Universidade Federal de Viçosa - Campus Florestal

Documentação do Diagrama de Classes-Sprint 2

Equipe Rocket(5)

Arasy: Tales of a Magic Pact

Versão 0.1-alpha

Thiago Cândido Rocha - 4225

Florestal - MG

2023

Sumário

Introdução

Este documento é referente ao diagrama de classes desenvolvido na Sprint 2 do projeto, o qual engloba dois casos de uso: CSU02 (Interagir com o Mundo) e CSU03 (Visualizar inventário).

O documento será dividido em quatro seções principais, com as duas últimas se referindo aos casos de uso específicos da atual *sprint* de desenvolvimento. As seções anteriores irão abordar a nova documentação dos casos de uso e classes relacionadas desenvolvidos em sprints anteriores.

É importante ressaltar, no entanto, que o aspecto iterativo incremental do processo sugere que o modelo seja retrabalhado em outras sprints, podendo resultar em alterações no diagrama conforme a necessidade surgir.

CSU00 - Realizar Autenticação

O CSU00 é um caso de uso responsável pelo sistema de autenticação e carregamento de progresso do jogador. Nele observa-se uma alta presença do banco de dados que é o seu ator secundário, e por esse motivo foi escolhido para ser iniciado na primeira sprint e posteriormente refinado, possivelmente, nas sprints finais do projeto. Está relacionado com as seguintes regras de negócio: (RN05, RN06, RN07).

Base Sequence	1- O aluno digita, via teclado, o seu ID em uma caixa de texto e clica no botão 'JOGAR'; 2- O sistema realiza uma busca do ID do aluno no Banco de Dados; 3- O ID do aluno é autenticado no Banco de Dados e suas informações são carregadas; 4- O sistema exibe uma 'Tela de Carregamento' enquanto os dados do armazenados no banco são inicializados no sistema; 5- O jogo é devidamente carregado e o personagem é colocado na posição adequada, definida pelos dados carregados do banco; 6- Um contador de 'Tempo de Sessão' é inicializado e o caso de uso se encerra.
Branch Sequence	Fluxo Alternativo 1: Campos não preenchidos A) Caso o aluno não preencha o campo, o botão 'JOGAR' não estará disponível para interação e o caso de uso retorna ao passo 1. Fluxo Alternativo 1 - O aluno clicou no botão 'SAIR': A) O sistema é fechado e o caso de uso se encerra.
Exception Sequence	Fluxo de Exceção 2 - O algoritmo de verificação de ID válido detectou um ID inválido: A) O sistema retorna ao passo 1 do fluxo de execução, uma exceção é lançada e a seguinte mensagem de erro é exibida: 'Desculpe, mas o seu ID parece estar errado! Que tal uma ajudinha do seu professor?'. Fluxo de Exceção 3 - As credenciais do aluno não foram encontradas no banco de dados: A) O sistema retorna ao passo 1 do fluxo de execução, uma exceção é lançada e a seguinte mensagem de erro é mostrada na tela: "Poxa, não conseguimos te achar no nosso universo, mas o seu professor pode te ajudar!".

	Fluxo de Exceção 4 - Ocorreu uma falha no processo de carregamento dos dados do jogo: A) O sistema retorna ao passo 1 do fluxo de execução e a seguinte mensagem de erro é mostrada na tela: "Oops! Não conseguimos carregar o seu mundo, tente novamente!".

Entidades relacionadas

EntidadeAluno

A classe EntidadeAluno é responsável por armazenar algumas informações carregadas no banco de dados para o sistema do jogo, utilizando composição para adquirir um comportamento temporário de outras classes de Entidade, como por exemplo a classe EntidadePersonagem.

É importante notar que os atributos de métricas não devem, necessariamente, ser armazenados diretamente na classe para não violar o princípio de responsabilidade única. Visando contornar o problema, foi criado um componente responsável por armazenar e calcular as métricas específicas da sessão do Aluno que atualmente está utilizando o sistema.

Dos atributos da classe:

- idAluno : String
  - Atributo privado do tipo String que armazena o id carregado do banco de dados.
  - É esperado que o seu padrão seja no formato {Letra maiúscula}x2{dígitos}x6, como definido na regra de negócio associada (vide SRS).

- nome : String
  - Atributo privado do tipo String que armazena o nome completo do aluno, carregado do banco de dados.

- personagem : EntidadePersonagemPrincipal
  - Armazena uma instância da classe como um atributo privado, garantindo que seus comportamentos públicos possam ser acessíveis pela classe EntidadeAluno.
  - É implementado como um Singleton e, portanto, a instância não precisa ser recebida como parâmetro.

- metricas : ComponenteMetricasDaSessao
    - Armazena uma instância da classe como um atributo privado, garantindo que seus comportamentos públicos possam ser acessados direto pela classe EntidadeAluno.
    - Recebe uma instância da classe via construtor.

Dos métodos da classe:

- retornaParCoordenadasDoPersonagem() : ArrayList<Double>(2)
    - Método que deve instanciar uma ArrayList para armazenar o par de coordenadas do personagem.
    - As coordenadas são obtidas via getters implementados na classe EntidadePersonagem.

- atualizaMetricas() : void
    - Método que deve invocar os métodos públicos do componente de métricas da classe para realizar atualizações sobre os dados.

EntidadeQuest

A classe EntidadeQuest é responsável por armazenar algumas informações carregadas no banco de dados para o sistema do jogo, utilizando composição para adquirir um comportamento temporário de outras classes de Entidade, como por exemplo o seu componente ComponenteGerenciadorTarefas. Além disso, é uma classe que está associada a um padrão de design comportamental chamado State, para a exibição dinâmica de informações da quest.

Os atributos de uma Quest carregados do banco devem ser armazenados em uma classe chamada ComponenteDadosQuest cuja instância será passada como parâmetro para os métodos de exibição do padrão State.

Dos atributos da classe:

- estado : EstadoQuest
    - Armazena uma instância privada da interface EstadoQuest definida pelas classes que a implementam.
    - Define um comportamento para a classe de acordo com os diferentes estados.
    - A variável pode ser de qualquer instância das classes implementadas pela interface, garantindo que o sistema se torne bastante expansível.

- gerenciadorTarefas : ComponenteGerenciadorTarefa
    - Armazena uma instância da classe ComponenteGerenciadorTarefa.
    - Possibilita que a própria Quest possa gerenciar suas tarefas a partir de métodos públicos do componente.

- dadosDaQuest : ComponenteDadosQuest
    - Armazena os dados carregados do banco referentes a uma quest.

Dos métodos da classe:

- mostrarDadosQuest() : void
    - Método público que deve chamar o respectivo método de exibição da variável de estado da classe (this.estado.mostrarDadosQuest()).

- avancarEstado() : void
    - Método público que deve chamar o respectivo método da variável de estado da classe (this.estado.avancarEstado()), ao mesmo tempo que altera a sua instância atual.

- inicializaEstado(int) : void
    - Método privado responsável por inicializar a variável estado do tipo EstadoQuest conforme o valor inteiro carregado do banco.
    - É utilizado principalmente no construtor da classe.

- calculaEstadoNumerico() : int
    - Método privado responsável por, dada a atual instância da variável estado da classe, convertê-la em um valor inteiro correspondente.
    - É um método utilizado em getters que precisem do valor numérico do estado, como em ocasiões onde uma quest do sistema deve ser salva no banco com um novo estado.

EntidadeTarefa

A classe EntidadeTarefa é uma classe abstrata responsável por armazenar os atributos comuns às suas subclasses. A implementação permite que novas subclasses sejam integradas no sistema uma vez que toda especialização da classe é representada de forma genérica como a sua superclasse.

No momento a classe é modelada de forma que a integração de um padrão State também seja possível para as Tarefas do sistema, mas a modelagem poderá sofrer alterações nas próximas sprints.

Dos atributos da classe:

- estado : int
    - Atualmente é um atributo privado do tipo inteiro que é carregado do banco de dados. O valor 0 define uma Tarefa como pendente e o estado 1 como concluída.

- dadosDaTarefa : ComponenteDadosTarefa
  - É uma instância das subclasses ComponenteDadosTarefaViagem ou ComponenteDadosTarefaColeta que são definidas pelas subclasses de EntidadeTarefa.
  - A classe abstrata possui construtor com modificador protected e só pode ser instanciada nos construtores das classes concretas que herdam de EntidadeTarefa.

Dos métodos da classe:

- Construtor
  - O construtor da classe deverá ser definido como protected.

- avancarEstado() : void
  - É um método concreto e protected da classe acessível por suas subclasses, responsável apenas por alterar o valor do estado de uma Tarefa.

Componentes relacionados

ComponenteMetricasDaSessao

A classe ComponenteMetricasDaSessao é uma classe concreta e especialização da classe EntidadeTarefa. Nela são armazenados atributos associados ao item e sua respectiva quantidade necessária para que a tarefa seja concluída.

Dos atributos da classe:

- idQuest : String
  - Atributo privado que armazena o id de uma Quest carregada do banco.
  -
- tituloQuest : String
  - Atributo privado que armazena o título de uma Quest carregada do banco.

- descricaoQuest : String
  - Atributo privado que armazena toda a descrição principal de uma Quest carregada do banco.

ComponenteDadosQuest

A classe ComponenteDadosQuest é uma classe concreta responsável apenas por armazenar os atributos de uma EntidadeQuest e realizar operações básicas sobre os mesmos. A sua implementação permite uma passagem mais genérica de parâmetros para os métodos (como os das classes do padrão State) que precisam exibir informações referentes a uma Quest do sistema.

Dos atributos da classe:

- idQuest : String
  - Atributo privado que armazena o id de uma Quest carregada do banco.

- tituloQuest : String
  - Atributo privado que armazena o título de uma Quest carregada do banco.

- descricaoQuest : String
  - Atributo privado que armazena toda a descrição principal de uma Quest carregada do banco.

ComponenteGerenciadorTarefas

A classe ComponenteGerenciadorTarefas é uma classe concreta responsável por gerenciar e controlar a lista de tarefas associadas a uma Quest do sistema. Confere o comportamento de gerência para a própria EntidadeQuest a partir de seus métodos públicos.

Dos atributos da classe:

- listaTarefasDaQuest : ArrayList<EntidadeTarefa>
  - Armazena em uma arraylist todas as tarefas associadas a uma quest.
  - Todas as tarefas, independentemente do seu tipo, devem ser instanciadas e armazenadas de forma genérica como EntidadeTarefa.
  - Para acessar os métodos específicos das especializações das Tarefas é necessário verificar o tipo da instância da classe e realizar um casting para a subclasse de EntidadeTarefa associada.
- tempoGastoConclusao : double
  - É um atributo privado que define o tempo gasto para a conclusão de todas as Tarefas de uma quest.
  - A princípio, as manipulações sob o atributo serão implementadas na Sprint de retorno do CSU00 para a sua finalização.

Dos métodos da classe:

- verificaConclusaoDeTarefasViagem(coordX : double, coordY : double) : void
  - Método responsável por verificar, para cada tarefa do tipo TarefaViagem, a sua conclusão.
  - Invoca o método público verificaEstadoConclusao da classe EntidadeTarefaViagem passando como parâmetro os valores recebidos pelo método.

- verificaConclusaoDeTarefasColeta(nomeItem : String, quantidadeArmazenada : int) : void
  - Método responsável por verificar, para cada tarefa do tipo TarefaColeta, a sua conclusão.
  - Invoca o método público verificaEstadoConclusao da classe EntidadeTarefaColeta passando como parâmetro os valores recebidos pelo método.

- verificaConclusaoDaListaDeTarefas() : boolean
  - Método responsável por percorrer toda a lista de tarefas e verificar se o estado de todos os objetos da lista está configurado como “concluído”.

ComponenteDadosTarefa

A classe ComponenteDadosTarefa é uma classe de componente abstrata que armazena informações comuns dos diferentes tipos de Tarefas do sistema.

Dos atributos da classe:

- idTarefa : String
  - Atributo privado que armazena o id de uma tarefa carregada do banco.
  - Os tipos das tarefas podem ser definidos pelo formato do seu id.
- tituloTarefa : String
  - Armazena o título da tarefa carregada do banco de dados.

- descricaoTarefa : String
  - Armazena uma descrição da tarefa carregada do banco de dados.

Sistemas relacionados

SistemaControladorDeAcesso

O SistemaControladorDeAcesso é considerado um dos mais críticos e importantes do sistema, uma vez que é nele que toda a lógica de carregamento das informações contidas no banco de dados é feita para as classes da aplicação. Está indiretamente associado ao SistemaControladorDeQuests e ao SistemaLogicaGeral, uma vez que fornece informações via parâmetro para os mesmos.

Dos atributos da classe:

- usuarioAtual : EntidadeAluno
  - Armazena uma instância do aluno que logou no sistema.

- instanciaBD : SistemaControladorDeAcesso
  - Armazena a própria instância da classe utilizando o padrão Singleton.
  - Garante que possa existir apenas uma única instância da classe e sua conexão com o banco de dados para toda a aplicação.

Dos métodos da classe:

- verificaFormatoId(id : String) : boolean
  - Método responsável por verificar se um id recebido como parâmetro está em um formato válido, conforme as Regras de Negócio.
  - Deve-se utilizar técnicas de verificações por expressões regulares (regex).

- carregaQuestsDoBanco() : ArrayList<EntidadeQuest>
  - Método responsável por carregar todas as quests e suas respectivas tarefas do banco de dados após a autenticação de um Aluno.
  - Utiliza consultas SQL e deve instanciar previamente uma lista genérica de quests.
  - A lista de quests carregadas deve ser passada como parâmetro para o SistemaControladorDeQuests.

- login(id : String) : boolean
  - Método responsável por instanciar a classe EntidadeAluno e seus respectivos atributos e classes relacionadas.
  - Utiliza consultas SQL e deve verificar se o id recebido como entrada é válido, utilizando o método verificaFormatoId(id : String).

SistemaControladorDeQuests

O SistemaControladorDeQuests é responsável por manter a integridade de todas as quests da sessão e acionar os métodos que verificam a conclusão de quests sempre que os eventos associados ocorrerem.

Dos atributos da classe:

- questsPendentesDaSessao : ArrayList<EntidadeQuest>
  - Atributo responsável por manter todas as quests que se encontram em estado pendente na sessão. Normalmente serão as quests que acabaram de ser carregadas do banco de dados e que cujas pré-condições de início não foram satisfeitas.

- questsAtivasDaSessao : ArrayList<EntidadeQuest>
  - Atributo responsável por manter todas as quests que se encontram em estado ativo na sessão. São quests cuja verificação de conclusão deverá ser feita sempre que algum evento associado às tarefas do sistema ocorrer.

- questsCompletasDaSessao : ArrayList<EntidadeQuest>
  - Atributo responsável por manter todas as quests que foram concluídas durante a sessão de jogo. É importante notar que o SistemaControladorDeAcesso não deverá carregar quests concluídas do banco de dados, uma vez que o atributo sempre é inicializado como uma arraylist vazia no início de cada sessão.

Dos métodos da classe:

- inicializaQuestsDaSessao(questsDoBanco : ArrayList<EntidadeQuest>) : void
  - É um método que recebe as quests carregadas do banco de dados e as subdivide nas listas de quests referentes a cada estado.

- avaliaQuestsAtivasPorCoordenadas(coordenadaX : double, coordenadaY : double) : void
  - O método é responsável por, para cada quest da lista de quests ativas da sessão, verificar para cada tarefa de uma quest que seja uma instância de EntidadeTarefaViagem, invocando o método que verifica a conclusão da tarefa.
  - Pode-se utilizar instanceof para verificar se um objeto é instância de uma certa classe.

- avaliaQuestsAtivasPorItemColetado(nomeItem : String, quantidadeColetada : int) : void

- O método é responsável por, para cada quest da lista de quests ativas da sessão, verificar para cada tarefa de uma quest que seja uma instância de EntidadeTarefaColeta, invocando o método que verifica a conclusão da tarefa.
- Pode-se utilizar instanceof para verificar se um objeto é instância de uma certa classe.

CSU01 - Movimentar Personagem

O CSU01 é um caso de uso responsável pelo sistema de movimentação do personagem e possui uma natureza mais simples que o CSU00. É a partir dele que podemos extrair algumas métricas dos jogadores durante uma sessão, como por exemplo distância percorrida pelo mapa.

Base Sequence	1- O aluno pressiona uma das teclas de movimentação (W, A, S, D ou Up, Down, Right, Left) para mover o personagem; 2- O personagem se movimenta (o sistema atualiza a posição do personagem no mapa): para baixo (se a tecla da seta 'Down' ou 'S' foi pressionada), para cima (se a tecla seta 'Up' ou 'W' foi pressionada), para a direita (se a tecla seta 'Right' ou 'D' foi pressionada) ou para esquerda (se a tecla seta 'Left' ou 'A' foi apertada); 3- As coordenadas do personagem são atualizadas e exibidas na tela 'Coordenadas do personagem'; 4- A distância total percorrida do personagem durante a sessão é incrementada com o valor da "velocidade" do personagem e o caso de uso se encerra.
Branch Sequence	
Exception Sequence	Fluxo de Exceção 2 - O aluno tentou se mover para uma direção inacessível: A) Se a direção para a qual o aluno tentar se movimentar contiver um objeto de cenário com colisão ou se chegar na borda do mapa, a posição do personagem não será atualizada e o caso de uso retorna ao passo 1.
Sub UseCase	
Note	RN02;RN03; RN09.

Entidades relacionadas

EntidadePersonagemPrincipal

A classe EntidadePersonagemPrincipal, como o próprio nome sugere, relaciona-se diretamente ao personagem controlado pelo aluno durante a sessão. É uma classe

diretamente associada com a engine do jogo e, portanto, possui poucos atributos e métodos modelados no diagrama de classes.

É importante ressaltar, no entanto, que a classe é implementada como um singleton, para garantir que exista apenas uma única instância da classe durante toda a sessão de jogo.

EntidadeTarefaViagem

A classe EntidadeTarefaViagem é uma classe concreta e especialização da classe EntidadeTarefa. Nela são armazenados atributos de coordenada que definem um ponto do mapa onde a conclusão de uma tarefa desse tipo ocorre.

Dos métodos da classe:

- Construtor:
  - O construtor da classe recebe todos os atributos referentes à uma EntidadeTarefa juntamente com os atributos referentes à própria classe. O construtor deve invocar o construtor da superclasse e definir os próprios atributos.

- verificaEstadoConclusao(coordX : double, coordY : double) : void
  - O método público deve receber um par de coordenadas referentes à atual posição do personagem no mapa e invocar o método avancarEstado da superclasse caso a condição de conclusão seja válida (coordenadas iguais ou distância euclidiana entre pontos menor ou igual a um limiar).

Componentes relacionados

ComponenteDadosTarefaViagem

A classe ComponenteDadosTarefaViagem é responsável por armazenar informações referentes às coordenadas necessárias para a conclusão de uma tarefa do tipo TarefaViagem. É uma especialização da classe abstrata ComponenteDadosTarefa e seu construtor deve invocar o construtor de sua superclasse para definir os atributos comuns.

Dos atributos da classe:

- coordenadaFinalX : double
  - Armazena a coordenada X que indicará o ponto final para a conclusão da Tarefa.

- coordenadaFinalY : double

- Armazena a coordenada Y que indicará o ponto final para a conclusão da Tarefa.

Sistemas relacionados

SistemaLogicaGeral

É uma classe de sistema responsável por uma interação mais forte com a engine do jogo. É nela que são definidos aspectos como a câmera associada ao personagem principal e o spawn do personagem de acordo com as coordenadas do banco.

Dos métodos da classe:

- spawnarPersonagemNoMundo(spawnX : double, spawnY : double) : void
  - Método que recebe valores das últimas coordenadas de um aluno carregado do banco para efetuar o spawn do personagem na mesma posição da última sessão.

- spawnarColetaveisNoMundo(itensColetavei : ArrayList<EntidadeItem>) : void
  - Método que recebe uma lista com todos os itens associados a quests que não foram concluídas e os posiciona no mundo do jogo.

CSU02 - Interagir com o Mundo

O CSU02 é responsável pelo sistema de interação do jogador com o mapa. Nele existe uma forte presença de elementos da engine, uma vez que são abordados assuntos como o tratamento de colisões entre entidades e afins. Está diretamente relacionado à mecânica de coleta de itens e à RN02.

Base Sequence	1- O aluno se movimenta para a coordenada de um objeto do mapa e o sistema detecta a colisão; 2- O aluno pressiona a tecla 'E' e a interação com o objeto é iniciada; 3- A interação termina e o caso de uso se encerra.
Branch Sequence	Fluxo Alternativo 2 - O aluno interagiu com um objeto que correspond e a um item coletável: A) O objeto é alocado no inventário do personagem, respeitando suas regras de negócio, e o caso de uso se encerra.
Exception Sequence	Fluxo de Exceção 2(1) - O objeto não possui nenhum tipo de interação com o jogador: A) O objeto permanece imutável e o caso de uso se encerra. Fluxo de Exceção 2(2) - Ocorreu algum problema na interação com o o bjeto: A) Uma exceção é lançada e tratada. B) O objeto permanece imutável e o caso de uso se encerra.
Sub UseCase	
Note	RN02;

Entidades relacionadas

EntidadeItem

A classe armazena informações pertinentes dos itens coletáveis que são posicionados no mapa sempre que uma nova sessão de jogo é iniciada. Os itens também servem como uma alternativa para apresentar informações específicas nas instâncias das classes EntidadeTarefaColeta.

Dos atributos da classe:

- nomeItem : String
  - O atributo serve para armazenar o nome de um item carregado do banco de dados.

- idItem : String
  - O atributo serve para armazenar o id de um item carregado do banco de dados.

- coordenadaXColetada : double
  - O atributo serve para armazenar a coordenada X que indica a posição de um item carregado do banco de dados no mapa.

- coordenadaYColetada : double
  - O atributo serve para armazenar a coordenada X que indica a posição de um item carregado do banco de dados no mapa.

- flagContabilizavel : boolean
  - É um atributo que indica se um item deve ser contabilizado na contagem para a conclusão de uma tarefa:
    - Imagine que uma tarefa que precisasse de 10 pedras tivesse sido concluída anteriormente.
    - E agora, você possui uma segunda tarefa que te pede para coletar 3 pedras em lugares diferentes.
    - A flag que indica se um item deve ser contabilizável deve ser false para:
      - Todas as tarefas que já foram concluídas (portanto, associadas a quests concluídas).
      - Todas as tarefas associadas a quests pendentes (uma vez que não podem ser concluídas sem antes estarem ativas).
    - Uma alternativa seria remover do inventário todos os itens associados a tarefas de coleta que se encontram na lista de tarefas de quests concluídas.

EntidadeTarefaColeta

A classe EntidadeTarefaColeta é uma classe concreta e especialização da classe EntidadeTarefa. Nela são armazenados atributos associados ao item e sua respectiva quantidade necessária para que a tarefa seja concluída.

Dos atributos da classe:

- nomeItemNecessario : String
  - É um atributo que se refere ao nome de um item que é necessário para que uma tarefa seja concluída. A verificação de conclusão deverá ser feita ao contar a quantidade de itens com um mesmo nome armazenados no inventário.
- quantidadeNecessaria : int
  - É um atributo que define a quantidade necessária de um mesmo item para que a Tarefa seja concluída.

Dos métodos da classe:

- Construtor
  - O construtor da classe recebe todos os atributos referentes à uma EntidadeTarefa juntamente com os atributos referentes à própria classe. O construtor deve invocar o construtor da superclasse passando o atributo comum (idTarefa).

- verificaEstadoConclusao(nomeItem : String, quantidadeDoItem : String) : void
  - O método deve receber o nome de um item recentemente armazenado no inventário do personagem e a sua quantidade armazenada. A comparação é feita com os atributos da classe, invocando o método avancarEstado da superclasse caso a condição de conclusão seja verdadeira.

Componentes relacionados

ComponenteDadosTarefaColeta

A classe ComponenteDadosTarefaColeta é responsável por armazenar informações referentes ao item necessário para a conclusão de uma tarefa. É uma especialização da classe abstrata ComponenteDadosTarefa e seu construtor deve invocar o construtor de sua superclasse para definir os atributos comuns.

Dos atributos da classe:

- itemDaTarefa : EntidadeItem
  - Armazena uma referência do item necessário para a conclusão de uma quest.
  - Modelado para permitir que o componente possa coletar informações do seu item através dos seus métodos públicos.

ComponenteItemGrafico

É uma classe concreta de componente utilizada para representar graficamente informações pertinentes dos itens durante o jogo. Por se tratar de uma classe que lida com elementos gráficos, está fortemente associada à engine e/ou à bibliotecas externas.

CSU03 - Visualizar Inventário

O CSU03 é um caso de uso responsável pelo sistema de armazenamento e exibição dos itens coletados pelo jogador ao longo de sua jornada. Nele existe uma forte presença de exibição de informações para o usuário do sistema a partir de telas, podendo ou não ser integradas com a engine. Está fortemente relacionado à RN10.

Base Sequence	1- O aluno aperta a tecla "I" para abrir o inventário; 2- O sistema exibe a 'Tela do Inventário' que contêm todos os itens at é então coletados pelo aluno; 3- O aluno utiliza o mouse para percorrer pelos itens armazenados; 4- Caso o cursor do mouse passe em cima de em um dos itens, uma p equena tela contendo as informações do registro é exibida; 5- O caso de uso retorna ao passo 3.
Branch Sequence	Fluxo Alternativo 3 - A tecla "Esc" ou "I" foi pressionada: A) O aluno pressionou as teclas correspondentes e o inventário foi fec hado; B) O caso de uso se encerra. Fluxo Alternativo 3 - O aluno clicou na opção "X": A) O aluno clicou na opção "X" localizada no canto superior direito da t ela do inventário; B) A tela do inventário é fechada e o caso de uso se encerra.
Exception Sequence	Fluxo de Exceção 1 - Um dos itens coletados pelo aluno já está armaze nado no inventário: A) No caso do aluno ter interagido com um item coletável por meio do CSU02 e este já se encontra armazenado no inventário, a quantidade d o item que está armazenada é incrementada e o item não é armazenad o em um novo "slot".
Sub UseCase	
Note	RN10;

Entidades relacionadas

EntidadeMochila

É uma classe concreta responsável por armazenar uma lista de itens coletados pelo jogador para uma visualização das suas informações.

Dos atributos da classe:

- inventario : ArrayList<EntidadeItem>
    - Atributo que armazena todos os itens coletados pelo jogador.

Dos métodos da classe:

- adicionaItem(item : EntidadeItem) :void
    - Método que recebe um item como parâmetro e simplesmente o adiciona no inventário.

Componentes relacionados

ComponenteGerenciadorMochila

É uma classe responsável por gerenciar toda a lógica da Mochila através da invocação de métodos públicos da classe EntidadeMochila. É uma classe que foi desenvolvida para separar a lógica da entidade.

Dos atributos da classe:

- mochila : EntidadeMochila
    - O atributo armazena uma referência da classe e permite que cálculos mais complexos sejam realizados dentro do componente por meio dos métodos públicos mais simples da classe de entidade, como getters e setters.

Dos métodos da classe:

- contaQuantidadeItem(nomeDoItem : String) : int
    - O método recebe um nome de um dado item para contar a sua quantidade armazenada no inventário.
    - A contagem é efetuada contabilizando o número de classes EntidadeItem que possuam o mesmo nome de um item como atributo.
    - Não é feita uma contagem por id uma vez que itens iguais em coordenadas diferentes devem possuir IDs diferentes.

- guardarItem(itemColetado : EntidadeItem) : void
    - O método recebe um item coletado pelo aluno ao tratar uma colisão com o mesmo.
    - Invoca o método adicionarItem do atributo mochila da classe.

- retornarParNomeQuantidadeDoNovoItem() : Entry<String, Integer>

- É uma classe que retorna um par “Chave, Valor” que define o nome do item contabilizado e sua respectiva quantidade.
- Utiliza o resultado do método contaQuantidadeItem.

ComponenteMochilaGrafica

É uma classe concreta de componente utilizada para representar graficamente o inventário do jogador. Por se tratar de uma classe que lida com elementos gráficos, está fortemente associada à engine e/ou às bibliotecas externas.

Referências

[1] LanguageTool - Para correção de textos e verificação ortográfica. Pode ser acessado em: <https://languagetool.org/pt-BR>. Último acesso em: 06/11/2023.

[2] Documento de Análise de Artefato - Diagrama de Classes - Sprint 2 - Versão do commit "Documento de Análise do Diagrama de Classes - Sprint 2". Pode ser acessado no repositório de produto da Equipe 5, link acima. Último acesso em: 06/11/2023.