INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

Campus São João da Boa Vista

Trabalho Final de Curso

4º ano – Curso Técnico em Informática

Prof. Breno Lisi Romano e Prof. Luiz Angelo Valota Francisco

DESENVOLVIMENTO DAS ITERAÇÕES DO MÓDULO 7

Aluno: Lucas Henrique Bensi Gutierres

Prontuário: BV1520032

São João da Boa Vista – SP

2019

Resumo

Este documento delata o processo de desenvolvimento do módulo 07 do projeto gerações, introduzindo conceitos sobre a programação, descrevendo o levantamento de requisitos, desenvolvimento dos casos de uso. Mostrando dificuldades que surgiram ao longo do caminho e habilidades desenvolvidas durante a elaboração do projeto gerações.

Sumário

1 Introdução

1.1 Contextualização/Motivação

São João da Boa Vista é um município brasileiro do interior do Estado de São Paulo. Segundo a estimativa do IBGE de 2017, a cidade tem uma população de 90.089 habitantes e seu IDH é de 0,797, considerado o 28º melhor do estado. O município superou outras 347 localidades analisadas no Índice de Desenvolvimento Urbano para Longevidade, realizado em parceria com o Instituto de Longevidade Mongeral Aegon. É considerada a cidade que oferece a melhor qualidade de vida para os idosos segundo uma pesquisa desenvolvida pela Fundação Getúlio Vargas. No estudo foram avaliadas 398 cidades brasileiras entre 50 mil e 100 mil habitantes. Prova disso é o Índice de Envelhecimento da cidade, que em 2018, totaliza 111,73% e a porcentagem da população com 60 anos ou mais que é 18,04%. O estudo considera o acesso aos serviços públicos e privados de saúde, situação financeira, educação, trabalho e cultura. Também analisa o número de idosos com internet fixa [1][2].

Em 1910 surgiu a Escola de Aprendizes e Artífices de São Paulo, destinada às camadas mais desfavorecidas, originando o Centro Federal de Educação Tecnológica de São Paulo – CEFET, que foram autorizadas a funcionar em São João da Boa Vista no primeiro bimestre de 2007 [3].

Nos tempos atuais esse centro de ensino é conhecido como o Instituto Federal de Educação, Ciência e Tecnologia, não só fornecendo cursos técnicos, porém também cursos de licenciatura, bacharelado, pós-graduação e engenharia. O curso técnico integrado em informática, tem como objetivo de ensino desenvolver programas em linguagens comercialmente utilizadas como ferramentas de apoio à gestão administrativa de uma organização, utilizando redes de computadores. E competências: interpretar e analisar dados, colaborar no levantamento de informações junto aos usuários participando da elaboração de anteprojetos de sistemas, desenvolver e manter programas, participar da implantação e manutenção de sistemas, desenvolver testes e simulações, gerando os arquivos necessários; analisar os resultados dos programas, identificando desvios e realizando correções. O técnico, ainda, poderá utilizar softwares para interpretar dados, efetuar levantamento da melhor linguagem de programação para atender os usuários e registrar informações sobre o desenvolvimento dos programas [4]. Durante o ano de conclusão desse curso, na matéria de pratica de desenvolvimento de sistemas que visa a demonstração prática de Gerenciamento de projetos de Software, Processos de Desenvolvimento de Software Tradicionais e Metodologias Ágeis. Definição de Cronogramas para controle do Desenvolvimento. Modelagem de Sistemas (Linguagem de Modelagem Unificada – UML). Investigação de Técnicas de Estimativas

de Desenvolvimento de Software, Arquitetura MVC (Model View Controller) e Padrões de Projeto (Design Patterns). Demonstração prática das técnicas de desenvolvimento de sistemas de software em um Estudo de Caso, abordando as seguintes atividades: Levantamento de Requisitos, Modelagem, Banco de Dados, Implementação, Testes e Documentação do Projeto Interdisciplinar [5], é proposto aos alunos o desenvolvimento de um sistema administrado pelo professor Breno Lisi Romano. Foi proposto a criação de um software para cuidado de idosos, possuindo como publico alvo instituições de longa permanência. O sistema foi dividido em 3 subsistemas para melhor organização no desenvolvimento do projeto, cada subsistema sendo dividido novamente e totalizando em 9 módulos. O modulo 7 é responsável pelo gerenciamento das atividades físicas e recreativas dos idosos, registros de consultas físicas e o gerenciamento do plano mensal de atividades.

Figura 1 - Perspectiva do modulo 07

1.2 Objetivo Geral

O objetivo geral desse trabalho final de curso compreende a tudo sobre o desenvolvimento das iterações do modulo 07, desde os protótipos iniciais, as iterações finais do projeto, apresentando as divisões as adaptações e as dificuldades surgidas durante seu desenvolvimento.

1.3 Objetivos Específicos

Os objetivos específicos apresentados neste trabalho são:

- A confecção dos protótipos a partir dos requisitos denotados;
- Desenvolvimento dividido em iterações;
- Integração das iterações ao banco de dados;
- A refatoração dos códigos apresentados nos protótipos;
- Dificuldades e Adaptações durante o desenvolvimento;

2 Desenvolvimento

2.1 Levantamento Bibliográfico

2.1.1 Noções básicas sobre linguagens de programação

As linguagens de programação são amplamente conhecidas e utilizadas por profissionais da área de computação. Para que linguagens de programação possam ser utilizadas, elas precisam ter sido projetadas e implementadas. Alguém precisou descobrir do que os profissionais de computação precisam e o que eles esperam de uma linguagem de programação. Isso inclui estudos sobre o que é essencial nas linguagens para que os problemas possam ser resolvidos, uma tarefa que também é realizada pelos projetistas de linguagens. Ainda mais do que isso: o que se deseja de uma linguagem de programação precisa ser passível de implementação. Todas essas considerações devem deixar claro que o uso de uma linguagem não dota o usuário do conhecimento necessário para projetar e implementar uma linguagem de programação.

Podemos iniciar pela caracterização do que é um programa, é uma entidade que se manifesta como um documento, especificando uma sequencia de operações a serem executadas e durante a sua execução, efetivamente levando a cabo as operações especificadas.

Um programa, portanto é uma maquina abstrata, manipulando e produzindo dados. Ele é ao mesmo tempo a descrição de uma máquina (o documento) e a própria maquina (quando está em execução).

Uma linguagem de programação determina os recursos disponíveis e sua forma de utilização para construir maquinas, de tal forma que elas possam ser simuladas adequadamente em computadores. Um conjunto de recursos que podem ser compostos para constituir programas específicos, mais um conjunto de regra de composição que garantem que todos os programas podem ser implementados em computadores com qualidade apropriada.

2.1.2 Programação Orientação a Objetos

Programação Orientada a Objetos ou POO é um modelo de programação de software baseado na composição e interação entre diversas unidades chamadas de objetos.

Objetos representam entidades possuindo características próprias (atributos) e executa determinadas ações (métodos), sendo esses atributos e métodos provenientes da classe que origina o objeto. Os programas são arquitetados através de objetos que interagem entre si.

Classe é uma coleção de objetos que são descritos com os mesmos atributos e as mesmas operações é um tipo abstrato de dados, um "molde" para a criação de objetos. Uma classe representa uma ideia ou conceito e classificam objetos que tenham propriedades similares, as classes são os blocos de construção mais importantes dos sistemas orientados a objetos, e devem possuir responsabilidades bem definidas dentro da aplicação.

Atributo ou Propriedade é uma característica particular de uma classe, como exemplo a idade de uma pessoa.

Métodos são funções e procedimentos contidos na classe, o ato de invocar um método é a passagem de mensagens para o objeto, toda classe possui um método especial chamado de método construtor, que inicializa o objeto instanciado. As classes também usam métodos destruidores para finalizar os objetos após seu uso, um método são comandos que possui um nome de identificação, é similar a uma função ou procedimento. Um método possui um nome e um corpo onde ficam os comandos que serão executados quando o método for chamado, e também podem receber dados para processamento e retornar informações. Os métodos também precisam ter um tipo de retorno, que pode ser um tipo de valor, de referência ou ainda void que não retornam valor.

Exemplos de método:

Get:

```
public String getNome(){
return nome;
}
```

Set:

```
public void setNome(String nome){
This.nome = nome;
}
```

Herança, em orientação a objetos, é a capacidade de um novo objeto tomar atributos e operações de um objeto existente. Assim, podem-se criar classes mais complexas sem a necessidade de repetir código. Por exemplo, para evitar repetição ao definir classes com características em comum e que são relacionadas entre si. A herança é um relacionamento entre classes, que permite que uma classe herde os membros (atributos e métodos) de outra classe.

Encapsulamento é a combinação de atributos e operações dentro de uma classe, deixando visível apenas o que é necessário para a comunicação entre dois objetos. Há três níveis principais de acesso dos objetos: público permite acesso a todos os objetos; privado, somente acesso a instancia e protegido, garante acesso para o objeto e para as subclasses.

Abstração é transformar o requisito real em algo abstrato, não se preocupando com todos os detalhes, mas tratando dos detalhes importantes ao programa. A abstração permite a sintetização de um requisito facilmente e ajuda a alcançar sua reutilização. Muitas vezes os programas são complexos e especializados, sempre que possível, é importante criar objetos que resolvam um domínio inteiro de requisitos.

Polimorfismo é um mecanismo que seleciona as funcionalidades utilizadas de forma dinâmica por um programa no decorrer de sua execução. Com o Polimorfismo, os mesmos atributos e objetos podem ser utilizados em objetos distintos, porém, com implementações lógicas diferentes.

Exemplo: podemos dizer que uma classe chamada Vendedor e outra chamada Diretor podem ter como base uma classe chamada Pessoa, com um método chamado CalcularVendas. Se este método (definido na classe base) se comportar de maneira diferente para as chamadas feitas a partir de uma instância de Vendedor e para as chamadas feitas a partir de uma instância de Diretor, ele será considerado um método polimórfico, ou seja, um método de várias formas[5].

Assim podemos ter na classe base o método CalcularVendas:

```
public decimal CalcularVendas()

    {

        decimal valorUnitario = decimal.MinValue;

        decimal produtosVendidos = decimal.MinValue;



        return valorUnitario * produtosVendidos;

    }
```

Figura 2 - Exemplo de Polimorfismo

Na classe Vendedor temos o mesmo método, mais com a codificação diferente:

```
public decimal CalcularVendas()

    {

        decimal valorUnitario = 50;

        decimal produtosVendidos = 1500;



        return valorUnitario * produtosVendidos;

    }
```

Figura 3 - Segundo Exemplo de Polimorfismo

O mesmo ocorre na classe Diretor:

```
public decimal CalcularVendas()

    {

        decimal valorUnitario = 150;

        decimal produtosVendidos = 3800;

        decimal taxaAdicional = 100;



        return taxaAdicional + (valorUnitario * produtosVendidos);

    }
```

Figura 4 - Ultimo exemplo de Polimorfismo

2.1.3 PHP

PHP: Hypertext Preprocessor, que originalmente se chamava Personal Home Page (PHP) foi criada por Rasmus Lerdorf em 1995, é uma linguagem de script capaz de lidar com várias funções de backend como coletar formulários de dados, gerenciar arquivos do servidor e modificar bases de dados. O PHP é incorporado dentro do HTML tendo seu código adjacente ao HTML.

Exemplo:

<html>

<head>

<title>Exemplo.com</title>

</head>

<body>

<p>olá,

<?php

$firstname = ‘Joao’;

$lastname = ‘Batista’;

Echo “$firstname $lastname”;

?>

</body>

</html>

Tudo entre <?php ?> faz parte do script.

Variáveis são identificadas pelo $ precedendo seu nome e possuem um tipo de acordo com o valor atribuído a elas. São valores mutáveis que só existem durante a execução de um programa,

também são armazenadas na memoria RAM e seu conteúdo é destruído após a execução do programa.

Tipo Boolean expressa um valor lógico que pode ser verdadeiro ou falso, utilizando TRUE ou FALSE respectivamente.

Tipo numérico é separado entre integer e float, que representam números inteiros e decimais respectivamente.

Tipo string é uma cadeia de caracteres alfanuméricos, para declarar a variável é utilizadas aspas simples (‘ ‘) ou aspas duplas (“ “).

Tipo array é uma lista de valores armazenados na memoria que podem ser de tipos diferentes e podem ser acessados a qualquer momento, pois cada valor é relacionado a uma chave.

Tipo objeto é uma entidade com um determinado comportamento definido por seus métodos e suas propriedades, para criar um objeto deve se utilizar o operador new.

Tipo recurso é uma variável que mantem uma referencia de recurso externo, eles são criados e utilizados por funções como aquelas que criam conexões de banco de dados, quando as funções mysql_connect() e pg_connect() são conectadas ao banco de dados, retornam uma variável de referencia do tipo recurso.

Exemplo:

Resource mysql_connect ([string $server[, string $username[, string $password
[, bool $new_link [, int $client_flags]]]]])

Resource pg_connect (string $connection_string)

Superglobais são variáveis disponibilizadas pelo próprio PHP que são acessíveis a partir de qualquer lugar na execução do script, elas carregam alguns conteúdos, dependendo de como o script foi invocado, como a seguir:

$GLOBALS contém uma lista com todas as variáveis globais disponíveis para o script.

$_SERVER contém informações sobre o ambiente em que o script está rodando, como endereço do servidor, da requisição, nome do script acessado, entre outros.

$_GET contém um vetor com as variáveis informadas em sua requisição, como exemplo a requisição http://localhost/teste.php?name=joao&idade20 retorna um vetor com as posições joao e idade, contendo os respectivos valores.

$_POST Funciona do mesmo modo que **$_GET**, porém contém as informações enviadas pelo método POST, normalmente utilizado no envio de formulários.

$_FILES contém um vetor com informações dos arquivos enviados para upload.

$_COOKIE contém um vetos com as variáveis da sessão do usuário.

$_REQUEST contém um vetos com as informações de **$_GET**, **$_POST**, e **$_COOKIE**.

$_ENV contém um vetor com variáveis de ambiente.[7]

2.1.4 MySQL

MySQL é um servidor e gerenciador de banco de dados relacional, que utiliza a linguagem padrão SQL (Structured Query Language), criado pelos desenvolvedores David Axmark, Allan Larsson e Michael Widenius durante a década de 90, que buscavam uma uma interface SQL compatível com as rotinas que utilizavam.

Para criar um banco de dados no MySQL é utilizado o utilitário mysql, executando os comandos CREATE DATABASE e USE para utilizar do banco de dados, exemplo:

```
CREATE DATABASE exemplo;
USE exemplo;
```

Dentro do banco de dados é utilizado tabelas para a armazenar os dados, é possível cria-las utilizando o comando CREATE TABLE, exemplo:

```
CREATE TABLE tabela_exemplo (
campo_exemplo tipo_de_dado [NULL ou NOT NULL]
[DEFAULT dado_padrão]
);
```

tabela_exemplo e campo_exemplo são os nomes da tabela e o campo respectivamente, tipo_de_dado é substituído por algum tipo de dado, podendo ser:

Int um número inteiro;

Decimal um número decimal, de ponto fixo;

Float um número de 32 bits, possui ponto flutuante de precisão simples;

Double um número de 64 bits, possui ponto flutuante de precisão dupla;

Char uma cadeia de caracteres de tamanho fixo;

Varchar uma cadeia de caracteres de tamanho variável;

Date um valor referente a data ano-mês-dia;

Time um valor referente ao horário no formato horas:minutos:segundos;

Para incluir um ou mais dados em uma tabela do banco de dados, é utilizado o comando INSERT, existindo duas possíveis sintaxes para o comando, exemplo:

INSERT INTO tabela_nome VALUES (valor1, valor2, ...);

Na variação do comando acima, os valores valor1 e valor2 são incluídos na mesma ordem que foram definidos os campos durante a criação da tabela

Ou

INSERT INTO tabela_nome VALUES (campo_nome1, campo_nome2, ...)
VALUES (valor1, valor2, ...)

Nessa variação do comando insert, os valores são inseridos na ordem especifica entre parênteses, logo após o nome da tabela, campos não listados receberão o valo NULL.

Para alterar os registros de uma tabela ou a tabela em si é utilizado o comando UPDATE, seguindo a sintaxe a seguir para alterar dados:

UPDATE tabela_nome SET campo1=valor1 [, campo2=valor2, ...,]
[WHERE condição];

O **WHERE** é utilizado para criar uma certa condição na alteração dos dados. Seu uso não é necessário sem a condição.

Outro tipo de alteração que podemos fazer em uma tabela é em sua estrutura, utilizando o comando **ALTER TABLE**. Podem ser usadas as seguintes sintaxes:

ALTER TABLE tabela_nome ADD campo_nome tipo_de_dado;

Essa sintaxe é utilizada para adicionar um campo a uma tabela existente.

ALTER TABLE tabela_nome RENAME TO novo_nome_tabela;

Essa sintaxe é utilizada para alterar o nome da tabela existente.

O comando DELETE exclui um ou mais registros de determinada tabela. Sua sintaxe é a seguinte:

DELETE FROM tabela_nome [WHERE condição];

Por exemplo, para excluir o produto de código 2, podemos digitar o comando a seguir:

DELETE FROM produtos WHERE codigo =2;

Para excluir todos os produtos existentes na tabela, basta digitar:

DELETE FROM produtos;

Para excluir uma tabela inteira (dados e estrutura) do banco de dados, usamos o comando DROP TABLE. Sua sintaxe é a seguinte:

DROP TABLE nome_tabela;

Existe também o comando DROP DATABASE, que serve para excluir um banco de dados inteiro.

Com o comando SELECT é possível executar diversos tipos de consultas sobre as tabelas de um banco de dados, sua sintaxe básica é a seguinte:

SELECT lista_de_campos FROM lista_de_tabelas [WHERE condição];

Se a lista de campos for substituída por um asterisco (*), serão retornados todos os campos existentes nas tabelas em uso. Se a cláusula WHERE for omitida, serão mostrados todos os registros das tabelas determinadas na lista de tabelas. Por exemplo, para selecionar os campos código, nome e preço da tabela de produtos, o comando utilizado é:

SELECT codigo_produto, nome_produto, preco FROM produtos;

Para ordenar os registros retornados por uma consulta, é utilizado ORDER BY, seguida pelo nome dos campos a serem ordenados, os campos serão avaliados da esquerda para a direita.

Podemos também fazer uma ordenação decrescente dos resultados, e para isso basta utilizar a opção DESC após o nome do campo:

```
SELECT nome_produto FROM produtos ORDER BY nome_produto DESC;
```

É possível também utilizar LIMIT para determinar o número máximo de registros que uma consulta pode retornar. Se, por exemplo, um usuário fizer uma busca pela palavra “CD”, e houver mais de 500 CDs na loja, obviamente não será mostrado os 500 CDs na mesma tela, pois a página ficaria muito grande. Então, seria determinado que devem ser retornados no máximo 10 CDs, utilizando o seguinte comando:

```
SELECT * FROM produtos WHERE nome_produto LIKE ‘CD %’ LIMIT 10;
```

Dessa forma exibiríamos apenas os 10 primeiros CDs ao usuário. É claro que, para obtermos 10 registros por vez, devemos utilizar a opção ORDER BY, assim, evitamos que os registros apareçam em uma ordem diferente a cada chamada. Por exemplo:

```
SELECT * FROM produtos WHERE nome_produto LIKE ‘CD %’
ORDER BY nome_produto LIMIT 10;
```

O parâmetro OFFSET pode ser utilizado em conjunto com o LIMIT, para determinar a partir de qual registro a consulta deve retornar. Portanto, para retornar os próximos 10 registros, utilizamos o seguinte comando:

```
SELECT * FROM produtos WHERE nome_produto LIKE ‘CD %’
ORDER BY nome_produto LIMIT 10 OFFSET 10;[6]
```

2.2 Desenvolvimento da pesquisa

2.2.1 Requisitos do modulo 07

Para desenvolver um software é necessário ter uma visão para organizar o projeto, é necessário a analise do projeto, tanto a função de analista para planejar as funcionalidades, criar diagramas e a documentação geral do projeto.

A imagem abaixo apresenta o diagrama de casos de uso do modulo 07.

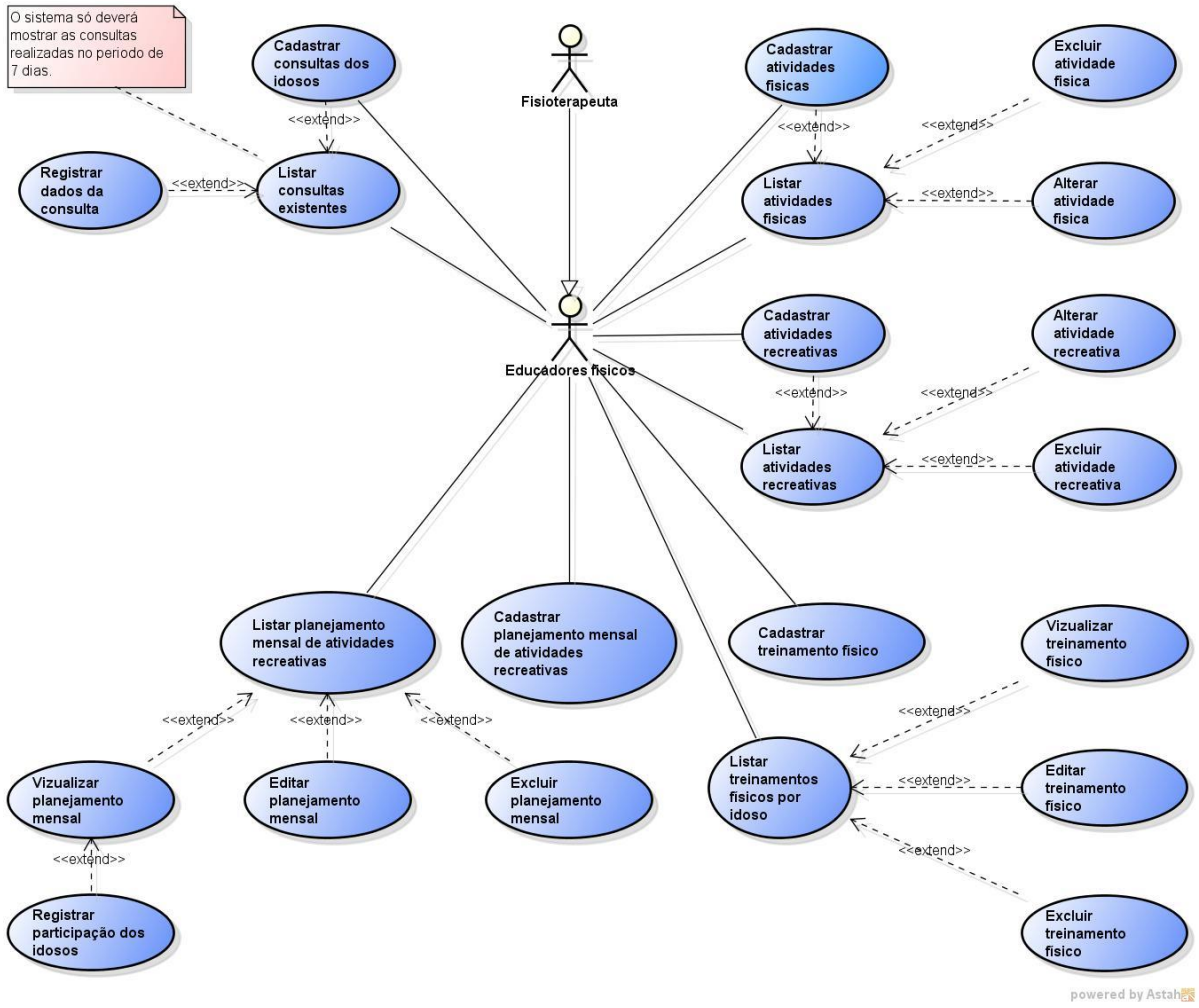

Figura 5 - Diagrama de casos de uso do módulo 07

Os bonecos na ilustração representam os funcionários da instituição, e os balões ligados a eles representando suas possíveis ações.

Um dos pontos principais do desenvolvimento de um software é a levantamento de requisitos funcionais, que definem as funções do sistema delimitando o que é necessário para o funcionamento do software. A tabela abaixo representa os requisitos funcionais do modulo 07.

Identificador	Descrição do Requisito
RF #01	O educador físico para ter acesso às informações do idoso, terá que fazer login na aba LOGIN através do prontuário e da senha. • **Prontuário do educador** → char (7) * • **Senha do educador** → varchar (20) * Uma vez acessado, terá acesso à página inicial. Com as seguintes funcionalidades: ✓ Lista de idosos → direcionado para outra aba com a opção de busca por idoso ✓ Atividades Recreativas → direcionado para a aba de Gerenciamento das atividades, do plano mensal (realizado em grupo), e o registro de participação nas atividades (descrição visual e textual)

	✓ Atividades Físicas → direcionado para a aba de Gerenciamento dos exercícios, registro das consultas periódicas ✓ Chat → servindo como um lembrete para o educador físico * Obrigatório
RF #02	Na janela que diz respeito ao chat, que se encontrara possivelmente no canto inferior direito da tela, haverá informações voltadas ao idoso, de alerta informando: ✓ Data de atividade recreativa/física* ✓ Nome do idoso* ✓ Prontuário do idoso* ✓ Dúvidas * Obrigatório
RF #03	Encaminha para uma aba com uma lista de idosos residentes nessa instituição. No começo da página haverá um campo para a inserção do prontuário do idoso, que quando inserido o sistema irá direcionar para o idoso específico. Pela utilização do slidedown, ao passar o cursor do mouse no nome do idoso devem ser apresentados as seguintes informações: • **Foto *** • **Nome** → varchar (100) * • **Idade** → char (3) * Quando dado um click no nome do idoso, deverá atualizar a página a partir da mesma self, apresentando, além das informações já citadas: • **CPF** → varchar (14) * • **Peso** → float (xxx,x kg)* • **Altura** → float (x,xx m)* • **Idade** → char (3) * • **Estado civil** → varchar (15) • **Data de entrada na instituição** → date (xxxx/xx/xx)* Sendo apresentadas, anotações básicas sobre o idoso (informações pessoais), o usuário do sistema deverá ter acesso a partir dessa aba (aba do idoso) para a janela de atividades recreativas e físicas. * Obrigatório
RF #04	Na janela que diz respeito às atividades recreativas, que deverá ser dividida em duas partes. De início uma área voltada ao registro de participação nas atividades de recreação, enfim, seguindo para a definição do plano mensal das atividades recreativas. • Registro de participação → cabe ao usuário digitalizar informações voltadas a atividade do dia realizada em grupo ou individualmente. • Plano mensal → detalha todas as atividades de recreação acontecidas em grupo, programadas ao longo de todo o mês. *Obrigatório
RF #05	Após ter sido realizada qualquer atividade recreativa em questão, o usuário deverá fazer uma anotação, no momento da atividade, e repassar para o sistema. Segue abaixo dados que deverão ser preenchidos pelo usuário de forma a concluir o registro de participação dos idosos. • **Data** → date (XXXX/XX/XX) * • **Hor**a → float (xx:xx) *

	• **Atividade** (ex: bingo, dominó entre outras opções que a instituição disponibilizar) → varchar (50) * Deverá ser possível também registrar, individualmente, cada um dos idosos que participaram da atividade. Sendo assim, uma lista com o nome de todos os idosos da instituição deverá estar presente nessa área destinada a registros e ao lado, uma de participação. Vale ressaltar, que uma lista de presença encontrada ao lado direito da lista de todos os residentes na instituição em questão obrigatoriamente deverá ser preenchida após todas as atividades ocorridas. A partir de um checkbox o educador (usuário responsável) tenderá a anotar todas as presenças e faltas de cada idoso. * Essa aba poderá contar ainda com um campo destinado para fotos, sendo de total responsabilidade do educador registrá-la. Por fim, dois botões ao final desse espaço deverão estar presentes, com o intuito de que fique salvo as alterações feitas segue abaixo botões de ordem obrigatória: ✓ Salvar * ✓ Cancelar * * Obrigatório
RF #06	Para que haja uma melhor definição das atividades realizadas no momento de recreação, uma aba deverá ser reservada. O plano mensal dessas atividades será distribuído em tabela com os campos a seguir: • **Data da atividade** → date (XXXX/XX/XX) * • **Hor**a → float (xx:xx) * • **Atividade recreativa** (ex: bingo, dominó entre outras opções que a instituição disponibilizar que deverão ser registradas em uma janela específica) → varchar (50) * • **Duração** (definido em formato de hora) → varchar (10) * Ao final de todas o registro de dados, o usuário irá disponibilizar de dois botões, um de inserção “inserir” e um para cancelamento “cancelar”. * * Obrigatório
RF #07	Na janela que diz respeito ao cadastro das atividades físicas, deverá ser apresentada a seguinte tabela em listagem (1,2,3,4,5,6, ...): ✓ Nome da atividade* ✓ Tipo da atividade* Apresentando também as seguintes funcionalidades: ✓ Exclusão → irá gerar um alert com a seguinte mensagem “Você realmente deseja excluir as informações selecionadas?”, com os botões “sim” ou “não”. * ✓ Inserção → para inserir uma nova atividade física, deverão ser informados: nome, tipo de atividade e foto, com os botões “inserir” ou “cancelar”. * ✓ Atualização → um dado só será atualizado a partir da abertura de um alert, com os dados já cadastrados no momento da inserção, sendo exibido: nome, tipo de atividade e foto. E, logo, abaixo dois botões, um de confirmação “confirmar” e outro de cancelamento “cancelar”. * ✓ Além desses tópicos, a tabela deverá ter um campo de inserção de nome, para filtrar a informação que o usuário necessita, facilitando no manuseio do sistema e economizando o tempo do usuário. E juntamente, um campo para inserção de foto, dessa forma, o educador físico deixa mais claro qual a atividade escolhida. * * Obrigatório

RF #08	Na janela que diz respeito ao cadastro das atividades recreativas, deverá ser apresentada a seguinte tabela em listagem (1,2,3,4,5,6, ...): ✓ Nome da atividade* ✓ Tipo da atividade* Apresentando também as seguintes funcionalidades: ✓ Exclusão → irá gerar um alert com a seguinte mensagem "Você realmente deseja excluir as informações selecionadas?", com os botões "sim" ou "não". ✓ Inserção → para inserir uma nova atividade recreativa, deverão ser informados: nome, tipo de atividade e foto, com os botões "inserir" ou "cancelar". * ✓ Atualização → um dado só será atualizado a partir da abertura de um alert, com os dados já cadastrados no momento da inserção, sendo exibido: nome, tipo de atividade e foto. E, logo, abaixo dois botões, um de confirmação "confirmar" e outro de cancelamento "cancelar". ✓ Além desses tópicos, a tabela deverá ter um campo de inserção de nome, para filtrar a informação que o usuário necessita, facilitando no manuseio do sistema e economizando o tempo do usuário. E juntamente, um campo para inserção de foto, dessa forma, o educador físico deixa mais claro qual a atividade escolhida. * Obrigatório
RF #09	Todas as funcionalidades descritas no requisito funcional 03 (RF #03), somente e deverão ser modificadas por pessoas específicas, tendo em vista, que o sistema terá que ser controlado para que não haja alguma alteração indevida. Sendo assim, a inserção, a alteração e exclusão de dados das atividades recreativas caberão ao preparador físico ou fisioterapeuta. Segue abaixo, alguns detalhes que o usuário deverá estar atento. • **Prontuário do educador** → char (7) * Somente com prontuário e senha o educador físico ou ainda, fisioterapeuta poderá alterar ou inserir qualquer que seja a informação. Esse passará por uma verificação, sendo liberado ou não o acesso. Se sim, segue normalmente o fluxo do sistema. Login → Pagina inicial → Atividades físicas/ recreativas • **Prontuário do idoso** → char (7) * Para ser realizada a funcionalidade escolhida pelo usuário (alteração ou inclusão) o mesmo deverá ter disponível no prontuário do idoso, assim todas as suas tarefas recreativas serão selecionadas e a escolhida poderá então ser mexida com a total precisão. Uma vez alterada, qual seja a informação o mesmo não poderá recuperar os dados. Sendo assim, o usuário deverá estar atento ao que mexe e o quanto é valiosa a informação. A cada alteração no banco de dados, o educador deverá acrescentar alguma: • **Observação** → varchar (500) ** Para descrever detalhes que não puderam ser informados nas categorias anteriores, como por exemplo, "o idoso X reclamou de dor na perna enquanto realizava Y atividade". * Obrigatório
RF #10	Na janela que diz respeito à consulta, terá três tópicos, que são: • Recente → apresentará informações das consultas recentes, que vão ser as consultas realizadas nos últimos 7 dias, assim o sistema fará essa atualização. Essas informações serão apresentadas em uma tabela com os seguintes dados:

	• **Peso** → float (xxx,x kg) • **Altura** → float (x,xx m) • **IMC** → float (xx,xx kg*/m)* Será gerado automaticamente quando o peso e altura forem inseridos. • **Pressão** → float (xx,x mmhg) • **Envergadura** → float (x,xx m) • **Observações adicionais** → varchar (500) • Nova → o médico vai inserir as informações que ele obteve do idoso naquela consulta, que será: • **Peso** → float (xxx,x kg) • **Altura** → float (x,xx m) • **IMC** → float (xx,xx kg*/m)* Será gerado automaticamente quando o peso e altura forem inseridos. • **Pressão** → float (xx,x mmhg) • **Envergadura** → float (x,xx m) • **Observações adicionais** → varchar (500) Esse campo servirá para que o médico insira informações adicionais, se caso o idoso apresentar algum resultado diferente do que geralmente costuma ter das informações citadas acima, apresentar sintomas anormais, etc. • Histórico → apresentará as informações de todas as consultas que foram feitas desde que o idoso entrou na instituição. Essas informações serão apresentadas em uma tabela com os seguintes dados: • **Peso** → float (xxx,x kg) • **Altura** → float (x,xx m) • **IMC** → float (xx,xx kg*/m)* Será gerado automaticamente quando o peso e altura forem inseridos. • **Pressão** → float (xx,x mmhg) • **Envergadura** → float (x,xx m) • **Observações adicionais** → varchar (500) * Obrigatório
RF #11	As atividades físicas regulares, além de serem registradas pelo educador, precisam constar sua data de início, data prevista de término, e o objetivo para qual está sendo realizada. Além disso, será necessário um select de *sim* ou *não* referente à participação do idoso na atividade física. Todas as respostas dos selects serão armazenadas em uma tabela, e ao final do mês, através desses dados, será contabilizada a frequência percentual desse idoso nas atividades físicas da instituição. • **Nome** → varchar (100) • **Prontuário** → varchar (xxx.xxx.xxx-xx) • **Data de início** → date (XXXX/XX/XX) * • **Data prevista de término** → date (XXXX/XX/XX) * • **Objetivo** → varchar (500) *

	O objetivo terá uma barra de progresso semelhante ao Redmine, cuja data determinará o quão próximo o objetivo está de ser realizado. Assim, o educador terá uma visão mais ampla de como está a situação de cada idoso. * Obrigatório
RF #12	Uma janela específica par o treino dos idosos deverá estar presente no sistema, com todas as funcionalidades necessárias para que se tenha um fluxo. Cabe ao educador físico registrar as atividades por idoso, sendo necessário: • **Prontuário → Select*** • **Dia da semana → Select*** • **Atividade → Select*** • **Repetições → varchar (100) *** As elaborações dos treinos deverão ser feitas ao longo da consulta periódica destinada a cada idoso. Dessa forma o avaliador, já terá uma noção de qual é a situação do idoso e de qual exercício em específico aquele idoso necessita. Segue abaixo um exemplo de treino, para melhor entendimento: • Exemplo de Treino: Idoso: Tio Breno Atividade 01: Caminhada / 02 vezes por semana / 10 minutos por dia Atividade 02: Alongamento do pescoço / 01 vezes por semana / 10 vezes por dia Atividade 03: Alongamento do joelho / 03 vezes por semana / 05 vezes por dia * Obrigatório
RF #13	O sistema deverá controlar os privilégios de acesso às funcionalidades desse módulo, ou seja, o acesso que o educador físico terá não será o mesmo que um usuário comum. **Educadores e fisioterapeutas:** podem incluir, atualizar, modificar, excluir, selecionais quaisquer dados referentes aos exercícios físicos, atividades recreativas e treinos, tendo total liberdade de modificação com exceção dos dados pessoais do idoso, como nome, CPF e RG. **Usuário comum:** poderá apenas visualizar e selecionar, e caso tenha alguma dúvida de modificação dos dados, deve entrar em contato com a instituição pelo formulário de contato.

2.2.2 Desenvolvimento dos casos de uso

Para organizar o desenvolvimento necessário pelo modulo 07, o trabalho foi separado em iterações, colaborando para a gestão de tempo e o foco do projeto. A seguir é uma imagem da tabela do planejamento das iterações para o desenvolvimento dos casos de uso.

Planejamento das Iterações para o Desenvolvimento dos Casos de Uso - Módulo de Cadastros					
Iteração	Data Prevista	Casos de Uso	Equipe Responsável	Desenvolvido ?	Testado?
#01	02/07/2019	1.1 - Cadastrar atividades físicas	Tatiane (Desenvolvedor) e Lucas Henrique (Desenvolvedor)	OK	OK
		1.2 - Listar atividades físicas		OK	OK
		1.3 - Alterar atividade física		OK	OK
		1.4 - Excluir atividade física		OK	OK
#02	23/08/2019	1.5 - Cadastrar treinamento físico por idoso	Grupo 01: Adryelle (Analista), Tatiane (Desenvolvedor) e João Gabriel (DBA)	OK	OK
		1.6 - Listar treinamento físico por idoso		OK	OK
		1.13 - Cadastrar atividades recreativas	Grupo 02: Dennison (Analista), Lucas Henrique (Desenvolvedor) e Rita (DBA)	OK	OK
		1.15 - Excluir atividade recreativa		OK	OK
		1.16 - Listar atividades reacreativas		OK	OK
#03	20/09/2019	1.7 - Alterar treinamento físico por idoso	Grupo 01: Adryelle (Analista), Tatiane (Desenvolvedor) e João Gabriel (DBA)	OK	OK
		1.8 - Excluir treinamento físico por idoso		OK	OK
		1.9 - Visualizar treinamento físico por idoso		OK	OK
		1.14 - Alterar atividades recreativa	Grupo 02: Dennison (Analista), Lucas Henrique (Desenvolvedor) e Rita (DBA)	OK	OK
#04	18/10/2019	1.17 - Cadastrar planejamento mensal de atividades recreativas	Grupo 02: Dennison (Analista), Lucas Henrique (Desenvolvedor) e Rita (DBA)	NOK	NOK
		1.18 - Listar planejamento mensal de atividades recreativas		NOK	NOK
		1.19 - Visualizar planejamento mensal de atividades recreativas		NOK	NOK
		1.20 - Alterar planejamento mensal de atividades recreativas		NOK	NOK
		1.21 - Excluir planejamento mensal de atividades recreativas		NOK	NOK
#05	08/11/2019	1.10 - Cadastrar consultas dos idosos	Grupo 01: Adryelle (Analista), Tatiane (Desenvolvedor) e João Gabriel (DBA)		
		1.11 - Registrar dados da consulta		NOK	NOK
		1.12 - Listar consultas existentes		NOK	NOK
		1.22 - Registrar participação dos idosos		NOK	NOK

Figura 6 - Tabela das iterações do módulo 07

Será apresentado nessa seção o processo de desenvolvimento de alguns casos de uso selecionados do modulo 07:

2.2.2.1 Cadastrar Atividade Recreativa

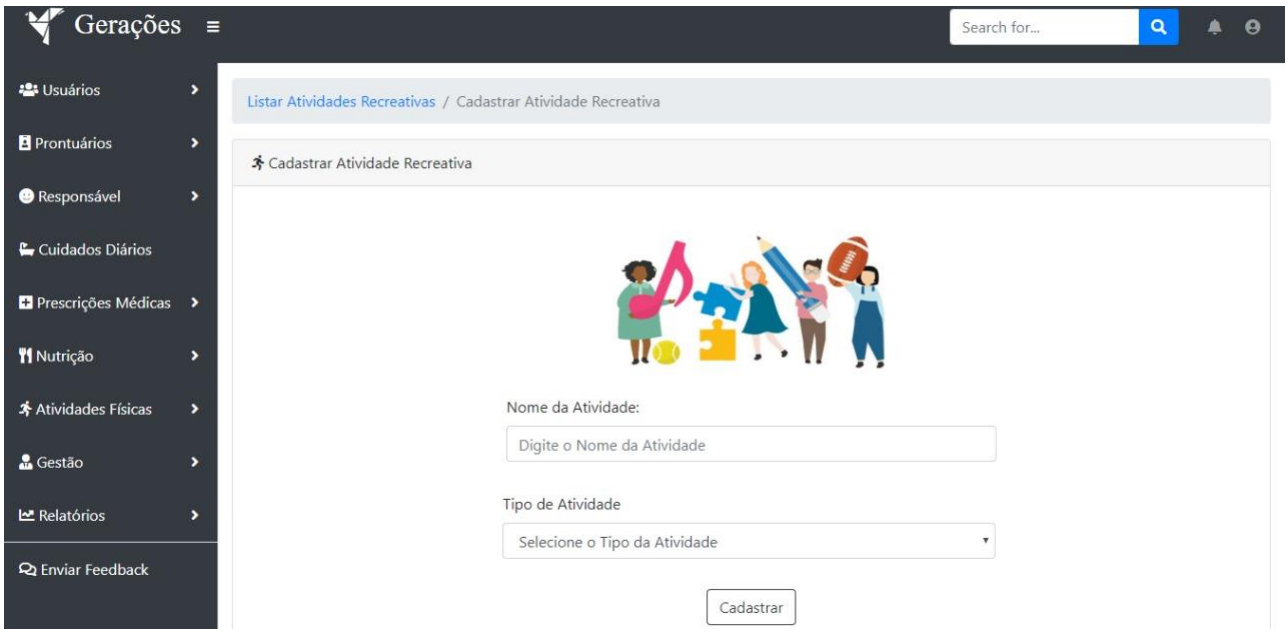

Figura 7 - Interface da pagina de cadastrar atividade recreativa

Uma das paginas com mais importância no modulo de atividades físicas, é a de cadastrar uma atividade recreativa, um usuário logado no servidor com permissão de educador físico tem acesso à tela e pode criar a atividade recreativa que tem em mente no banco de dados do projeto gerações. Caso o usuário crie uma atividade sem nome ou sem tipo, o servidor o alertara do erro, atividades do mesmo tipo não podem possuir o mesmo nome.

```
public function cadastrar(){

    if(isset($_POST['ARE_NOME']) &&  isset($_POST['FK_TIPOS_ATIVIDADES_RECREATIVAS_TAF_ID'])){

        $AtividadeRecreativa = new AtividadeRecreativa();

        $AtividadeRecreativa->__set('ARE_NOME',$_POST['ARE_NOME']);

        $AtividadeRecreativa->__set('FK_TIPOS_ATIVIDADES_RECREATIVAS_TAR_ID',$_POST['FK_TIPOS_ATIVIDADES_RECRETIVAS_TAR_ID']);

       $AtividadeRecreativaDAO = new $AtividadeRecreativaDAO();

       $AtividadeRecreativaDAO->inserir($AtividadeRecreativa);
    }

    header('Location: /md7/CadastrarAtividadeRecreativa');
    die();
}
```

Figura 8 - codigo da função de cadastrar atividade recreativa

```
    Cadastrar Atividade Recreativa
</div>

<div class="card-body">

    <div class="text-center">

        <img src="<?php echo $this->view->include ?>resources/img/atividade_recreativa.jpg" class="rounded"  height="200px" width="320px"/>

    </div>

    <form action="" method="post">
        <div class="form-group form-row mr-0 ml-md-0">
            <div class="form-group col-md-3">

            </div>
            <div class="form-group col-md-6">
                <label for="nomeatv" text-align="center" >Nome da Atividade:</label>
                <input type="text" name="nome" class="form-control" id="nomeatv" placeholder="Digite o Nome da Atividade" required="required" maxlength="100" >
            </div>
        </div>

        <div class="form-group form-row mr-0 ml-ml-0">
            <div class="form-group col-md-3">

            </div>

            <div class="form-group col-md-6">
                <label for="inputState">Tipo de Atividade</label>
                <select id="inputState" required name="tipo" class="form-control" >
                    <option selected disabled hidden>Selecione o Tipo da Atividade</option>

                    <?php

                    foreach($tipos as $tipo)
                    {
                        ?>
                            <option value="<?php echo $tipo->__get('TAR_ID'); ?>"><?php echo $tipo->__get('TAR_NOME'); ?></option>
                        <?php
                    }

                    ?>

                </select>
            </div>
```

Figura 9 - Codigo da interface de cadastrar atividade recreativa

```
public function alterar(){

    $AtividadeRecreativa = new AtividadeRecreativa();

    $AtividadeRecreativa->__set('ARE_ID',$_GET['ARE_ID']);

    $AtividadeRecreativaDAO = new AtividadeRecreativaDAO();

    $AtividadeRecreativa = $AtividadeRecreativaDAO->buscarPorId($AtividadeRecreativa);

    $this->getView()->mostrar = $AtividadeRecreativa;

    $this->render( '../md7/AlterarAtividadeRecreativa');
}

public function alteracao(){

    if(isset($_POST['ARE_ID']) && isset($_POST['ARE_NOME']) && isset($_POST['FK_TIPOS_ATIVIDADES_RECREATIVAS_TAR_ID']) )
    {
        $AtividadeRecreativa = new Produto();

        $AtividadeRecreativa->__set('ARI_ID',$_POST['ARI_ID']);

        $AtividadeRecreativa->__set('ARI_NOME',$_POST['ARI_NOME']);

        $AtividadeRecreativa->__set('FK_TIPOS_ATIVIDADES_RECREATIVAS_TAR_ID',$_POST['FK_TIPOS_ATIVIDADES_RECREATIVAS_TAR_ID']);

        $AtividadeRecreativaDAO = new AtividadeRecreativaDAO();

        $AtividadeRecreativaDAO->alterar($AtividadeRecreativa);
    }
```

2.2.2.2 Listar Atividades Recreativas

Um dos casos de uso do modulo 07, listar atividades recreativas cria uma lista de tamanho modificável dando as opções de alterar a atividade, exclui-la ou até mesmo adicionar uma atividade nova, levando para suas interfaces respectivas.

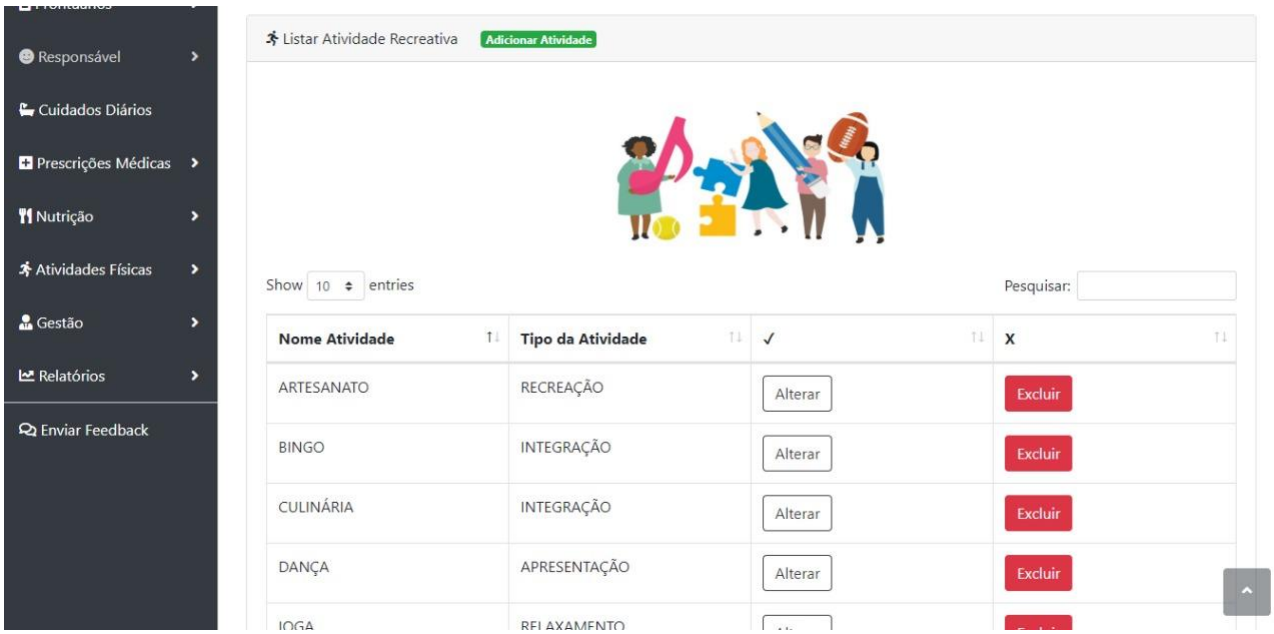

Figura 10 - Interface de listar atividades recreativas

```php
<tbody>
    <?php

    foreach($atividades as $atv)
    {
        ?>
        <tr>
            <form action="/md7/AlterarAtividadeRecreativa" method="post">
                <input type="hidden" name="id" value="<?php echo $atv->__get('ATR_ID'); ?>">

                <td><?php echo $atv->__get('ATR_NOME'); ?></td>

                <?php
                    foreach($tipos as $tipo)
                    {
                        if($tipo->__get('TAR_ID') == $atv->__get('FK_TIPO_ATIVIDADES_RECREATIVAS_TAR_ID'))
                        {
                            echo '<td>' . $tipo->__get('TAR_NOME') . '</td>';
                        }
                    }
                ?>

                <td><input type="submit" class="btn btn-outline-dark" value="Alterar"/></td>
                <td><a onclick="defineId('<?php echo $atv->__get('ATR_ID'); ?>', '<?php echo $atv->__get('ATR_NOME'); ?>')"
                    href="" class="btn btn-danger" data-toggle="modal" data-target="#modalConfirmDelete">Excluir</a></td>
            </form>
        </tr>
        <?php
    }

    ?>

</tbody>
```

Figura 11 - Código da interface de listar atividades recreativas

Excluir é uma função integral a listagem de atividades recreativas, pois ocorre na mesma interface. Após clicar na opção excluir uma janela aparece perguntando se deseja realmente fazer a ação.

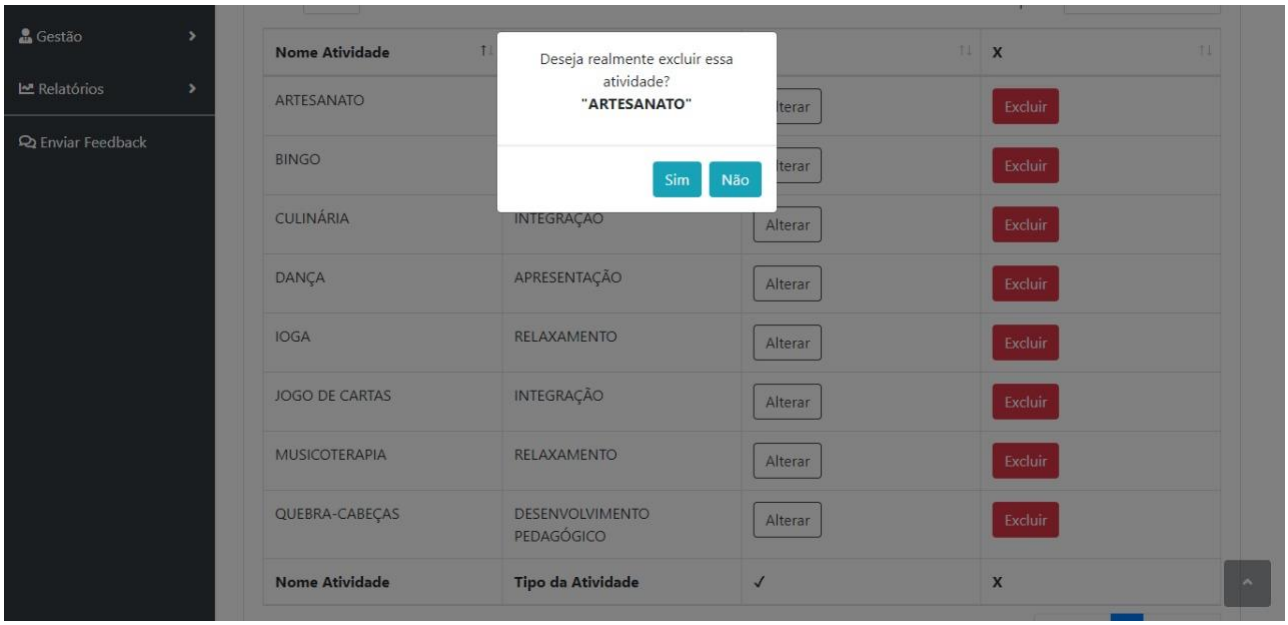

Figura 12 - Interface de excluir atividade recreativa

```php
public function excluir(){

    if(isset($_GET['ARE_ID'])){

        $AtividadeRecreativa = new AtividadeRecreativa();

        $AtividadeRecreativa->__set('ARI_ID',$_GET['ARI_ID']);

        $AtividadeRecreativaDAO = new AtividadeRecreativaDAO();

        $AtividadeRecreativaDAO->excluir($AtividadeRecreativa);
    }

    header('Location: /');
    die();
}
```

Figura 13 - Código de a função excluir

2.2.2.3 Alterar Atividade Recreativa

Alterar atividade recreativa possui acesso exclusivo pela função listar, onde é selecionada uma atividade recreativa especifica e alterada seu nome e/ou seu tipo. Caso não for escolhido um tipo de atividade recreativa ou um nome a pagina alerta o usuário, nomes de atividades do mesmo tipo não podem ser iguais.

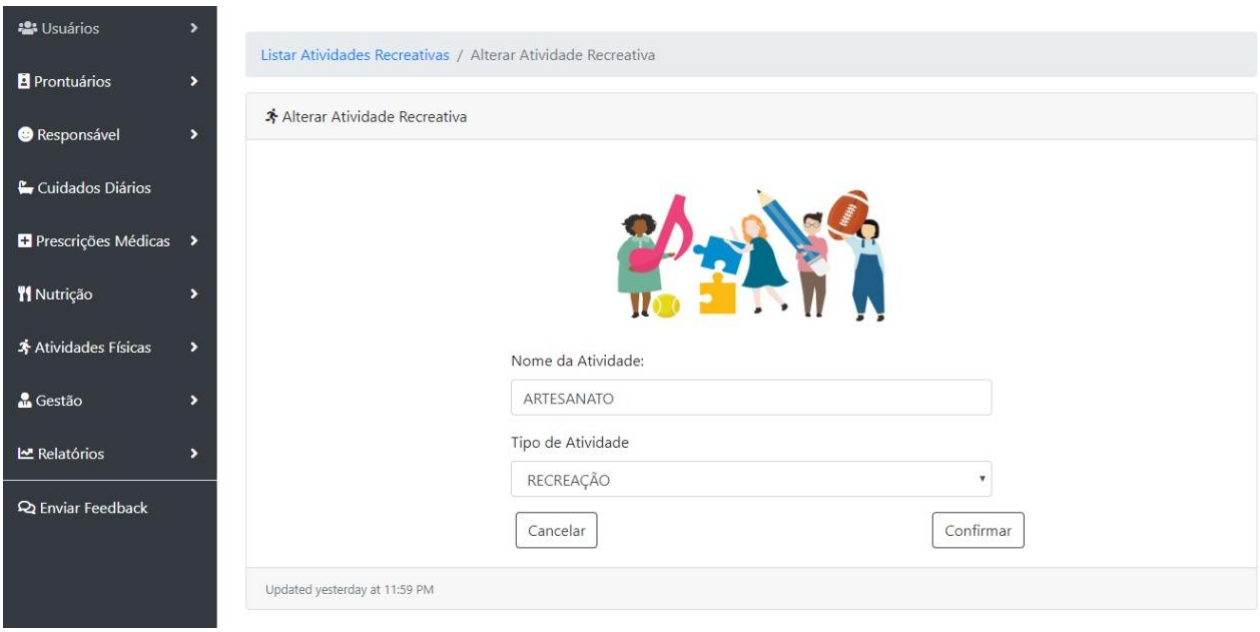

Figura 14 - Interface de alterar atividade recreativa

```
<input type="hidden" name="id" value="<?php echo $atvcomeco->__get('ATR_ID'); ?>" id="idAtv">
<input type="hidden" name="action" value="finish">

<div class="form-row">

    <div class="form-group col-md-3">

    </div>

    <div class="form-group col-md-6">
      <label for="nomeatv" text-align="center" >Nome da Atividade:</label>
      <input type="text" class="form-control" value="<?php echo $atvcomeco->__get('ATR_NOME') ?>" id="nomeatv"
             name="nome" required placeholder="Digite o Nome da Atividade">
    </div>

</div>

<div class="form-row">
    <div class="form-group2 col-md-3">

    </div>

    <div class="form-group col-md-6">
        <label for="inputState">Tipo de Atividade</label>
        <select id="inputState" required name="tipo" class="form-control">
            <?php

            foreach($tipos as $tipo)
            {
                if($tipo->__get('TAR_ID') == $atvcomeco->__get('FK_TIPO_ATIVIDADES_RECREATIVAS_TAR_ID'))
                {
                    echo '<option selected value="' . $tipo->__get('TAR_ID') . '">' . $tipo->__get('TAR_NOME') . '</option>';
                }else{
                    echo '<option value="' . $tipo->__get('TAR_ID') . '">' . $tipo->__get('TAR_NOME') . '</option>';
                }
            }

            ?>
        </select>
    </div>
</div>

<div class="row">
    <div class="form-group2 col-md-3">
    </div>
```

Figura 15 - Código da interface de alterar atividade recreativa

2.2.3 Dificuldades e adaptações durante a refatoração do código

Durante a criação dos protótipos para os requisitos funcionais, é de grande ajuda a reutilização de códigos apresentada pela POO economizando o tempo que levaria a confecção de cada protótipo e o desenvolvimento do produto final.

A refatoração do código é um dos maiores obstáculos do projeto por unir o código de todos os desenvolvedores, mesmo o projeto sendo separado em módulos, para chegar ao objetivo de completa-lo é necessária à união do código de todos. Dado a tarefa nada fácil, pois cada

desenvolvedor possui seu próprio ponto de vista e maneira de programar, porém é necessário superar o desafio como uma equipe para conclusão do projeto gerações.

3 Conclusões e Recomendações

Durante o ano 2019, foi proposto para o quarto ano de informática, do Instituto Federal de Ciências e Tecnologia de São Paulo em São João da Boa Vista, desenvolver um projeto para gerenciamento de institutos de longa permanência. São João da Boa Vista sendo uma das melhores cidades para o desenvolvimento do projeto gerações, por ter sido eleita como a melhor cidade para se viver após os 60 anos. O desenvolvimento do projeto foi separado em 3 subsistemas e esses subsistemas separados em 9 módulos cada um responsável por uma área. Neste documento foi acompanhado o modulo responsável pelo treinamento físico dos idosos, tanto por atividades físicas como por atividades recreativas, e o processo de desenvolvimento dos casos de uso do módulo e explicando brevemente suas funcionalidades. Para melhor organização do trabalho no projeto o modulo 07 levantou os requisitos necessários para a criação do sistema, desenvolvendo os casos de uso em cima dos requisitos funcionais que são necessários para o funcionamento do projeto.

O desenvolvimento dos casos de uso foi separado em iterações para melhor organização do foco do modulo e obter um melhor resultado final.

A confecção deste trabalho leva a ver o quão longe o modulo de treinamento físico conseguiu chegar e a soma de seus esforços para o desenvolvimento do projeto obter êxito, e como a união dos quartos anos consegue completar um projeto enorme como o proposto.

Por gerenciar mal meu tempo não pude detalhar melhor o processo de desenvolvimento do modulo 07, e pouca experiência desenvolvendo projetos desse porte, que acabou me ajudando a melhorar minhas habilidades como desenvolvedor. Recomendo que no futuro, criar módulos menores e com mais união em suas funções, uma das minhas maiores dificuldades com o desenvolvimento no projeto foi a comunicação interna no modulo principalmente de função para função.

Referências Bibliográficas

[1] Prefeitura de São João da Boa Vista. Portal da prefeitura de São João da Boa Vista <http://www.saojoao.sp.gov.br/home/cidade.php?cod=1> Acesso em 22 de agosto de 2019.

[2] Globo noticias <http://g1.globo.com/sp/sao-carlos-regiao/noticia/2017/03/pesquisa-aponta-sao-joao-da-boa-vista-como-melhor-cidade-para-idosos.html> Acesso em 22 de agosto de 2019

[3] Pratica de desenvolvimento de sistemas instituto federal de ciências e tecnologia de são João da boa vista

<https://sbv.ifsp.edu.br/wiki/index.php/Prática_de_Desenvolvimento_de_Sistemas_(PDS)_(Técnico_Integrado_em_Informática)> Acesso em 22 de agosto de 2019

[4] IFSP Campus São João da Boa Vista <https://www.sbv.ifsp.edu.br/sobre-campus> Acesso em 22 de Agosto de 2019.

[5] David J. Barnes, Michael Kolling Programação orientada a objetos com Java. 4° Edição.

[6] MILANI, ANDRE MySQL – Guia do programador. 1° Edição.

[7] DALL'OGLIO, Pablo. PHP: Programando com Orientação a Objetos. 2ª Edição.