# doc

## 1. Estrutura do Sistema

```
graph TD
    Main --> GerenciadorJogador
    Main --> GerenciadorRodada
    Main --> FileTemaRepository
    Main --> FilePalavraRepository
    GerenciadorRodada --> Rodada
    Rodada --> Jogador
    Rodada --> Palavra
    Palavra --> Tema
    FileTemaRepository --> Tema
    FilePalavraRepository --> Palavra
```

## 2. Fluxo Principal do Jogo

## Passo 1: Inicialização

- **Arquivos CSV**: Carregados pelos repositórios (`FileTemaRepository` e `FilePalavraRepository`)
- **Singleton**: Garante uma única instância dos repositórios
- **Exemplo**:

```
temaRepo = FileTemaRepository.getSoleInstance();
palavraRepo = FilePalavraRepository.getSoleInstance();
```

## Passo 2: Menu Principal

```
sequenceDiagram
    participant Usuário
    participant Main
    Usuário->>Main: Escolhe opção
    alt Jogar
        Main->>Main: iniciarJogo()
    else Cadastrar Tema
        Main->>Main: cadastrarTema()
    else Cadastrar Palavra
        Main->>Main: cadastrarPalavra()
    end
```

## Passo 3: Início do Jogo

- **Criação do Jogador**:

```
Jogador jogador = gerenciadorJogador.criarJogador(nome);
```

- **Lógica de Repositório**:
  - Adiciona jogador na memória
  - Não persiste em arquivo (apenas memória)

## Passo 4: Criação da Rodada

- **Seleção de Tema**:

```
Tema tema = temas.get(random.nextInt(temas.size()));
```

- **Seleção de Palavras**:

```
List<Palavra> palavrasEscolhidas = palavrasTema.subList(0, 3);
```

## 3. Loop Principal da Rodada

## Passo 5: Exibição do Estado

```
private static void exibirStatus(Rodada rodada) {
    System.out.println("Palavras: " + mascararPalavras(rodada));
    System.out.println(Boneco.montar(rodada.getErros()));
}
```

- **Mascaramento**:

  ```
  _ _ A _ _  B _ _ C _
  ```

## Passo 6: Processamento de Tentativas

- **Tentativa de Letra**:

  ```
  rodada.tentarLetra(input.charAt(0));
  ```

- **Lógica de Acerto/Erro**:
  - Letra existente: revela posições
  - Letra inexistente: incrementa erros

## Passo 7: Tentativa de Arriscar

- **Validação**:

  ```
  boolean acerto = rodada.arriscar(tentativa);
  ```

- **Comparação Exata**:

  ```
  if (!original.equals(tentativa)) return false;
  ```

# 4. Finalização da Rodada

## Passo 8: Cálculo de Pontos

```
public void calcularPontos() {
    if (acertouArriscar || todasLetrasReveladas()) {
        pontos = 100 + (letrasOcultas * 15);
    }
}
```

### Passo 9: Persistência dos Dados

- **Palavras/Temas**: Salvos automaticamente via `File...Repository`
- **Jogadores**: Persistidos apenas em memória (não implementado em arquivo)

## 5. Diagrama de Sequência de uma Tentativa

```
sequenceDiagram
    participant Jogador
    participant Main
    participant Rodada
    participant Boneco
    Jogador->>Main: Digita 'A'
    Main->>Rodada: tentarLetra('A')
    Rodada->>Rodada: Verifica palavras
    alt Letra existe
        Rodada-->>Main: true
    else Letra não existe
        Rodada->>Rodada: erros++
    end
    Main->>Boneco: montar(erros)
    Boneco-->>Main: ASCII art
    Main-->>Jogador: Exibe novo estado
```

## 6. Fluxo de Exceções

```
graph TD
    A[Operação no repositório] --> B{Erro de IO?}
```

```
B →|Sim| C[RepositoryException]
B →|Não| D[Operação concluída]
C → E[Tratamento no Main]
```