Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Sistemas de Informação

Desenvolvimento de um Jogo de Simulação Empresarial como Ferramenta de Apoio ao Ensino de Conceitos Administrativos

Miguel Sales de Almeida Lopes

JUIZ DE FORA
JULHO, 2026

Desenvolvimento de um Jogo de Simulação Empresarial como Ferramenta de Apoio ao Ensino de Conceitos Administrativos

Miguel Sales de Almeida Lopes

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Sistemas de Informação

Orientador: Ciro de Barros Barbosa

JUIZ DE FORA

JULHO, 2026

Desenvolvimento de um Jogo de Simulação Empresarial como Ferramenta de Apoio ao Ensino de Conceitos Administrativos

Miguel Sales de Almeida Lopes

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Aprovada por:

Ciro de Barros Barbosa
Professor Doutor em Ciência da Computação pela University of Twente, Holanda

Igor de Oliveira Knop
Professor Doutor em Modelagem Computacional pela Universidade Federal de Juiz de Fora

Ronney Moreira de Castro
Professor Doutor em Informática pela Universidade Federal do Estado do Rio de Janeiro

JUIZ DE FORA
10 DE JULHO, 2026

Resumo

O ensino de conceitos administrativos é, em muitos casos, limitado a abordagens teóricas que dificultam a compreensão dos desafios reais enfrentados pelas organizações. Este trabalho apresenta o desenvolvimento de uma aplicação *web* de simulação empresarial, voltada ao apoio do ensino de gestão empresarial, na qual professores configuram modelos de simulação fundamentados na Dinâmica de Sistemas e alunos competem entre si gerenciando empresas fictícias em tempo real. A plataforma integra a SimDS, responsável pela execução dos modelos de simulação, com uma arquitetura *web* composta por um *backend* em Java e Spring Boot e um *frontend* em React, comunicando-se por meio de WebSocket para a transmissão dos resultados a cada passo da simulação. O sistema permite que múltiplos jogadores ajustem parâmetros de suas empresas durante a partida e acompanhem seu desempenho por meio de gráficos e placares atualizados automaticamente. A aplicação foi implantada em produção, com *frontend* hospedado na Vercel e *backend* em container Docker no Render. A solução desenvolvida demonstra a viabilidade de uma plataforma configurável, acessível via navegador e capaz de oferecer uma experiência competitiva e dinâmica para o ensino de conceitos administrativos.

Palavras-chave: jogos empresariais, dinâmica de sistemas, gamificação, simulação, ensino de administração.

Abstract

The teaching of administrative concepts is often limited to theoretical approaches that limit the understanding of the real challenges faced by organizations. This work presents the development of a *web* application for business simulation, aimed at supporting management education, in which teachers configure simulation models grounded in System Dynamics and students compete with each other by managing fictitious companies in real time. The platform integrates SimDS, responsible for executing the simulation models, with a *web* architecture composed of a *backend* in Java and Spring Boot and a *frontend* in React, communicating through WebSocket for the transmission of results at each simulation step. The system allows multiple players to adjust their companies' parameters during the match and track their performance through automatically updated charts and leaderboards. The application was deployed to production, with the *frontend* hosted on Vercel and the *backend* in a Docker container on Render. The developed solution demonstrates the feasibility of a configurable, browser-accessible platform capable of offering a competitive and dynamic experience for teaching administrative concepts.

Keywords: Business games, System Dynamics, Gamification, Simulation, Management education.

Agradecimentos

À Deus, pois sem Ele eu não estaria aqui e nada disso seria possível. Obrigado por me proteger e me guiar.

Aos meus pais, por terem me apoiado e me amado incondicionalmente desde o meu primeiro dia de vida. Não há palavras que possam descrever o quanto eu amo vocês e sou grato por tudo que fizeram e abriram mão para eu chegar onde estou hoje. Obrigado mãe, por me mostrar que quando queremos fazer grandes coisas, sacrifícios são necessários. Eles vão doer, mas no final tudo vai valer a pena. Obrigado pai, por me ensinar que a vida será cheia de obstáculos e muitos deles vão me abalar, mas nada disso importa se eu me levantar e tentar de novo. Vocês conseguiram. O filho de vocês vai se formar em uma universidade federal.

À todos os meus tios e tias, primos e primas, por sempre terem estendido a mão quando eu precisei, desde a escola até o vestibular. Obrigado tia Heloísa, queria que você estivesse aqui também.

À Luísa, minha parceira e namorada, por estar comigo durante os momentos bons, e principalmente nos ruins. Obrigado por não me deixar desistir e me fazer enxergar que sou capaz. Com você, a vida é muito melhor. Eu te amo.

Aos meus amigos de Três Rios e todos que eu conheci durante a gradução. Quero levar vocês para o resto da vida.

À Cleo e ao Batata, meus amiguinhos de quatro patas, que me ensinaram a ser alguém mais paciente e carinhoso.

Ao professor Ciro, por toda a paciência e apoio durante a escrita deste trabalho.

Ao Departamento de Ciência da Computação, o Instituto de Ciências Exatas e a Universidade Fedaral de Juiz de Fora e todos os seus professores por terem me proporcionado uma educação pública, gratuita e de qualidade.

"Todos os homens sonham, mas não da mesma forma. Os que sonham de noite, nos recessos poeirentos das suas mentes, acordam de manhã para verem que tudo, afinal, não passava de vaidade. Mas os que sonham acordados, esses são homens perigosos, pois realizam os seus sonhos de olhos abertos, tornando-os possíveis.".

T.E. Lawrence, Os Sete Pilares da Sabedoria.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
JPA	Java Persistence API
JWT	JSON Web Token
MVC	Model-View-Controller
ORM	Object-Relational Mapping
REST	Representational State Transfer
UFJF	Universidade Federal de Juiz de Fora
UUID	Universally Unique Identifier

1 Introdução

Este capítulo apresenta o contexto do trabalho, contemplando a apresentação do tema, a justificativa, os objetivos geral e específicos, a metodologia adotada e a estrutura do trabalho.

1.1 Apresentação do tema

Atualmente, com o avanço tecnológico e digital da sociedade, tornou-se possível o desenvolvimento de novas formas de ensino e aprendizagem (DIAS, 2012). Nesse contexto, a gamificação pode ser entendida como a incorporação de mecânicas e elementos típicos de jogos, como pontuação, desafios e feedback, em contextos que não são originalmente lúdicos, com o objetivo de tornar processos de ensino mais dinâmicos e engajadores (MEROTO et al., 2024). Este trabalho, no entanto, situa-se predominantemente do lado da simulação para o ensino, com enfoque na representação matemática do comportamento das empresas simuladas, sendo os aspectos competitivos um reforço opcional dessa experiência.

Os jogos empresariais destacam-se como ferramentas educacionais que ilustram essa aplicação de elementos lúdicos ao ensino. Eles podem auxiliar e facilitar a inserção dos estudantes no mercado de trabalho, além de proporcionar uma base de conhecimento mais relevante para a sociedade e para organizações (SANTOS et al., 2014).

Diante disso, este trabalho propõe o desenvolvimento de uma plataforma *web* de simulação computacional, com elementos de jogos empresariais, voltada ao apoio do ensino de conceitos administrativos em ambiente educacional.

1.2 Justificativa

O ensino de conceitos administrativos, em muitos casos, limita-se a abordagens teóricas que dificultam a compreensão dos desafios organizacionais. Os jogos empresariais e os ele-

mentos de gamificação apresentam-se como uma estratégia para estimular o aprendizado ativo e o desenvolvimento do pensamento estratégico (SANTOS et al., 2014).

Conforme será discutido no Capítulo 3 deste trabalho, parte das soluções existentes possui limitações relevantes: muitas não são baseadas em modelos matemáticos reais, restringindo-se a regras simplificadas que pouco refletem a dinâmica do mercado. Outras são de acesso pago, de difícil utilização ou pouco flexíveis, não permitindo que o professor adapte o cenário de simulação às necessidades de cada contexto.

Assim, a Dinâmica de Sistemas surge como uma abordagem adequada para suprir essa lacuna, visto que estes modelos matemáticos permitem representar de forma mais realista o funcionamento das empresas, tornando o processo de ensino mais interativo e próximo das situações reais do mercado (BASTOS, 2003).

Portanto, justifica-se a proposta de uma aplicação baseada em modelos dinâmicos de simulação, com elementos de jogos empresariais, permitindo que múltiplos alunos compitam em tempo real, tomem decisões com base em gráficos atualizados a cada passo da simulação e que professores possam configurar os parâmetros do modelo de acordo com o contexto pedagógico desejado, objetivando contribuir para o auxílio no ensino de gestão empresarial.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver uma aplicação *web* que ofereça uma alternativa ao ensino de conceitos de gestão empresarial, integrando um modelo de simulação fundamentado na Dinâmica de Sistemas a uma plataforma configurável, na qual os alunos possam experimentar e tomar decisões sobre empresas fictícias em tempo real.

1.3.2 Objetivos Específicos

- Disponibilizar uma plataforma *web* na qual professores e alunos possam criar, configurar e acompanhar simulações empresariais.

- Fundamentar o comportamento das simulações em modelos de Dinâmica de Sistemas, por meio da integração com a SimDS.
- Possibilitar a adaptação do cenário de simulação pelo professor, de acordo com o contexto pedagógico desejado.
- Permitir que os alunos acompanhem, em tempo real, o impacto de suas decisões ao ajustar os parâmetros de suas empresas durante a partida.
- Oferecer recursos visuais que favoreçam a análise do desempenho das empresas simuladas.
- Estimular o engajamento dos alunos por meio de elementos competitivos, como a comparação de desempenho entre as empresas simuladas.

1.4 Metodologia

A metodologia adotada neste trabalho foi voltada ao desenvolvimento de uma aplicação web que utiliza a simulação do funcionamento de empresas fictícias como apoio ao ensino de conceitos administrativos. O trabalho centrou-se no projeto e na implementação da ferramenta, desde o levantamento de requisitos até a realização de testes.

Inicialmente, foi realizado o levantamento de requisitos funcionais e não funcionais do sistema, com o objetivo de definir o escopo da aplicação e identificar as principais funcionalidades a serem implementadas. Nessa etapa, foram estabelecidos os recursos disponíveis na plataforma e as limitações do sistema.

Em seguida, foi definida a arquitetura do sistema, estabelecendo a organização dos componentes da aplicação e a separação de responsabilidades entre o servidor e o cliente. Foram tomadas decisões relacionadas à estrutura do *backend*, do *frontend* e à comunicação entre eles, considerando os requisitos de tempo real e a necessidade de suportar múltiplos usuários simultâneos.

Após a definição da arquitetura, foram selecionadas as linguagens de programação, *frameworks* e ferramentas utilizadas no desenvolvimento da aplicação. Essa escolha levou em consideração critérios como compatibilidade com aplicações *web*, facilidade de

desenvolvimento, desempenho, familiaridade e adequação aos objetivos do projeto.

Então, foi realizada a implementação do sistema, contemplando o desenvolvimento das funcionalidades previstas nos requisitos, a integração com a SimDS e os mecanismos de comunicação em tempo real entre os participantes da simulação.

Por fim, foram realizados testes das funcionalidades desenvolvidas com o objetivo de verificar o correto funcionamento dos principais recursos da aplicação, assegurando que o sistema opera de acordo com os requisitos definidos e com o escopo proposto neste trabalho.

1.5 Estrutura do trabalho

Este trabalho está organizado em cinco capítulos. O Capítulo 2 apresenta a fundamentação teórica, abordando os conceitos de gamificação aplicada ao ensino, jogos de negócios, conceitos administrativos e Dinâmica de Sistemas. O Capítulo 3 analisa os trabalhos relacionados, incluindo jogos empresariais existentes, trabalhos acadêmicos relevantes e uma discussão comparativa entre eles e o presente trabalho. O Capítulo 4 descreve o desenvolvimento da aplicação, cobrindo os requisitos levantados, a arquitetura definida, as tecnologias utilizadas, a implementação do *backend* e do *frontend*, e o processo de *implantação*. Por fim, o Capítulo 5 apresenta as conclusões do trabalho.

2 Fundamentação teórica

Neste capítulo são apresentados os principais conceitos que norteiam o desenvolvimento e a implementação do jogo de simulação empresarial proposto.

2.1 Gamificação aplicada ao ensino

A gamificação tem sido empregada em contextos de ensino para promover ambientes lúdicos e motivadores, favorecendo a aprendizagem e a assimilação de conceitos por parte dos estudantes. Segundo Fadel et al. (2014), a aplicação da gamificação, por meio de mecanismos como recompensas e *feedback* em tempo real, pode trazer diversos benefícios para os alunos, como o aumento do engajamento e da motivação, maior participação nas atividades de ensino e melhor absorção de conceitos teóricos.

Estudos também demonstraram resultados positivos do uso da gamificação em contextos reais de ensino. Observou-se, por meio de uma pesquisa realizada por Buckley e Doyle (2014), um aumento significativo no conhecimento geral dos alunos após a adoção de práticas de gamificação, com maior participação e engajamento dos estudantes nas atividades realizadas.

No entanto, a literatura aponta que a gamificação não deve ser compreendida como uma solução universal apropriada para todos os contextos de ensino. Dessa forma, torna-se fundamental o equilíbrio dos conceitos de jogos com as necessidades acadêmicas e pedagógicas de cada cenário educacional, assegurando que a aprendizagem continue sendo o ponto focal dessas ferramentas (MEROTO et al., 2024).

2.2 Jogos de negócios

Os jogos de negócios, também conhecidos como jogos empresariais, são ferramentas educacionais que simulam cenários reais vivenciados pelas organizações. Essas ferramentas permitem que os estudantes façam escolhas estratégicas, acompanhando o impacto de suas

decisões no desempenho da empresa simulada, auxiliando no desenvolvimento de análise crítica e aplicação prática de conceitos (SAUAIA, 2006).

Os jogos empresariais também possibilitam, de acordo com Pretto, Filardi e Pretto (2010), relacionar diversas variáveis de uma organização ao longo do tempo. Esses processos ocorrem em períodos sequenciais, criando resultados que refletem as decisões tomadas anteriormente, proporcionando aos estudantes uma análise próxima da dinâmica real vivenciada pelas organizações.

Apoiadas por modelos dinâmicos, essas ferramentas têm o potencial de auxiliar na compreensão dos resultados imediatos das decisões escolhidas, além dos efeitos acumulados ao longo do tempo.

2.3 Conceitos Administrativos

A gestão empresarial envolve a utilização de diversos conceitos administrativos fundamentais para o gerenciamento adequado de uma empresa, auxiliando os gestores na tomada de decisão (CHIAVENATO, 2003).

Esses conceitos, por estarem relacionados ao sucesso de uma empresa, exigem dos gestores um bom entendimento de cada um deles e de seus relacionamentos. Diferentes variáveis podem ter impacto umas sobre as outras, definindo um ambiente complexo (CHIAVENATO, 2003). Esse cenário justifica a busca por soluções que permitam aos estudantes vivenciarem ambientes de experimentação, através de simulações que se aproximam de situações reais de mercado.

A complexidade citada acima se manifesta, por exemplo, na relação entre preço e demanda: um aumento no preço de um produto pode reduzir sua demanda, o que por sua vez impacta o faturamento da empresa. Da mesma forma, uma queda na produtividade eleva os custos de produção, afetando a competitividade da organização no mercado. Essas interações refletem como decisões tomadas em uma área da gestão empresarial produzem efeitos que se propagam por outras, exigindo do gestor uma visão integrada do funcionamento da empresa.

2.4 Dinâmica de Sistemas

A Dinâmica de Sistemas, conceito desenvolvido por Jay W. Forrester, é uma abordagem que permite estudar como sistemas complexos evoluem através da representação de suas variáveis e interações (BASTOS, 2003). Tal técnica permite representar relações de causa e efeito entre variáveis, bem como os impactos acumulados das decisões adotadas.

Essa forma de modelagem tem sido utilizada para representar o funcionamento de empresas e mercados, relacionando diferentes variáveis como demanda, preço, produção e concorrência ao longo de um determinado período. Assim, essa abordagem pode ser considerada adequada para aplicações voltadas ao ensino de gestão empresarial, nas quais as decisões dos estudantes afetam o desempenho da organização simulada (BARBOSA; BONIFáCIO, 2017).

2.5 SimDS

A SimDS[1] é um conjunto de classes Java para modelagem e simulação baseada em Dinâmica de Sistemas, desenvolvido pelo Prof. Ciro de Barros Barbosa. Sua fundamentação teórica está apoiada em dois trabalhos publicados pelo autor (BARBOSA; KNOP, 2009; BARBOSA; BONIFáCIO, 2017), detalhados no Capítulo 3 deste trabalho.

Essas classes implementam uma abordagem de modelagem dividida em dois níveis. O primeiro, chamado *Expert Model*, define a estrutura geral do modelo de simulação, como as classes que representam as entidades da simulação e seus comportamentos. Essas classes possuem propriedades ajustáveis, variáveis de acumulação e calculadas, além de relações que definem como as instâncias dessas classes interagem entre si durante a simulação (BARBOSA; KNOP, 2009).

O segundo nível, denominado *User Model*, instancia essa estrutura, criando objetos concretos a partir das classes definidas no *Expert Model* e definindo valores iniciais para suas propriedades (BARBOSA; KNOP, 2009). Essa separação pode ser comparada à relação entre classes e objetos no paradigma de programação orientada a objetos.

[1]Nome adotado neste trabalho para referenciar a ferramenta, visto que não há uma denominação oficial.

A título de exemplo, o código a seguir apresenta um *Expert Model* de administração de empresas, definido na sintaxe textual da SimDS. Nele, a classe *empresa* reúne variáveis como preço, produtividade e demanda, além dos estoques de caixa e de produtos, enquanto a classe *ambiente* concentra os valores compartilhados entre as empresas, como custo unitário e demanda geral do mercado.

```
model EmpAdmin {
   timescale = dias;
   class empresa {
        property preco 80.0;
        proc produtividade if(empregados*10>100:100;empregados*10);
        property empregados 4;
        proc score caixa;
        proc custo read(r1, custoUnitario);
        proc precoTotal read(r1, somaPrecos);
        proc demandaTotal read(r1, demandaGeral);
        proc sal read(r1, salario);
        proc demandaLocal demandaTotal * preco / precoTotal;
        rate (estoque) venda - if(demandaLocal<=estoque:demandaLocal;
           estoque);
        rate (estoque) producao produtividade;
        rate (caixa) faturamento - venda * preco;
        rate (caixa) gastos - ((custo * produtividade) + (sal *
           empregados));
        stock estoque 0;
        stock caixa 10000;
   };
   class ambiente {
        property custoUnitario 18;
        property demandaGeral 80;
        proc somaPrecos read(r2, preco) + 40;
        property salario 500;
   };
   relation r1 empresa, ambiente;
   multirelation r2 ambiente, empresa;
};
```

Esse mesmo modelo pode ser representado graficamente na notação de estoques e fluxos da Dinâmica de Sistemas, como ilustra a Figura 2.1, na qual os retângulos representam os estoques (caixa e estoque) e as demais variáveis influenciam seus fluxos ao longo da simulação.

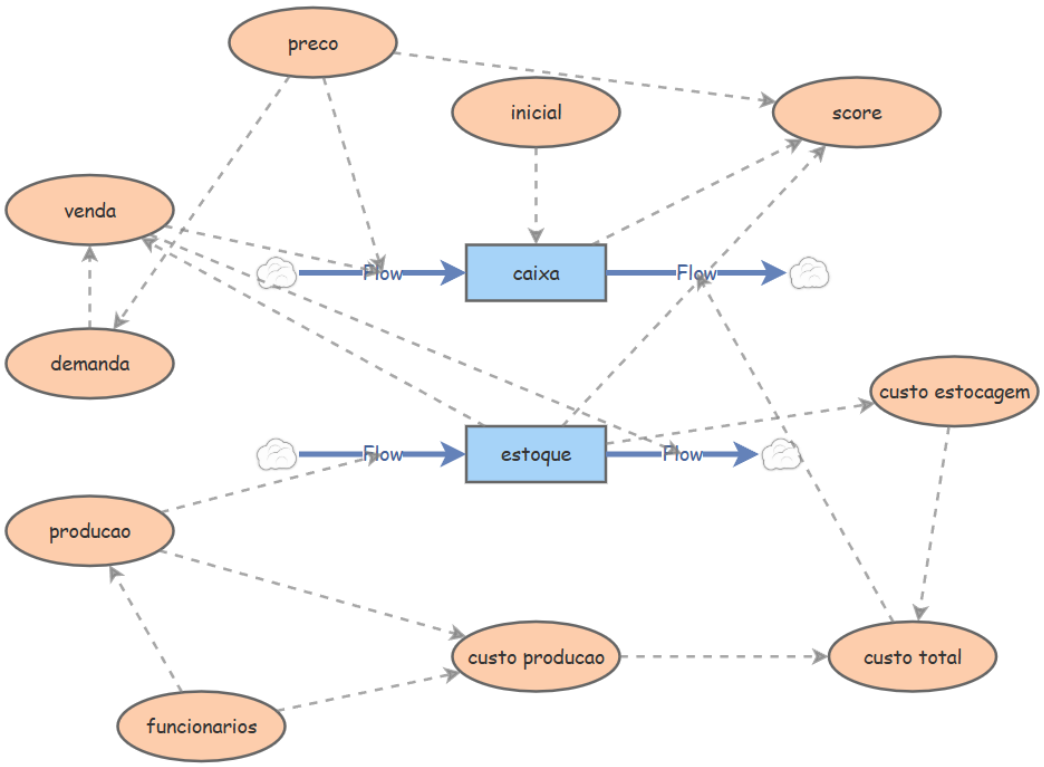

Figura 2.1: Representação gráfica do *Expert Model* na notação de estoques e fluxos. Fonte: elaborado pelo Prof. Ciro de Barros Barbosa na ferramenta Insight Maker.

A SimDS é disponibilizada como um conjunto de classes Java, contendo os pontos de extensão necessários para executar e acompanhar o andamento da simulação. Esse conjunto foi utilizado como base para a integração com o sistema desenvolvido neste trabalho, conforme detalhado nos capítulos seguintes.

3 Trabalhos relacionados

Este capítulo apresenta uma análise de jogos empresariais existentes e de trabalhos acadêmicos relacionados ao desenvolvimento deste projeto, seguida de uma discussão comparativa entre eles e a proposta apresentada neste trabalho.

3.1 Jogos empresariais

Nesta seção são descritos alguns jogos empresariais existentes, destacando suas principais características, pontos fortes e limitações. No mercado existem soluções desenvolvidas por empresas especializadas em simulações empresariais, muitas delas com alto nível de complexidade. No entanto, parte dessas soluções apresenta diversos pontos relevantes que impedem ou dificultam seu acesso e uso em contextos educacionais, como a disponibilização somente através de licenças pagas, a necessidade de instalação local, a utilização de modelos de simulação fixos, com pouca flexibilidade de acordo com o contexto de uso, e a ausência de interface em português.

A análise apresentada nesta seção tem enfoque em jogos empresariais com escopo menor e acesso gratuito, voltados a experiências introdutórias de gestão empresarial.

3.1.1 *Nem parece jogo de Pizzaria*

O jogo *Nem parece jogo de Pizzaria* foi desenvolvido pelo Sebrae, uma entidade brasileira que tem como objetivo o desenvolvimento de micro e pequenas empresas. O jogo está disponível de forma gratuita por meio da plataforma digital do próprio Sebrae, com acesso realizado via navegador, após cadastro do usuário. [2]

[2]Disponível em: ⟨https://sebrae.com.br/sites/PortalSebrae/cursosonline/⟩

Figura 3.1: Menu inicial do jogo *Nem Parece Jogo de Pizzaria*. Fonte: Sebrae.

O principal objetivo do jogo é simular o funcionamento de uma pizzaria, através da administração de um ambiente virtual. Ao longo da simulação, o jogador tem contato com diferentes conceitos relacionados à gestão empresarial, como a montagem do ponto de venda, a elaboração do cardápio, a contratação, qualificação e demissão de funcionários, o gerenciamento de estoque e a aquisição de equipamentos e estoque de ingredientes.

Além disso, o jogo aborda aspectos competitivos do mercado, como a concorrência por clientes, bem como estratégias de marketing, ativação de ações promocionais e contratação de meios de divulgação, como jornal, rádio, internet e televisão. As decisões do jogador são realizadas por meio de interfaces pré-definidas, sem que as regras que determinam o comportamento da simulação sejam apresentadas ao usuário. Dessa forma, o entendimento da relação entre conceitos ao longo da simulação acaba sendo limitado.

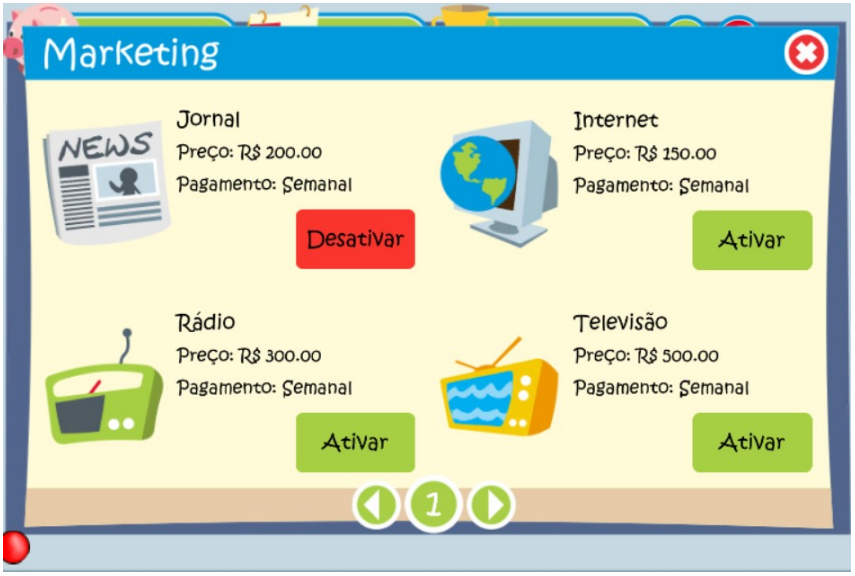

Figura 3.2: Tela de seleção de estratégias de marketing do jogo *Nem Parece Jogo de Pizzaria*. Fonte: Sebrae.

Entre os pontos positivos do jogo, destacam-se a interface visualmente agradável e a presença de efeitos visuais que contribuem para uma experiência mais imersiva. O jogo também permite o salvamento do progresso do usuário, possibilitando a continuidade da simulação em diferentes momentos, o que pode favorecer o uso em contextos educacionais.

Por outro lado, o jogo é voltado a uma experiência individual, sem suporte a múltiplos jogadores, o que reflete seu escopo introdutório e reduz o potencial competitivo em comparação a soluções *multiplayer*. Além disso, o processo de instrução inicial é baseado em um tutorial estático, que não acompanha o jogador ao longo das etapas da simulação, podendo dificultar a compreensão das mecânicas do jogo para usuários sem familiaridade prévia.

Figura 3.3: Exemplo de um dos tutoriais do jogo *Nem Parece Jogo de Pizzaria*. Fonte: Sebrae.

3.1.2 Fazendinha de Negócios

O jogo *Fazendinha de Negócios*, também desenvolvido pelo Sebrae, integra o conjunto de jogos educacionais disponibilizados gratuitamente pela instituição em sua plataforma digital, com acesso realizado via navegador.[3]

Figura 3.4: Menu inicial do jogo *Fazendinha de Negócios*. Fonte: Sebrae.

O principal objetivo do jogo é simular a administração de uma fazenda fictícia.

[3]Disponível em: ⟨https://sebrae.com.br/sites/PortalSebrae/cursosonline/⟩

O jogador controla diretamente o fazendeiro, que se movimenta pelo cenário realizando ações como coletar a produção, encaixotá-la e enviá-la para a cidade. É possível adquirir insumos e realizar melhorias na fazenda, como o aumento da capacidade do celeiro e da mão de obra disponível, o que impacta na produtividade ao longo do jogo. O jogador acompanha a evolução do caixa e precisa cumprir metas mensais de desempenho, o que traz uma noção de planejamento financeiro e gestão de recursos disponíveis.

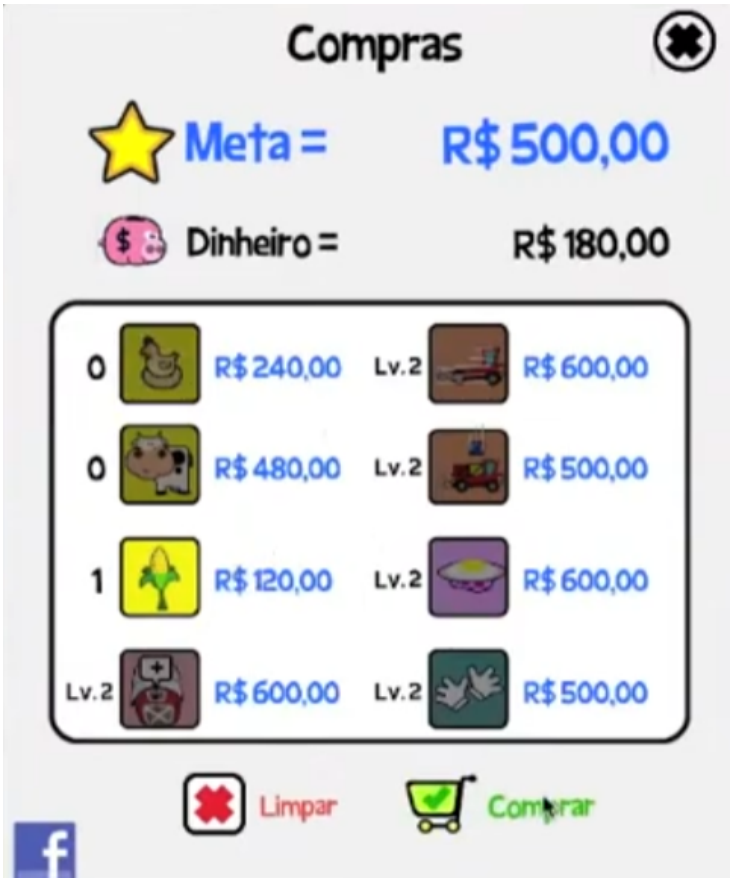

Figura 3.5: Tela de compra de insumos e melhorias da *Fazendinha de Negócios*. Fonte: Sebrae.

Entre os pontos positivos do jogo, destaca-se a interface interativa e o caráter lúdico da ambientação, com o fazendeiro se movimentando visivelmente pelo cenário, o que contribui para o engajamento do jogador. A organização das atividades e a presença de metas bem definidas facilitam o acompanhamento da evolução ao longo da simulação, tornando o processo de aprendizagem mais estruturado e acessível.

Por outro lado, assim como no jogo analisado anteriormente, a experiência é voltada a um único jogador, sem suporte a modo competitivo entre participantes. As

regras que determinam o comportamento da simulação também seguem o mesmo padrão simples e trivial, limitando-se a relações diretas entre produção, venda e lucro. Isso reduz a profundidade da experiência e limita a compreensão do impacto real das decisões tomadas ao longo do jogo.

Figura 3.6: Tela principal do jogo *Fazendinha de Negócios*. Fonte: Sebrae.

3.2 Trabalhos acadêmicos

Nesta seção serão analisados dois trabalhos diretamente relacionados ao desenvolvimento deste projeto. O primeiro apresenta a abordagem de modelagem que fundamenta a SimDS, descrevendo suas linguagens de especificação e mecanismos de validação, enquanto o segundo sugere uma arquitetura conceitual para ambientes de simulação voltados ao aprendizado, construída sobre conceitos definidos no primeiro. Ambos serviram como referência para as decisões de projeto descritas nos capítulos seguintes.

3.2.1 *Meta-modelagem no domínio de sistemas dinâmicos*

O trabalho *Tool supported meta-modeling and code generation for the dynamic systems domain*, de Barbosa e Knop (2009), propõe uma abordagem para o desenvolvimento de soluções de simulação baseadas em conceitos de Dinâmica de Sistemas, permitindo a definição de linguagens de modelagem para especificar os domínios dessas simulações, jun-

tamente a mecanismos de validação. A motivação do trabalho é oferecer uma ferramenta que combine elevação do nível de abstração do modelo, geração automática de código e validação semântica, características que não eram reunidas em conjunto pelas soluções comerciais existentes na época, como Stella e PowerSim.

Para isso, o trabalho apresenta a formalização das linguagens textuais utilizadas para definição do *Expert Model* e do *User Model*, apresentando sua sintaxe completa. A ferramenta desenvolvida inclui um compilador capaz de validar sintática e semanticamente os modelos escritos nessas linguagens, garantindo que apenas modelos corretos sejam executados.

O trabalho também descreve a possibilidade de geração de um conjunto de classes Java a partir dos modelos definidos, permitindo que sistemas externos se integrem à simulação. Essa característica foi relevante para o desenvolvimento deste projeto, conforme detalhado nos capítulos seguintes. Por fim, como estudo de caso, os autores aplicam essa abordagem no domínio de gerenciamento de projetos de *software*, demonstrando sua flexibilidade de uso para diferentes contextos.

3.2.2 Ambiente de simulação baseado em modelos

O trabalho *Model-Based Environment for Learning Simulations*, de Barbosa e Bonifácio (2017), propõe uma arquitetura para ambientes de jogos de negócios, onde o comportamento do jogo é definido pelos próprios usuários por meio de modelos. A motivação central é permitir a configuração e reutilização de modelos de simulação, aplicando-os em múltiplas simulações simultâneas, voltadas ao ensino de gestão empresarial.

A arquitetura proposta define quatro perfis de usuário: o pesquisador é responsável por criar e validar modelos, o professor por configurar e iniciar as partidas, o aluno por interagir com a simulação e o administrador por gerenciar usuários e monitorar o servidor.

Os principais componentes da arquitetura são o *View Manager*, o *Artefacts Manager* e o *Execution Engine*. O primeiro é responsável pelo controle de acesso e exibição dos dados conforme o perfil do usuário, o segundo gerencia modelos e jogos, e o terceiro executa as simulações de forma centralizada, suportando múltiplas partidas simultâneas

por meio de *threads*. A implementação descrita no trabalho utiliza tecnologia Java com JSPs e Servlets, com comunicação via TCP entre os componentes.

Como estudo de caso, os autores apresentam um jogo empresarial com duas empresas competindo por demanda de mercado com base em preço e investimento em *marketing*. Os resultados da simulação são exibidos por meio de gráficos atualizados sob demanda, mediante ação do usuário. Esse trabalho serviu como principal referência conceitual para o desenvolvimento deste projeto, especialmente em relação a separação de perfis, fluxo de execução da simulação e organização dos componentes do sistema.

3.3 Discussão dos trabalhos relacionados

Os jogos empresariais analisados, embora apresentem uma abordagem mais acessível e introdutória aos conceitos de gestão, possuem limitações relevantes para uso em contextos educacionais mais exigentes. Por serem jogos de jogador único, não permitem a competição direta entre participantes, o que reduz o potencial de engajamento e a dinâmica competitiva em sala de aula. Além disso, as decisões do jogador são baseadas em regras simplificadas, sem um modelo matemático real por trás, o que limita a compreensão do impacto das escolhas ao longo da simulação. Por fim, esses jogos não oferecem flexibilidade de configuração, sendo voltados a um cenário fixo que não pode ser adaptado pelo professor de acordo com o contexto pedagógico desejado.

Existem também soluções mais complexas e robustas disponíveis no mercado, como Sim Companies e Virtonomics. No entanto, essas ferramentas apresentam características que dificultam sua adoção em contextos educacionais, como a disponibilização mediante licenças pagas, interfaces exclusivamente em inglês e uma curva de aprendizado elevada. Além disso, parte dessas soluções é voltada a um único contexto de simulação fixo, sem possibilidade de adaptação. Outras oferecem cenários diferentes, mas cada um é disponibilizado como uma solução separada, exigindo novas aquisições para diferentes contextos, o que reduz a flexibilidade de uso em ambientes educacionais.

O trabalho de Barbosa e Bonifácio (2017), por sua vez, propõe uma arquitetura conceitual próxima ao que foi desenvolvido neste projeto. No entanto, apresenta algumas limitações relacionadas às tecnologias utilizadas. A atualização dos gráficos da simulação,

por exemplo, é realizada de forma manual pelo usuário, sem atualização em tempo real. A comunicação entre os componentes é feita via TCP, com parte da aplicação rodando como um programa *desktop*, o que pode exigir uma infraestrutura mais complexa para implantação. Por fim, o *User Model* é definido manualmente, sem geração dinâmica a partir dos participantes do jogo.

Este trabalho busca suprir as limitações identificadas, propondo uma plataforma web de simulação empresarial, que combina um modelo matemático baseado em Dinâmica de Sistemas com elementos competitivos em tempo real, permitindo que os participantes alterem os parâmetros da simulação a qualquer momento durante a partida. A comunicação entre o sistema e os participantes é feita via WebSocket[4], permitindo que os gráficos e o placar sejam atualizados a cada passo da simulação, sem necessidade de ação do usuário. O *User Model* é gerado dinamicamente pelo sistema a partir dos participantes do jogo, dispensando configuração manual. A plataforma é desenvolvida com tecnologias modernas e amplamente adotadas no mercado, com frontend e backend separados em duas aplicações independentes, o que favorece a manutenção e a escalabilidade do sistema. A aplicação é disponibilizada em nuvem, acessível via navegador sem necessidade de instalação, e permite que o professor configure o modelo de simulação de acordo com o contexto pedagógico desejado, possibilitando, por exemplo, a definição de um mercado compartilhado entre as empresas simuladas, onde as decisões de cada participante afetam diretamente os resultados dos demais.

[4]Protocolo de comunicação que permite troca de mensagens bidirecionais em tempo real entre cliente e servidor.

4 Desenvolvimento

Este capítulo descreve o desenvolvimento da aplicação proposta, apresentando os requisitos levantados, a arquitetura definida, as tecnologias utilizadas, a implementação do *backend* e do *frontend*, e o processo de *implantação* em produção.

4.1 Contexto geral do sistema

O sistema desenvolvido neste trabalho é uma plataforma *web* de simulação empresarial voltada ao ensino de conceitos administrativos.

Esta solução permite que professores criem jogos nos quais os alunos assumem o papel de gestores de empresas fictícias, tomando decisões com base nas métricas de desempenho exibidas em tempo real, e competindo entre si pela vitória da partida. O comportamento da simulação é definido por um modelo matemático configurável pelo professor, que pode especificar variáveis, regras de cálculo e relações entre elas, como a disputa por demanda de mercado. O professor também é capaz de definir os parâmetros da partida, como o número máximo de jogadores, a quantidade de passos da simulação e o intervalo de tempo entre eles, permitindo adequar a duração e a escala do jogo conforme desejado.

A plataforma é organizada em torno de três perfis de usuários: os administradores, responsáveis pelo gerenciamento das organizações cadastradas no sistema e de seus participantes, os professores, que configuram o modelo de simulação da organização, criam os jogos e acompanham as partidas em andamento, e os alunos, que participam dos jogos gerenciando suas empresas fictícias, ajustando os parâmetros da simulação e acompanhando seu desempenho por meio de uma interface visual. Fora das partidas, os alunos têm acesso a uma página inicial com os jogos disponíveis para entrada, seu histórico de partidas e sua melhor pontuação, além de um *ranking* geral da organização.

Após o início da partida, a simulação é executada passo a passo pelo servidor. A cada passo, os resultados são exibidos em tempo real para todos os participantes, que

acompanham a evolução de suas empresas por meio de gráficos e de um placar geral de pontuação atualizado automaticamente. Durante a partida, os alunos podem ajustar os parâmetros de suas empresas definidos pelo professor no modelo, como preço de venda e quantidade produzida, observando o impacto de suas decisões nos passos seguintes da simulação. Ao final, o sistema determina o vencedor com base na pontuação acumulada e exibe o resultado final para todos os participantes.

4.2 Requisitos

Os requisitos do sistema foram levantados de acordo com as necessidades do projeto e organizados em dois grupos: os requisitos funcionais, que descrevem as funcionalidades esperadas do sistema, e os requisitos não funcionais, que estão mais relacionados a critérios de qualidade e restrições técnicas da aplicação.

4.2.1 Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades que o sistema deve oferecer aos seus usuários. A Tabela 4.1 apresenta os requisitos funcionais levantados, organizados por identificador e prioridade.

Tabela 4.1: Requisitos Funcionais

ID	**Descrição**	**Prioridade**
RF01	O sistema deve permitir o cadastro e gerenciamento de organizações.	Alta
RF02	O sistema deve permitir o cadastro e gerenciamento de usuários que estão atrelados a uma organização.	Alta
RF03	O sistema deve possibilitar a autenticação dos usuários por meio de login e senha.	Alta

ID	**Descrição**	**Prioridade**
RF04	O sistema deve permitir que o professor configure o modelo de simulação da organização.	Alta
RF05	O sistema deve permitir que o professor crie jogos e configure seus parâmetros, como número máximo de jogadores, quantidade de passos e intervalo entre eles.	Alta
RF06	O sistema deve permitir que alunos entrem em jogos disponíveis.	Alta
RF07	O sistema deve executar a simulação passo a passo após o início da partida.	Alta
RF08	O sistema deve exibir os resultados de cada passo da simulação em tempo real para todos os participantes.	Alta
RF09	O sistema deve permitir que os alunos ajustem os parâmetros de suas empresas durante a partida.	Alta
RF10	O sistema deve exibir gráficos de desempenho e um placar geral de pontuação durante a partida.	Alta
RF11	O sistema deve exibir e registrar o vencedor ao final da simulação com base na pontuação acumulada.	Alta
RF12	O sistema deve exibir uma página inicial com jogos disponíveis, histórico de partidas e melhor pontuação do aluno.	Média
RF13	O sistema deve disponibilizar um *ranking* geral de vencedores da organização.	Média

4.2.2 Requisitos Não Funcionais

Os requisitos não funcionais definem as restrições e critérios de qualidade que o sistema deve cumprir. A Tabela 4.2 apresenta os requisitos não funcionais levantados, organizados por identificador e categoria.

Tabela 4.2: Requisitos Não Funcionais

ID	**Descrição**	**Categoria**
RNF01	O sistema deve ser disponibilizado em nuvem, acessível por meio de um navegador *web*, sem necessidade de instalação local.	Disponibilidade
RNF02	O sistema deve suportar múltiplos usuários simultâneos em uma mesma partida.	Desempenho
RNF03	A transmissão de dados da simulação entre servidor e clientes deve ser feita via *WebSocket*.	Disponibilidade
RNF04	O sistema deve autenticar os usuários por meio de tokens JWT, garantindo acesso seguro às rotas da aplicação.	Segurança
RNF05	O acesso às funcionalidades deve ser restrito de acordo com o perfil do usuário.	Segurança
RNF06	O *frontend* e o *backend* devem ser aplicações independentes, comunicando-se por meio de uma API REST.	Manutenibilidade
RNF07	O sistema deve validar os valores inseridos pelos alunos, respeitando os limites mínimos e máximos definidos no modelo.	Confiabilidade

ID	Descrição	Categoria
RNF08	A interface do sistema deve ser otimizada para uso em dispositivos *desktop*, com suporte a diferentes resoluções de tela.	Usabilidade

4.3 Arquitetura

A ferramenta desenvolvida neste trabalho possui duas aplicações distintas: o *frontend*, responsável pela interface visual e interação do usuário com o sistema, e o *backend*, responsável pela autenticação, processamento das regras de negócio, execução da simulação e persistência de informações no banco de dados. Essa separação foi adotada por favorecer a manutenção e a escalabilidade do sistema, permitindo que cada aplicação seja modificada de forma independente. A comunicação entre elas é feita de duas formas: por meio de uma API REST para operações como registro de usuários, criação de jogos e autenticação, e por meio de *WebSocket* para o envio dos dados da simulação em tempo real durante as partidas, este sendo um protocolo mais adequado para esse tipo de comunicação contínua e bidirecional. Os dados relevantes para a aplicação são persistidos em um banco de dados PostgreSQL, que é acessado somente pelo *backend*.

A Figura 4.1 ilustra a arquitetura geral do sistema, apresentando os principais componentes e as formas de comunicação entre eles.

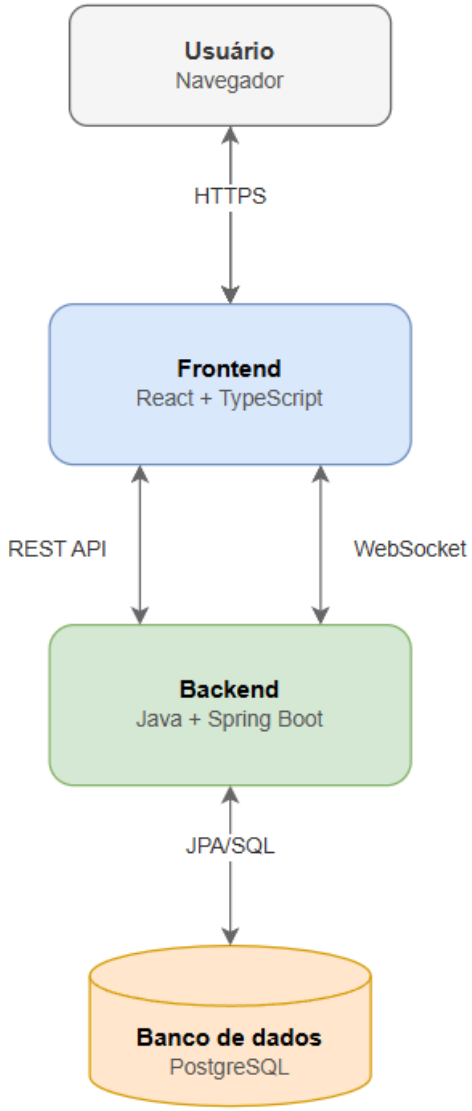

Figura 4.1: Arquitetura geral do sistema. Fonte: elaborado pelo autor.

O *backend* foi desenvolvido seguindo uma arquitetura em camadas, separando responsabilidades entre os diferentes componentes do sistema. O código foi organizado em pacotes que refletem essa separação: o *controller* contém as classes responsáveis por receber as requisições HTTP e *WebSocket* e redirecioná-las para o *service*, que concentra as regras de negócio da aplicação e utiliza o *repository* para acessar o banco de dados, persistindo e recuperando os dados conforme necessário.

Foram definidos também pacotes com responsabilidades mais específicas: *entity*, *dto*, *config* e *simulation*. O primeiro define as entidades do sistema e seus relacionamentos com o banco de dados, o segundo concentra os objetos de transferência de dados das re-

quisições e respostas da API, o terceiro agrupa as configurações de segurança, autenticação e *WebSocket*, e o último contém as classes responsáveis pela execução e gerenciamento das simulações.

Já o *frontend* foi organizado seguindo uma estrutura baseada em funcionalidades, onde cada *feature* da aplicação possui sua própria pasta com os componentes, *hooks* e páginas relacionadas, facilitando a manutenção e o isolamento das responsabilidades de cada módulo. As principais features são: *auth*, responsável pelo fluxo de autenticação, *game*, concentrando os componentes da página da partida, e *organization*, que agrupa as funcionalidades de gerenciamento da organização. Também existe uma pasta *shared*, que contém componentes e utilitários reutilizáveis entre as diferentes features da aplicação.

4.4 Tecnologias utilizadas

As tecnologias utilizadas no desenvolvimento da aplicação foram selecionadas com base em critérios como compatibilidade com os requisitos do sistema, familiaridade com as ferramentas e utilização no mercado. A combinação dessas tecnologias permitiu desenvolver uma aplicação web moderna, capaz de suportar múltiplos usuários simultaneamente, comunicação em tempo real e integração com o mecanismo de simulação utilizado neste trabalho.

4.4.1 Backend

O *backend* da aplicação foi desenvolvido utilizando a versão 17 da linguagem Java, em conjunto com o *framework* Spring Boot 3.5.0. A escolha do Java como linguagem principal se deu pela necessidade de integração com a SimDS, visto que garantia compatibilidade direta sem a necessidade de camadas adicionais de integração.

O *Spring Boot* foi escolhido como *framework* principal por facilitar e simplificar a configuração e o desenvolvimento da aplicação, reduzindo a necessidade de configurações manuais e acelerando o processo de desenvolvimento. Dentre os módulos empregados, o *Spring Web* foi responsável pela criação dos endpoints REST, o *Spring Data* JPA pelo acesso ao banco de dados por meio do ORM Hibernate, e o *Spring Security* pelo controle

de autenticação e autorização dos usuários.

A autenticação da plataforma é baseada em *tokens* JWT, implementada com a biblioteca JJWT 0.12.3. Essa abordagem permite que o servidor carregue no próprio *token* de autenticação as informações necessárias para o controle de acesso, como o perfil do usuário e o identificador da organização à qual pertence.

Para a comunicação em tempo real durante a simulação, foi empregado o módulo de *WebSocket* nativo do *Spring Boot*, fazendo com que as informações geradas durante a simulação fossem transmitidas automaticamente para os participantes, sem a necessidade de qualquer ação do usuário para atualizar os dados exibidos em tela.

O banco de dados adotado foi o PostgreSQL, integrado à aplicação por meio do *Spring Data* JPA. O *Lombok* foi escolhido por reduzir a verbosidade do código, eliminando a necessidade de escrever manualmente *getters* e *setters* para todas as classes. Por fim, o Maven foi utilizado como ferramenta de *build* e gerenciamento de dependências.

4.4.2 Frontend

O *frontend* da aplicação foi desenvolvido com React 19. O React foi escolhido por permitir um desenvolvimento baseado em componentes e *hooks*, que gerencia o estado da interface de forma reativa, atualizando automaticamente os elementos em tela conforme os dados mudam. Essa característica foi especialmente relevante para a página da partida, por exemplo, onde as informações são atualizadas à medida que os dados chegam do servidor via *WebSocket*.

O *TypeScript* foi empregado como linguagem principal por fornecer tipagem estática ao código, contribuindo para a detecção de erros em tempo de compilação e para a manutenção do código, especialmente em uma aplicação como essa que possui diversos tipos de dados e componentes distintos.

Para a estilização, foram adotados o *Tailwind* CSS e *shadcn*/ui. O *Tailwind* foi escolhido por permitir estilizar a interface sem a necessidade de escrever arquivos CSS separados, agilizando o desenvolvimento. O *shadcn*/ui foi empregado por fornecer componentes prontos e acessíveis, como modais, botões e formulários, evitando a necessidade de implementá-los do zero.

O gerenciamento das requisições ao servidor foi feito com *TanStack Query* v5, pois simplifica o controle de *cache* e revalidação dos dados consumidos pela interface. As requisições em si foram realizadas com Axios, e o roteamento da aplicação foi implementado com *React Router* DOM v7.

Para a exibição dos gráficos de desempenho durante as partidas, foi adotada a biblioteca *Recharts*, que oferece componentes de gráficos configuráveis e de fácil integração com React. A biblioteca *React Toastify* foi empregada para exibir notificações de *feedback* ao usuário, como mensagens de sucesso e erro. Por fim, a biblioteca *canvas-confetti* ficou responsável por exibir um efeito visual na tela de encerramento da partida.

A comunicação em tempo real com o servidor foi implementada utilizando a API nativa de *WebSocket* do navegador, sem depender de bibliotecas adicionais.

4.4.3 Ferramentas de apoio

Foram empregadas também algumas ferramentas de apoio que não fazem parte da *stack* da aplicação em si, mas que contribuíram para o desenvolvimento e a produtividade.

O *Docker* foi adotado para rodar o banco de dados PostgreSQL localmente por meio de um *container*, eliminando a necessidade de instalação do banco na máquina. O *Beekeeper Studio* foi utilizado como interface visual para consulta e manipulação dos dados armazenados no banco. O *Insomnia* foi empregado para o teste manual dos *endpoints* REST do *backend*, permitindo verificar o comportamento das rotas antes da integração com o *frontend*.

Finalmente, o Git foi adotado para realizar o versionamento do código ao longo do desenvolvimento da aplicação, organizado em repositórios separados para o *frontend* e o *backend*, enquanto o GitHub foi a solução para realizar a hospedagem dos repositórios.

4.5 Implementação do backend

Esta seção descreve os componentes implementados no *backend* e as decisões técnicas tomadas ao longo do desenvolvimento dessa aplicação.

4.5.1 Entidades

As entidades da aplicação são compostas por quatro entidades principais: *Organization*, *User*, *Game* e *GamePlayer*. Todas fazem a extensão da *BaseEntity*, que centraliza os campos comuns *id*, *createdAt* e *updatedAt*, sendo o identificador um UUID e as datas de criação e atualização preenchidos automaticamente pelo banco de dados.

A entidade *Organization* representa uma instituição cadastrada no sistema e armazena, além do seu nome, o *Expert Model*, modelo que serve como base para a geração das simulações daquela organização. Cada organização pode ter vários usuários e jogos associados a ela.

User simboliza qualquer usuário cadastrado na plataforma e está sempre vinculado a uma organização. Além dos dados de autenticação, como email e senha armazenada como *hash*, guarda o perfil do usuário por meio do enum *UserRole*: *ADMIN*, *TEACHER* ou *PLAYER*, que determina quais ações e permissões o usuário terá no sistema. Essa entidade implementa a interface *UserDetails* do Spring Security, integrando-se ao mecanismo de autenticação do *framework*.

Já a entidade *Game* representa uma partida, e armazena suas configurações: nome, número máximo de jogadores, duração em passos, intervalo entre passos e o status atual, controlado pelo enum *GameStatus* com os valores *WAITING*, *RUNNING* e *FINISHED*. Também armazena em formato JSONB os aliases das propriedades do modelo, permitindo que sejam definidos nomes mais amigáveis para as variáveis exibidas aos alunos durante a partida, e uma referência ao *GamePlayer* vencedor da partida, definida ao final da simulação.

Por fim, *GamePlayer* representa a participação de um usuário em um jogo específico, armazenando o nome da empresa escolhida pelo aluno e a pontuação final obtida ao término da partida.

A Figura 4.2 apresenta o diagrama de classes com as entidades da aplicação e seus relacionamentos.

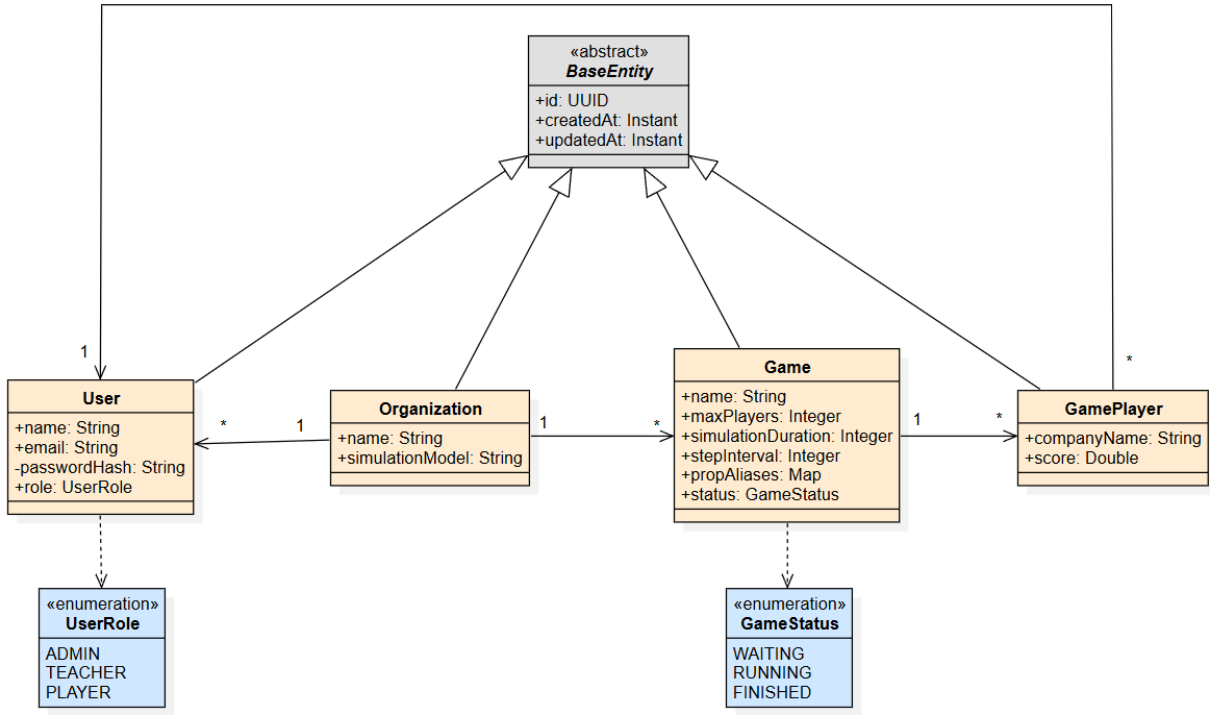

Figura 4.2: Diagrama de classes das entidades da aplicação. Fonte: elaborado pelo autor

4.5.2 Autenticação

A autenticação da aplicação é baseada em JWT. Ao realizar o login, o usuário envia suas credenciais e recebe um *token* assinado, que deve ser incluído no cabeçalho de todas as requisições. O *token* possui três informações: o email do usuário (*sub*), seu perfil (*role*) e o identificador da organização à qual pertence (*orgId*).

O processamento do *token* é feito pelo *JwtAuthFilter*, um filtro que intercepta todas as requisições antes de chegarem aos *controllers*. Ele valida o *token*, extrai as informações e registra o usuário no contexto de segurança do *Spring Security*. Rotas públicas como a de *login*, por exemplo, são liberadas sem autenticação.

O controle de acesso para a aplicação é baseado no perfil do usuário. Administradores têm acesso total à plataforma, incluindo o gerenciamento de todas as organizações cadastradas. Professores podem gerenciar usuários e jogos da própria organização, mas não têm acesso a outras organizações. Já os alunos podem apenas entrar em jogos disponíveis e participar das partidas.

4.5.3 Simulação em tempo real

A simulação em tempo real envolve a integração entre as estruturas desenvolvidas neste trabalho e a SimDS (BARBOSA; KNOP, 2009; BARBOSA; BONIFáCIO, 2017). Enquanto a SimDS fornece o motor da simulação e a infraestrutura de compilação e execução dos modelos, este trabalho implementa a camada de integração entre a aplicação web e esse motor, realizando a geração dinâmica do *User Model*, a execução da simulação por jogo, a transmissão dos resultados via *WebSocket* e o recebimento das decisões dos alunos.

Expert Model* e *User Model

O *Expert Model* da organização é armazenado no campo *simulationModel* da entidade *Organization*, conforme descrito na Seção 4.5.1. Quando uma partida é iniciada, o *backend* utiliza esse modelo para gerar dinamicamente o *User Model*, instanciando uma empresa para cada jogador participante e uma instância global compartilhada entre todos.

Essa geração é realizada pelo método *generateUM()*, que utiliza a classe *SimulationUtils* da SimDS para extrair os valores padrão de cada classe e as relações definidas no modelo. Com base nesses dados, o método constrói o *User Model* em texto, criando as instâncias e configurando suas propriedades e relações de forma automática, dispensando uma configuração manual do modelo a cada nova partida, tornando o processo independente do número de jogadores. Para identificar a qual jogador cada instância pertence durante a execução, é mantido um mapeamento chamado *playerInstMap*, que associa o UUID de cada *GamePlayer* ao nome de sua instância no modelo (*emp1*, *emp2* e assim por diante). Esse mapeamento é utilizado durante a simulação para relacionar os dados de cada empresa ao jogador correto.

Execução da simulação

O *GameService* é responsável por gerenciar o inicío da partida: gera o *User Model* via *generateUM()* e repassa o *Expert Model*, o *User Model* e o *playerInstMap* para o *GameWebSocketHandler*, que instancia um *GameSimulation* para o jogo e o executa em uma thread dedicada. Essa abordagem permite que múltiplas partidas ocorram simultaneamente sem que uma bloqueie a execução das outras.

O *GameSimulation* é a classe responsável por gerenciar a execução da simulação. Ela instancia o *Simulador* da SimDS, passando o *Expert Model* e o *User Model* como entrada, delegando a ele a execução dos passos. A cada passo, o *Simulador* chama o método *execute()* do *GameOutputChart*, classe implementada neste trabalho que estende *SimulationOutput* da SimDS, conectando a simulação à camada de comunicação do sistema. O *GameOutputChart* itera sobre todas as instâncias do modelo a cada passo. Para instâncias de empresa, utiliza o *playerInstMap* para identificar o jogador correspondente e envia os dados exclusivamente a ele. Para a instância global, transmite os dados para todos os participantes. Ao final da simulação, o método *finaliza()* é chamado pela SimDS, momento em que o *GameOutputChart* notifica todos os participantes do encerramento e aciona o *GameService* para registrar a pontuação final de cada jogador e determinar o vencedor.

Comunicação via *WebSocket*

A comunicação em tempo real entre o servidor e os participantes da partida é gerenciada pelo *GameWebSocketHandler*, que mantém o mapeamento entre as sessões *WebSocket* ativas e os jogadores conectados. Quando um jogador acessa a página da partida, uma conexão *WebSocket* é estabelecida, associando a sessão ao identificador do jogador e do jogo correspondente. Durante a simulação, o *GameOutputChart* utiliza o *GameWebSocketHandler* para transmitir os dados gerados a cada passo. Os dados da empresa são enviados exclusivamente ao jogador correspondente via *sendToPlayer()*, enquanto mensagens com dados globais e de placar são transmitidas a todos os participantes via *broadcast()*. Ao encerrar a simulação, uma mensagem de finalização é enviada a todos, sinalizando ao frontend que a partida terminou.

O *WebSocket* também é utilizado no sentido inverso: durante a partida, os alunos podem alterar os parâmetros de suas empresas pela interface, e essas alterações são enviadas ao servidor por meio desse protocolo. O *GameWebSocketHandler* recebe a mensagem, identifica o jogador e repassa a atualização ao *GameSimulation*, que aplica o novo valor diretamente na instância correspondente no modelo, respeitando os limites mínimo e máximo definidos no *Expert Model*.

4.6 Implementação do frontend

Esta seção descreve a organização e os principais componentes implementados no *frontend*, cobrindo desde o fluxo de autenticação até a interface da partida.

4.6.1 Autenticação e controle de rotas

O fluxo de autenticação do *frontend* é centralizado no *AuthContext*, um componente React que disponibiliza um contexto para toda a aplicação com as informações do usuário autenticado: o token JWT, o email, o perfil e o identificador da organização. Esse contexto também expõe as funções de *login* e *logout*, sendo o ponto central de controle da sessão do usuário.

Além disso, o controle de acesso às rotas é feito por meio de dois componentes: o *PrivateRoute*, que redireciona para a tela de login caso o usuário não esteja autenticado, e o *RoleRoute*, que define o permissionamento do usuário a determinadas rotas de acordo com seu perfil. A rota */admin*, por exemplo, é acessível apenas por administradores, enquanto a rota */manage* é acessível por professores e administradores. Alunos autenticados têm acesso apenas ao *dashboard* e à página da partida.

Figura 4.3: Login da aplicação. Fonte: elaborado pelo autor.

4.6.2 *Dashboard*

O *dashboard* é a página inicial da aplicação após o *login* e agrupa as principais informações e ações disponíveis ao usuário.

A barra de navegação superior exibe o nome da aplicação e, à direita, um conjunto de ações que varia de acordo com o perfil do usuário autenticado. Professores visualizam o botão de acesso ao painel de gerenciamento da organização, um ícone de configurações e a opção de criar um novo jogo. Administradores visualizam, além das opções do professor, um ícone que redireciona ao painel administrativo, onde é possível gerenciar todas as organizações cadastradas no sistema. Alunos visualizam apenas a opção de sair da aplicação, sem acesso às funcionalidades de gerenciamento.

Abaixo, há três *cards* que apresentam um resumo da participação do usuário naquela organização: o número de jogos disputados, a posição no *ranking* geral da organização e a melhor pontuação já registrada pelo usuário. Esses indicadores oferecem uma visão rápida do desempenho do usuário sem a necessidade de navegar para outras páginas.

Após os *cards*, a página é dividida em dois blocos. O bloco principal, à esquerda, exibe a tabela de jogos do usuário, com informações sobre o nome da empresa utilizada em cada partida, o *status* do jogo, a pontuação obtida e a data de criação. À direita, um painel lateral exibe o *ranking* de vencedores da organização, listando os usuários com maior número de vitórias.

Por último, na parte inferior da página, uma seção exibe os jogos disponíveis para entrada no momento. Cada *card* de jogo exibe seu nome, o *status* de disponibilidade, uma barra de progresso indicando a ocupação atual em relação ao número máximo de jogadores e um botão para entrar na partida.

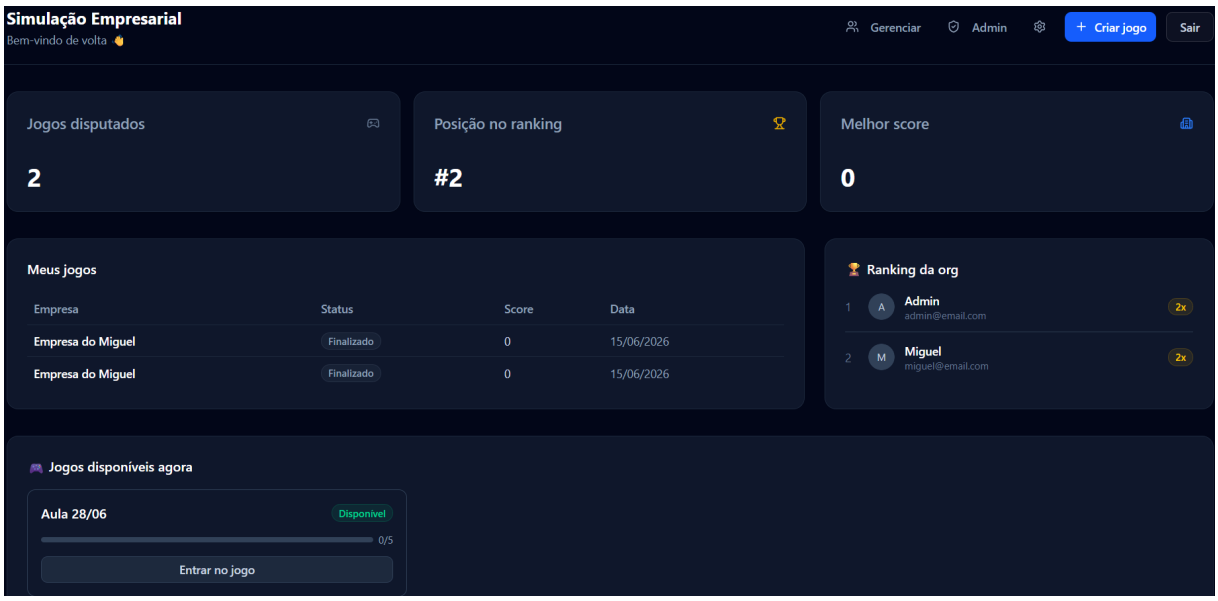

Figura 4.4: *Dashboard* da aplicação. Fonte: elaborado pelo autor.

Para criar uma nova partida, professores e administradores podem acessar o modal de criação de jogo diretamente pelo *dashboard*. Nele, é possível configurar o nome do jogo, o número máximo de jogadores, a duração da simulação em passos e o intervalo de tempo entre eles. Também é possível definir *aliases* para as variáveis do modelo, permitindo que os nomes técnicos das propriedades sejam substituídos por termos mais compreensíveis para os alunos durante a partida.

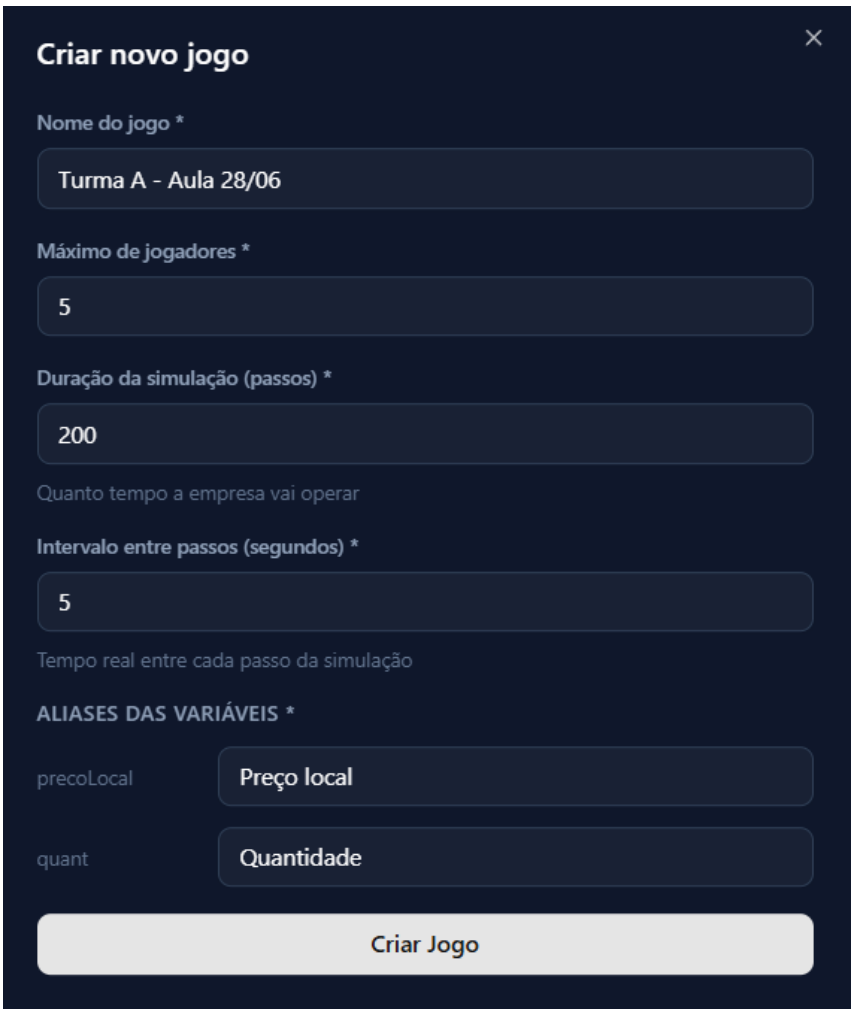

Figura 4.5: Modal de criação de jogo. Fonte: elaborado pelo autor.

4.6.3 Página da partida

A página da partida é a interface mais densa da aplicação e concentra todas as informações e ações disponíveis durante uma simulação em andamento. O cabeçalho exibe o nome do jogo, a empresa do aluno, o passo atual em relação ao total de passos da simulação, o tempo restante, o tempo total decorrido desde a entrada na página e um indicador do estado da conexão *WebSocket*. Para professores e administradores, enquanto o jogo está com status *WAITING*, um botão adicional permite iniciar a partida.

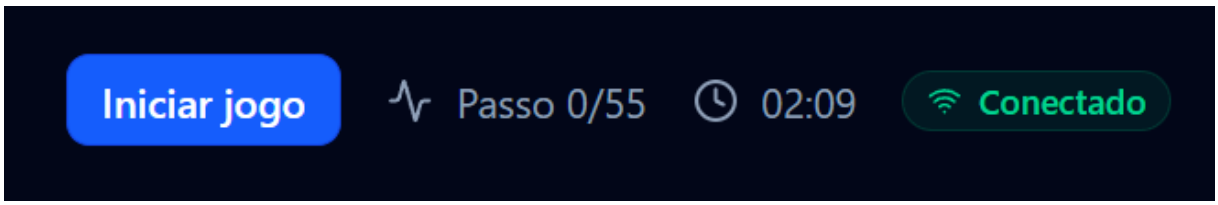

Figura 4.6: Seção direita do cabeçalho da página da partida, exibindo informações da simulação. Fonte: elaborado pelo autor.

A área principal da página é dividida em dois blocos. O primeiro exibe um gráfico de linhas com a evolução de todas as variáveis do modelo ao longo dos passos da simulação, atualizado em tempo real conforme os dados chegam via *WebSocket*. As variáveis exibidas no gráfico não são fixas: são determinadas dinamicamente a partir da primeira mensagem recebida do servidor, permitindo que o gráfico se adapte automaticamente a qualquer modelo configurado pelo professor. O segundo bloco apresenta os campos de variáveis ajustáveis da empresa do aluno, com seus respectivos aliases, valores mínimo e máximo definidos no *Expert Model*, e um botão para aplicar as mudanças, que envia os novos valores ao servidor via *WebSocket*. Antes do envio, os valores informados são limitados aos intervalos definidos no modelo, tanto ao sair do campo quanto no momento da submissão, evitando que o aluno insira valores fora dos limites configurados pelo professor.

Figura 4.7: Gráfico de desempenho e variáveis ajustáveis da empresa. Fonte: elaborado pelo autor.

À direita da página, um painel lateral exibe o placar da partida em tempo real, listando todos os participantes ordenados pela pontuação atual. O placar é atualizado a cada passo da simulação, permitindo que os alunos acompanhem sua posição em relação aos demais competidores durante toda a partida.

Figura 4.8: Placar da partida em tempo real. Fonte: elaborado pelo autor.

Ao final da simulação, a tela da partida é substituída pela tela de encerramento,

exibida assim que o *frontend* recebe a mensagem de finalização via *WebSocket*. A tela apresenta a classificação final de todos os participantes, destacando a posição do próprio aluno e do vencedor da partida. Caso o aluno autenticado seja o vencedor, é exibida uma animação de confete, implementada com a biblioteca *canvas-confetti*, reforçando o caráter competitivo e o *feedback* positivo ao final do jogo.

4.6.4 Painéis de gerenciamento

A página de gerenciamento da organização, acessível por professores e administradores, centraliza a administração dos usuários vinculados à própria organização. A página exibe uma tabela com os usuários cadastrados, contendo nome, email e perfil de cada um, além de ações de edição e remoção disponíveis ao lado de cada linha. No canto superior direito, o botão *Novo usuário* abre um modal de cadastro, no qual é possível definir o nome, o email, a senha e o perfil do novo usuário: *ADMIN*, *TEACHER* ou *PLAYER*.

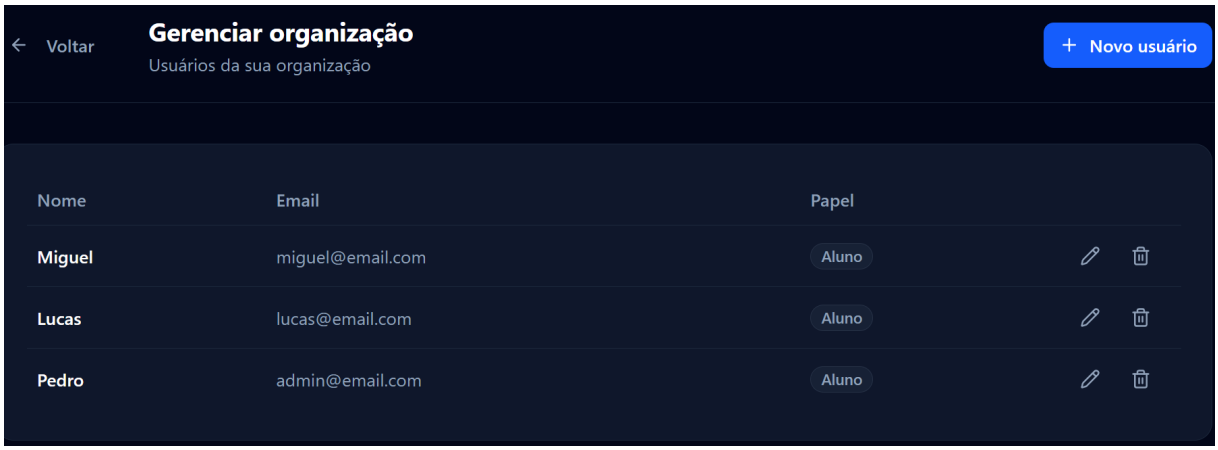

Figura 4.9: Gerenciamento de usuários da organização. Fonte: elaborado pelo autor.

Já o *Painel Admin*, acessível apenas por administradores, oferece uma visão geral de todas as organizações cadastradas no sistema, exibidas em *cards* com o nome de cada instituição. A partir de cada *card*, é possível acessar a página de detalhes daquela organização, com a mesma tabela de usuários descrita anteriormente, permitindo que o administrador gerencie os usuários de qualquer organização da plataforma, e não apenas da sua própria.

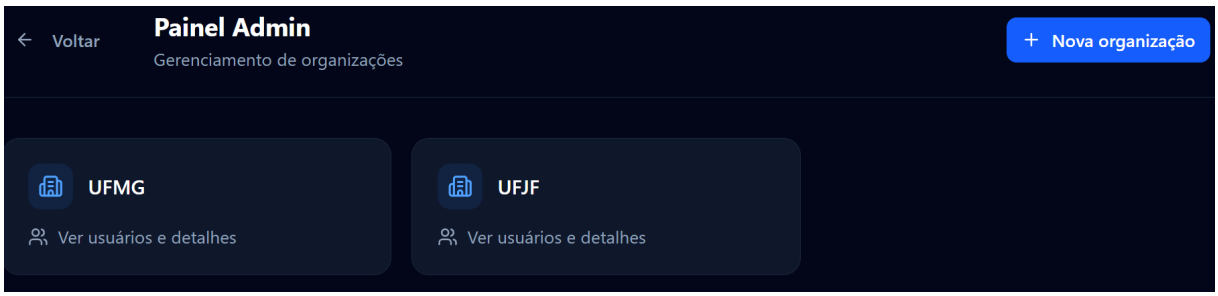

Figura 4.10: Painel administrativo com a lista de organizações. Fonte: elaborado pelo autor.

No canto superior direito do *Painel Admin*, o botão *Nova organização* abre um modal de cadastro, no qual é possível definir o nome da nova organização e o *Expert Model* que servirá de base para as simulações criadas dentro dela. O modelo é informado em um campo de texto, no formato textual definido pela SimDS, e pode ser ajustado posteriormente conforme as necessidades pedagógicas da organização ao longo do tempo.

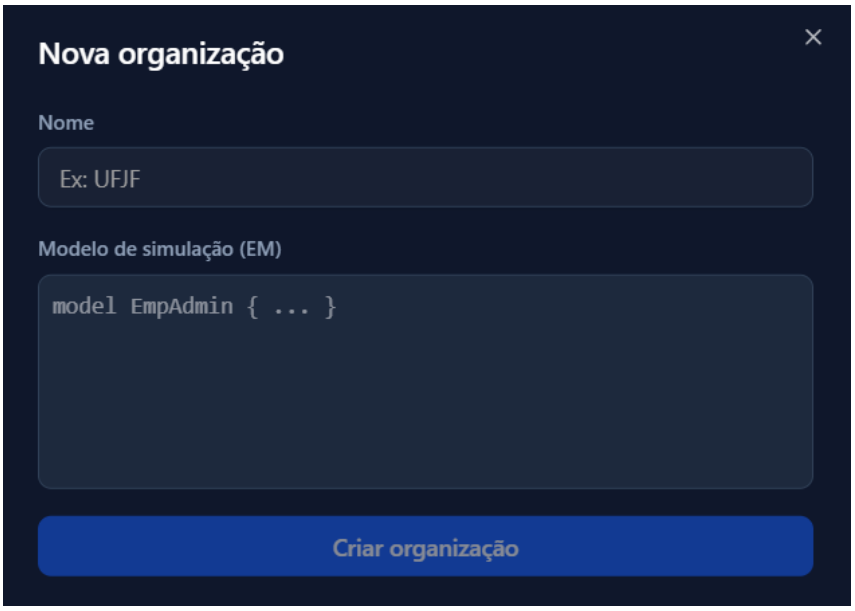

Figura 4.11: Modal de cadastro de organização, com a definição do *Expert Model*. Fonte: elaborado pelo autor.

4.7 Implantação

A aplicação foi disponibilizada em nuvem, com o *frontend* e o *backend* em serviços distintos. Essa separação reflete a própria arquitetura do sistema, onde cada aplicação pode ser atualizada e escalada de forma independente.

O *frontend* foi publicado na Vercel, plataforma especializada em hospedagem de aplicações *frontend*. A integração com o repositório no GitHub permite que qualquer atualização enviada para a *branch* principal realize automaticamente um novo *build* e publicação da aplicação, sem necessidade de intervenção manual. A URL de acesso ao frontend é ⟨tcc-front-olive.vercel.app⟩.

O *backend* foi implantado no Render por meio de um *container Docker*, construído a partir de um *Dockerfile* com *build multi-stage*. Essa abordagem separa a etapa de compilação do projeto da etapa de execução, gerando uma imagem final reduzida que contém apenas o artefato necessário para rodar a aplicação. O *backend* está acessível em ⟨simul-back.onrender.com⟩. Já para o banco de dados, o PostgreSQL também foi configurado no *Render*, integrado ao *backend* por meio de uma variável de ambiente contendo a URL de conexão. Localmente, durante o desenvolvimento, o banco foi executado via *Docker Compose*, com uma instância PostgreSQL 15 configurada em *container*.

As demais configurações sensíveis da aplicação, como a chave de assinatura dos tokens JWT e a URL do *frontend* utilizada nas configurações de CORS, foram definidas como variáveis de ambiente no *Render*, mantendo essas informações fora do código-fonte e evitando exposição acidental em repositórios públicos.

A Figura 4.7 ilustra a aplicação em execução real, com uma partida de exemplo criada para uma turma, utilizando uma variação do modelo apresentado na Seção 2.5, com a adição de limites mínimo e máximo às propriedades ajustáveis. No passo 28 de 100, com preço local de R$125,00 e 25 funcionários configurados, o gráfico evidencia o crescimento consistente do caixa e do *score* da empresa ao longo da simulação, confirmando o funcionamento correto da integração entre a SimDS, o backend e o frontend em um ambiente implantado em produção.

5 Conclusão

Este capítulo retoma o objetivo do trabalho, sintetiza o que foi desenvolvido, discute as limitações identificadas e as possibilidades de trabalhos futuros, encerrando com as considerações finais sobre o projeto.

5.1 Objetivo

Este trabalho teve como objetivo desenvolver uma aplicação *web* que oferecesse uma alternativa ao ensino de conceitos de gestão empresarial, integrando um modelo de simulação fundamentado na Dinâmica de Sistemas a uma plataforma configurável, na qual os alunos pudessem experimentar e tomar decisões sobre empresas fictícias em tempo real. A proposta originou-se da limitação identificada nos jogos empresariais analisados no Capítulo 3, que se restringem a regras simplificadas sem um modelo matemático real e não oferecem suporte a múltiplos jogadores, reduzindo o potencial competitivo entre os participantes.

Diferente dos jogos analisados, o modelo de simulação da plataforma desenvolvida não é fixo: o professor pode definir seu próprio *Expert Model*, conforme descrito na Seção 4.5.1, adaptando o cenário da partida ao contexto pedagógico desejado. Em razão do prazo estabelecido para o desenvolvimento deste trabalho, apenas um modelo foi construído e testado como demonstração dessa capacidade, mas a arquitetura da plataforma permite a criação de outros cenários sem necessidade de alterações no código.

5.2 Síntese do trabalho

Dos objetivos específicos definidos na introdução, os relacionados à disponibilização da plataforma *web*, à fundamentação das simulações em modelos de Dinâmica de Sistemas por meio da integração com a SimDS, à adaptação do cenário de simulação pelo professor e ao acompanhamento, pelos alunos, do impacto de suas decisões em tempo real foram plenamente atendidos, conforme detalhado no Capítulo 4. Os recursos visuais para análise

de desempenho também foram implementados, por meio dos gráficos e do placar em tempo real disponíveis na página da partida. Já os elementos competitivos previstos no último objetivo específico foram implementados tecnicamente, com o sistema de pontuação e a definição de um vencedor ao final da partida, mas seu efeito sobre o engajamento dos alunos não foi avaliado neste trabalho, uma vez que o escopo do projeto se concentrou no desenvolvimento da plataforma, sem aplicação em um contexto real de sala de aula.

A SimDS foi integrada ao backend da aplicação, responsável por compilar e executar os modelos de simulação definidos pelos professores, enquanto a camada de integração desenvolvida neste trabalho ficou responsável pela geração dinâmica do *User Model*, pela execução da simulação por jogo e pela transmissão dos resultados aos participantes.

A comunicação entre servidor e clientes foi implementada via *WebSocket*, permitindo que os alunos acompanhem a evolução da simulação a cada passo sem necessidade de atualização manual, conforme ilustrado na Figura 4.7. Essa comunicação também ocorre no sentido inverso: os alunos podem ajustar os parâmetros de suas empresas durante a partida, enviando os novos valores ao servidor pelo mesmo protocolo, conforme descrito na Seção 4.5.3.

A plataforma desenvolvida conta ainda com painéis de gerenciamento para professores e administradores, apresentados na Seção 4.6.4, permitindo a configuração de organizações, modelos de simulação e usuários. Por fim, a aplicação foi implantada em produção, com o *frontend* hospedado na Vercel e o *backend* em *container Docker* no *Render*, tornando-se acessível via navegador sem necessidade de instalação local.

5.3 Limitações

A principal limitação do sistema está relacionada à forma como os dados da simulação são armazenados. Os passos intermediários de cada partida não são persistidos no banco de dados: o histórico exibido no gráfico da página da partida é acumulado apenas no estado do *frontend*, à medida que os dados chegam via *WebSocket*, sendo perdido caso o aluno atualize a página ou a conexão seja interrompida. Ao final da simulação, apenas o resultado final de cada jogador é persistido no banco de dados. Essa decisão evita o acúmulo desproporcional de dados, considerando o volume gerado pela combinação de

múltiplos jogadores, múltiplos passos e múltiplos jogos, mas tem como consequência direta a perda do histórico visual em caso de reconexão durante a partida. Pelo mesmo motivo, também não é possível acompanhar o histórico detalhado de uma simulação já finalizada, restando apenas a pontuação final registrada.

Outra limitação é a ausência de testes automatizados no sistema. A verificação do funcionamento das funcionalidades implementadas, incluindo a integração com a SimDS e a comunicação via *WebSocket*, foi feita inteiramente por meio de testes manuais ao longo do desenvolvimento, o que aumenta o risco de regressões não detectadas em alterações futuras.

O modelo de simulação utilizado durante os testes do sistema, descrito no Capítulo 4, é funcional e demonstra bem o funcionamento da plataforma, mas tem caráter demonstrativo. Ele não foi aplicado em um contexto real de sala de aula, então o impacto pedagógico da ferramenta ainda não foi avaliado na prática.

Por fim, a modelagem de dados vincula um único *Expert Model* a cada organização, o que reflete uma decisão de simplificação do escopo do projeto. Professores de uma mesma instituição que desejem utilizar cenários diferentes precisam alternar manualmente o modelo configurado entre uma partida e outra.

5.4 Trabalhos futuros

Como continuidade deste trabalho, sugere-se a aplicação da plataforma em um contexto real de sala de aula, com turmas de Administração ou Sistemas de Informação, permitindo avaliar seu impacto pedagógico de forma prática e coletar *feedback* direto de professores e alunos.

Outra possibilidade é o desenvolvimento de uma versão com interface responsiva otimizada para dispositivos móveis, já que a aplicação atual, embora acessível via navegador em qualquer dispositivo, foi projetada e testada apenas para uso em telas *desktop*. Um aplicativo nativo também poderia oferecer notificações sobre eventos da partida, como seu início pelo professor ou o encerramento da simulação, dispensando a necessidade de o aluno manter a aba aberta durante toda a espera.

Sugere-se também a implementação de um mecanismo de persistência dos passos

intermediários da simulação, possibilitando a reconstrução do histórico de uma partida já finalizada e abrindo espaço para análises mais detalhadas do desempenho dos jogadores ao longo do tempo.

Propõe-se também a criação de um *marketplace* de modelos de simulação, permitindo que diferentes organizações compartilhem entre si os *Expert Models* configurados. Essa funcionalidade reduziria a necessidade de cada professor construir seu próprio modelo do zero, possibilitando reaproveitar e adaptar cenários já validados por outras instituições.

5.5 Considerações finais

O desenvolvimento deste trabalho permitiu unir os conceitos de Dinâmica de Sistemas a uma implementação prática, integrando a SimDS a uma plataforma *web* acessível e voltada ao ensino de gestão empresarial. Ao longo do projeto, decisões técnicas como a comunicação em tempo real via *WebSocket*, a geração dinâmica do *User Model* e a separação entre *frontend* e *backend* em aplicações independentes se mostraram adequadas para sustentar múltiplos jogadores competindo simultaneamente, conforme demonstrado na Seção 4.7.

Em relação ao trabalho de Barbosa e Bonifácio (2017), discutido na Seção 3.3, este projeto representa uma nova implementação da mesma proposta conceitual, atualizando a arquitetura original com comunicação em tempo real e implantação em nuvem acessível por qualquer navegador. As diferenças técnicas entre as duas propostas mostram como a mesma ideia original pode ser reimplementada com ferramentas mais modernas, sem alterar sua essência conceitual.

Bibliografia

BARBOSA, C. B.; KNOP, I. Tool supported meta-modeling and code generation for the dynamic systems domain. Trabalho não publicado. 2009.

BARBOSA, C. d. B.; BONIFáCIO, A. L. Model-based environment for learning simulations. In: *Anais do XXXVII International Sodebras Congress*. [S.l.: s.n.], 2017.

BASTOS, A. A. P. *A dinâmica de sistemas e a compreensão de estruturas de negócios*. Dissertação (Mestrado) — Universidade de São Paulo, Faculdade de Economia, Administração e Contabilidade, 2003.

BUCKLEY, P.; DOYLE, E. Gamification and student motivation. *Interactive Learning Environments*, 2014.

CHIAVENATO, I. *Introdução à Teoria Geral da Administração*. 7. ed. Rio de Janeiro: Elsevier, 2003.

DIAS, P. Comunidades de educação e inovação na sociedade digital. *Educação, Formação & Tecnologias*, 2012.

FADEL, L. M. et al. (Ed.). *Gamificação na educação*. São Paulo: Pimenta Cultural, 2014.

MEROTO, M. B. d. N. et al. Jogando para aprender: como a gamificação está mudando a educação. *Revista Foco*, 2024.

PRETTO, F.; FILARDI, F.; PRETTO, C. Jogos de empresas: uma estratégia de motivação no processo de ensino e aprendizagem na teoria das organizações. *Estratégia e Negócios*, 2010.

SANTOS, M. S. et al. A teoria dos jogos empresariais como estratégia de ensino-aprendizagem nos cursos de administração de empresas. *Revista Práxis*, 2014.

SAUAIA, A. C. A. Cases and business games: The perfect match! *Developments in Business Simulation and Experiential Learning*, 2006.