

UNIVERSIDADE SALVADOR - UNIFACS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Equipe:

João Luccas Marques - RA: 1272212801,

Elaine Santana,

Gabriel Reis badaró,

Guilherme Goes Xavier,

Herbert Lopes Santana,

Sabrina Filgueiras Alves Raiol

RELATÓRIO A3 - Usabilidade e desenvolvimento Web



UNIFACS

Salvador, BA

2023

Descrição dos requisitos da aplicação

Tecnologias necessárias para execução do projeto:

- Node.js

Tecnologias utilizadas:

Servidor:

- Node.js (Express)

Banco de dados;

- Sqlite3

Cliente:

- ReactJs
- React-router-dom
- Vite (Build)

Instruções para instalação e execução da aplicação

1. Instalar o node:

- Faça a instalação do node.js na máquina, segue link do site oficial para dowload: <https://nodejs.org/en/download/current>

2. Clonar o projeto com o git e usar na sua IDE de preferência:

- git clone <https://github.com/devJoaoLuccas/EntregaFinalA3-USD.git>

3. De dentro da pasta raiz do projeto, rodar o comando para instalar as dependências:

- npm install

Após instalar o npm na paz raiz, executar o seguinte comando:

- npm run install:all

4. Executar o server e client:

- No primeiro terminal inicie o servidor de desenvolvimento através do comando:
 - npm run dev:back
- No segundo terminal inicie o cliente através do comando:
 - npm run dev:front

5. Acesse o cliente através da url:

- <http://localhost:5173/>

OBJETIVO

O projeto "The Game Bay" é uma iniciativa focada em proporcionar aos usuários de jogos eletrônicos uma plataforma eficiente e intuitiva para gerenciar suas coleções e descobrir novos títulos. Este relatório destaca o compromisso em aplicar as heurísticas de Nielsen ao design do sistema, buscando garantir a usabilidade, eficiência e satisfação do usuário. No universo dinâmico dos jogos virtuais, onde a diversidade de plataformas, gêneros e comunidades é vasta, surge a necessidade de uma ferramenta que unifique e simplifique o gerenciamento de bibliotecas pessoais de jogos. A "The Game Bay" aspira a ser essa solução, integrando funcionalidades essenciais como cadastro de usuários, autenticação, operações CRUD para jogos e plataformas, categorização de jogos, avaliação e notas, promovendo uma experiência centralizada e envolvente.

Protótipos figma, decisões e heurísticas

A consistência no design foi priorizada para criar uma experiência coesa. Cores consistentes (roxo, preto e rosa) foram escolhidas, já que estas são facilmente associadas ao mundo de jogos, remetendo a cores comumente utilizadas no universo “gamer”, além de fazer alusão à temática cyberpunk, muito comum em diversas paletas associadas a jogos e tecnologia. Além disso, na psicologia das cores a cor roxa é utilizada como liberdade e estabilidade, trazendo um sentimento de algo mais leve. Foram empregadas em todo o site para indicar categorias, botões e áreas interativas, seguindo um padrão de design que facilita a identificação e navegação, se adequando assim à quarta heurística de Nielsen, como é possível analisar na Figura 1, representando a tela de login do “The Game Bay”. É possível analisar que aplicamos as cores principais do projeto de forma que gerasse conforto e leveza para o usuário.

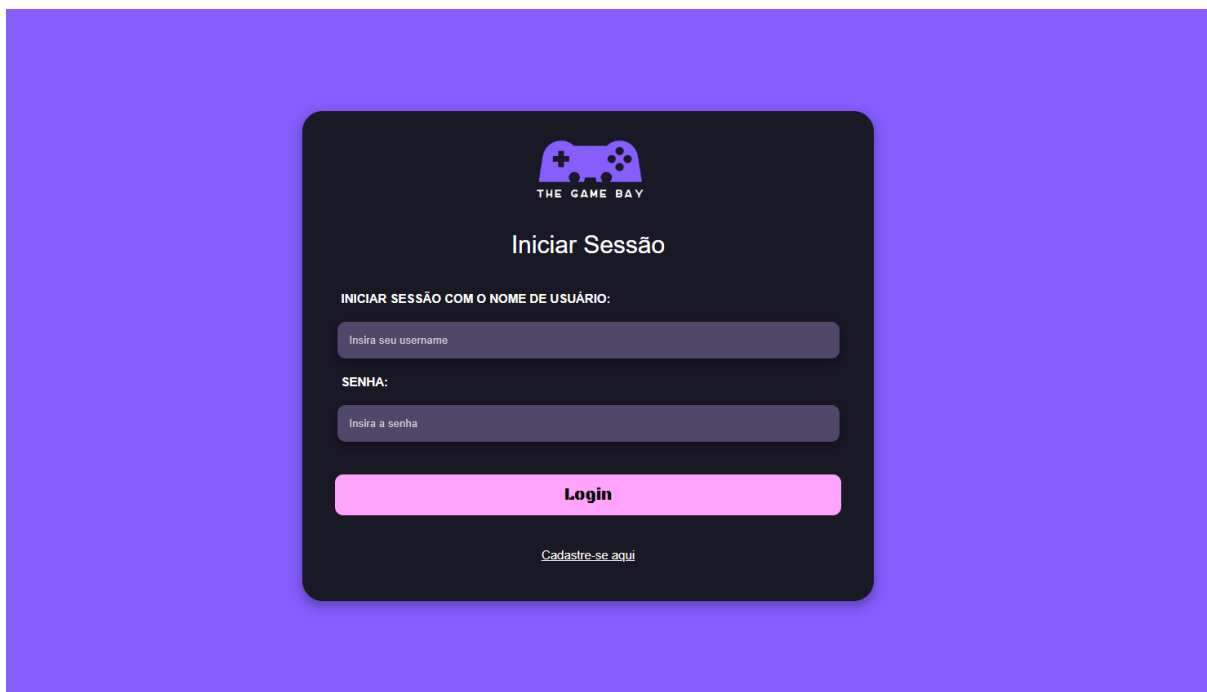


Figura 1: Tela de Login

A 8ª heurística de Nielsen, que propõe estética e designs minimalistas, foi escolhida cuidadosamente para proporcionar uma experiência visualmente agradável e fácil de entender, com o intuito de manter a estética limpa e focada na funcionalidade, para que dessa forma o usuário consiga navegar de forma simples e intuitiva. Além de tornar as informações visíveis, considerando a inclusão de recursos de pesquisa robustos que permitam aos usuários encontrar rapidamente os jogos desejados, mesmo que não se lembre exatamente dos detalhes. Isso pode incluir uma busca intuitiva por nome, categoria, plataforma, etc. Como é possível verificar na Figura 2, onde apresentamos o Menu Principal, é possível notar um menu de fácil uso e somente as funcionalidades essenciais.

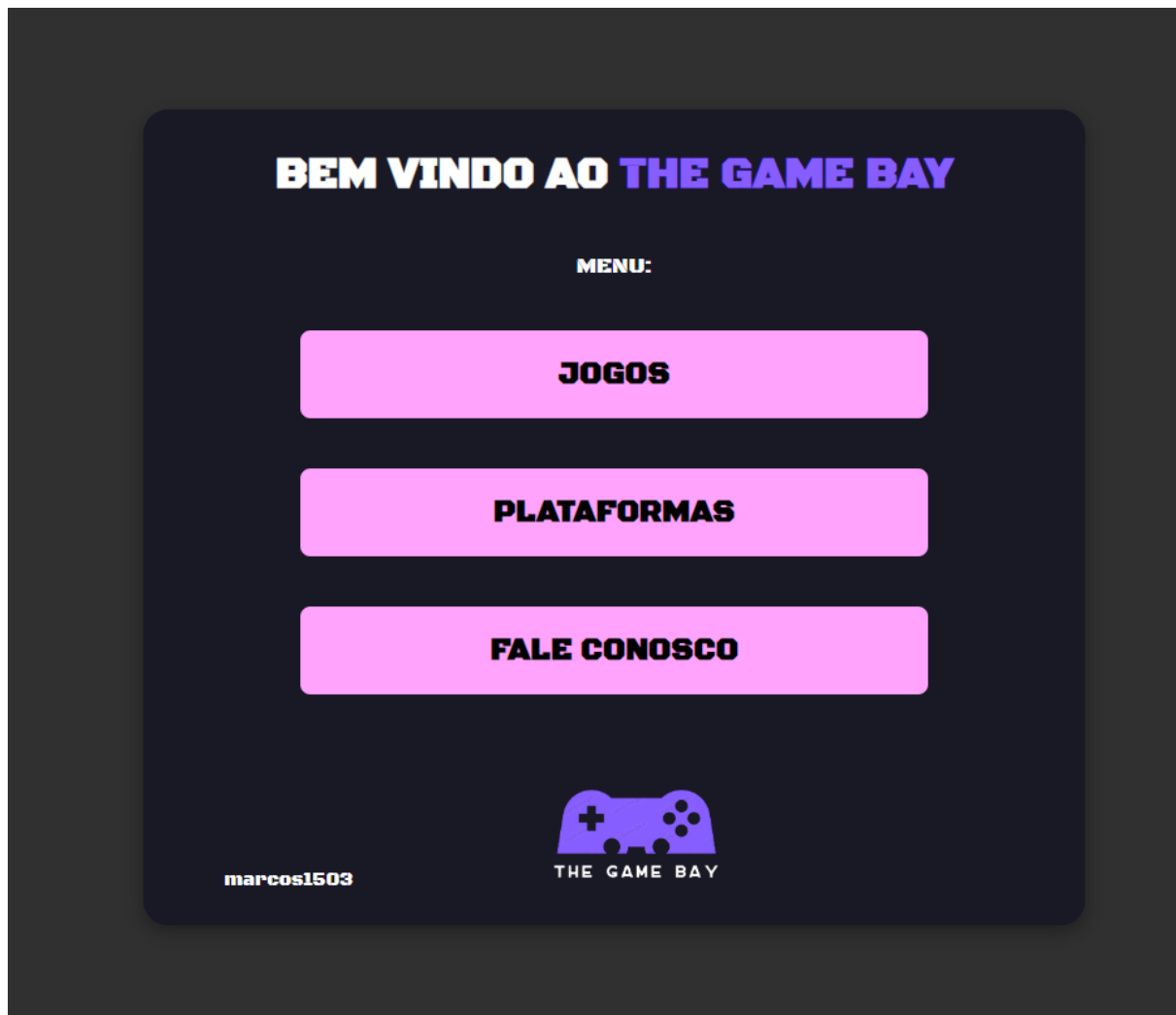


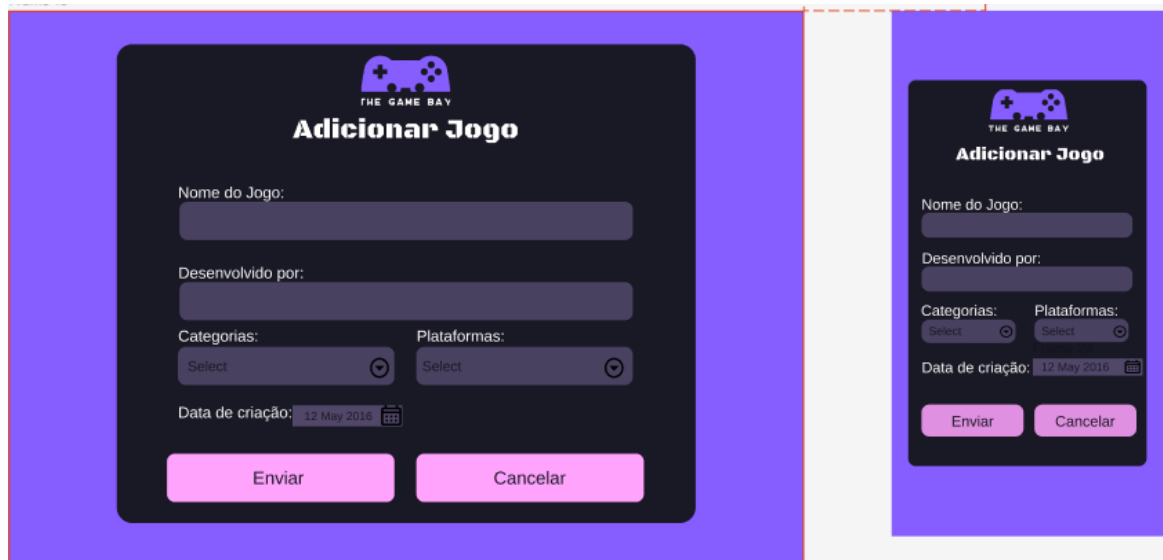
Figura 2: Menu Principal

Na Figura 2, também é possível analisar que alteramos a cor do background, tentando enfatizar o nome do site e a logo, pois são roxos, chamando a atenção do usuário.

A 3ª heurística "Controle do Usuário e Liberdade" foi aplicada permitindo que os usuários desfaçam ou refaçam ações facilmente. A funcionalidade de "Cancelar" foi incorporada nos formulários de cadastro e edição para oferecer aos usuários maior controle sobre suas interações, também foram criadas alertas e confirmações antes de executar uma função, como cadastrar ou deletar um jogo.

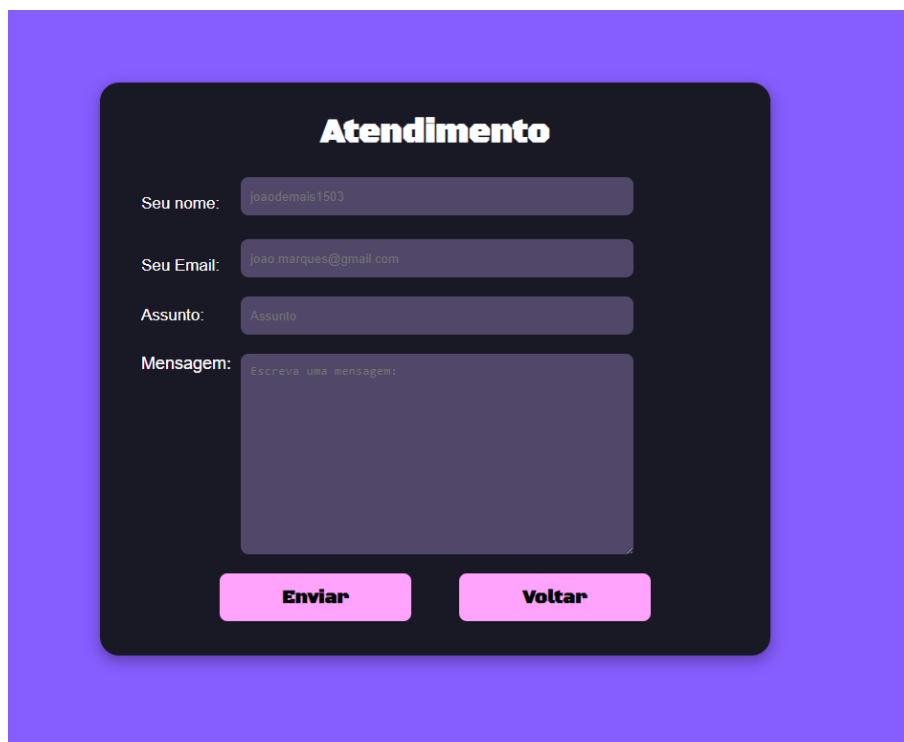
A tela de adição de jogo, apresentada na Figura 3, é projetada para facilitar a inclusão de novos títulos, sendo disponibilizada apenas a usuários com cargos de gerenciamento ou administração, com base na quinta heurística de Nielsen, para evitar erros e conflitos, visto que jogos adicionados, removidos, ou modificados, devem ser visíveis para todos os usuários. Ao acessar essa tela, o administrador encontra campos para inserir informações como o nome do jogo, categorias, plataformas, por qual empresa foi desenvolvido, e data de criação.

Caso um usuário comum deseje a adição de um jogo ou uma plataforma que sentiu falta ao navegar pelo sistema, ele deve enviar um ticket para a equipe através do “fale conosco”, apresentado na Figura 4, visando ainda mais a preocupação com a experiência do usuário do site, para ele não ter jogos ou plataformas repetidos, nós adicionamos a partir da solicitação dele.



The image displays two versions of a web form titled "Adicionar Jogo" (Add Game) for "THE GAME BAY". The desktop version on the left features a dark blue header with a game controller icon, followed by input fields for "Nome do Jogo:", "Desenvolvido por:", "Categorias:" (with a "Select" dropdown), "Plataformas:" (with a "Select" dropdown), and "Data de criação:" (with a date picker set to "12 May 2016"). At the bottom are "Enviar" and "Cancelar" buttons. The mobile version on the right is a condensed version of the same form, with the "Categorias:" and "Plataformas:" fields stacked horizontally and the date picker set to "12 May 2016". It also includes "Enviar" and "Cancelar" buttons.

Figura 3: Tela de adicionar jogos



The image shows a "Atendimento" (Support) form on a dark blue background. The title "Atendimento" is in white. Below it are input fields for "Seu nome:" (containing "joaodemais1503"), "Seu Email:" (containing "joao.marques@gmail.com"), and "Assunto:" (containing "Assunto"). The "Mensagem:" field is a large text area with the placeholder text "Escreva uma mensagem:". At the bottom are "Enviar" and "Voltar" buttons.

Figura 4: Tela de atendimento

Descrição do BackEnd

Para criarmos o nosso servidor, utilizamos *Node.js* com a biblioteca *Express* para criar a nossa *API Rest*. Implementamos alguns conceitos da arquitetura de MVC (Model, View e Controller), utilizando ao máximo boas práticas de clean code.

Levando em conta os requisitos do projeto, optamos por utilizar o banco de dados em *SQLite*. É possível visualizar as configurações do nosso banco no arquivo ***src/backend/src/configDb.js***, onde retornamos uma função assíncrona responsável por criar e "abrir" a rota para o banco de dados. As queries responsáveis pelo *CRUD* de dados estão na pasta *Models* do nosso backend. Visando a organização e manutenção do nosso projeto, criamos um arquivo separado para cada tabela do nosso banco, onde é possível identificar as queries de Criação de tabelas, inserção dos 10 dados padrões ao inicializar o banco, seleção de dados, inserção de dados, atualização das informações e exclusão. Também, foram criadas tabelas relacionais para lidar com as relações de 1 * X (1 para muitos), como a ***notaJogo.js***.

Arquitetura

A arquitetura e organização das pastas está dividido da seguinte formas:

1. Router

A pasta Router é responsável por cuidar das rotas da nossa aplicação através do arquivo *routes.js*, lidando com as respostas (res) e as requisições (req) dos métodos HTTP: ***POST***, ***PUT***, ***DELETE*** e ***GET***

2. Model:

Responsável pela criação das tabelas do nosso banco de dados e por armazenar as queries referentes aquela tabela, exportando as funções que irão ser chamadas no routes

FrontEnd

Consumo da API no FrontEnd

Conforme solicitado pelos professores, utilizamos o *framework React* para auxiliar no processo de criação do nosso front-end, facilitando a manipulação de objetos devido à utilização dos *hooks* ***useState*** e ***useEffect***. Para fazer chamadas na API, utilizamos a função *fetch* do *JavaScript*, passando a rota desejada e configurando-a para funcionar da melhor maneira. Um exemplo está na Figura 5; caso o usuário escolha atualizar o jogo, será executado um bloco de código para se conectar com a API.

Na mesma figura, observa-se o uso do método ***HTTP PUT***, passando as informações atualizadas dentro do corpo da requisição. É importante lembrar que estamos utilizando *JSON* para a comunicação entre cliente e servidor

```
37     const handleEnviar = () => {
38         const confirmation = window.confirm('Tem certeza que deseja atualizar o jogo ${jogos.name_game}?');
39
40         if(confirmation) {
41             fetch('http://localhost:3000/updateJogo', {
42                 method: 'PUT',
43                 headers: {
44                     "Content-Type": "application/json"
45                 },
46                 body: JSON.stringify({
47                     name_game:nameGame,
48                     developed_by:developedBy,
49                     category_name:category,
50                     data_criacao:date,
51                     idJogo:idJogos
52                 })
53             })
54                 .then((resp) => {
55                     if(!resp.ok) {
56                         throw new Error('Erro ao atualizar o jogo');
57                     }
58                     return resp.json();
59                 })
60                 .then(() => {
61                     window.alert(`O jogo ${jogos.name_game} foi atualizado com sucesso!`);
62                     navigate(`/jogos/${idJogos}`)
63                 })
64                 .catch((err) => {
65                     console.log(`Erro ao atualizar o jogo ${err}`)
66                     window.alert('Erro ao atualizar o jogo')
67                 })
68         }
69     }
70 }
```

Figura 5: Exemplo de conexão com a API

Components

No React, é recomendado utilizar componentes para melhorar a resposta, dinamismo e manutenção dos nossos códigos. Por isso, criamos partes de códigos que foram reaproveitadas ao longo do projeto na forma de componentes, como botões, listas, *navbar*, entre outros. Também organizamos os componentes por categorias, como é possível identificar na Figura 6, para facilitar a importação e utilização deles. Alguns desses componentes recebem propriedades de acordo com a necessidade

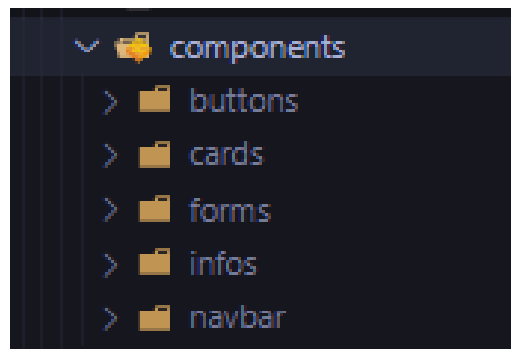


Figura 6: Pasta components

Rotas

Para aplicar as rotas em nossa aplicação, utilizamos a biblioteca ***react-router***. Visando a organização do nosso projeto, as páginas estão armazenadas dentro de *src/pages*, onde cada arquivo é responsável por uma resposta diferente da nossa aplicação. As rotas são implementadas no *main.jsx*, onde existe uma variável chamada *router* que utiliza um *hook* ***createBrowserRouter***, responsável por configurá-las.

Conclusão

Ao analisar, em resumo, o site está caminhando para oferecer uma experiência fácil e flexível para a avaliação de jogos, respeitando a experiência do usuário (UX), e as Heurísticas de Nielsen, permitindo assim, um gerenciamento de forma fácil e agradável dos seus jogos favoritos.

A aplicação das heurísticas de Nielsen nesse projeto resultou em um site de biblioteca de jogos que busca oferecer uma experiência de usuário eficiente e visualmente agradável. As decisões de design foram tomadas pela usabilidade, com atenção à visibilidade, consistência, controle do usuário e prevenção de erros em seus dados pessoais. O design minimalista complementa a funcionalidade, criando um ambiente simples para os usuários gerenciarem suas bibliotecas de jogos de forma descomplicada.

Sobre a implementação do servidor e cliente, podemos concluir que foram utilizadas boas práticas de desenvolvimento, existindo um padrão de organização e seguindo um modelo de arquitetura, focamos também em preservar a fácil manutenção e compreensão do código, além da possibilidade de escalabilidade da nossa aplicação.