AGENDA DE CONSULTA HOSPITALAR

Filipe Alves Garcia
Ezequias Ferreira de Souza
Curso: Tecnologia em Análise e Desenvolvimento de Sistemas Período: 6º
Área de Pesquisa: Ciências Exatas e da Terra

Resumo:O presente trabalho tem como objetivo expor o desenvolvimento e a implantação de um sistema de informação para gerenciamento de consultas. Os procedimentos de controle de horários, registros de pacientes, quantidade de agendamentos por dia, controle de retorno de consultas entre outros, são realizados de forma dificultosa, geralmente artesanal. Com o intuito de solucionar tais questões, ou minimizá-las, foi desenvolvido um sistema para definir por inteiro oque seria feito, nele foi utilizado a linguagem *Java*, a ferramenta *Workbench*e o Banco de dados do pacote *WampSever*, depois que o sistema de teste foi concluído,foi realizado um novo desenvolvimento que se resultouno sistema final, onde o cliente que usara a ferramenta deverá ter seus*login* e senha para que seja capaz de registrar, pesquisar, alterar, editar e excluirinformações dos pacientes,e também manter informações dos médicos, agendar e confirmaroucancelar o agendamento, saber quais agendamentos já marcados de futuros de pacientes e exibi-los. Desenvolvido em linguagem *Java* e utiliza o banco de dados *PostgreSQL*para armazenar informações.

Palavras Chaves: Java, NetBeans,PostgreSQL.

1. INTRODUÇÃO

Atualmente organizações de saúde têm tido um grande aumento na quantidade de serviços, por conta de que a expectativa de vida brasileira, a diminuição da mortalidade infantil e outros fatores socioeconômicos positivos tem aumentado ao longo do tempo.

Sabendo que durante a atividade médica entre outras, são feitas uma grande quantidade de informação para cada paciente, e com o aumento de pacientes também cresce o aumento de informação a ser gerenciada, um exemplo: em grandes organizações médicas o aumento de informação não é um grande problema, pois em si já estão automatizadas. Mas ainda existem pequenas clínicas e consultórios em que a administração é realizada de forma artesanal, com um grande número de documentos em papéis e até em forma manuscrita que gera desperdício de tempo e dinheiro.

O método artesanal é trabalhoso e demanda atenção constante. Muitas vezes ineficiente, está sujeito à perda das informações, conflitos de dados, ilegibilidade das anotações e degrada-se com mais facilidade; mofo, por exemplo. Tais eventos, se ocorrerem, são capazes de prejudicar o desempenho e a qualidade dos serviços prestados pela organização (HOGARTH; SABBATINI, 1998 *apud* STOLF, 2007, p. 14).

No presente trabalho é proposto substituir o método artesanal por um *software* e desenvolver um sistema *desktop* na linguagem Java utilizando o *NetBeans IDE 8.2* em conjunto do *PostgreSQL*com o objetivo de melhor gerenciar as

consultas, de maneira a padronizar e facilitar o controle das atividades de um consultório visando um melhor controle e agilidade na prestação de serviços ao paciente.Em fim é possível substituir o método artesanal por um software simples de controle?

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo serão citados os principais conteúdos e conceitos necessários para a realização dos objetivos neste trabalho. Inicialmente será abordada a metodologia de desenvolvimento de *software*, em seguida o processo de *software*com: Metodologia Ágil, Especificação de *Software*, Engenharia e Análise de Requisitos, Projeto e Implementação de *Software*,*Scrum*, Linguagem *Java*,*Workbench, PostgreSQLeNetBeans IDE*.

2.1. METODOLOGIA DE DESENVOLVIMENTO DE *SOFTWARE*

Segundo Rezende (1997),metodologia para estar completa deve conter uma abordagem organizada para que possa atingir um objetivo, por meio de etapas preestabelecidas, ou seja um roteiro, um processo dinâmico e iterativo para desenvolvimento estruturado de projetos, sistemas ou *software*, visando à qualidade, produtividade e efetividade de projetos.

> Metodologia de *software* não é apenas uma técnica, porque pode utilizar qualquer outra técnica de desenvolvimento de projeto, sistema ou *software*. Alguns exemplos de técnicas são: Análise Estruturada, Análise de Pontos por Função, Análise Orientada a Objetos entre outras.
>
> A metodologia deve auxiliar o desenvolvimento de projetos, consistem em várias, sistemas ou *software*, de modo que os mesmos atendam de maneira adequada às necessidades do cliente ou usuário, com os recursos disponíveis e dentro de um prazo ideal definido em conjunto com os envolvidos. Não deve limitar a criatividade profissional, mas deve ser um instrumento que determine um planejamento metódico, que harmonize e coordene as áreas envolvidas. O que limita a criatividade não é a metodologia, mas os requisitos de qualidade, produtividade e efetividade de um projeto (REZENDE, 2005).

2.1.1. FASES DA METODOLOGIA DE DESENVOLVIMENTO DE SISTEMAS

As fases de uma metodologia também podem ser chamadas de ciclo de vida de sistema ou processos de *software*. Embora essas fases sejam apresentadas em sequência, o processo de desenvolvimento de Sistemas de Informação deve ser dinâmico e interativo. Essa interatividade permite que a equipe desenvolvedora retorne às fases anteriores ou desmembre o sistema em módulos ou protótipos, possibilitando a continuação da construção de uma parte do Sistema de Informação. Todavia, sempre devem ser respeitados os pontos de avaliação da qualidade e

aprovação, que requerem revisão da qualidade do projeto e validação formal dos envolvidos.

> As fases para o desenvolvimento de projetos, sistemas ou *softwares* são: estudo preliminar, ou anteprojeto, ou estudo inicial, ou primeira visão; análise do sistema atual, ou reconhecimento do ambiente; projeto lógico, ou especificação do projeto, ou *design*; projeto físico, ou execução, ou implementação do projeto, ou programação; projeto de implantação, ou projeto de disponibilização e uso (REZENDE, 2005).

2.1.1.1. DEFINIÇÕES DAS FASES DE DESENVOLVIMENTO DE SISTEMAS

Estudos preliminares: São a visão global do projeto, com a primeira definição dos requisitos funcionais, objetivos, abrangências, integrações, limitações, impactos e áreas envolvidas.

Análise do Sistema: É a visão do atual sistema, relatando requisitos funcionais atuais.

Projeto Lógico: É a confecção de propostas de soluções, definição dos requisitos funcionais, desenho da lógica do projeto. Definição de o que o projeto fará.

Projeto Físico: Execução, confecção de programas e seus testes, *layout* das entradas e saídas.

Projeto de implantação: Disponibilidade, execução do planejamento de implantação, treinamento e capacitação do cliente ou usuário.

2.1.2. PREMISSAS DE DESENVOLVIMENTO DE SISTEMAS

> As premissas da metodologia são a modularidade e sua própria existência. A modularidade não tolera o desenvolvimento de projetos, sistemas ou *software* sem metodologia. A segunda premissa retrata que sempre um sistema ou *software* deve ser desenvolvido com uma metodologia, mesmo que ainda não esteja fortemente sedimentada (REZENDE, 2005).

2.1.3. METODOLOGIA ÁGIL

Em 2001, *Kent Beck* e 16 outros notáveis desenvolvedores, produtores e consultores de *software*|***BEC01a***| (conhecidos como a “Aliança Ágil”) assinaram o “Manifesto para o Desenvolvimento Ágil de *Software*”. Um manifesto é normalmente associado com um movimento político emergente – um movimento que ataque a velha guarda e sugira mudanças revolucionárias. De algum modo, isso é exatamente o que o desenvolvimento ágil é. Embora as ideias subjacentes que guiam o desenvolvimento ágil tenham estado conosco há muitos anos, somente durante a década de 90 é que essas ideias foram cristalizadas em um “movimento”. Em essência, os métodos ágeis foram desenvolvidos em um esforço para vencer as fraquezas percebidas e reais da engenharia de *software* convencional. O desenvolvimento ágil pode fornecer importantes benefícios, mas ele não é aplicável a todos os projetos, produtos, pessoas e situações. Ele também não é contrário à

sólida prática de engenharia de *software* e pode ser aplicado como uma filosofia prevalecente a todo o trabalho de *software* (PRESSMAN, 2006).

> A engenharia de *software* ágil combina uma filosofia e um conjunto de diretrizes de desenvolvimento. A filosofia encoraja a satisfação do cliente e a entrega incremental do *software* logo de início; equipes de projeto pequenas, altamente motivadas, métodos informais; produtos de trabalho de engenharia de *software* mínimos e simplicidade global do desenvolvimento. As diretrizes de desenvolvimento enfatizam a entrega em contraposição à análise e ao projeto (apesar dessas atividades não serem desencorajadas) e a comunicação ativa e contínua entre desenvolvedores e clientes (PRESSMAN, 2006).

2.2. PROCESSOS DE SOFTWARE

Em engenharia de *software*, processos podem ser separados para atividades como Especificação de *software*, Projeto e Implementação de *software*. E em cada atividade existente, podem-se definir subprocessos, como determinação de requisitos funcionais, análise de sistema, diagramação, programação, testes e implantação.

Sommerville (2003) define um processo de *software* como um conjunto de atividades e resultados associados que levam à produção de um produto de *software*. Os processos de *software* são complexos e, como todos os processos intelectuais, dependem de julgamento humano. Não há um processo ideal, e diferentes organizações desenvolveram abordagem inteiramente diferente. Até dentro da mesma empresa pode haver muitos processos diferentes utilizados para o desenvolvimento de *software*.

2.2.1.ESPECIFICAÇÃO DE SOFTWARE

Segundo Sommerville (2003) a especificação de *software* destina-se a estabelecer quais funções são requeridas pelo sistema e as restrições sobre operação e o desenvolvimento do sistema. Essa etapa do processo de *software* geralmente é chamada de engenharia de requisitos que se encontrados erros produzem problemas posteriores no projeto e na implementação do sistema.

2.2.1.1. ENGENHARIA E ANÁLISE DE REQUISITOS

> Na etapa de especificação no desenvolvimento de um *software* são obtidos requisitos ou informações que são necessárias para modelagem do sistema. Para poder analisar e entender o problema a ser solucionado e estabelecer com certeza o que o sistema deve fazer.
>
> Com a análise de requisitos serão determinados e analisados requisitos com a finalidade de converter a explicação de alto nível desses requisitos em uma lista precisa que possa ser usada como informação para o restante da fase de análise.

> A organização dos requisitos em grupos com diferentes níveis de descrições permite que não apareçam problemas durante o processo de engenharia de requisitos.(REZENDE, 2005).

Segundo Resende(2005), a análise de requisitos é fundamental para desenvolver um sistema ou *software* que atenda e satisfaça totalmente a todos os desenvolvedores, clientes e usuários, jáquando a análise de requisitos é bem definida e relatada evita muitos problemas futuros e um deles é a alta manutenção de sistemas e *software*.

2.2.1.1.1. REQUISITOS FUNCIONAIS

> Os requisitos de um *software*, também chamados de requerimento de *software* ou de requisitos funcionais de um sistema, devem ser elaborados no início de um projeto de sistema ou *software*. (REZENDE, 2005).
>
> Os requisitos funcionais são declarações de funções que o sistema deve fornecer como o sistema deve reagir a entradas específicas e como deve se comportar em determinadas situações. Em alguns casos os requisitos funcionais podem também explicitamente declarar o que o sistema não deve fazer. (SOMMERVILLE, 2003).

2.2.2. PROJETO E IMPLEMENTAÇÃO DE *SOFTWARE*

Sommerville (2003) descreve que a etapa de implementação de desenvolvimento de *software* é o processo para a conversão de uma especificação de sistema em um sistema executável que envolve todo o processo do projeto e programação de *software* podendo ser também um aperfeiçoamento da especificação de *software* quando aliado a uma metodologia de desenvolvimento.

> Um projeto de *software* é uma descrição de estrutura de *software* a ser implementada, dos dados que são parte do sistema, das interfaces entre componentes do sistema e, algumas vezes, dos algoritmos utilizados. O processo de projeto de *software* envolve acrescentar forma e detalhes, à medida que o projeto é desenvolvido com retornos constantes a fim de corrigir projetos anteriores. (SOMMERVILLE, 2003).

2.2.3. UML (Linguagem de Modelagem Unificada)

UML é uma linguagem para visualização, especificação, construção e documentação de artefatos de um *software* orientado a objeto.

Sua vantagem é que ela serve para as quatro atividades: análise, *design*, implementação e teste.

Ela é composta por itens, relacionamentos e diagramas, sendo eles:

Itens: Estruturais (classe, interface, casos de uso e componentes), comportamentais (interações, máquinas de estado), grupos de elementos (pacotes, *frameworks* e subsistemas) e Anotações (notas).

Relacionamentos: Dependência; Associação; Generalização e Realização, os relacionamentos são entre itens como classes e casos de uso.

Diagramas: são divididos em nove diagramas que são: de classes; de objetos; de casos de uso; de sequência; de colaborações; de gráficos de estados; de atividades; de componentes e de implantação.

2.2.3.1. DIAGRAMA CASOS DE USO

Segundo Bezerra(2007) O diagrama de casos de uso tem como objetivo ajudar na comunicação entre o ponto de vista do analista com o ponto de vista do cliente, descrevendo um cenário aonde mostra todas funcionalidades do sistema e futuros riscos que podem ou não acontecer no ponto de vista do usuário.

> O diagrama de caso de uso é representado por atores, casos de usos e relacionamentos. Os relacionamentos podem ser associações entre atores e casos de uso e generalizações entre os atores e generalizações são representados por *extends* e *includes*. (BEZERRA, 2007).

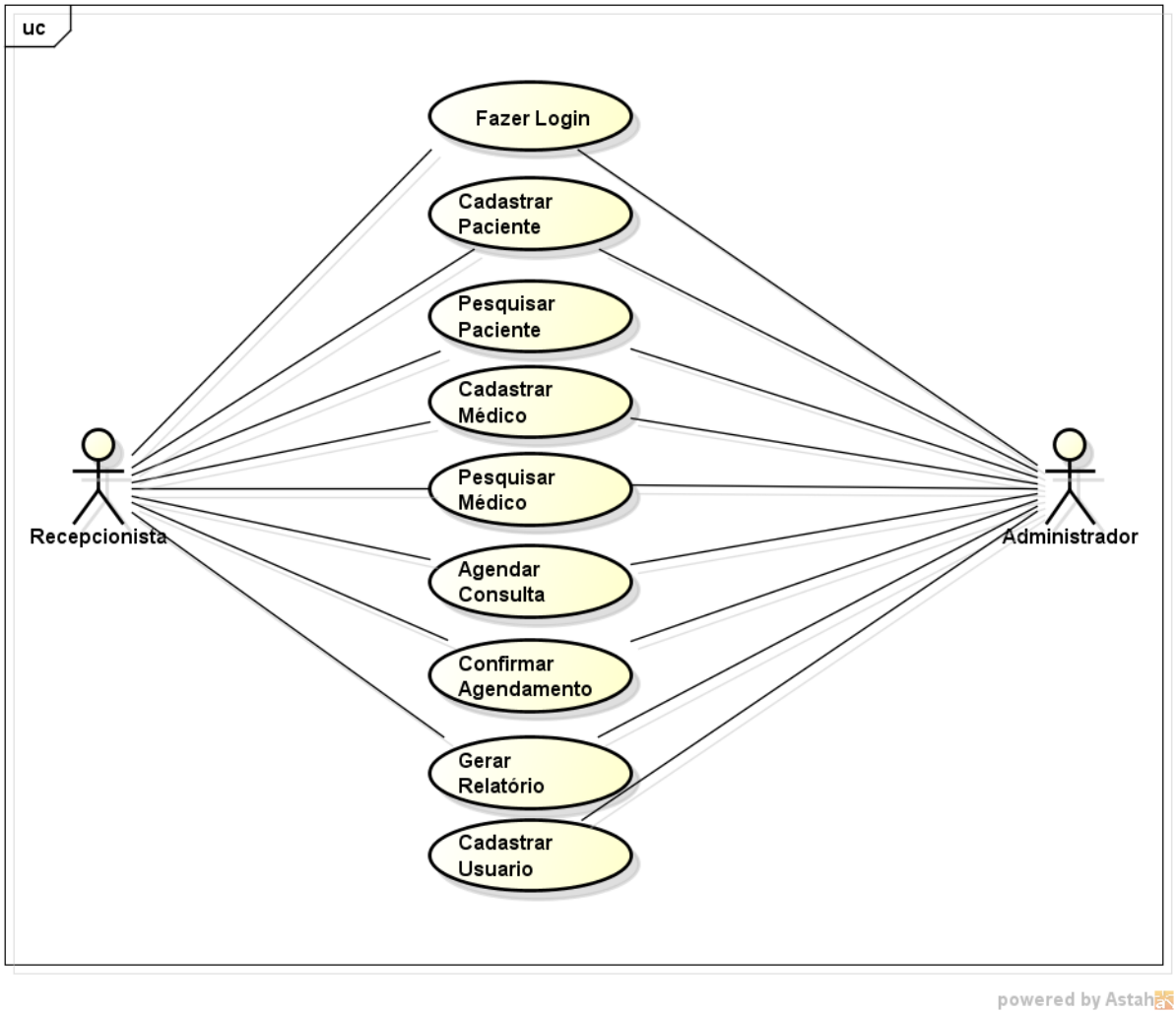

Figura 1:Modelo de Diagrama Casos de Uso

Cenário Principal:
1 Recepcionista faz *login* que dá acesso a tela principal do Sistema.
2 Recepcionista passa informações do paciente para o Sistema.
3 Sistema salva informações.
4 Recepcionista solicita pesquisa de usuário ao Sistema.

5 Sistema exibe informações.
6 Recepcionista passa dados do médico.
7 Sistema salva dados.
8Recepcionista solicita pesquisa de médico.
9 Sistema exibe informações.
10 Recepcionista envia dados para Marcar Horário de Consulta ao Sistema.
11 Sistema salva dados.
12Recepcionista atende paciente e confirma agendamento.
13 Sistema finaliza agendamento.
14 Recepcionista gera relatório de agendamento para dia 21/11/2017.
15 Sistema por meio de data digitada exibe informações.

Cenário Alternativo:
1 Caso Recepcionista tenta fazer *login* com campos em branco.
2 Sistema não permite e avisa erro.

2.2.4. IMPLEMENTAÇÃO

Implementação é a fase do ciclo de vida de um *software* (programa computacional, documentação e dados), no contexto de um sistema de informação, que corresponde à elaboração e preparação dos módulos necessários à sua execução.

O diagrama de caso de uso é representado por atores, casos de uso e relacionamentos. Os relacionamentos podem ser associações entre atores e casos de uso; generalizações entre os atores e generalizações, *extends* e *includes*.

2.2.4.1. PROGRAMAÇÃO ORIENTADA A OBJETOS

Orientação a objetos é uma maneira de programar que ajuda na organização e resolve muitos problemas enfrentados pela programação procedural. A programação orientada a objeto, segundo a maioria dos especialistas, fundamenta-se em quatro princípios: abstração, encapsulamento, herança e polimorfismo. (MECENAS, 2008).

O paradigma Orientado a Objetos (POO) é o mais avançado no quesito de reusabilidade de código e não resolve tudo aquilo que se propõe a fazer, ferindo, inclusive, seus próprios preceitos e ocasionando no sistema dificuldade de compreensão, manutenção, forte acoplamento e até mesmo redundância de código (WINCK & JUNIOR, 2006).

A programação orientada a objetos se ocupa de realizar um projeto de *software* utilizando uma linguagem de programação orientada a objetos na qual aceita implementação direta de objetos e fornece recursos para definir as classes de objetos. Os sistemas orientados a objetos devem ser de fácil manutenção, uma vez que os objetos são independentes. (SOMMERVILLE, 2003).

3. LINGUAGEM JAVA

Java é uma linguagem criada a partir da linguagem de programação C. C, influenciada e testada por programadores profissionais, a linguagem C trabalha no mais baixo nível, já outras linguagens trabalham no nível mais alto.

> *Java* é uma das filhas de C. Criada por James Gosling, da *Sun Microsystems*, com o nome de "*Oak*", a linguagem deveria incorporar os benefícios da programação orientada a objetos sem, no entender do criador, a desnecessária complexidade de C (MECENAS 2008).

Segundo Mecenas (2008) a linguagem *Java* é dividida nos seguintes componentes:

Applets: *Applet* é um componente de *software* que executa uma função limitada em outro ambiente de programa, como um navegador da *Web*. Os *applets* Java fornecem recursos interativos em um navegador da *Web* por meio do *Java Virtual Machine* (JVM).

JVM *(Java Virtual Machine):*O mecanismo responsável pela interpretação do código intermediário *Java*. A Máquina Virtual *Java*, embora possa parecer estranho, não conhece a linguagem *Java*. Ela somente reconhece o formato de arquivo. *Class*que contém os *bytecodes*, os quais serão traduzidos para o código binário da plataforma hospedeira.

JSDK *(Java Software Development Kit)* ou JDK *(Java Development Kit):*O conjunto de ferramentas de desenvolvimento oferecido pela *Sun* e outros fabricantes, que inclui, como estrutura mínima, um compilador e um interpretador.

JRE *(Java RuntimeEnvironment):*O ambiente de execução *Java* existente nas máquinas dos usuários finais, que apenas executam aplicações *Java*. O ambiente inclui, portanto, a Máquina Virtual *Java* e as classes que formam a plataforma *Java*.

JDBC *(Java DatabaseConnectivity):*Arquitetura baseada em interface e especificações que permite acesso multiplataforma a bancos de dados. O acesso aos dados baseia-se em *drivers* JDBC, que são implementados por conjuntos de classes que dominam a arquitetura e os comandos dos bancos de dados.

Servlets:Extensões do padrão CGI, escritas em puro *Java*, usadas para a geração de conteúdo dinâmico para a *Web*.

JSP *(JavaServerPages):*Conjunto de API's que permitem a criação de páginas *web* dinâmicas, com acesso a dados e interação com o usuário. Atualmente, há uma forte tendência a se utilizar JSP para a parte visual das aplicações e *Servlets* para a lógica de validação e controle. Há muitos aplicativos e *sites* que funcionam somente se o *Java* estiver instalado, muitos outros aplicativos e *sites* são desenvolvidos e disponibilizados com o suporte dessa tecnologia. O *Java* é rápido, seguro e confiável. A tecnologia *Java* está em todo lugar, pode ser encontrada em *laptops*, *datacenters*, consoles de jogo, supercomputadores científicos, telefones celulares e até na Internet.

EJB*(Enterprise JavaBeans):* Define padrão para componentes de negócio distribuídos. A sigla também é utilizada para referir-se aos componentes criados segundo o padrão. Os componentes fornecem serviços básicos como: gerenciamento de transações e persistência.

RMI *(RemoteeMethodInvocation)*: Solução para a distribuição de aplicações *Java,* permitindo a comunicação entre objetos localizados em máquinas virtuais diferentes.

J2SE *(Java 2 Standard Edition):*A denominação da primeira edição de *Java*,contendo as principais classes da plataforma *Java*.

J2ME *(Java 2Micro Edition):* *A* extensão da plataforma J2SE para o desenvolvimento de aplicações *Java* em pequenos dispositivos como telefones celulares, sendo este apenas um dos exemplos de suas múltiplas aplicações.

J2EE *(Java 2 Enterprise Edition):* A extensão da plataforma J2SE para o desenvolvimento de aplicações distribuídas.Engloba um conjunto de tecnologias que, em conjunto, fornecem API's e um ambiente para desenvolvimento e execução de aplicações corporativas.

4. MYSQL WORKBENCH

SegundoBenthin(2010), a versão comunitária do *MySQL Workbench* inclui uma grande coleção de recursos. Administradores e desenvolvedores de banco de dados podem usá-lo para planejar tabelas, *views*, índices, *stored procedures* e *triggers*; interpretar esquemas de bancos de dados existentes para visualizá-los ("engenharia reversa"); sincronizar esquemas com bancos de dados existentes ("gerenciamento de mudanças") e exportar e imprimir modelos de diagramas.

5. NETBEANS IDE

> *NetBeans IDE (IntegratedDevelopmentEnvironment)* é uma ferramenta de desenvolvimento modular para uma ampla gama de tecnologias de desenvolvimento de aplicações. O IDE de base inclui um editor avançado, bem como ferramentas para controle de versão e colaboração de desenvolvedores. (Netbeans.org).

5.1. SUPORTE PARA AS TECNOLOGIAS MAIS RECENTES DE JAVA

NetBeans IDE oferece suporte abrangente para as mais recentes tecnologias *Java* e as últimas melhorias de *Java* antes de outros IDE's. Este IDE fornece suporte para JDK, o JDK é um ambiente de desenvolvimento para construir aplicações, e componentes usando a linguagem de programação *Java* uma máquina virtual, *Java EE Enterprise Edition* é o padrão em *software* empresarial voltada para a comunidade, e *Java* FX é o próximo passo na evolução do *Java* como uma plataforma *richclient*. Ele é projetado para fornecer uma plataforma leve acelerada por *hardwareJava* UI para aplicações de negócios da empresa. (Oracle.com).

O IDE fornece modelos de projeto e exemplos de projetos que ajudam a criar aplicativos *Java*, aplicações Java EE, aplicações HTML5, aplicação PHP e aplicativos C/C++ além de outras linguagens disponíveis no IDE.

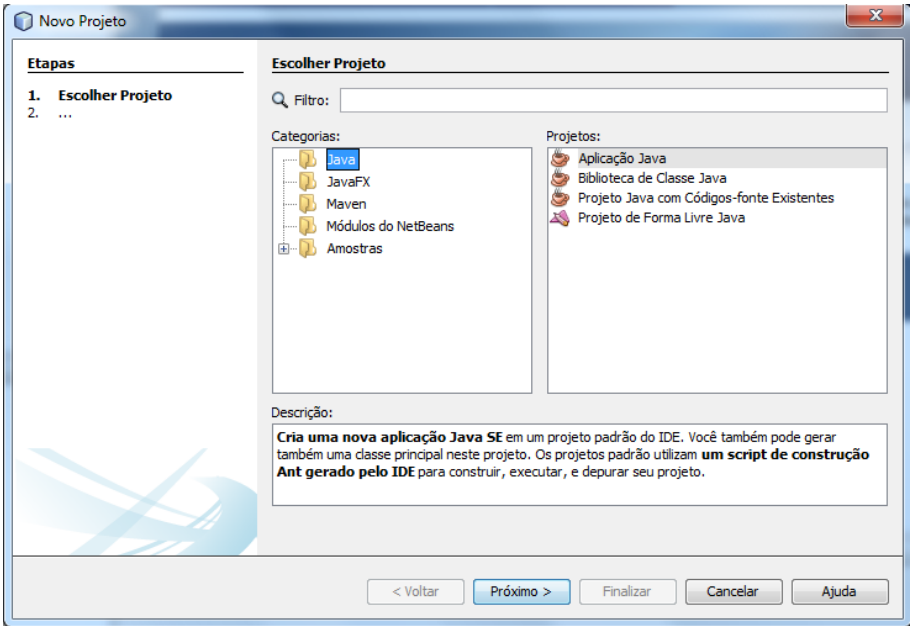

Figura 3: Escolha do projeto.

O IDE suporta várias linguagens divididas por categorias, entre elas estão: *Java*, Java FX, *Maven* e outras. Após selecionar a categoria temos ainda opções de projetos disponíveis para cada uma dessas categorias, como mostra a figura 3.

> *Segundo o site NetBeans.org: NetBeans IDE* oferece aplicações de esqueleto em forma de modelos de projeto para todas as tecnologias que ele suporta. Além disso, proporciona um conjunto de aplicações de exemplo.
>
> Um IDE é muito mais que um editor de texto. O Editor do *NetBeans* recua linhas, associa palavras e colchetes e realça códigos-fonte sintática e semanticamente. Ele também fornece modelos de código, dicas de codificação e ferramentas de refatoração.

A figura 4 mostra um exemplo da edição de um código dentro do *NetBeansIDE*.

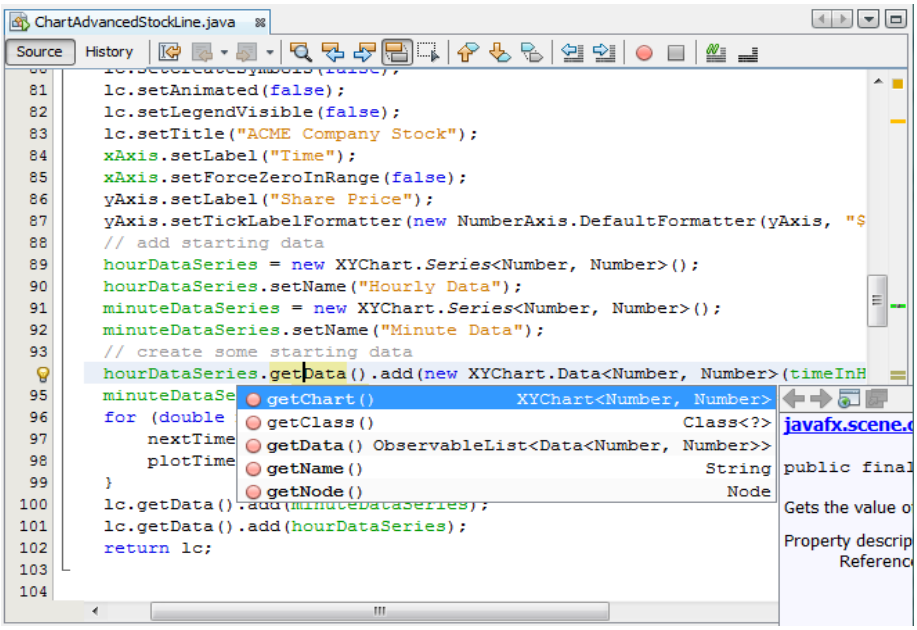

Figura 4: Edição de Código.

5.2. BANCOS DE DADOS E SERVIÇOS

A janela Serviços dá acesso a muitos recursos auxiliares, tais como bancosde dados, servidores, serviços *web* e rastreadores de problemas.
Você pode iniciar e parar os bancos de dados e servidores diretamente no IDE. Ao trabalhar com bancos de dados, você pode adicionar remover e modificar os seus dados no IDE. Quando você implantou um aplicativo para um servidor, você pode gerenciar seus recursos mobilizados, porque eles são exibidos no nó Servidores.

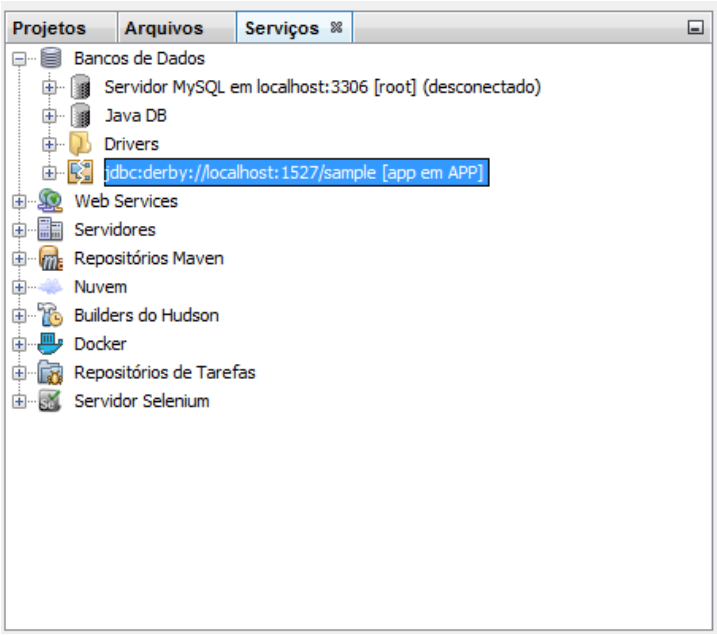

Figura 5: Serviços Básicos.

5.3. DESENVOLVIMENTO DE INTERFACE

Com relação a interface gráfica pode ser construída dentro do próprio IDE, com isso facilita a união da interface e do código fonte do projeto em construção, vantagem essa muito utilizada pelos usuários dessa ferramenta facilitando assim a visualização do usuário final de um *software*.

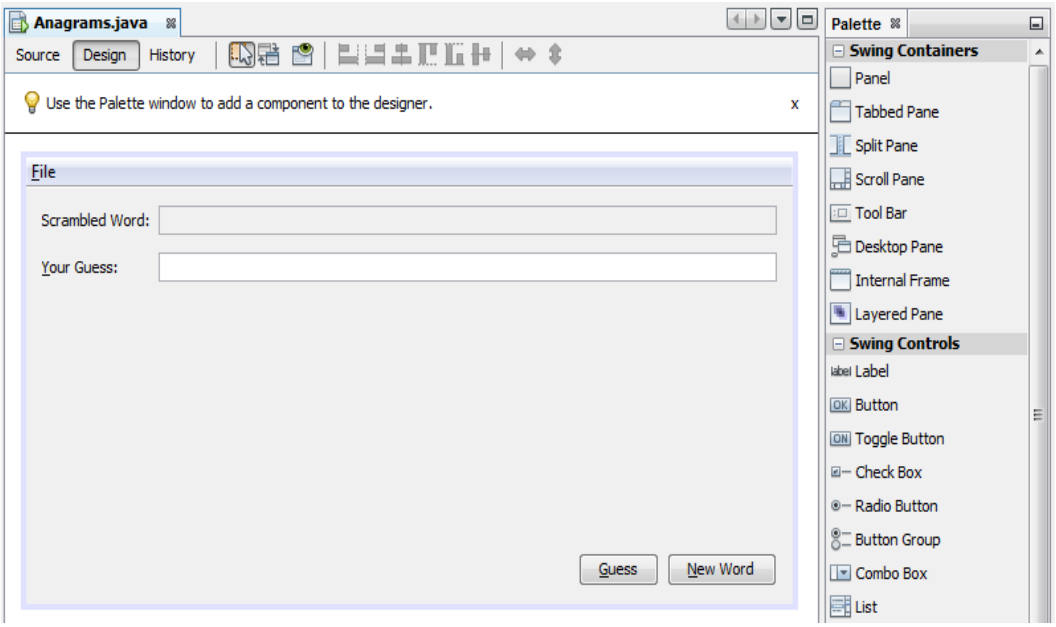

Figura 6: Exemplo de interface Gráfica.

Criação de uma interface gráfica de um *software* contendo dois botões, um título e duas caixas de texto mostrados na figura 6.

5.4.DEPURADOR

O *NetBeansDebugger* permite a você inserir pontos de interrupção no seu código-fonte, adicionar *watches* de campo, avançar pelo código, executar métodos, obter telas e monitorar a execução durante sua ocorrência. Você também pode conectar o depurador a um processo já em execução.(Netbeans.org).

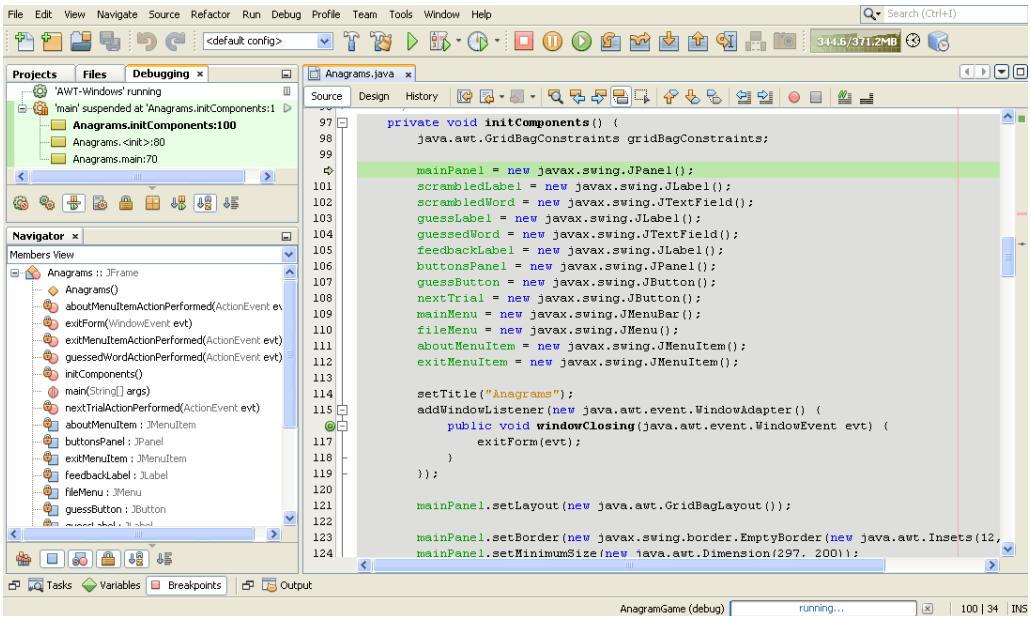

Figura 7: Exemplo de *Debugger*.

O IDE inclui um depurador visual que permite obter telas da GUI e explorar visualmente a GUI das aplicações em

JavaFX e *Swing*. Ele permite exibir propriedades do componente, a hierarquia dos componentes no contêiner e localizar o código-fonte dos componentes. Você pode usar o depurador visual para adicionar com facilidade *listeners* para ações da GUI sem precisar pesquisar em todo o código-fonte.(Netbeans.org).

5.4.2. CRIADOR DE PERFIL

Segundoo siteNetbeans.org,O *NetBeans Profiler* proporciona assistência especializada para otimizar a velocidade e o uso de memória de sua aplicação e facilita a construção de aplicações *Java SE*, *JavaFX* e *Java EE* confiáveis e dimensionáveis.

Selecione entre várias das tarefas comuns de criação de perfil, tais como criação de perfil padrão da CPU ou da memória ou monitoramento simples. As opções foram predefinidas para valores *default* por especialistas; você pode personalizar as definições para uma aplicação específica.(Netbeans.org).

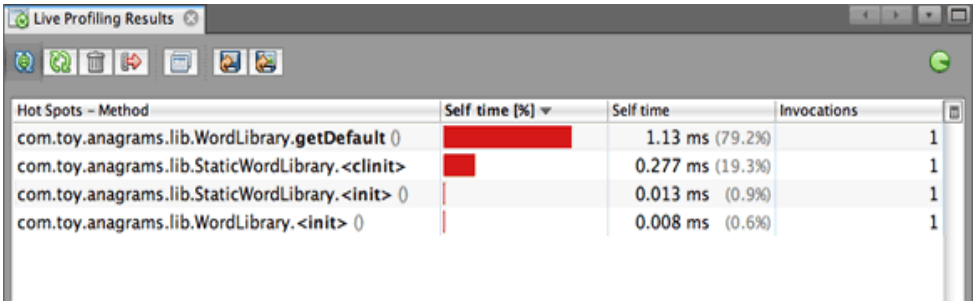

Figura 8:Resultados em Tempo Real.

5.5.PROJETO EXIBIÇÃO

A janela Projetos é o principal ponto de entrada para as fontes do projeto. Ele mostra uma visão lógica do conteúdo do projeto.

Além da janela Projetos, o IDE fornece a janela Arquivos, para que você possa ver todos os arquivos que pertencem a um projeto, e a janela de Favoritos, onde você pode adicionar pastas e arquivos de modo que eles podem ser pesquisados dentro do IDE.

Figura ilustrando uma janela detalhando de um projeto, juntamente com uma janela com os arquivos compostos nesse projeto.(Netbeans.org).

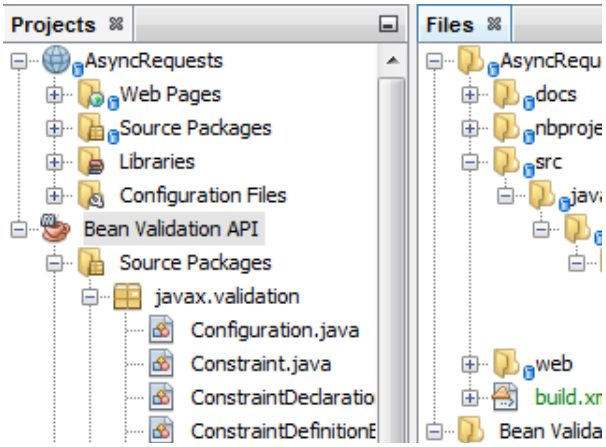

Figura 9: Gerenciamento de Projeto

A janela Navegador oferece uma *view* compacta do arquivo selecionado atualmente e simplifica a navegação entre diferentes partes do arquivo.

Por exemplo, para arquivos Java, a janela Navegador mostra uma lista de construtores, métodos e campos, enquanto para arquivos HTML e CSS ele mostra a estrutura lógica do documento selecionado(Netbeans.org).Exemplo na figura 10:

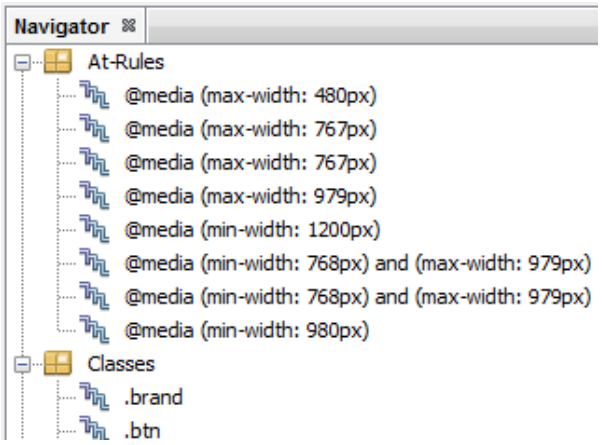

Figura 10: Gerenciamento do Navegador.

6.PROGRAMA AGENDA DE CONSULTA

A ideia de desenvolver o *software* agenda de consulta surgiu depois de ter visto em vários estabelecimentos médicos aonde seu método de agenda era artesanal (feita em cadernos, etc.) e geralmente seus pacientes eram chamados por ordem de chegada, tendo isso foi proposto um desenvolvimento de um projeto aonde cobriria toda a questão.

6.1. TELA DE LOGIN

Figura 11: Tela de *Login*.

A agenda de consulta terá uma tela de *login*aonde poderão entrar dois tipos de usuários, oadministrador e orecepcionista, onde o administrador terá acesso a todo o programa e o recepcionista terá acesso apenas as telas relacionadas ao paciente.

6.2. TELA PRINCIPAL

Figura 12: Tela Principal.

A tela principal é a tela de onde serão chamados todos outros formulários que nela poderão ser chamados pelo *jMenuBar que é como um Menu* ou *jButton*.

```
private void jButtonCadMedicoActionPerformed(java.awt.event.ActionEvent evt) {
try {
        conecta.sqlExecuta("select *from usuarios where nome_usuario='"+jLabelUsuario.getText()+"'");
        conecta.rs.first();// seta como primeiro resultado
        if(conecta.rs.getString("tipo_usuario").equals("Administrador")){
            if(telamed==null){
            telamed = new JMedico();
            telamed.setVisible(true);
            telamed.setResizable(false);
           } else{
                telamed.setVisible(true);
                telamed.setResizable(false);
            }
        }else{JOptionPane.showMessageDialog(null, "Você não tem permissão para entrar aqui!");}
    } catch (SQLException ex) {

    }
}
```

Figura 13: Chamada do formulário médico.

A chamada do formulário médico por meio de um *jButton* como mostrado na figura 13, verifica na tabela usuários por meio do nome do usuário que fez *login*, se o tipo do usuário for igual a Administrador ele chamara o formulário medico, mas caso não seja ele resultará na mensagem: “Você não tem permissão de entrar aqui! ”.

6.3. TELAS DE CADASTROS

6.3.1. TELA DE USUÁRIO

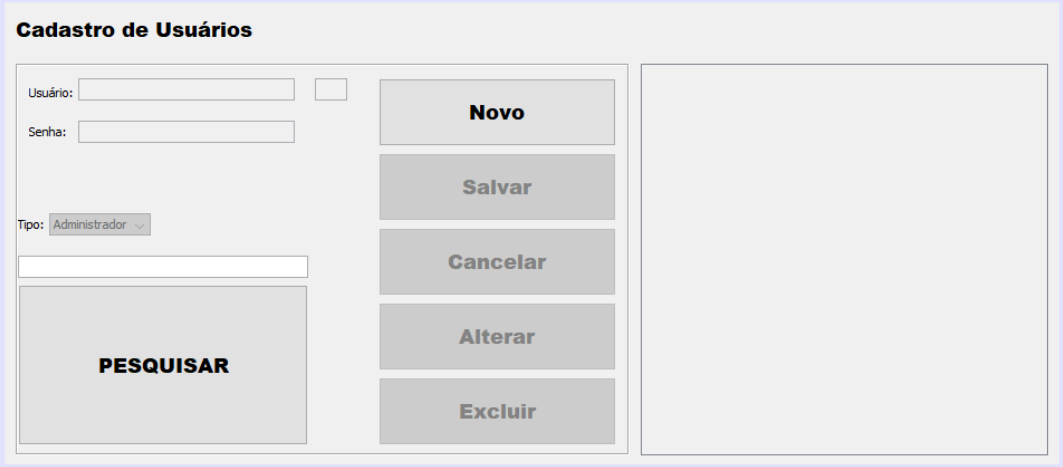

Figura 14: Tela de cadastro de usuários.

A tela de usuários como mostrada na figura 14 terá o campo de usuário e senha que serão cadastrados, contém uma *jTable* onde mostrara usuários já cadastrados, e também poderá ser feita a pesquisa dos mesmos para futuramente alterar ou excluir usuários.

6.3.2. TELA DE PACIENTE

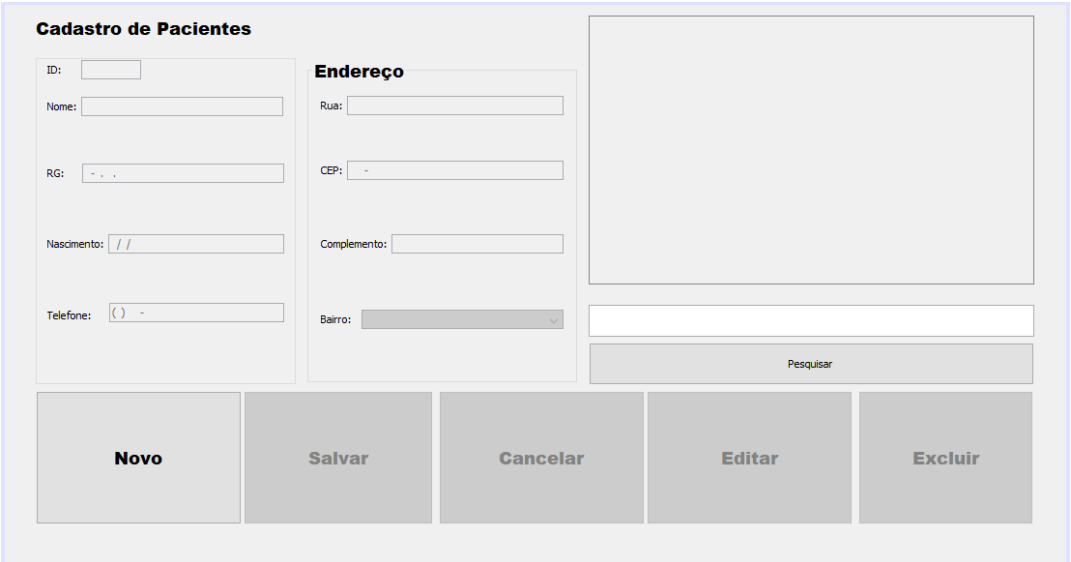

Figura 15: Tela de cadastro de pacientes.

Como exibido na figura 15 a tela de paciente conterá os campos de Nome, RG, Nascimento, Telefone, Rua, CEP, Complemento e Bairro que poderão ser

cadastrados alterados e excluídos, todos serão mostrados na *jTable* e também poderão ser pesquisados por meio do *jButtonPesquisar*.

6.3.3. TELA DE MÉDICOS

Como mostrados a seguir na figura 16 a tela de médicos irá cadastrar o Nome, CRM e a especialização do médico, também conterá um *jTable* onde exibirá os médicos já cadastrados que poderão ser pesquisados por meio do campo e o *jButtonPesquisar*.

Figura 16: Tela de cadastro de médicos.

6.3.4. TELA DE AGENDAMENTO

Figura 17: Tela agendamento de pacientes.

Na tela de agendamento como mostrado na figura 17 acima, deverá ser pesquisado por meio do *jButtonPesquisar* o nome do paciente, se encontrado o administrador deverá preencher o campo de data, selecionar por meio da *jComboBoxMedico* o médico que já está cadastrado, selecionar o turno de atendimento por meio da *JComboBoxTurno*, o motivo por ser atendido e por fim concluir o agendamento por meio do *jButtonConcluir* ou cancelar o agendamento por meio do *jButtonCancelar*.

6.3.5. TELA DE ATENDIMENTO

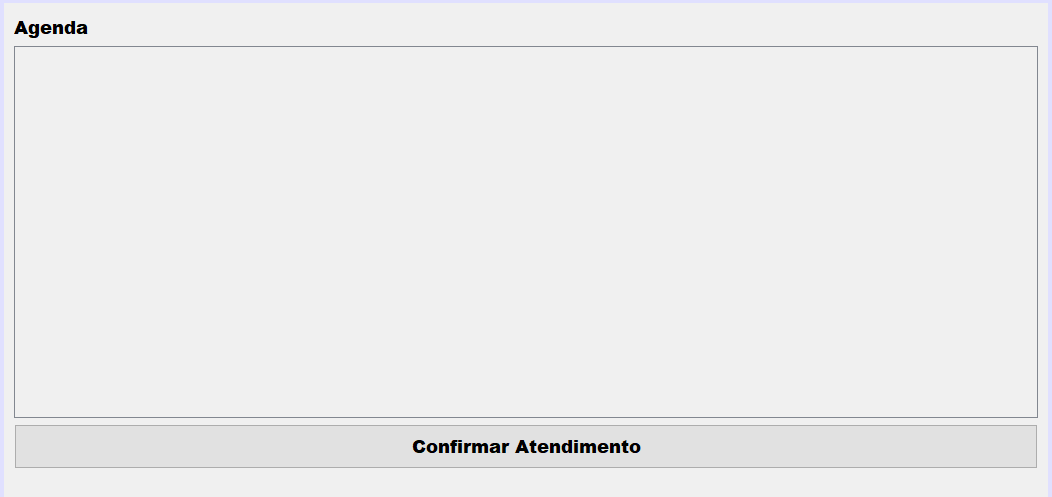

Figura 18: Tela de Atendimento.

Na tela de atendimento ou fila de espera como mostrado na figura 18 acima,será mostrado todos os agendamentos na *jTableAtendimento*que estão marcados para data atual em que o *software* foi executado, assim o administrador ou recepcionista poderá selecionar o paciente que vai estar disponível e confirmar assim que o paciente terminar de ser atendido.

```
1 select *from agendamento inner join pacientes on agenda_codpac=id_paciente inner join medicos on agenda_codmedico=id_medico where agenda_data='10/07/1996' and agenda_status='Aberto' order by agenda_turno;
```

Figura 19: Instrução SQL Preencher jTableAtendimento.

Na instrução SQL acima terá como resultado todos os dados da tabela agendamento pedido, onde a coluna agenda_data seja igual a data recebida ali no momento e a colunaagenda_status tem que estar com o *status* definido como aberto.

7.POSTGRESQL

Segundo*PostgreSQL*, a ferramenta *postgre* é um sistema de banco de dados de fonte aberto, possuindo mais de 15 anos de desenvolvimento ativo e uma arquitetura que lhe garante confiabilidade, integridade de dados e correção, ele é executado em todos os principais sistemas é compatível com o ACID e inclui a maioria dos tipos de dados SQL 2008, *incluindo integer, numeric, boolean, char, varchar, date, intervalo* e *timestamp*.

8. CONCLUSÃO

Atualmente é necessário cada vez mais de suporte para realizar um processo em pouco tempo. O *software* é uma ferramenta que pode auxiliar no controle e gestão de um processo. Com isso foi desenvolvido neste trabalho um *software* de Agenda De Consulta. O *Software* de Agenda de Consulta é uma ferramenta para auxiliar no controle de registro e atendimento de clientes em um consultório médico substituindo o método artesanal de um modo simples. Essa ferramenta *desktop* foi

desenvolvida na linguagem *Java* através do ambiente de desenvolvimento *NetBeans IDEversão 8.2*e a ferramenta de banco de dados *PostgreSQL*. Conforme os requisitos levantados o sistema foi desenvolvido com suas principais funcionalidades, cadastro de pacientes, de médicos, a agenda e consulta dos pacientes, alcançando os objetivos do trabalho proposto.

9. REFERÊNCIAS BIBLIOGRÁFICAS

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas UML**, Rio de Janeiro, Elisevier, 2007 – 2ª Reimpressão.

BENTHIN, Falko.***MySQL Workbench:Planejamento de bancos de dados com o MySQL Workbench***.Edição 1. Linux Magazine. 2010. Junho 2010. Disponível na Internet: <http://lnm.com.br/lm/article/3530>.

MECENAS, Ivan. **Java 6 Fundamentos, Swing, BlueJ e JDBC**. 3ª Edição, Rio de Janeiro, Alta Book, 2008.

Netbeans.org. Disponível em: < https://netbeans.org/features/index_pt_BR.html >. Acesso em: 18 maio 2017.

Oracle.com. Disponível em:
< http://www.oracle.com/technetwork/java/javaee/overview/index.html>. Acesso em: 15 maio. 2017.

PRESSMAN, Roger S. **Engenharia de Software**. 6ª ed. Editora McGraw-Hill, São Paulo, 2006.

PostgreSQLAbout. Disponível em:< https://www.postgresql.org/about/> Acesso em 28/11/2017.

REZENDE, D. A. **Engenharia de Software Empresarial**. Rio de Janeiro: Brasport, 1997.

REZENDE, Denis Alcides. **Engenharia de Software e Sistemas de Informação**. 3ª ed. rev. e ampl., Rio de Janeiro: Brasport, 2005.

SOMMERVILLE, Ian. **Engenharia de Software**. 6ª edição, São Paulo, Addison Wesley, 2003.

SCHWABER, K. **Agile Project Management With Scrum**, Microsoft, 2004.
WINCK, D.; JÚNIOR, G. **AspectJ: Programação Orientada a Aspectos com Java**. 1.ed. São Paulo: Novatec, 2006.

VARASCHIM, Jacques Douglas**. Implantando o SCRUM em um Ambiente de Desenvolvimento de Produtos para Internet.** Rio de Janeiro, 2009.

10. APÊNDICE

10.1. DIAGRAMA DE OBJETOS

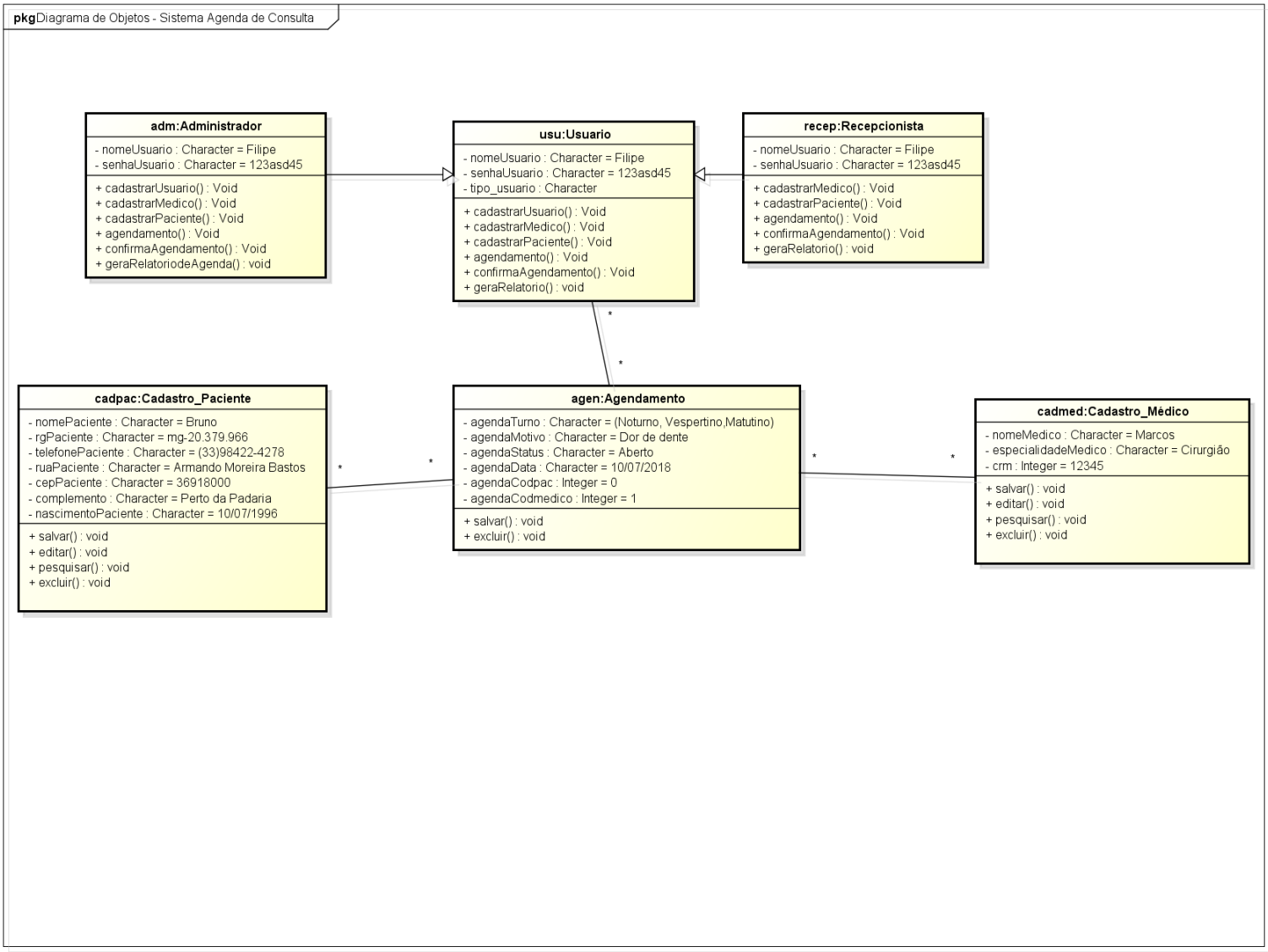

Figura 20: Diagrama de Objetos.

10.2.DIAGRAMA DE SEQUÊNCIA

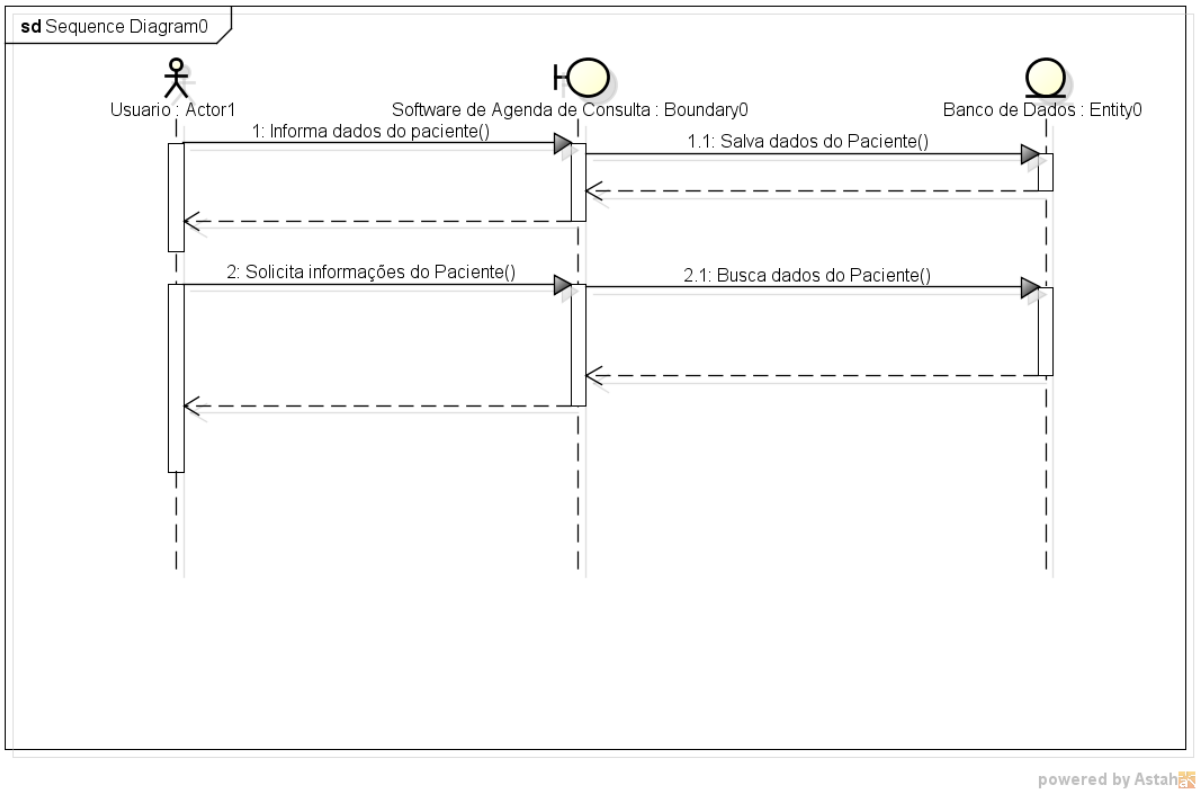

Figura 21: Diagrama de Sequência.

10.3. DIAGRAMA DE MÁQUINA DE ESTADO

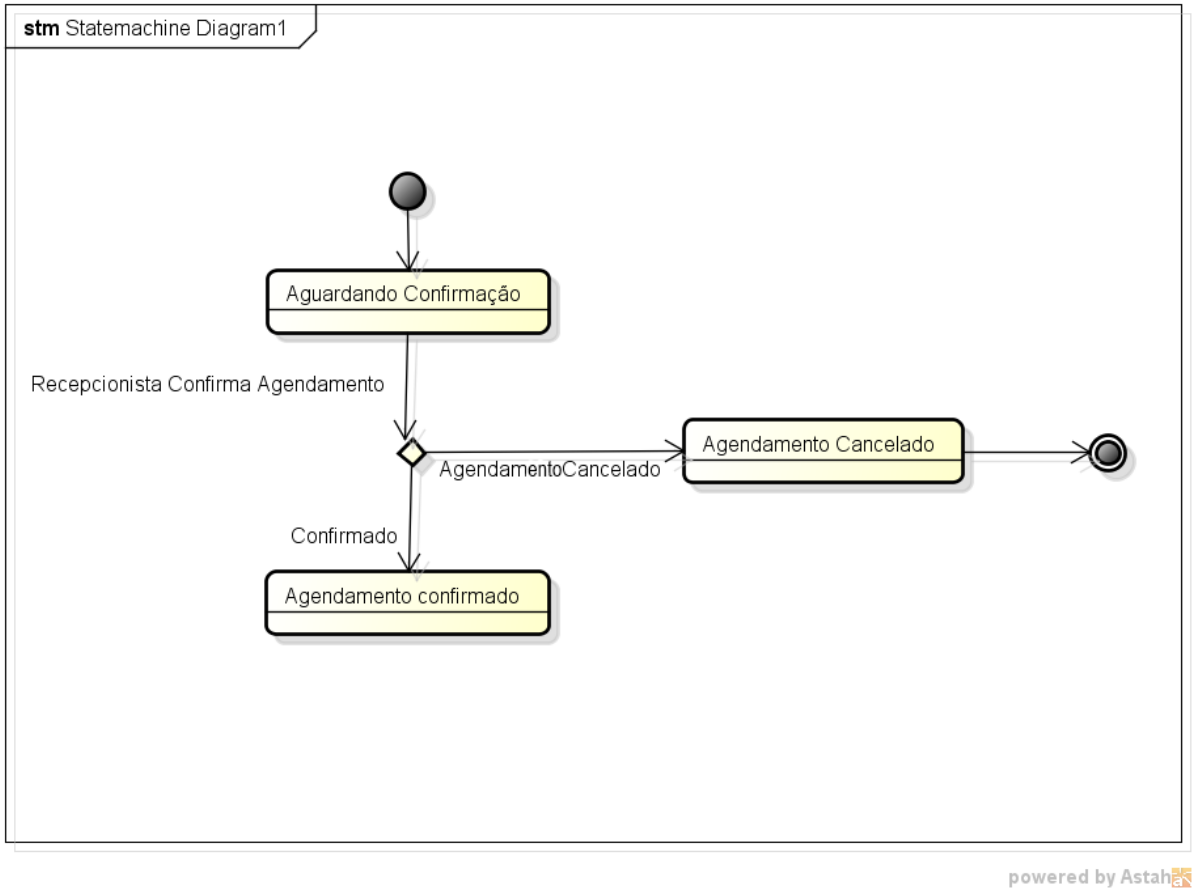

Figura 22: Diagrama de Estado.

10.4.DIAGRAMA DE ATIVIDADE

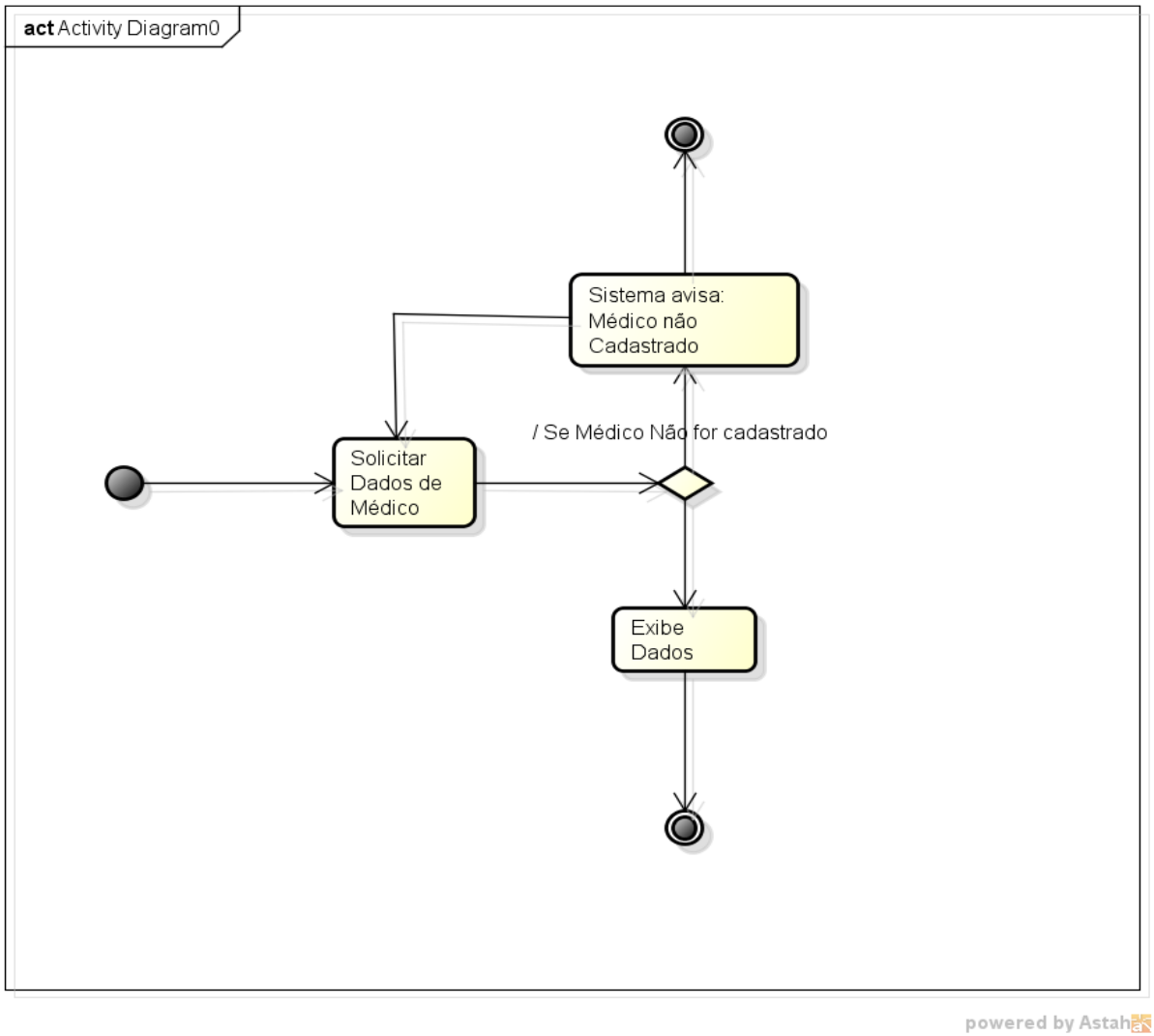

Figura 23: Diagrama de Atividades.

10.5. DIAGRAMA DE CLASSES

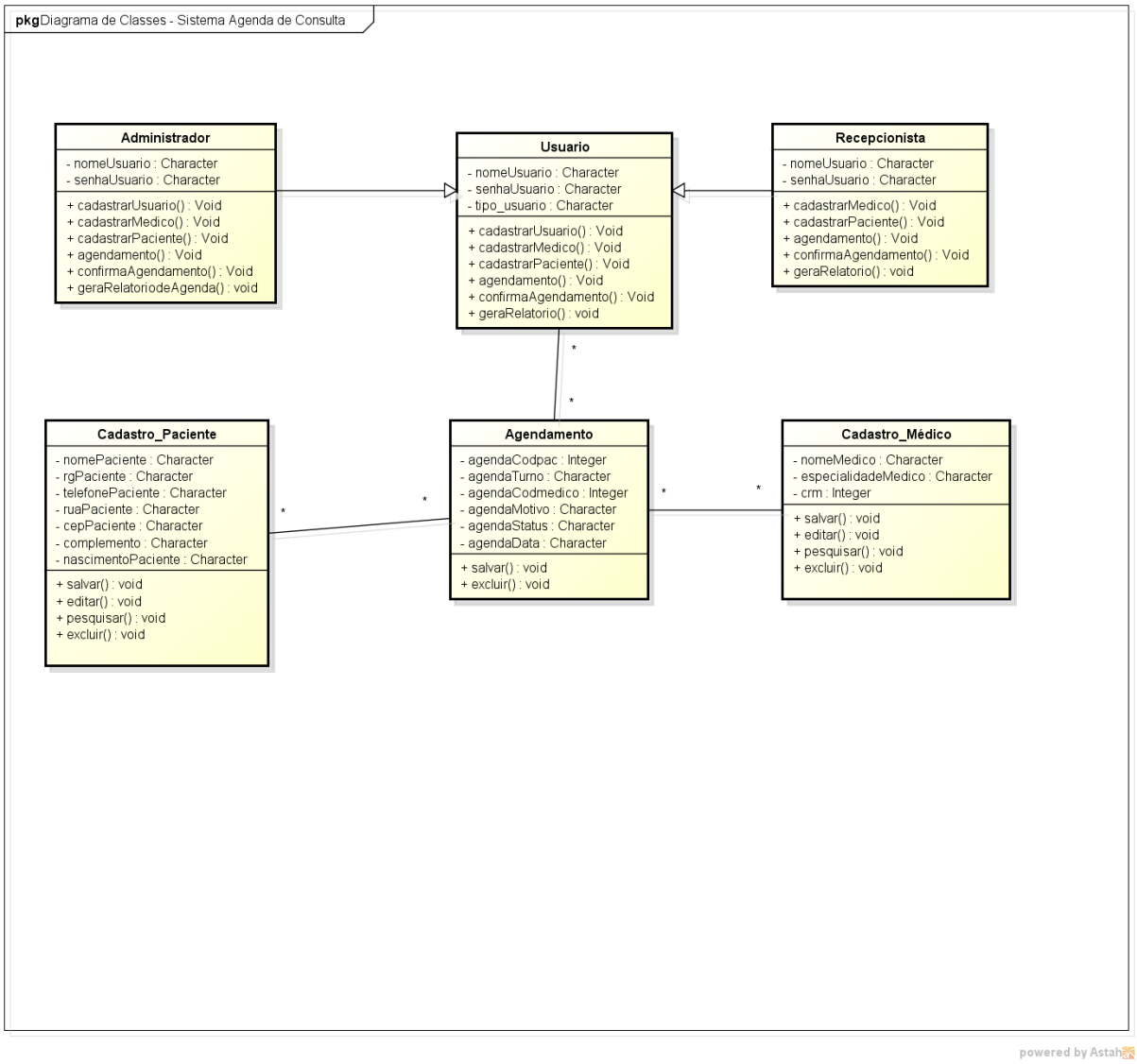

Figura 23: Diagrama de Classe.

10.6. DIAGRAMA DE COMPONENTES

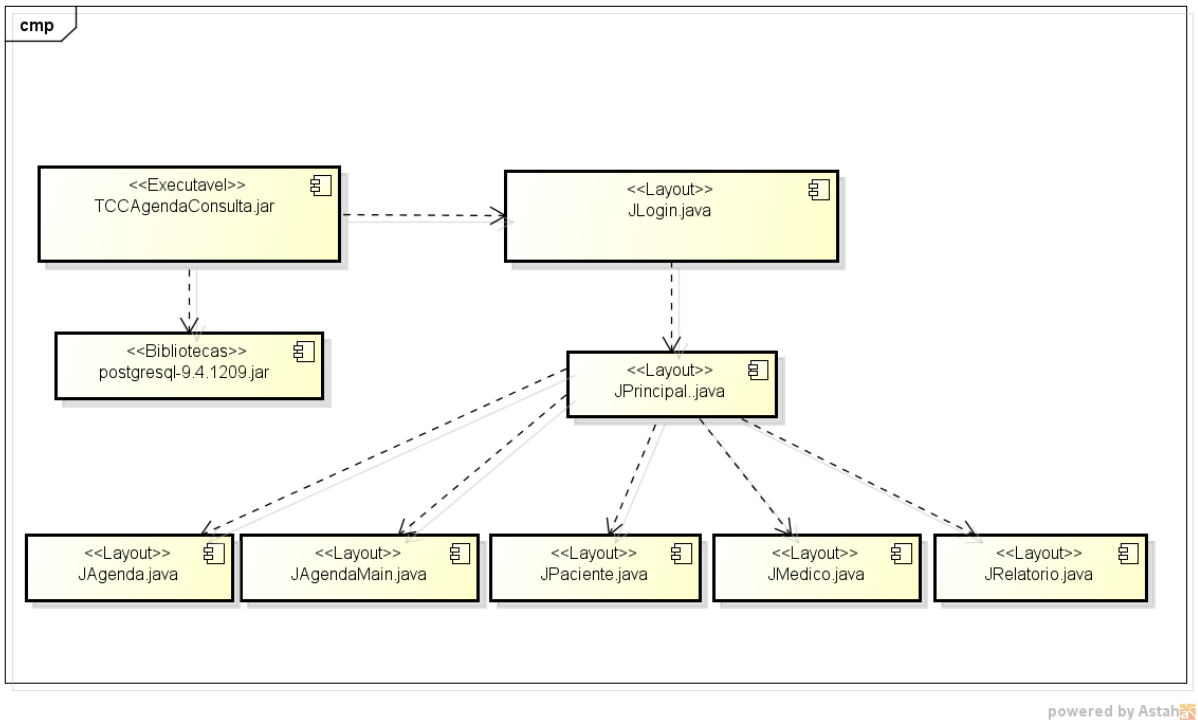

Figura 24: Diagrama de Componentes.