Trabalho de Engenharia de Software

Relatório

Ana Júlia Rita Lima Cardoso, George Dutra, Iago Riveiro Santos Dutra, Luan Rodrigues de Carvalho, Luís Henrique Domingues Bueno

Trabalho desenvolvido e apresentado para
Ciência de Dados e Inteligência Artificial

Escola de Matemática Aplicada
Fundação Getúlio Vargas
Brasil
Dezembro 2023

Sumário

1 Casos de Uso e suas descrições

1.1 Caso 1: Fazer login

Fluxo Principal - Caminho Feliz:

1.1 - Abra o aplicativo/site da academia, e visualize a tela de boas vindas;

1.2 - Insira nas caixas de texto respectivamente seu usuário e senha;

1.3 - Clique em "Entrar"(Fim de caso);

Fluxo Alternativo 1.1 -Senha inválida/incorreta

1.1.1 - Visualize mensagem de erro indicando erro da senha;

1.1.2 - Volte a (1.2);

Fluxo Alternativo 1.2 - Usuário não cadastrado:

1.2.1 - Visualize mensagem de erro indicando que o usuário não existe;

1.2.2 - Volte a (1.2);

1.2 Caso 2: Alterar Cargas de Treino

Fluxo Principal - Caminho Feliz:

2.1 - Abra o sistema da academia (já logado);

2.2 - Selecione a opção "Meus Treinos"na área do menu;

2.3 - Toque na aba "Progredir Cargas";

2.4 - Selecionando o exercício que quer progredir;

2.5 - Clique em "Progredir"(Fim de caso).

Fluxo Alternativo 2.1 - Não selecionou um exercício:

2.1.1 - Será exibida uma mensagem dizendo para selecionar um exercício;

2.1.2 - Volte a (2.4);

1.3 Caso 3: Criar treinos

Fluxo Principal - Caminho Feliz:

3.1 - Abra o sistema da academia(já logado);

3.2 - Selecione a área de "Treinos"na área do menu;

3.3 - Dentro daquele área, clique em "Novo treino";

3.4 - Selecione o aluno para criar novo treino no canto direito;

3.5 - Escreva os exercícios com suas especificações(peso, séries, repetições)

3.6 - Clique em "Salvar"(Fim de caso).

Fluxo Alternativo 3.1 - Aluno não selecionado:

3.1.1 - Visualize a mensagem de erro e selecione um aluno.

Fluxo Alternativo 3.2 - Campo não preenchido

3.2.1 - Visualize a mensagem de erro e preencha todos os campos

---

1.4 Caso 4: Fazer matrícula

Fluxo Principal - Caminho Feliz:

4.1 - Abra o sistema da academia(já logado);

4.2 - Abra a aba "Matrículas"na área do menu;

4.3 - Clique em "Nova Matrícula";

4.4 - Preencha os campos de nome completo, RG, CPF, número de celular e endereço de residência;

4.5 - Preencha dados médicos;

4.6 - Clique em enviar e matrícula está finalizada(Fim de caso).

Fluxo Alternativo 4.1 - Senha inválida

4.1.1- Caso a senha não tenha o padrão de ao menos uma letra maiúscula, uma minúscula e um número vai aparecer uma mensagem de erro. Redefina a senha.

Fluxo Alternativo 4.2 - Quantidade de números no cadastro menor ou maior que o esperado

4.2.1 - Caso nos campos de CPF, RG e telefone não for colocada a quantidade esperada de dígitos, aparecerá o erro indicando quantos números devem, ser colocados e quantos têm.

1.5 Caso 5: Atualizar Medidas

Fluxo Principal - Caminho Feliz:

5.1 - Abra o sistema da academia (já logado);

5.2 - Na área do menu, selecione "Progresso";

5.3 - Clique no botão "Definir medidas";

5.4 - Preencha os campos de registro;

5.5 - Clique em "Confirmar";

5.6 - Visualize a nova medida na aba "Visualizar medidas"em "Progresso"(Fim do caso).

Fluxo Alternativo 5.1 - Campos não preenchidos:

5.1.1 - Caso algum dos campos não estiver preenchido, aparecerá a mensagem de erros para preencher todos os campos.

5.1.2 - Volte a (5.4).

1.6 Caso 6: Cadastrar Novas Aulas

Fluxo Principal - Caminho Feliz:

6.1 - Abra o sistema da academia (já logado);

6.2 - Na área do menu, selecione "Aulas";

6.3 - Na aba "Aulas"serão exibidas todas as aulas e datas registradas, clique em "Nova aula";

6.4 - Coloque o nome da aula, a data, a descrição da aula e o número máximo de alunos;

6.5 - Clique em "Criar".

6.6 - Visualize a aula.

Fluxo Alternativo 6.1 - Campos não preenchidos:

6.1.1 - Caso algum dos campos não estiver preenchido, aparecerá a mensagem de erros para preencher.

6.1.2 - Volte a (6.4).

1.7 Caso 7: Solicitar troca de treino

Fluxo Principal - Caminho Feliz:

7.1 - Abra o sistema da academia (já logado);

7.2 - Selecione a aba de "Meus Treinos"na área do menu;

7.3 - Selecione "Solicitar Alteração";

7.4 - Na caixa de texto que surge, escolha o motivo de troca de treino e escreva o detalhamento do pedido com alguma exigência ou limitação;

7.5 - Após revisar a solicitação, clique em "Enviar Solicitação"(Fim de Caso).

Fluxo Alternativo 7.1 - Motivo não selecionado

7.1.1 - Caso não tenha selecionado um motivo, o erro vai aparecer para que possa selecionar seu motivo.

7.1.2 - Volte a (7.4).

1.8 Caso 8: Cadastrar forma de pagamento

Fluxo Principal - Caminho Feliz:

8.1 - Abra o sistema da academia (já logado);

8.2 - Abra a aba "Perfil"na área do menu;

8.3 - Clique em "Registrar Cartão";

8.4 - Preencha as informações pedidas;

8.5 - Clique em "Registrar Cartão"(Fim de Caso).

Fluxo Alternativo 8.1 - Campos não preenchidos:

8.1.1 - Caso algum dos campos não estiver preenchido, aparecerá a mensagem de erros para preencher.

8.1.2 - Volte a (8.4).

1.9 Caso 9: Marcar aulas coletivas

Fluxo Principal - Caminho Feliz:

9.1 - Abra o sistema da academia (já logado);

9.2 - Seleciona "Aulas"na área do menu;

9.3 - Seleciona "Nova Aula"para poder se inscrever em uma nova aula;

9.4 - Selecione uma aula disponível dentre as apresentadas no dropdown;

9.5 - Veja suas especificações e clique em "Entrar";

9.6 - Visualize se entrou na aula(Fim de Caso);

Fluxo Alternativo 9.2 - Já inscrito

9.2.1 - Caso o aluno já esteja inscrito na aula, aparece a mensagem de erro, dizendo que ele já está matriculado na aula.

- Volte ao ponto 9.4 para selecionar outra aula;
- Clique em "Cancelar"caso não queira se inscrever em mais nada.

1.10 Caso 10: Definir metas

Fluxo Principal - Caminho Feliz:

10.1 - Abra o sistema da academia (já logado);

10.2 - Seleciona "Progresso"na área do menu;

10.3 - O aluno seleciona "Definir metas";

10.4 - O aluno preenche os campos de metas de treino necessárias;

10.5 - Clique em "confirmar";

10.6 - Todas as metas aparecerão numa aba de visualização para o aluno verificar depois(Fim de Caso).

Fluxo Alternativo 10.1 - Campos obrigatórios não preenchidos

10.1.2 - Caso o aluno não tenha preenchido todos os campos obrigatórios, aparecerá o erro, indicando para o aluno que ele deve preencher os campos obrigatórios.

10.1.2 - Volte a (10.4).

2 Estórias de usuário

2.1 Estória 1:

Como usuário(professor ou aluno), preciso ser capaz de fazer meu login no sistema para acessar minha respectiva página.

2.2 Estória 2:

Alterar cargas: Como aluno, preciso ser capaz de atualizar as cargas de peso de treino no aplicativo da academia para acompanhar meu progresso e atualizar o nível de dificuldade dos meus exercícios.

2.3 Estória 3:

Criar Treinos: Como professor, é preciso poder criar os treinos dos alunos para acompanharem os métodos adequados a eles.

2.4 Estória 4:

Fazer Matrícula: Como professor, preciso fazer a matrícula dos alunos, colocando suas limitações e preocupações, para que o contrato com a academia seja adequado às suas necessidades.

2.5 Estória 5:

Atualizar Medidas: Como aluno, gostaria de atualizar minhas medidas pessoais (peso, medidas e etc.) para acompanhar meu progresso nas metas estabelecidas.

2.6 Estória 6:

Criar Aulas: Como professor, preciso criar novas aulas (boxe, dança, etc.) para disponibilizar a todos os alunos as aulas fornecidas pela academia.

2.7 Estória 7:

Solicitar troca de treino: Como um aluno, gostaria de solicitar mudanças em meu plano de treinamento para meu professor, para assim poder evoluir ou adaptá-lo a uma lesão.

2.8 Estória 8:

Cadastrar forma de pagamento: Como aluno, gostaria de poder cadastrar uma nova forma de pagamento para pagar a academia e para não precisar solicitar isso ao atendente em caso de bloqueio de cartão.

2.9 Estória 9:

Marcar aulas: Como um aluno, preciso ser capaz de entrar em uma aula para garantir minha participação e uso dos recursos disponíveis da academia.

2.10 Estória 10:

Definir metas: Como um aluno, preciso ser capaz de definir minhas metas de treino, para que eu possa acompanhar meu progresso.

3 Metáfora do Sistema

O sistema virtual de uma academia pode ser comparado a uma "Cozinha de Restaurante Fitness". Nesta metáfora, o sistema virtual assume o papel dos chefes e cozinheiros que preparam pratos saudáveis para os clientes, explicando mais sobre o assunto, podemos dividir em algumas categorias, sendo elas:

Chefes Virtuais: O sistema virtual é como o chef de cozinha do restaurante em questão. Ele é o encarregado de criar receitas de exercícios, planos de treinamento e rotinas personalizadas para cada cliente, no caso, os alunos.

Cozinheiros Virtuais: Os "cozinheiros"representam os diferentes componentes do sistema virtual, como programas de treinamento, metas estabelecidas e aulas oferecidas, além de também podermos compreender como os personais da academia virtual. Eles trabalham juntos para fornecer um serviço completo aos clientes.

Clientes Famintos por Saúde: Os alunos da academia são como os clientes famintos por saúde e condicionamento físico. Eles chegam à "cozinha"em busca de refeições de exercícios e aulas bem preparadas para atender às suas necessidades específicas.

Cardápio de Exercícios: O cardápio da academia é composto por exercícios variados e programas de treinamento. Os chefes e cozinheiros virtuais ajudam os clientes a escolher os pratos (exercícios) adequados para seus objetivos. Mais além, podemos entender que as metas criadas pelos alunos sejam suas comidas favoritas e nossa cozinha fará o possível para servi-la da melhor forma possível.

Pratos Personalizados: Assim como um chef personaliza pratos com base nas preferências do cliente, o sistema virtual personaliza rotinas de exercícios com base nas metas e capacidades de cada cliente como citado anteriormente, levando em questão, o peso, medida, tudo relacionado a metas e progresso do cliente.

Serviço de Mesa e Feedback: Os garçons e atendentes representam o sistema virtual que fornece feedback e orientação aos clientes durante seus treinos. Eles se asseguram de que os pratos sejam servidos da maneira correta e de que os clientes tenham uma experiência satisfatória.

Qualidade e Sabor: A qualidade dos exercícios e a eficácia do treinamento são comparáveis ao sabor dos pratos em um restaurante. O sistema virtual garante que os exercícios e aulas sejam eficazes e agradáveis para cada cliente, levando em consideração suas limitações e preocupações, que até poderiam ser interpretadas como alergias e desgostos.

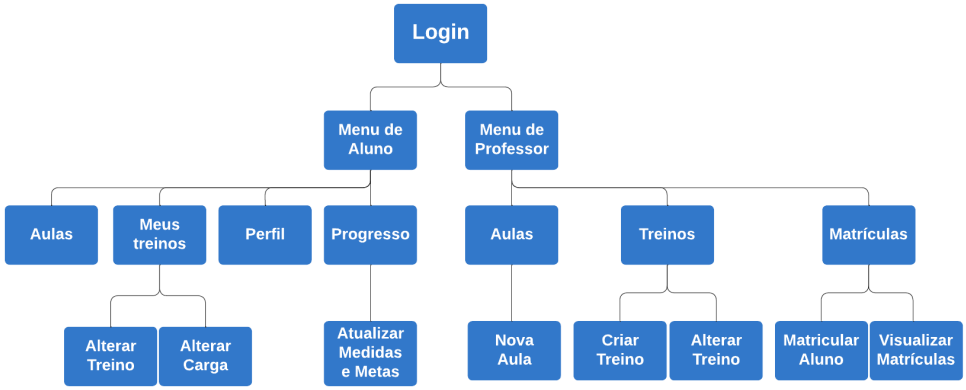
Nesta metáfora, a academia é vista como um restaurante fitness onde os clientes recebem pratos personalizados de exercícios preparados por chefes e cozinheiros virtuais. O sistema virtual desempenha o papel central na criação de experiências de treinamento deliciosas e saudáveis para os frequentadores famintos por condicionamento físico.

4 Estrutura e Desenvolvimento de Código

O desenvolvimento em Python do projeto ocorreu em um repositório GitHub criado e acessado pelos integrantes do grupo. O repositório segue um padrão estrutural comum em projetos Python, centrado em dois diretórios principais: *src* e *test*.

4.1 Planejamento

No início do projeto, buscamos planejar o funcionamento do sistema e dividir tarefas. Com o decorrer das discussões e estudando melhor o módulo *TkInter*, percebemos que seria útil modelar o sistema seguindo a ideia de um padrão de projeto Composite. A ideia principal é que a raiz do projeto comece na tela de login, e dali se divida em diferentes telas que levam a diferentes partes do sistema. No fim, as janelas do sistema ficaram com a seguinte forma:



4.2 Estrutura

Na pasta raiz, vemos alguns diretórios auxiliares para processos do sistema, como *img* para guardar imagens relevantes para o visual do programa, e *relatorio*, onde estão guardados diagramas e relatorios do trabalho.

No diretório *src*, é possível ver as sub-pastas que agrupam os principais módulos do sistema, e um arquivo *main.py* que representa o núcleo do sistema e por onde o mesmo pode ser inicializado. É possível reparar que os sub-diretórios seguem um raciocínio parecido com o resultado do planejamento mostrado acima. Eles armazenam as seguintes divisões do sistema:

- *db*: Módulo de acesso ao banco de dados relacional do sistema;
- *exc*: Arquivo de erros e exceções especiais levantadas pelo sistema;
- *gui*: Módulo gráfico do sistema, armazenam padrões visuais e classes template para serem utilizadas pelas diferentes telas do programa;
- *login*: Módulo de login, que exibe a tela de login, recebe e valida as informações do usuário, e acessa o sistema;
- *student*: Módulo de processos realizados pelo aluno, a partir da tela de Menu do Aluno;
- *teacher*: Módulo de processos realizados pelo professor, a partir da tela de Menu do Professor.

Por fim, no diretório *test*, é possível encontrar os arquivos de testes unitários para validar o funcionamento de algumas unidades essenciais do sistema.

4.3 Arquivos e Processos

Nessa seção, discutiremos os arquivos presentes na pasta *src* dentro do diretório raiz.

No arquivo *main* foi desenvolvida a solução para o primeiro desafio do projeto: devido a modelagem do sistema centrado em um banco de dados relacional, se mostrou necessário que de qualquer ponto do programa, fosse possível criar um acesso ao banco de dados. Em outros projetos, quando trabalhamos com *pygame* por exemplo, vimos algumas soluções para este tipo de problema, como por exemplo passar instâncias de classes como parâmetros umas para as outras, de forma a fazer com que todo o sistema "se conheça". Neste projeto porém, conhecemos uma solução muito mais versátil e eficiente: o uso do padrão de projeto Singleton. Com ele, criamos uma classe Singleton chamada System, que armazena em si o acesso ao banco de dados. Se, em qualquer ponto do código, tentarmos criar uma instância de System, estaremos acessando a instância criada no *boot* do sistema, e por ela, acessando o banco de dados. Como todo o aplicativo é exibido ao usuário através de uma única janela do *TkInter*, utilizamos a System também para armazenar a instância da classe principal Window, assim como armazenamos nela o ID do usuário logado. Além disso, o arquivo main.py serve como ponto de partida do sistema, ou seja, é executando-o que iniciamos corretamente o processo de *boot* do programa.

Dentro das sub-pastas de *src*, temos diversos arquivos que desempenham papéis diferentes. As pastas *db*, *gui*, e *exc* já foram bem explicadas na seção Estrutura, mas os arquivos das pastas *login*, *teacher* e *student* merecem mais da nossa atenção para serem explicados.

Todos os arquivos nestes módulos seguem aproximadamente um padrão de três arquivos: arquivos *Frame*, arquivos *Factory* e arquivo de processos.

Frames são objetos gráficos plotados dentro da classe principal Window do *TkInter*, que está armazenada na Singleton System no núcleo do sistema. Tudo que plotamos nas Frames é exibido ao usuário, e muitos Widgets das Frames são usados para interagir com ele, como as caixas de texto, botões e afins. Toda vez que mudamos de uma tela para outra (por exemplo, da tela de login para o menu), precisamos destruir tudo de uma frame para começar a próxima de forma a evitar vazamento de memória. Observando esse comportamento de destruir e construir objetos constantemente, observamos que seria mais eficiente e consistente aplicar o padrão de projeto Factory.

Portanto sempre que queremos mudar de tela, pedimos à uma classe *frame_factory* para que retorne a Frame específica para aquela situação. Um exemplo disso é o arquivo *student_frame_factory.py*, que define a classe que retorna todas as frames que vemos ao fazer login como estudante.

Essas Frames são definidas uma a uma em seus respectivos arquivos. Estes arquivos tem como foco implementar somente a lógica de design das Frames, como botões, caixas de texto e mensagens de erro, e boa parte dessas Frames e desses Widgets derivam de classes Template, definidas no diretório *gui*. O arquivo *login_frame.py* define a aparência da Frame que incorpora a tela de login que vemos ao iniciar o programa.

Quando inserimos uma informação ou apertamos um botão em uma tela, é executado um comando no sistema. Quando esse comando envolve o restante do sistema e outros processos, incorporamos o processo num arquivo próprio. Um exemplo disso é o arquivo *login_class.py*, que implementa a classe Login que recebe os dados inseridos

pelo usuário, valida eles (levantando exceções caso estejam incorretos), e indica para o sistema qual tela chamar após o processo de login terminar.

4.4 Padrões de Projeto

Por fim, podemos identificar ao todo o uso de 6 padrões de projeto no sistema da academia Chi-Trapézio:

4.4.1 Singleton

Como já foi explicado, a classe *System* definida no arquivo *main.py* é um Singleton, ou seja, possui uma única instância global que pode ser acessada de qualquer local do sistema, com o objetivo de guardar dados importantes e facilitar o acesso ao banco de dados.

4.4.2 Factory

Já explicamos também que os arquivos *frame_factory* retornam instâncias das diferentes Frames utilizadas no programa.

4.4.3 Composite

No início da seção de planejamento explicamos como o modo de pensar em telas e menus nos levou a desenvolver um sistema parecido com uma árvore de composições, que leva o usuário em diferentes níveis de acesso de acordo com qual caminho ele seleciona nos menus.

4.4.4 Facade

A interface gráfica do sistema, implementada através do TkInter, implementa diversas formas de visualizar, editar e adicionar dados no banco de dados relacional do programa. Ela faz isso de forma visual e simplificada, e como manda o padrão de projeto Facade, facilita o acesso às partes mais complexas do sistema.

4.4.5 Template

Como explicamos anteriormente, o diretório *gui* armazena diversos Templates de classes utilizadas em todas as Frames do projeto, como botões, textos e afins. Esses templates facilitam etapas como a padronização de cores, fontes e formatos do aplicativo.

4.4.6 Mediator

O padrão Mediator foi particularmente útil na interação entre as ações do usuário e o banco de dados relacional. Como o nosso BD foi implementado numa VM na nuvem, foi necessário criar uma conexão SSH direta ao terminal da VM, pelo qual enviamos as queries e recebemos os resultados.

A classe DBConnector agiu como mediador entre os inputs do usuário e o banco de dados, convertendo comandos em queries válidas e retornos do SQL em tabelas fáceis de visualizar.