

Programação Modular

Trabalho Final

Clínica Saracura

Estevão Andrade¹, Lucas Silvestre¹

¹Sistemas de Informação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, MG – Brazil

`estevaomoraes1@hotmail.com, lsilvs@dcc.ufmg.br`

1. Descrição do Problema

Este projeto visa a prática do conteúdo e conceitos apresentados durante a disciplina de Programação Modular do curso de Sistemas de Informação da UFMG. A proposta do projeto é desenvolver um programa para controlar o atendimento a clientes de uma clínica de saúde. A clínica oferece serviços de consultas em várias especialidades e a realização de exames de imagens (tomografias, scan, Raio X, etc.). A clínica funciona das 08 às 17 horas e tanto as consultas quanto os exames são realizados a cada 30 minutos. Os usuários deverão ser capazes de marcar uma consulta ou exame através do sistema. O cancelamento de atendimento já realizado também poderá ser feito.

Para usar os serviços da clínica, o cliente deve agendar o atendimento com antecedência. No momento do agendamento, o sistema verifica se o cliente já está cadastrado. Se não estiver, o sistema deve cadastrar o nome do cliente, número da identidade e do CPF, endereço, telefone e data de nascimento.

Para agendar uma consulta, o cliente escolhe a especialidade e o sistema lista os médicos cadastrados nesta especialidade. O cliente então seleciona um dos médicos cadastrados e a agenda do médico é apresentada para que o cliente escolha dia e horário da consulta.

Para o agendamento de exames, o cliente deve fornecer o nome do exame. O sistema verifica se aquele exame é realizado pela clínica e fornece uma listagem das próximas datas disponíveis para sua realização. O usuário escolhe uma das datas, ou desiste do agendamento. Quando o usuário escolher uma data o sistema fornece os horários disponíveis.

O projeto deve ser desenvolvido sob o paradigma de Orientação a Objetos e usando Interface Gráfica. Os dados de teste deverão ser criados pelos projetistas e persistidos da maneira que achar melhor.

2. Diagrama de Classes

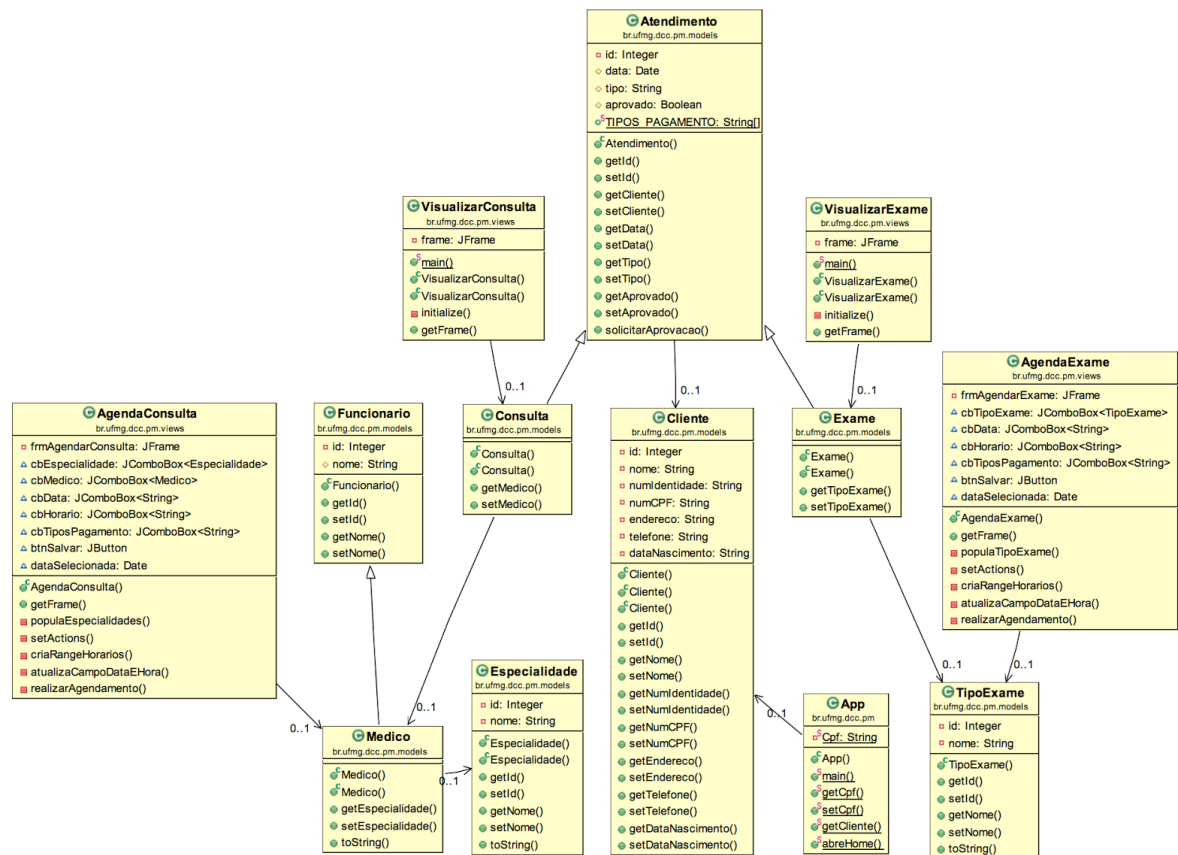


Diagrama de Classes 1 - Visão Geral

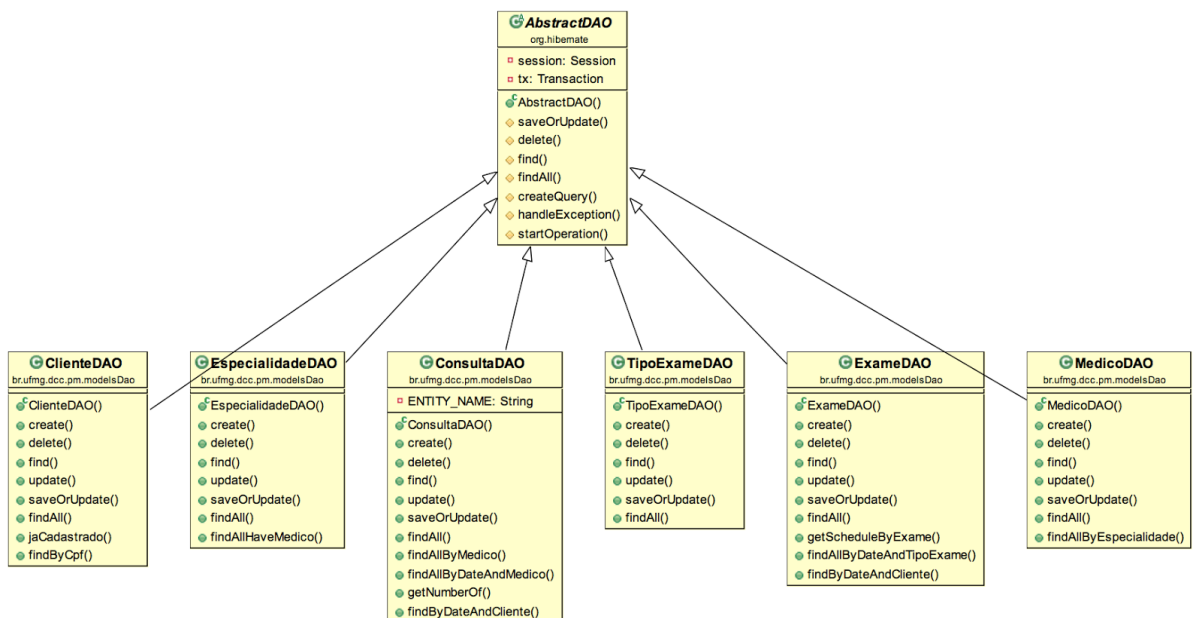


Diagrama de Classes 2 - Data Access Object

3. Implementação

O projeto foi implementado utilizando-se a linguagem Java e todos os conceitos aprendidos em sala de aula.

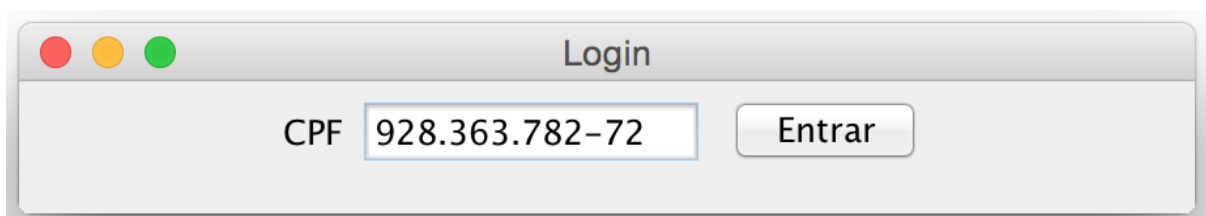
3.1. Entidades

As entidades foram definidas de forma a abranger os conceitos exigidos e a viabilizar um programa intuitivo e de fácil compreensão. A entidade Funcionário foi criada como super classe de Médico e poderá ser estendida a Técnico e Administrativo em uma possível evolução do projeto. Todo Funcionário tem um nome e um identificador, mas somente o Médico possui uma Especialidade. Logo, os atributos comuns encontram-se na super classe e os específicos na sub classe. A relação entre Médico e Especialidade é uma relação direta e unitária, ou seja, um Médico possui somente uma Especialidade. Similarmente à entidade Funcionário, temos a entidade Atendimento, uma super classe para Consulta e Exame. Uma Consulta somente se difere de um Exame por um atributo. Enquanto a Consulta possui um Médico, um Exame possui um TipoExame. Todo e qualquer Atendimento possui um Cliente e uma data (tipo Date do java.util).

3.2. Interface Gráfica

Toda interface gráfica utilizada foi implementada com WindowsBuilder utilizando JFrame, JPanel, JLabel, JComboBox, entre outros.

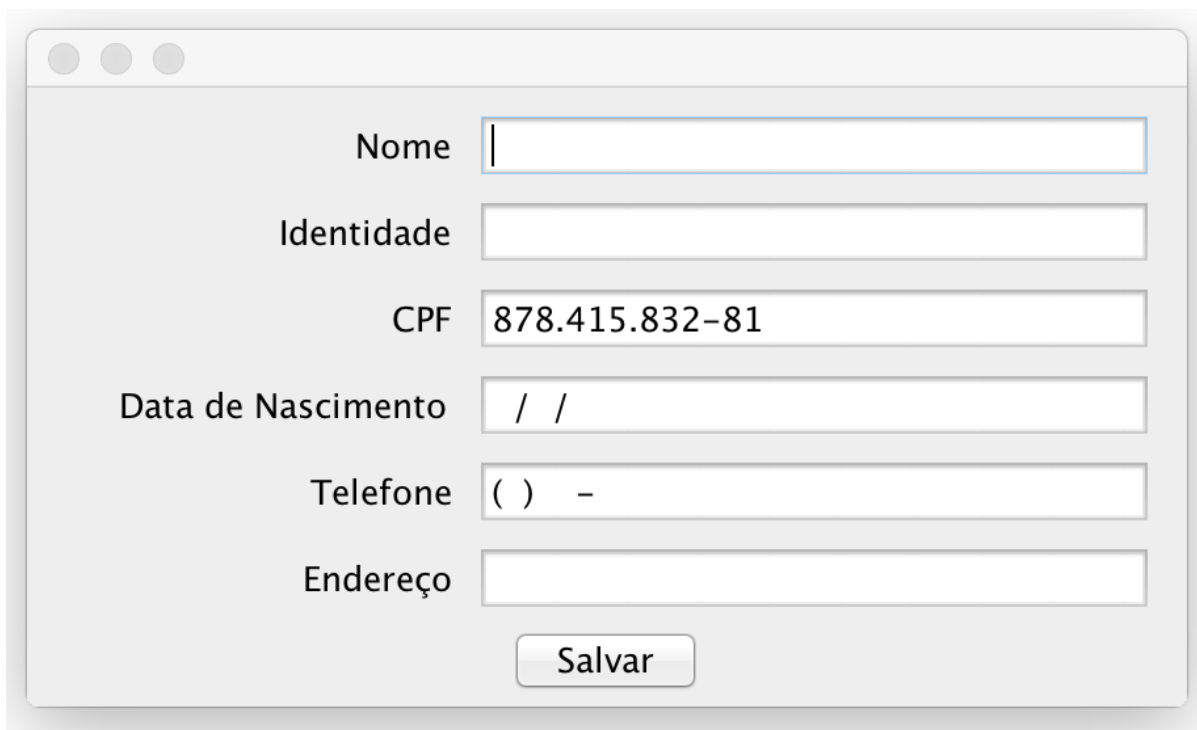
Na imagem Tela 1 temos a janela de login onde o usuário é requisitado a inserir seu CPF. Após inserí-lo, o sistema verifica se esse usuário já está cadastrado. Caso negativo, a tela de cadastro é exibida (Tela 2).



Tela 1 - Login

A Tela 2 é a tela de cadastro de cliente. O CPF digitado na Tela 1 é utilizado e pré-preenchido para o cadastro. Caso o CPF não seja um CPF válido, a exceção InvalidCpfException é lançada e o usuário é informado. O usuário então digita seu nome, identidade, data de nascimento, endereço e telefone e salva seus dados.

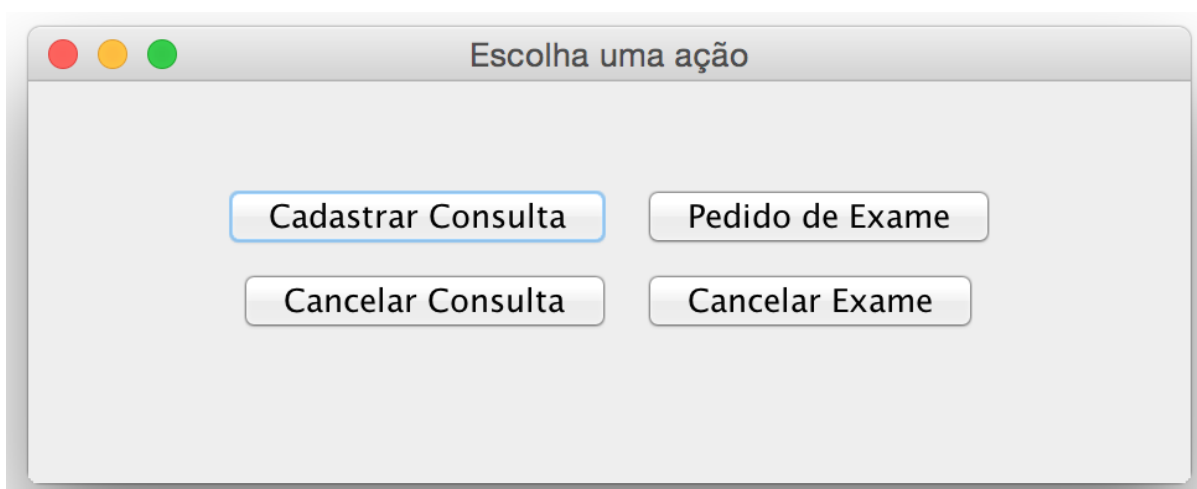
Uma vez logado, o sistema registra o cliente e toda ação subsequente utilizará seus dados sem a necessidade de os informá-lo novamente.



A screenshot of a web form titled 'Tela 2 - Cadastro de Cliente'. The form is enclosed in a light gray border with a standard macOS-style title bar at the top containing three colored window control buttons (red, yellow, green). The form contains several input fields with labels to their left: 'Nome' with an empty text box; 'Identidade' with an empty text box; 'CPF' with a text box containing '878.415.832-81'; 'Data de Nascimento' with a text box containing '/' and '/' as placeholders; 'Telefone' with a text box containing '() -' as placeholders; and 'Endereço' with an empty text box. Below the input fields is a single button labeled 'Salvar'.

Tela 2 - Cadastro de Cliente

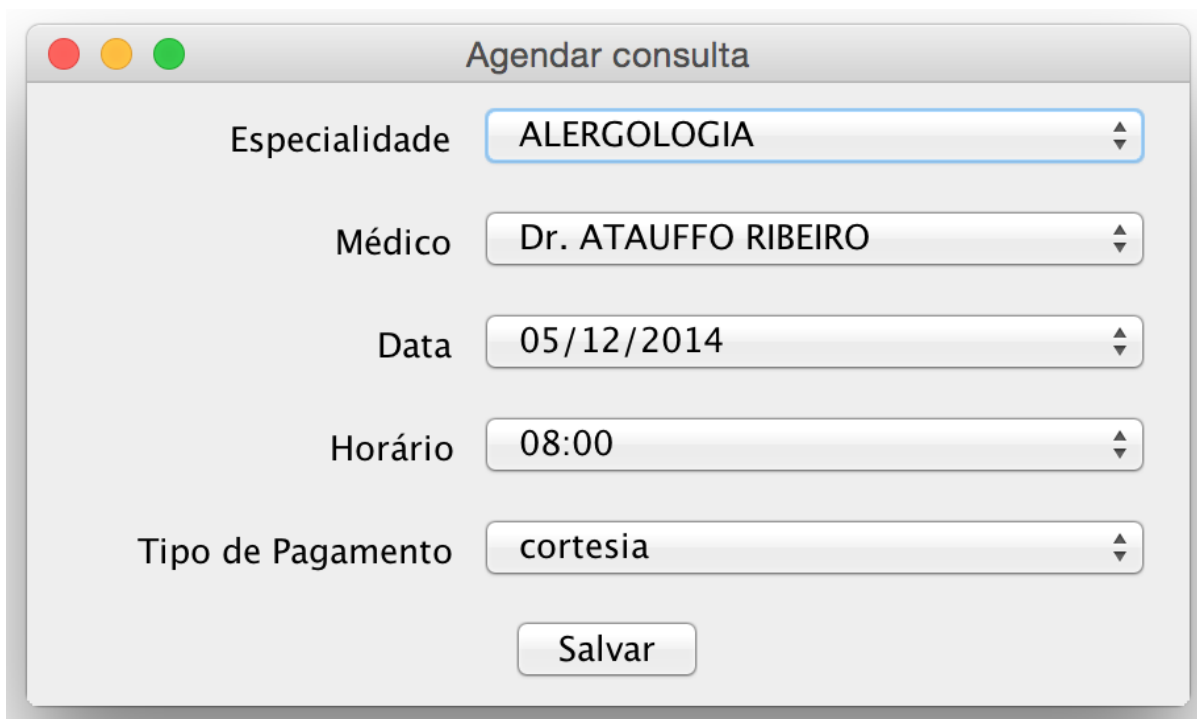
Após se cadastrar ou logar com um CPF já cadastrado, o cliente vê a Tela 3. Essa tela é um menu com as opções do sistema. O cliente poderá agendar uma nova consulta, agendar um novo exame, cancelar uma consulta previamente agendada ou cancelar um exame previamente agendado.



A screenshot of a web form titled 'Tela 3 - Menu de opções'. The form has a light gray border and a title bar with three colored window control buttons (red, yellow, green). The title 'Escolha uma ação' is centered at the top. The main area of the form contains four buttons arranged in a 2x2 grid: 'Cadastrar Consulta' (top-left), 'Pedido de Exame' (top-right), 'Cancelar Consulta' (bottom-left), and 'Cancelar Exame' (bottom-right). The 'Cadastrar Consulta' button is highlighted with a blue border.

Tela 3 - Menu de opções

Ao selecionar a opção "Cadastrar Consulta", o cliente poderá escolher uma especialidade. Ao selecioná-la, a lista com os médicos da especialidade selecionada será exibida. Após escolher o médico, uma lista com os próximos vinte dias será preenchida e, por fim, ao selecionar o dia, a lista com os horários disponíveis será exibida.



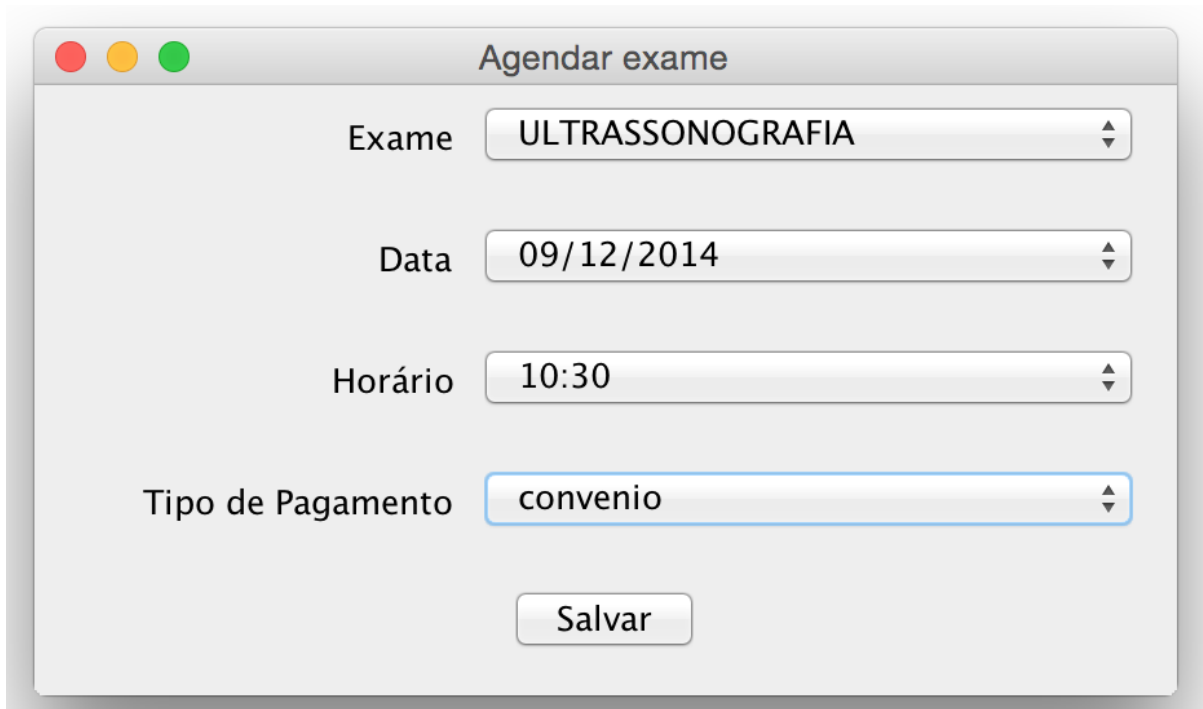
The image shows a software window titled "Agendar consulta". It contains five dropdown menus arranged vertically, each with a label to its left. The first dropdown is labeled "Especialidade" and shows "ALERGOLOGIA". The second is labeled "Médico" and shows "Dr. ATAUFFO RIBEIRO". The third is labeled "Data" and shows "05/12/2014". The fourth is labeled "Horário" and shows "08:00". The fifth is labeled "Tipo de Pagamento" and shows "cortesia". Below these dropdowns is a button labeled "Salvar".

Field	Value
Especialidade	ALERGOLOGIA
Médico	Dr. ATAUFFO RIBEIRO
Data	05/12/2014
Horário	08:00
Tipo de Pagamento	cortesia

Salvar

Tela 4 - Marcação de Consulta

Da mesma forma que em "Cadastro de Consulta", o "Pedido de Exame" exibe uma lista com os exames ofertados pela clínica. Ao selecionar o exame desejado o cliente poderá escolher uma data e ao selecionar a data os horários disponíveis serão apresentados.



Agendar exame

Exame ULTRASSONOLOGRAFIA

Data 09/12/2014

Horário 10:30

Tipo de Pagamento convenio

Salvar

Tela 4 - Marcação de Exame

3.3. Persistência

A parte de persistência do projeto foi implementada utilizando SQLite como banco de dados e Hibernate como ORM. O SQLite foi escolhido por ser um banco de fácil manipulação e por possibilitar o encapsulamento do mesmo junto ao projeto sem a necessidade de instalação ou configuração de um banco de dados na máquina em que o projeto irá rodar. Já o Hibernate foi escolhido por abstrair a parte de consultas ao banco de dados e facilitar a conexão com o mesmo.

Para integrar o Hibernate com o projeto, foi utilizado a classe `HibernateFactory` para gerenciar as sessões com o banco de dados. A classe `HibernateFactory` é uma implementação de um famoso padrão de projeto chamado Singleton. Um Singleton é uma classe que só possui uma instância e aqui é muito útil pois evita a abertura de várias conexões com o banco de dados utilizando sempre a mesma para as execuções. Outra classe utilizada foi a `SQLiteDialect`. Nela temos atributos e métodos que dão instruções ao Hibernate de como as consultas devem ser construídas.

O Hibernate possui ainda um arquivo de configurações onde informamos as configurações básicas para o seu funcionamento. Esse arquivo se encontra em `/src/main/resources/hibernate.cfg.xml` e possui informações sobre qual o banco de dados utilizado e quais as entidades que são mapeadas. Para que uma entidade seja mapeada, além de constar no arquivo de configuração citado anteriormente, podemos indicar na classe da entidade algumas instruções. Tais instruções são implementadas através do recurso Java chamado Annotations.

4. Decisões de projeto

Para facilitar a implementação foi decidido que o login do cliente se daria utilizando somente o CPF. Além disso, qualquer usuário pode se cadastrar caso ainda não o tenha feito. Como o programa visa atender aos clientes, o cadastro de novos médicos, novos exames ou especialidades foi suprimido. O registro de médicos, exames e especialidades deverão ser cadastrados previamente no banco de dados.

Outra decisão importante foi em relação à agenda de consultas e exames. Cada atendimento possui um atributo data do tipo `java.util.Date` onde armazenamos a data e hora do atendimento. Para verificar se aquele horário está disponível, basta realizar uma consulta no banco de dados para o horário desejado. Essa implementação foi possível pois, conforme especificado no documento do projeto, os atendimentos são realizados sempre de trinta em trinta minutos a começar das oito horas da manhã e encerram-se às cinco horas da tarde.

Os botões de ação foram implementados utilizando `ActionListeners`, dessa forma todo o controle de ações é realizado no método `actionPerformed`.

Todo o desenvolvimento do projeto foi realizado utilizando-se o sistema de controle de versão Git e hospedado no GitHub. Essa decisão foi baseada na confiabilidade do Git e por permitir o desenvolvimento paralelo e remoto.

5. Sugestões para evolução

O projeto poderá ser evoluído implementando-se um login com senha para o usuário. Os médicos poderiam ser classificados por médicos de convênio ou médicos particulares. Os funcionários poderiam ter acesso ao sistema de forma a registrar atendimentos por telefone. Os dirigentes poderiam acessar o sistema para autorizar ou recusar atendimentos (essa função é atualmente um método que simula a autorização).

6. Bibliografia

Hibernate ORM documentation. Disponível em:

<<http://hibernate.org/orm/documentation/>>. Acesso em: 20 nov. 2014.

SQLite for Java. Disponível em:

<http://www.tutorialspoint.com/sqlite/sqlite_java.htm>. Acesso em: 20 nov. 2014.

WindowBuilder - is a powerful and easy to use bi-directional Java GUI designer.

Disponível em: <<https://eclipse.org/windowbuilder/>>. Acesso em: 30 nov. 2014.