

**UNIVERSIDADE FEDERAL DO RIO GRANDE
CENTRO DE CIÊNCIAS COMPUTACIONAIS
SISTEMAS PARA INTERNET II**

RELATÓRIO DO PROJETO DE UM SISTEMA PARA UMA CLÍNICA MÉDICA

ALUNOS
FELIPE FREITAS, 104532
GABRIEL MORAES, 88919
VINICIUS LUCENA, 85522

SUMÁRIO

INTRODUÇÃO	3
REQUISITOS BÁSICOS DO SISTEMA	3
FUNCIONALIDADES EXTRAS (seção 3.4)	4
ESTRUTURA DO PROJETO	5
TECNOLOGIAS	5
CAMADAS LÓGICAS	5
ESTRUTURA DO XML	6
DIAGRAMA DE CLASSES	7
HIERARQUIA DE PÁGINAS	9
RESULTADOS	9
FUNÇÕES DO ATENDENTE	9
FUNÇÕES DO MÉDICO	12
FUNÇÕES DO PACIENTE	14
FUNCIONALIDADES EXTRAS	16
CONCLUSÃO	18
OVERVIEW DO SISTEMA	18

1. INTRODUÇÃO

O presente relatório tem por objetivo explicar o design e implementação de um sistema para uma clínica médica fictícia. O sistema deve ser capaz de atender um conjunto de requisitos, listados na seção 1.1.

O objetivo do sistema é ser um sistema totalmente funcional e prático para clínicas médicas. Incluindo funcionalidades comuns à esse tipo de empresas, porém com uma diferença: um sistema de armazenamento em arquivos XML. Para o desenvolvimento do sistema, é necessário criar o lado do médico e do atendente e, como função extra, o lado do paciente. Os requisitos específicos estão listados abaixo.

Vale salientar que o relatório contará com diversas imagens para o melhor entendimento do leitor. Levando em consideração que as imagens não são autoexplicativas, as mesmas contam com uma explicação (acima ou abaixo).

Incluímos no sistema a opção de um paciente efetuar login para acessar seu histórico de consultas e agendar novas consultas. Caso o paciente não esteja cadastrado, há duas opções: 1) A atendente pode cadastrá-lo ou 2) Ele mesmo pode se cadastrar.

Quando a organização do relatório, estruturamos da seguinte maneira: Em um primeiro momento, iremos falar sobre os requisitos básicos do sistema de maneira superficial, explicando a função de cada um tipos de usuários (paciente, médico e atendente). Após isso, iremos comentar a estrutura do projeto, quais as tecnologias utilizadas, a divisão em camadas lógicas e mais algumas outras coisas. Após esse momento, iremos descrever o que foi implementado, seguindo os itens descritos na seção 1.1. Por fim, na conclusão iremos tratar de alguns pontos positivos e negativos do sistema.

1.1. REQUISITOS BÁSICOS DO SISTEMA

FUNÇÕES DO ATENDENTE (seção 3.1)

- Cadastrar médicos e pacientes

O atendente deve ser capaz de realizar o cadastro tanto de médicos, quanto de pacientes, sendo que médicos possuem algumas informações além do paciente. Caso a função extra não seja implementada, a atendente será a única responsável pelos cadastros.

- Agendar consultas

O atendente é o único com permissão para autenticar consultas, caso ele mesmo as agende, as mesmas serão diretamente autenticadas. Caso o paciente agende, as mesmas passarão pelo atendente para confirmação.

- Visualizar consultas

O atendente tem a opção de ver consultas de determinado médico, assim como de determinado paciente. É uma forma de visualização e controle das consultas.

- Visualizar histórico de consultas dos médicos
- Visualizar históricos de consultas dos pacientes

- Confirmar agendamento de consultas

O atendente é responsável por confirmar consultas marcadas por pacientes para garantir que um paciente marque várias consultas com médicos diferentes ou cometa qualquer tipo de engano.

FUNÇÕES DO MÉDICO (seção 3.2)

- **Alterar seu próprio cadastro**

O médico é cadastrado somente através do atendente, porém, ele mesmo pode alterar seu cadastro depois de estar logado no sistema.

- **Visualizar sua agenda de consultas**

O médico pode ver sua agenda de consultas, verificar quais seus pacientes recentes, sendo próximos ou anteriores.

- **Preencher observações e receita nas suas consultas**

Junto à visualização da agenda ele pode preencher observações e receitas em cada paciente, assim como diagnosticá-los.

- **Visualizar histórico de consultas dos pacientes**

No caso de um médico estar prestes a atender um paciente, ele pode ver o histórico daquele paciente com informações de diagnóstico e receita que o paciente obteve daquela consulta anterior.

FUNÇÕES DO PACIENTE (seção 3.3)

- **Realizar o próprio cadastro**

O paciente deve poder se registrar com seus dados para acessar o sistema.

- **Alterar seu cadastro**

Assim como o médico, quando o paciente já está logado no sistema, ele é capaz de alterar seu próprio cadastro.

- **Realizar agendamento de consultas**

O paciente consegue agendar consultas, essas devem ser validadas pela atendente para impedir problemas.

- **Visualizar seu histórico de consultas**

O paciente pode visualizar suas consultas para ver quando e com qual médico consultou.

Embora esses foram os requisitos básicos do projeto, fomos mais além e, implementamos algumas outras funcionalidades que julgamos interessantes e que melhoraram a qualidade final do trabalho.

1.2. FUNCIONALIDADES EXTRAS (seção 3.4)

- **Página personalizada para cada médico**

Uma forma de retornar o nome do médico e o horário do dia para desejar um Bom Dia/Boa tarde/Boa noite/Boa madrugada ao médico.

- Ordenar os agendamentos em ordem cronológica

Para mostrar todas as tabelas de histórico e agenda, organizamos em ordem cronológica.

- Tela de login

A tela de login se viu necessária para o controle de acesso para cada funcionalidade, assim conseguimos controlar se o acesso é de um médico, um paciente ou um atendente.

Na próxima seção veremos como o projeto foi estruturado, quais foram as tecnologias empregadas, tanto no front-end quanto no back-end. Como se deu a estruturação lógica do sistema, como foram estruturados os arquivos XML e por fim, o diagrama de classes e a hierarquia de páginas.

2. ESTRUTURA DO PROJETO

2.1. TECNOLOGIAS

FRONT-END

Para o front-end, foram utilizadas as linguagens HTML, CSS e JavaScript, junto com o framework Bootstrap. A utilização desse conjunto é bem comum hoje em dia na criação de sites por permitir uma grande variedade de funcionalidades e personalização do site. Além disso, utilizamos a biblioteca jQuery de JavaScript, em conjunto com o Bootstrap para criar algumas funções específicas. E, por fim, a técnica AJAX para uma interação entre o *user-side* e o *server-side* (arquivos XML).

FUNCIONAMENTO DO AJAX

O AJAX (Asynchronous JavaScript And XML) serve para trocar dados entre o cliente e o servidor de maneira assíncrona, dessa maneira, conseguimos atualizar o conteúdo de uma página sem precisar recarregá-la.

A figura 1 mostra como utilizamos essa técnica no nosso sistema.

```
onClick="ajaxPost('../server/verAgendaSimplificadaPaciente.php', '#result-historico')"
```

Figura 1. Exemplo de uso do AJAX

BACK-END

Em relação ao back-end, utilizamos a linguagem de programação PHP, visto que é amplamente utilizada na indústria de TI, conta com uma comunidade grande e colaborativa, além disso possui inúmeras funções já implementadas e foi construída voltada para web.

2.2. CAMADAS LÓGICAS

A figura 2 mostra a estruturação do projeto quanto as camadas lógicas.

CAMADA DE APRESENTAÇÃO:

- Responsável por interagir com o usuário.

- Onde está o localizado o *user-side*.
- Onde fica as páginas web.

CAMADA DE PROCESSAMENTO:

- Responsável por processar todas as requisições dos usuários.
- Decisão de projeto: Criamos diversos scripts em php que lidam com as operações que os usuários querem realizar. Fizemos isso para aumentar a reutilização de código, além disso, é mais simples de identificar algum bug caso dê algum problema, uma vez que os scripts são inteiramente separados uns dos outros.

CAMADA DE PERSISTÊNCIA:

- Responsável por guardar os dados do sistema: dados de cadastro de pacientes e médicos, além de guardar informações de quem pode logar como atendente.

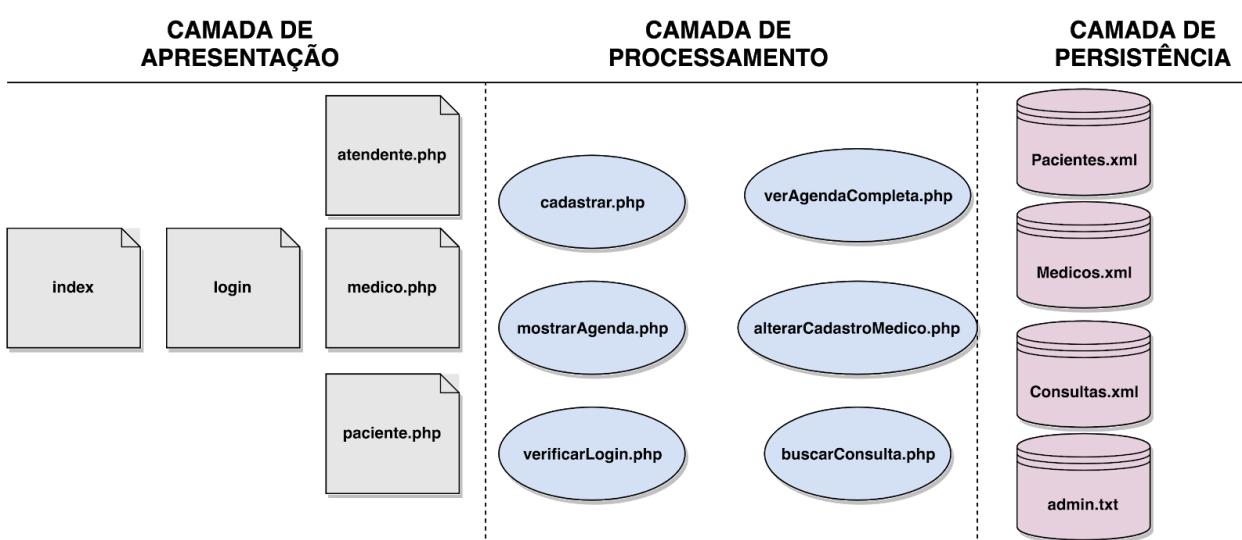


Figura 2. Camadas lógicas.

2.3. ESTRUTURA DO XML

A figura 3a mostra a árvore da estrutura XML que é responsável por guardar as informações dos médicos. A figura 3b mostra um exemplo de como fica guardado os dados de um médico cadastrado.

Uma decisão de projeto foi separarmos o "banco de dados" em quatro arquivos diferentes. Mas por quê fizemos isso?

Tem diversos motivos:

- 1) Diminuir a complexidade do sistema.

Uma vez que é mais fácil manipular somente os dados que estamos interessados. Se estamos implementando a função cadastrar médico, não faz

sentido carregar um arquivo XML que guarda informações de pacientes e de consultas. Isso nos leva ao segundo argumento.

2) Desempenho

Dividindo o banco em três arquivos, podemos carregar somente o arquivo que nos interessa naquele momento, isso reduz (e muito) o processamento para inserir, consultar e remover dados do arquivo.

- OBS: Se quisermos utilizar um banco de dados, cada arquivo XML se transformaria em uma tabela no banco, simplesmente isso.

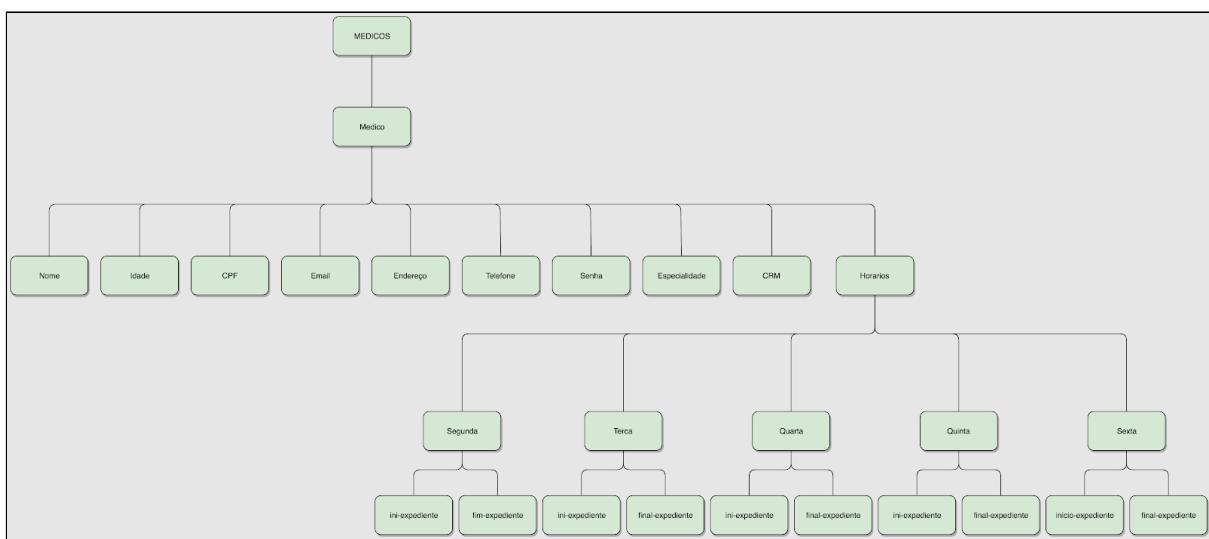


Figura 3a. Estrutura do XML que guarda informações de cadastro dos médicos.

```

<Medico>
  <nome>Cicera Tavares</nome>
  <idade>1900-01-20</idade>
  <cpf>111.111.111-11</cpf>
  <email>vinicius.lucena97@gmail.com</email>
  <endereco>Rua Santos Dumont, n434</endereco>
  <telefone>(02) 3122-12212</telefone>
  <senha>senha</senha>
  <crm>123456</crm>
  <especialidade>Todas</especialidade>
  <horarios>
    <segunda>
      <ini-expediente-seg>10:00</ini-expediente-seg>
      <fim-expediente-seg>18:00</fim-expediente-seg>
    </segunda>
    <terca>
      <ini-expediente-ter>10:00</ini-expediente-ter>
      <fim-expediente-ter>18:00</fim-expediente-ter>
    </terca>
    <quarta>
      <ini-expediente-qua>10:00</ini-expediente-qua>
      <fim-expediente-qua>18:00</fim-expediente-qua>
    </quarta>
    <quinta>
      <ini-expediente-qui>10:00</ini-expediente-qui>
      <fim-expediente-qui>18:00</fim-expediente-qui>
    </quinta>
    <sexta>
      <ini-expediente-sex>10:00</ini-expediente-sex>
      <fim-expediente-sex>18:00</fim-expediente-sex>
    </sexta>
  </horarios>
</Medico>
  
```

Figura 3b. Exemplo do XML de um médico cadastrado.

2.4. DIAGRAMA DE CLASSES

A figura 4 ilustra o diagrama de classes do projeto. Essa estrutura foi utilizada para a criação do sistema, pensando-se no funcionamento como um todo, porém, essa foi a base para o *back-end* do mesmo.

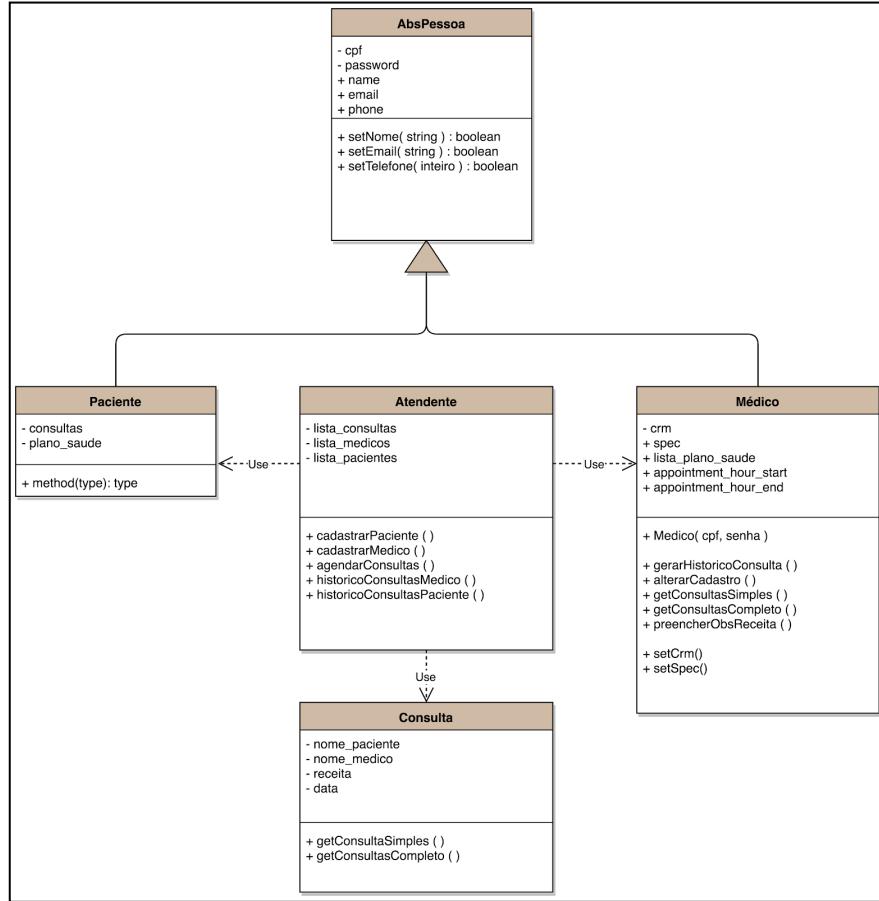


Figura 4. Diagrama de classes.

Ao todo foram utilizadas 5 classes.

1. CLASSE *AbsPessoa*

- É uma superclasse abstrata, isto é, não implementa todos os seus métodos e, portanto, não pode ser instanciada.
- Ela funciona para que não precisemos replicar atributos, visto que tanto paciente quanto médico são pessoa.

2. CLASSE *Paciente*

- É uma subclasse da classe *AbsPessoa*
- Responsável por guardar informações dos pacientes

3. CLASSE *Atendente*

- É a principal classe do sistema, visto que a atendente é responsável por administrá-lo.
- Implementa todas as funções que somente a atendente deve realizar.

4. CLASSE *Médico*

- Guarda as informações dos médicos.
- Responsável por implementar as funcionalidades do médico.

5. CLASSE *Consulta*

- Guarda informações sobre uma consulta ou agendamento.
 - Quando é feito um agendamento, um objeto da classe consulta é instanciado, passando o nome do paciente, nome do médico, data e horário do agendamento, com o campo receita e diagnóstico vazios. No momento em que uma consulta é realizada, os campos receita e diagnóstico são preenchidos exclusivamente pelo médico.
 - Em resumo, a mesma classe representa uma consulta e um agendamento.
 - Poderíamos não usar classes? A resposta é sim, contudo uma enorme vantagem em estruturar o sistema com classes é a modularidade.
- Exemplo:** E se no futuro, quisermos cadastrar novos atendentes? Bom, podemos fazer com que a classe Atendente também herde os métodos e atributos da classe *AbsPessoa*, dessa maneira, resolvemos facilmente o problema.

2.5. HIERARQUIA DE PÁGINAS

Em relação às páginas implementadas no front-end, foram utilizadas apenas 4 páginas. Não precisamos de mais páginas porque utilizamos a tecnologia AJAX (discutido na seção 2.1) para atualizar a própria página, sem precisar que o servidor redirecione para outra página. Posteriormente foi adicionada uma página extra, justamente em relação à função extra (paciente). A página de registro do mesmo, que será citada posteriormente.

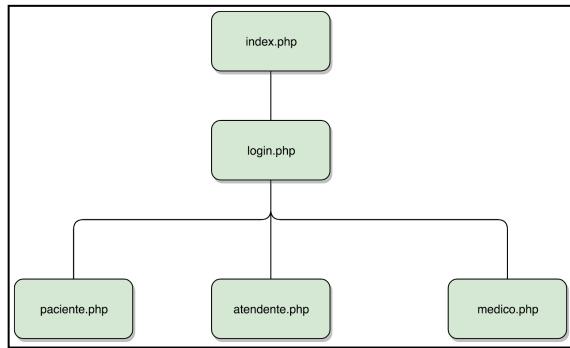


Figura 5. Hierarquia de Páginas.

3. RESULTADOS

Nesta seção do trabalho, apresentaremos algumas das funções que foram implementadas.

3.1. FUNÇÕES DO ATENDENTE

- **Cadastrar médicos e pacientes**

Consideramos o caso do médico ter diversos horários, dependendo do dia da semana. Os campos que incluímos no formulário podem facilmente serem alterados. Dependendo da clínica, pode ser útil saber outras informações, como o tipo sanguíneo, por exemplo.

Cadastro

Nome: Arminda de Jesus
Data de Nascimento: 20/03/1950
CPF: 182.312.381-28
Email: ariminiana@gmail.com
Endereço: Rua das Flores, n 456
Telefone: (18)99828-4850
Senha:

Médico

CRM: 381231
Specialidade: Pediatra
Horário/Dia

SEG	09:00	até	10:00
TER	18:00	até	22:00
QUA	23:00	até	05:00
QUI	18:00	até	20:00
SEX	10:00	até	16:00

Paciente Médico

Cadastrar

Figura 6. Tela de cadastro de médico

A figura 7 descreve os passos realizados para o atendente cadastrar um médico ou um paciente no sistema de clínica médica.

Passos:

- 1) Atendente preenche o formulário, o qual é enviado via AJAX para o servidor.
- 2) O script *cadastrar.php* é responsável por identificar quem está sendo cadastrado (médico ou paciente), após essa etapa, acessa o "banco de dados", verifica se o registro já existe, caso negativo, insere no banco.
- 3) Retorna uma mensagem para o usuário dizendo se a ação pôde ser realizada ou houve algum problema (caso o registro já exista).

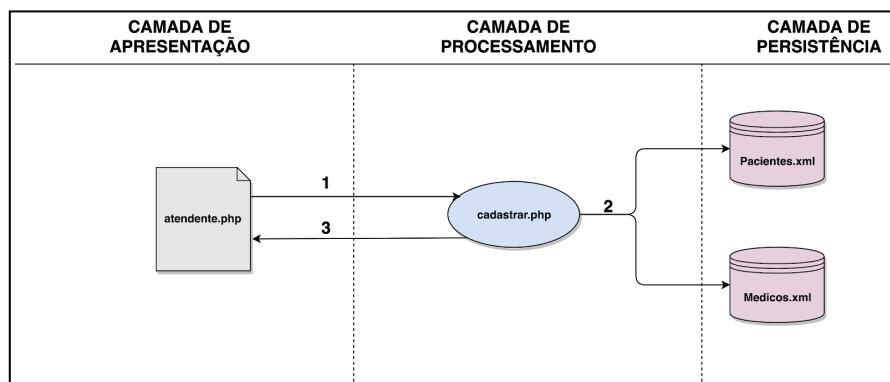


Figura 7. Passos para cadastrar um médico ou paciente.

A figura 7 é bem geral, o mesmo procedimento ocorre com todas as funcionalidades do sistema. O que muda é o script php na camada de processamento e a página de exibição de conteúdo na camada de apresentação.

Note que no passo 3, nós devolvemos o resultado do cadastro (se deu certo ou não) para a própria página, para fazer isso, utilizamos o AJAX (essa é outra questão de projeto).

• **Agendar consultas**

A figura 8 mostra a implementação da função agendar consulta. Essa função é feita pela atendente que tem como finalidade agendar uma consulta de um paciente, para determinado médico

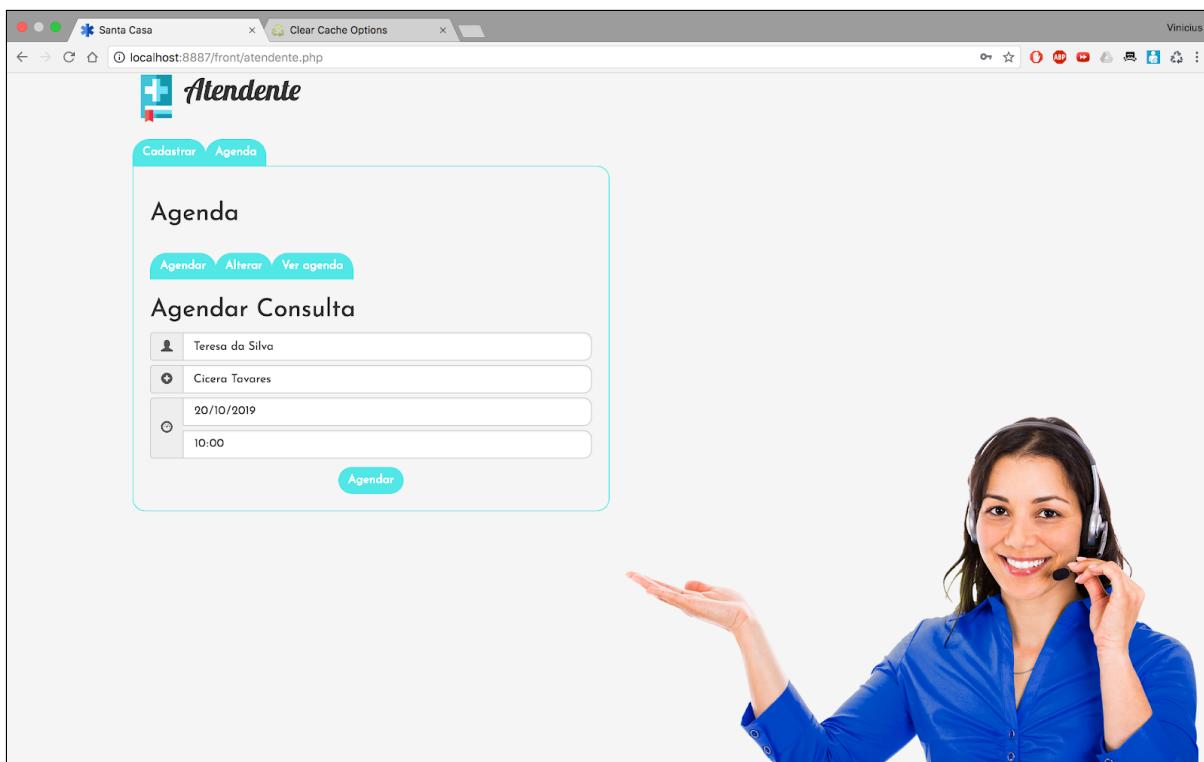


Figura 8. Tela de agendamento de consultas.

- **Visualizar histórico de consultas dos médicos**

A figura 9 mostra como ficou a tela de visualização da agenda do médico por parte do atendente. Quando o atendente for digitar o nome do médico, aparece uma lista dos médicos já cadastrados.

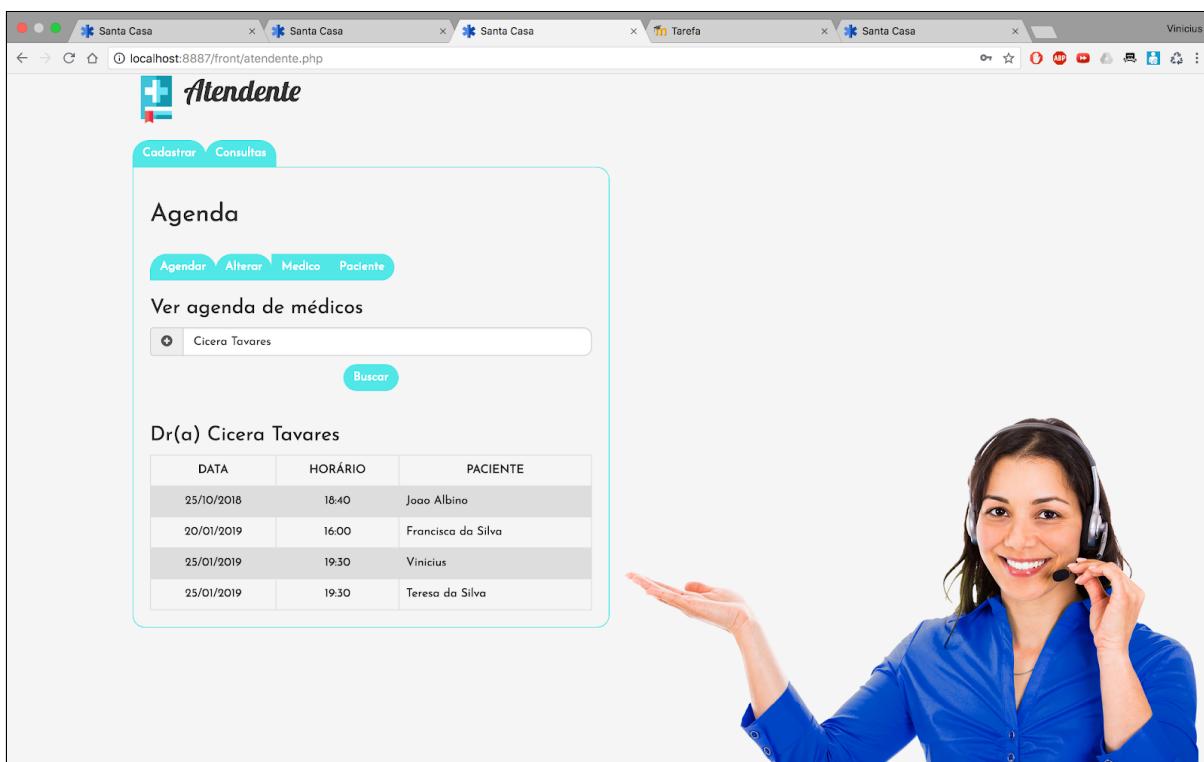


Figura 9. Tela de visualização da agenda dos médicos.

- **Confirmar agendamento de consultas**

Quando um paciente, no conforto do seu lar, realiza um agendamento, os dados da consulta vão para uma lista chamada ConsultasPendentes.xml. Ao passo que o atendente acessa essa lista e escolhe se irá confirmar o agendamento ou não. Caso o agendamento seja confirmado, os dados da consulta são movidos para o arquivo Consultas.xml, caso o agendamento seja cancelado, os dados são removidos do arquivo ConsultasPendentes.xml. Todo isso é demonstrado visualmente pela figura 10a.

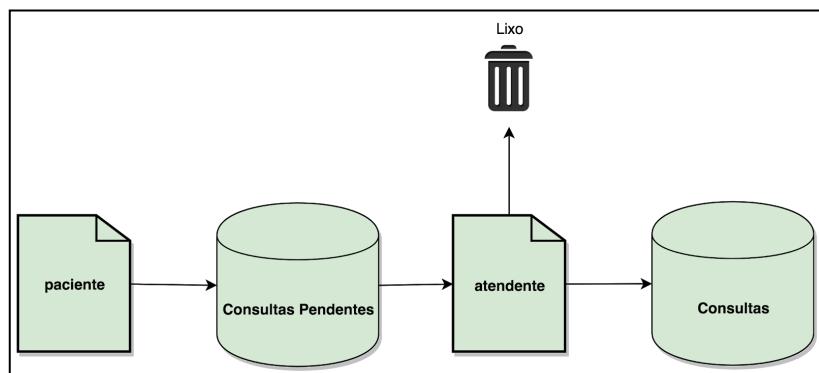


Figura 10a. Esquema do paciente agendar uma consulta.

A figura 10b mostra como ficou a tela de confirmação de agendamentos de consultas por parte do atendente. A parte de programação é muito semelhante a função do médico receitar ou diagnosticar um paciente.

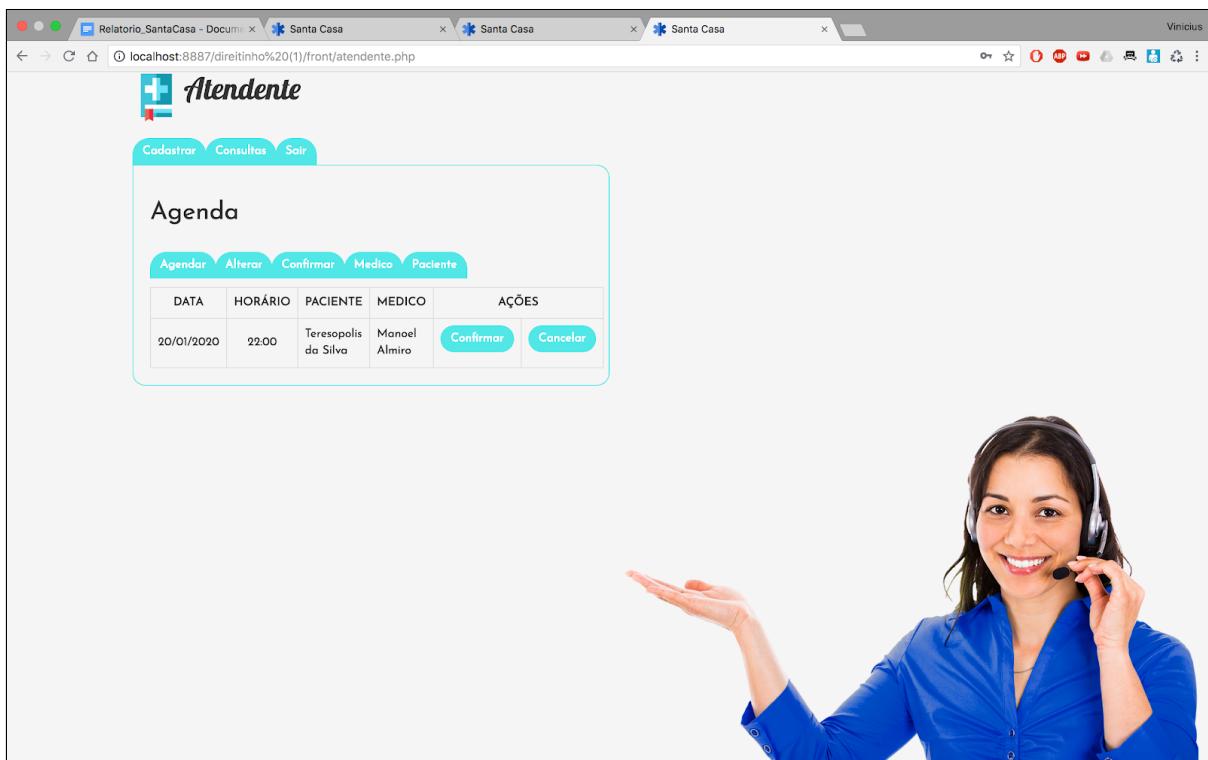


Figura 10b. Tela de confirmação ou cancelamento de um agendamento de uma consulta por parte do atendente.

3.2. FUNÇÕES DO MÉDICO

- Alterar seu próprio cadastro

Os dados de cadastro de um médico devem poder ser alterados exclusivamente por ele. A figura 12 mostra como ficou a tela de alteração de cadastro do médico.

Implementamos esta função da seguinte maneira:

- 1) Ao clicar no botão "meu perfil", é ativado uma função, escrita em javascript, para mostrar o formulário.
- 2) O formulário é aut preenchido com as informações do médico que está atualmente logado no sistema.

A figura 11 mostra como foram feitos os dois primeiros campos do formulário (nome e idade). Note que cada campo possui um identificador (é importante para que o servidor accesse os dados enviados).

O principal motivo pelo qual estamos mostrando esta figura é para mostrar o atributo value. Note que incluímos um código PHP para acessarmos o superglobal Session e pegar os dados do médico atualmente logado.

```
<div class="input-group">
  <span class="input-group-addon"><i class="glyphicon glyphicon-user"></i></span>
  <input id="name" class="form-control" type="text" name="name" value=<?php echo $_SESSION['nome'];?>" placeholder="Nome" required>
  <input id="age" class="form-control" type="date" name="age" value=<?php echo $_SESSION['idade'];?> required>
</div>
```

Figura 11. Parte do formulário aut preenchido do médico.

- 3) Após o médico alterar o que ele desejar, ao clicar no botão salvar, uma requisição HTTP é realizada via AJAX e os dados do formulário são passados para o servidor.
- 4) O servidor accessa o arquivo Medicos.xml e atualiza os dados do referente médico.
- 5) Uma mensagem de confirmação é devolvida para o médico.

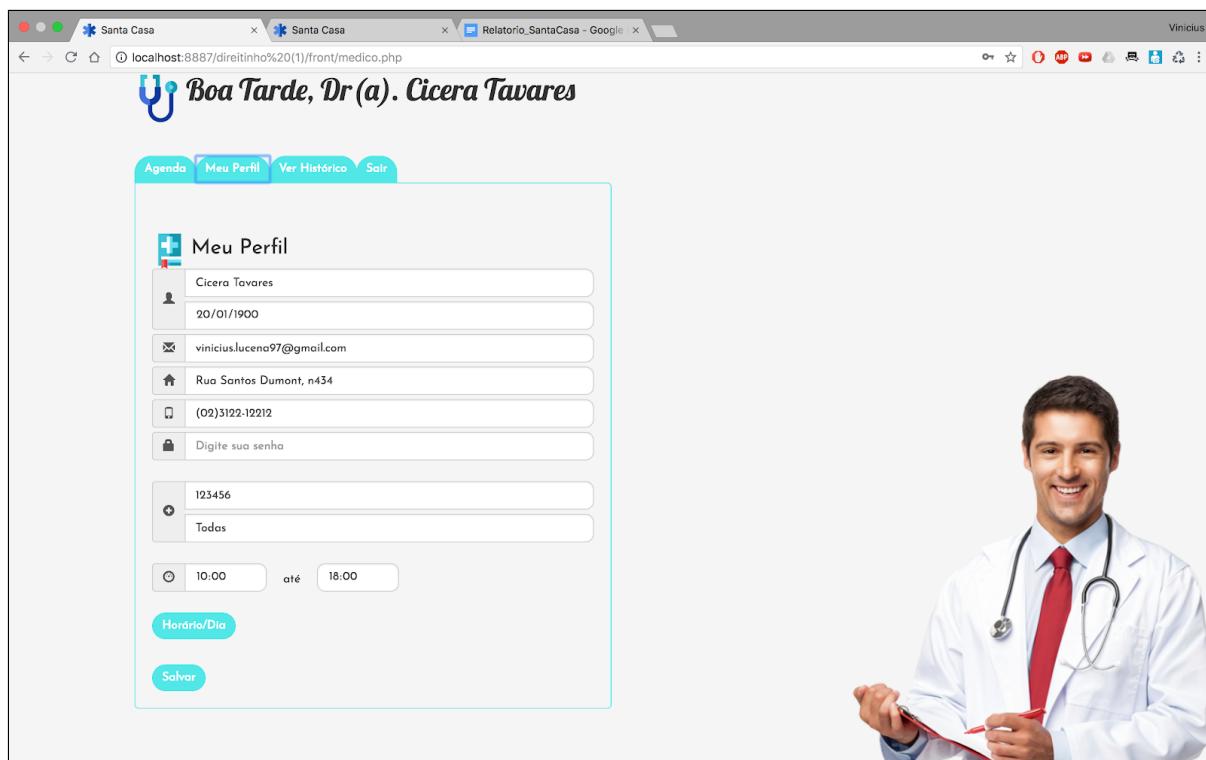


Figura 12. Médico alterando seu cadastro.

• Visualizar sua agenda de consultas

Quando o médico desejar ver sua agenda de consultas, basta clicar no botão "Agenda" que os seguintes passos são executados:

- 1) É feita uma requisição HTTP ao servidor.
- 2) O servidor pega o nome do médico com o superglobal `$_SESSION`.
- 3) O servidor acessa o arquivo `Consultas.xml` buscando as consultas que combinam com o nome do médico.
- 4) Nesta etapa, como temos a lista de consultas do médico, geramos a tabela dentro do loop que percorre essa lista e, dentro do loop, geramos as linhas da tabela.

A figura 13 mostra como ficou essa função para o usuário. Para mais detalhes, verificar o script `verAgendaCompleta.php`, na pasta `server`.



Figura 13. Tela do médico vendo sua agenda de consultas.

• Preencher observações e receita nas suas consultas

É importante que o médico receite um medicamento ou faça alguma observação para cada consulta. Fizemos tal função pensando em facilitar a vida do médico. Quando o médico visualiza sua agenda de consultas, é mostrado a opção de "Receitar" e "Diagnosticar", para cada consulta.

Na função do médico visualizar sua agenda de consulta já explicamos como foi que geramos a tabela de consultas e os botões de receber e diagnosticar. Ao clicar em cada botão, é mostrado um texto para que o médico adicione uma nova receita ou para que o médico realize o diagnóstico.

O campo para adicionar o texto fica em uma linha na tabela, contudo essa linha fica oculta (classe collapse) e só aparece se clicarmos no botão.

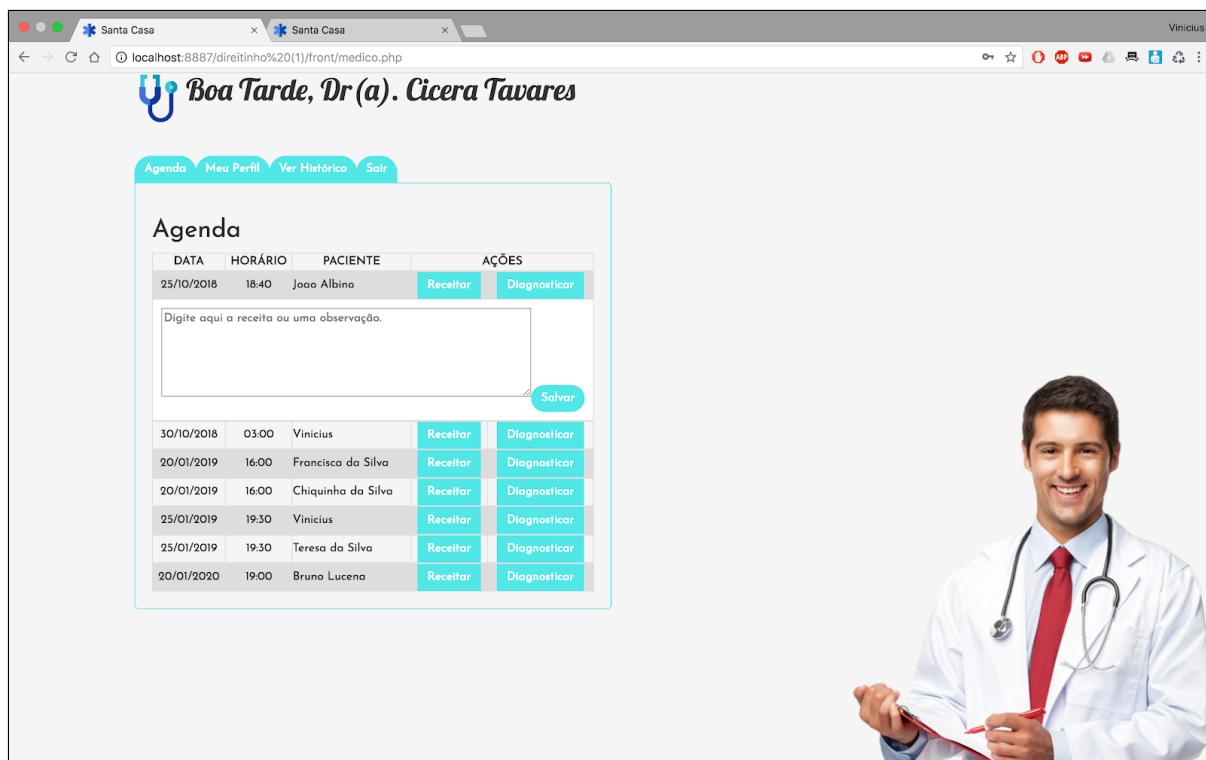


Figura 14. Tela do médico receitando um paciente.

- **Visualizar histórico de consultas dos pacientes**

É importante que o médico consiga visualizar o histórico completo de consultas de cada paciente. O histórico completo consiste, além das informações básicas sobre a consulta, as quais os pacientes e atendentes têm acesso, como também à receitas e diagnósticos realizados por outros médicos.

Entendemos que a receita e o diagnóstico são informações confidenciais, que somente os médicos podem acessar.

A figura 15 mostra a tela de visualização das consultas dos pacientes.

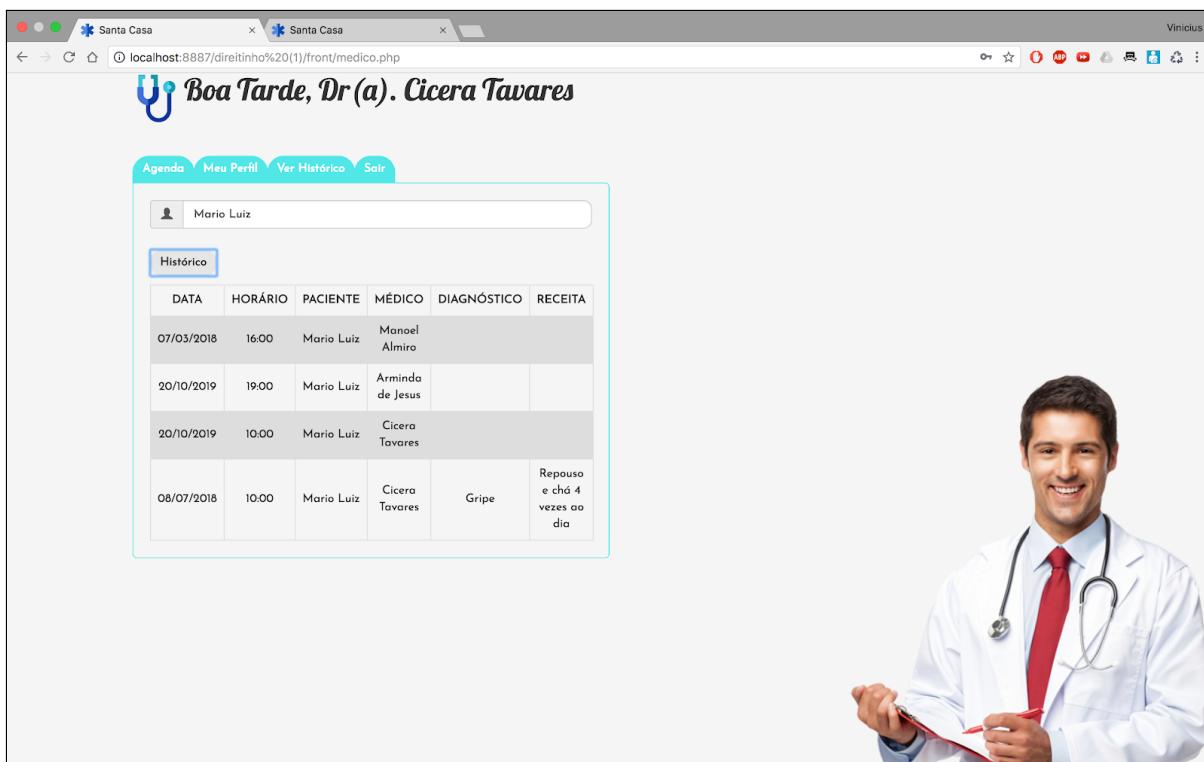


Figura 15. Tela de visualização de consultas dos pacientes pelo médico.

3.3. FUNÇÕES DO PACIENTE

- Realizar seu cadastro

Para a realização do cadastro do paciente, foi adicionado na tela de login um link para o mesmo se cadastrar, como é possível ver na imagem 16 da sessão 3.4 que mostra a tela de login. Essa função não existia anteriormente, pois apenas a atendente cadastrava, tanto médicos, quanto pacientes. Clicando nesse link, o paciente é redirecionado à página de cadastro, que pode ser vista abaixo. Nesta página são requisitados dados do paciente para que o mesmo consiga se cadastrar, a partir daí, ele pode inserir os dados e se cadastrar e clicar no botão “Cadastrar”, onde, caso o cadastro ocorra com sucesso ele será avisado e redirecionado automaticamente para a página de login, ou clicar na seta localizada no canto superior esquerdo e voltar para a tela de login.

The screenshot shows a web browser window titled 'Santa Casa' with the URL 'localhost:8080/front/registro.html'. The page has a light blue header with a logo featuring a cross and a red heart. Below the header is a teal-colored form titled 'Cadastro'. The form contains fields for personal information: Nome (Name), Data de Nascimento (Birth Date) in mm/dd/yyyy format, CPF, E-mail, Endereço (Address), Telefone (Phone), and Senha (Password). A 'Cadastrar' (Register) button is at the bottom right of the form.

Figura 16. Tela de cadastro

● Alterar seu cadastro

Para a alteração do cadastro, o paciente dispõe, em sua própria página, após realizar o login, da função de ver seu perfil e fazer alterações no mesmo, salvando após realizar essas alterações. Para não causar nenhum problema de alteração sem intenção, o perfil do paciente é resgatado da base de dados (arquivo XML), utilizando a SESSION criada a partir do login. E então, o salvamento é feito como uma alteração no arquivo XML de pacientes.

The screenshot shows a web browser window titled 'Santa Casa' with the URL 'localhost:8080/front/paciente.php'. The page features a red blood drop icon and the greeting 'Boa Tarde, Sr(a). Mario Luiz'. At the top, there is a navigation bar with tabs: 'Perfil' (selected), 'Agendar', 'Minhas Consultas', and 'Sair'. Below the navigation is a teal-colored form titled 'Meu Perfil'. It displays the patient's information: Nome (Nome: Mario Luiz, Data de Nascimento: 12/10/1963), E-mail (marioluiz@bol.com.br), Endereço (Rua Matildes Amaral, 527), Telefone (53)98100-0875, and Senha (Digitte sua senha). A 'Salvar' (Save) button is located at the bottom left of the form.

Figura 17. Perfil do paciente

Como é possível observar a imagem acima, os campos do perfil do paciente são preenchidos assim que o botão perfil é clicado, isso acontece através da SESSION em PHP inserida no *value* de cada *input*. Isso é feito no código da maneira que podemos ver na figura abaixo.

```
value="<?php echo $_SESSION['nome']; ?>"
```

Figura 18. *Input value* por sessão

- **Realizar agendamento de consultas**

O agendamento de consultas por parte do paciente trabalha com a ideia de uma verificação da atendente. Assim, o paciente agenda a sua consulta e a mesma precisa ser aprovada pela atendente. Enquanto a consulta não é aprovada, ela é mantida em um arquivo XML separado das consultas gerais, caso seja aprovada, ela é movida para o XML de consultas, caso contrário, ela é somente excluída do XML de consultas a serem aprovadas.

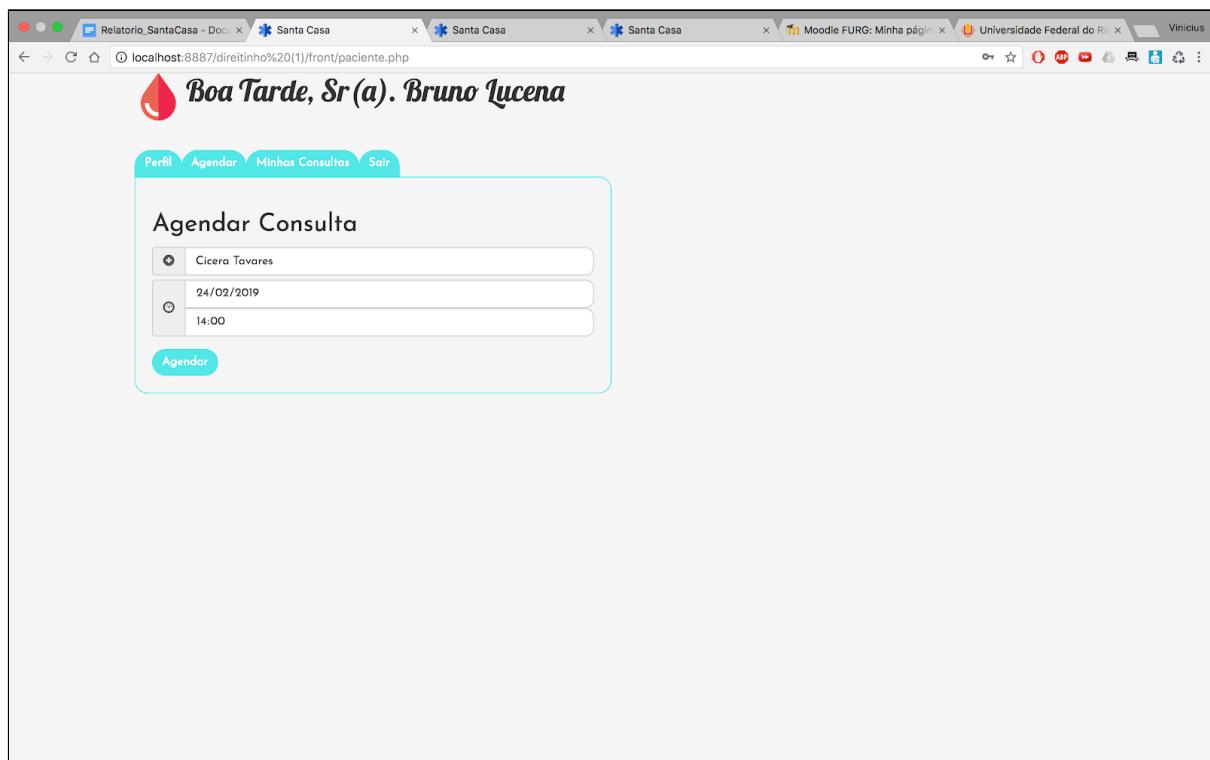


Figura 19. Paciente realizando um agendamento de uma consulta.

- **Visualizar seu histórico de consultas**

A visualização do histórico de consultas por parte do paciente se dá da mesma forma como um médico consegue visualizar o histórico de seus pacientes, porém, de maneira simplificada. Essa parte, partindo do ponto de vista de um paciente, e considerando a nossa interpretação, seria para o paciente ter noção das consultas que ele já fez, quando foram e com qual médico elas foram realizadas.

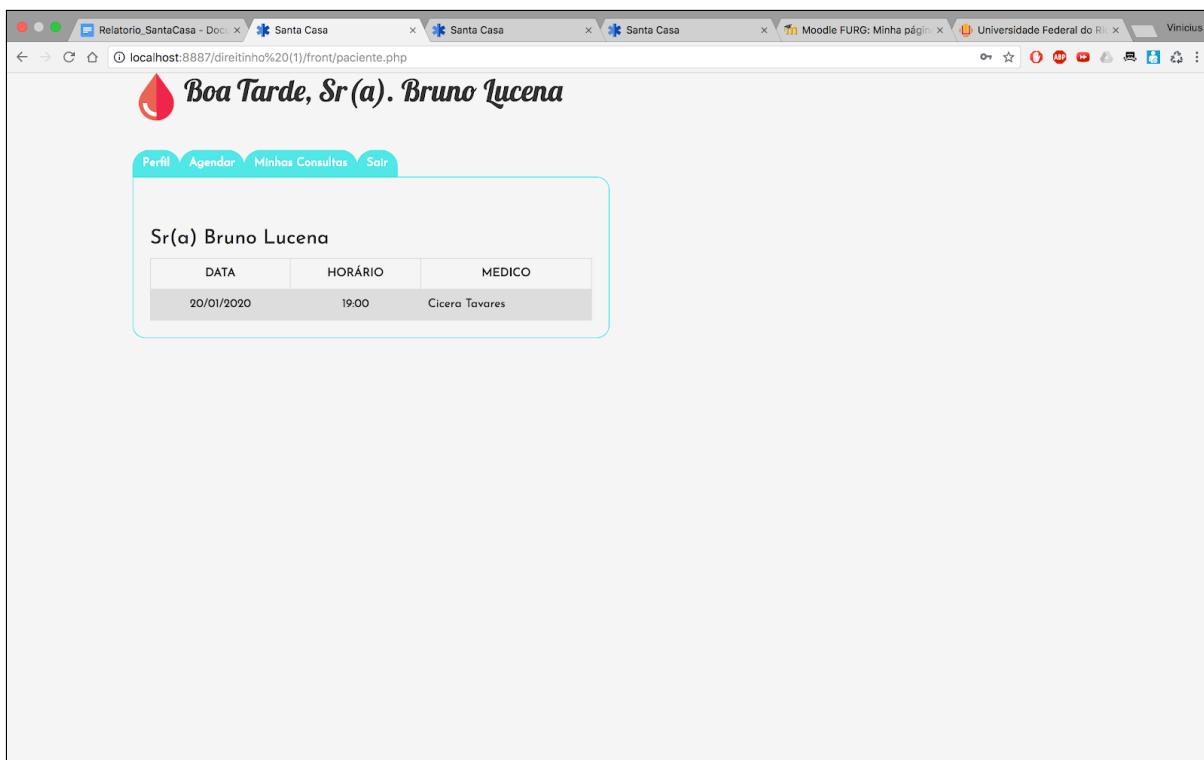


Figura 20. Paciente visualizando suas consultas.

3.4. FUNCIONALIDADES EXTRAS

Nesta parte do trabalho iremos apresentar algumas funcionalidades que incluímos ao decorrer do desenvolvimento do sistema, a fim de melhorar a experiência de usuário.

- **Página personalizada para cada médico.**

Como se pode notar pela figura 21, a página do médico possui uma mensagem para cada médico, de acordo com o horário do computador e de acordo com o nome de cada um. Para saber o horário atual do computador, foi utilizado a função *date()* do PHP.



Figura 21. Página personalizada para cada médico.

A figura 22 mostra o código PHP embutido na página do médico. Podemos notar que o nome do médico foi acessado pelo superglobal Session. A função é extremamente simples, basta verificar em qual intervalo o horário atual da máquina está definido. O comando *date()* com o parâmetro 'H' retorna o horário atual com o timezone UTF-0. Por isso que o intervalo de horários é um pouco diferente do nosso.

```

<?php
    echo "<h1>";
    echo ((date('H') >= 3 && (date('H')) < 9) ? "Boa Madrugada, " : "");
    echo ((date('H') >= 9 && (date('H')) < 15) ? "Bom Dia, " : "");
    echo ((date('H') >= 15 && (date('H')) < 21) ? "Boa Tarde, " : "");
    echo ((date('H') >= 21 || (date('H')) < 3) ? "Boa Noite, " : "");
    echo "Dr(a). " . $_SESSION['nome'] . "</h1><br><br>";
?

```

Figura 22. Mensagem personalizada para cada médico.

• Tela de login

A figura 23 mostra a tela de login do sistema. A pessoa pode logar como Paciente, Médico ou Atendente. Os passos executados para realizar o login são ilustrados pela figura 25. Também é possível que o paciente se cadastre no sistema através do link “Cadastrar-se”, cuja funcionalidade foi explicada anteriormente na seção 3.3.

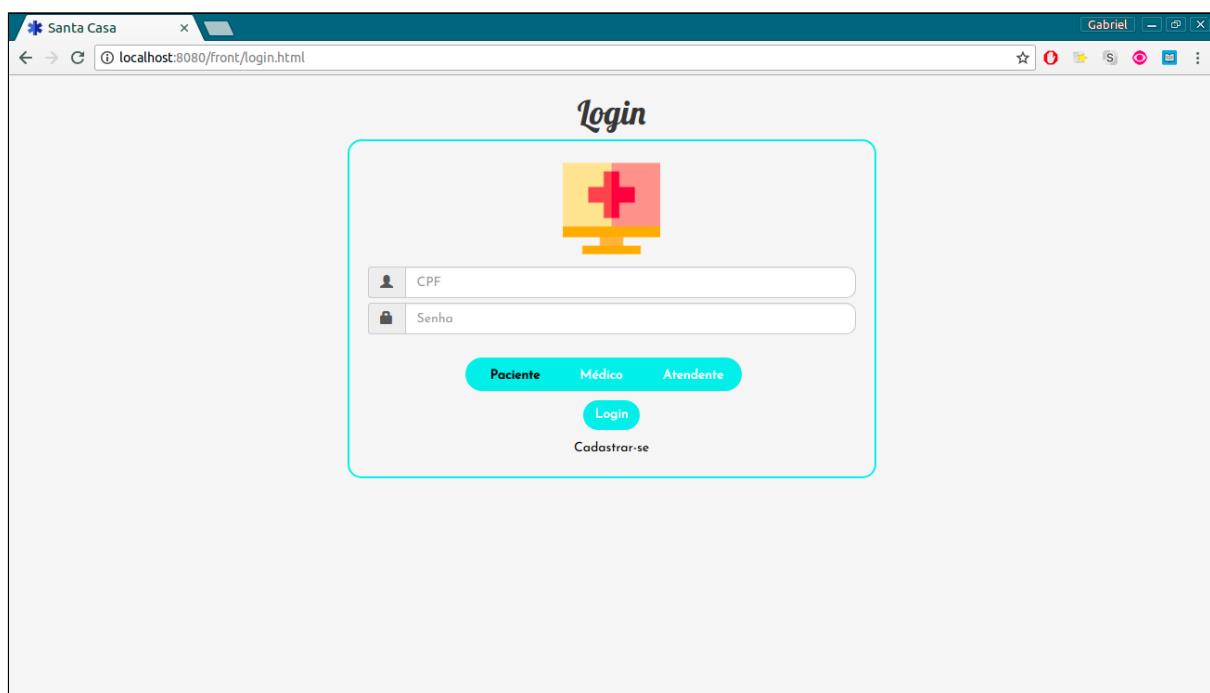


Figura 23. Tela de login.

• Sair

Como utilizamos *SESSION*, achamos necessário incluir uma pequena função extra chamada *sair()*. Ela existe nos 3 tipos de login (paciente, médico e atendente). Essa função foi implementada através de um botão inserido nas páginas dos citados acima, chamando através do Ajax, o arquivo contendo o seguinte conteúdo (imagem abaixo).

```

<?php
    require_once "../front/atendente.php";
    if (session_status() != PHP_SESSION_NONE) {
        session_unset();
        session_destroy();
    }

    exit();
?>

```

Figura 24. Sair

A figura 25 mostra os passos executados para realizar um login no sistema. Nesta imagem, dividimos o sistema em dois componentes: *user-side* e *server-side*.

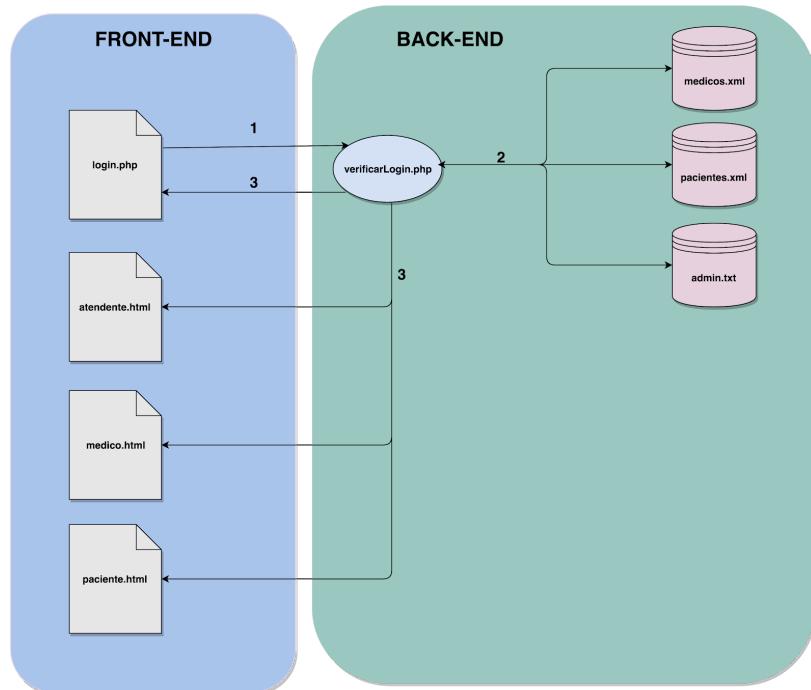


Figura 25. Passos para realizar login no sistema.

Passos para realizar login:

- 1) Usuário preenche o formulário informando seu CPF e sua senha. Os dados são enviados para o servidor e o script *verificarLogin.php* é executado.
- 2) É identificado que está tentando acessar o sistema (paciente, médico ou atendente). É realizado uma consulta no banco de dados para verificar se o CPF e a senha estão cadastrados.
- 3) Dependendo do resultado do passo 2, em caso de êxito, o servidor redireciona a respectiva página do usuário. Caso as informações não constem no banco, o servidor retorna para a página inicial.

4. CONCLUSÃO

Durante o desenvolvimento do projeto houveram diversos desafios, especialmente porque os três integrantes envolvidos nunca desenvolveram aplicações web anteriormente, dessa maneira tivemos que aprender (quase) tudo do zero.

De maneira geral, ficamos plenamente satisfeitos com o sistema desenvolvido, cremos que o sistema esteja em um ponto no qual poderíamos disponibilizá-lo para ser usado na vida real. Claro que sempre há o que melhorar, como salientamos acima, porém, o sistema está digno de um sistema simples e utilizável, até mais do que alguns que vemos em alguns sites.

Ressaltamos que caso houvesse mais tempo e necessidade, o sistema está apto a receber aperfeiçoamentos e funções adicionais.

Em relação à este relatório, tentamos fazer o mais explicativo possível para o seu entendimento, por isso contamos com imagens, diagramas e esquemas para tornar a visualização do sistema como um todo, assim como sua criação, mais simples.

4.1. OVERVIEW DO SISTEMA

PONTOS POSITIVOS

- O paciente pode se cadastrar, agendar consultas e ver seu histórico.
- Para adicionar novos administradores do sistema, basta abrir o arquivo admin.txt e adicionar um novo CPF e uma nova senha. (Caso deseja incluir um novo computador na clínica, por exemplo).
- Extensibilidade. Se for interessante guardar informações pessoais da atendente (como nome, endereço, telefone, etc) basta que a classe Atendente herde os métodos e atributos da classe AbsPessoa.
- Como há diversos scripts separados, há um fraco acoplamento entre os módulos do sistema.

PONTOS NEGATIVOS

- Não nos preocupamos com o controle de concorrência. Pode acontecer de dois processos concorrentes acessarem o mesmo recurso ao mesmo tempo.
Contraponto: Podemos corrigir isso com a utilização de primitivas do sistema operacional (*mutexes* e *semáforos*) para realizar o controle.
- A opção do atendente alterar uma consulta ficou trabalhosa para o atendente, visto que o mesmo terá que digitar os dados da consulta antiga e os dados da nova para realizar a alteração.
Contraponto: Poderíamos inserir um botão de alterar consulta ao lado de cada consulta (similar ao que fizemos na função do médico receitar um paciente) .

Por fim, como próximos passos, implementaríamos justamente o que ficou como ponto negativo entre outras coisas, para tornar o sistema o mais robusto possível.