O que cada membro da equipe trabalhou:

Renan: Foi feita a pesquisa sobre o paradigma Funcional, para fazer a comparação em qual paradigma escolher e após feita as escolhas, foi estudado os 2 paradigmas escolhidos: POO e Imperativa e com isso foi aprofundado sobre as questões de complexidade. Eu consegui ampliar mais meus conhecimentos sobre esses paradigmas em específicos, assim podendo utilizar mais em meus projetos, além de ver um pouco sobre o paradigma funcional.

Pablo: Pesquisa de informações na internet, levantamento de tema, estruturação dos slides, separação de processos e apresentações

Eduardo: Levantamento sobre o que é o paradigma na programaçã,

Vinicius: Pesquisa sobre o paradigma OO e realização de alguns slides. Aprendi mais aprofundamento sobre o paradigma OO, onde irá me ajudar futuramente

Helon: Fiz a conclusão sobre a pesquisa referente aos paradigmas, trazendo questões como quando usar ou não tal paradigma, o impacto deles no software a ser desenvolvido, o impacto na equipe e sobre as linguagens modernas em sua maioria possuírem suporte multiparadigma. Com este trabalho, ficou mais claro a diferença entre os paradigmas na minha mente, pois o único do qual eu tinha uma profundidade maior de conhecimento era o de programação orientada a objetos (POO), e neste trabalho tive a oportunidade de entender mais sobre o imperativo durante as pesquisas, e sobre o funcional durante a mostra de trabalhos dos colegas de classe.

Enois - VSCode: Versionamento de Código e Terminal

1. Módulo de Versionamento de Código:

- **Descrição:** O VSCode integra com sistemas de controle de versão, como Git, diretamente no editor.
- **Funções principais:**
    - Visualização de alterações no código.
    - Comparação de versões de arquivos.
    - Commit, push e pull de alterações.
- **Vantagens:** Facilita o controle de versões, colaboração entre equipes e gerenciamento de histórico de código sem sair do editor.

2. Módulo Terminal:

- **Descrição:** O VSCode possui um terminal embutido, permitindo rodar comandos diretamente dentro do editor.
- **Funções principais:**
    - Execução de scripts e comandos sem precisar sair da IDE.
    - Suporte a múltiplos terminais simultâneos.
    - Integrado com o sistema operacional (Linux, macOS, Windows).
- **Vantagens:** Agiliza o fluxo de trabalho, pois permite a execução de tarefas de terminal e código em um único ambiente.

Observações sobre os Paradigmas no VSCode:

- **Imperativo:** O VSCode facilita a escrita de código sequencial e direto, com fácil integração ao terminal para rodar comandos simples.
- **POO:** Com extensões como o IntelliSense, o VSCode ajuda a trabalhar com conceitos de objetos e classes, oferecendo autocompletar e sugestões de código.
- **Funcional:** O editor suporta linguagens funcionais (como Haskell e Elixir) e oferece funcionalidades como destaque de sintaxe e refatoração de funções, além de possibilitar a execução de código de forma eficiente pelo terminal.

The Rapazes - Paradigmas de Programação Imperativo, POO, Funcional

1. Imperativo:

- **Descrição:** Baseado em uma sequência de instruções, onde o programador controla o fluxo de execução passo a passo.
- **Vantagens:** Simples e direto para tarefas pequenas.

- **Desvantagens:** Difícil de manter, propenso a erros e não escalável para projetos grandes.

2. Programação Orientada a Objetos (POO):

- **Descrição:** Organiza o código em objetos (instâncias de classes) que encapsulam dados e comportamentos.
- **Vantagens:** Melhora a modularidade, reutilização e manutenção de código.
- **Desvantagens:** Pode ser mais complexo para iniciantes e exigir mais esforço no design.

3. Funcional:

- **Descrição:** Foca em funções puras, imutabilidade e ausência de efeitos colaterais.
- **Vantagens:** Código mais previsível, fácil de testar e ideal para concorrência.
- **Desvantagens:** Curva de aprendizado maior e nem sempre natural para quem vem de paradigmas imperativos.

CREV - Paradigmas de Programação:

1. **Imperativo:** É o paradigma mais simples, baseado em uma sequência de instruções para execução rápida. No entanto, é difícil de manter e propenso a erros.
2. **POO (Programação Orientada a Objetos):** Surge para melhorar a modularidade e manutenção, utilizando conceitos como Encapsulamento, Herança e Polimorfismo.
3. **Funcional:** Trata a computação como a avaliação de funções matemáticas. A ênfase está em funções puras, imutabilidade e tratamento de dados sem efeitos colaterais.
4. **Lógico:** Baseado na lógica formal e matemática, usa fatos e regras para declarar relações e resolver problemas por meio de busca de provas lógicas.

YET - Programação Funcional:

A **programação funcional** foca no uso de funções matemáticas para resolver problemas computacionais. Os conceitos principais incluem:

- **Funções puras** (sem efeitos colaterais)
- **Imutabilidade**
- **Funções de ordem superior**

Exemplos de linguagens funcionais puras: Haskell, Elm, PureScript e Elixir.

CKS - Paradigma Funcional em Java:

No **Java 8**, o paradigma funcional pode ser implementado com ferramentas como:

- **Streams**: `map`, `filter`
- **Funções Lambda**
- **Interfaces Funcionais**
- **Imutabilidade** com estruturas auxiliares

Vantagens: Código mais limpo, conciso e legível.
Desvantagens: Curva de aprendizado acentuada, especialmente para quem vem do paradigma imperativo/tradicional.

Diferenças Conceituais: O Java é predominantemente imperativo e orientado a objetos, enquanto o paradigma funcional oferece uma abordagem diferente, focada em funções e imutabilidade.

Dica: Para aprofundar no paradigma funcional, confira a playlist de Otávio Lemos.