**Documento Base: Sistema de Classificação e Ranking de Jogos de Tabuleiro**

**1. Objetivos do Sistema**

1. Criar um ranking dinâmico e preciso, com pontuação geral entre 1 e 10 (com três casas decimais) para cada jogo.
2. Refletir as preferências do utilizador de forma transparente e personalizada.
3. Combinar dois modos de interação:
   * Um **modo básico simples**, baseado na dimensão "Vontade de Jogar".
   * Um **modo avançado** que permite comparar jogos com base em várias dimensões configuráveis.
4. Implementar **pesos ajustáveis para cada dimensão**, com valores padrão definidos, mantendo sempre um total de 100%.
5. Aplicar um **decaimento temporal diário** para priorizar as preferências mais recentes, sem descartar completamente comparações antigas.
6. Garantir que as pontuações finais dos jogos sejam **sempre entre 1 e 10** e representativas das avaliações feitas até o momento, utilizando uma normalização adaptativa.

**2. Estrutura do Sistema**

**2.1. Modos de Interação**

**Modo Básico**

* Baseado exclusivamente na dimensão **"Vontade de Jogar"**.
* Cada comparação apresenta dois jogos (Jogo A e Jogo B).
* O utilizador escolhe qual prefere jogar no momento.
* Pontuações resultantes:
  + O jogo escolhido recebe **1 ponto** em "Vontade de Jogar".
  + Em caso de empate, ambos os jogos recebem **0.5 ponto**.
* O **Modo Básico** é o ponto de partida obrigatório para todos os utilizadores.

**Modo Avançado**

* Permite ativar dimensões adicionais para avaliação.
* Comparações no Modo Avançado incluem perguntas específicas relacionadas às dimensões ativas.
* As pontuações acumuladas são divididas entre as dimensões selecionadas.
* O sistema mantém consistência ao adicionar dimensões, preservando as pontuações existentes de "Vontade de Jogar".

**2.2. Dimensões de Avaliação**

As dimensões refletem características específicas dos jogos e captam nuances distintas. São:

1. **Vontade de Jogar:** Desejo de jogar o jogo em qualquer momento.
2. **Design:** Qualidade estética e visual do jogo, incluindo componentes e ilustrações.
3. **Diversão:** Capacidade de gerar momentos de riso, emoção e engajamento.
4. **Rejogabilidade:** Potencial do jogo de proporcionar experiências únicas em múltiplas partidas.
5. **Satisfação Pós-Jogo:** Sentimento de realização ou diversão ao término de uma partida.
6. **Facilidade de Ensinar:** Simplicidade e clareza das regras para novos jogadores.
7. **Complexidade:** Nível de profundidade estratégica ou dificuldade percebida.
8. **Originalidade:** Grau de inovação nas mecânicas e no tema.
9. **Interatividade:** Nível de envolvimento entre jogadores durante a partida.
10. **Imersão no Tema:** Capacidade do jogo de transportar os jogadores para o seu universo temático.
11. **Equilíbrio:** Percepção de justiça e igualdade de oportunidades entre jogadores.
12. **Tempo de Jogo Ideal:** Adequação da duração do jogo às preferências do utilizador.
13. **Valor pelo Preço:** Relação entre o custo e a experiência oferecida pelo jogo.

**Configuração das Dimensões:**

* O utilizador escolhe quais dimensões ativar.
* "Vontade de Jogar" é obrigatória e está sempre ativa.
* Dimensões ativas entram imediatamente no cálculo do ranking, mesmo sem dados completos.

**2.3. Pesos das Dimensões**

1. Cada dimensão ativa recebe um peso que define sua contribuição no cálculo da pontuação final.
2. **Pesos Iniciais Padrão:**
   * Vontade de Jogar: 50%
   * Design: 10%
   * Diversão: 10%
   * Rejogabilidade: 10%
   * Satisfação Pós-Jogo: 5%
   * Facilidade de Ensinar: 5%
   * Complexidade: 2.5%
   * Originalidade: 2.5%
   * Interatividade: 2.5%
   * Imersão no Tema: 2.5%
   * Equilíbrio: 2.5%
   * Tempo de Jogo Ideal: 2.5%
   * Valor pelo Preço: 2.5%
3. **Regras para Alteração de Pesos:**
   * O utilizador pode ajustar os pesos conforme preferir.
   * O total dos pesos deve ser **exatamente 100%**.
   * Alterações nos pesos acionam o **reprocessamento imediato** das pontuações de todos os jogos.

**2.4. Balanceamento de Aparições**

1. **Garantia de Equidade:**
   * O sistema prioriza jogos com menor número de comparações realizadas.
   * Em caso de empate no número de aparições:
     + Prioriza jogos cuja última avaliação tenha ocorrido há mais tempo.
2. **Seleção para Comparações:**
   * A lógica de balanceamento aplica-se tanto ao Modo Básico quanto ao Avançado.
   * A prioridade dos jogos é reavaliada após cada comparação.

**2.5. Pontuação e Normalização**

**Pontuação Base:**

* Cada comparação resulta em:
  + **1 ponto** para o jogo escolhido na dimensão avaliada.
  + **0.5 ponto** para cada jogo em caso de empate.

**Normalização por Frequência:**

* A pontuação acumulada é dividida pelo número de vezes que o jogo apareceu:

Uma imagem com texto, Tipo de letra, file, captura de ecrã

Descrição gerada automaticamente

**Normalização Adaptativa (Intervalo [1, 10]):**

1. Identificar o intervalo dinâmico:
   * PminP\_{\text{min}}Pmin​: A pontuação mais baixa entre todos os jogos.
   * PmaxP\_{\text{max}}Pmax​: A pontuação mais alta entre todos os jogos.
2. Ajustar a pontuação para o intervalo [1, 10]:

Uma imagem com Tipo de letra, texto, captura de ecrã, design

Descrição gerada automaticamente

**2.6. Decaimento Temporal**

1. **Frequência de Aplicação:**
   * O decaimento é aplicado **uma vez por dia**.
2. **Fórmula:**
   * Cada comparação recebe um fator de relevância temporal (T):

Uma imagem com Tipo de letra, texto, Gráficos, captura de ecrã

Descrição gerada automaticamente

* λ=0.01\lambda = 0.01λ=0.01: Constante de decaimento.
* Δt\Delta tΔt: Tempo em dias desde a comparação.

**2.7. Pontuação Final**

A pontuação final é calculada como a média ponderada das dimensões ativas:

Uma imagem com Tipo de letra, texto, Gráficos, tipografia

Descrição gerada automaticamente

**2.8. Challenges**

1. Challenges focam em dimensões específicas, incentivando o utilizador a enriquecer o perfil dos jogos.
2. Cada challenge consiste em um conjunto de comparações relacionadas à dimensão alvo.
3. Os resultados atualizam automaticamente a pontuação dos jogos.

**3. Transparência e Visualização**

Cada jogo terá um ecrã dedicado com os seguintes detalhes:

1. **Pontuação Geral:** Exibição do ranking geral e da pontuação final.
2. **Pontuação por Dimensão:** Tabela detalhada com pontuações, pesos e contribuições de cada dimensão.
3. **Histórico de Comparações:** Lista completa de comparações realizadas, incluindo adversários, dimensões e resultados.
4. **Impacto Temporal:** Gráfico de evolução da pontuação ao longo do tempo.
5. **Pesos Configurados:** Exibição dos pesos atribuídos a cada dimensão.

**Estrutura da Base de Dados Atualizada**

**1. Tabela: Users (Utilizadores)**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único do utilizador. |
| name | VARCHAR(255) | Nome do utilizador. |
| email | VARCHAR(255) | Endereço de e-mail (único). |
| password\_hash | VARCHAR(255) | Hash da senha. |
| created\_at | TIMESTAMP | Data de criação do registo. |
| updated\_at | TIMESTAMP | Última atualização do registo. |

**2. Tabela: Games (Jogos)**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único do jogo. |
| name | VARCHAR(255) | Nome do jogo. |
| bgg\_id | INT | ID do jogo na base de dados do BGG. |
| image\_url | TEXT | URL da imagem principal do jogo. |
| year\_published | INT | Ano de publicação do jogo. |
| min\_players | INT | Número mínimo de jogadores. |
| max\_players | INT | Número máximo de jogadores. |
| playtime | VARCHAR(50) | Duração estimada do jogo. |
| complexity | DECIMAL(3, 2) | Complexidade do jogo (peso no BGG). |
| created\_at | TIMESTAMP | Data de criação do registo. |
| updated\_at | TIMESTAMP | Última atualização do registo. |

**3. Tabela: GameCategories (Categorias de Jogos)**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único da categoria. |
| name | VARCHAR(255) | Nome da categoria (ex.: Estratégia, Família). |
| created\_at | TIMESTAMP | Data de criação do registo. |
| updated\_at | TIMESTAMP | Última atualização do registo. |

**Tabela Relacional: GameCategoryAssignments**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único do registo. |
| game\_id | UUID | Relacionamento com a tabela Games. |
| category\_id | UUID | Relacionamento com a tabela GameCategories. |
| created\_at | TIMESTAMP | Data de criação do registo. |

**4. Tabela: GameMechanics (Mecânicas de Jogos)**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único da mecânica. |
| name | VARCHAR(255) | Nome da mecânica (ex.: Deck Building, Worker Placement). |
| created\_at | TIMESTAMP | Data de criação do registo. |
| updated\_at | TIMESTAMP | Última atualização do registo. |

**Tabela Relacional: GameMechanicAssignments**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único do registo. |
| game\_id | UUID | Relacionamento com a tabela Games. |
| mechanic\_id | UUID | Relacionamento com a tabela GameMechanics. |
| created\_at | TIMESTAMP | Data de criação do registo. |

**5. Tabela: GameAuthors (Autores do Jogo)**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único do autor. |
| name | VARCHAR(255) | Nome do autor (ex.: Reiner Knizia). |
| created\_at | TIMESTAMP | Data de criação do registo. |
| updated\_at | TIMESTAMP | Última atualização do registo. |

**Tabela Relacional: GameAuthorAssignments**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único do registo. |
| game\_id | UUID | Relacionamento com a tabela Games. |
| author\_id | UUID | Relacionamento com a tabela GameAuthors. |
| role | VARCHAR(50) | Papel do autor (ex.: Designer, Artista). |
| created\_at | TIMESTAMP | Data de criação do registo. |

**6. Tabela: UserGames (Jogos do Utilizador)**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único do registo. |
| user\_id | UUID | Relacionamento com a tabela Users. |
| game\_id | UUID | Relacionamento com a tabela Games. |
| initial\_score | DECIMAL(10, 3) | Pontuação inicial atribuída (padrão: 5.5). |
| created\_at | TIMESTAMP | Data de criação do registo. |
| updated\_at | TIMESTAMP | Última atualização do registo. |

**7. Tabela: Dimensions (Dimensões de Avaliação)**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único da dimensão. |
| name | VARCHAR(255) | Nome da dimensão (ex.: Vontade de Jogar). |
| default\_weight | DECIMAL(5, 2) | Peso padrão da dimensão. |
| is\_mandatory | BOOLEAN | Indica se é obrigatória (ex.: Vontade de Jogar). |
| created\_at | TIMESTAMP | Data de criação do registo. |
| updated\_at | TIMESTAMP | Última atualização do registo. |

**8. Tabela: AdjustedScores (Pontuações por Jogo e Dimensão, com Histórico)**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único do registo. |
| user\_id | UUID | Relacionamento com a tabela Users. |
| game\_id | UUID | Relacionamento com a tabela Games. |
| dimension\_id | UUID | Relacionamento com a tabela Dimensions. |
| score | DECIMAL(10, 3) | Pontuação calculada na dimensão. |
| frequency | INT | Número de vezes avaliado nesta dimensão. |
| timestamp | TIMESTAMP | Data e hora da atualização. |

**9. Tabela: AdjustedRankings (Ranking Normalizado, com Histórico)**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único do registo. |
| user\_id | UUID | Relacionamento com a tabela Users. |
| game\_id | UUID | Relacionamento com a tabela Games. |
| normalized\_score | DECIMAL(10, 3) | Pontuação final normalizada (1 a 10). |
| rank\_position | INT | Posição no ranking do utilizador. |
| timestamp | TIMESTAMP | Data e hora da atualização. |

**10. Tabela: Comparisons (Comparações Realizadas)**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id | UUID | Identificador único da comparação. |
| user\_id | UUID | Relacionamento com a tabela Users. |
| game\_a\_id | UUID | Jogo A comparado (Games.id). |
| game\_b\_id | UUID | Jogo B comparado (Games.id). |
| chosen\_game\_id | UUID | Jogo escolhido na comparação. |
| dimension\_id | UUID | Dimensão avaliada (Dimensions.id). |
| comparison\_date | TIMESTAMP | Data da comparação. |
| created\_at | TIMESTAMP | Data de criação do registo. |
|  |  |  |

**Stack Tecnológico Mais Apropriado**

Aqui está o stack otimizado para custos zero (ou muito baixos) e aproveitamento do NAS:

**1. Frontend**

**Tecnologias:**

* **React.js**: Biblioteca leve e eficiente para criar uma interface responsiva e moderna.
* **Tailwind CSS**: Para estilização rápida e altamente customizável.
* **Workbox**: Para transformar a web app numa PWA (Progressive Web App), garantindo instalação no ecrã inicial do telemóvel e funcionalidades offline.

**Porquê?**

* React é amplamente suportado e tem excelente documentação.
* Tailwind CSS reduz significativamente o esforço de estilização.
* Workbox facilita a configuração de PWA com caching offline.

**Instalação no NAS Synology:**

* O React gera um **build estático** (HTML, CSS, JS) que pode ser servido diretamente pelo **Web Station** do Synology, com **nginx** ou **Apache**.

**2. Backend**

**Tecnologias:**

* **Node.js** com **Express.js**:
  + Leve, rápido e fácil de configurar no Synology.
  + Pode ser executado no ambiente Docker disponível no NAS.
* **Prisma ORM**:
  + Simplifica a interação com a base de dados relacional.
  + Ideal para modelos de dados complexos, como rankings, histórico e sincronização com o BGG.

**Porquê?**

* Node.js é eficiente e bem documentado, com ótimo suporte para integrar APIs.
* O Prisma facilita a gestão de modelos e consultas SQL complexas, reduzindo o tempo de desenvolvimento.

**Instalação no NAS Synology:**

* Usar **Docker** para executar a aplicação Node.js com um container leve.
* Configuração simples e suporte para reinícios automáticos.

**3. Base de Dados**

**Tecnologias:**

* **PostgreSQL**:
  + Base de dados relacional robusta e gratuita.
  + Perfeito para lidar com relacionamentos entre tabelas e consultas avançadas.

**Porquê?**

* PostgreSQL é altamente confiável e ideal para armazenar os dados estruturados do sistema (jogos, rankings, histórico, comparações).
* Integra-se perfeitamente com o Prisma ORM.

**Instalação no NAS Synology:**

* O Synology suporta **PostgreSQL** diretamente ou através de um container Docker.
* Configuração:
  + Garantir backups regulares com as ferramentas nativas do NAS.

**4. Integração com o BoardGameGeek**

**Tecnologias:**

* API XML do BGG:
  + Usar o endpoint XML da API do BGG para importar dados.
  + Bibliotecas como **xml2js** (Node.js) facilitam a conversão do XML para JSON.

**Porquê?**

* A API XML é gratuita e cobre todas as necessidades (coleções, categorias, mecânicas, autores).

**Processo de Sincronização:**

* Criar um cron job com **node-cron** para sincronizar periodicamente os dados com o BGG.

**5. Hospedagem e Infraestrutura**

**Frontend (React.js)**

* **Web Station** do Synology (com nginx):
  + Servir os ficheiros estáticos gerados pelo React build.
  + Certificar-se de que o domínio local (ou público) está configurado corretamente.

**Backend (Node.js + Prisma)**

* Executar o backend como um container Docker no NAS Synology:
  + Fácil de configurar e gerenciar.
  + Pode ser acedido via URL interna ou externa (com redirecionamento de portas).

**Base de Dados**

* PostgreSQL instalado diretamente no NAS ou como container Docker.

**Certificados SSL**

* Usar **Let's Encrypt**, que é integrado no Synology, para configurar HTTPS sem custos adicionais.

**6. Notificações e Offline**

**Notificações Push:**

* **Web Push API**:
  + API nativa do navegador para enviar notificações push, sem necessidade de plataformas externas.
  + Pode ser configurada no backend com bibliotecas como **web-push** (Node.js).

**Offline:**

* IndexedDB ou LocalForage para armazenar dados do utilizador localmente.
* PWA configurada com caching dinâmico para garantir acessibilidade offline.

**7. Ferramentas e Fluxos de Desenvolvimento**

**Desenvolvimento Local**

* **Frontend:**
  + Desenvolver e testar o React localmente antes de fazer deploy no NAS.
* **Backend:**
  + Usar **Docker Compose** localmente para replicar a configuração do NAS.

**Fluxo de Deploy**

* Utilizar **rsync** ou outro método simples para transferir os ficheiros do frontend/backend para o NAS.
* Configurar **CI/CD local** com ferramentas como Git hooks para facilitar atualizações.

**Vantagens de Usar o NAS Synology**

1. **Custos Zero:**
   * Todo o hosting é feito no hardware que já possuis, sem custos adicionais.
2. **Escalabilidade para Projetos Pequenos/Médios:**
   * O Synology suporta bem cargas moderadas de utilização para aplicações pessoais e pequenos grupos.
3. **Infraestrutura Flexível:**
   * Com suporte a Docker, nginx e PostgreSQL, o NAS pode funcionar como uma solução completa.

**Resumo do Stack para Custos Zero**

**Frontend**

* **React.js** com **Tailwind CSS**.
* PWA configurada com **Workbox**.

**Backend**

* **Node.js** com **Express.js**.
* **Prisma ORM** para interagir com PostgreSQL.

**Base de Dados**

* **PostgreSQL** instalado diretamente no NAS ou via Docker.

**Infraestrutura**

* **NAS Synology**:
  + Servidor para frontend, backend e base de dados.
  + Certificados HTTPS com **Let's Encrypt**.

**Extras**

* Web Push API para notificações.
* IndexedDB para suporte offline.