UNIVERSIDADE DO OESTE DE SANTA CATARINA
CAMPUS SÃO MIGUEL DO OESTE

CURSO CIÊNCIA DA COMPUTAÇÃO

RELATÓRIO - TRABALHO A1

JOÃO GABRIEL DE ABREU

WILLIAN PABLO COLOMBO

SÃO MIGUEL DO OESTE -
SC 2023

SUMÁRIO

Lista de Figuras

1 INTRODUÇÃO (STORYTELLING)

O aplicativo “Condomine” é um aplicativo de Campo Minado feito no Flutter, realizado para Desktop e sendo possível jogá-lo em Windows e em Linux.

Era uma vez um jovem chamado Jorge, que estava entediado e procurava algo emocionante para jogar em seu computador. Ele já havia experimentado diversos jogos, mas nenhum deles conseguia capturar sua atenção por muito tempo. Jorge estava em busca de algo novo, algo que o desafiasse de maneira única.

Então, ele decidiu pesquisar na internet por jogos clássicos e encontrou uma lista com sugestões. Entre elas, estava o jogo "Campo Minado". Jorge, curioso, clicou no link para saber mais sobre o jogo. Ele descobriu que se tratava de um jogo de lógica em que o objetivo era evitar as minas ocultas no campo.

Intrigado, Jorge decidiu baixar o jogo em seu computador e iniciá-lo. Assim que o jogo abriu e ele iniciou uma partida, notou um tabuleiro de campos vazios, alguns dos quais continham minas escondidas. O desafio era desvendar o campo com cuidado, clicando em cada campo para revelar seu conteúdo. Jorge sabia que uma única explosão poderia acabar com o jogo, então ele precisava tomar decisões estratégicas.

À medida que Jorge avançava no jogo, ele se viu cada vez mais envolvido. O suspense de cada clique, a tentativa de deduzir a localização das minas com base nas dicas numéricas e a sensação de alívio quando encontrava uma célula segura eram emocionantes. O jogo desafiava sua mente e mantinha sua atenção focada.

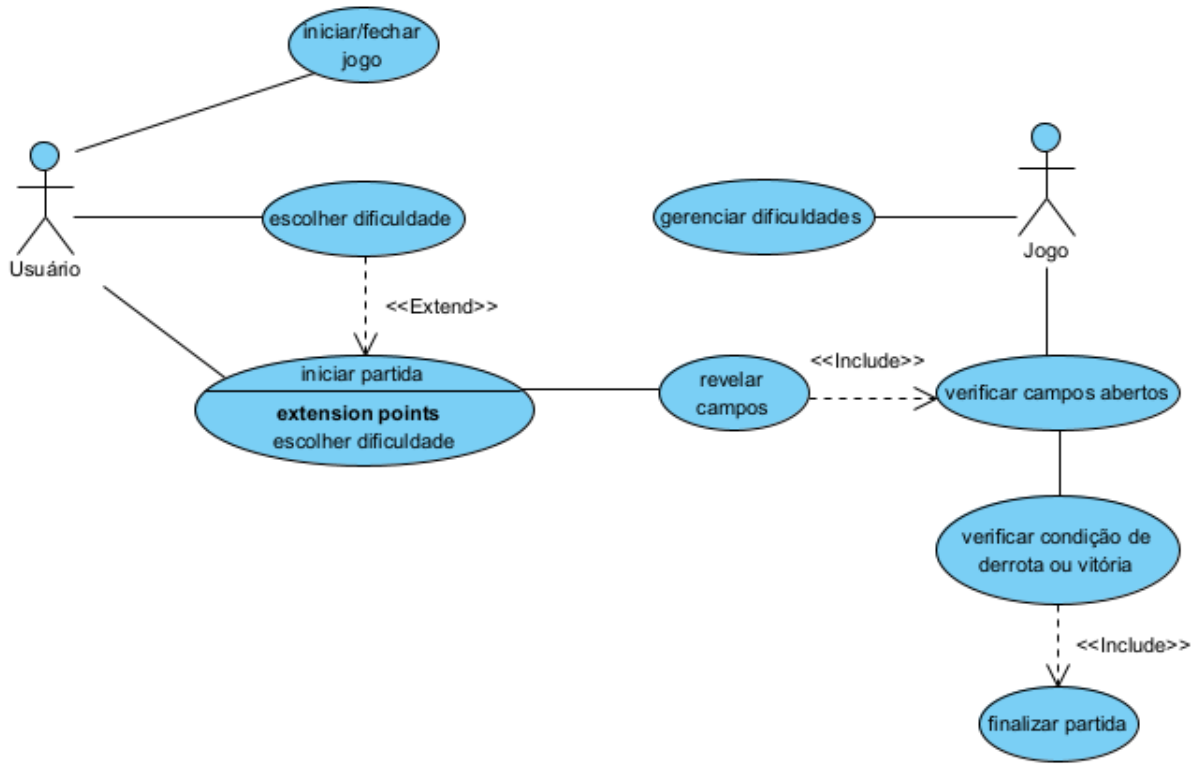
Jorge ficou impressionado com a simplicidade e a complexidade ao mesmo tempo do "Campo Minado". Ele percebeu que cada movimento era crucial e que uma decisão errada poderia levá-lo à derrota. Conforme Jorge progredia, ele começou a descobrir técnicas avançadas para melhorar seu desempenho. Ele aprendeu a usar as dicas numéricas com mais eficácia, a marcar os campos suspeitos e a analisar o campo de uma maneira mais calculada. O jogo o desafiava a pensar de forma lógica e a aprimorar suas habilidades de resolução de problemas.

Além disso, também oferecia diferentes níveis de dificuldade, permitindo que ele ajustasse o desafio de acordo com sua habilidade e disposição. Podia escolher entre tabuleiros menores e menos minas para uma experiência mais tranquila, ou enfrentar tabuleiros maiores e mais minas para um desafio verdadeiramente desafiador.

2 DESENVOLVIMENTO DA ATIVIDADE

2.1 Diagrama de caso de uso

Figura 1: Diagrama de caso de uso



Fonte: Os autores (2023)

Neste diagrama, o usuário inicia o jogo e é apresentado com a opção de selecionar o nível de dificuldade e de jogar a partida, não sendo obrigatório selecionar a dificuldade. Durante o jogo, ele pode revelar um campo no tabuleiro. O sistema, então, verifica se o campo revelado contém uma mina e, caso positivo, o jogo termina. Se o usuário revelar um quadrado sem uma mina, o jogo continua. O sistema também verifica se todos os campos sem minas foram reveladas e, nesse caso, o usuário vence a partida, se não, o usuário perde a partida.

2.2 Diagrama de classes

Essa estrutura de classes permite a criação e manipulação do jogo de Campo

Minado, com a definição de dificuldade, geração do tabuleiro, posicionamento das bombas e interação com as células do tabuleiro.

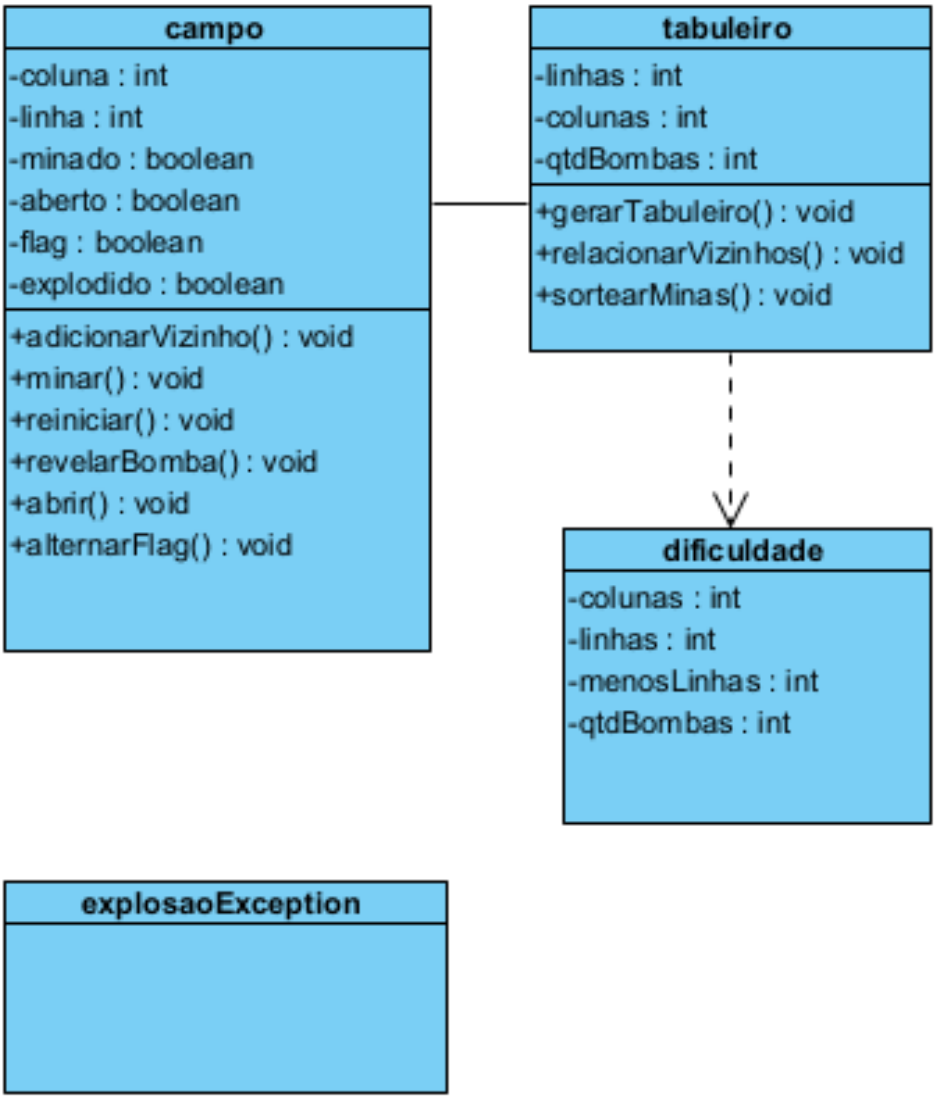
A classe Tabuleiro depende da classe Dificuldade para obter as configurações de dificuldade, como o número de linhas, colunas e bombas. O método gerarTabuleiro() da classe Tabuleiro utiliza essas informações da classe Dificuldade para criar o tabuleiro com os campos do campo minado de acordo com a configuração de dificuldade.

A classe Campo está relacionada à classe Tabuleiro por meio do relacionamento de associação. Cada instância da classe Tabuleiro contém várias instâncias da classe Campo representando os campos do campo minado. Os métodos da classe Campo são utilizados pela classe Tabuleiro para manipular o estado de cada campo do jogo durante a partida.

Portanto, a classe Tabuleiro depende da classe Dificuldade para obter as configurações de dificuldade, e a classe Campo está associada à classe Tabuleiro

para representar as células do campo minado.

Figura 2: Diagrama de classes



Fonte: Os autores (2023)

3 METODOLOGIA

Para o desenvolvimento do aplicativo, primeiramente foi realizado um cronograma, com os requisitos e as funcionalidades que o jogo deveria ter, como as classes, métodos e atributos necessários e principalmente como seria os primeiros passos do desenvolvimento.

A primeira tarefa foi realizar o desenvolvimento das classes tabuleiro e campo, definindo primeiramente o tamanho do tabuleiro e a quantidade de campos para que tivéssemos uma base e implementar a lógica dentro do jogo. Então foram adicionadas as funcionalidades principais como as bombas, bandeiras, abrir campos, etc. As tarefas foram realizadas conforme o cronograma, cada método e objeto foi documentado durante a realização delas.

Após a implementação das classes campo e tabuleiro, foi realizado o desenvolvimento da classe dificuldade, que muda a quantidade de campos e a probabilidade de ter minas nos campos. após as funcionalidades do jogo terem sido implementadas, foi iniciado a definição do menu inicial e a aparência do aplicativo, como a aparência dos botões, as cores no aplicativo, ícones e a logo. A seguir, foi realizado testes em todas as funcionalidades do aplicativo para identificar e corrigir qualquer que pudesse acontecer

4 TESTES

Foram realizados testes para garantir as funcionalidades do aplicativo, fazendo com que não tivesse erros ou comportamentos inesperados. Estes testes verificam o comportamento do campo minado em diferentes ações, como a abertura de campos com e sem explosões, a adição de vizinhos e a contagem de minas na vizinhança.
Em todas as fases do desenvolvimento, foram realizados testes de “caixa branca”, quando foram implementadas dificuldades diferentes, telas diferentes, botões, e ações diferentes na partida. Os testes de caixa preta foram os seguintes:

1. Teste 'Abrir Campo COM Explosão':

Neste teste, um objeto de campo c é criado com coordenadas (0, 0). Em seguida, a função minar() é chamada no objeto c, colocando uma mina no campo. O teste verifica se a função abrir() lançará uma exceção.

2. Teste 'Abrir Campo SEM Explosão':

Neste teste, um objeto de campo c é criado com coordenadas (0, 0). A função abrir() é chamada no objeto c. O teste verifica se a propriedade aberto do objeto c é verdadeira.

3. Teste 'Adicionar NÃO Vizinho':

Neste teste, dois objetos de campo c1 e c2 são criados com coordenadas diferentes. A função adicionarVizinho() é chamada no objeto c1, passando o objeto c2 como

argumento. O teste verifica se a lista de vizinhos do objeto c1 está vazia.

Figura 3: Testes 1

```
test('Abrir Campo COM Explosão', () {
  Campo c = Campo(0, 0);
  c.minar();

  expect(c.abrir, throwsException);
});

Run | Debug
test('Abrir Campo SEM Explosão', () {
  Campo c = Campo(0, 0);
  c.abrir();
  expect(c.aberto, isTrue);
});

Run | Debug
test('Adicionar NÃO Vizinho', () {
  Campo c1 = Campo(0, 0);
  Campo c2 = Campo(1, 3);
  c1.adicionarVizinho(c2);
  expect(c1.vizinhos.isEmpty, isTrue);
});
```

Fonte: Os autores (2023)

4. Teste 'Adicionar Vizinho':

Neste teste, quatro objetos de campo c1, c2, c3 e c4 são criados com diferentes coordenadas. A função adicionarVizinho() é chamada no objeto c1, passando os objetos c2, c3 e c4 como argumentos. O teste verifica se o comprimento da lista de vizinhos do objeto c1 é igual a 3.

5. Teste 'Minas na Vizinhança':

Neste teste, um objeto de campo c1 é criado com coordenadas (3, 3).

Três objetos de campo c2, c3 e c4 são criados com diferentes coordenadas, e c2 e c4 têm minas. A função adicionarVizinho() é chamada no objeto c1, passando os objetos c2, c3 e c4 como argumentos. O teste verifica se a quantidade de minas na vizinhança do objeto c1 é igual a 3.

Figura 4: Testes 2

```
test('Adicionar Vizinho', () {
  Campo c1 = Campo(3, 3);
  Campo c2 = Campo(3, 4);
  Campo c3 = Campo(2, 2);
  Campo c4 = Campo(4, 4);

  c1.adicionarVizinho(c2);
  c1.adicionarVizinho(c3);
  c1.adicionarVizinho(c4);

  expect(c1.vizinhos.length, 3);
});

Run | Debug
test('Minas na Vizinhança', () {
  Campo c1 = Campo(3, 3);
  Campo c2 = Campo(3, 4);
  c2.minar();
  Campo c3 = Campo(2, 2);
  Campo c4 = Campo(4, 4);
  c4.minar();
  c1.adicionarVizinho(c2);
  c1.adicionarVizinho(c3);
  c1.adicionarVizinho(c4);

  expect(c1.qtdeMinasNaVizinhanca, 2);
});
```

Fonte: Os autores (2023)

5 CONCLUSÃO

Nesse relatório, foram apresentados os principais aspectos do desenvolvimento de um aplicativo de Campo Minado utilizando a plataforma Flutter. Esse projeto foi realizado seguindo um cronograma para garantir a entrega eficiente e bem sucedida do aplicativo.

No início do relatório, foi apresentado o diagrama de classe, que fornece uma representação visual da estrutura do aplicativo e das relações entre as diferentes classes. Esse diagrama é uma ferramenta essencial para entender a arquitetura do aplicativo e facilitar o desenvolvimento e a manutenção do código.

Em seguida, foram descritos os casos de uso do aplicativo, destacando as funcionalidades principais que foram implementadas. Esses casos de uso permitem uma compreensão mais clara das interações entre o usuário e o aplicativo.

Além disso, foram detalhados os testes realizados no aplicativo, com o objetivo de verificar a correta implementação das funcionalidades e identificar possíveis problemas ou erros. Esses testes desempenham um papel fundamental na garantia da qualidade do aplicativo, permitindo que os desenvolvedores encontrem e corrijam quaisquer falhas antes da entrega. Com base nas informações apresentadas, pode-se concluir que o aplicativo está pronto para ser lançado e disponibilizado aos usuários, fornecendo uma experiência de jogo interessante e desafiadora.