INSTITUTO FEDERAL DO PIAUÍ – IFPI
PRÓ-REITORIA DE ENSINO
DIRETORIA DE ENSINO
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

RELATÓRIO TÉCNICO DE SOFTWARE

KAMI: UMA PLATAFORMA PERSONALIZÁVEL E EM TEMPO REAL PARA ORGANIZAÇÃO E SINCRONIZAÇÃO DE SESSÕES DE RPGS ONLINE

ALAN LEAL CARVALHO FILHO

Prof. Me. Aislan Rafael Rodrigues de Sousa
ORIENTADOR

PICOS, PI
2024

ALAN LEAL CARVALHO FILHO

KAMI: UMA PLATAFORMA PERSONALIZÁVEL E EM TEMPO REAL PARA ORGANIZAÇÃO E SINCRONIZAÇÃO DE SESSÕES DE RPGS ONLINE

Relatório Técnico de Software apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Orientador: Prof. Me. Aislan Rafael Rodrigues de Sousa

PICOS
2024

RESUMO

Este trabalho apresenta o desenvolvimento de uma plataforma personalizável para organização e sincronização em tempo real de sessões de RPGs (*Role-Playing Games*) online. O problema abordado é a falta de uma plataforma que atenda tanto jogadores iniciantes quanto experientes, com funcionalidades complexas e que ao mesmo tempo seja de fácil utilização. Além disso, a falta de suporte a dispositivos móveis e as interfaces complicadas das plataformas existentes reduzem o interesse dos usuários e dificultam a utilização das mesmas. Os objetivos da plataforma proposta neste trabalho envolvem a criação de uma interface amigável para atender usuários iniciantes, mas, sem limitar usuários experientes, com elementos que facilitem a compreensão das funcionalidades da plataforma, como a criação de fichas de personagens e de macros para rolagem de dados, além da adaptação para que a plataforma esteja disponível por completo para dispositivos móveis. Com a conclusão do projeto pode-se constatar que o produto final obteve uma boa aceitação por parte dos usuários e conseguiu atingir os objetivos propostos.

Palavras-chave: RPGs online; plataforma personalizável; sincronização em tempo real; gerenciamento de sessões.

SUMÁRIO

1. INTRODUÇÃO

A partir do ano de 2020, houve um aumento significativo na procura por novas formas de entretenimento em grupo de forma online, sendo este causado pelo início da pandemia de Covid-19, uma vez que, com o isolamento social, surgiu uma grande necessidade de socialização sem que houvesse interações presenciais (NIC.br).

Uma das soluções encontradas pelas pessoas foram os jogos eletrônicos, de acordo com uma pesquisa feita pela bandeira de cartões Visa, as transações relacionadas a plataformas de jogos tiveram um aumento de 140% em 2020 quando comparado ao ano de 2019 (VALOR INVESTE, 2021).

Vindo do inglês *Role-Playing Game*, o RPG, é um jogo onde um grupo de pessoas se reúne de forma presencial ou virtual, onde cada um irá interpretar um personagem em um mundo fictício para atingir um objetivo final predeterminado por um outro jogador que será o mestre da campanha (RPG TIPS, 2022).

Os RPGs acabaram sendo uma boa opção encontrada pelas pessoas no meio dos jogos virtuais, apesar de, originalmente serem jogos presenciais de tabuleiro, com o avanço da tecnologia, foram adaptados para serem jogados de forma virtual, porém, essa adaptação ainda possui algumas dores a serem resolvidas.

O presente trabalho se propõe a desenvolver uma plataforma que solucione parte dessas dificuldades que ainda podem ser encontradas durante a fase de criação de elementos necessários para o jogo e de organização dos mesmos durante as sessões, além da formação de campanhas.

Atualmente, já existem algumas plataformas que possuem uma proposta parecida, porém, tais plataformas majoritariamente oferecem funcionalidades complexas de serem utilizadas e baixa personalização, devido ao fato que muitas vezes suas funções são ligadas diretamente a um sistema de regras de RPG específico, além do fato que essas plataformas não oferecem boas experiências para usuários de dispositivos móveis, uma vez que, seus sites não possuem uma boa adaptação para estes dispositivos e também não possuem aplicativos móveis de forma gratuita.

1.1. OBJETIVOS

Para o sucesso de um projeto, é fundamental decidir o que será realizado desde o início. A definição clara dos objetivos a serem alcançados é extremamente importante para manter o foco durante o desenvolvimento, proporcionando um resultado mais eficaz ao atingir o produto final.

A elaboração de um objetivo geral desempenha um papel crucial ao proporcionar uma visão abrangente do que o produto final deve alcançar. Simultaneamente, a definição de pequenos objetivos específicos exerce uma função orientadora fundamental no processo de desenvolvimento, atuando como marcos intermediários que direcionam de maneira incremental a realização da ideia final.

1.1.1. Geral

Desenvolver uma solução personalizável para organização e sincronização em tempo real de sessões de RPGs online, que seja simples de utilizar para jogadores iniciantes e com funcionalidades mais complexas para jogadores experientes, com suporte a múltiplas plataformas e otimização para dispositivos móveis.

1.1.2. Específicos

- Realizar pesquisas sobre RPGs e seu funcionamento para identificar suas reais dores;
- Propor uma interface amigável e intuitiva para a necessidade dos jogadores;
- Implementar as funcionalidades básicas para criação e gerenciamento de fichas de personagens, para jogadores iniciantes;
- Garantir a compatibilidade do sistema com múltiplas plataformas, incluindo dispositivos móveis;
- Realizar questionários de avaliação do sistema final com seus usuários para avaliar possíveis pontos de aprimoramento.

2. TECNOLOGIAS ENVOLVIDAS

O projeto de *backend* foi desenvolvido utilizando Typescript, na versão 4.9.5. O Typescript é uma extensão da linguagem de programação Javascript, com ele, passa a ser possível a utilização de tipagem estática no Javascript (TYPESCRIPT, 2023).

Para o armazenamento de informações foi utilizado o PostgreSQL, versão 15.2. O PostgreSQL é um sistema de gerenciamento de banco de dados relacional (RDBMS) de código aberto e altamente extensível. Conhecido por sua

confiabilidade, conformidade com padrões e recursos avançados (POSTGRESQL, 2023).

Para realizar o acesso e gerenciamento do banco de dados no *backend* foi escolhido o Prisma, versão 4.12.0. Prisma é um *Object-Relational Mapping* (ORM) moderno para Node.js e TypeScript. Ele simplifica a interação com bancos de dados SQL, oferecendo uma interface de programação em TypeScript que permite realizar operações de banco de dados de forma segura e tipada (PRISMA, 2023).

A criação da API do *backend* e gerenciamento de suas rotas foi realizado utilizando Express.Js, versão 4.18.1. Express.Js é um *framework web* minimalista e flexível para Node.js, projetado para simplificar o desenvolvimento de aplicativos *web* e APIs. Ele fornece uma estrutura leve que facilita a criação rápida de servidores *web* robustos e escaláveis em Node.js (EXPRESS, 2022).

Com a necessidade de criar funcionalidades com transmissão de dados em tempo real, foi escolhido o Socket.IO, na versão 4.5.4. O Socket.IO é uma biblioteca JavaScript e Typescript para desenvolvimento de aplicações em tempo real bidirecionais. Construída sobre o WebSocket, ela simplifica a comunicação em tempo real entre clientes (navegadores) e servidores (SOCKET.IO, 2022).

Para o desenvolvimento do *frontend*, foi escolhido o Vue.js, versão 3.2.45. O Vue.js é um framework progressivo de JavaScript para a construção de interfaces de usuário. Sua estrutura se concentra na camada de visualização, proporcionando uma abordagem reativa e baseada em componentes. Também permite a criação de componentes reutilizáveis, facilitando a construção de interfaces complexas divididas em partes menores e gerenciáveis. Além da utilização também do Socket.IO, na mesma versão do servidor, para realizar a conexão com o servidor e outros clientes (VUE.JS, 2022).

Ainda para o desenvolvimento do *frontend*, também foi utilizado a ferramenta Vite, versão 4.0.0. O Vite é uma ferramenta que traz algumas ajudas ao desenvolvedor *frontend* durante o processo de criação de um novo projeto, como a geração automática do código base padrão das tecnologias selecionadas pelo desenvolvedor, assim poupando tempo no início de um projeto, além da criação de um servidor local de desenvolvimento e também geração de *builds* para o ambiente de produção (VITE, 2022).

Para o controle de versão de todo o projeto foi utilizado o Git (GIT, 2024), com o Github como repositório remoto (GITHUB, 2024), devido a sua fácil utilização e grande conjunto de funcionalidades.

A hospedagem da aplicação foi realizada utilizando um servidor privado virtual (VPS) da empresa Vultr (VULTR, 2024), devido a seu baixo custo e bom desempenho, além de uma grande possibilidade de expansão para o futuro do projeto. O servidor utiliza o sistema operacional Ubuntu 20.04 (UBUNTU, 2020) e o Nginx, na versão 1.18.0, como servidor HTTP e *Proxy* reverso, o mesmo foi escolhido por seu desempenho e facilidade de uso e poder ser utilizado para servir tanto às requisições do *backend* quanto do *frontend* sem grandes diferenças de configurações (NGINX, 2020).

3. MODELAGEM DO PROJETO

A modelagem de um projeto é uma forma da ciência e engenharia criar a abstração de um sistema com um certo nível de precisão e detalhe (GOMAA, 2011, p. 3).

3.1. LEVANTAMENTO DE REQUISITOS

Os requisitos são definidos em:

- Requisitos funcionais:
    1. Autenticação e Autorização - Os usuários devem poder se cadastrar, acessar o sistema e gerenciar suas contas, além de poderem decidir se suas criações na plataforma serão públicas ou privadas;
    2. Criação de Fichas de RPG - Os usuários devem poder criar, editar, compartilhar e apagar, da forma que desejarem, fichas de RPG personalizadas;
    3. Criação de Macros de Rolagem de Dados - Os usuários devem poder criar, editar, compartilhar e apagar, como preferirem, macros que executem rolagens de dados, edições de valores em fichas, cálculos de resultados, entre outras funções;
    4. Sincronização em Tempo Real - A plataforma deve sincronizar ações realizadas por usuário através de todos os dispositivos e instâncias sendo utilizadas simultaneamente pelo mesmo em tempo real para garantir uma experiência fluida e evitar discrepâncias nas informações exibidas ao usuário.

- Requisitos não-funcionais:
    1. Disponibilidade - A plataforma e as informações nela armazenada devem estar disponíveis a todo momento, com o mínimo de interrupções possíveis;
    2. Compatibilidade Multiplataforma - O sistema deve ser acessível pela maior gama de dispositivos possíveis, com prioridade para versão WEB responsiva;
    3. Escalabilidade - A arquitetura do sistema deve ser escalável, para lidar com o crescimento no número de usuários;
    4. Desempenho - A plataforma deve oferecer tempos de resposta baixos, criando uma experiência fluida e sem incômodos ou interrupções para o usuário.

3.2. ARQUITETURA DO SISTEMA

Com a popularização dos softwares em todas as áreas, o desenvolvimento de sistemas também fica mais complexo. Desenvolvedores precisam lidar com mais e diferentes tipos de problemas, tanto simples quanto complexos (KARAM; TSUI; BERNAL, 2016, p. 24).

A arquitetura do sistema foi projetada unindo tecnologias atuais e padrões de projeto consolidados com a intenção de criar uma plataforma robusta e eficiente, uma vez que nos requisitos definidos, disponibilidade, desempenho e sincronização em tempo real são pontos muito importantes para a plataforma.

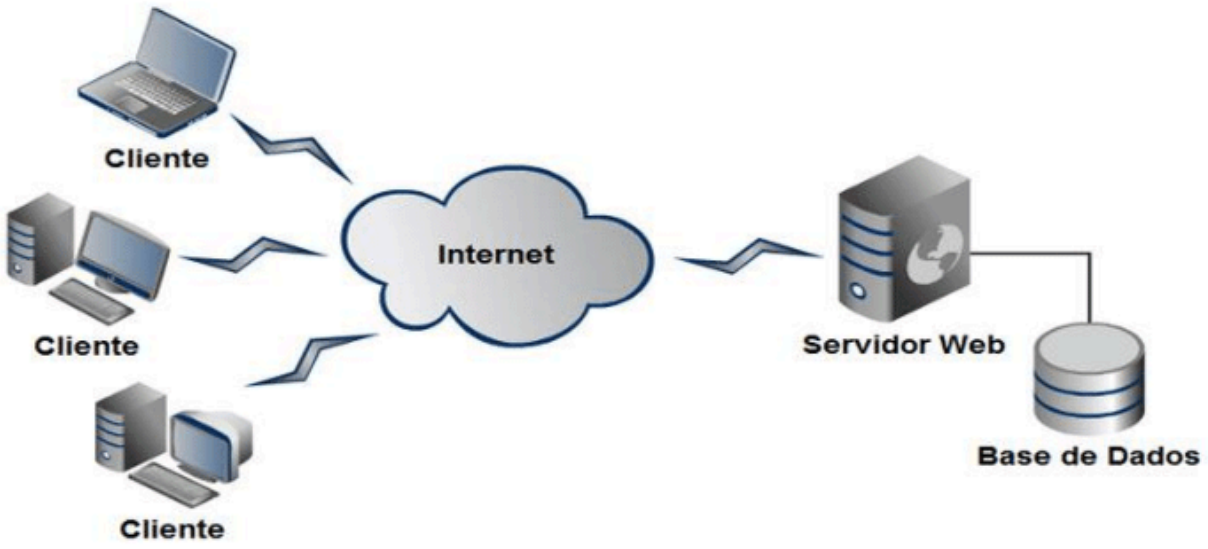
No *backend*, a escolha do TypeScript proporciona uma camada adicional de segurança e clareza no código, visando que a utilização de tipagem estática previne diversos tipos de problemas durante o desenvolvimento, enquanto o Express.js simplifica o desenvolvimento de APIs RESTful. O Prisma, atuando como um ORM, facilita a interação com o banco de dados, garantindo uma abstração eficiente para consultas SQL.

No *frontend*, a arquitetura se baseia na versatilidade do Vue.Js 3 para criar uma Single Page Application (SPA) reativa e de alta performance. A estrutura componentizada promove a modularidade e reusabilidade do código. A utilização do Socket.IO traz um nível de abstração para a utilização de Websockets tanto para o lado de servidor, quanto para o lado do cliente, simplificando significativamente a

utilização desta tecnologia. O uso do Vite também tem um papel fundamental, além de proporcionar um servidor de desenvolvimento eficiente, ele também gera a versão de produção do *frontend*, compilando todos os arquivos com extensão “vue” em três arquivos para serem servidos aos usuários, sendo eles, correspondentes respectivamente a um arquivo HTML, um arquivo CSS e um arquivo JavaScript, assim criando a aplicação final.

O formato arquitetural do sistema se encaixa no padrão de cliente-servidor, onde diversos clientes se conectam através da internet a um servidor e tal servidor estará conectado a um banco de dados, seja ele executado localmente ou em um servidor externo, representado na Figura 1.

Figura 1 - Arquitetura Cliente-Servidor



Fonte: II Congresso Brasileiro de Recursos Digitais na Educação - Scientific Figure on ResearchGate

3.3. INTERFACE

Toda a interface do sistema foi desenvolvida utilizando Vue.Js, na versão 3.2.45. O Vue.Js, permite a criação de interfaces utilizando JavaScript ou TypeScript, HTML e CSS, além de permitir a utilização de módulos externos sem maiores complicações para configuração.

Seus arquivos com extensão “vue” permitem a criação simples de páginas e componentes seguindo um modelo com três seções, sendo elas, *script*, *template* e *style*, onde, além de permitir a utilização dos métodos e atributos nativos do

JavaScript, HTML e CSS, traz outras funcionalidades que reduzem significativamente a complexidade para realizar a criação de páginas e componentes altamente responsivos durante o desenvolvimento, ainda mantendo um bom desempenho.

O design da plataforma aposta em interfaces simples e de fácil entendimento para os usuários, como o exemplo da página inicial na Figura 2, com somente uma breve descrição da plataforma seguida de algumas informações sobre suas estatísticas de utilização.

Figura 2 - Página inicial da plataforma

Fonte: própria. Disponível em: https://kamiapp.com.br/

As páginas que não necessitam de autenticação, como a página inicial, seguem o padrão da Figura 2 com uma barra de navegação ao topo, com um submenu de cascata caso o usuário esteja autenticado, como pode ser visto na Figura 3, nele dando acesso às páginas que necessitam de autenticação, como o *Dashboard*, página de fichas e página de macros, além de um botão para realizar o encerramento da sessão.

Figura 3 - Menu cascata

Fonte: própria.

Já para as páginas onde o usuário deve estar autenticado foi utilizado uma aproximação diferente, trazendo um menu lateral recolhível, que apresenta informações similares ao menu ao topo, porém, com mais destaque nas informações, como pode ser visualizado na Figura 4.

Figura 4 - Página *Dashboard*

Fonte: própria.

Na página de *Dashborad* também se toma uma aproximação simplificada, exibindo somente um acesso rápido das fichas e macros acessados recentemente

pelo usuário, assim reduzindo a quantidade de passos necessários para que o usuário chegue até as funcionalidades da plataforma.

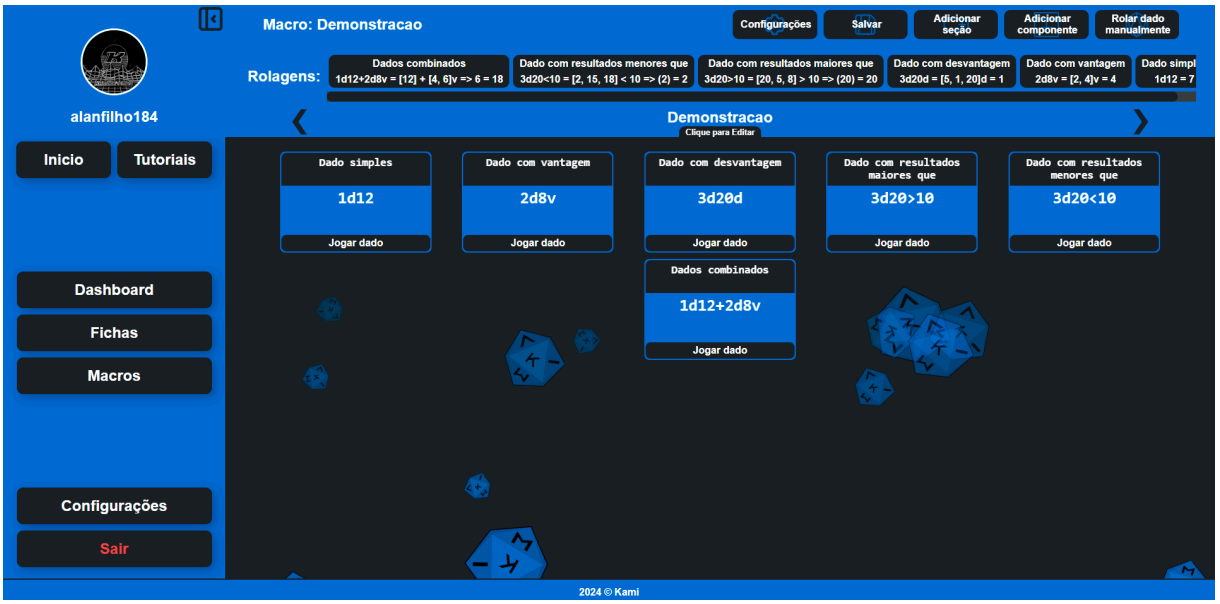
Partindo para a página de uma ficha, um menu com funcionalidades relacionadas diretamente a ficha foi adicionado ao topo para dar fácil acesso a funções essenciais para a criação de fichas de personagem, logo abaixo deste, é exibida uma seção da ficha, esta seção é um conjunto dos componentes adicionados pelo usuário. 5 tipos de componentes estão disponíveis nesta versão da plataforma, sendo eles: Componente de Texto; Componente de Número, Componente de Imagem; Componente de Lista e Componente de Barra, respectivamente apresentados na Figura 5:

Figura 5 - Página de uma Ficha

Fonte: própria.

A página de um macro é similar a da ficha, porém, com algumas pequenas modificações para que se encaixe em seu propósito, no lugar da variedade de componentes da ficha, o macro possui apenas um componente, mas, ele consegue realizar diversas funções, como demonstrado na Figura 6. A similaridade do macro com a ficha além de facilitar o desenvolvimento, reduz a complexidade da plataforma pro usuário, pois, ao entender como um módulo funciona, também conseguirá utilizar o outro sem maiores dificuldades.

Figura 6 - Página de um Macro



Fonte: própria.

Todas as páginas da plataforma foram desenvolvidas para serem responsivas, para assim ser compatível com o máximo de dispositivos diferentes possível. Algumas, além da responsividade, possuem uma versão dedicada para dispositivos móveis.

As páginas para dispositivos móveis seguem um padrão comum, com o botão no canto superior esquerdo que ao receber um clique, exibe o menu de navegação da plataforma, como pode ser visto na Figura 7 referente à página inicial do sistema.

Figura 7 - Página inicial para dispositivos móveis

Fonte: própria.

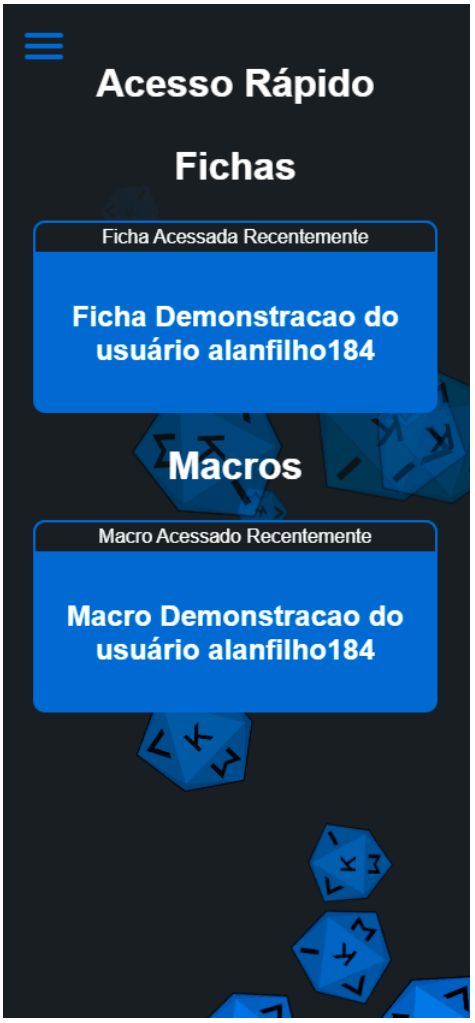
O menu de navegação em dispositivos móveis, visto na Figura 8, só se altera com o estado de autenticação do usuário, com a parte em cinza sendo substituída por um botão para login caso o usuário não esteja autenticado, fora isso, o menu se mantém no mesmo formato por toda a plataforma, diferentemente da versão para desktop.

Figura 8 - Menu para dispositivos móveis

Fonte: própria.

A página *Dashboard*, assim como as outras páginas que dão acesso a funcionalidades, foi desenvolvida para ser responsiva e se adaptar ao dispositivo, como pode ser observado na Figura 9.

Figura 9 - Página *Dashboard* para dispositivos móveis



Fonte: própria.

Já para a página de uma ficha, apesar dela também ser responsiva, ela foi desenvolvida com dois códigos diferentes, onde um é utilizado para dispositivos com telas maiores que 800 pixels de largura e outro para dispositivos menores, na Figura 10 pode-se visualizar a página de uma ficha para dispositivos móveis com telas menores que 800 pixels de largura. Existe uma mudança no posicionamento do menu de ações da ficha e no funcionamento dos componentes, o menu vai para a parte inferior da tela com somente ícones ao invés de botões com texto e os componentes ao invés de utilizarem campos de texto para que possam ser editados diretamente, agora possuem um submenu que abre ao serem clicados com suas

informações expandidas para proporcionar uma melhor visualização em dispositivos pequenos.

Figura 10 - Página de uma ficha para dispositivos móveis

Fonte: própria.

Para a página de um macro também é seguido esse conceito com dois códigos diferentes, a similaridade entre as páginas também se mantém, como pode ser observado na Figura 11.

Figura 11 - Página de um macro para dispositivos móveis

Fonte: própria.

4. SOFTWARE

O produto desenvolvido pode ter seu código-fonte para o *backend* encontrado no seguinte repositorio do Github: https://github.com/ifpi-picos/tcc-kami-backend, já o código para o *frontend* pode ser acessado em: https://github.com/ifpi-picos/tcc-kami-frontend.

A plataforma em si está disponível em sua versão de produção no endereço: https://kamiapp.com.br/ e em sua versão de desenvolvimento em: https://beta.kamiapp.com.br/.

4.1. IMPLANTAÇÃO

Durante o desenvolvimento, foi decidido criar a plataforma de forma modular, sendo a primeira fase, a base para o projeto, a segunda, as fichas de personagem e o terceiro, o sistema de macros.

Seguindo este modelo, a plataforma foi publicada em beta fechada para usuários selecionados assim que sua primeira fase foi concluída, para que com as opiniões de tais usuários ajustes em questões de design, usabilidade e adaptabilidade em diferentes tipos de dispositivos, pudessem ser feitos antes de dar início as próximas fases.

Com a finalização do módulo correspondente a ficha, nesta segunda etapa, foi disponibilizada uma beta pública, para que o máximo de usuários possível pudessem testar as novas funcionalidades e validar seu design, desempenho, responsividade e outras características.

Após a publicação da primeira versão beta pública, o desenvolvimento do módulo de macros se iniciou, e da mesma forma do módulo da ficha, com sua finalização ele foi adicionando a uma nova versão beta pública, ao lado dos outros módulos.

A segunda versão beta pública foi disponibilizada por um período indeterminado para que correções de erros, melhorias de usabilidade e responsividade fossem realizadas, levando como base as experiências reportadas pelos usuários, até que o sistema atingisse um estado satisfatório e uma versão estável fosse atingida e finalmente disponibilizada para todos.

4.2. TESTES

Como já mencionado no tópico anterior, o desenvolvimento do sistema foi realizado em múltiplos módulos com fases de validação junto a usuários reais ao fim do desenvolvimento de cada módulo.

As validações com os usuários foram realizadas permitindo que, um grupo selecionado de usuários utilizassem uma versão de desenvolvimento do sistema, que depois reportaram sua experiência com a plataforma, relatando erros encontrados, sugestões de mudanças, preferências e opiniões de forma direta, além

de eventuais conversas sobre o rumo do projeto, possuindo as mesmas teor técnico em relação a desenvolvimento e em relação ao RPG em si.

Apesar de somente a primeira fase possuir um grupo específico de usuários para realizar estes testes, este mesmo grupo ainda participou das etapas subsequentes, continuando a dar feedback no mesmo formato da primeira etapa, enquanto os demais usuários possuíam uma forma de comunicação indireta, utilizando de formulários para avaliar a plataforma.

5. CONSIDERAÇÕES FINAIS

Mesmo com algumas dificuldades durante o processo de idealização, planejamento e desenvolvimento da plataforma, o produto final foi bastante satisfatório, tendo conseguido atingir seus principais objetivos.

Julgado pelos próprios usuários participantes da versão beta de testes, a plataforma conseguiu entregar uma interface limpa e que deixa claro como o usuário consegue utilizar o sistema, o desempenho da aplicação é bom, não apresentando travamentos, lentidão e afins, a experiência de utilização em diferentes dispositivos é fluída e sem problemas de compatibilidade, uma vez que a interface é responsiva e se a adapta a quaisquer proporções e tamanhos de dispositivos, além de que todas as alterações realizadas na plataforma em um dispositivo, são imediatamente comunicadas a todos os outros que também estejam utilizando daquele recurso, sem que nenhum atraso seja percebido e sem a necessidade de nenhuma ação do usuário.

Futuramente, após a realização das alterações necessárias notadas durante esta fase de testes, mais testes de usabilidade serão realizados, considerando o estado atual do sistema, um teste em escala junto a toda base de usuários é o ideal para este momento. A criação de um sistema de feedback integrado diretamente na plataforma é uma boa solução para conseguir mais resposta dos usuários, já que exige menos esforço para sua realização e causa menos disrupção nos seus fluxos de uso.

O desenvolvimento de software ágil é um processo iterativo e incremental, onde novas funcionalidades serão implementadas e outras serão atualizadas dentro de um período de tempo pré-determinado (VALENTE, 2020).

Durante a fase de desenvolvimento do software e de avaliação pelos usuários, ideias de novas funcionalidades surgiram e por mais que algumas delas não tenha chegado a ser implementadas nesta versão da plataforma, ainda são boas sugestões para o desenvolvimento no futuro do projeto, como, por exemplo: a adição de um novo módulo responsável pelo gerenciamento de campanhas completas, realizando o armazenamento da história base e os personagens não jogáveis, anotações sobre acontecimentos durante as sessões e até mesmo a formação de grupos de usuários para jogar tais campanhas. Outra ideia que surgiu e que também seria uma ótima adição para a plataforma seria um módulo para gerenciamento de conteúdo, como, por exemplo: arquivos de texto; imagens; vídeos e afins, que possam ser facilmente compartilhados dentro da plataforma com outros usuários, durantes sessões, assim ajudando ainda mais na organização das campanhas.

REFERÊNCIAS

EXPRESS. Express: Node.js web application framework. Versão 4.18.1. Disponível em: https://expressjs.com/. Acesso em: 16 fevereiro 2024.

GIT. Git: ”--local-branching-on-the-cheap”. Disponível em: https://git-scm.com/. Acesso em: 16 fevereiro 2024.

GITHUB. Github: Let's build from here. Disponível em: https://github.com/. Acesso em: 16 fevereiro 2024.

GOMAA, Hassan. Software Modeling and Design: Uml, Use Cases, Patterns, and Software Architectures. Cambridge University Press, p. 3, 2011.

II Congresso Brasileiro de Recursos Digitais na Educação - Scientific Figure on ResearchGate, 2013. Disponível em: https://www.researchgate.net/figure/Figura-2-Cenario-proposto-arquitetura-cliente-servidor_fig2_337991059. Acesso em: 16 fevereiro 2024.

KARAM, Orlando; TSUI, Frank; BERNAL, Barbara. Essentials of Software Engineering. Jones & Bartlett Learning, LLC, p. 24, 2016.

LARGHI, Nathália. Com pandemia, mercado de games cresce 140% no Brasil, aponta estudo. Valor Investe, 2021. Disponível em: https://valorinveste.globo.com/objetivo/gastar-bem/noticia/2021/01/23/com-pandemia-mercado-de-games-cresce-140percent-no-brasil-aponta-estudo.ghtml. Acesso em: 27 junho 2023.

NGINX. Nginx: Advanced Load Balancer, Web Server, & Reverse Proxy. Versão 1.18.0. Disponível em: https://www.nginx.com/. Acesso em: 16 fevereiro 2024.

NIC.br (Núcleo de Informação e Coordenação do Ponto BR). 2021. Pesquisa sobre o uso das tecnologias de informação e comunicação nos domicílios brasileiros: pesquisa TIC Domicílios (Edição COVID-19 - Metodologia adaptada), ano 2020. Disponível em: https://cetic.br/pt/arquivos/domicilios/2020/domicilios/

OLIVEIRA, Luiz. O que é RPG de mesa: Motivos para você jogar hoje. RPG Tips, 2022. Disponível em: https://rpgtips.com.br/o-que-e-rpg-de-mesa/. Acesso em: 27 junho 2023.

POSTGRESQL. PostgreSQL: The World's Most Advanced Open Source Relational Database. Versão 15.2. Disponível em: https://www.postgresql.org/. Acesso em: 16 fevereiro 2024.

PRISMA. Prisma: Simplify working and interacting with databases. Versão 4.12.0. Disponível em: https://www.prisma.io/. Acesso em: 16 fevereiro 2024.

SOCKET.IO. Socket.IO: Bidirectional and low-latency communication for every platform. Versão 4.5.4. Disponível em: https://socket.io/. Acesso em: 16 fevereiro 2024

TYPESCRIPT. TypeScript: JavaScript With Syntax For Types. Versão 4.9.5. Disponível em: https://www.typescriptlang.org/. Acesso em: 16 fevereiro 2024.

UBUNTU. Ubuntu: Focal Fossa. Disponível em: https://releases.ubuntu.com/focal/. Acesso em: 16 fevereiro 2024.

VALENTE, Marco. Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade. Editora: Independente, 2020.

VITE. Vite: Next Generation Frontend Tooling. Versão 4.0.0. Disponível em: https://vitejs.dev/. Acesso em: 16 fevereiro 2024.

VUE.JS. Vue.js: The Progressive JavaScript Framework. Versão 3.2.45. Disponível em: https://vuejs.org/. Acesso em: 16 fevereiro 2024.

VULTR. Vultr: The Everywhere Cloud. Disponível em: https://www.vultr.com/. Acesso em: 16 fevereiro 2024.