Sistema especialista com temática de jogo para alunos do Ensino Fundamental

Vinicius Emmanuel Teixeira Pinheiro [1]; *Maria Aparecida Reis França* [2]
[1, 2] *Universidade de Uberaba*
viniciusemmanuel@edu.uniube.br; *maria.franca@uniube.br;*

Resumo
Este trabalho tem como objetivo apresentar um *software* desenvolvido, com vistas a proporcionar uma experiência de um jogo ao estudante, percorrendo uma sequência de telas na qual ele possa ser apresentado a situações que envolvam sua emoção e compreensão. Neste *software* foi utilizado o *framework* Adonis Js e a biblioteca React Js, tendo em seu desenvolvimento a predominância da linguagem JavaScript. O usuário irá utilizar o software on-line, absorver os conteúdos apresentados e em relação às questões suas respostas com base nos dados dos especialistas serão contabilizadas e posteriormente compartilhadas com os responsáveis pela instituição de ensino. Todo esse fluxo e requisitos dos parâmetros de análise foram definidos por especialistas e compartilhados para o desenvolvimento, tendo somente transformado esse conhecimento em um programa computacional.

Palavras-chave: Framework; API; Sistema especialista;

1 Introdução

O objetivo geral deste trabalho foi desenvolver um *software* que pudesse conduzir os alunos por uma sequência de telas, em que cada uma retrata um conceito importante sobre sua educação social e emocional. Nestas sequências, comumente chamadas de trilhas, o aluno irá conhecer conceitos e modos, por exemplo de comportamento ou emoção. Sua imersão no *software* será inteiramente como um jogo, tendo como objetivo o trabalho transparente para o aluno.

Para que isso seja possível, o desenvolvimento foi baseados nos documentos fornecidos e a partir deles transformar mais de 400 telas em uma dinâmica interativa com o usuário mantendo a temática de um jogo e com a transparência na coleta de dados relevantes para sua avaliação socioemocional, onde posteriormente esses dados sejam um auxílio para as escolas na formação desse usuário.

2 Metodologia

Sistemas especialistas, em sua construção, são criados para resolver problemas com características definidas por especialistas no campo de atuação. O seu campo de aplicação pode ser qualquer um que necessite da tomada de decisão de um especialista como: decisões financeiras; planejamento logístico e de tráfego; diagnósticos e monitoramento médicos e de máquinas e instalações.

No processo de desenvolvimento utilizou-se a linguagem JavaScript, tanto para a parte de *backend*, quanto no *frontend* da aplicação. A escolha de uma linguagem predominante entre os dois ambientes se deve ao fato de poder obter produtividade e alternância entre os ambientes de desenvolvimento com mais agilidade. Quanto ao desenvolvimento, foi

utilizado os *frameworks* React Js e Adonis Js.

React Js é uma biblioteca para criar interfaces baseadas em componentes, com a facilidade de gerenciar seu estado dentro de cada componente.

Adonis Js é um *framework* para o desenvolvimento *backend* da aplicação, tendo uma estrutura MVC[1], e muito inspirado em outros dois *frameworks* de sucesso, Laravel e Ruby on Rails. Além da linguagem mencionada, foi utilizado para armazenamento das informações o gerenciador de banco de dados Mysql e como servidor *web* o Nginx.

Um dos problemas iniciais foi determinar a tabela de respostas do aluno. Isso porque o sistema teria que atender de forma dinâmica, respostas que contêm somente texto e outras com resposta de única e múltipla escolha. A solução encontrada foi salvar em JSON[2] a resposta do aluno, mesmo na situação em que somente contém texto, para mantermos um padrão. Com isso, é salvo como *key* do JSON, o mesmo nome do tipo da tela. Dentro do JSON, no caso das respostas que o usuário tem que selecionar uma opção, é salvo o Id (identificador) da alternativa ou alternativas selecionadas com *true* e as demais como *false*. No caso de resposta que contém somente texto, é salvo a *key* conforme mencionado anteriormente, mais a resposta do usuário. Com isso, tanto no *frontend* quanto no *backend*, se manteve um padrão e um escopo de código para cada tipo de respostas, juntamente com os cálculos referentes à pontuação.

No *frontend*, foi utilizada a biblioteca React Js, que basicamente possibilita a construção de interfaces *web* somente utilizando javascript e a sintaxe do React.

Na parte de estruturação de pastas, ficou definido em:

- Componentes: fragmentos de html, como *button*, ou *section* com estilos de CSS base já definidos;

- Módulos: conjunto de componentes com regras, validações. Esses módulos abstraem uma parte do que está sendo renderizado na tela, principalmente para manter regras, validações para cada escopo fechado, principalmente para manter uma organização de código que possibilita uma fácil manutenção ou alteração;

- *Template*: são responsáveis pela chamada dos módulos e cada um representa um *background* do projeto com seus estilos próprios;

Para instanciar os módulos em cada *template,* a depender do conteúdo que será apresentado na tela, criou-se uma função para estruturar as informações que vem do retorno do *backend* a cada chamada pelo método *http get*, transformando a chamada dos módulos com os componentes de forma dinâmica. Essa estrutura consiste em um objeto do tipo de caixa de texto, questão e um ou mais objetos de personagens.

Em relação ao *backend*, foi utilizado o *framework* Adonis Js tendo o principal motivo de escolha, dentro de vários *frameworks* da linguagem javascript para

[1] MVC - Model View Controller, é um padrão de arquitetura de software, separando a aplicação em 3 camadas.

[2] JSON - JavaScript Object Notation, é um formato leve de intercâmbio de dados.

o *backend* foi a estrutura e as facilidades para adicionar validadores, *middlewares* e um Orm sólido. Como foi utilizado o conceito de API[3] para sua construção, no servidor utilizamos o JWT [4]para validação e segurança nos request do *frontend*.

Nesse modelo, o JWT contém informações do usuário sendo apenas o Id e informações de Ip, navegador para validação do *token*, sendo uma forma de coibir o sequestro de *tokens*. Além do JWT para validar os usuários, também foi feito uma autenticação para comunicação de servidor para servidor, onde o servidor que executaria o post para API, também terá que enviar sua *key* para validação, além da validação do IP desse servidor. Essa autenticação é realizada, pois, as informações do usuário tanto para seu login, quanto para um novo cadastro, serão realizadas dessa maneira.

Um dos problemas iniciais, foi pensar na dinâmica de interação do usuário, que se resume a escolher uma trilha onde em cada trilha ele poderá responder alguma questão ou simplesmente absorver o conteúdo proposto. A forma adotada para controlar essas interações, além de ser responsabilidade exclusiva do *backend*, foi retornar para o *frontend* as informações necessárias para que pudesse interagir, como por exemplo os botões de voltar ou prosseguir, a URL para o envio da resposta do usuário. Sendo assim, todo o desenvolvimento foi pensando em manter o *frontend* sendo renderizado com as informações que vierem do *backend*. Como a linguagem adotada foi javascript, uma facilidade proporcionada é de trabalhar com dados estruturados em JSON, sendo muito simples para sua manipulação, tanto no *frontend* quanto *backend*. Mesmo sendo somente dados e nada de html ou *blobs*. Sempre houve uma preocupação com a quantidade de informação que iria trafegar, isso porque, mesmo de início, o *software* irá atender poucas escolas, a projeção de escala do projeto é que atenda várias escolas no território nacional.

A estrutura do *backend*, foi separada em:

- Rotas: arquivo onde todas as rotas da aplicação são definidas, com a chamada referente aos *controllers* e seu método;
- *Controllers*: arquivos com regras da aplicação, como, por exemplo, a execução de um método do *model* para carregar dados do banco ou inserir dados no banco;
- *Models*: são os arquivos que referenciam as tabelas no banco de dados e configurações de seus relacionamentos com outras tabelas;
- *Migrations*: são todos os arquivos de criação de tabela no banco de dados. Também é o histórico do *database;*
- *Middlewares*: são funções que são executadas antes da chamada de um *controller* na rota que está inserido. Isso é realizado tanto na autenticação do usuário, quanto na autenticação do servidor;

[3] API - Application Programing Interface, é um conjunto de padrões de programação que permitem a construção de aplicativos e sua utilização.

[4] JWT - JSON Web Token, é um método de autenticação entre duas partes, onde autêntica uma requisição web.

- *Validator*: são funções para validar os dados enviados à API. Também são executados antes da chamada do controller na rota referenciada;

- *Exceptions*: arquivo que centraliza a captura de *bugs* da aplicação e utilizando uma ferramenta de terceiros (Sentry), podemos ver qual parte, horário e erro ocorreu a aplicação.

Toda essa estrutura foi pensada a partir das explicações sobre o que era esperado do projeto e pelas planilhas que foram compartilhadas para servir como base do desenvolvimento. Nessas planilhas estava a representação de cada etapa de cada trilha. Como seria feito o cálculo para os pontos dos usuários nos diferentes tipos de questões e a sequência da trilha que em determinadas etapas ela pode seguir por trajetos diferentes na mesma trilha.

Segundo Weber (2017), com base nos dados de especialistas e transformando seu conhecimento na forma de programa computacional, o *software* contém a capacidade de realizar o processo de inferência nos dados coletados.

De acordo com Buchanan, Barstow e Bechtel (1983 apud WEBER, 2017, p. 23)

> a aquisição do conhecimento pode ser definida com a transferência e transformação do conhecimento especializado, com potencial para a resolução de problemas de alguma fonte de conhecimento para o [Sistema Especialista].

Sistemas especialistas são aplicáveis onde o conhecimento para extrair o resultado é específico para a situação. São sistemas em que as habilidades de especialistas para aplicar uma decisão se resumem na execução do sistema, abstraindo seus conceitos em forma de código. Com isso, temos o sistema desenvolvido, um sistema especialista para extrair o nível sócio emocional de pessoas cursando o Ensino Médio. Toda essa base teórica para abstrair em código foi representada em planilhas nas quais as psicopedagogas definiram suas regras. Toda a avaliação em relação às respostas, conteúdo e questões está registrada nesse documento.

Souza (2017, p. 09) afirma que

> sistemas especialistas normalmente são empregados quando se busca transferir o conhecimento de um humano a sistema computacional, permitindo assim uma posterior emulação do raciocínio inteligente humano.

Além disso, Maher (1987 apud SOUZA, 2017, p. 09)

> explica que sistemas especialistas ou baseados em conhecimento, são sistemas interativos construídos baseados no julgamento, experiência, regras práticas gerais, intuição, entre expertises para prover conselhos e soluções em diferentes assuntos e áreas de conhecimento.

Vale ressaltar que sistemas especialistas baseados em regras são os métodos mais usuais utilizados para a representação de conhecimento (CARRICO *et al*. 1989 apud SOUZA, 2017).

Sistemas Especialistas, de acordo com Feigenbaum (1977 apud BARRELLA,

2000) são sistemas que visam solucionar problemas que, na maioria das vezes, são solucionados somente por especialistas.

Ainda segundo Barrella (2000), os sistemas especialistas são pensados para atender a uma aplicação com processos explícitos e com o conhecimento humano limitado, sendo construídos com a capacidade de tomar decisão como um especialista tendo como referência o conhecimento em sua base de dados provenientes de um especialista.

Podendo ser justificados em

> sistemas que manipulam informações armazenadas em bases de conhecimento e que retratam o raciocínio de especialistas. (ROCKART, 1986 apud BARRELLA, 2000, p. 21).

3 Resultados

Para avaliar o desempenho do software desenvolvido, foi realizado testes na *API* (*backend*), na parte de resposta do usuário. Foi realizado nessa situação por exigir mais dos recursos e ser um fluxo maior onde ocorre validação da resposta e interações com o banco de dados. Para esse teste foi utilizado o programa AB Apache, em um servidor Ubuntu 18.04 com as configurações de processador Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz com 1 núcleo e 1 GB de memória ram, que está na região sul do Brasil no ambiente da Azure. Todos os testes foram realizados em uma máquina separada, estando na região do Triângulo Mineiro.

Foram dois testes nas seguintes configurações:

1° Comando no terminal *bash*: “ab -n 1000 -c 100 http://191.234.160.71/api/v1/request_test/132”, com os resultados:

- Tempo por *request*: 36.504 (ms) (em todos os pedidos simultâneos).

- Porcentagem de solicitações atendidas dentro de um determinado período (ms):
  50% 3399
  66% 3483
  75% 3601
  80% 3788
  90% 4302
  95% 4461
  98% 4621
  99% 4659
  100% 4679 (pedido mais longo)

2° Comando no terminal *bash* ab -n 25000 -c 200 http://191.234.160.71/api/v1/request_test/137

- Tempo por *request*: 50.398 (ms) (em todos os pedidos simultâneos).

- Porcentagem de solicitações atendidas dentro de um determinado período (ms):
  50% 10179
  66% 10733
  75% 11107
  80% 11264
  90% 11688
  95% 11941
  98% 12121
  99% 12353
  100% 20945 (pedido mais longo)

Em todos os testes com a ferramenta AB Apache, a média do recurso de memória ram utilizado pelo Node Js foi de 110 mb, utilizando a ferramenta PM2 (Advanced, Production Process Manager

for Node Js) para monitorar a execução e sem nenhum erro na aplicação.

Além do teste de desempenho, todo o fluxo de resposta do aluno foi construído junto com teste funcionais para garantir a integridade e validar o funcionamento da aplicação.

4 Discussão

Neste trabalho o primeiro desafio foi encontrar uma solução para que os dados da posição do personagem no banco de dados, fosse renderizado conforme seu dado. A sua posição na tela poderia ser ao centro e no canto direito e esquerdo, porém, somente no canto direito, o personagem teria a perspectiva de profundidade diferente dos outros dois que podem aparecer na tela. Para esse problema, foi utilizado o conceito de *flex-box* do Css (Cascading Style Sheets). Essa primeira etapa no desenvolvimento foi decisiva para validar a ideia e o fluxo para a composição dos elementos na tela conforme os dados salvos. Com isso, validou uma solução para a construção de 400 telas dinamicamente.

Conforme mencionado anteriormente, outra parte mais complexa, foi a definição do formato da estrutura para salvar a resposta do usuário. O benefício de uma linguagem predominante e juntamente com a utilização dos *frameworks*, agilizou o progresso de produção do software, com isso sendo entregue dentro do prazo estipulado.

É certo que sem os *frameworks* seria possível realizar tudo o que foi desenvolvido, porém, para um desenvolvimento mais rápido e principalmente com um prazo para a entrega, é essencial levar em consideração a utilização de algum desses recursos para o desenvolvimento. Neste caso, como o domínio do autor é maior no uso da biblioteca React Js e com o *framework* Adonis Js e, principalmente, projeto não havia restrição de uso, optou-se por utilizá-los. Isso não significa que utilizando os *frameworks* o desenvolvedor não necessite de conhecimento do que está sendo desenvolvido e da organização de código, pois a manutenção deste *software* ou adição de funcionalidades, em relação a organização do código, deve estar clara e coesa, para que outros desenvolvedores também consigam desenvolver. Um dos principais questionamentos durante o desenvolvimento era se todo o código escrito estava claro e objetivo.

5 Conclusão

Em relação ao *software* desenvolvido, foi significante ver seu resultado, com a dinâmica dos personagens, as questões e principalmente por ter realizado dentro de um prazo estipulado. Sendo assim, a base de conhecimento em relação ao processo de desenvolvimento de *software* para obter o resultado desejado é de grande importância e principalmente pela manutenibilidade do software e não sendo simplesmente uma codificação de códigos. Além de que nesse tempo, algo novo sempre é aprendido, como por exemplo, simular um ambiente de produção com máquina virtual. Em relação ao tema, percebe-se a importância de cuidar dos fatores sócio emocionais o mais precoce possível. Com o resultado obtido, o projeto tem funcionalidades a serem implementadas futuramente, por exemplo, utilizar inteligência artificial para avaliar as respostas de texto e com isso percebe-se que o resultado, cumpriu o esperado.

Referências

BARRELLA, Wagner Däumicher. **Sistemas Especialistas Modulados e Abrangentes para a Gestão de Operações**. Orientador: Israel Brunstein.

2000. 179f. Tese (Doutorado em Engenharia) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2000.

SOUZA, Danilo Curvelo. **Sistema Especialista Baseado em Regras Ponderando por Tendências Aplicado ao Monitoramento de Processos Industriais**. Orientador: Adrião Duarte Dória Neto. 2017. 101f. Tese (Doutorado em Ciências) – UFRN, Universidade do Rio Grande do Norte, Natal, 2017.

WEBER, Cláudio José. **Desenvolvimento de um sistema especialista para seleção de componentes mecânicos**. Orientador: Gilberto Francisco Martha de Souza. 2017. 256f. Tese (Doutorado em Ciências) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2017.