Relatório Intermediário
CES-22 - Programação Orientada a Objeto

Adriano Soares, Matheus Vidal e Pedro Alves

Maio, 2019

Turma COMP 21

Prof.º Yano

Instituto Tecnológico de Aeronáutica (ITA)

São José dos Campos, São Paulo, Brasil.

{sadrianorod, matheusvidaldemenezes, alvesouza.pedro97}@gmail.com

git:vidalmatheus

I. Introdução

O grupo Adriano Soares Rodrigues, Matheus Vidal de Menezes e Pedro Alves de Souza Neto decidiu construir uma *web application* de marcação de consultas para a Divisão de Saúde (DS) do Departamento de Ciência e Tecnologia Aeroespacial (DCTA).

A principal motivação foi o fato da DS não possuir um sistema automatizado de marcação de consultas, fazendo com que um morador militar ou aluno do ITA tenha que se dirigir até a DS apenas para marcar uma consulta.

Apesar do principal objetivo ser a marcação de consulta *online*, foram vislumbradas outras funcionalidades necessárias ou para o correto funcionamento, ou para melhor usabilidade para paciente e médico. Inclusive funcionalidades que podem ser aprimoradas num sistema maior no futuro.

II. Análise de Requisitos

Primeiramente, tivemos dificuldades de termos uma reunião presencial com os responsáveis adequados na DS, devido a desencontro de horários. Assim, o sistema teve seu desenvolvimento iniciado com base na experiência de vários alunos do ITA que passaram informações relevantes sobre o processo de marcação de consultas na Divisão de Saúde.

Almejando sempre boas práticas de Engenharia de Software, incorporamos diagramas da UML para nos ajudar a modelar tal aplicação real. Tais diagramas foram sofrendo modificações ao longo de mais informações consistentes que se obtinham. **Por isso, os diagramas presentes neste relatório estão um pouco diferentes daqueles apresentados em aula.** Segue o Diagrama de Caso de Uso.

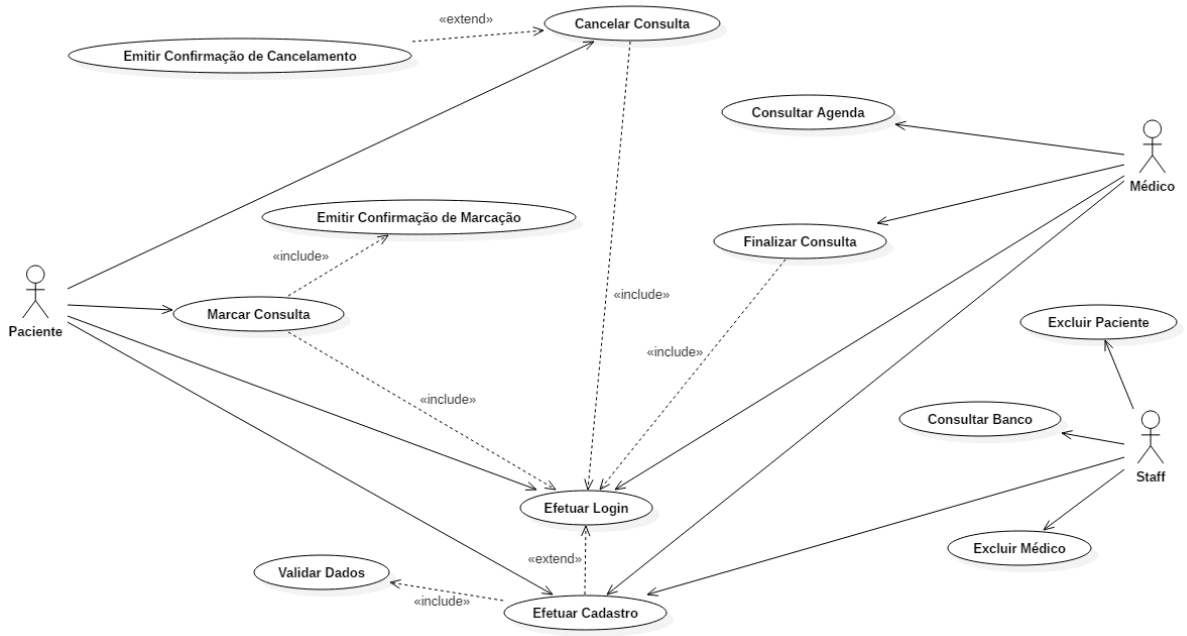


Figura 1. Diagrama de Casos de Uso do sistema.

Para entender melhor como se dará o processo, segue os principais Fluxos de Evento, tanto do paciente, quanto do médico no sistema:

<Marcar Consulta>
Atores: Paciente e Funcionário
Pré-Condição: O paciente não tem cadastro ainda.
Fluxo de Eventos Principal:

1. O paciente seleciona Cadastro.
2. O paciente fornece dados: CPF, senha, SARAM, nome, data de nascimento, sexo,endereço, telefone, e-mail, posto/graduação militar ou caso especial (aluno do ITA).
3. O funcionário valida o número SARAM, confirmando o cadastro do paciente.
4. O paciente recebe confirmação por e-mail que o cadastro foi confirmado.

<Marcar Consulta>

Atores: Paciente

Pré-Condição: Paciente já cadastrado.

Fluxo de Eventos Principal:

1. O paciente efetua *login* com CPF e senha.
2. O paciente seleciona Marcar Consulta.
3. O paciente escolhe o dia da semana que quer marcar consulta.
4. O paciente escolhe a especialidade do médico.
5. O paciente escolhe o médico.
6. O paciente escolhe o horário da consulta de 30 min, dentre os 15 slots diários.
7. O paciente confirma seus dados.
8. O paciente seleciona Enviar.
9. O sistema exibe mensagem de operação foi concluída.
10. O sistema envia, uma confirmação por e-mail.

<Cancelar Consulta>

Atores: Paciente, Médico e Funcionário

Pré-Condição: O paciente deve estar com consulta marcada e ter efetuado *login*.

Fluxo de Eventos Principal:

1. O sistema exibe as consultas marcadas.
2. O paciente seleciona Cancelar Consulta.

<Consultar Agenda>

Atores: Médico

Pré-Condição: O médico deve estar cadastrado.

Fluxo de Eventos Principal:

1. O sistema exibe a agenda semanal
2. O médico seleciona o dia desejado.
3. O médico consulta aos slots de consultas marcadas ou livres do dia.

<Finalizar Consulta>

Atores: Médico

Pré-Condição: O médico deve estar cadastrado e ter terminado uma consulta.

Fluxo de Eventos Principal:

1. O sistema exibe os slots de consultas marcas no dia.
2. O médico seleciona Finalizar Consulta na consulta que já foi terminada.
3. O sistema retira o slot da tela.
4. O sistema torna o horário disponível de novo para marcação de consulta.

<Efetuar Cadastro>

Atores: Médico e Funcionário

Pré-Condição: O médico não tem cadastro ainda.

Fluxo de Eventos Principal:

1. O médico seleciona Cadastro.
2. O médico fornece dados: CPF, senha, SARAM, nome, especialidade, posto/graduação militar.
3. O funcionário valida o número SARAM, confirmando o cadastro do médico.
4. O médico recebe confirmação por e-mail que o cadastro foi confirmado.

<Consultar Banco>

Atores: Funcionário

Fluxo de Eventos Principal:

1. O funcionário faz *login* como administrador.
2. O sistema exibe todas as consultas marcadas da semana.

<Excluir Médico>

Atores: Funcionário

Fluxo de Eventos Principal:

1. O funcionário faz *login* como administrador.
2. O funcionário seleciona Excluir Médico.
3. O sistema exibe todos os médicos.
4. O funcionário exclui o médico desejado.

<Excluir Paciente>

Atores: Funcionário

Fluxo de Eventos Principal:

1. O funcionário faz *login* como administrador.
2. O funcionário seleciona Excluir Paciente.
3. O sistema exibe todos os pacientes.
4. O funcionário exclui o paciente desejado.

III. Divisão de Trabalho

A equipe foi subdividida em três áreas principais: *Front-End*, *Back-End* (Banco de Dados) e *Back-end* (Integração), com seus respectivos líderes, Adriano Soares, Matheus Vidal e Pedro Alves. O motivo disso foi o fato de dar mais autonomia para cada integrante, tal como uma situação real de trabalho. Essa divisão, porém, não é excludente, a fim de que cada integrante consiga dialogar e até mesmo ajudar na codificação de cada parte.

IV. Tecnologias Utilizadas

Front-end: HTML5, CSS3, Bootstrap
Back-end (Banco de Dados): PostgreSQL, pgAdmin 4
Back-end (Integração): Flask (python)
Diagramas: StarUML
IDE: vscode

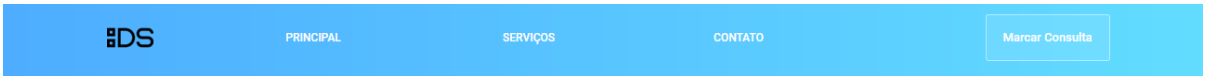
V. *Front-end*: Adriano Soares

Diante da responsabilidade de toda a interação usuário com o website, foi necessário o estudo das principais ferramentas de *Front-End*: HTML5, CSS3, *Bootstrap*.

É importante salientar que o *Bootstrap* é um *framework* que agiliza a criação do website, uma vez que já apresenta padrões de materiais para utilização, sendo de extrema importância para a criação do DS.com uma vez que o tempo de projeto é curto, então necessitou-se otimizar o tempo utilizando padrões já disponibilizados.

Em paralelo, estudei React como alternativa final para compartimentar o código e deixá-lo mais organizado, já que a ideia do React é simplificar o código, através da divisão em blocos de código, deixando muito mais subdividido e organizado do que um file HTML.

Assim, após semanas de estudo para poder entender e aprender a utilizar as ferramentas de *Front-End*, a página principal do site foi criada e já inserida no projeto. Logo abaixo, temos algumas Figuras do código criado e compartilhado no repositório do github.

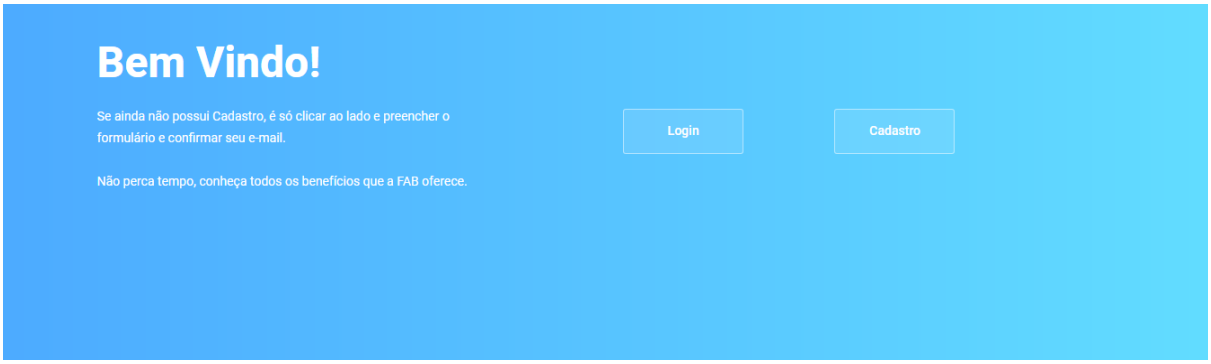



```
<!--================Header da Area de Menu =================-->
    <body>
        <header class="header_area">
            <div class="main_menu">
                <nav class="navbar navbar-expand-lg navbar-light">
                    <div class="container box_1620">
                        <a class="navbar-brand logo_h" href="index.html">   <img class="img-fluid" src="img/DS.png" alt=""></a>

                        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
                            aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                            <span class="icon-bar"></span>
                            <span class="icon-bar"></span>
                            <span class="icon-bar"></span>
                        </button>
                        <div class="collapse navbar-collapse offset" id="navbarSupportedContent">
                            <ul class="nav navbar-nav menu_nav justify-content-center">
                                <li class="nav-item active"><a class="nav-link" href="index.html">Principal</a></li>       &e
                                <li class="nav-item"><a class="nav-link" href="#">Serviços</a>         
                                <li class="nav-item"><a class="nav-link" href="#">Contato</a></li>         
                            </ul>
                            <ul class="nav navbar-nav navbar-right" >
                                <li class="nav-item"><a href="#" class="tickets_btn">Marcar Consulta</a></li>
                            </ul>
                        </div>
                    </div>
                </nav>
            </div>
        </header>

```

Figura 2. Código do header da barra de menu.




```
<!--================Area Azul Principal da Página =================-->
        <section class="home_banner_area">
            <div class="banner_inner">
                <div class="container">
                    <div class="row">
                        <div class="col-lg-5">
                            <div class="banner_content">
                                <h2>Bem Vindo!</h2>
                                <p>Se ainda não possui Cadastro, é só clicar ao lado e preencher o formulário e confirmar seu e-mail.
                                    <br/><br/>Não perca tempo, conheça todos os benefícios que a FAB oferece.</p>
                            </div>
                        </div>
                        <div class="col-lg-7">
                            <div class="home_left_img">
                                      <a class="banner_btn" href="#">Login</a>
                                       <a class="banner_btn" href="#">Cadastro</a>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </section>
```

Figura 3. Código da parte principal do site, logo abaixo da barra de menu.


```
<!--================Area de Ícones ================-->
<section class="work_area p_120">
    <div class="container">
        <div class="work_inner row">
            <div class="col-lg-4">
                <div class="work_item">
                    <i class="lnr lnr-star"></i>
                    <a href="#"><h4>Personalizado</h4></a>
                    <p>Sempre focamos em garantir a qualidade que nosso militares merecem com atendimento personalizado e longe de filas de es
                </div>
            </div>
            <div class="col-lg-4">
                <div class="work_item">
                    <i class="lnr lnr-heart-pulse"></i>
                    <a href="#"><h4>Cuide-se</h4></a>
                    <p>Focamos na prevenção das doenças facilitando seus agendamentos de consultas pelo nosso website.</p>
                </div>
            </div>
            <div class="col-lg-4">
                <div class="work_item">
                    <i class="lnr lnr-thumbs-up"></i>
                    <a href="#"><h4>Curta</h4></a>
                    <p>Aqui temos os melhores profissionais para seu atendimento, marque sua consulta e curta a vida com saúde.</p>
                </div>
            </div>
        </div>
    </div>
</section>
<!--================Fim da Area de Ícones ================-->
```

Figura 4. Código da parte descritiva com os tópicos: “personalizado”, “cuide-se” e “curta”.

VI. *Back-end*: Matheus Vidal

Como uma aplicação real, trouxe ao grupo conhecimentos e boas práticas de Engenharia de Software, utilizando a UML. Fiquei responsável pelo Modelo Conceitual do projeto, Análise de Requisitos (Diagramas de Casos de Uso e Fluxo de Eventos) e Modelagem dos Dados (criação do Banco de Dados).

Quanto a Análise de Requisitos, está bem descrita no tópico II.

A fim de modelar o sistema em mais baixo nível, a partir do Diagrama de Casos de Uso, foi montado o Diagrama de Entidade-Relacionamento do sistema:

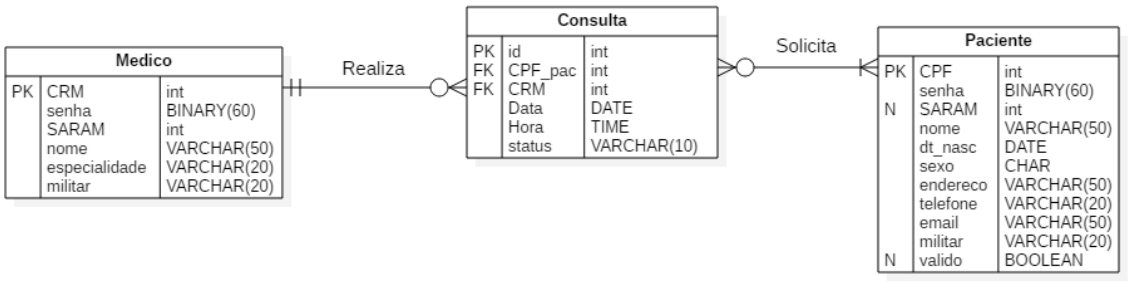


Figura 5. Diagrama de Entidade-Relacionamento do sistema.

Note o uso correto das cardinalidades que originaram a tabela Consulta com as chaves estrangeiras da tabela Médico e Paciente:

a) “1 médico realiza nenhuma ou muitas consultas. ”
b) “1 consulta é realizada por apenas 1 médico. ”
c) “1 paciente solicita nenhuma ou muitas consultas. ”
d) “1 consulta é solicitada por, no mínimo, 1 e, no máximo, muitos pacientes. ”

Assim, foi gerado o script em .sql, para criar o Banco de Dados “ds”. Um trecho do script é mostrado abaixo:

```
-- Database: ds
-- DROP DATABASE ds;
CREATE DATABASE ds
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'Portuguese_Brazil.1252'
    LC_CTYPE = 'Portuguese_Brazil.1252'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;
DROP TABLE IF EXISTS Medico CASCADE;
DROP TABLE IF EXISTS Paciente CASCADE;
DROP TABLE IF EXISTS Consulta CASCADE;
-- ------------------------------------------------
-- Table `ds`.`medico`
-- ------------------------------------------------
DROP TABLE IF EXISTS "ds".medico ;
CREATE TABLE Medico (
    CRM bigint NOT NULL,
     senha bytea NOT NULL,
     SARAM int,
    Nome VARCHAR(50) NOT NULL,
    Especialidade VARCHAR(20) NOT NULL,
    militar VARCHAR(20) NOT NULL,
    PRIMARY KEY (CRM)
);
```

Exemplo de Consulta ao Banco de consultas marcadas:

```
select p.Nome, k.Nome, data, hora
from
    (medico as m inner join consulta as c on (m.CRM=c.CRM) ) as k
    inner join paciente as p on (k.CPF_pac=p.CPF)
where status = 'marcado'
```

VII. *Back-end*: Pedro Alves

Para a programação do servidor, será utilizado a linguagem de programação Python, utilizando o *framework* Flask.

Eu contribuí para o projeto dos bancos de dados, como quais dados o banco deve ter, casos de uso e a medida de segurança com o armazenamento de senhas, usando criptografia, especificamente o módulo "flask_bcrypt" que possui uma função de criptografia ("generate_password_hash") que não possui inversa, ou seja, a senha na tentativa de acesso de *login*, será convertida pela mesma, para ser comparado com a senha do banco de dados.

Desenhei o diagrama de classes (ainda deve receber alterações), o principal desse diagrama será evitar as possibilidades de dados serem capturados pelo *Front-End*, pois acontece casos de *hackers* obter o *hash* da senha do usuário, sem precisar invadir o servidor, pois a comparação é feita no front.

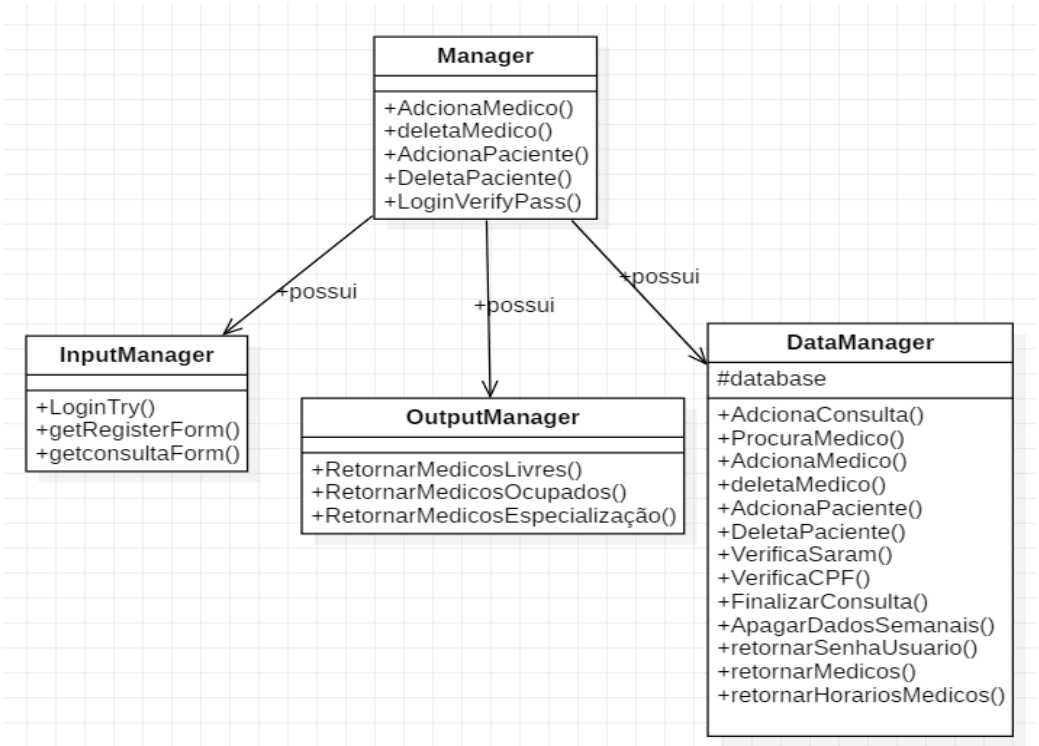


Figura 6. Diagrama de classes.

Da Figura 6, o que sofrerá menos alteração será a classe “DataManager”, ela fará a ligação do Back com o banco de dados. A maioria das funcionalidades serão abstratas (não serão referentes à uma instância da classe). As classes “InputManager”, “OutputManager” provavelmente possuirão outras classes referente ao seus respectivos dados e formulários. O “Manager” será o responsável pela interação entre o “DataManager” e as outras classes, de certo modo ela será o que controla tudo.