

Padrão de Projeto

Alexandre Fabian

Padrão:

Singleton

Projeto:

App Questoes

Projeto utilizando Linguagem Java para a plataforma Android de PW3

Singleton

É um padrão de projeto criacional

Intenção

- Garantir que uma classe tenha somente uma instância e
- fornecer um ponto global de acesso para a mesma. (GAMMA et al., 2000)
- O Singleton pode ser reconhecido por um método de criação estático, que retorna o mesmo objeto em cache.

Motivação

É importante para algumas classes ter uma, e apenas uma, instância.

- Como garantimos que uma classe tenha somente uma instância e que essa instância seja facilmente acessível?

Possível Resposta:

- Uma variável global torna um objeto acessível, mas não impede você de instanciar múltiplos objetos.

Resposta:

- Uma solução melhor seria tornar a própria classe responsável por manter o controle da sua única instância.

Este é o padrão Singleton

Use o padrão Singleton quando:

- for preciso haver apenas uma instância de uma classe, e essa instância tiver que dar acesso aos clientes através de um ponto bem conhecido;
- a única instância tiver de ser extensível através de subclasses, possibilitando aos clientes usar uma instância estendida sem alterar o seu código.

Estrutura

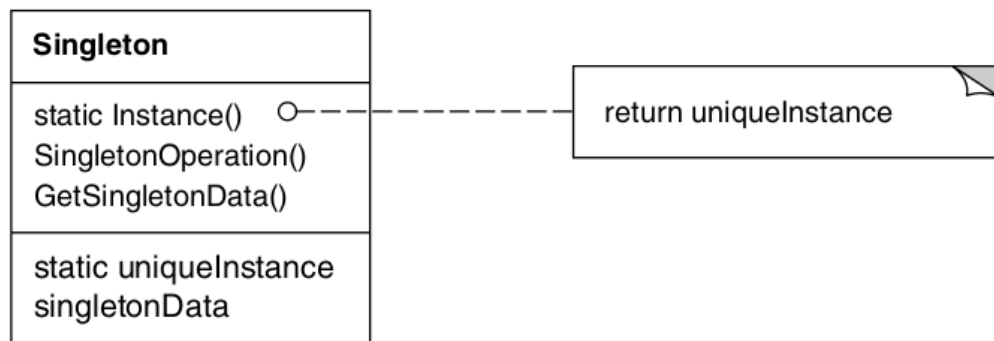


Figure 1: Singleton

Vantagens:

1. Acesso controlado à instância única.
2. Espaço de nomes reduzido.
3. Permite um refinamento de operações e da representação.
4. Permite um número variável de instâncias:

O padrão torna fácil mudar de idéia, permitindo mais de uma instância da classe Singleton.

Desvantagens

- Muitos desenvolvedores consideram o padrão Singleton um antipadrão. É por isso que seu uso está diminuindo no código Java. (SHVETS, 2019)

Como aplicar:

Garantindo uma única instância.

- Ocultando a operação que cria a instância, usando uma operação com uma função-membro **estática**
- esta operação tem acesso a variável que mantém a única instância,
- e garante que seja iniciada uma unica vez antes de retornar o seu conteúdo.

Sobre o exemplo

App para treinar questões do Encceja:

- Banco de dados no Firebase para as questões e Storage das imagens.
- Banco de dados local para guardar as informações de desempenho

Casos de uso

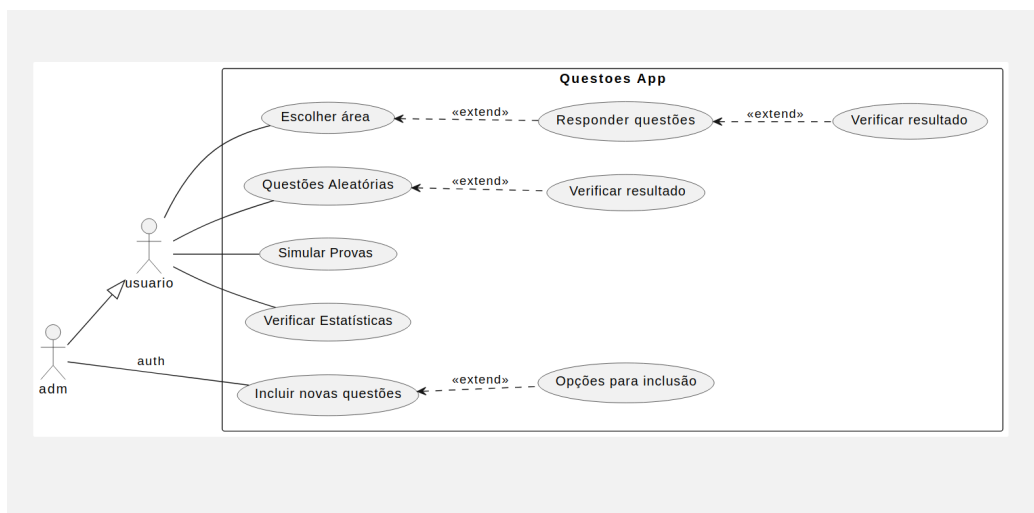


Figure 2: Casos de uso

Diagrama de Classes

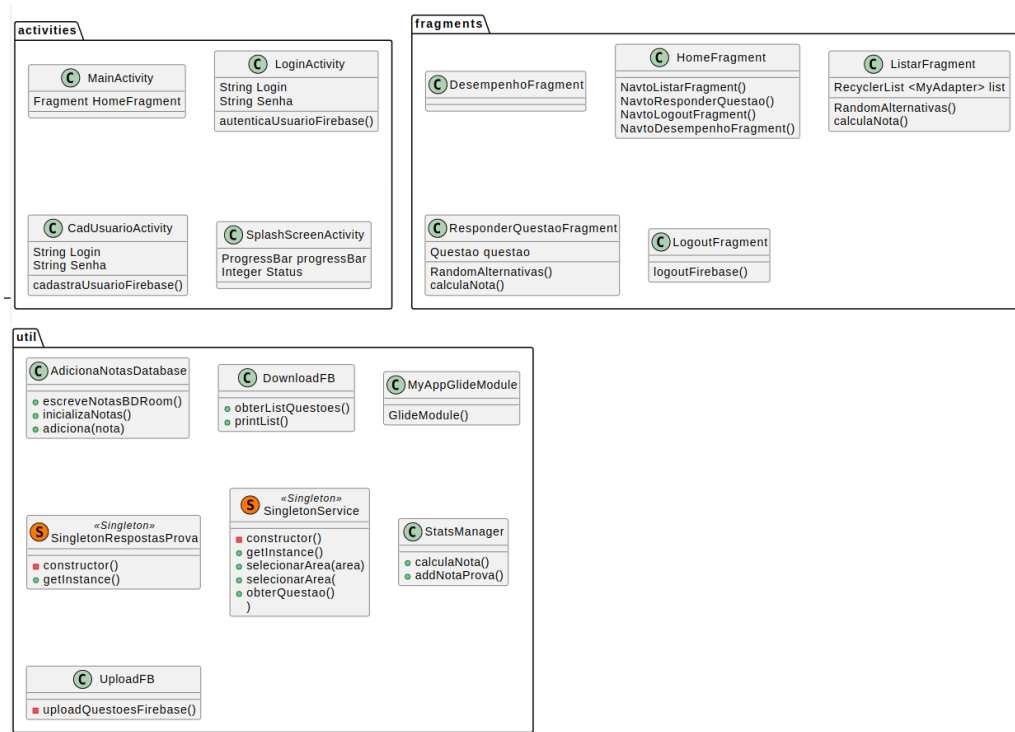


Figure 3: Diagrama de Classes

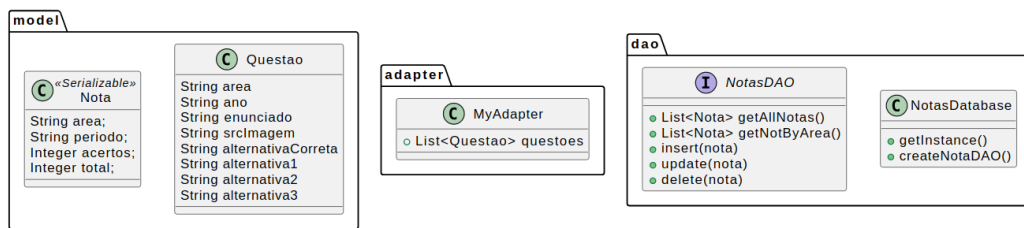
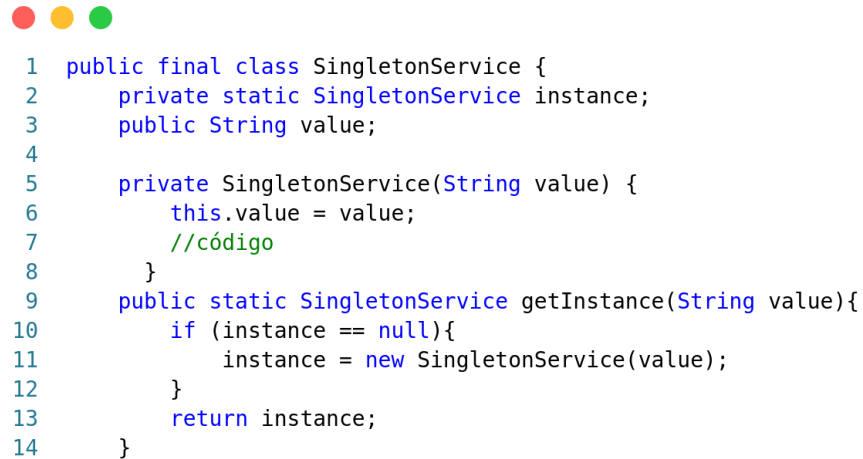


Figure 4: Diagrama de Classes

Singleton

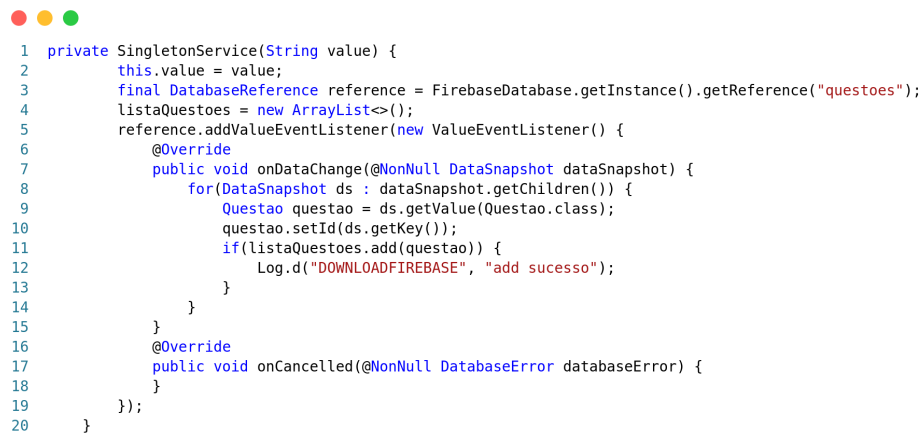


```

1  public final class SingletonService {
2      private static SingletonService instance;
3      public String value;
4
5      private SingletonService(String value) {
6          this.value = value;
7          //código
8      }
9      public static SingletonService getInstance(String value){
10         if (instance == null){
11             instance = new SingletonService(value);
12         }
13         return instance;
14     }

```

Figure 5: Classe Singleton



```

1  private SingletonService(String value) {
2      this.value = value;
3      final DatabaseReference reference = FirebaseDatabase.getInstance().getReference("questoes");
4      listaQuestoes = new ArrayList<>();
5      reference.addValueEventListener(new ValueEventListener() {
6          @Override
7          public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
8              for(DataSnapshot ds : dataSnapshot.getChildren()) {
9                  Questao questao = ds.getValue(Questao.class);
10                 questao.setId(ds.getKey());
11                 if(listaQuestoes.add(questao)) {
12                     Log.d("DOWNLOADFIREBASE", "add sucesso");
13                 }
14             }
15         }
16         @Override
17         public void onCancelled(@NonNull DatabaseError databaseError) {
18         }
19     });
20 }

```

Figure 6: Construtor Privado

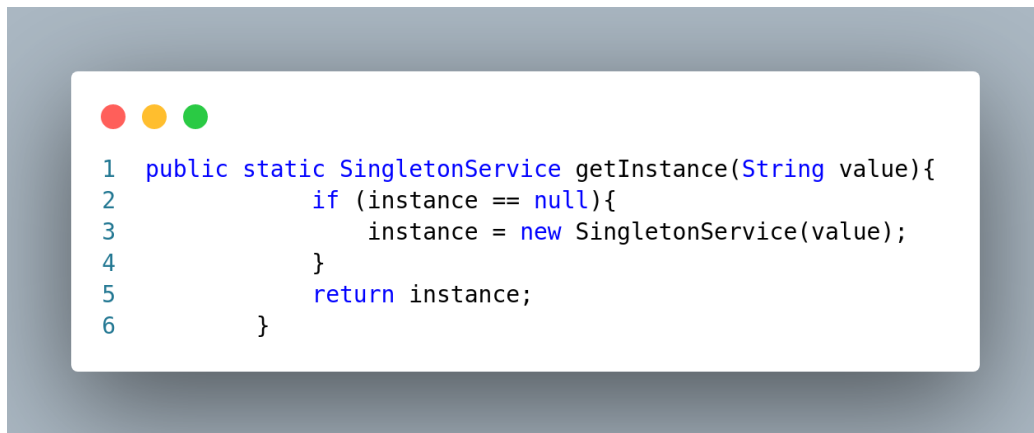


Figure 7: Construtor Publico

App

- [Link para video de demonstracao.](#)

Screenshots:

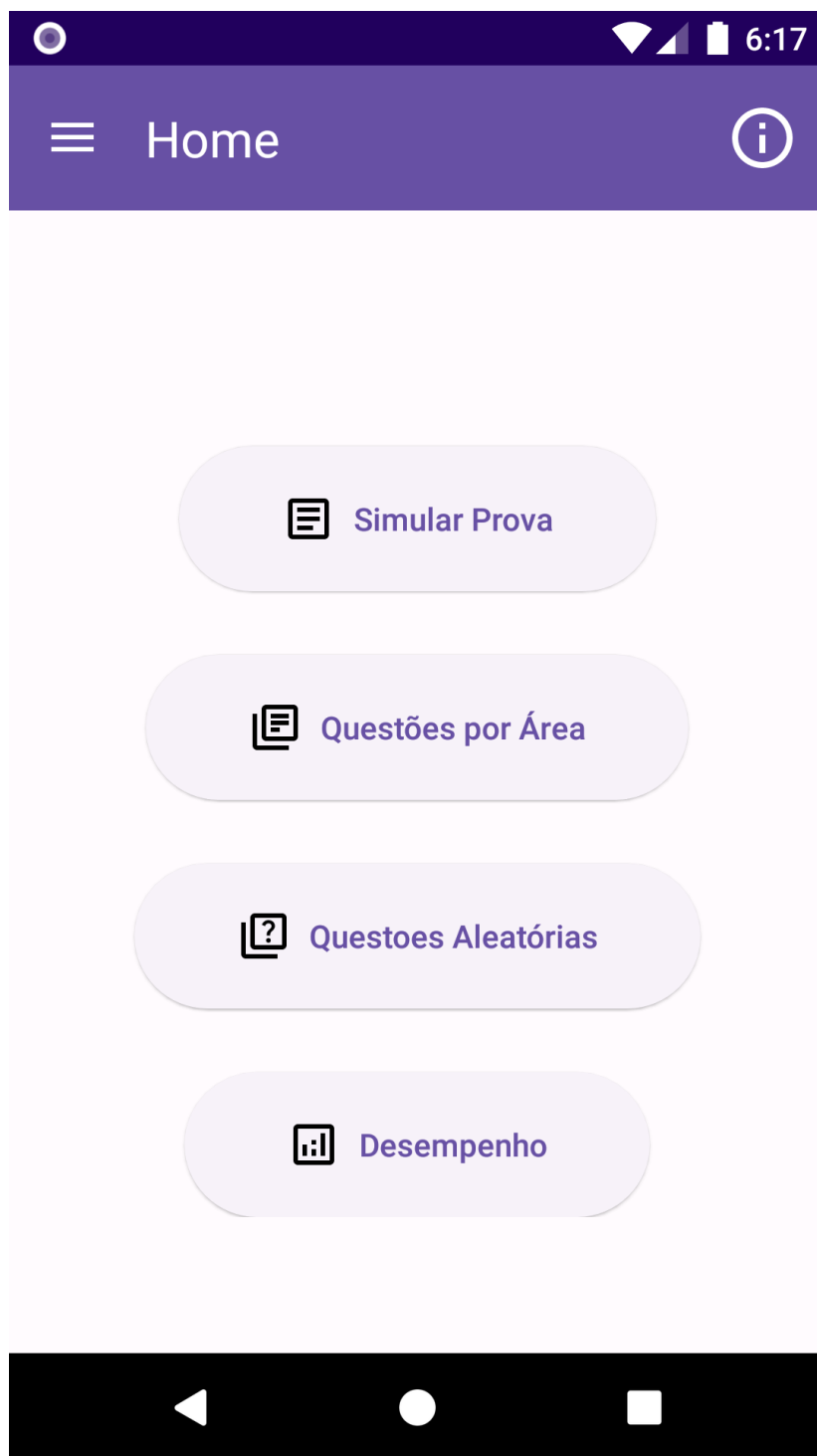


Figure 8: Home

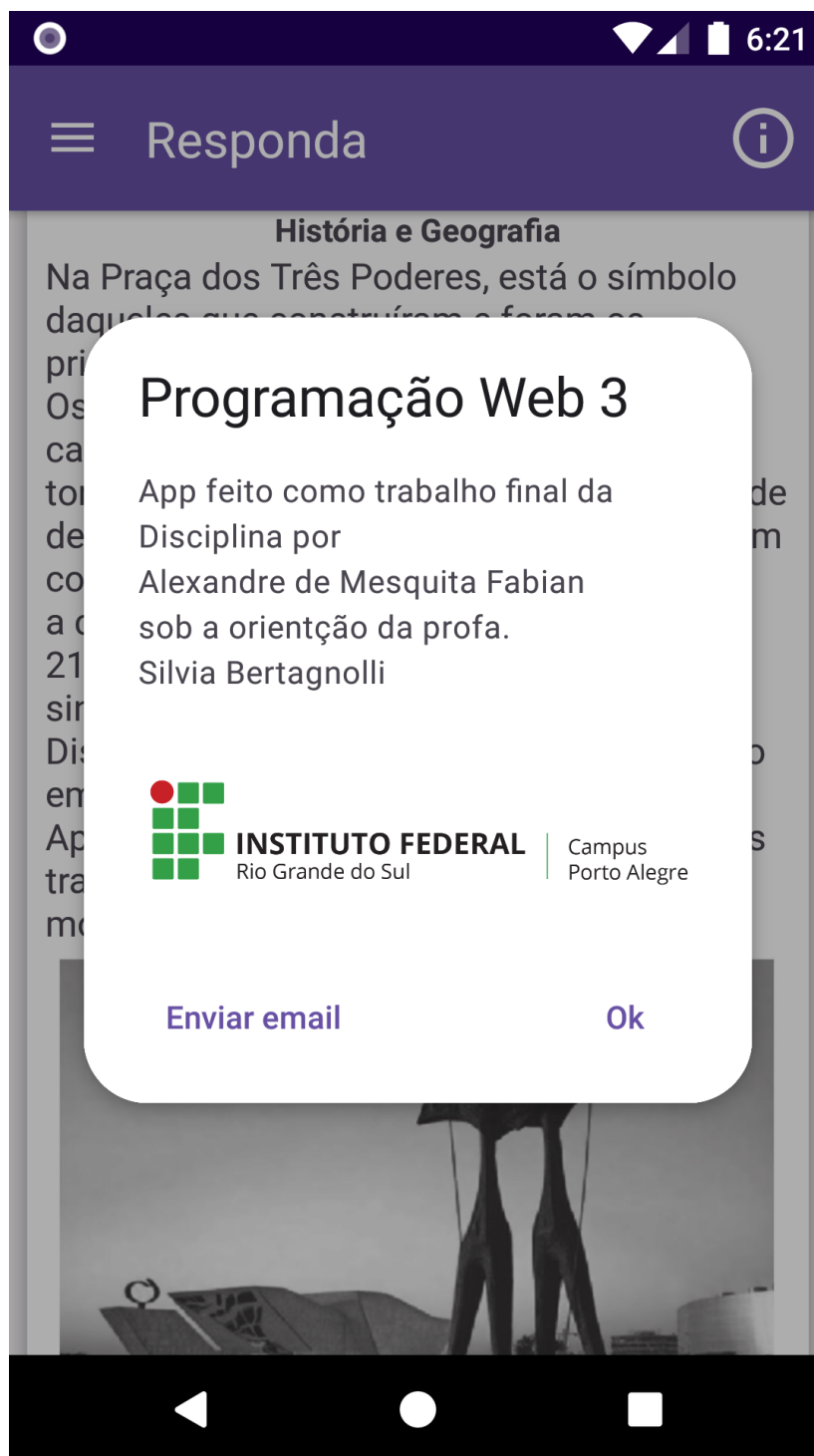


Figure 9: Sobre

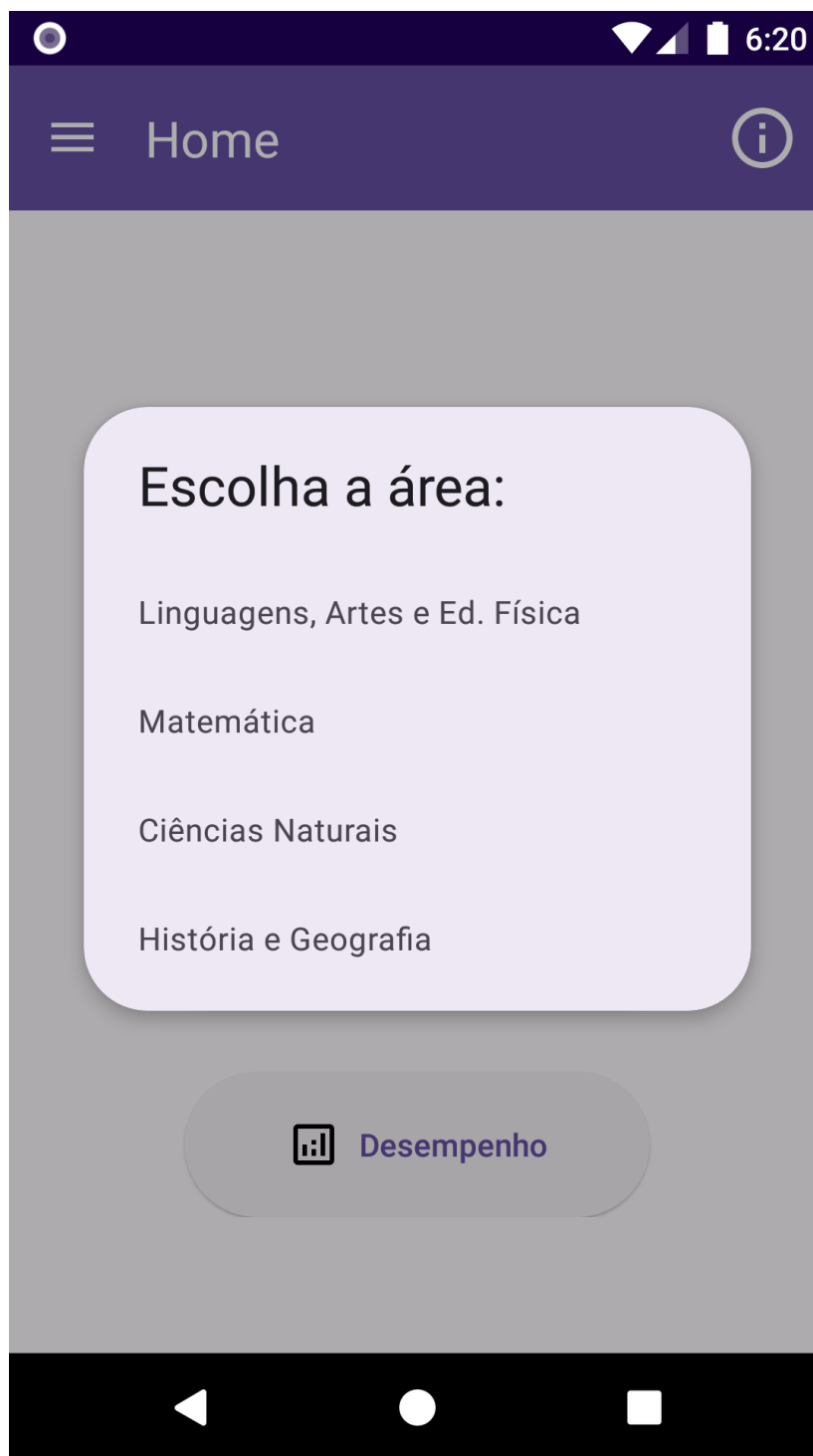


Figure 10: Escolha Área



Figure 11: Desempenho

Sobre

- Link para a apresentação no Github:
<https://github.com/amfabian/docs/tree/main/ES3/singleton>
- Link para o App no Github:
<https://github.com/amfabian/QuestoesApp>

Referências

- SHVETS, Alexander. Dive Into Design Patterns. Kamianets-Podilskyi, Ucrânia. Refactoring.Guru, 2019. 409 p.
- GAMMA, Erich; JOHNSON, Ralph; HELM, Richard; VLISSIDES, John. Padrões de Projetos: Soluções reutilizáveis de software orientados a objetos. São Paulo: Bookman, 2000. 360 p. Tradução Luiz A. Meirelles Salgado.