Projeto Integrador II - Dalibor, um jogo e aprendizado

Emanuel Felipe , Gabriel Rodrigues, Isabella Saueressig Ricci, Raul Carneiro, Ricardo Lewkowicz

Instituto de ensino superior de Brasília (IESB)

Abstract. *This paper discusses the development of an application that presents a Role-Playing videogame as its main topic. Furthermore, this paper will also show the systems presented in the application, the methods used for calculating data, as well as the demonstration of all the tests made throughout the project, which also made a big impact on data balancing.*

Resumo. *Esse artigo discorre sobre o desenvolvimento de um aplicativo que apresenta um jogo do gênero RPG como seu principal tópico. Nele também serão abordados os sistemas presentes no aplicativo, os métodos utilizados para o cálculo de dados, assim como a demonstração dos testes feitos ao longo do projeto, que influenciaram na questão do balanceamento de dados.*

1. Introdução

O conceito de *Role Playing Game* (RPG) **[SITOWSKI, 2015]**, pro português traduzido como "jogo narrativo", é um gênero de jogo que vem de partidas puramente abstratas como o *Dungeons & Dragons* **[GYGAX e ARNESON 1974]**, onde um grupo de jogadores se une e interpretam situações fictícias e fantásticas sem apoio material algum, objetivando a criatividade e interpretação de cada pessoa envolvida com o intuito de um entretenimento construtivo.

A evolução do gênero de jogos RPG promoveu produtos cada vez mais rentáveis ao mercado, com uma proposta que aproxima o jogador das obras de ficção medieval-fantasiosa populares, como por exemplo a saga *Senhor dos Anéis* **[J. R. R. TOLKIEN 1954],** antes mesmo de sua adaptação às telas de cinema.

A invenção dos computadores pessoais, inaugurada pela Apple, foi um avanço considerável não somente à tecnologia, mas também ao mundo todo e às empresas de jogos que rapidamente se consolidaram em torno da possibilidade de virtualmente renderizar o conteúdo que existia somente na imaginação dos jogadores de RPG com jogos como *The Legend of Zelda* **[MIYAMOTO e TEZUKA 1986]**, *Dragon Quest* **[HORII 1986]** e a série de jogos *Ultima* **[GARRIOTT 1980]**.

Esse mercado emergente, então, se tornou cada vez maior e com o avanço da Internet foi possível conectar todos esses jogadores em uma mesma partida, inaugurando o conceito de Massive Multiplayer Online (MMO) RPG, traduzido como Jogo de RPG em Massa para Multijogadores, com jogos como *Ultima Online* **[GARRIOTT 1997]** , *Runescape*[**GOWER 1998**] e *World of Warcraft* **[PARDO 2004]**.

Com o início da pandemia COVID-19 **[DIOGO, 2021]**, os eventos sociais, que envolvem uma grande quantidade de pessoas em um mesmo local, tornaram-se cada vez

mais escassos. Entre esses eventos, as jogatinas de RPG de mesa em ludotecas são um exemplo de interação social cortada pela pandemia. Esse fato é justificado pelo aumento de casos de contaminação e morte causados pelo vírus em questão, chegando a um número impactante de 477.307 mortes atualmente **[G1, 2021]**, o que impossibilitava a realização de eventos como esses.

Visando uma forma de promover o isolamento social, evitando a aglomeração de pessoas, será desenvolvido um aplicativo que apresentará um jogo no estilo RPG como destaque. Esse aplicativo será uma alternativa para as pessoas que costumam sair de casa para jogar uma partida de RPG de mesa, trazendo uma experiência similar sem a necessidade de aglomeração.

2. Objetivo Geral

Nosso projeto é tomar a proposta base que o gênero RPG oferece, com seu mundo fantástico e histórias épicas, evoluindo o conceito para os dispositivos móveis e as inovações que o jogo embarcado em um celular ou tablet pode conter como meio de promover espaço neste mercado e proporcionar entretenimento ao público gamer.

3. Objetivo Específico

O trabalho será concluído mediante as seguintes etapas:

3.1. Construção da interface do aplicativo via Android Studio;

3.2. Criação de personagens e suas características via Kotlin em Android Studio;

3.3. Modelo de batalha entre personagem de jogador e monstro criado pelo programa como meio de evolução do personagem;

3.3.1. Conceito de “Masmorra”, espaço onde o combate será promovido de forma mais desafiadora e recompensante;

3.4. Geolocalização do jogador como meio de promover atividades ímpares em posições geográficas diferentes;

3.5. Integração da aplicação com banco de dados Redis;

3.6. Tutoriais e sugestões ao jogador em forma de Chatbot.

4. Referencial Teórico

4.1. Requisições

A integração entre o usuário e o servidor está sendo feita por requisições GET e POST via API e hospedagem Heroku. Atualmente contamos com as seguintes requisições:

- GET (“/”) - Requisição que gera a lista das demais requisições disponíveis em nossa API.

- POST ("/jogadores/criarjogador") - Envia-se uma classe, nome e elemento ao servidor para a criação de um novo personagem.
- GET (“/jogadores”) - Disponibiliza uma lista de todos os jogadores cadastrados e seus atributos.
- GET (“/monstros”) - Disponibiliza uma lista de todos os monstros cadastrados e seus atributos.
- POST ("/batalha/PERSONAGEMX") - Envia-se um personagem para uma batalha, a função de batalha então monta um monstro que possui atributos calibrados com o nível do jogador.
- POST (“/loja/PERSONAGEMX/OPÇÃOY”) - Requisita-se a compra de um item OPÇÃOY ao inventário de PERSONAGEMX, caso o personagem possua moedas o suficiente.
- POST (“/inventario/PERSONAGEMX”) - Recebe-se o inventário do PERSONAGEMX.

4.2. Sistema de Personagens

O primeiro passo do usuário no jogo é criar um personagem, este personagem é um avatar do jogador que irá praticar todas as atividades que o aplicativo tem a oferecer, a principal delas sendo as batalhas. O personagem apresenta 5 atributos quantitativos distintos que influenciam sua performance:

- Vida: pontos de vida são subtraídos a cada golpe recebido durante o combate, quando se atinge 0 (zero) pontos de vida o personagem perde a batalha;
- Mana: qualificar o poder das magias do personagem;
- Ataque: define o poder dos ataques do personagem;
- Defesa: ao receber um ataque os pontos de defesa irão subtrair o ataque do oponente e consequentemente diminuindo a quantidade de pontos de vida reduzidos;
- Velocidade: define qual parte irá atacar primeiro ao início de um combate;
- Elemento: define um elemento de afinidade do personagem entre Água, Fogo, Natureza, Ordem, Caos ou Luz. A mecânica dos elementos será exposta posteriormente no Sistema de Batalha;
- Moedas de Ouro: dinheiro recompensado ao jogador e utilizado no jogo.

Durante a criação do personagem o jogador pode optar por três classes distintas: Cavaleiro, Arqueiro e Mago. Cada classe apresenta características e atributos iniciais diferentes que, por consequência, introduzem vantagens e desvantagens a cada uma delas. O Cavaleiro apresenta atributo Ataque elevado, porém, atributo Mana reduzido. Já o Arqueiro apresenta Velocidade elevada e Vida reduzida. Por fim, o Mago possui Defesa reduzida, mas com Mana elevada.

O personagem apresenta a opção de selecionar um Ídolo, item especial que possibilita a mudança do elemento do personagem. Esse item também fornece o aumento do poder de magias que apresentam o mesmo elemento do Ídolo atual.

Todo personagem possui um atributo numérico inteiro Nível que qualifica seu avanço no jogo, onde jogadores com nível maior possuem personagens mais fortes que jogadores de nível menor. Para se evoluir o nível do personagem precisa-se antes atingir uma meta de Pontos de Experiência, atributo recompensado às vitórias de batalhas, este que mede o progresso do personagem entre seu nível atual e o próximo nível.

Após a criação do personagem é possível acessar um menu que apresenta todas as características do personagem, como os atributos atuais, a classe, as magias disponíveis e a quantidade necessária de experiência para alcançar o próximo nível.

4.3. Sistema de Itens

O personagem é capaz de utilizar itens que o auxiliam durante as batalhas, denominamos estes itens por "Elixir". Os elixires podem ser adquiridos em troca das moedas de ouro do jogador em uma loja no jogo.

Um Elixir pode ser consumido para fornecer um bônus temporário em um atributo específico do jogador, ou seja, um Elixir de Vida, por exemplo, pode aumentar os pontos de vida de um personagem e um Elixir de Mana pode aumentar seus pontos de Mana.

4.4. Sistema de Batalha

O aplicativo apresenta um sistema de batalha, na qual o jogador participa de embates contra monstros visando o ganho de pontos de experiência que, por consequência, promovem o aumento do nível e aperfeiçoamento dos atributos do personagem.

As batalhas ocorrem em turnos feito um jogo de xadrez, de forma a um participante agir e o outro sucessivamente, sem haver concorrência entre seus movimentos. O participante da batalha que possuir a maior Velocidade irá iniciar o combate podendo, por exemplo, efetuar o primeiro golpe.

O personagem pode efetuar quatro tipos de ações diferentes durante uma batalha: Ações Básicas, Magias, Inventário e Fugir.

- A opção Ações Básicas é dividida em:
    - Atacar: possibilita ao personagem efetuar um ataque básico no adversário, um golpe fraco mas que não possui custo ao personagem;
    - Defender: coloca o personagem em uma posição de defesa, possibilitando uma redução na força do ataque causado pelo adversário.

- Magias: expõe ao jogador a lista de magias do personagem, as magias são habilidades que consomem pontos de Mana e geralmente removem mais pontos de vida do adversário;

- Inventário: possibilita o uso de itens presentes no inventário do personagem para trazer vantagens a ele na batalha, como aumentar seus pontos de atributos ou aumentar a poder de ataque das magias;

- Fugir: nesta opção o personagem possui uma determinada chance de evadir do combate e evitar a derrota.

O sistema de batalhas segue o método do “Pedra - Papel - Tesoura”**[BOSSCHE, 2010]**, em que cada elemento do combate apresenta uma fraqueza e força. Nesse caso, esses elementos são representados em seis tipos diferentes: Fogo, Água, Natureza, Ordem, Caos e Luz. O método segue da seguinte forma:

- Fogo é forte contra Natureza;
- Natureza é forte contra Água;
- Água é forte contra Fogo.

Esses três elementos seguem a teoria do método da "Pedra - Papel - Tesoura". Os outros dois elementos, Ordem e Caos, são considerados elementos neutros, em que ambos não apresentam vantagens ou desvantagens contra os três elementos principais, porém, são capazes de neutralizar uns aos outros. Ou seja:

- Ordem é forte contra Caos;
- Caos é forte contra Ordem.

Já o elemento Luz é ligado a magias de cura, contudo não possui vantagens ou fraquezas em relação aos demais elementos.

Com o sistema “Pedra-Papel-Tesoura” esclarecido o jogador consegue utilizar dessa mecânica para ganhar uma vantagem na batalha ao utilizar magias do elemento superior ao de seu adversário para causar dano extra, contudo o elemento do adversário não é exposto ao jogador, o que força este a testar magias de diferentes elementos para, assim, descobrir a fraqueza de seu adversário.

O jogador é vitorioso na batalha ao fazer os pontos de vida de seu adversário atingirem 0 ou inferior, a recompensa de sua vitória é composta de pontos de experiência e dinheiro. No entanto, caso o jogador venha a ter seus pontos de vida zerados ele é derrotado e perde uma quantidade do seu dinheiro atual.

4.5. Sistema de Monstros e Masmorras

A oposição do jogador nas batalhas é feita pelo monstro, também um personagem, porém não jogável. Por ser um personagem, o monstro compartilha de alguns atributos pertencentes também ao jogador, como os pontos de vida, elemento, magias e atributos baseados na dificuldade da Masmorra (ambiente onde a batalha ocorre).

Por possuírem elemento específico, os monstros também são expostos à mecânica “Pedra-Papel-Tesoura” tanto ao seu favor como ao contrário disso. Caso o

monstro utilize magia do mesmo elemento do jogador o efeito será enfraquecido, mas se utilizar de magia de elemento superior ao do jogador, haverá um bônus atribuído ao efeito da magia.

Existe um tipo especial de monstro denominado “Chefe”, este possui um grau de dificuldade e recompensa elevados obrigando, assim, o jogador a arquitetar melhor seus movimentos durante a batalha.

Para ingressar em uma batalha com um monstro, o jogador deve buscar por ícones de Masmorra que estarão expostos na extensão ao Google Maps disponível em nosso aplicativo. Cada masmorra apresenta uma temática e grau de dificuldade específicos, este último sendo representado pelo nível recomendado para ingressar na masmorra selecionada. A masmorra de chefe é evidenciada com um ícone diferenciado.

5. Metodologia

Para a composição das funções que guiam a aplicação, nos baseamos em fórmulas e cálculos matemáticos que preservam certa previsibilidade do comportamento dos personagens jogadores e monstros, tanto durante a batalha, como também a métrica que dita a evolução do personagem do jogador enquanto recebe cada vez mais pontos de experiência e, por consequência, aumenta seu nível e atributos.

Testes foram providenciados com o intuito de expormos de forma prática a teoria escrita nas fórmulas e cálculos que guiam o prosseguimento do jogo, de forma a mantermos atenção nos resultados obtidos durante a criação da aplicação, assim garantindo cada vez mais um jogo balanceado, onde o jogador não é impossibilitado de evoluir por conta de monstros fortes demais e também não é fácil demais o desenvolvimento de um personagem por só haver monstros fracos demais para o personagem.

5.1 Fórmulas

No sistema de Batalha, diversas variáveis, que apresentam valores diferentes, são utilizadas para conduzir fatores diferentes na luta. Variáveis como o dano do atacante, os pontos de vida do alvo e a velocidade do atacante são alguns exemplos de variáveis utilizadas para o fluxo da batalha. No entanto, para que seja possível utilizá-las, é necessário o uso de fórmulas que vão transformar essas variáveis em valores balanceados para o equilíbrio da luta.

Nesse caso, fórmulas como o Cálculo do Dano Mágico e do Aumento de Nível são algumas reformuladas do jogo Pokémon, com alguns ajustes feitos a partir dos testes feitos no projeto, visando o balanceamento.

5.1.1 Cálculo do Dano Básico

Para calcular o valor do dano básico do atacante na batalha, é necessário fazer o uso de uma fórmula que utiliza-se do valor atual do atributo Ataque do atacante e do valor atual do atributo de Defesa do alvo. A fórmula está representada na imagem a seguir:

```
dano = (ataquePersonagem - 3 .. ataquePersonagem).random() - (defesaInimigo/2)
```

Figura 1. Exemplo da Fórmula de Cálculo do Dano Básico

A fórmula funciona da seguinte maneira: primeiramente, é definido um valor mínimo e máximo do atributo Ataque do atacante. O valor mínimo consiste no resultado da subtração do Ataque atual por três, enquanto o valor máximo consiste no valor do Ataque atual. Após a definição do valor mínimo e máximo, a definição do valor do Ataque será feita a partir da escolha aleatória entre os números presentes situados entre o valor mínimo e o valor máximo. Logo após a escolha do número, esse valor é subtraído pelo resultado da divisão do atributo Defesa atual do alvo por dois. Por fim, o resultado da subtração define o valor do dano que o alvo vai receber na batalha.

Em uma situação exemplo, o atacante apresenta um Ataque com valor de 45 e o alvo apresenta uma Defesa com valor de 50. Nesse caso, é feita a definição do Ataque mínimo e máximo, sendo respectivamente 42 e 45, e logo após, é definido um valor aleatório, que nesse caso, será o número 46. Em seguida, é feita a divisão da Defesa do alvo por 2, resultando no valor 25. Com isso, é feita a subtração entre o resultado da divisão com o valor do ataque, gerando o valor 21, que será o valor do dano que o alvo receberá na batalha.

5.1.2 Cálculo do Dano Mágico

Para calcular o valor do dano mágico do atacante na batalha, é necessário fazer o uso de uma fórmula que utiliza-se de diversos atributos. Entre eles, temos: o nível atual do atacante, o valor do dano básico da magia utilizada, o valor da Mana do atacante, o valor da Defesa do alvo, o elemento da magia utilizada, o elemento do atacante, e por último, o elemento do alvo.

Tirando os atributos anteriores, também será utilizado ferramentas que proporcionarão um acréscimo ou decréscimo no valor do dano, dependendo da situação atual do atacante. Nesse caso, essas ferramentas são: STAB, que consiste no bônus utilizado quando o atacante faz o uso de uma magia que apresenta o elemento similar ao do atacante, e o Efetivo, que consiste no bônus utilizado quando o atacante utiliza uma magia que apresenta um elemento forte contra o elemento do alvo. A fórmula está representada na imagem a seguir:

```
dano = (((((2 * nivelAtacante) + 2) * poderAtaque * (statusAtaqueAtacante)/statusDefesaVitima) / 50) + 2) * stab * bonusEfetivo
```

Figura 2. Exemplo da Fórmula de Cálculo do Dano Mágico

A fórmula funciona da seguinte maneira: primeiramente, é feito uma multiplicação do nível atual do atacante por dois, na qual o resultado da mesma é somado por mais dois. Logo após, o valor resultante é multiplicado pelo valor do dano básico da magia utilizada, e em seguida, o resultado dessa operação é multiplicado pelo resultado da divisão entre o valores da Mana atual do atacante e da Defesa atual do alvo. Com isso, o resultado dessa multiplicação é dividido por cinquenta, e logo após, é feita uma soma por dois. Por fim, o resultado dessa operação toda passa por uma multiplicação entre os bônus de batalha, resultando no valor do dano do ataque mágico.

Dependendo dos elementos da magia, do atacante e do alvo, os bônus de batalha podem passar por algumas alterações nos seus valores. Nesse caso, como explicado anteriormente, o bônus do STAB apenas estará presente caso o atacante apresentar o mesmo elemento da magia utilizada. Já em relação ao Efetivo, quando a magia utilizada apresenta o elemento que apresenta uma vantagem contra o elemento do alvo, o bônus contribuirá para o aumento do valor do dano mágico. Caso a magia apresenta o mesmo elemento que o monstro, o bônus não é aplicado ao valor final. E em situações em que a magia apresenta o elemento que possui uma desvantagem contra o elemento do alvo, o dano do ataque sofre um decréscimo, diminuindo o valor final do dano.

Em uma situação exemplo, o atacante apresenta uma Mana com valor de 50, o elemento Fogo e um nível com valor 20, o alvo apresenta uma Defesa com valor de 45 e o elemento Natureza, a magia utilizada apresenta um dano de 10 e o elemento Fogo. Nesse caso, é feita a multiplicação do nível do atacante por 2, e logo após, a soma do resultado por 2, resultando no valor 42. Com isso, esse valor é multiplicado pelo valor do dano da magia, resultando no valor 420, e logo após, é feita a multiplicação com o valor resultante da divisão entre a Mana e Defesa do atacante e do alvo, respectivamente, sendo 1,11, e por fim: 466,2.

Esse resultado é dividido por 50, resultando em 9,32, na qual é somado por 2, transformando em 11,32. Por fim, seguindo as condições de bônus de ataque, o valor final é multiplicado 4, resultando no valor 45,28, representando o dano final do ataque mágico.

5.1.3 Cálculo do Acréscimo da Defesa

Quando o personagem escolhe a opção de Defender na batalha, sua Defesa atual sofre um pequeno acréscimo temporário, visando reduzir o dano gerado pelo inimigo. Nesse caso, o valor atual da Defesa é multiplicado por 3, retornando ao valor anterior no fim do turno.

5.1.4 Cálculo da Cura

Para calcular o valor da cura, é necessário fazer o uso de uma fórmula que utiliza-se do nível atual do atacante e do valor atual do atributo Mana do atacante. A fórmula está representada na imagem abaixo:

```
cura = (nivelAtacante * statusMaxManaAtacante)/2
```

Figura 3. Exemplo da Fórmula de Cálculo do Cura

A fórmula funciona da seguinte maneira: primeiramente, é feita a multiplicação entre o nível do atacante e o valor do atributo Mana do atacante. Logo após, o resultado dessa operação é dividido por 2, gerando no valor final da cura.

Em uma situação exemplo, o atacante apresenta uma Mana com valor de 50, e um nível com valor 20. Nesse caso, é feita a multiplicação entre o nível e a Mana, resultando no valor 1000. Em seguida, esse valor é dividido por 2, resultando no valor final de 500. Com isso, os pontos de Vida do atacante passam por um acréscimo de 500 pontos.

5.1.5 Cálculo da Redução da Mana

Quando o personagem utiliza qualquer tipo de magia na batalha, é necessário fazer o uso de uma fórmula que irá reduzir os pontos de Mana atual do personagem, com o intuito de limitar o personagem de ficar abusando de usar as magias durante a batalha toda. A fórmula está representada abaixo:

```
this.pontosMana = this.pontosMana - valor / 2
```

Figura 4. Exemplo da Fórmula de Redução da Mana

A fórmula funciona da seguinte forma: é feita uma subtração entre o valor dos pontos de Mana atual e o resultado da divisão do dano gerado pela magia por dois. Com isso, os pontos da Mana recebem o resultado da operação. Essa operação ocorre sempre quando o personagem utiliza uma magia, no entanto, caso seus pontos de Mana chegarem a zero, ele não poderá utilizar mais magias durante a batalha.

Em uma situação exemplo, o personagem apresenta um valor de 40 pontos de Mana. Ao utilizar uma magia que gera um dano de 30, é feita a operação da redução, na qual os pontos são subtraídos pelo dano dividido por 2, resultando no valor 15. Com isso, é feita a subtração, fazendo com que os pontos de Mana sejam 25 após o uso da magia.

5.1.5 Cálculo do Aumento de Nível

O jogador recebe pontos de experiência ao vencer combates, o cálculo dos pontos de experiência recebidos é representado na seguinte fórmula:

```
val xpganho = monstro.nivel * 100
```

Figura 5. Exemplo da Fórmula de XP ganho

Nessa fórmula, o nível atual do monstro enfrentado é multiplicado por 100, resultando nos pontos de experiência que são somados a cada batalha vencida. Esse processo se repete até ser atingida uma meta, definida na fórmula abaixo:

```
xpProxNv += this.nivel * 100
```

Figura 6. Exemplo da Fórmula de XP Para o Próximo Nível

Nessa fórmula, o nível atual do personagem é multiplicado por 100, definindo o valor total de pontos de experiência necessários para atingir o próximo nível. Desta forma nós conseguimos manter um avanço de nível com dificuldade incremental, de forma que seja mais difícil atingir um nível mais alto e, assim, valorizando jogadores mais esforçados ou veteranos.

Quando o personagem aumenta o seu nível, os valores atuais dos seus atributos passam por uma fórmula para determinar os acréscimos que terão no novo nível. A fórmula apresenta apenas dois componentes para o seu cálculo: o valor base do atributo passando pelo acréscimo e o novo nível conquistado pelo personagem. No entanto, a classe do personagem também apresenta uma influência na fórmula em questão, influenciando no valor adicionado ao final da operação.

A fórmula está representada na imagem a seguir:

```
this.maxVida = ((2 * statusBaseVida) * nivel)/50 + nivel + 10
this.maxMana = ((2 * statusBaseMana) * nivel)/50 + nivel + 12
this.maxAtaque = ((2 * statusBaseAtaque) * nivel)/50 + nivel + 12
this.maxDefesa = ((2 * statusBaseDefesa) * nivel)/50 + nivel + 12
this.maxVelocidade = ((2 * statusBaseVelocidade) * nivel)/50 + nivel + 14
```

Figura 7. Exemplo da Fórmula de Cálculo do Aumento de Nível do Jogador

Como explicado anteriormente, a fórmula é utilizada para cada atributo do personagem, com elas apresentando diferenças no atributo utilizado e no último valor

adicionado na operação. Nesse caso, a fórmula em questão é utilizada para os personagens que apresentam a classe Arqueiro. Cada classe possui uma proficiência, ou vantagem, em um determinado atributo, assim como uma desvantagem em um outro atributo. Essa característica é demonstrada por meio da adição final da fórmula, na qual caso o personagem apresentar essa vantagem em um atributo, o valor padrão desse componente, que é o número 12, sofre um acréscimo de dois números, resultando no valor 14. Caso contrário, o valor sofre um decréscimo de dois números, resultando no valor 10. O valor padrão é apenas utilizado quando o personagem não apresenta uma vantagem e nem uma desvantagem no atributo em questão.

Em relação ao Arqueiro, essas características estão presentes no atributo Velocidade, na qual ele apresenta uma vantagem, e no atributo Vida, na qual ele apresenta uma desvantagem. Já em relação ao valor base do atributo, ele consiste no valor padrão que cada classe possui no nível 1, sendo ele utilizado para determinar as mudanças dos atributos no próximo nível.

A fórmula funciona da seguinte maneira: o valor base do atributo é multiplicado por 2, e seu resultado é multiplicado pelo novo nível do personagem. Logo após, o resultado dessa multiplicação é dividido por 50, com o valor resultante sendo somado pelo novo nível e, por fim, pelo valor final, variando dependendo da classe do personagem, como explicado anteriormente. O valor resultante da operação é definido como o novo valor do atributo, tornando o personagem mais forte que antes.

Em uma situação exemplo, o personagem aumenta de nível para o 30, e logo em seguida, passa pela fórmula do aumento de nível. Por ser da classe Arqueiro, ele apresenta como valores bases: 5, 6, 6, 6 e 7, para os atributos Vida, Mana, Ataque, Defesa e Velocidade, respectivamente. Com isso, os valores são introduzidos a fórmula, com a Vida apresentando 46 pontos, a Mana, o Ataque e a Defesa apresentando 49 pontos, e por fim, a Velocidade apresentando 52 pontos.

Essa fórmula é também utilizada para determinar os valores dos atributos dos monstros, com a diferença que o nível será determinado por um valor aleatório entre um valor mínimo e máximo, definido pelo nível atual da Masmorra e pela soma do nível atual por um, e o valor somado ao final é determinado aleatoriamente em um conjunto de números de 0 a 6. Essa operação está demonstrada na imagem abaixo:

```
this.maxVida = ((statusBaseVida) * (nivelMasmorra..nivelMasmorra+1).random())/2 + (0..6).random()
this.maxMana = ((statusBaseMana) * (nivelMasmorra..nivelMasmorra+1).random())/2 + (0..6).random()
this.maxAtaque = ((statusBaseAtaque) * (nivelMasmorra..nivelMasmorra+1).random())/2 + (0..6).random()
this.maxDefesa = ((statusBaseDefesa) * (nivelMasmorra..nivelMasmorra+1).random())/2 + (0..6).random()
this.maxVelocidade = ((statusBaseVelocidade) * (nivelMasmorra..nivelMasmorra+1).random())/2 + (0..6).random()
```

Figura 8. Exemplo da Fórmula de Cálculo do Aumento de Nível do Monstro

5.2 Testes

Durante o desenvolvimento do projeto, foram feitos diversos testes para conferir os resultados de cada ação do personagem nas batalhas. Os resultados dos testes encontram-se a seguir:

5.2.1 Teste - Ataque Básico

- Ao fim das 10 ações de Ataque Básico:
    - O Jogador iniciou o combate 9 vezes, com a última sendo iniciada pelo Monstro;
    - O Monstro conseguiu efetuar 4 ações;
    - O Jogador derrotou 6 Monstros;
    - O Jogador apresentou 10 de dinheiro na primeira batalha e 70 no final da última;
    - O Jogador aumentou de nível três vezes: do nível 1 pro 2 na segunda batalha, do nível 2 pro 3 na quarta batalha e do nível 3 pro 4 na décima batalha;
    - Os atributos Ataque, Defesa e Mana do Personagem iniciaram com 13 pontos na primeira batalha e finalizaram com 17 pontos na última batalha;
    - O atributo Vida do Personagem iniciou com 11 na primeira batalha e finalizou com 14 na última batalha;
    - O atributo Velocidade do Personagem iniciou com 15 na primeira batalha e finalizou com 19 na última batalha;
    - A média do dano infligido ao Monstro foi de 7 pontos;
    - O maior dano infligido pelo Personagem foi de 11 pontos, enquanto o menor foi de 4 pontos;
    - O Jogador sofreu 0 mortes no total.

5.2.2 Teste - Defesa

- Ao fim das 10 ações de Defesa:
    - O Jogador iniciou o combate 8 vezes, enquanto o Monstro iniciou 2 vezes;
    - Dois Monstros foram gerados durante o processo;
    - No total, os Monstros efetuaram 4 ataques no Jogador;
    - Um dos Monstros conseguiram derrotar o Jogador;
    - No começo dos testes, o Jogador apresentou 10 de dinheiro, já no final, ele apresentou 5 de dinheiro.

5.2.3 Teste - Ataque Mágico

- Ao fim das 10 ações de Magia:
    - O Jogador iniciou o combate 10 vezes;
    - O Jogador utilizou apenas o elemento Água;
    - No total, o Jogador derrotou 7 Monstros;
    - No total, os Monstros efetuaram 8 ações;
    - A geração dos Monstros atribuiu o elemento Água em dois Monstros, o elemento Fogo em dois Monstros e o elemento Caos em seis monstros;

- O Jogador efetuou dois ataques super efetivos (na lutas contra os Monstros que detinham o elemento Fogo) e oito ataques normais;
- O Jogador aumentou de nível três vezes: do nível 1 pro 2 na segunda batalha, do nível 2 pro 3 na quarta batalha e do nível 3 pro 4 na nona batalha;
- No total, o Jogador sofreu 0 mortes.

5.2.4 Comparação entre o Ataque Básico e o Ataque Mágico

Após a análise dos testes feitos, pode-se notar algumas diferenças nos desempenhos dos ataques. No uso do Ataque Básico, foi observado que o Monstro não apresentou uma quantidade considerável de efetuação de ações, ou seja, o Jogador efetuava um golpe fatal na sua primeira ação, deixando o Monstro incapacitado.

Já no uso da Magia, o dano do golpe dependia exclusivamente da comparação entre elementos do Jogador e do Monstro, onde ataques super efetivos tornavam-se fatais, entretanto as possibilidades de realização de ataques normais eram maiores, o que permitia uma janela de ataque do Monstro.

5.3 Balanceamento

Durante o desenvolvimento do projeto, foram feitos diversos experimentos para testar as as variáveis e fórmulas presentes no aplicativo, visando o equilíbrio entre elas. Devido a isso, foram feitas reformulações em alguns desses elementos, possibilitando o balanceamento no sistema de batalha.

5.3.1 Balanceamento da Fórmula do Cálculo do Dano Básico

Essa fórmula, como explicado anteriormente, é utilizada para definir o dano do ataque básico do personagem, fazendo o uso do atributo Ataque e Defesa do atacante e da vítima, respectivamente. O resultado esperado desse cálculo seria um número influenciado pela comparação entre o Ataque e a Defesa, sofrendo mudanças a partir dessa comparação.

Nesse caso, em uma situação que o inimigo apresenta a Defesa com um valor na mesma faixa do Ataque do atacante, a fórmula vai apresentar um valor “médio”, ou seja, um valor que não ultrapassa do Ataque atual, como também não é muito baixa. No entanto, caso a Defesa for maior que o valor do Ataque, o dano terá um valor muito mais baixo que o esperado, e isso se reflete também quando a Defesa é menor que o Ataque, resultando em um dano maior.

Anteriormente, a fórmula era representada da seguinte maneira:

```
dano = (statusAtaqueAtacante - 2 .. statusAtaqueAtacante + 2).random() * (statusAtaqueAtacante / statusDefesaVitima)
```

Figura 9. Exemplo da Fórmula Antiga de Cálculo do Dano Básico

Inicialmente, o dano era resultado de uma multiplicação entre o valor escolhido aleatoriamente dentro de um conjunto de números, e o resultado da divisão entre o Ataque do atacante e a Defesa da vítima. Esse conjunto é representado por números entre um valor mínimo e máximo, definidos a partir da subtração do Ataque por dois e pela soma do mesmo por dois, respectivamente.

O fator que prejudicou os resultados esperados da fórmula foi a divisão feita entre o Ataque e a Defesa. Nesse caso, em situações em que o jogador e o inimigo apresentavam os atributos com valores similares, devido a forma como a divisão passa um número inteiro invés de um número decimal, em muitas situações a Defesa da vítima não apresentaria nenhum efeito no cálculo do dano. Essa situação tornava o atacante extremamente forte, ignorando qualquer tipo de defesa que o inimigo apresentasse no momento.

Em uma situação exemplo, o jogador apresenta 20 de Ataque e o inimigo 19 de Defesa. Nesse caso, é feita a divisão entre esses dois valores, resultando em um número decimal 1,05, porém, por ele passar um número inteiro, esse valor apenas pega o primeiro número, ou seja, apenas gera o número 1. Com isso, o dano gerado pelo ataque básico sofre nenhuma penalidade causada pela defesa do inimigo, o que prejudica o fluxo da batalha.

Com esses problemas em mente, foram feitas algumas mudanças no cálculo, resultando na seguinte fórmula:

```
dano = (ataquePersonagem - 3 .. ataquePersonagem).random() - (defesaInimigo/2)
```

Figura 10. Exemplo da Fórmula Nova de Cálculo do Dano Básico

Nesse caso, foram feitas alterações no cálculo do valor mínimo e máximo do conjunto, na divisão da Defesa do inimigo e na operação entre os dois valores. O valor mínimo é resultado da subtração do valor atual do Ataque do personagem por três e o valor máximo é o mesmo valor do Ataque. Nessa fórmula, após feita a definição do valor aleatório do ataque, é feita uma subtração com o resultado da divisão da Defesa por dois, resultando no dano do ataque básico.

Diferente da fórmula antiga, essa operação evita as situações em que a defesa não apresenta uma influência no ataque, possibilitando as situações de geração de dano mencionadas anteriormente. Então, mesmo quando a defesa apresentar um valor muito baixo, ela ainda terá algum efeito no cálculo do dano.

Utilizando o mesmos valores do exemplo demonstrado anteriormente, será feita a subtração do Ataque 20 pelo resultado da divisão da Defesa 19 por 2. Nesse caso, a divisão gera um valor 9,5, que transformado para um valor inteiro torna-se 9, o que resulta no valor 11, sendo definido como o dano do ataque básico.

Como mostrado no exemplo anterior, a Defesa mantém a sua influência na definição do dano, o que corrige o problema da fórmula antiga, mantendo o balanceamento do Ataque Básico.

5.3.2 Balanceamento da Fórmula do Aumento de Nível

Essa fórmula, como mostrada anteriormente, é utilizada para calcular a mudança dos atributos de um personagem, sempre que ocorre o aumento de nível. Dentro dela, são utilizados os valores base de um atributo, assim como o nível atual do personagem para determinar a mudança no valor. Anteriormente, a fórmula apresentava a seguinte estrutura:

```
this.maxVida += ((2 * statusBaseVida) * nivel)/100 + nivel + 8
this.maxMana += ((2 * statusBaseMana) * nivel)/100 + nivel + 10
this.maxAtaque += ((2 * statusBaseAtaque) * nivel)/100 + nivel + 10
this.maxDefesa += ((2 * statusBaseDefesa) * nivel)/100 + nivel + 10
this.maxVelocidade += ((2 * statusBaseVelocidade) * nivel)/100 + nivel + 12
```

Figura 11. Exemplo da Fórmula de Cálculo do Aumento de Nível

Inicialmente, os valores finais que são adicionados no fim da operação apresentavam os seguintes valores: 8, 10 e 12, na qual cada um era utilizado para atributos que a classe do personagem tem desvantagem, neutralidade e vantagem, respectivamente, como explicado anteriormente. No entanto, vendo que o personagem apresenta atributos um pouco mais fracos que os dos Monstros, esses valores sofreram um aumento e o valor que divide foi reduzido também. Com isso, os novos valores são: 10, 12 e 14 para os bônus e 50 para o divisor.

Com essas alterações feitas, a fórmula apresenta a nova estrutura seguinte:

```
this.maxVida = ((2 * statusBaseVida) * nivel)/50 + nivel + 10
this.maxMana = ((2 * statusBaseMana) * nivel)/50 + nivel + 12
this.maxAtaque = ((2 * statusBaseAtaque) * nivel)/50 + nivel + 12
this.maxDefesa = ((2 * statusBaseDefesa) * nivel)/50 + nivel + 12
this.maxVelocidade = ((2 * statusBaseVelocidade) * nivel)/50 + nivel + 14
```

Figura 12. Exemplo da Fórmula de Cálculo do Aumento de Nível

Como mostrado acima, foram feitas as mudanças mencionadas anteriormente, tornando o personagem mais equilibrado para as batalhas. Já em relação ao Monstro, para seguir o balanceamento com o Jogador, foram adicionados um valor aleatório, de 1 a 6, que é somado ao fim da sua operação de nível, assim como a definição do nível por meio da escolha aleatória entre o valor do nível atual e da soma dele por 1.

Nesse caso, a fórmula do Monstro apresentava a seguinte estrutura anteriormente:

```
this.maxVida = ((2 * statusBaseVida) * nivel)/50 + nivel + 10
this.maxMana = ((2 * statusBaseMana) * nivel)/50 + nivel + 12
this.maxAtaque = ((2 * statusBaseAtaque) * nivel)/50 + nivel + 12
this.maxDefesa = ((2 * statusBaseDefesa) * nivel)/50 + nivel + 12
this.maxVelocidade = ((2 * statusBaseVelocidade) * nivel)/50 + nivel + 14
```

Figura 13. Exemplo da Fórmula de Cálculo do Aumento de Nível

Enquanto a nova fórmula é representada da seguinte maneira:

```
this.maxVida = ((statusBaseVida) * (nivelMasmorra..nivelMasmorra+1).random())/2 + (0..6).random()
this.maxMana = ((statusBaseMana) * (nivelMasmorra..nivelMasmorra+1).random())/2 + (0..6).random()
this.maxAtaque = ((statusBaseAtaque) * (nivelMasmorra..nivelMasmorra+1).random())/2 + (0..6).random()
this.maxDefesa = ((statusBaseDefesa) * (nivelMasmorra..nivelMasmorra+1).random())/2 + (0..6).random()
this.maxVelocidade = ((statusBaseVelocidade) * (nivelMasmorra..nivelMasmorra+1).random())/2 + (0..6).random()
```

Figura 14. Exemplo da Fórmula de Cálculo do Aumento de Nível

Com essas mudanças feitas, tanto o Monstro quanto o Jogador estão balanceados para manter o equilíbrio da batalha.

6. Resultados Obtidos

Tendo em mente todos os tópicos abordados neste artigo, desde as mecânicas e funcionalidades do jogo apresentadas no Referencial Teórico, aos cálculos e testes propostos na Metodologia, o resultado do esforço dos membros componentes deste Projeto Integrador demonstra um jogo divertido que utiliza de práticas existentes no mercado de jogos hodierno.

Há dificuldade e sensação de recompensa durante as atividades do jogo, as batalhas demandam raciocínio do jogador e a adversidade proposta no jogo é suficiente para cativar a noção de desafio. Os vários componentes da aplicação, quando interpretados em conjunto, demonstram complexidade não restritiva ao público comum, porém razoável e engajadora.

Entretanto, mesmo com as fórmulas e cálculos apresentados atingindo nossas expectativas, houve um erro com a manipulação dos retornos oriundos da API que impossibilitou a apresentação dos dados da batalha ao usuário, mesmo com os dados da

batalha devidamente processados. Espera-se a correção para a apresentação do Projeto à banca, contudo junto a este artigo está uma aplicação defeituosa.

Logo, compreende-se um Projeto Integrador com resultados quase satisfatórios, expondo um sistema complexo criado com um intuito claro de entretenimento que foi cumprido e manteve possibilidade da atualização do jogo com tendências pendentes, como monetização e ambiente competitivo, assim como a finalização da parte visual do projeto.

7. Conclusão

Foram atingidas duas metas estabelecidas pela equipe: o desenvolvimento de uma aplicação de entretenimento e o aprendizado profundo proposto pelo Projeto Integrador. Juntos, conseguimos estudar diversas tecnologias complementares entre si que proporcionaram uma estrutura robusta.

Resta, ainda, ânsia para o desenvolvimento de um sistema multijogador com comunidade composta de múltiplos usuários igualmente envolvidos no ambiente da aplicação. Diversos jogos hoje em dia possuem fóruns e campos exclusivos para os jogadores interagirem e desenvolverem conhecimento acerca do jogo, criando um verdadeiro ecossistema que quebra a barreira tecnológica imposta pelo dispositivo hospedeiro do aplicativo.

8 . Bibliografia

PETERSON, Jon. Website que mostra a linha do tempo do jogo. Dungeons & Dragons, 2021. Disponível em: History: Forty Years of Adventure | Dungeons & Dragons - Acessado em: 18 de Março de 2021.

DOUGHAN, David. Bibliografia sobre o autor J.R.R Tolkien. The Tolkien Society, 2021. Disponível em: Biography – The Tolkien Society - Acessado em: 18 de Março de 2021.

WATERS, Darren. Artigo sobre o número de vendas de Dungeons & Dragons. BBC News, 2004. Disponível em: BBC NEWS | What happened to Dungeons and Dragons? - Acessado em: 18 de Março de 2021.

GADELHA, Julia. Artigo sobre a Evolução dos Computadores. Universidade Federal Fluminense, ??. Disponível em: A EVOLUÇÃO DOS COMPUTADORES - Acessado em: 18 de Março de 2021.

WELL, Thomas. Artigo sobre a história do Dragon Quest. Thomas Well, 2019. Disponível em: Enix Origins: The Story Behind Dragon Quest - Acessado em: 18 de Março de 2021.

WEBSTER, Andrew. Timeline da franquia The Legend of Zelda. The Verge, 2016. Disponível em: 30 years of Zelda: a timeline of the legend so far - Acessado em: 18 de Março de 2021.

MOWATT, Todd. Entrevista do Shigeru Miyamoto, criador do The Legend of Zelda. Amazon, 2007. Disponível em: In the Game: Nintendo's Shigeru Miyamoto - Acessado em: 18 de Março de 2021.

BARTON, Matt. Artigo sobre a história dos jogos RPG para computadores. Gamasutra, 2007. Disponível em: The History of Computer Role-Playing Games Part 2: The Golden Age (1985-1993) - Acessado em: 18 de Março de 2021.

DODSON, Sean. Artigo sobre o desenvolvimento de RuneScape. The Guardian, 2003. Disponível em: Rune to move | Games - Acessado em: 18 de Março de 2021.

WILLIAMS, Mike. Artigo sobre o desenvolvimento do World of Warcraft. USGamer, 2019. Disponível em: How World of Warcraft Was Made: The Definitive Inside Story of Nearly 20 Years of Development - Acessado em: 18 de Março de 2021.

DIOGO, Darcianne. Notícia sobre o número de casos e mortes de COVID-19. G1, 2021. Disponível em: Covid-19: DF registra 1.322 casos em 24 horas e total de mortes chega a 5.206 - Acessado em: 18 de Março de 2021.

SITOWSKI, André. Artigo sobre o gênero RPG. Medium, 2015. Disponível em: RPG — Tudo que você precisa saber sobre esse tal Role Playing Game - Acessado em: 18 de Março de 2021.

BOSSCHE, Andrew. Artigo sobre o uso do método “Pedra-Papel-Tesoura” nos jogos. Gamasutra, 2010. Disponível em: Analysis: Everything is Rock, Paper, Scissors - Acessado em: 15 de Maio de 2021.

POKEDREAM, Calculadora de Status Base do jogo Pokémon Gold. Pokedream, ??. Disponível em: Gold/Silver - Stat Calculator - PokeDream - Acessado em: 25 de Maio de 2021.

CHARD, Ben; LAU, Vincent. Artigo sobre a explicação do cálculo de Status do jogo Pokémon Shield. Gamer Guides, 2019. Disponível em: Understanding Stats - Pokémon 101 - Advanced Trainer Info | Pokémon: Sword & Shield - Acessado em: 25 de Maio de 2021.

G1. Reportagem sobre o número de casos e mortes de COVID19. G1, 2021. Disponível em: Brasil registra 2.693 novas mortes por Covid após feriado prolongado; casos passam de 17 milhões no total - Acessado em: 9 de Junho de 2021.