**TEMA:** Brawhalla

**TEXTO DE REQUISITOS:**

O software escolhido, pode ser utilizado para solucionar os seguintes problemas:

* Ausência da coordenação Mão-Olho;
* Escassez de atividades recreativas envolvendo jogos eletrônicos.

**Requisitos funcionais:**

**Definição:**

Requisitos funcionais são declarações de como um programa deve funcionar, desde sua função até seu desempenho e velocidade de processamento. Definindo o modo que o programa se comporta.

**Requisitos:**

**BANCO DE DADOS**

O primeiro requisito do software é a criação de um banco de dados que possua a capacidade de armazenar uma imensa quantidade de informações, para que assim o usuário seja capaz de criar uma conta em uma das diversas plataformas na qual o jogo se encontra, de modo que essa conta seja armazenada no banco de dados.

O banco de dados deve armazenar as informações em tempo real do que está acontecendo no software, como por exemplo: armazenar o elo dos jogadores em cada temporada assim como o processo para a promoção ao próximo elo.

**INTERFACE**

È necessário a criação de uma interface que demonstre:

* Os modos de jogo;
* As opções de customização da conta;
* Acesso às configurações e controles;
* Acesso a loja e ao passe de batalha;
* Seleção de personagens;
* O mapa selecionado.

**Algoritmo de colisão**

Devido a ser um jogo de luta é necessário a existência de um algoritmo que delimite o espaço que cada item ocupe no mapa, fazendo com que os jogadores não atravessem as paredes, que ataques sejam acertados e itens seja equipados.

**Algoritmo de Atributos**

Um algoritmo que defina o modo que os multiplicadores dos atributos influenciam as ações dos personagens, como por exemplo: ataques, vida ,velocidade de ataque, entre outras coisas.

**Requisitos não Funcionais:**

**Definição:**

Requisitos não funcionais são os requisitos relacionados ao uso do programa em quesitos como: desempenho, confiabilidade, segurança, manutenibilidade, usabilidade, entre outras aplicações. Dependendo do desenvolvedor escolher se ele irá atender a esses requisitos ou não.

**Requisitos:**

**Interface gráfica avançada**

Uma interface gráfica agradavel aos olhos é um exemplo de requisito não funcional, ja que não altera o funcionamento do programa em nenhum quesito ou área, servindo puramente para estética e fluidez na navegação do usuário.

**Servidores locais**

A criação de servidores locais serve apenas para que as partidas sigam de forma mais fluída e sem travamentos, além de facilitar o armazenamento de dados.

**Conexão de plataformas**

A conexão entre plataformas existe somente para facilitar a entrada do usuário no jogo, usando a conta da plataforma escolhida, além de impossibilitar a entrada de outras pessoas na conta do usuário.

### **Desempenho**

* O jogo deve ser capaz de suportar milhares de pessoas o acessando ao mesmo tempo.

### **Disponibilidade**

* O sistema deve ter uma disponibilidade mínima de 99,9%, o que significa uma janela de inatividade de no máximo 8,76 horas por ano.
* O tempo de recuperação do sistema após uma falha crítica deve ser inferior a 1 hora, para melhor experiência do usuário.

### **Usabilidade**

* A interface do sistema deve ser intuitiva e permitir que um novo usuário consiga realizar as principais funcionalidades em até 10 minutos de uso.

### **Manutenibilidade**

* O código do sistema deve ser modular e documentado de forma clara, para que novas funcionalidades possam ser implementadas com facilidade.
* O sistema deve ser fácil de atualizar sem causar interrupções nos serviços.

### **Portabilidade**

* O sistema deve ser capaz de ser executado em diferentes ambientes operacionais, como Windows, Linux e macOS.

### **Legalidade e Conformidade**

* O sistema deve estar em conformidade com a Lei Geral de Proteção de Dados (LGPD) no Brasil, garantindo a privacidade e segurança dos dados dos usuários.
* O sistema deve atender aos requisitos de acessibilidade exigidos pela legislação vigente.

**O que é um diagrama de arquitetura de software?**

Um diagrama de arquitetura de software é uma representação visual da estrutura e organização de um sistema de software. Ele oferece uma visão abstrata e de alto nível das diferentes componentes do sistema, suas interações e como elas se relacionam.

Esses diagramas fornecem uma representação gráfica das partes do sistema, como módulos, componentes, camadas, classes, interfaces e suas relações, ajudando a compreender a arquitetura geral do software. Eles são criados para auxiliar no projeto, comunicação, documentação e entendimento das decisões de design tomadas ao desenvolver um sistema de software.

Com um diagrama de arquitetura de software é possível identificar os principais componentes, visualizar suas interações e entender como as partes do sistema trabalham em conjunto para cumprir os requisitos do software.

### **Modelo Entidade Relacionamento**

**O Modelo Entidade Relacionamento (também chamado Modelo ER, ou simplesmente MER)**, como o nome sugere, é um modelo conceitual utilizado na [**Engenharia de Software**](https://www.devmedia.com.br/principios-da-engenharia-de-software/29630) para descrever os objetos (entidades) envolvidos em um domínio de negócios, com suas características (atributos) e como elas se relacionam entre si (relacionamentos).

Em geral, este modelo representa de forma abstrata a estrutura que possuirá o [**banco de dados**](https://www.devmedia.com.br/cursos/banco-de-dados) da aplicação. Obviamente, o banco de dados poderá conter várias outras entidades, tais como chaves e tabelas intermediárias, que podem só fazer sentido no contexto de [**bases de dados relacionais**](https://www.devmedia.com.br/curso/curso-modelagem-de-bancos-de-dados-relacionais/409).

### **Entidades**

Os objetos ou partes envolvidas um domínio, também chamados de entidades, podem ser classificados como físicos ou lógicos, de acordo sua existência no mundo real. Entidades físicas: são aquelas realmente tangíveis, existentes e visíveis no mundo real, como um cliente (uma pessoa, uma empresa) ou um produto (um carro, um computador, uma roupa). Já as entidades lógicas são aquelas que existem geralmente em decorrência da interação entre ou com entidades físicas, que fazem sentido dentro de um certo domínio de negócios, mas que no mundo externo/real não são objetos físicos (que ocupam lugar no espaço). São exemplos disso uma venda ou uma classificação de um objeto (modelo, espécie, função de um usuário do sistema).

### **Diagrama Entidade Relacionamento**

Enquanto o MER é um modelo conceitual, o Diagrama Entidade Relacionamento (Diagrama ER ou ainda DER) é a sua representação gráfica e principal ferramenta. Em situações práticas, o diagrama é tido muitas vezes como sinônimo de modelo, uma vez que sem uma forma de visualizar as informações, o modelo pode ficar abstrato demais para auxiliar no desenvolvimento do sistema. Dessa forma, quando se está modelando um domínio, o mais comum é já criar sua representação gráfica, seguindo algumas regras.

O diagrama facilita ainda a comunicação entre os integrantes da equipe, pois oferece uma linguagem comum utilizada tanto pelo analista, responsável por levantar os requisitos, e os desenvolvedores, responsáveis por implementar aquilo que foi modelado.

*Fonte: https://www.devmedia.com.br/mer-e-der-modelagem-de-bancos-de-dados/14332*

### **1. Entidades Principais**

jogador

#### **1.1 Personagem (Legend)**

Cada personagem (ou "Legend") tem atributos e características únicas. As entidades principais relacionadas aos personagens são:

* **Nome**: Nome do personagem (exemplo: "Bodvar", "Ragnar").
* **Armas**: Cada personagem usa um ou dois tipos de armas, com diferentes movimentos e habilidades. Exemplos de armas são: espadas, machados, lanças, arcos, entre outros.

#### **1.2 Mapa (Stage)**

Os mapas onde as batalhas acontecem, com diferentes formas e elementos que afetam a jogabilidade. Alguns atributos do mapa são:

* **Plataformas**: Locais elevados que o jogador pode usar para se mover.
* **Tamanho**: O tamanho do mapa pode influenciar a dinâmica da luta.
* **Obstáculos**: Elementos como paredes ou plataformas móveis que alteram o movimento dos jogadores.

#### **1.3 Armas**

* **Nome da Arma**: Cada tipo de arma tem seu próprio estilo de combate.
* **Ataques Básicos**: Ataques simples, realizados com um botão de ação.
* **Ataques Especiais**: Ataques mais poderosos, que geralmente têm efeitos mais devastadores ou únicos.

#### **1.4 Jogador (Player)**

Representa o indivíduo controlando o personagem:

* **Nickname**: Nome do jogador.
* **Ranking**: O nível ou classificação do jogador em partidas competitivas.
* **Pontos (Points)**: A quantidade de pontos ou vitórias acumuladas.

#### **1.5 Item**

Itens que aparecem no mapa durante a partida:

* **Nome**: O nome do item (exemplo: "Bomb", "Spear", "Swords").
* **Efeito**: O que o item faz, como causar dano ao ser jogado ou dar ao personagem um bônus temporário.

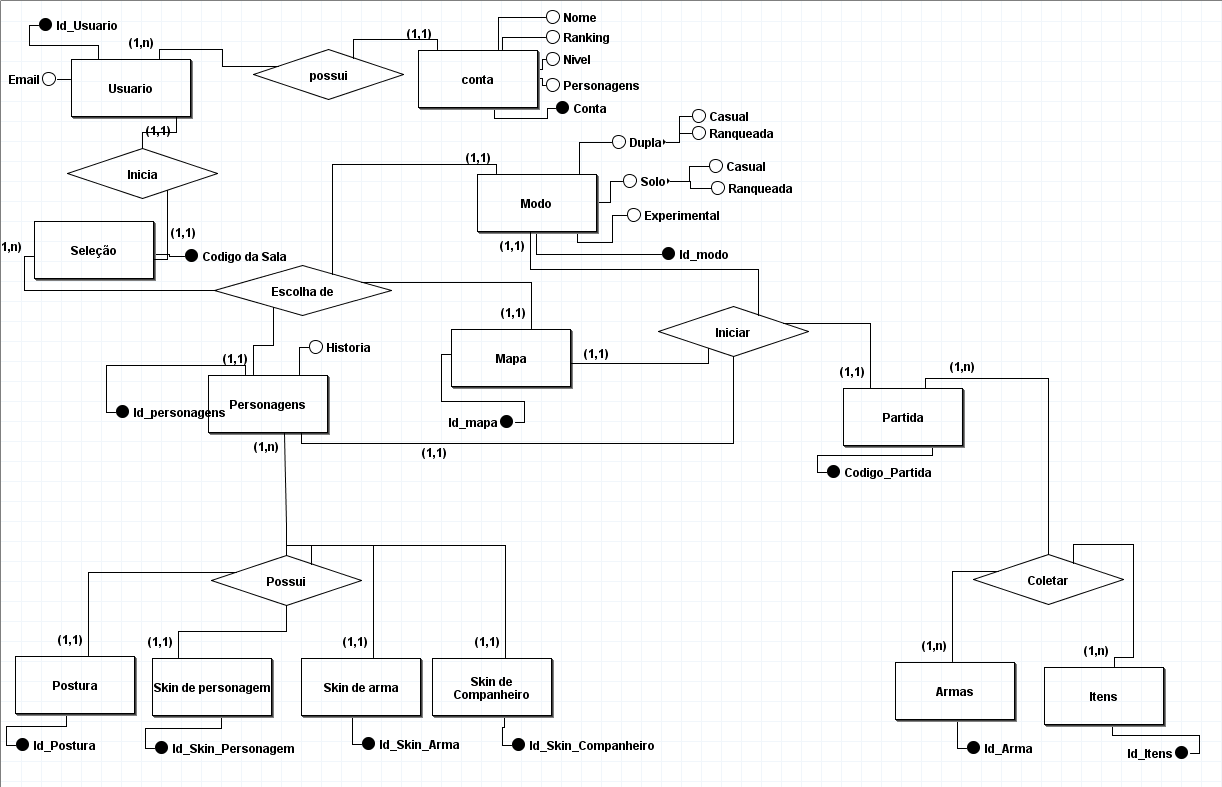
#### **1.6 Equipe**

Se for um jogo em equipe (2v2, por exemplo):

* **Jogadores**: Indivíduos que formam uma equipe.
* **Estratégia de Combate**: A tática usada pela equipe para derrotar os oponentes.

Essas são algumas das principais entidades e atributos que estruturam o jogo Brawlhalla. O jogo é projetado para permitir diversas estratégias e combinações, dependendo da escolha de personagem, armas e das táticas de jogo dos jogadores.

**Mer e Der:**



Dicionário de Dados:

**Usuário:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Id\_Usuário | Inteiro | Identifica o usuário no banco de dados. |
| Email | Email | Armazena o email do usuário |

**Conta:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Nome | VARCHAR(100) | Armazena o nome utilizado na conta. |
| Ranking | INT | Ranking do jogador |
| Nivel | INT | Nível da conta |
| Personagens | TEXT | LIsta de personagens do usuário |
|  |  |  |

**Modo:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Id\_Modo | INT(Chave Primária) | Identificador único do modo de jogo |
| Tipo | ENUM | Tipo de jogo (Solo, Dupla, Casual) |

**Mapa:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Id\_Mapa | INT(Chave Primária) | Identificador único do mapa |

**Personagens:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Id\_Personagens | INT(Chave Primária) | Identificador único do personagem |

**Postura:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Id\_Postura | INT(Chave Primária) | Identificador único da postura |

**Skin de Personagem:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Id\_Skin\_Personagem | INT(Chave Primária) | Identificador único da skin do personagem. |

**Skin de Arma:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Id\_Skin\_Arma | INT(Chave Primária) | Identificador único da skin de arma |

**Skin de Companheiro:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Id\_Skin\_Companheiro | INT(Chave Primária) | Identificador único da skin do companheiro |

**Partida:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Código\_Partida | INT(Chave Primária) | Código único da partida |

**Armas:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Id\_Arma | INT(Chave Primária) | Identificador único de cada arma |

**Itens:**

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo de Dado | Descrição |
| Id\_Itens | INT(Chave Primária) | Identificador único de cada item |