

Padrão de Projeto

Alexandre Fabian

Padrão:

Singleton

Projeto:

App Questoes

Projeto utilizando Linguagem Java para a plataforma Android de PW3

É um padrão de projeto criacional

- 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.

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?

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

- 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;

Use o padrão Singleton quando:

- a única instância tiver de ser extensível através de subclasses, possibilitando aos clientes usar uma instância estendida sem alterar o seu código.

Estrutura

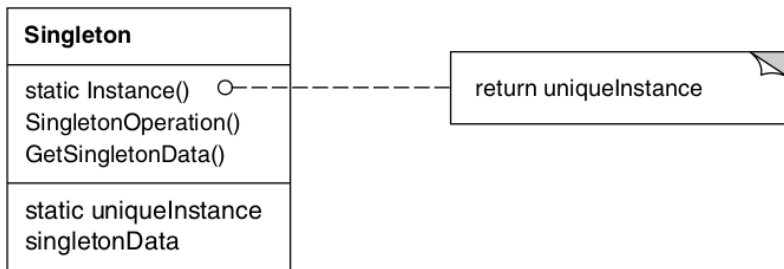


Figure 1: Singleton

Vantagens:

- ① Acesso controlado à instância única.
- ② Espaço de nomes reduzido.
- ③ Permite um refinamento de operações e da representação.

Vantagens:

④ Permite um número variável de instâncias:

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

- 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 única vez antes de retornar o seu conteúdo.

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

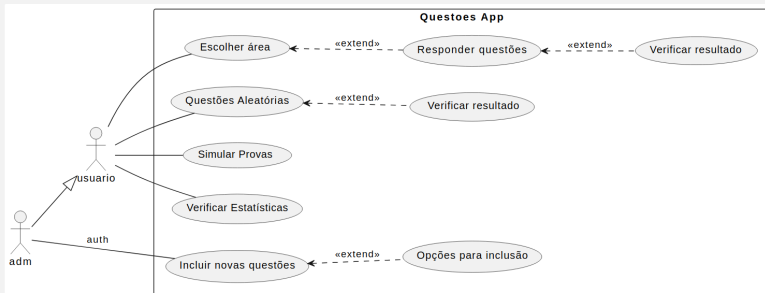


Figure 2: Casos de uso

Diagrama de Classes

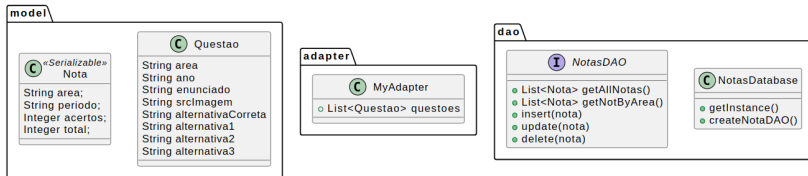
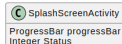
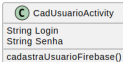
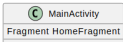


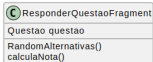
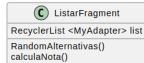
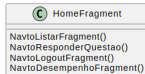
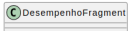
Figure 3: model, adapter e dao

Diagrama de Classes

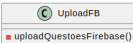
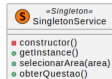
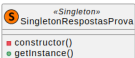
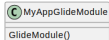
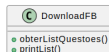
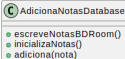
activities




fragments



util



Exemplo da utilização do 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

Exemplo da utilização do 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

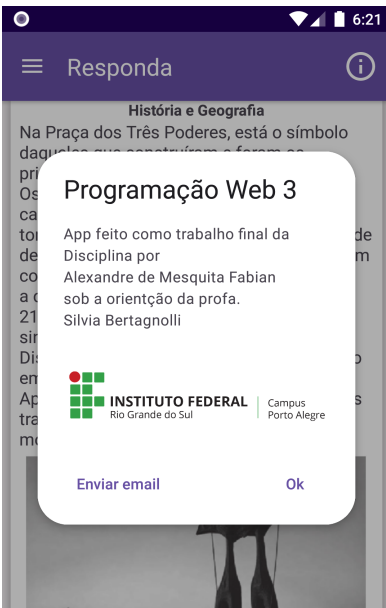
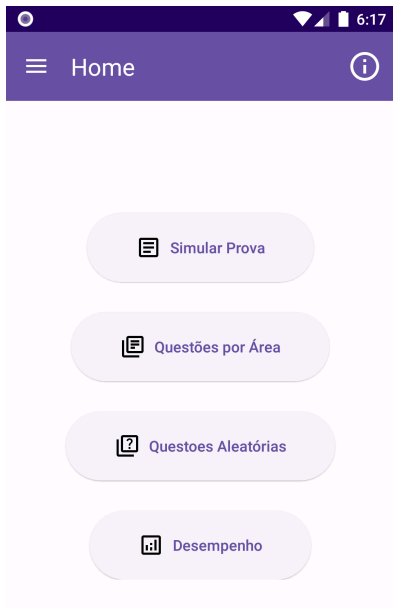
Exemplo da utilização do Singleton



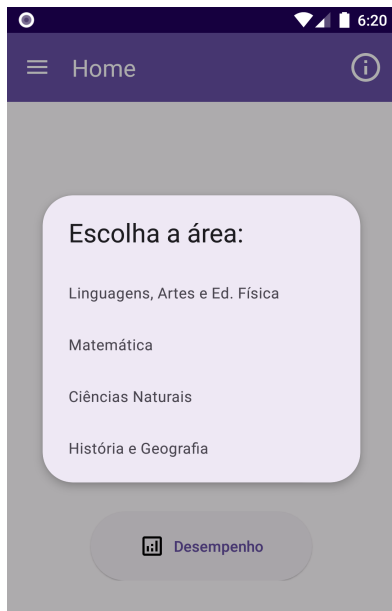
```
1 public static SingletonService getInstance(String value){  
2     if (instance == null){  
3         instance = new SingletonService(value);  
4     }  
5     return instance;  
6 }
```

Figure 7: Construtor Publico

Screenshots do App: Home e Sobre



Screenshots do App: Escolha Área e Desempenho



- Link para video de demonstracao.
- Link para a apresentação no Github.
- Link para o App no Github.

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.