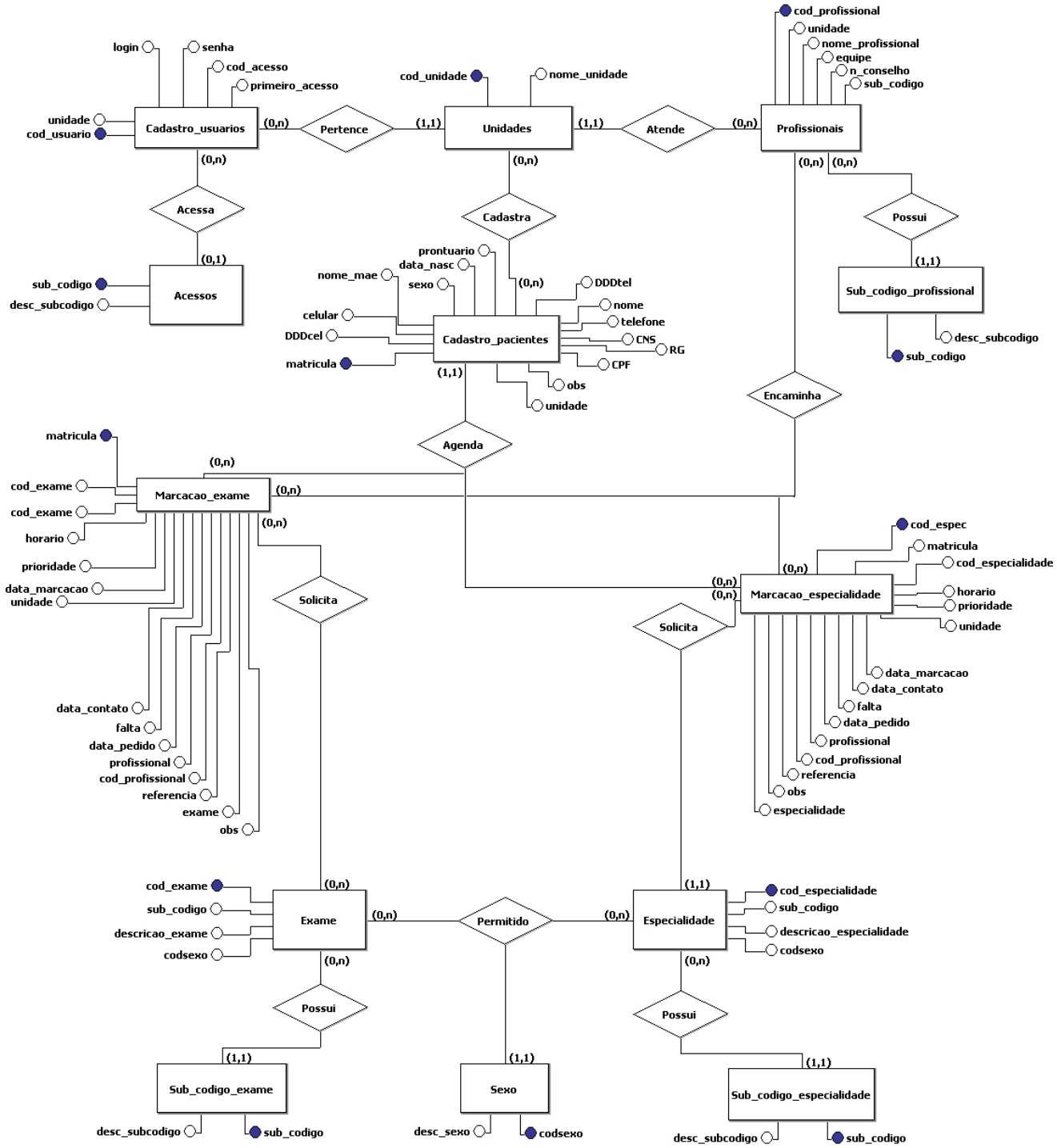
UNICAMP. **Modelo Entidade-Relacionamento**. 2009. Disponível em: <http://www.ic.unicamp.br/~beatriz/cursos/mc536/slides/MER-p-pagina.pdf>. Acesso em 13 out. 2012.

WILLRICH, Roberto**. Linguagens de Programação**. 2005**.** Disponível em: <http://algol.dcc.ufla.br/~monserrat/icc/Introducao_linguagens.pdf>. Acesso em 06 out. 2012.

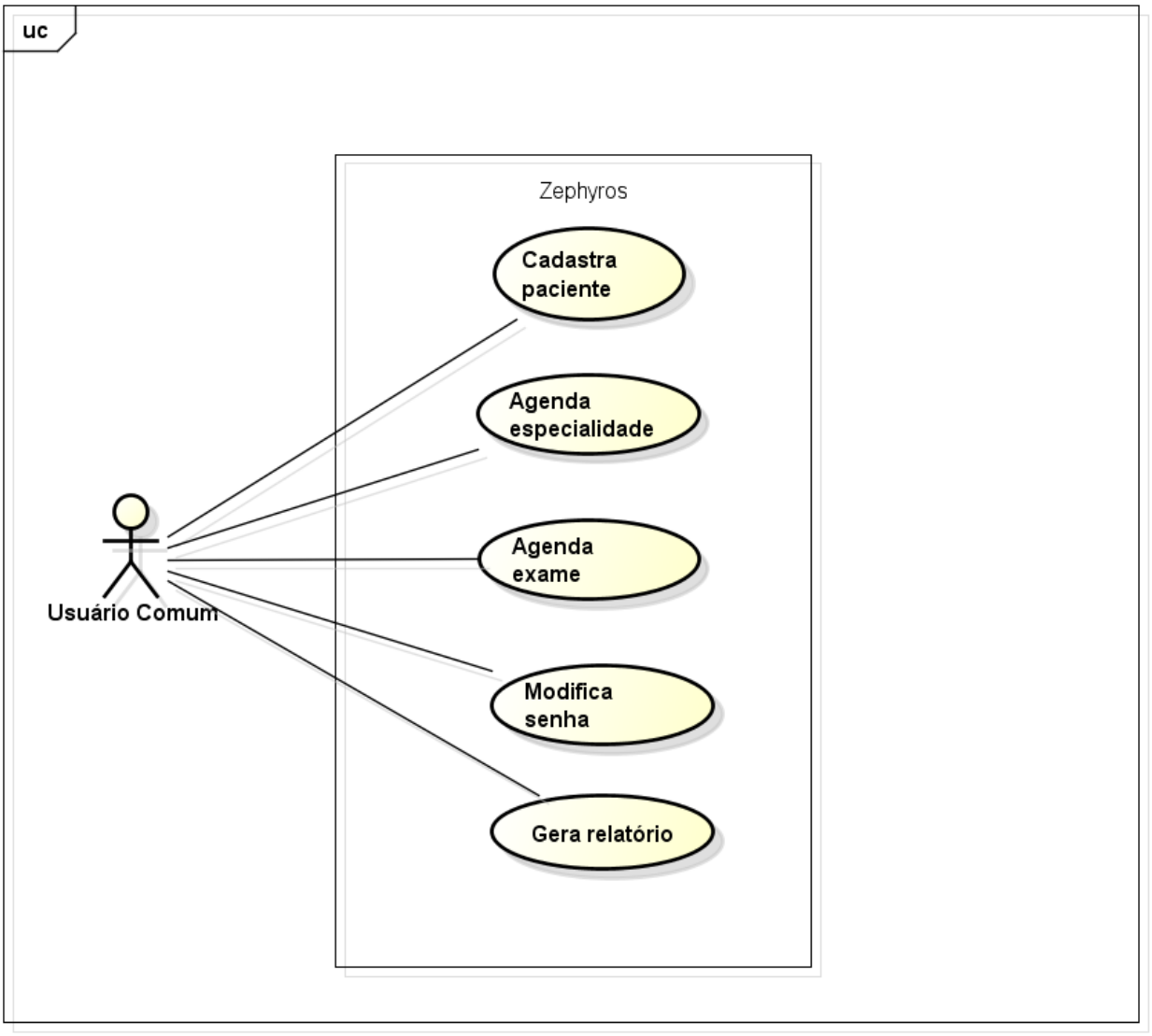
YEUNG, Jick. **Using Duration Constraint in Sequence Diagram**. 2010. Disponível em: <http://s1.linkvp.com/vpuml/tutorials/durationconstraint_screenshots/20101004/01sample-sequence-diagram.png>. Acesso em 19 out. 2012.

APÊNDICE: DOCUMENTAÇÕES PRINCIPAIS DO SISTEMA ZEPHYROS

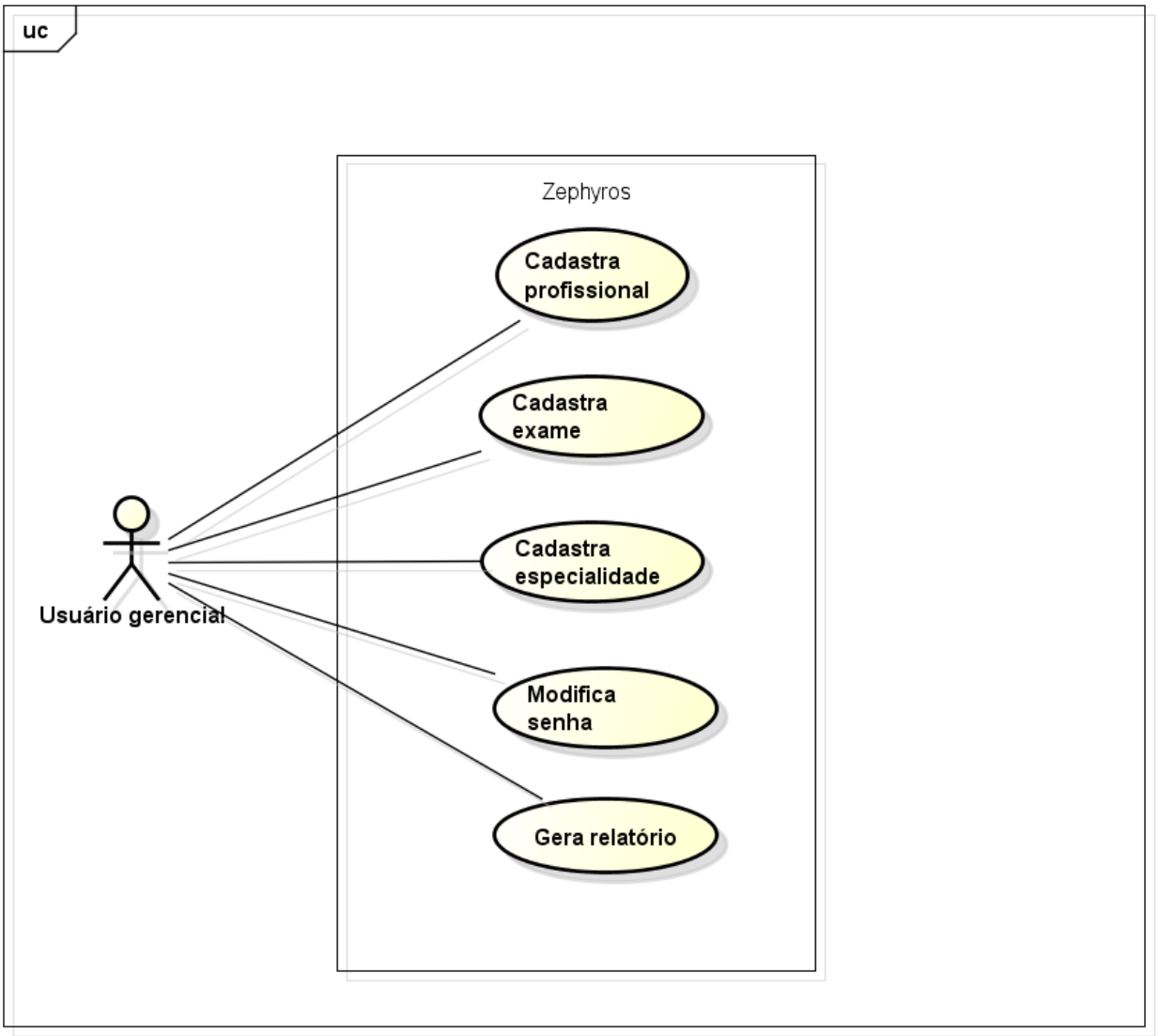
Apêndice A – Modelo Entidade-Relacionamento



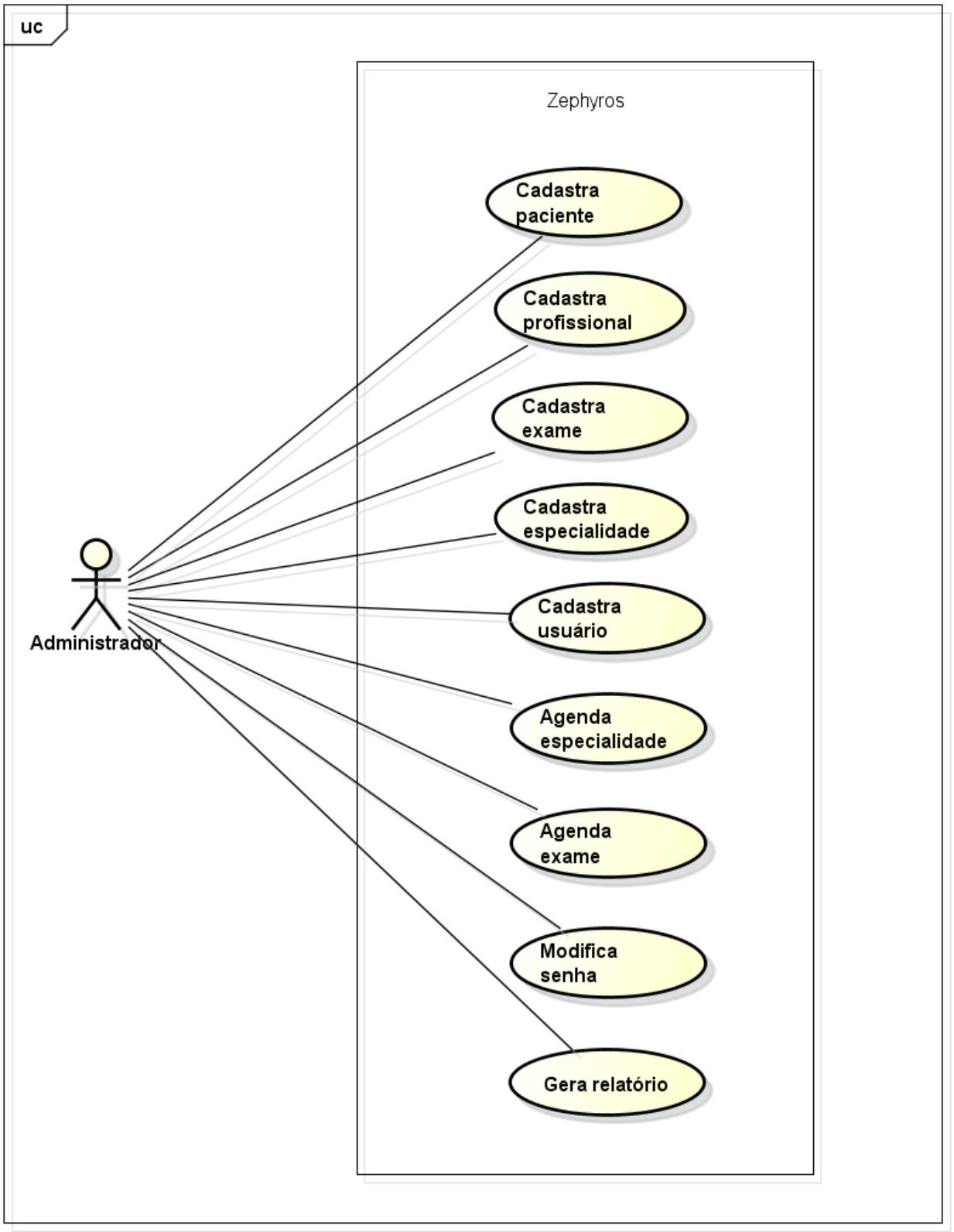
Apêndice B – Caso de Uso (Permissões de acesso – Usuário Comum)



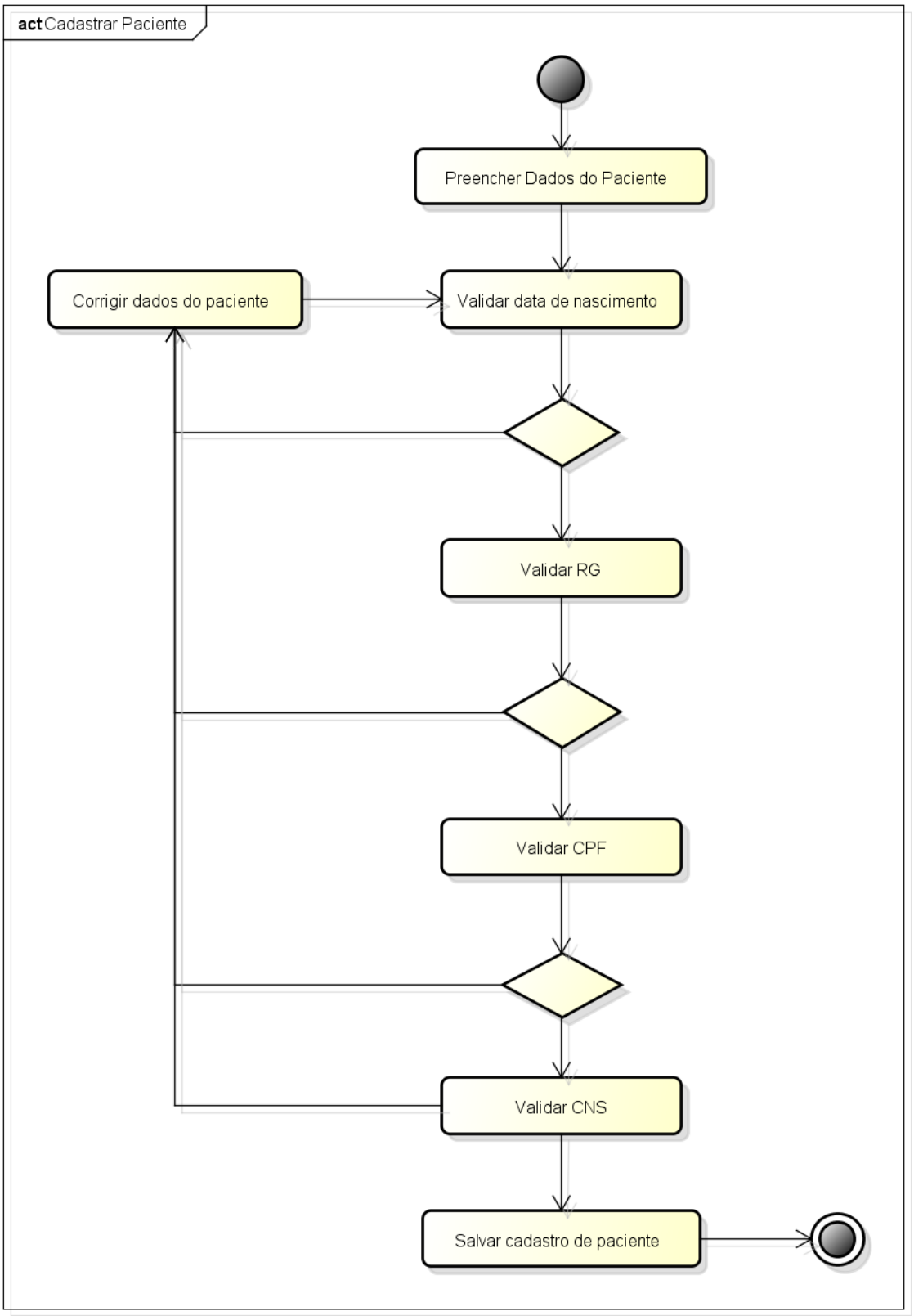
Apêndice C – Caso de Uso (Permissões de acesso – Usuário Gerencial)



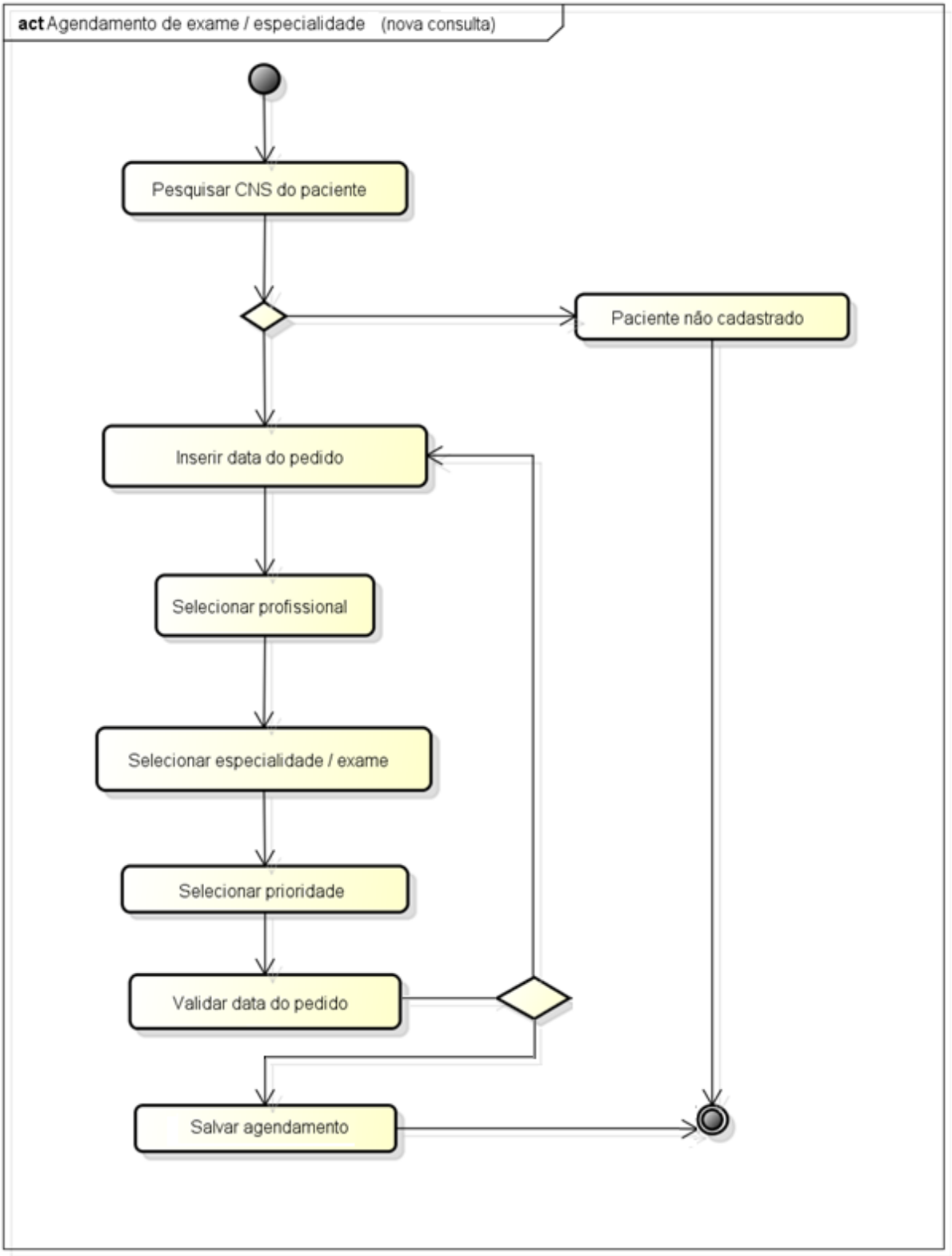
Apêndice D – Caso de Uso (Permissões de acesso – Administrador do Sistema)



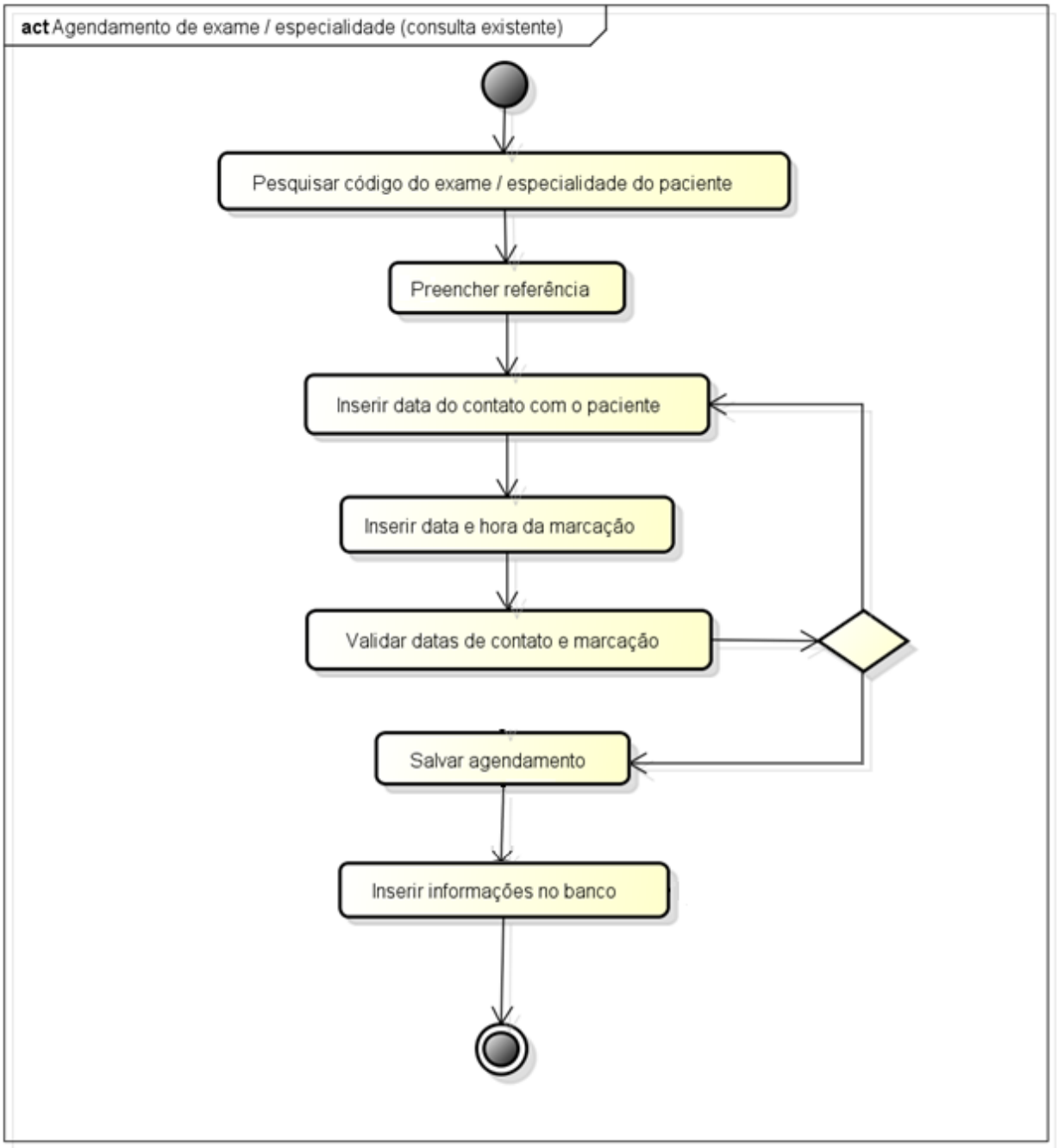
Apêndice E – Diagrama de Atividade – Cadastrar Paciente



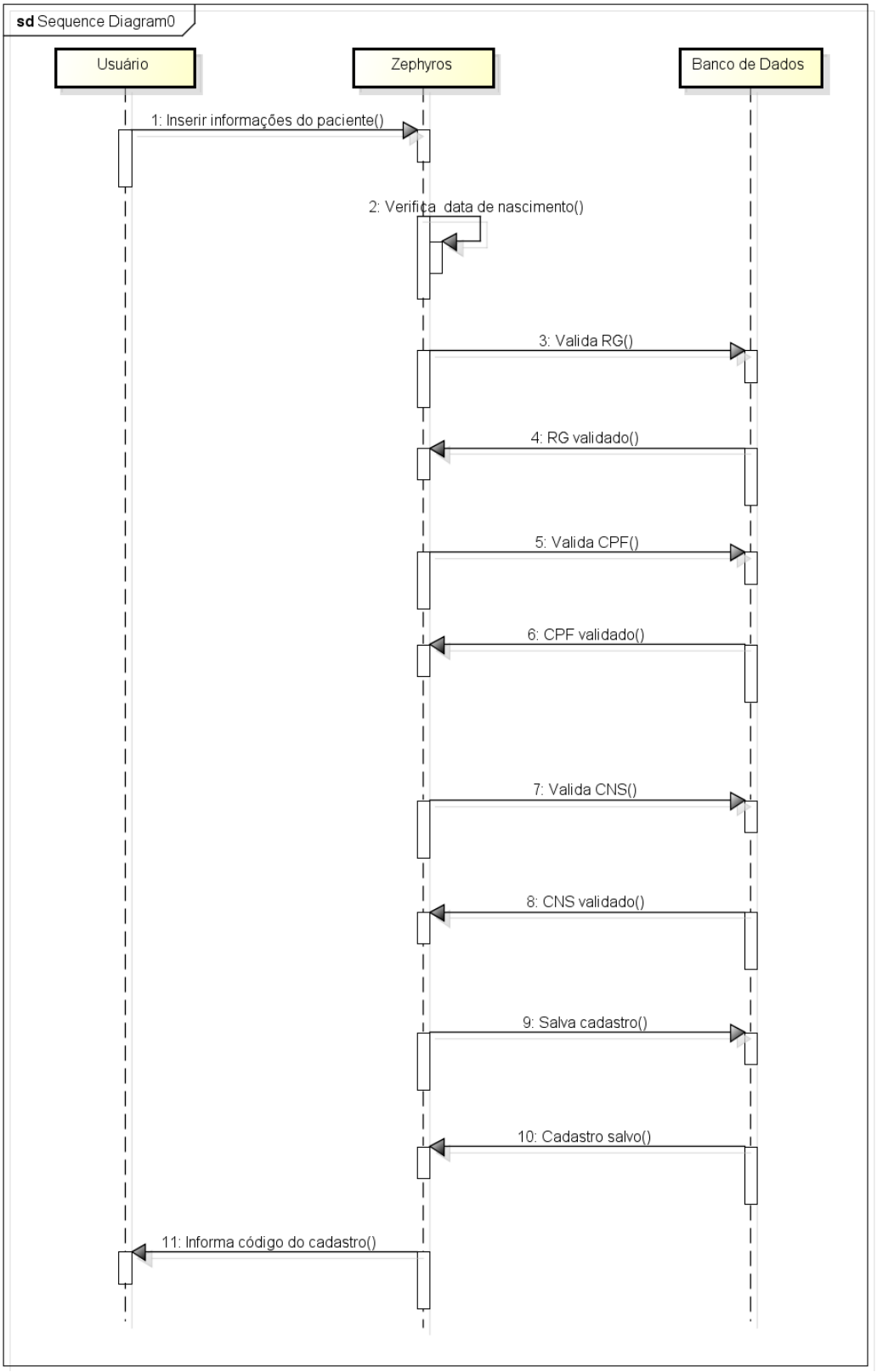
Apêndice F – Diagrama de Atividade – Agendar especialidade / exame (nova consulta)



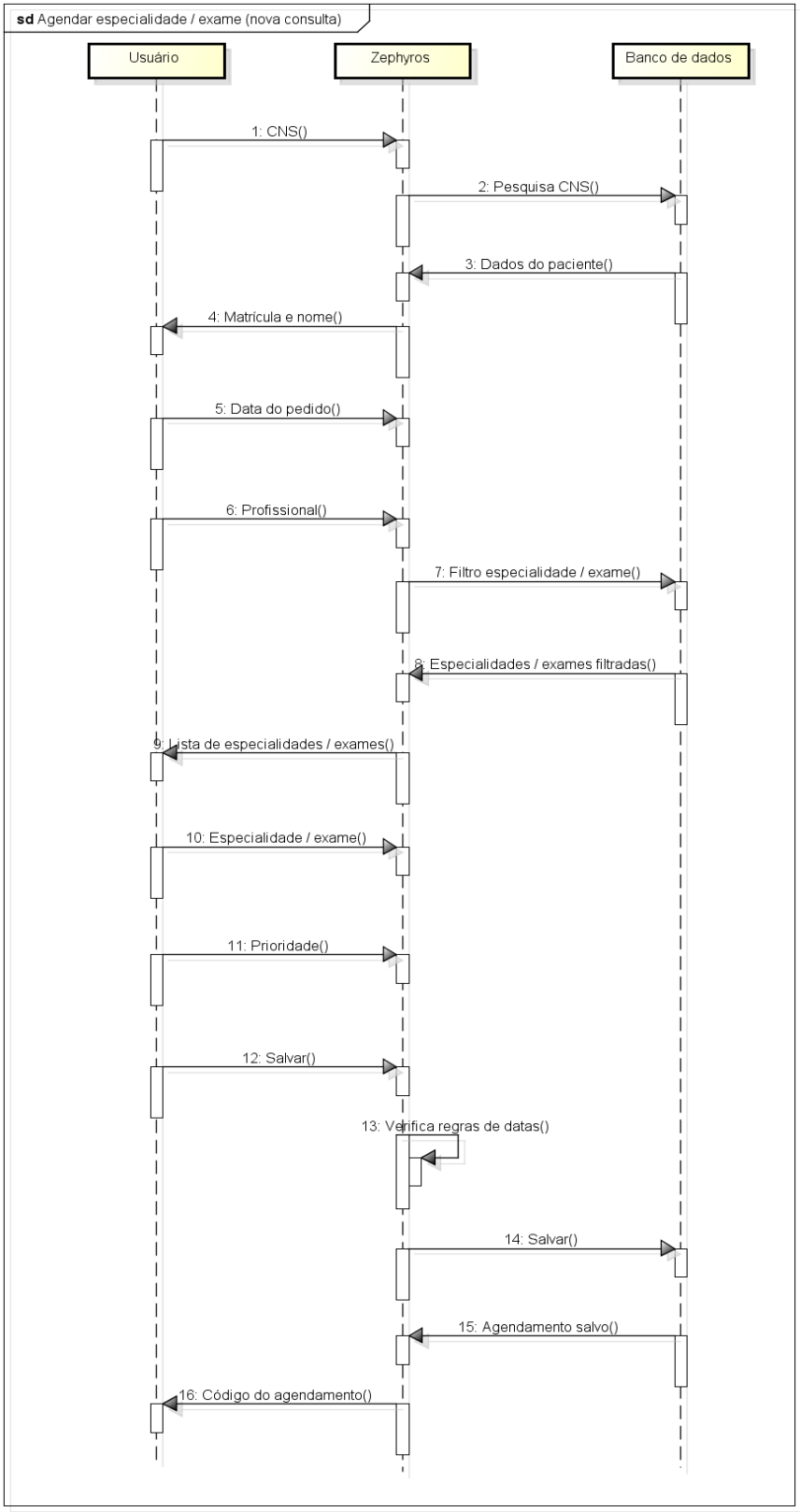
Apêndice G – Diagrama de Atividade – Agendar exame / especialidade (consulta existente)



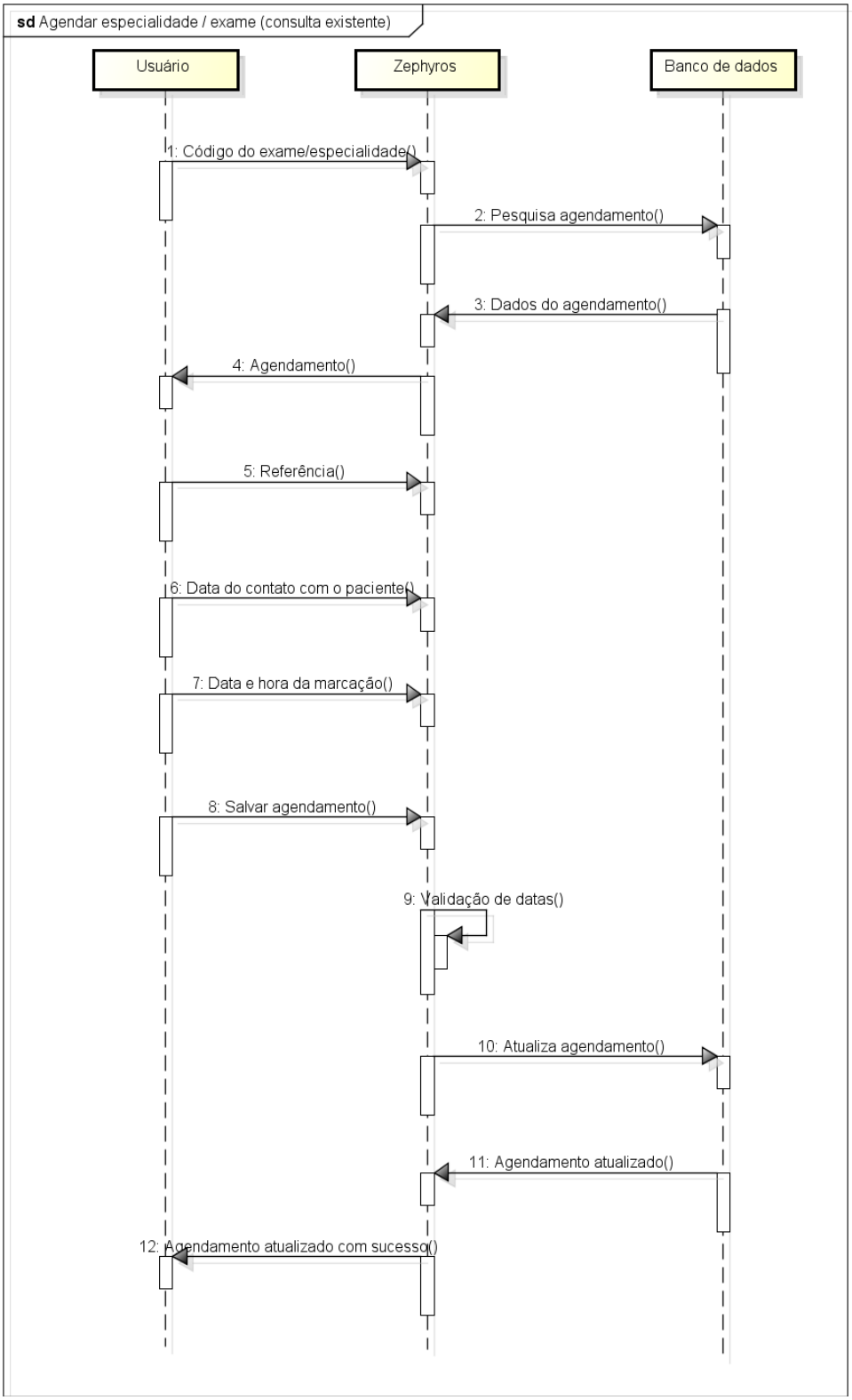
Apêndice H – Diagrama de Sequência – Cadastrar Paciente



Apêndice I – Diagrama de Sequência – Agendar especialidade/exame (nova consulta)



Apêndice J – Diagrama de Sequência – Agendar especialidade/exame (nova consulta)



Apêndice K – Diagrama de Classes – Sistema Zephyros