PROVA DE TÉCNICO DE INFORMÁTICA

ALUNO: JHEIMISON GOMES 29/10/2025

1. R=

O padrão de arquitetura MVC é um modelo de organização de software que separa a aplicação em três camadas principais, com o objetivo de dividir responsabilidades**,** facilitar a manutenção e melhorar a reutilização de código**.**

Model (Modelo)
Função principal: representar os dados e as regras de negócio da aplicação. É responsável por acessar o banco de dados, armazenar informações e implementar a lógica que define como os dados podem ser criados, lidos, atualizados e excluídos.

View (Visão)
Função principal: cuidar da interface com o usuário, ou seja, o que é exibido na tela. A View não contém lógica de negócio; ela apenas mostra os dados vindos do Model, geralmente por meio do Controller.

Controller (Controlador)
Função principal: intermediar a comunicação entre a View e o Model. O Controller recebe as requisições do usuário (como cliques ou envio de formulários), processa essas requisições, chama o Model para obter ou modificar dados e retorna a resposta adequada à View.

2. R=

Manter uma separação clara entre Model, View e Controller é fundamental porque o padrão MVC foi criado justamente para organizar o código, facilitar a manutenção e permitir o reuso e a evolução do sistema sem causar efeitos colaterais em outras partes da aplicação. Quando essa separação não é respeitada, o sistema se torna confuso, difícil de entender e mais suscetível a erros.

Exemplo 1 – Lógica de negócio dentro da View, se o código da interface (por exemplo, HTML com PHP embutido) contiver lógica de negócio, como cálculos de impostos ou regras de desconto, a manutenção se torna difícil. Problema: ao alterar a regra de negócio, é necessário editar várias páginas diferentes. Consequência: alto risco de inconsistências e erros, além de dificuldade em testar a lógica isoladamente.

Exemplo 2 – Controller acessando diretamente o banco de dados se o Controller fizer consultas SQL diretamente, sem passar pelo Model, ele perde o encapsulamento dos dados. Problema: duplicação de código e dificuldade para mudar o banco de dados no futuro. Consequência: qualquer alteração na estrutura do banco exigirá mudanças em vários Controllers.

3. R=

Em uma aplicação JavaFX que segue o padrão MVC, as camadas Controller, Model e View se comunicam de maneira organizada e unidirecional, respeitando as responsabilidades de cada uma.

Comunicação entre Controller e Model**,** essa comunicação ocorre quando o usuário realiza uma ação (como clicar em um botão) e o Controller precisa consultar, atualizar ou manipular os dados da aplicação.

Exemplo: @FXML

```
private void salvarProduto() {

Produto p = new Produto(txtNome.getText(), Double.parseDouble(txtPreco.getText()));

 produtoModel.salvar(p); // Chama o Model para salvar os dados

}
```

Comunicação entre Controller e View, A comunicação entre o Controller e a View ocorre principalmente para atualizar a interface com base nas mudanças de estado do sistema.

Exemplo: @FXML

```
private Label lblMensagem;

@FXML

private void salvarProduto() {

Produto p = new Produto(txtNome.getText(), Double.parseDouble(txtPreco.getText()));

if (produtoModel.salvar(p)) {

lblMensagem.setText("Produto salvo com sucesso!");

} else {

lblMensagem.setText("Erro ao salvar o produto.");

    }
```

```
}
```