Rafael Rani Santos Martins

Ambiente de Autoria de Jogos Sérios, GREAT: Módulo de Resolução de Tarefas Offline

São Luís/MA

2019

Rafael Rani Santos Martins

Ambiente de Autoria de Jogos Sérios, GREAT: Módulo de Resolução de Tarefas Offline

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Universidade Federal do Maranhão

Orientador: Prof. Dr. Mario Antônio Meireles Teixeira

São Luís/MA

2019

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Santos Martins, Rafael Rani.
AMBIENTE DE AUTORIA DE JOGOS SÉRIOS, GREAT: : MÓDULO DE RESOLUÇÃO DE TAREFAS OFFLINE / Rafael Rani Santos Martins. – 2019.
43 f.

Orientador(a): Prof. Mario Antônio Meireles Teixeira.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, Centro de Ciências Exatas e Tecnologia, 2019.

1. Ferramenta de Autoria. 2. Jogos Educacionais. 3. Jogos Sérios. 4. Modo offline. I. Meireles Teixeira, Prof. Mario Antônio. II. Título.

Rafael Rani Santos Martins

Ambiente de Autoria de Jogos Sérios, GREAT: Módulo de Resolução de Tarefas Offline

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. São Luís/MA, 16 de julho de 2019:

Prof. Dr. Mario Antônio Meireles Teixeira
Universidade Federal do Maranhão
Orientador

Profª. M.ª Alana Oliveira Meireles Teixeira
Universidade Federal do Maranhão
Membro da banca

Prof. Dr. Carlos de Salles Soares Neto
Universidade Federal do Maranhão
Membro da banca

São Luís/MA
2019

Dedico este trabalho primeiramente a Deus, a minha família e a todos os professores do Departamento de Informática da Universidade Federal do Maranhão que de alguma forma contribuíram em minha graduação ao longo destes anos.

Agradecimentos

Gostaria de agradecer e dedicar esta dissertação ao Senhor da minha vida, Jesus Cristo, por ter me dado vida, sabedoria e permitido chegar até aqui.

A minha mãe Iran Cerqueira, que já se foi, mas continua sendo minha maior força e inspiração na vida.

A minha noiva Thaynara Lima, por seu companheirismo, paciência nos meus dias ruins, pelos conselhos e carinho. Com certeza, o fardo se tornou mais leve por saber que tenho seu amor.

Ao meu orientador e professor DR. Mário Antônio Meireles Teixeira e ao meu amigo Eduardo do LAWS, por me indicar o caminho a seguir durante a realização deste trabalho.

Agradeço a esta universidade, ao departamento de informática e seu corpo docente que oportunizaram a janela que hoje vislumbro um horizonte superior.

"O sucesso não é garantido, mas o fracasso é certo se você não estiver emocionalmente envolvido em seu trabalho."

(Biz Stone – Fundador do Twitter)

Resumo

Uma forma muito comum e popular de entretenimento praticada por público de todas as idades desde muito tempo são os jogos. E devido à grande expansão da tecnologia digital ocorrida nos últimos anos, os jogos digitais hoje estão presentes em quase todos os dispositivos eletrônicos, podendo também ser acessados através da web. Existe uma categoria de jogos chamada serious games, cuja finalidade vai além de simplesmente entreter o jogador, pois são usados de forma lúdica com o propósito de ensinar, treinar, ajudar em tratamento de saúde, entre outros fins. No contexto educacional, os jogos são uma ótima forma de facilitar o aprendizado, pois atraem a atenção do aluno, que muitas vezes por conta de um mundo cada vez mais conectado e interativo acaba tendo sua atenção "roubada". Com o intuito de promover o ensino através do uso de jogos digitais, foi criado o ambiente de autoria de jogos digitais chamado GREAT (Game Ready Authoring Tool), que permite ao professor, mesmo sem nenhum conhecimento em programação, criar jogos para serem acessados pelos alunos através da internet, para que estes possam aprender enquanto se divertem. O presente trabalho tem como propósito a construção de um protótipo de um módulo para resolução de tarefas em modo offline, que irá compor o ambiente de autoria de jogos digitais GREAT, oferecendo a opção para o aluno que não dispõe de acesso à internet o tempo inteiro, de poder baixar os jogos ao acessá-los, para posteriormente jogar offline e, quando novamente estiver conectado, fazer o upload de seu progresso e verificar seus resultados. O módulo foi desenvolvido usando o framework web Flask e usa os princípios REST para permitir a comunicação com a ferramenta GREAT, além de fazer uso da abordagem PWA a fim de dar ao aluno que acessa pelo smartphone uma experiência semelhante ao uso de um aplicativo mobile.

Palavras-chave: Jogos Sérios; Jogos Educacionais; modo offline; Ferramenta de Autoria.

Abstract

A very common and popular form of entertainment practiced by the public of all ages has long been games. And due to the great expansion of digital technology in recent years, digital games today are present in almost all electronic devices, and can also be accessed through the web. There is a category of games called serius games, whose purpose goes beyond simply entertaining the player, as they are used in a playful way for the purpose of teaching, training, help in health treatment, among other purposes. In the educational context games are a great way to facilitate learning, because they attract the attention of the student, who often because of an increasingly connected and interactive world ends up having their attention "stolen". In order to promote teaching through the use of digital games, the authoring environment for digital games called GREAT (Game Ready Authoring Tool) was created, which allows the teacher, even without any programming knowledge, to create games to be accessed by students through the internet, so they can learn while having fun. The purpose of the present work is to construct a prototype of a module for task resolution in offline mode, which will compose the environment of authoring of GREAT digital games, offering the option for the student who does not have access to the Internet. you can download the games by accessing them, then playing offline, and when you are connected again, upload your progress and check your results. The module was developed using the Flask web framework and uses the REST principles to allow communication with the GREAT tool, in addition to making use of the PWA approach in order to give the student accessing the smartphone an experience similar to the use of a mobile application.

Keywords: Serious Games; Educational Games; Offline mode; Authoring Tool.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SD	Sistema Distribuído
SO	Sistema Operacional
REST	REpresentational State Transfer
API	Application Programming Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
PHD	Philosophiæ Doctor
SOAP	Simple Object Access Protoco
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
CRUD	Create, Read, Update, Delete
JSON	JavaScript Object Notation
XML	Extensible Markup Language
HTML	HyperText Markup Language
PC	Personal Computer
GREAT	Game Ready Authoring Tool
SOA	Service-Oriented Architecture
PWA	Progressive Web Apps
NFC	Near Field Communication
UML	Unified Modeling Language

Sumário

1 INTRODUÇÃO

Desde seu surgimento em meados dos anos 50, os jogos digitais sempre foram uma excelente forma de entretenimento e diversão para crianças, jovens, adultos e até mesmo idosos. Atualmente a indústria de jogos usufrui de melhores tecnologias gráficas e computacionais do que até meados dos anos 90, o que expande as possibilidades na construção de um jogo como o uso de imagens mais realistas, Realidade Virtual e Aumentada e uso de inteligência artificial nos personagens do jogo.

Os jogos digitais estão cada vez mais presentes em nossa sociedade através da expansão da tecnologia digital, pois não se limitam mais a estarem somente nos consoles e desktops como também em notebooks, smartphones, tablets, videogames portáteis, Smart TV's e outros dispositivos eletrônicos, podendo também ser acessados através da web.

Existe a categoria de jogos denominada de jogos sérios (serious games) que possuem objetivos que vão além do mero entretenimento, pois estes são voltados ao aprendizado, treinamento e até mesmo tratamento de saúde de forma lúdica. O estímulo de funções cognitivas como raciocínio, abstração, linguagem, atenção, criatividade, capacidade de resolução de problemas, memória, entre outros, além da motivação e aquisição de conhecimentos são elementos fundamentais em um jogo sério (MACHADO; MORAES; NUNES, 2009).

No contexto educacional, os jogos sérios são uma ótima maneira de estimular o estudo, pois facilita o aprendizado "prendendo" a atenção do aluno por meio do entretenimento, típico de qualquer jogo. Segundo dados de pesquisa realizada em julho de 2013, [ElearningBrasil 2013], 92 por cento dos participantes concluem que a aplicação de jogos educativos ao e-Learning pode aumentar o desempenho e a capacidade de aprendizado do aluno, podendo ser utilizados para ensinar e treinar o mesmo, promovendo-lhe maior competência e destreza. Estes mesmos participantes acreditam que os jogos educativos ajudam a mudar a forma de avaliação tradicional, fazendo com que os alunos se sintam empolgados em fazê-la e em aprender algo novo, e não mais pressionados (OLIVEIRA; OLIVEIRA; TEIXEIRA,).

Com a finalidade de promover o ensino através do uso dos games foi criado o ambiente de autoria de jogos digitais chamado GREAT (Game Ready Authoring Tool) (OLIVEIRA; NETO; TEIXEIRA, 2014), que se trata de uma aplicação desenvolvida para a criação de jogos educacionais, o qual permite ao professor, mesmo sem nenhum conhecimento em programação, criar jogos de sua própria autoria para serem acessados pelos alunos através da internet, a fim de que estes possam aprender sobre o conteúdo abordado no jogo enquanto se divertem.

O GREAT é um sistema web distribuído em vários serviços, fazendo uso de princípios REST para permitir a comunicação entre os módulos que compõem a aplicação. Esta ferramenta também faz uso da abordagem PWA (Progressive Web App) para promover aos usuários que a acessam pelo smartphone uma experiência semelhante ao uso de uma aplicativo mobile, podendo oferecer funcionamento offline, notificações push entre outras funcionalidades antes restritas apenas a aplicativos nativos.

No intuito de aprimorar esta ferramenta de incentivo educacional, este trabalho propõe a construção de um protótipo de um módulo de resolução de tarefas em modo offline, o módulo Playground, para que seja possível ao aluno jogar, mesmo estando desconectado da internet, e posteriormente, ao obter acesso, fazer o upload de seu progresso e verificar sua pontuação alcançada. O módulo foi desenvolvido usando o framework do Python para web chamado Flask, além de usar os princípios REST para a comunicação com a ferramenta GREAT através do protocolo HTTP e fazer uso da abordagem PWA.

1.1 Objetivo Geral

Este trabalho tem como objetivo a construção de um protótipo de módulo ao ambiente de autoria de jogos sérios, GREAT (Game Ready Authoring Tool), o módulo Playground, que faça com que o aluno (jogador) tenha a possibilidade de baixar o jogo do seu PC ou dispositivo móvel ao acessá-lo, jogar offline e fazer upload de suas respostas.

1.1.1 Objetivos Específicos

a) Buscar jogos cadastrados no sistema;

b) Obter os jogos através do método de requisição à API do GREAT;

c) Salvar o jogo no cache do navegador para posterior acesso em modo offline;

d) Realizar upload das respostas para obtenção dos resultados.

1.2 Organização do Trabalho

O capítulo 2 trata do conceito de Sistemas Distribuídos, explicando seu funcionamento e citando suas características que garantem vantagem sobre os sistemas unificados tradicionais. Apresenta também Arquitetura Orientada a serviços, tecnologia utilizada da implementação dos sistemas distribuídos que permite a comunicação até mesmo entre sistemas escritos em diferentes linguagens. Neste capítulo é mostrado ainda o conceito de API, interface de Programação de Aplicativos, criadas para serem consumidas por aplicações.

O capítulo 3 apresenta o modelo REST, utilizado para projetar arquiteturas de sistemas distribuídos ou API's, que usam comunicação via rede através do protocolo HTTP, abordando seus cinco princípios fundamentais e como os utilizar da forma correta.

O capítulo 4 apresenta a abordagem PWA para aplicações web, que tem a finalidade de promover ao usuário de uma experiência mais completa, muito próxima a de um aplicativo mobile, citando suas principais funcionalidades, vantagens e desvantagens frente às aplicações nativas.

O capítulo 5 apresenta a aplicação web desenvolvida (Módulo Playground) mostrando suas modelagens, especificações, funcionalidades e interfaces.

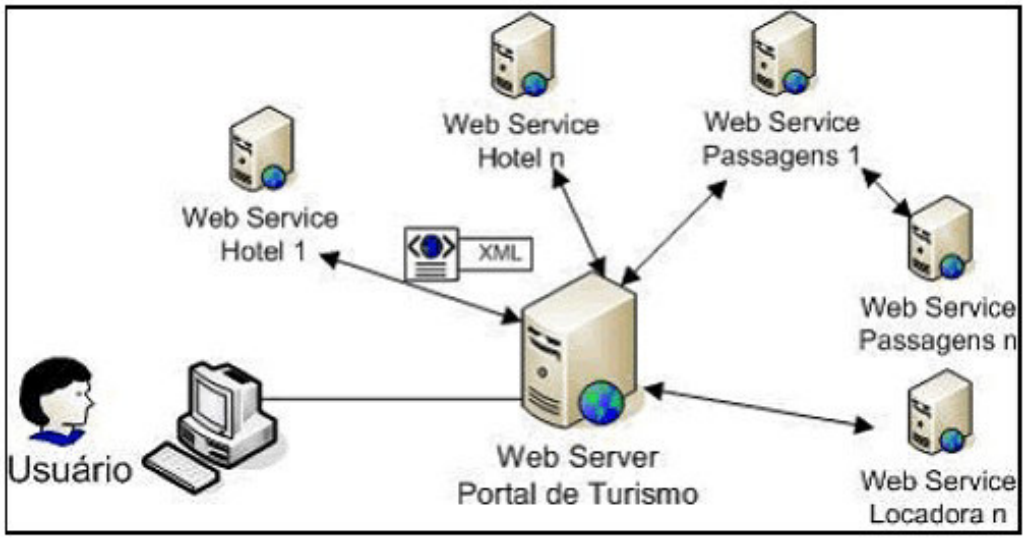
2 SISTEMAS DISTRIBUÍDOS

Segundo Tanenbaum e Steen (2007), “Um sistema distribuído é um conjunto de computadores independentes entre si que se apresenta a seus usuários como um sistema único e coerente”, com base nesta informação, podemos então considerar um sistema distribuído como uma aplicação integrada, composta por uma colação de computadores e softwares interconectados em uma rede.

2.1 Como Funciona um Sistema Distribuído

Nos Sistemas Distribuídos, os módulos que compõem o sistema são divididos entre as diferentes máquinas sem interação direta do usuário. Os componentes localizados nos computadores interligados em rede se comunicam e coordenam suas ações apenas trocando mensagens (COULOURIS; DOLLIMORE; KINDBERG, 2005), conforme mostrado na figura 1.

Figura 1 – Esquema de comunicação em um SD



Fonte: CulturaMix (2013)

2.2 Características dos Sistemas Distribuídos

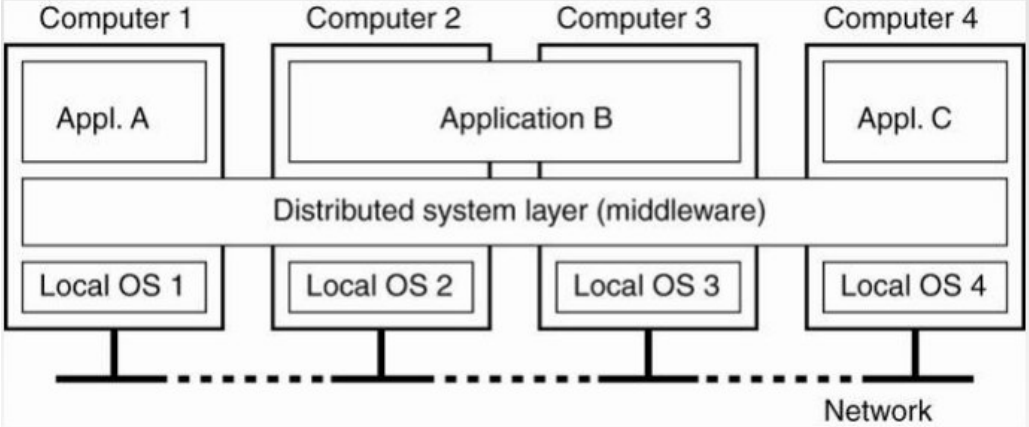
2.2.1 Heterogeneidade

Um sistema distribuído pode possuir componentes em diferentes tipos de entidades computacionais, escritas em diferentes linguagens de programação, controladas por diferen-

tes S.O. e conectadas por diferentes tipos de rede. Com o objetivo de tornar a comunicação entre tais entidades possível de forma transparente são utilizados os protocolos de rede. No entanto, há aplicações que necessitam de um Middleware para assegurar a coerência na comunicação.

O middleware permite que diferentes aplicações executando em diferentes plataformas se comuniquem de forma transparente em um SD. Trata-se de uma camada intermediária que atua entre a aplicação e a plataforma (S.O.) composta por um conjunto de funcionalidades e padronizações, oferecendo uma abstração para a comunicação e representação dos dados (TANENBAUM; STEEN, 2007). A forma como o Middleware funciona é mostrado na figura 2, onde é mostrado como uma camada entre a aplicação e o sistema operacional, estando presente em todos os nós que compõem o SD, permitindo assim a comunicação entre eles.

Figura 2 – Middleware



Fonte: Tanenbaum e Steen (2007)

2.2.2 Escalabilidade

Por ser um sistema aberto, um SD está sujeito a modificações ao longo do tempo. Novos recursos podem ser adicionados ao sistema, bem como como outros podem deixar de existir, o mesmo vale para usuários. Em um sistema centralizado, onde um único servidor é acessado por todos os usuários, pode haver um gargalo quando houver um aumento na quantidade de acessos. Por outro lado, um Sistema Distribuído é capaz de manter o desempenho mesmo quando há um aumento significativo no número de usuários e recursos.

2.2.3 Segurança

Mecanismos de privilégios de usuário e criptografia são comumente utilizados para garantir a segurança nos Sistemas Distribuídos, com a finalidade de manter os recursos protegidos de acessos indevidos.

2.2.4 Tolerância a Falhas

O fato de um sistema estar distribuído em diversas máquinas ao invés de uma só, faz com que o mesmo possua maior tolerância a falhas, pois estas são normalmente parciais, quando apenas um ou mais componentes do sistema falham e não todos. Os Sistemas Distribuídos podem continuar a prover seus serviços mesmo com a falta de um ou mais componentes.

2.2.5 Transparência

Um dos objetivos de um Sistema Distribuído é a realização de acesso a processos e recursos remotos como se fossem locais, ocultando do usuário que tais componentes estão fisicamente distribuídos por vários computadores (TANENBAUM; STEEN, 2007). Os tipos de transparência existentes são:

a) **Transparência de Acesso:** Oculta diferenças na representação dos dados e no modo de acesso dos usuários aos recursos;

b) **Transparência de Localização:** Tem como objetivo não transparecer a localização física dos recursos;

c) **Transparência de Migração:** Oculta a mudança da localização de recursos do sistema;

d) **Transparência de Relocação:** Oculta a mudança da localização de recursos do sistema durante sua utilização;

e) **Transparência de Replicação:** Oculta a existência de múltiplas réplicas de um mesmo recurso;

f) **Transparência de Concorrência:** Oculta o acesso concorrente de dois ou mais usuários ao mesmo recurso compartilhado;

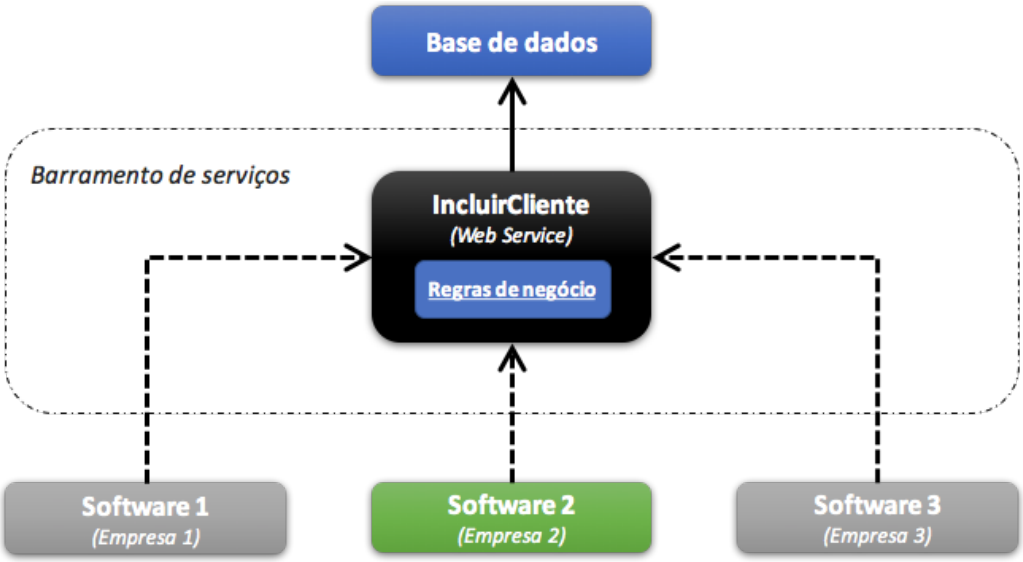
g) **Transparência de Falha:** Oculta a falha que acontece quando um recurso do sistema deixa de funcionar bem e as possíveis recuperações.

2.3 Arquitetura Orientada a Serviços (SOA)

Arquitetura orientada a serviços (SOA), trata-se de uma tecnologia que permite a comunicação entre módulos ou mesmo sistemas distintos, escritos em diferentes linguagens e em diferentes sistemas operacionais, utilizando-se dos protocolos padrões da web (http e https por exemplo). Tal arquitetura permite a troca de informações e integração entre diferentes aplicações web possibilitando o desenvolvimento de sistemas distribuídos e API's (Application Programming Interface) para serem consumidas pelas aplicações (GUEDES,

2017). Os principais modelos utilizados para fazer essa comunicação são SOAP e REST. A figura 3 mostra o esquema de funcionamento do SOA.

Figura 3 – Esquema de um SOA



Fonte: Guedes (2017)

Na imagem acima pode ser observado um exemplo de uso do SOA, no qual uma aplicação se comunica com o banco de dados e fornece o mesmo serviço Incluir Cliente para os softwares das empresas 1, 2 e 3 que se comunicam com a aplicação.

2.4 Interface de programação de aplicações (API)

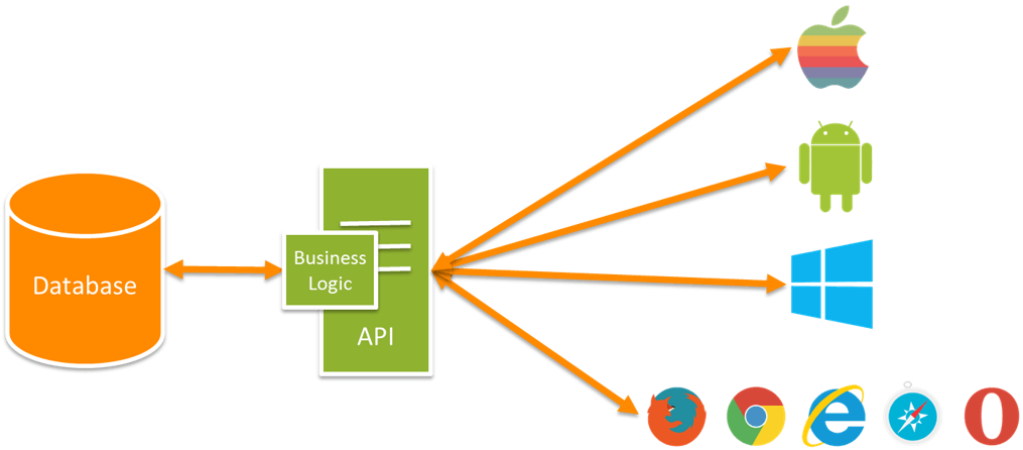
O termo API (Application Programming Interface) em português, interface de Programação de Aplicativos, trata-se de um conjunto de padrões e rotinas de programação para acesso a um aplicativo de software ou plataforma baseado na Web (NASCIMENTO, 2017).

As API's são interfaces criadas para serem consumidas por aplicações, as quais poderão solicitar acesso e obter recursos até mesmo de forma transparente, ou seja, ocultando do usuário final o modo de acesso e a localização do recurso requerido. API's possibilitam a conexão entre sistemas com diferentes tecnologias, como diferentes bancos de dados e linguagens de programação de maneira ágil e segura.

Costuma-se ser criado nas API's uma espécie de portão (gateway de API), pelo qual apenas um conjunto específico de informações estará disponível para ser acessado por terceiros através de requisições, essa informações acessíveis são definidas pelo desenvolvedor da API. Esta é uma forma de garantir a segurança do sistema, pois impede que dados sigilosos possam ser acessados por usuários não autorizados.

Um exemplo de API muito utilizada por aplicações é o Google Maps, que torna possível visualizar de dentro da página que o acessa a localização geográfica de um determinado ponto ou mesmo uma rota para chegar até lá partindo do ponto em que o usuário se encontra. Outra API comumente usada é a dos Correios, que retorna o endereço de um local, através do CEP fornecido pelo usuário, dentro da página acessada. A figura 4 mostra o esquema de funcionamento de uma API no qual a API se comunica com seu banco de dados e fornece serviços e recursos que podem ser acessados por diversas aplicações.

Figura 4 – Esquema de uma API



Fonte: Laboissonniere (2017)

3 MODELO ARQUITETURAL REST

REST, abreviação de "REpresentational State Transfer", em português Transferência de Estado Representacional, não é uma tecnologia, uma biblioteca, nem tampouco uma arquitetura, trata-se de um modelo constituído por um conjunto de princípios e boas práticas que são utilizados para projetar arquiteturas de sistemas distribuídos ou API's, que usam comunicação via rede através do protocolo HTTP.

O REST foi um modelo descrito por Roy T. Fielding, um dos principais criadores de protocolos Web essenciais, como HTTP e URIs, em sua tese de PHD, que é considerada pelos desenvolvedores "A Bíblia do REST", e acabou sendo definido como o modelo a ser utilizado na evolução da arquitetura do protocolo HTTP (TILKOV, 2007).

Foi percebido pelos desenvolvedores que o REST também poderia ser utilizado na implementação de aplicações web, com o objetivo de integrar diferentes sistemas, tornando o REST uma alternativa ao SOAP (FERREIRA, 2017).

Quando uma aplicação ou serviço segue à risca todos os princípios do REST se diz que é uma aplicação RESTful. A seguir será mostrado os 5 princípios fundamentais e como os utilizar da forma correta.

3.1 Identificação de Recursos

Um recurso é uma abstração sobre um tipo de informação gerenciada por uma aplicação, por exemplo, uma aplicação de E-commerce gerencia os seguintes recursos: produtos, clientes, vendas, etc (FIELDING; TAYLOR, 2000). Segundo este princípio do REST, todo recurso deve ter uma identificação única, a fim de que a aplicação possa diferenciar qual recurso deve ser manipulado em uma determinada requisição.

O recurso deve ser identificado na web através de uma URI (Uniform Resource Identifier), em português Identificador de Recursos Universal. Uma URI em REST une o Protocolo (http://) a localização do recurso, URL (ex: dominio.com), o nome do recurso, URN (ex: /produtos/) e pode também possuir o ID de um recurso específico (/135). Algumas boas práticas no uso das URI's são:

3.1.1 Uso de URI's legíveis

As URI's são as portas de acesso aos recursos da aplicação e devem ser legíveis por humanos a fim de que facilitem o entendimento dos clientes que utilizarão o serviço, além de dispensar documentações extensas (FERREIRA, 2017).

Exemplos de URIs:

- http://example.com/customers/1234
- http://example.com/orders/2007/10/776654
- http://example.com/products/4554
- http://example.com/processes/salary-increase-234

3.1.2 Utilização de um padrão único nas URI's

Outra boa prática REST aconselha a manter o mesmo padrão de nomenclatura nas URI's de todos os recursos da aplicação, a fim de tornar o acesso intuitivo (FERREIRA, 2017). Deve-se evitar situações como estas:

- http://loja.com.br/produto (Singular);
- http://loja.com.br/clientes (Plural);
- http://dominio.com.br/processosAdministrativos (Camel Case);
- http://dominio.com.br/processos-judidicais (kebab case);

3.1.3 Evitar a adição da operação a ser realizada na URI

As operações de CRUD nos recursos que uma aplicação gerencia devem ser feitas utilizando os métodos HTTP, como diz um dos princípios REST. Portanto deve-se evitar a utilização de operações na definição de uma URI (FERREIRA, 2017), como por exemplo:

- http://dominio.com.br/produtos/cadastrar;
- http://dominio.com.br/clientes/40/excluir;
- http://dominio.com.br/vendas/25/atualizar.

3.1.4 Evitar modificações nas URI's

Modificações nas URI's certamente impactarão os clientes que utilizam o serviço, pois houve alteração na porta de acesso a um determinado recurso utilizado pelo cliente, portanto se deve evitar ao máximo alterações nas URI's. Em casos críticos nos quais a alteração for realmente necessária, deve-se verificar a possibilidade de manter a URI antiga fazendo um redirecionamento para a nova URI e se não for possível notificar os clientes do serviço previamente (FERREIRA, 2017).

3.1.5 Evitar a adição do formato de representação do recurso na URI

Comumente os serviços REST suportam vários formatos de representação de recursos tais como JSON, XML e HTML. O formato não deve ser especificado na URI de um recurso (FERREIRA, 2017). Esta informação deve ser feita através do Content Negotiation, portanto deve-se evitar URI's como estas:

- http://servicorest.com.br/produtos/xml;
- http://servicorest.com.br/clientes/112?formato=json.

3.2 Vinculação de Recursos

Segundo este princípio do REST, sempre que for possível deve-se utilizar links para referenciar os recursos que possam ser identificados. O uso de links é comumente utilizado nas aplicações web como uma forma de navegar entre suas diversas funcionalidades e recursos oferecidos. Os sites costumam possuir menus de navegação contendo hiperlinks, com a finalidade de guiar os usuários até a funcionalidade ou recurso que procura (FERREIRA, 2017).

Dentro deste princípio, é encontrado o conceito de HATEOAS (Hypermedia As The Engine Of Application State), que significa em português Hipermídia como mecanismo do estado da aplicação. Como exemplo de aplicação pode-se citar um E-commerce, no qual o cliente navega entre seus links até ser conduzido a um determinado produto que procurava. Caso o produto esteja em estoque o site exibe o botão de compra, caso contrário apresenta um botão de notificar quando o produto estiver disponível. HATEOAS é portanto o uso de hipermidia juntamente com a utilização de links como forma de guiar o usuário ao estado atual dos recursos e suas possíveis transições no momento (FERREIRA, 2017).

3.3 Comunicação sem Estado (Stateless)

Segundo Fielding e Taylor (2000) ser stateless é a capacidade que um servidor tem de processar requisições de um cliente sem precisar usufruir de nenhum dado já previamente guardado nele. Ou seja, toda informação que o servidor necessita para compreender e processar deve ser enviada. Logo, quem deve manter o estado não é o servidor, e sim o cliente.

Comunicar sem manter o estado da comunicação do cliente é uma maneira de prover maior escalabilidade a uma aplicação, pois se o estado do cliente for mantido no servidor, a quantidade de clientes que podem interagir com a aplicação no mesmo instante será consideravelmente afetada, pois recursos físicos como memória e disco são limitados (FIELDING; TAYLOR, 2000).

Desenvolvedores costumam armazenar dados de autenticação/autorização em sessão, pois essa é a maneira mais comum ao se desenvolver uma aplicação web tradicional. No entanto, segundo os princípios REST, esta não é a maneira mais correta de se implementar tal requisito de segurança. A utilização de tokens de acesso, gerados pelo serviço, é a melhor solução para este problema, pois são armazenados pelos clientes, via cookies ou HTML 5 Web Storage, sendo também enviados a cada nova requisição ao serviço (FERREIRA, 2017).

3.4 Utilização dos Métodos HTTP para Manipular Recursos

Um cliente que faz requisições HTTP a um serviço via URI, deve manipular os recursos que acessa utilizando os métodos padrões do protocolo HTTP. É possível criar, excluir, atualizar entre outras opções, informando o URI do recurso e o tipo de operação que deseja realizar. Os métodos GET, POST, DELETE e PUT são geralmente os mais usados nas aplicações. A tabela 1 mostra os principais métodos HTTP e seus respectivos cenários de utilização:

Tabela 1 – Principais métodos HTTP e seus respectivos cenários de utilização

Método	**Utilização**
GET	Obter os dados de um recurso.
POST	Criar um novo recurso.
PUT	Substituir os dados de um determinado recurso.
PATCH	Atualizar parcialmente um determinado recurso.
DELETE	Excluir um determinado recurso.
HEAD	Utilizado apenas para se obter os cabeçalhos de resposta
OPTIONS	Obter quais manipulações podem ser realizadas em um determinado recurso.

Fonte: Ferreira (2017)

A tabela 2 mostra exemplos de como se pode aplicar os métodos HTTP em um serviço REST. Para exemplificar é utilizado um recurso chamado cliente.

Tabela 2 – Exemplo de aplicação dos métodos HTTP em um recurso

Método	**URI**	**Utilização**
GET	/clientes	Recuperar os dados de todos os clientes.
GET	/clientes/id	Recuperar os dados de um determinado cliente.
POST	/clientes	Criar um novo cliente.
PUT	/clientes/id	Atualizar os dados de um determinado cliente.
DELETE	/clientes/id	Excluir um determinado cliente.

Fonte: Ferreira (2017)

3.5 Suporte a Diferentes Representações

O suporte a múltiplas representações em um serviço REST é considerada uma boa prática, pois torna-se mais fácil a inclusão de novos clientes. Suportar apenas um tipo de formato limita a aplicação, pois esta não poderá se comunicar com clientes não adaptados ao formato específico do serviço. A maioria das aplicações REST suportam os formatos: JSON, XML e HTML, que atualmente são os três formatos principais (FERREIRA, 2017).

Ao suportar mais de uma representação, uma aplicação REST espera que o cliente especifique em qual formato deseja que seja feita a representação do recurso requerido. Essa negociação do formato é chamada de **Content Negotiation** e deve ser feita através de um cabeçalho HTTP, chamado de **accept** (TILKOV, 2007).

O exemplo abaixo mostra uma requisição HTTP a um recurso especificando o formato XML:

GET / customers/1234 HTTP/1.1

Host: example.com

Accept: application/vnd.mycompany.customer+xml

4 A ABORDAGEM PWA

Grande parte dos acessos aos serviços web atualmente são realizados através de dispositivos móveis. Justamente por isso, tornou-se uma prática muito comum realizada por desenvolvedores de aplicações web conduzir seus usuários à instalação de uma app mobile através de seus smartphones, a fim de promover a seus clientes uma experiência mais completa, podendo lhes enviar notificações (notificações push), oferecer suporte offline, entre outras funcionalidades (ANDERSON, 2018).

Mas seria essa a melhor opção, tendo em vista que grande parte dos usuários, por diversos motivos como a falta de espaço em seus dispositivos ou comodidade desistem de instalar um aplicativo que não tem uma funcionalidade primordial? Por exemplo: a maioria dos clientes não mantém em seus smartphones aplicativos de e-commerce, pois para grande parte este aplicativo só será útil no momento de uma compra naquela loja (ANDERSON, 2018).

A abordagem PWA (Progressive Web App) é uma forma de promover ao usuário uma experiência mais completa, muito próxima a de um aplicativo mobile, combinando os recursos oferecidos pelos navegadores modernos com os benefícios da experiência móvel sem a necessidade de instalação de um aplicativo(ANDERSON, 2018) (MATHEUS, 2018). As principais funcionalidades oferecidas por esta tecnologia são:

- Notificações push;
- Ícone na tela home do smartphone;
- Splash screen;
- Processos rodando em background;
- Suporte a funcionamento em modo offline;
- Atualização automática;
- Acesso à câmera e galeria;
- Acesso à geolocalização;
- Acesso aos contatos.

Pode-se citar como casos de sucesso de PWA's: Flipkart (maior e-commerce da Índia) e Alibaba (maior e-commerce da China).

4.1 Vantagens das PWA's

A principal vantagem dos PWA's em relação aos aplicativos mobiles é a possibilidade de resolver o problema do cliente no momento em que ele acessa o serviço, sem precisar instalar um aplicativo, evitando assim perda de tráfego e aumentando a conversão. Além disso um PWA é acessível a qualquer dispositivo ou sistema operacional, e por ser responsivo se ajusta a qualquer formato de tela, reduzindo assim os custos com desenvolvimento para vários tipos de dispositivos diferentes. Segundo Fernandes (2018), pode-se ainda citar outras vantagens em relação às aplicações nativas como:

- Utilização de linguagens web como HTML, CSS, Javascript;
- Poucas alterações no código fonte do site;
- Envio de notificações push;
- Suporte à utilização offline;
- Acesso à API's nativas como câmera, microfone, geolocalização, etc;
- Aplicações mais leves (em média menos de 1MB);

4.2 Pontos Fracos

Por outro lado os PWA's, ainda não são compatíveis com alguns recursos que apenas aplicativos mobile acessam, como Near Field Communication (NFC) e bluetooth, e por ser uma aplicação web a interface com o usuário não fica tão fluida quanto um aplicativo mobile, razão pela qual nem sempre os PWA's serão a melhor opção para todos os serviços web, pois isso irá depender dos requisitos exigidos pela aplicação. Ainda segundo Fernandes (2018) se pode citar:

- Ausência de recursos requeridos pela aplicação em alguns navegadores;
- Não é possível incluí-los nas lojas de aplicativos;
- Interface web pode sofre perda de performance em aplicações mais robustas;
- Podem não passar a mesma legitimidade de uma aplicação mobile;
- Sem acesso à alguns recursos mobile como sensores, comunicação com outros apps, vibração, etc.

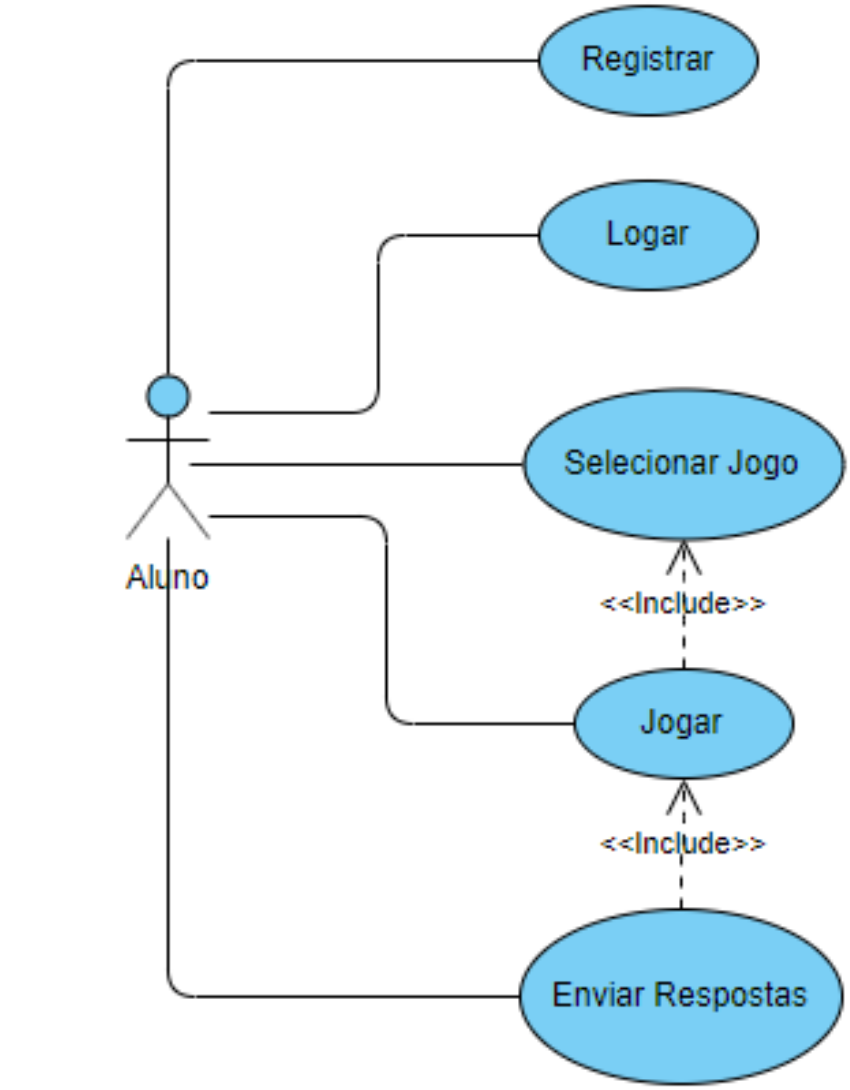
5 ESTUDO DE CASO: GREAT - MÓDULO DE RESOLUÇÃO DE TAREFAS OFFLINE

5.1 Modelagem da Aplicação

Para modelar a aplicação será usada a linguagem UML, padrão na elaboração da estrutura de projetos de software. A UML é utilizada para modelar um software ou aplicação de forma padronizada. Possui diagramas que auxiliam no "desenho" da aplicação, e tem como objetivo o entendimento e documentação da mesma.

Na figura 5 é mostrado o diagrama de caso de uso do módulo desenvolvido.

Figura 5 – Diagrama de caso de uso (Módulo Playground).



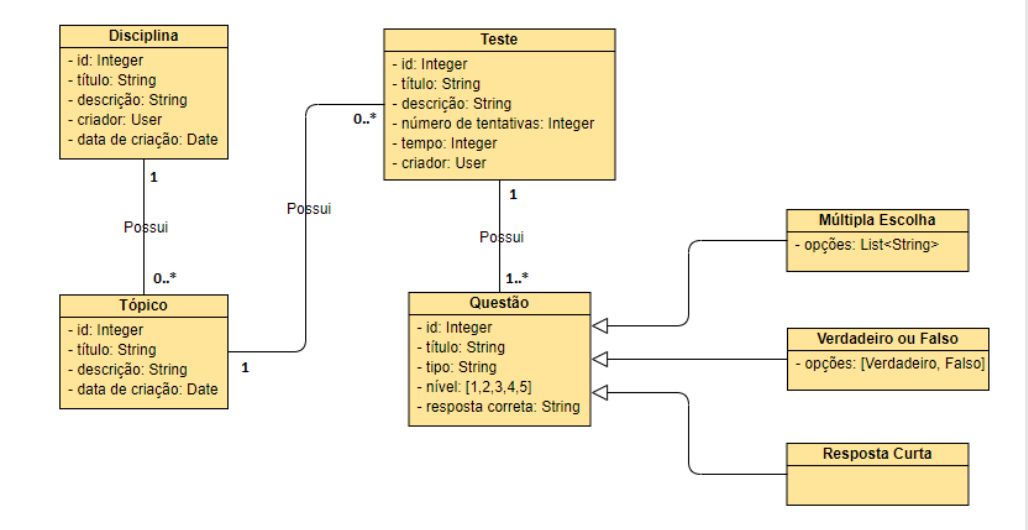
Fonte: O autor

- **UC01 – Registrar:** Caso não tenha registro na aplicação, o aluno se registra inserindo nome de usuário e senha.
- **UC02 – Logar:** Caso o aluno já possuia registro na aplicação, ele insere o nome de usuário e senha afim de fazer sua autenticação para acessar o ambiente interno da aplicação.
- **UC03 - Secionar Jogo:** O aluno seleciona uma das disciplinas disponíveis, em seguida a aplicação exibe os tópicos disponíveis na disciplina selecionada e, após clicar no tópico desejado é exibido os títulos dos testes disponíveis. O aluno então escolhe o questionário que deseja responder e é conduzido para a próxima função.

- **UC04 - Jogar:** A aplicação exibe o teste, onde o aluno seleciona as questões que deseja responder e digita sua resposta num campo de texto ou seleciona a resposta escolhida dentre as opções disponíveis, de acordo com o tipo da questão.
- **UC05 - Enviar Respostas:** Após escolher suas respostas o aluno clica no botão "Enviar Respostas" e o aplicativo envia as respostas à ferramenta GREAT através de uma requisição por meio do protocolo HTTP, em seguida é exibida uma tela de confirmação das respostas enviadas.

Na Figura 6 é apresentado o diagrama de classes do módulo desenvolvido. Apresentando as classes Disciplina, Tópico, Teste e Questões, as quais podem ser do tipo Múltipla Escolha, Verdadeiro ou Falso ou do tipo Resposta Curta.

Figura 6 – Diagrama de classes do módulo playground da ferramenta GREAT.



Fonte: O autor

A aplicação não usa banco de dados próprio para guardar as informações, pois visa obter os recursos através de requisições GET à API do GREAT usando o protocolo HTTP e seguindo o modelo REST. Tais informações são guardadas em documentos JSON como mostrado nos códigos abaixo.

```
1 [
2     {
3         "id": "0",
4         "titulo": "Matematica",
5         "descricao": "descricao a sobre a disciplina de matematica",
6         "criador": "usuario x",
7         "data de criacao": "DD/MM/AAAA",
8         "topicos":
9         [
```

```
            {
                "id": "0",
                "titulo": "Equacoes",
                "descricao": "descricao a sobre ao topico de Equacoes",
                "data de criacao": "DD/MM/AAAA",
                "testes":
                [
                    {
                        "id": "0",
                        "titulo": "equacoes simples",
                        "descricao": "descricao do questionario",
                        "numero de tentativas": "3",
                        "tempo": "5",
                        "criador": "usuario x"
                    },
                    {
                        "id": "1",
                        "titulo": "equacoes do 2 grau",
                        "descricao": "descricao do questionario",
                        "numero de tentativas": "3",
                        "tempo": "5",
                        "criador": "usuario x"
                    }
                ]
            }
        ]
    }
]
```

Este primeiro documento contém as disciplinas, tópicos e seus respectivos atributos juntamente com os testes disponíveis.

```
{
    "id": "2",
    "titulo": "Computacao > Geral",
    "descricao": "Perguntas de assuntos gerais de multipla escolha..",
    "numero de tentativas": "3",
    "tempo": "5",
    "criador": "usuario x",
    "questoes":
    [
        {
            "id": "3",
            "titulo": "O trecho de codigo a seguir: ' x=3+2; if(x==5)",
            "tipo": "MultiplaEscolha",
            "nivel": "1",
            "resposta correta": "6",
            "opcoes":
            [
```

```
18                  "5",
19                  "6",
20                  "4",
21                  "7"
22              ]
23          },
24          {
25              "id": "4",
26              "titulo": "O techo de codigo ' for(i=0;i<=10;i+=2)",
27              "tipo": "VerdadeiroOuFalso",
28              "nivel": "1",
29              "resposta correta": "Falso",
30              "opcoes":
31              [
32                  "Verdadeiro",
33                  "Falso"
34              ]
35          },
36          {
37              "id": "5",
38              "titulo": "Quem foi o criador da ling C e do sistema UNIX ?",
39              "tipo": "respostaCurta",
40              "nivel": "1",
41              "resposta correta": "Dennis Ritchie",
42              "opcoes":[""]
43          }
44      ]
45  }
```

Este segundo documento possui o teste selecionado pelo usuário juntamente com as opções e demais atributos.

5.2 Tecnologias Utilizadas no Desenvolvimento da Aplicação

Para que o aluno possa baixar o jogo de sua escolha na ferramenta GREAT e conseguir jogar em modo offline para posteriormente salvar seu progresso no banco de dados da aplicação, foi desenvolvido o módulo playground, que permite que permite que o aluno vença a barreira da falta de conexão com a internet, o que muitas vezes impede ou atrapalha nos estudos e na execução de tarefas durante a vida acadêmica.

Este protótipo foi desenvolvido utilizando o framework web Flask, escrito em Python e baseado na biblioteca WSGI Werkzeug e na biblioteca de Jinja2. Foi utilizado também os princípios REST para estabelecer a comunicação via internet com a API da ferramenta GREAT, através dos métodos do protocolo HTTP para a obtenção e envio de recursos.

Para o funcionamento da aplicação offline foi utilizado um service work em javascript que permite que as páginas web e os recursos sejam armazenados no navegador via cache.

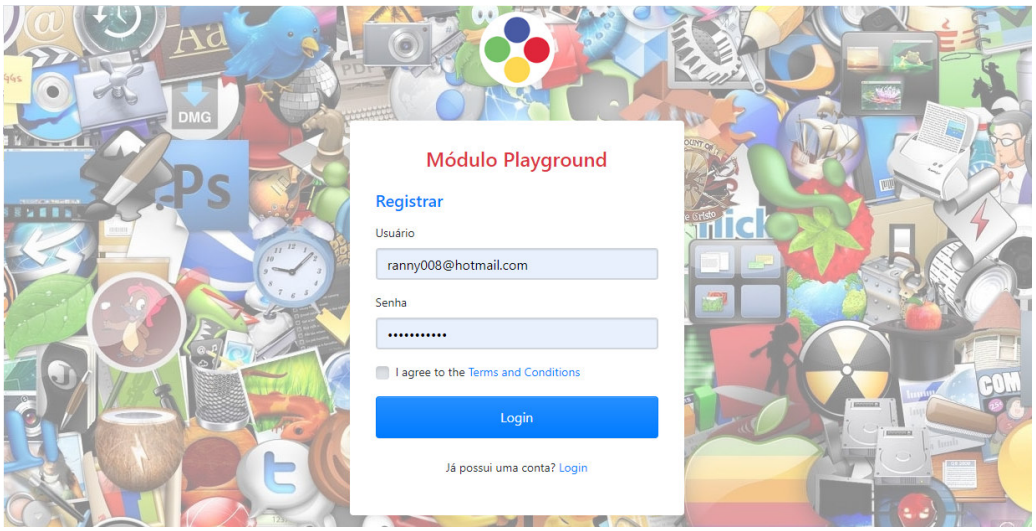
Foi utilizada também a framework Bootstrap no desenvolvimento do front-end da aplicação com a finalidade de deixar o sistema responsivo, de forma que se adapte à tela de qualquer dispositivo.

5.3 Funções da Aplicação de Interfaces Externas

A aplicação possui as funções abaixo:

a) **Registro:** Permite ao aluno se registrar na aplicação informando nome de usuário e senha, conforme mostra a figura 7.

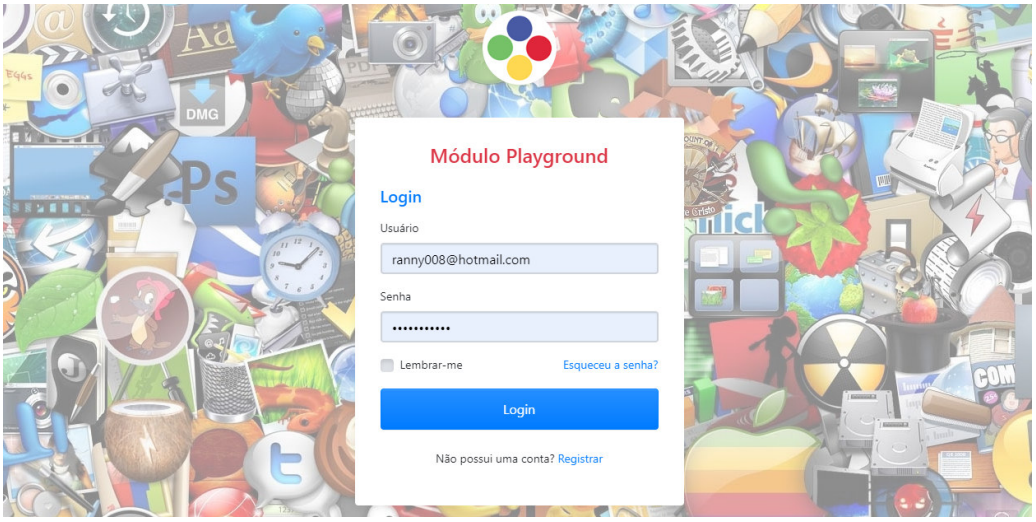
Figura 7 – Tela de registro da aplicação web.



Fonte: O autor

b) **Login:** Solicita que o aluno faça a autenticação para poder acessar o ambiente da aplicação fornecendo nome de usuário e senha, como mostra a figura 8.

Figura 8 – Tela de login da aplicação web.



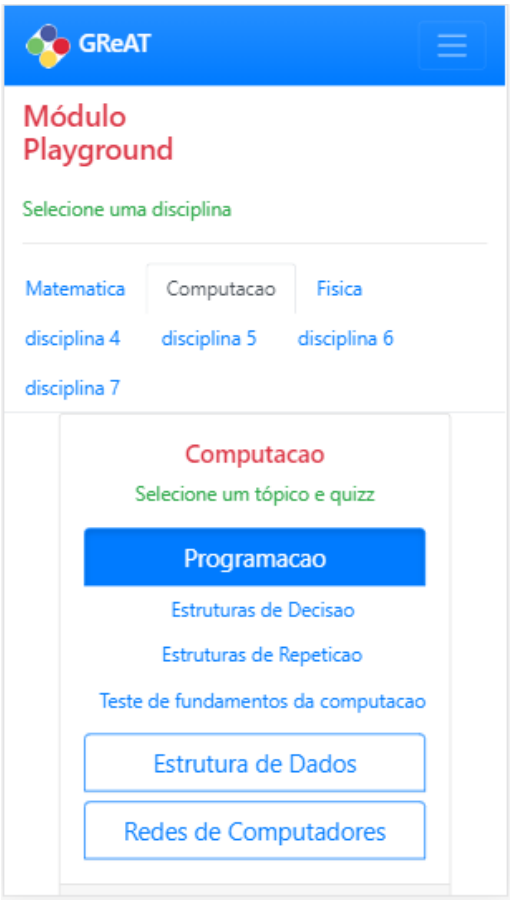
Fonte: O autor

c) **Seleção de jogos:** Permite que o aluno escolha o jogo que deseja jogar através de um menu de navegação onde ele seleciona primeiramente a disciplina, em seguida o tópico e por fim o questionário. A aplicação faz uma requisição à API do GREAT, onde obtém as disciplinas, tópicos e seus respectivos questionários que estão disponíveis para serem jogados pelos alunos. As figuras 9 e 10 mostram a interface com o usuário na tela de um desktop e de um smartphone, onde é feita a seleção do jogo.

Figura 9 – Interface de seleção de jogos em tela de desktop

Fonte: O autor

Figura 10 – Interface de seleção de jogos em tela de smartphone



Fonte: O autor

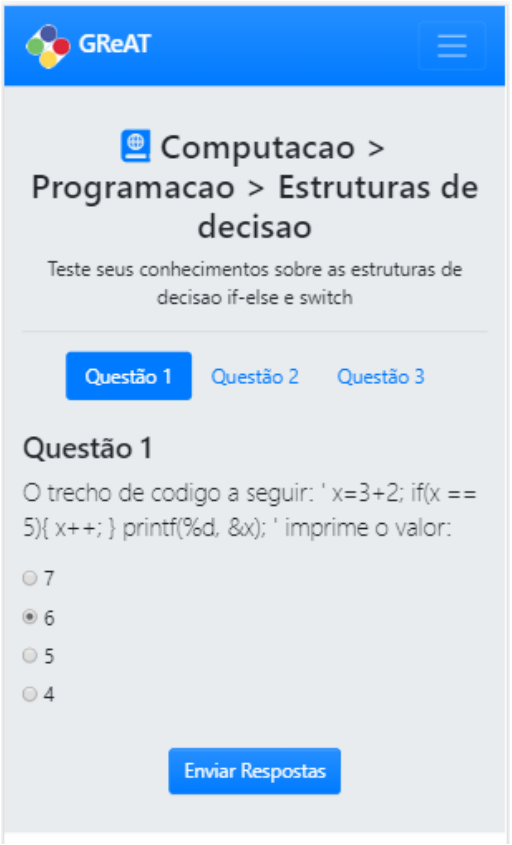
d) **Funcionamento offline:** A aplicação utiliza um service work em javascript para fazer o armazenamento das páginas web e recursos do sistema na memória cache do navegador e assim permitir o funcionamento offline.

e) **Execução dos jogos:** A aplicação permite que o aluno jogue, mesmo sem conexão com a internet, exibindo as perguntas do questionário uma a uma de forma aleatória e também as alternativas para cada pergunta apresentada, também de forma aleatória. O aluno tem a opção de responder cada pergunta na ordem que desejar. As perguntas são do tipo múltipla escolha, verdadeiro ou falso e resposta curta. O teste pode conter perguntas de todos estes tipos juntos. As figuras 11, 12, 13 e 14 mostram a interface com o usuário onde o jogo é exibido na tela de um desktop e de um smartphone.

Figura 11 – Tela de exibição de pergunta do tipo múltipla escolha em desktop

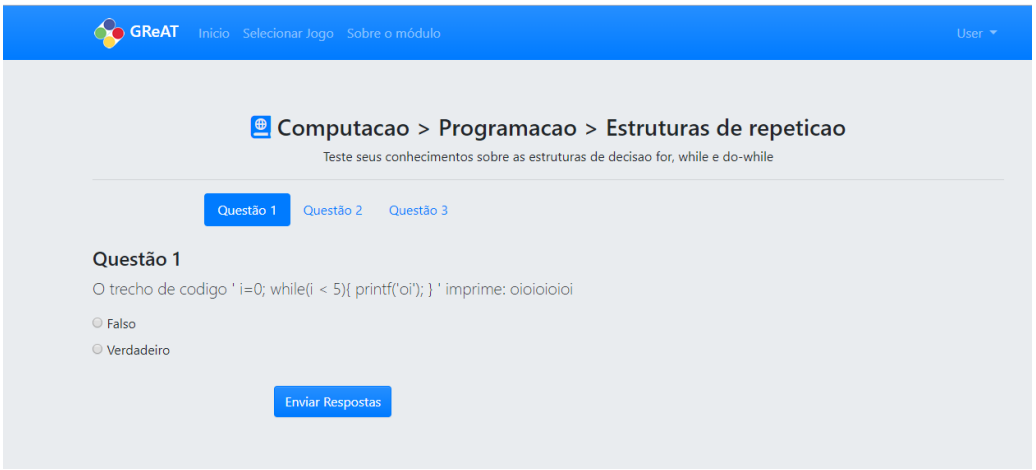
Fonte: O autor

Figura 12 – Tela de exibição de pergunta do tipo múltipla escolha em smartphone



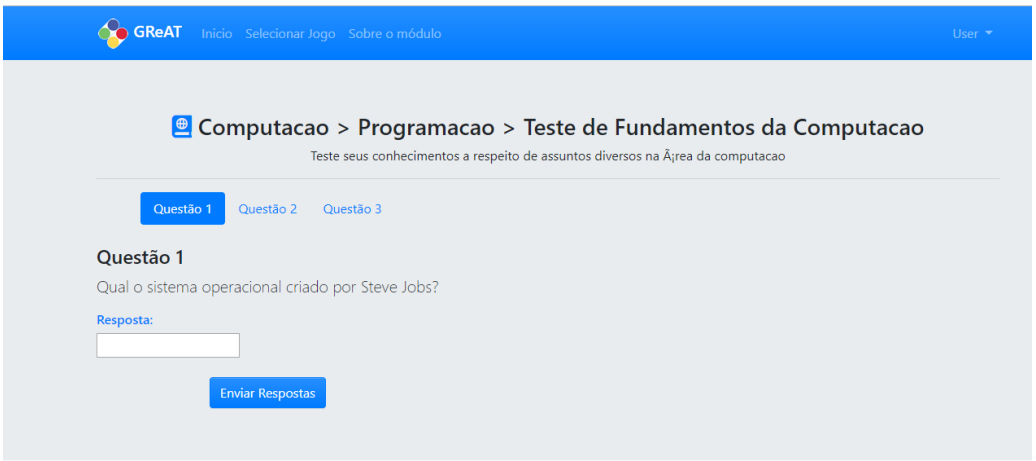
Fonte: O autor

Figura 13 – Tela de exibição de pergunta do tipo verdadeiro ou falso em desktop



Fonte: O autor

Figura 14 – Tela de exibição de pergunta do tipo resposta curta em desktop

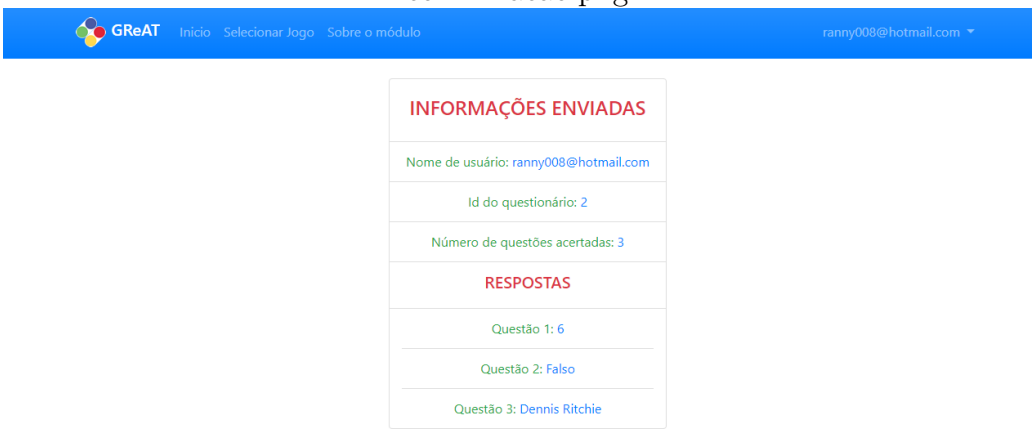


Fonte: O autor

f) **Upload das respostas:** Permite que o aluno envie as opções selecionadas no teste para a GREAT a fim de registrar suas respostas e obter seu resultado. A aplicação faz uma requisição a ferramenta e envia as respostas do aluno por meio do protocolo HTTP. A figura 15 mostra a tela de confirmação que mostra as informações enviadas para a ferramenta GREAT.

Figura 15 – Tela de exibição de informações enviadas

confirmacao.png



Fonte: O autor

6 CONSIDERAÇÕES FINAIS

Os jogos digitais provaram-se como um poderoso aliado para o estudo, aprendizado e treinamento nas mais diversas áreas do conhecimento, além de ser uma ótima opção para promover ao estímulo de importantes funções cognitivas. Os números mostrados em pesquisa falam por si.

Neste contexto, aplicações como o ambiente de autoria de jogos sérios, GREAT, constituem-se como ferramentas fundamentais no desenvolvimento, evolução e inserção na sociedade deste novo tipo de abordagem educacional.

A elaboração deste trabalho abordou conceitos de Sistemas Distribuídos e suas vantagens frente aos sistemas unificados tradicionais, utilizando a arquitetura orientada a serviços implementada através do modelo REST e o uso de API's como forma de tornar uma aplicação expansível e escalável. Foi apresentado também o conceito de PWA e suas características e vantagens que tornam desta abordagem, neste caso, uma alternativa melhor do que o desenvolvimento de uma aplicação mobile.

O módulo desenvolvido acrescenta novas funcionalidades à ferramenta GREAT, e espera-se que mais funcionalidades sejam implementadas e aperfeiçoadas futuramente, tornando o ambiente cada vez mais robusto e agradável aos alunos e professores, para que possa servir no cotidiano como um reforço no aprendizado dos alunos ou mesmo como uma ferramenta para professores de auxílio no ensino e avaliação.

6.1 Trabalhos Futuros

É sugerido como trabalho futuro um mecanismo de busca dos testes, realizando buscas por nome, data de criação, autor, nível de dificuldade, etc. Também é incentivado que seja implementada a funcionalidade de notificação push, para que os alunos sejam notificados a cada vez que um novo teste for adicionado na ferramenta, de forma a aumentar o engajamento dos usuários e deixar a aplicação mais robusta.

Referências

ANDERSON, A. *O que é PWA (Progressive Web App) e porque isso pode aumentar seus resultados mobile*. 2018. <https://vizir.com.br/2017/08/o-que-e-pwa-progressive-web-app-porque-isso-pode-aumentar-seus-resultados-mobile/>. Acesso em: 11-05-2019. 27

COULOURIS, G. F.; DOLLIMORE, J.; KINDBERG, T. *Distributed systems: concepts and design*. [S.l.]: pearson education, 2005. 17

CULTURAMIX. *O Que é um Sistema Distribuído?* 2013. <https://tecnologia.culturamix.com/dicas/o-que-e-um-sistema-distribuido>. Acesso em: 19-10-2018. 17

FERNANDES, D. *PWA: O que é? Vale a pena? Quando utilizar?* 2018. <https://blog.rocketseat.com.br/pwa-o-que-e-quando-utilizar/>. Acesso em: 23-05-2019. 28

FERREIRA, R. *REST: Princípios e boas práticas*. 2017. <https://blog.caelum.com.br/rest-principios-e-boas-praticas/>. Acesso em: 04-03-2019. 22, 23, 24, 25, 26

FIELDING, R. T.; TAYLOR, R. N. *Architectural styles and the design of network-based software architectures*. [S.l.]: University of California, Irvine Doctoral dissertation, 2000. v. 7. 22, 24

GUEDES, M. *ARQUITETURA DE SOFTWARE Você sabe o que é Arquitetura Orientada a Serviços (SOA)?* 2017. <https://www.treinaweb.com.br/blog/voce-sabe-o-que-e-arquitetura-orientada-a-servicos-soa/>. Acesso em: 12-02-2019. 20

LABOISSONNIERE, M. *How to Use APIs: A Painless Introduction + Tutorials*. 2017. <https://snipcart.com/blog/apis-integration-usage-benefits>. Acesso em: 03-02-2019. 21

MACHADO, L. S.; MORAES, R. M.; NUNES, F. Serious games para saúde e treinamento imersivo. *Abordagens práticas de realidade virtual e aumentada*, SBC Porto Alegre, v. 1, p. 31–60, 2009. 14

MATHEUS. *O que é PWA (Progressive Web Apps)?* 2018. <https://www.opus-software.com.br/o-que-e-pwa/>. Acesso em: 11-05-2019. 27

NASCIMENTO, A. *Introdução de Como Funciona uma API*. 2017. <https://alexnascimento.com.br/artigo/introducao-de-como-funciona-uma-api>. Acesso em: 06-12-2018. 20

OLIVEIRA, A.; NETO, C. d. S. S.; TEIXEIRA, M. M. Um ambiente de autoria de jogos sérios pelo usuário final aplicados a educação. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2014. v. 25, n. 1, p. 1058. 14

OLIVEIRA, R. de A.; OLIVEIRA, A.; TEIXEIRA, M. M. Aplicando gamificação na avaliação de aprendizagem de disciplinas de graduação em computação. 14

TANENBAUM, A. S.; STEEN, M. V. *Distributed systems: principles and paradigms*. [S.l.]: Prentice-Hall, 2007. 17, 18, 19

TILKOV, S. A brief introduction to rest. *InfoQ, Dec*, v. 10, 2007. 22, 26