PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

SISTEMA WEB DE APRENDIZAGEM COM USO DE GAMIFICAÇÃO

GUILHERME HENRIQUE MENDONÇA NASCENTE

GOIÂNIA
2023

GUILHERME HENRIQUE MENDONÇA NASCENTE

SISTEMA WEB DE APRENDIZAGEM COM USO DE GAMIFICAÇÃO

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Prof. Me. Eugênio Júlio M. Cândido Carvalho

Banca examinadora:

Profa. Dra. Carmen Cecilia Centeno

Prof. Dr. Leonardo Guerra de Rezende Guedes

Prof. Me. Olegário Correa da Silva Neto

GOIÂNIA
2023

GUILHERME HENRIQUE MENDONÇA NASCENTE

SISTEMA WEB DE APRENDIZAGEM COM USO DE GAMIFICAÇÃO

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Ciência de Computação, em _____/_____/_____.

Profa. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de Conclusão de Curso

Orientador(a): Me. Eugênio Júlio M. Candido Carvalho

Profa. Dra. Carmen Cecilia Centeno

Prof. Dr. Leonardo Guerra de Rezende Guedes

Prof. Me. Olegário Correa da Silva Neto

GOIÂNIA
2023

AGRADECIMENTOS

Agradeço a Deus pela força e coragem que me deu durante a realização do TCC.

Agradeço aos meus pais, a minha irmã e ao meu cachorro Chico que me deram todo amor e apoio e sempre me motivaram a continuar estudando e nunca desistir, mesmo que as coisas parecessem impossíveis.

Agradeço aos meus amigos que estiveram comigo desde o início da faculdade. São eles: Watyson Soares, Márcio Antusa e Fernando Gomes. Eles sempre me apoiaram e ajudaram de alguma forma, e também aprendi muito com eles.

Agradeço ao orientador professor Eugênio Júlio por ter me orientado e ajudado a conseguir realizar o TCC e obrigado por ter me ensinado tantas coisas.

Agradeço à professora Carmen Cecília pelo apoio e por permitir de ter sido seu monitor por bastante tempo. Obrigado por acreditar em mim e por contribuir com este trabalho pelas sugestões e feedbacks.

Agradeço ao professor Alexandre Ribeiro que sempre me incentivou para dar o meu melhor. Graças a ele, nunca desisti de estudar aquilo que eu tanto gosto, a programação.

Agradeço aos meus colegas de trabalho por sempre terem me dado espaço para realização deste trabalho.

Agradeço ainda aos professores Leonardo Guerra e Olegário Correa, por terem contribuído com o TCC, seus feedbacks foram fundamentais para aprimoramento deste projeto.

Obrigado a todos. Vocês foram responsáveis por me tornar a pessoa que sou hoje.

RESUMO

Apresenta-se um trabalho sobre desenvolvimento de sistema web de aprendizagem com uso de gamificação, que busca ser uma alternativa para professores utilizarem como ferramenta para motivar os alunos a aprenderem o conteúdo. Este trabalho tem como objetivo documentar os requisitos, regras de negócio e casos de uso, e implementar um sistema web que faz o uso dos elementos da gamificação por meio de um jogo de memória, porém em forma de protótipo. Foram utilizadas as tecnologias Flutter, Spring e PostgreSQL e as arquiteturas cliente-servidor, camadas e MVVM para a construção deste sistema. A utilização destas tecnologias e arquiteturas possibilitou o desenvolvimento do protótipo do sistema Web que permite ao professor criar seus próprios jogos de memória e gerar a partida para que os alunos possam jogar o jogo de memória, e durante a partida, quando selecionado um par de cartas, aparecerá as opções “Está certo!” e “Está errado!” para que o aluno possa decidir entre as duas opções.

Palavras-chave: Gamificação. Jogo de memória. Sistema web de aprendizagem. Criador. Jogador. Partida. Cartas. Pontuação. Histórico. Flutter. Spring. PostgreSQL. Cliente-servidor. Camadas. MVVM.

ABSTRACT

This paper presents a work on the development of a learning web system with the use of gamification, which seeks to be an alternative for teachers to use as a tool to motivate students to learn the content. This work aims to document the requirements, business rules, and use cases, and to implement a web system that makes use of gamification elements through a memory game, but in prototype form. The technologies Flutter, Spring and PostgreSQL and the client-server, layers and MVVM architectures were used to build this system. The use of these technologies and architectures enabled the development of the prototype Web system that allows the teacher to create his own memory games and generate the game so that the students can play the memory game, and during the game, when a pair of cards is selected, the options "It's right!" and "It's wrong!" will appear so that the student can decide between the two options.

Keywords: Gamification. Memory game. Learning web system. Creator. Player. Game. Cards. Score. History. Flutter. Spring. PostgreSQL. Client-server. Layers. MVVM.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

DTO - *Data Transfer Object*

HTTP - *Hypertext Transfer Protocol*

IDE - *Integrated Development Environment*

JPA - *Java Persistence API*

JPQL - *Java Persistence Query Language*

JSON - *JavaScript Object Notation*

JVM - *Java Virtual Machine*

JWT - *JSON Web Tokens*

MER - Modelo entidade-relacionamento

MVVM - *Model-view-ViewModel*

ORM - *Object Relational Mapper*

SQL - *Structured Query Language*

XML - *Extensible Markup Language*

YAML - *YAML Ain't Markup Language*

SUMÁRIO

1. INTRODUÇÃO

A educação passou por muitas transformações ao longo de anos. Uma das transformações evidentes é o uso da tecnologia de informação, que ajudou os professores a melhorarem a qualidade do ensino. Mas, mesmo com uso da tecnologia, a desmotivação e a falta de engajamento dos alunos ainda permanece na sala de aula. (CAROLINA TOMÉ KLOCK et al., 2014)

De acordo com Tolomei, as instituições de ensino, na maioria delas, independente do nível de educação, têm dificuldades para engajar os alunos utilizando os recursos educacionais tradicionais. Por conta desta dificuldade, é necessário encontrar novas maneiras para engajar e motivar os alunos da nova geração (2023) nas atividades educacionais. Umas das maneiras é o uso da gamificação. (TOLOMEI, 2017)

A gamificação na educação é uma técnica que envolve o uso de mecanismos, estética e pensamento dos jogos para incentivar os alunos a promover conhecimento e resolver problemas. É o uso de elementos de jogos em contextos não relacionados com jogos. Para que o aluno possa se sentir motivado a aprender, pode aumentar progressivamente a dificuldade dos exercícios, permitir cometer erros e promover competições e colaboração entre os alunos. (CAROLINA TOMÉ KLOCK et al., 2014)

O jogo da memória é um jogo popular de cartas jogado por crianças e adultos em todo o mundo. Ter uma boa memória é uma das qualidades necessárias para conseguir ganhar o jogo. Qualquer número de jogadores pode jogar. Cada jogador deve escolher duas cartas, se ambas forem iguais, permanece virada, mas se não, devem virar de novo. Cada jogador deve lembrar a posição e a identidade das cartas já vistas. O jogo somente termina quando todas cartas estão viradas. (ZWICK et al., 1993)

Diante da dificuldade de engajamento dos alunos, o uso de gamificação pode ser uma ferramenta que visa prender a atenção do aluno. Como o jogo de memória é popular no mundo todo, este trabalho tem objetivo de construir um aplicativo que possa ser utilizado em sala de aula, introduzindo a gamificação no mesmo. O aplicativo terá a versão Web que permitirá que usuários possam criar seus próprios jogos de memória e compartilhar estes jogos para outros usuários jogarem e ganharem pontos.

1.1 Objetivos

1.1.1 Objetivos Gerais

Este trabalho tem como objetivo de implementar o jogo de memória para Web. Será usado a tecnologia Flutter, Spring e PostgreSQL para construir a aplicação.

1.1.2 Objetivos Específicos

- Construir um sistema Web em que o professor possa criar seu próprio jogo de memória e depois disponibilizar para alunos jogarem.
- Documentar requisitos, diagrama de casos de uso, diagrama de classe e diagrama modelo entidade-relacionamento (MER).
- Utilizar a linguagem Dart e o framework Flutter para construir front-end.
- Utilizar a linguagem Java e o framework Spring para construir back-end.
- Utilizar sistema de banco de dados PostgreSQL para construir tabelas e relações e usar para armazenamento de dados.

1.2. Metodologia

A metodologia usada para desenvolvimento deste projeto foram:

- Pesquisa bibliográfica sobre gamificação.
- Pesquisa bibliográfica das ferramentas, linguagens e frameworks utilizados.
- Desenvolvimento de um sistema Web.
- Descrição do sistema (regras, requisitos, diagrama de classe e MER, arquitetura e protótipo).

1.3. Visão Geral

Este trabalho está dividido em 9 capítulos:

1) Introdução: Introdução, objetivos, metodologia e visão geral deste projeto.

2) Gamificação e Projeto: Gamificação e as ideias do projeto.

3) Ferramentas: Ferramentas, linguagens e frameworks utilizados para construção do sistema.

4) Regras de negócios e Requisitos: Regras de negócio, requisitos funcionais, requisitos não funcionais.

5) Caso de uso: Diagrama de caso de uso e os casos de uso descritivos.

6) Diagrama de Classe e MER: Diagrama de classe e MER.

7) Arquitetura: Arquiteturas usadas no desenvolvimento.

8) Protótipo: Protótipo do aplicativo Web.

9) Conclusão: Conclusão do TCC.

2. GAMIFICAÇÃO E PROJETO

2.1. Gamificação

A tecnologia apoia determinadas estratégias na educação, porém os fatores como motivação, cooperação e engajamento dos estudantes ainda são considerados difíceis de ultrapassar. A gamificação é uma das formas de motivar o aluno, pois ela pode aumentar a participação dos alunos extraindo os elementos agradáveis e divertidos dos jogos de forma adaptada ao ensino. De acordo com Tolomei e Carolina Tomé Klock, alguns elementos de jogos conhecido são: (TOLOMEI, 2017)

Tabela 1 - Elementos de jogos.

Pontuação	O usuário ganha uma determinada pontuação quando completa uma tarefa ou objetivo.
Níveis	Mostra o progresso do usuário dentro do sistema, geralmente sendo utilizado em conjunto com os pontos.
Ranking	Mostra progresso de todo usuário, ordenado de forma decrescente pela pontuação do usuário, pode criar um senso de competição.
Medalhas/ Conquistas	São uma espécie de prêmios que o usuário recebe por conta das suas tarefas realizadas.
Desafio	O usuário pode realizar tarefas desafiadoras, sendo recompensado de alguma maneira por isso, que podem ser pontos ou medalhas. Cria o sentimento de desafio para o usuário do sistema.
Regras	O sistema pode ser composto por diversas regras que definem como usuário deve interagir com o jogo, servem para limitar as ações do usuário.

Fonte: Autoria própria. Adaptada do Tolomei e Carolina Tomé Klock.

Os elementos de jogos aplicados na gamificação estão diretamente relacionados aos desejos humanos, como por exemplo: pontos são conectados com a necessidade de recompensa; níveis são úteis para demonstrar status; desafios permitem concluir realizações; rankings estimulam a competição. É importante notar que aplicar gamificação em uma tarefa, não é necessário a utilização de todos os elementos do jogo, ou seja, pode ser utilizado um número reduzido, até uma quantia maior de elementos disponíveis. (CAROLINA TOMÉ KLOCK et al, 2015; SILVA et al, 2019)

De acordo com Silva et al, os jogos têm 4 elementos fundamentais que são utilizados para gamificação, são eles:

- Voluntariedade: A voluntariedade implica na aceitação das regras, objetivos e feedbacks.
- Objetivo: Os objetivos vão orientar o jogador a se concentrar para atingir o propósito e os objetivos devem ser bastante claros, senão o jogador pode se sentir perdido, levando-o ao fracasso.
- Regra: As regras vão definir como o jogador deverá se comportar ou organizar suas ações para a realização das tarefas impostas pelo jogo.
- Feedback: Os feedbacks vão informar ao jogador os resultados das ações realizadas por ele dentro do jogo e devem ser imediatos e claros para o jogador se atualizar sobre seu estado no jogo.

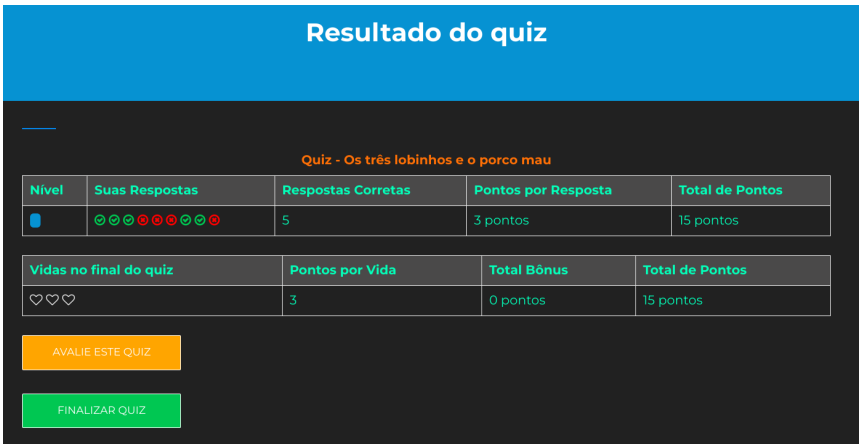
Outra característica importante sobre a gamificação é que pode ser usado as tecnologias digitais para implementar uma estratégia baseada na gamificação da sala de aula. Um dos exemplos da aplicação de gamificação com uso de tecnologia digital é o Game Arkos. (SILVA et al, 2019)

Game Arkos foi criado em 2011 pelos Prof. Odair Artmann, Prof. Bruno Kosinski e Sven Kottmann. É um portal com objetivo de promover a leitura através da gamificação para alunos do ensino fundamental até o ensino médio. O aluno acessa o portal por meio do usuário e senha, procura o livro que leu e responde a perguntas sobre o livro. A quantidade de perguntas depende do livro e o aluno vai ter três vidas, a cada pergunta errada, ele perde uma vida. Se ele perder todas as vida, ele perde a possibilidade de ganhar o ponto bônus no final, mas as perguntas continuam. Caso acerte, ganha pontos. Além disso, pode ter medalhas e adesivos virtuais, além de subir de nível e competir com os colegas. As figuras 1 e 2 mostram exemplos do jogo sobre o livro “Os três lobinhos e o Porco Mau”. (GAME ARKOS, 2023; TOLOMEI, 2017)

Figura 1 - Exemplo de um jogo sobre o livro “Os três lobinhos e o Porco Mau”.

Fonte: Game Arkos.

Figura 2 - Resultado do Quiz.



Fonte: Game Arkos.

2.2. Projeto

Como dito anteriormente, este projeto tem objetivo de construir um sistema para aplicar gamificação na metodologia de qualquer disciplina, usando para isso o jogo de memória. O jogo de memória usado será diferente do tradicional. Ao invés de duas cartas iguais, pode ser usado duas cartas diferentes, mas uma é associada com a outra. Por exemplo, uma carta contendo o texto "Revolução Francesa" e outra carta contendo uma imagem da Revolução Francesa.

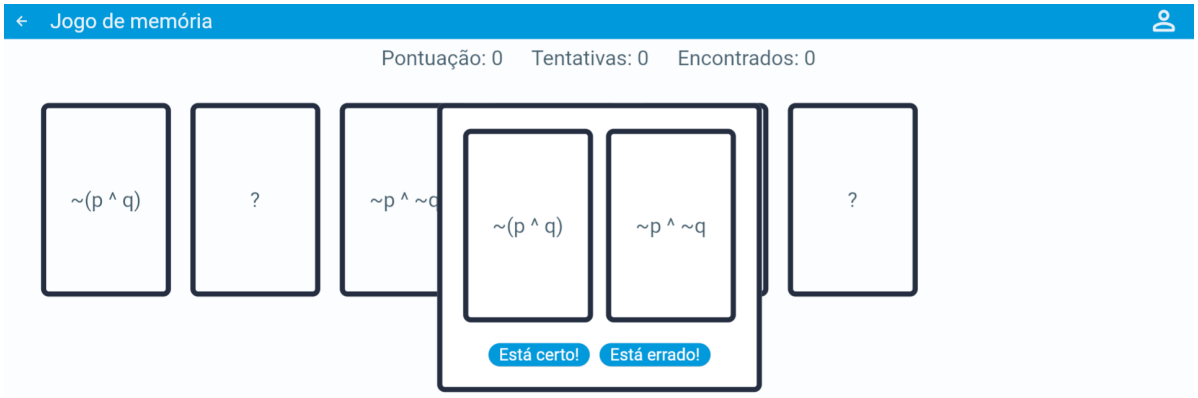
Outra diferença está no fato de que é o jogador que vai decidir se está certo ou errado nas duas cartas que ele escolheu como mostra a figura 3. Se as cartas estiverem corretas e o jogador escolher que está certo, ele ganha pontos e as cartas permanecem viradas. Caso

contrário, ele perde pontos e as cartas são viradas novamente. Porém se as cartas estiverem erradas e o jogador escolher que está certo, ele perde os pontos e as cartas são viradas novamente. Caso contrário, ele não ganha nem perde pontos e as cartas são viradas novamente. O jogador só termina o jogo até conseguir completar todas as cartas.

Ou seja, os objetivos do jogador são: selecionar se estão certas ou erradas um par de cartas selecionadas, o que serve como uma espécie de desafio para o jogador, já que ele precisa ter conhecimento sobre determinado conteúdo da carta para decidir se estão de fato certo ou errado; e terminar todas as cartas para completar o jogo.

E as regras são: só pode selecionar duas cartas por vez e não mais que isso; escolher somente uma das opções se está certo ou errado as cartas selecionadas; as cartas só serão viradas para cima quando as cartas forem certas e jogador selecionou a opção certa. O jogo termina quando as cartas estiverem viradas para cima.

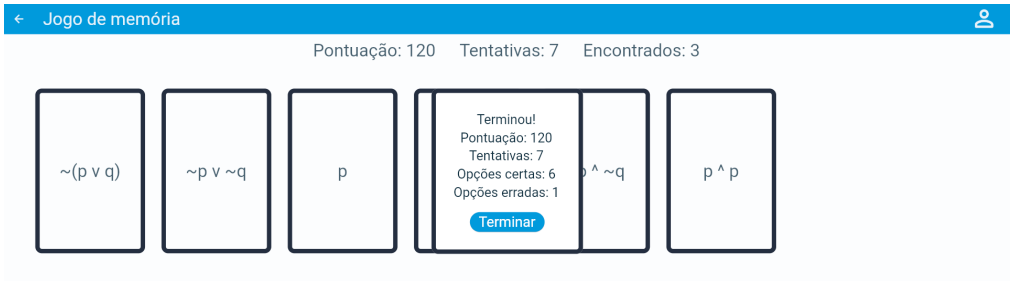
Figura 3 - O jogador escolhe se está certo ou errado nas cartas escolhidas.



Fonte: Autoria própria.

As pontuações vão sendo calculadas à medida que usuário for jogando o jogo de memória. Além da pontuação, também são computadas: a quantidade de tentativas, ou seja, a quantidade de vezes que usuário selecionou um par de cartas, e de pares encontrados, que se trata da quantidade de vezes que usuário encontrou e acertou as cartas. Isso serve como feedback pois eles darão a informação de como está o jogador durante o jogo. Outro feedback aparecerá quando o jogador completar o jogo, que vai mostrar as informações de pontuação, quantidade de tentativas, quantidade de opções certas e quantidade de opções erradas, como mostra a figura 4.

Figura 4 - Jogo completado pelo jogador.



Fonte: Autoria própria.

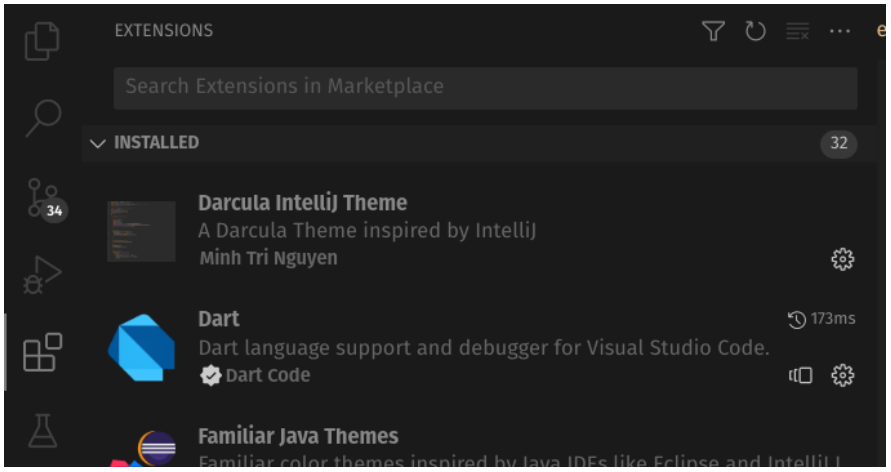
Além do uso dos elementos de jogos, o projeto também oferecerá a criação ou edição do jogo de memória; compartilhamento de jogo de memória com outros jogadores e jogadores podem jogar o jogo de memória salvo anteriormente; e visualizar o histórico das partidas que pode ser usado como feedback. Isto será explicado com mais detalhes nos capítulos 4 e 5.

3. FERRAMENTAS

3.1. VS Code

O Visual Studio Code é um editor de código desenvolvido pela Microsoft, disponível para Windows, macOS e Linux. Ele vem com suporte integrado para JavaScript, TypeScript e Node.js e através de extensões como mostra a figura 5, pode possuir suporte para outras linguagens como C++, C#, Java, Python, PHP, Go, .NET, Dart e Flutter. (VISUAL STUDIO CODE, 2023a)

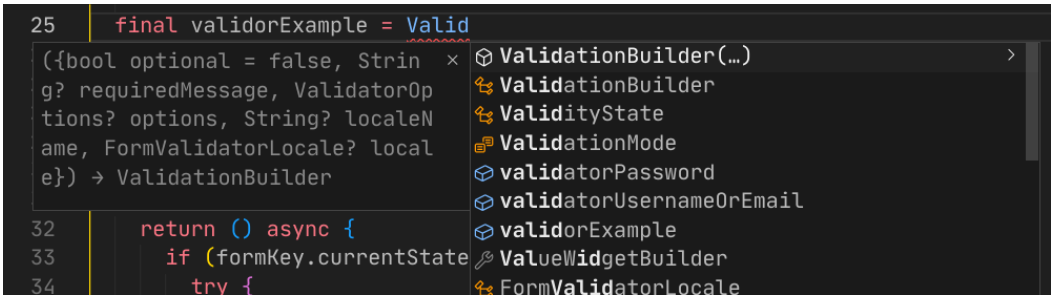
Figura 5 - Extensões do VS Code.



Fonte: Autoria própria.

Ele também possui um recurso chamado IntelliSense que pode apresentar uma lista de sugestões disponíveis à medida que usuário vai digitando e usuário pode escolher a sugestão para ser completada automaticamente sem precisar digitar a palavra toda. Além disso, também mostra as informações sobre determinado comando como por exemplo as informações sobre parâmetros. Por exemplo, se digitar a palavra “Valid”, como mostra a figura 6, já mostra uma lista de sugestões que podem ser completadas, e para cada sugestão pode haver informações sobre parâmetros desta sugestão. (VISUAL STUDIO CODE, 2023b)

Figura 6 - Exemplo de lista de sugestão.



Fonte: Autoria própria.

Também traz algumas ferramentas que ajudam a alterar o código com facilidade, como por exemplo, renomear o nome da variável ou função e códigos que usarem esta variável ou função, vão ser renomeados também.

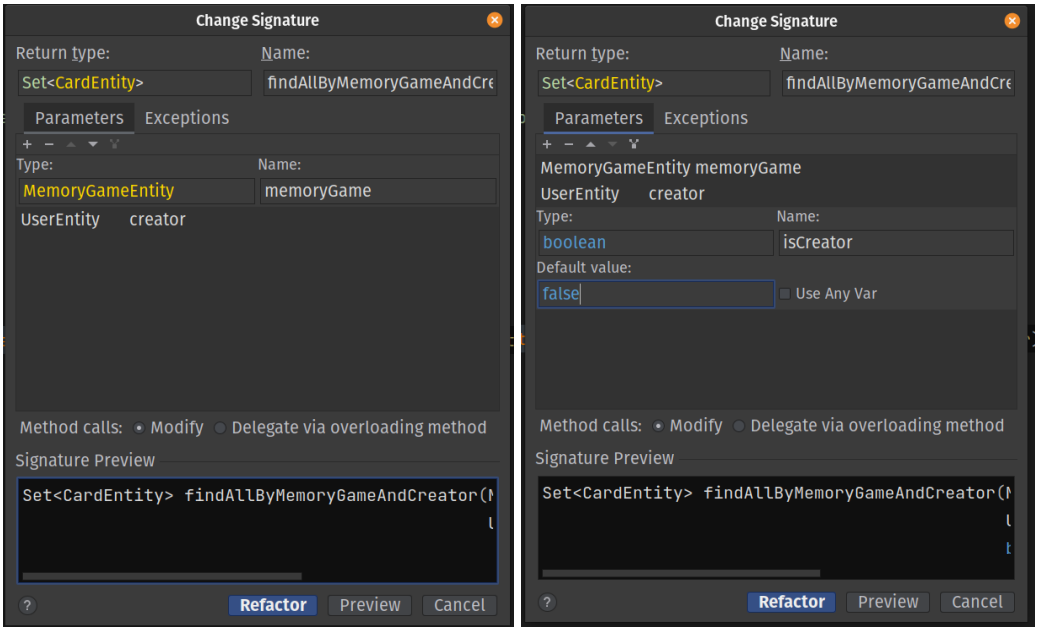
O Visual Studio Code foi escolhido para desenvolver aplicação Flutter com a linguagem Dart, pois ele oferece suporte para aplicação executar no navegador Chrome, oferece o modo debug para que seja feito o rastreio da execução da aplicação em pontos específicos do código e também facilita a mudança de nome.

3.2. IntelliJ IDEA

O IntelliJ IDEA é a *Integrated Development Environment* (IDE) feito pela JetBrains para o desenvolvimento em Java e Kotlin. Possui diversos recursos que facilitam o desenvolvimento como por exemplo, assistência inteligente à programação, refatorações, navegação instantânea pelo código, ferramentas de desenvolvimento incorporadas, suporte ao desenvolvimento corporativo e para a Web, entre outros. (INTELLIJ IDEA, 2023b)

Ele oferece várias opções de refatoração, como por exemplo mudar a assinatura do método. A figura 7 mostra que foi adicionado um parâmetro booleano chamado de *isCreator* na assinatura do método *findAllByMemoryGameAndCreator*. É possível também definir um valor padrão para um parâmetro adicionado para que seja adicionado automaticamente este valor em códigos que usam este método. (INTELLIJ IDEA, 2023a)

Figura 7 - Exemplo de modificação da assinatura do método.



Fonte: Autoria própria.

O IntelliJ IDEA foi escolhido para desenvolver aplicação *Spring* com a linguagem Java por conter ferramentas avançadas de refatoração e apresentar sugestões inteligentes, além de ter a opção de debugar a aplicação em execução em pontos específicos do código.

3.3. PostgreSQL

O PostgreSQL é um banco de dados que segue o padrão *Structured Query Language* (SQL). SQL é uma linguagem que permite a interação com dados armazenados no banco de dados relacional. PostgreSQL estende a linguagem SQL combinada com muitos recursos que armazenam e dimensionam com segurança as cargas de trabalho de dados mais complicadas. E traz muitos recursos destinados a auxiliar na proteção da integridade dos dados e a criar ambientes tolerantes a falhas. (MILANI, 2008; PostgreSQL, 2023)

De acordo com Milani, as suas características principais são:

- Permite realizar consultas complexas com uso de *SELECT*, *WHERE*, *GROUP BY* e entre outros. (MILANI, 2008)
- Possui vários tipos de dados, como Inteiro, Numérico, String, Booleano, Data/Hora, UUID e entre outros. (PostgreSQL, 2023)
- Restrições para manter integridade dos dados, como *UNIQUE*, *NOT NULL*, chave primária, chave estrangeira e entre outros. Por exemplo, definir como

> *UNIQUE* em uma determinado campo de uma tabela, garante que aquele campo nunca vai ter o valor repetido. (MILANI, 2008; PostgreSQL, 2023)

Uma das características principais citada anteriormente, é a consulta complexa que pode ser realizada para buscar registros que dependem do relacionamento com outras tabelas. A figura 8 mostra a consulta para obter informações de um determinado jogador pertencente a um jogo de memória ou mais de um determinado criador. Irá retornar dois resultados possíveis, primeiro resultado é quando encontra o registro de um jogador que pertence a pelo menos um jogo de memória de um criador, e outro resultado é quando não encontra este jogador, ou seja, ele não faz parte de nenhum jogo de memória de um criador.

Figura 8 - Comando SQL para buscar um jogador de um jogo de memória de um criador.

```sql
select pl.*
from creator cr
inner join memory_game mg on mg.creator_id = cr.id
inner join player_memory_game pmg on pmg.memory_game_id = mg.id
inner join player pl on pl.id  = pmg.player_id
inner join user_mg us on us.id = pl.user_id
inner join user_mg us2 on us2.id = cr.user_id
where us2.username = 'eugenio'
and us.username = 'fer_junnon';
```

Fonte: Autoria própria.

O PostgreSQL foi escolhido para criar as tabelas e relações entre elas, e definir as restrições como *UNIQUE* e *NOT NULL* para garantir a integridade de dados.

3.4. Java

Java é uma linguagem multiplataforma graças ao *Java Virtual Machine* (JVM), máquina virtual do Java, que consegue executar o código compilado em Bytecodes, que é uma linguagem de máquina da JVM, em diferentes plataformas. Ela é também uma linguagem multi paradigmas, tem suporte para programação orientada a objetos, programação genérica e programação funcional. (DEITEL et al, 2017)

Com a programação orientada a objetos, a linguagem permite criar classes, definindo os atributos (variáveis) e métodos (funções), e a instância desta classe será o objeto. Cada objeto de uma classe tem seu próprio conjunto de atributos de instância, que vão armazenar o

estado do objeto. E outra característica importante do Java, é que ele possui gerenciamento de memória segura com uso de *Garbage Collector*, ou Coletor de Lixo, que é responsável por limpar memória não usada sem a intervenção do usuário. (SEBESTA, 2011; DEITEL et al, 2017)

A figura 9 mostra um exemplo feito em Java com a classe chamada *Player* que vai representar o jogador. Ele contém os atributos *user* que vai representar o usuário cadastrado do sistema. O atributo *memoryGameSet* que vai possuir conjunto de jogo de memória que o jogador possui. E o atributo *gameplaySet* vai possuir o histórico dos jogos que ele jogou. E também mostra os métodos *addMemoryGame* para adicionar o jogo de memória na lista e *addGameplay* para adicionar a jogada na lista de histórico.

Figura 9 - Exemplo de uma classe Jogador feita em Java.

```
public class Player {

    no usages
    private User user;
    1 usage
    private Set<MemoryGame> memoryGameSet = new HashSet<>();
    1 usage
    private Set<Gameplay> gameplaySet = new HashSet<>();

    no usages
    public Player addMemoryGame(MemoryGame memoryGame) {
        memoryGameSet.add(memoryGame);
        return this;
    }

    no usages
    public Player addGameplay(Gameplay gameplay) {
        gameplaySet.add(gameplay);
        return this;
    }
}
```

Fonte: Autoria própria.

Java foi usado como linguagem de backend, por possuir várias bibliotecas por padrão como por exemplo, java.util que traz várias estruturas de dados já implementados como *HashSet* e *ArrayList*. E também por ser multiparadigmas que permite usar programação orientada a objetos por meio das classes e funcional através do *Stream* que contém *map, filter*, *reduce* e entre outros.

3.5. Lombok

O Projeto Lombok é uma biblioteca Java que permite diminuir código como por exemplo, os métodos *get* e *set* para cada atributo da classe Java. Através das anotações *@Getter* e *@Setter* em cima da classe, é gerado automaticamente todos os métodos *get* e *set*, deixando código mais simples. Ele também oferece outras anotações para serem gerados automaticamente como *@AllArgsConstructor*, *@NoArgsConstructor* em relação ao constructor e *@EqualAndHashCode* em relação a comparação entre dois objetos e do código Hash. (PROJECT LOMBOK, 2023)

Como mostra a figura 10, com uso do Lombok, os métodos *get* e *set* não estão mais presentes, pois as anotações *@Getter* e *@Setter* anotada em cima da classe de jogador, já gera automaticamente em tempo de compilação, desta forma a classe fica mais simples, sem necessidade de escrever estes métodos.

Figura 10 - Exemplo da classe de Jogador com anotações *@Getter* e *@Setter*.

```
@Getter
@Setter
public class Player {

    no usages
    private User user;
    no usages
    private List<MemoryGame> memoryGameList = new ArrayList<>();
    no usages
    private List<Gameplay> gameplayList = new ArrayList<>();
}
```

Fonte: Autoria própria.

Lombok foi escolhido para não ser necessário criar *getters* e *setters* manualmente para cada atributo, o que permite ter somente o código necessário.

3.6. MapStruct

As aplicações geralmente requerem mapeamento entre diferentes classes de objetos, como por exemplo, converter de classe de entidade para classe *Data Transfer Object* (DTO). MapStruct é um framework de mapeamento que gera a implementação de mapeamento entre classes diferentes. (MAPSTRUCT, 2022)

A criação de mapeamento é simples, apenas deve ser criado uma interface do Java com anotação *@Mapper* e criar métodos definindo tipo de parâmetros de entrada e retorno de resultado. Desta forma, MapStruct analisa os tipos e gera a implementação que converte do tipo do parâmetro de entrada para tipo de retorno quando o projeto é construído. Não é preciso configurar caso as classes diferentes tenham os mesmos nomes e tipos de atributos. (MAPSTRUCT, 2022)

Por exemplo, a figura 11 mostra o mapeamento em relação às cartas ("*card*" no projeto) e tem um método *toCardEntity* que vai receber o parâmetro do tipo *CardRequestDto* para converter para tipo de retorno que é *CardEntity*.

Figura 11 - Exemplo da definição de mapeamento de *CardRequestDto* para *CardEntity*.

```
@Mapper
public interface CardMapper {
    1 usage  1 implementation
    CardEntity toCardEntity(CardRequestDto cardRequestDto);
```

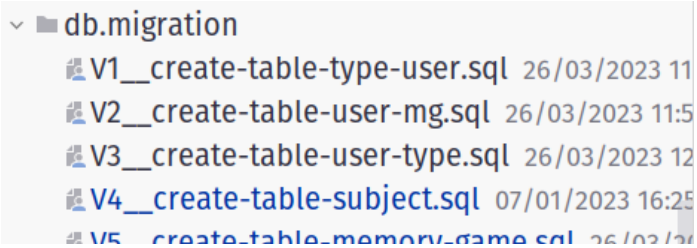
Fonte: Autoria própria.

MapStruct foi usado para facilitar a criação de mapeamento entre classes diferentes sem precisar implementar manualmente.

3.7. Flyway

O *Flyway* é um framework que permite controlar as criações das tabelas do banco de dados e usa o versionamento das migrações. Ele verifica a versão da migração e aplica novas migrações automaticamente antes que o restante do projeto seja iniciado. Para que as migrações ocorram automaticamente, é preciso criar arquivos SQLs com comandos necessários e armazenar na pasta db.migration como a figura 12. Desta forma, antes de o projeto começar a funcionar, *Flyway* executa migrações não aplicadas, como mostrado na figura 13, e depois disso, o projeto começa a rodar. Cada arquivo começa com número da versão em sequência, assim ele executa em ordem crescente das versões de migrações. (FLYWAY, 2023a, 2023b)

Figura 12 - Os arquivos SQLs na pasta db.migration.



Fonte: Autoria própria.

Figura 13 - As migrações aplicadas pelo Flyway mostradas no terminal.

```
2023-03-30T22:11:15.136-03:00  INFO 1013246 --- [  restartedMain] o.f.core.internal.command.DbMigrate      : Migrating schema "public" to version
 "1 - create-table-type-user"
2023-03-30T22:11:15.163-03:00  INFO 1013246 --- [  restartedMain] o.f.core.internal.command.DbMigrate      : Migrating schema "public" to version
 "2 - create-table-user-mg"
2023-03-30T22:11:15.187-03:00  INFO 1013246 --- [  restartedMain] o.f.core.internal.command.DbMigrate      : Migrating schema "public" to version
 "3 - create-table-user-type"
2023-03-30T22:11:15.207-03:00  INFO 1013246 --- [  restartedMain] o.f.core.internal.command.DbMigrate      : Migrating schema "public" to version
 "4 - create-table-subject"
2023-03-30T22:11:15.227-03:00  INFO 1013246 --- [  restartedMain] o.f.core.internal.command.DbMigrate      : Migrating schema "public" to version
 "5 - create-table-memory-game"
2023-03-30T22:11:15.244-03:00  INFO 1013246 --- [  restartedMain] o.f.core.internal.command.DbMigrate      : Migrating schema "public" to version
 "6 - create-table-card"
2023-03-30T22:11:15.262-03:00  INFO 1013246 --- [  restartedMain] o.f.core.internal.command.DbMigrate      : Migrating schema "public" to version
 "7 - create-table-memory-game-subject"
```

Fonte: Autoria própria.

Flyway foi escolhido para conseguir criar tabelas e relações no banco de dados pelos arquivos SQLs executados antes da execução do projeto. Cada versão da migração cria uma tabela como: tabela de usuário, tipo de usuário, de cartas e entre outros.

3.8. Spring

Spring é um framework open source para a plataforma Java que foi criado pelo australiano Rod Johnson em 2002, que facilita a construção de aplicações Web, Micro-serviço e outros. (WALLS, 2022; SPRING, 2023)

De acordo com site de *Spring*, as suas principais características são (SPRING, 2023):

- Usa a linguagem Java como base, embora tenha opções de usar Kotlin ou Groovy.
- Permite configurar de forma fácil graças ao *Spring Boot*, para que usuário possa focar no desenvolvimento sem se preocupar com configurações.
- Permite a construção de aplicação mais segura como exemplo, autenticação de usuário, através do *Spring Security*. Os desenvolvedores do *Spring* trabalham

> com profissionais de segurança para corrigir e testar quaisquer vulnerabilidades relatadas.

Alguns projetos conhecidos do *Spring* são *Spring Boot*, *Spring MVC*, *Spring Security* e *Spring Data JPA*.

3.8.1. Spring Boot

Spring Boot é uma ferramenta que configura todas as configurações no início do projeto por padrão sem a intervenção do usuário, ou seja, oferece a capacidade de criar aplicações que podem ser executadas sem a necessidade de configuração no formato *Extensible Markup Language* (XML) ou muitos códigos adicionais. Ele utiliza uma abordagem opinativa, ou seja, traz as práticas recomendadas que funcionarão para a maioria dos usuários na maioria das situações. Desta forma, outros usuários que têm experiência com essa estrutura terão facilidade imediata com a nova aplicação e poderão começar imediatamente. Portanto, o usuário pode alterar o comportamento padrão se ele desejar. (AZURE, 2023)

Outra característica importante do *Spring Boot* são as starters. Elas são dependências que agrupam outras dependências com um propósito em comum. Dessa forma, somente uma configuração é realizada no seu gerenciador de dependências como por exemplo Maven. Por exemplo, para incluir *Spring Data JPA* para acesso ao banco de dados, deve incluir a dependência *spring-boot-starter-datajpa* no projeto e essa dependência vai conter muitas das dependências necessárias para colocar um projeto em funcionamento rapidamente. (WALLS, 2022)

3.8.2. Spring MVC

Spring MVC é uma estrutura MVC completa orientada a *Hypertext Transfer Protocol* (HTTP), baseada em Servlets, que permite que você crie métodos com as anotações *@Controller* ou *@RestController* para lidar com solicitações HTTP recebidas. Os métodos em seu controlador são mapeados para HTTP usando as anotações *@RequestMapping*, *@GetMapping*, *@PostMapping*, *@PutMapping* e *@DeleteMapping*.

HTTP é um protocolo de cliente-servidor que permite a obtenção de recursos, como por exemplo, a estrutura de dados no formato de *JavaScript Object Notation* (JSON). Um

método HTTP, como *GET*, *POST*, *PUT* e entre outros, define a operação que o cliente deseja executar. Por exemplo, quando um cliente deseja buscar um recurso, deve usar *GET*, ou enviar as entradas de um formulário, deve usar o *POST*. Podem ser enviados vários tipos de respostas como texto, descrição do layout, imagens, vídeos, scripts, JSON, XML e entre outros. (MOZILLA, 2023)

Por exemplo, a classe *MemoryGameController* da figura 14, na linha 20, está anotada com *@RequestMapping* com endereço de "/jogo-de-memoria", o que significa que para ter acesso a este controlador, deve estar presente na URL para mandar requisição HTTP o caminho "..../jogo-de-memoria/…". E na linha 30, está presente *@GetMapping* do método *getAllMemoryGame()* e para ter acesso a este método, deve constar o caminho que foi usado no *@RequestMapping* na URL e usar a requisição HTTP *GET*, para assim receber a resposta.

Figura 14 - Exemplo de definição de Controller do Jogo de Memória.

```
19  @RestController
20  @RequestMapping("/jogo-de-memoria")
21  @CrossOrigin("*")
22  public class MemoryGameController {
23
        8 usages
24      @Autowired
25      private MemoryGameService memoryGameService;
26
        6 usages
27      @Autowired
28      private MemoryGameMapper memoryGameMapper;
29
        no usages
30      @GetMapping
31      @PreAuthorize("hasRole('ROLE_CRIADOR') or hasRole('ROLE_JOGADOR')")
32      public ResponseEntity getAllMemoryGame() throws Exception {
33          Set<MemoryGameEntity> memoryGameSet = memoryGameService.findAll();
34
35
36          return ResponseEntity.ok(memoryGameSet.stream() Stream<MemoryGameEntity>
37                                                  .map(memoryGameMapper::toMemoryGameResponseDto) Stream<MemoryGameResponseDto>
38                                                  .collect(Collectors.toSet()));
39      }
```

Fonte: Autoria própria.

3.8.3. Spring Security e JWT

Spring Security fornece autenticação, autorização e proteção contra ataques comuns. Com suporte de primeira classe para proteger aplicações imperativas e reativas, é o padrão para proteger aplicações baseadas em Spring. (DOCS SPRING SECURITY, 2023a)

Ele oferece várias configurações para poder deixar a aplicação mais protegida e controlar acesso para impedir que usuários não autorizados possam usar. Com ela é possível

controlar o acesso de um tipo usuário específico, como por exemplo, usando a anotação *@PreAuthorize* que ele oferece. (DOCS SPRING SECURITY, 2023b)

Por exemplo, a figura 15 mostra que só é possível salvar o jogo de memória no sistema, apenas se usuário for do tipo criador, que tem a permissão de criar o jogo, caso contrário vai dar erro. Isto é possível pois o método *saveMemoryGame* está anotado com *@PreAuthorize*.

Figura 15 - O método getCardsByCreatorAndMemoryGame.

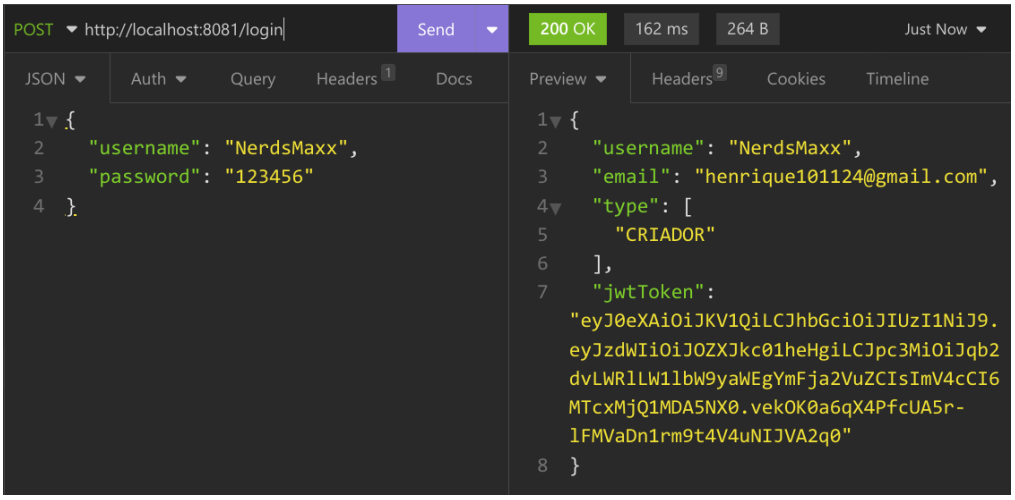
```
@PostMapping
@PreAuthorize("hasRole('ROLE_CRIADOR')")
public ResponseEntity saveMemoryGame(
        @RequestBody @Valid MemoryGameRequestDto memoryGameRequestDto,
        UriComponentsBuilder uriBuilder) throws Exception {
    MemoryGameEntity memoryGame = memoryGameService.save(memoryGameRequestDto);

    var uri = uriBuilder.path("/jogo-de-memoria/{id}") UriComponentsBuilder
                        .buildAndExpand(memoryGame.getId()) UriComponents
                        .toUri();

    return ResponseEntity.created(uri)
                         .body(memoryGameMapper.toMemoryGameResponseDto(memoryGame));
}
```

Fonte: Autoria do autor próprio.

Também é possível configurar para usar *JSON Web Tokens* (JWT), que é um dos padrões mais usados nas aplicações de produção. JWT é um padrão aberto seguindo RFC 7519, que define uma maneira de transmitir informações com segurança entre as partes como um objeto JSON. Essas informações podem ser verificadas e confiáveis porque são assinadas digitalmente. Os JWTs podem ser assinados usando um segredo (com o algoritmo HMAC) ou um par de chaves pública/privada usando RSA ou ECDSA. (AUTH0, 2023)

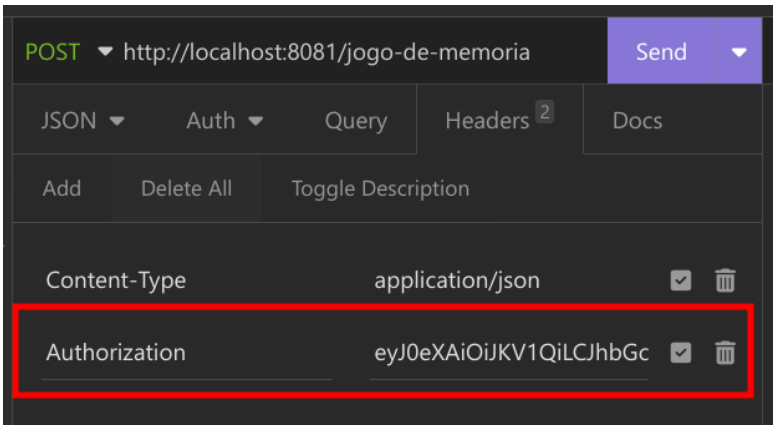
A figura 16 mostra que um usuário com nome de “NerdsMaxx” fez login e retornou um token JWT. Este token deve ser inserido no cabeçalho Authorization, como mostrado na figura 17, em outras requisições HTTP para sistema, para que usuário tenha acesso, caso contrário usuário não consegue ter acesso.

Figura 16 - Login feito com sucesso do usuário NerdsMaxx.



Fonte: Autoria do autor próprio.

Figura 17 - Token inserido no cabeçalho Authorization.



Fonte: Autoria do autor próprio.

3.8.4. Spring Data JPA, JPA e Hibernate

O *Spring Data* é um projeto do *Spring* para acesso e manipulação de dados. Ela facilita a configuração e utilização com banco de dados, seja ele relacional ou não. Ele tem subprojetos adaptados para cada tipo de banco de dados diferente, como *Spring Data JPA* para banco relacional (PostgreSQL, MySQL e etc) e *Spring Data MongoDB* para MongoDB.

Spring Data JPA traz consigo uma biblioteca Hibernate, que é uma implementação da especificação de *Java Persistence API* (JPA). A JPA é uma API padrão da linguagem Java que descreve uma interface comum para frameworks de persistência de dados. Também define um meio de mapeamento objeto-relacional para objetos feito em Java e traz uma forma de criar consulta complexa de banco de dados pelo *Java Persistence Query Language* (JPQL)

ao invés de SQL. JPQL é uma linguagem que permite especificar consultas que se referem a classes e propriedades de classes e é similar ao SQL. (IBM, 2023; BAUER et al, 2016)

Com JPA e *Hibernate*, é possível mapear a classe com uma determinada tabela no banco de dados relacional, usando o conceito de *Object Relational Mapper* (ORM). Na classe pode definir os atributos que vão representar as colunas da tabela, que quando obter dados do banco, ele vai gerar objetos com estes dados, facilitando o acesso e a manipulação dos dados pela orientação a objeto. Também é possível definir e ter acesso aos atributos que vão representar os relacionamentos entre as tabelas. ORM é uma forma de lidar de banco de dados pela orientação a objeto, para não ser preciso lidar com SQL diretamente. (BAUER et al, 2016)

Anotando em cima da classe com *@Entity* e *@Table*, já consegue ser mapeado para uma determinada tabela. Os atributos podem ser configurado com anotações *@Id* (para indicar que é uma chave primária), *@Column* (configurar se pode ser nulo ou não, ou se é unique), *@OneToOne* (para indicar que é relação de 1:1), *@OneToMany* (para indicar que é relação de 1:N), *@ManyToOne* (para indicar que é relação de N:1) e *@ManyToMany* (para indicar que é relação N:N). A figura 18 mostra este exemplo de uma das entidades. (BAUER et al, 2016)

Figura 18 - A classe *SubjectEntity* com anotações *@Entity, @Table, @Id, @Column, @ManyToMany*.

```
@Entity  21 usages
@Table(name = "subject")
public class SubjectEntity {
    @Id  no usages
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, unique = true)  no usages
    private String subject;

    @ManyToMany(mappedBy = "subjectSet")  no usages
    private Set<MemoryGameEntity> memoryGameSet;
}
```

Fonte: Autoria do autor próprio.

Outro recurso do *Spring Data JPA* é o uso da anotação *@Query*, que permite escrever consulta pelo JPQL. Por exemplo, na figura 19, tem-se o método *findAllByUsernamePlayerAndCreator* que usa anotação *@Query* para buscar as informações sobre matérias de um determinado jogador pelo nome de usuário do jogador e do criador que foi passado como parâmetro. (WALLS, 2022)

Figura 19 - O método findAllByUsernamePlayerAndCreator.

```
@Query("SELECT DISTINCT sub " +
       "FROM PlayerEntity pl " +
       "JOIN pl.memoryGameSet mg " +
       "JOIN mg.subjectSet sub " +
       "JOIN pl.user us " +
       "WHERE us.username = :username " +
       "AND mg.creator = :creator")
Set<SubjectEntity> findAllByUsernamePlayerAndCreator(String username, CreatorEntity creator);
```

Fonte: Autoria do autor próprio.

Spring foi escolhido como uma ferramenta para construção de backend por trazer uma variedade de soluções, como *Spring Boot* que facilita a configuração inicial, *Spring Data JPA* que facilita a manipulação com banco de dados, *Spring Security* que permite controle de acesso de usuário e *Spring MVC* que expõe os endereços que podem ser acessado externamente para ter acesso ao backend.

3.9. Dart e Flutter

Dart é uma linguagem de programação usada no framework Flutter e tem capacidade de executar no smartphone, computador e navegadores de internet. No caso do navegador, o código Dart é compilado para JavaScript, que por sua vez é executado em um navegador. (DART DEV, 2023a)

É uma linguagem "*type-safe*", ou seja, faz verificação de tipo estático para garantir que o valor de uma variável sempre corresponda ao tipo estático da variável que foi definida antes. E também oferece null safety, o que significa que os valores não podem ser nulos, a menos que o usuário deixe explícito que pode ser nula. (DART DEV, 2023a)

Já o Flutter é um framework para construir a interface de forma fácil e criar aplicativos multiplataforma compilados nativamente a partir de uma única base de código. As plataformas disponíveis são Android, IOS, Web e Desktop. (FLUTTER, 2023)

Flutter usa widgets para construir interface gráfica. Widgets são componentes visuais para definir a interface de um aplicativo (seja mobile, desktop e web). Ele oferece uma variedade de widgets como botões, cards, menus e muitos outros, para agilizar o desenvolvimento, assim como também personalizar e/ou criar widgets livremente para ser usado em outras telas. Cada widget pode receber um parâmetro de widget ou lista de widgets, que por sua vez podem também receber um parâmetro de widget ou lista de widgets. O parâmetro do widget é chamado de filho e quem recebe é chamado de pai. (WINDMILL et al., 2020)

A figura 20 mostra um widget *Column*, que é responsável por colocar os widgets em coluna, recebe uma lista de widgets como parâmetro. Nesta lista vão ter widgets filhos como *TextFormField*, responsável pelo campo de texto para que o usuário possa escrever, e SizedBox que é capaz de definir um tamanho específico de um espaço, ou de um widget filho caso seja passado como parâmetro. O primeiro *TextFormField* é responsável pelo nome de usuário ou email, já o segundo é responsável pela senha, ou seja, estes dois são responsáveis pelo login do usuário.

Figura 20 - Exemplo do um Widget no Flutter.

```
32  Column(
33  children: [
34    TextFormField(
35      validator: _controller.validatorUsernameOrEmail,
36      onChanged: (value) => _controller.username = value,
37      style: Theme.of(context).textTheme.headlineSmall,
38    ), // TextFormField
39    const SizedBox(
40      height: 30,
41    ), // SizedBox
42    TextFormField(
43      validator: _controller.validatorPassword,
44      obscureText: true,
45      autocorrect: false,
46      onChanged: (value) => _controller.password = value,
47      style: Theme.of(context).textTheme.headlineSmall,
48    ), // TextFormField
49  ],
50  ), // Column
```

Fonte: Autoria do autor próprio.

Flutter foi escolhido para construir front-end, pois usa apenas uma linguagem de programação e é multiplataforma, o que permite construir para web. A linguagem Dart traz um recurso de *null-safety* para trabalhar com variáveis nulas e pode ser compilado para JavaScript na web.

Além dos recursos nativos do Dart e Flutter, existem pacotes disponíveis no Pub.dev, que é repositório de pacotes oficial do Dart e Flutter, que podem ser usado para adicionar mais funcionalidades para aplicação ou facilitar desenvolvimento, como por exemplo *json_serializable*, *injectable*, *get_it* e entre outros.

3.10. json_serializable

Um pacote feito para facilitar a geração de código de mapeamento de JSON para objeto Dart e de objeto Dart para JSON. JSON significa *JavaScript Object Notation* e foi desenvolvida para usar como uma estrutura de transferência de dados. Para gerar código JSON, deve ter anotação *@JsonSerializable* em cima da classe, além de poder ajustar algumas configurações pela anotação. E também pode ser personalizado campos individuais usando anotação *@JsonKey* e fornecendo argumentos personalizados. (GOOGLE DEV, 2023; JACKSON, 2016)

Por exemplo, a figura 21 mostra a classe *MemoryGameModel* anotado com *@JsonSerializable* e é preciso criar dois métodos, um *fromJson* e outro *toJson*, que vão chamar as funções gerados automaticamente pelo pacote. E também foi usado *@JsonKey* para mudar o nome do campo do JSON caso o nome de campo trazido pelo servidor seja diferente do que está definido na classe Dart.

Figura 21 - A classe MemoryGameModel anotada com @JsonSerializable.

```dart
@JsonSerializable()
class MemoryGameModel {
  final String name;
  final String creator;
  final String? code;

  @JsonKey(name: 'subjectSet')
  final List<String>? subjectList;

  @JsonKey(name: 'cardSet')
  final List<CardModel>? cardList;

  const MemoryGameModel({
    required this.name,
    required this.creator,
    required this.subjectList,
    this.cardList,
    this.code,
  });

  factory MemoryGameModel.fromJson(Map<String, dynamic> json) => _$MemoryGameModelFromJson(json);

  Map<String, dynamic> toJson() => _$MemoryGameModelToJson(this);
}
```

Fonte: Autoria do autor próprio.

Ele foi escolhido para gerar duas funções para uma determinada classe para que não precise ser escrito manualmente. Uma função que converte para formato Json e outra função que converte para objeto da linguagem Dart.

3.11. *get_it e injectable*

O pacote *get_it* é um registrador de classes ou objetos para Dart com alguns itens que não são widgets. Ele permite registrar uma fábrica para sempre gerar o objeto pelo constructor ou constructor personalizado. Também permite registrar uma classe como singleton para que tenha apenas uma instância para toda aplicação. Ele também serve para injeção de dependências, já que com ele é possível instanciar uma classe já injetando as dependências automaticamente. (BURKHART, 2021)

Já o pacote *injectable*, ele traz um conjunto de anotações como *@singleton*, *@lazySingleton* e *@injectable*, para ser gerado um código automaticamente com *get_it*, facilitando o desenvolvimento. A figura 22 mostra o uso de *@injectable* na classe *MemoryGameService*, que é uma classe responsável por se comunicar com backend para buscar os dados. Ele depende das classes *Auth* e *HttpInterface*. (AKARIE, 2023b)

Figura 22 -A classe *MemoryGameService* anotada com *@injectable*.

```dart
@injectable
class MemoryGameService extends Service {
  const MemoryGameService(super.auth, @Named.from(HttpImpl) super.http);

  static const String path = 'jogo-de-memoria';

  Future<List<MemoryGameModel>> getAllMemoryGame() async {
    Response response = await http.get(
      path,
      auth: auth,
    );

    dynamic json = jsonDecode(convert(response.body));

    if (json is List) {
      return json.map((memoryGame) => MemoryGameModel.fromJson(memoryGame)).toList();
    }

    throw CustomException('Não foi possível obter os jogos de memória.');
  }
```

Fonte: Autoria do autor próprio.

Estes pacotes foram escolhidos para lidar com objetos que não fazem parte da interface gráfica, como por exemplo, a classe de serviço, que têm a responsabilidade de comunicar com backend e transformar os dados recebidos em JSON para objeto em Dart.

3.12. shared_preferences

O pacote *shared_preferences* é uma ferramenta que permite o armazenamento persistente para dados simples. Usa *LocalStorage* para a plataforma Web. Ele pode armazenar os dados sem data de expiração e pode ser acessado a qualquer momento, mesmo que o navegador seja fechado. (FLUTTER DEV, 2023; W3 SCHOOLS, 2023).

A classe *LocalStorage* (diferente do que foi mencionado antes), como mostra a figura 23, foi criado para facilitar o uso do pacote *shared_preferences* como por exemplo, o método setString permite inserir uma chave e o valor em formato de String no localStorage da Web e o método getString permite obter o valor de acordo com a chave dada, caso o valor não esteja presente, ele retorna nulo.

Figura 23 - A classe LocalStorage.

```dart
class LocalStorage {
  LocalStorage._();

  static late final SharedPreferences _sharedPreferences;

  static void init() async {
    _sharedPreferences = await SharedPreferences.getInstance();
  }

  static Future<void> setString(String key, String value) async {
    await _sharedPreferences.setString(key, value);
  }

  static String? getString(String key) {
    return _sharedPreferences.getString(key);
  }
```

Fonte: Autoria do autor próprio.

O pacote *shared_preferences* foi escolhido para lidar com armazenamento de dados na Web usando localStorage, o que permite permanecer estes dados mesmo que a página reinicie.

3.13. auto_route

É um pacote de navegação que permite navegar de uma página a outra através de rotas, a passagem de parâmetros, e usa geração de código para simplificar a configuração de rotas com uso da anotação *@MaterialAutoRoute*. Ele permite também restringir a página para

um tipo de usuário específico com uso de guardas de rotas. Por exemplo, para navegar para a página de edição de cartas, deve-se inserir a rota dela e os parâmetros necessários, como mostra a figura 24. (AKARIE, 2023a)

Figura 24 - Navegação para outra página.

```
context.router.push(
  CardsEditingRoute(memoryGameName: memoryGame.name),
);
```

Fonte: Autoria do autor próprio.

Com a anotação *@MaterialAutoRouter*, é possível configurar as rotas que serão usadas na Web, por exemplo, na figura 25, a primeira rota é de login, onde será usado página de login para que usuário possa logar. (AKARIE, 2023a)

Figura 25 - Configuração de rotas por meio da anotação @MaterialPageRoute.

```
@MaterialAutoRouter(
  routes: <AutoRoute>[
    AutoRoute(
      page: LoginPage,
      name: 'LoginRoute',
      path: '/login',
    ), // AutoRoute
    AutoRoute(
      page: CreateAccount,
      name: 'CreateAccountRoute',
      path: '/create',
    ), // AutoRoute
    AutoRoute(
      page: ChangePasswordPage,
      name: 'ChangePasswordRoute',
      path: '/change-password',
    ), // AutoRoute
    AutoRoute(
      page: DashboardPage,
      name: 'DashboardRoute',
      path: '/dashboard',
      guards: [AuthGuard],
    ), // AutoRoute
```

Fonte: Autoria do autor próprio.

A figura 26 mostra um exemplo de uma guarda de rota, que vai permitir que usuário acesse outras rotas desde que esteja logado, senão irá para página de login.

Figura 26 - Uma guarda de rota para controlar o acesso do usuário a páginas.

```dart
class AuthGuard extends AutoRouteGuard {
  @override
  void onNavigation(NavigationResolver resolver, StackRouter router) {
    final Auth auth = getIt<Auth>();

    if (auth.isValid()) {
      resolver.next();
    } else {
      router.push(const LoginRoute());
    }
  }
}
```

Fonte: Autoria do autor próprio.

O pacote *auto_route* foi escolhido para lidar com a navegação de uma página para outra e controlar acesso das páginas dependendo se usuário for logado ou não.

3.14. form_validator

Os usuários podem incluir informações incorretas no campo de texto e caso não seja validado, pode gerar algum resultado não desejável. Por exemplo, na criação de conta, uma senha deve ter no mínimo 6 caracteres, mas sem validação o usuário pode digitar 4 caracteres e mandar para servidor, mas o servidor não aceitará esta conta nova, já que não está de acordo com a regra de no mínimo 6 caracteres. Para tornar a aplicação mais segura e fácil de usar, as informações fornecidas devem ser validadas, caso não sejam válidas, deve ser exibida a mensagem de erro. (DOCS FLUTTER DEV, 2023)

O pacote *form_valiidator* permite construir as validações personalizadas, como por exemplo, na figura 27, a variável *validatorUsername*, que é responsável por validar o nome de usuário, vai receber parâmetro *requiredMessage* que é uma mensagem de erro que vai aparecer caso usuário não digite o nome, e com com *minLength*, vai limitar no mínimo 5 caracteres e aparecer a mensagem de erro caso usuário não digite no mínimo 5 caracteres, como mostra a figura 28. (THEMISIR.COM, 2023)

Figura 27 - Um validador personalizado para nome de usuário.

```
final validatorUsername =
    ValidationBuilder(requiredMessage: 'É obrigatório preencher usuário.')
        .minLength(5, 'Nome de usuário deve ser no mínimo 5 caracteres')
        .build();
```

Fonte: Autoria do autor próprio.

Figura 28 - Exemplo de mensagens de erro.

Fonte: Autoria do autor próprio.

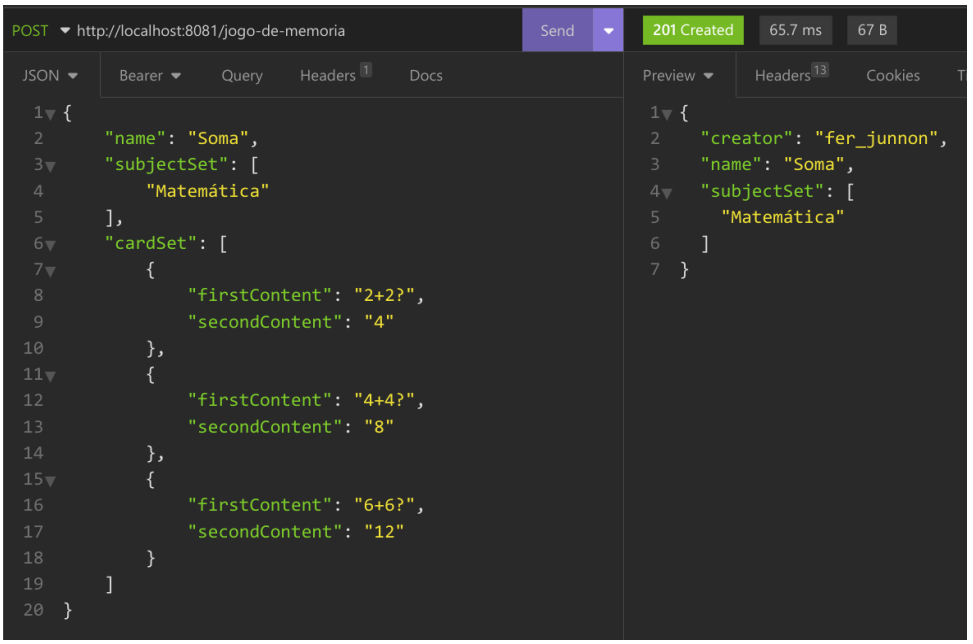
O pacote *form_validator* foi escolhido para facilitar a criação de validadores personalizados para validar as entradas dos usuários.

3.15. Insomnia

Insomnia é um cliente de API que permite enviar solicitações para API pelo aplicativo. Ele permite testar a requisição com métodos HTTP como *GET*, *POST*, *PUT* ou *DELETE* pela URL. Também permite adicionar token de autenticação caso o API exija usuário logado. (DOCS INSOMNIA REST, 2021)

A figura 29 mostra um exemplo de criação de um jogo de memória chamado Soma. O método selecionado foi *POST*, já que envolve criação de recurso e os dados foram mandados no formato de JSON. Depois do envio desta requisição, é retornada um código 201 que significa que foi criado um recurso e traz uma resposta a esta criação também no formato JSON. Insomnia também permite enviar dados de outros formatos como XML e *YAML Ain't Markup Language* (YAML). (DOCS INSOMNIA REST, 2021)

Figura 29 - Exemplo de envio de requisição que cria jogo de memória.



Fonte: Autoria do autor próprio.

A Insomnia foi escolhida para testar as solicitações e verificar se os resultados gerados pelo API são resultados esperados ou errados.

4. REQUISITOS FUNCIONAIS, NÃO FUNCIONAIS E REGRAS DE NEGÓCIO

Requisitos funcionais:

Tabela 2 - Tabela de Requisitos Funcionais

(continua)

Identificador	**Nome**	**Descrição**
RF001	Cadastrar usuário	No cadastro de conta, o usuário deve registrar seu nome, nome de usuário, email, senha e informar o tipo de usuário que ele deseja ser, podendo ser criador, jogador ou ambos.
RF002	Logar no sistema	O usuário deve digitar nome de usuário e senha, desde que sejam corretos, para logar no sistema.
RF003	Alterar a senha	Caso o usuário esqueça a senha, ele pode alterar a senha, digitando seu email, nova senha e confirmação da nova senha para que usuário consiga logar no sistema posteriormente.
RF004	Criar ou editar jogo de memória	O usuário do tipo criador (mesmo se ele for do tipo de jogador também) deve digitar o nome do jogo de memória, incluir as matérias se ele desejar e criar as cartas na criação do jogo de memória. Ele pode editar cartas criadas anteriormente. Mas caso seja edição do jogo de memória, o usuário pode editar o nome do jogo de memória, adicionar ou remover matérias, editar as cartas já existentes e adicionar novas cartas.
RF005	Gerar a partida	Cada partida criada pelo usuário do tipo criador (mesmo se ele for do tipo jogador também) terá seu código gerado para que outros jogadores possam usá-lo para entrar na partida.
RF006	Entrar para partida pelo código	Usuário do tipo jogador (mesmo se for do tipo criador também) pode entrar para partida pelo código e começar a jogar.

Tabela 2 - Tabela de Requisitos Funcionais.

(continuação)

Identificador	**Nome**	**Descrição**
RF007	Jogar jogo de memória durante a partida	Durante a partida, quando usuário do tipo jogador (mesmo se for do tipo criador também) selecionar duas cartas, aparecerá duas opções "Está certo!" e "Está errado!". Se as duas cartas forem corretas: ● Se for clicado na opção "Está certo!", as cartas não podem ser selecionadas. ● Se for clicado na opção "Está errado!", as cartas são viradas e podem ser selecionadas novamente. Se as duas cartas forem erradas: ● Se for clicado na opção "Está certo!", as cartas são viradas e podem ser selecionadas novamente. ● Se for clicado na opção "Está errado!", as cartas são viradas e podem ser selecionadas novamente. O usuário vai repetir esses passos até que todas as cartas sejam viradas para cima. Também irá atualizar o resultado do usuário durante a partida que são pontuação, tentativas e encontrados.
RF008	Jogar o jogo de memória individual-mente	Caso o usuário tenha tipo jogador, os jogos de memória, que foram jogados anteriormente e salvo no sistema, vão ter opção de jogar o jogo como se fosse uma partida, mas individualmente. E se ele tiver o tipo criador também, ele pode jogar os jogos de memória que ele criou. Porém, se o usuário tiver somente o tipo criador, ele poderá somente testar o jogo.
RF009	Acompanhar as partidas atuais	A lista de partidas atuais ocorrendo no momento mostrará as partidas atuais que usuário do tipo criador (mesmo se ele for do tipo jogador também) criou e usuário poderá escolher uma das partidas para ver os resultados dos jogadores que estiveram presentes na partida.

Tabela 2 - Tabela de Requisitos Funcionais.

(conclusão)

Identificador	**Nome**	**Descrição**
RF010	Consultar histórico das partidas jogados anteriormen-te	O histórico das partidas jogados anteriormente mostrará os resultados das partidas que usuário do tipo jogador (mesmo se ele for do tipo criador também) jogou anteriormente.
RF011	Consultar histórico das partidas criadas anteriormen-te	O histórico das partidas criadas anteriormente mostrará o histórico das partidas que usuário do tipo criador (mesmo se ele for do tipo jogador também) criou anteriormente e poderá escolher uma das partidas para ver os resultados dos jogadores que estiveram presentes na partida.
RF012	Consultar a pontuação	A tela de pontuação deve mostrar os resultados do(s) usuário(s). Ela aparece: ● Quando o usuário do tipo jogador finaliza a partida, na qual usuário entrou pelo código, mostra os resultados de todos usuários que finalizaram a partida até o momento. ● Quando o usuário do tipo jogador finaliza a partida na qual usuário jogou individualmente, só mostra o resultado do próprio usuário. ● Quando usuário do tipo criador clica em uma partida atual na tela de acompanhamento de partidas atuais e visualiza os resultados dos usuários que participaram. ● Quando usuário do tipo criador clica em uma partida na tela de histórico das partidas criadas anteriormente e visualiza os resultados dos usuários que participaram. ● Quando usuário do tipo jogador visualiza os seus resultados das partidas jogadas anteriormente.

Fonte: Autoria própria.

Requisitos não funcionais:

Tabela 3 - Requisitos Não Funcionais.

(continua)

Identificador	**Nome**	**Descrição**
RNF001	Autenticação	Usuário deve ser autenticado pelo nome de usuário e senha para que consiga ter acesso ao sistema. Caso não autentique, ele não poderá ter acesso aos recursos do sistema.
RNF002	Portabilidade	O sistema poderá rodar em qualquer navegador que seja compatível com Flutter, independente do sistema operacional.
RNF003	Desempenho	O sistema deve ser rápido para responder às solicitações.
RNF004	Usabilidade	O sistema deve ser fácil de aprender para que usuário possa ter facilidade de aproveitar o que o sistema oferece.
RNF005	Disponibilidade	O sistema deve estar disponível por 24 horas para que usuário possa acessar o sistema por qualquer horário.
RNF006	Permissões do usuário do tipo criador	O usuário do tipo criador têm as seguintes permissões: ● Criação ou edição de jogo de memória. ● Testar o jogo de memória. ● Gerar código para criar uma partida. ● Acompanhamento das partidas atuais (partidas que não passaram de 2 horas). ● Histórico das partidas criadas anteriormente (partidas que passaram de 2 horas).
RNF007	Permissões do usuário do tipo jogador	O usuário do tipo jogador têm as seguintes permissões: ● Entrar na partida pelo código. ● Jogar jogos de memória jogados anteriormente. ● Histórico das partidas jogadas anteriormente.
RNF008	Tipos de usuário	O usuário deve ser do tipo criador ou jogador, mas também pode ser ambos.
RNF009	Lista de jogo de memória	Caso usuário tenha dois tipos, a lista de jogos de memória devem ser separados dos que foram criados e dos que foram jogados.

Tabela 3 - Requisitos Não Funcionais.

(continuação)

Identificador	**Nome**	**Descrição**
RNF010	Tela de dashboard	A tela de dashboard deve mostrar as opções de criar jogo de memória, acompanhar as partidas atuais, histórico das partidas criadas anteriormente e histórico das partidas jogados anteriormente. A tela mostrará a lista de jogos de memória que o usuário possui. Haverá as opções de visualizar a lista de jogo de memória que o usuário criou e jogou. Algumas opções podem não aparecer se usuário contém apenas um tipo, porém se conter ambos, aparecem todas as opções.
RNF011	Jogo de memória	O usuário só pode selecionar 2 cartas por vez durante a partida. Se o usuário acertar um par de cartas (cartas corretas e opção "Está certo!"), as cartas não podem ser selecionadas novamente, caso contrário, o usuário pode selecionar novamente. A partida só termina quando o usuário acertar todas as cartas.
RNF012	Jogos de memória salvos	Os usuários que entraram na partida pelo código, o jogo de memória usado na partida deve ser salvo na conta destes usuários.
RNF013	Acompanhamento das partidas atuais	A lista de partidas atuais ocorrendo no momento deve ser restrita somente ao usuário criador (mesmo se ele for do tipo jogador também). São consideradas as partidas atuais somente aqueles que não passaram de 2 horas. (consultar RN002 na tabela de regras de negócio). Para cada partida atual, deve conter o código e nome do jogo de memória.
RNF014	Histórico das partidas jogados anteriormente	O histórico das partidas jogados anteriormente deve ser restrita apenas para usuário que tenha tipo de jogador (mesmo se ele for do tipo criador também). São consideradas partidas jogados anteriormente somente aqueles que o usuário jogou tanto com outros usuários, quanto individualmente (inclusive de jogos de memória criado pelo próprio usuário, se ele for do tipo criador também).

Tabela 3 - Requisitos Não Funcionais.

(conclusão)

Identificador	**Nome**	**Descrição**
RNF015	Histórico das partidas criadas anteriormente	O histórico das partidas criadas anteriormente deve ser restrita apenas para usuário que tenha tipo de criador (mesmo se ele for do tipo jogador também). São consideradas partidas anteriores somente aqueles que passaram de 2 horas e não podem ser entradas novamente. (consultar RN002 na tabela de regras de negócio). Para cada partida, deve conter: ● Nome de jogo de memória. ● Código usado para partida. ● Quantidade de jogadores que participaram. ● Data e hora que começou a partida. ● Data e hora que último jogador terminou.
RNF016	Resultado do usuário na tela de pontuação	O resultado do usuário na tela de pontuação deve conter: ● Nome de jogo de memória. ● Pontuação. ● Nome do jogador (na visão do usuário do tipo criador) ou criador (na visão do usuário do tipo jogador). ● Quantidade de opções certas, ou seja, quantidade de vezes que usuário acertou a opção. ● Quantidade de opções erradas, ou seja, quantidade de vezes que usuário errou a opção. ● Quantidade de tentativas, ou seja, quantidade de vezes que usuário selecionou pares de cartas. ● Data e hora de quando usuário começou a jogar o jogo de memória. ● Data e hora de quando usuário terminou de jogar o jogo.

Fonte: Autoria própria.

Regras de negócio:

Tabela 4 - Regras de negócio

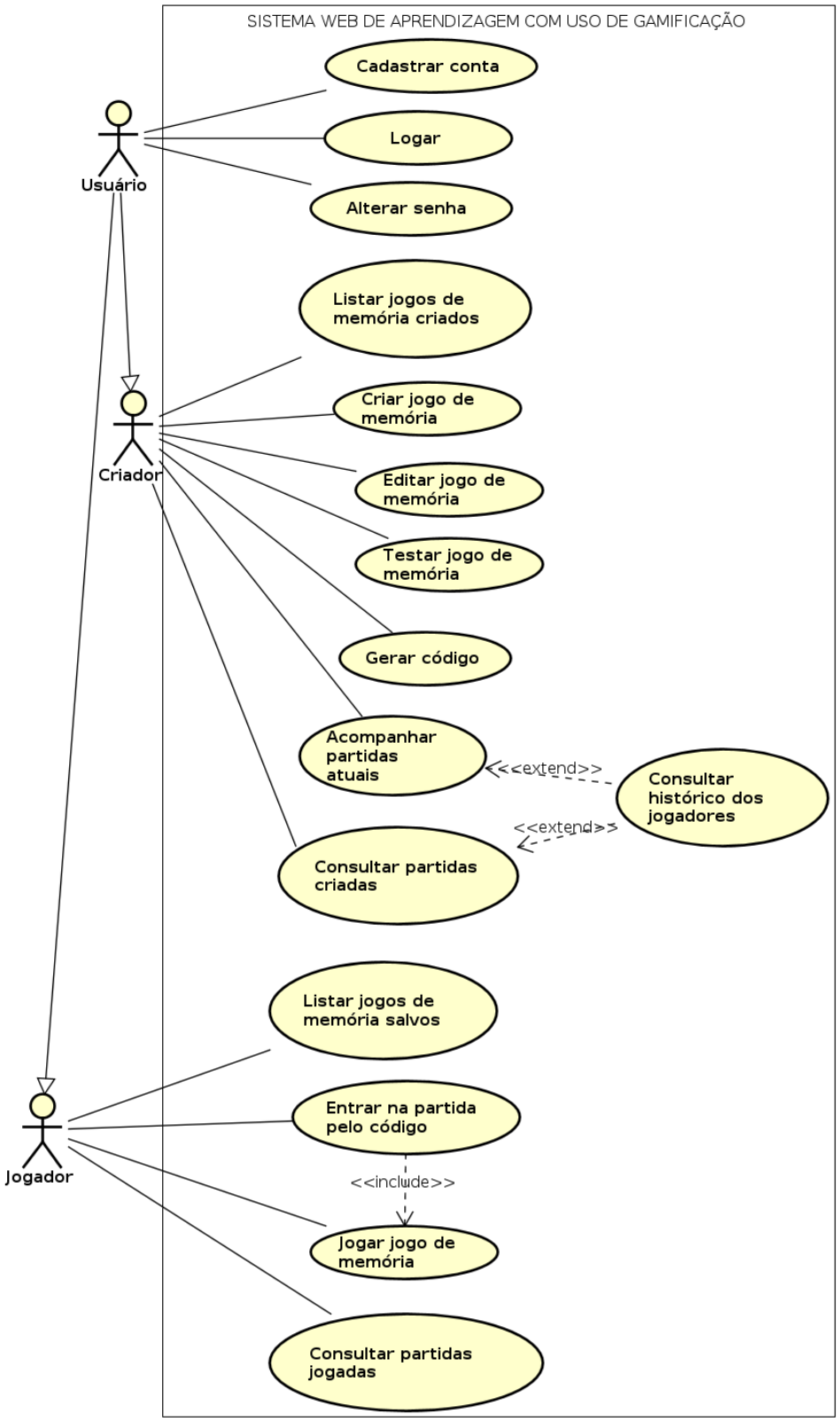
Identificador	**Nome**	**Descrição**
RN001	Jogo de memória	É obrigatório que jogo de memória tenha nome e no mínimo um par de cartas, porém as matérias não são obrigatórias associar com jogo de memória.
RN002	Código da partida	O código gerado pelo usuário criador terá duração de 2 horas. Passado 2 horas, o código não é mais válido e os usuários que não conseguiram entrar na partida não poderão usar este código que foi invalidado.
RN003	Pontuação da partida	Se as duas cartas selecionadas forem corretas: ● Se for clicado na opção "Está certo!", ganha 50 pontos na pontuação. ● Se for clicado na opção "Está errado!", perde 25 pontos na pontuação. Se as duas cartas selecionadas forem erradas: ● Se for clicado na opção "Está certo!", perde 30 pontos na pontuação. ● Se for clicado na opção "Está errado!", não ganha nem perde a pontuação. Pontuação final: x = quantidade de opção “Está certo!” quando as duas cartas estão certas. y = quantidade de opção “Está errado!” quando as duas cartas estão certas. z = quantidade de opção “Está certo!” quando as duas cartas estão erradas. $Pontuação = (50 * x) - (25 * y) - (30 * z)$

Fonte: Autoria própria.

5. DIAGRAMA DE CASO DE USO E CASOS DE USO DESCRITIVOS:

Diagrama de caso de uso:

Figura 30 - Diagrama de caso de uso



Fonte: Autoria própria.

Casos de usos descritivos:

Tabela 5 - Caso de uso de cadastrar a conta.

Identificador: CSU 001
Nome: Cadastrar conta.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que os atores criem contas para entrar no sistema.
Atores: Qualquer ator.
Pré-condições: O ator deve ter acesso a internet.
Pós-condições: N/A.
Cenário Principal:

1. O sistema apresenta uma página de login que vai estar presente um botão de cadastrar uma nova conta.
2. O ator clica nesta opção e vai para a tela de cadastro. Nesta tela, vão estar presentes os campos de texto de nome, nome de usuário, email e senha e as opções de tipo de usuário.
3. O ator digita seu nome, nome de usuário, email, senha e seleciona que tipo de usuário deseja ser (jogador, criador ou ambos).
4. O ator consegue cadastrar no sistema e volta para tela de login para logar no sistema.

Extensões (ou Fluxo Alternativo):

a. Usuário digita nome com menos de 6 caracteres:
   1. O sistema exibe mensagem de erro: "O nome deve ter no mínimo 6 caracteres.".
   2. Usuário digita nome com mais ou igual a 6 caracteres e consegue cadastrar no sistema.

b. Usuário digita nome de usuário com menos de 6 caracteres:
   3. O sistema exibe mensagem de erro: "O nome de usuário deve ter no mínimo 6 caracteres.".
   4. Usuário digita nome de usuário com mais ou igual a 6 caracteres e consegue cadastrar no sistema.

c. Usuário digita email inválido:
   1. O sistema exibe mensagem de erro: “Email inválido. O e-mail deve ser válido.”
   2. Usuário digita e-mail válido e consegue cadastrar no sistema.

d. O usuário digita a senha com menos de 6 caracteres:
   1. O sistema exibe mensagem de erro "A senha deve ter no mínimo 6 caracteres.".
   2. Usuário digita a senha com mais ou igual a 6 caracteres e consegue cadastrar no sistema.

e. O usuário tenta logar e o servidor deu erro.
   1. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 6 - Caso de uso de logar.

Identificador: CSU 002
Nome: Logar.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que os atores loguem no sistema.
Atores: Qualquer ator.
Pré-condições: O ator deve ter acesso a internet.
Pós-condições: N/A.
Cenário Principal:

1. O sistema apresenta uma página de login que vão estar presente os campos de texto de nome de usuário e senha.
2. O ator digita nome de usuário e senha;
3. O ator consegue entrar no sistema;
4. O ator vai para tela de dashboard.

Extensões (ou Fluxo Alternativo):

a. Usuário digita nome de usuário com menos de 6 caracteres:

1. O sistema exibe mensagem de erro: "O nome de usuário deve ter no mínimo 6 caracteres.".
2. Usuário digita nome de usuário correto com mais ou igual a 6 caracteres e consegue logar no sistema.

b. O usuário digita a senha com menos de 6 caracteres:

1. O sistema exibe mensagem de erro "A senha deve ter no mínimo 6 caracteres.".
2. Usuário digita a senha correta com mais ou igual a 6 caracteres e consegue logar no sistema.

c. O usuário digita nome de usuário e senha respeitando limite de quantidades de caracteres mas o sistema nega o login por elas não estarem corretas.

1. O sistema exibe a mensagem de erro "O nome de usuário/senha estão erradas! Tente novamente.".
2. Usuário digita o nome de usuário e senha corretamente e consegue logar no sistema.

d. O usuário tenta logar e o servidor deu erro.

1. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 7 - Caso de uso de alterar a senha.

Identificador: CSU 003
Nome: Alterar senha.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que os atores alterem a senha.
Atores: Qualquer ator.
Pré-condições: O ator deve ter acesso a internet.
Pós-condições: N/A.
Cenário Principal:

1. O sistema apresenta uma página de login que vai estar presente um botão de alterar uma senha.
2. O ator clica nesta opção e vai para a tela de alteração de senha. Nesta tela, vão estar presentes os campos de texto de nome de usuário e a senha.
3. O ator digita nome de usuário e senha.
4. O ator consegue alterar a senha no sistema e volta para tela de login para logar no sistema.

Extensões (ou Fluxo Alternativo):

a. Usuário digita nome de usuário com menos de 6 caracteres:

3. O sistema exibe mensagem de erro: "O nome de usuário deve ter no mínimo 6 caracteres.".
4. Usuário digita nome de usuário correto com mais ou igual a 6 caracteres e consegue logar no sistema.

b. O usuário digita a senha com menos de 6 caracteres:

3. O sistema exibe mensagem de erro "A senha deve ter no mínimo 6 caracteres.".
4. Usuário digita a senha correta com mais ou igual a 6 caracteres e consegue logar no sistema.

c. O usuário digita nome de usuário e senha respeitando limite de quantidades de caracteres mas o sistema não acha usuário no banco de dados.

3. O sistema exibe a mensagem de erro "Usuário não encontrado! Digite um nome de usuário válido.".
4. Usuário digita o nome de usuário corretamente e consegue alterar a senha no sistema.

d. O usuário tenta alterar a senha e o servidor deu erro.

2. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 8 - Caso de uso de listar jogos de memória criados.

Identificador: CSU 004
Nome: Listar jogos de memória criados.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que o ator possa visualizar os jogos de memórias criados pelo criador.
Atores: Criador.
Pré-condições: O ator deve ter acesso a internet, estar devidamente logado e criado pelo menos um jogo de memória.
Pós-condições: N/A.
Cenário Principal:

1. Na tela de dashboard, o sistema apresenta a opção de visualizar os jogos de memória criados anteriormente.
2. O ator clica nesta opção.
3. O sistema carregará os jogos de memórias criados pelo criador anteriormente.

Observação: Caso o usuário seja somente do tipo criador, o sistema já apresenta automaticamente a lista.

Extensão (ou Fluxo Alternativo):
a. O sistema não encontra os jogos de memória criados pelo ator.

1. O sistema exibe a mensagem: "Crie seu jogo de memória agora :) !".

b. O sistema tentou carregar a lista e o servidor deu erro:

2. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 9 - Caso de uso de criar jogo de memória.

Identificador: CSU 005
Nome: Criar jogo de memória.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que os atores criem jogo de memória.
Atores: Criador.
Pré-condições: O ator deve ter acesso a internet, estar devidamente logado.
Pós-condições: N/A.
Cenário Principal:

1. Na tela de dashboard, o sistema apresenta a opção de criação de jogo de memória.
2. O ator clica nesta opção.
3. Dentro da tela de criação, o ator escreve o nome do jogo de memória.
4. O ator escreve as matérias separadas pela vírgula “,”. (opcional)
5. O ator clica na carta de ‘+’ para adicionar as cartas.
6. O sistema apresenta a tela de criação ou edição de duas cartas.
7. O ator escreve o conteúdo nas duas cartas.
8. O ator clica no botão de “Aplicar alteração” ou “Cancelar”.
    - 8.1. O ator clica no botão de “Aplicar alteração”.
        - 8.1.1. O ator visualiza as cartas criadas por ele.
    - 8.2. O ator clica no botão de “Cancelar”.
        - 8.2.1. As cartas são canceladas e o ator visualiza as mesmas cartas apresentadas anteriormente, caso as cartas tenham sido criadas antes.
10. Se o ator desejar criar mais cartas, o ator pode clicar na carta de ‘+’ novamente para adicionar outra carta, ou se desejar editar um par de cartas, deve clicar no botão de "Editar" para editar um par de cartas, repetindo o passo 6.
11. O ator clica no botão de salvar para salvar o jogo de memória junto com as cartas.
12. O ator volta para tela de dashboard.

Extensões (ou Fluxo Alternativo):
a. O ator tenta salvar sem nome do jogo de memória ou nome do jogo com menos de três caracteres:

1. O sistema exibe mensagem de erro: "O nome do jogo de memória deve ter no mínimo três caracteres.".
2. Usuário digita nome do jogo de memória com três caracteres ou mais e consegue salvar jogo de memória.

b. O ator tenta salvar o jogo de memória sem cartas:

1. O sistema exibe mensagem de erro "Jogo de memória deve ter no mínimo um par de cartas.".
2. Usuário cria par de cartas ou mais e consegue salvar jogo de memória.

c. O ator tenta salvar o jogo de memória e o servidor deu erro:

1. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 10 - Caso de uso de editar jogo de memória.

Identificador: CSU 006
Nome: Editar jogo de memória.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que os atores editem o jogo de memória.
Atores: Criador.
Pré-condições: O ator deve ter acesso a internet, estar devidamente logado e criado jogo de memória antes.
Pós-condições: N/A.
Cenário Principal:

1. Na tela de dashboard, o sistema apresenta a opção de edição em cada jogo de memória já criado.
2. O ator clica nesta opção.
3. Dentro da tela de edição, o ator pode:
    - 3.1. Mudar nome do jogo de memória.
    - 3.2. Adicionar ou remover matérias, escreve as matérias separadas pela vírgula “,”.
    - 3.3. Clicar em uma carta já criada.
        - 3.3.1. O sistema apresenta a tela de edição de duas cartas.
        - 3.3.2. O ator edita os conteúdos nas duas cartas.
        - 3.3.3. O ator clica na opção de “Aplicar alteração”.
        - 3.3.4. O ator visualiza as cartas editadas por ele.
    - 3.4. Clicar na carta de ‘+’ para adicionar um par de cartas.
        - 3.4.1. O sistema apresenta a tela de criação de duas cartas.
        - 3.4.2. O ator escreve o conteúdo nas duas cartas.
        - 3.4.3. O ator clica no botão de “Aplicar alteração”.
        - 3.4.4. O ator visualiza as cartas criadas por ele.
4. Se o ator deseja criar ou editar outras cartas, deve clicar na carta de ‘+’ ou em uma das cartas já criadas, repetindo o passo 3.3 ou 3.4.
5. O ator clica no botão de salvar para salvar o jogo de memória editada junto.
6. O ator volta para tela de dashboard.

Extensão (ou Fluxo Alternativo):

a. O ator tenta salvar sem nome do jogo de memória ou nome do jogo com menos de três caracteres:

1. O sistema exibe mensagem de erro: "O nome do jogo de memória deve ter no mínimo três caracteres.".
2. Usuário digita nome do jogo de memória com três caracteres ou mais e consegue salvar jogo de memória.

b. O ator tenta salvar o jogo de memória sem cartas:

1. O sistema exibe mensagem de erro "Jogo de memória deve ter no mínimo um par de cartas.".
2. Usuário cria par de cartas ou mais e consegue salvar jogo de memória.

c. O ator tenta salvar o jogo de memória e o servidor deu erro:

1. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 11 - Caso de uso de testar o jogo de memória.

Identificador: CSU 007
Nome: Testar jogo de memória.
Responsável: Guilherme Henrique Mendonça Nascente
Descrição/Resumo: Possibilita que atores possam testar os jogos de memória criados.
Atores: Criador.
Pré-condições: O ator deve estar devidamente cadastrado, ter acesso a internet, ter somente o tipo de criador e criado pelo menos um jogo de memória.
Pós-condições: N/A
Cenário Principal:

1. O sistema apresenta a opção de testar jogo de memória em cada jogo de memória jogado anteriormente;
2. O ator clica nesta opção.
3. Dentro da tela de partida de jogo de memória que apresenta as cartas, o ator pode jogar novamente o jogo de memória;
4. O ator clica em duas cartas.
5. Aparece a tela destas duas cartas e as opções “Está certo!” e “Está errado!”;
6. O ator escolhe a opção que acha estar certa.
7. As cartas vão ser desviradas caso as cartas estejam erradas, senão elas vão continuar viradas para cima. E a pontuação vai ser calculada de acordo com a regra de negócio RN003 (consultar na tabela de regras de negócio).
8. Repete o passo 4 até o ator acertar todas as cartas.
9. Depois de acertar todas as cartas, apresenta a pontuação, porém não vai ser salvo o resultado no sistema.
10. O ator aperta no botão “Voltar para dashboard”
11. O ator volta para tela de dashboard.

Extensão (ou Fluxo Alternativo):
a. O sistema tenta carregar o jogo de memória e o servidor deu erro:

1. O sistema exibe a mensagem de erro "Tem algo errado com servidor, contate com administrador.".

Fonte: Autoria própria.

Tabela 12 - Caso de uso de gerar código para jogo de memória.

Identificador: CSU 008
Nome: Gerar código.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que os atores criem um código para partida do jogo de memória e compartilhem com outros atores.
Atores: Criador.
Pré-condições: O ator deve ter acesso a internet, estar devidamente logado e criado jogo de memória antes.
Pós-condições: N/A.
Cenário Principal:

1. Na tela de dashboard, o sistema apresenta a opção de geração de código em cada jogo de memória já criado.
2. O ator clica nesta opção;
3. Dentro da tela de geração de código.
4. O ator clica na opção de "Gerar código".
5. Aparece o código gerado e deve copiar o código para compartilhar com outros.
6. Se o ator desejar de gerar outro código do mesmo jogo de memória, deve clicar em "Gerar outro código" e repetir o passo 5.
7. Se o ator desejar gerar código em outro jogo de memória, repete o passo 1.
8. O ator volta para tela de dashboard.

Extensão (ou Fluxo Alternativo):
a. O ator tenta gerar código do jogo de memória e o servidor deu erro:

1. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 13 - Caso de uso de acompanhar as partidas atuais.

Identificador: CSU 009
Nome: Acompanhar partidas atuais.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que o ator possa visualizar as pontuações dos jogadores das partidas atuais geradas pelo próprio ator.
Atores: Criador.
Pré-condições: O ator deve ter acesso a internet, estar devidamente logado e gerado pelo menos uma partida (que não tenha passado 2 horas, ou seja, válido para entrar).
Pós-condições: N/A.
Cenário Principal:

1. Na tela de dashboard, o sistema apresenta a opção de acompanhamento de partidas atuais.
2. O ator clica nesta opção.
3. Dentro da tela de acompanhamento, mostra as partidas atuais geradas pelo próprio autor como nome do jogo de memória e código da partida no momento.
4. O ator clica em uma das partidas e o sistema mostrará as informações de todos jogadores que terminaram o jogo.
5. O ator pode clicar na opção de “Recarregar” para que possa verificar se mais jogadores terminaram o jogo.
6. O ator pode clicar na opção de voltar para tela anterior que mostra as partidas e repetir o passo 3, ou pode clicar na opção de voltar para tela de dashboard.

Extensão (ou Fluxo Alternativo):

a. O sistema não encontra as partidas atuais ocorrendo no momento:
   1. O sistema exibe a mensagem: "Partidas atuais não encontradas!".

b. O sistema não encontra os jogadores que estão jogando ou terminaram.
   1. O sistema exibe a mensagem: "Jogadores não encontrados!".

c. O sistema tentou carregar as pontuações e o servidor deu erro:
   1. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 14 - Caso de uso de consultar histórico das partidas criadas anteriormente.

Identificador: CSU 010
Nome: Consultar partidas criadas.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que o ator possa visualizar as pontuações dos jogadores das partidas criadas anteriormente pelo próprio ator.
Atores: Criador.
Pré-condições: O ator deve ter acesso a internet, estar devidamente logado e criado pelo menos uma partida anteriormente (que foi passado 2 horas, ou seja, inválido, não permitido entrar).
Pós-condições: N/A.
Cenário Principal:

1. Na tela de dashboard, o sistema apresenta a opção de acompanhamento de partidas criadas anteriormente.
2. O ator clica nesta opção;
3. Dentro da tela de histórico, mostra as partidas anteriores geradas pelo próprio autor como nome do jogo de memória, código da partida, quantidades de jogadores, data e hora que começou a partida e data e hora que último jogador terminou.
4. O ator clica em uma das partidas pelo botão “Detalhes” e o sistema mostrará as informações de todos jogadores que participaram da partida.
5. O ator pode clicar na opção de voltar para tela anterior que mostra as partidas e repete o passo 4.
6. O ator volta para tela de dashboard.

Extensão (ou Fluxo Alternativo):

a. O sistema não encontra as partidas geradas anteriormente:
   2. O sistema exibe a mensagem: "Partidas não encontradas!".

b. O sistema não encontra os jogadores que estão jogando ou terminaram.
   2. O sistema exibe a mensagem: "Jogadores não encontrados!".

c. O sistema tentou carregar as pontuações e o servidor deu erro:
   2. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 15 - Caso de uso de listar jogos de memória jogados anteriormente.

Identificador: CSU 011
Nome: Listar jogos de memória salvos.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que o ator possa visualizar os jogos de memórias jogados salvos.
Atores: Jogador.
Pré-condições: O ator deve ter acesso a internet, estar devidamente logado e jogado pelo menos uma partida anteriormente.
Pós-condições: N/A.
Cenário Principal:

1. Na tela de dashboard, o sistema apresenta a opção de visualizar os jogos de memória jogados anteriormente.
2. O ator clica nesta opção.
3. O sistema carregará os jogos de memórias jogados pelo jogador anteriormente.

Observação: Caso o usuário seja somente do tipo jogador, o sistema já apresenta automaticamente a lista.

Extensão (ou Fluxo Alternativo):
a. O sistema não encontra os jogos de memória jogados pelo ator.

1. O sistema exibe a mensagem: "Não jogou jogo de memória, tente encontrar um código ;)".

b. O sistema tentou carregar a lista e o servidor deu erro:

1. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 16 - Caso de uso de entrar na partida pelo código.

Identificador: CSU 012
Nome: Entrar na partida pelo código.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que atores possam entrar na partida pelo código compartilhado por outro ator.
Atores: Jogador
Pré-condições: O ator deve ter acesso a internet, estar devidamente logado e ter acesso a algum código.
Pós-condições: N/A.
Cenário Principal:

1. O sistema apresenta a opção de entrada para partida pelo código ao ser feito o login no sistema;
2. O ator clica nesta opção;
3. Dentro da tela de entrada para partida pelo código.
4. O ator digita o código.
5. O sistema procura a partida pelo código.
6. Se o sistema achar, o ator aperta no botão “Jogar”.
    6.1. O ator entra para a partida.
7. Se o sistema não achar:
    7.1. O ator aperta no botão “Procurar outra partida”.
        7.1.1. Repete o passo 4.

7.2. O ator aperta no botão “Sair”
    7.2.1. O ator volta para tela de dashboard.

Extensão (ou Fluxo Alternativo):

a. O sistema não encontra a partida pelo código digitado pelo ator.
1. O sistema exibe a mensagem de erro: "Não foi criado um gameplay ou código foi expirado!".
2. O ator deve digitar outro código para tentar encontrar outra partida ou voltar para tela de dashboard.

b. O sistema tenta pesquisar a partida e o servidor deu erro:
1. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 17 - Caso de uso de jogar jogo de memória durante a partida.

Identificador: CSU 013
Nome: Jogar jogo de memória durante a partida.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que atores possam jogar os jogos de memória na partida, na qual foi criado pelo criador pelo compartilhamento do código.
Atores: Jogador.
Pré-condições: O ator deve ter acesso a internet, estar devidamente logado e ter entrado em alguma partida.
Pós-condições: N/A
Cenário Principal:

1. Na tela de dashboard, o sistema apresenta as cartas depois que o ator entra para uma partida.
2. O ator clica em duas cartas.
3. Aparece a tela destas duas cartas e as opções “Está certo!” e “Está errado!”;
4. O ator escolhe a opção que acha estar certa.
5. As cartas vão ser desviradas caso as cartas estejam erradas, senão elas vão continuar viradas para cima. E a pontuação vai ser calculada de acordo com a regra de negócio RN003 (consultar na tabela de regras de negócio).
6. Repete o passo 2 até o ator acertar todas as cartas.
7. Mostra o resultado do usuário como pontuação, quantidade de tentativas, quantidade de opções certas e quantidade de opções erradas.
8. Clica no botão “Terminar” e vai para a tela de pontuações onde é apresentado os resultados do próprio autor e de outros atores que jogaram a mesma partida. Os resultados apresentados vão conter nome do jogo de memória, pontuação, nome do jogador, quantidade de tentativas, quantidade de opções certas e quantidade de opções erradas, data e hora que determinado usuário começou e terminou a partida.
9. O ator aperta no botão “Voltar para dashboard”.
10. O ator volta para tela de dashboard.

Extensão (ou Fluxo Alternativo):
a. O sistema tenta carregar o jogo de memória e o servidor deu erro:

1. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 18 - Caso de uso de jogar o jogo de memória jogados anteriormente novamente.

Identificador: CSU 014
Nome: Jogar os jogos de memória jogados anteriormente novamente, porém individual.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que atores possam jogar os jogos de memória na partida.
Atores: Jogador.
Pré-condições: O ator deve ter acesso a internet, estar devidamente logado e ter jogado em pelo menos uma partida anteriormente.
Pós-condições: N/A
Cenário Principal:

1. Na tela de dashboard, o sistema apresenta a opção de jogar jogo de memória em cada jogo de memória jogado anteriormente.
2. O ator clica nesta opção.
3. Dentro da tela de partida de jogo de memória que apresenta as cartas, o ator pode jogar novamente o jogo de memória;
4. O ator clica em duas cartas.
5. Aparece a tela destas duas cartas e as opções “Está certo!” e “Está errado!”;
6. O ator escolhe a opção que acha estar certa.
7. As cartas vão ser desviradas caso as cartas estejam erradas, senão elas vão continuar viradas para cima. E a pontuação vai ser calculada de acordo com a regra de negócio RN003 (consultar na tabela de regras de negócio).
8. Repete o passo 4 até o ator acertar todas as cartas.
9. Mostra o resultado do usuário como pontuação, quantidade de tentativas, opções certas e opções erradas.
10. Clica no botão “Terminar” e vai para a tela de pontuações onde é apresentado o resultado do próprio autor. O resultado apresentado vai conter nome do jogo de memória, pontuação, nome do criador do jogo de memória, quantidade de tentativas, quantidade de opções certas e quantidade de opções erradas, data e hora que o próprio autor começou e terminou a partida.
11. O ator aperta no botão “Voltar para dashboard”
12. O ator volta para tela de dashboard.

Extensão (ou Fluxo Alternativo):
a. O sistema tenta carregar o jogo de memória e o servidor deu erro:

1. O sistema exibe a mensagem de erro "Deu algo errado! Favor contatar com administrador!".

Fonte: Autoria própria.

Tabela 19 - Caso de uso de consultar histórico das partidas jogadas anteriormente.

Identificador: CSU 015
Nome: Consultar partidas jogadas.
Responsável: Guilherme Henrique Mendonça Nascente.
Descrição/Resumo: Possibilita que atores possam visualizar o histórico das partidas jogadas anteriormente.
Atores: Jogador.
Pré-condições: O ator deve ter acesso a internet, estar devidamente logado e jogado algumas partidas anteriores.
Pós-condições: N/A
Cenário Principal:

1. Na tela de dashboard, o sistema apresenta a opção de histórico das partidas anteriores.
2. O ator clica nesta opção.
3. Dentro da tela de histórico, mostra os resultados das partidas anteriores. O resultado apresentado vai conter nome do jogo de memória, pontuação, nome do criador do jogo de memória, quantidade de tentativas, quantidade de opções certas e quantidade de opções erradas, data e hora que o próprio autor começou e terminou a partida.
4. O ator aperta no botão “Voltar para dashboard”
5. O ator volta para tela de dashboard.

Extensão (ou Fluxo Alternativo):

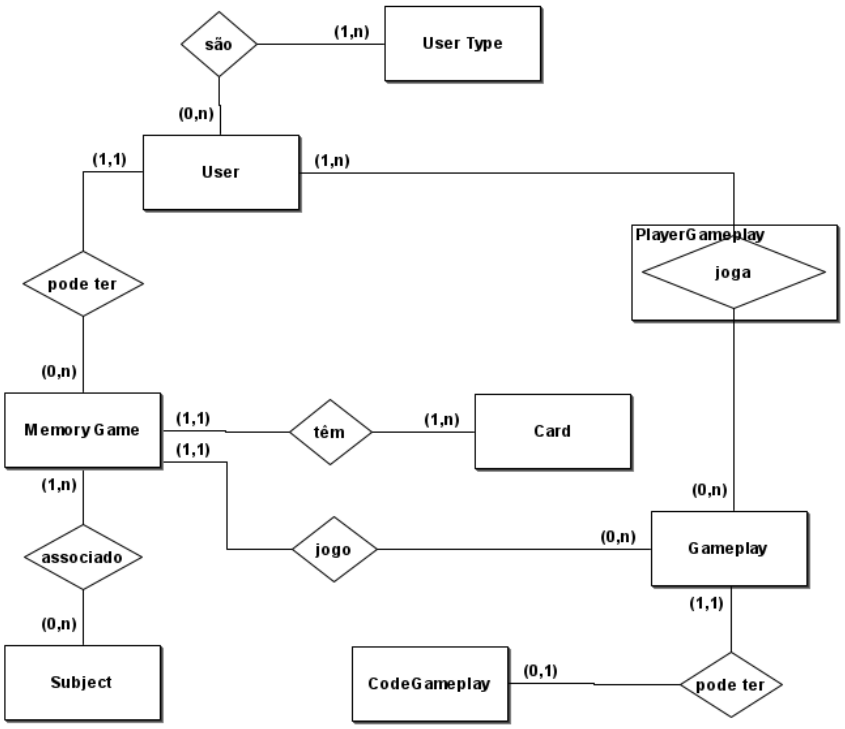
a. O sistema não encontra as partidas anteriores:
   1. O sistema exibe a mensagem: "Não foram encontradas partidas anteriores.".

b. O sistema tenta carregar o histórico das partidas anteriores e o servidor deu erro:
   1. O sistema exibe a mensagem de erro "Tem algo errado com servidor, contate com administrador.".

Fonte: Autoria própria.

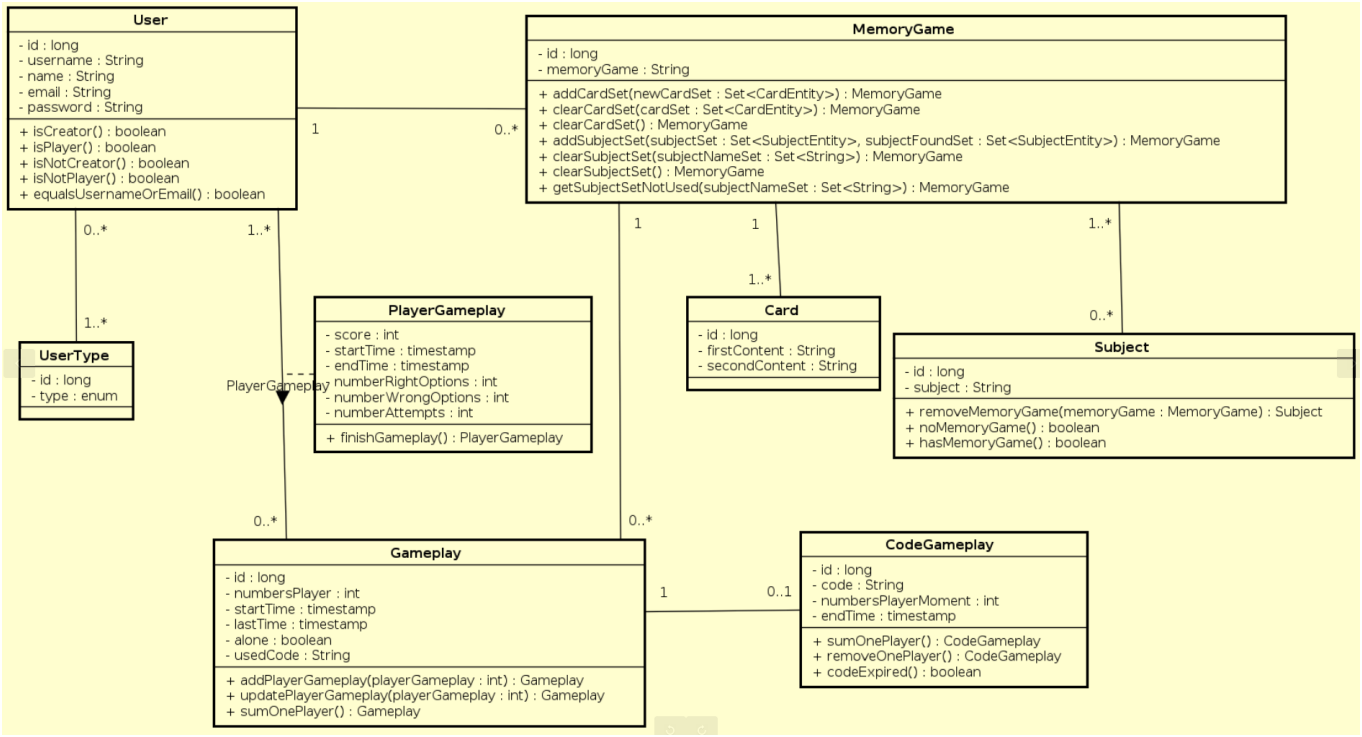
6. MODELO ENTIDADE-RELACIONAMENTO E DIAGRAMA DE CLASSE

Figura 31 - Modelo de Entidade-Relacionamento do projeto.



Fonte: Autoria própria.

Figura 32 - Diagrama de classe do projeto.



Fonte: Autoria própria.

Descrição dos atributos apresentados no diagrama de classe da figura 32:

Tabela 20 - Descrição dos atributos da classe de User (Usuário).

Atributo	**Tipo**	**Descrição**
id	Inteiro longo	Identificador único do usuário.
username	Texto	Nome de usuário. No mínimo 6 caracteres.
name	Texto	Nome do usuário. No mínimo 6 caracteres.
email	Texto	E-mail do usuário.
password	Texto	Senha do usuário (criptografado). No mínimo 6 caracteres na senha normal.

Fonte: Autoria própria.

Tabela 21 - Descrição dos atributos da classe de UserType (Tipo de usuário).

Atributo	**Tipo**	**Descrição**
id	Inteiro longo	Identificador único do tipo de usuário.
type	Enum	Tipo de usuário.

Fonte: Autoria própria.

Tabela 22 - Descrição dos atributos da classe de MemoryGame (Jogo de memória).

Atributo	**Tipo**	**Descrição**
id	Inteiro longo	Identificador único do jogo de memória.
memoryGame	Texto	Nome do jogo de memória. No mínimo 3 caracteres.

Fonte: Autoria própria.

Tabela 23 - Descrição dos atributos da classe de Card (Carta).

Atributo	**Tipo**	**Descrição**
id	Inteiro longo	Identificador único do jogo de memória.
firstContent	Texto	Primeiro conteúdo da carta. No mínimo 1 caracter.
secondContent	Texto	Segundo conteúdo da carta. No mínimo 1 caracter.

Fonte: Autoria própria.

Observação: A classe carta da tabela 23 armazena a informação de um par de cartas que são associadas. Primeiro conteúdo seria uma carta e segundo conteúdo seria outra carta, mas ambos associados.

Tabela 24 - Descrição dos atributos da classe de Subject (Matéria).

Atributo	**Tipo**	**Descrição**
id	Inteiro longo	Identificador único da matéria.
subject	Texto	Nome da matéria.

Fonte: Autoria própria.

Tabela 25 - Descrição dos atributos da classe de Gameplay (Partida).

Atributo	**Tipo**	**Descrição**
id	Inteiro longo	Identificador único da matéria.
usedCode	Texto	Código usado para partida.
alone	Booleano	Para identificar se o jogador está jogando individualmente ou não. Se for falso, então está jogando em uma partida gerada pelo um criador.
startTime	Timestamp	Data e hora que começou a partida.
lastTime	Timestamp	Data e hora do último jogador que terminou a partida.

Fonte: Autoria própria.

Tabela 26 - Descrição dos atributos da classe de PlayerGameplay (Relação do Jogador com Partida).

Atributo	**Tipo**	**Descrição**
playerId	Inteiro longo	Identificador único do usuário que tenha tipo de jogador.
gameplayId	Inteiro longo	Identificador único da partida.
score	Inteiro	Pontuação da partida
numberRightOptions	Inteiro	Quantidade de opções certas escolhidas pelo usuário.
numberWrongOptions	Inteiro	Quantidade de opções erradas escolhidas pelo usuário.
numberAttempts	Inteiro	Quantidade de tentativas que o usuário selecionou o par de cartas.
startTime	Timestamp	Data e hora que usuário começou a partida.
endTime	Timestamp	Data e hora que usuário terminou a partida.

Fonte: Autoria própria.

Tabela 27 - Descrição dos atributos da classe de CodeGameplay (Código).

Atributo	**Tipo**	**Descrição**
id	Inteiro longo	Identificador único da matéria.
code	Texto	Código gerado para uma partida.
numbersPlayer Moment	Inteiro	Quantidade de usuário jogando no momento da partida.
endTime	Timestamp	Data e hora que código vai ser expirado.

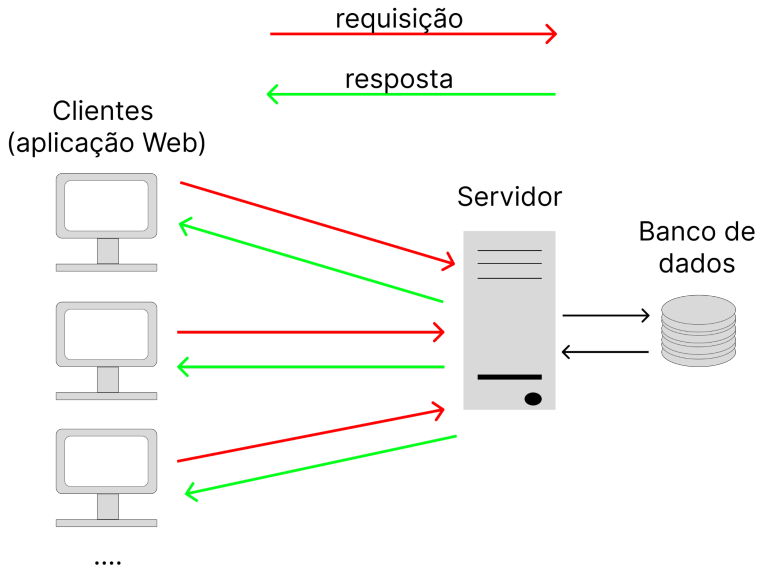
Fonte: Autoria própria.

7. ARQUITETURA

7.1. Projeto

A arquitetura usada no projeto como um todo foi arquitetura cliente-servidor. Como mostra a figura 33, a arquitetura cliente-servidor se divide em dois grupos: servidores, que fornecem serviços ou recursos, e clientes, que solicitam estes serviços e recursos, recebendo-os como respostas. A aplicação Front-end feita em Flutter é cliente, já a aplicação Back-end é servidor. (CONTROLE NET, 2023)

Figura 33 - Arquitetura cliente e servidor.



Fonte: Autoria própria.

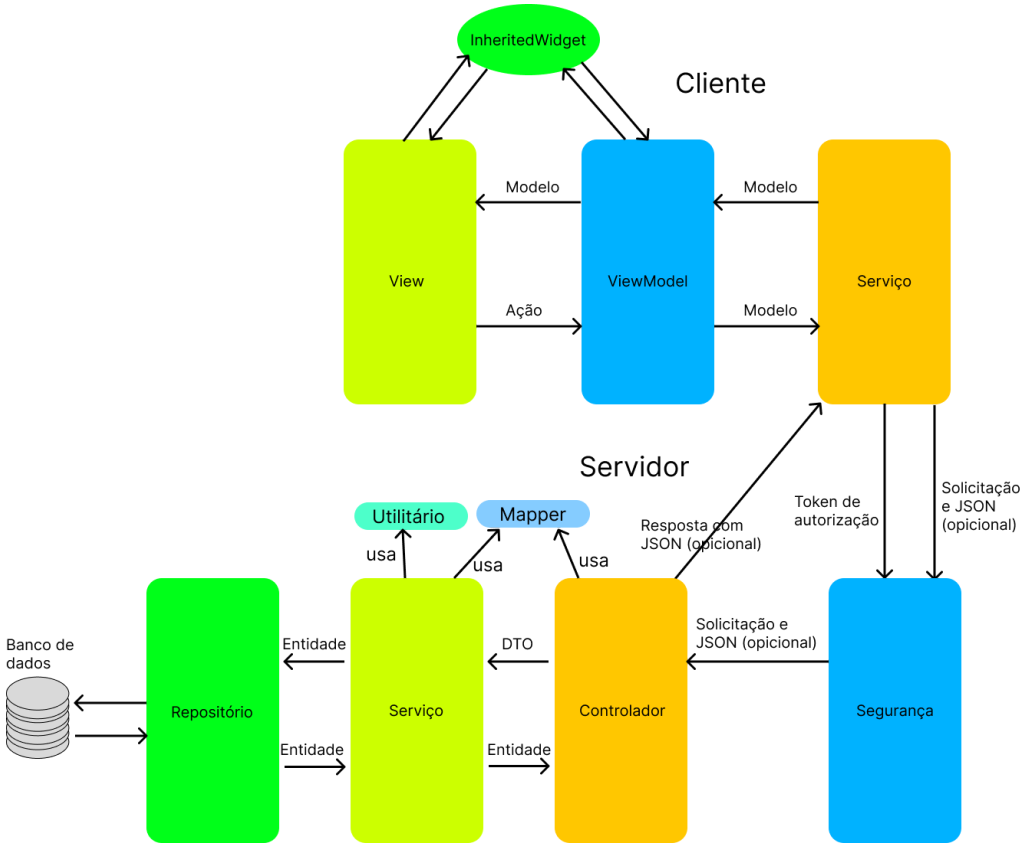
Mas além da arquitetura cliente-servidor, também foi usada duas características importantes da API REST:

- A comunicação entre cliente e servidor é stateless (sem estado). Ou seja, nenhuma informação do cliente é armazenada entre as solicitações e as solicitações são separadas e desconectadas. A única forma de ter acesso ao servidor é pelo token JWT, toda solicitação com exceção de solicitação de logon, cadastro de conta e alteração de senha, deve ser mandado com token incluído, desde que seja válido. (RED HAT, 2020)
- Servidor expõe os serviços que podem ser consumidos pelos clientes, e os clientes devem consumir através da requisição HTTP. As requisições enviadas pelo cliente e as respostas pelo servidor podem estar no formatos XML, JSON,

Texto, Imagens e entre outros tipos de formatos. Neste projeto, foi usado apenas o formato JSON. (RED HAT, 2020)

A figura 34 mostra arquitetura usada no desenvolvimento do projeto.

Figura 34 - Arquitetura do projeto todo.



Fonte: Autoria própria.

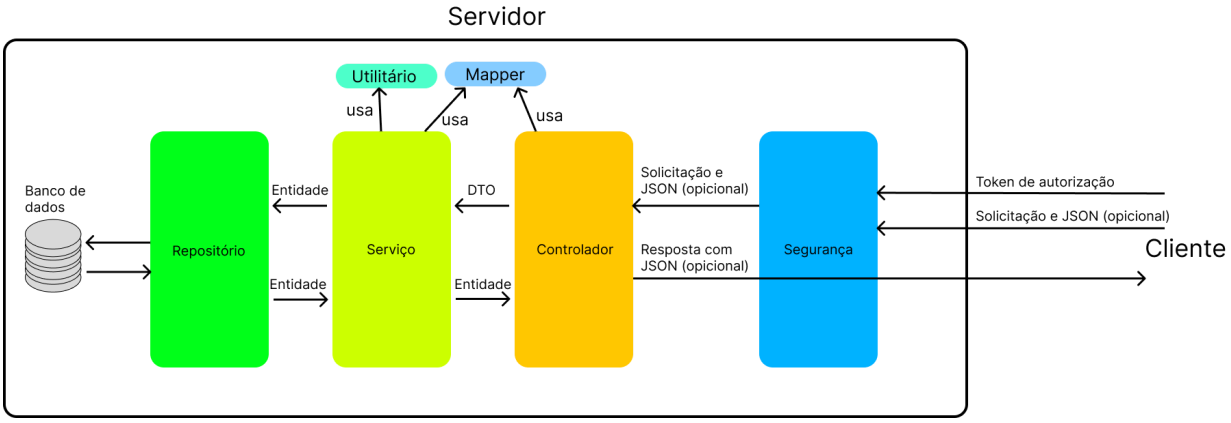
7.2. Backend

O backend usa arquitetura em camada. Cada camada deve ter uma determinada responsabilidade diferente de uma da outra e se comunicar entre elas.

> "Arquitetura em camadas é um dos padrões arquiteturais mais usados, desde que os primeiros sistemas de software de maior porte foram construídos nas décadas de 60 e 70. Em sistemas que seguem esse padrão, as classes são organizadas em módulos de maior tamanho, chamados de camadas. As camadas são dispostas de forma hierárquica, como em um bolo. Assim, uma camada somente pode usar serviços — isto é, chamar métodos, instanciar objetos, estender classes, declarar parâmetros, lançar exceções, etc. — da camada imediatamente inferior." (VALENTE, 2020, cap. 7, seção 7.2)

Como a figura 35 mostra: as camadas estão divididas em Segurança, Controlador, Serviço e Repositório e alguns dados que são transferidos entre eles:

Figura 35 - Arquitetura em camadas de Segurança, Controlador, Serviço e Repositório.



Fonte: Autoria própria.

- Camada de segurança: é responsável por fazer a validação do token de autorização e permitir acesso do usuário caso o token de autorização seja válido, senão é negado o acesso. O token usado é JWT (JSON Web Token) da Auth0, que permite transferir informações com segurança entre as partes e o token é verificável e confiável pois é assinado digitalmente. (AUTH0, 2023)
- Camada de controlador (controller): é responsável por expor os serviços para que ela possa ser consumida por clientes e lidar com solicitações HTTP e entregar dados em formato de JSON recebido da camada de serviço. Junto com a solicitação HTTP pode vir os dados de entradas em formato JSON que vão se converter para DTO e esta camada deve passar o DTO para a camada de serviço responsável. (COLLINGS, 2021)
- Camada de Repositório (repository): é responsável pela comunicação e operação com o banco de dados. As camadas de serviço devem usar camada de repositório caso queira lidar com banco de dados. (COLLINGS, 2021)
- Camada de Serviço (service): é responsável pelas regras de negócios que vai salvar ou modificar os dados de acordo com dados de entradas passados pelo cliente através do DTO. Para salvar ou recuperar dados armazenados no banco de dados, ele faz através da camada de repositório. Também é responsável por enviar resposta de dados de acordo com a solicitação do Cliente e deve ser enviado para camada de controlador para converter para DTO e enviar para Cliente. (COLLINGS, 2021)
- DTO: é um objeto que serve como uma forma de transferência de dados entre as camadas de uma aplicação ou entre sistemas diferentes. Ele define apenas os dados necessários para a transferência e também permite misturar dados de diferentes

entidades. Como por exemplo, para passar as informações de um determinado usuário, deve passar somente as informações necessárias, como nome completo, email e nome de usuário e não deve ser passado as informações privadas como por exemplo, a senha. E também podem ser passados os jogos de memória que ele contém e o jogo de memória é uma entidade diferente da entidade de usuário. (LUBOWA, 2021; ANDERSON et al., 2022)

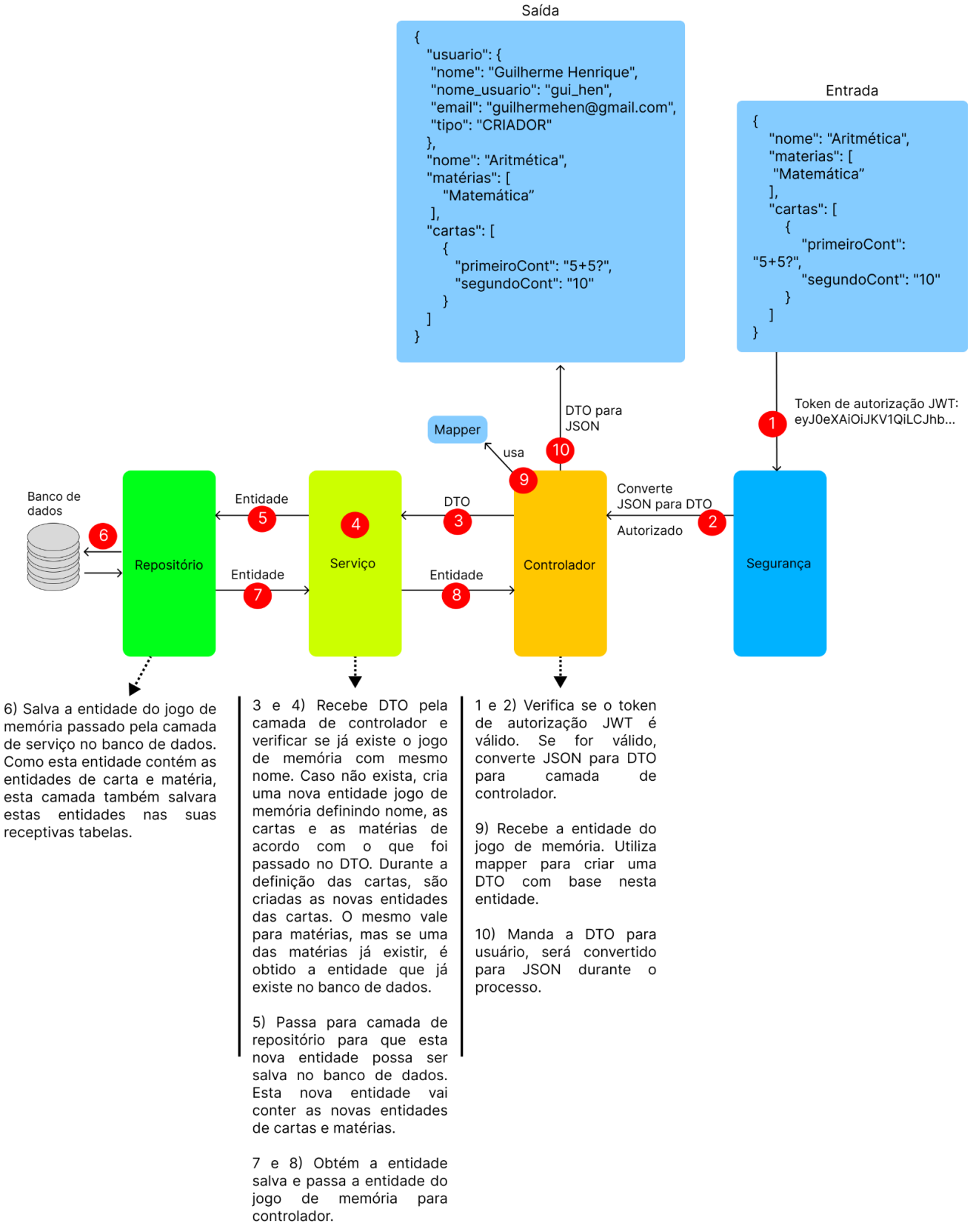
- Entidade: é um objeto de domínio utilizado para representar uma tabela do banco de dados, sendo que cada instância da entidade corresponde a uma linha da tabela. Como por exemplo a entidade de jogo de memória representa a tabela do Jogo de Memória e a instância representa uma linha desta tabela, que conterá identificador e nome de jogo de memória. A instância também permite acessar outras entidades por meio das relações que têm com as outras tabelas, seja um-para-um, um-para-muitos ou muitos-para-muitos, como por exemplo, uma instância do Jogo de Memória permite acessar as cartas associado com jogo de memória. (ORACLE, 2009)

Além das camadas, na arquitetura também estão presentes dois utilitários, que são utilitário (util) e mapper.

- Utilitário: é responsável por oferecer funções utilitários para facilitar algumas tarefas que não fariam sentido estar dentro do Serviço, Controlador ou Repositório. Como por exemplo, a função de gerar código de uma determinada partida para um jogo de memória faria mais sentido utilizar numa classes utilitárias, pois o serviço não precisa conhecer a implementação, só necessita de um código novo para criar uma partida e salvar no banco de dados.
- Mapper: é responsável por fazer mapeamento entre classes diferentes dos objetos e permitir que possa criar uma DTO com base nos dados da entidade ou vice-versa. Também não faria sentido estar dentro do Serviço, Controlador ou Repositório. Como por exemplo, a conversão da entidade de usuário para DTO do usuário, contendo somente nome, nome de usuário e email. (MAPSTRUCT, 2022)

A figura 36 mostra um exemplo de um jogo de memória chamado “Aritmética” criado pelo usuário “gui_hen” a ser adicionado no sistema.

Figura 36 - Exemplo de jogo de memória que vai ser adicionado no sistema.



Fonte: Autoria própria.

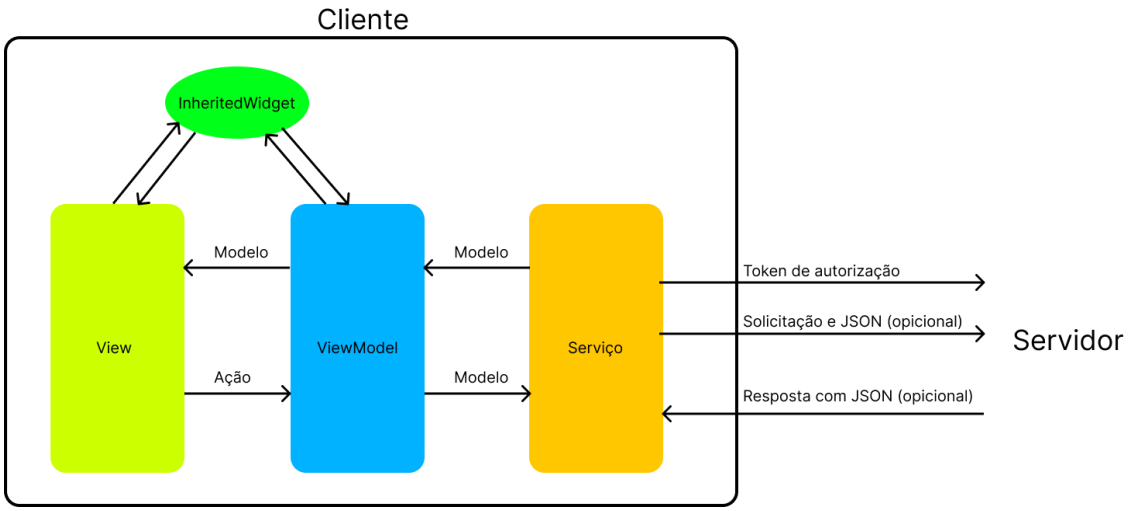
7.3. Frontend

O frontend usa uma arquitetura adaptada baseada na arquitetura chamada MVVM. MVVM significa *Model*, *View e ViewModel* e são elas:

- *Model* (modelo): responsável por encapsular os dados entregues por alguma fonte externa, como por exemplo, servidor. Esta camada pode conter lógica de negócios, validação de código, etc. A camada *ViewModel* usa a *Model* para exibir as informações na View e também mandar para o servidor, quando convertido para formato JSON. (STONIS et al., 2022; MOHITE, 2020)
- *View*: responsável pela construção de interface do usuário. Quando o usuário interage com a interface e aciona alguma ação, faz com que *ViewModel* realize alguma tarefa e se comunique com *Model*. Assim que o *ViewModel* terminar a tarefa e retornar dados necessários, ele atualizará o *View*. (STONIS et al., 2022; MOHITE, 2020)
- *ViewModel*: é o intermediador entre *View* e *Model*, que aceita todos os eventos do usuário e solicita dados ao *Model*. Depois que o *Model* tiver dados, ele retornará ao *ViewModel* e, em seguida, o *ViewModel* notificará esses dados para o *View*. *ViewModel* não está restrito a somente uma view, pode ser usado por várias *Views*. (STONIS et al., 2022; MOHITE, 2020)

O que foi adaptado nesta arquitetura, como mostra a figura 37, é a adição da camada de serviço (*Service*), que é responsável por buscar dados pelo servidor, podendo também mandar dados para o servidor salvar no sistema e também representa parte do conceito do *Model* apresentado anteriormente. *Model* nesta arquitetura adaptada, vai ser responsável somente carregar os dados para ser transferido para outras camadas. Ela vai ser responsável por transformar dados recebidos em *Model* para que possa ser usado pela camada *ViewModel* para preparar outros modelos para ser exibido pela *View*.

Figura 37 - Arquitetura MVVM adaptada.



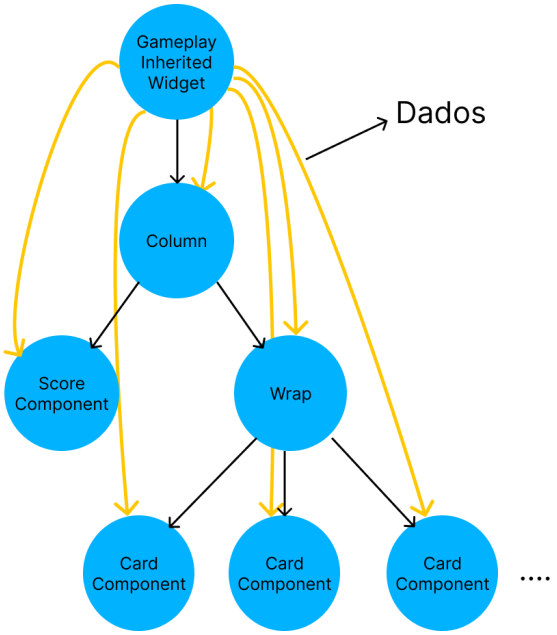
Fonte: Autoria própria.

Outra adaptação para a arquitetura é o componente importante para a arquitetura, é o recurso de *InheritedWidget*, que Flutter oferece. Ele é responsável por auxiliar a compartilhar e atualizar os dados para diversos componentes na árvore. Flutter utiliza a árvore como base para construção dos componentes da tela. (API FLUTTER DEV, 2023b)

A figura 38 mostra um exemplo dos componentes da tela de partida. Muitas informações precisam ser administradas durante a partida, como por exemplo, é necessário saber se um usuário clicou nas duas cartas, calcular a pontuação quando usuário seleciona uma das opções "Está certo?" e "Está errado?" e atualizar a pontuação mostrada na tela da partida. Para obter e atualizar essas informações, foi usado uma classe estendida da classe *InheritedWidget* do Flutter, chamada de *GameplayInheritedWidget*. Como explicado anteriormente, ele permite compartilhar as informações entre os componentes.

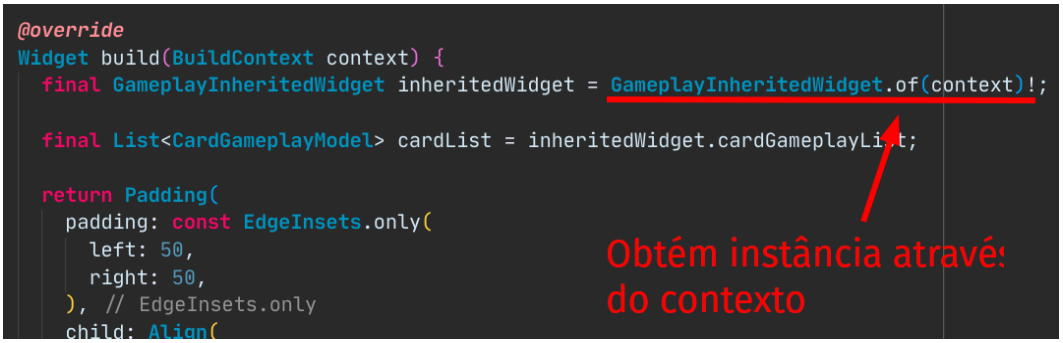
A vantagem deste tipo de abordagem é que não é necessário passar as informações pelos parâmetros em cada componente. Para ter acesso e atualizar as informações pela classe *InheritedWidget*, é preciso passar o argumento chamado contexto do tipo *BuildContext* e com este contexto, é possível localizar a instância da classe *InheritedWidget* na árvore e ter acesso às informações, como mostra a figura 39.

Figura 38 - Exemplo de InheritedWidget.



Fonte: Autoria própria.

Figura 39 - Instância do GameplayInheritedWidget através do contexto.



Fonte: Autoria própria.

Outro ponto importante do Flutter é o *Future*, representa o resultado de uma operação assíncrona e pode ter dois estados: incompleto ou concluído. Quando um *Future* está no estado incompleto, significa que ainda não foi recebido dados ou algum tipo de erro, já no estado completo, foi recebido dados se operação ocorreu com sucesso ou ocorreu algum erro. Um dos usos para *Future* é a obtenção de dados pela rede, já que pode ter algum tempo de demora e a aplicação precisa continuar realizando o trabalho enquanto não recebe os dados. (DART DEV, 2023b)

Este recurso é usado para obter a lista de jogo de memória, históricos das partidas anteriores e as cartas para criar as partidas. É também usado para mandar alguma requisição com JSON para servidor, já que o servidor pode mandar alguma resposta para a aplicação. A camada *View* estará usando um componente *FutureBuilder*, um componente que Flutter traz

de forma nativa, porém adaptado para facilitar a manipulação do *Future*, chamado *CustomFutureBuilder*. (API FLUTTER DEV, 2023a)

Como mostra a figura 40, deve adicionar: um *Future* para receber dados vindo do servidor*;* um componente visual no parâmetro *onLoading* para quando dados não foram chegados ainda; um outro componente visual no parâmetro *onData* para quando dados são carregados com sucesso. Mas além disso, também está presente o parâmetro *onError*, que deve ser definido um componente visual para quando os dados não chegaram com sucesso.

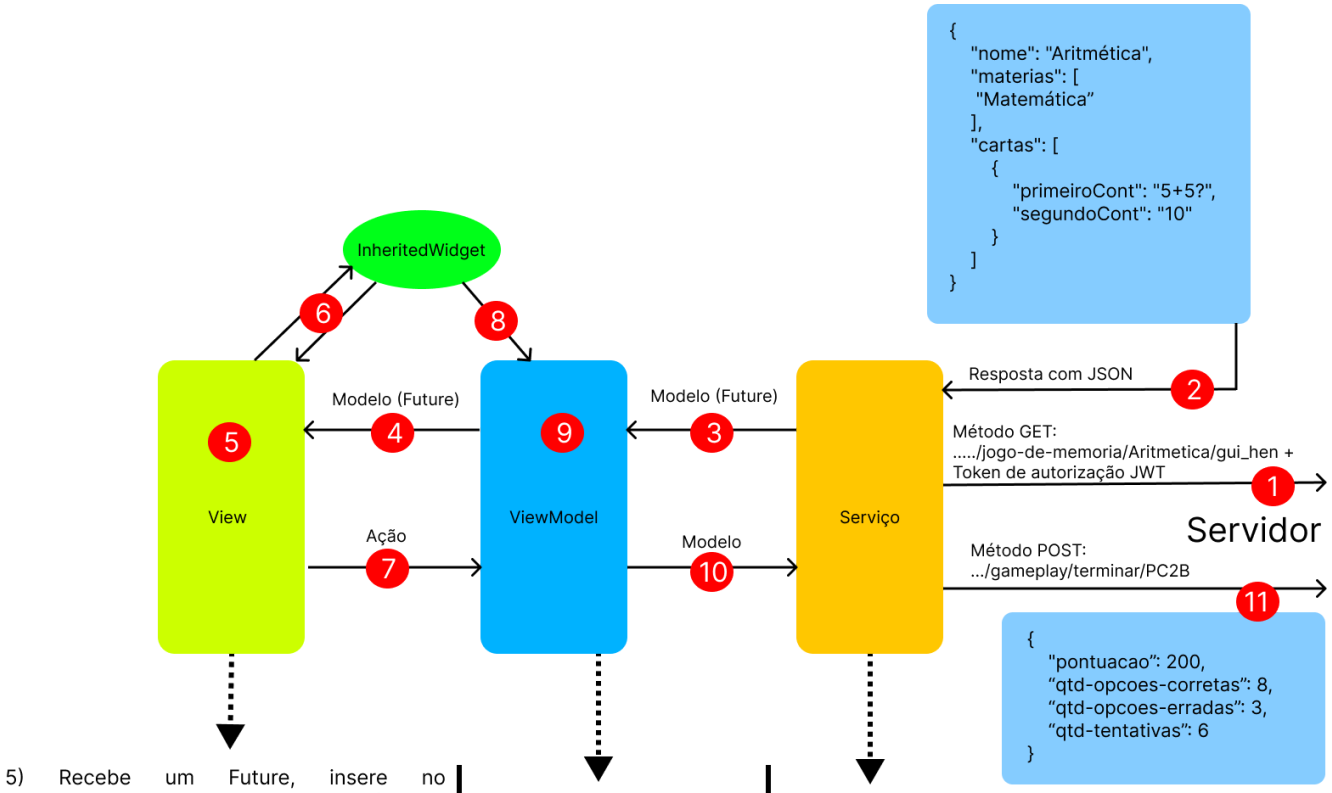
Figura 40 - Exemplo de componente *CustomFutureBuilder*.

```
CustomFutureBuilderWidget<PlayerAddedModel, PlayerAddedModel>(
  future: viewModel.futurePlayerAddedModel,
  onLoading: (context) => const CircularProgressIndicator(),
  onData: (context, value) => Column(
    children: [
      SelectableText(
        value.memoryGame.name,
        style: Theme.of(context).textTheme.headlineLarge,
      ), // SelectableText
      SelectableText(
        'Do criador: ${value.memoryGame.creator}',
        style: Theme.of(context).textTheme.headlineLarge,
      ), // SelectableText
      ElevatedButton(
        onPressed: viewModel.onPressedEnterInGameplay(value),
        child: const Text('Jogar'),
      ), // ElevatedButton
    ],
  ), // Column
```

Fonte: Autoria própria.

A figura 41 mostra um exemplo de obtenção de jogo de memória chamada “Aritmética”, criado pelo usuário “gui_hen”, para carregar as cartas para a partida e por fim mandar o resultado do usuário na partida para o servidor.

Figura 41 - Exemplo de obtenção do jogo de memória.



5) Recebe um Future, insere no CustomFutureBuilder e carrega a interface de acordo com estado do Future (carregamento, dados recebidos ou erro). Se os dados foram recebidos com sucesso, deve carregar as cartas do jogo de memória para usuário começar jogar.

6)
1°) Quando usuário clicar em uma carta, é armazenado esta primeira carta no InheritedWidget.
2°) Quando é clicado outra carta, é armazenado a segunda carta no InheritedWidget.
3°) Com isso é feito a alteração na View pelo InheritedWidget para aparecer as duas cartas e as opções de "Está certo!" e "Está errado!".
4°) De acordo com a opção escolhido, as cartas podem ser viradas novamente ou não e é atualizado a pontuação do jogo, quantidade de opções certas, opções erradas e tentativas na tela.
5°) Repete o 1° passo até ser completado todas as cartas.

7) Quando usuário completa todas as cartas, e clica no botão terminar, aciona a ação que ViewModel deve executar.

4) Obtém um Future que contém o modelo do jogo de memória (inclusive cartas) e repassa para camada de View.

8) Com isso, ViewModel obtém pontuação do jogo, quantidade de opções certas, opções erradas e tentativas do usuário pelo InheritedWidget.

9) Obtém as informações do resultado do usuário e cria um modelo (model) com base nestas informações.

10) Manda o modelo para camada de serviço.

1 e 2) Solicita com método GET o jogo de memória com nome de "Aritmética" criado pelo usuário gui_hen para servidor e receber a resposta no formato JSON.

3) Cria um modelo (model) com base na resposta de JSON recebida e passa para camada ViewModel. Como a resposta pode demorar vindo do servidor, deve encapsular em um objeto da classe Future.

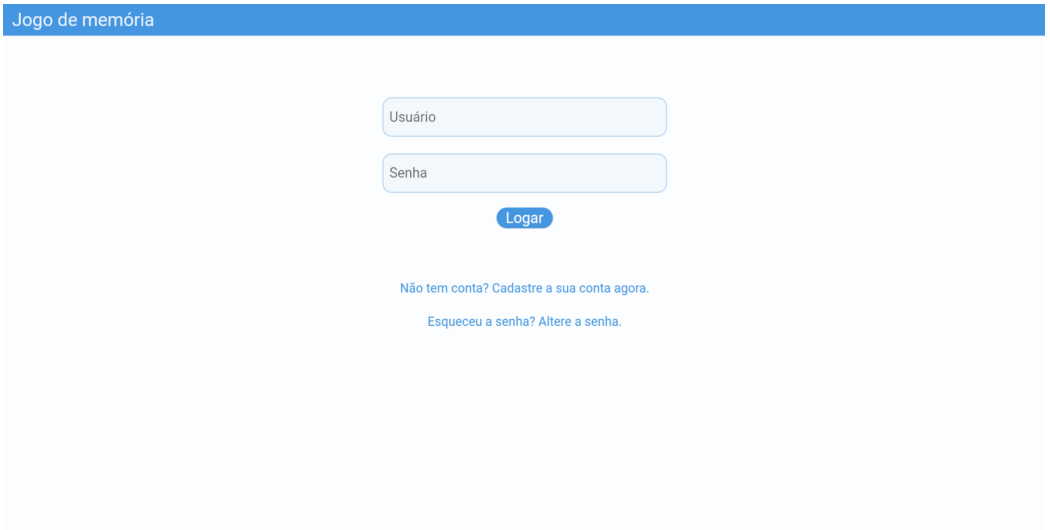
11) Obtém o modelo que contém a respeito do resultado do usuário e manda para servidor no formato JSON para que este resultado seja salvo no sistema.

Fonte: Autoria própria.

8. PROTÓTIPO

Neste capítulo será apresentado o protótipo do projeto. A figura 42 mostra o login do usuário com os campos de usuário e senha para que usuário possa logar no sistema. A figura 43 mostra cadastro de conta para usuário criar uma nova conta para entrar no sistema inserindo os dados nos campos nome, usuário, email, senha e as opções de tipo de jogador e criador que usuário deseja ser. A figura 44 mostra a tela de alteração de senha para o usuário alterar a senha caso esqueça da senha anterior. Para mudar a senha, deve ser inserido seu nome de usuário, senha nova e confirmação de senha nos campos usuário, senha e confirmação de senha.

Figura 42 - Tela de login.



Fonte: Autoria própria.

Figura 43 - Cadastro de conta.

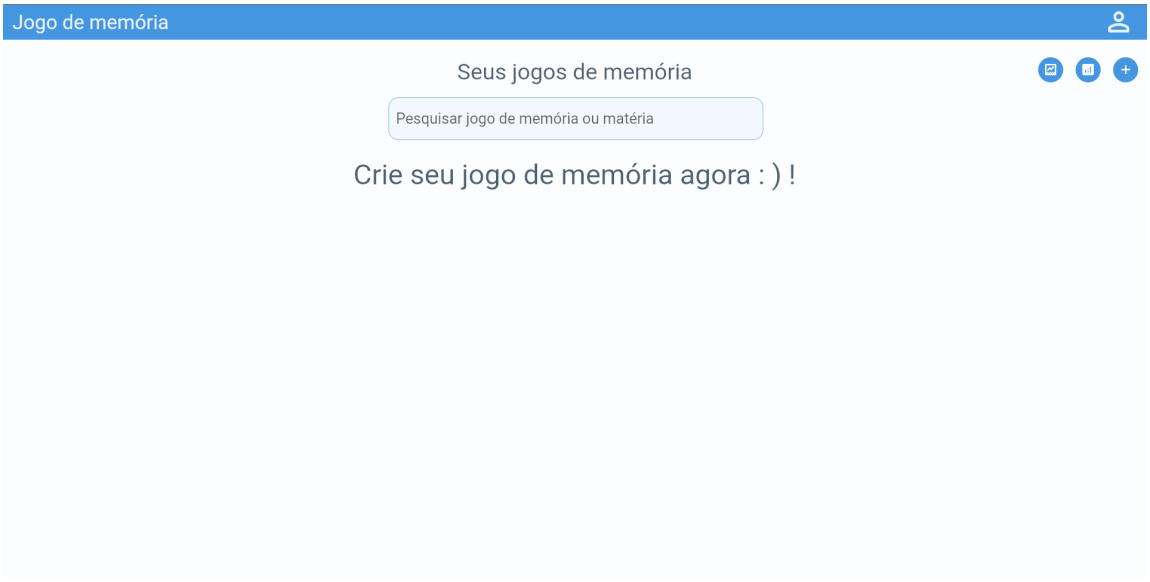
Fonte: Autoria própria.

Figura 44 - Alteração de senha.

Fonte: Autoria própria.

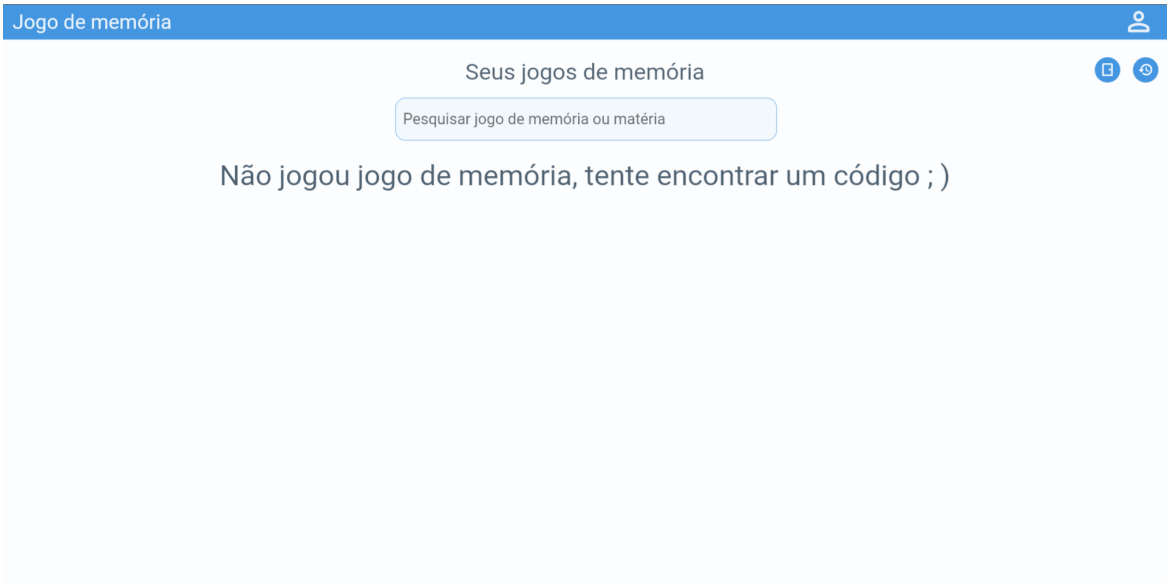
Quando o usuário criado recentemente é logado, é mostrado a mensagem “Crie seu jogo de memória agora :)!” se usuário for do tipo de criador, ou “Não jogou jogo de memória, tente encontrar um código ;)” se usuário for do jogador como mostra as figuras 45 e 46.

Figura 45 - Tela de dashboard com mensagem inicial para criador.



Fonte: Autoria própria.

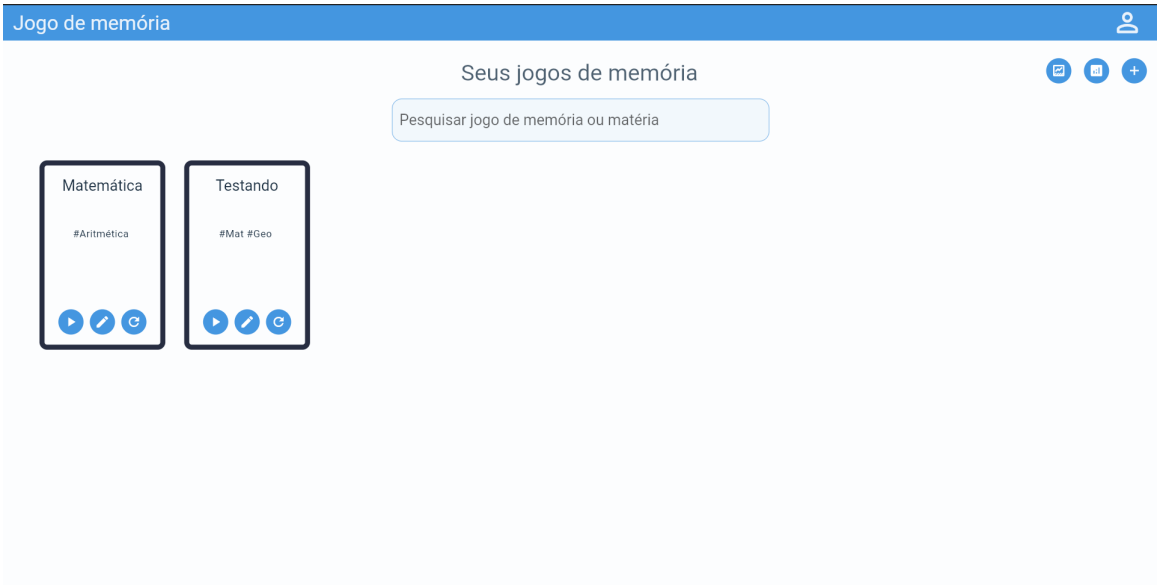
Figura 46 - Tela de dashboard com mensagem inicial para jogador.



Fonte: Autoria própria.

Caso o usuário já tenha jogos de memória salvo no sistema, como mostra a figura 47, serão mostrados os jogos disponíveis na tela de dashboard. Se usuário for do tipo criador, será mostrado os jogos de memória criados anteriormente, mas se for do tipo jogador, será mostrado os jogos de memória jogados anteriormente.

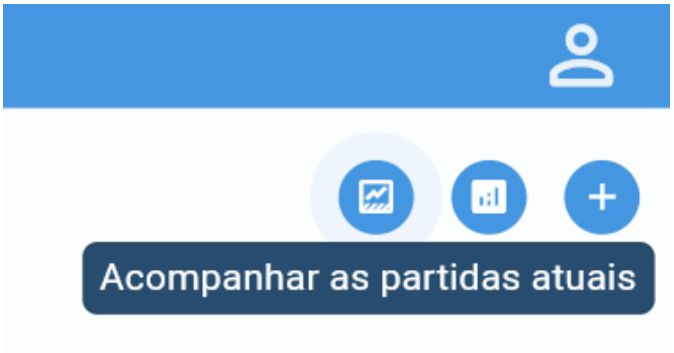
Figura 47 - Tela de dashboard com jogos de memória.



Fonte: Autoria própria.

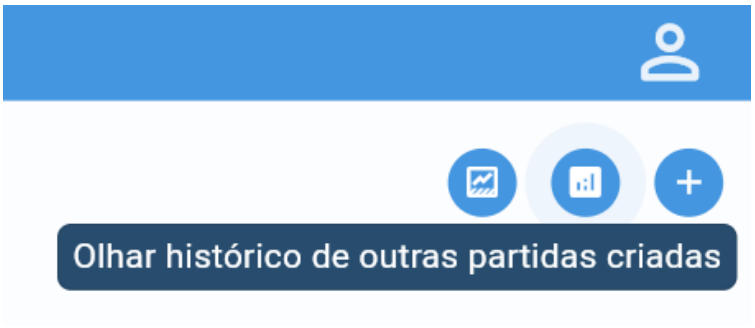
Caso usuário tenha somente o tipo de criador, aparecerá três opções no canto superior direito, mostrados nas figuras 48, 49 e 50. São eles: “Acompanhar as partidas atuais”, “Olhar histórico de outras partidas criadas” e “Criar um jogo de memória”.

Figura 48 - Opção de acompanhar as partidas atuais.



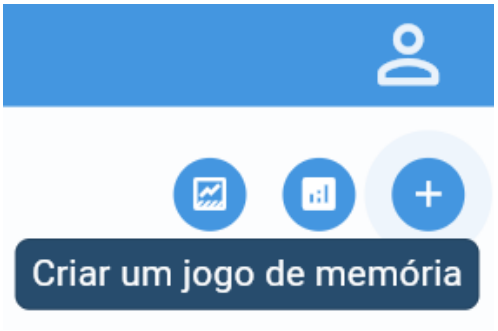
Fonte: Autoria própria.

Figura 49 - Opção de olhar o histórico de outras partidas criadas.



Fonte: Autoria própria.

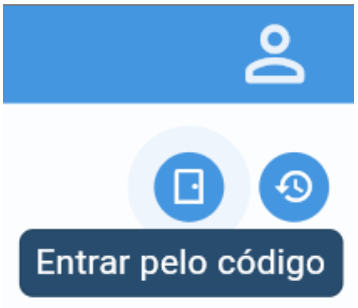
Figura 50 - Opção de criar um jogo de memória.



Fonte: Autoria própria.

Caso usuário tenha somente tipo de jogador, aparecerá duas opções no canto superior direito, mostrados nas figuras 51 e 52. São eles: “Entrar pelo código” para entrar em uma partida e “Olhar histórico de outras jogadas” para mostrar histórico das partidas jogados anteriormente.

Figura 51 - Opção de entrar pelo código.



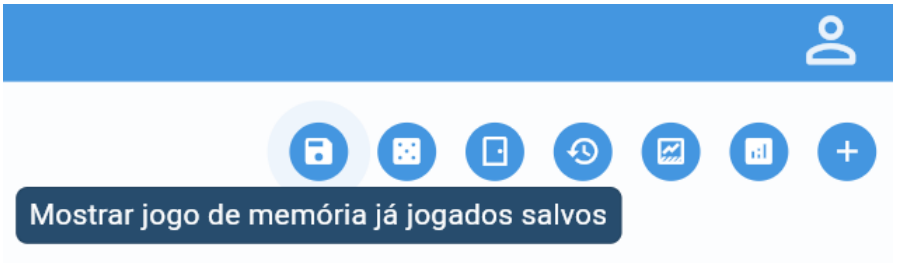
Fonte: Autoria própria.

Figura 52 - Opção de olhar histórico de outras jogadas.

Fonte: Autoria própria.

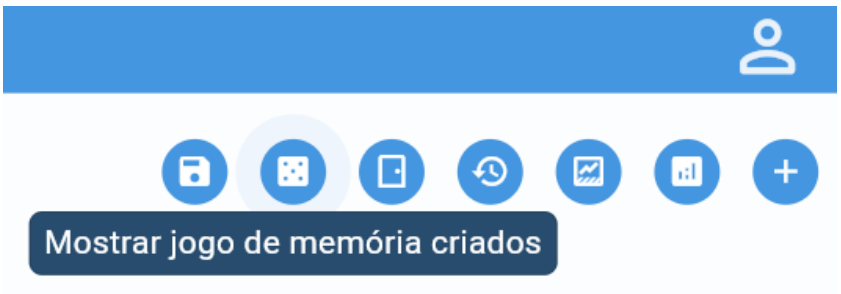
Caso usuário tenha tanto o tipo criador e jogador, aparecerá todas as opções apresentadas anteriormente e duas opções a mais, que são “Mostrar jogos de memória já jogados salvos” e “Mostrar jogos de memória criados”. Isto é mostrado nas figuras 53 e 54.

Figura 53 - Opção de mostrar jogos de memória já jogados salvos.



Fonte: Autoria própria.

Figura 54 - Opção de mostrar jogos de memória criados.



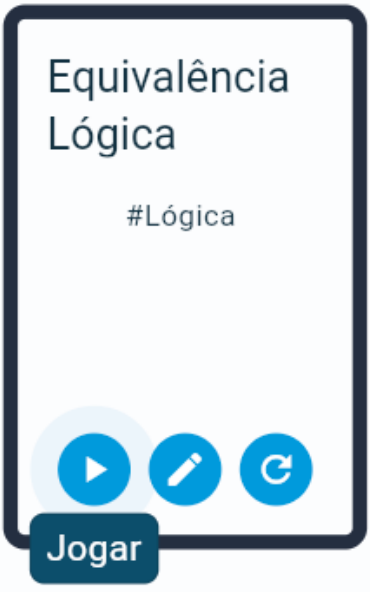
Fonte: Autoria própria.

Cada jogo de memória, em formato de carta, apresentada na tela de dashboard terá algumas opções dependendo do tipo de usuário. Caso o usuário seja somente do tipo criador, como mostrado na figura 55, serão apresentadas as opções “Testar”, “Editar” e “Gerar código” (gerar uma partida). Mas se o usuário for tanto do tipo criador e jogador, como mostra a figura 56, a opção “Testar” será mudado para “Jogar”, no qual vai registrar no histórico quando o usuário jogar o jogo de memória.

Figura 55 - As opções presentes em cada jogo de memória no formato de carta quando usuário é somente criador.

Fonte: Autoria própria.

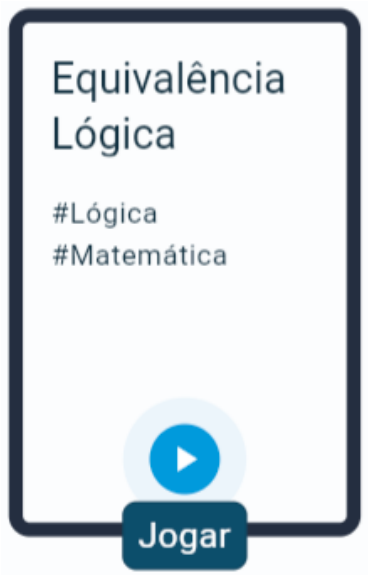
Figura 56 - As opções presentes em cada jogo de memória no formato de carta quando usuário é tanto criador quanto jogador.



Fonte: Autoria própria.

Caso o usuário seja somente do tipo jogador, aparecerá somente a opção de jogar o jogo de memória como mostra a figura 57.

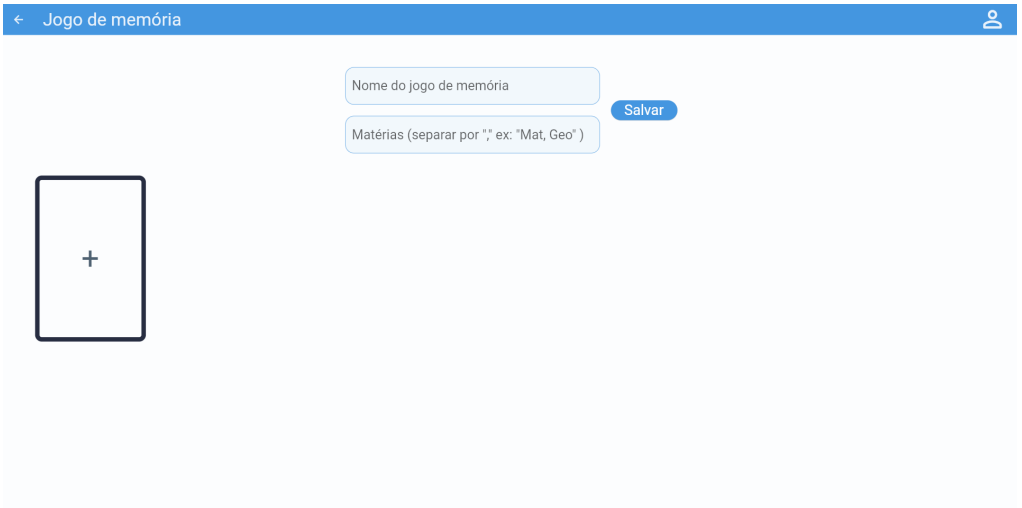
Figura 57 - A opção presente no jogo de memória quando usuário é somente jogador.



Fonte: Autoria própria.

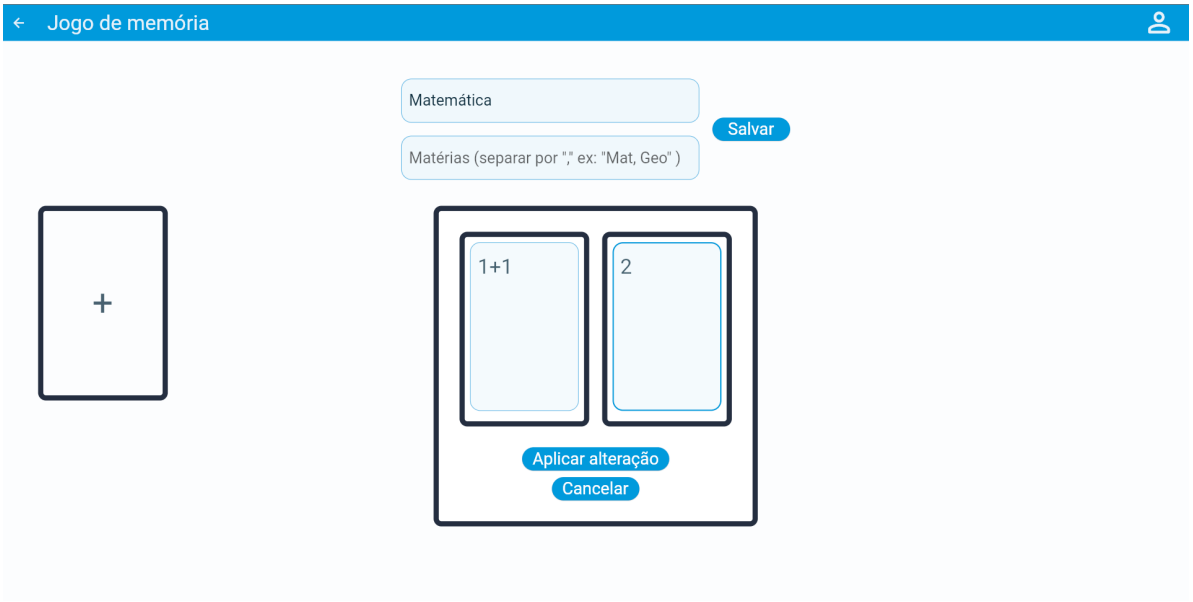
Caso o usuário tenha o tipo de criador, ele poderá criar o jogo de memória inserindo o nome do jogo de memória e criando as cartas. As matérias são opcionais e devem ser inseridas separado por vírgula caso seja mais de uma matéria. Para criar um par de cartas, deve ser clicado na carta de "+" e escrever conteúdo em cada uma das cartas e clicar no botão "Aplicar alteração", como é mostrado na figura 58 e 59.

Figura 58 - A tela de criação de jogo de memória.



Fonte: Autoria própria.

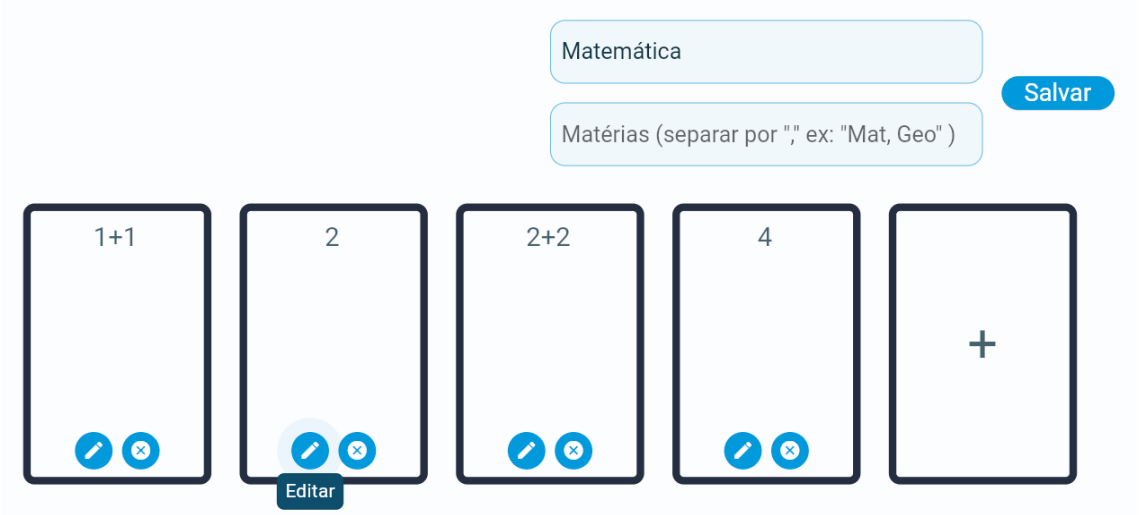
Figura 59 - A tela de criação ou edição de cartas.



Fonte: Autoria própria.

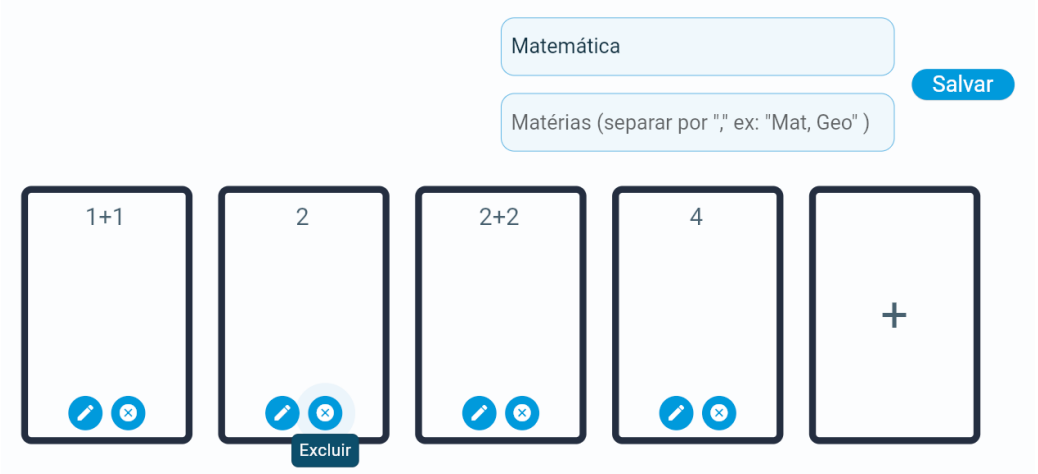
Mostrado na figura 60, o usuário também pode editar as cartas já existentes e editar os conteúdos do par de cartas. A tela que aparece para editar é igual a figura 59, a diferença é que vai aparecer o conteúdo que foi escrito anteriormente. Ou excluir um par de cartas caso deseje, como são mostrados nas figuras 61 e 62. Quando o usuário clica no botão “Salvar” aparece a mensagem que o jogo foi salvo e apresenta as opções “Criar outro jogo de memória” ou “Voltar para dashboard” como mostrado na figura 63.

Figura 60 - A opção de editar um par de cartas.



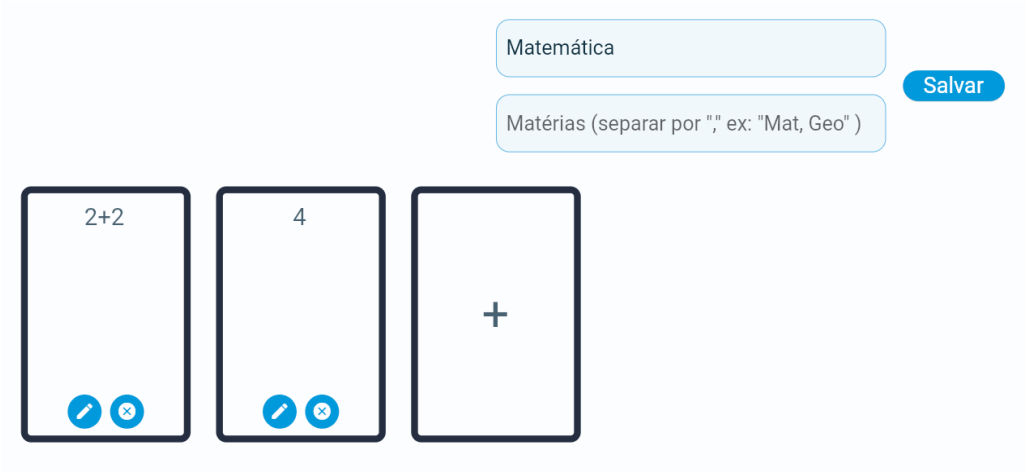
Fonte: Autoria própria.

Figura 61 - A opção de excluir um par de cartas.



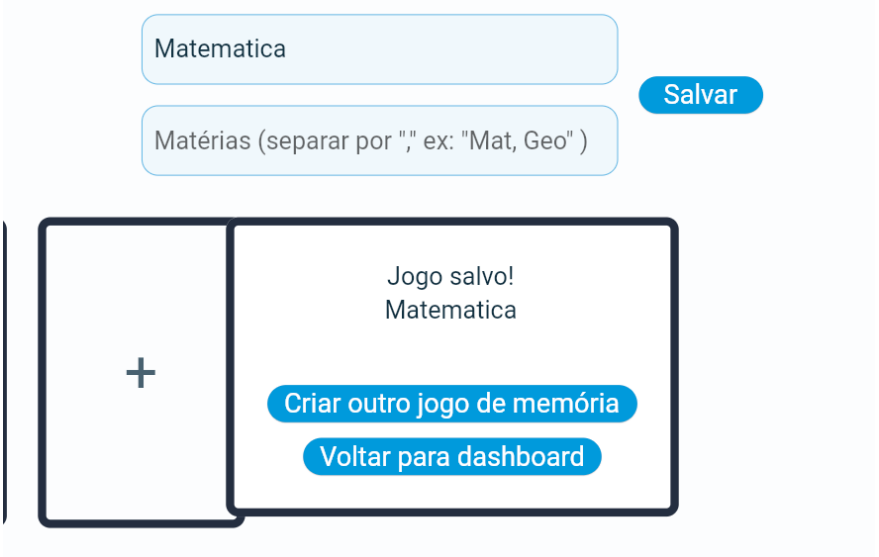
Fonte: Autoria própria.

Figura 62 - Um par de cartas “1+1” e “2” excluídas.



Fonte: Autoria própria.

Figura 63 - Mensagem de jogo salvo com nome de jogo de memória criado.



Fonte: Autoria própria.

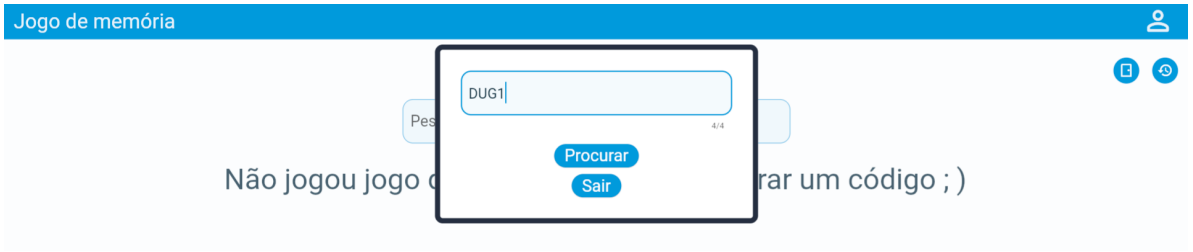
Além de criar um jogo de memória, pode gerar um código para compartilhar com outros usuários para que eles possam entrar na partida e jogar um jogo de memória. A figura 64 mostra a geração de código do jogo de memória chamado “Equivalência Lógica”.

Figura 64 - Tela de geração de código para jogo de memória.

Fonte: Autoria própria.

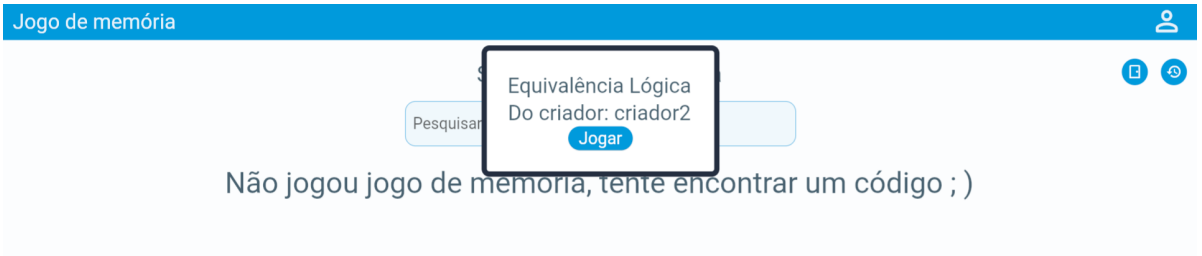
Quando um outro usuário que seja jogador receber código, ele poderá usar a opção de entrar na partida pelo código para digitar o código e encontrar a partida como mostra a figura 65. E quando é encontrada a partida, aparece o nome de jogo de memória e nome de usuário do criador que criou o jogo como mostrado na figura 66.

Figura 65 - Tela de procura de jogo de memória pelo código.



Fonte: Autoria própria.

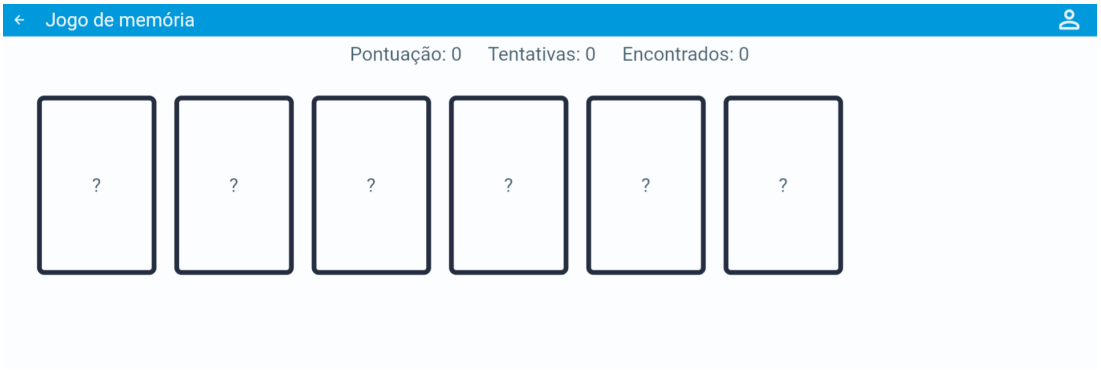
Figura 66 - Jogo de memória encontrado.



Fonte: Autoria própria.

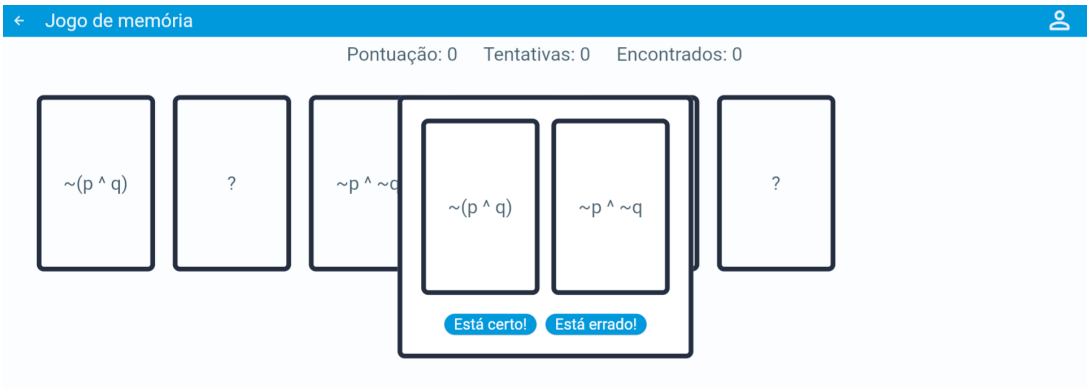
Assim que o usuário entra na partida, todas cartas vão estar viradas e deve selecionar um par de cartas para responder se está certo ou está errado com as opções “Está certo!” e “Está errado!” como mostram as figuras 67 e 68.

Figura 67 - Tela da partida do jogo de memória.



Fonte: Autoria própria.

Figura 68 - Seleção de opção para um par de cartas selecionadas.



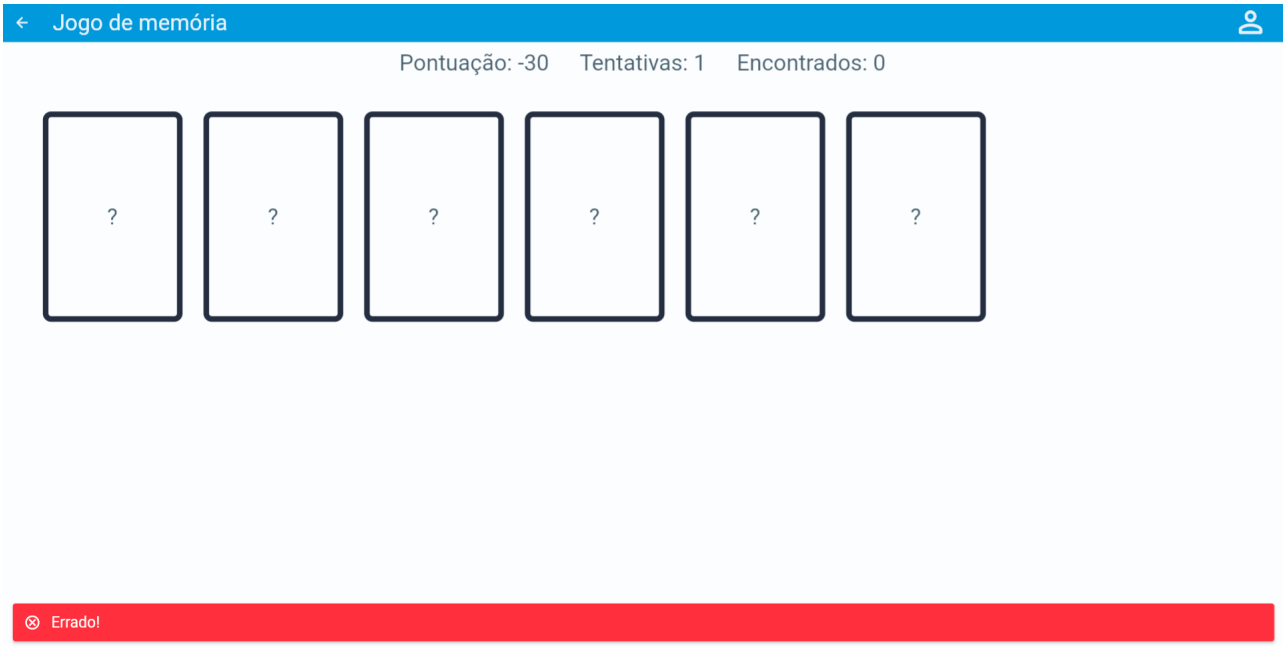
Fonte: Autoria própria.

Se o usuário errar a opção, as cartas se viram novamente e aparece uma barra vermelha com a mensagem “Errado!” no canto inferior da tela, como é mostrado na figura 69. Mas se o usuário acertar a opção, não importando se o usuário selecionou as cartas certas ou não, aparece uma barra verde com a mensagem “Ok!” como mostra as figuras 70 e 71.

Se a opção certa for “Está errado!” , ou seja, as cartas estão erradas, então elas são viradas, como mostra na figura 70. Porém se a opção certa for “Está certo!”, ou seja, as cartas estão certas, as cartas se mantêm com conteúdo revelado e não podem mais ser selecionadas novamente, como mostra na figura 71.

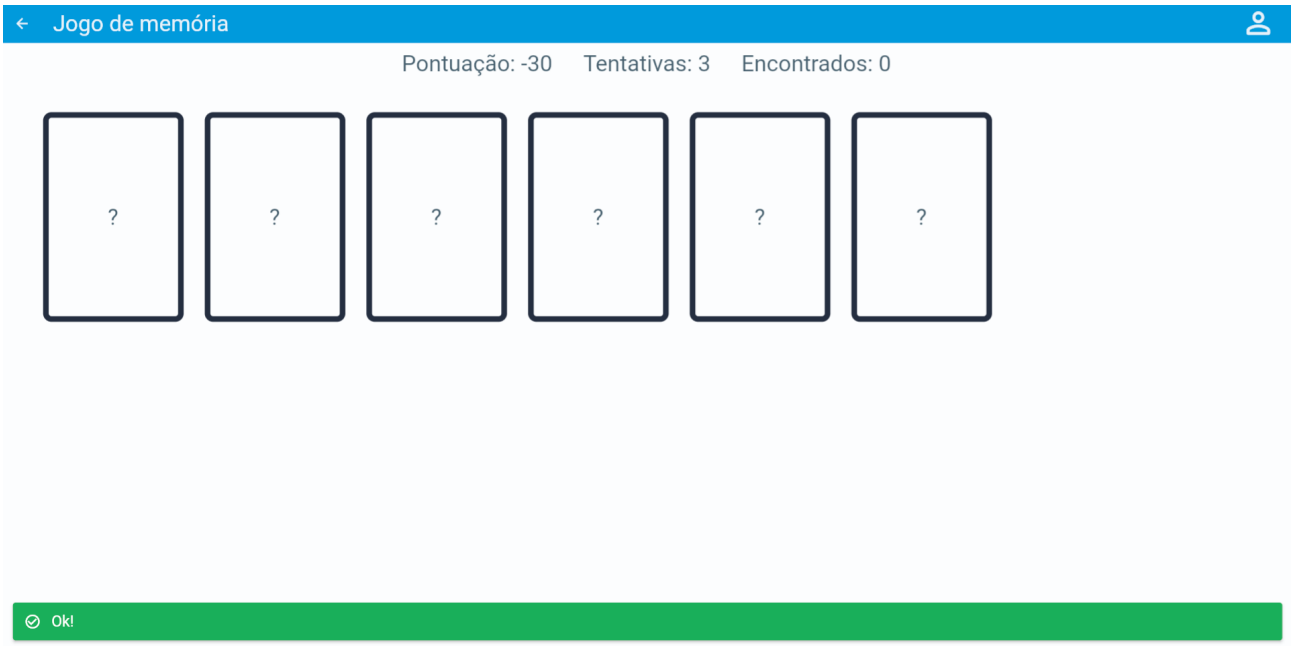
Além disso, a pontuação, quantidade de tentativas e quantidade de encontrados vão sendo atualizados à medida que usuário jogador for avançando na partida, como mostram as figuras 69, 70 e 71.

Figura 69 - Mensagem “Errado!” quando é selecionado a opção errada.



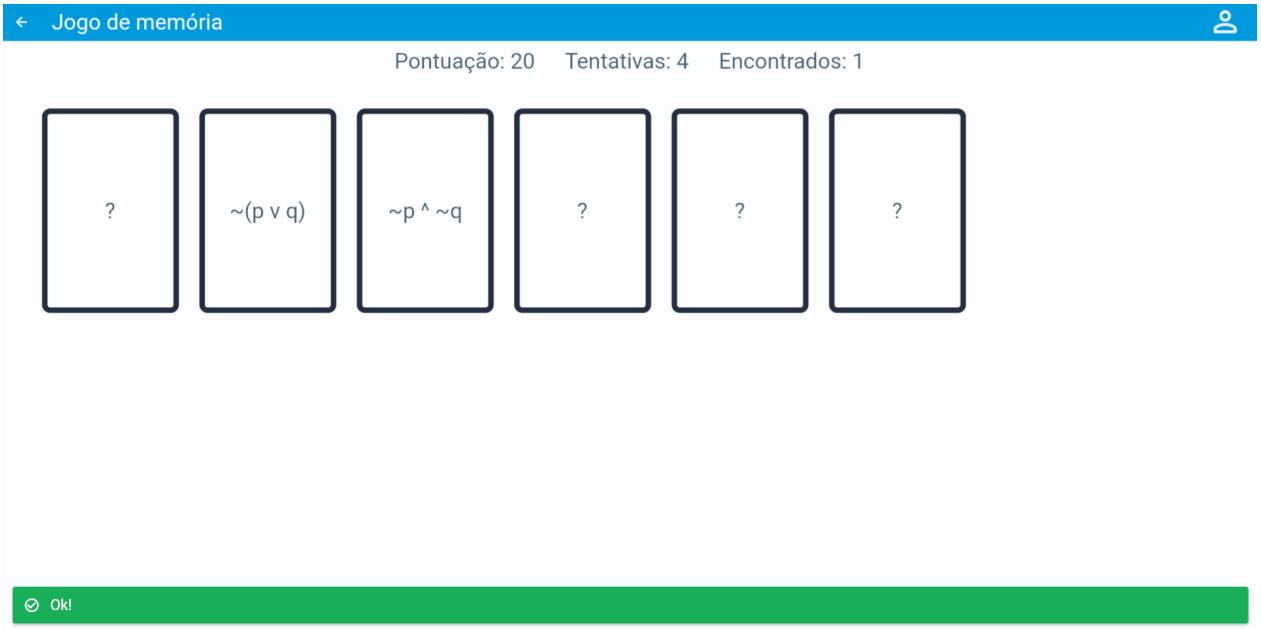
Fonte: Autoria própria.

Figura 70 - Mensagem “Ok!” quando é selecionado a opção certa.



Fonte: Autoria própria.

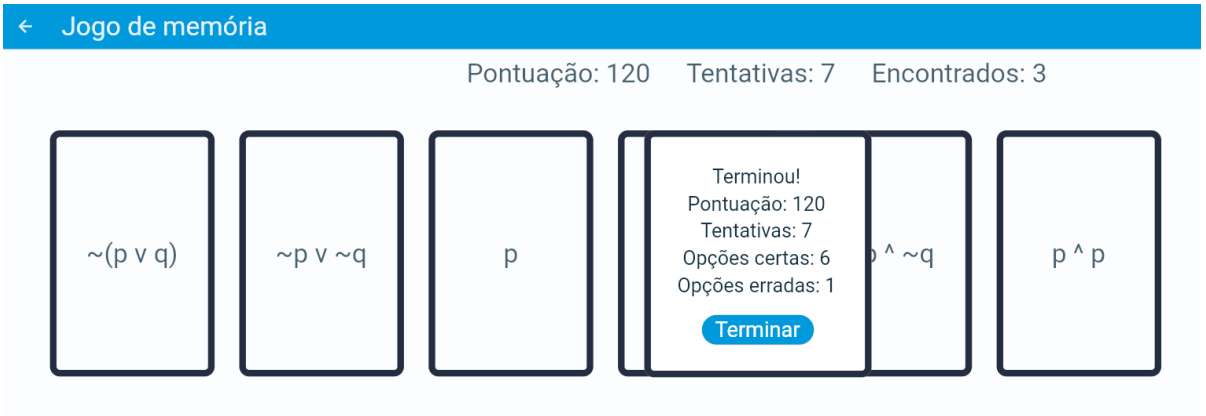
Figura 71 - As cartas selecionadas certas.



Fonte: Autoria própria.

Quando o usuário terminar o jogo, irá aparecer os resultados desta partida como mostra a figura 72. E também aparece o botão "Terminar" para que o usuário vá para a próxima tela.

Figura 72 - As pontuações quando completa o jogo.



Fonte: Autoria própria.

Na próxima tela, figura 73, vão aparecer as pontuações dos outros jogadores durante a partida. O usuário pode voltar para tela de dashboard pelo botão "Voltar para dashboard" ou atualizar as pontuações pelo botão "Recarregar".

Figura 73 - Tela de pontuações da partida atual com outros jogadores.

Fonte: Autoria própria.

Mas além de jogar o jogo de memória pelo código, caso usuário jogador escolha jogar individualmente pela opção “Jogar” no jogo de memória na tela de dashboard, como apresentada nas figuras 56 e 57 anteriormente, a tela de pontuações só apresentará somente deste usuário já que não se trata de partida compartilhada com outros usuários como mostra figura 74.

Figura 74 - Tela de pontuações da partida individual de um jogador.

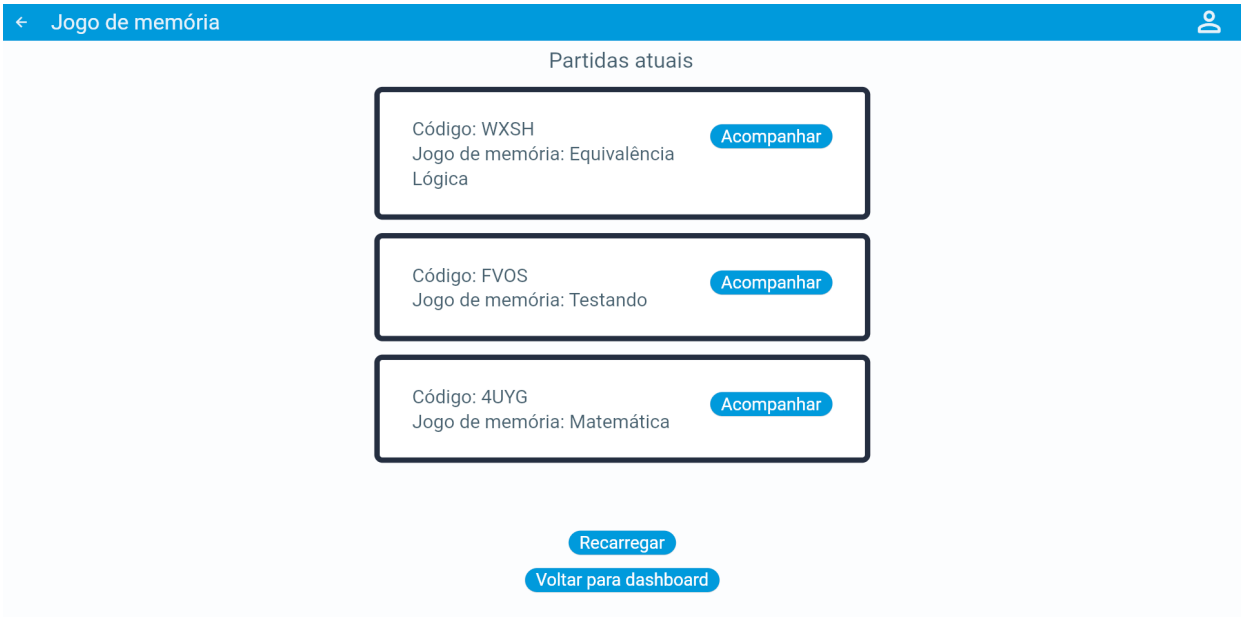
Fonte: Autoria própria.

O usuário que seja do tipo criador, consegue acompanhar as partidas atuais ocorrendo no momento pela opção apresentada na figura 48. Como mostra a figura 75, é exibido a lista

de partidas atuais, e pelo botão “Acompanhar” pode visualizar as pontuações dos jogadores que terminaram em uma determinada partida, na qual vai na tela apresentada na figura 76.

Mas caso não tenha partidas atuais, aparecerá a mensagem “Partidas atuais não encontradas!" mostrado na figura 77. E se tiver presente uma partida atual, mas não têm jogadores participando, aparecerá a mensagem “Jogadores não encontrados!” mostrado na tela da figura 78.

Figura 75 - Tela de acompanhamento de partidas atuais ocorrendo no momento.



Fonte: Autoria própria.

Figura 76 - Tela de pontuações de uma partida atual selecionada anteriormente.

Fonte: Autoria própria.

Figura 77 - Tela de partidas atuais com a mensagem “Partidas atuais não encontrados”.

Fonte: Autoria própria.

Figura 78 - Tela de pontuações com mensagem “Jogadores não encontrados!”.

Fonte: Autoria própria.

O usuário criador também pode consultar as partidas que ele criou anteriormente e que não estão mais ativas como mostrado na figura 49. Como mostra a figura 79, para cada partida é mostrado as informações a respeito da mesma, e também oferece a opção “Detalhes” para visualizar o histórico dos jogadores que jogaram a partida. A figura 80 mostra as pontuações dos jogadores que jogaram de uma determinada partida que foi selecionado anteriormente.

Porém, caso não tenha as partidas criadas anteriormente, aparecerá a mensagem “Partidas não encontradas!" mostrado na figura 81. Caso esteja presente uma partida, mas não tenham jogadores que participaram da partida, aparecerá a mensagem “Jogadores não encontrados!” mostrado anteriormente na tela da figura 78.

Figura 79 - Tela de partidas criadas anteriormente.

Fonte: Autoria própria.

Figura 80 - Tela de pontuações de uma partida anterior selecionada anteriormente.

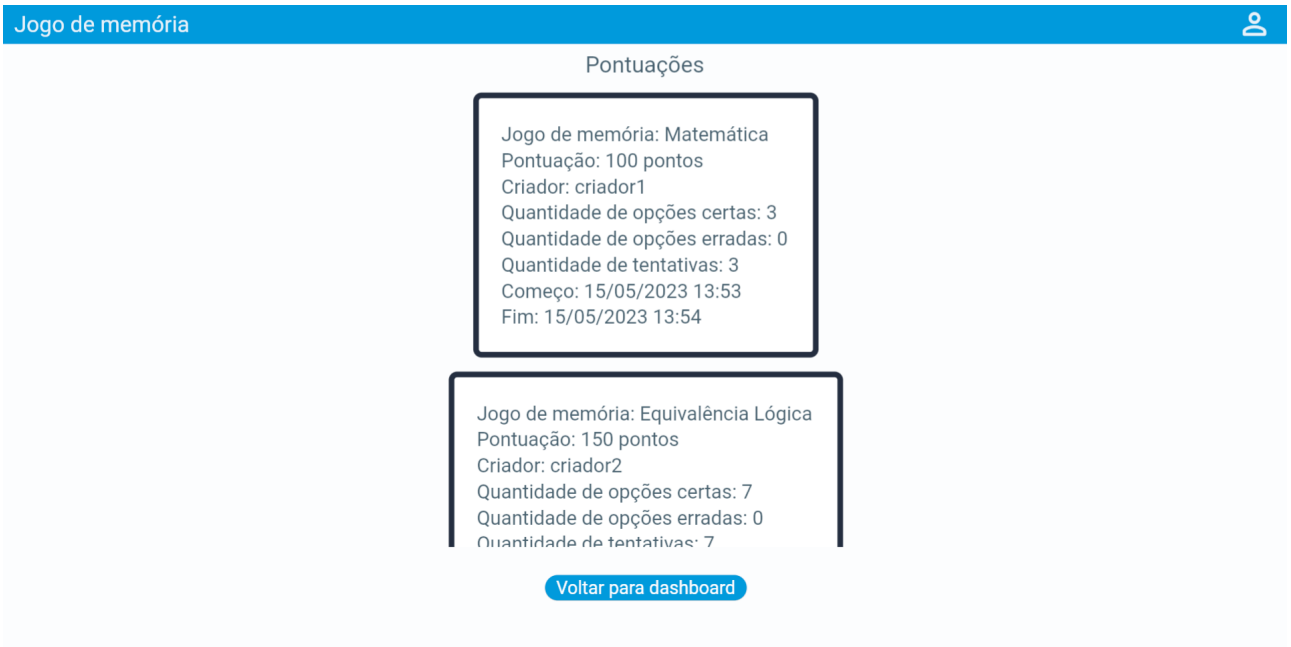
Fonte: Autoria própria.

Figura 81 - Tela de partidas anteriores com a mensagem “Partidas não encontradas!”.

Fonte: Autoria própria.

Assim como usuário do tipo criador pode visualizar o histórico das partidas anteriores, o usuário que tenha tipo de jogador pode visualizar o histórico das partidas que ele jogou anteriormente como mostra a figura 82. Caso não tenha partidas anteriores, aparecerá a mensagem “Não foram encontradas partidas anteriores.” como mostra na tela da figura 83.

Figura 82 - Tela de pontuações das partidas anteriores que o jogador jogou.



Fonte: Autoria própria.

Figura 83 - Tela de pontuações com mensagem “Não foram encontradas partidas anteriores.”.

Fonte: Autoria própria.

12. CONSIDERAÇÕES FINAIS

Conforme apresentado na introdução, o uso da tecnologia da informação ajudou os professores, porém a falta de engajamento e da motivação dos alunos ainda continuam na maior parte das instituições de ensino, independente do nível de educação. Por conta desta barreira, foi encontrado a necessidade de usar alguma técnica que pudesse motivar os alunos a aprenderem, como por exemplo a técnica de gamificação com uso do jogo de memória.

Dessa maneira, o projeto teve como objetivo geral de construir um sistema Web de jogo de memória com uso de gamificação, que oferecesse aos professores criar seus próprios jogos de memória e compartilhassem com alunos para que eles jogassem. Além disso, foi acrescentada uma dificuldade em que, durante a partida do jogo de memória, o aluno tem que escolher se estão certas ou erradas as cartas que foram selecionadas e ganhasse ou perdesse a pontuação de acordo com a opção escolhida. O sistema também oferece aos professores e alunos visualizarem o histórico das partidas atuais ou anteriores.

Para a partida de jogo de memória, foram usados alguns conceitos dos jogos, como: objetivos (exemplo: a aluno deve completar todas as cartas), regras (exemplo: se as cartas já estão viradas, não podem usar mais estas cartas), feedback (exemplo: a tela de pontuação para informar o resultado ao aluno) e uma espécie de desafio, como dito anteriormente, em que aluno deve escolher se estão certas ou erradas as cartas que foram selecionadas, o que necessita o aluno ter conhecimento do conteúdo que professor criou para este jogo de memória.

Para desenvolvimento do sistema, foram usadas as tecnologias: framework Flutter, linguagem de programação Dart e um conjunto de bibliotecas de terceiros para construção de website, do lado do cliente; framework Spring, linguagem de programação Java e um conjunto de bibliotecas de terceiros para construção de backend, do lado de servidor; e PostgreSQL para construção de banco de dados para armazenar dados da aplicação em conjunto com servidor.

Para o lado do servidor, foi usado uma arquitetura em camadas para cada camada ter sua determinada responsabilidade, com uso de métodos HTTP para facilitar o consumo de serviços exposto pelo Controlador. E para o lado do cliente, foi usado uma arquitetura adaptada de MVVM em conjunto com InheritedWidget e Future, que são recursos que o framework Flutter oferece, para separar em camadas e facilitar o compartilhamento de dados e lógica entre componentes diferentes.

Durante o trabalho, observa-se que foi possível construir este sistema com as ferramentas propostas usando alguns conceitos dos jogos apresentados anteriormente, como objetivos, regras, feedback e desafio para motivar o aluno, e facilitando a criação ou edição e compartilhamento do jogo de memória por parte do professor.

Diante desses fatos, o objetivo geral foi atendido, a criação deste sistema Web, onde foi utilizado a gamificação, oferece uma ferramenta que professores podem usar em suas salas de aulas para motivar os alunos por meio do jogo de memória e a possibilidade de acompanhar as partidas atuais e visualizar o histórico das partidas anteriores, servindo para tanto professor, quanto aluno.

Para sugestão de trabalho de futuro, poderiam ser oferecidas:

- a opção de rankear as pontuações. Ou seja, ordenar de forma decrescente e mostrar as posições dos alunos na qual poderia servir como uma espécie de competição entre alunos, na qual é um dos elementos presente nos jogos.
- a opção de limitar o tempo da partida do aluno, como por exemplo, limitar o tempo em 5 minutos, mas isso dependerá do tempo escolhido do professor. Desta forma, o aluno terá desafio de completar o jogo de memória antes que o acabe.
- aplicativos para Android e IOS, já que framework Flutter é uma tecnologia multiplataforma que pode ser utilizada para construir aplicativos em plataforma diferente.

REFERÊNCIAS

AKARIE, M. M. ***auto_route 6.0.5.*** Pub.dev, 2023a. Disponível em: <https://pub.dev/packages/auto_route>. Acesso em: 21 mar. 2023.

AKARIE, M. M. ***injectable 2.1.1.*** Pub.dev, 2023b. Disponível em: <https://pub.dev/packages/injectable>. Acesso em: 30 mar. 2023.

ANDERSON, et al. **Criar DTOs (Objetos de Transferência de Dados).** Disponível em: <https://learn.microsoft.com/pt-br/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5>. Acesso em: 08 mai. 2023.

API FLUTTER DEV. ***FutureBuilder<T> class.*** Flutter Dev, 2023a. Disponível em: <https://api.flutter.dev/flutter/widgets/FutureBuilder-class.html>. Acesso em: 10 maio. 2023.

API FLUTTER DEV. ***InheritedWidget class.*** Flutter Dev, 2023b. Disponível em: <https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html>. Acesso em: 10 maio. 2023.

AUTH0. ***Introduction to JSON Web Tokens.*** Auth0, 2023. Disponível em: <https://jwt.io/introduction>. Acesso em: 09 mar. 2023

AZURE. **O que é o Java Spring Boot?** Microsoft Azure, 2023. Disponível em: <https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-java-spring-boot/> Acesso em: 21 mar. 2023

BAUER, C.; KING, G.; GREGORY, G. ***Java persistence with Hibernate.*** Shelter Island: Manning Publcations Co, 2016.

BURKHART, T. ***get_it 7.2.0.*** Pub.dev, 2021. Disponível em: <https://pub.dev/packages/get_it>. Acesso em: 21 mar. 2023

CAROLINA TOMÉ KLOCK, A.; FARIAS DE CARVALHO, M.; EDUARDO ROSA, B.; GASPARINI, I. **Análise das técnicas de Gamificação em Ambientes Virtuais de Aprendizagem.** RENOTE, Porto Alegre, v. 12, n. 2, 2014. DOI: 10.22456/1679-1916.53496. Disponível em: <https://seer.ufrgs.br/index.php/renote/article/view/53496>. Acesso em: 22 nov. 2022.

CAROLINA TOMÉ KLOCK, A.; FELIPE DA CUNHA, L.; GASPARINI, I. **Um modelo conceitual para a gamificação de Ambientes Virtuais de Aprendizagem.** Revista Novas Tecnologias na Educação, Porto Alegre, v. 13, n. 1, 2015. DOI: 10.22456/1679-1916.57654. Disponível em: <https://seer.ufrgs.br/index.php/renote/article/view/57654>. Acesso em: 30 nov. 2022.

COLLINGS, T. ***Controller-Service-Repository.*** Medium, 2021. Disponível em: <https://tom-collings.medium.com/controller-service-repository-16e29a4684e5>. Acesso em: 08 mai. 2023.

CONTROLE NET. **Cliente-Servidor, uma estrutura lógica para a computação centralizada.** Controle Net, 2023. Disponível em: <https://www.controle.net/faq/cliente-servidor-uma-estrutura-para-a-computacao-centralizada>. Acesso em: 09 maio. 2023.

DART DEV. ***Dart overview.*** Dart, 2023a. Disponível em: <https://dart.dev/overview>. Acesso em: 13 mar. 2023

DART DEV. ***Asynchronous programming: futures, async, await.*** Dart Dev, 2023b. Disponível em: <https://dart.dev/codelabs/async-await>. Acesso em: 10 maio. 2023.

DEITEL, Paul; DEITEL, Harvey. **Java: como programar.** 10 ed. São Paulo: Pearson Education do Brasil, 2017.

DOCS FLUTTER DEV. ***Build a form with validation.*** Flutter, 2023. Disponível em: <https://docs.flutter.dev/cookbook/forms/validation>. Acesso em: 06 abr. 2023

DOCS INSOMNIA REST. ***Send Your First Request.*** Insomnia, 2021. Disponível em: <https://docs.insomnia.rest/insomnia/send-your-first-request>. Acesso em: 21 mar. 2023.

DOCS SPRING SECURITY. ***Spring Security. Overview.*** Spring, 2023a. Disponível em: <https://docs.spring.io/spring-security/reference/index.html>. Acesso: 21 mar. 2023.

DOCS SPRING SECURITY. ***Expression-Based Access Control.*** Spring, 2023b. Disponível em: <https://docs.spring.io/spring-security/reference/5.8/servlet/authorization/expression-based.html#_access_control_using_preauthorize_and_postauthorize>. Acesso: 21 mar. 2023.

FLUTTER DEV. ***shared_preferences 2.0.20.*** Pub.dev, 2023. Disponível em: <https://pub.dev/packages/shared_preferences>. Acesso em: 21 mar. 2023

FLYWAY. ***Homepage.*** REDGATE, 2023a. Disponível em: <https://flywaydb.org/>. Acesso em: 21 mar. 2023

FLYWAY. ***Migrations - Flyway - Product Documentation.*** REDGATE, 2023b. Disponível em: <https://documentation.red-gate.com/fd/migrations-184127470.html>. Acesso em: 31 mar. 2023.

GAME ARKOS. **Nossa Equipe.** Game Arkos, 2023. Disponível em: <https://gamearkos.com.br/sobre>. Acesso em: 10 de mai. 2023.

GOOGLE DEV. ***json_serializable 6.6.1.*** pub.dev, 2023. Disponível em: <https://pub.dev/packages/json_serializable> Acesso em: 21 mar. 2023.

INTELLIJ IDEA. ***Change signature.*** IntelliJ IDEA, 2023a. Disponível em: <https://www.jetbrains.com/help/idea/change-signature.html#change_method_signature_refactoring>. Acesso em: 06 abr. 2023.

INTELLIJ IDEA. ***What is IntelliJ IDEA?*** IntelliJ IDEA, 2023b. Disponível em: <https://www.jetbrains.com/idea/features/#deep-code-insight>. Acesso em: 06 abr. 2023.

JACKSON, W. ***JSON Quick Syntax Reference.*** [s.l.] Berkeley, Ca Apress, 2016.

LUBOWA, E. ***Why you need to use DTO's in your REST API.*** Medium, 2021. Disponível em: <https://medium.com/@enocklubowa/why-you-need-to-use-dtos-in-your-rest-api-d9d6d7be5450>. Acesso em: 08 mai. 2023.

MAPSTRUCT. ***MapStruct 1.5.3.Final Reference Guide.*** MapStruct, 2022. Disponível em: <https://mapstruct.org/documentation/stable/reference/pdf/mapstruct-reference-guide.pdf>. Acesso em: 09 mar. 2023

MILANI, A. **PostgreSQL - Guia do Programador.** [s.l.] Novatec Editora, 2008.

MOHITE, J. ***Flutter: MVVM Architecture.*** Medium, 2020. Disponível em: <https://medium.com/flutterworld/flutter-mvvm-architecture-f8bed2521958>. Acesso em: 09 maio. 2023.

MOZILLA. ***An overview of HTTP.*** Mozilla, 2023. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>. Acesso em 28 mar. 2023

ORACLE. ***What is a JPA Entity?*** Oracle, 2009. Disponível em: <https://docs.oracle.com/cd/E16439_01/doc.1013/e13981/undejbs003.htm>. Acesso em: 10 maio. 2023.

POSTGRESQL. ***PostgreSQL: About.*** PostgreSQL, 2023. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 07 mar. 2023

PROJECT LOMBOK. ***Stable.*** Disponível em: <https://projectlombok.org/features/>. Project Lombok, 2023. Acesso em: 09 mar. 2023

RED HAT. **O que é API REST?** Red Hat, 2020. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>. Acesso em: 06 abr. 2023

SEBESTA, R. W. **Conceitos de Linguagens de Programação - 9.ed.** [s.l.] Bookman Editora, 2011.

SILVA, J. B. DA .; SALES, G. L.; CASTRO, J. B. DE .. **Gamificação como estratégia de aprendizagem ativa no ensino de Física. Revista Brasileira de Ensino de Física**, v. 41, n. 4, p. e20180309, 2019. Disponível em: <https://www.scielo.br/j/rbef/a/Tx3KQcf5G9PvcgQB4vswPbq/#>. Acesso em: 10 de mai. 2023.

SPRING. ***Why Spring?*** Spring, 2023. Disponível em: <https://spring.io/why-spring>. Acesso em: Acesso: 21 mar. 2023.

STONIS, MICHAEL, et al. ***Model-View-ViewModel***. Microsoft, 2022. Disponível em: <https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm>. Acesso em: 08 mai. 2023.

THEMISIR.COM. ***form_validator 2.1.1.*** pub.dev, 2023. Disponível em: <https://pub.dev/packages/form_validator>. Acesso em: 21 mar. 2023

TOLOMEI, B. V. **A Gamificação como Estratégia de Engajamento e Motivação na Educação.** EaD em Foco, [S. l.], v. 7, n. 2, 2017. DOI: 10.18264/eadf.v7i2.440. Disponível em: <https://eademfoco.cecierj.edu.br/index.php/Revista/article/view/440>. Acesso em: 30 nov. 2022.

VALENTE, M. T. **Cap. 7: Arquitetura – Engenharia de Software Moderna.** Engenharia de Software Moderna, 2020. Disponível em: <https://engsoftmoderna.info/cap7.html>. Acesso em: 09 maio. 2023.

VISUAL STUDIO CODE. ***Documentation for Visual Studio Code.*** Microsoft, 2023a. Disponível em: <https://code.visualstudio.com/docs>. Acesso em: 06 abr. 2023

VISUAL STUDIO CODE. ***IntelliSense in Visual Studio Code.*** Microsoft, 2023b. Disponível em: <https://code.visualstudio.com/docs/editor/intellisense>. Acesso em: 06 abr. 2023

W3 SCHOOLS. ***HTML5 Web Storage.*** W3 SCHOOLS, 2023. Disponível em: <https://www.w3schools.com/HTML/html5_webstorage.asp>. Acesso em: 21 mar. 2023

WALLS, C. ***Spring in Action.*** Shelter Island: Manning Publcations Co, 2022.

WINDMILL, E.; RISCHPATER, R. ***Flutter in action.*** [s.l.] Shelter Island, Ny Manning Publications Co, 2020.

ZWICK, URI; PATERSON, MICHAEL S.. ***The Memory Game.*** Theoretical Computer Science, v. 110, p. 169-196, 1993. Disponível em: <https://www.sciencedirect.com/science/article/pii/030439759390355W>. Acesso em: 07. de dez. 2022.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Guilherme Henrique Mendonça Nascente do Curso de Ciência da Computação, matrícula 2019.2.0028.0052-4, telefone: (62) 99223-1289, e-mail henrique101124@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado SISTEMA WEB DE APRENDIZAGEM COM USO DE GAMIFICAÇÃO, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 16 de junho de 2023.

Assinatura do autor: _____

Nome completo do autor: Guilherme Henrique Mendonça Nascente

Assinatura do professor–orientador: _____

Nome completo do professor-orientador: Eugênio Júlio Messala Cândido Carvalho