

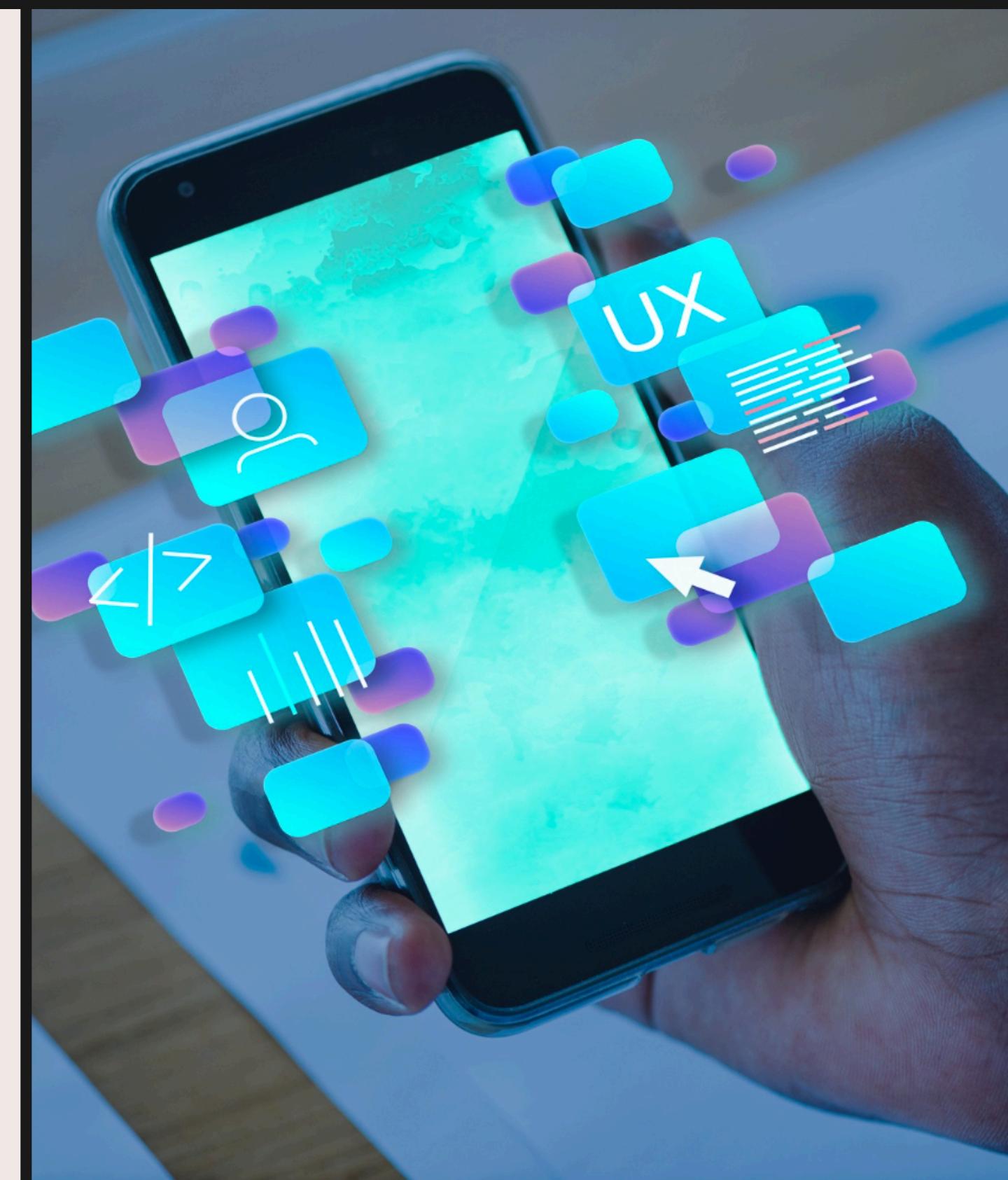
Análise Comparativa de Gerenciadores de Estado em Flutter com Foco na Incidência de Antipattern

1. Introdução

1.1 Gerenciadores de Estado

Gerenciadores de estado no Flutter são **ferramentas ou padrões** que ajudam a **controlar e atualizar os dados exibidos na interface**. Eles organizam como o app guarda informações, reage a mudanças e mantém a **UI sincronizada com o estado atual**.

Basicamente, evitam bagunça no código e facilitam lidar com mudanças na tela.



1. Introdução

1.2 O que é Antipattern?

Um antipadrão é uma solução comumente adotada que acaba gerando mais problemas do que benefícios. Enquanto os padrões de projeto ajudam a organizar o código de forma eficiente, **os antipadrões acontecem quando adotamos soluções rápidas ou convenientes demais, que deixam o sistema confuso, frágil ou difícil de manter a longo prazo.**



2. Problema

Ainda não está claro **como diferentes gerenciadores de estado do Flutter influenciam a qualidade do código e a ocorrência de antipadrões em projetos reais.** Falta uma análise sistemática que compare modelos para entender se algum deles tende a produzir arquiteturas mais complexas, mais acopladas ou mais propensas a erros.

3. Objetoivo geral

Avaliar como diferentes gerenciadores de estado em Flutter influenciam a ocorrência de antipadrões em projetos reais, com foco na comparação entre **Bloc** e **MobX**.

3.1 Objetivos específicos

- **Identificar e catalogar** os antipadrões relacionados ao gerenciamento de estado presentes em projetos Flutter que utilizam Bloc e MobX.
- **Comparar Bloc e MobX** quanto à incidência, diversidade e severidade dos antipadrões identificados.
- **Avaliar** como acoplamento, complexidade, modularização e tamanho dos módulos se relacionam com os antipadrões encontrados em cada gerenciador.
- **Examinar a relação** entre antipadrões e comportamentos de performance.
- **Propor recomendações** práticas para mitigação de antipadrões, baseadas nas evidências observadas nos projetos Bloc e MobX analisados.



4. Estudos Relacionados

State management patterns and their potential drawbacks

No artigo "*State management patterns and their potential drawbacks*", Manuel Plavsic destaca vários antipadrões comuns em gerenciamento de estado de aplicações mobile, como prop drilling, uso inadequado de InheritedWidget e Provider, e problemas com fallback providers e fallback values, entre outros.

5. Stakeholders e contribuições

5.1 Partes interessadas

O estudo é relevante para **desenvolvedores e equipes** que usam Flutter, ajudando na escolha e manutenção de gerenciadores de estado. Também beneficia **arquitetos, pesquisadores, estudantes** e a **comunidade open-source**, ao oferecer evidências sobre qualidade de código, antipadrões e boas práticas.

5. Stakeholders e contribuições

5.2 Contribuições

- Impacto positivo na qualidade futura de projetos Flutter, ao levantar antipadrões comuns.
- Insights que podem influenciar escolhas arquiteturais em novos produtos.
- Melhora na tomada de decisão técnica a partir dos resultados obtidos em projetos reais.

6. Questões de Pesquisa e Métricas Utilizadas

RQ	Questão de Pesquisa	Métricas (todas obtidas por análise estática)
RQ1	Quais são os antipadrões mais recorrentes em projetos Flutter relacionados ao gerenciamento de estado?	<ul style="list-style-type: none"> Frequência de cada antipadrão por projeto • Nº total de antipadrões por grupo
RQ2	Como Bloc e MobX estruturam o fluxo de estado e que diferenças essa estrutura gera no código?	<ul style="list-style-type: none"> Nº de arquivos por fluxo de estado • Profundidade de diretórios • Grau de acoplamento (imports diretos) • Complexidade média dos módulos
RQ3	Em que situações os antipadrões surgem com mais frequência ao utilizar Bloc?	<ul style="list-style-type: none"> Nº de antipadrões por módulo Bloc • Ocorrências por tipo de arquivo (event/state/bloc/widget)
RQ4	Em que situações os antipadrões surgem com mais frequência ao utilizar MobX?	<ul style="list-style-type: none"> Nº de antipadrões por módulo MobX • Ocorrências por tipo de arquivo (store/actions/reactions/widget)
RQ5	Como Bloc e MobX se comparam quanto à quantidade, tipo e impacto dos antipadrões identificados?	<ul style="list-style-type: none"> Distribuição percentual por tipo • Severidade estática (baixa/média/alta)
RQ6	Quais práticas podem reduzir o surgimento de antipadrões ao utilizar Bloc ou MobX?	<ul style="list-style-type: none"> Relação entre presença de boas práticas (extração de widgets, uso de select(), modularização) e redução de antipadrões

RQ7	Como decisões de modularização influenciam o surgimento de antipadrões?	<ul style="list-style-type: none"> • N° de módulos/pacotes • Tamanho médio dos módulos (LOC) • Antipadrões por módulo
RQ8	Quais diferenças aparecem na navegação e compreensão do código entre projetos Bloc e MobX?	<ul style="list-style-type: none"> • N° de passos entre arquivos (grafo de imports) • N° de arquivos necessários para localizar estados/ações
RQ9	Quais antipadrões estão associados a problemas potenciais de performance?	<ul style="list-style-type: none"> • Tamanho de widgets dentro de BlocBuilder / Observer • Falta de const • N° de reações (MobX) • Tamanho de árvores reativas — <i>proxies de rebuilds desnecessários</i>
RQ10	Quais características arquiteturais estão mais correlacionadas com antipadrões?	<ul style="list-style-type: none"> • N° de dependências cruzadas • Complexidade ciclomática • Profundidade de camadas • Imports circulares
RQ11	Como diferentes ferramentas de análise estática diferem na detecção de antipadrões entre Bloc e MobX?	<ul style="list-style-type: none"> • N° de regras aplicadas por ferramenta • Diferença entre antipadrões detectados • N° de falsos positivos/negativos estimáveis por padrões contraditórios
RQ12	Como os antipadrões observados influenciam o esforço de manutenção do código?	<ul style="list-style-type: none"> • Tempo entre commits sucessivos que afetam o mesmo módulo com antipadrões

7. Hipóteses formais

H1 – Ocorrência geral de antipadrões

- **H₀₁**: Não há diferença significativa na quantidade de antipadrões entre projetos que utilizam Bloc e MobX.
- **H₁₁**: Projetos que utilizam MobX apresentam maior quantidade de antipadrões que projetos que utilizam Bloc.

H2 – Acoplamento e modularização

- **H₀₂**: Não há diferença significativa no acoplamento entre módulos em projetos com Bloc e MobX.
- **H₁₂**: Projetos com MobX apresentam maior acoplamento entre módulos do que projetos com Bloc.

H3 – Complexidade do fluxo de estado

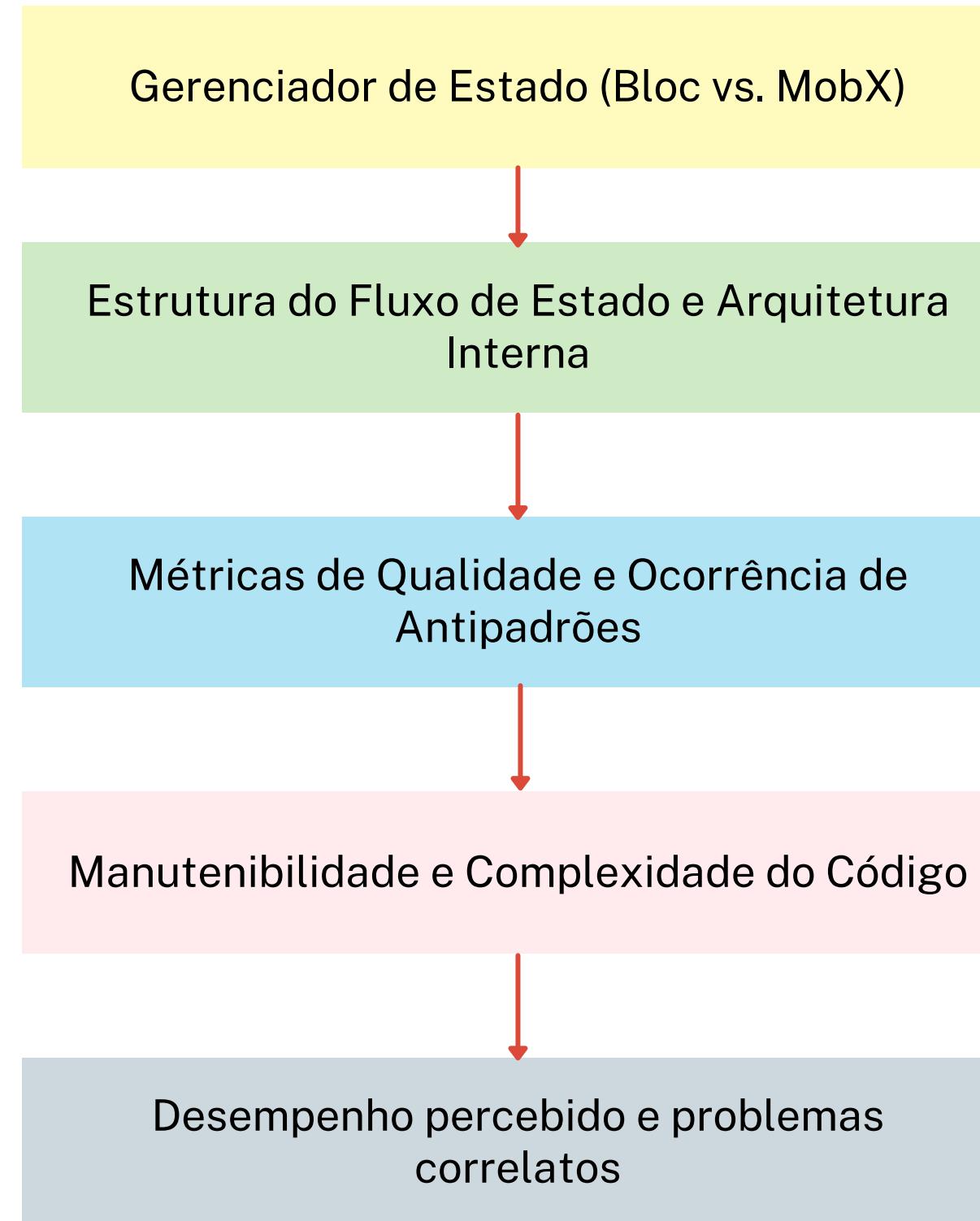
- **H₀₃**: Bloc e MobX apresentam complexidade média de fluxo de estado equivalente.
- **H₁₃**: O Bloc apresenta maior complexidade estrutural (mais arquivos, mais camadas) do que o MobX.

7. Hipóteses formais

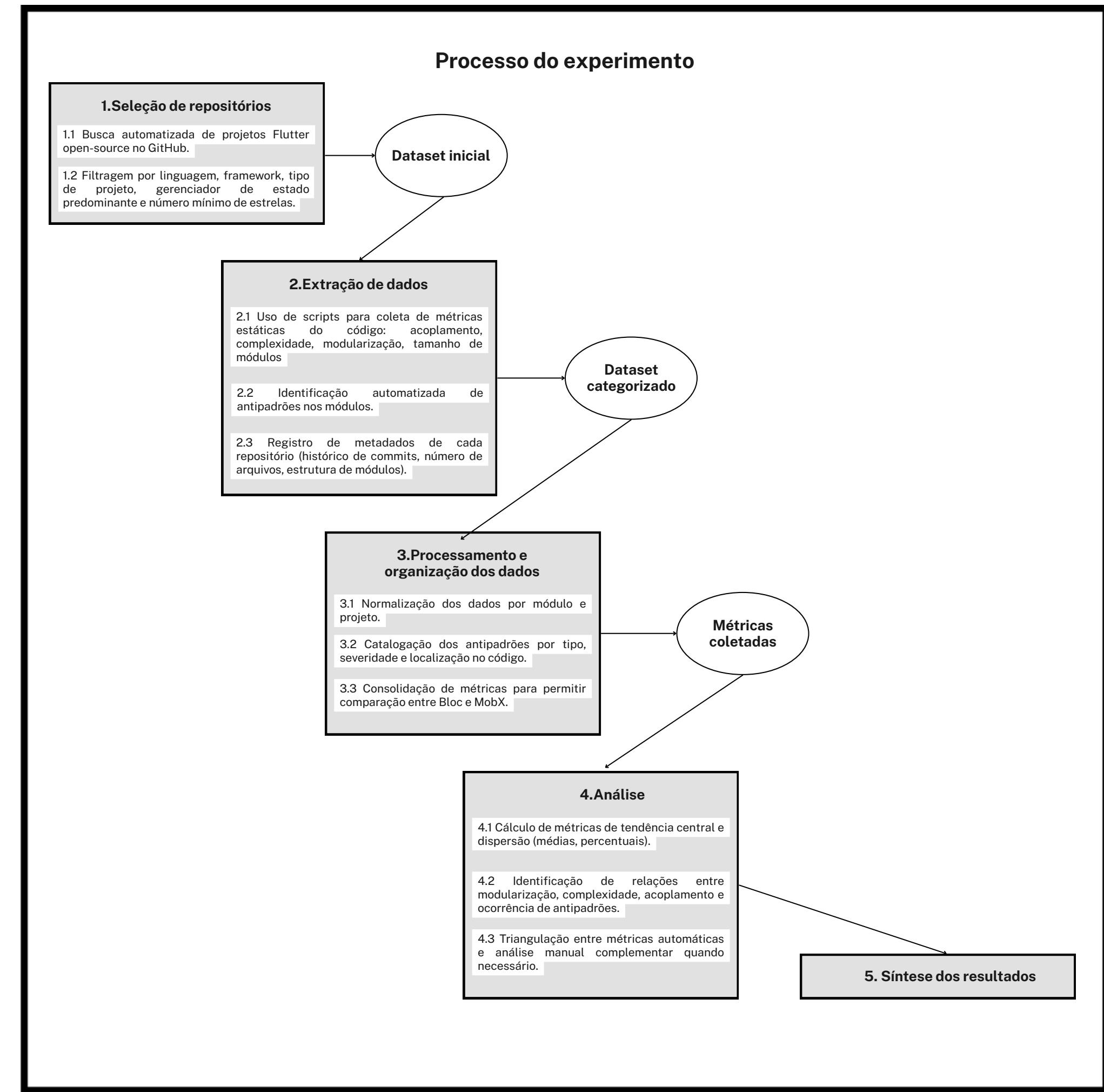
H6 – Severidade dos antipadrões

- **H₀₆**: A severidade dos antipadrões é equivalente entre Bloc e MobX.
- **H₁₆**: Antipadrões encontrados em projetos MobX têm maior severidade e impacto arquitetural.

8. Modelo Conceitual



O gerenciador de estado (Bloc ou MobX) influencia diretamente como o fluxo de estado e a arquitetura interna são estruturados. Essa estrutura, por sua vez, afeta as métricas de qualidade e a quantidade de antipadrões presentes, o que impacta a manutenibilidade e a complexidade do código. Esses fatores acabam refletindo também no desempenho percebido da aplicação e nos problemas associados.



8. Variáveis, fatores e objetos de estudo

8.1 Objetos de estudo

Os objetos de estudo desta pesquisa são (1) o gerenciador de estado adotado pelos repositórios analisados e (2) os artefatos produzidos durante esse processo, ou seja, o código-fonte.

8. Variáveis, fatores e objetos de estudo

8.2 Variáveis independentes

Fator	Descrição	Níveis
Gerenciador de estado	Tecnologia analisada	<i>Bloc, MobX</i>
Módulo analisado	Parte do sistema	múltiplos módulos por repositório
Tipo de antipadrão	Categoria segundo literatura	~10 categorias predefinidas

8. Variáveis, fatores e objetos de estudo

8.3 Variáveis dependentes

- Quantidade de antipadrões
- Tipos e severidade dos antipadrões
- Passos para navegar e entender o código
- Rebuilds desnecessários
- Manutenibilidade percebida
- Sintomas de performance

8. Variáveis, fatores e objetos de estudo

8.4 Variáveis de controle / bloqueio

As variáveis de controle incluem linguagem e framework (Flutter/Dart), tipo do projeto (aplicação), uso predominante de um gerenciador de estado (Bloc ou MobX) e número mínimo de estrelas.

9. Ameaças à validade

Para garantir resultados confiáveis, identificamos ameaças que podem comprometer a robustez e interpretabilidade do estudo e apresentamos estratégias de mitigação.

9. Ameaças à validade

1. Número de repositórios

- **Categoria:** Validade de conclusão / Selection
- **Descrição:** Número limitado de projetos Bloc e MobX disponíveis com estrutura adequada e número mínimo de estrelas.
- **Impacto:** Pode dificultar comparações estatísticas confiáveis entre os grupos.
- **Mitigação:** Selecionar todos os repositórios que atendam aos critérios; balancear Bloc e MobX; complementar com descrição detalhada do contexto.

9. Ameaças à validade

2. Diferenças de arquitetura / proposta dos projetos

- **Categoria:** Validade interna / Selection
- **Descrição:** Projetos têm arquiteturas ou **objetivos diferentes**, o que pode tornar alguns **antipadrões aceitáveis** em determinados contextos.
- **Impacto:** Comparações diretas podem ser enviesadas se não considerarmos o contexto.
- **Mitigação:** Normalizar métricas por tamanho do módulo; registrar arquitetura base; analisar antipadrões considerando contexto de cada projeto.

9. Ameaças à validade

3. Mistura de gerenciadores de estado em um projeto

- **Categoria:** Validade interna / Confounding
- **Descrição:** Alguns projetos podem usar Bloc, MobX e outros gerenciadores juntos, **dificultando atribuição de antipadrões a um gerenciador específico.**
- **Impacto:** Pode distorcer resultados sobre incidência de antipadrões por gerenciador.
- **Mitigação:** Excluir projetos com mistura significativa; analisar apenas módulos predominantes; documentar claramente os casos de mistura.

Obrigado!



ALGUMA PERGUNTA?

Merry Christmas

