UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Kevin Rafael Velez Bernal

DINÂMICA DIGITAL GAMIFICADA PARA SELEÇÃO DE MÉTRICAS ÁGEIS DE SOFTWARE

Florianópolis

2024

Kevin Rafael Velez Bernal

DINÂMICA DIGITAL GAMIFICADA PARA SELEÇÃO DE MÉTRICAS ÁGEIS DE SOFTWARE

Trabalho de Conclusão de Curso submetido ao Curso de Bacharelado em Sistemas de Informação para a obtenção do Grau de Bacharel em Sistemas de Informação. Orientador: Prof. Dr. Jean Carlo Rossa Hauck

Florianópolis

2024

RESUMO

O gerenciamento de projetos de software tem sido normalmente realizado com ajuda de métodos ágeis que contribuem de diversas formas para a medição nos projetos. Existem diversas métricas tipicamente utilizadas nos métodos ágeis e a escolha de quais métricas utilizar pode ser complexa. Nesse sentido, a gamificação pode contribuir para que esse processo de seleção das métricas mais adequadas não seja ignorado ou feito às pressas. Por isso, foi desenvolvida a dinâmica gamificada Metrics Poker, que auxilia na seleção de métricas ágeis em projetos de software. Entretanto, a dinâmica foi desenvolvida utilizando artefatos físicos, o que dificulta a sua aplicação em equipes distribuídas. O presente trabalho propõe o desenvolvimento de uma versão digital da dinâmica Metrics Poker.Este trabalho propõe uma solução digital para a dinâmica gamificada Metrics Poker, originalmente criada para auxiliar equipes ágeis na escolha de métricas de software. A solução digital foi desenvolvida utilizando tecnologias APIs REST no backend como Supabase com PostgreSQL como banco de dados com estrutura relacional modelada para persistência de sessões, métricas, participantes e histórico. A autenticação foi feita com o Supabase via Google OAuth2, Vue.js para frontend, e JavaScript como linguagem principal.Design e interface das cartas e o layout simulam um jogo de poker, incentivando o engajamento visual e a clareza nas escolhas. O principal objetivo da solução é facilitar a seleção de métricas ágeis de forma colaborativa e interativa, com acessibilidade para equipes remotas. A proposta foi implementada seguindo uma abordagem iterativa e incremental, começando pela modelagem do backend. Testes foram realizados para garantir segurança, funcionalidade e usabilidade. A aplicação foi projetada para atender aos requisitos ágeis e proporcionar uma experiência colaborativa para equipes de desenvolvimento de software.

Sumário

1. Introdução..... 5
1.1. Contextualização..... 5
1.2. Objetivos..... 7
1.2.1. Objetivos Gerais..... 7
1.2.2. Objetivos Específicos..... 7
1.3. Método de Pesquisa..... 7
1.4. Medição de software..... 9
1.5. Dinâmica para seleção de métricas..... 11
1.6. Processo ENgAGED..... 14
2. Trabalhos Relacionados..... 18
2.1. Ensino de Engenharia de Software com SimulES-W..... 20
2.2. Sharkworld..... 21
2.3. Problems and programmers..... 22
2.4. Discussão..... 24
3. Proposta..... 26
3.1. Concepção da Dinâmica..... 26
3.2. Requisitos..... 28
3.3. Protótipos de tela..... 30
3.4. Arquitetura da solução..... 35
3.5. Tecnologias utilizadas..... 36
3.5.1. Front-end..... 37
3.5.2. Back-end..... 38
3.5.3. REST..... 39
3.6. Banco de Dados..... 41
3.6.1. Tabelas..... 42
3.6.2. Relação entre tabelas..... 44
3.7. Desenvolvimento..... 46
3.7.1. Backend..... 47
3.7.1.1. Configuração da API..... 47
3.7.1.2. Estrutura da API..... 48
3.7.1.3. Endpoints da API..... 48
3.7.1.4. Documentação da API..... 49
3.7.1.5. Autenticação..... 49
3.7.2. Frontend..... 52
3.7.2.1. RF01 - O sistema deve permitir ao Dealer se cadastrar com seu e-mail... 52
3.7.2.2. RF02 - O sistema deve permitir ao usuário visualizar o tabuleiro..... 54
3.7.2.3. RF03 - O sistema deve permitir ao usuário visualizar as regras da dinâmica 57
3.7.2.4. RF04 - O sistema deve permitir ao usuário a visualização das cartas..... 57
3.7.2.5. RF05 - O sistema deve permitir aos usuários a informação de que a dinâmica acabou..... 60
3.7.2.6. RF06 - O sistema deve criar uma sessão de jogo para a partida iniciada. 62
3.7.2.7. RF07 - O sistema deve permitir a cada um dos membros da equipe

cadastrar um nome fictício....63
3.7.3. Fluxo de funcionamento.... 63
3.7.3.1. Criação da Sessão.... 63
3.7.3.2. Estágios da dinâmica....64
3.7.3.2.1. Não iniciado (not_started)....64
3.7.3.2.2. Iniciado (started)....64
3.7.3.2.3. Votação (select_metrics).... 64
3.7.3.2.4. Finalizado (ended)....65
4. Conclusão.... 66
5. Referências.... 68
6. Apêndices....73
6.1. Desenvolvimento.... 73
6.1.1. Testes da API....73
6.2. Artigo.... 80
6.3. Código Fonte.... 88

1. Introdução

1.1. Contextualização

Os métodos ágeis de desenvolvimento de software têm sido largamente utilizados pelas organizações de desenvolvimento de software (DIGITAL, 2023). Dentre os métodos ágeis, o mais utilizado atualmente tem sido o Scrum, com 63% dos usuários de métodos ágeis (DIGITAL, 2023). Os principais benefícios de um time que utiliza métodos ágeis são o melhor alinhamento com o negócio e melhora na colaboração (DIGITAL, 2023).

Nos métodos tradicionais, os projetos são normalmente gerenciados com apoio de um processo de medição de software. Nesse sentido, medição consiste na atribuição de um número a um determinado atributo, e serve para descrever o grau em que esse atributo se aplica a um artefato (ABBAD-ANDALOUSSI, 2023). Nos métodos ágeis, métricas ágeis são utilizadas para avaliar o processo de desenvolvimento de software (DAVIS, 2015). Métricas de software estão relacionadas à avaliação quantitativa de características e propriedades de produtos de software (ISO/IEC/IEEE 24765, 2017).

O processo de Medição de Software consiste no planejamento, coleta das medidas, avaliação e análise (ISO/IEC/IEEE…, 2009). No processo de planejamento há a definição de metas, a qual envolve identificar o que se deseja alcançar com a medição; há também a seleção de métricas relevantes e o estabelecimento do plano de medição, que descreve a forma como será coletada, analisada e relatada a métrica escolhida. Na avaliação ocorre a coleta de dados em si, baseada nas métricas e o plano de medição, que pode incluir ferramentas automatizadas ou revisões manuais que ajudam a identificar defeitos. Por fim, a análise dos dados obtidos que identifiquem áreas para melhoria, assim como também a interpretação e relatório que apresente de forma clara as descobertas feitas (ISO/IEC/IEEE…, 2009).

Para a medição de projetos ágeis existem abordagens distintas. Dentre as diversas métricas existentes, têm sido principalmente utilizadas a velocidade (36%) e a entrega de valor (29%) por projetos que adotam métodos ágeis

(DIGITAL, 2023). Para gerenciar o atendimento dos objetivos organizacionais de projetos, a abordagem OKR (*Objective Key Results*) também tem sido utilizada. Os OKRs são utilizados em empresas como a Google para conseguir medir o alcance de objetivos em projetos de software (GOOGLE, 2020).

Por existir uma variedade de contextos e escopos de projetos de software, há uma dificuldade de selecionar qual métrica ágil é mais adequada para um projeto específico (REIS, 2023). Assim, uma dinâmica gamificada foi desenvolvida para facilitar a seleção das métricas (REIS, 2023). A dinâmica gamificada foi desenvolvida por Reis (2023) em um trabalho anterior que, com a ajuda de mapeamentos sistemáticos da literatura (MSL), selecionou métricas de maneira dinâmica a partir de uma pesquisa feita com líderes de projetos de todas as regiões do Brasil.

A dinâmica funciona da seguinte forma: do lado esquerdo do tabuleiro é colocada uma carta com a dor que o time possui. Na parte de cima há cinco espaços disponíveis para colocar as cartas de um determinado grupo de métricas, que serão mencionados e explicados posteriormente. Na parte de baixo do tabuleiro há o mesmo espaço disponível para outras cinco cartas de outro grupo de métricas. Cada um dos integrantes do time recebe 3 fichas - preferência pessoal, relevância para a dor, facilidade de coleta - e coloca cada uma delas virada para baixo, em uma das cartas do tabuleiro. As duas métricas mais votadas são as selecionadas para a dor em questão (REIS, 2023).

No entanto, equipes de desenvolvimento de software podem estar distribuídas em diferentes localidades, de acordo com Digital (2023). Na pesquisa feita, surpreendentes 91% dos participantes relataram que suas equipes estão totalmente distribuídas, sendo a maior porcentagem já encontrada nos 17 anos em que a pesquisa vem sendo realizada, muitas vezes desenvolvendo suas atividades somente de maneira remota (DIGITAL, 2023), e, nesses casos, a dinâmica física não é aplicável. Por isso, existe a necessidade do desenvolvimento de uma versão digital da dinâmica para seleção de métricas ágeis em projetos de software.

1.2. Objetivos

1.2.1. Objetivos Gerais

O objetivo geral do presente projeto é desenvolver uma dinâmica gamificada digital online para apoio à seleção de métricas ágeis. A dinâmica foi inicialmente concebida no trabalho de Reis (2023).

1.2.2. Objetivos Específicos

Para que o objetivo geral seja alcançado, faz-se necessário que os seguintes objetivos específicos sejam alcançados:

- Sintetizar a teoria da área de seleção de métricas ágeis em termos de princípios, o que abrange e principalmente como é feito o processo de medição de software.
- Levantar o estado da arte em relação a jogos digitais para ensino de conceitos de medição de software.
- Desenvolver uma dinâmica digital online para a seleção de métricas ágeis que possam ser utilizadas em times distribuídos de desenvolvimento de projetos de software.

1.3. Método de Pesquisa

Do ponto de vista da natureza da pesquisa a ser empreendida, este trabalho se classifica como uma pesquisa aplicada (ou tecnológica), que tem por objetivo gerar produtos e/ou processos inéditos, com finalidades imediatas, com base em conhecimentos prévios. Quanto aos objetivos da pesquisa, este trabalho se caracteriza como uma pesquisa exploratória, pois visa proporcionar maior familiaridade com o problema investigado a fim de torná-lo explícito.

A metodologia de pesquisa deste projeto é dividida nas seguintes etapas:

Etapa 1. Análise da fundamentação teórica: Será realizada uma análise sobre a área de medição de software:

Atividade 1.1: Analisar a área de métricas ágeis utilizadas dentro de métodos ágeis tal como o Scrum, levantando o estado da arte de diferentes métricas ágeis;

Atividade 1.2: Analisar e levantar o estado da arte da gamificação dentro do gerenciamento de projetos de software;

Etapa 2. Desenvolvimento da dinâmica gamificada digital online para apoio de levantamento de métricas ágeis para o desenvolvimento de projetos de software

Atividade 2.1: Analisar o projeto da dinâmica de seleção de métricas ágeis;

Atividade 2.2: Explicação de como a dinâmica foi concebida;

Atividade 2.3: Implementação da dinâmica online utilizando Vue.js, incluindo a codificação;

Etapa 3. Avaliação do produto

Atividade 3.1: Teste das funcionalidades do produto criado.

Fundamentação Teórica

Neste capítulo são apresentados os principais conceitos relacionados a este trabalho. Inicialmente são apresentados os conceitos de medição de software, seguidos da apresentação do jogo Metrics Poker e, por fim, a abordagem ENgAGED, que embasa este trabalho.

1.4. Medição de software

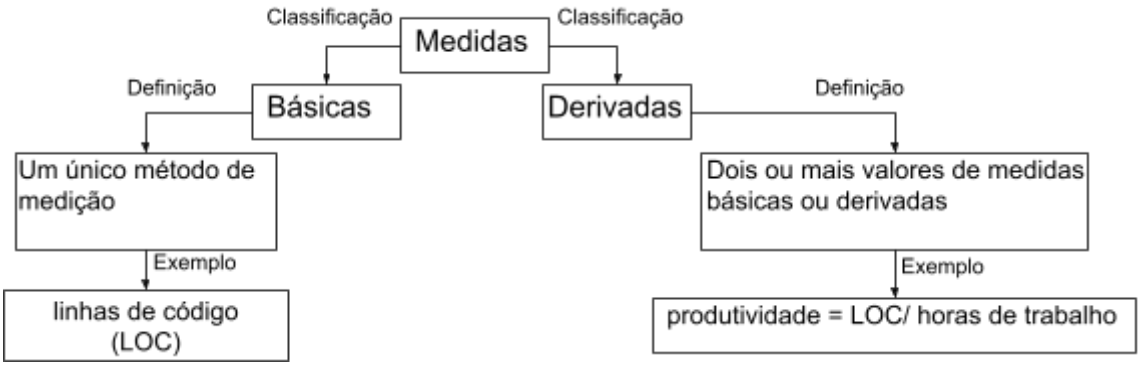
Para explicar medição de software, é importante antes trazer uma possível definição de medição. Medição tem como objetivo determinar o valor de uma medida, através de um conjunto de operações (ISO/IEC/IEE 15939, 2017). A sequência lógica de operações usadas para quantificar um atributo são os métodos de medição, que podem ser (INMETRO, 2012):

- subjetivo: o julgamento humano é envolvido na quantificação;
- objetivo: as regras numéricas são baseadas para a quantificação.

O procedimento de medição é feito de acordo com um determinado método e com um conjunto de operações descritas especificamente e utilizadas em um determinado contexto (INMETRO, 2012).

Para o processo de medição de software existe um propósito e um resultado esperado. O propósito do processo de medição apoia os objetivos da organização na coleta, análise, armazenamento e relato dos dados relativos ao produto desenvolvido (SOFTEX, 2023). A organização que possui um processo padrão consegue comparar dados históricos de projetos e, a partir dela, é possível extrair informações e estabelecer boas práticas, além de identificar possíveis problemas (SOFTEX, 2016).

Figura 1 - Método de medição



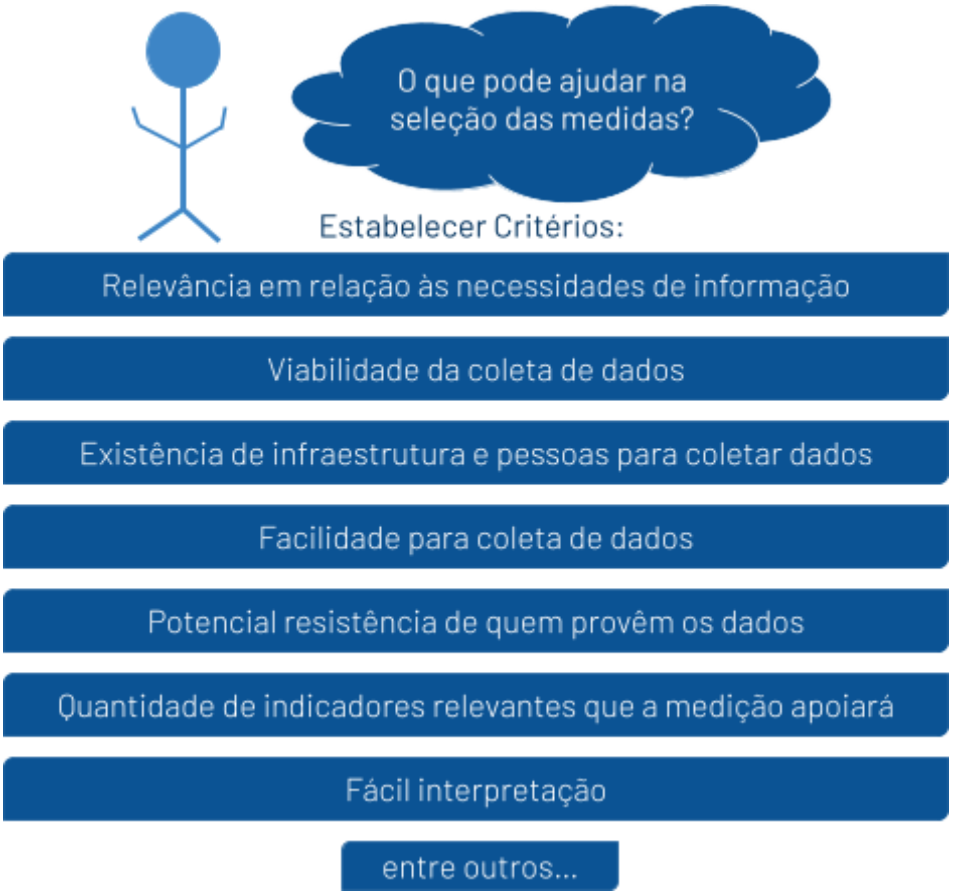
Fonte: adaptado do ISO/IEC 15939:2017

A maturidade de outros processos organizacionais influenciam no processo de medição, e por isso é feito de maneira progressiva (SOFTEX, 2016).

O que se espera alcançar é obtido de acordo com os objetivos do negócio no que diz respeito a processos, produtos e recursos. Recursos podem ser ferramentas e pessoas. Produtos podem ser os objetivos alcançados com a execução do projeto de software, como uma aplicação, o código fonte, casos de teste, projetos, programas. Processos tais como atividades de análise (SOFTEX, 2016).

O processo de medição inicia ao deixar bem claro qual o objetivo da medição, onde se espera chegar, e o que se espera a partir da medição. Em seguida, deve-se definir quais medidas podem contribuir para alcançar tal objetivo (Figura 2).

Figura 2 - Estabelecimento de critérios de medidas



Fonte: adaptado de SOFTEX, 2016.

A quantidade de seleção de medidas não necessariamente precisa ser grande, pois pode haver o risco de que seja necessário muito esforço para a coleta. Por outro lado, escolher poucas medidas pode levar a ignorar aspectos que poderiam ser relevantes (SOFTEX, 2016).

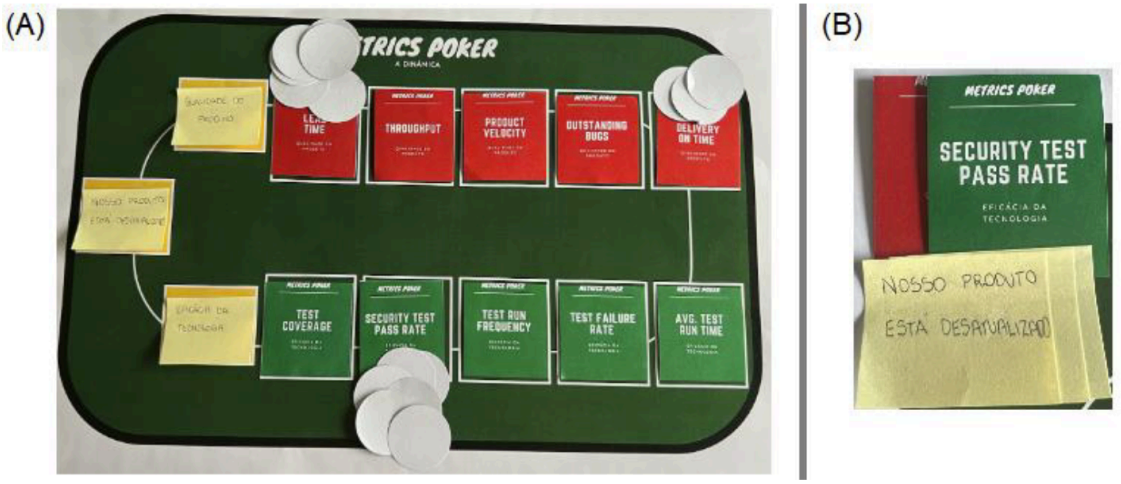
Uma vez selecionadas as medidas, elas devem ser documentadas na forma de um plano de medição, com: nome, descrição, unidade de medida e relação com a necessidade de medição (SOFTEX, 2016). O documento do plano de medição deve sofrer constante atualização, caso ocorram alterações na necessidade de informações.

1.5. Dinâmica para seleção de métricas

A dinâmica gamificada para seleção de métricas utilizada é o Metrics Poker. A versão física foi desenvolvida no trabalho: Metrics Poker - Uma

Dinâmica para Apoiar o Ensino de Medição em Projetos de Software, apresentado no CBIE 2023 (REIS *et al*., 2023). Basicamente, a dinâmica funciona da seguinte forma (Figura 3): alunos são separados em grupos de 3 até 8 participantes, que é o número aproximado de pessoas encontrado em um time de desenvolvedores. Uma pessoa é escolhida como *Dealer,* que vai se certificar do cumprimento das regras da dinâmica por parte dos outros participantes, os restantes são os Membros da Equipe (REIS ET. ALL. 2023).

Figura 3 - Metrics Poker tabuleiro



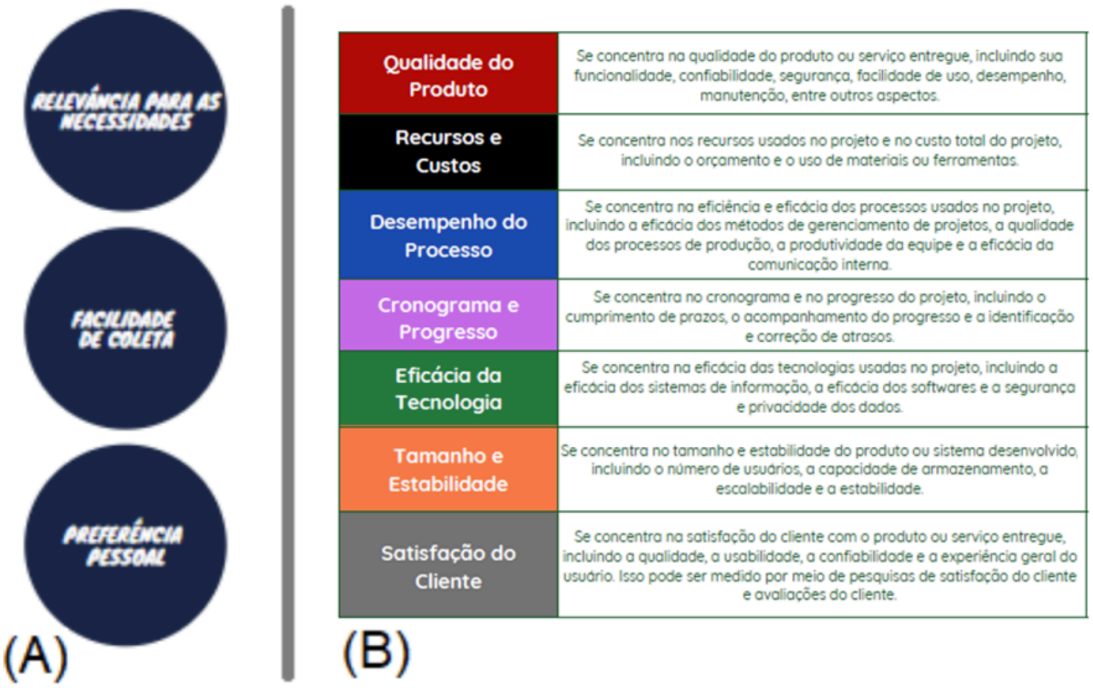
Fonte: REIS *et al*., 2023

O objetivo da dinâmica é que, com base nas necessidades da equipe, seja possível identificar as métricas que são adequadas, sendo que há dois turnos. No primeiro, os Membros da Equipe selecionam quais os grupos de métricas, conforme a figura 3 (B) (Qualidade do Produto, Recursos e Custos, Desempenho do Processo, Cronograma e Progressão, Eficácia da Tecnologia, Tamanho e Estabilidade, Satisfação do Cliente). No segundo turno, a partir dos grupos de métricas escolhidos no primeiro, é hora de selecionar uma métrica específica dentro do grupo de métricas. Sendo necessário para vencer a dinâmica a escolha de pelo menos 2 (duas) métricas que façam sentido para o problema em questão (Reis *et al*., 2023).

Os materiais necessários são:

- Mesa: é o tabuleiro onde serão colocadas as cartas ou fichas, conforme a figura 4(A).
- Cartas das Métricas: Coloridas, sendo cada grupo de uma cor, com nome e descrição de cada métrica 4(B).
- Fichas de “aposta” em cada métrica: Cada uma com um nome em um verso conforme figura 4(A) nos mostra.
- Tabela dos Grupos de Métricas: com as cores, título e descrição de cada grupo de métricas, conforme na figura 4 (B)
- Regras: Detalhamento em um documento com informações da execução da dinâmica.

Figura 4 - Fichas (A) e Grupos de Métricas (B)



Qualidade do Produto	Se concentra na qualidade do produto ou serviço entregue, incluindo sua funcionalidade, confiabilidade, segurança, facilidade de uso, desempenho, manutenção, entre outros aspectos.
Recursos e Custos	Se concentra nos recursos usados no projeto e no custo total do projeto, incluindo o orçamento e o uso de materiais ou ferramentas.
Desempenho do Processo	Se concentra na eficiência e eficácia dos processos usados no projeto, incluindo a eficácia dos métodos de gerenciamento de projetos, a qualidade dos processos de produção, a produtividade da equipe e a eficácia da comunicação interna.
Cronograma e Progresso	Se concentra no cronograma e no progresso do projeto, incluindo o cumprimento de prazos, o acompanhamento do progresso e a identificação e correção de atrasos.
Eficácia da Tecnologia	Se concentra na eficácia das tecnologias usadas no projeto, incluindo a eficácia dos sistemas de informação, a eficácia dos softwares e a segurança e privacidade dos dados.
Tamanho e Estabilidade	Se concentra no tamanho e estabilidade do produto ou sistema desenvolvido, incluindo o número de usuários, a capacidade de armazenamento, a escalabilidade e a estabilidade.
Satisfação do Cliente	Se concentra na satisfação do cliente com o produto ou serviço entregue, incluindo a qualidade, a usabilidade, a confiabilidade e a experiência geral do usuário. Isso pode ser medido por meio de pesquisas de satisfação do cliente e avaliações do cliente.

Fonte: REIS et al., 2023.

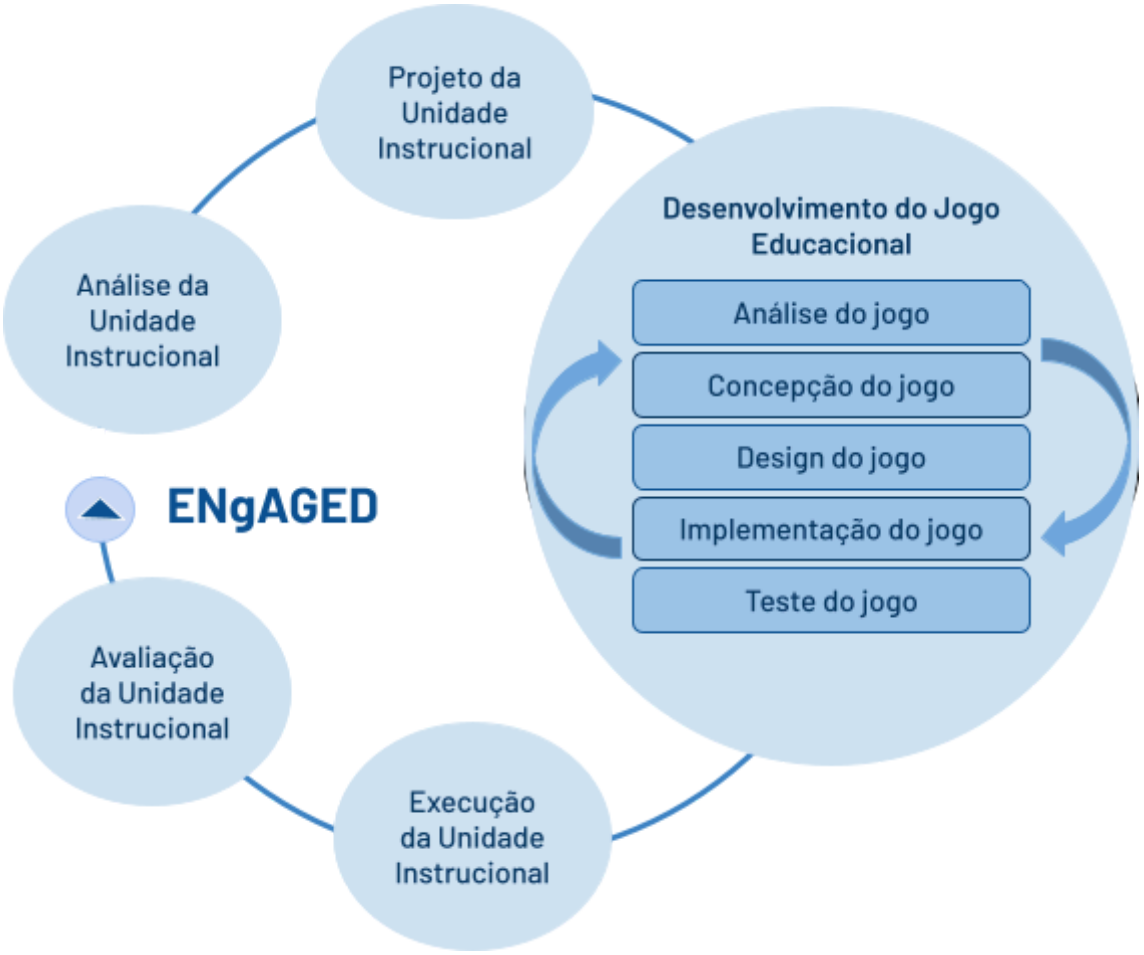
Um exemplo de partida da dinâmica é o seguinte: 2 problemas são apresentados aos membros da equipe, problemas estes que fazem parte do escopo de gerenciamento de um projeto de software. Para conseguir resolvê-los, a equipe vai em busca de selecionar as melhores métricas, os papéis de Dealer e de Membros da Equipe são definidos entre os participantes

da dinâmica. A equipe tem à disposição as regras, o tabuleiro e a tabela de grupos de métricas. Cada um dos membros da equipe recebe as cartas (que são as métricas), além de receber três fichas, cada uma com um critério de seleção. A dinâmica inicia com os membros do grupo escolhendo quais os grupos de métricas farão mais sentido selecionar para o problema proposto; em seguida, no segundo turno, todos apostam suas fichas, viradas para baixo, nas cartas que acham pertinentes (REIS *et al*., 2023).

1.6. Processo ENgAGED

Para o desenvolvimento de dinâmicas gamificadas digitais existe o processo ENgAGED, que vem do inglês *EducatioNAl GamEs Development*, que nada mais é que um processo de desenvolvimento de jogos, com o objetivo de utilizar no ensino superior de computação (BATTISTELLA, 2016).

Figura 5 - ENgAGED - Educational Games Development



Fonte: adaptado de BATTISTELLA (2016).

Dentre as fases do processo de ENgAGED, a fase 1 é Análise de Unidade Instrucional, que envolve: especificação da unidade instrucional, caracterização do público alvo, definição dos objetivos de desempenho (BATTISTELLA e WANGENHEIM, 2015).

A fase 2 vem a ser o Projeto da Unidade Instrucional, onde é necessário: definir a avaliação do aluno, definir conteúdo da estratégia instrucional, decidir pelo desenvolvimento ou utilizar jogo desenvolvido, revisar modelo de avaliação da dinâmica (BATTISTELLA, 2016).

A fase 3 diz respeito ao Desenvolvimento do Jogo Educacional. Nessa fase será feita a Análise do Jogo: que envolve o levantamento de requisitos da dinâmica sobre quais serão as funcionalidades e como será a interface da dinâmica; Concepção da dinâmica: implica na descrição das características da dinâmica; Design da dinâmica: ocorre a definição da linguagem de programação ou *game engine,* produção de ilustrações ou imagens dos elementos da dinâmica, modelagem da dinâmica; Implementação do Jogo: produção em código dos elementos da dinâmica; Testes da dinâmica: fazer validação de feedback e verificar possíveis erros (BATTISTELLA, 2016). Em resumo a análise da dinâmica, concepção da dinâmica, design da dinâmica, implementação da dinâmica e, por fim, o teste do mesmo, conforme mostra a figura 6.

Figura 6 - Fase 3 - Desenvolvimento do Jogo Educacional

Fonte: Elaborado com DALL-E 3 (2024), adaptado de BATTISTELLA (2016).

Fase 4 - Execução da unidade instrucional: planejamento da execução da dinâmica, o qual envolve a definição do material necessário para a execução da dinâmica em si e determinação de onde será feito, qual o local, data e hora da execução da dinâmica (BATTISTELLA, 2016). Instalação da dinâmica digital, essa tarefa depende muito da arquitetura da dinâmica, no caso de ser cliente-servidor, então a dinâmica precisa ser disponibilizado neste momento no servidor, caso seja um aplicativo para *smartphones* deve-se instalar o aplicativo nos celulares dos alunos que vão jogar e no caso de ser *stand-alone* então é preciso que se instale a dinâmica nos computadores utilizados pelos alunos que irão executar a dinâmica (BATTISTELLA, 2016). Por fim, a execução da dinâmica em sala de aula é a última atividade desta fase (BATTISTELLA, 2016).

Fase 5 - Avaliação da Unidade Instrucional, que envolve a condução da avaliação: a coleta de dados é feita com base na atividade definida na fase 2, *revisar modelo de avaliação da dinâmica* com a ajuda de instrumentos que

permitam a condução da avaliação (BATTISTELLA, 2016). Por fim, na última atividade, a análise dos dados coletados na avaliação, essa revisão deve ser feita, também, baseada na atividade definida na fase 2, *revisar modelo de avaliação da dinâmica* (BATTISTELLA, 2016).

2. Trabalhos Relacionados

Neste capítulo são apresentados os trabalhos relacionados à presente pesquisa. São considerados trabalhos relacionados aqueles que apresentem jogos educacionais voltados para a área de Engenharia de Software, a qual é a mesma em que se encontra a dinâmica gamificada deste trabalho.

Os trabalhos relacionados foram inicialmente encontrados na Revisão Sistemática de Literatura: "*Games for Teaching Computing in Higher Education – A Systematic Review*" que apresenta uma revisão dos jogos utilizados para o ensino de computação no ensino superior, destacando a diversidade de abordagens e os benefícios do uso de jogos no processo educativo (BATTISTELLA & VON WANGENHEIM, 2016).

A revisão de Batistela & Wangenheim (2016) destaca que o uso de jogos no ensino superior pode aumentar o engajamento dos alunos, melhorar a retenção de conhecimento e fornecer uma experiência de aprendizado mais prática e interativa. Além disso, a revisão sugere que os jogos educacionais são eficazes em promover habilidades de resolução de problemas, trabalho em equipe e pensamento crítico, essenciais para a formação de profissionais de computação (BATTISTELLA & VON WANGENHEIM, 2016).

A revisão do trabalho analisou 107 jogos, dos quais 10 foram inicialmente selecionados para este trabalho por sua relevância e potencial similaridade. A Tabela 1 apresenta os 10 trabalhos inicialmente selecionados. Após uma análise mais aprofundada, três jogos foram escolhidos por sua maior similaridade com o estudo em questão.

Tabela 1 - Lista dos 10 jogos inicialmente escolhidos

Jogo	**Título**	**Referência**
SimulES-W	*Teaching Software Engineering with SimulES-W*	(MONSALVE, WERNECK & LEITE, 2011)
SimSE	*SimSE: an interactive simulation game for software engineering education*	(NAVARRO & HOEK, 2004)

Problems and Programmers	*Problems and Programmers: an educational software engineering card game*	(BAKER, NAVARRO & HOEK, 2003)
Deliver!	*DELIVER! – An educational game for teaching Earned Value Management in computing courses*	(VON WANGENHEIM, SAVI & BORGATTO, 2012)
SDM	*SDM – An educational game for Software Engineering*	(KOHWALTER; CLUA & MURTA, 211)
SimVBSE	*SimVBSE : Developing a Game for Value-Based Software Engineering*	(JAIN & BOEHM ,2006)
Sharkworld	*Make It Fun Or Real – Design Dilemmas And Their Consequences On The Learning Experience.*	(HARTEVELD; BEKEBREDE; LO; PLOMBER & JORDAAN, 2011)
Software Risk Management Game	*Using games in Software Engineering education to teach Risk Management*	(TARAN, G., 2007)
AMEISE	*AMEISE – A media education initiative for Software Engineering*	(MITTERMEIR ET AL., 2003)
Detective Game – what killed the project?	*Development of an educational game for teaching project management in undergraduate computing programs*	(SOARES, G.M. & RAUSIS, B.Z., 2011)

Os três jogos revisados variam em termos de objetivos educacionais, mecânicas de jogo e públicos-alvo. Eles são utilizados para ensinar uma ampla gama de conceitos de computação, desde a programação básica até

tópicos mais avançados, como redes e segurança (BATTISTELLA & VON WANGENHEIM, 2016). Os trabalhos relacionados são apresentados nas próximas seções.

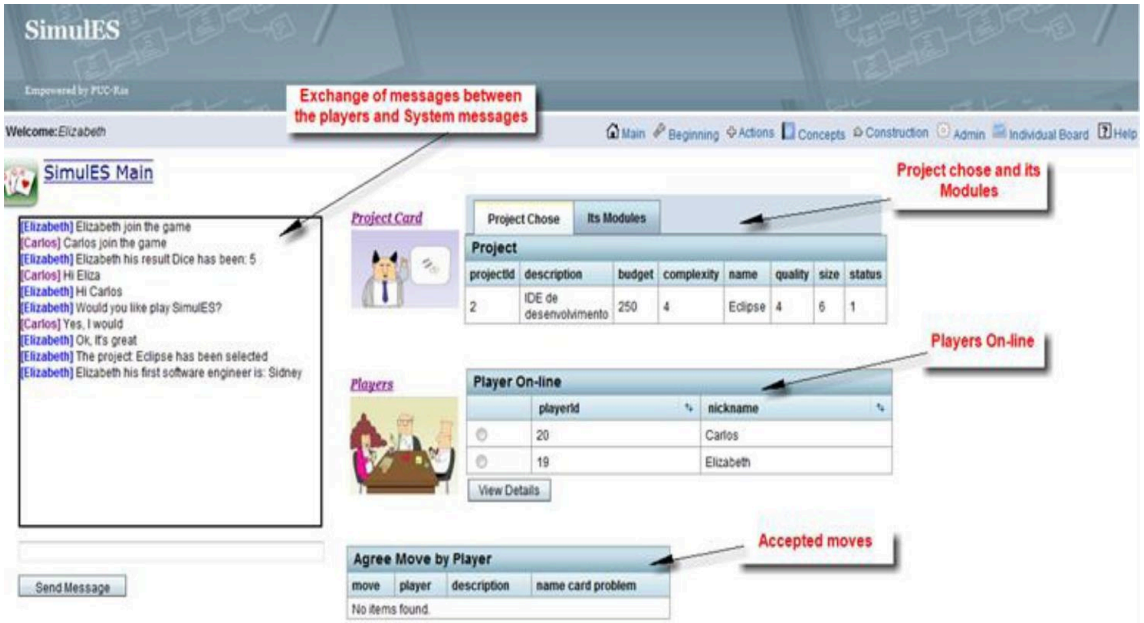
2.1. Ensino de Engenharia de Software com SimulES-W

O artigo "Teaching Software Engineering with SimulES-W" descreve o SimulES-W, um jogo educacional desenvolvido para ensinar Engenharia de Software. Baseado na dinâmica "Problems and Programmers", o SimulES-W apresenta inovações importantes, como ser uma plataforma web, cobrir uma visão ampla do processo de software e ser altamente personalizável. Os jogadores assumem diferentes papéis, como engenheiro de software, coordenador técnico, controlador de qualidade e gerente de projeto. Durante a dinâmica, eles enfrentam desafios que simulam situações reais de desenvolvimento de software, gerenciando recursos e resolvendo problemas para concluir projetos com qualidade e dentro do orçamento (MONSALVE, WERNECK & LEITE, 2011).

A dinâmica é estruturada em várias rodadas, onde os jogadores devem construir, inspecionar e integrar artefatos em um módulo. Além disso, a dinâmica incentiva discussões sobre problemas e conceitos de engenharia de software, promovendo um aprendizado ativo e colaborativo.

Os componentes da dinâmica incluem tabuleiros, cartas de projeto, engenheiros de software, problemas e conceitos, além de dados e marcadores que determinam as ações dos jogadores, conforme é possível observar na figura 7, SimulES-W foi projetado para ser uma ferramenta flexível e adaptável a diferentes contextos educacionais, tornando o ensino de Engenharia de Software mais interativo e eficaz (MONSALVE, WERNECK & LEITE, 2011).

Figura 7 - Página principal do SimulES-W



Fonte: SUESCÚN, 2018.

2.2. Sharkworld

O artigo "Make It Fun Or Real – Design Dilemmas And Their Consequences On The Learning Experience" discute o desenvolvimento da dinâmica Sharkworld, focado no ensino de gerenciamento de projetos. Sharkworld foi desenvolvido pela Ranj Serious Games para a OTIB, um instituto holandês de material educacional para empresas de instalação técnica e escolas vocacionais. Após receber elogios e prêmios, a dinâmica encontrou interesse como ferramenta de ensino em universidades e empresas de consultoria (HARTEVELD, BEKEBREDE, PLOMBER & JORDAAN, 2011).

Na dinâmica, os alunos devem gerenciar um grande projeto de construção de um aquário de tubarões na China. Eles enfrentam desafios relacionados ao planejamento, orçamento, qualidade e satisfação dos stakeholders. A pesquisa investiga como as escolhas de design, como flexibilidade e fidelidade, impactam a experiência de aprendizado dos alunos. O estudo mostrou que, embora os alunos tenham achado a dinâmica valiosa e envolvente, eles expressaram frustração com elementos que tornaram a

dinâmica menos realista e divertida, afetando negativamente a percepção de aprendizado (HARTEVELD, BEKEBREDE, PLOMBER & JORDAAN, 2011).

Figura 8 - Sharkworld (captura de tela).
Na parte de cima estão as ferramentas que podem ser usadas.
Figura 8 -

Fonte: HARTEVELD, BEKEBREDE, PLOMBER & JORDAAN, 2011.

A dinâmica foi projetada para equilibrar fidelidade e diversão, mas descobriu-se que esses dois elementos frequentemente entram em conflito. A flexibilidade nas escolhas de design permitiu que os alunos experimentassem diferentes abordagens para a gestão de projetos, mas isso também resultou em frustração, quando as simulações não refletiam com precisão as complexidades do mundo real. A fidelidade, por outro lado, tornou a dinâmica mais educativa, mas menos divertida, criando um dilema de design que impacta a eficácia geral da dinâmica como ferramenta de ensino (HARTEVELD, BEKEBREDE, PLOMBER & JORDAAN, 2011).

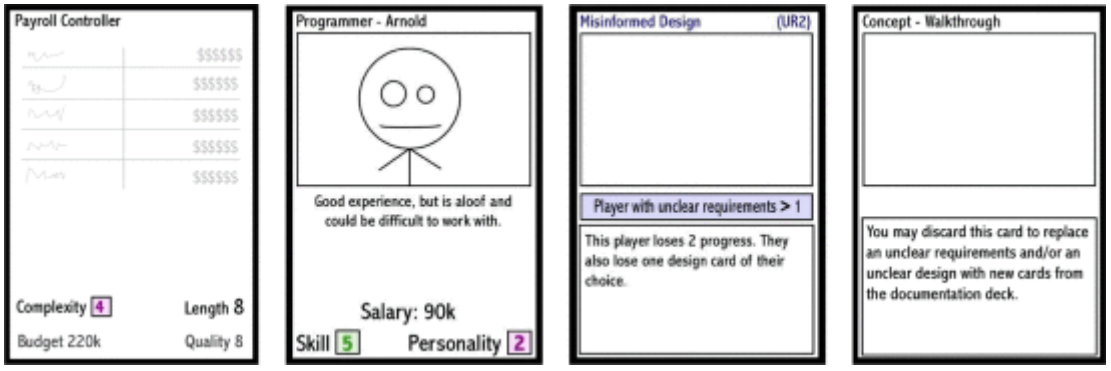
2.3. Problems and programmers

O artigo "An Experimental Card Game for Teaching Software Engineering Processes" descreve o desenvolvimento da dinâmica de cartas educacional *Problems and Programmers*, que simula o processo de

engenharia de software desde a especificação de requisitos até a entrega do produto. A dinâmica é projetada para fornecer uma experiência prática de alto nível do processo de engenharia de software em um curto período de tempo, destacando o processo de desenvolvimento ao invés dos artefatos produzidos. *Problems and Programmers* promove práticas adequadas de engenharia de software, recompensando boas práticas e penalizando desvios. Além disso, a dinâmica é competitiva e física, incentivando a aprendizagem colaborativa e garantindo que todos os mecanismos subjacentes do processo de engenharia de software sejam visíveis e, portanto, mais compreensíveis (BAKER e VAN DER HOEK, 2005).

A dinâmica é estruturada de forma a permitir que os jogadores assumam o papel de gerentes de projeto, competindo para completar um projeto antes de seus oponentes. Os jogadores devem equilibrar preocupações concorrentes, como orçamento e exigências do cliente, enquanto seguem práticas de engenharia de software para evitar consequências adversas que possam atrasá-los. A dinâmica utiliza um modelo de ciclo de vida em cascata, permitindo aos jogadores retornar a fases anteriores com penalidades, promovendo um entendimento das consequências de suas ações no processo de desenvolvimento de software (BAKER e VAN DER HOEK, 2005).

Figura 9 - Exemplos das cartas de: projeto, programador, problema e conceito



Fonte: BAKER, NAVARRO & VAN DER HOEK, 2003.

A dinâmica foi avaliada em um experimento com 28 estudantes de graduação, que forneceram feedback positivo sobre sua utilidade e diversão. Os resultados mostraram que a dinâmica pode reforçar conceitos de engenharia de software ensinados em cursos introdutórios e ensinar novos

conhecimentos processuais de forma eficaz (BAKER & VAN DER HOEK, 2005)

2.4. Discussão

Os jogos dos trabalhos relacionados são todos da área de engenharia de software, assim como a dinâmica física desenvolvida no trabalho de Reis (2023). Além dessa característica em comum, na tabela 2 é possível observar quais características principais apresentam os jogos dos trabalhos relacionados. Na última coluna estão as características que são apresentadas pelo *Metrics Poker.*

Tabela 2 - Características dos 3 trabalhos relacionados com o Metrics Poker

Característica	SimulES-W	*Sharkworld*	*Problems and programmers*	*Metrics Poker*
Jogo de tabuleiro para ES (engenharia de software)	parcial	não	parcial	sim
Multijogador	sim	sim	sim	sim
Colaborativo	não	sim	parcial	sim
Jogadores assumem diferentes papéis	sim	parcial	sim	parcial
Aprender a selecionar métricas	não	não	não	sim
Jogo digital	sim	sim	não	não

O SimulES-W é um jogo digital que foi baseado no *Problems and programmers,* esse, inclusive, é um dos motivos pelo qual foi escolhido, já que o presente trabalho é um jogo digital baseado no trabalho prévio de Luiz Reis, que desenvolveu a mesma dinâmica em um jogo de tabuleiro físico. O SimulES-W tem um jogo de tabuleiro para engenharia de software, multijogador e competitivo onde os jogadores assumem papéis diferentes e criam *cards* de projetos com a informação de id, nome, descrição, valor, complexidade, qualidade, tamanho e status, todas essas informações estão relacionadas à métricas ágeis.

Sharkworld assim como o SimulES-W é um jogo digital e colaborativo que permite aos participantes trabalho em equipe. A dinâmica, porém, não é tão realista e não apresenta um tabuleiro e conforme o trabalho relacionado apresentou não é tão divertido.

Problems and programmers é um jogo de tabuleiro físico com cartas, bem similar ao *Metrics Poker*. Um ponto positivo é que ele engloba desde a especificação de requisitos até a entrega do produto. Por outro lado, isso faz com que, no meio da dinâmica, algumas etapas não sejam aprofundadas, como é o caso da seleção de métricas de software. O papel principal na dinâmica é de um gestor de projetos, sendo que no Metrics Poker os jogadores desempenham muito mais papel de desenvolvedores que contribuem para selecionar as métricas de software.

Existe um *gap* que não é preenchido por nenhum dos jogos relacionados, que é a aprendizagem de como selecionar as métricas ágeis para um problema em específico que há no projeto a ser desenvolvido e é justamente essa lacuna nos trabalhos relacionados que o presente trabalho procura preencher. Além disso, o Metrics Poker é um jogo físico, e a proposta agora é disponibilizar a dinâmica no meio digital, para que as equipes possam jogar pelo computador.

3. Proposta

Neste capítulo, a partir da proposta da dinâmica *Metrics Poker,* já desenvolvida no trabalho de Reis (2023), é desenvolvida uma proposta de dinâmica digital[1]. A dinâmica de Reis (2023), porém, foi para uma dinâmica física em um tabuleiro, assim sendo, foram feitas algumas adaptações, para que seja possível desenvolver a dinâmica no formato digital.

3.1. Concepção da Dinâmica

O objetivo da dinâmica proposta por Reis (2023) é possibilitar a seleção das métricas que, de acordo com as necessidades da equipe, sejam as mais votadas. As métricas são votadas por cada jogador que, ao colocar uma ficha sobre a métrica, atribui um ponto para a carta selecionada. O jogador, no entanto, não recebe pontos. Além disso, não ocorre a formação de times dentro de uma mesma equipe, pois a dinâmica é realizada por todos os participantes do projeto.

Não há uma competição entre os participantes, e sim uma colaboração para alcançar o desafio almejado, que é encontrar as métricas pertinentes para que a equipe as utilize. Para isso, ocorrem 2 turnos, no primeiro são selecionados os grupos de métricas e, no segundo, são identificadas as métricas a serem utilizadas dentro dos grupos já selecionados.

A lista de grupos e métricas apresentada no trabalho de Reis (2023) foi construída com base em um Mapeamento Sistemático da Literatura (MSL). Esse mapeamento seguiu um protocolo detalhado, incluindo a definição de critérios de inclusão e exclusão e a realização de buscas em bases como ACM, IEEExplore e Scopus. A metodologia e a seleção dos estudos primários foram amplamente baseadas em um estudo prévio realizado por Stéphanie Leal e colaboradores (Leal et al., 2022).

O MSL identificou métricas frequentemente utilizadas em métodos ágeis e as classificou segundo um padrão clássico de classificação de métricas (Leal et al., 2022). A seguir, são apresentados os Grupos de Métricas e suas respectivas métricas relevantes:

[1] Disponível em https://metricspoker.netlify.app/

- Efetividade da Tecnologia: Frequência de execução de testes, Taxa de falha em testes, Tempo médio de execução de testes, Cobertura de testes, Taxa de aprovação em testes de segurança.
- Desempenho do Processo: Bugs corrigidos, Horas gastas em tarefas, Horas gastas em correção de bugs, Tempo de ciclo, Qualidade dos atributos das tarefas.
- Tamanho e Estabilidade: Itens do backlog de produtos modificados, Número de linhas de código, Componentes frágeis.
- Recursos e Custos: Esforço restante, Esforço, Precisão na estimativa de esforço, Total de horas efetivamente disponíveis da equipe, Tamanho do backlog.
- Satisfação do Cliente: NPS (Net Promoter Score).
- Qualidade do Produto: Bugs pendentes, Entregas no prazo, Velocidade do produto, Tempo de entrega, Taxa de produção (Throughput).
- Cronograma e Progresso: Número de tarefas concluídas, Crescimento do escopo, Mudança de prioridade, Diagramas de Fluxo Cumulativo, Revisão de Solicitações de Merge (Merge Request Review).

Os grupos e métricas foram derivados de um estudo que buscava responder a perguntas de pesquisa como: "Quais métricas são utilizadas em métodos ágeis?" e "Como essas métricas são classificadas?". Esses insights, obtidos de artigos publicados e relatos empíricos, foram posteriormente adaptados e refinados para uso na dinâmica proposta por Reis (2023).

Existem 2 tipos de jogadores: o Dealer: aquele que possui ciência de todas as regras da dinâmica e tem um mínimo conhecimento em métricas, já que é o responsável pela gamificação; e Membros da Equipe: fazem parte do dia a dia da equipe e não necessariamente precisam ter conhecimento sobre métricas, já que haverá explicações ao longo da dinâmica.

O Dealer, junto com o restante da equipe, deve indicar 3 dores a serem resolvidas pela equipe, podendo ser OKRs (*Objective Key Results*)[2] ou não. A priorização e definição dos problemas fogem do escopo da dinâmica.

[2] https://www.whatmatters.com/get-started

Uma vez definidos os problemas, a equipe seleciona dois Grupos de Métricas que têm relação com a dor. Portanto, deve haver um consenso para que sejam dispostos no tabuleiro. O Dealer, por sua vez, lê a descrição de cada métrica (cada participante também consegue ler no verso da carta). Cada participante recebe os 3 Tickets, cada um com um critério diferente (preferência pessoal, relevância para as necessidades, facilidade de coleta).

Em seguida, os Tickets devem ser associados a alguma métrica no tabuleiro por cada um dos participantes, de tal forma que o outro participante não consiga ver a ação dos outros, para que assim não seja influenciado pela decisão da outra pessoa. Apenas após todos atribuírem todos os 3 Tickets a pelo menos uma métrica é que os outros jogadores conseguem saber quais Tickets foram associados às respectivas métricas

Para encerrar a dinâmica, o Dealer é quem indica a mais votada das métricas de cada critério. Em caso de empate, o somatório de todos os Tickets associados à métrica é o critério de desempate. Se a métrica já havia sido selecionada anteriormente, a segunda mais votada (do respectivo critério) é escolhida.

A dinâmica é feita de forma virtual, com cartas, da mesma cor para cada grupo de métricas. No verso da carta há a explicação completa do grupo ao qual ela faz parte e sobre a métrica especificamente. Além disso, há 1 tabuleiro e 3 Tickets para cada integrante do grupo.

Por ser uma dinâmica online para equipes ágeis de projetos de software, os integrantes do grupo podem estar distribuídos e a dinâmica ocorre normalmente.

3.2. Requisitos

Com base na dinâmica proposta por Reis (2023), foram extraídos requisitos funcionais para a implementação de uma versão digital. A dinâmica digital desenvolvida tem os requisitos funcionais e não funcionais com suas descrições listados na tabela 3:

Tabela 3 - Código, título e detalhamento dos requisitos funcionais

#	Título	Detalhamento
RF01	O sistema deve permitir ao *Dealer* se cadastrar com seu e-mail	Para que ocorra a dinâmica, o sistema deve permitir o Dealer se cadastrar apenas com seu e-mail
RF02	O sistema deve permitir ao usuário visualizar o tabuleiro	O tabuleiro deve aparecer tanto para o Dealer quanto para os membros da equipe, cada um com as devidas permissões de acordo com as regras da dinâmica
RF03	O sistema deve permitir ao usuário visualizar as regras da dinâmica	Haverá uma tela específica explicando as regras da dinâmica
RF04	O sistema deve permitir ao usuário a visualização das cartas	Haverá uma tela na qual poderão ser vistas todas as cartas existentes da dinâmica
RF05	O sistema deve permitir aos usuários a informação de que a dinâmica acabou	Uma última tela com as métricas selecionadas e os critérios com os quais cada uma foi selecionada
RF06	O sistema deve criar uma sessão de jogo para a partida iniciada	O objetivo é que uma sessão seja criada a partir do login feito pelo Dealer, de tal forma que o Dealer tenha visualização levemente diferente dos demais participantes da dinâmica
RF07	O sistema deve permitir a cada um dos membros da equipe	Ao receber o *link* gerado pela sessão do *Dealer* , os demais jogadores ao clicar no link devem cadastrar um *nickname* (nome

	cadastrar um nome fictício	fictício)

Tabela 4 - Requisitos não funcionais

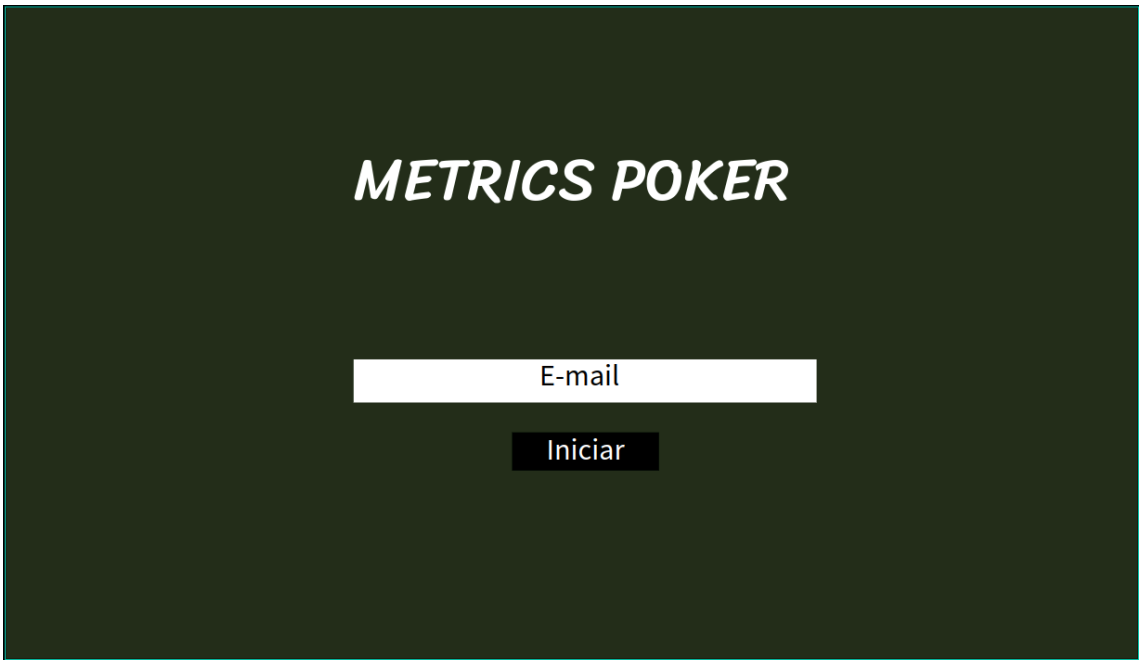
#	Título	Detalhamento
RNF01	A interface do usuário deve ser intuitiva e fácil de usar, permitindo que novos usuários compreendam seu funcionamento em menos de 10 minutos.	A disposição dos elementos na tela, o acesso e informações devem estar de forma clara e direta em toda a estrutura do site criado para a dinâmica
RNF02	O sistema deve ser feito utilizando a linguagem JavaScript	Javascript é a linguagem utilizada no desenvolvimento de novos sites na web e a linguagem em que o desenvolvedor do projeto possui maior conhecimento
RNF03	O sistema deve utilizar o framework Vue.JS	Para desenvolvimento das interfaces deve ser utilizado o Vue.JS
RNF04	O sistema é destinado a equipes distribuídas	O sistema possibilita que pessoas em diferentes locais do mundo consigam acessar e participar de forma remota

3.3. Protótipos de tela

Tomando por base os requisitos funcionais e não-funcionais identificados, foram desenvolvidos protótipos de telas. A seguir, encontram-se cada um dos protótipos de tela referenciados com os respectivos requisitos funcionais levantados:

A primeira interação do usuário com a dinâmica digital é feita a partir da tela ilustrada na figura 10, onde é possível o usuário identificar o nome da dinâmica, inserir seu e-mail para, em seguida, clicar em iniciar, dando prosseguimento à tela seguinte, conforme pede o requisito funcional RF01.

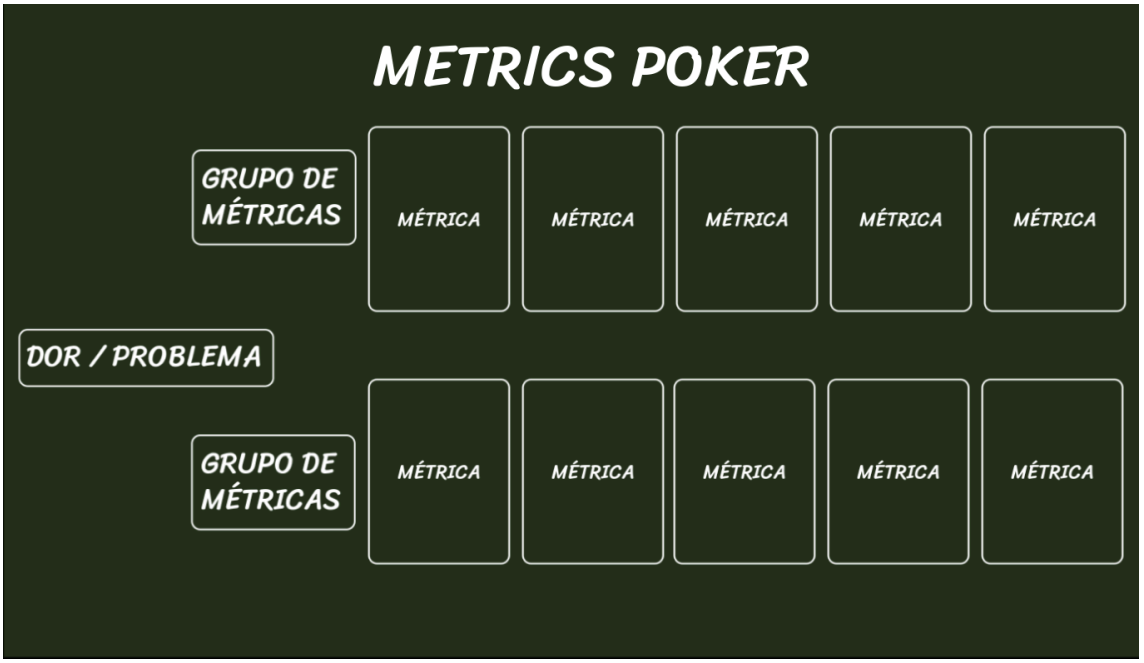
Figura 10 - Requisito funcional RF01



Fonte: Autor

Depois de clicado em iniciar, a próxima tela é prototipada na figura 11, onde é possível o usuário visualizar a tela principal da dinâmica, que é o tabuleiro, onde, à esquerda, há a dor ou problema identificado que será avaliado, além de 2 filas de grupos de métricas que serão selecionadas no decorrer da dinâmica, conforme pede o requisito funcional RF02.

Figura 11 - Requisito Funcional RF02



Fonte: Autor

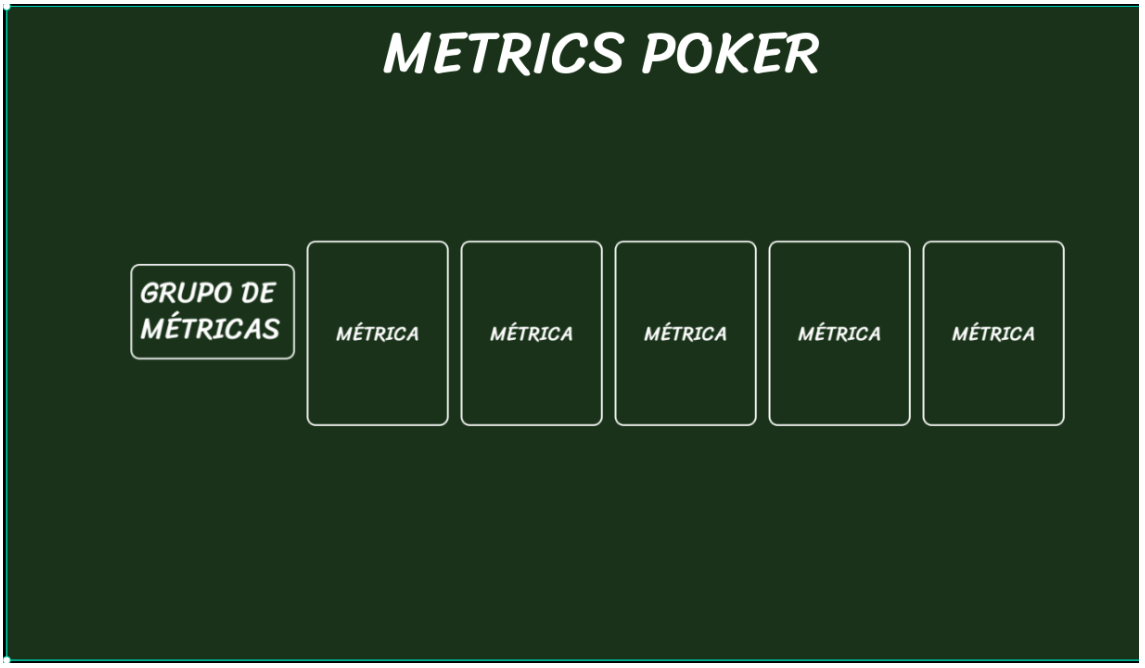
Haverá ainda uma tela onde é possível o usuário visualizar as regras da dinâmica como um todo, do início ao fim, para que assim todos consigam participar e sanar suas dúvidas conforme ilustra a figura 12, conforme pede o requisito funcional RF03.

Figura 12 - Requisito funcional RF03

Fonte: Autor

Para visualizar cada grupo de métricas, a figura 13 exemplifica a forma como elas devem aparecer na tela sem ser o tabuleiro, apenas listadas na tela, para que seja possível ver o nome de cada uma delas na frente da carta e, ao clicar o verso, revela a descrição da métrica. No protótipo está em um grupo de métricas, porém, nessa tela, a ideia é que sejam mostrados todos. Foi colocado um como exemplo, apenas para ilustrar, conforme pede o requisito funcional RF04.

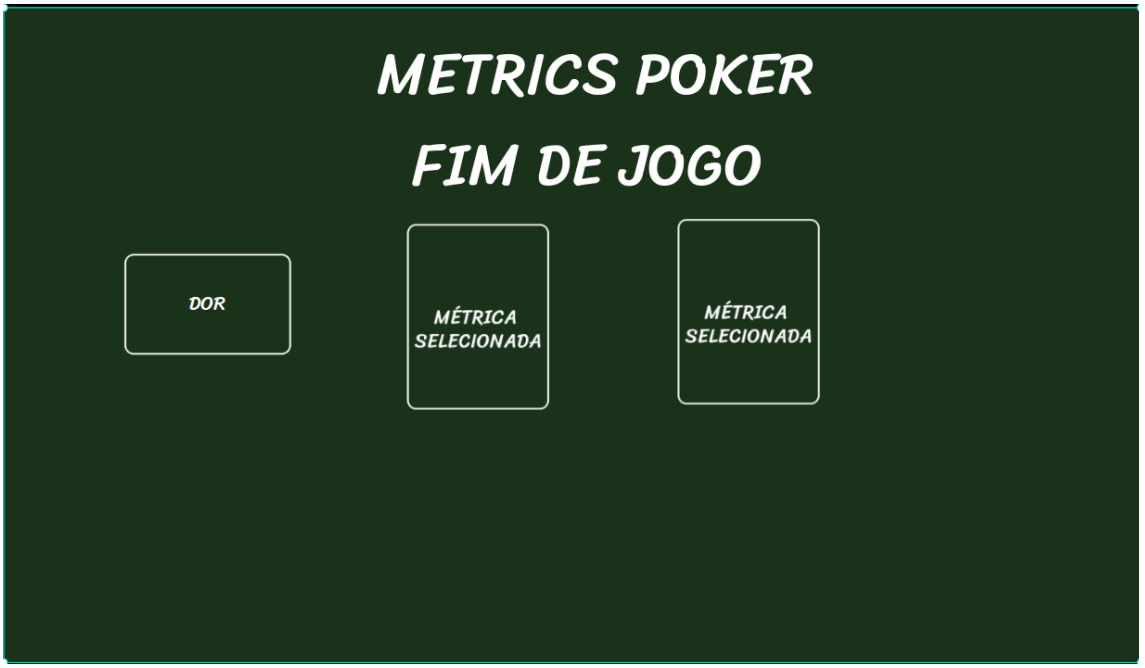
Figura 13 - Requisito funcional RF04



Fonte: Autor

Ao finalizar a dinâmica, o usuário tem um feedback de que ela terminou. Para isso, a tela informa “fim de jogo” além de mostrar, a partir da dor, quais as métricas que foram selecionadas na dinâmica, tal qual pede o requisito funcional RF05, conforme é ilustrado na figura 14.

Figura 14 - Requisito funcional RF05

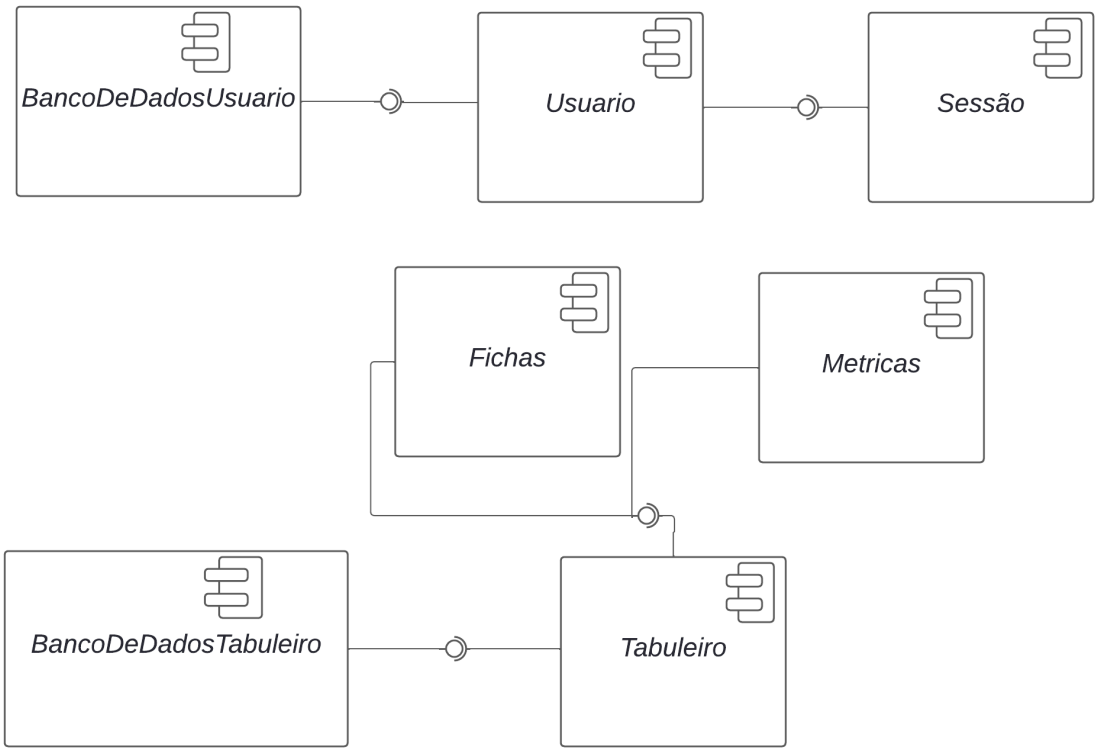


Fonte: Autor

3.4. Arquitetura da solução

Os componentes do projeto, que estarão em pastas na implementação do projeto, iniciam a partir do componente de banco de dados do usuário para conseguir gerar uma sessão de login. Com relação ao banco de dados do tabuleiro, ele deve armazenar as informações de fichas e métricas e, com isso, finalizar a estrutura de componentes do backend, conforme mostrado na figura 15.

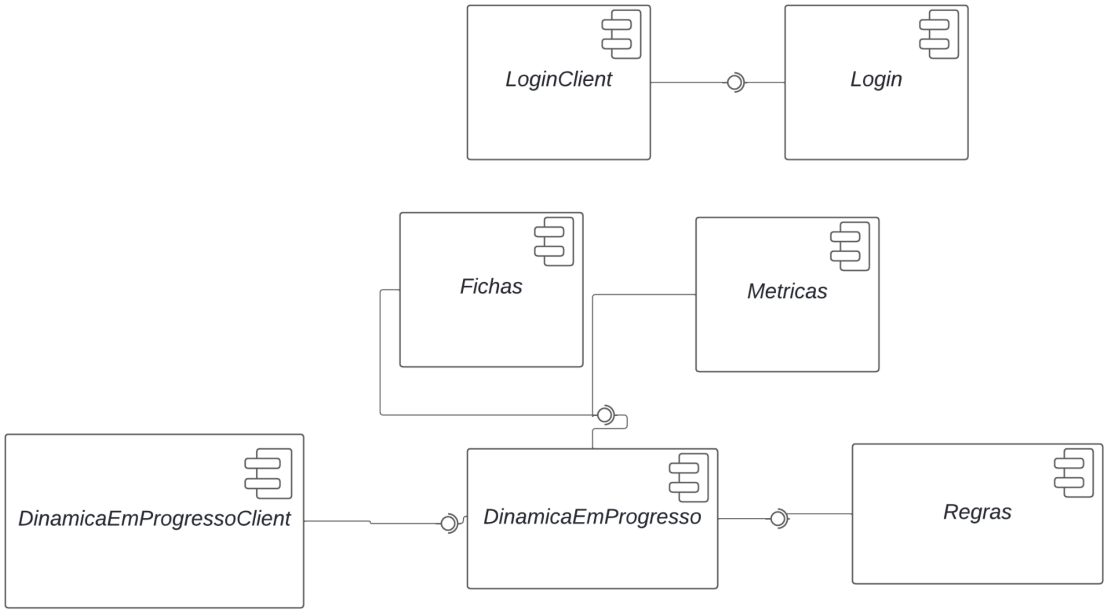
Figura 15 - - Diagrama de componentes do backend do software desenvolvido



Fonte: Autor

Os componentes no front-end seguem uma ideia similar, porém adaptada ao diagrama de componentes para o backend, onde as telas farão parte dos componentes, assim como a comunicação com o backend será feita através de componentes *client,* como por exemplo *dinâmicaEmProgressoClient* e *loginClient*, conforme é possível observar na figura 16.

Figura 16 - Diagrama de componentes do frontend do software desenvolvido



Fonte: Autor

3.5. Tecnologias utilizadas

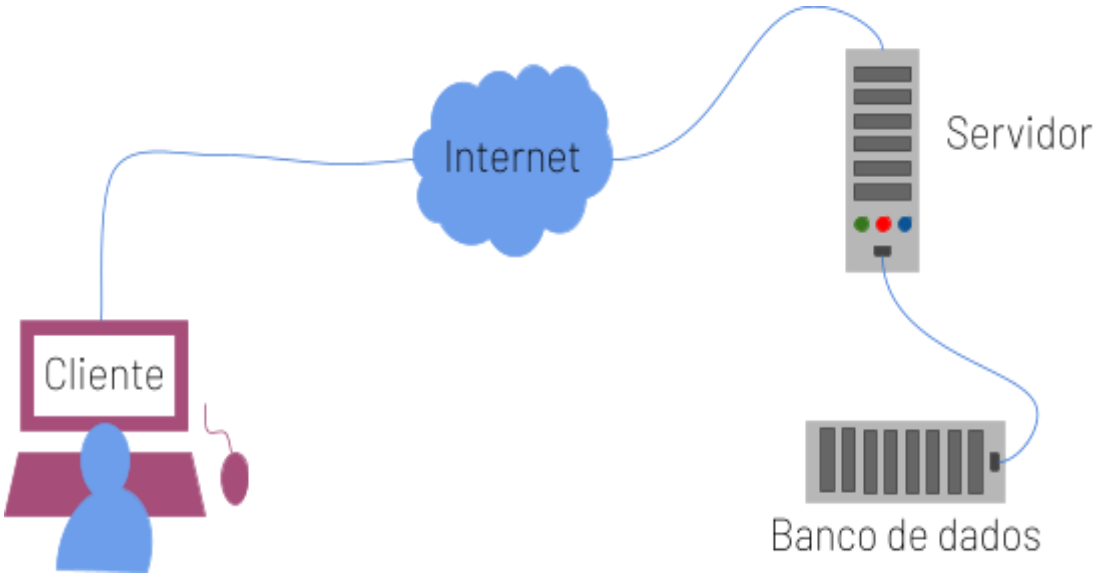
As tecnologias escolhidas para desenvolver o projeto estão listadas a seguir, e cada uma delas foi escolhida por conta da experiência prévia do responsável pelo trabalho, não havendo uma seleção com base em outros critérios, além do critério de facilidade de implementação pelo responsável.

O presente trabalho irá criar uma aplicação web que garante uma comunicação cliente servidor com o uso de diferentes protocolos de comunicação (ADHIKARI, 2016). Protocolos são um conjunto de regras para a troca de mensagens em uma rede de comunicação (ADHIKARI, 2016). Sendo assim, existem diferentes tipos de protocolos com diferentes atributos e

que estão em diferentes camadas utilizados em aplicações web tais como HTTP (*HyperText transfer Protocol*) e o REST (*Representational State Transfer*) (ADHIKARI, 2016).

Um exemplo de comunicação é possível ver na figura 17, na qual o cliente envia uma requisição HTTP para o servidor, a fim de estabelecer a comunicação, em seguida o servidor envia uma resposta HTTP para o cliente (ADHIKARI, 2016).

Figura 17 - Comunicação em aplicação web



Fonte: adaptado de ADHIKARI (2016).

As tecnologias utilizadas para na implementação deste trabalho foram dividas em frontend e backend e estão apresentadas a seguir:

3.5.1. Front-end

Front-end é a prática de disponibilizar a aplicação web para o usuário final, de forma que ele consiga ver e interagir com a aplicação diretamente, de tal forma que ao abrir o site a informação esteja em um formato fácil de visualizar e que seja relevante (MADURAPPERUMA, 2022). Para isso algumas ferramentas são utilizadas conforme mostram os próximos parágrafos:

A linguagem de programação utilizada para desenvolvimento do front-end neste trabalho é o JavaScript[3]. JavaScript é uma linguagem leve e interpretada usada principalmente para páginas web, rodando do lado do cliente e que pode executar ações a partir de um evento, sendo fácil de utilizar e ao mesmo tempo poderosa, além de ser utilizada em outros ambientes sem browser como no caso do Node.js[4].

Juntamente com o JavaScript será utilizado o CSS[5] (*Cascading Style Sheets*), Folhas de Estilo em Cascata para a estilização. CSS é a linguagem do

[3] https://developer.mozilla.org/en-US/docs/Web/JavaScript
[4] https://nodejs.org/en
[5] https://developer.mozilla.org/en-US/docs/Web/CSS

estilo da página web que é a responsável por descrever a forma como os elementos estarão dispostos na tela do usuário e essa linguagem de estilo é padronizada por especificações da W3C[6].

HTML[7] (*Hypertext Markup Language*), linguagem de marcação de hipertexto, será utilizado para definição do conteúdo e estrutura da página web. O hipertexto faz referência aos links entre cada uma das páginas web, dentro ou fora do mesmo site.

O framework Vue.JS[8] também será utilizado, assim como o Tailwind.CSS[9] para deixar a dinâmica mais interativa. O Vue.JS é utilizado para desenvolvimento de interfaces de usuário, projetado para ser adotado de maneira incremental, e é tão popular que outras bibliotecas surgiram exclusivamente para ele como é o caso do Vuetify[10] e Vuelidate[11].

O Tailwind CSS, disponibiliza classes CSS com estilos correspondentes responsivos. O Tailwind CSS escaneia todos os arquivos HTML, componentes JavaScript e qualquer outro template por classes CSS, gerando os estilos correspondentes em um arquivo CSS estático.

O JSON[12] (*JavaScript Object Notatio*n) será utilizado como forma de disponibilizar as informações para que o front-end consiga tratá-las seja para editar, excluir ou criar o que for necessário.

3.5.2. Back-end

O back-end é a camada responsável pelo processamento dos dados e pelas regras de negócio de uma aplicação, diferentemente do front-end, que se preocupa com a interface do usuário (YUNRUI, 2018).

A linguagem de programação utilizada para desenvolvimento do back-end será, assim como no front-end, o JavaScript. Junto com JavaScript será utilizado o framework Node.JS. Node.JS é orientado a eventos de forma

[6] https://www.w3.org/Style/CSS/#specs
[7] https://developer.mozilla.org/en-US/docs/Web/HTML
[8] https://vuejs.org/guide/introduction.html
[9] https://tailwindcss.com/docs/installation
[10] https://vuetifyjs.com/en/introduction/why-vuetify/#what-is-vuetify3f
[11] https://vuelidate-next.netlify.app/
[12] https://www.json.org/json-en.html

assíncrona em tempo de execução no JavaScript, tendo sido criado para que seja possível construir aplicações web escaláveis.

O banco de dados utilizado é o Supabase[13], uma plataforma de código aberto que oferece uma alternativa ao Firebase, mas com base no PostgreSQL[14], um banco de dados SQL relacional robusto e escalável. Supabase é ideal para aplicações que exigem não só flexibilidade e escalabilidade, mas também uma estrutura de dados relacional, o que facilita a integridade dos dados e o uso de consultas SQL. A plataforma inclui autenticação, armazenamento de arquivos, funções serverless e uma API REST gerada automaticamente a partir de tabelas e permissões configuradas no próprio banco. Além disso, o Supabase possui uma documentação completa, tutoriais e uma comunidade ativa que auxiliam no desenvolvimento e gerenciamento dos dados, tornando-o uma excelente escolha para projetos que demandam um sistema mais completo e integrado.

O uso do Node.js, Supabase e JWT[15] (J*SON Web Token*) proporciona uma infraestrutura robusta e eficiente para o desenvolvimento de aplicações modernas, suportando uma grande quantidade de conexões simultâneas e permitindo um desenvolvimento rápido e ágil.

3.5.3. REST

REST[16] (*Representational State Transfer* - Transferência de Estado Representacional) é um estilo arquitetural para sistemas distribuídos que apresenta um conjunto de restrições (FIELDING, 2000).

A primeira restrição é o estilo arquitetural Cliente-Servidor, ou seja, que a questão de armazenamento de dados fique separado das preocupações com a interface do usuário, por melhorar a escalabilidade e a simplicidade dos componentes do servidor, além de possibilitar que os componentes possam evoluir de forma independente (FIELDING, 2000).

A segunda é *Stateless,* o que significa que o cliente, a cada solicitação feita, deve conter todas as informações necessárias e também não pode se

[13] https://supabase.com/docs
[14] https://www.postgresql.org/about/
[15] https://jwt.io/
[16] https://restapitutorial.com/

aproveitar de contexto algum armazenado no lado do servidor, portanto o cliente é responsável por manter o estado da sessão (FIELDING, 2000).

O Cache é a terceira restrição mostra que ao responder uma requisição deve conter o rótulo se a informação permite ou não armazenar o cache, seja implicitamente ou explicitamente (FIELDING, 2000) .

Quarta restrição, a Interface Uniforme entre os componentes, de forma que a visibilidade das iterações é aprimorada e a arquitetura geral do sistema é simplificada pois há uma padronização na forma que as informações são enviadas (FIELDING, 2000).

Sistema em camadas é a quinta restrição, e permite que a arquitetura seja feita em camadas hierárquicas, de tal forma que cada componente não veja outras camadas com as quais não está interagindo (FIELDING, 2000).

Por fim, a sexta restrição é o código sob demanda, que é uma restrição opcional, o servidor pode estender temporariamente o cliente, transferindo lógica para o cliente como, por exemplo, código JavaScript (FIELDING, 2000)

3.6. Banco de Dados

O Supabase foi escolhido para a persistência dos dados. O Supabase utiliza um banco de dados Postgres SQL. Um banco de dados SQL (Structured Query Language, ou Linguagem de Consulta Estruturada) é um sistema que organiza e armazena dados de forma estruturada, usando tabelas com linhas e colunas. Ele é baseado em um modelo relacional, onde os dados são organizados em tabelas que podem se relacionar entre si, o que facilita a organização e o acesso a grandes volumes de dados de forma eficiente.

A estrutura SQL permite realizar operações complexas de busca, inserção, atualização e exclusão de dados usando uma linguagem de consulta chamada SQL. Esse tipo de banco de dados é amplamente usado em aplicações onde é importante garantir a consistência, integridade e confiabilidade dos dados, como em sistemas financeiros, e-commerces, redes sociais e muitos outros.

Algumas características chave dos bancos de dados SQL incluem:

- Integridade referencial: As relações entre tabelas são mantidas através de chaves estrangeiras, que garantem que os dados se mantenham consistentes e evitem redundâncias.

- Transações: São operações que garantem que um conjunto de instruções SQL seja executado de maneira confiável, mesmo em caso de falhas. Isso é fundamental em sistemas onde não se pode ter erros em atualizações ou gravações de dados.

- Linguagem SQL: Usada para definir, manipular e consultar os dados, ela permite que os desenvolvedores criem queries complexas para obter informações específicas.

- Escalabilidade: Bancos de dados SQL podem ser escalados verticalmente (aumentando o poder de um único servidor) e, em algumas implementações, horizontalmente (distribuindo dados entre múltiplos servidores).

3.6.1. Tabelas

Sessões de Jogo: A tabela de Sessões de Jogo (*game_sessions*) possui as colunas: identificador (*id*), criado em (*created_at*), criado por (*created_by*), nome do jogo (*name*), status do jogo (*status*), dor 1 (*problemA*), dor 2 (*problemB*) e dor atual em análise (currentProblem). Essas colunas permitem armazenar informações essenciais sobre cada sessão, incluindo o problema que está sendo analisado no momento. Abaixo, na figura 18, é possível ver as colunas no banco de dados com a descrição, o tipo de dado aceito (*Data Type*) e o formato (*Format*).

Figura 18 - Tabela de Sessões de Jogo (game_sessions)

Name	Description	Data Type	Format
id	No description	bigint	int8
created_at	No description	timestamp with time zone	timestamptz
created_by	No description	uuid	uuid
name	name of game	text	text
status	which status is the game	text	text
problemB	No description	bigint	int8
problemA	No description	bigint	int8
currentProblem	No description	bigint	int8

Fonte: Autor

Dores: A tabela de Dores (*problems*) inclui as colunas: identificador (*id*), criado em (*created_at*), nome (*name*), descrição (*description*), e grupos de métricas (*metricsGroups*). Essa tabela armazena os diferentes problemas que podem ser analisados nas sessões de jogo, permitindo associar múltiplas dores a uma mesma sessão. Abaixo estão as colunas, com suas descrições, tipos de dados e formatos.

Figura 19 - Tabela das Dores (problems)

Name	Description	Data Type	Format
id	No description	bigint	int8
created_at	No description	timestamp with time zone	timestamptz
name	No description	text	text
description	No description	text	text
metricsGroups	grupos de métricas escolhidos pelos participantes do jogo por v otação é escolhido o mais votado	text	text

Fonte: Autor

Métricas: A tabela de Métricas (*metrics*) contém as colunas: identificador (*id*), criado em (*created_at*), valor (*value*), nome da métrica (*name*), descrição (*description*), sessão de jogo associada (*game_session*), contagem (*count*), criado por (*created_by*), dor associada (*problem*), e participantes (*participants*). Esta tabela registra as métricas associadas a cada sessão e dor, permitindo monitorar os dados relevantes de cada análise de forma individual. Abaixo estão as colunas e seus respectivos detalhes.

Figura 20 - Tabela de Métricas(metrics)

Name	Description	Data Type	Format
id	No description	bigint	int8
created_at	No description	timestamp with time zone	timestamptz
value	No description	text	text
name	No description	text	text
description	No description	text	text
game_session	No description	bigint	int8
count	No description	bigint	int8
created_by	No description	uuid	uuid
problem	No description	bigint	int8
participants	participantes que votaram na métrica	ARRAY	_jsonb

Fonte: Autor

Participantes: A tabela de Participantes (*participants*) inclui as colunas: identificador (*id*), criado em (*created_at*), apelido (*nickname*), dor 1 (*problemA*), dor 2 (*problemB*), relevância da dor 1 (*relevanceA*), relevância da dor 2 (*relevanceB*), facilidade da dor 1 (easeA), facilidade da dor 2 (easeB), preferência pela dor 1 (*preferenceA*), preferência pela dor 2 (*preferenceB*), e sessão de jogo associada (*game_session*). Esta tabela permite registrar os participantes e suas avaliações sobre os problemas em análise durante uma sessão de jogo específica.

Figura 21 - Tabela dos Participantes (participants)

Name	Description	Data Type	Format
id	No description	uuid	uuid
created_at	No description	timestamp with time zone	timestamptz
nickname	No description	text	text
problemA	No description	text	text
problemB	No description	text	text
relevanceA	No description	text	text
easeA	No description	text	text
preferenceA	No description	text	text
game_session	No description	bigint	int8
relevanceB	métrica relevante para o problema b	text	text
easeB	métrica escolhida por facilidade para coleta do problema B	text	text
preferenceB	Métrica escolha por preferência pessoal do problema B	text	text

Fonte: Autor

As relações entre as tabelas do banco de dados são estabelecidas por meio de chaves estrangeiras que conectam as informações de forma lógica e organizada.

3.6.2. Relação entre tabelas

Relação entre *game_sessions* e *problems*: A tabela *game_sessions* possui três colunas (*problemA*, *problemB* e *currentProblem*) que funcionam

como chaves estrangeiras para a tabela *problems*. Essas chaves permitem que cada sessão de jogo esteja associada a diferentes problemas, indicando quais problemas estão sendo abordados e analisados durante a sessão.

Relação entre *metrics* e *game_sessions*: A coluna *game_session*, na tabela *metrics*, é uma chave estrangeira que aponta para a tabela *game_sessions*. Isso permite que cada métrica esteja vinculada a uma sessão específica de jogo, facilitando o monitoramento de dados relacionados a cada sessão.

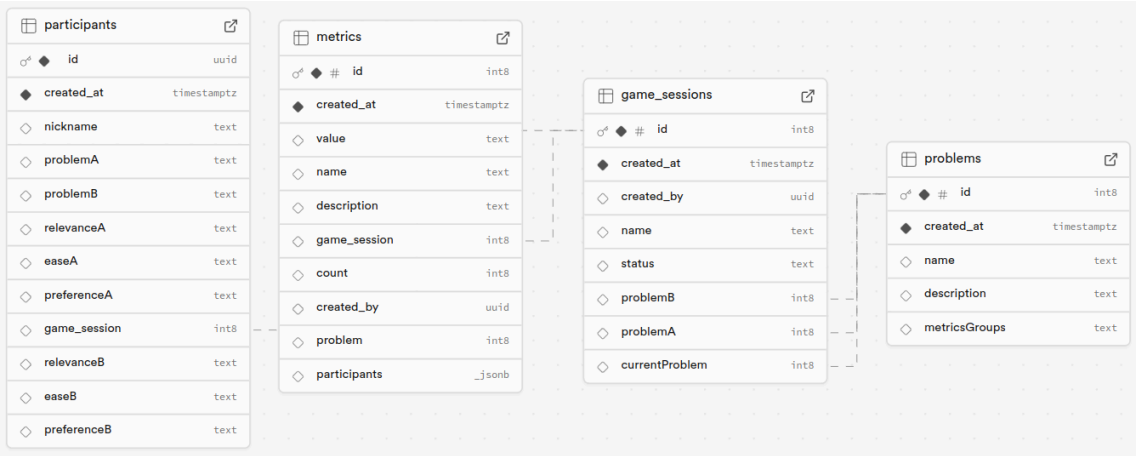
Relação entre *metrics* e *problems*: A coluna *problem* na tabela *metrics* é uma chave estrangeira que se relaciona com a tabela *problems*. Essa relação associa cada métrica a um problema específico, permitindo que as métricas sejam contextualizadas dentro do problema que estão avaliando.

Relação entre *participants* e *game_sessions*: A coluna *game_session* na tabela *participants* é uma chave estrangeira que conecta cada participante a uma sessão de jogo específica. Com essa relação, é possível saber quais participantes estão envolvidos em cada sessão e acompanhar suas avaliações sobre os problemas.

Essas chaves estrangeiras estruturam o banco de dados de maneira relacional, garantindo que as informações de sessões, problemas, métricas e participantes estejam organizadas e possam ser facilmente recuperadas conforme as necessidades do sistema.

Abaixo é possível ver todas as tabelas e a relação entre cada uma delas no diagrama ER (Diagrama Entidade-Relacionamento) uma representação gráfica que ajuda a modelar as estruturas e relações de um banco de dados. Ele mostra as entidades (tabelas), atributos (colunas ou campos das tabelas) e relacionamentos (conexões entre as tabelas) de forma visual, facilitando a compreensão e o planejamento de um banco de dados.:

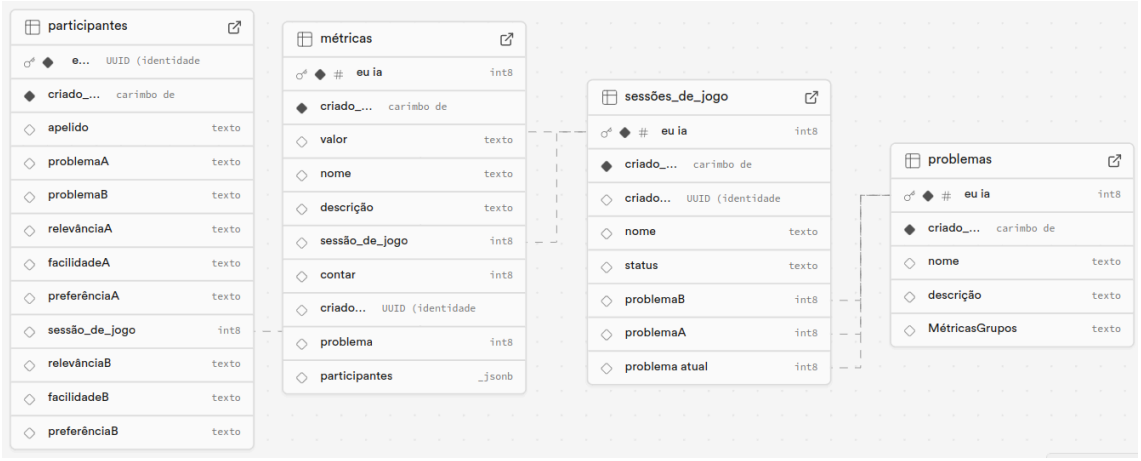
Figura 22 - Esquema Relacional (ER)



Fonte: Autor

A figura 22 acima mostra a versão original feita com tabelas e atributos em inglês. Por outro lado, a figura 23 abaixo é uma versão com a tradução dos campos para o Português.

Figura 23 - Esquema Relacional traduzido para melhor compreensão



Fonte: Autor

3.7. Desenvolvimento

O foco do trabalho foi no frontend, portanto ferramentas que facilitaram o desenvolvimento do backend foram utilizadas. A principal delas é o Supabase. Já no frontend, o principal *framework* que contribuiu foi o Vue.JS.

Infraestrutura Inicial: O desenvolvimento começou pela modelagem do backend, incluindo a persistência dos seguintes dados:

- Usuários.
- Sessões de jogo.
- Ações dos participantes (como votos).
- Histórico de jogos.
- Estética: Após a infraestrutura, foi trabalhado o design e a interface.
- Sprints: O projeto é gerido em sprints de duas semanas.

3.7.1. *Backend*

A ferramenta *open source* Supabase pode funcionar também como backend. Para o desenvolvimento do backend do projeto, foram utilizados os serviços fornecidos pelo Supabase, uma plataforma que simplifica o gerenciamento de banco de dados e a criação de *APIs RESTful* para consulta e manipulação de dados. O Supabase oferece *endpoints RESTful* automáticos e seguros para as tabelas do banco de dados, o que facilita a criação de *APIs* sem a necessidade de configurações complexas.

3.7.1.1. Configuração da *API*

A *URL* base do projeto é: https://clrfpadopbydsjiqejxb.supabase.co

O Supabase possui uma *API RESTful* segura que requer uma *API Key* para cada requisição. A chave pública (*anon*) é destinada a operações de leitura em ambientes onde a segurança de nível de linha (*RLS*) está ativada e as políticas de acesso estão configuradas. Para operações que exigem permissões mais elevadas, como inserções, atualizações ou exclusões, há uma chave de serviço (*service_role*), que permite cumprir as políticas de segurança. No entanto, essa chave é confidencial e não deve ser compartilhada publicamente.

3.7.1.2. Estrutura da *API*

A API do Supabase é baseada no modelo REST, com endpoints que permitem realizar as operações *CRUD (Create, Read, Update, Delete)* diretamente nas tabelas do banco de dados. Cada entidade do banco de dados pode ser acessada através de um *endpoint* específico, onde são permitidas as operações de:

- *GET*: Para realizar consultas e obter dados de uma tabela específica.
- *POST*: Para inserir novos registros nas tabelas.
- *PATCH* ou *PUT*: Para atualizar registros existentes.
- *DELETE*: Para excluir registros da tabela.

As operações são realizadas com base no identificador único de cada registro, o que garante integridade e segurança nas operações.

3.7.1.3. Endpoints da *API*

Game_sessions

- *GET: /rest/v1/game_sessions?id=eq.{ID}*
- *POST: /rest/v1/game_sessions*
- *PATCH: /rest/v1/game_sessions?id=eq.{ID}*
- *DELETE: /rest/v1/game_sessions?id=eq.{ID}*

Metrics

- *GET*: /rest/v1/metrics?id=eq.{ID}
- *POST*: /rest/v1/metrics
- *PATCH*: /rest/v1/metrics?id=eq.{ID}

Participants

- *GET: /rest/v1/participants?id=eq.{ID}*
- *POST: /rest/v1/participants*
- *PATCH: /rest/v1/participants?id=eq.{ID}*
- *DELETE: /rest/v1/participants?id=eq.{ID}*

Problems

- *GET: /rest/v1/problems?id=eq.{ID}*
- *POST: /rest/v1/problems*
- *PATCH: /rest/v1/problems?id=eq.{ID}*

Observação: Para cada *endpoint*, {*ID*} representa o identificador do registro, permitindo realizar operações diretamente sobre registros específicos.

3.7.1.4. Documentação da *API*

A documentação da *API* gerada pelo Supabase é automaticamente disponibilizada para cada tabela e entidade configurada no banco de dados. Através do console do Supabase, é possível visualizar e testar os endpoints diretamente, além de ter acesso aos detalhes sobre os métodos suportados, parâmetros, chaves de acesso e tipos de dados para cada tabela. Isso facilita a integração e uso da *API* durante o desenvolvimento.

3.7.1.5. Autenticação

O Supabase tem disponível a gestão de usuários para autenticação utilizando provedores como o Google, entre outros. O escolhido para efetuar o login neste trabalho foi o provedor Google, que fornece um token de autenticação de usuário para que consiga acessar outras rotas da aplicação desenvolvida. No Supabase, as *Row-Level Security* (*RLS*) *Policies* permitem que você controle quais operações (inserção, seleção, atualização e exclusão) estão disponíveis para diferentes tipos de usuários em cada tabela do banco de

dados. Essas políticas ajudam a definir regras específicas de acesso, restringindo ou permitindo ações com base em condições personalizadas, como o papel do usuário (exemplo: *public* - público ou *authenticated* - autenticado) ou dados específicos da tabela, como o campo *created_by*.

Abaixo, segue um resumo das políticas configuradas para cada tabela conforme descrito:

Tabela ***game_sessions***

DELETE: Permite a exclusão de registros para usuários baseando-se no campo created_by.

- Aplica-se a usuários anônimos.
- Papel aplicado: *public*.

SELECT: Permite o acesso de leitura para todos os usuários.

- Aplica-se a usuários anônimos.
- Papel aplicado: *public*.

UPDATE: Permite a atualização para usuários autenticados.

- Aplica-se a usuários anônimos.
- Papel aplicado: *authenticated*.

INSERT: Permite a inserção de novos registros apenas para usuários autenticados.

- Aplica-se a usuários anônimos.
- Papel aplicado: *authenticated*.

Tabela ***metrics***

INSERT: Permite a inserção apenas para usuários autenticados.

- Aplica-se a usuários anônimos.
- Papel aplicado: *authenticated*.

SELECT: Permite o acesso de leitura para todos os usuários.

- Aplica-se a usuários anônimos.

- Papel aplicado: *public*.

UPDATE: Permite a atualização para todos os usuários baseando-se no campo *updated_by*.

- Aplica-se a usuários anônimos.
- Papel aplicado: *public*.

Tabela ***participants***

INSERT: Permite a inserção apenas para usuários autenticados.

- Aplica-se a usuários anônimos.
- Papel aplicado: *authenticated*.

SELECT: Permite o acesso de leitura para todos os usuários autenticados

- Aplica-se a usuários anônimos.
- Papel aplicado: *authenticated*.

UPDATE: Permite a atualização de registros para todos os usuários com uma política definida.

- Aplica-se a usuários anônimos.
- Papel aplicado: *public*.

DELETE: Permite a exclusão de registros apenas para usuários autenticados.

- Aplica-se a usuários anônimos.
- Papel aplicado: *authenticated*.

Tabela ***problems***

INSERT: Permite a inserção apenas para usuários autenticados.

- Aplica-se a usuários anônimos.
- Papel aplicado: *authenticated*.

SELECT: Permite o acesso de leitura para todos os usuários.

- Aplica-se a usuários anônimos.
- Papel aplicado: *public*.

UPDATE: Permite a atualização de registros para usuários autenticados com uma política específica.

- Aplica-se a usuários anônimos.
- Papel aplicado: *authenticated*.

Essas políticas configuradas no Supabase garantem que apenas usuários específicos possam realizar determinadas ações em cada tabela, aumentando a segurança e controle de acesso do banco de dados. As configurações de *RLS* permitem a adaptação ao acesso de acordo com as necessidades, controlando rigorosamente quem pode visualizar, modificar ou inserir dados.

3.7.2. Frontend

As cartas do jogo foram projetadas para simular um baralho de poker, utilizando símbolos e cores que remetem aos grupos de métricas. O layout prioriza clareza e semântica, destacando as métricas selecionadas no resultado final com cor verde para indicar sucesso.

Essa estrutura detalhada visa facilitar a dinâmica colaborativa, assegurando que todos os participantes compreendam suas ações e o progresso do jogo em cada etapa.

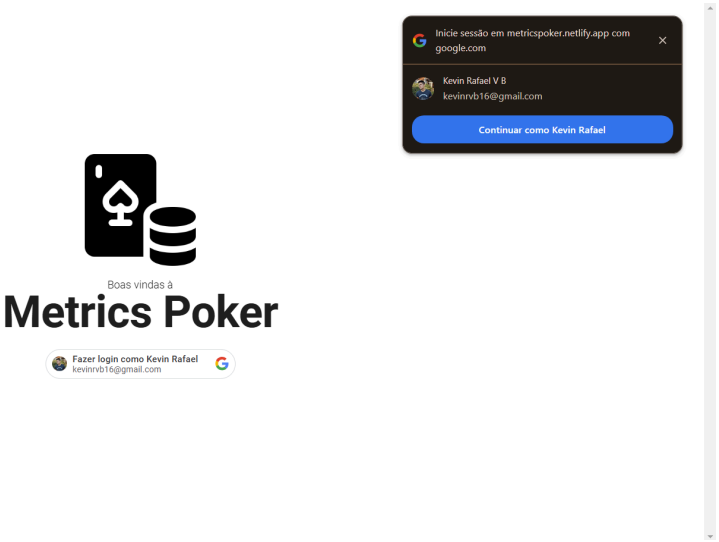
O desenvolvimento do frontend foi feito com base nos requisitos funcionais (RF) que abaixo são especificados cada um deles com as imagens de como ficaram as telas, com certa diferença dos protótipos desenvolvidos na sessão

3.7.2.1. RF01 - O sistema deve permitir ao *Dealer* se cadastrar com seu *e-mail*

O login é feito utilizando o Google como provedor, o que torna mais rápido e prático para o *Dealer* fazer o *login*, bastando ter uma conta Google. Para os demais participantes da dinâmica, basta criar um *nickname* único por

cada sessão de jogo. A figura 24, abaixo, mostra como ficou a tela de *login* para o *Dealer*. O Google permite fazer o login manualmente como está no centro da imagem, ou ao lado direito quando identifica que já existe um usuário logado, apenas clicando em ‘Continuar como …’ o login já é efetuado.

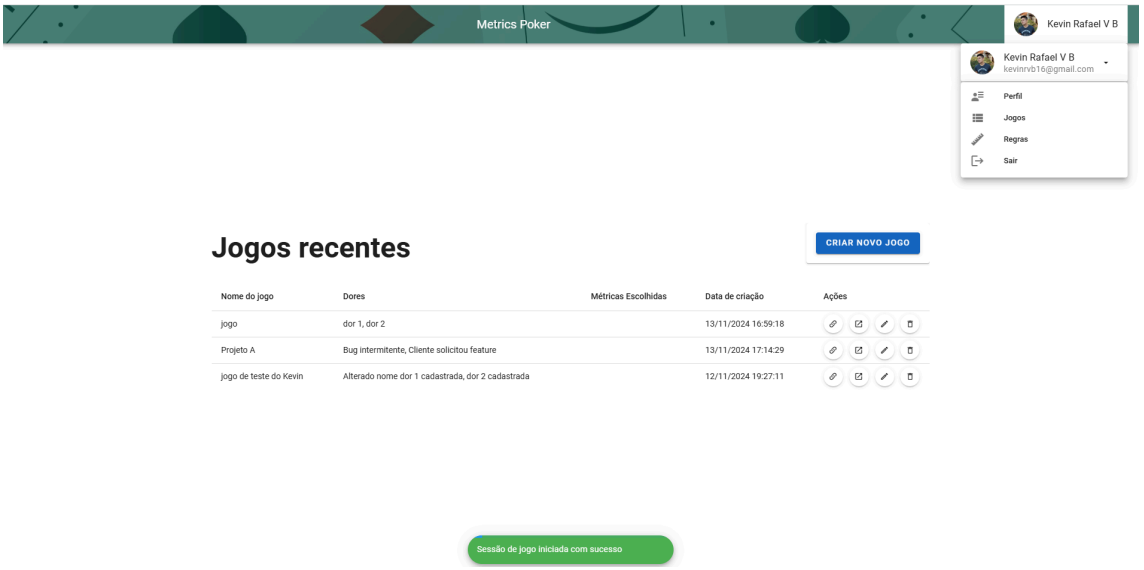
Figura 24 - Tela de Login do Metrics Poker



Fonte: Autor

Na figura 25 é possível visualizar os jogos recentes. É possível também editar as dores e o nome do jogo criado. Em cada linha é possível copiar os links gerados para a sessão de jogo, além de excluir o jogo e de abrir em nova guia o jogo. Para que seja possível acessar essa tela, é necessário estar autenticado.

Figura 25 - Tela de listagem de sessões de jogo do Metrics Poker

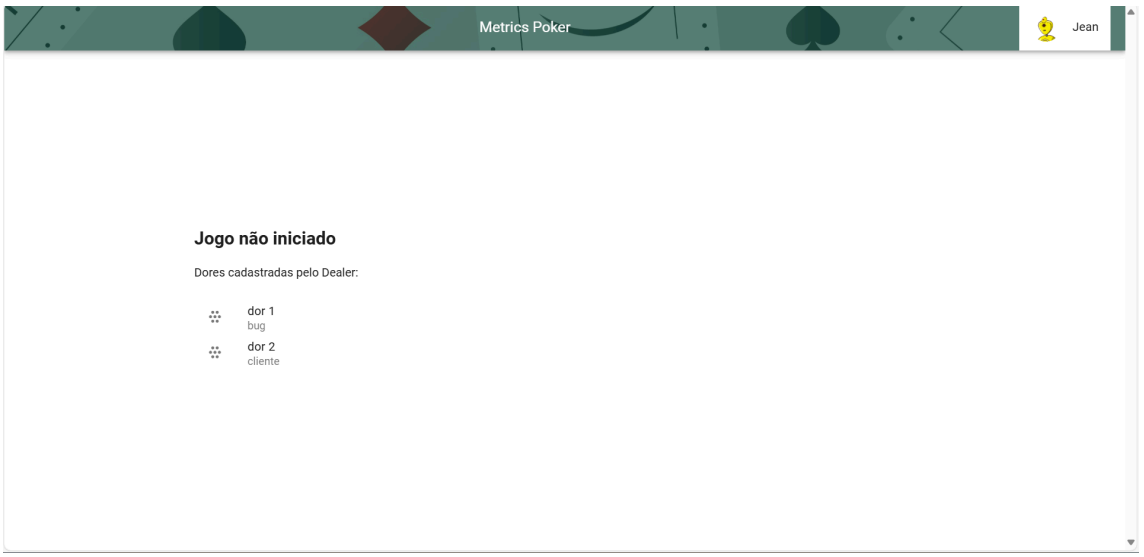


Fonte: Autor

3.7.2.2. RF02 - O sistema deve permitir ao usuário visualizar o tabuleiro

O tabuleiro se dividiu em várias telas para que não fosse informação demais em uma única tela. Portanto, a primeira tela do tabuleiro é essa em que os participantes estão aguardando outros participantes entrarem no jogo.

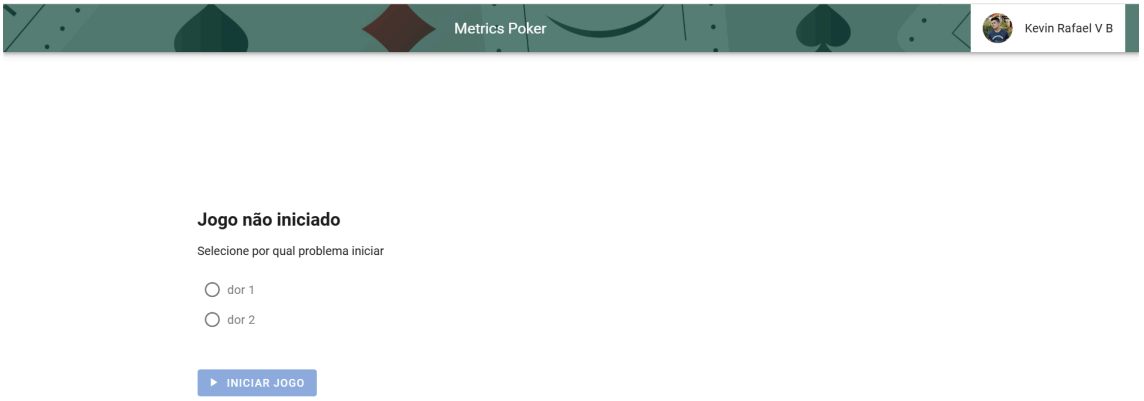
Figura 26 - Print Jogo não iniciado, visão do participante do jogo



Fonte: Autor

Neste momento, existem 2 possíveis telas de acordo com quem está acessando o jogo: a visão da dinâmica por parte do Dealer (figura 27), ou os participantes do jogo que acessaram através do link compartilhado cadastrando seu nickname (figura 26) na sessão de jogo criada pelo Dealer.

Figura 27 - Print Jogo não iniciado, visão do Dealer do jogo

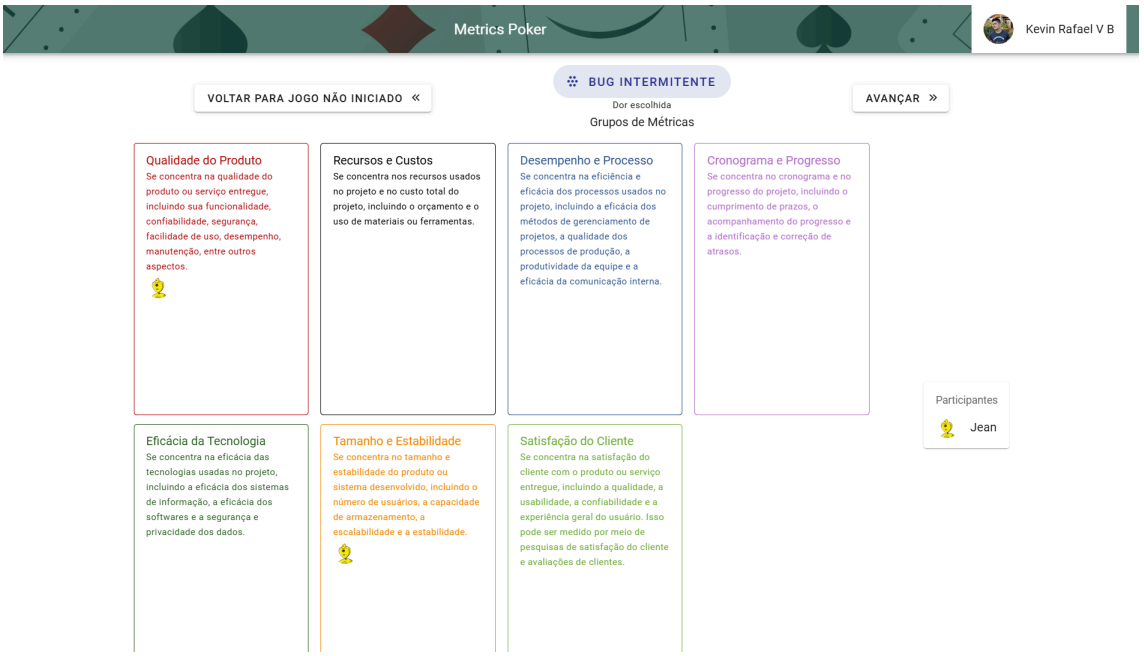


Fonte: Autor

A tela seguinte é de seleção dos grupos de métricas. As duas visualizações, tanto de Dealer como de membro da equipe, estão dispostas a seguir, nas figuras 38 e 39.

Para o Dealer, é possível ver os grupos de métricas com suas respectivas descrições. Além disso, abaixo de cada descrição, vão aparecendo os avatares dos participantes que já votaram nos respectivos grupos de métricas selecionadas, que são 2 por participante. Quando o Dealer perceber que todos votaram, ele é o responsável por dar continuidade à dinâmica clicando no botão superior direito ‘avançar’, conforme demonstra a figura. Ao lado direito, é possível visualizar os participantes da dinâmica com seus avatares criados e respectivos nicknames.

Figura 28 - Tela de seleção de grupos de métricas, o Dealer consegue ver o que cada participante votou



Fonte: Autor

Os participantes podem clicar nas cartas para selecionar, podem selecionar apenas duas, e, ao clicar em enviar, todas ficam desabilitadas. Um ícone de correto aparece no canto superior direito da carta para indicar que ela foi selecionada, indicando um feedback ao participante.

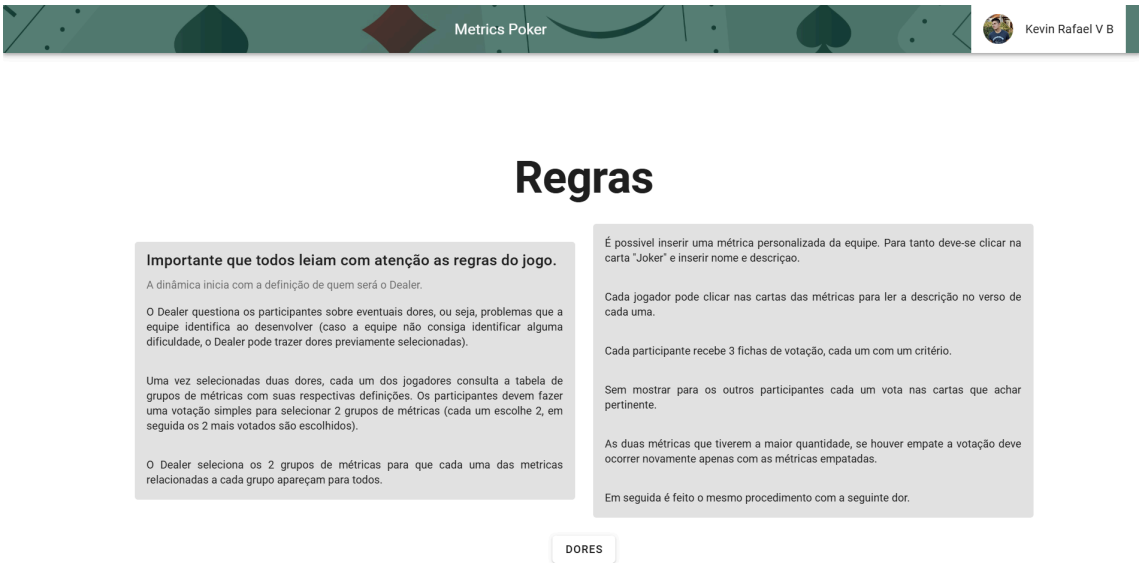
Figura 29 - Tela de seleção de grupos de métricas, visão após o participante ter escolhido os grupos de métricas

Fonte: Autor

3.7.2.3. RF03 - O sistema deve permitir ao usuário visualizar as regras da dinâmica

O Dealer e os jogadores podem ver as regras do jogo ao clicar no menu superior direito e clicar em ‘regras’, onde aparece a tela contendo a informação de cada uma das regras. Na figura 30 é possível ver as regras.

Figura 30 - Print da tela de regras do jogo



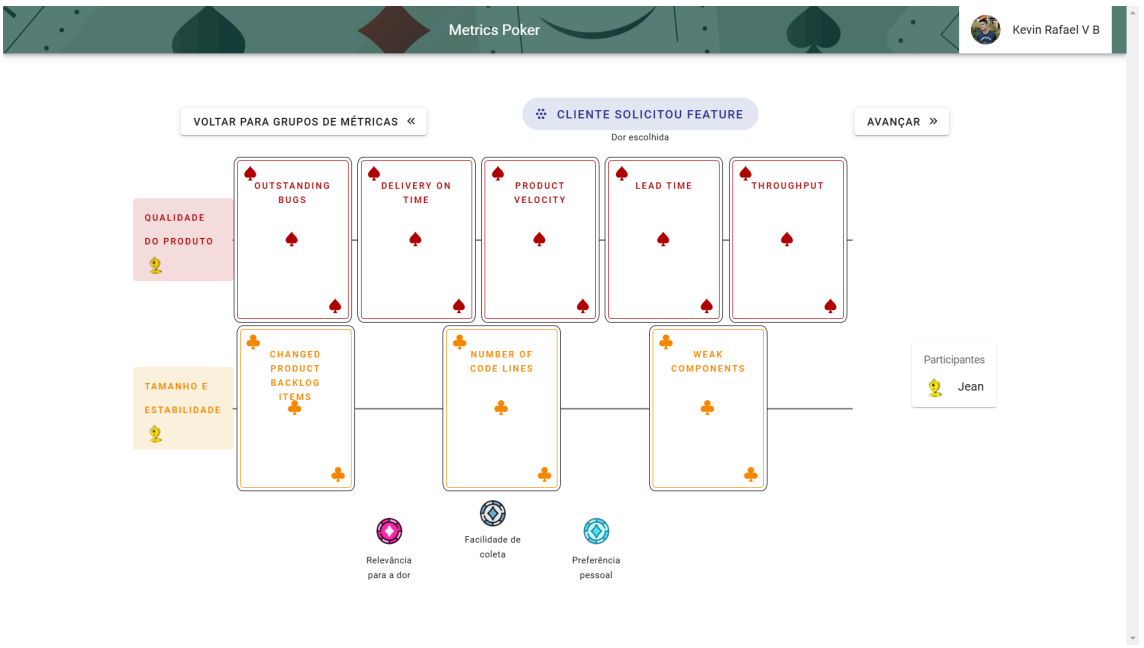
Fonte: Autor

3.7.2.4. RF04 - O sistema deve permitir ao usuário a visualização das cartas

Após a seleção do grupo de métricas, depois que todos selecionaram e o Dealer clicar em avançar, aparece a tela da figura 31, onde é possível visualizar as cartas para votação com cada uma das fichas disponíveis na parte de baixo da página. Na parte de cima continua disponível a dor em análise para acompanhamento, assim como a direita os participantes.

Ao clicar nas cartas, a carta vira e mostra a descrição da métrica, para que fique claro o que ela significa. À esquerda aparece a qual grupo de métrica pertence cada uma das cartas e quais participantes votaram nos grupos.

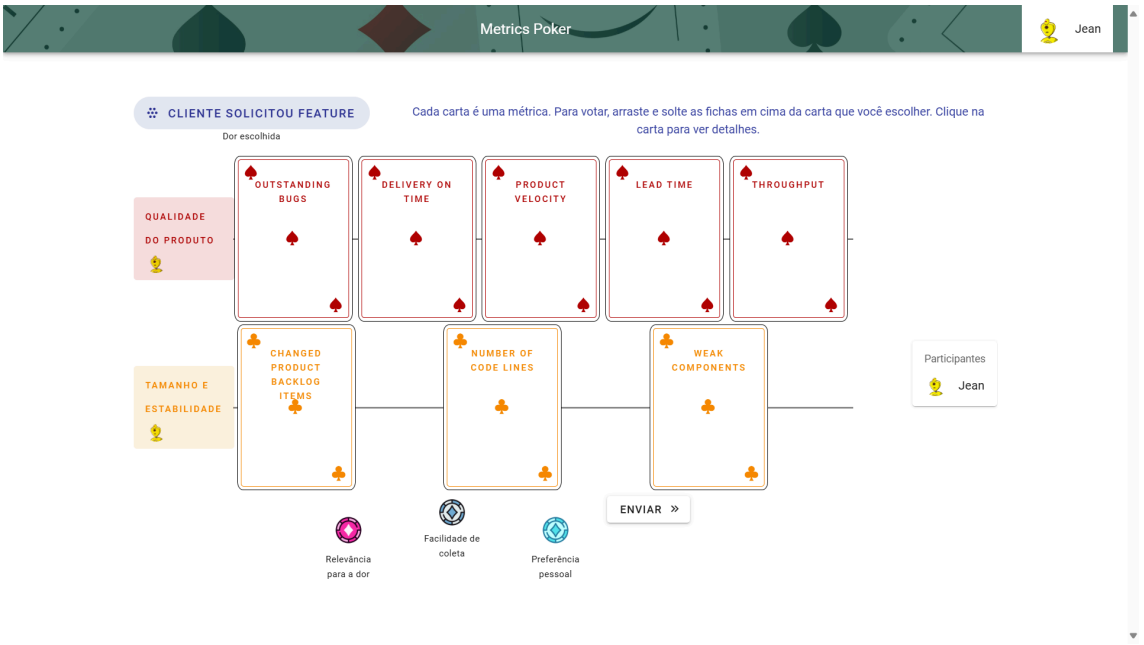
Figura 31 - Print das métricas para votação com as fichas abaixo dela que devem ser arrastadas até as cartas pelos participantes, mas não pelo Dealer



Fonte: Autor

Para os participantes há mais ações disponíveis, pois devem arrastar as fichas abaixo para dentro de cada uma das cartas de acordo com a preferência pessoal, facilidade na coleta e relevância para a dor. Ao clicar na carta, os participantes conseguem visualizar a descrição de cada uma das métricas. Após votar cada ficha em alguma carta, e clicar em enviar, basta aguardar o *Dealer* avançar.

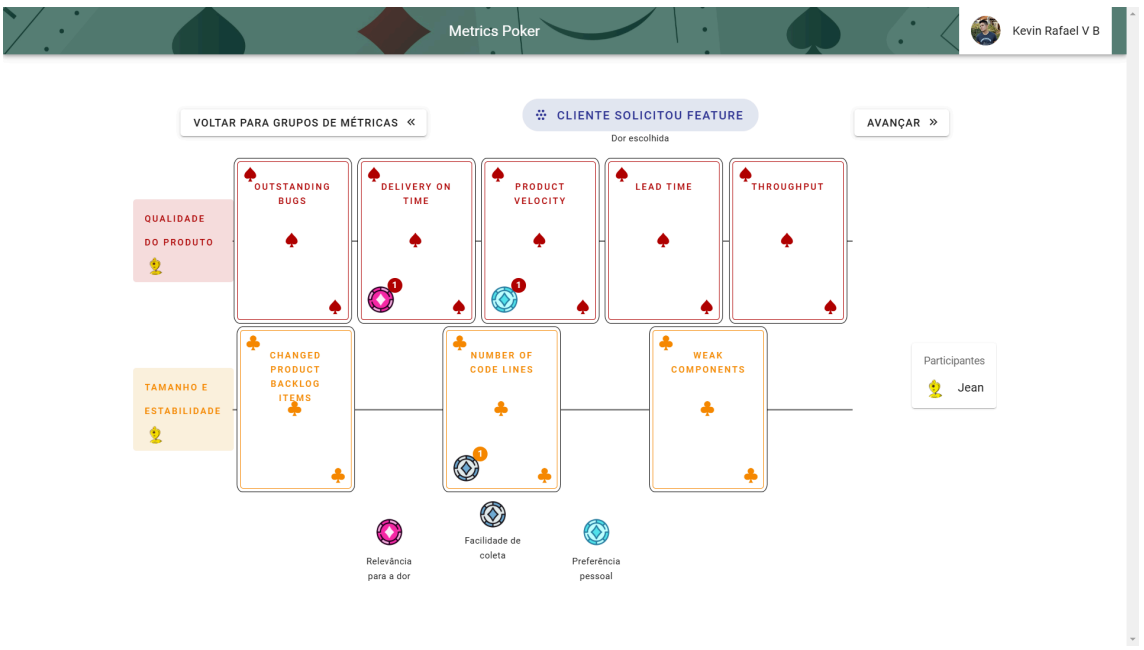
Figura 32 - Visão do participante com opção de arrastar as fichas para cada carta escolhida e depois enviar para o Dealer.



Fonte: Autor

Após a votação, a tela fica da seguinte forma, informando ao Dealer quantas pessoas votaram e mostrando para o membro da equipe um *feedback* de quais cartas foram votadas com as fichas.

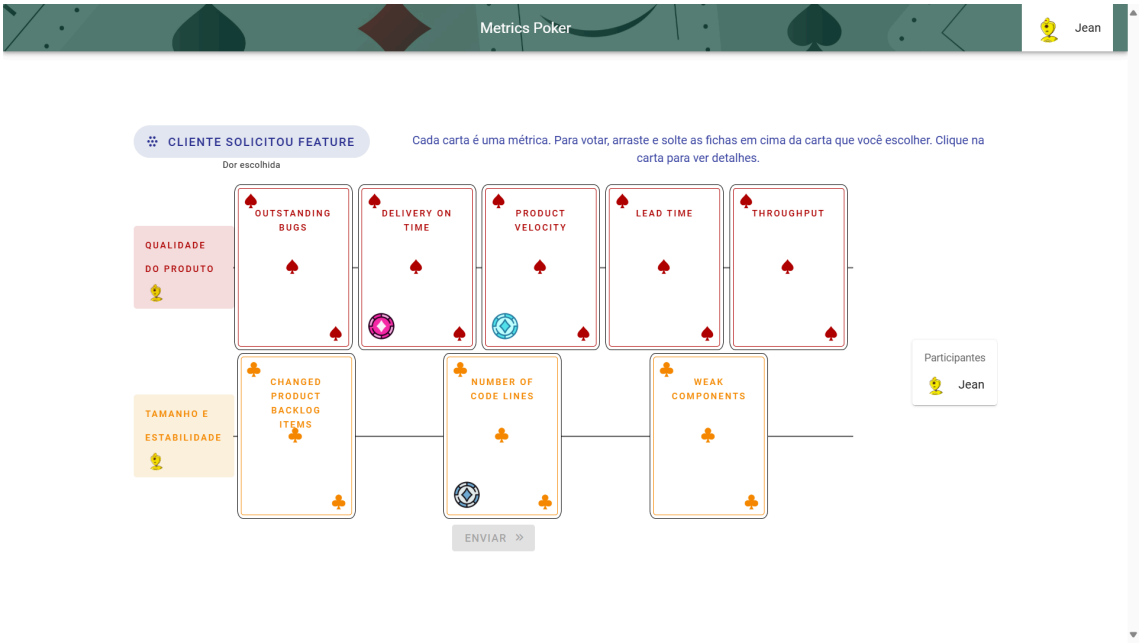
Figura 33 - Print da tela de votar nas métricas. visão do Dealer



Fonte: Autor

A visão para o participante nesse momento é de votação apenas de suas próprias fichas, não sendo possível visualizar a votação dos demais membros.

Figura 34 - Print da tela de votar nas métricas. visão dos participantes



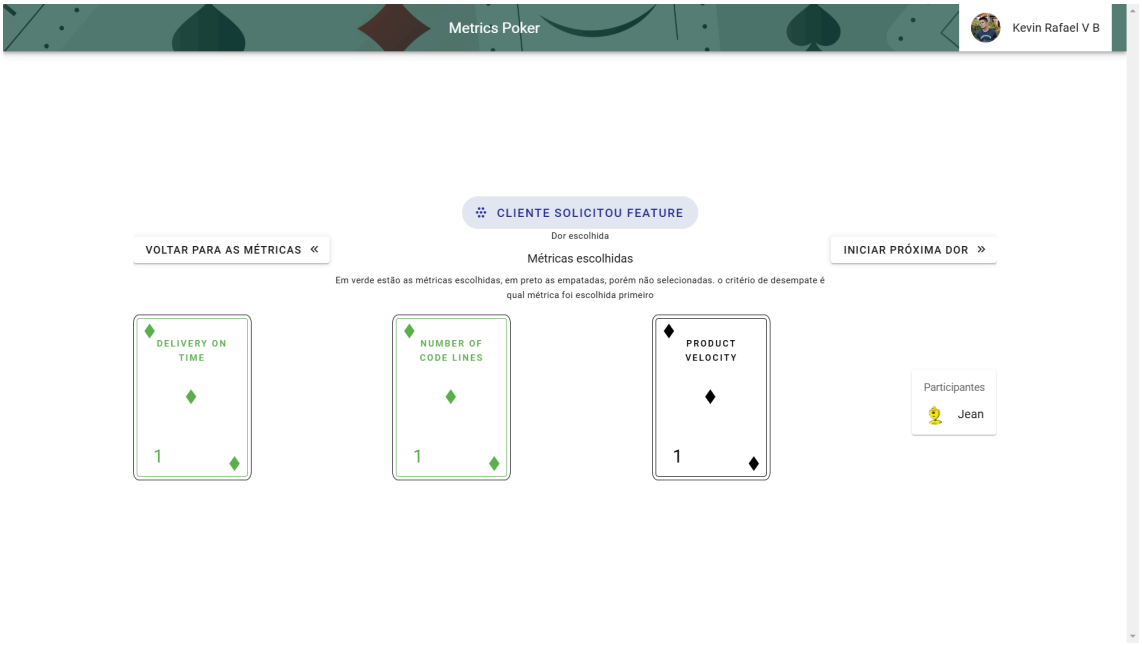
Fonte: Autor

O *Dealer* tem a opção de avançar para ver as métricas mais votadas e, em caso de empate, aparece em preto aquelas que empataram mas que não foram selecionadas pelo critério de que em caso de empate as duas primeiras selecionadas serão escolhidas:

3.7.2.5. RF05 - O sistema deve permitir aos usuários a informação de que a dinâmica acabou

Ao terminar a análise da dor, a figura 44 contém print da tela que possui a informação das métricas selecionadas que indica o fim da análise da dor proposta.

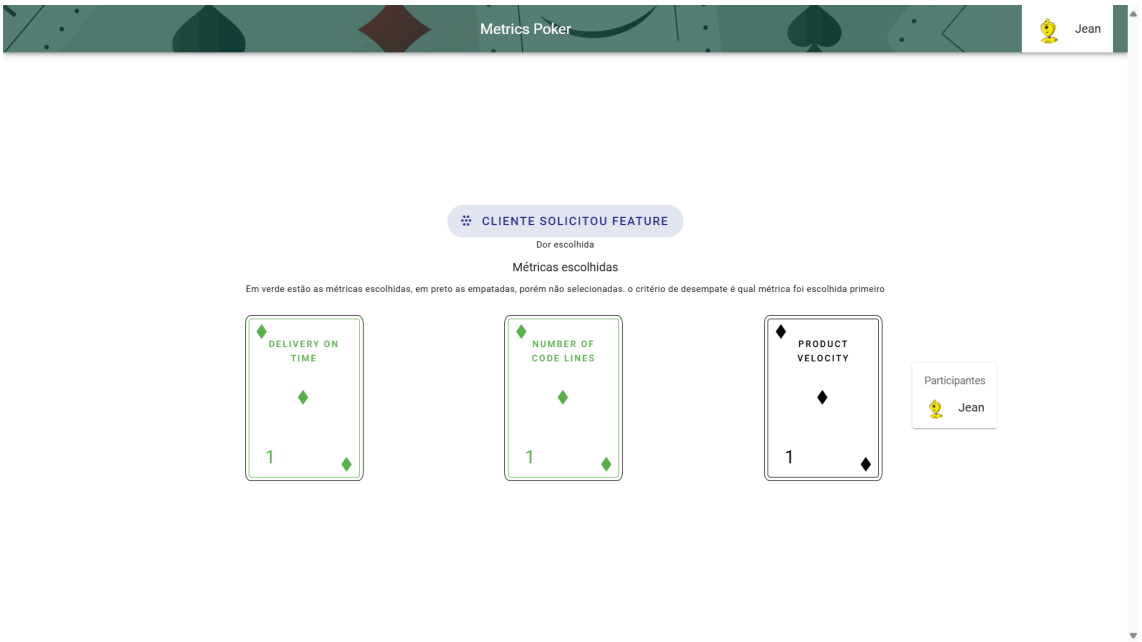
Figura 35 - Print da tela de resultado da votação nas métricas. visão do Dealer



Fonte: Autor

Nessa tela, tanto na figura 35 quanto 36 é possível observar que a visão tanto para o *Dealer* quanto para os participantes é bem similar, alterando apenas as opções de ações de botões que são de responsabilidade do Dealer.

Figura 36 - Print da tela de resultado da votação nas métricas. visão do participante

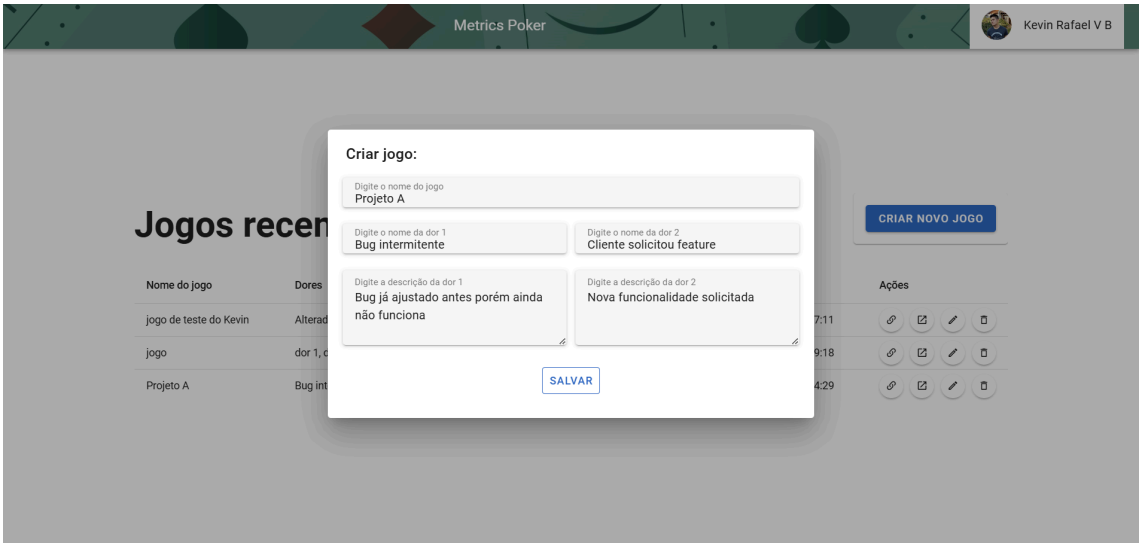


Fonte: Autor

3.7.2.6. RF06 - O sistema deve criar uma sessão de jogo para a partida iniciada

A figura 37 mostra a criação do jogo através de um modal que ao clicar em ‘criar novo jogo’ aparece na tela de listagens de jogos, disponível apenas para o *Dealer*. Todos os campos são validados para que tenham algum conteúdo, não podendo ser cadastrado novo jogo sem nome, dor 1, dor 2 e descrição de cada uma delas.

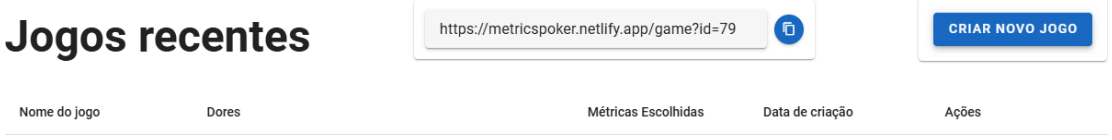
Figura 37 - Print do cadastro de Sessão de jogo e das duas dores



Fonte: Autor

Ao clicar em salvar, aparece o link gerado na tela disponível para ser copiado e compartilhado com os outros integrantes do jogo conforme mostra a figura 38:

Figura 38 - Print do link gerado a partir da criação do jogo



Fonte: Autor

3.7.2.7. RF07 - O sistema deve permitir a cada um dos membros da equipe cadastrar um nome fictício

Figura 39 - Print da tela de cadastro de participante através do nickname

Fonte: Autor

3.7.3. Fluxo de funcionamento

A dinâmica digital online está disponível no link https://metricspoker.netlify.app/ e inicia com uma sessão criada exclusivamente pelo *Dealer* (Professor ou Líder). Os alunos ou membros da equipe acessam a sessão através de um link gerado pela plataforma. A autenticação é feita via login de usuário utilizando o Supabase como biblioteca e Google como provedor OAuth2. O ID do usuário autenticado é armazenado no banco de dados para persistência.

3.7.3.1. Criação da Sessão

O primeiro usuário a logar no sistema assume o papel de *Dealer*, sendo responsável por conduzir as etapas do jogo, independentemente de sua função no time. Ao entrar, o *Dealer* acessa uma tela com uma lista de sessões

existentes e a opção de criar uma nova sessão. Ele define o nome do jogo e cadastra as dores (*problems*) que serão analisadas.

3.7.3.2. Estágios da dinâmica

A dinâmica é dividida em quatro *status* principais:

3.7.3.2.1. Não iniciado (not_started)

Dealer: Pode cadastrar o nome do jogo, as dores, e copiar o link para compartilhamento. Visualização da lista de participantes. Pode mudar o status para "Iniciado".

Participantes:

Apenas visualizam as dores e o link, além da lista de outros participantes. Não possuem ações disponíveis neste estágio.

3.7.3.2.2. Iniciado (started)

Dealer: Visualiza os participantes e pode alterar o status para "Votação" ou "Não Iniciado".

Participantes: Escolhem os próprios grupos de métricas (sem visualizar as escolhas dos demais). Continuam visualizando as dores e os participantes. Em caso de empate na seleção dos grupos de métricas, são escolhidos os dois primeiros grupos selecionados.

3.7.3.2.3. Votação (select_metrics)

Dealer: Visualiza os grupos e métricas escolhidos por cada participante. Pode mudar o status para "Finalizado" ou retornar para "Iniciado".

Participantes: Apostam fichas nas métricas escolhidas. Visualizam os grupos e métricas mais votadas, incluindo a quantidade de votos, assim como o total de fichas disponíveis para votação.

3.7.3.2.4. Finalizado (ended)

O Dealer pode reiniciar o fluxo para uma nova dor cadastrada, retornando ao status "Iniciado".

É possível navegar entre os status sequencialmente, garantindo que etapas não sejam puladas acidentalmente.

4. Conclusão

Este trabalho apresenta o desenvolvimento de um jogo digital baseado na versão física do Metrics Poker, um jogo colaborativo voltado para estimativas de métricas que auxiliam no processo de medição de projetos de software. Inspirado no processo físico de tomada de decisão, o jogo digital busca reproduzir e aprimorar a experiência coletiva, trazendo interatividade e flexibilidade para um ambiente virtual. Além disso, ele visa proporcionar um ambiente de colaboração mais acessível e dinâmico, de modo tal que permita aos usuários participarem remotamente e em tempo real.

Foi realizada uma análise da literatura que abordou temas como gamificação, dinâmicas de jogos colaborativos e metodologias ágeis, com foco nas práticas de estimativa de métricas em projetos de *software*. Essa pesquisa fundamentou o embasamento teórico para a criação de um sistema que pudesse replicar as vantagens do jogo físico em uma plataforma digital, mantendo o engajamento dos usuários.

Para alcançar o desenvolvimento da interface e da lógica do jogo, foi necessário projetar a experiência do usuário, levando em conta o fluxo de interação para garantir uma participação intuitiva e engajante dos jogadores. O uso do framework Vue.js e da biblioteca Vuetify facilitou a criação de componentes responsivos e reutilizáveis, contribuindo para uma interface amigável e coerente com os princípios do *design*.

Por fim, o terceiro objetivo abordou a implementação das funcionalidades que permitissem a atualização das escolhas de métricas em tempo real entre os jogadores. Esta funcionalidade foi projetada para que, ao selecionar uma métrica, os jogadores pudessem ver instantaneamente a escolha dos demais, possibilitando uma colaboração efetiva e uma melhor dinâmica de estimativa em grupo.

Para trabalhos futuros, há a possibilidade de criar um guia de usuário dentro do que já foi desenvolvido, expandir as funcionalidades do jogo, como a implementação de estudos de caso reais, onde o impacto do jogo na precisão das estimativas possa ser avaliado. Em termos de melhorias técnicas, poderiam ser exploradas opções como o agrupamento por equipes, permitindo

que múltiplos grupos de jogadores participem simultaneamente, cada um com seu *Dealer*, responsável pela mediação e organização das jogadas. Essa adaptação abriria caminho para novos modos de jogo e estratégias de interação que favorecem a cooperação e o aprendizado em equipe, considerando que as interações ocorrem em grupo, pode-se integrar uma ferramenta de videoconferência de código aberto, tornando a dinâmica mais prática e eficiente.

5. Referências

ADHIKARI, A. **Full stack JavaScript**: Web application development with MEAN. 40 p. Bachelor of Engineering. Helsinki Metropolia University of Applied Sciences. 2016. Disponível em: https://www.theseus.fi/bitstream/handle/10024/116597/ThesisAA.pdf?sequence=1&isAllowed=y. Acesso em: 5 mai. 2024.

AMINE ABBAD-ANDALOUSSI, On the relationship between source-code metrics and cognitive load: A systematic tertiary review. **Journal of Systems and Software**, 198, 111619, 2023. DOI: ttps://doi.org/10.1016/j.jss.2023.111619.

ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO – SOFTEX. **MPS.BR – Melhoria de Processo do Software Brasileiro**: Guia de Implementação – Parte 2: Fundamentação para Implementação do Nível F do MR-MPS-SW:2016. 2006. Disponível em: https://www.softex.br/wpcontent/uploads/2016/04/MPS.BR_Guia_de_Implementacao_Parte_2_2016.pdf. Acesso em: 21 abr. 2024.

_____. **MPS.BR - Melhoria de Processo do Software Brasileiro**: Guia Geral MPS de Software. 2023. Disponível em: https://softex.br/download/guia-geral-mps-de-software2023/#. Acesso em: 21 abr. 2024.

BAKER, A; NAVARRO, E.; VAN DER HOEK, A. **An Experimental Card Game for Teaching Software Engineering Processes**. Journal of Systems and Software, 2005. Disponível em: https://ics.uci.edu/~emilyo/papers/JSS2005.pdf. Acesso em: 18 mai. 2024.

BAKER, A., NAVARRO, E. & VAN DER HOEK, A. Van Der, “Problems and Programmers: an educational software engineering card game”. *In*: Int.Conf.on Software Engineering, 25. **Proceedings** [...] Portland, OR. 2003.

BATTISTELLA, P.; VON WANGENHEIM, C. **Games for Teaching Computing in Higher Education – A Systematic Review**. IEEE Technology and Engineering Education, v. 1, n. 3, p. 23-33, mar. 2016. Disponível em: https://softwarequalitygroup.paginas.ufsc.br/files/2020/02/ITEE-Games-for-Teaching-Computing-in-Higher-Education_Vdraft.pdf. Acesso em: 18 mai. 2024.

BATTISTELLA, P. **ENgAGED: Um processo de desenvolvimento de jogos para ensino em computação**. 2016. 401 f. Tese (Doutorado em Ciência da Computação) - Universidade Federal de Santa Catarina, Florianópolis, 2016. Disponível em: https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/175816/345443.pdf?sequence=1&isAllowed=y. Acesso em: 28 abr. 2024.

DALL-E 3 . Disponível em: https://www.bing.com/images/create/uma-imagem-simples-com-os-seguintes-passos-e-um-c3adc/1-6632a0ddf80e4318baeec4eb56dd7727?id=bQjNTtLyMFO0hJKAdu7wDw%3d%3d&view=detailv2&idpp=genimg&idpclose=1&thId=OIG4.qwIc00iEJzyCeZySf9p6&FORM=SYDBIC. Acesso em: 3 mai. 2024.

FIELDING, R., **Architectural styles and the design of network-based software architectures**. CHAPTER 5: Representational state transfer (REST). 2000. 162 f. Dissertação (Doutorado em Filosofia) - University of California, Irvine. Disponível em: https://ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. Acesso em: 12 mai. 2024.

HARTEVELD, C.; BEKEBREDE, G.; LO, J.; PLOMBER, A.; JORDAAN, B. **Make It Fun Or Real**: Design Dilemmas And Their Consequences On The Learning Experience. ISAGA Conference, 2011. Disponível em: https://www.researchgate.net/publication/319449031. Acesso em: 18 mai. 2024.

INSTITUTO NACIONAL DE METROLOGIA, QUALIDADE E TECNOLOGIA (INMETRO). **Vocabulário Internacional de Metrologia - VIM 2012**. Traduzido por Grupo de trabalho luso-brasileiro. Duque de Caxias, RJ : INMETRO, 2012. Disponível em:

https://www.gov.br/inmetro/pt-br/centrais-de-conteudo/publicacoes/documentos-tecnicos-em-metrologia/vim_2012.pdf. Acesso em: 21 abr. 2024.

ISO/IEC/IEEE International Standard - **Systems and software engineering – Vocabulary**. ISO/IEC/IEEE 24765:2017(E), p. 1–541, 2017.

JAIN, A. & BOEHM B. SimVBSE : Developing a Game for Value-Based Software Engineering. *In*: Conf. on Software Engineering Education e Training, 19. **Proceedings [...].** Oahu, HI. 2006.

KOHWALTER, T.C.; CLUA, E.W.G. & MURTA, L.G.P., SDM. An educational game for Software Engineering. *In*: Brazilian Symposium on Games and Digital Entertainment. **Proceedings [...].** Salvador, Brazil. 2011.

MADURAPPERUMA, I. H.; SHAFANA, M. S.; SABANI, M. J. A. State-of-art frameworks for front-end and back-end web development. In: International 2011. p. 31-39.

NAVARRO, E. & HOEK, A. Van Der, “SimSE: an interactive simulation game for software engineering education”. *In*: Int. Conf. on Computers and Advanced Technology in Education, 7. **Proceedings** [...] Kauai, HI. 2004.

REIS, L. **Dinâmica de seleção de métricas em projetos ágeis de software**. 10 jul. 2023. Disponível em https://repositorio.ufsc.br/handle/123456789/248524. Acesso em: 18 nov. 2024.

REIS, L. *et al*. Metrics Poker - Uma Dinâmica para Apoiar o Ensino de Medição em Projetos de Software. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 34., 2023, Passo Fundo/RS. **Anais [...].** Porto Alegre: Sociedade Brasileira de Computação, 2023. p. 764-775. Disponível em: https://sol.sbc.org.br/index.php/sbie/article/view/26710/26529. Acesso em: 28 abr. 2024.

SOARES, G.M.; RAUSIS, B.Z. **Development of an educational game for teaching project management in undergraduate computing programs**, Project Thesis, Information Systems, UFSC, Brasil. 2011.

SUESCÚN, E. et al. SimulES-W: A collaborative game to improve software engineering teaching. Computación y sistemas, v. 22, n. 3, p. 953–983, 2018. Disponível em: https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462018000300953. Acesso em: 15 jun. 2024.

TARAN, G., Using games in Software Engineering education to teach Risk Management. *In*: Conf. on Software Engineering Education e Training, 20. **Proceedings [...]**. Dublin, Irleand, 2007, pp. 211–220.

VON WANGENHEIM, C.G.; SAVI, R. & BORGATTO, A.F. DELIVER! – An educational game for teaching Earned Value Management in computing courses. **Information and Software Technology***,* 54(3), 2012, pp. 286–298.

YUNRUI, Q. Front-end and back-end separation for warehouse management system. *In*: International Conference on Intelligent Computation Technology and Automation (ICICTA), 11. **Anais**...IEEE, 2018.

17th Annual State Of Agile Report. *In*: Digital.ai. Disponível em: https://digital.ai/resource-center/analyst-reports/state-of-agile-report/. Acesso em: 6 abr. 2024.
Conference on Science and Technology, 2. 2022 August 24. **Proceedings** [...]. Sri Lanka, 2022. p 62-67. Disponível em: http://ir.lib.seu.ac.lk/bitstream/123456789/6339/3/62-67.pdf. Acesso em: 5 maio 2024.

MITTERMEIR, R. et al., AMEISE – A media education initiative for Software Engineering. *In*: Int. Workshop on Interactive Computer Aided Learning. **Proceedings [...].** Villach, Austria. 2003.

MONSALVE, E. S.; WERNECK, V. M. B.; LEITE, J. C. S. P. Teaching Software Engineering with SimulES-W. *In*: SOFTWARE ENGINEERING EDUCATION CONFERENCE, 2011, Waikiki, Honolulu. **Proceedings […].** Waikiki: IEEE

Leal, S., Hauck, J., Bertan, M., & Vieira, G. (2022, November). How agile organizations use metrics: A systematic literature mapping. In Proceedings of the XXI Brazilian Symposium on Software Quality (pp. 1-11).

6. Apêndices

6.1. Desenvolvimento

6.1.1. Testes da API

Os testes da *API* foram realizados utilizando o Postman, uma ferramenta que facilita o envio de requisições *HTTP* para testar os *endpoints*. Para cada entidade, foram realizados os seguintes testes:

GET: Consulta para recuperar dados de uma entidade, verificando se os registros são retornados corretamente de acordo com os parâmetros
POST: Inserção de um novo registro, com validação de dados e verificação da criação bem-sucedida.
PATCH: Atualização de um registro existente usando o identificador (*ID*), garantindo que os dados sejam atualizados conforme esperado.

DELETE: Exclusão de um registro específico e confirmação de que ele foi removido com sucesso.

Cada requisição foi executada com a *API Key* pública (*anon*) para garantir que os dados pudessem ser acessados ou manipulados conforme as políticas de segurança configuradas.

Abaixo, segue um exemplo de teste realizado no Postman para a entidade *game_sessions*, demonstrando um *GET* para recuperar uma sessão de jogo específica:

- Método: *GET*

URL:
https://clrfpadopbydsjiqejxb.supabase.co/rest/v1/game_sessions?id=eq.{*ID*}

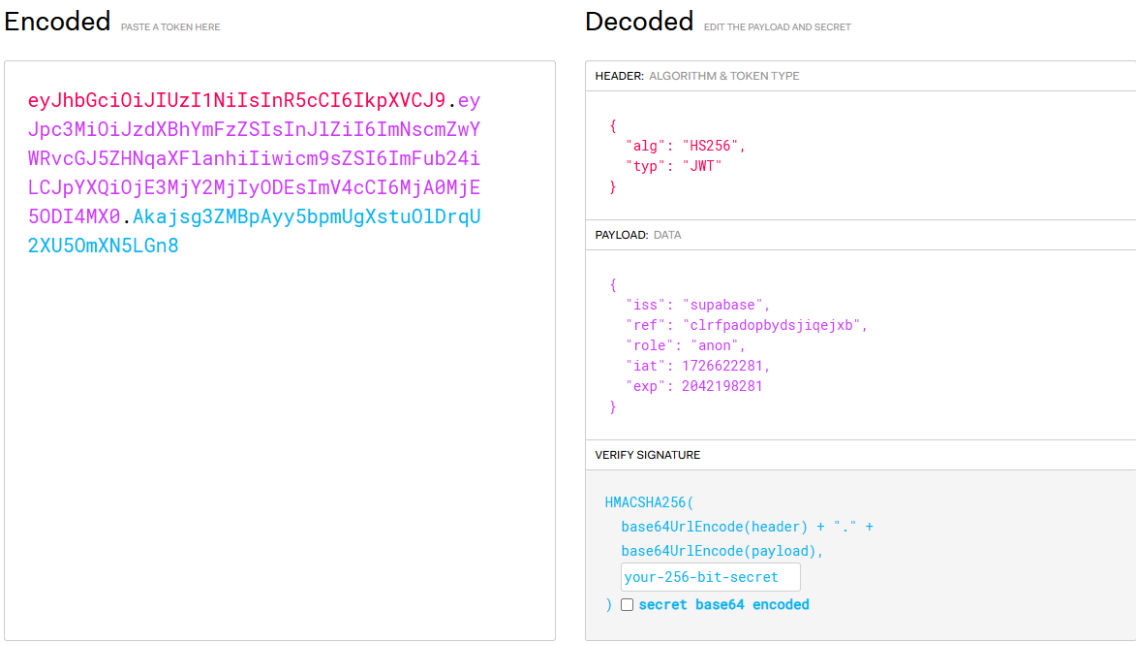
- Cabeçalhos:

apikey:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3…OlDrqU2XU5OmXN5LGn8

Esta APIKey é pública.

Abaixo a APIKey decodificada com as informações necessárias para autenticação:

Figura 40 - apikey decodificada



Fonte: jwt.io

- *Authorization: Bearer {API_KEY}*

- Resposta Esperada:

[

{

"id": 76,

"created_at": "2024-11-12T22:27:11.987188+00:00",

"created_by": "04bf26b1-17df-46fe-a94a-53e5b050bf1d",

"name": "jogo de teste do Kevin",

"status": "ended",

"problemB": 23,

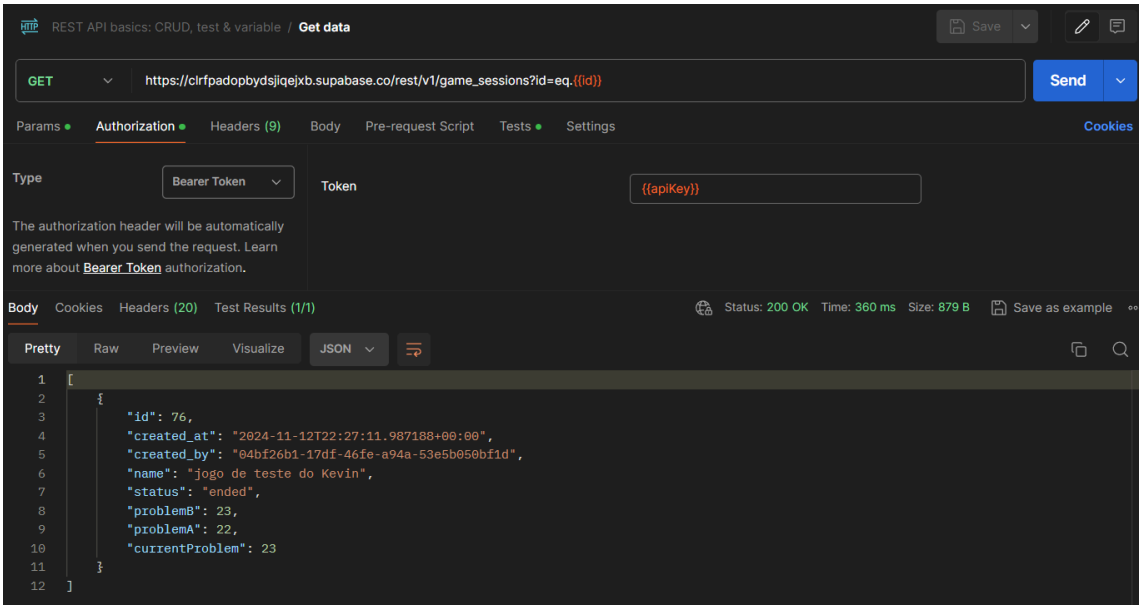
"problemA": 22,

"currentProblem": 23

```
    }
]
```

Conforme mostra a figura 41 abaixo o teste ocorreu com sucesso

Figura 41 - Resultado de teste de endpoint para trazer dados da Sessão de Jogo.

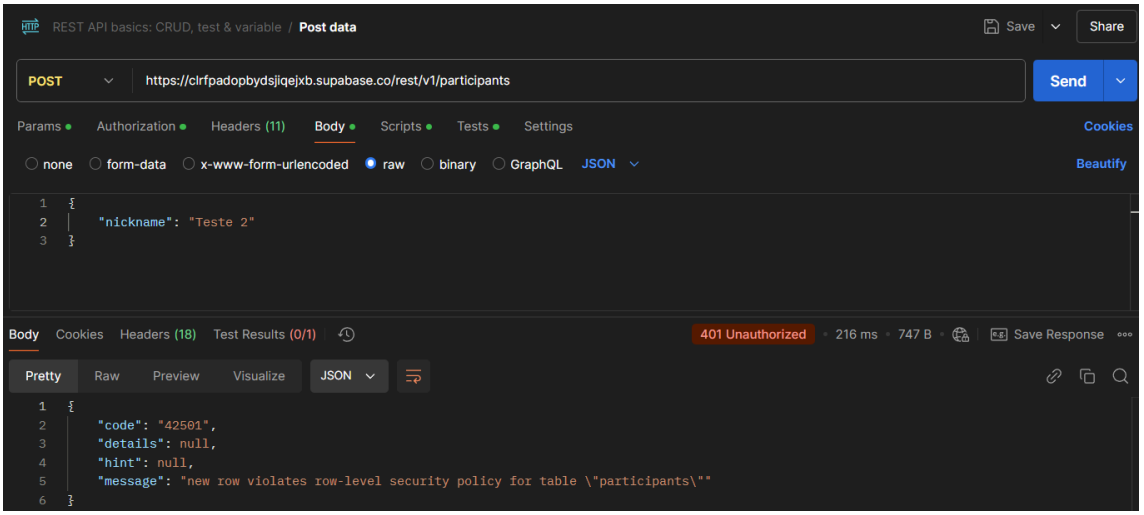


Fonte: Autor

A seguir, há outro teste feito com a entidade *participants*.

A *APIKey* é a mesma utilizada anteriormente, como ela é pública, o endpoint retornou com erro de *RLS (row-level security)*, pois essa inserção na tabela *participants* só pode ser feita com a chave privada.

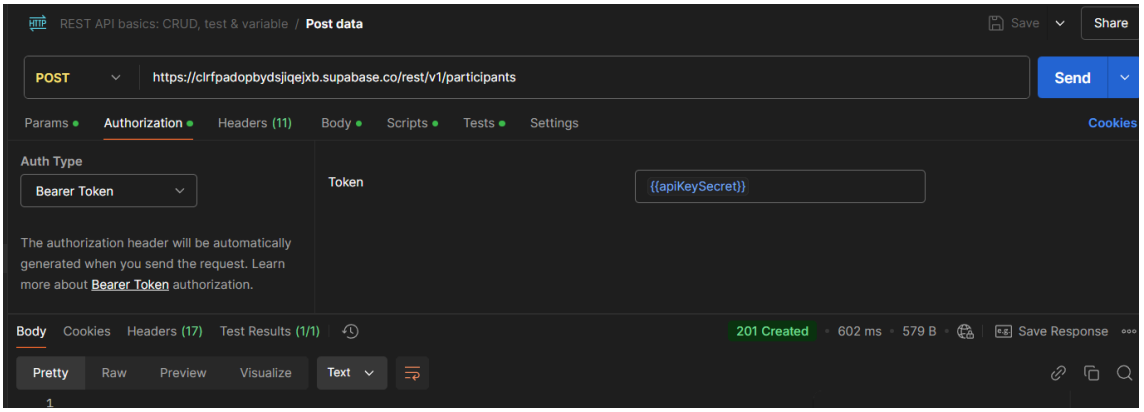
Figura 42 - Erro de row-level security quando chave privada não é utilizada.



Fonte: Autor

O teste com a chave privada na figura 43 mostra que foi possível adicionar corretamente um participante quando a política de segurança é respeitada.

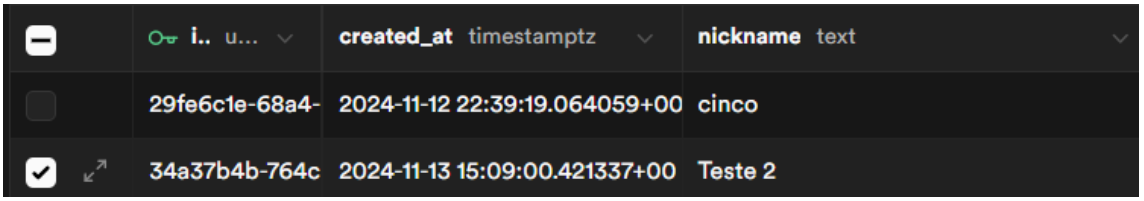
Figura 43 - Cadastrar participante com a chave privada correta.



Fonte: Autor

Na figura 44 é possível observar um print do banco de dados em que o participante com nickname ‘Teste 2’ foi inserido com sucesso.

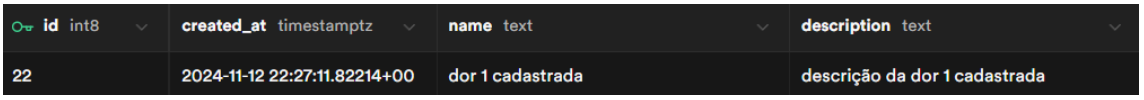
Figura 44 - Participante com nickname igual a ‘Teste 2’ no banco de dados



Fonte: Autor

Em seguida, um terceiro teste foi feito com a entidade de *problems,* que são as dores cadastradas pelo dealer no momento da criação do jogo. A figura 45, abaixo, mostra que o nome da dor 1 era ‘dor 1 cadastrada’ e através do endpoint foi alterado para ‘alterado nome dor 1 cadastrada’:

Figura 45 - Banco de dados antes da alteração.

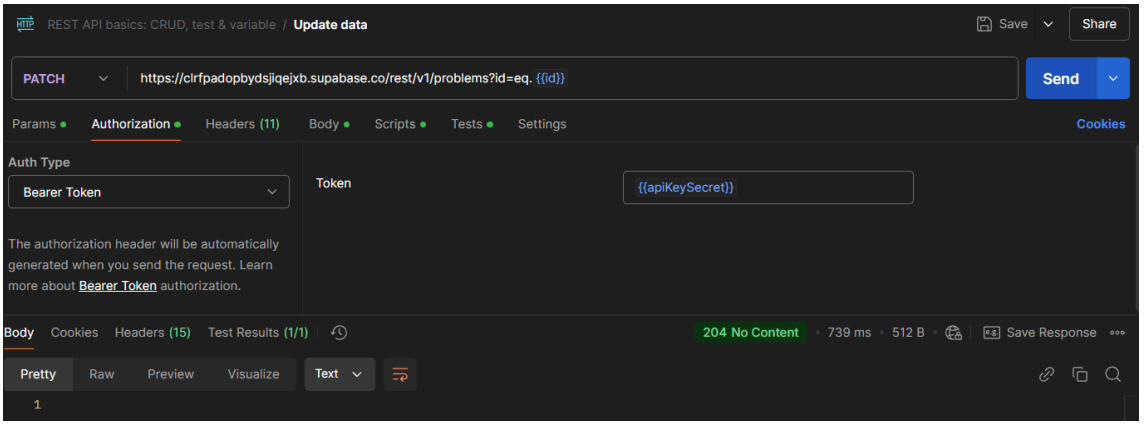


id int8	created_at timestamptz	name text	description text
22	2024-11-12 22:27:11.82214+00	dor 1 cadastrada	descrição da dor 1 cadastrada

Fonte: Autor

Ao acessar o endpoint com o id 22 nos parâmetros, a *apiKeySecret* na *url* correta a alteração é realizada com sucesso no endpoint, essa chave é secreta e portanto não deve ser compartilhada.

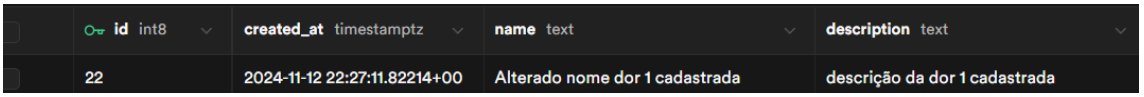
Figura 46 - Abaixo é possível ver o retorno do endpoint com código 204 com sucesso.



Fonte: Autor

Abaixo, na figura 47, a linha na tabela de *problems* que representa as dores com o id 22 foi alterado e reflete a alteração solicitada com o endpoint:

Figura 47 - A alteração de fato refletiu no banco de dados.

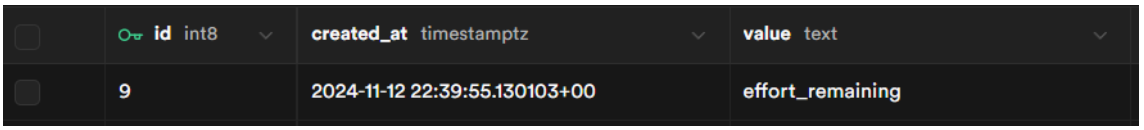


id int8	created_at timestamptz	name text	description text
22	2024-11-12 22:27:11.82214+00	Alterado nome dor 1 cadastrada	descrição da dor 1 cadastrada

Fonte: Autor

O último teste foi de deletar, feito com a tabela *metrics* a linha abaixo, através do id. No teste, a métrica de id igual 9 foi deletada através do Postman.

Figura 48 - Linha na tabela metrics antes de ser deletada através do enpoint



Fonte: Autor

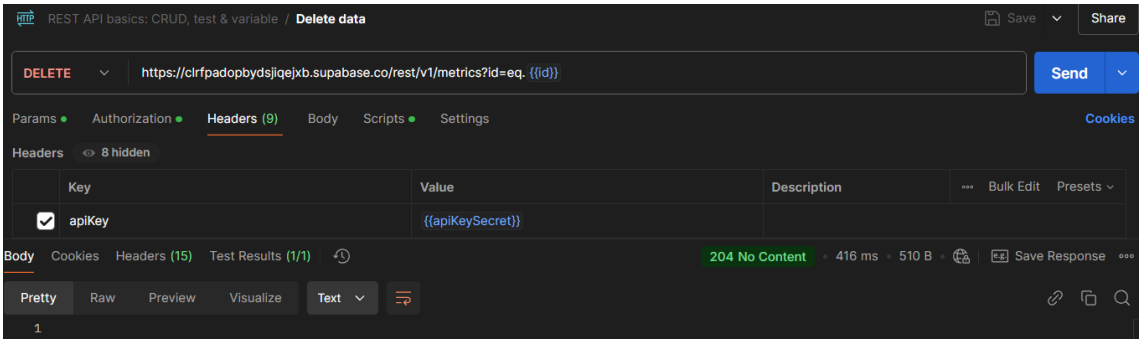
O print a seguir mostra que a métrica de id 9 não consta mais no banco de dados.

Figura 49 - Métrica de id igual a 9 deletada com sucesso

Fonte: Autor

Na figura 50 é possível identificar o retorno do postman ao acessar o endpoint de exclusão de métricas.

Figura 50 - Print do postman, em verde retorno 204 indica sucesso ao excluir a métrica



Fonte: Autor

A utilização do Supabase para o backend proporcionou uma solução eficiente e segura para a criação e manipulação de dados via *API RESTful*. A documentação automática e o suporte a políticas de segurança baseadas em papéis facilitaram o desenvolvimento e garantiram controle de acesso aos dados. Os testes no Postman confirmaram que os *endpoints* funcionam conforme o esperado, garantindo que as operações *CRUD* estão devidamente configuradas e seguras.

6.2. Artigo

Dinâmica Digital Gamificada para Seleção de Métricas Ágeis de Software

Kevin Rafael Velez Bernal[1]

[1]Departamento de Informática e Estatística - Universidade Federal de Santa Catarina (UFSC) - Florianópolis, SC - Brazil

kevinrvb16@gmail.com

Abstract. *The selection of appropriate metrics for agile software projects is a challenge, especially for distributed teams. This paper presents a digital solution for Metrics Poker, a gamified dynamic originally designed as a physical activity, aimed at supporting agile teams in the collaborative selection of metrics. The proposed solution leverages Vue.js on the front-end and Supabase on the back-end to provide an interactive and accessible experience. Results show that the digitalization improves usability and accessibility for remote teams while preserving the engagement of the original physical dynamic.*

Resumo. *A escolha de métricas adequadas para projetos ágeis de software é um desafio, especialmente em equipes distribuídas. Este artigo apresenta uma solução digital para o Metrics Poker, uma dinâmica gamificada originalmente física, projetada para apoiar equipes ágeis na seleção colaborativa de métricas. A proposta utiliza Vue.js no front-end e Supabase no back-end para oferecer uma experiência interativa e acessível. Resultados mostram que a digitalização aumenta a usabilidade e a acessibilidade para equipes remotas, preservando o engajamento da dinâmica física.*

1. Introdução

Métricas ágeis desempenham um papel essencial no monitoramento e melhoria de projetos de software. Contudo, selecionar as métricas mais relevantes para contextos específicos pode ser complexo e suscetível a decisões apressadas. A dinâmica Metrics Poker, inicialmente física, foi concebida para apoiar essa escolha de forma colaborativa e organizada. Com o aumento de equipes distribuídas, surge a necessidade de uma versão digital para expandir sua aplicabilidade.

Este trabalho propõe uma solução digital para o Metrics Poker, que simula uma experiência de jogo de cartas para engajar equipes no processo de decisão. A digitalização visa aumentar a acessibilidade e a eficiência da dinâmica.

2. Trabalhos Relacionados

Jogos educacionais têm sido usados para ensinar conceitos de engenharia de software. Exemplos incluem SimulES-W, que simula o processo de desenvolvimento, e Problems and Programmers, que aborda práticas de engenharia de software por meio de um jogo de cartas. Contudo, nenhum deles aborda a seleção de métricas de forma gamificada, especialmente para métodos ágeis. O Metrics Poker preenche essa lacuna ao integrar a escolha de métricas ao contexto ágil de maneira colaborativa.

3. Proposta

O Metrics Poker digital foi desenvolvido para apoiar equipes distribuídas na seleção de métricas. Os participantes, liderados por um Dealer, seguem duas etapas principais: escolha dos grupos de métricas e votação nas métricas individuais. A interface simula cartas de poker, com detalhes sobre cada métrica disponível no verso.

A implementação utiliza:

- Front-end: Vue.js, para interfaces dinâmicas.

- Back-end: Supabase com PostgreSQL, garantindo persistência e autenticação com Google OAuth2. A figura 1 mostra a estrutura de componentes do backend.

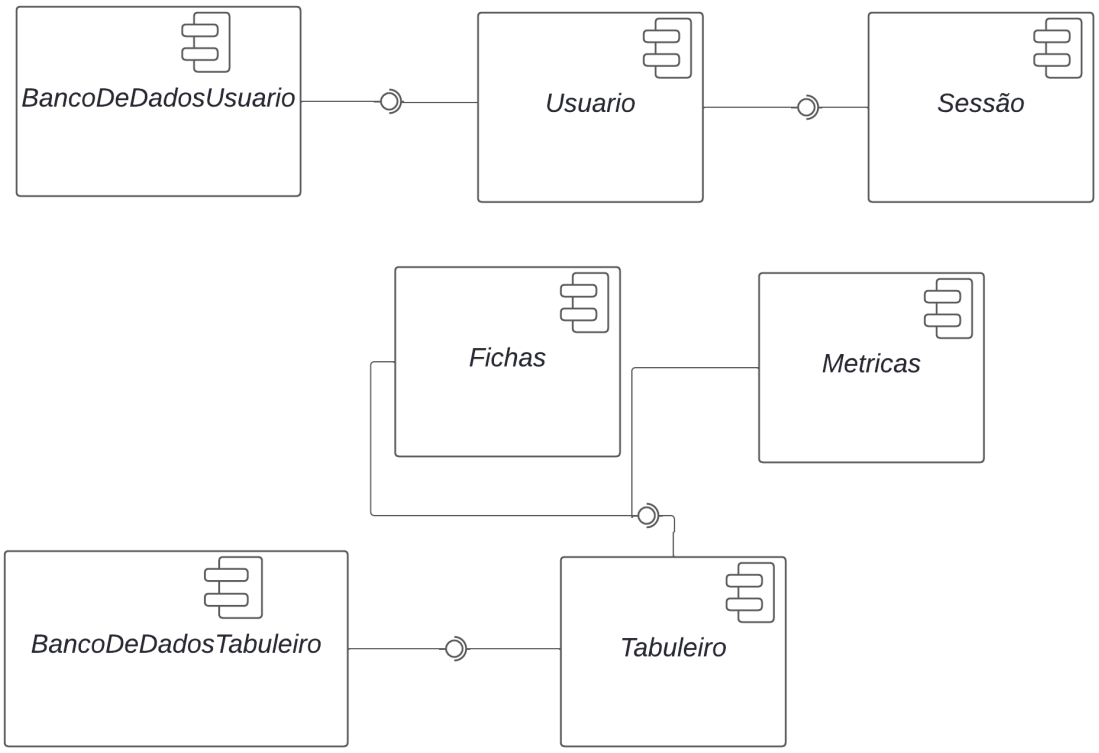


Figura 1 - Diagrama de componentes do backend do software desenvolvido

4. Metodologia

O desenvolvimento seguiu uma abordagem iterativa e incremental, com sprints quinzenais.

As funcionalidades principais incluem: Login de participantes. Criação de sessões de jogo com dores e métricas. Seleção e votação nas métricas por parte dos participantes. Feedback final sobre as métricas selecionadas.

Os requisitos foram definidos com base na dinâmica física original, adaptados para o formato digital.

5. Resultados

A aplicação está disponível em https://metricspoker.netlify.app/ onde é possível ver a tela de login conforme a figura 2:

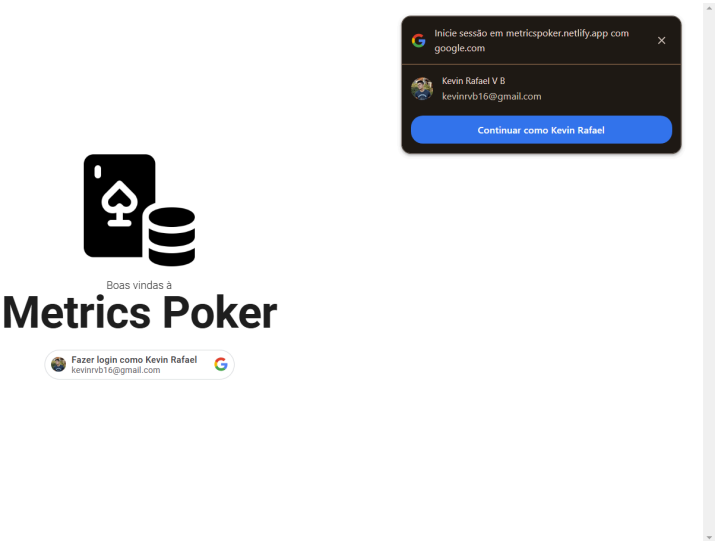


Figura 2 - Tela de Login do Metrics Poker

Após o login é possível visualizar a lista de jogos recentes e menu superior direito conforme a figura 3 mostra:

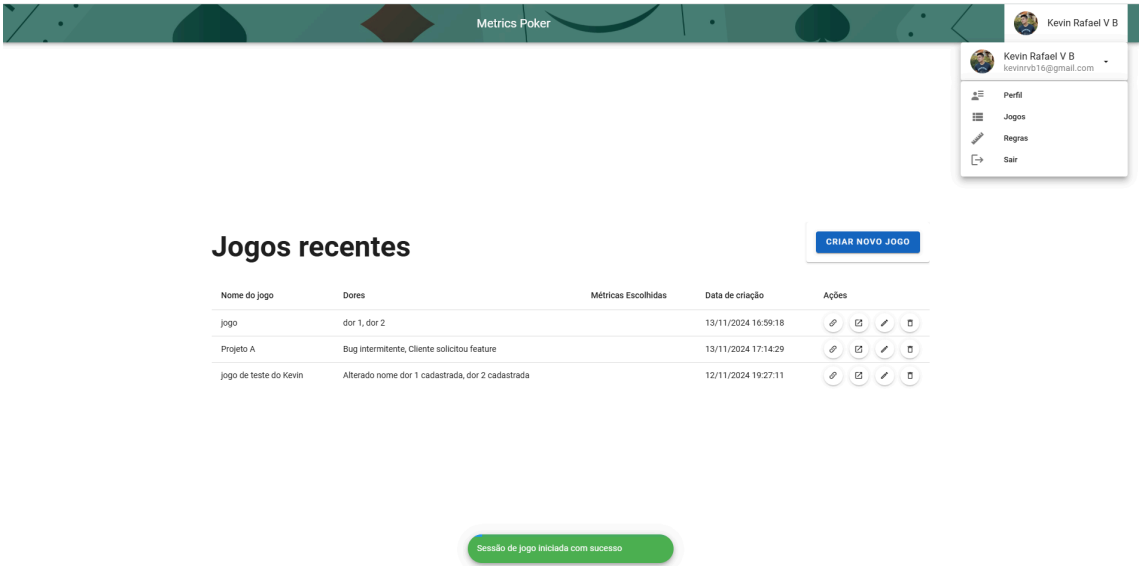


Figura 3 - Tela de listagem de sessões de jogo do Metrics Poker

Clicando em ‘criar novo jogo’ aparece o modal da figura 4, onde é cadastrado os 2 problemas e suas respectivas descrições que serão analisadas na dinâmica.

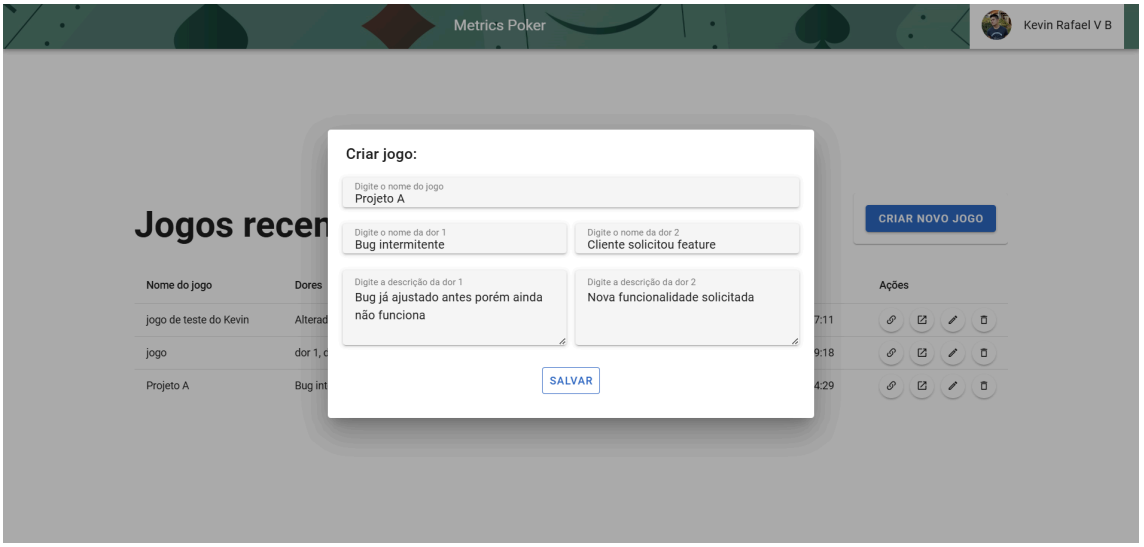


Figura 4 - Print do cadastro de Sessão de jogo e das duas dores

Clicando em 'salvar' no botão da figura 4, o que aparece em tela é o que apresenta a figura 5, que mostra o link gerado para ser compartilhado com os outros participantes.

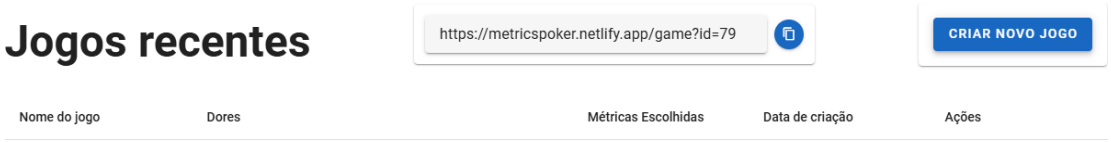


Figura 5 - Print do link gerado a partir da criação do jogo

A partir do momento que o link é compartilhado e outros participantes acessam o jogo, é possível visualizar a tela de acesso em que é necessário apenas o nickname conforme mostra a figura 6

Figura 6 - Print da tela de cadastro de participante através do nickname

Ao acessar com o nickname a tela seguinte indica o status de jogo ainda não iniciado, pois deve-se aguardar até que todos os participantes entrem com seu nickname.

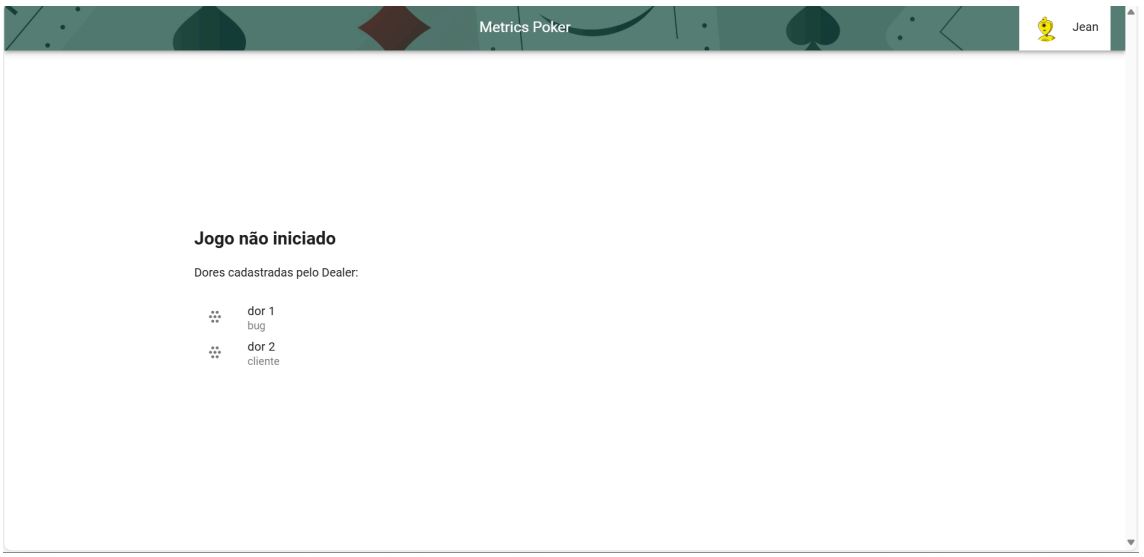


Figura 7 - Print Jogo não iniciado, visão do participante do jogo

Nas figuras 7 e 8 é possível ver o status, os problemas e descrições cadastradas, sendo que apenas o Dealer consegue selecionar por qual delas a dinâmica vai iniciar a escolha de grupos de métricas e em seguida as métricas.

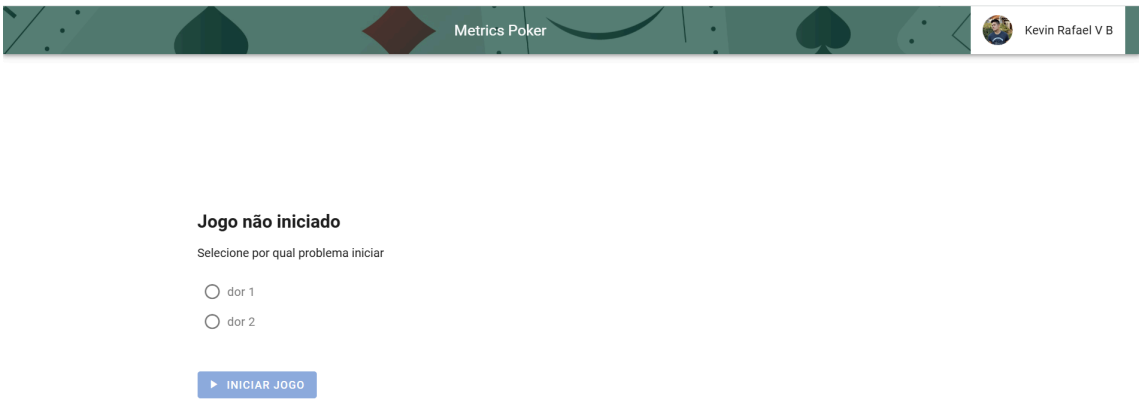


Figura 8 - Print Jogo não iniciado, visão do Dealer do jogo

O Dealer após clicar em iniciar jogo possibilita a mudança de status para jogo iniciado, sendo assim a tela altera automaticamente para ambos os papéis conforme a figura 9 mostra.

Figura 9 - Tela de seleção de grupos de métricas, o Dealer consegue ver o que cada participante votou

O grupo de métricas é apresentado para todos, sendo que apenas os participantes podem clicar selecionando apenas 2 das 7 cartas apresentadas. Assim que cada participante seleciona a carta e clica em 'enviar' como está na figura 10, o Dealer consegue visualizar quem votou e em quais cartas votou.

Figura 10 - Tela de seleção de grupos de métricas, visão do participante após ter escolhido os grupos de métricas

O Dealer é responsável por clicar em 'avançar' para mudar o status e ir para a tela de seleção de métricas, onde são dispostas os dois grupos de métricas mais votados pelos participantes conforme mostram as figuras 11 e 12.

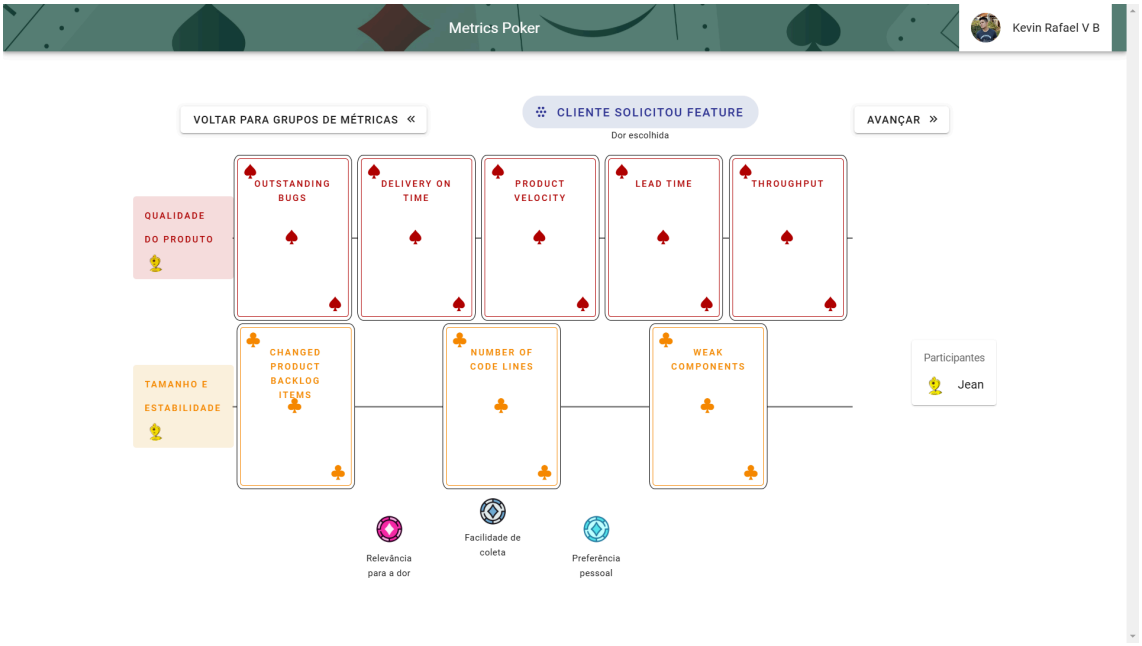


Figura 11 - Print das métricas para votação com as fichas abaixo dela que devem ser arrastadas até as cartas pelos participantes, mas não pelo Dealer.

Os participantes devem neste momento arrastar as fichas para as cartas que escolherem, após clicar em enviar o Dealer consegue ver quais fichas foram colocadas em quais métricas.

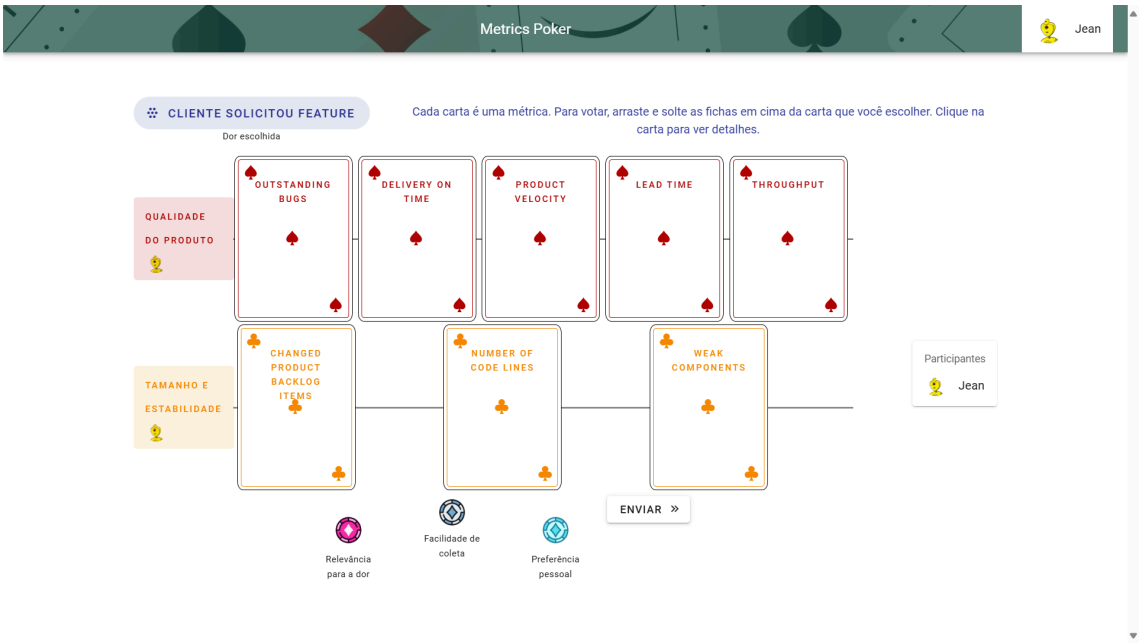


Figura 12 - Visão do participante com opção de arrastar as fichas para cada carta escolhida e depois enviar para o Dealer.

Depois que todos votaram e o Dealer clica em avançar aparece o resultado das duas métricas selecionadas para o respectivo problema/dor conforme mostra a figura 14.

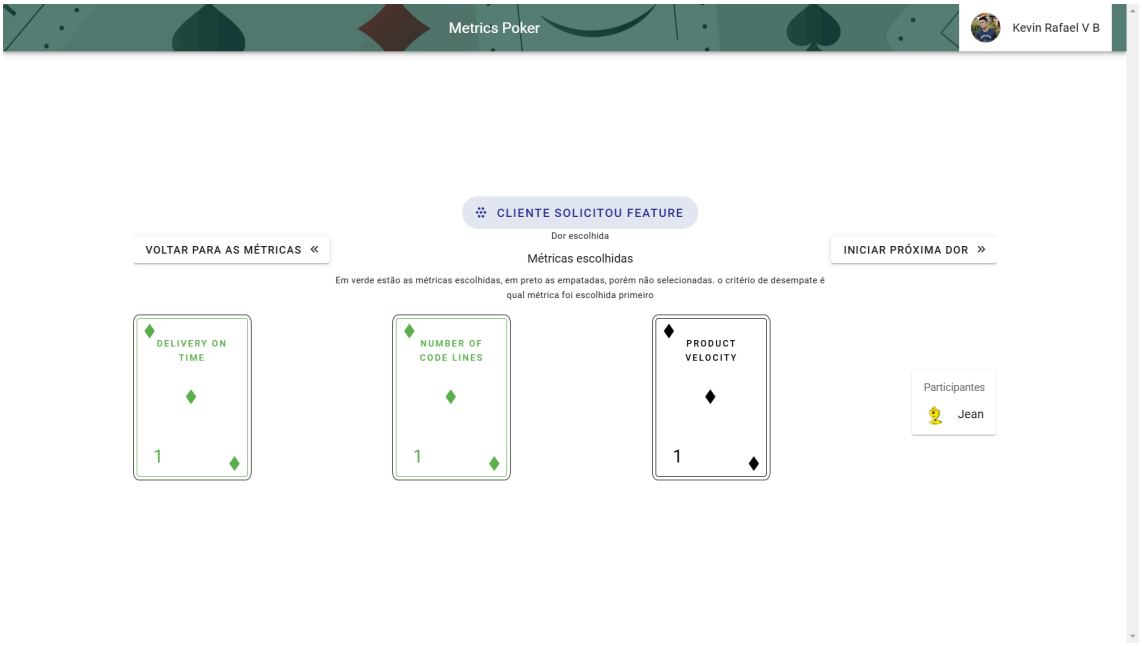


Figura 13 - Print da tela de resultado da votação nas métricas. visão do Dealer

O fluxo explicado anteriormente é repetido para a segunda dor ao clicar em 'iniciar próxima dor'.

A solução digital foi avaliada em termos de usabilidade e funcionalidade. Resultados destacam:

- Acessibilidade: Permitiu que equipes distribuídas participassem remotamente.

- Engajamento: A interface de cartas contribuiu para uma experiência intuitiva.

- Precisão: A integração com Supabase garantiu a consistência dos dados entre sessões.

6. Conclusão e Trabalhos Futuros

O Metrics Poker digital amplia a aplicabilidade da dinâmica física para contextos distribuídos, mantendo o engajamento e a colaboração entre os participantes. Trabalhos futuros incluem a implementação de gamificação adicional e integração com outras ferramentas ágeis, como JIRA e Trello, para maior automatização do processo.

7. Referências

Reis, L. et al. Metrics Poker: Uma Dinâmica para Apoiar o Ensino de Medição em Projetos de Software. CBIE (2023).

Battistella, P. et al. Educational Games Development. IEEE Transactions, (2016)..

Fielding, R. Architectural Styles and the Design of Network-based Software Architectures. UC Irvine (2000).

6.3. Código Fonte

Disponível em https://codigos.ufsc.br/gqs/metrics-poker