UNIVERSIDADE FEDERAL DE SANTA CATARINA
TECNOLOGIAS DA INFORMAÇÃO E COMUNICAÇÃO

MATEUS MARTINELLI PAEGLE

GAMEANALYTICSAPP: PROJETO DE UM APLICATIVO PARA ESTUDO DE ANATOMIA HUMANA UTILIZANDO ANALÍTICA DE APRENDIZAGEM PARA JOGOS

Araranguá, 04 de dezembro de 2020

MATEUS MARTINELLI PAEGLE

GAMEANALYTICSAPP: PROJETO DE UM APLICATIVO PARA ESTUDO DE ANATOMIA HUMANA UTILIZANDO ANALÍTICA DE APRENDIZAGEM PARA JOGOS

Trabalho de conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos necessários para obtenção do Grau de Bacharel em Tecnologias da Informação e Comunicação. Sob a orientação do Professor Dr. Robson Rodrigues Lemos.

Araranguá, 04 de dezembro de 2020

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Paegle, Mateus
GAMEANALYTICSAPP : PROJETO DE UM APLICATIVO PARA ESTUDO DE ANATOMIA HUMANA UTILIZANDO ANALÍTICA DE APRENDIZAGEM PARA JOGOS / Mateus Paegle ; orientador, Robson Lemos, 2020.
77 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Tecnologias da Informação e Comunicação, Araranguá, 2020.

Inclui referências.

1. Tecnologias da Informação e Comunicação. 2. Jogos Sérios. 3. Ambiente de Ensino Virtual. 4. Escalabilidade. 5. Analítica de Aprendizagem. I. Lemos, Robson. II. Universidade Federal de Santa Catarina. Graduação em Tecnologias da Informação e Comunicação. III. Título.

MATEUS MARTINELLI PAEGLE

GameAnalyticsApp: PROJETO DE APLICATIVO PARA ESTUDO DE ANATOMIA HUMANA UTILIZANDO ANALÍTICA DE APRENDIZAGEM PARA JOGOS

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Tecnologias da Informação e Comunicação" e aprovado em sua forma final pelo Curso de "Bacharel em Tecnologias da Informação e Comunicação"

Araranguá, 11 de Dezembro de 2020.


Documento assinado digitalmente
Vilson Gruber
Data: 13/12/2020 08:22:18-0300
CPF: 175.317.788-07

Prof. Vilson Gruber, Dr.

Coordenador do Curso

Banca Examinadora:


Documento assinado digitalmente
Robson Rodrigues Lemos
Data: 11/12/2020 16:12:38-0300
CPF: 613.256.949-91

Prof. Robson Rodrigues Lemos, Dr.

Orientador

Universidade Federal de Santa Catarina


Documento assinado digitalmente
Patricia Jantsch Fiuza
Data: 11/12/2020 16:27:59-0300
CPF: 806.421.879-91

Prof.a Patrícia J. Fiuza, Dra.

Universidade Federal de Santa Catarina


Documento assinado digitalmente
Cristian Cechinel
Data: 11/12/2020 17:06:37-0300
CPF: 454.628.449-72

Prof. Cristian Cechinel, Dr.

Universidade Federal de Santa Catarina

“Dedico este trabalho a todos que de alguma forma fizeram parte desta jornada.”
Mateus Martinelli Paegle

AGRADECIMENTOS

“Agradeço a todos que de alguma forma fizeram parte da jornada que é a graduação. Aos meus pais que nunca deixaram de acreditar. A minha esposa Fabiana e a minha filha Maria Laura que sempre estavam lá para incentivar. Ao meu orientador Prof. Robson, um dos grandes responsáveis por este projeto acontecer e a aluna Thais pelas suas contribuições neste projeto.”

Mateus Martinelli Paegle

“Só é útil o conhecimento que nos torna melhores” (Sócrates).

RESUMO

O uso de tecnologias modernas para aplicações Web em conjunto com boas práticas no desenvolvimento de *software* permite criar uma base de código com qualidade e com potencial de escalabilidade para adição de funcionalidades durante o ciclo de desenvolvimento de software. Este trabalho de conclusão de curso tem como objetivo o desenvolvimento de uma base de código utilizando tecnologias Web baseadas em aplicações do tipo *single-page* e melhores práticas no desenvolvimento de *software*, tais como padrões de projeto (*design patterns*). Utilizando como base o projeto do EducaAnatomia3D (2020), desenvolveu-se uma estrutura para uma aplicação móvel onde o jogo sério poderá expandir em termos de possíveis funcionalidades futuras de forma rápida e escalável. Além disso a coleta de informações a respeito da sessão de estudo foi incorporada no projeto possibilitando explorar técnicas de analítica de aprendizagem e visualização dos dados através de painéis de controlo (*dashboards*). Dentro deste contexto, os resultados preliminares do projeto e desenvolvimento de um aplicativo intitulado *GameAnalyticsApp* são apresentados neste trabalho e disponíveis em uma url para acesso via web *browser* para testes durante o ciclo de desenvolvimento do *software*.

Palavras-chave: Jogos Sérios, Ambiente de Ensino Virtual, Escalabilidade, Analítica de Aprendizagem.

ABSTRACT

The use of modern technologies for Web applications along with the best practices in software development allows to create a code base with quality and scalability potential for adding features during the software development cycle. This final project study aims to develop a code base using Web technologies based on single-page applications and best practices in software development, such as design patterns. Based on the EducaAnatomia3D project (2020), a structure for a mobile application was developed where the serious game can expand in terms of possible future features in a fast and scalable way. In addition, the collection of information about the study session was incorporated into the project, enabling the exploration of learning analytics and data visualization techniques through dashboards. Within this context, the preliminary results of the design and development of an application called GameAnalyticsApp are presented and are available in a url for access via web browser for testing during the software development cycle.

Keywords: Serious Games, Virtual Learning Environment, Scalability, Learning Analytics.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 JUSTIFICATIVA

Ao iniciar o desenvolvimento de uma nova aplicação, se faz necessária a definição de estruturas de apoio que auxiliarão o crescimento e manutenção dela. Pensando neste cenário, a utilização de Padrões de Projetos (*Design Patterns*) aliada às boas práticas no desenvolvimento de software manterão o crescimento da aplicação de forma saudável.

Este crescimento saudável também está aliado à utilização de tecnologias atuais, onde é possível o manejo das melhores ferramentas, a fim de proporcionar uma experiência de uso agradável para os usuários.

Ao utilizar a tecnologia como aliada para o ensino, cria-se novas formas de imersão aos conteúdos, que se tornam complexos caso a abordagem tradicional seja empregada.

Para o estudo da anatomia humana, o custo de se manter um laboratório de anatomia é alto. Os procedimentos que precisam ser realizados em um cadáver para mantê-lo em um estado de conservação para utilização como ferramenta de estudos, são elevados. Neste aspecto, é possível a utilização de modelos geométricos 3D para que assim os alunos possam interagir, diminuindo o custo de manutenção de um laboratório.

Aliando o uso de tecnologia com o ensino, se torna viável a coleta dos dados de uso do aluno dentro de um software, para que através da analítica de aprendizagem, seja possível desenvolver informações para identificação da evolução do aprendizado em paralelo com as dificuldades encontradas nos estudos.

1.2 OBJETIVOS

Os objetivos do trabalho estão divididos em objetivo geral e objetivos específicos.

1.2.1 Objetivo Geral

O objetivo geral corresponde ao desenvolvimento de uma aplicação web adotando padrões de projetos que permitam a escalabilidade de aplicações educacionais para analítica de aprendizagem.

1.2.2 Objetivos Específicos

Para se alcançar o objetivo geral, serão realizados os seguintes objetivos específicos:

Adotar tecnologias de desenvolvimento web baseadas em *single-page applications*;

Adotar uma arquitetura cliente-servidor utilizando a abordagem *Frontend* e *Backend*;

Desenvolver o *Frontend* observando padrões de projeto que permitam a escalabilidade da aplicação web para dispositivos móveis;

Desenvolver o *Backend* observando padrões de projeto que permitam a escalabilidade da aplicação;

Projetar a aplicação web para coletar os dados necessários para analítica de aprendizagem;

Disponibilizar arquivos resultantes da coleta de dados para o projeto de painéis de controle (*dashboards*) para analítica de aprendizagem.

1.3 METODOLOGIA

Para o desenvolvimento do Jogo Sério *GameAnalyticsApp*, foi adotada a metodologia de Aprendizagem baseada em jogos digitais (do inglês, *Digital Game-based Learning* - DBGL), que ajuda a superar problemas de aprendizagem.

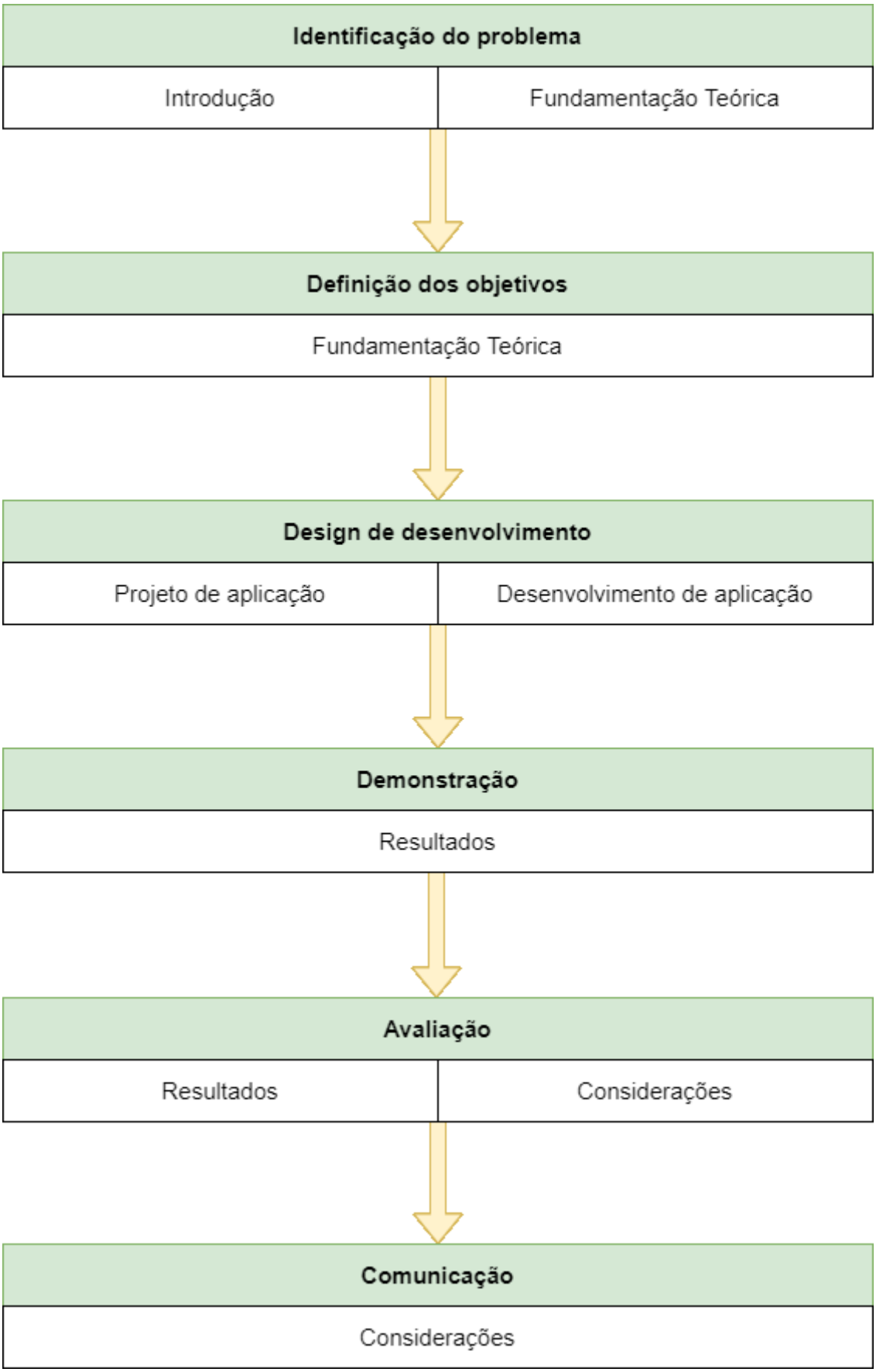
Com o objetivo de se criar uma aplicação de fácil manutenção com possíveis novas funcionalidades, foi desenvolvida uma arquitetura cliente-servidor utilizando tecnologias modernas, mantendo cada peça do sistema bem definida e isolada, de acordo com as melhores práticas do ciclo de desenvolvimento de *software*.

Para o desenvolvimento do protótipo funcional, foi adotada a metodologia *Design Science Research* (DSR), que busca gerar conhecimento sobre os artefatos ou prescrever uma solução. Mesmo sendo uma abordagem orientada para a solução de problemas, o objetivo não é desenvolver uma solução ótima, mas sim, uma solução satisfatória em comparação com as existentes.

Peffers et al. (2007) propôs uma metodologia DSR dividida em seis etapas: identificação da problemática, definição dos objetivos, design e desenvolvimento, demonstração, avaliação e comunicação.

Na Figura 1 é demonstrada a aplicação da metodologia DSR nesta pesquisa, relacionando as etapas da metodologia DSR com a estrutura deste trabalho.

Figura 1 - Aplicação da metodologia DSR no contexto do *GameAnalyticsApp*



Fonte: Elaborado pelo autor

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em seis capítulos, sendo que no presente capítulo é apresentado alguns pontos para o início do desenvolvimento do projeto como: justificativa, objetivos do trabalho e a metodologia utilizada.

No segundo capítulo buscamos estabelecer a fundamentação teórica no sentido de projetar e desenvolver uma aplicação voltada a jogos sérios.

Mais adiante, no terceiro capítulo, apresentamos as ferramentas para construção de uma aplicação web escalável e de fácil manutenção e no capítulo quatro, o projeto para o desenvolvimento e ferramentas necessárias para atingir os objetivos de escalabilidade são apresentadas do ponto de vista tecnologia computacional.

No capítulo cinco, descrevemos o funcionamento da primeira versão funcional da aplicação e no capítulo seis faço minhas considerações a respeito do trabalho, bem como a indicação de possibilidades de evolução para a aplicação.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 JOGOS SÉRIOS NA ÁREA DA SAÚDE

O conhecimento da anatomia é fundamental para a formação das profissões da área da saúde. As metas de ensino de anatomia são: conhecer as estruturas anatômicas e as relações entre elas; reconhecer as estruturas anatômicas por meio de técnicas de imagens; e entender as bases anatômicas da Patologia.

Aprimorar os recursos didáticos aplicados ao ensino de anatomia tende satisfatoriamente para o direcionamento das ações, estimula a participação do aluno como sujeito ativo na busca por novas informações promovendo suporte indispensável ao processo de ensino-aprendizagem (Guiraldes *et al.* 1995).

O ensino de anatomia humana envolve momentos distintos. O primeiro momento é o de exposição de conceitos teóricos e definições dos sistemas e órgãos do corpo humano. O segundo é uma abordagem estuda as características gerais e suas inter-relações, utilizando peças anatômicas e cadáveres em laboratório. Porem a vasta nomenclatura anatômica pode tornar o aprendizado da disciplina complexo levando a uma experiência de aprendizado fraca resultando em uma abordagem superficial para a aprendizagem e até mesmo podendo levar à evasão do curso, com isso, os ambientes virtuais assumem um papel grande no ensino fornecendo um novos ambiente de estudo que não seja o laboratório.

Uma grande vantagem da utilização de ambiente virtuais para o ensino de anatomia está relacionado ao uso de modelos tridimensionais substituindo a peça anatômica cadavérica, pois a preservação de estruturas anatômicas pequenas como os vasos sanguíneos e nervos é um dos grandes desafios dos laboratórios de anatomia.

Ferramentas virtuais e jogos digitais vem conquistando espaço nas aulas teóricas e práticas de anatomia, os quais vem sendo cada vez mais utilizados pelos docentes como ferramenta que desperta um grande interesse do estudante e exclui por muitas vezes os problemas de aprendizagem.

O uso de jogos no contexto educacional pode ser denominado como jogos sérios. Os jogos se tornaram aliados para a sala de aula permitindo que o ambiente educacional saia do ensino tradicional e passe a funcionar dentro da informação.

O uso de jogos sérios para treinar, aprender e executar atividades reais em ambientes virtuais pode melhorar o desempenha dos estudantes, pois possibilita a vivencia de

experiências de aprendizagem produzidas individualmente de acordo com o estilo do estudante (Prikladnicki e Wangenheim 2008).

Ao incorporar o uso de jogos sérios nas disciplinas de anatomia facilita a aprendizagem e combinando modelos geométricos da anatomia humana com *softwares* personalizados pode-se proporcionar aos alunos novas formas de interagir com a anatomia que não poderiam ser alcançadas através de imagens estáticas e modelos anatômicos tradicionais.

Existem inúmeras vantagens em se utilizar jogos digitais para treinamento médico e muitas evidências de resultados positivos utilizando simulações virtuais, jogos e aplicações móveis, melhorando a motivação, participação e gestão de tempo. Os recursos disponíveis continuam aumentando, sendo uma área em constante expansão (MCCOY; LEWIS, 2015), existindo a necessidade de uma pesquisa mais profunda para se avaliar os benefícios educacionais específicos de cada área.

2.2 ANALÍTICA DE APRENDIZAGEM PARA JOGOS SÉRIOS

Com o avanço das tecnologias digitais, a educação buscou atualizar suas metodologias de ensino, em especial a metodologia de jogos sérios.

Os jogos sérios tornam o processo de aprendizagem divertido, desafiador e gratificante fazendo com que o usuário fique envolvido no jogo e não note que está em sistema de ensino-aprendizagem. Entretanto o sucesso no jogo não é garantia de aprendizado, com isso a avaliação do usuário dentro do jogo sério é fundamental para o processo de ensino-aprendizagem.

O método mais comum utilizado para esta avaliação é através de um teste antes e depois que o usuário participa do jogo, porém, este método consome muito tempo e fornece informações limitadas. Com isso a analítica de aprendizagem se torna um método mais eficaz para a avaliação do ensino-aprendizagem.

A analítica de aprendizagem, refere-se à coleta, análise e visualização de uma grande quantidade de dados relacionados aos processos educacionais (Slimani A. et. al. 2018). A extração dos dados educacionais pode ocorrer através de web logs, registros de interações do usuário, mecanismos de rastreamento, localização e detectores de movimentos (Massa S. M.; Kühn F. D. 2018)(Roque, F. et al. 2019). Após a extração, os dados devem ser processados através da mineração de dados e por fim devem ser utilizadas técnicas de visualização de informações para auxiliar na interpretação dos dados e identificação de possíveis padrões.

A analítica de aprendizagem pode ser utilizada como parte de uma abordagem de avaliação, afim de verificar se os jogos estão alcançando bons resultados de aprendizagem, fornecendo dados sobre a interação do usuário com o jogo em tempo real.

A visualização de dados trata-se de uma necessidade essencial dentro de um jogo sério, pois é através da representação dos dados que ocorre a comprovação da validade e eficiência educacional, fornecendo meios para avaliar o conhecimento obtido pelo usuário (Alonso-Fernandez C. et. al. 2017).

A analítica de aprendizagem é uma área de conhecimento emergente que fornece relatórios detalhados sobre a utilização de um jogo baseado em processos de mineração de dados. Schneider e Lemos apresentam uma revisão sistemática da literatura a respeito de Analítica de Aprendizagem para Jogos Sério. A analítica de aprendizagem, refere-se a coleta, análise e visualização de uma grande quantidade de dados relacionados aos processos educacionais e esses dados devem ser processados através de mineração de dados utilizando técnicas de visualização de informações para auxiliar na interpretação dos dados e identificações de padrões. (SCHNEIDER E LEMOS, 2020, p. 150-174).

Normalmente está visualização de dados se dá através de painéis de controle (dashboards) de forma significativa não sobrecarregando de informações, possuindo somente informações que tenham propósito, objetivando uma experiência satisfatória para o usuário. A visualização deve ocorrer de forma significativa, de forma que o painel não fique sobrecarregado de informações e só possua informações que tenham um propósito, para não dificultar a experiência do usuário (Perez-Colado I. J. et. al. 2018). As visualizações de um painel são caracterizadas por apresentarem as atividades realizadas pelos usuários ao longo do tempo.

A coleta da informação pode acontecer ainda (além do pré e pós-teste) através da coleta de informações realizadas pelo usuário durante a partida que pode ocorrer através de registros de acesso, mecanismos de rastreamento, sensores como rastreadores oculares, rastreamento de localização e movimentos detectores.

Dentro do contexto do *GameAnalyticsApp* serão coletados dados para o usuário do tipo estudante e professor. Para o usuário do tipo estudante serão coletados dados referentes ao tempo de cada sessão de estudo por tópicos referentes ao conteúdo do Sistema Cardiovascular, assim como, o desempenho do estudante nos questionários referentes ao conteúdo do Sistema Cardiovascular. Para o usuário do tipo professor, serão coletados dados referentes ao tempo de cada sessão de estudo e desempenho nos questionários dos estudantes por tópicos do Sistema Cardiovascular, pertencentes a uma dada sala virtual. Para possibilitar a análise exploratória

dos dados coletados serão disponibilizados recursos de visualização de dados a partir de diferentes cenários para usuários do tipo estudante e professor.

3 PROJETO DE UMA APLICAÇÃO PARA O ENSINO DE ANATOMIA DO SISTEMA CARDIOVASCULAR

Para o projeto e desenvolvimento de um aplicativo web, inicialmente precisa-se definir como será o seu funcionamento do ponto de vista de uma arquitetura para o sistema computacional e a partir disso, como será sua estrutura com base nas definições de projeto.

Para o projeto e desenvolvimento do *GameAnalyticsApp*, foi adotada uma arquitetura "cliente-servidor". Nas próximas etapas, estão descritas as principais ferramentas adotadas para a criação do aplicativo baseado nesta arquitetura.

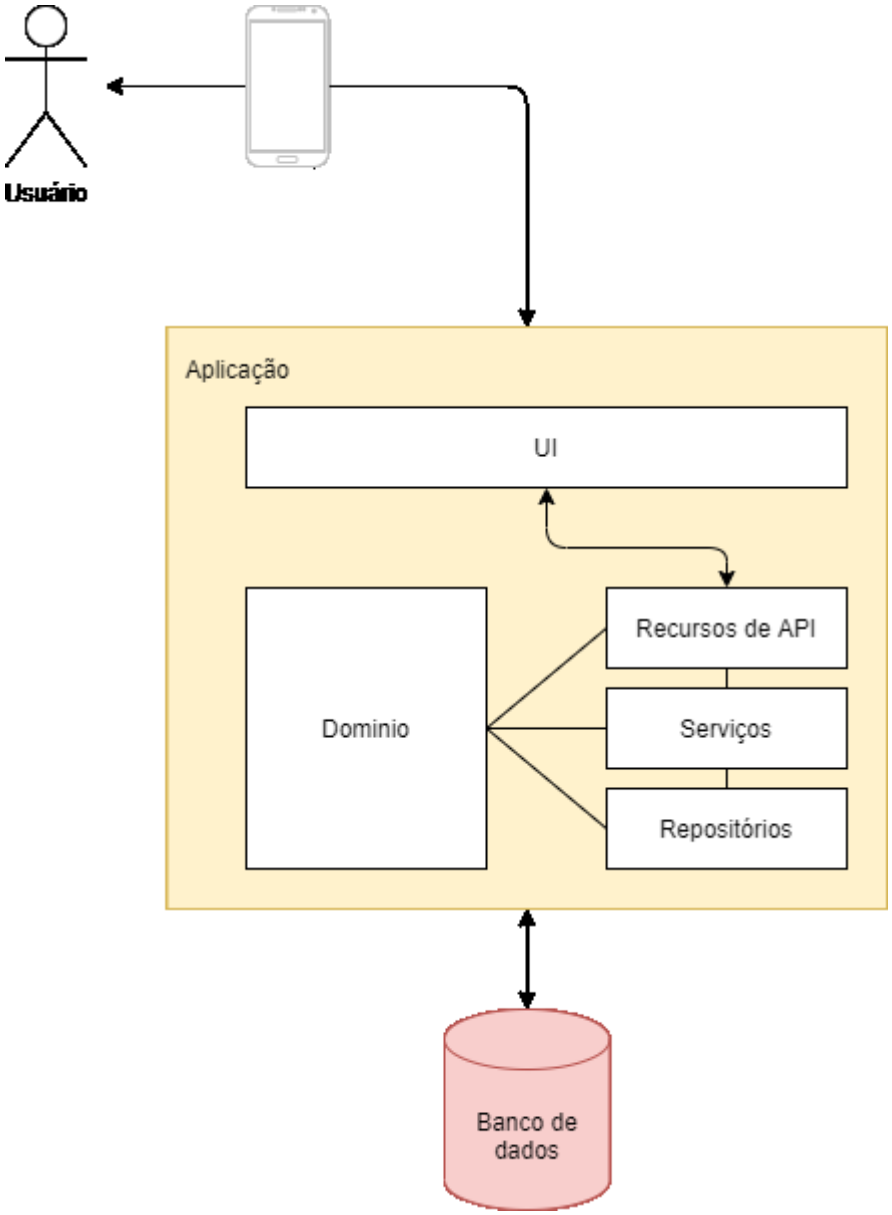
3.1 ARQUITETURA PARA APLICAÇÃO WEB

Para a criação de aplicações web existem dois modelos muito utilizados. São eles: Arquitetura monolíticas e orientadas a micro serviços. Para este projeto adotou-se a arquitetura monolítica.

A arquitetura monolítica é a mais comum e consiste basicamente na obtenção da aplicação centralizada em um único serviço.

Na Figura 2 apresenta-se um possível modelo de funcionamento da arquitetura monolítica:

Figura 2 - Funcionamento de uma arquitetura monolítica

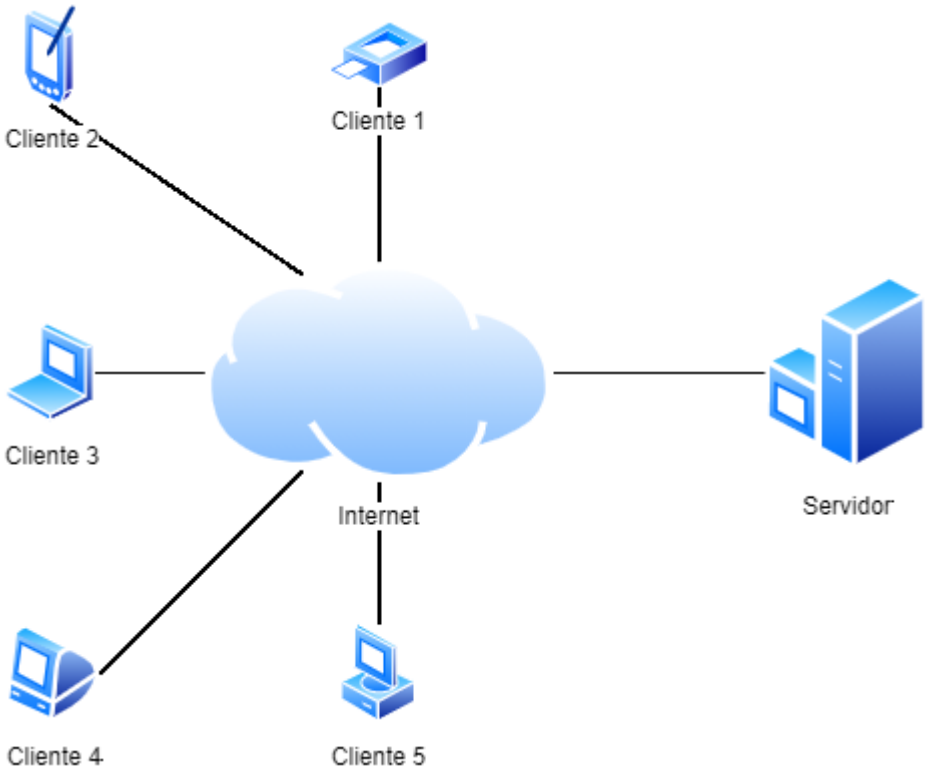


Fonte: Elaborado pelo autor

Como definiu-se que o aplicativo será web, será manejado o modelo cliente-servidor para disponibilizar as aplicações aos usuários.

Neste exemplo abaixo, um cliente solicita um recurso ao servidor, que retorna à informação requisitada. Na figura 3, segue demonstração do funcionamento de um modelo cliente-servidor.

Figura 3 - Modelo Cliente-Servidor



Fonte: Elaborado pelo autor

3.1.1 Tecnologias para o *Backend*

O *Backend* está relacionado ao que acontece por trás da aplicação (por exemplo, requisições de acesso ao banco de dados). Assim como, o *Backend* está associado a todo tipo de tecnologia que fornece estrutura e apoio às ações do usuário.

3.1.1.1 Ambiente para Execução de Procedimentos *Node.js*

O *Node.js* é um ambiente de execução *Javascript server-side*, construído sobre o motor *JavaScript* do Google Chrome, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos. Com o *Node.js* pode-se criar aplicações *Javascript standalone*, não dependentes do browser para execução. (NODE.JS, 2020)

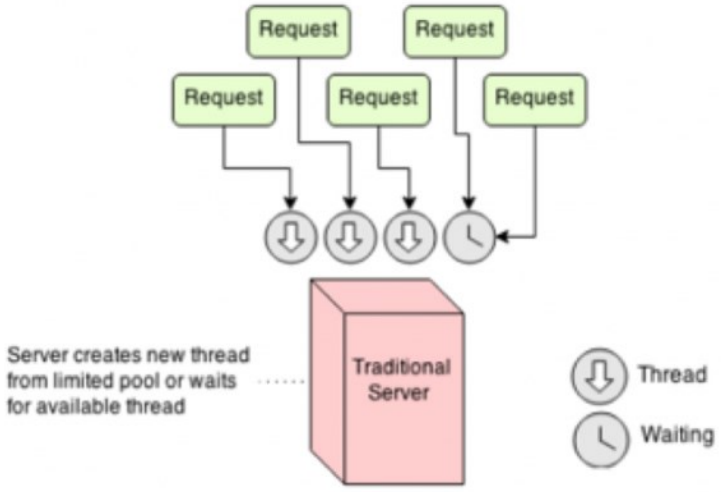
Apesar de ser uma tecnologia relativamente nova, grandes empresas, como Linkedin, Uber e Netflix, já utilizam.

A principal diferença entre *Node.js* (2020) e outras tecnologias do mercado como PHP, Java ou C# é o fato de sua execução ser *single-thread*, ou seja, apenas uma *thread* é

responsável pela execução da aplicação, diferente das tecnologias citadas acima que *são multi-threads*.

No modelo tradicional dos servidores web, a cada requisição recebida pelo servidor, uma nova *thread* é criada para responder a esta requisição. Isso significa que recursos (tais como, memória RAM) serão alocados para atender este processo. Como os recursos são limitados, quando este limite é atingido, as novas requisições terão que esperar a liberação destes para ser atendida. A Figura 4 mostra o modelo tradicional de *web servers*.

Figura 4 - Modelo tradicional de *web servers*.

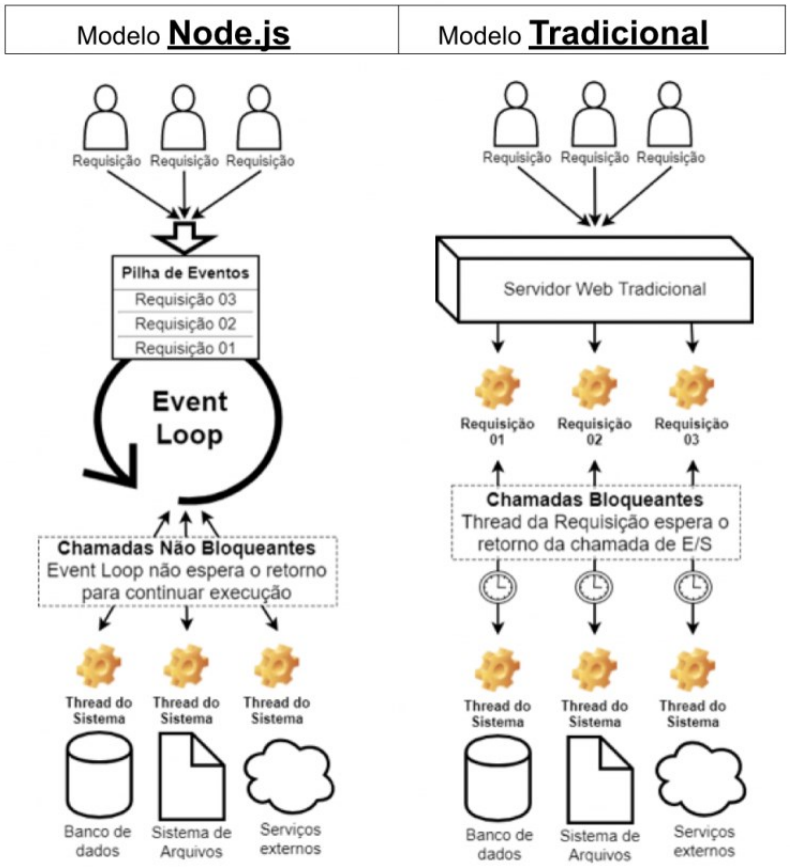


Fonte https://www.opus-software.com.br/node-js/

Já no *Node.js* apenas uma *thread* é responsável por executar as requisições, que é chamada de *Event Loop*. Desta forma, para tratar múltiplas requisições, o *Node.js* conta com o chamado Entrada e Saída (E/S) não bloqueantes, ou seja, as operações de entrada e saída (acesso ao banco de dados ou ao sistema de arquivos, por exemplo) são assíncronas e não bloqueia a *thread* enquanto aguardam a resposta de acesso ao recurso. (NODE.JS, 2020)

Na Figura 5 apresenta-se uma comparação entre a forma de execução com *Event Loop* e o modelo tradicional de servidor web.

Figura 5 - Comparação entre os modelos Node.js e Tradicional



Fonte https://www.opus-software.com.br/node-js

3.1.1.2 O *Framework Express*

A documentação do *framework Express* o define como: "O *Express* é um *framework* para aplicativo da web do *Node.js* (2020) mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel".

O *framework Express* pode ser comparado com o *Laravel* para o PHP. Com o *framework Express* é possível criar abstrações de rotas, *middlewares* e muitas outras funções para facilitar a criação, tanto de aplicações web quanto de *Application Program Interface* (API's). Desta forma, o *framework Express* auxilia no desenvolvimento de uma aplicação para Web. (EXPRESS, 2020)

O processo de instalação do Node.js e Express, pode ser encontrado no ANEXO INo ANEXO I.

3.1.1.3 Exemplo "*Hello world*!"

A partir disso, edita-se o arquivo *package.json*, e na seção *scripts*, adiciona-se a chave *start*, com o valor node index.js, Figura 6.

Figura 6 - Edição do arquivo *package.json*

```
"scripts": {
  "start": "node index.js"
},
```

Fonte: Elaborado pelo autor

Dando continuidade ao procedimento, pode-se criar a primeira aplicação *Express* (2020). Na pasta raiz, o arquivo *index.js* deve ser criado, conforme Figura7:

Figura 7 - Criação da aplicação Express

```
const express = require('express');
const app = express();

app.get('/', (request, response) => {
  return response.send("Hello world!");
});

app.listen(3333);
```

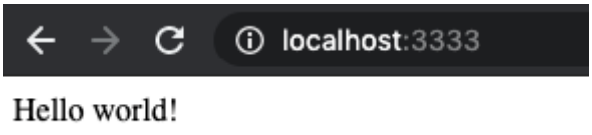
Fonte: Elaborado pelo autor

A partir disso, basta executar a aplicação, no terminal com o comando:

$ npm start

No navegador, é necessário acessar o *link* (http://localhost:3333), e a aplicação retornará com "*Hello world*!", assim como fica demonstrado na Figura 8.

Figura 8 - Exemplo de saída do "*Hello World*"



Fonte: Elaborado pelo autor

3.1.1.4 A ferramenta *express-generator*

O *express-generator* é uma ferramenta de linha de comando disponibilizada pela equipe de desenvolvimento do *Express*. É um gerador de aplicativos *Express* que cria uma estrutura básica para sua aplicação.

Para utilizá-la, o comando abaixo deve ser executado para iniciar a instalação:

$ npm install express-generator -g

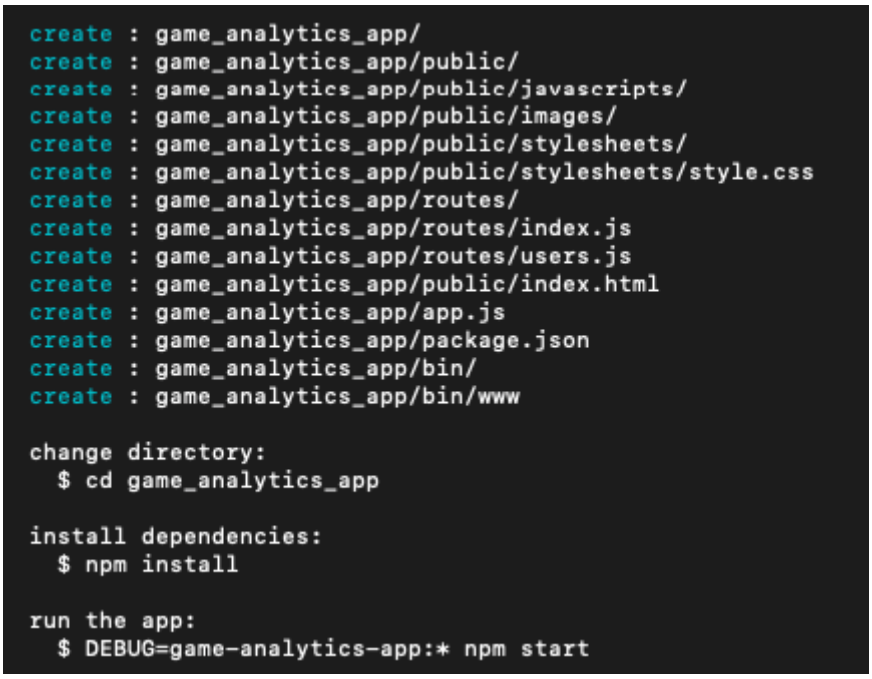
Em seguida emprega-se o comando abaixo para gerar a estrutura base para a aplicação:

$ express --no-view game_analytics_app

A opção --no-view, cria a estrutura base sem uma *template engine*, ideal para se construir api's. Para todas as opções disponíveis, é necessário o comando express -h.

Ao executar o comando de criação, toda a estrutura e arquivos necessários para aplicação *express* será criada. Na Figura 9, tem-se a saída padrão que ele gera:

Figura 9 - Saída padrão da ferramenta express-generator



Fonte: Elaborado pelo autor

Conforme o informado na saída do comando, é necessária instalação das dependências, para isso a pasta criada deve ser acessada para executar o comando npm install.

Após a instalação, o comando "DEBUG=game-analytics-app:* npm start deve" ser inicializado. Para otimizar os processos, é possível editar a chave *scripts.start* no arquivo

package.json, alterando para “DEBUG=game-analytics-app:* node ./bin/www”, conforme Figura 10.

Figura 10 - *Script start* no arquivo *package.json*

```
"scripts": {
  "start": "DEBUG=game-analytics-app:* node ./bin/www"
},
```

Fonte: Elaborado pelo autor

A partir dos passos realizados anteriormente, o comando de inicialização da aplicação se resumiu a npm start.

Executado o npm start, a aplicação será iniciará na porta 3000. A partir disso, no navegador, deve-se informar o *link* (http://localhost:3000), e a saída da Figura 11 estará disponível: (EXPRESS, 2020)

Figura 11 - Mensagem de início de acesso ao *Express*

Fonte: Elaborado pelo autor

3.1.2 Tecnologias para o *Frontend*

Na arquitetura cliente-servidor, existe o chamado *client-side* que envolve linguagens que são processadas pelo navegador do usuário. É fica o “rosto" da aplicação, sendo ela um site ou aplicativo.

Para se desenvolver o *Frontend* algumas linguagens de programação são aplicadas, tais como *JavaScript* e linguagens de marcação como HTML e CSS.

Com a evolução dos computadores e da internet, várias ferramentas auxiliam no desenvolvimento de aplicações ricas em detalhes.

3.1.2.1 Aplicações Web do tipo *single-page*

Uma *single-page application*, é uma aplicação que carrega uma única página HTML e todos os *assets* (como *JavaScript* e CSS) necessários para a aplicação ser executada. Quaisquer interações com a página ou páginas subsequentes não necessitam de outras requisições para o servidor, o que significa que a página não é recarregada.

3.1.2.2 A biblioteca *React* para criação de interfaces

React é uma biblioteca utilizada para a criação de interfaces de usuários (do inglês, *user interfaces* - UIs) interativas. Pode-se criar *views* para cada estado da aplicação. A partir disso, o *React* irá atualizar e renderizar de forma eficiente apenas os componentes necessários na medida em que os dados mudam. (REACT, 2020)

Com o React pode-se criar componentes encapsulados que gerenciam seu próprio estado e combiná-los para criar UIs complexas.

3.1.2.3 A biblioteca *React Router* para declaração de rotas

No desenvolvimento de uma *single-page application*, a primeira coisa a ser definida é como controlar as rotas da aplicação. *React Router* é uma biblioteca completa que permite definir rotas, informado quais os componentes serão renderizados pelo *React*. (REACT ROUTER, 2020)

3.1.2.4 A biblioteca *Axios* para requisições assíncronas

Através do *JavaScript* é possível enviar requisições e obter respostas sem que a página seja recarregada. Este método é conhecido como XMLHttpRequest (AJAX).

O *Axios* é uma biblioteca que facilita a utilização do XMLHttpRequest para realizar essas requisições. (AXIOS, 2020)

3.1.2.5 A Biblioteca CSS *Bootstrap* para definição de estilos de página

O *Bootstrap* é um framework CSS para construção de aplicações, onde é possível obter estilos de componentes já definidos e prontos para serem utilizados.

3.1.2.6 Criando um projeto utilizando *React*, *React Router* e *Axios*

3.1.2.6.1 A ferramenta create-react-app

O create-react-app é uma ferramenta de linha de comando que facilita a criação de uma *single-page application* em *React* (2020).

O *create-react-app* configura um ambiente de desenvolvimento para o uso das funcionalidades mais recentes do *JavaScript*, definindo uma pipeline para o *Frontend*.

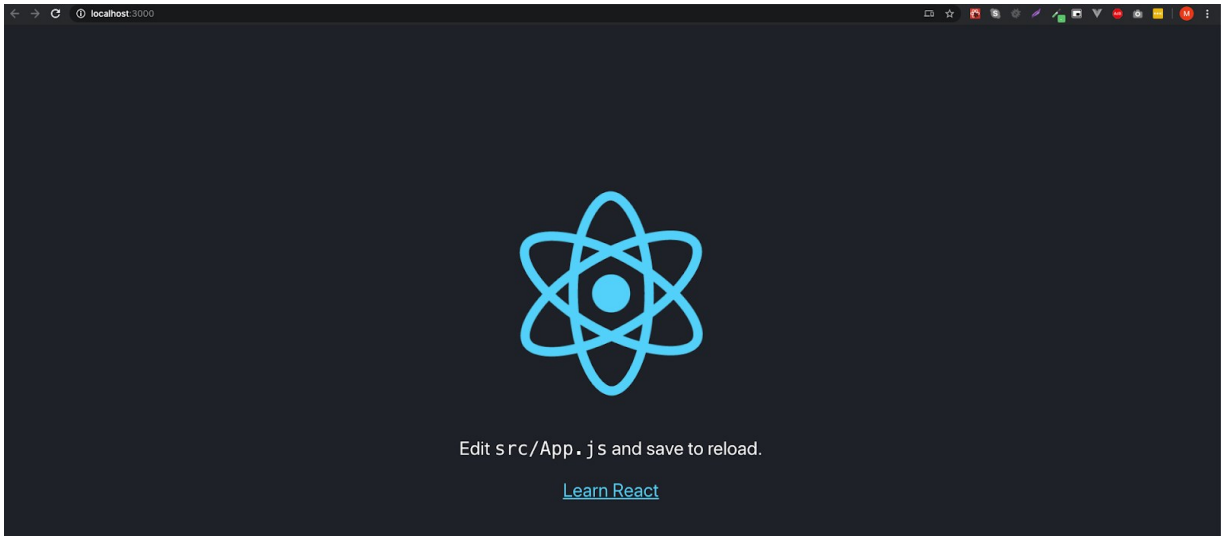
Para utilizá-la, deve-se executar o seguinte comando:

$ npx create-react-app game_analytics_frontend

Com o projeto criado, a pasta que contém o projeto deve ser acessada e na linha de comando, executado o npm start.

Ao rodar o npm start, a aplicação será executada na porta 3000. Ao informar no navegador o link http://localhost:3000, obtém-se o resultado apresentado na Figura 12:

Figura 12 - Janela de iniciação de uma aplicação *React*



Fonte: Elaborado pelo autor

O processo de instalação do React Route e Axios, pode ser encontrado no anexo II.

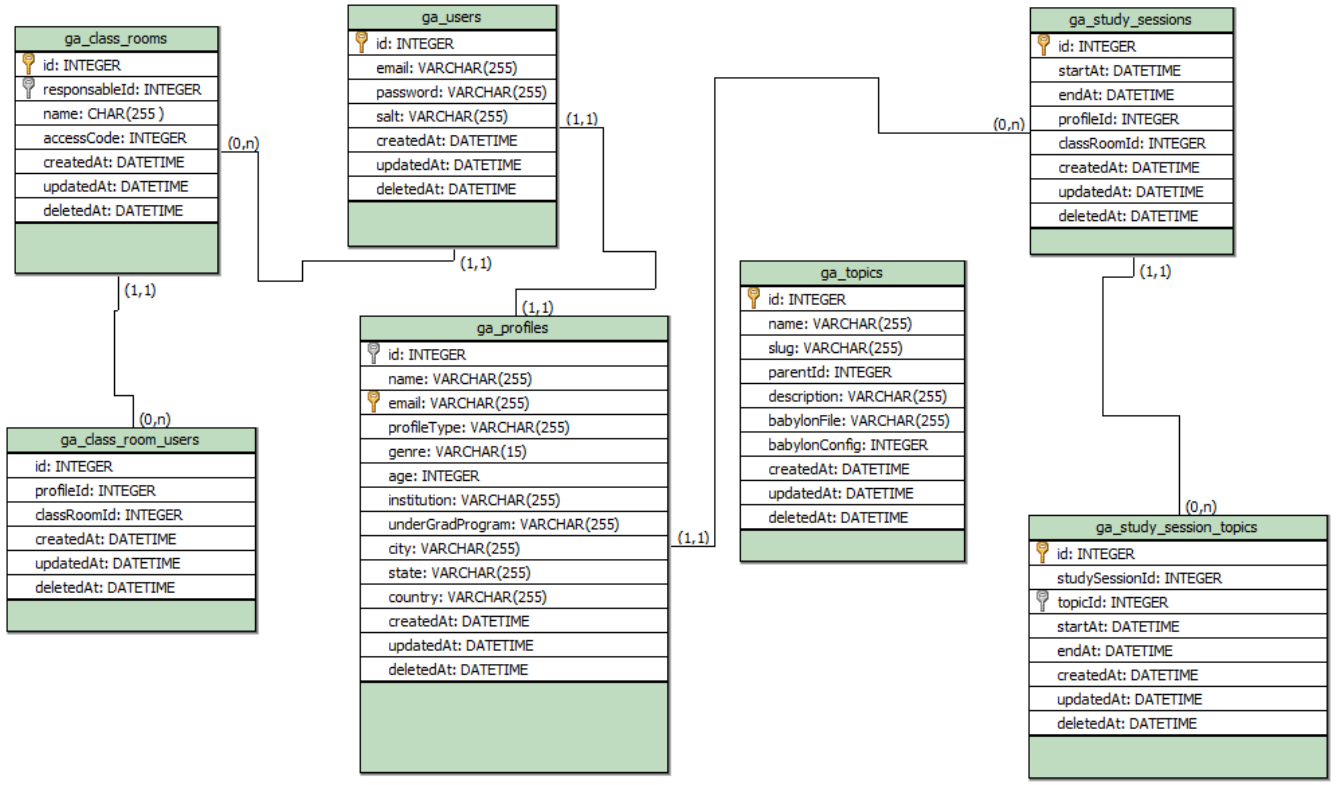
3.1.3 Projeto do Banco de Dados

3.1.3.1 O Modelo Entidade Relacionamento

Um modelo de entidade-relacionamento, ou MER, é usado para descrever um domínio e/ou seus requisitos de uma maneira abstrata onde seus principais componentes são as entidades (coisas/objetos) e suas relações, que expressam suas dependências.

Normalmente implementado como banco de dados onde as entidades são representadas como tabelas. Alguns campos apontam para índices de outras tabelas, representando seus relacionamentos. (WIKIPEDIA, 2020)

Figura 13 - Modelo Entidade Relacionamento



Fonte: Elaborado pelo autor

A seguir, são descritas as entidades criadas para armazenamento das informações da aplicação:

ga_users: Tabela usada para armazenar os usuários do sistema quando o *signup* é feito pela própria aplicação.

ga_profiles: Tabela usada para armazenar informações do usuário. Nesta tabela, aparece o relacionamento de 1 para 1 não obrigatório com a tabela *ga_users* através do atributo e-mail. Esses dados são mantidos em uma tabela separada da *ga_users* para que no futuro, se houver *signup* através de *OAuth*, como *Facebook* ou *Google*, perfis específicos possam ser criados para os usuários que optarem por este tipo de cadastro.

ga_topics: Tabela usada para armazenar os tópicos ou assuntos de estudos dentro da aplicação. Nesta tabela, ocorre um relacionamento de 1 para 1 não obrigatório com ela mesma, através do atributo *parentId*, onde é possível definir o tópico pai e assim criar relacionamento entre eles.

ga_study_sessions: Tabela usada para armazenar informações de início e término de uma sessão de estudo. Nesta tabela, surge o relacionamento de N para 1 não obrigatório com a tabela *ga_class_rooms*, através do atributo *classRoomId* e um relacionamento de N para 1 obrigatório com a tabela *ga_profiles*, através do atributo *profileId*.

ga_study_session_topics: Tabela usada para armazenar informações de início e término de um estudo de tópico. Nesta tabela, acontece um relacionamento de N para 1 obrigatório com a tabela *ga_study_sessions*, através do atributo *studySessionId* e um relacionamento de N para 1 obrigatório com a tabela *ga_topics*, através do atributo *topicId*.

ga_class_rooms: Tabela usada para armazenar salas de aula onde o professor poderá comparar o desempenho dos participantes. Nela, o relacionamento será de 1 para N com a tabela *ga_profiles*, através do atributo *responsableId*.

ga_class_room_users: Tabela usada para conter os participantes de uma sala de aula. Nesta tabela, descreve-se o relacionamento de N para 1 obrigatório com a tabela *ga_class_rooms*, através do atributo *classRoomId* e um relacionamento de N para N obrigatório com a tabela *ga_profiles*, através do atributo *profileId*.

3.1.3.2 O Banco de Dados *PostgreSQL*

O *PostgreSQL* é um banco de dados objeto-relacional criado pela universidade de *Berkley* em 1986. (POSTGRESQL, 2020)

Atualmente conta com uma equipe de desenvolvimento global formada por empresas especialistas em *PostgreSQL* (2020) e é distribuído sob a licença BSD, o que torna seu código fonte disponível e seu uso livre para aplicações comerciais ou não.

O desenvolvimento do *PostgreSQL* (2020) sempre se pautou em altos padrões de qualidade, o que torna este banco muito confiável e faz dele, uma opção tão boa quanto aquelas utilizadas pela Oracle ou o Microsoft SQL Server.

3.1.3.3 A biblioteca *Sequelize* para mapeamento relacional de objetos

Sequelize é uma biblioteca *promise-based* para *Node.js* (2020) que viabiliza a técnica chamada *Object-Relational Mapping* (ORM), que aproxima o paradigma de desenvolvimento orientado a objetos ao paradigma de banco de dados relacional. (SEQUELIZE, 2020)

Atualmente o *Sequelize* suporta conexões aos bancos de dados *PostgresSQL*, *MySQL*, *MariaDB*, *SQLite* e *Microsoft SQL*.

Criado pelo alemão Sascha Depold, hoje o *Sequelize* conta com várias *features* como suporte a transações, relacionamentos, *eager* e *lazy loading* entre muitas outras.

3.1.4 Projeto da Interface do Aplicativo

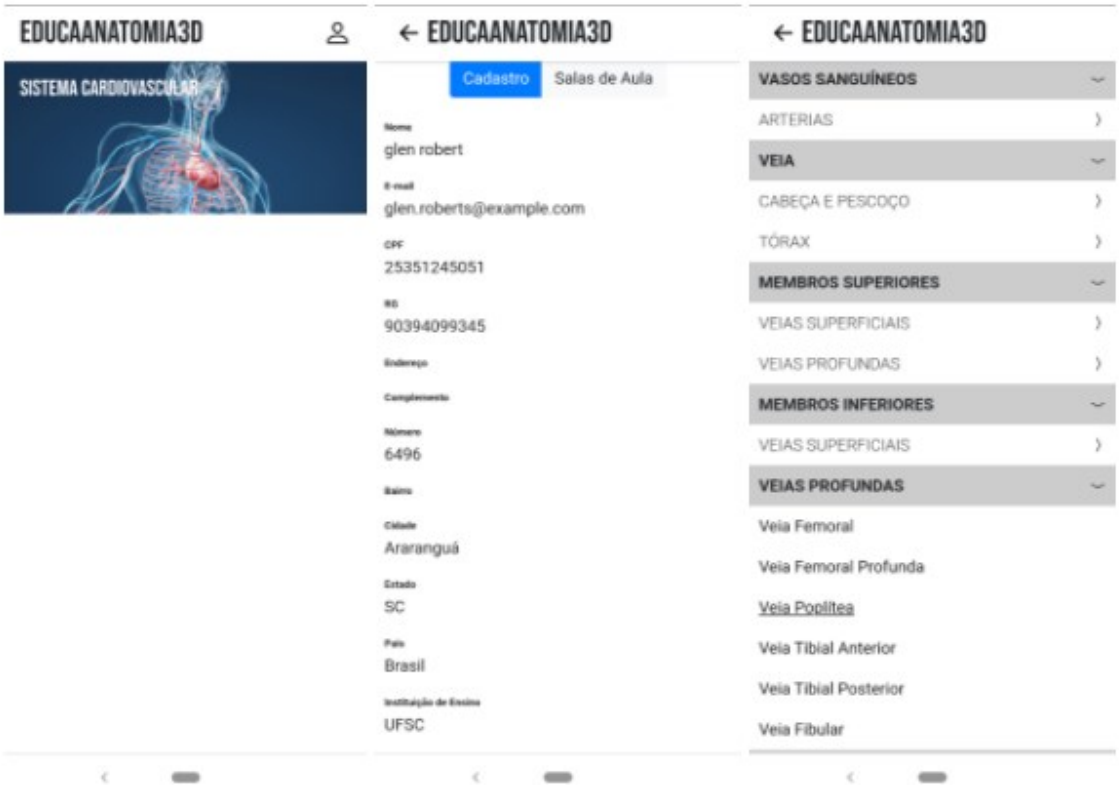
O projeto da interface do aplicativo tem como objetivo sua utilização em dispositivos móveis, podendo se tornar uma aplicação *Progressive Web App* (PWA) no futuro, sendo toda a sua interface adaptada para uso em dispositivos de tela pequena.

3.1.4.1 Projeto da Interface para Apresentação do Conteúdo

Atendendo a abordagem do projeto adotou-se uma navegação para dispositivos móveis, onde a interface segue o padrão utilizado em outros aplicativos existentes no mercado, trazendo informações dispostas em listas.

Na figura 14 apresentam-se alguns exemplos do padrão adotado para o projeto da interface do *GameAnalyticsApp* para dispositivos móveis.

Figura 14 - Padrão de interface adotado para o projeto



Fonte: Elaborado pelo autor

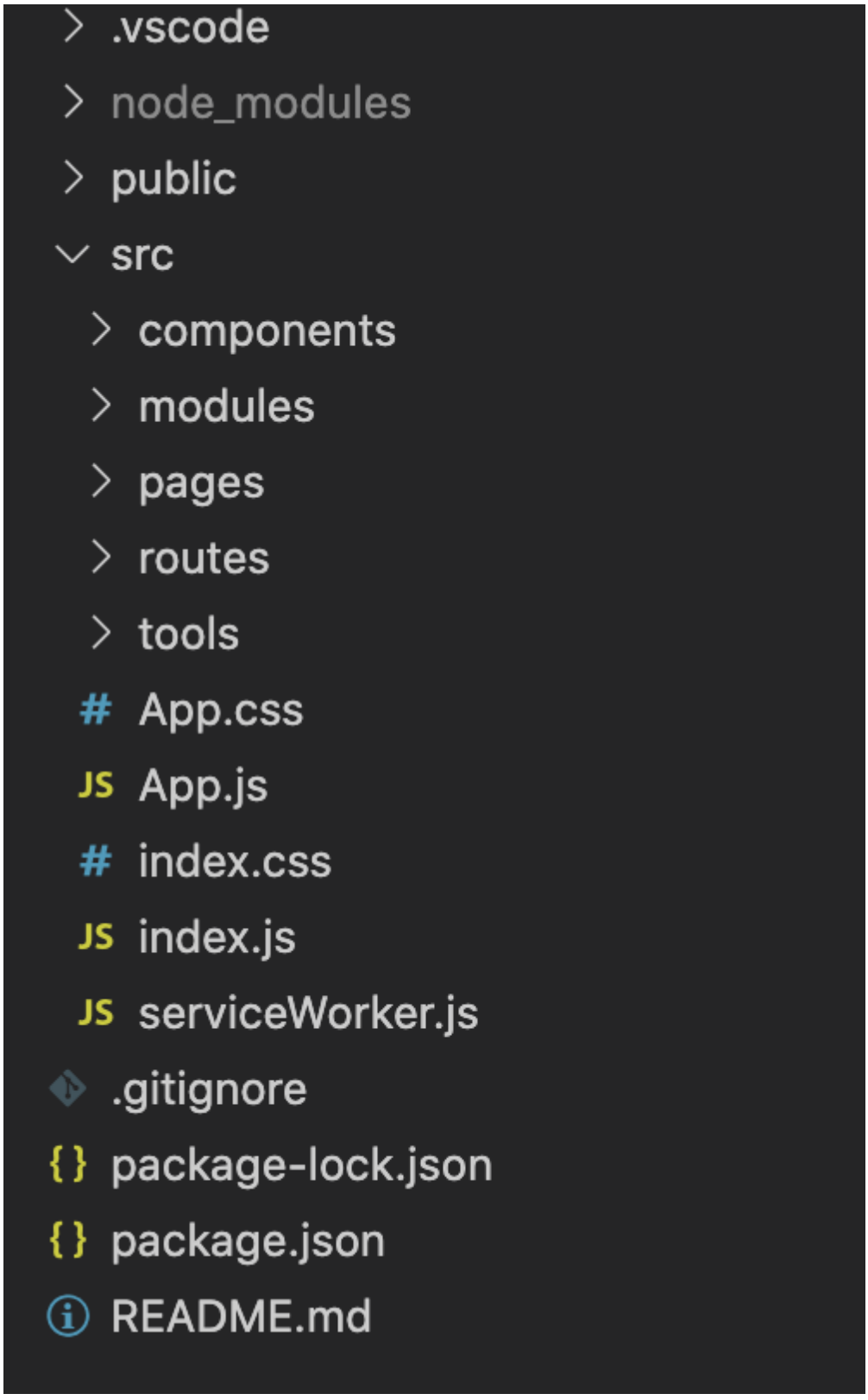
4 DESENVOLVIMENTO DE UMA APLICAÇÃO PARA O ENSINO DE ANATOMIA DO SISTEMA CARDIOVASCULAR

4.1 DESENVOLVIMENTO DO *FRONTEND*

No emprego da ferramenta para geração de projetos *React* (2020), create-react-app, por padrão uma estrutura de pasta é criada.

Para uma melhor organização de arquivos, e evitar a mistura de arquivos gerados pela ferramenta com os arquivos criados em desenvolvimento, cria-se a pasta src para armazenar arquivos pertinentes ao desenvolvimento. Na Figura 15 apresenta-se a estrutura de pastas adotadas para a aplicação relacionadas ao desenvolvimento.

Figura 15 - Estrutura de pastas



Fonte: Elaborado pelo autor

Descreve-se a seguir o funcionamento de cada uma das pastas:

components: Neste diretório se encontram todos os componentes compartilhados entre as páginas/componentes da aplicação. Sempre que um componente é compartilhado entre duas ou mais páginas/componentes, cria-se o componente dentro deste diretório.

modules: Neste diretório se encontram as *actions* e *slices* responsáveis por manter o estado da aplicação e realizar consultas nos servidores.

pages: Sempre que a biblioteca *React* (2020) é manejada para o desenvolvimento de aplicações, tudo pode ser interpretado como uma componente. Porém quando todos os componentes são combinados a fim de criar a visão final, ele será chamado de *page*. Neste diretório encontram-se todas as *pages* desenvolvidas para mostrar o conteúdo da aplicação.

routes: Com a evolução dos navegadores, torna-se possível através de técnicas específicas controlar as rotas a partir do *frontend*. Neste diretório estão os arquivos de configurações e definições das rotas que contém acesso a aplicação.

tools: Neste diretório estão os arquivos que armazenam funções que nos auxiliam durante o desenvolvimento, mantendo funções que podem ser utilizadas em vários locais centralizadas.

Como mencionado acima, com a evolução dos navegadores, é possível controlarmos as rotas de navegação pela aplicação *Frontend*.

Uma biblioteca muito empregada para manipulação do *React* (2020), para criar aplicações *single-page applications* (SPA) é a *react-router-dom*. Com o *react-router-dom*, de forma declarativa, pode-se definir cada rota de navegação na aplicação e no caso do usuário fornecer uma rota inválida, envia-se o usuário para uma rota padrão.

As Figuras 16, 17 e 18, apresentam a forma de definição das rotas da aplicação:

Figura 16 - Declaração de rotas

```
/* Biblioteca utilizada para declaracao de rotas */
import { BrowserRouter as Router, Route, Switch, useLocation } from 'react-router-dom';
```

Fonte: Elaborado pelo autor

Figura 17 - Importação dos componentes usados

```
/* Importacao dos nossos components pages para utilizar na declaracao das rotas */
import HomePage from '../pages/HomePage';
import SubjectPage from '../pages/sistema/SubjectPage';
import AboutPage from '../pages/sistema/AboutPage';
import LoginPage from '../pages/LoginPage';
import SignupPage from '../pages/SignupPage';
import ProfilePage from '../pages/ProfilePage';
import AboutStudyPage from '../pages/sistema/AboutStudyPage';
```

Fonte: Elaborado pelo autor

Figura 18 - Declaração das rotas e definição de qual componente será mostrado na tela

```
function AnimationApp() {
  const _location = useLocation();
  const _lastLocation = useLastLocation();

  return (
    <PrivateRoute>
      <TopBar />
      <TransitionGroup>
        <CSSTransition
          key={_location.key}
          timeout={{enter: 800, exit: 400}}
          classNames={pageSlider(_location, _lastLocation)}
        >
          <Switch location={_location}>
            <Route exact path="/" component={HomePage} />
            <Route exact path="/perfil" component={ProfilePage} />
            <Route exact path="/assuntos/:slug" component={SubjectPage} />
            <Route exact path="/assuntos/:slug/:topicSlug" component={AboutPage} />
            <Route exact path="/assuntos/:slug/:topicSlug/description" component={AboutStudyPage} />
          </Switch>
        </CSSTransition>
      </TransitionGroup>
    </PrivateRoute>
  );
}
```

Fonte: Elaborado pelo autor

Uma parte muito importante da aplicação é seu estado. Estado nada mais é do que a situação atual da aplicação em um determinado momento. Em qual rota o usuário se encontra? Qual componente está visível na página? Qual foi a resposta do servidor para uma dada requisição? Todas essas situações correspondem a estados dentro da aplicação.

Durante o desenvolvimento adotou-se uma ferramenta chamada *Redux Toolkit* para controlar o estado da aplicação.

Redux e uma biblioteca criada por Dan Abramov para implementar o conceito de fluxo unidirecional de dados e o *Redux Toolkit* é sua implementação para utilização dos *hooks* do *React* (2020). (REDUX TOOLKIT, 2020)

As Figuras 19, 20 e 21, apresentam o mecanismo de configuração da aplicação para utilizar o *redux toolkit*:

Figura 19 - Declaração da biblioteca *redux toolkit*

```
/* Biblioteca utilizada para configuracao do redux toolkit */
import {combineReducers, configureStore} from '@reduxjs/toolkit';
import thunk from 'redux-thunk';
```

Fonte: Elaborado pelo autor

Figura 20 - Importação dos *slices*

```
/* Importacao dos slices utilizados na aplicacao */
import appSlice from './app/slice';
import authSlice from './auth/slice';
import subjectsSlice from './subject/slice';
import topicSlice from './topic/slice';
import profileSlice from './profile/slice';
import classRoomSlice from './classRoom/slice';
```

Fonte: Elaborado pelo autor

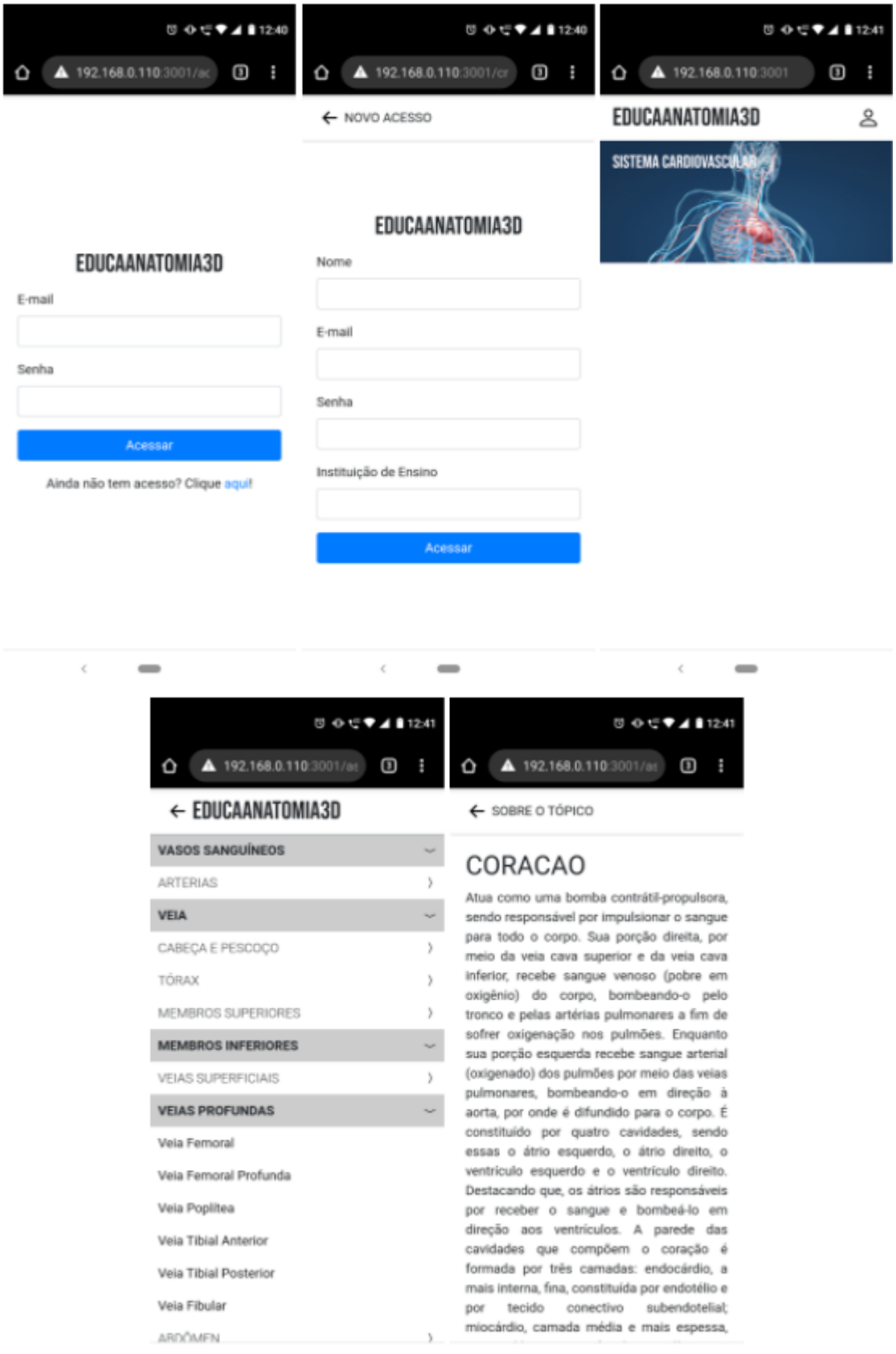
Figura 21 - Definição dos *reducers* e configuração da *store*

```
const middleware = [
  thunk
];

const rootReducer = combineReducers({
  app: appSlice,
  auth: authSlice,
  subjects: subjectsSlice,
  topic: topicSlice,
  profile: profileSlice,
  classRoom: classRoomSlice,
});

const config = {
  reducer: rootReducer,
  middleware: middleware
};

export default configureStore(config);
```

Fonte: Elaborado pelo autor

A Figura 22 apresenta as principais páginas da *interface* do *Frontend* do EducaAnatomia3D em sua versão para aplicativos móveis. (EDUCAANATOMIA 3D, 2020)

Figura 22 - Aplicação EducaAnatomia 3D



Fonte: Elaborado pelo autor

4.2 DESENVOLVIMENTO DO *BACKEND*

A aplicação está dividida em quatro módulos principais: *routes*, *controllers*, *helpers* e *models*.

O desenvolvimento inicia-se com a criação da rota e o seu registro dentro da aplicação. A Figura 23 apresenta o registro das rotas na aplicação.

Figura 23 - Registro das rotas na aplicação

```
const topicRouter = require('./routes/topic.router');
const signRouter = require('./routes/sign.router');
const studySessionRouter = require('./routes/study-session.router');
const studySessionTopicRouter = require('./routes/study-session-topic.router');
const meRouter = require('./routes/me.router');
const classRoomRouter = require('./routes/class-room.router');
const subjectRouter = require('./routes/subject.router');

        app.use('/sign', signRouter);
        app.use('/topics', topicRouter);
        app.use('/sessions', studySessionRouter);
        app.use('/sessions', studySessionTopicRouter);
        app.use('/me', meRouter);
        app.use('/class-rooms', classRoomRouter);
        app.use('/subjects', subjectRouter);
```

Fonte: Elaborado pelo autor

Após a definição do registro, no arquivo referente a rota que queremos criar, definimos como a mesma funcionará. A Figura 24 apresenta as definições da rota de tópicos(/*topics*).

Figura 24 - Definição das ações envolvidas para cada rota

```
const express = require('express');
const router = express.Router();

const TopicController = require('../controllers/topic.controller');
const { errorHandler } = require('../tools');
const { bearer } = require('../middlewares');

router.get('/', bearer, async function (req, res) {
  try {
    const topicController = new TopicController();
    res.send(await topicController.get());
  } catch (err) {
    errorHandler(err, res);
  }
});

router.get('/:idTopic', bearer, async function (req, res) {
  try {
    const topicController = new TopicController();
    res.send(await topicController.getById(req.params.idTopic));
  } catch (err) {
    errorHandler(err, res);
  }
});

router.get('/:slugNode/tree', bearer, async function (req, res) {
  try {
    const topicController = new TopicController();
    res.send(await topicController.treeStructure(req.params.slugNode));
  } catch (err) {
    errorHandler(err, res);
  }
});

router.get('/slug/:slug', bearer, async function (req, res) {
  try {
    const topicController = new TopicController();
    res.send(await topicController.getBySlug(req.params.slug));
  } catch (err) {
    errorHandler(err, res);
  }
});

module.exports = router;
```

Fonte: Elaborado pelo autor

Após a definição das ações são definidos os *controllers*, onde as ações realmente acontecem. Consultas no banco de dados, validação de campos obrigatórios, aplicação de regras de negócio entre outras várias possíveis ações que são necessárias para a aplicação.

Na Figura 25 mostra-se como estão definidas as ações de um *controller* simples, em particular o *controller topic*:

Figura 25 - Ações realizadas pelo *controller topic*

```
const { Topic } = require('../models');
const { notFound } = require('../exceptions');
const { topicWithChildrens } = require('../helpers/topic.helper');

class TopicController {
  async get() {
    return await Topic.findAll();
  }

  async getById(idTopic) {
    return await Topic.findOne({
      where: {
        id: idTopic
      }
    });
  }

  async getBySlug(slug) {
    return await Topic.findOne({
      where: {
        slug
      }
    });
  }

  async treeStructure(slugNode) {
    const mainNode = await Topic.findOne({
      where: {
        slug: slugNode
      }
    });

    if (!mainNode) {
      throw new notFound();
    }

    return topicWithChildrens(mainNode);
  }
}

module.exports = TopicController;
```

Fonte: Elaborado pelo autor

Apesar de parecer simples, existe uma função chamada *topicWithChildrens* que é responsável por montar a estrutura em árvore dos tópicos adotados na aplicação. Essa função não fica declarada dentro do *controller*, usamos um *helper* para armazenar esse método e mantermos nosso *controller* da forma mais simples possível.

A função *topicWithChildrens* é uma função especial, a qual recebe um id de um tópico e percorre ele buscando seus filhos. Porém esses filhos podem ter mais filhos, quem podem ter mais filhos. E para fazer essas buscas cada vez mais profundas na árvore, este método utiliza a si mesmo, chamamos essa abordagem de recursividade. Na Figura 26 apresenta-se a definição deste método.

Figura 26 - Método responsável pela construção da árvore de tópicos fazendo uso de recursividade

```
const { Topic } = require('../models');

const getChildrens = async parentId => {
  const childrens = await Topic.findAll({
    where: {
      parentId,
    }
  });

  if (childrens.length == 0) {
    return null;
  }

  const result = [];
  for (let i = 0; i < childrens.length; i++) {
    result.push(await topicWithChildrens(childrens[i]));
  };

  return result;
};

const topicWithChildrens = async topic => ({
  id: topic.id,
  courseId: topic.courseId,
  name: topic.name,
  slug: topic.slug,
  description: topic.description,
  babylon: topic.babylon,
  childrens: await getChildrens(topic.id),
});

module.exports = {
  getChildrens,
  topicWithChildrens,
};
```

Fonte: Elaborado pelo autor

Com a utilização do ORM *Sequelize* pode-se fazer todo o gerenciamento do banco de dados através de scripts. (SEQUELIZE, 2020)

Pode-se criar *migrations*, que consistem na criação, manutenção e exclusão de tabelas em na base de dados. Na Figura 27 apresenta-se um exemplo de uma *migration* para a criação da tabela *ga_topics* e seus relacionamentos com outras tabelas.

Figura 27 - *Migration* utilizada para criar a tabela *ga_topics* e suas chaves estrangeiras

```
'use strict';

module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.createTable('ga_topics', {
      id: {
        type: Sequelize.INTEGER,
        primaryKey: true,
      },
      courseId: {
        type: Sequelize.INTEGER,
      },
      name: {
        type: Sequelize.STRING,
        allowNull: false,
      },
      slug: {
        type: Sequelize.STRING,
        allowNull: false,
      },
      parentId: {
        type: Sequelize.INTEGER,
        allowNull: true,
      },
      description: {
        type: Sequelize.TEXT,
        allowNull: true,
      },
      babylon: {
        type: Sequelize.TEXT,
        allowNull: true,
      },
      createdAt: {
        type: Sequelize.DATE,
        allowNull: false,
      },
      updatedAt: {
        type: Sequelize.DATE,
        allowNull: false,
      },
      deletedAt: {
        type: Sequelize.DATE,
      },
    });

    await queryInterface.addConstraint('ga_topics', {
      type: 'foreign key',
      fields: ['parentId'],
      name: 'fk_parentId_ga_topics',
      references: {
        table: 'ga_topics',
        field: 'id',
      },
    });

    await queryInterface.addConstraint('ga_topics', {
      type: 'foreign key',
      fields: ['courseId'],
      name: 'fk_parentId_ga_courses',
      references: {
        table: 'ga_courses',
        field: 'id',
      },
    });
  },

  down: async (queryInterface, Sequelize) => {
    await queryInterface.dropTable('ga_topics');
  }
};
```

Fonte: Elaborado pelo autor

Os chamados modelos (*models*) servem para que possamos de uma maneira mais alto nível, acessar os dados disponíveis nas tabelas. Os modelos são representações das tabelas dentro da aplicação.

Na Figura 28 apresenta-se a representação da tabela *ga_topics* mapeado no modelo *Topic*.

Figura 28 - Mapeamento da tabela *ga_topics* para o modelo *topic*

```
module.exports = function (sequelize, DataTypes) {
  const Topic = sequelize.define('Topic', {
    id: {
      type: DataTypes.INTEGER,
      primaryKey: true,
    },
    name: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    slug: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    parentId: {
      type: DataTypes.INTEGER,
      allowNull: true,
    },
    description: {
      type: DataTypes.TEXT,
      allowNull: true,
    },
    babylon: {
      type: DataTypes.TEXT,
      allowNull: true,
    },
    babylonConfig: {
      type: DataTypes.JSON,
      allowNull: true,
    },
    createdAt: {
      type: DataTypes.DATE,
    },
    updatedAt: {
      type: DataTypes.DATE,
    },
    deletedAt: {
      type: DataTypes.DATE,
    },
  }, {
    paranoid: true,
    timestamps: true,
    tableName: 'ga_topics',
  });

  return Topic;
};
```

Fonte: Elaborado pelo autor

Para exemplificar, na Figura 29 apresenta-se um exemplo de como buscar um tópico a partir do atributo *slug*.

Figura 29 - Buscando um registro a partir do seu slug

```
const model = await Topic.findOne({
  where: {
    id: idTopic
  }
});
```

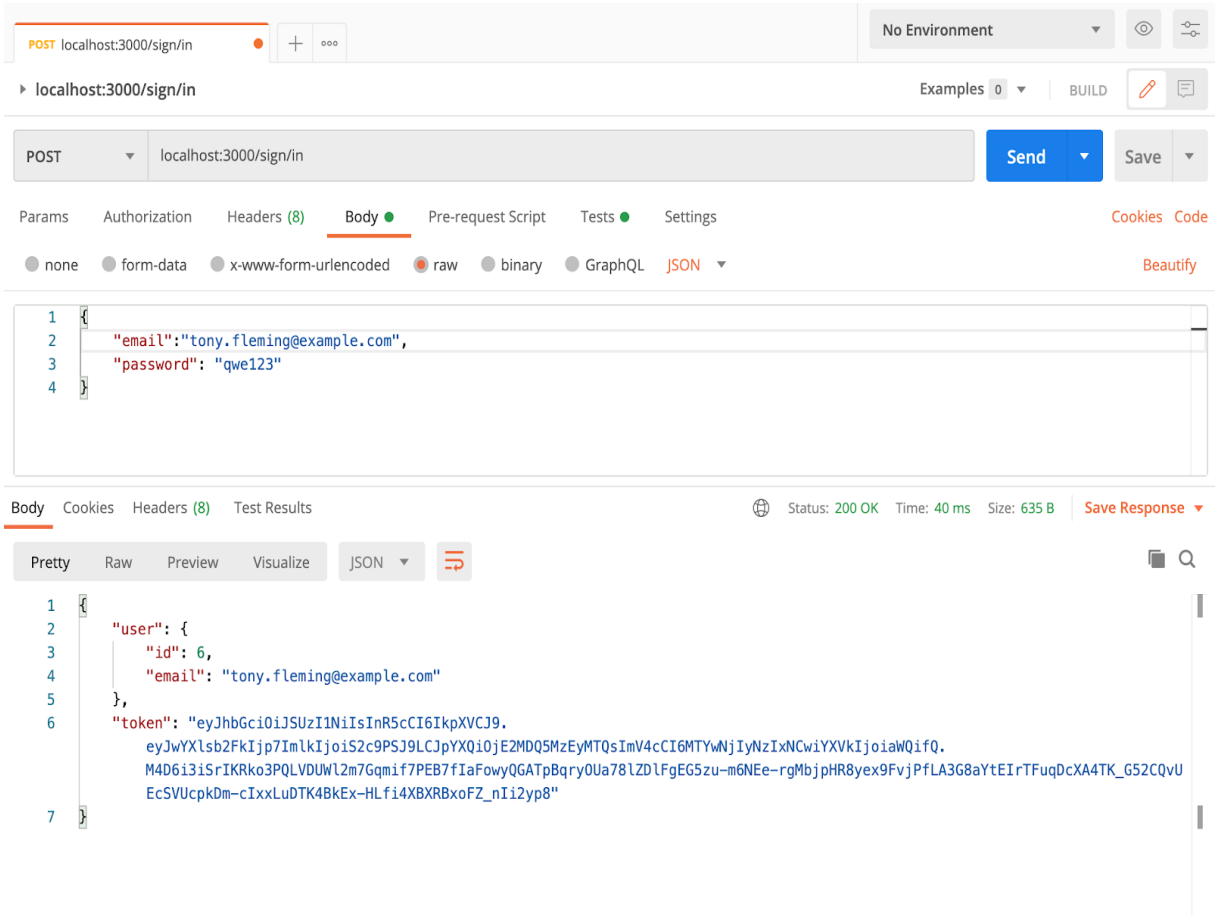
Fonte: Elaborado pelo autor

No exemplo apresentado acima na Figura 29, teríamos na variável *model* a representação do banco de dados em um objeto, onde pode-se acessar cada um dos atributos através do seu nome, por exemplo: *model.name*.

Para testar as rotas criadas para a aplicação utilizou-se uma ferramenta conhecida como *Postman*. *Postman* é uma ferramenta para auxiliar e testar o processo de desenvolvimento. Através do *Postman* pode-se criar várias configurações diferentes, testar as rotas com diferentes valores, verificar saídas, escrever testes automatizados entre várias outras funcionalidades. (Acredito que é interessante referenciar a ferramenta "*Postman*" com alguma obra)

Basicamente, após definirmos o nome da rota, e fazer a configuração do *controller* no *Postman*, basta adicionar a url da rota criada, definir o tipo de *request* (*GET*, *POST*, *PUT*, *DELETE*), acionar no botão *SEND* e aguardar a resposta do servidor. Na Figura 30 apresenta-se um exemplo utilizando a rota de *signin* criada para a aplicação com o objetivo de acessarmos o EducaAnatomia3D. (EDUCAANATOMIA 3D, 2020)

Figura 30 - Exemplo de utilização do *Postman*



Fonte: Elaborado pelo autor

4.3 PROPOSTA PARA VISUALIZAÇÃO DA ANALÍTICA DE APRENDIZAGEM

Os tópicos relacionados as artérias e veias do Sistema Cardiovascular estão subdivididos em cinco partes do corpo: Cabeça e Pescoço, Tórax, Abdômen, Membros Superiores e Membros Inferiores. A partir disso a representação das artérias e veias para cada parte do corpo corresponde a uma estrutura hierárquica. Desta forma, do ponto de vista de visualização de dados para a analítica de aprendizagem serão adotadas principalmente técnicas de visualização de dados para estruturas hierárquicas como árvores de preenchimento (por exemplo, *Treemap*) e baseadas em árvores por ligação de nós (por exemplo, árvore de nós ligados).

No contexto do *GameAnalyticsApp* serão coletados dados para o usuário do tipo: estudante e professor. A partir do modelo da base de dados definido para o *GameAnalyticsApp* (Figura 13) pode-se obter uma representação hierárquica no formato JSON (*Javascript Object*

Notation) com o conjunto de informações contidas em uma sessão de estudos. A Figura 31 abaixo apresenta uma representação das informações coletadas em uma sessão de estudos no formato de arquivo JSON.

Figura 31 - Representação hierárquica de dados coletados para uma sessão de estudo no formato JSON

```
[
  {
    "id": 1,
    "name": "Sistema Cardiovascular",
    "parent": "null",
    "text": "Caracteriza-se por um sistema que transporta o sangue por todo o corpo, sendo formado
    "children": [
      {
        "id": 11,
        "name": "Coração",
        "parent": "Sistema Cardiovascular",
        "text": "Atua como uma bomba contrátil-propulsora, sendo responsável por impulsionar o sang
        "slug": "coracao"
      },
      {
        "id": 12,
        "name": "Vasos Sanguíneos",
        "parent": "Sistema Cardiovascular",
        "text": "Existem três tipos de vasos sanguíneos: artérias, veias e capilares. As artérias c
        "children": [
          {
            "id": 121,
            "name": "Artérias",
            "parent": "Vasos Sanguíneos",
            "text": "As artérias podem ser classificadas em: Artérias de grande calibre (elásticas)
            "children": [
              {
                "id": 1211,
                "name": "Cabeça e Pescoço",
                "parent": "Artérias",
                "text": null,
                "children": [
                  {
```


Fonte: Elaborado pelo autor

O projeto de graduação do TIC apresentado neste trabalho é parte de um projeto de mestrado do PPGTIC em andamento. Ambos projetos estão inseridos no projeto de pesquisa EducaAnatomia3D (2020). Desta forma, Scheneider utilizará o aplicativo *GameAnalyticsApp* como uma ferramenta fundamental para dar continuidades ao seu projeto de mestrado. Dentro deste contexto, Scheneider desenvolveu a partir da disponibilização do arquivo no formato JSON de dados coletados para uma sessão de estudos do *GameAnalyticsApp* possíveis cenários de visualização para os usuários do tipo estudante e professor utilizando a biblioteca de visualização D3. A seguir são apresentados alguns cenários preliminares de visualização de dados para os usuários do tipo estudante e professor. (SCHENEIDER, 2020)

4.3.1 Visualização de Dados para os Estudantes

4.3.1.1 Visualização do tempo de estudo de todas as sessões de estudos

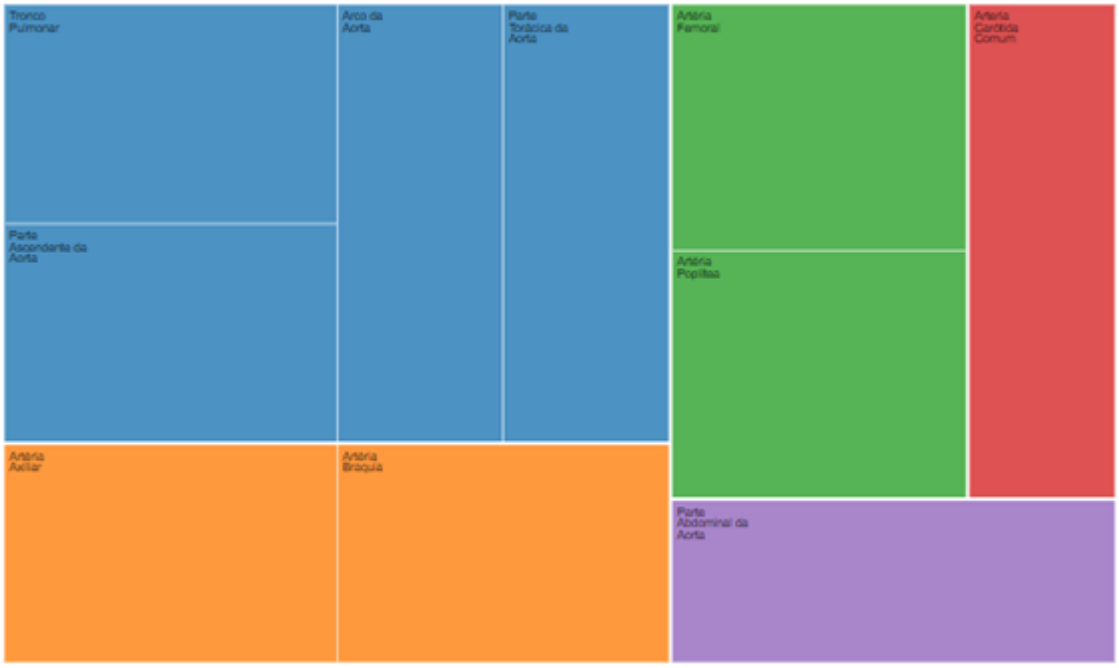
A Figura 32 apresenta a visualização do tempo de estudo de todas as sessões de estudos. Os tópicos a serem estudados estão organizados de forma hierárquica, por esse motivo o cenário de visualização proposto é através da representação de dados hierárquico conhecida como *Treemap*. Essa visualização possui o objetivo de apresentar o tempo de estudos de cada tópico durante todas as sessões de estudos de um estudante. O *Treemap* é dividido entre tópicos relacionados a artérias (azul) e veias (laranja). Tópicos relacionados a artérias e veias são subdivididos em 5 partes do corpo (i.e.; cabeça e pescoço, tórax, abdômen, membro inferior e membro superior).

Figura 32 - Visualização do *Treemap* dividido entre conteúdos de artérias (azul) e veias (laranja)

Fonte: Elaborado pelo autor

Dentro do contexto deste cenário, a Figura 33 apresenta um possível resultado quando o estudante selecionar algo na parte de representação de artérias. Assim, é possível visualizar todos os conteúdos de artérias subdivididos pelas regiões do corpo cabeça e pescoço (vermelho); tórax (laranja); abdômen(azul); membros superiores (roxo); e membros inferiores (verde)

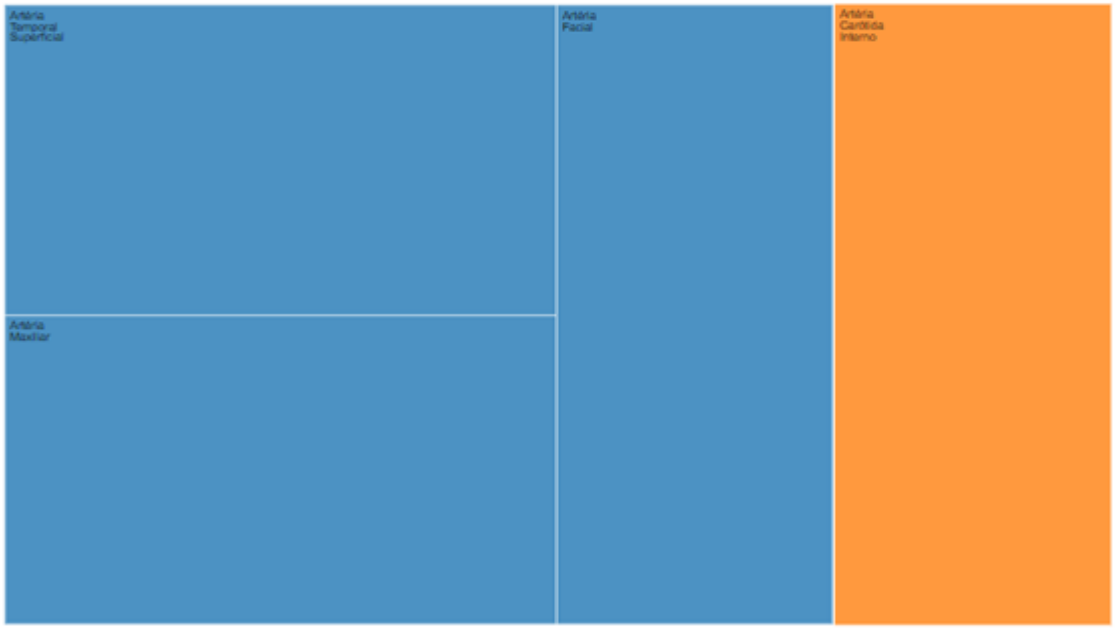
Figura 33 - Visualização dos conteúdos de artérias e suas subdivisões por regiões



Fonte: Elaborado pelo autor

Ao selecionar Artéria Carótida Comum (retângulo em vermelho), abrirá um novo subconjunto de conteúdos como mostra a Figura 34.

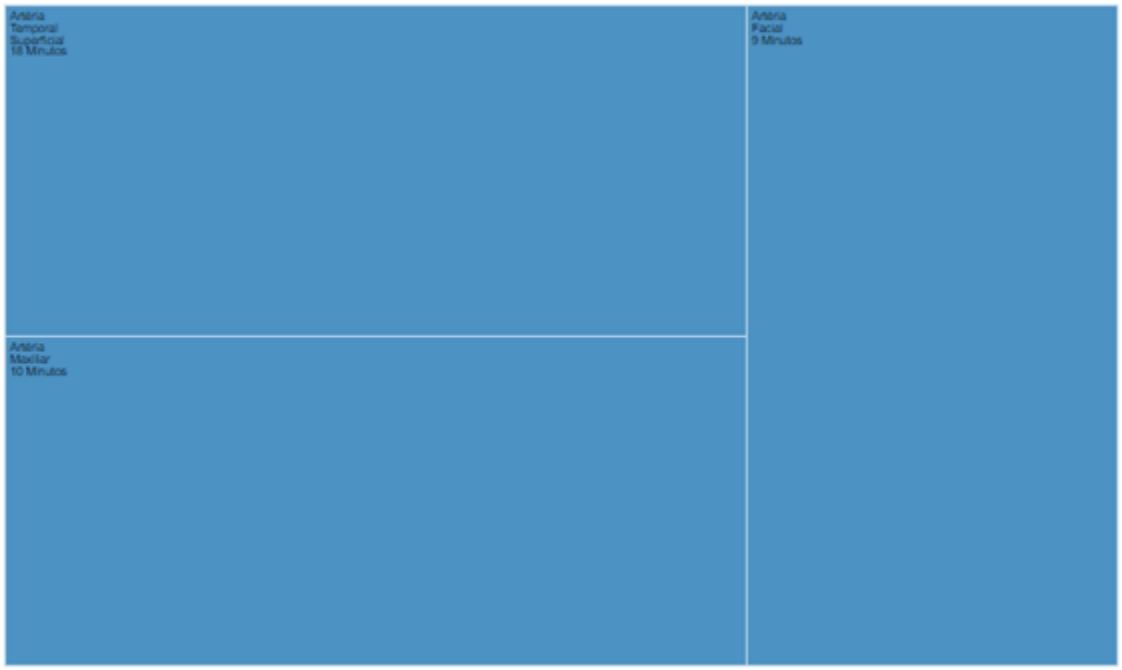
Figura 34 - Visualizar todos os conteúdos de artérias Carótida Comum e suas subdivisões



Fonte: Elaborado pelo autor

E, por fim, quando selecionado os tópicos correspondentes na hierarquia em questão serão apresentados o tempo que o aluno passou estudando cada tópico, como apresentado na Figura 35.

Figura 35 - Visualização do tempo de estudo de cada tópico



Fonte: Elaborado pelo autor

4.3.1.2 Visualização do tempo de estudo de uma sessão de estudos

Pode-se também explorar a visualização do conteúdo estudado durante uma sessão de estudos e o tempo que o estudante se dedicou a cada um dos tópicos. A Figura 36 apresenta a visualização do conteúdo estudado durante uma sessão de estudos.

Figura 36 - Visualização do conteúdo estudado durante uma sessão de estudo

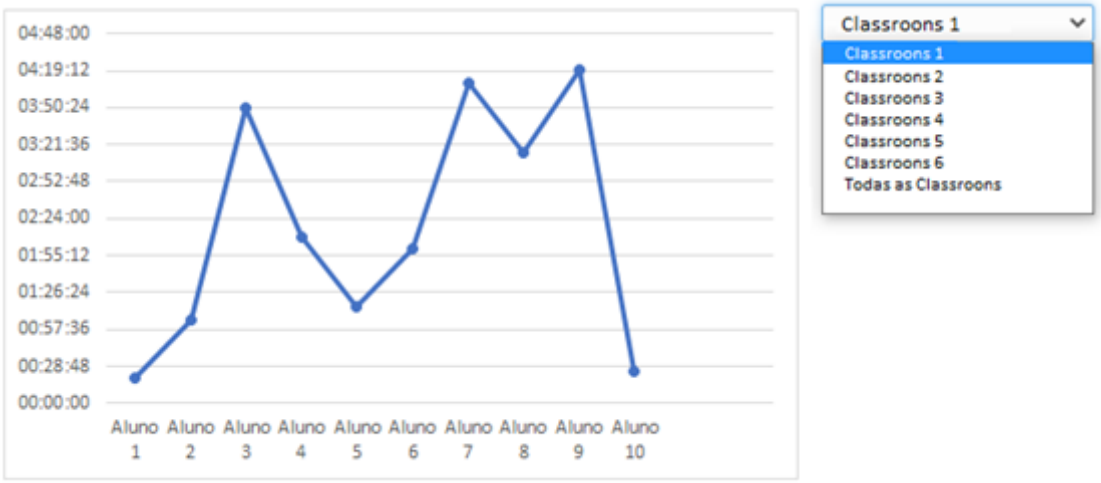
Fonte: Elaborado pelo autor

4.3.2 Visualização de Dados para os Professores

4.3.2.1 Visualização do tempo de acesso de cada aluno nas salas virtuais

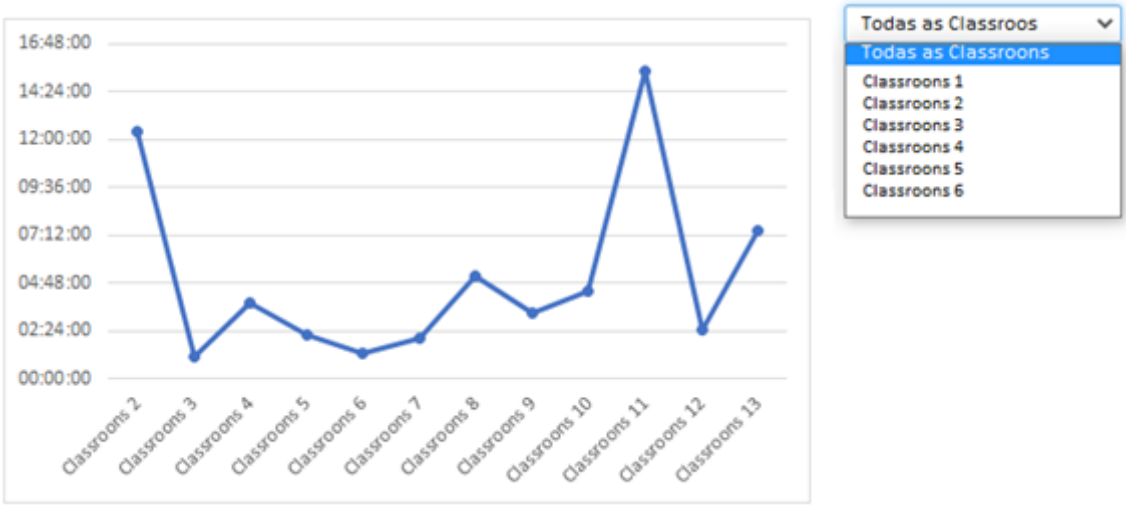
A seguir são apresentados cenários de gráficos de visualização para acompanhamento do professor nas salas virtuais. A Figura 37 apresenta o tempo médio de estudos dos alunos nas respectivas salas virtuais. E, a Figura 38 apresenta o tempo total de estudos dos alunos nas respectivas salas virtuais.

Figura 37 - Visualização do tempo médio de estudos dos alunos por sala virtual



Fonte: Elaborado pelo autor

Figura 38 - Visualização do tempo médio de cada sala virtual de estudo

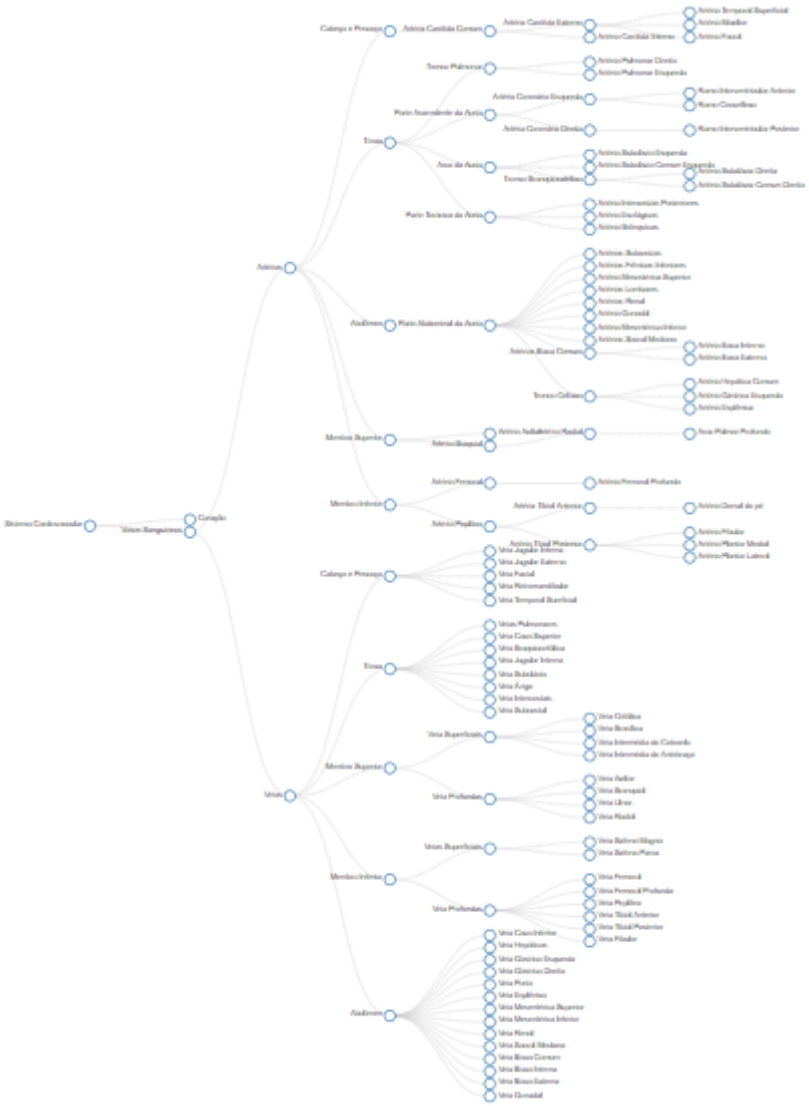


Fonte: Elaborado pelo autor

4.3.2.2 Visualização do tempo de acesso por tópico nas salas virtuais

Para visualização do tempo de acesso por tópico nas salas virtuais pode-se adotar uma visualização de árvore por nós-ligados como uma interface correspondendo ao primeiro passo para acessar informações específicas da sala virtual por tópicos (Figura 39). Os cenários restantes a partir deste passo inicial estão atualmente em fase de elaboração.

Figura 39 - Visualização da árvore de tópicos



Fonte: Elaborado pelo autor

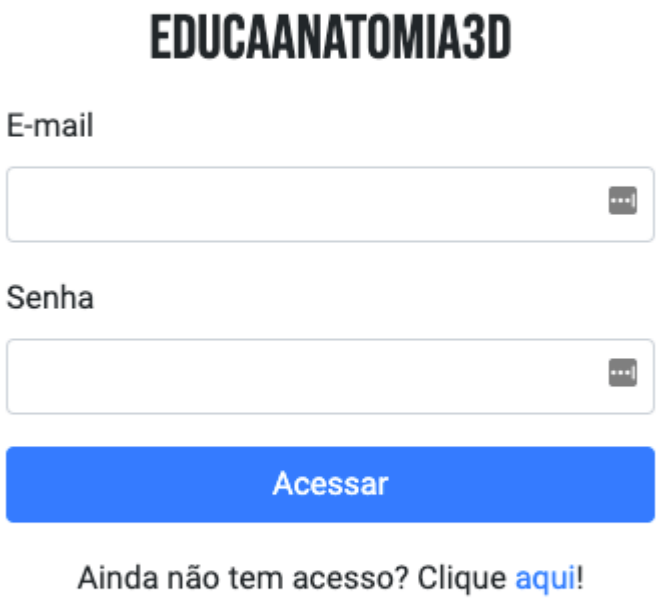
5 RESULTADOS

5.1 FLUXO DE OPERAÇÃO DO APLICATIVO *GAMEANALYTICSAPP*

A primeira versão do aplicativo *GameAnalyticsApp* encontra-se no link, www.educaanatomia3d.com.br (estará disponível em breve). A seguir é apresentado o fluxo de funcionamento atual da aplicação. (EDUCAANATOMIA 3D, 2020)

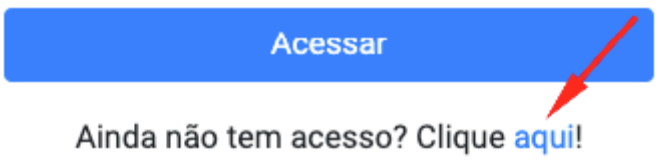
Ao acessar a url para o aplicativo *GameAnalyticsApp* a tela de acesso será mostrada. A partir da tela de acesso o usuário poderá informar seu *email* e senha para se identificar, conforme Figura 40. Caso o usuário ainda não esteja cadastrado, o usuário pode criar o acesso clicando no link localizado após o botão "Acessar", conforme indicado na Figura 41.

Figura 40 - Tela de acesso da aplicação



Fonte: Elaborado pelo autor

Figura 41 - Link para criar novo usuário de acesso



Fonte: Elaborado pelo autor

Na tela de criação de novo acesso, o usuário deve preencher todos os campos do formulário e em seguida deve clicar em Acessar. A Figura 42 apresenta o formulário para a página de cadastro.

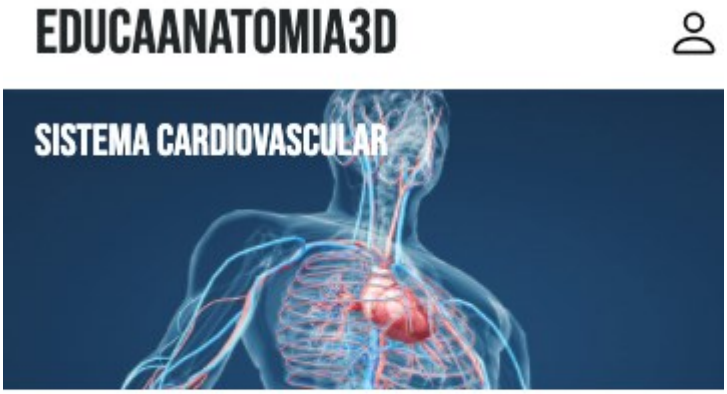
Figura 42 - Tela para criação de novo acesso

Fonte: Elaborado pelo autor

Na tela principal do *GameAnalyticsApp*, após o acesso, o usuário terá acesso a todos os módulos de ensino disponíveis na aplicação. Neste primeiro momento somente o módulo do Sistema Cardiovascular está disponível.

A Figura 43 apresenta a tela principal do *GameAnalyticsApp*.

Figura 43 - Tela principal da aplicação



Fonte: Elaborado pelo autor

Ao acessar o módulo do Sistema Cardiovascular, os tópicos de estudo estarão disponíveis para acesso ao usuário, conforme apresentado a Figura 44.

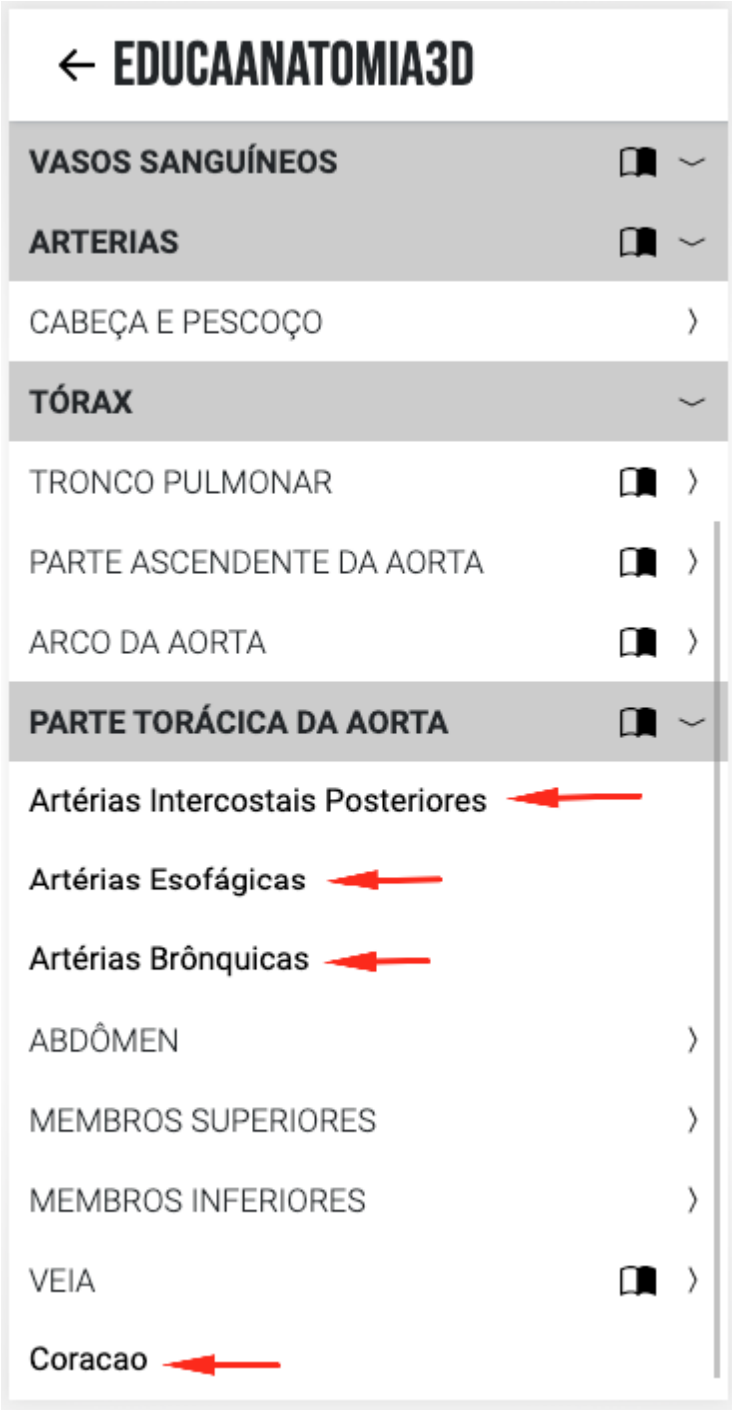
Figura 44 - Tópicos disponíveis para o Sistema Cardiovascular

Fonte: Elaborado pelo autor

Na tela apresentada acima na Figura 44 também é possível acessar o conteúdo de estudo relacionado a um dado tópico. O conteúdo pode ser acessado de duas maneiras:

1. Escrita em *Camelcase* e em negrito, conforme Figura 45 (como indicado por setas em vermelho).

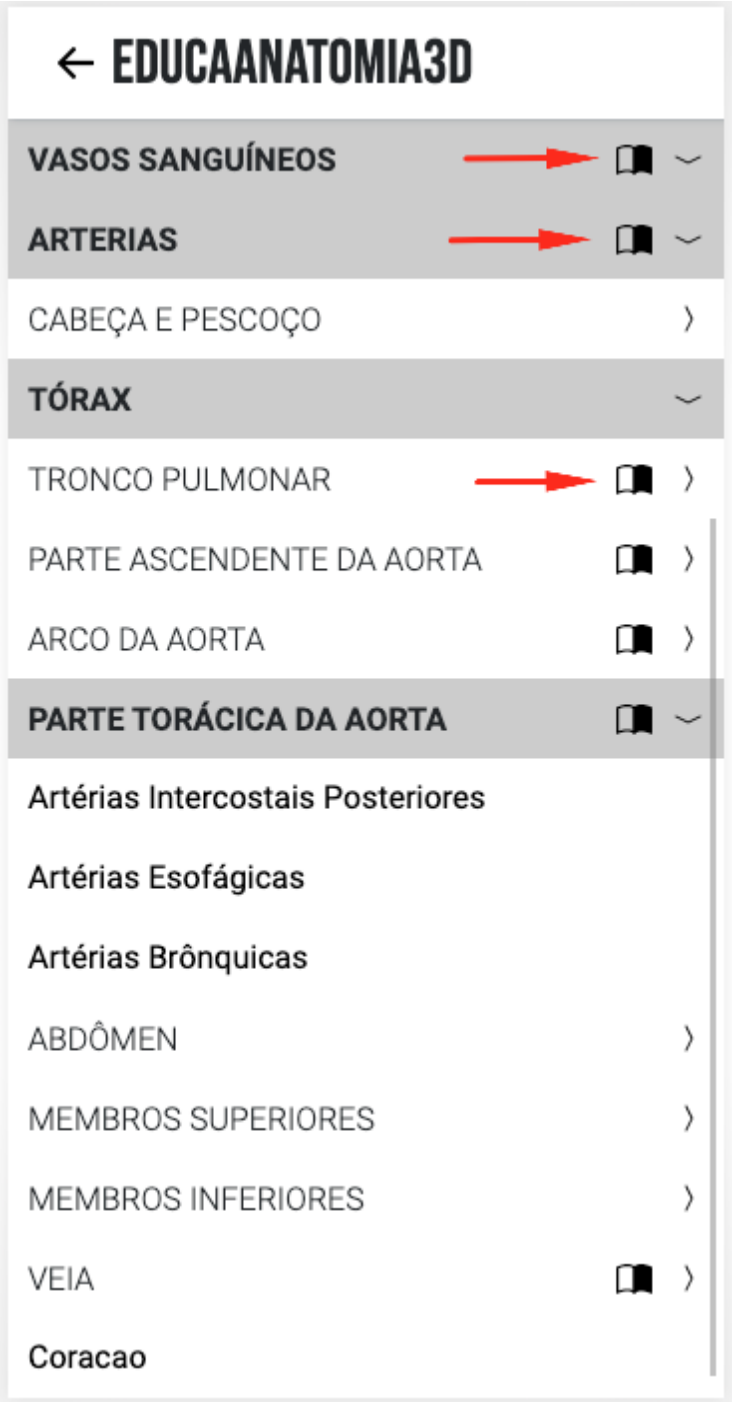
Figura 45 - Acessando conteúdo de estudo



Fonte: Elaborado pelo autor

2. Pelo ícone representado por um livro, conforme Figura 46 (como indicado por setas em vermelho).

Figura 46 - Acessando conteúdo de estudo pelo ícone livro



Fonte: Elaborado pelo autor

Ao acessar o conteúdo de estudo, teremos duas opções: a) interagir com a representação da estrutura anatômica por meio de um objeto 3D associado ao conteúdo selecionado; b) e, acessar informações sobre o conteúdo em formato de texto.

A Figura 47 apresenta a tela onde será mostrado o objeto 3D.

Figura 47 - Objeto 3D

Fonte: Elaborado pelo autor

Para acessarmos informações do conteúdo em formato texto, deve-se clicar no ícone no cabeçario (*header*) da aplicação em formato de livro, conforme apresentado na Figura 48.

Figura 48 - Ícone de acesso ao conteúdo em formato texto

Fonte: Elaborado pelo autor

Ao clicar neste ícone será apresentado o conteúdo em formato texto, conforme apresentado na Figura 49.

Figura 49 - Tela com o conteúdo em formato texto

Fonte: Elaborado pelo autor

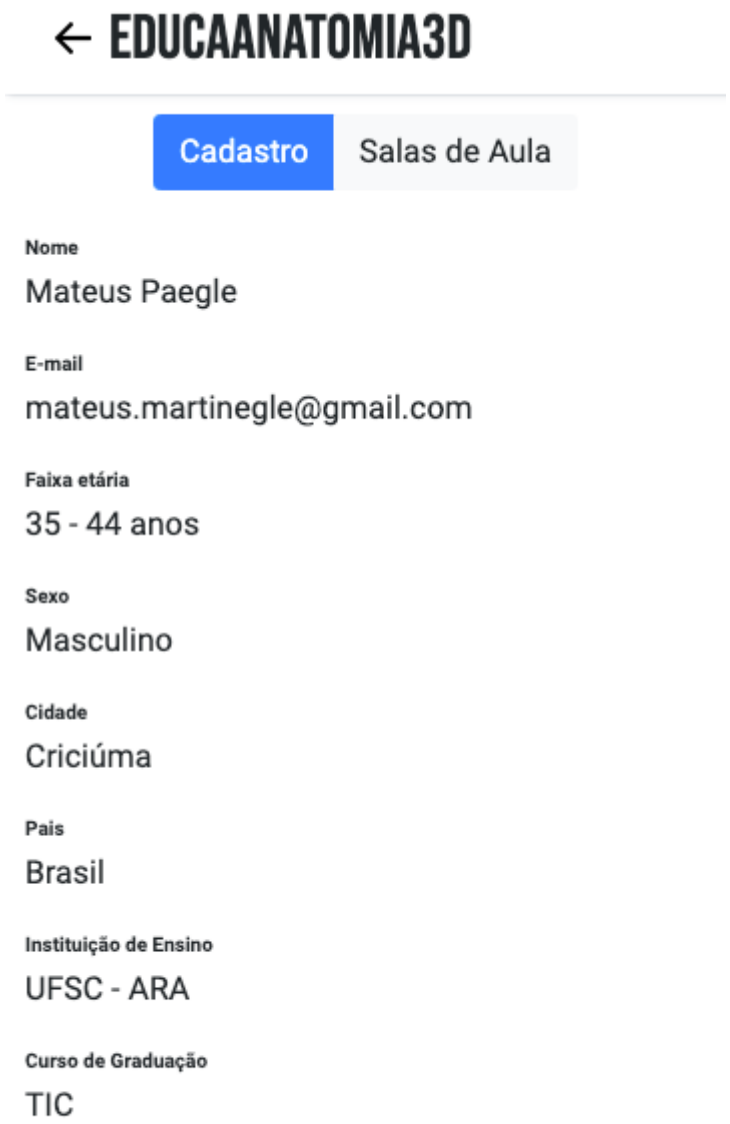
Além disso, na tela inicial pode-se acessar o perfil do usuário atual, através do ícone no formato de uma pessoa, conforme Figura 50.

Figura 50 - Acesso ao perfil do usuário logado

Fonte: Elaborado pelo autor

Na tela de cadastro é somente apresentado as informações que o usuário utilizou para criar sua conta. A Figura 51 apresenta a tela de cadastro.

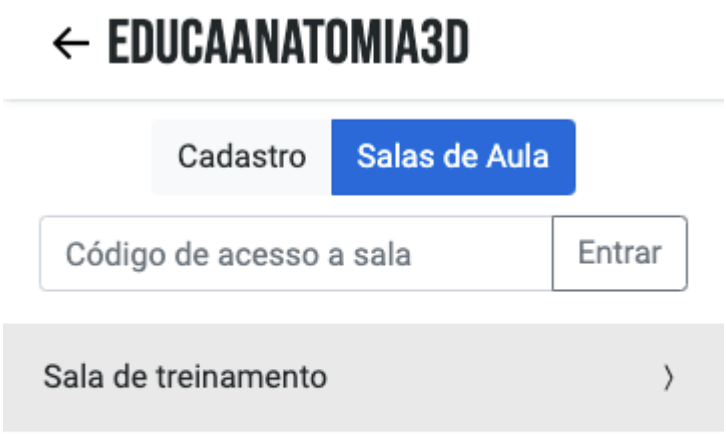
Figura 51 - Lista com as informações do usuário



Fonte: Elaborado pelo autor

Além disso, esta tela permite visualizar as salas em que o usuário tem acesso e se assim desejar o usuário pode selecionar uma sala para realizar uma sessão de estudos. A Figura 52 apresenta a tela com as informações para sala de aulas.

Figura 52 - Seleção e ingresso em uma sala de aula



Fonte: Elaborado pelo autor

Para o usuário participar de uma sala, basta informar o código de acesso no campo "Código de acesso a sala" e clicar em "Entrar". Ao realizar esta ação, a sala será apresentada na lista de salas e para participar de uma sala basta selecioná-la.

Quando o usuário estiver participando de uma sala um ícone no formato de medalha será exibido no topo da aplicação conforme apresentado na Figura 53.

Figura 53 - Ícone informando que o usuário está participando de uma sala

Fonte: Elaborado pelo autor

6 CONSIDERAÇÕES FINAIS E TRABALHO FUTUROS

O projeto e desenvolvimento do *GameAnalyticsApp* utilizando tecnologias do tipo *single-page application* como o *React* (2020) e melhores práticas de desenvolvimento de software irão possibilitar que a aplicação, a partir deste estágio, possa crescer de forma organizada.

A modularização do *Backend* voltada a domínio, ajudou a manter cada módulo da aplicação em seu lugar e outros desenvolvedores poderão seguir o mesmo padrão, pois existe uma estrutura lógica ao se criar e expandir o sistema para novas funcionalidades (*features*). Ainda no *Backend* a utilização do *Object Relational Mapping* (ORM) *Sequelize* ajudou a manter organizado os acessos a registros necessários no banco de dados. (*SEQUELIZE*, 2020)

Ao se utilizar um framework para o desenvolvimento *Frontend*, aproveitamos todo o potencial de performance que podemos ter em um browser. Através do *React* (2020) utilizamos o conceito de componentização onde após criar nossos componentes podemos, como um quebra-cabeça, juntar os componentes (peças) e montar as visualizações de forma prática e rápida.

Como trabalhos futuros pretende-se adicionar novos módulos, tais como o Sistema Esquelético, o Sistema Muscular e o Sistema Nervoso. Também, pretende-se criar uma nova etapa de aprendizagem utilizando um jogo do tipo questionário (*Quiz* do tipo pergunta/resposta). Além disso, pretende-se disponibilizar painéis de controle para acompanhamento do usuário do tipo estudante e professor para acompanhar o processo de ensino e compreensão das etapas de ensino a fim de tomar possíveis decisões para que o processo melhore a partir de modificações ou recomendações futuras.

Vale ressaltar que para adicionar os objetos 3d utilizando a biblioteca *Babylon* e a biblioteca de gráficos D3.js, eles devem ser integrados ao *React* para manter a compatibilidade.

E ainda para disponibilizar os arquivos contendo os objetos 3d que o *Babylon* precisa para funcionar, pode-se utilizar o ambiente AWS utilizando o serviço S3, que é um serviço de armazenamento de documentos.

REFERÊNCIAS

Axios, (2020). Disponível em: <https://github.com/axios/axios>. Acesso em: Novembro de 2020.

EducaAnatomia3d, (2020). Disponível em: <http://www.labanatomiainterativa.ufsc.br/ea3d/>. Acesso em: Outubro de 2020.

Express.js, (2020). Disponível em: <https://expressjs.com/pt-br/>. Acesso em: Outubro de 2020.

Node.js, (2020). Disponível em: <https://nodejs.org>. Acesso em: Outubro de 2020.

Node Version Manager NVM, (2020). Disponível em: <https://github.com/nvm-sh/nvm>. Acesso em: Outubro de 2020.

Opus Software, (2020). Disponível em: <https://www.opus-software.com.br/node-js>. Acesso em: Outubro de 2020.

Postgresql, (2020). Disponível em: <https://wiki.postgresql.org/wiki/Introdu%C3%A7%C3%A3o_e_Hist%C3%B3rico>. Acesso em: Novembro de 2020.

React, (2020). Disponível em: <https://pt-br.reactjs.org/>. Acesso em: Outubro de 2020.

React Router, (2020). Disponível em: <https://reactrouter.com/>. Acesso em: Outubro de 2020.

Redux Toolkit, (2020). Disponível em: <https://redux-toolkit.js.org/introduction/quick-start>. Acesso em: Novembro de 2020.

Peffers, Ken et al. **A Design Science Research Methodology for Information Systems Research**. Journal of Management Information Systems, v. 24, n. 3, p.45-77, 2007.

Guiraldes, D. C., Oddó Atria, H. and Ortega, F. (1995) **Métodos computacionales y gráficos de aopyo al aprendizaje de la anatomía humana: vision de los estudiantes/Computer and graphic methods of support to the human anatomy learning: the students point of view**. Revista Chilena de Anatomia. V. 13, n. 1, p. 76-71.

Prikladnicki, R. and Wangenheim, C. G. (2008). **O Uso de jogos Educacionais para o ensino de gerencia de projetos de software**. In: FÓRUM DE EDUCAÇÃO EM ENGENHERIA DE SOFTWARE, 1., Fortaleza. Anais eletrônicos. Rio de Janeiro: PUC.

Mccoy, Lise; Lewis, Joy H.; Dalton, David. **Gamification and Multimidia for Medical Education: A Landscape Review**. The Journal Of The American Osteopathic Association, [s.l.], v. 116, n. 1, p1-13, 1 jan. 2016. American Osteopathic Association.

Scheneider, T., & Lemos, R. (2020). **The Use of Learning Analytics Interactive Dashboards in Serious Games**: A Review of the Literature. International Journal for Innovation Education and Research, 8(3), 150-174. Disponível em: <https://doi.org/10.31686/ijier.vol8.iss3.2220>. Acesso em: Novembro de 2020.

SCHENEIDER, Thais. **Painéis interativos de analítica de aprendizagem para apoiar estudantes em um jogo sério no ensino de anatomia humana** (título provisório). Início: 2019. Dissertação (Mestrado em Programa de Pós-Graduação em Tecnologias da Informação e Comunicação) - Universidade Federal de Santa Catarina. (Orientador). (em andamento).

Sequelize, (2020). Disponível em: <https://sequelize.org/>. Acesso em: Novembro de 2020.

Wikipedia, (2020). Disponível em: <https://pt.wikipedia.org/wiki/Modelo_entidade_relacionamento>. Acesso em: Novembro de 2020.

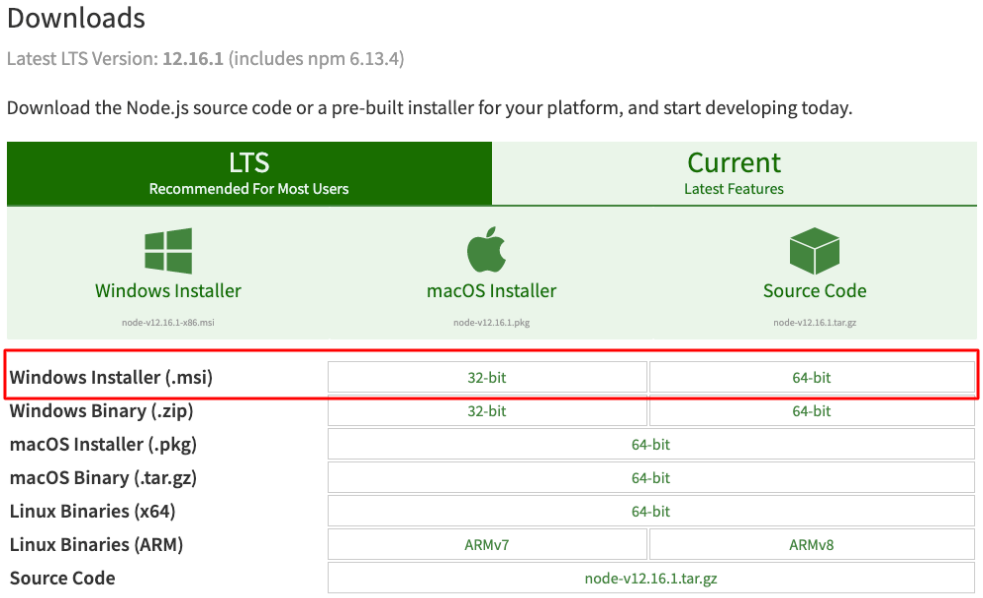
ANEXOS

ANEXO I – O PROCESSO DE INSTALAÇÃO DO *NODE.JS* E *EXPRESS*

Instalação do Node.js para Windows 10

O *Node.js* está disponível a partir do link https://nodejs.org/en/download/. De acordo com as instruções descritas na Figura 54 pode-se selecionar o *Windows Installer* (.msi) e seguir o processo de instalação. Para conferir a versão basta ir no *Windows Powershell* e digitar node -v. (*NODE.JS*, 2020)

Figura 54 - Página para download do *Node.js*



Fonte https://nodejs.org/en/download/

Instalação do Node.js para Sistemas Unix like (linux/macos)

Já no sistema *Unix like*, a forma mais simples de se instalar o *Node.js* é utilizar uma ferramenta chamada nvm (*Node Version Manager*) disponível no link (https://github.com/nvm-sh/nvm) (NODE VERSION MANAGER NVM, 2020)

Para isso, basta executar o comando:

$ wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash

Este comando baixará os arquivos necessários e adicionará os comandos necessários no arquivo de profile.

Após a finalização da execução do *script*, basta executar o comando:

$ nvm install 12

O comando acima instalará a última versão estável (lts), que é a versão 12.*.*. A fim de instalar esta versão, é possível utilizar o comando:

$ nvm install --lts

O Processo de Instalação do Framework Express

As principais etapas de instalação do *framework Express* (2020) são descritas a seguir.

Cria-se o diretório onde a aplicação será criada:

$ mkdir game_analytics_app

$ cd game_analytics_app

Utiliza-se o comando npm init para criação da aplicação *Node.js* (2020). Este comando gera o arquivo *package.json*. Neste arquivo serão armazenadas todas as informações de pacotes utilizados pela aplicação e outras informações como, versão, scripts de inicialização, etc.

$ npm init -y

Usa-se então a opção -y para aceitar todos os valores *default*. A partir disso pode-se instalar o *Express* (2020) como uma dependência da aplicação:

$ npm i express --save

Utiliza-se a opção --save para efetuar o registro como uma dependência da aplicação no arquivo *package.json*.

ANEXO II – INSTALANDO REACT ROUTER E AXIOS

Instalando React Router e Axios

Para instalar as ferramentas *React Router* (2020) e *Axios* basta executar o comando abaixo:

$ npm i --save react-router-dom axios

Com isso, obtêm-se os pacotes instalados e prontos para uso. (AXIOS, 2020)