**Relatório do Sistema de recomendação.**

No projeto do sistema de recomendação, foram aplicados os paradigmas de programação imperativa, orientada a objetos e funcional. Abaixo, detalharei como cada paradigma foi utilizado e como eles facilitam diferentes aspectos do desenvolvimento de software:

**1. Programação Imperativa:**

* **Utilização:** A programação imperativa é utilizada em partes do código onde é necessário definir explicitamente a sequência de passos para alcançar um resultado. Por exemplo, no menu principal do sistema, onde há uma sequência de opções e um loop para solicitar a entrada do usuário.
* **Controle explícito:** Este paradigma oferece controle direto sobre o fluxo de execução do programa, facilitando a compreensão e o gerenciamento do código.
* **Sequencialidade:** A execução das instruções ocorre em ordem sequencial, o que simplifica a lógica do programa em certos contextos.

**2. Programação Orientada a Objetos (POO):**

* **Utilização:** A POO é aplicada na modelagem do sistema, onde as entidades do mundo real são representadas como objetos que possuem atributos e comportamentos.
* **Encapsulamento:** Os dados são encapsulados dentro dos objetos, o que promove a modularidade e a segurança do código.
* **Reutilização de código:** Através do conceito de herança e polimorfismo, é possível reutilizar e estender o código de forma eficiente.
* **Abstração:** Permite representar conceitos complexos do mundo real de forma mais clara e organizada, simplificando o desenvolvimento e manutenção do software.
* **Mensagens entre objetos:** Os objetos se comunicam enviando mensagens uns aos outros, o que facilita a interação e colaboração entre diferentes partes do sistema.

**3. Programação Funcional:**

* **Utilização:** A programação funcional é utilizada em certas operações de processamento de dados, como cálculos estatísticos ou transformações de dados.
* **Imutabilidade:** Dados imutáveis facilitam a compreensão do código e reduzem a possibilidade de efeitos colaterais.
* **Funções de primeira classe:** Funções podem ser tratadas como qualquer outra variável, permitindo passá-las como argumentos e retorná-las como resultados de outras funções.
* **Composição de funções:** Permite combinar funções simples para construir operações mais complexas, o que torna o código mais conciso e legível.
* **Recursão:** Facilita a resolução de problemas que podem ser divididos em subproblemas menores e iguais, simplificando a implementação de algoritmos complexos.
* **Legibilidade e manutenção:** A POO e a programação funcional, através da abstração e da composição de funções, respectivamente, tornam o código mais legível e fácil de manter, facilitando a identificação de bugs e a implementação de novos recursos.
* **Reusabilidade:** A POO permite a reutilização de código através da herança e do polimorfismo, enquanto a programação funcional incentiva a criação de funções modulares e genéricas, que podem ser facilmente reutilizadas em diferentes partes do sistema.
* **Gerenciamento de estado:** A imutabilidade de dados na programação funcional ajuda a evitar efeitos colaterais inesperados e simplifica o gerenciamento de estado, tornando o código mais previsível e fácil de depurar.
* **Expressividade e flexibilidade:** A combinação dos paradigmas permite uma expressividade e flexibilidade maior no desenvolvimento de software, onde diferentes abordagens podem ser adotadas conforme a necessidade específica de cada parte do sistema.

**4. Testes Unitários:**

Os testes unitários foram escritos para garantir a correção das funcionalidades do sistema de recomendação. Eles cobrem diferentes aspectos do sistema e são executados de forma automatizada para verificar se o sistema se comporta conforme o esperado em diferentes cenários.

* **Fixture Sistema:** Foi criada uma fixture para criar uma instância do SistemaRecomendacao que será usada em vários testes. Isso ajuda a evitar a repetição de código e mantém os testes mais organizados.
* **Teste para filme\_bem\_avaliado:** Este teste é parametrizado para avaliar se a função filme\_bem\_avaliado funciona corretamente em diferentes cenários. Ele verifica se o sistema identifica corretamente se um filme foi bem avaliado, considerando uma variedade de casos, como quando mais de 50% das avaliações têm 4 ou mais estrelas e quando menos de 50% das avaliações têm 4 ou mais estrelas.
* **Teste para calcular\_media\_recomendacao:** Este teste verifica se a função calcular\_media\_recomendacao retorna a média correta das notas dos filmes no conjunto de dados. Isso é importante para garantir que a função de cálculo da média esteja correta e que as recomendações sejam baseadas em valores precisos.
* **Teste adicional para filme\_bem\_avaliado:** Este teste verifica se a função filme\_bem\_avaliado funciona corretamente ao calcular a proporção de avaliações bem avaliadas em relação ao total de avaliações. Ele testa se o sistema identifica corretamente se um filme foi bem avaliado com base na proporção de avaliações com 4 ou mais estrelas em relação ao total de avaliações.

Em resumo, a aplicação dos paradigmas de programação imperativa, orientada a objetos e funcional no projeto do sistema de recomendação permite alcançar uma estrutura modular, legível, reutilizável e flexível, facilitando o desenvolvimento, manutenção e evolução do software. Cada paradigma traz suas próprias vantagens e, quando aplicados de forma combinada e equilibrada, podem melhorar significativamente a qualidade e eficiência do código.