Projeto Padrões e Projetos

The Revenge of The Descedant

START MENU

Cronograma

30/05

Finalização do Back-end, inicio do Front-end

02/06

Implementação do Front-end

13/06

Versão inicial do projeto

16/06

Versão final do projeto

Diagrama UML

SIM

NÃO

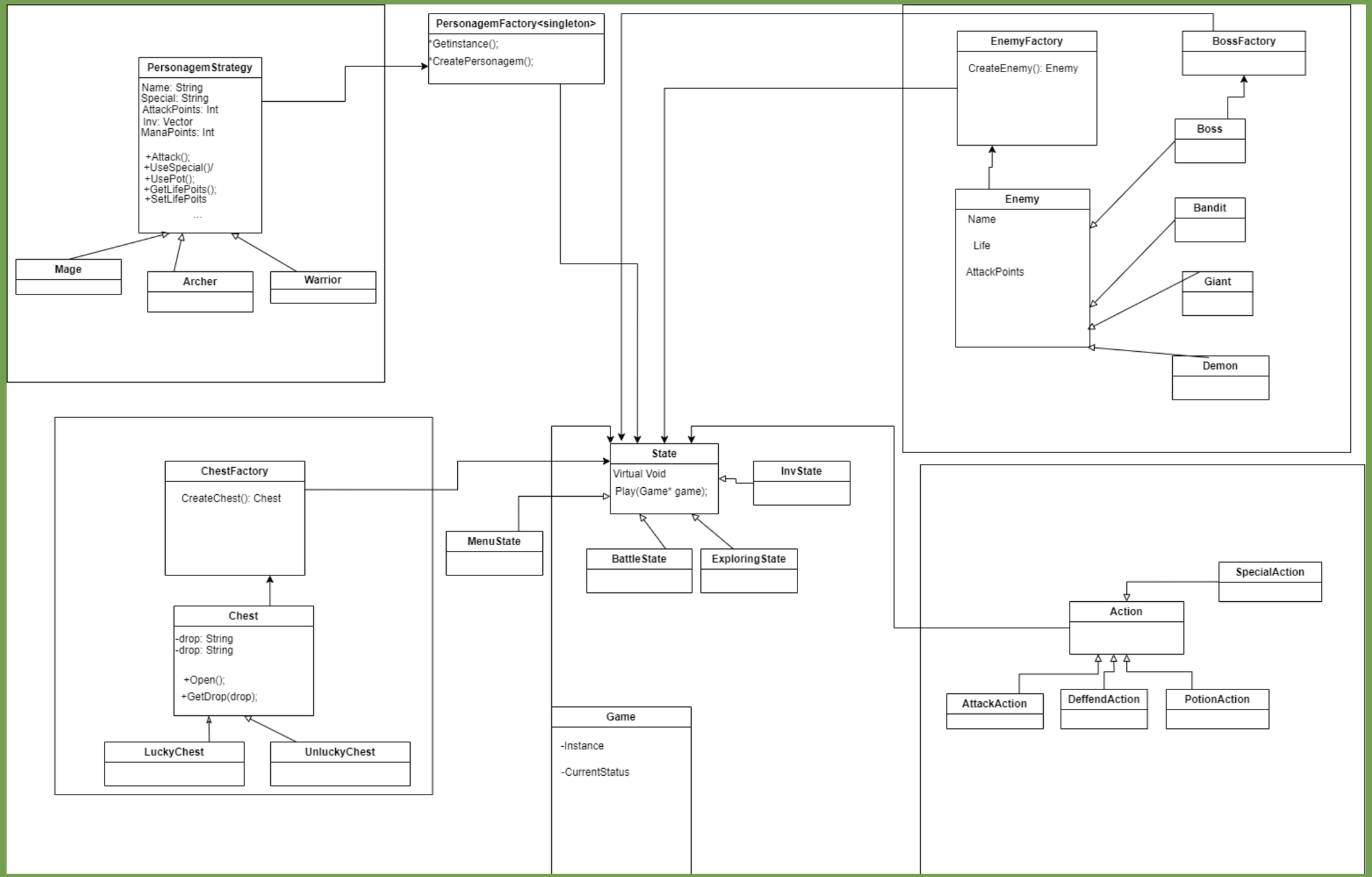
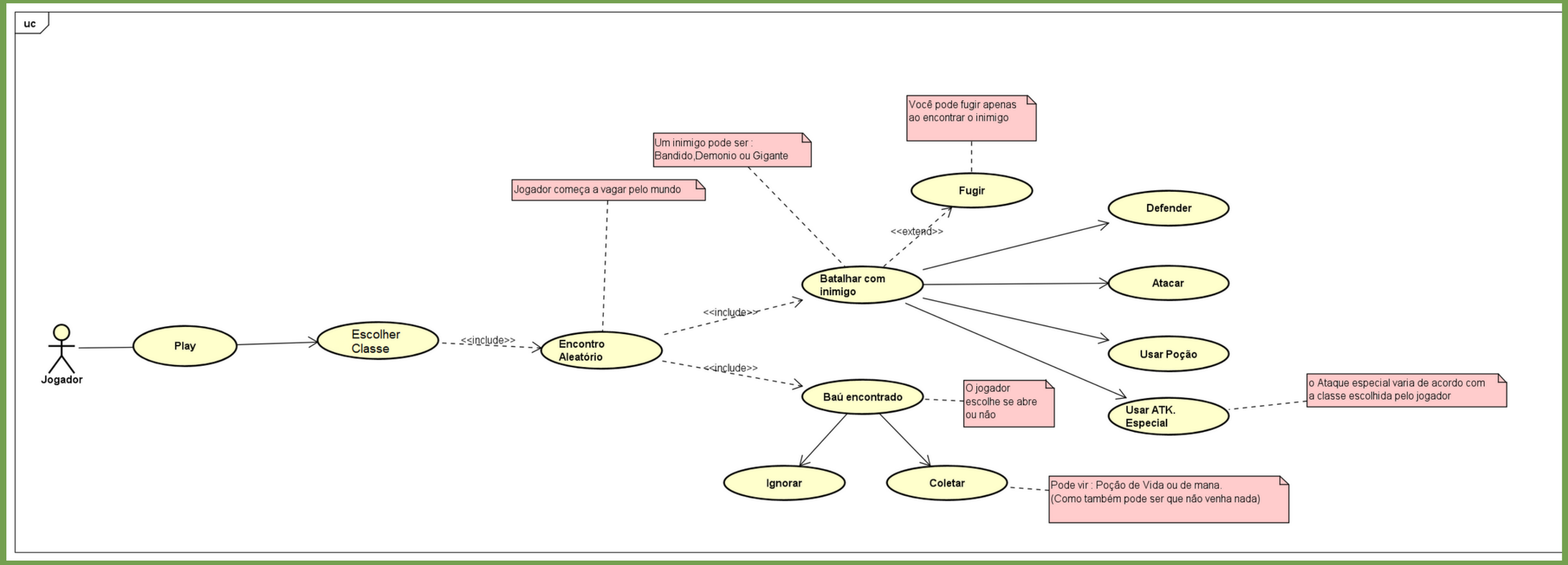

PersonagemStrategy
Name: String
Special: String
AttackPoints: Int
Inv: Vector
ManaPoints: Int
+Attack();
+UseSpecial()/
+UsePot();
+GetLifePoits();
+SetLifePoits
...
Mage
Archer
Warrior
PersonagemFactory<singleton>
*Getinstance();
*CreatePersonagem();
EnemyFactory
CreateEnemy(): Enemy
BossFactory
Boss
Enemy
Name
Life
AttackPoints
Bandit
Giant
Demon
ChestFactory
CreateChest(): Chest
Chest
-drop: String
-drop: String
+Open();
+GetDrop(drop);
LuckyChest
UnluckyChest
State
Virtual Void
Play(Game* game);
InvState
MenuState
BattleState
ExploringState
Game
-Instance
-CurrentStatus
SpecialAction
Action
AttackAction
DeffendAction
PotionAction

Diagrama de Caso de Uso

SIM

NÃO


uc
Você pode fugir apenas ao encontrar o inimigo
Um inimigo pode ser : Bandido,Demonio ou Gigante
Jogador começa a vagar pelo mundo
Fugir
Defender
<<extend>>
Batalhar com inimigo
Atacar
Usar Poção
Usar ATK. Especial
o Ataque especial varia de acordo com a classe escolhida pelo jogador
Jogador
Play
Escolher Classe
<<include>>
Encontro Aleatório
<<include>>
<<include>>
Baú encontrado
O jogador escolhe se abre ou não
Ignorar
Coletar
Pode vir : Poção de Vida ou de mana.
(Como também pode ser que não venha nada)

Padrões Implementados

Singleton

Garante instância única do jogo e do personagem.

Method Factory

Usado na instanciação do personagem, inimigos e baús.

Facade

Usado para exibição da interface gráfica.

Strategy

Usado na implementação dos movimentos do jogador quando em batalha.

State

Usado para alterar entre o estado de Exploração e de Batalha.

Singleton

```
class PersonagemFactory {
private:
    static std::shared_ptr<PersonagemFactory> instance;

    PersonagemFactory() {} // Construtor privado para evitar instanciação direta

public:
    static std::shared_ptr<PersonagemFactory> getInstance() {
        if (!instance) {
            instance = std::shared_ptr<PersonagemFactory>(new PersonagemFactory());
        }
        return instance;
    }

    std::shared_ptr<Personagem> createPersonagem() {
         int escolha;
        std::cout << "Selecione o tipo de personagem: " << std::endl;
        std::cout << "1. Mago" << std::endl;
        std::cout << "2. Guerreiro" << std::endl;
        std::cout << "3. Arqueiro" << std::endl;
        std::cout << "Digite aqui a opção: ";
        std::cin >> escolha;

        switch (escolha) {
            case 1:
                return std::make_shared<Mago>("Mago", 50, 100, 200,8);
            case 2:
                return std::make_shared<Guerreiro>("Guerreiro", 40, 80, 250,7);
            case 3:
                return std::make_shared<Arqueiro>("Arqueiro", 35, 120, 180,5);
            default:
                std::cout << "Opção inválida. Criando um Guerreiro por padrão." << std::endl;
                return std::make_shared<Guerreiro>("Guerreiro", 40, 80, 250,3);

    }
  }
};

std::shared_ptr<PersonagemFactory> PersonagemFactory::instance = nullptr;
```

Method Factory

```
#ifndef CHESTFACTORY_H
#define CHESTFACTORY_H

#include "Chest.h"

Chest *create_random_chest() {
    int random = rand() % 11; // gera um núm
    if (1) {
        return new ExplosiveBox();
    } else if (random <= 5) {
        return new LootBox();
    } else {
        return new EmptyChest();
    }

};
#endif
```

State

```
class GameState {
public:
  virtual ~GameState() {}
  virtual void play(Game *game) = 0;
};

class ExploringState : public GameState {
public:
  void play(Game *game) override;
};

class MenuState : public GameState {
public:
  void play(Game *game) override;
};


class BattleState : public GameState {
private:
  std::unique_ptr<Enemy> enemy;
  bool isPlayerTurn;
  std::map<int, std::unique_ptr<Action>> actions;

public:
  void play(Game *game) override;
};
```

Strategy

```
class Action {
public:
  virtual void execute() = 0;
  virtual ~Action() {}
};

class AttackAction : public Action {
private:
  Personagem &personagem; // Referência para o personagem principal
  Enemy &enemy;           // Referência para o inimigo

public:
  AttackAction(Personagem &character, Enemy &enemy)
      : personagem(character), enemy(enemy) { … }
  void execute() override { … }

private:
  int calculateDamage(int attackPoints) { … }
};
class DefendAction : public Action { … };
class PotionAction : public Action {
private:
  std::shared_ptr<Personagem> personagem;

public:
  PotionAction(std::shared_ptr<Personagem> character) : personagem(character) {}

  void execute() override { … }
};

class EspecialAction : public Action {
private:
  Personagem &personagem; // Referência para o personagem principal
  Enemy &enemy;           // Referência para o inimigo

public:
  EspecialAction(Personagem &character, Enemy &enemy)
      : personagem(character), enemy(enemy) {}

  void execute() override { … }

private:
  int calculateDamage(int attackPoints) { … }
};
```

Facade

```
void ExibirTelaMago() {
    sf::RenderWindow window(sf::VideoMode(800, 600), "RPG");

    sf::Texture texture;
    if (!texture.loadFromFile("AncestralMago.png")) {
        // Tratar erro ao carregar a textura
        return;
    }

    sf::Sprite sprite(texture);

    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed ||
                event.type == sf::Event::KeyPressed ||
                event.type == sf::Event::MouseButtonPressed) {
                window.close();
            }
        }

        window.clear();
        window.draw(sprite);
        window.display();
    }
};
```

Interface

SIM

NÃO


ESCOLHA SUA CLASSE
1. MAGO 2. GUERREIRO 3. ARQUEIRO


VOCÊ ENCONTRO UM INIMIGO
HP : ?????
ATAQUE : ????


VOCÊ ENCONTRO UM INIMIGO
HP : ?????
ATAQUE : ????


BÁU ENCONTRADO
DESEJA ABRIR O BÁU?
1. SIM
2. NÃO


POÇÃO DE MANA ENCONTRADA


POÇÃO DE VIDA ENCONTRADA


BAÚ VAZIO


BÁU EXPLOSIVO ENCONTRADO


VOCÊ ENCONTRO UM INIMIGO
HP : ?????
ATAQUE : ????

Melhorias Futuras

Exploração

Desenvolver um sistema de exploração mais interativo ao jogador.

Eventos

Adicionar novos tipos de eventos no jogo.

Equipe

Anderson Lucas

Full-stack, Design UI

Allan Pontes

Back-end, Design UI

Lucas Daris

Back-end, Sound