DIMITRY ALVES MENDES
GUILHERME DA SILVA LIMA
GUSTAVO SOARES OLIVEIRA
JOÃO FELIPE MATOS DE PINHO
LUCAS DOS SANTOS GOMES
MATHEUS BRAZ DE SIQUEIRA
NICOLAS FABRÍCIO LIMA
VINÍCIUS FERNANDES TAVARES BITTENCOURT
VINICIUS TAVARES DA SILVA

ABC VIRTUAL

SÃO PAULO – SP

2024

ETEC ALBERT EINSTEIN

DIMITRY ALVES MENDES
GUILHERME DA SILVA LIMA
GUSTAVO SOARES OLIVEIRA
JOÃO FELIPE MATOS DE PINHO
LUCAS DOS SANTOS GOMES
MATHEUS BRAZ DE SIQUEIRA
NICOLAS FABRÍCIO LIMA
VINÍCIUS FERNANDES TAVARES BITTENCOURT
VINICIUS TAVARES DA SILVA

ABC VIRTUAL

Trabalho de Conclusão de Curso apresentado do curso de Desenvolvimento de Sistemas da Etec Albert Einstein, como requisito parcial para o título de Técnico em Desenvolvimento de Sistemas sob orientação da Prof. Cristiane Tuji da Silva

SÃO PAULO – SP
2024

RESUMO

O texto a seguir aborda a importância da educação, especialmente nas fases iniciais da vida de uma pessoa, destacando que ela é fundamental para o desenvolvimento individual de cada um e para uma vida em conjunto no mundo mais justa. Apesar de tal importância, a educação brasileira ainda está longe da qualidade ideal para sua população, dificultando o desenvolvimento cognitivo do próximo. Diante deste contexto, surge a ideação e o desenvolvimento de uma plataforma digital com o objetivo de auxiliar profissionais da educação, responsáveis e os próprios alunos no processo de alfabetização infantil, com a utilização de métodos inovadores e tecnologias inovadoras.

Palavras-chave: Educação, alfabetização, desenvolvimento, inovação, importância.

ABSTRACT

The following text addresses the importance of education, especially in the early stages of a person's life, highlighting that it is fundamental for each person's individual development and for a fairer life together in the world. Despite such importance, Brazilian education is still far from the ideal quality for its population, hindering the cognitive development of others. Given this context, the idea and the development of a digital platform emerged with the aim of helping education professionals, guardians and students themselves in the process of children's literacy, using innovative methods and innovative technologies.

Keywords: Education, literacy, development, innovation, importance.

SUMÁRIO

ABSTRACT 6

SUMÁRIO 7

1. INTRODUÇÃO 10

2. EDUCAÇÃO INFANTIL 12

3. PROBLEMATIZAÇÃO 13

4. INTERNET E EDUCAÇÃO 14

5. GAMIFICAÇÃO NA APRENDIZAGEM 16

6. PROPOSTA DE VALOR 17

7. METODOLOGIA 18

8. GERENCIAMENTO 19

9. DESENVOLVIMENTO 20

9.1. Diagrama Banco de Dados 20

9.2. Endpoints Backend 22

9.2.1. Authenticação 22

9.2.2. Estudantes 22

9.2.3. Temporadas 22

9.2.4. Fases 22

9.2.5. Responsáveis 24

9.2.6. Produtos 24

9.2.7. Assinatura 24
9.3. Desenvolvimento do Site 25
9.3.1. Tela de Cadastro 25
9.3.2. Tela de Login 26
9.3.3. Tela Inicial do Site 27
9.3.4. Sessão de Aprendizagem 28
9.3.5. Sessão de Realidade Virtual 29
9.3.6. Perfil do Aluno 30
9.3.7. Loja Virtual 31
9.4. Jogos Realidade Virtual 32
10. RESULTADOS 33
11. CONSIDERAÇÕES FINAIS 34
REFERÊNCIAS 35
APÊNDICES 38
APÊNDICE A - BACKEND 38
Arquivos Controllers 38
Authentication.JS 38
Levels.JS 40
Payments.JS 41
Products.JS 42
Responsibles.JS 43

Seasons.JS 44
Students.JS 45
Validate.JS 46
Arquivos Database 47
Connection.JS 47
Arquivos Middlewares 47
Authentication.JS 47
Arquivos Models 47
Authentication.JS 48
Levels.JS 49
Products.JS 51
Responsibles.JS 52
Seasons.JS 54
Students.JS 56
Arquivos Routes 57
Authentication.JS 57
Levels.JS 58
Payments.JS 58
Products.JS 58
Responsibles.JS 59
Seasons.JS 59

Studensts.JS 59
Arquivos Schemas 60
Levels.JS 60
Products.JS 61
Responsibles.JS 62
Seasons.JS 63
Students.JS 64
App.JS 64
Server.JS 65
APÊNDICE B - REALIDADE VIRTUAL 66
Index.HTML 66
Arquivos Pages 66
Classroom.HTML 66
Arquivos Scripts 67
Arquivos Components 68
Camera.JS 72
Letters-placement.JS 73
Sign-in.JS 74
Arquivos Styles 74
Index.CSS 75
Reset.CSS 80

ÍNDICE DE FIGURAS

Figura 1:Diagrama Banco de Dados 20

Figura 2: Tela de Cadastro 23

Figura 3: Tela de Login 24

Figura 4: Tela Inicial do Site 25

Figura 5: Sessão de Aprendizagem 26

Figura 6: Sessão de Realidade Virtual 27

Figura 7: Perfil do Aluno 28

Figura 8: Loja Virtual 29

Figura 9: Sala de Aula em Realidade Virtual 30

1. INTRODUÇÃO

A educação é uma ferramenta essencial para a formação de um cidadão e para a sociedade. Ela é responsável por disseminar e multiplicar os conhecimentos, e pelo desenvolvimento de habilidades úteis em prol da vivência do indivíduo em sua comunidade. Segundo (PAULO FREIRE, 1979), a educação, como apenas uma ferramenta, não tem a capacidade de mudar o mundo. Agora, cidadãos que utilizam de tal ferramenta são capazes de fazerem uma sociedade mais justa e beneficiária a todos.

Para sermos capazes de evoluir educacionalmente, as instituições de ensino detêm um papel fundamental no processo. Não só elas, as pessoas também necessitam se dispor para praticar os estudos de forma independente. Contudo, essa disciplina não surge do nada. As escolas precisam fazer a diferença para o estudante, disponibilizando uma boa estrutura e professores capacitados e dispostos a despertar o interesse pelo saber do aluno. Além de que, a escola é onde as pessoas possuem o primeiro contato em conjunto, aprendendo a socializar e conviver bem com o próximo.

Entretanto, esse tratamento nem sempre é proporcionado para o aluno, visto que, até mesmo para os profissionais da área ainda há falta de recursos. Diante do exposto, os estudantes, ao decorrer dos anos, possuirão uma defasagem de ensino, podendo acarretar problemas futuros. Este quadro se complica mais, se analisado em visão de crianças que acabaram de entrar nesse mundo dos estudos, pois nessa fase, é onde se encontra a alfabetização infantil, processo importante na formação de vida, em razão de promover hábitos de leitura, comunicação em grupo, escrita e construção dos conhecimentos.

Na atualidade, as ferramentas digitais podem fornecer um enorme auxílio no quesito de acompanhamento e conteúdo dos estudantes dentro e fora do ambiente escolar. Tendo em vista que, a internet dá acesso a uma extrema quantidade de informações disponíveis em todo o mundo, além de ainda poder contar com

professores no auxílio dos estudantes, pois com a internet a comodidade para executar tal ação fica facilitada.

Com isso, foi feita uma pesquisa em benefício de uma alfabetização digna para as crianças e, diante das informações auferidas, foi iniciado o desenvolvimento de uma plataforma digital que propõem auxiliar no processo de alfabetização das crianças, de forma adaptável e inclusiva, dependendo da situação que o aluno se encontra, tudo em conjunto a métodos inovadores, possibilitados graças a modernização do mundo. Surgindo uma plataforma atrativa para a criança se interessar pelo conteúdo, jogos intuitivos que possam diversificar a monótona rotina de estudos, mas ainda assim alavancar os conhecimentos do aluno. Sem contar, com a possibilidade do uso da realidade virtual em alguns conceitos de aula, realidade na qual ainda não é habitual no dia a dia das pessoas, entretanto, viabiliza uma vastidão de oportunidades de ensino, garantindo a aproximação da criança ao material de estudo.

A sociedade e a ciência estão constantemente evoluindo e nosso trabalho também faz parte disso. Uma escola virtual é um pequeno passo para um grande futuro da educação, mesmo assim tem uma grande importância para nossa sociedade pois providencia novos meios educacionais. A ciência tem sua parte nisso também e explora esses meios para visar a educação de todos e podendo assim melhorá-la mais ainda futuramente e trazer o melhor de nossos estudantes.

2. EDUCAÇÃO INFANTIL

O ensino ou a educação infantil é a primeira fase da educação básica, onde a criança é introduzida no cenário escolar, como creches e pré-escolas, é um pilar imprescindível para o seguimento nos estudos, já que serve como base para os estudos mais avançados, vindos do ensino fundamental e médio. Para comprovar essa importância, basta olhar a Constituição de 1988, onde é declarado que o ensino básico infantil, entre 0 e 6 anos, se tornou um dever do Estado, criando a obrigação do atendimento para essa faixa etária nas escolas.

Ela detém tamanha importância, porque estabelece toda a base da personalidade de uma pessoa, já que, existem diversos estudos e até ditados populares que dizem, que a primeira experiência é a que fica marcada, e isso se aplica na educação também, com experiências positivas nesse período da vida, se tende a reforçar a empatia, autoconfiança, responsabilidade, solidariedade, disciplina etc. Assim, se torna imprescindível a discussão de sua essencialidade para todo mundo.

Partindo do pressuposto de uma boa experiência, manter a criança sempre com atividades diversificadas, estimula a curiosidade do saber e a busca pelas respostas a essas curiosidades de maneira própria. Visto que, crianças nessa idade procuram sempre inúmeros estímulos para se entreterem, assim, com atividades monótonas, ou manter sempre as propostas, faz o aluno perder o interesse pelos estudos. Alterando essa repetitiva rotina, é possível despertar habilidades fundamentais para o ser humano.

Outro pilar que é imprescindível para a educação é a afetividade no ciclo social da criança, ela é vital durante toda vida do ser humano, mas se torna mais importante quando o assunto é desenvolvimento infantil. Tendo um bom apoio ao redor facilita o processo cognitivo da criança, já que são aspectos que andam em paralelo. Como Jean Piaget (1995) diz, afetividade constitui aspecto indissociável da inteligência, pois ela impulsiona o sujeito a realizar as atividades propostas. De maneira que seja alcançado melhores resultados quando se trata de seus interesses.

3. PROBLEMATIZAÇÃO

Sabendo de toda a importância que a educação infantil tem na vida de uma pessoa, imagina-se que o governo invista muitos recursos para haver uma qualidade devida a todas as pessoas. Mas se engana quem pensa de tal forma, visto que durante essa faixa etária, segundo dados do MEC, em 2022, de 10 crianças apenas 4 se encontravam alfabetizadas, número que aumentou desde pesquisas feitas antes da pandemia de covid-19. Ocorrido esse, no qual as crianças ficaram sem nenhum acompanhamento educacional, onde poderia ter sido valorizado o meio digital, já que era a única forma de comunicação durante todo o período.

Outra pesquisa que vale ressaltar, é a de que 71% dos professores brasileiros passam por estado de estresse pela sobrecarga no seu trabalho, dados auferidos pelo IPEC em 2022. Avaliando essa pesquisa, é possível concluir que a partir da sobrecarga em cima dos professores, nem todas as crianças terão o mesmo acompanhamento na escola, criando um déficit nas crianças com uma maior dificuldade no processo de aprendizagem, uma vez que, o contato entre professor e aluno é essencial para um bom desempenho do estudante, pois os educadores sabem a forma certa de incentivar e despertar o melhor do seu aluno. Assim essa defasagem gerada consequentemente amplia a chance de abandono dos estudos, em razão de ocorrer um acúmulo de conteúdos a serem aprendidos.

Além da sobrecarga nos professores, ocorre uma sobrecarga no ambiente residencial para os responsáveis, porque os temas não desenvolvidos com excelência em aula, são levados para dentro de casa. Criando um estresse desnecessário, pois os pais que já passam pela correria do trabalho, terão de ajudar no ensino dos filhos, mesmo não sabendo os meios certos para ensinar como os professores.

Com o conhecimento dos dados acima, surge a criação do ABC Virtual, com uma proposta de auxiliar os educadores, gestores de ensino, e principalmente, fornecer um apoio ao aluno diante dos materiais estudados dentro da sala de aula, a fim de erradicar o analfabetismo existente no território nacional.

4. INTERNET E EDUCAÇÃO

Com o avanço da modernidade e das tecnologias, as formas de conseguir informações ficaram facilitadas a qualquer pessoa que possua um celular ou computador e acesso a internet. Mas essa maneira mais fácil de se manter informado com as tecnologias, nem sempre gera no indivíduo o conhecimento que maneiras tradicionais gerariam, o que é um fato preocupante. Visto que, quantidade não é igual a qualidade, assim, para haver um bom aprendizado é necessário um afunilamento dos materiais disponibilizados digitalmente.

Com o conhecimento acima, profissionais da educação e responsáveis por estudantes detém uma resistência no quesito ensino de forma remota, não confiando inteiramente no que é passado aos alunos e no que eles conseguem aprender de fato, ainda mais com desinformações que diversas vezes são passadas em vários meios de comunicação. Contudo, é importante que esta barreira seja quebrada, em razão da internet ser o futuro.

Abordando a importância da internet, ela se torna um meio válido pelas diversas possibilidades que ela pode trazer as pessoas quando tratada de forma consciente e saudável. Pois, a aprendizagem on-line oferece uma ampla gama de recursos, incluindo vídeos, textos, áudios, simulações interativas entre outras coisas, e se bem escolhido todos com uma alta qualidade, que não é viável a todos quando se fala de no meio presencial. Podendo assim, atender diversos tipos de pessoa e os mais variados tipos de aprendizagem, conseguindo manter a atenção do público e o interesse pelo aprendizado.

Com a atenção e o desenvolvimento do aluno de forma online, existe outra vantagem no qual a internet se destaca e pode-se tirar proveito para atrair ainda mais quem utilizar, a personalização do aprendizado, com dados referentes ao que se é estudado por um estudante, é possível fazer pesquisas para tratar os erros de maneira mais pessoal a cada um. Desse modo, podendo auxiliar professores e educadores, já que com uma análise de perfil de cada aluno, assim o profissional da educação que

se utilizará dos meios virtuais, pode tratar conteúdos específicos e direcionados a cada estudante.

Vale ditar outra possibilidade que a internet traz e que tem efeito positivo quando se fala de crianças e um processo de alfabetização, a brincadeira. Quando a criança brinca ela consegue se expressar, interagir, aprender a lidar com o mundo que a cerca e formar sua personalidade. Desta maneira percebe-se a importância do brincar como uma forma de a criança expressar-se e desenvolver suas habilidades de criação, de relacionar-se e de interagir. Como na internet pode existir essa brincadeira, criando assim uma harmonia excelente para a criança, fora apenas daquele conhecimento de livros e aulas didáticas, surtindo um aprendizado tão bom quanto.

5. GAMIFICAÇÃO NA APRENDIZAGEM

A gamificação é definida como o processo de implementar mecânicas relacionadas a jogos em ambientes não relacionados a jogos com a intenção de incentivar o aprendizado e a realização dos seus objetivos, trazendo consigo também, além dos jogos, um sistema de recompensas. Recentemente, esta estratégia vem inovando a indústria com o tempo, influenciando tanto usuários como funcionários a fazerem suas tarefas intuitivamente, além de garantir a coleta de dados e feedback, para que as equipes possam implementar futuras melhorias em seus projetos.

Nossa equipe vê esta estratégia como nosso principal diferencial em nosso projeto, tornando a sua implementação necessária para atingirmos nosso principal objetivo, e sua implementação será a criação de jogos intuitivos, um personagem carismático e um sistema de recompensa e entretenimento para os usuários, para que eles possam se sentir incentivados a progredir e a constantemente utilizar nossa plataforma.

É importante ressaltar que os jogos virtuais trazem benefícios como estímulo ao raciocínio lógico, leitura e entretenimento as pessoas que os jogam. Podendo desfrutar também destes benefícios com a aplicação na parte educacional do projeto.

6. PROPOSTA DE VALOR

A nossa proposta de valor consiste em desenvolver uma plataforma de alfabetização inovadora e *gamificada*, que busca continuamente proporcionar o melhor ensino para os alunos. O objetivo é aumentar o interesse pelo aprendizado através do uso de jogos, Realidade Virtual (RV) e conteúdos diversificados, tornando o processo educacional mais imersivo e divertido.

Nossa abordagem não apenas visa envolver os estudantes em um ambiente de aprendizado interativo, mas também oferecer um excelente custo-benefício, acessível a pessoas de todas as classes sociais. Utilizamos a Realidade Virtual para democratizar o acesso à educação, proporcionando uma visão ampla do futuro e um ensino mais envolvente.

Queremos entregar um projeto que auxilie crianças, adultos e até idosos que não tiveram a oportunidade de aprender as noções básicas do português brasileiro. Nossa missão é promover a inclusão educacional, garantindo que todos tenham acesso a uma educação de qualidade, independentemente de suas condições sociais. Ao utilizar tecnologias avançadas e métodos de ensino inovadores, pretendemos transformar o aprendizado em uma experiência prazerosa e eficaz, que contribua significativamente para o desenvolvimento pessoal e profissional dos nossos alunos.

Levando em consideração esses aspectos, podemos afirmar que nossa proposta de valor é um compromisso com a educação inclusiva e de alta qualidade, que utiliza a tecnologia para superar barreiras e oferecer oportunidades iguais a todos.

7. METODOLOGIA

O tipo de pesquisa utilizada neste trabalho foi a exploratória, pois desta maneira é possível ter dados de emergentes clientes do projeto. Também se trouxe uma grande consideração a vivência de cada integrante que fez parte do projeto, assim foi criado um amplo material para estudo. Consequentemente, com a análise dos estudos, surge a ideia do ABC Virtual, visto que, foi enorme o retorno negativo em volta deste assunto por parte das pessoas entrevistadas e dos relatos individuais de cada integrante.

Para embasamento argumentativo deste estudo, a abordagem utilizada foi a pesquisa qualitativa, assim foram realizadas consultas bibliográficas sem artigos online, com o intuito de estruturar completamente o problema e focar na solução mais adequada. Desta forma, foi possível entender todos os problemas do tema trabalhado e as causas destes problemas, desde a falta de estrutura, até a falta de um acompanhamento por parte dos profissionais em volta do aluno.

8. GERENCIAMENTO

Para desenvolver o projeto, a equipe adotou a estrutura SCRUM para os participantes, deixando todos com cargos definidos, sendo um *Product Owner*, um *Scrum Master*, dois desenvolvedores *Front-End*, um *Desiner*, dois desenvolvedores *Back-End*, um desenvolvedor de *games* e um documentador.

Dessa forma, a equipe foi realizando reuniões semanais (weekly) para avaliar tudo que foi feito e o que será feito adiante. O Trello foi uma ferramenta muito útil para acompanhamento das tarefas de cada membro em cada parte do projeto.

9. DESENVOLVIMENTO

9.1. Diagrama Banco de Dados

O banco de dados do projeto foi feito em volta de uma tabela de alunos, onde contém todas as informações referentes ao aluno para poder ser efetuado processos de login entre as demais funcionalidades da plataforma, com dados também do seu responsável, da fase em que se encontra e da assinatura. Tabela de assinatura na qual contém as datas de aquisição e expiração dos planos de estudos.

Há uma ligação entre os alunos e os produtos no qual poderão ser comprados na loja virtual, de forma que é necessária uma tabela de intermediação para se reconhecer o produto ao aluno correto. Também é possível encontrar a tabela de fases que se liga com uma unidade de estudo específica, assim seu id vira chave estrangeira em alunos e unidades. Por fim, a tabela de realidade virtual, com toda a referência do jogo em realidade virtual, que se liga com unidades, pois cada unidade terá apenas uma fase em realidade virtual.

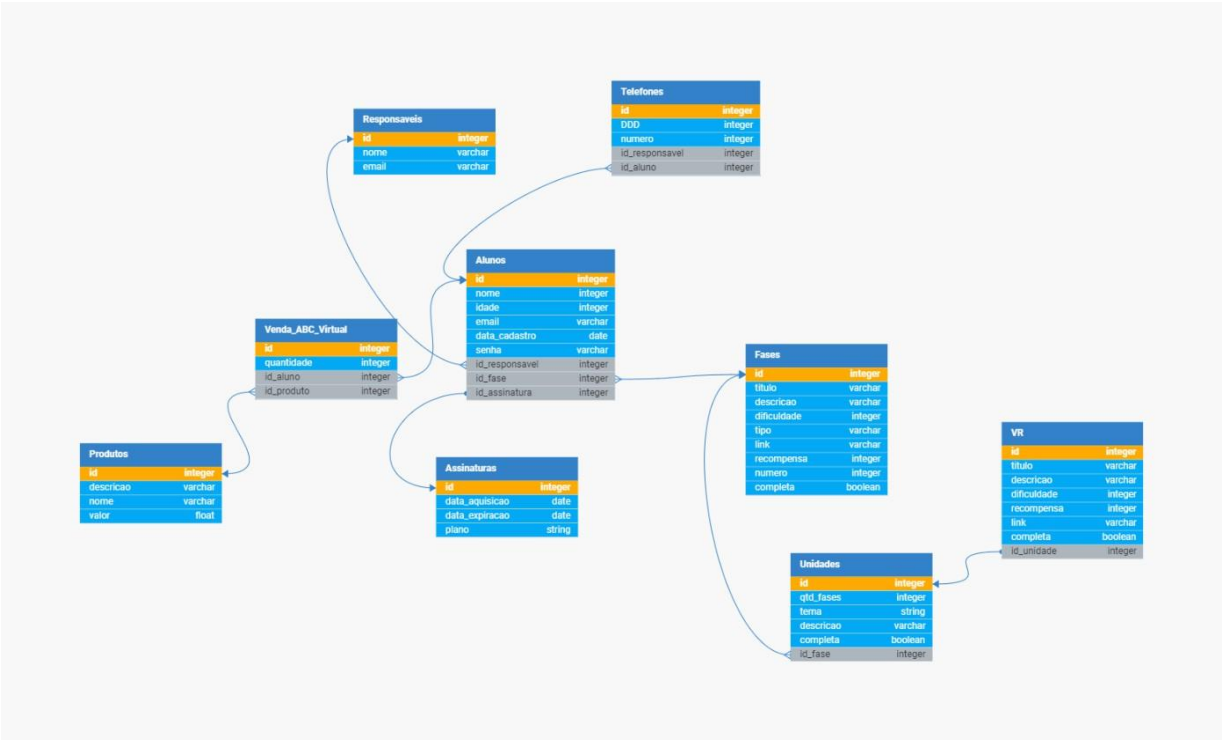


Figura 1:Diagrama Banco de Dados

9.2. Endpoints Backend

9.2.1. Authenticação

- POST /authentication/signup - Cadastrar estudante e responsável;
- POST /authentication/signin - Logar utilizando credenciais do estudante.

9.2.2. Estudantes

- GET /students - Retorna todos os estudantes;
- GET /students/:id - Retorna um estudante com ID específico;
- PUT /students/:id - Atualiza os dados de um estudante com ID específico;
- DELETE /students/:id - Deleta um estudante com ID específico.

9.2.3. Temporadas

- POST /seasons - Cria uma temporada;
- GET /seasons - Retorna todas as temporadas;
- GET /seasons/:id - Retorna uma temporada com ID específico;
- PUT /seasons/:id - Atualiza os dados de uma temporada com ID específico;
- DELETE /seasons/:id - Deleta uma temporada com ID específico.

9.2.4. Fases

- POST /levels - Cria uma fase;
- GET /levels - Retorna todas as fases;
- GET /levels/last - Retorna a última fase do estudante que está autenticado;
- GET /levels/:id - Retorna uma fase com ID específico;
- GET /levels/season/:id - Retorna todos as fases de uma temporada com ID específico;
- PUT /levels/finish/:id - Atualiza as fases finalizadas do estudante autenticado

acrescentando mais uma fase com ID específico, também acrescenta moedas ao estudante de acordo com a recompensa da fase;

- PUT /levels/:id - Atualiza os dados de uma fase com ID específico;
- DELETE /levels/:id - Deleta uma fase com ID específico.

9.2.5. Responsáveis

- GET /responsibles - Retorna todos os responsáveis;
- GET /responsibles/:id - Retorna um responsável com ID específico;
- PUT /responsibles/:id - Atualiza os dados de um responsável com ID específico;
- DELETE /responsibles/:id - Deleta um responsável com ID específico.

9.2.6. Produtos

- POST /products - Cria um produto;
- GET /products - Retorna todos os produtos;
- GET /products/:id - Retorna um produto com ID específico;
- PUT /products/buy/:id - Realiza a compra de um produto com ID específico para o estudante autenticado;
- PUT /products/:id - Atualiza os dados de um produto com ID específico;
- DELETE /products/:id - Deleta um produto com ID específico.

9.2.7. Assinatura

- POST /payment - Realiza o pagamento da assinatura utilizando Mercado Pago.

9.3. Desenvolvimento do Site

O site foi projetado pela plataforma Figma, permitindo assim uma visão ampla e imediata de todo o funcionamento e campos que ele apresenta de fato. Em todas as páginas o usuário é acompanhado pelo astronauta que é mascote do projeto.

9.3.1. Tela de Cadastro

Na tela de cadastro, há a requisição dos dados pessoais para efetuar o cadastro na plataforma, junto a opção de criar cadastro com o Google.

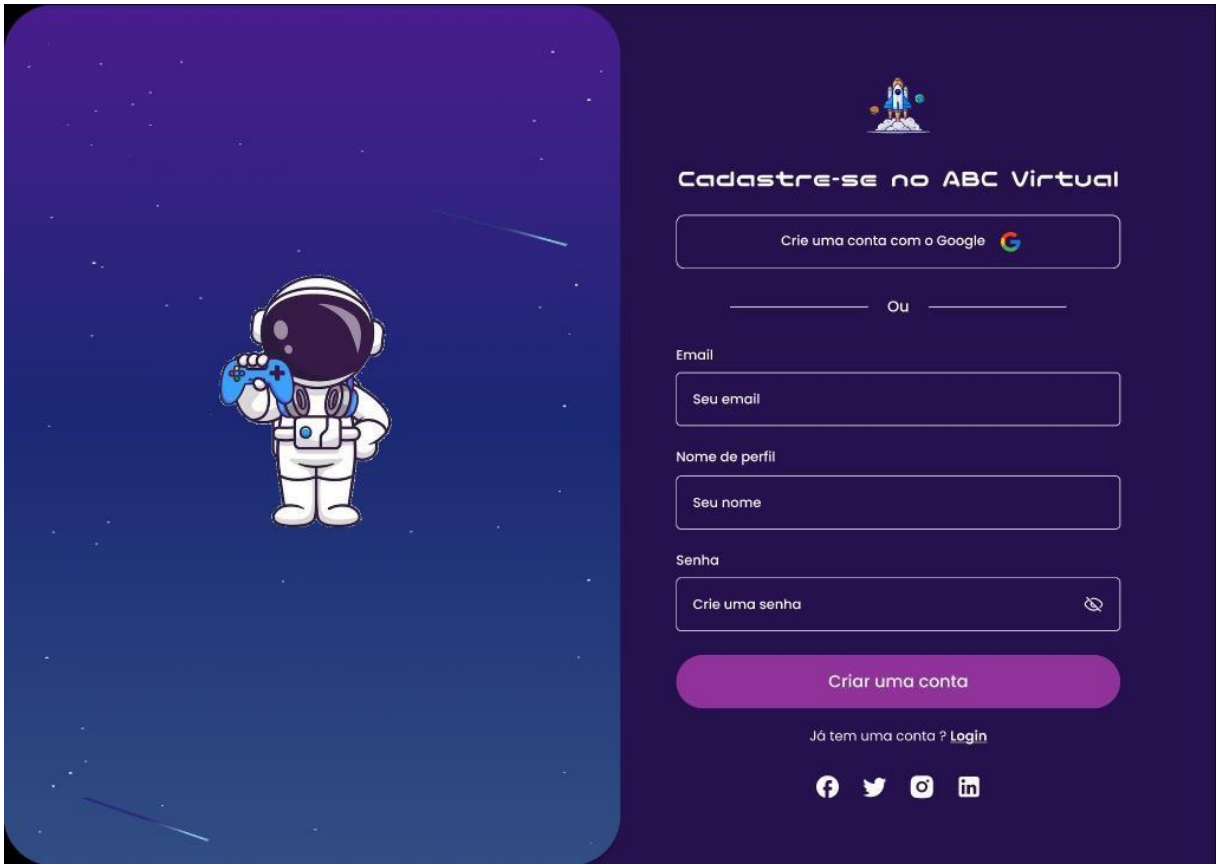


Figura 2: Tela de Cadastro

9.3.2. Tela de Login

Na tela de login, os usuários necessitam informar seus dados feitos no momento do cadastro para poder entrar dentro da sua conta na plataforma.

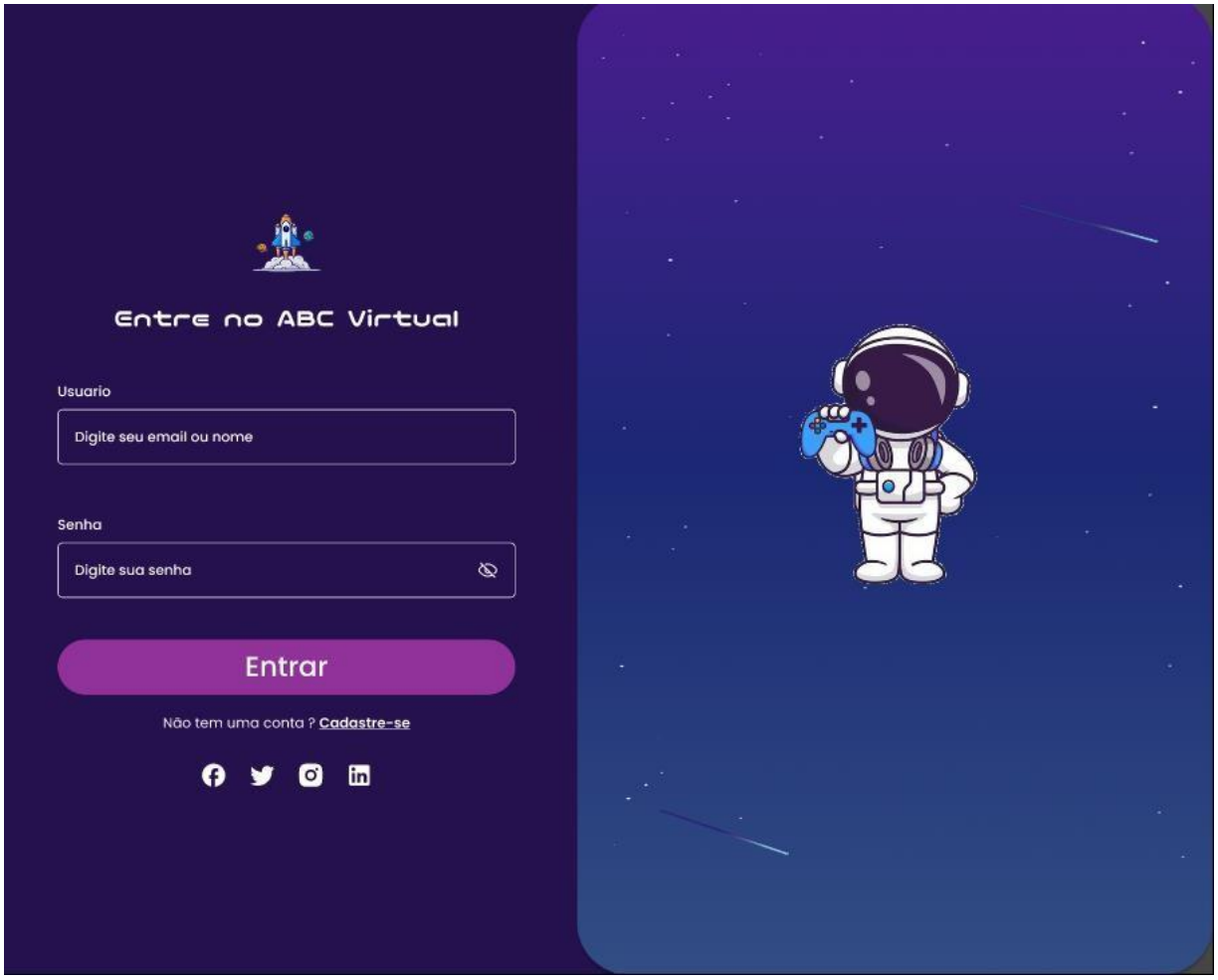


Figura 3: Tela de Login

9.3.3. Tela Inicial do Site

Após concluir o processo de cadastro ou login dentro da plataforma, o usuário é direcionado a tela inicial do site, onde há informações sobre o objetivo buscado pelo projeto, sobre o grupo, sobre como funcionará todo o processo de aprendizagem e ao fim os planos de assinaturas e seus benefícios respectivamente.

Figura 4: Tela Inicial do Site

9.3.4. Sessão de Aprendizagem

Na sessão de aprendizagem o usuário é introduzido ao seu aprendizado, com os planetas referenciando cada aula que o aluno terá que realizar. Apenas após a conclusão da unidade atual, que o aluno desbloqueará a próxima unidade.

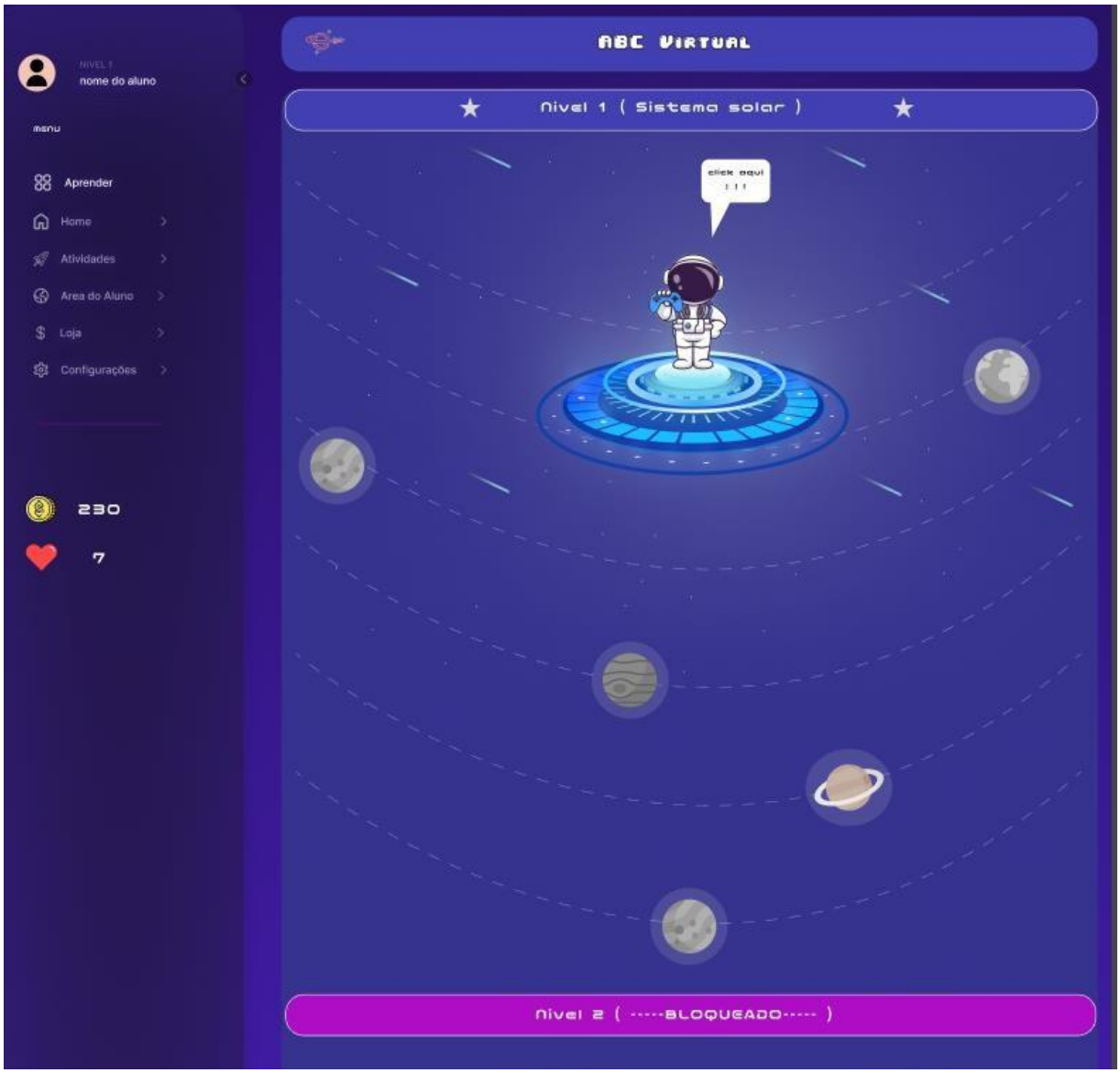


Figura 5: Sessão de Aprendizagem

9.3.5. Sessão de Realidade Virtual

Em cada unidade como mostrado anteriormente terá um mini-game em realidade virtual para exercitar os conhecimentos aprendidos na unidade, de forma que esses mini-games terão uma sessão específica, com a apresentação deles e sobre as outras atividades.

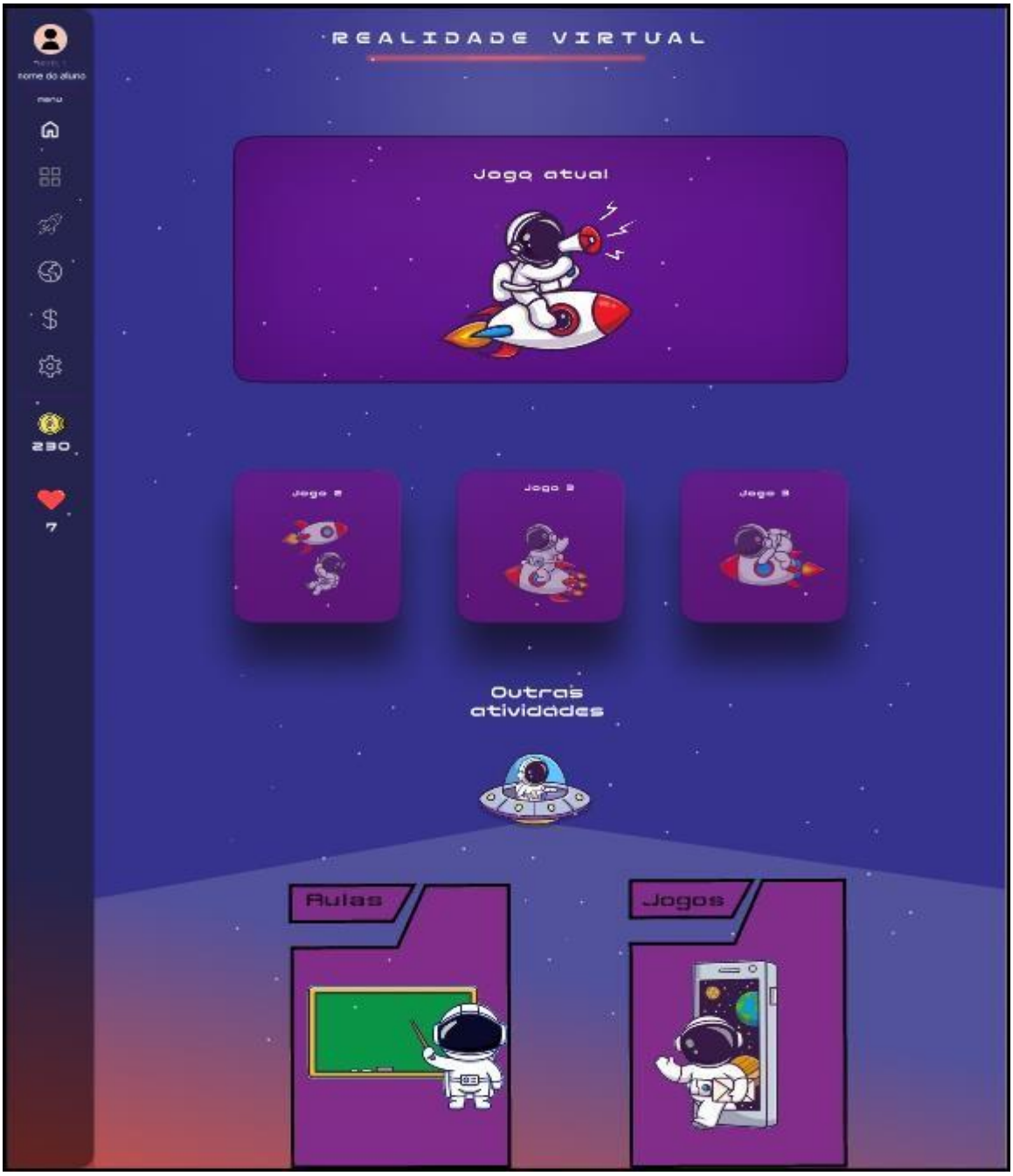


Figura 6: Sessão de Realidade Virtual

9.3.6. Perfil do Aluno

O perfil do aluno terá toda a análise de perfil do aluno, com suas respectivas pontuações nos jogos, um gráfico com essas pontuações e uma tabela na qual é possível verificar qual o seu melhor desempenho.

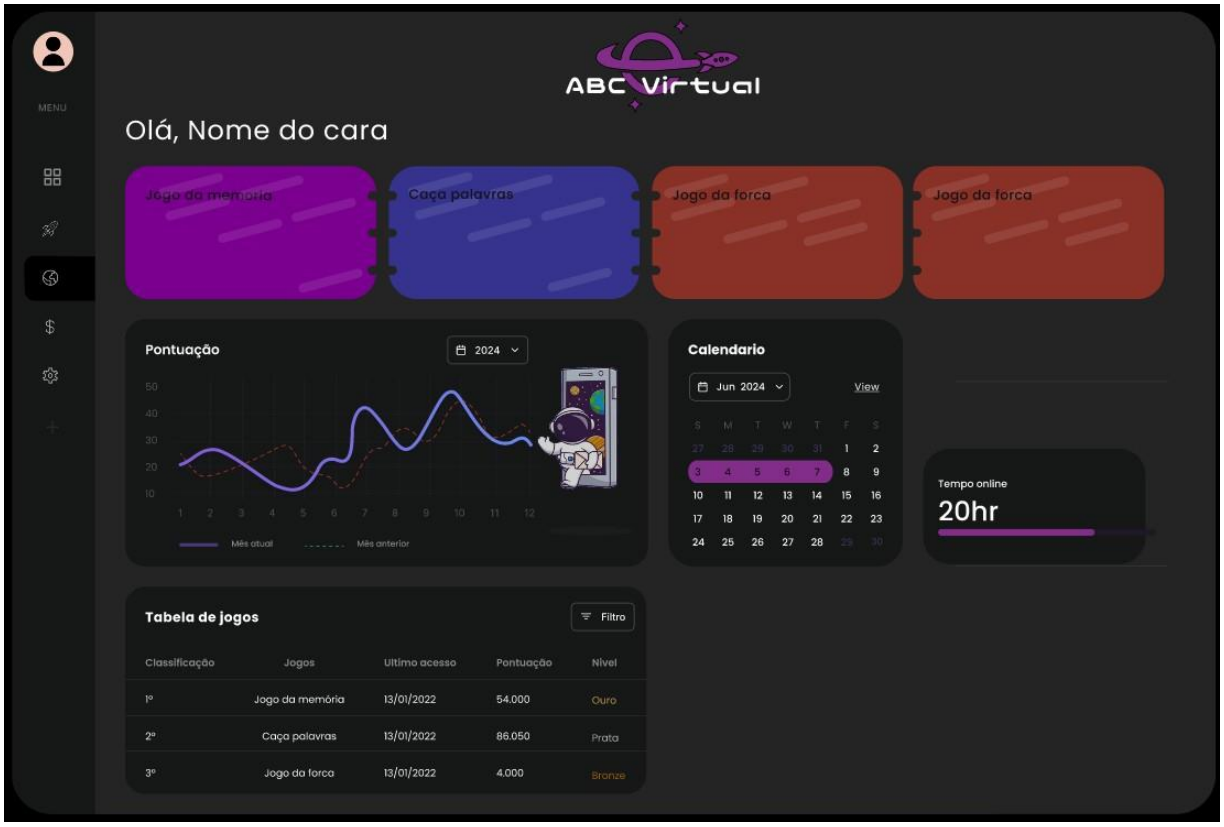


Figura 7: Perfil do Aluno

9.3.7. Loja Virtual

Dentro da plataforma há uma loja virtual, com itens reais que poderão ser comprados a partir de moedas, que os alunos ganham praticando os mini-games de realidade virtual.

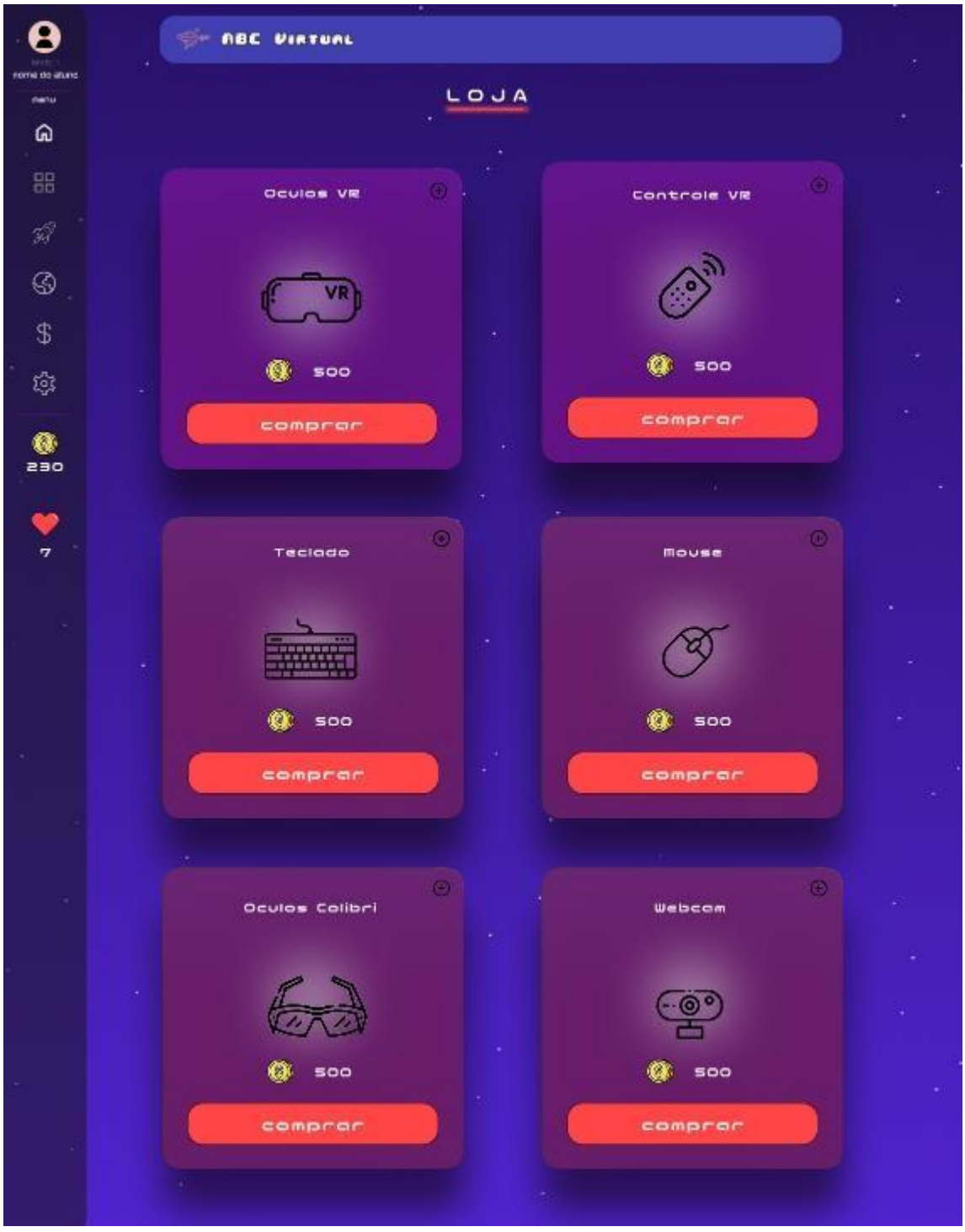


Figura 8: Loja Virtual

9.4. Jogos Realidade Virtual

Dentro da plataforma, haverá jogos em realidade virtual e aumentada, que poderão ser desbloqueadas pelos alunos conforme avancem nas fases. Esses jogos não são obrigatórios para a continuação das fases, serão fases extras que tornam o aprendizado referente à um módulo muito mais imersivo, simulando salas de aula e até o espaço dentro dos jogos.

Figura 9: Sala de Aula em Realidade Virtual

10. RESULTADOS

Foi planejado e desenvolvido um projeto que pudesse atender as necessidades da educação básica, primeiramente para as crianças, no processo de alfabetização, visto que, é um problema que pode afetar a vida da pessoa futuramente. Diante da problemática, o sistema almeja diminuir este déficit que se existe no Brasil, no quesito alfabetização, para poder garantir um futuro digno a todos.

As resoluções iniciais do projeto, podem ser visualizados dentro do tópico anterior do trabalho, onde é possível entender toda a proposta e conceito de uma plataforma de fácil acesso e bem intuitiva, para responsáveis e para os alunos que usufruirão desta tecnologia, a fim de não criar uma preocupação para quem for utilizar, além de aproximar e gerar curiosidade nas crianças que usarem.

Também pode-se encontrar o primeiro mini-game da plataforma em realidade virtual desenvolvido para a aprendizagem, como ilustrado nas figuras e explicações, ele faz a apresentação da criança ao seu nome, com uma similaridade ao jogo da forca, havendo possibilidades de escolhas que são as possíveis letras do nome, e a apresentação do nome na lousa conforme for se completando, além de possuir uma quantidade de erros, que se ultrapassada o mini-game é finalizado.

11. CONSIDERAÇÕES FINAIS

Considerando as informações apresentadas neste trabalho, é evidente que a educação é um direito fundamental para todos, principalmente nas fases iniciais da vida, onde a criança forma seus princípios e valores, portanto com a devida relevância futuramente a sociedade poderá ser muito mais justa. Contudo, não temos que relevar apenas o quesito financeiro da educação, mas também a parte humana, valorizar o apoio das pessoas ao redor do estudante, graças a elas é possível educar também o lado emocional da criança, papel fundamental para o restante da vida.

Além disso, a integração das tecnologias digitais no processo educacional, como o ABC Virtual, representa uma oportunidade significativa para tornar a educação mais acessível, envolvente e eficaz. No entanto, vale destacar o tratamento cuidadoso ao integrar as pessoas aos recursos digitais disponíveis, garantindo que conteúdos passados aos estudantes estejam de acordo com todo o ideal e valor com os princípios da educação.

Em suma, a educação é um direito fundamental de todos os cidadãos da sociedade, na tentativa de erradicar os problemas existentes e buscar soluções inovadoras e inclusivas, é possível construir um sistema educacional que atenda às necessidades de todos os alunos e promova o desenvolvimento humano e social.

REFERÊNCIAS

Letramento e Alfabetização. Disponível em: <https://multivix.edu.br/blog/letramento-alfabetizacao/> Acesso em 28 de abr. 2024.

Importância da Educação para a Sociedade. Disponível em: <https://mudes.org.br/empresa/a-importancia-da-educacao-para-a-sociedade/> Acesso em 28 de abr. de 2024.

Alfabetização e sua Importância para o Futuro das Crianças. Disponível em: <https://www.fadc.org.br/noticias/a-alfabetizacao-e-sua-importancia-para-o-futuro-das-criancas> Acesso em 29 de abr. 2024.

De que Forma a Internet Transformou o que Aprendemos. Disponível em: <https://blog.simple2u.com.br/internet-na-educacao/#De_que_forma_a_Internet_tem_transformado_a_maneira_como_aprendemos> Acesso em 29 de abr. de 2024.

Educação Infantil e Soluções Tecnológicas. Disponível em: <https://movplan.com.br/blog/educacao-infantil-entenda-o-que-e-qual-a-importancia-e-conheca-solucoes-tecnologicas/> Acesso em 29 de abr. de 2024.

A Importância do Brincar na Educação Infantil. Disponível em: <https://docs.uninove.br/arte/fac/publicacoes_pdf/educacao/v5_n1_2014/natali.pdf> Acesso em 25 de mai. de 2024.

Novos Desafios na Educação. Disponível em: <https://cead.ufop.br/site_antigo/arquivos/texto1.pdf> Acesso em 25 de mai. de 2024.

Outros Tempos, Outros Espaços. Disponível em: < https://repositorio.minedu.gob.pe/bitstream/handle/20.500.12799/243/2006_Saraiva_Outros%20tempos%2c%20outros%20espa%c3%a7os%20%20Internet%20e%20ed

uca%c3%a7%c3%a3o.pdf?sequence=1&isAllowed=y> Acesso em 25 de mai. de 2024.

APÊNDICES

APÊNDICE A - BACKEND

Arquivos Controllers

Authentication.JS

```
import bcrypt from "bcrypt"
import 
        import AuthenticationModel
import AuthenticationModel from "../models/Authentication.js"
import ResponsiblesModel from "../models/Responsibles.js"
import StudentsModel from "../models/Students.js"

import ValidateController from "./Validate.js"

class AuthenticationController {
  async signup(req, res) {
    const salt = await bcrypt.genSalt(10)
    const hash = await bcrypt.hash(req.body.studentPassword, salt)

    const student = {
      firstName: req.body.studentFirstName,
      lastName: req.body.studentLastName,
      email: req.body.studentEmail,
      password: hash,
      age: req.body.studentAge,
      responsible_id: ""
    }

    const responsible = {
      firstName: req.body.responsibleFirstName,
      lastName: req.body.responsibleLastName,
      email: req.body.responsibleEmail,
      age: req.body.responsibleAge
    }

```

```
    const responsibleExistsCheck = req.body.responsibleExistsCheck

    const responsibleFindByEmail = await ResponsiblesModel.findByEmail(req.body.responsibleEmail)

    const studentFindByEmail = await StudentsModel.findByEmail(req.body.studentEmail)

    if(!(ValidateController.name(req.body.studentFirstName)) || !(ValidateController.name(req.body.studentLastName)) || !(ValidateController.email
    (req.body.studentEmail)) || !(ValidateController.password(req.body.studentPassword)) || !(ValidateController.age(req.body.studentAge))) {
      return res.status(400).json({ message: "Student data is invalid!" })
    }

    if(!(ValidateController.name(req.body.responsibleFirstName)) || !(ValidateController.name(req.body.responsibleLastName)) || !(ValidateController.
    email(req.body.responsibleEmail)) || !(ValidateController.age(req.body.responsibleAge)) || responsibleExistsCheck === undefined) {
      return res.status(400).json({ message: "Responsible data is invalid!" })
    }

    if(studentFindByEmail !== null) return res.status(409).json({ message: "There is already a student with this email!" })

    if(responsibleExistsCheck === "true") {
      if(responsibleFindByEmail === null) {
        return res.status(409).json({ message: "This person is not registered!" })
      } else {
        student.responsible_id = responsibleFindByEmail._id

        await AuthenticationModel.signup({ student, responsible: responsibleFindByEmail }, responsibleExistsCheck )
      }
    } else {
      if(responsibleFindByEmail === null) {
        await AuthenticationModel.signup({ student, responsible }, responsibleExistsCheck )
      } else {
        return res.status(409).json({ message: "This person is already registered!" })
      }
    }

    return res.status(200).json({ message: "User registered successfully!" })
  }

  async signin(req, res) {
    const { studentEmail, studentPassword } = req.body

    const studentFindByEmail = await StudentsModel.findByEmail(studentEmail)

    if(studentFindByEmail === null) return res.status(404).json({ message: "There is already a student with this email!" })

    const studentPasswordCompare = bcrypt.compare(studentPassword, studentFindByEmail.password)

    if(!studentPasswordCompare) return res.status(400).json({ message: "Incorrect password!" })

    const token = jwt.sign({
      id: studentFindByEmail._id,
      firstName: studentFindByEmail.firstName,
      lastName: studentFindByEmail.lastName,
      email: studentFindByEmail.email,
      age: studentFindByEmail.age,
      registration: studentFindByEmail.registration,
      coins: studentFindByEmail.coins,
      last_level: studentFindByEmail.last_level,
      products: studentFindByEmail.products,
      responsible_id: studentFindByEmail.responsible_id
    }, process.env.JWT_SECRET, { expiresIn: "24h" })

    return res.status(200).json({ token, message: "User successfully authenticated!" })
  }
}

export default new AuthenticationController
```

Levels.JS

```
import LevelsModel from "../models/Levels.js"
import SeasonsModel from "../models/Seasons.js"
import StudentsModel from "../models/Students.js"

import ValidateController from "./Validate.js"

class LevelController {
    async create(req, res) {
        const { title, description, difficulty, minigame, reward, season_id } = req.body

        if(!(ValidateController.name(title)) ||  !(ValidateController.name(description)) || !(ValidateController.number(difficulty)) || !
        (ValidateController.name(minigame)) || !(ValidateController.number(reward))) {
            return res.status(400).json({ message: "Invalid level data!" })
        }

        const season = await SeasonsModel.findById(season_id)

        if(season === undefined || season === null) return res.status(404).json({ message: "This season doesn't exist!" })

        const lastLevel = await LevelsModel.findLast()

        const lastLevelNumber = lastLevel === null ? 1 : lastLevel.number + 1

        await LevelsModel.create({
            level: {
                title,
                description,
                difficulty,
                minigame,
                reward,
                 number: lastLevelNumber,
                 season_id: season_id
             }
        })

        res.status(200).json({ message: "Level created successfully!" })
    }

    async findAll(req, res) {
        const levels = await LevelsModel.findAll()

        res.status(200).json({ levels, message: "All levels searched successfully!" })
    }

    async findById(req, res) {
        const { id } = req.params

        const level = await LevelsModel.findById(id)

        if(level === undefined || level === null) return res.status(404).json({ message: "This level doesn't exist!" })

        res.status(200).json({ level, message: "Level successfully searched for ID!" })
    }

    async findAllBySeason(req, res) {
        const { id } = req.params

       const levels = await LevelsModel.findAllBySeason(id)

       if(levels === undefined || levels === null) return res.status(404).json({ message: "This season doesn't exist!" })

       res.status(200).json({ levels, message: "Level successfully searched for season!" })
    }

    async findLast(req, res) {
        const level = await LevelsModel.findLast()

        res.status(200).json({ level, message: "Last level successfully searched!" })
    }

    async update(req, res) {
        const { id } = req.params

        const { title, description, difficulty, minigame, reward } = req.body

        if(!(ValidateController.name(title)) ||  !(ValidateController.name(description)) || !(ValidateController.number(difficulty)) || !
        (ValidateController.name(minigame)) || !(ValidateController.number(reward))) {
            return res.status(400).json({ message: "Invalid level data!" })
        }

        const level = await LevelsModel.update(id, { title, description, difficulty, minigame, reward })

        if(level === undefined || level === null) return res.status(404).json({ message: "This level doesn't exist!" })

        res.status(200).json({ level, message: "Level updated successfully!" })
```

```
    }

    async finish(req, res) {
        const levelId = req.params.id
        const session = req.session

        const level = await LevelsModel.findById(levelId)

        if(level === undefined || level === null) return res.status(404).json({ message: "This level doesn't exist!" })

        const student = await StudentsModel.findById(session.id)

        if(student === undefined || student === null) return res.status(404).json({ message: "This student doesn't exist!" })

        const coinsAfterFinishingLevel = student.coins + level.reward

        await LevelsModel.finish(student._id, { last_level: level.number, coins: coinsAfterFinishingLevel })

        res.status(200).json({ level, message: "Level finished successfully!" })
    }

    async delete(req, res) {
        const { id } = req.params

        const level = await LevelsModel.delete(id)

        if(level === undefined) return res.status(404).json({ message: "This level doesn't exist!" })

            res.status(200).json({ message: "Level deleted successfully!" })
        }
    }

    export default new LevelController
```

Payments.JS

```
import { Payment, MercadoPagoConfig } from 'mercadopago';
import { v4 as uuidv4 } from 'uuid';

class PaymentController {
    async create(req, res) {
        const client = new MercadoPagoConfig({ accessToken: 'TEST-320461010060425-051822-f1c2675a5d2d5f687837e9f88e17d38d-411772534' });

        const payment = new Payment(client);

        payment.create({
            body: {
                transaction_amount: req.transaction_amount,
                token: req.token,
                description: req.description,
                installments: req.installments,
                payment_method_id: req.paymentMethodId,
                issuer_id: req.issuer,
                    payer: {
                    email: req.email,
                    identification: {
                type: req.identificationType,
                number: req.number
            }}},
            requestOptions: { idempotencyKey: uuidv4() }
        })
        .then(result => console.log(result))
        .catch(error => console.log(error));
    }
}
export default new PaymentController
```

Products.JS

```
import ProductsModel from "../models/Products.js"
import StudentsModel from "../models/Students.js"

import ValidateController from "./Validate.js"

class ProductsController {
    async create(req, res) {
        const { name, description, price, amount } = req.body

        if(!(ValidateController.name(name)) ||  !(ValidateController.name(description)) || !(ValidateController.number(price)) || !(ValidateController.number(amount))) {
            return res.status(400).json({ message: "Invalid product data!" })
        }

        await ProductsModel.create({ name, description, price, amount })

        res.status(200).json({ message: "Product created successfully!" })
    }

    async findAll(req, res) {
        const products = await ProductsModel.findAll()

        res.status(200).json({ products, message: "All products searched successfully!" })
    }

    async findById(req, res) {
        const { id } = req.params

        const product = await ProductsModel.findById(id)

        if(product === undefined || product === null) return res.status(404).json({ message: "This product doesn't exist!" })

        res.status(200).json({ product, message: "Product successfully searched for ID!" })
    }

    async update(req, res) {
        const { id } = req.params

        const { name, description, price, amount } = req.body

        if(!(ValidateController.name(name)) ||  !(ValidateController.name(description)) || !(ValidateController.number(price)) || !(ValidateController.number(amount))) {
            return res.status(400).json({ message: "Invalid product data!" })
        }

        const product = await ProductsModel.update(id, { name, description, price, amount })

        if(product === undefined) return res.status(404).json({ message: "This product doesn't exist!" })

        res.status(200).json({ product, message: "Product updated successfully!" })
    }

    async buy(req, res) {
        const session = req.session

        const productId = req.params.id

        const product = await ProductsModel.findById(productId)

        if(product === undefined || product === null) return res.status(404).json({ message: "This product doesn't exist!" })

        if(product.amount <= 0) return res.status(410).json({ message: "This product is sold out!" })

        const student = await StudentsModel.findById(session.id)

        if(student === undefined || student === null) return res.status(404).json({ message: "This student doesn't exist!" })

        if(student.coins < product.price) return res.status(410).json({ message: "The student does not have enough coins" })

        let productAlreadyPurchased = null

        student.products.forEach(productId => {
            String(productId) === String(product._id)
                ? productAlreadyPurchased = true : productAlreadyPurchased = false
        })

        if(productAlreadyPurchased === true) return res.status(403).json({ message: "This student has already purchased this product!" })

        const amountAfterPurchase = { amount: product.amount - 1 }
        const coinsAfterPurchase = student.coins - product.price

        const productAndStudent = await ProductsModel.buy(student._id, productId, { amountAfterPurchase, product: { products: productId }, coinsAfterPurchase })

        if(productAndStudent === undefined) return res.status(500).json({ message: "Failed to make purchase!" })

```

```
            res.status(200).json({ productAndStudent, message: "Product purchased successfully!" })
        }

        async delete(req, res) {
            const { id } = req.params

            const product = await ProductsModel.delete(id)

            if(product === undefined) return res.status(404).json({ message: "This product doesn't exist!" })

            res.status(200).json({ message: "Product deleted successfully!" })
        }
    }

    export default new ProductsController
```

Responsibles.JS

```
import ResponsiblesModel from "../models/Responsibles.js"

import ValidateController from "./Validate.js"

class ResponsibleController {
    async findAll(req, res) {
        const responsibles = await ResponsiblesModel.findAll()

        res.status(200).json({ responsibles, message: "All those responsible successfully searched!" })
    }

    async findById(req, res) {
        const { id } = req.params

        const responsible = await ResponsiblesModel.findById(id)

        if(responsible === undefined || responsible === null) return res.status(404).json({ message: "This responsible doesn't exist!" })

        res.status(200).json({ responsible, message: "Responsible successfully searched for ID!" })
    }

    async update(req, res) {
        const { id } = req.params

        const { firstName, lastName, email, age } = req.body

        if(!(ValidateController.name(firstName)) ||  !(ValidateController.name(lastName)) || !(ValidateController.email(email)) || !
        (ValidateController.number(age))) {
            return res.status(400).json({ message: "Invalid responsible data!" })
        }

          const responsible = await ResponsiblesModel.update(id, { firstName, lastName, email, age })

          if(responsible === undefined) return res.status(404).json({ message: "This responsible doesn't exist!" })

          res.status(200).json({ responsible, message: "Responsible updated successfully!" })
      }

      async delete(req, res) {
          const { id } = req.params

          const responsible = await ResponsiblesModel.delete(id)

          if(responsible === undefined) return res.status(404).json({ message: "This responsible doesn't exist!" })

          res.status(200).json({ message: "Responsible deleted successfully!" })
      }
  }

  export default new ResponsibleController
```

Seasons.JS

```
import SeasonModel from "../models/Seasons.js"

import ValidateController from "./Validate.js"

class SeasonController {
    async create(req, res) {
        const { title, description, theme } = req.body

        if(!(ValidateController.name(title)) ||  !(ValidateController.name(description)) || !(ValidateController.name(theme))) {
            return res.status(400).json({ message: "Invalid season data!" })
        }

        await SeasonModel.create({ title, description, theme })

        res.status(200).json({ message: "Season created successfully!" })
    }

    async findAll(req, res) {
        const seasons = await SeasonModel.findAll()

        res.status(200).json({ seasons, message: "All seasons searched successfully!" })
    }

    async findById(req, res) {
        const { id } = req.params

        const season = await SeasonModel.findById(id)

        if(season === undefined || season === null) return res.status(404).json({ message: "This season doesn't exist!" })

        res.status(200).json({ season, message: "Season successfully searched for ID!" })
    }

    async update(req, res) {
        const { id } = req.params

        const { title, description, theme } = req.body

        if(!(ValidateController.name(title)) ||  !(ValidateController.name(description)) || !(ValidateController.name(theme))) {
            return res.status(400).json({ message: "Invalid season data!" })
        }

        const season = await SeasonModel.update(id, { title, description, theme})

        if(season === undefined) return res.status(404).json({ message: "This season doesn't exist!" })

        res.status(200).json({ season, message: "Season updated successfully!" })
    }

    async delete(req, res) {
        const { id } = req.params

        const season = await SeasonModel.delete(id)

        if(season === undefined) return res.status(404).json({ message: "This season doesn't exist!" })

        res.status(200).json({ message: "Season deleted successfully!" })
    }
}

export default new SeasonController
```

Students.JS

```
import bcrypt from "bcrypt"

import StudentsModel from "../models/Students.js"
import ValidateController from "./Validate.js"

class StudentController {
    async findAll(req, res) {
        const students = await StudentsModel.findAll()

        res.status(200).json({ students, message: "All students searched successfully!" })
    }

    async findById(req, res) {
        const { id } = req.params

        const student = await StudentsModel.findById(id)

        if(student === undefined || student === null) return res.status(404).json({ message: "This student doesn't exist!" })

        res.status(200).json({ student, message: "Student successfully searched for ID!" })
    }

    async update(req, res) {
        const { id } = req.params

        const { firstName, lastName, email, password, age  } = req.body

        const salt = await bcrypt.genSalt(10)
        const hash = await bcrypt.hash(password, salt)

        if(!(ValidateController.name(firstName)) || !(ValidateController.name(lastName)) || !(ValidateController.email(email)) || !
        (ValidateController.password(password)) || !(ValidateController.age(age))) {
            return res.status(400).json({ message: "Student data is invalid!" })
          }

        const student = await StudentsModel.update(id, { firstName, lastName, email, password: hash, age })

        if(student === undefined) return res.status(404).json({ message: "This student doesn't exist!" })

        res.status(200).json({ student, message: "Student updated successfully!" })
    }

    async delete(req, res) {
        const { id } = req.params

        const student = await StudentsModel.delete(id)

        if(student === undefined) return res.status(404).json({ message: "This student doesn't exist!" })

        res.status(200).json({ message: "Student deleted successfully!" })
    }
}

export default new StudentController
```

Validate.JS

```
class ValidateController {
  name(value) {
    if(value === undefined) {
      return false
    } else {
      return (/^[a-záàâãéèêíïóôõöúçñ]+/i).test(value)
    }
  }

  email(value) {
    if(value === undefined) {
      return false
    } else {
      return (/\S+@\S+\.\S+/).test(value)
    }
  }

  password(value) {
    if(value === undefined || value.length < 8) {
      return false
    } else {
      return true
    }
  }

  age(value) {
    if(value === undefined) {
      return false
    } else {
      return true
    }
  }

  number(value) {
    if(value === undefined || !((/^[0-9]+$/).test(value))) {
      return false
    } else {
      return true
    }
  }
}

export default new ValidateController
```

Arquivos Database

Connection.JS

```
import mongoose from "mongoose"

export const connection = async () => {
    try {
        await mongoose.connect(process.env.DATABASE_HOST)
        console.log("Connected to the database!")
    } catch (error) {
        console.log("Failed to connect to database!")
    }
}
```

Arquivos Middlewares

Authentication.JS

```
import dotenv from "dotenv"
import jwt from "jsonwebtoken"

dotenv.config()

export const authentication = async (req, res, next) => {
    try {
        const bearer = req.headers["authorization"]

        if(!bearer) return res.status(401).json({ message: "Invalid token." })

        const token = bearer.replace("Bearer ", "")

        try {
            req.session = jwt.verify(token, process.env.JWT_SECRET)
        } catch (error) {
            return res.status(401).json({ message: "Expired token." })
        }

        next()
    } catch(error) {
        console.log(`There was an error checking the token: ${error}`)
    }
}
```

Arquivos Models

Authentication.JS

```
import ResponsiblesModel from "../models/Responsibles.js"
import StudentsModel from "../models/Students.js"

class AuthenticationModel {
    async signup(data, responsibleExists) {
      try {
        if(responsibleExists === "true") {
          return await StudentsModel.create(data.student)
        }

        const responsible = await ResponsiblesModel.create(data.responsible)

        data.student.responsible_id = responsible._id

        return await StudentsModel.create(data.student)
      } catch (error) {
        console.log(`Failed to register student: ${error}`)
      }
    }
}

export default new AuthenticationModel
```

Levels.JS

```
import mongoose from "mongoose"

import LevelsSchema from "../schemas/levels.js"
import studentsSchema from "../schemas/students.js"

const levels = mongoose.model("levels", LevelsSchema)
const students = mongoose.model("students", studentsSchema)

class LevelsModel {
    async create(data) {
      try {
        const level = await levels(data.level).save()
      } catch (error) {
        console.log(`Failed to create a level: ${error}`)
      }
    }

    async findAll() {
        try {
            return await levels.find()
        } catch (error) {
            console.log(`Failed to fetch all levels: ${error}`)
        }
    }

    async findById(id) {
        try {
            return await levels.findById(id)
        } catch (error) {
            return undefined
```

```
        }
    }

    async findAllBySeason(id) {
        try {
            return await levels.find({ season_id: id })
        } catch(error) {
            return undefined
        }
    }

    async findLast() {
        try {
            return await levels.findOne().sort({ field: "asc", _id: -1 })
        } catch(error) {
            return undefined
        }
    }

    async update(id, data) {
        try {
            return await levels.findByIdAndUpdate(id, data)
        } catch (error) {
            return undefined
        }
    }

    async finish(id, data){
        try {
            return await students.findByIdAndUpdate(id, data)
        } catch (error) {
            return undefined
        }
    }

    async delete(id) {
        try {
            return await levels.findByIdAndDelete(id)
        } catch (error) {
            return undefined
        }
    }
}

export default new LevelsModel
```

Products.JS

```
import mongoose from "mongoose"

import ProductsSchema from "../schemas/products.js"
import StudentsSchema from "../schemas/students.js"

const products = mongoose.model("products", ProductsSchema)
const students = mongoose.model("students", StudentsSchema)

class ProductsModel {
    async create(data) {
      try {
        await products(data).save()
      } catch (error) {
        console.log(`Failed to create a product: ${error}`)
      }
    }

    async findAll() {
        try {
            return await products.find()
        } catch (error) {
            console.log(`Failed to fetch all products: ${error}`)
        }
    }

    async findById(id) {
        try {
            return await products.findById(id)
        } catch (error) {
            return undefined
    }
}

async update(id, data) {
    try {
        return await products.findByIdAndUpdate(id, data)
    } catch (error) {
        return undefined
    }
}

async buy(studentId, productId, data) {
    try {
        const product = await products.findByIdAndUpdate(productId, data.amountAfterPurchase)
        const student = await students.findByIdAndUpdate(studentId, { $push: data.product, coins: data.coinsAfterPurchase })

        return { product, student }
    } catch (error) {
        return undefined
    }
}

async delete(id) {
    try {
        return await products.findByIdAndDelete(id)
    } catch (error) {
        return undefined
    }
```

```
58            }
59        }
60    }
61
62    export default new ProductsModel
```

Responsibles.JS

```
1   import mongoose from "mongoose"
2
3   import ResponsiblesSchema from "../schemas/responsibles.js"
4
5   const responsibles = mongoose.model("responsibles", ResponsiblesSchema)
6
7   class ResponsiblesModel {
8       async create(data) {
9         try {
10          return await responsibles(data).save()
11        } catch (error) {
12          console.log(`Failed to create a responsible: ${error}`)
13        }
14      }
15
16      async findAll() {
17        try {
18          return await responsibles.find()
19        } catch (error) {
20          console.log(`Failure to search for all those responsible: ${error}`)
21        }
22      }
23
24      async findById(id) {
25        try {
26          return await responsibles.findById(id)
27        } catch (error) {
28          console.log(`Failed to search for responsible person by ID: ${error}`)
29        }
30      }
```

```
    }

    async findByEmail(email) {
      try {
        return await responsibles.findOne({ email })
      } catch (error) {
        console.log(`Failed to search for responsible person by email: ${error}`)
      }
    }

    async update(id, data) {
      try {
          return await responsibles.findByIdAndUpdate(id, data)
      } catch (error) {
          return undefined
      }
    }

    async delete(id) {
        try {
            return await responsibles.findByIdAndDelete(id)
        } catch (error) {
            return undefined
        }
    }
}

export default new ResponsiblesModel
```

Seasons.JS

```
import mongoose from "mongoose"

import SeasonsSchema from "../schemas/seasons.js"

const seasons = mongoose.model("seasons", SeasonsSchema)

class SeasonsModel {
    async create(data) {
      try {
        await seasons(data).save()
      } catch (error) {
        console.log(`Failed to create a season: ${error}`)
      }
    }

    async findAll() {
        try {
            return await seasons.find()
        } catch (error) {
            console.log(`Failed to fetch all seasons: ${error}`)
        }
    }

    async findById(id) {
        try {
            return await seasons.findById(id)
        } catch (error) {
            return undefined
        }
    }
```

```
31
32      async update(id, data) {
33          try {
34              return await seasons.findByIdAndUpdate(id, data)
35          } catch (error) {
36              return undefined
37          }
38      }
39
40      async delete(id) {
41          try {
42              return await seasons.findByIdAndDelete(id)
43          } catch (error) {
44              return undefined
45          }
46      }
47  }
48
49  export default new SeasonsModel
```

Students.JS

```
import mongoose from "mongoose"

import StudentsSchema from "../schemas/students.js"

const students = mongoose.model("students", StudentsSchema)

class StudentsModel {
  async create(data) {
    try {
      return await students(data).save()
    } catch (error) {
      console.log(`Failed to create a student: ${error}`)
    }
  }

  async findAll() {
    try {
      return await students.find()
    } catch (error) {
      console.log(`Failure to search for all those students: ${error}`)
    }
  }

  async findById(id) {
    try {
      return await students.findById(id)
    } catch (error) {
      console.log(`Failed to search for student by ID: ${error}`)
    }
  }
```

```
31
32      async findByEmail(email) {
33        try {
34          return await students.findOne({ email })
35        } catch (error) {
36          console.log(`Failed to search for student person by email: ${error}`)
37        }
38      }
39
40      async update(id, data) {
41        try {
42            return await students.findByIdAndUpdate(id, data)
43        } catch (error) {
44            return undefined
45        }
46      }
47
48      async delete(id) {
49          try {
50              return await students.findByIdAndDelete(id)
51          } catch (error) {
52              return undefined
53          }
54      }
55  }
56
57  export default new StudentsModel
```

Arquivos Routes

Authentication.JS

```
1  import express from "express"
2
3  import AuthenticationController from "../controllers/Authentication.js"
4
5  export const authenticationRouter = express.Router()
6
7  authenticationRouter.post("/signup", AuthenticationController.signup)
8  authenticationRouter.post("/signin", AuthenticationController.signin)
```

Levels.JS

```
import express from "express"

import LevelController from "../controllers/Levels.js"
import { authentication } from "../middlewares/authentication.js"

export const levelRouter = express.Router()

levelRouter.post("/", LevelController.create)

levelRouter.get("/", LevelController.findAll)
levelRouter.get("/last", LevelController.findLast)
levelRouter.get("/:id", LevelController.findById)
levelRouter.get("/season/:id", LevelController.findAllBySeason)

levelRouter.put("/finish/:id", authentication, LevelController.finish)
levelRouter.put("/:id", LevelController.update)

levelRouter.delete("/:id", LevelController.delete)
```

Payments.JS

```
import express from "express"

import PaymentController from "../controllers/Payments.js"

export const paymentRouter = express.Router()

paymentRouter.post("/process_payment", PaymentController.create)
```

Products.JS

```
import express from "express"

import ProductsController from "../controllers/Products.js"
import { authentication } from "../middlewares/authentication.js"

export const productRouter = express.Router()

productRouter.post("/", ProductsController.create)

productRouter.get("/", ProductsController.findAll)
productRouter.get("/:id", ProductsController.findById)

productRouter.put("/buy/:id", authentication, ProductsController.buy)
productRouter.put("/:id", ProductsController.update)

productRouter.delete("/:id", ProductsController.delete)
```

Responsibles.JS

```
import express from "express"

import ResponsibleController from "../controllers/Responsibles.js"

export const responsibleRouter = express.Router()

responsibleRouter.get("/", ResponsibleController.findAll)
responsibleRouter.get("/:id", ResponsibleController.findById)

responsibleRouter.put("/:id", ResponsibleController.update)

responsibleRouter.delete("/:id", ResponsibleController.delete)
```

Seasons.JS

```
import express from "express"

import SeasonController from "../controllers/Seasons.js"

export const seasonRouter = express.Router()

seasonRouter.post("/", SeasonController.create)

seasonRouter.get("/", SeasonController.findAll)
seasonRouter.get("/:id", SeasonController.findById)

seasonRouter.put("/:id", SeasonController.update)

seasonRouter.delete("/:id", SeasonController.delete)
```

Studensts.JS

```
import express from "express"

import StudentController from "../controllers/Students.js"
import { authentication } from "../middlewares/authentication.js"

export const studentsRouter = express.Router()

studentsRouter.get("/", authentication, StudentController.findAll)
studentsRouter.get("/:id", StudentController.findById)

studentsRouter.put("/:id", StudentController.update)

studentsRouter.delete("/:id", StudentController.delete)
```

Arquivos Schemas

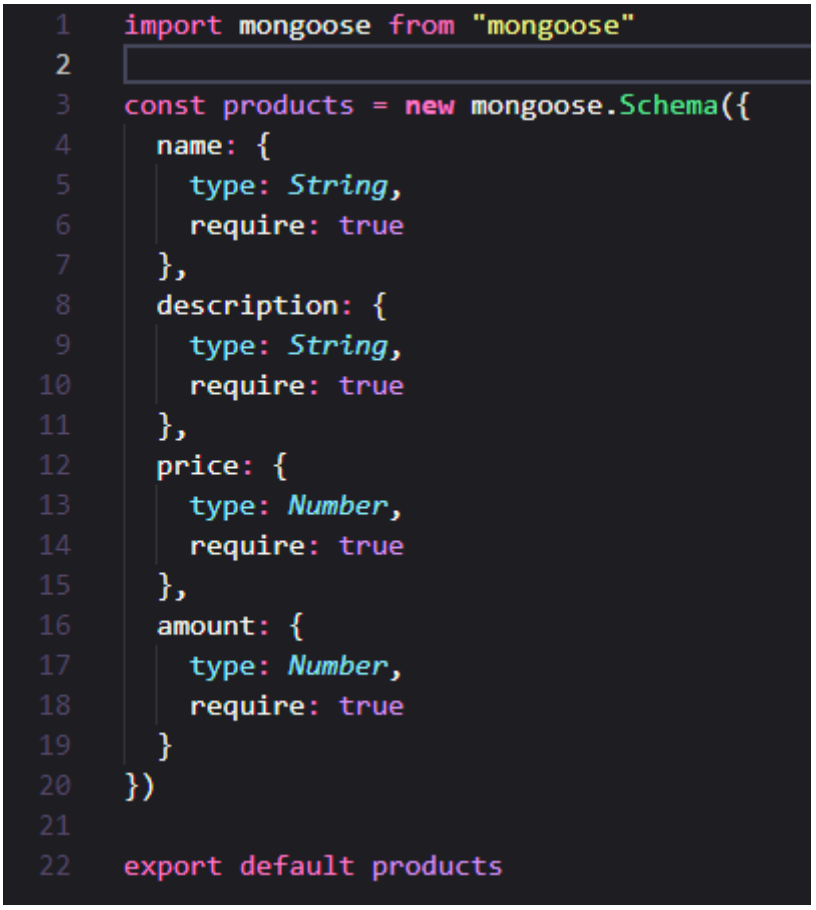
Levels.JS

```
import mongoose from "mongoose"

const levels = new mongoose.Schema({
  title: {
    type: String,
    require: true
  },
  description: {
    type: String,
    require: true
  },
  difficulty: {
    type: Number,
    require: true
  },
  minigame: {
    type: String,
    require: true
  },
  reward: {
    type: Number,
    require: true
  },
  number: {
    type: Number,
    require: true
  },
  season_id: {
    type: mongoose.Types.ObjectId,
    require: true
  }
})

export default levels
```

Products.JS

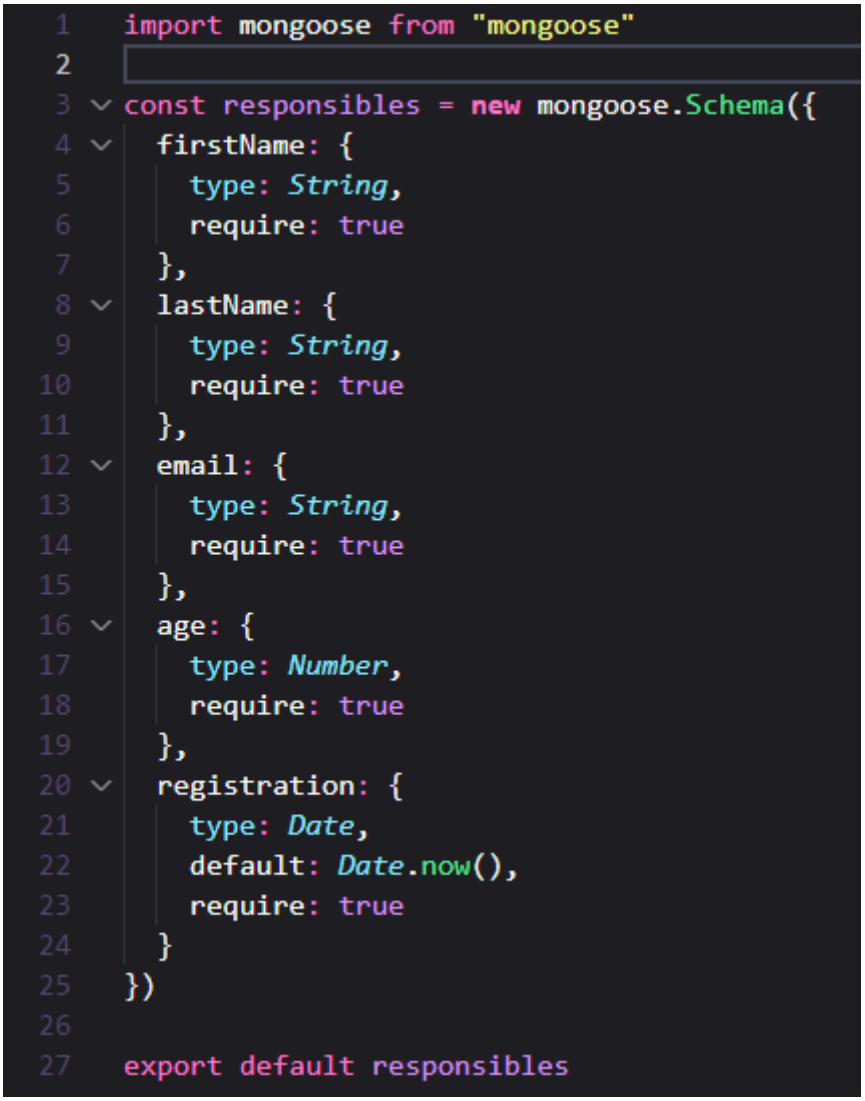


```
import mongoose from "mongoose"

const products = new mongoose.Schema({
  name: {
    type: String,
    require: true
  },
  description: {
    type: String,
    require: true
  },
  price: {
    type: Number,
    require: true
  },
  amount: {
    type: Number,
    require: true
  }
})

export default products
```

Responsibles.JS



```
import mongoose from "mongoose"

const responsibles = new mongoose.Schema({
  firstName: {
    type: String,
    require: true
  },
  lastName: {
    type: String,
    require: true
  },
  email: {
    type: String,
    require: true
  },
  age: {
    type: Number,
    require: true
  },
  registration: {
    type: Date,
    default: Date.now(),
    require: true
  }
})

export default responsibles
```

Seasons.JS

```
import mongoose from "mongoose"

const seasons = new mongoose.Schema({
  title: {
    type: String,
    require: true
  },
  description: {
    type: String,
    require: true
  },
  theme: {
    type: String,
    require: true
  }
})

export default seasons
```

Students.JS

```
import mongoose from "mongoose"

const students = new mongoose.Schema({
  firstName: {
    type: String,
    require: true
  },
  lastName: {
    type: String,
    require: true
  },
  email: {
    type: String,
    require: true
  },
  password: {
    type: String,
    require: true
  },
  age: {
    type: Number,
    require: true
  },
  registration: {
    type: Date,
    default: Date.now(),
    require: true
  },
  coins: {
    type: Number,
    default: 0,
    require: true
  },
  last_level: {
    type: Number,
    default: 0,
    require: true
  },
  products: [ mongoose.Types.ObjectId ],
  responsible_id: {
    type: mongoose.Types.ObjectId,
    require: true
  }
})

export default students
```

App.JS

```
import express from "express"
import dotenv from "dotenv"
import cors from "cors"

import { connection } from "./database/connection.js"

import { authenticationRouter } from "./routes/authentication.js"
import { studentsRouter } from "./routes/students.js"
import { responsibleRouter } from "./routes/responsibles.js"
import { seasonRouter } from "./routes/seasons.js"
import { levelRouter } from "./routes/levels.js"
import { productRouter } from "./routes/products.js"
import { paymentRouter } from "./routes/payment.js"

dotenv.config()

await connection()

export const app = express()

app.use(cors())

app.use(express.urlencoded({ extended: false }))
app.use(express.json())

app.use("/authentication", authenticationRouter)
app.use("/students", studentsRouter)
app.use("/seasons", seasonRouter)
app.use("/levels", levelRouter)
app.use("/responsibles", responsibleRouter)
```

```
app.use("/products", productRouter)
app.use("/payment", paymentRouter)
```

Server.JS

```
import { app } from "./app.js"

app.listen(process.env.PORT, () => console.log("Server running!"))
```

APÊNDICE B - REALIDADE VIRTUAL

Index.HTML

```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="shortcut icon" href="/src/assets/imgs/pencil-icon.png" type="image/x-icon">
  <link rel="stylesheet" href="./src/styles/reset.css">
  <link rel="stylesheet" href="./src/styles/index.css">
  <title>Cenário interativo - VR</title>
</head>
<body>
  <main>
    <div class="glass-container">
      <form>
        <h1>Sala de Aula Virtual</h1>
        <div class="input-container">
          <input type="text" id="name" placeholder="Máx. 20 caracteres" autocomplete="off" maxlength="20" minlength="2"
          required>
          <label for="input" class="label">Insira seu nome</label>
          <div class="underline"></div>
        </div>
        <button type="submit" class="sigin">
          <span class="shadow"></span>
          <span class="edge"></span>
          <span class="front">Entrar</span>
        </button>
      </form>
    </div>
  </main>
  <script src="./src/scripts/sign-in.js"></script>
</body>
</html>
```

Arquivos Pages

Classroom.HTML

```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="shortcut icon" href="/src/assets/imgs/pencil-icon.png" type="image/x-icon">
  <title>Cenário interativo - VR</title>

  <script src="../scripts/lib/aframe-v1-5-0.min.js"></script>
  <!-- <script src="https://aframe.io/releases/1.4.0/aframe.min.js"></script> -->
  <script src="../scripts/lib/aframe-extras.js"></script>

  <script src="https://cdn.jsdelivr.net/gh/c-frame/aframe-physics-system@v4.2.2/dist/aframe-physics-system.min.js"></script>
  <!-- <script src="https://cdn.jsdelivr.net/gh/c-frame/aframe-extras@7.0.0/dist/aframe-extras.min.js"></script> -->

  <script src="../scripts/components/player-actions.js"></script>
  <script src="../scripts/components/letter-animation.js"></script>
  <script src="../scripts/components/letter-click.js"></script>
  <script src="../scripts/components/name.js"></script>
  <script src="../scripts/components/collider.js"></script>
</head>
```

```
<body>
  <a-scene id="vr-screen" loading-screen="dotsColor: blue; backgroundColor: black">
    <!-- Primeira pessoa -->
    <a-entity movement-controls="constrainToNavMesh: true; controls: checkpoint, gamepad, trackpad, keyboard, nipple;">
      <a-camera wasd-controls="acceleration: 0.00001" id="camera" look-controls="pointerLockEnabled: true" player-actions>
        <a-cursor id="cursor" letter-click></a-cursor>
      </a-camera>
    </a-entity>

    <!-- Sala de Aula -->
    <a-entity id="classroom" scale="1.1 1.1 1.1" gltf-model="url(../assets/models/japanese_classroom.glb )"></a-entity>

    <!-- Lights -->
    <a-entity position="0 50 -10" a-entity id="sun" scale="5 5 5" gltf-model="url(../assets/models/sun.glb)" light="type:
    point; intensity: 2;"></a-entity>
    <a-light type="directional" target="#sun" intensity="0.1"></a-light>
    <a-light type="directional" position="-5 3 -5" intensity="0.1"></a-light>
    <a-light type="directional" position="10 1 20" intensity="0.1"></a-light>
    <a-light type="directional" position="-10 1 20" intensity="0.1"></a-light>

    <!-- Piso -->
    <a-plane src="../assets/textures/classroom-floor.jpg" position="0 0 0" rotation="-90 0 0" width="30" height="30"
    color="tan"></a-plane>
  <!-- Céu -->
  <a-sky color="#89CFF0"></a-sky>

  <!-- Nome na lousa -->
  <a-entity position="1.69 2 -4.5" rotation="0 -90 0">
    <a-text shader="msdf" font="../assets/fonts/WalterTurncoat-Regular.json" scale="1.5 1.5 1.5" value="Seu nome:"
    align="center" color="#FFF" />
  </a-entity>
  <a-entity position="1.69 1.4 -4.5" rotation="0 -90 0">
    <a-text name id="a-text-name" shader="msdf" font="../assets/fonts/WalterTurncoat-Regular.json" scale="1.5 1.5 1.5"
    letter-spacing="10" align="center" color="#FFF" />
  </a-entity>

  <!-- Erros na lousa -->
  <a-entity position="-12.1 2.3 -5.3" rotation="0 90 0">
    <a-text shader="msdf" font="../assets/fonts/WalterTurncoat-Regular.json" scale="1.5 1.5 1.5" value="Erros:"
    align="center" color="#FFF" />
  </a-entity>
  <a-entity position="-12.1 1.42 -4.1" rotation="0 90 0">
    <a-text shader="msdf" font="../assets/fonts/WalterTurncoat-Regular.json" scale="3 3 3" value="0 0 0" letter-spacing="10"
    color="#FFF" />
  </a-entity>
  <a-entity position="-12.1 1.4 -4.12" rotation="0 90 0">
    <a-text id="a-text-errors" shader="msdf" font="../assets/fonts/WalterTurncoat-Regular.json" scale="3 3 3"
    letter-spacing="11.6" color="#FFF" />
  </a-entity>

    <!-- Pshysics -->
    <!-- <a-sphere dynamic-body position="0 1.8 -2" scale="0.2 0.2 0.2"></a-sphere> -->
    <!-- <a-box position="0 0.5 -5" width="3" height="5" depth="1"></a-box> -->
  </a-scene>
  <script src="../scripts/letters-placement.js"></script>
  <!-- <script>
    var scene = document.querySelector('a-scene');
    var cursor = document.querySelector('a-cursor');
    scene.addEventListener('enter-vr', () => {
      cursor.removeAttribute('position');
      cursor.removeAttribute('geometry');
    });
    scene.addEventListener('exit-vr', () => {
      cursor.setAttribute('position', '0 0 -0.05');
      cursor.setAttribute('geometry', 'primitive: ring; radiusOuter: 0.0009; radiusInner: 0.0005" material="color: black;
      shader: flat');
    });
  </script> -->
</body>
```

Arquivos Scripts

Arquivos Components

Collider.JS

```
AFRAME.registerComponent('collider-test', {
  schema: {

  },

  init: function () {
    const element = this.el;
    element.addEventListener('collide', function(e) {
      console.log(e);
      console.log('Player has collided with body #' + e.detail.body.id);

      console.log(e.detail.target.el);  // Original entity (playerEl).
      console.log(e.detail.body.el);    // Other entity, which playerEl touched.
      console.log(e.detail.contact);    // Stats about the collision (CANNON.ContactEquation).
      console.log(e.detail.contact.ni); // Normal (direction) of the collision (CANNON.Vec3).
    });
  },

  update: function () {
    // Do something when component's data is updated.
  },

  remove: function () {
    // Do something the component or its entity is detached.
  },

  tick: function (time, timeDelta) {
    // Do something on every scene tick or frame
  }
});
```

Letter-animation.JS

```
AFRAME.registerComponent('letter-animation', {
init: function() {
    const NAME = sessionStorage.getItem('name') ?? login();
    let splitName = NAME.toLowerCase().normalize("NFD").replace(/[\u0300-\u036f]/g, "").split('');
    const letterEl = this.el;
    const [ x, y, z ] = letterEl.getAttribute('position');
    const maxY = y + (letterEl.getAttribute('scale').x * 0.2);

    // Início da animação
    letterEl.setAttribute('animation', `property: position; dur: 2000; easing: easeInOutSine; from: ${x} ${y} ${z}; to: ${x} $
    {maxY} ${z}`);
    let up = true;

    // Loop de animação
    const animationLoop = () => {
      let animation = `property: position; dur: 2000; easing: easeInOutSine; from: ${x} ${up?maxY:y} ${z}; to: ${x} ${up?
      y:maxY} ${z}`;
      up = !up;
      letterEl.setAttribute('animation', animation);
    };
    letterEl.addEventListener('animationcomplete', animationLoop);
  }
});

// Retornar ao login
function login() {
  const a = document.createElement("a");
  a.style.display = "none";
  a.href = '../../index.html';
  document.body.appendChild(a);
  a.click();
  return 'Lucas';
}
```

Letter-click.JS

```
var errors = 0;

AFRAME.registerComponent('letter-click', {
  schema: {
    // errors: {type: 'number', default: 0},
    alert: {type: 'boolean', default: true},
    start: {default: undefined},
    rAF: {default: window.requestAnimationFrame},
    rAFStop: {default: window.cancelAnimationFrame},
    interval: {default: undefined},
  },
  init: function() {
    this.el.onclick = () => errors += verifyLetterClick();
  },
  tick: function() {
    if(errors >= 3) {
      if(this.data.alert) {
        alert('GAME OVER: Quantidade máxima de erros permitida');
        this.data.alert = false;
      };
      sessionStorage.removeItem('name');
      login();
    };
  },
});

window.addEventListener("gamepadconnected", (e) => {
  const gp = navigator.getGamepads()[e.gamepad.index];
  console.log(e);
  console.log(`Gamepad connected at index ${gp.index}: ${gp.id}. It has ${gp.buttons.length} buttons and ${gp.axes.length}
  axes.`);
  console.log(gp.buttons);

  gameLoop();
});

var start;
var rAF = window.requestAnimationFrame;
var rAFStop = window.cancelAnimationFrame;

window.addEventListener("gamepaddisconnected", function() {
  gamepadInfo.innerHTML = "Waiting for gamepad.";

  rAFStop(start);
});

if(!('GamepadEvent' in window)) {
  // No gamepad events available, poll instead.
  var interval = setInterval(pollGamepads, 500);
}

function pollGamepads() {
  var gamepads = navigator.getGamepads ? navigator.getGamepads() : (navigator.webkitGetGamepads ? navigator.webkitGetGamepads
  : []);
  for(let i = 0; i < gamepads.length; i++) {
    var gp = gamepads[i];
    if(gp) {
      gamepadInfo.innerHTML = "Gamepad connected at index " + gp.index + ": " + gp.id + ". It has " + gp.buttons.length + "
      buttons and " + gp.axes.length + " axes.";
      gameLoop();
      clearInterval(interval);
    }
  }
}

function buttonPressed(b) {
  if(typeof(b) == "object") {
    return b.pressed;
  }
  return b == 1.0;
}
```

```
var pauseClick = false;
var crouchDown = true;
function gameLoop() {
  var gamepads = navigator.getGamepads ? navigator.getGamepads() : (navigator.webkitGetGamepads ? navigator.webkitGetGamepads
  : []);
  if(!gamepads) return;

  // 0 = C / 1 = A / 4 = D / 3 = B
  var gp = gamepads[0];
  if(!pauseClick) {
    if(buttonPressed(gp.buttons[7])) { // R1
      console.log(gp.buttons[7]);
      errors += verifyLetterClick();
    } else if(buttonPressed(gp.buttons[6])) { // R2
      crouchHere(document.querySelector('a-camera'), crouchDown ? 'down' : 'up');
      crouchDown = !crouchDown;
    }
    pauseClick = true;
  }
  pauseClick = gp.buttons.filter(e => e.pressed || e.touched).length > 0 && pauseClick;

  start = rAF(gameLoop);
};

// Retorna à página de Login
function login() {
  const a = document.createElement("a");
  a.style.display = "none";
  a.href = '../../index.html';
  document.body.appendChild(a);
  a.click();
  return 'Lucas';
}
// Última letra escondida
function hiddenLastLetter(splitName) {
  const A_SCENE = document.querySelector('#vr-screen');
  const A_ENTITY = document.createElement('a-entity');
  const A_LETTER = document.createElement('a-entity');
  const A_CROWN = document.createElement('a-entity');

  A_ENTITY.setAttribute('id', 'last-letter');
  A_ENTITY.setAttribute('position', '0.2 0.83 -4.55');
  A_ENTITY.setAttribute('animation', 'property: rotation; loop: true; dur: 3000; easing: linear; from: 0 0 0; to: 0 360 0');

  A_LETTER.setAttribute('class', 'letters');
  A_LETTER.setAttribute('position', '0 0 0');
  A_LETTER.setAttribute('scale', '0.5 0.5 0.5');
  A_LETTER.setAttribute('gltf-model', `url(../assets/models/letter-${splitName[0]}.glb)`);
  A_LETTER.setAttribute('data-letter', splitName[0]);

  A_CROWN.setAttribute('id', 'crown');
  A_CROWN.setAttribute('position', '0 0.12 0');
  A_CROWN.setAttribute('scale', '0.2 0.2 0.2');
  A_CROWN.setAttribute('gltf-model', 'url(../assets/models/crown.glb)');

  A_ENTITY.appendChild(A_LETTER);
  A_ENTITY.appendChild(A_CROWN);
  A_SCENE.appendChild(A_ENTITY);
}

function crouchHere(element, direction) {
  const position = element.getAttribute('position');
  const axisY = position.y;
  let count = axisY;

  let intervalId = setInterval(() => {
    count = (direction === 'down' ? (count - 0.1) : (count + 0.1));
    element.setAttribute('position', {
      x: element.getAttribute('position').x,
      y: direction ==='down' ? (count >= 0.8 ? count : 0.8) : (count <= 1.6 ? count : 1.6),
      z: element.getAttribute('position').z,
    });
    direction === 'down' ? (count <= 0.8 && clearInterval(intervalId)) : (count >= 1.6 && clearInterval(intervalId));
  }, 18);
}
```

Name.JS

```
AFRAME.registerComponent('name', {
  schema: {
    underlineLetters: {type: 'string', default: ''}
  },

  init: function() {
    const NAME = sessionStorage.getItem('name').normalize("NFD").replace(/[^\w\s]/g, '') ?? 'Aluno';
    for(letter of NAME) {
      this.data.underlineLetters += letter === ' ' ? ' ' : '_';
    }
    this.el.setAttribute('value', this.data.underlineLetters);
  }
});
```

Player-actions.JS

```
// Função de agachar (serve para agachar e levantar)
function crouch(key, element, direction) {
  if(key === 'Shift') {
    const position = element.getAttribute('position');
    const axisY = position.y;
    let count = axisY;

    let intervalId = setInterval(() => {
      count = (direction === 'down' ? (count - 0.1) : (count + 0.1));
      element.setAttribute('position', {
        x: element.getAttribute('position').x,
        y: direction ==='down' ? (count >= 0.8 ? count : 0.8) : (count <= 1.6 ? count : 1.6),
        z: element.getAttribute('position').z,
      });
      direction === 'down' ? (count <= 0.8 && clearInterval(intervalId)) : (count >= 1.6 && clearInterval(intervalId));
    }, 18);
  }
}
```

```
// Componente/atributo colocado na câmera para ter as funções de movimento
AFRAME.registerComponent('player-actions', {
  init: function() {
    const element = this.el;

    // Agachar / Levantar
    window.addEventListener('keydown', e => crouch(e.key, element, 'down'));
    window.addEventListener('keyup', e => crouch(e.key, element, 'up'));

    // Pular
    window.addEventListener('keypress', e => {
      if(e.code === 'Space') {
        const position = element.getAttribute('position');
        const axisY = position.y;
        let count = axisY;
        let direction = 'up'

        let intervalId = setInterval(() => {
          count = direction === 'up' ? count + 0.1 : count - 0.1;
          if(count >= 3) direction = 'down';
          element.setAttribute('position', {
            x: element.getAttribute('position').x,
            y: direction === 'up' ? (count <= 3 ? count : 3) : (count >= 1.6 ? count : 1.6),
            z: element.getAttribute('position').z,
          });
          (count <= 1.6 && direction === 'down') && clearInterval(intervalId);
        }, 15);
      }
    });
  }
});
```

Camera.JS

```
const camera = document.querySelector('#camera');

// ponteiro do mouse no centro da tela
document.addEventListener('click', () => {
  document.body.requestPointerLock();
});

// Atualizar a rotação da câmera com o movimento do mouse
document.addEventListener('mousemove', e => {
  if(document.pointerLockElement === document.body) {
    const sensitivity = 0.001;
    camera.components['look-controls'].yawObject.rotation.y -= e.movementX * sensitivity;
    camera.components['look-controls'].pitchObject.rotation.x -= e.movementY * sensitivity;
  }
});
```

Letters-placement.JS

```
const A_SCENE = document.querySelector('#vr-screen');
var alphabet = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'];
const y = 1.3;
let x = -9.2;
const NAME = sessionStorage.getItem('name') ?? login();

// Retornar ao login
function login() {
  const a = document.createElement("a");
  a.style.display = "none";
  a.href = '../../index.html';
  document.body.appendChild(a);
  a.click();
  return 'Lucas';
}

// Confere se dois arrays são iguais
const checkArrays = (a1, a2) => JSON.stringify(a1) === JSON.stringify(a2);

// Deixa o nome minúsculo, sem espaço e sem caracteres especiais
let splitName = NAME.toLowerCase().normalize("NFD").replace(/[\u0300-\u036f]/g, "").split('');
const NAME_LETTERS_AND_POSITIONS = getName(NAME, []);

// Criando letras e posicionando em cima das mesas
for(let i = 0; i < 5; i++) {
  let z = -1.05;
  for(let j = 0; j < 5; j++) {
    let letter = false;
    let chooseLetterName = false;
    for(const EACH_NAME_LETTER_AND_POSITION of NAME_LETTERS_AND_POSITIONS) {
      alphabet = alphabet.filter(l => l != EACH_NAME_LETTER_AND_POSITION[0]);
      chooseLetterName = checkArrays([i, j], EACH_NAME_LETTER_AND_POSITION[1]);
      if(chooseLetterName && NAME_LETTERS_AND_POSITIONS.indexOf(EACH_NAME_LETTER_AND_POSITION) != NAME_LETTERS_AND_POSITIONS.
      length - 1) letter = EACH_NAME_LETTER_AND_POSITION[0];
    }
    letter = letter || randomLetter(alphabet);

    const A_ENTITY = document.createElement('a-entity');
    A_ENTITY.setAttribute('class', 'letters');
    A_ENTITY.setAttribute('position', `${x} ${y} ${z}`);
    A_ENTITY.setAttribute('scale', `0.8 0.8 0.8`);
    A_ENTITY.setAttribute('rotation', `0 90 0`);
    A_ENTITY.setAttribute('gltf-model', `url(../assets/models/letter-${letter}.glb)`);
    A_ENTITY.setAttribute('animation__rotation', `property: rotation; loop: true; dur: 3000; easing: linear; from: 0 0 0; to:
    0 360 0`);
    A_ENTITY.setAttribute('letter-animation', 'true');
    A_ENTITY.setAttribute('data-letter', letter);

    A_SCENE.appendChild(A_ENTITY);
    z -= 1.74 ;
  }
  x += 1.83;
};
```

```
// Escolha aleatória de letras
function randomLetter(letters) {
  const randomPercent = Math.random() * 100;
  const n_of_l = letters.length;
  for(let i = 0; i <= n_of_l; i++) {
    if(randomPercent <= ((100 / n_of_l) * (i + 1))) {
      return letters[i];
    }
  }
};

// Divide as letras do nome nas posições da sala
function getName(name, letters_of_name_position) {
  const NAME_LETTERS = name.toLowerCase().normalize("NFD").replace(/[\u0300-\u036f]/g, "").split('').filter(letter => letter
  != ' ');
  NAME_LETTERS.forEach((letter, index) => {
    if(letters_of_name_position[index] === undefined) {
      letters_of_name_position[index] = [letter, [Math.round(Math.random() * 4), Math.round(Math.random() * 4)]];

      if(index > 0) {
        for(const EACH_NAME_LETTER_AND_POSITION of letters_of_name_position) {
          if(!(letters_of_name_position[index][1] === EACH_NAME_LETTER_AND_POSITION[1])) {
            if(checkArrays(letters_of_name_position[index][1], EACH_NAME_LETTER_AND_POSITION[1])) {
              letters_of_name_position.pop();
              return getName(name, letters_of_name_position);
            }
          }
        }
      }
    }
  });
  return letters_of_name_position;
}
```

Sign-in.JS

```
const form = document.querySelector('form');
const input = form.querySelector('#name');

const saveName = () => sessionStorage.setItem('name', input.value.trim());

function sigin() {
  const a = document.createElement("a");
  a.style.display = "none";
  a.href = './src/pages/classroom.html';
  document.body.appendChild(a);
  a.click();
};

form.onsubmit = e => {
  e.preventDefault();
  saveName();
  sigin();
};
```

Arquivos Styles

Index.CSS

```css
main {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 100vw;
  height: 100vh;
  background-image: url(../assets/imgs/galaxy.jpg);
  background-repeat: no-repeat;
  background-size: cover;
}

.glass-container {
  width: min(90%, 600px);
  height: clamp(10px, 90%, 350px);
  background-color: rgba(255, 255, 255, 0.05);
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.25);
  backdrop-filter: blur(10px);
  border-radius: 15px;
  padding: 30px 15px;
}

form {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
  width: 100%;
  height: 100%;
}

form > h1 {
  text-align: center;
  font-size: 3.5rem;
  color: #FFF;
}
```

```css
/* Input */
.input-container {
  position: relative;
  margin: 0 auto 15px;
  width: 90%;
}

.input-container input[type="text"] {
  font-size: 2rem;
  width: 100%;
  border: none;
  border-bottom: 2px solid #ccc;
  color: #FFF;
  padding: 5px 0;
  background-color: transparent;
  outline: none;
}

.input-container .label {
  font-size: 1.6rem;
  position: absolute;
  top: 0;
  left: 0;
  color: #ccc;
  transition: all 0.3s ease;
  pointer-events: none;
}

.input-container input[type="text"]:focus ~ .label,
.input-container input[type="text"]:valid ~ .label {
  top: -25px;
  font-size: 2rem;
  color: #7d13ee;
}
```

```css
.input-container .underline {
  position: absolute;
  bottom: 0;
  left: 0;
  height: 2px;
  width: 100%;
  background-color: #7d13ee;
  transform: scaleX(0);
  transition: all 0.3s ease;
}

.input-container input[type="text"]:focus ~ .underline,
.input-container input[type="text"]:valid ~ .underline {
  transform: scaleX(1);
}

.input-container input[type="text"]::placeholder {
  opacity: 0;
  transition: all 0.3s ease;
}

.input-container input[type="text"]:focus::placeholder,
.input-container input[type="text"]:valid::placeholder {
  opacity: 1;
}
```

```css
/* Button */
.sigin {
  position: relative;
  background: transparent;
  padding: 0px;
  border: none;
  cursor: pointer;
  outline-offset: 4px;
  outline-color: deeppink;
  transition: filter 250ms;
  -webkit-tap-highlight-color: rgba(0, 0, 0, 0);
}

.shadow {
  position: absolute;
  top: 0;
  left: 0;
  height: 100%;
  width: 100%;
  background: hsl(226, 25%, 69%);
  border-radius: 8px;
  filter: blur(2px);
  will-change: transform;
  transform: translateY(2px);
  transition: transform 600ms cubic-bezier(0.3, 0.7, 0.4, 1);
}
```

```css
.edge {
  position: absolute;
  top: 0;
  left: 0;
  height: 100%;
  width: 100%;
  border-radius: 8px;
  background: linear-gradient(
    to right,
    hsl(248, 39%, 39%) 0%,
    hsl(248, 39%, 49%) 8%,
    hsl(248, 39%, 39%) 92%,
    hsl(248, 39%, 29%) 100%
  );
}

.front {
  display: block;
  position: relative;
  border-radius: 8px;
  background: #7d13ee;
  /* background: #6a5bcd; */
  padding: 16px 32px;
  color: white;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Oxygen,
    Ubuntu, Cantarell, "Open Sans", "Helvetica Neue", sans-serif;
  font-weight: 600;
  text-transform: uppercase;
  letter-spacing: 1.5px;
  font-size: 1.2rem;
  transform: translateY(-4px);
  transition: transform 600ms cubic-bezier(0.3, 0.7, 0.4, 1);
}
```

```css
.sigin:hover {
  filter: brightness(110%);
}

.sigin:hover .front {
  transform: translateY(-6px);
  transition: transform 250ms cubic-bezier(0.3, 0.7, 0.4, 1.5);
}

.sigin:active .front {
  transform: translateY(-2px);
  transition: transform 34ms;
}

.sigin:hover .shadow {
  transform: translateY(4px);
  transition: transform 250ms cubic-bezier(0.3, 0.7, 0.4, 1.5);
}

.sigin:active .shadow {
  transform: translateY(1px);
  transition: transform 34ms;
}

.sigin:focus:not(:focus-visible) {
  outline: none;
}
```

Reset.CSS

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  font-size: 1rem;
  font-weight: normal;
}

html {
  font-size: 62.5%;
  min-width: 250px;
}
```