RPG

▶ 👨‍💻 desenvolvido

Neste projeto, você implementei os princípios da arquitetura SOLID e os princípios de POO em uma estrutura de jogos de interpretação de papéis, mais conhecidos como jogos RPG (Role Playing Game).

▶ 🐉 **Contextualizando** 🐲

No universo de Trybers and Dragons - T&D, quase todos os seres que andam por essas terras pertencem a uma **raça** definida.

As diversas raças (como, por exemplo, Élfica, Orc ou Anã) definem as características das personagens dentro do jogo desde a sua criação, como os seus pontos de vida e a sua destreza. No entanto, existem seres bestiais denominados **monstros** que não possuem uma raça específica, mas podem lutar.

Alguns seres também possuem uma **energia** e, ao treinarem o uso da energia, passam a possuir um **arquétipo**. De modo geral, os arquétipos definem a vocação de uma personagem, suas habilidades e visão de mundo: como encaram as situações, exploram masmorras ou enfrentam monstros. Como exemplos de arquétipos presentes em T&D, podemos citar guerreiro, mago e necromante.

Boa parte dos seres podem ser considerados lutadores, bastando para isso possuir alguns atributos específicos. Em muitas ocasiões podem acontecer lutas entre personagens diversas, bem como entre personagens e monstros.

Agora, cabe a você, nobre ~~dev~~, explorar essas terras e cumprir as quests que surgirão ao longo da sua incrível ~~jornada~~ leitura do README.

Now, follow ~~the blind~~ the dungeon master!

▶ 📝 **Habilidades trabalhadas**

PROF

- Implementação da API através de uma camada de controle
- Implementação da camada de serviço
- Tratamento de erros
- Cobertura de código por testes unitários
- Dockerização da aplicação

Orientações

▶ 🐋 **Rodando no Docker**

> Rode o serviço `node` com o comando `docker-compose up -d`.

- Esse serviço irá inicializar um container chamado `trybers_and_dragons`.
- A partir daqui você pode rodar o container `trybers_and_dragons` via CLI ou abri-lo no VS Code.

> Use o comando `docker exec -it trybers_and_dragons bash`.

- Ele te dará acesso ao terminal interativo do container criado pelo compose, que está rodando em segundo plano.

> Instale as dependências [**Caso existam**] com `npm install`

- dentro do container.

> Testes com `npm testes`

- dentro do container.

⚠ Atenção ⚠ Caso opte por utilizar o Docker, **TODOS** os comandos disponíveis no `package.json` (npm start, npm test, npm run dev, ...) devem ser executados **DENTRO** do container, ou seja, no terminal que aparece após a execução do comando `docker exec` citado acima.

⚠ Atenção ⚠ O **git** dentro do container não vem configurado com suas credenciais. Ou faça os commits fora do container, ou configure as suas credenciais do git dentro do container.

⚠ Atenção ⚠ Não rode o comando npm audit fix! Ele atualiza várias dependências do projeto, e essa atualização gera conflitos com o avaliador.

- 🎛️ Linter: Para poder rodar os ESLint em um projeto, basta executar o comando npm install dentro do projeto e depois npm run lint

▶ **Passo a passo:**

1. Crie uma Classe Abstrata `Race`

- **Descrição**: A classe `Race` deve ser uma classe abstrata que define os atributos e métodos básicos para todas as raças.
- **Atributos**:
  - `name` (do tipo `string`): O nome da raça.
  - `dexterity` (do tipo `number`): A destreza da raça.
- **Métodos**:
  - `createdRacesInstances()`: Um método estático que retorna o número de instâncias criadas das classes derivadas da classe `Race`.
  - `maxLifePoints`: Um getter abstrato que deve ser implementado pelas classes derivadas para retornar o número máximo de pontos de vida da raça.

PROF

2. Crie Classes Derivadas de `Race`

- **Descrição**: Crie classes específicas para diferentes raças que herdam de `Race` e implementam o método `maxLifePoints`.
- **Raças**:
  - `Dwarf`: Deve ter `80` pontos de vida.
  - `Elf`: Deve ter `99` pontos de vida.
  - `Halfling`: Deve ter `60` pontos de vida.
  - `Orc`: Deve ter `74` pontos de vida.

3. Crie uma Classe Abstrata `Archetype`

- **Descrição**: A classe `Archetype` deve ser uma classe abstrata que define os atributos e métodos básicos para todos os arquétipos.
- **Atributos**:
  - `name` (do tipo `string`): O nome do arquétipo.
  - `special` (do tipo `number`): A potência do ataque especial.
  - `cost` (do tipo `number`): O custo energético do ataque especial.
- **Métodos**:
  - `createdArchetypeInstances()`: Um método estático que retorna o número de instâncias criadas das classes derivadas da classe `Archetype`.
  - `energyType`: Um getter abstrato que deve ser implementado pelas classes derivadas para retornar o tipo de energia utilizado (`'mana'` ou `'stamina'`).

4. Crie Classes Derivadas de `Archetype`

- **Descrição**: Crie classes específicas para diferentes arquétipos que herdam de `Archetype` e implementam o método `energyType`.
- **Arquétipos**:
  - `Mage`: Deve usar `mana`.
  - `Necromancer`: Deve usar `mana`.
  - `Warrior`: Deve usar `stamina`.
  - `Ranger`: Deve usar `stamina`.

5. Crie uma Interface `Energy`

- **Descrição**: Defina a interface `Energy` para representar o uso de energia no jogo.
- **Atributos**:
  - `type_` (do tipo `'mana'` ou `'stamina'`): O tipo de energia.
  - `amount` (do tipo `number`): A quantidade de energia.

6. Implemente o Gerenciamento de Instâncias

- **Descrição**: Implemente o método `createdRacesInstances()` na classe `Race` e `createdArchetypeInstances()` na classe `Archetype` para contar o número de instâncias criadas das respectivas classes derivadas.

PROF

7. Defina o Tipo de Energia para Arquétipos

- **Descrição**: Certifique-se de que cada arquétipo tenha um tipo de energia (`mana` ou `stamina`) e que isso seja refletido no método `energyType`.

8. Implemente o Método `maxLifePoints` para Raças

- **Descrição**: As classes derivadas de `Race` devem implementar o método `maxLifePoints` para fornecer o número máximo de pontos de vida específico para cada raça.

Estrutura do Projeto

O projeto está dividido em dois diretórios principais:

- `src/Races/`: Contém a classe abstrata `Race` e suas classes derivadas representando diferentes raças.
- `src/Archetypes/`: Contém a classe abstrata `Archetype` e suas classes derivadas representando diferentes arquétipos.