Desenvolvimento de uma aplicação Java em integração com MySQL para o uso de uma academia

1st Douglas Kenji Kihara, 2nd Gabriel Castagna Henrique, 3rd Arthur de Souza Spironello
Universidade Franciscana (UFN), Santa Maria - RS
1st douglas.kihara@ufn.edu.br, 2st gabriel.castagna@ufn.edu.br, 3st arthur.spironello@ufn.edu.br

***Resumo*—Este artigo faz parte de um projeto extensionista visando desenvolver um produto para a comunidade, foi elaborado em conjunto com a disciplina de Linguagem de Programação II e Banco de Dados II com vistas de desenvolver um software para auxiliar no controle de uma academia. Foi utilizada a linguagem Java na versão Desktop, sendo que os dados foram armazenados no banco de dados MySQL. O trabalho se iniciou com análise do processo de preenchimento em fichas de treino da academia.**

***Abstract*—This article is part of an extension project aimed at developing a product for the community, it was prepared together with the discipline of Programming Language II and Database II with a view to developing software to assist in the control of a gym. The Java language was used in the Desktop version, and the data were stored in the MySQL database. The work began with an analysis of the process of filling out training sheets at the academy.**

I. Introdução

De acordo com o Instituto Brasileiro de Geografia e Estatística (IBGE), no Brasil, em 2019, cerca de metade dos adultos não atingiram a recomendação mínima, recomendada pela Organização Mundial da Saúde (OMS), da prática de atividade física. A não realização dessas atividades, esteve relacionada com mais de 800 mil óbitos no mundo. Quando comparado a outros países, o Brasil apresenta uma das piores taxas de ociosidade do mundo.

Neste cenário, esta pesquisa se insere e, portanto, para ajudar a comunidade foi desenvolvido sobre o aspecto atividade física, uma aplicação direcionada a uma Academia, com o intuito de auxiliar na organização de seus clientes, funcionários, treinos, exercícios, equipamentos e frequência de seus alunos, proporcionando uma melhor organização neste ambiente e em seus praticantes.

Para o desenvolvimento desta aplicação foi utilizada a linguagem de programação Java, a qual é orientada a objetos e integrada ao ambiente de desenvolvimento NetBeans. A linguagem de banco de dados escolhida foi o MySQL e o Sistema Gerenciador de Banco de Dados (SGBD) utilizado foi o MySQL *Workbench*. A construção dos diagramas de banco de dados foi feita no programa brModelo. Ao final, com o programa elaborado, o usuário poderá realizar o controle de clientes, funcionários, equipamentos, exercícios e treinos, permitindo o controle de frequência de seus alunos.

II. Fundamentação Teórica

Originalmente, a linguagem Java foi desenvolvida na primeira metade da década de 90 nos laboratórios da Sun Microsystems tendo como objetivo ser mais simples e eficiente que suas predecessoras. Após adaptações em seus códigos, a linguagem passou a ser amplamente usada na construção de documentos web que permitam maior interatividade. Não somente isso, a linguagem permite aplicações com acesso remoto a banco de dados, banco de dados distribuídos, iteratividade em páginas *World Wide Web* (WWW).

Java é uma linguagem simples, de fácil aprendizado, possui uma grande quantidade de bibliotecas que contribuem para a funcionalidade básica, incluindo acesso à rede e a criação de interfaces gráficas. É baseada na Orientação a Objetos, sendo o encapsulamento em um bloco de *software* dos dados e métodos de manipulação, a linguagem possibilita a modularização das aplicações, reuso e manutenção do código já implementado.

A IDE utilizada para construção do *software* proposto foi NetBeans, pois é um ambiente de código aberto e gratuito feito para para escrever, compilar, depurar e implantar programas, neste caso, suportando a linguagem Java.

O projeto foi construído utilizando o padrão *Model-View-Controller* (MVC) que é comumente usados em projetos, pois a implementação de cada uma destas camadas proporciona uma melhora na manutenção, na identificação de possíveis erros, uniformidade na estrutura do *software*, estabelece um vocabulário comum de projeto entre os desenvolvedores, o possível reaproveitamento de classes e partes do projeto, entre diversos benefícios.

O MVC pode ser dividido em três elementos: o **Controller** que interpreta as entradas do mouse ou teclado enviados pelo usuário, mapeando essas ações que são enviadas para o *Model* e para a janela de visualização (*View*) sendo efetuada a alteração apropriada. O **Model** fica responsável por gerenciar um ou mais elementos de dados, saber o que o aplicativo quer fazer, sendo a principal estrutura computacional da arquitetura, pois é ele que modela o problema a ser resolvido. E por fim, a **View** responsável por representar as interações visuais com o usuário, em uma combinação com gráficos e textos.

A brModelo é uma ferramenta de apoio aos projetos BDs relacionais desenvolvida pelo Grupo de BD da UFSC. Esta ferramenta permite dar suporte a todas as etapas tradicionais de um projeto BD: conceitual, lógica e física; auxilia na tomada

de decisões e dá suporte a todos os conceitos do modelo entidade-relacionamento.

O MySQL é um SGBD multiusuário e multitarefa que utiliza a linguagem SQL (Structure Query Language), que é a linguagem mais popular para inserir, acessar e gerenciar um conteúdo armazenado em um banco de dados. É distribuído e suportado pela Oracle Corporation.

Para a conexão entre a aplicação Java e o MySQL é necessário um driver chamado ”mysql-connector-java”, esse driver é o que permite ser possível as duas linguagens se comunicarem.

III. Desenvolvimento

A. Diagramas

Este trabalho teve seu início na construção dos diagramas conceitual e lógico da academia com base nas fichas de treinos disponibilizadas pela própria academia. As entidades criadas foram: Frequência, Equipamento, Exercícios, Cliente, Funcionário e as entidades relacionamento criadas foram entre Equipamento/Exercícios e Cliente,Funcionário/Exercícios. Vale ressaltar, que, foi criada uma entidade Pessoa, na qual a entidade Cliente e entidade Funcionário herdam os seus atributos, essa característica ficou visível na geração do modelo lógico.

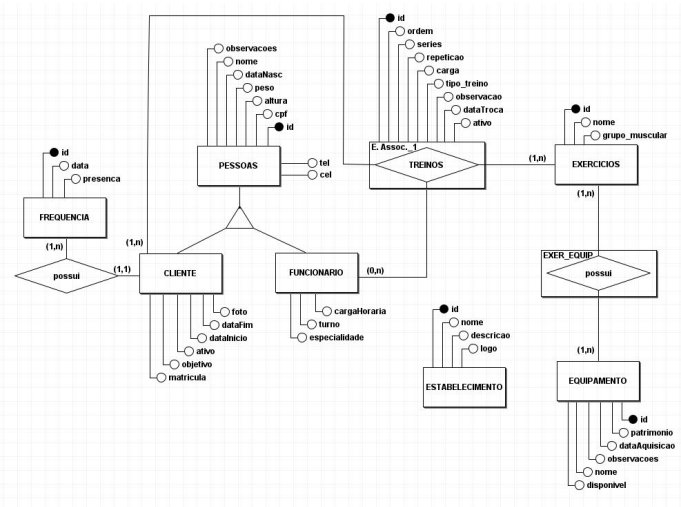

Figura 1: Modelo Entidade Relacionamento

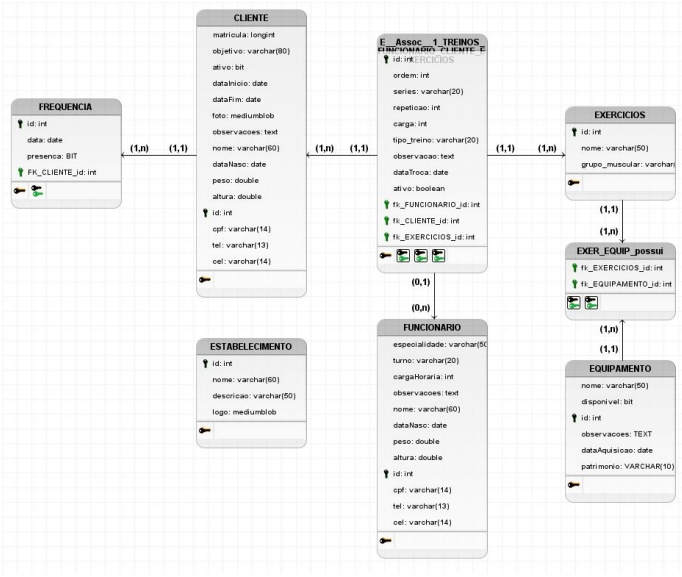

Figura 2: Modelo Lógico

B. Banco de Dados

Após a realização dos diagramas, foi dado início a construção do banco de dados no MySQL *WorkBench*. Primeiro passo foram as criações das tabelas e seus atributos, finalizada a criação de tabelas foi necessário confeccionar *Stored Procedures, Triggers e View*.

Stored Procedure, que traduzido significa Procedimento Armazenado, é um conjunto de comandos em SQL que podem ser executados como um sub rotina. Permitem centralizar a lógica de acesso aos dados em único local, otimizando o código e facilitando na manutenção. O *Procedure* criado para esta aplicação foi especificamente para gerar automaticamente uma matrícula ao cliente assim que inserido na tabela, neste foi utilizado a função LPAD(string, length, lpad_string) que preenche à esquerda uma string com outra string, até um certo comprimento definido. A estrutura do número da matricula foi pensado da seguinte forma, onze números, sendo os 4 primeiros o ano atual e os restantes são números gerados pela função LPDA(1,6,0). Um exemplo de resultado gerado é 20220000001.

Um *Trigger*, ou gatilho, é um tipo especial de *Stored Prodecure* que é acionado automaticamente sempre que o usuário executar alguma operação de modificação de dados em uma tabela especificada. As ações para o *trigger* disparar podem ser uma inserção, uma exclusão ou uma alteração de dados. No caso desta aplicação, os Triggers foram criados para impedir o cadastramento de clientes ou funcionários duplicados com base no nome e na data de nascimento deles, o trigger verifica se já existe cadastrado no banco os mesmos dados, caso existir uma mensagem é mostrada como ”Registro Existente”não permitindo a inclusão do cliente ou funcionário.

A View, em banco de dados, é uma tabela virtual baseada no conjunto de resultados de uma consulta SQL, mostra sempre os resultados dos dados atualizados, pois a view é recriada toda a vez que o usuário a consulta. A View foi criada especificamente para listar a frequência dos clientes na academia.

C. Java

1) ConnectionFactory: Primeiramente, para que a conexão entre o Java e o Banco de Dados seja possível, foi criado um pacote contendo uma classe com um método responsável por fazer esta conexão, chamado de ConnectionFactory e seu método getConnection(), nessa classe os atributos existentes são: o user, a password e a url. Que devem estar definidos com os valores correspondentes ao usuário, a senha e o caminho para acessar o banco de dados.

2) Model: Feita a conexão com o banco, o próximo passo foi criar o package Model, que para cada tabela existente, uma classe foi criada com os atributos privados e os seus métodos *getters* públicos, de modo que a consulta a esses atributos fossem seguras atráves dos *getters*.

Também, foi criado as classes DAO (*Data Acess Object*) de cada tabela. Uma classe DAO é responsável por trocar informações com o banco de dados e fornecer operações CRUD (*Create Read Update e Delete*) e de pesquisas, esta deve ser capaz de buscar dados no banco e transformar esses objetos em lista de objetos, usando listas genéricas ou não, também deverão receber os objetos para assim consigam converter em instruções SQL e mandar para o banco de dados.

Este padrão foi escolhido, pois deste modo toda interação com a base de dados se dará através destas classes, logo é colocado um nível de abstração no modo de busca e gravação de dados, tornando isso transparente para aplicação e facilitando muito na hora de fazer uma manutenção ou migração de banco de dados. Para a realização destas operações fez-se necessário a utilização da dependência ”mysql-connector-java”.

3) View: No package View encontra-se tudo que é relacionado a interfaces, para a construção das interfaces foi evitado o máximo de telas, mantendo tudo que era necessário em somente uma tela. O método mais eficiente para essa proposta foi a utilização do componente Tabbed Pane em conjunto com a classe CardLayout().

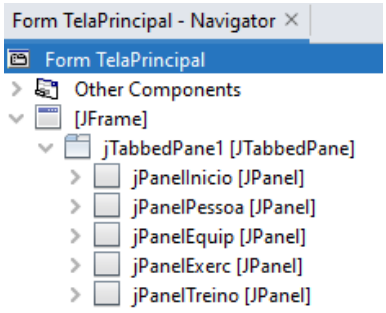

Figura 3: Visualização do JTabbedPane

Tabbed Pane como mostra a figura 3, permite a interação de vários documentos ou painéis dentro de uma única janela. Isto foi usado como uma espécie de menu, cada entidade do banco de dados possuía um painel. Separados desta forma, ainda, havia as telas de cadastro, visualização, edição e exclusão dos dados, portanto o uso do CardLayout foi essencial nesta aplicação, afim de ter evitado a construção de telas para cada evento a ser realizado.

O CardLayout permite a interação de vários componentes, esses chamados de *cards*, nos quais, geralmente são painéis, adicionados dentro de um *container*, os painéis são dispostos como se fossem um baralho, um sobre o outro, onde cada painel possui uma posição e é exibido um de cada vez. Desta forma, havia um *card* para cada evento, estes *cards* ficaram dispostos dentro de cada JPanel das classes criadas, como é mostrado na Figura 4, sendo um *card* para cadastro, visualização e exclusão. Porém, para o evento da edição foi criado uma tela *modal*, com o intuito de dar uma outra impressão ao usuário, quando o mesmo estivesse editando os dados, diferenciando, a tela de editar da tela de cadastro, evitando das mesmas ficarem na mesma tela.

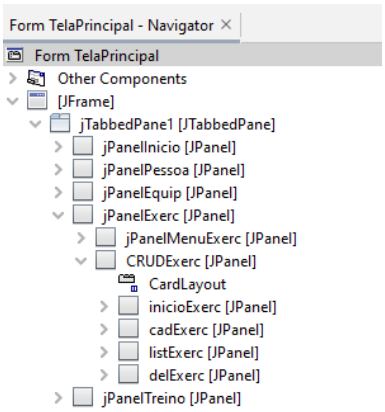

Figura 4: Integração do JTabbedPane com o CardLayout em conjunto com JPanel

4) Controller: No *package Controller* se encontra toda a parte relacionada a administração e processamento das requisições feitas em cada tela, ademais gerar respostas para elas. Para maior organização possível cada tela possui sua *controller* que utiliza tanto das classes do *packge Model* como do *package View* para efetuar as operações e aplicar as funções aos botões da tela que está se refere. Além destas *controllers*, existe a classe App.java que faz a distribuição das telas para as *controllers* devidas e também Util.java que dispõe funções que são usadas em todas as outras classes.

Conclusão

Com tudo que foi realizado neste projeto, é possível ter um maior entendimento e compreensão em forma prática de como são utilizadas a linguagem Java e o banco de dados relacional SQL para a criação de uma aplicação digital que poderia ser vista no dia-a-dia de um estabelecimento, também é valido salientar a concepção dos métodos e padrões utilizados no mercado como os métodos DAO e MVC. Deste modo, durante o planejamento, desenvolvimento e término do projeto, pudemos ter uma visão mais clara de como é o trabalho e a organização de uma equipe de desenvolvimento de software.

Referências

[1] L.S.Indrusiak, Linguagem Java, Rio Grande do Sul, Grupo Java RS, 1996

[2] https://www.oracle.com/br/tools/technologies/netbeans-ide.html Acesso em: 05/12/2022

[3] R. S Mello, C. H. Cândido, M. B. Neto, Ferramenta brModelo: Quinze Anos!