**GbClickerV2**

**Resumo:** O GbClickerV2 é um sistema que teve como base um antigo projeto meu (GbClicker) utilizando as mesmas ferramentas (HTML, CSS, JAVASCRIPT, PHP e MYSQL), porém, de uma maneira muito mais simples e desorganizada. É um projeto pessoal de um jogo do tipo *Clicker/Idle*, que consiste em: o usuário clicar para conquistar dinheiro **fictício**, que pode serutilizado para o aprimoramento das ferramentas utilizadas para melhorar o seu ganho dentro do jogo. Em um primeiro momento, o jogo não necessariamente terá alguma interação entre jogadores, porém pode ser alterado de acordo com o caminhar do projeto. Hoje dia 19/08/2023, resolvi utilizar o padrão de projeto MVC (Model, View and Control) pois, com o andamento do desenvolvimento do projeto, notei que os códigos estavam ficando difíceis de serem acessados e alterados. Também não é mais utilizado o aplicativo XAMPP para manusear o banco de dados ou criar servido, estou utilizando o servidor embutido no próprio PHP. Hoje dia 16/12/2023, trago notícias sobre o projeto, para melhorar qualidade do código e facilitar sua manutenção futura, comecei a seguir a abordagem de escrita de código “PSR-12” e utilizar como base o livro Código Limpo do Robert C. Martin. Também houve a implantação do *composer* para facilitar a utilização de código repetido entre arquivos PHP.

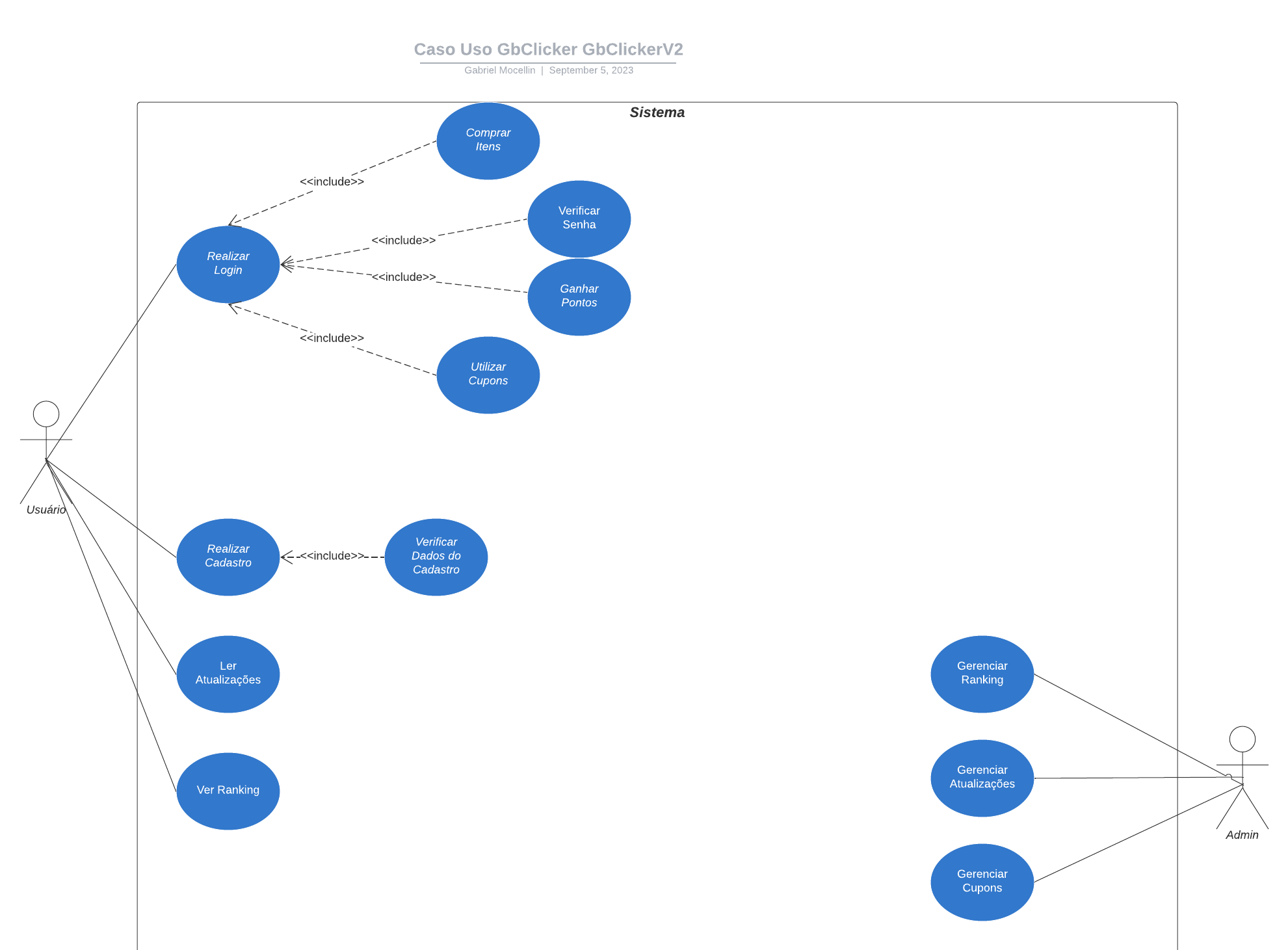
**Requisitos Funcionais**

|  |
| --- |
| **REQUISITOS** |
| **1RF:** O site deverá possuir uma tela inicial (*landing page*) com uma pequena introdução sobre o jogo. |
| **2RF:** A tela inicial (*landing page*) deve apresentar ao usuário as atualizações feitas com datas da manutenção e também fotos ou descrições do conteúdo atualizado. |
| **3RF:** A página inicial deverá conter um botão em que o usuário possa ser redirecionado para a página de login. |
| **4RF:** A página de login deverá conter um botão o usuário possa ser redirecionado para a página de registro. |
| **5RF:** A página de login deverá conter uma caixa de seleção para caso o usuário queira armazenar seus dados utilizando *cookies* para facilitar o login futuro. |
| **6RF:** Caso haja uma tentativa de registro com campos inválidos, deve ser informado para o usuário por meio de uma notificação os campos invalidados. |
| **5RF:** Os campos presentes na página de registro deverão possuir validações no *back-end* utilizando Javascript antes de serem enviados ao banco de dados. |
| **$RF:** Realizar a verificação de todos os campos enviados pelo usuário no lado do servidor para evitar SQL injection. |
| **9RF:** Ao usuário realizar o login deve ser iniciado o sistema de coleta de dinheiro por *minions*, que são comprados na loja. |
| **10RF:** Deve ser verificado na página de login se o usuário já está logado no sistema por meio da sessão do PHP. |
| **11RF:** Caso o usuário esteja com uma sessão iniciada anteriormente, ou seja, ter realizado o login previamente e não a encerrou, deve ser redirecionado para a página do jogo. |
|  |
| **8RF:** Deve ser disponibilizado ao usuário na página do jogo um menu, onde possa ser feita a navegação entre as páginas de máquinas, configurações, loja, página inicial, cupons, perfil e inventário. |
| **7RF:** Ao usuário realizar login com campos válidos, deve ser redirecionado à página inicial (onde o jogo acontece de fato). |
|  |
|  |
|  |
|  |
|  |

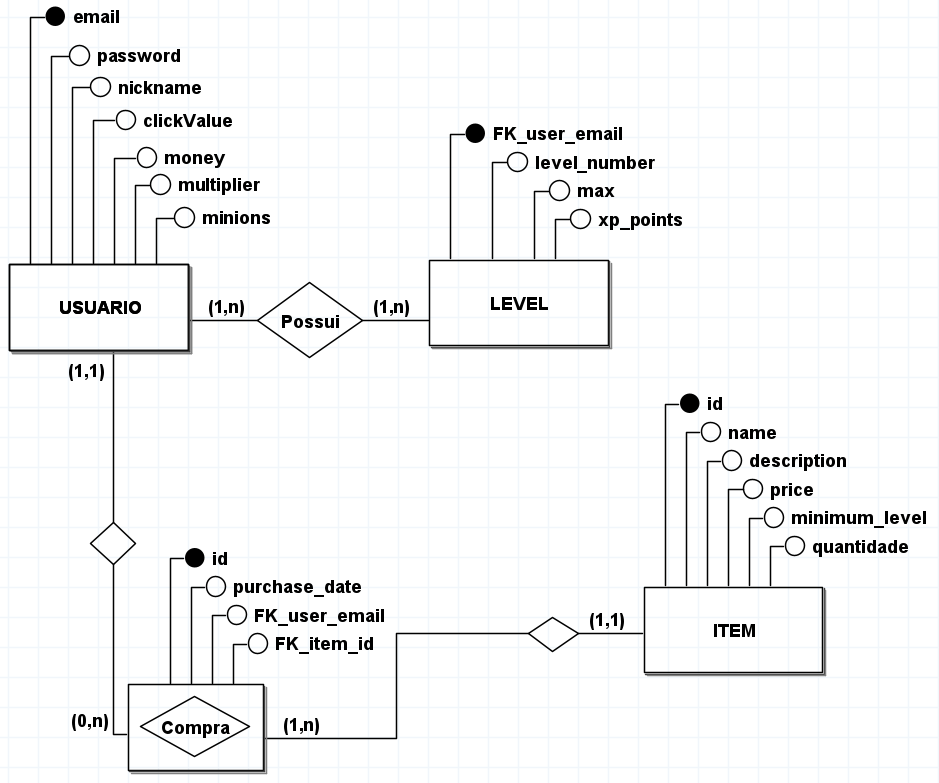
**Requisitos Não Funcionais**

|  |
| --- |
| **REQUISITOS** |
| **1RNF:** O software deverá ser desenvolvido utilizando as linguagens de programação PHP e JAVASCRIPT, para o banco de dados utilizará MYSQL, para estilização e estrutura utilizará HTML e CSS. |
| **2RNF:** A cada no máximo 5 dias, o desenvolvedor deverá verificar se o documento está com os requisitos atualizados referentes ao projeto. |
| **3RNF:** O projeto deverá utilizar o padrão de projeto MVC (*Model, View and Control*). |
| **4RNF:** O usuário deve ter seus dados carregados referentes ao jogo como itens e recursos adquiridos ao realizar o login. |
| **5RNF:** O jogo deve salvar os dados atuais do usuário a cada 1.5 segundos de maneira automática utilizando XMLHttpRequest. |

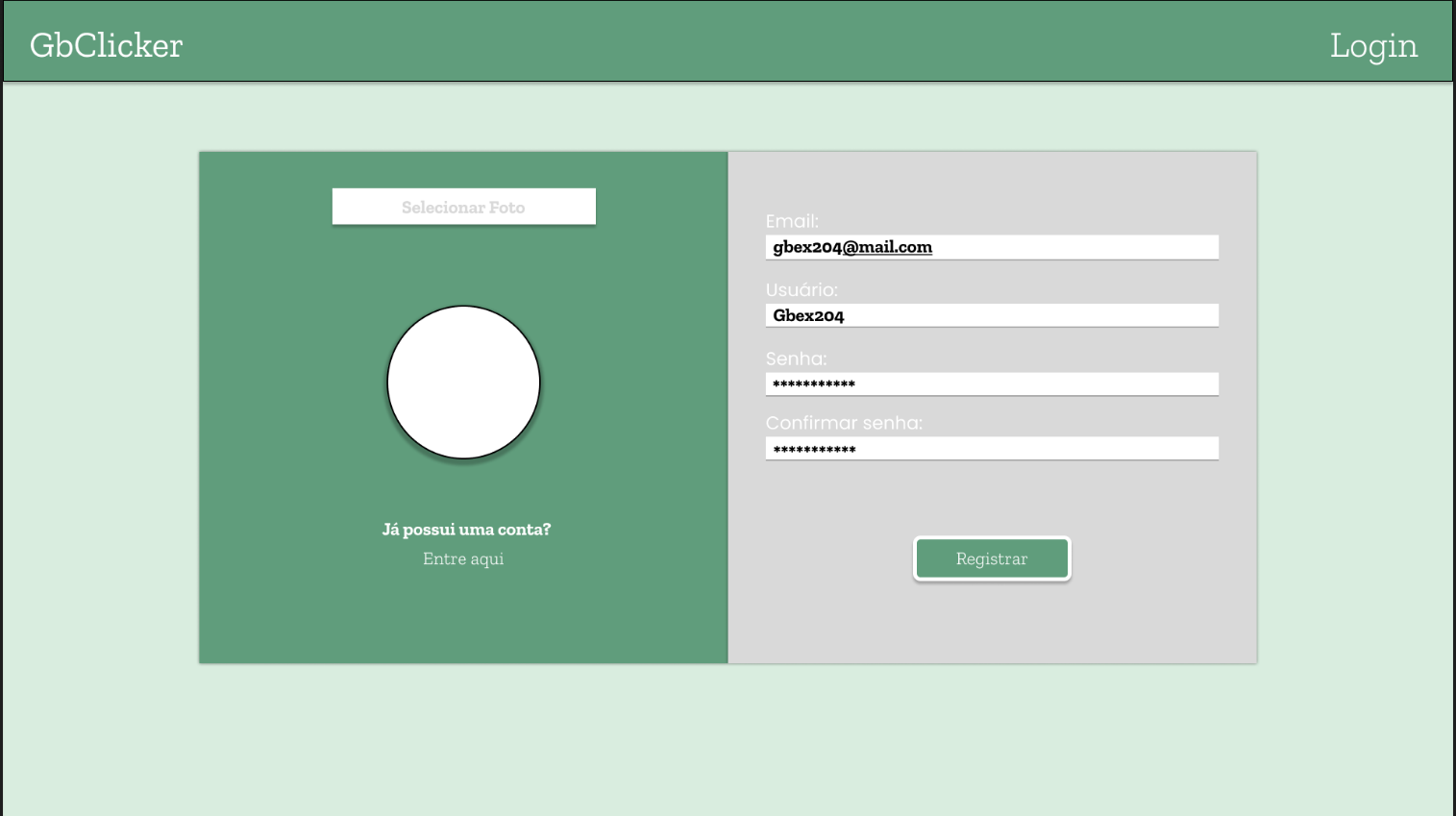
**Caso de uso**

****

**Modelo Relacional**

****

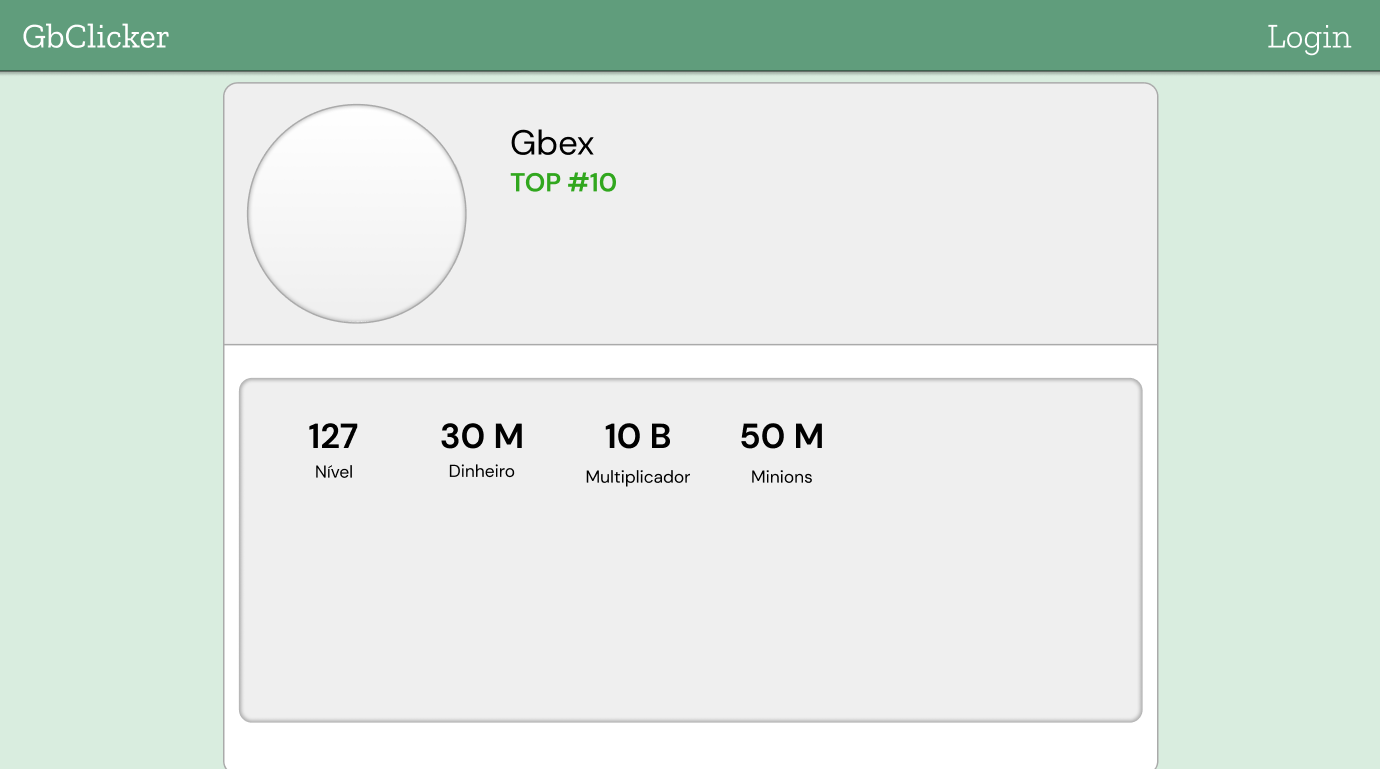
**Prototipação**

****

Página de registro do usuário



Página de login do usuário

****

Página de perfil de um usuário

**Problemas a serem resolvidos**

* Encontrar uma solução para quando a notificação esteja ativa, não seja necessária a divisão com width:100% para centralizá-la (presente na página de login e registro). Esta utilização impede o usuário de acessar o botão de login por 8 segundos (estipulados no Javascript).

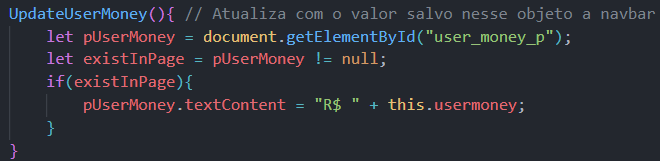
**Solucionado! (z-index negativo para a divisão de centralização)**

* Além disso, o código ainda está um pouco confuso, principalmente caso seja necessário o aumento de atributos que devem ser salvos ou resgatados do banco de dados relacionado ao usuário (talvez uma atenção a mais seja necessária).

**Melhorado! (troca do nome de funções e variáveis, diminuição do escopo de funções, e baseamento do código na abordagem PSR-12 e no livro Código Limpo.**

* Ao entrar em outras páginas que utilizam um objeto da classe game, ocorre tentativas de “atualizar” os dados presentes na página atual como ocorre na página inicial do jogo, onde são mostradas as mudanças no dinheiro atual, multiplicador e nível. Para corrigir esse erro deve ser verificado se ao buscar pelos *id’s* os textos utilizados para mostrar os dados são diferentes de nulo. Exemplo:

**Solucionado! (Função genérica criada para suprir as verificações e funções com escopo repetido)**



* O usuário consegue manipular os dados do jogo por conta de os dados serem atualizados de maneira dinâmica por meio do Javascript, que até o momento (16/12/2023) não possui um meio de prever o uso do console de desenvolvedor pelo usuário.