Um Backend Analítico para Acompanhamento Terapêutico de Crianças com Distúrbios do Desenvolvimento Neurológico

Title: An Analytical Backend for Therapeutic Monitoring of Children with Neurodevelopmental Disorders

João Pedro de Castro Ribeiro[1], **Rosilane Ribeiro Mota**[1]

[1]Belo Horizonte – MG – Brasil
Pontifícia Universidade Catolica de Minas Gerais (PUC-MG)

joaopedrocastro05@gmail.com, rosilane@pucminas.br

Abstract. *This paper develops an analytical backend for the therapeutic monitoring of children with Neurodevelopmental Disorders, including autism. The system integrates with the "Jogando Juntos" app to track children's progress through gamified activities, providing detailed feedback to caregivers. It uses Natural Language Processing techniques to analyze interactions, identifying typing and semantic understanding errors.*
Keywords *Neurodevelopmental Disorders, Therapeutic Monitoring, Natural Language Processing, Child Development, Educational Technology.*

Resumo. *Este trabalho desenvolve um backend analítico para monitoramento terapêutico de crianças com Distúrbios do Desenvolvimento Neurológico, incluindo autismo. O sistema integra-se ao aplicativo "Jogando Juntos" para rastrear o progresso das crianças através de atividades gamificadas, fornecendo feedback detalhado aos cuidadores. Utiliza técnicas de Processamento de Linguagem Natural para analisar interações e identificar erros de digitação e compreensão semântica.*
Palavras-Chave *Distúrbios do Desenvolvimento Neurológico, Monitoramento Terapêutico, Processamento de Linguagem Natural, Desenvolvimento Infantil, Tecnologia Educacional.*

1. Introdução

No cenário contemporâneo, a integração da tecnologia no dia a dia tem transformado inúmeras áreas da vida humana, especialmente a educação e a saúde. As crianças com Distúrbios do Desenvolvimento Neurológico (DDN), incluindo aquelas com Transtorno do Espectro Autista (TEA), enfrentam desafios particulares que exigem soluções especializadas para auxiliar em sua aprendizagem e comunicação.

Diante desse cenário, os sistemas de *backend*, particularmente relevantes nas áreas da saúde e educação, desempenham um papel importante em garantir a integridade, segurança e confiabilidade na transferência e manuseio de dados sensíveis. Esses sistemas são concebidos para funcionar em nível sistêmico, tratando de questões como o acesso restrito, o armazenamento seguro de dados e a conformidade com as normas. A confiabilidade desses *backends* é essencial para assegurar que as análises e informações geradas sejam precisas e confiáveis, oferecendo suporte aos usuários finais.

Diante disso, este trabalho visa abordar a carência de ferramentas analíticas especializadas que acompanhem o desenvolvimento de crianças com DDN por meio de atividades gamificadas. Atualmente, muitos sistemas carecem de uma capacidade analítica que foque nas nuances de aprendizagem e comunicação dessas crianças, e que forneça um *feedback* detalhado capaz de orientar os pais e especialistas de maneira eficaz.

Portanto, o objetivo deste trabalho é desenvolver um *backend* com alta disponibilidade para integrar com o aplicativo "Jogando Juntos" [Kaweski 2024], analisando as interações das crianças com o jogo e fornecendo *feedback* textual aos pais e especialistas sobre a pontuação obtida, número de acertos e erros mais cometidos. O sistema desenvolvido e apresentado neste artigo emprega análise fonética e técnicas de *Natural Language Processing* (*NLP*), identificando erros de digitação e compreensão semântica, facilitando assim o monitoramento e a orientação do desenvolvimento da criança.

Este trabalho é organizado em seções distintas. A Seção 2 explora pesquisas e trabalhos que fundamentam este artigo. Na Seção 3, os algoritmos, técnicas e tecnologias são detalhados. Depois, a Seção 4 apresenta gráficos com os resultados juntamente com alguns comparativos do desempenho das crianças. Finalmente, na Seção 5 é feita uma análise sobre os objetivos alcançados e algumas considerações para trabalhos futuros.

2. Trabalhos Relacionados

Existem diversos trabalhos que abordam o desenvolvimento de sistemas para o acompanhamento terapêutico de crianças com distúrbios do neurodesenvolvimento. Dentre eles, destaca-se o trabalho de [Cabral 2023], que propõe um sistema para monitoramento terapêutico gamificado, coletando e analisando dados em tempo real durante a interação das crianças com os jogos. A arquitetura do sistema desenvolvida contempla muitas tecnologias que também foram utilizadas no presente trabalho, como Java e Spring Boot. Seu sistema está hospedado no Heroku e Netlify, garantindo um ambiente confiável e de baixo custo. Esse trabalho tem como objetivo principal fornecer *feedbacks* constantes para profissionais de saúde e responsáveis, possibilitando um tratamento personalizado para cada criança. Entretanto, essa solução apresenta uma análise muito básica para os objetivos propostos neste trabalho, sem a possibilidade de inclusão de técnicas de NLP, como feito no estudo de [Silveira et al. 2019].

Em complemento, o aplicativo "Jogando Juntos" [Kaweski 2024] é um jogo similar ao Gartic [Moreira 2008], em que os jogadores se revezam desenhando e adivinhando palavras com base em categorias pré-definidas. O jogador que está desenhando deve tentar transmitir a palavra escolhida por meio de desenhos, enquanto os outros jogadores tentam adivinhar a palavra com base neles. O jogo estimula a criatividade, a comunicação e a interação social, sendo uma ferramenta valiosa para o aumento do vínculo familiar da criança com DDN, visto que se trata de um jogo colaborativo em rede para que a família jogue junto. Esse trabalho foi integrado ao sistema de *backend* apresentado neste artigo, que se comunica diretamente com o aplicativo "Jogando Juntos", coletando e fazendo análise dos dados gerados durante as sessões de jogo para fornecer *feedback* aos pais e especialistas.

Para que as análises fossem realizadas no sistema desenvolvido, foi utilizada NLP como uma das técnicas para examinar os erros dos jogadores, fornecendo análises

especializadas para cada jogador. Outros dois estudos também utilizaram a NLP na análise de seus dados. Um deles, realizado por Eric Luz (2020), usa NLP para analisar redações de estudantes e identificar dificuldades na escrita. Enquanto o segundo, [Silveira et al. 2019], aplica a técnica para avaliar o discurso de crianças com distúrbios da fala e linguagem.

Outro aspecto importante é a alta disponibilidade, crucial para sistemas *backend* na área da saúde. Em um trabalho realizado no ano de 2012 [Mojžišová e Mojžiš 2012], foi proposta uma plataforma unificada para a entrega de alertas e notificações em *smartphones*, baseada em um serviço *web RESTful* em Java, que oferece uma interface simples e acesso seguro. Essa abordagem visava fornecer uma solução flexível e escalável para o envio de alertas e notificações em tempo real, independentemente do sistema operacional utilizado.

3. Metodologia

Neste projeto, foram utilizados métodos quantitativos e qualitativos para desenvolver um sistema analítico destinado ao ensino de crianças com DDN. Além disso, foram gerados resultados textuais disponibilizados por este trabalho que indicam o histórico de interação da família (criança e cuidadores) com o aplicativo "Jogando Juntos" [Kaweski 2024].

Para gerar esses resultados, foram utilizadas técnicas, sendo elas a distância de Levenshtein[1], a Metaphone[2] e a NLP, que fazem uma comparação da similaridade da tentativa de resposta feita pelo jogador com a resposta correta, tornando possível identificar o tipo de erro cometido pelo jogador. A partir dessas identificações, são geradas métricas e relatórios do resultado e progresso dos jogadores. Essas métricas geradas incluem os tipos de erros mais cometidos, frequência de erros, pontuações feitas e progresso do jogador ao longo do tempo.

O sistema desenvolvido neste trabalho precisa estar totalmente integrado ao jogo "Jogando Juntos" [Kaweski 2024], pois é preciso incluir no Sistema de Gerenciamento de Banco de Dados (SGBD) do *backend* desenvolvido, informações sobre a interação do jogador com o aplicativo, tais como as partidas jogadas, respostas corretas e tentativas dos jogadores. Dessa forma, é necessário garantir um sistema com alta disponibilidade para viabilizar a comunicação entre o aplicativo "Jogando Juntos" e este sistema, armazenando e coletando os dados necessários para uma análise posterior pela *Application Programming Interface* (API) desenvolvida.

Foram identificados sete requisitos funcionais. Dentre eles, tem-se:

- **RF01:** Coletar dados das partidas, rodadas e tentativas.
- **RF02:** Armazenar dados do jogador.
- **RF03:** Analisar os dados coletados.
- **RF04:** Detectar o tipo de similaridade entre a resposta do jogador e a palavra correta.

[1]A distância de Levenshtein é uma métrica de diferença entre duas sequências de caracteres. Ela mede o número mínimo de operações necessárias para transformar uma sequência na outra, em que as operações podem ser inserções, remoções ou substituições de um único caractere [Levenshtein et al. 1966].

[2]Metaphone é um algoritmo fonético que indexa palavras por sua pronúncia. Ele foi desenvolvido para melhorar a pesquisa de palavras baseadas em sua similaridade fonética, ajudando a identificar palavras que soam de forma similar, mesmo que sejam ortograficamente diferentes [Philips 1990].

- **RF05:** Calcular a pontuação do jogador.
- **RF06:** Gerar relatórios e *dashboards* sobre o desempenho do jogador.
- **RF07:** Integrar com o aplicativo "Jogando Juntos".

Além desses, foram identificados como requisitos não-funcionais, alta disponibilidade, garantindo que o sistema esteja sempre disponível para receber informações das partidas concluídas do jogo "Jogando Juntos". O sistema também é projetado para ter um baixo tempo de resposta, utilizando *cache* em integrações feitas a serviços externos. Além disso, o sistema deve ser extensível, permitindo a adição de novas formas de calcular a similaridade das palavras facilmente. O sistema deve possuir persistência dos dados, assegurando que, uma vez salva as informações, elas não serão perdidas. Por último, o sistema deve possuir Integração Contínua e Entrega Contínua (CI/CD) para garantir um processo de desenvolvimento contínuo e automatizado, facilitando a integração e entrega de novas funcionalidades.

Para que o sistema desenvolvido atendesse a todos esses requisitos, foram escolhidas as tecnologias apresentadas nas subseções que se seguem.

3.1. Modelagem de Dados

O modelo de dados utilizado pelo sistema desenvolvido precisa suportar os dados dos jogadores, das partidas jogadas, das tentativas e desenhos feitos.

O diagrama de entidade-relacionamento demonstra as entidades do sistema, assim como suas relações (Figura 1). A tabela *PLAYER_ROUND* é utilizada para resgatar informações das tentativas feitas pelo jogador (coluna *GUESS_ATTEMPT*), comparando essas tentativas com o desenho que foi feito.

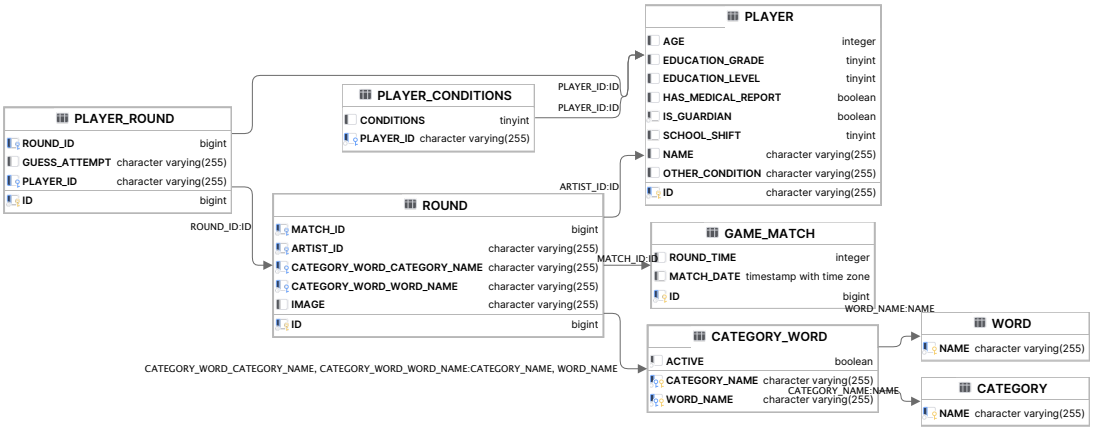

Figura 1. Diagrama de Entidade-Relacionamento do Sistema

Esta modelagem indica a relação entre as informações das partidas jogadas no aplicativo "Jogando juntos" [Kaweski 2024] e os demais dados do jogador, facilitando uma análise posterior de um jogador específico.

3.2. Coleta de Dados

A coleta de dados no sistema "Jogando Juntos" [Kaweski 2024] é automática e ocorre em tempo real, sendo consolidada e enviada ao final de cada partida jogada para uma *API RESTful*. Cada sessão de jogo gera um conjunto de dados, que inclui identificadores do jogador, a resposta correta, a categoria da palavra, as respostas dadas pelos jogadores, o tempo de cada rodada e a imagem desenhada, como descrito na modelagem da Figura 1.

Este processo de coleta de dados é necessário para que seja feita uma análise posterior dos erros cometidos, assim como a geração dos relatórios de acompanhamento que podem ser acessados por pais e especialistas.

3.3. Cálculo de Similaridade

O sistema desenvolvido é projetado para registrar e categorizar erros e acertos nas tentativas feitas pelos jogadores. Essas classificações incluem:

- **Igualdade Exata:** determina se as palavras são idênticas em todos os aspectos, incluindo letras e acentuação;
- **Equivalência Sem Acentos:** determina se as palavras são equivalentes ao ignorar os acentos, como "época" e "epoca";
- **Ortograficamente Similares:** determina se as palavras são semelhantes em termos de estrutura de caracteres, como "casa" e "casas";
- **Foneticamente parecidas:** determina se as palavras possuem a mesma pronúncia, como "sessão" e "seção";
- **Semanticamente Relacionadas:** determina se as palavras são sinônimos ou têm significados muito próximos, como "canino" e "cachorro";
- **Mesma Categoria:** verifica se as palavras pertencem à mesma categoria semântica, como "cachorro" e "gato", ou seja, ambos são animais;
- **Conexão Contextual:** analisa se existe uma ligação funcional ou contextual entre as palavras, por exemplo, "caneta" e "papel";
- **Outras Similaridades:** identifica outras formas de similaridade que não se encaixam nas categorias anteriores;
- **Diferentes:** determina se as palavras são completamente distintas, sem nenhuma conexão aparente.

Foi utilizado o padrão de *design Chain of Responsibility*[3] para implementar o cálculo de similaridade entre as respostas dos jogadores e as palavras-alvo. Esse padrão organiza as técnicas de similaridade em uma sequência lógica, da mais simples à mais complexa, permitindo um desacoplamento de cada técnica utilizada em um *handler* diferente. A sequência dos *handlers* utilizados pode ser visualizada na Figura 2.

[3]Chain of Responsibility é um padrão de *design* comportamental que permite que um pedido passe por uma cadeia de *handlers* até que um deles trate o pedido. Ele promove o desacoplamento entre o remetente do pedido e seus receptores ao permitir que múltiplos objetos tenham a oportunidade de processar o pedido [Gamma et al. 1995].

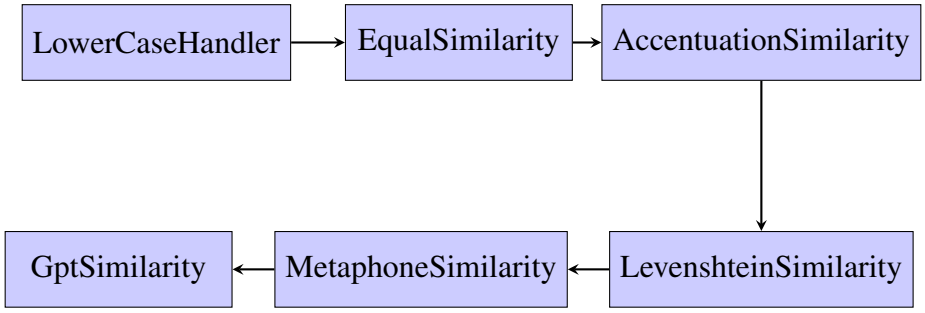

Figura 2. Cadeia de *handlers*

Cada *handler* tem uma responsabilidade única e só passa o pedido ao próximo *handler*, se ele não conseguir retornar o tipo de similaridade, sendo possível observar a responsabilidade de cada *handler* a seguir:

- **LowerCaseHandler:** embora não diretamente relacionado com uma categoria específica de similaridade, este *handler* funciona como um preprocessador, convertendo todas as entradas em letras minúsculas para padronizar as comparações nas etapas subsequentes;
- **EqualSimilarity:** este *handler* verifica a igualdade exata entre duas palavras. Se as palavras são idênticas em todos os aspectos, incluindo letras e acentuação, a análise para essa tentativa é concluída imediatamente, resultando em uma categorização de "Igualdade Exata";
- **AccentuationSimilarity:** este *handler* compara duas palavras ignorando acentos ortográficos, categorizando a similaridade como "Equivalência Sem Acentos" se forem idênticas nesse aspecto;
- **LevenshteinSimilarity:** utiliza a distância de Levenshtein para avaliar pequenas variações ortográficas entre palavras, indicando possíveis erros comuns de digitação ou variações singulares/plurais, e as categoriza como "Ortograficamente Similares";
- **MetaphoneSimilarity:** este *handler* analisa a similaridade fonética entre palavras, usando o algoritmo metaphone para detectar se elas soam de maneira similar, mesmo que escritas de formas diferentes, classificando-as como "Foneticamente Parecidas";
- **GptSimilarity:** este é o *handler* mais avançado, sendo utilizado apenas quando as outras análises falham em classificar a similaridade. Ele emprega técnicas de processamento de linguagem natural, utilizando a API da OpenAI para avaliar similaridades complexas, incluindo aquelas semânticas e contextuais.

Esse padrão utilizado garante uma economia de recursos, pois o último *handler*, que faz chamadas externas à API da OpenAI, só é utilizado quando é realmente necessário. Essas chamadas possuem um custo de acordo com o número de *tokens* utilizados, além de possuir uma latência maior devido ao tráfego de rede.

Por abordar mais de um tipo de similaridade e fazer uso de serviços externos, o **GptSimilarity** será melhor detalhado na seção seguinte.

3.3.1. Similaridade Semântica com GPT

No sistema desenvolvido neste artigo, optou-se pelo uso do modelo GPT-4o, escolhido por seu baixo custo e velocidade. Este modelo é usado com um *prompt* para avaliar a similaridade semântica entre a palavra-alvo e as tentativas de adivinhação dos usuários, baseando-se em categorias definidas com um critério de prioridade, definidas no *prompt* customizado.

O *prompt* orienta a classificação das respostas nas categorias apresentadas na Seção 3.3, ordenadas por prioridade em caso de empate.

Para a criação do *prompt*, foram empregadas técnicas de engenharia de *prompt* [OpenAI 2023] [Prompting Guide 2023], tais como pedir ao modelo para adotar uma persona. Estas técnicas foram fundamentais para garantir que o modelo de linguagem fornecesse *feedbacks* mais exatos.

A capacidade de avaliar se duas palavras são parecidas usando técnicas de NLP é fundamental para realizar uma análise mais completa sobre o tipo de erro cometido pelo usuário, permitindo, assim, uma abordagem mais instrutiva e adaptada às necessidades educacionais específicas.

3.4. Cálculo de Pontuação

No sistema desenvolvido, foi feito um cálculo de pontuação a partir dos dados obtidos do jogo, que podem ser visualizados posteriormente fora deste. Para o cálculo, foi utilizado o padrão de *design Strategy*[4], o que permite a flexibilização e ajuste do método de cálculo sem alterar o sistema principal. Esta abordagem assegura que diferentes lógicas de pontuação possam ser aplicadas conforme necessário, facilitando a adaptação e a experimentação com diferentes estratégias educacionais.

Utilizamos a estratégia `NormalizedSimilarityScoreStrategy` para calcular o *score* baseado na similaridade das respostas dos jogadores com a palavra-alvo. O *score* é normalizado para garantir que o resultado final esteja sempre entre 0 e 100, independentemente do número de tentativas ou da complexidade da resposta, proporcionando uma métrica de desempenho consistente e justa. A pontuação é calculada da seguinte forma:

- **Acerto Exato:** a resposta exata à palavra-alvo recebe a pontuação máxima de 2.0 pontos;
- **Erros Ortográficos e de Acentuação:** respostas que diferem apenas por pequenos erros ortográficos ou de acentuação recebem 1.0 ponto;
- **Similaridades Menos Diretas:** respostas que estão corretas em termos fonéticos, semânticos, categoriais ou contextuais são recompensadas com 0.5 ponto;
- **Resposta Completamente Diferente:** uma resposta que não se relaciona de maneira significativa com a palavra-alvo resulta em uma subtração de 1.0 ponto do *score* total.

[4]Strategy é um padrão de *design* comportamental que define uma família de algoritmos, encapsula cada um deles e os torna intercambiáveis. Este padrão permite que o algoritmo varie independentemente dos clientes que o utilizam [Gamma et al. 1995].

A pontuação total para um jogador é calculada somando-se os pontos de cada tentativa e normalizando este total para criar uma pontuação percentual. Esse método não apenas incentiva respostas exatas, mas também reconhece e valoriza tentativas que, embora não perfeitamente corretas, demonstram compreensão e raciocínio lógico relevante.

Essa metodologia de pontuação é utilizada para monitorar o progresso dos jogadores e fornecer *feedback* educativo. Em um dos *endpoints*, por exemplo, o *score* é calculado e apresentado com base nas datas das partidas, permitindo aos jogadores, pais e especialistas visualizarem a progressão das habilidades ao longo do tempo.

3.5. Tecnologias Utilizadas

A seleção das tecnologias para o desenvolvimento do *backend* foi orientada pela busca por eficiência, suporte da comunidade e adequação às demandas de um sistema interativo e escalável. Para isso, a linguagem Java 21 foi escolhida devido ao seu suporte à *virtual threads* [Kumar 2021], um recurso introduzido pelo Projeto Loom. Essa tecnologia facilita o gerenciamento de operações de entrada e saída (I/O) que bloqueiam a execução, melhorando a escalabilidade do sistema por meio de um uso mais eficiente dos recursos computacionais. As *virtual threads* são particularmente vantajosas para aplicações com uso intensivo de I/O, como a API desenvolvida neste projeto, que realiza diversas operações de banco de dados e chamadas para APIs externas, como a da OpenAI.

Além disso, o *framework* Spring Boot foi utilizado para simplificar a configuração e o desenvolvimento da aplicação, aproveitando recursos como a injeção de dependências e a integração facilitada com bancos de dados e outras APIs. A escolha do Spring Boot se justifica por sua popularidade e eficiência no desenvolvimento de aplicações Java corporativas [Mythily et al. 2022]. Adicionalmente, a biblioteca Caffeine [Manes 2024] foi empregada para otimizar o acesso aos dados frequentemente utilizados, implementando um mecanismo de *cache* que reduz a carga sobre os serviços externos e melhora o desempenho da aplicação. Outra solução de *cache* poderia ser o Redis por exemplo, mas pelo fato do Caffeine ser em memória, sua configuração foi mais simples.

Para o gerenciamento do banco de dados relacional, optou-se pelo H2 Database [Mueller 2024], escolhido por sua leveza e facilidade de integração, o que agiliza o processo de desenvolvimento e testes, além de facilitar a portabilidade da aplicação para outros ambientes. Embora outros bancos relacionais também atendessem igualmente, tais como MySql, MariaDb, Cassandra, o H2 foi escolhido por sua facilidade de uso.

A fim de garantir a qualidade e a eficácia do processo de desenvolvimento, foram adotadas práticas de CI/CD utilizando o GitHub Actions. Essa ferramenta automatiza os testes, a construção e a implantação da aplicação, garantindo que as atualizações sejam processadas de modo automático e eficiente, agilizando o processo de desenvolvimento e entrega do *software*.

Por fim, a hospedagem do sistema foi realizada na plataforma *Amazon LightSail*. A escolha dessa plataforma foi motivada por sua alta disponibilidade, com diversas opções de configurações para garantir a disponibilidade da aplicação, além de seu custo-benefício, com planos mensais de baixo custo que se adequam às necessidades do projeto.

4. Resultados Obtidos e Validação

Como resultados obtidos, foram feitas três validações diferentes para assegurar que o projeto cumpriu seu objetivo.

4.1. Integração com o Jogando Juntos

A integração entre o *backend* desenvolvido e o aplicativo "Jogando Juntos" [Kaweski 2024] foi realizada estabelecendo uma comunicação que garante a transmissão dos dados sem erros. A cada partida finalizada no aplicativo, um conjunto de dados abrangente era enviado para o *backend* por meio de uma *API RESTful* hospedada na AWS. Esses dados incluíam informações importantes, como os dados do jogador, a palavra correta, a categoria da palavra, as tentativas de resposta, o tempo de cada rodada e a imagem desenhada.

Essa integração permitia que o *backend* coletasse e armazenasse dados relevantes para o acompanhamento terapêutico sobre as interações dos jogadores com o aplicativo, fornecendo uma base para a análise e geração de relatórios personalizados. A utilização da *API RESTful* com uso de *Virtual Threads* garantiu a escalabilidade na solução aqui desenvolvida.

4.2. Visualização de Dados dos Relatórios Gerados

Os dados coletados pelo *backend* foram processados e transformados em relatórios textuais informativos, projetados para auxiliar pais e especialistas no acompanhamento do progresso das crianças com DDN em consonância com os requisitos validados com especialistas. A seguir, nas Figuras 3 e 4, tem-se um exemplo de relatório de um jogador.

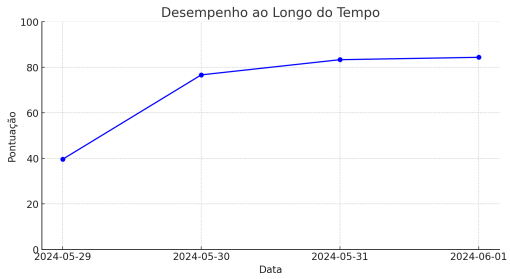

Figura 3. Desempenho ao Longo do Tempo

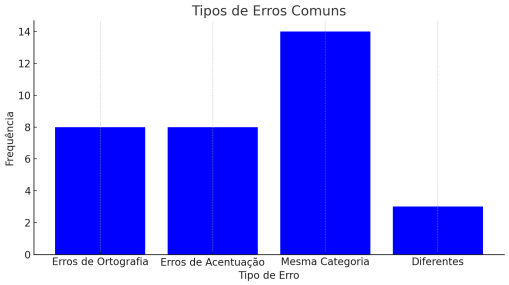

Figura 4. Tipos de Erros Comuns

Esses relatórios facilitaram a identificação das áreas de dificuldade que a criança possuía. Adicionalmente, a possibilidade de acompanhar o progresso ao longo do tempo permitia que pais e especialistas avaliassem a eficácia das intervenções e ajustassem as estratégias de ensino conforme necessário.

4.3. Comparação com outro backend desenvolvido

O sistema de Cabral (2023) foi desenvolvido com o *framework* Spring Boot e utilizava o Amazon *Relational Database Service* RDS. Já no presente trabalho, foi utilizado o H2 Database como banco de dados que oferece vantagens em termos de custo, pois não é necessário pagar pelo gerenciamento do banco de dados da AWS RDS. A Tabela 1 faz um comparativo com as tecnologias usadas entre os dois sistemas.

Feature	Sistema Terapêutico Gamificado	Sistema Desenvolvido
Linguagem Usada	Java	Java
Framework de Backend	Spring Boot	Spring Boot
Banco de Dados	PostgreSQL	H2 Database
Gerenciador de Pacotes	Maven	Maven
Hospedagem	Heroku	AWS Lightsail
Cache	-	Caffeine

Tabela 1. Tecnologias utilizadas

O sistema de Cabral (2023) oferecia algumas funcionalidades que o sistema desenvolvido neste artigo não contempla, como um *frontend* para visualização dos gráficos. No entanto, o sistema proposto neste artigo se destaca pelo uso de técnicas de NLP para melhor suportar uma análise do desempenho e aprendizado dos jogadores. Isso permitiu uma avaliação mais completa das respostas para os pais e especialistas, por levar em consideração o contexto e o significado semântico das palavras avaliadas. Além disso, a estrutura proposta por este *backend* viabiliza o seu uso com outras soluções gamificadas.

5. Conclusão e Trabalhos Futuros

Este trabalho pretendeu desenvolver um *backend* com alta disponibilidade para integrar com o aplicativo ”Jogando Juntos”, analisando as interações das crianças com o jogo e fornecendo *feedback* textual aos pais e especialistas sobre a pontuação obtida, número de acertos e erros mais cometidos. Para isso, foram utilizados métodos avançados como NLP. Assim, este sistema proporciona um suporte para pais e especialistas monitorarem o progresso das crianças por meio da visualização dos tipos de erros mais cometidos para o repertório de vocabulário escolhido por eles.

O sistema desenvolvido sugere possuir escalabilidade e alta disponibilidade, devido ao uso de *Virtual Threads* e do *Amazon LightSail*, cumprindo os objetivos propostos do trabalho. Nesse sentido, pensando em pesquisas e contribuições futuras, sugere-se a implementação de uma interface gráfica para uma fácil visualização dos *dashboards*, uma vez que facilitaria a interação dos usuários com as informações, tornando o acesso aos dados mais intuitivo e visualmente atraente. O trabalho também poderia ser repensado para uma solução genérica de monitoramento de partidas de jogos de saúde.

Além disso, um sistema de autenticação e autorização seria igualmente importante para garantir que apenas os pais e especialistas autorizados acessem informações sensíveis, melhorando a segurança dos dados. Essas melhorias não apenas expandiriam a usabilidade do sistema, mas também ampliariam seu impacto positivo, oferecendo suporte ainda mais efetivo ao desenvolvimento da criança.

Referências

Cabral, V. (2023). Desenvolvimento de um sistema computacional para monitoramento terapêutico gamificado. Trabalho de conclusão de curso, Pontifícia Universidade Católica de Minas Gerais. Disponível na biblioteca da PUC Minas.

Gamma, E., Helm, R., Johnson, R., e Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH.

Kaweski, L. (2024). Jogando juntos: Uma aplicação móvel gamificada para facilitar a interação de crianças que possuem distúrbios do desenvolvimento neurológico (ddn) com adultos. Trabalho de conclusão de curso, Pontifícia Universidade Católica de Minas Gerais. Disponível na biblioteca da PUC Minas.

Kumar, V. (2021). Exploring java virtual threads. *Oracle Technical Articles*. Accessed: 2024-06-01.

Levenshtein, V. I. et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Manes, B. (2024). Caffeine: A high performance caching library for java. `https://github.com/ben-manes/caffeine`. Accessed: 2024-06-01.

Mojžišová, A. e Mojžiš, M. (2012). Unified platform for the delivery of notifications to smartphones notification. In *Proceedings of the 13th International Carpathian Control Conference (ICCC)*, pages 490–494.

Moreira, H. (2008). Gartic. `https://gartic.io`. Accessed: 2024-06-01.

Mueller, T. (2024). H2 database engine. `http://www.h2database.com`. Accessed: 2024-06-01.

Mythily, M., Raj, A. S. A., e Joseph, I. T. (2022). An analysis of the significance of spring boot in the market. In *2022 International Conference on Inventive Computation Technologies (ICICT)*, pages 1277–1281. IEEE.

OpenAI (2023). Six strategies for getting better results. `https://platform.openai.com/docs/guides/prompt-engineering/six-strategies-for-getting-better-results`.

Philips, L. (1990). Hanging on the metaphone. *Computer Language*, 7(12):39–43.

Prompting Guide (2023). Prompting guide: Prompts. `https://www.promptingguide.ai/prompts`.

Silveira, A. C., Pires, C., e Silva, M. I. (2019). Análise do discurso de crianças com distúrbios da fala e linguagem utilizando técnicas de processamento de linguagem natural. *Revista Brasileira de Linguística Aplicada*.