Universidade Federal de Viçosa - Campus Florestal

Documentação do Diagrama de Classes

Sprint 1

Arasy: Tales of a Magic Pact

Versão 0.1-alpha

Thiago Cândido Rocha - 4225

Florestal - MG

2023

Universidade Federal de Viçosa - Campus Florestal

Documentação do Diagrama de Classes

Sprint 1

Sumário

Universidade Federal de Viçosa - Campus Florestal

Documentação do Diagrama de Classes

Sprint 1

Introdução

Este documento é referente ao diagrama de classes desenvolvido na Sprint 1 do projeto, o qual engloba dois casos de uso em um único diagrama: CSU00 (Realizar autenticação) e CSU01 (Movimentar personagem).

O documento será dividido em duas seções principais, cada uma se referindo a cada caso de uso e suas respectivas classes modeladas. 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.

Universidade Federal de Viçosa - Campus Florestal

Documentação do Diagrama de Classes

Sprint 1

CSU00

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!".

Sistemas relacionados

Aqui serão descritas todas as classes relacionadas com o caso de uso bem como a sua implementação para o funcionamento ideal do sistema.

Controle de Acesso: Aluno

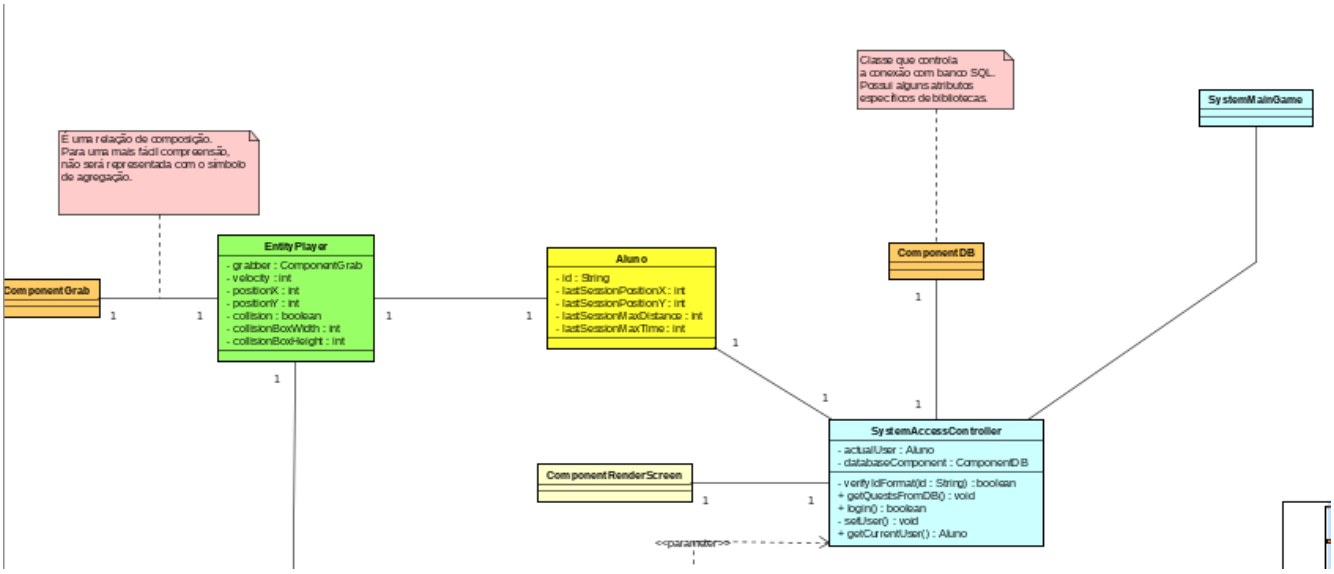
A classe Aluno está diretamente relacionada com o Banco de Dados, e ela é resultado de uma consulta correta no banco através de um ID que deve ser coletado pelo sistema controlador de acesso. Este sistema (SystemAccessController) estará responsável por toda a lógica da tela de login (que provavelmente será abordada como um componente dessa classe).

O SystemAccessController irá instanciar um aluno com informações coletadas via consultas no banco de dados e deverá ser a única classe responsável por manter a instância do aluno. Também, para aquele aluno logado, SystemAccessController deverá criar a instância do personagem (EntityPlayer).

Universidade Federal de Viçosa - Campus Florestal

Documentação do Diagrama de Classes

Sprint 1



Quests e Tasks

Após um login bem sucedido de um aluno, o sistema deverá carregar todas as informações de Quests e Tasks daquele aluno e repassar para o SystemQuestController, que é responsável por toda a gerência global de quests da sessão.

A classe SystemQuestController deverá instanciar as Quests através das consultas feitas pelo SystemAccessController (é importante verificar ainda se a passagem de parâmetros entre os System é factível). Todas as Quests dos alunos no banco de dados terão um atributo inteiro que identifica o seu estado, os quais a priori podem ser: Pendente, Iniciada, Concluída. SystemQuestController será responsável por instanciar as classes no código de acordo com o estado recolhido do banco, respeitando o padrão comportamental State que foi modelado.

Para cada sessão, SystemQuestController terá diferentes listas de quests que são armazenadas de acordo com o seu estado. A princípio, também é possível que tudo seja armazenado em uma única estrutura de dados.

Quests são classes compostas por Tasks (no nosso contexto, é feita uma composição com um componente de controle de tasks, garantindo acesso indireto às tarefas que a definem). Assim, podemos ter Quests com diferentes estados, diferentes nomes e descrições e tasks totalmente distintas, como o seguinte exemplo:

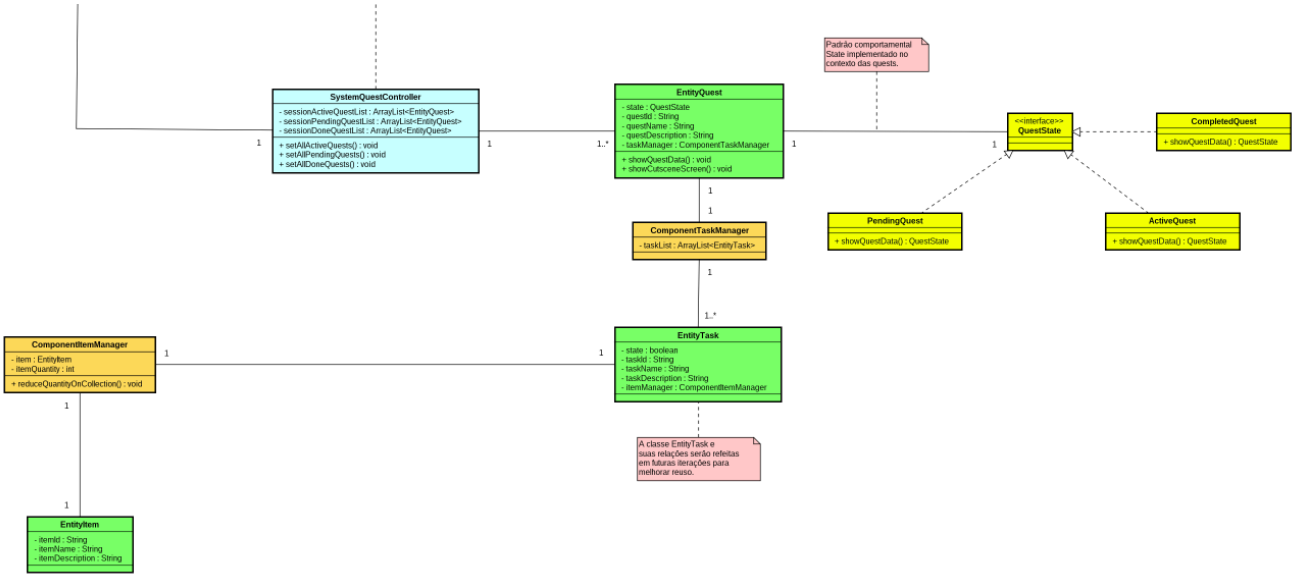
- Imagine que você, como jogador, aceitou uma quest chamada “Um peixe tirando onda” que basicamente pede para você ajudar uma tribo de peixes falantes a recuperar um artefato mágico. A quest, que antes de ser aceita pelo jogador estava no estado Pendente, passou para o estado Ativo e agora exibe informações mais detalhadas na tela.
- Como tarefas dessa quest, você precisa coletar alguns fragmentos de concha espalhados em diferentes pontos do mapa e também falar com o xamã da tribo, mas tudo isso pode ser feito na ordem que o jogador achar melhor.
- Já que temos diferentes tasks que compõem uma quest, e essas tasks podem tanto ser para coletar itens ou apenas para ir em uma localização (como o caso do xamã, porém mais simplificado), poderemos ter uma task para cada fragmento espalhado no mapa e cada uma com um contexto específico (por exemplo, cada task pode ter uma descrição específica de uma paisagem).
- Assim que todas as tarefas associadas àquela quest forem concluídas, a quest passará para o estado Completa e a sua forma de exibição em uma tela poderá, novamente, ser diferente.

O padrão State permite que classes assumam comportamentos diferentes de acordo com o seu estado, bem como possui natureza expansiva uma vez que basta a inserção de novos estados para termos novos comportamentos. Seria possível, por exemplo, a existência de um estado Oculto responsável por caracterizar os chamados *easter-eggs*, o mesmo sendo possível para as tasks.

Universidade Federal de Viçosa - Campus Florestal

Documentação do Diagrama de Classes

Sprint 1



CSU01

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.

Sistemas relacionados

Aqui serão descritas todas as classes relacionadas com o caso de uso bem como a sua implementação para o funcionamento ideal do sistema.

Personagem e controle de movimentações

Basicamente o controle de movimentações é inteiramente feito via um componente disponibilizado pela engine. É importante ressaltar, no entanto, que isso não limita a modelagem a um acoplamento forte com o motor gráfico, uma vez que bastaria substituir o componente por outro caso a plataforma seja alterada.

O funcionamento do sistema é relativamente simples e portanto pode ser representado com poucas classes na modelagem, em especial a classe EntityPlayer e seus componentes, que serão úteis para outros casos de uso.

Um EntityPlayer deve sempre possuir os atributos que representam sua coordenada (x e y no grid), velocidade (distância percorrida pela sprite a cada atualização dos frames) e, em normalmente atributos que delimitam a área de colisão da sprite.