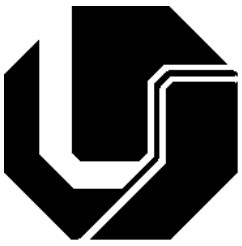


UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA – FEELT
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

IGOR MOREIRA FRANCISCO

PLATAFORMA WEB PARA AUXÍLIO DE TREINAMENTO DO USO DE PRÓTESES PARA INDIVÍDUOS COM AMPUTAÇÃO DE MEMBROS SUPERIORES

Uberlândia - MG
Dezembro/2020

IGOR MOREIRA FRANCISCO

PLATAFORMA WEB PARA AUXÍLIO DE TREINAMENTO DO USO DE PRÓTESES PARA INDIVÍDUOS COM AMPUTAÇÃO DE MEMBROS SUPERIORES

Trabalho apresentado como requisito parcial de avaliação na disciplina Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal de Uberlândia.

Banca Examinadora:

Prof. Edgard A. Lamounier Jr, PhD – Orientador (UFU)

Prof. Luciano Coutinho Gomes, Dr. – (UFU)

Prof. Adriano Alves Pereira, Dr. – (UFU)

Uberlândia - MG
Dezembro/2020

IGOR MOREIRA FRANCISCO

PLATAFORMA WEB PARA AUXÍLIO DE TREINAMENTO DO USO DE PRÓTESES PARA INDIVÍDUOS COM AMPUTAÇÃO DE MEMBROS SUPERIORES

Uberlândia, 21 de dezembro de 2020.

Prof. Edgard A. Lamounier Jr, Phd.	Prof. Luciano Coutinho Gomes, Dr.
Orientador	Membro 1

Prof. Adriano Alves Pereira, Dr.

Membro 2

Uberlândia - MG
Dezembro/2020

Resumo

A amputação de um membro superior é um processo traumático. O período de reabilitação pode levar meses, devido à dificuldade de adaptação do paciente com a prótese. Esse tempo, para o paciente, é algo cansativo e desanimador. Esse período ele será acompanhado por um terapeuta ocupacional diariamente e outros profissionais ocasionalmente, baseando-se nesse fato, esse trabalho propõe uma plataforma web para auxiliar o paciente e terapeuta ocupacional, em sua rotina de treinamento diário na utilização da prótese. A plataforma web tem como principal funcionalidade fornecer informações diárias para que o terapeuta acompanhe de perto o treinamento de diversos pacientes, sem precisar estar presente em consulta. Tanto paciente quanto terapeuta podem ter acesso ao sistema de qualquer local, seja através de smartphone ou computador com acesso à internet. É utilizado uma interface diretamente voltada para coleta dos dados e envio dos resultados, a coleta de dados é feita por tecnologias adaptadas tanto para o modo web quanto mobile. O trabalho descreve as ferramentas utilizadas, a construção do sistema, seu modo de utilização e funcionalidades.

Palavras-chave: Front-end; Back-end; React; Mobile, Terapeuta Ocupacional, NoSQL.

Abstract

Amputation of an upper limb is a traumatic process. The rehabilitation period can take months, due to the difficulty of the patient to adapt with the prosthesis. This time, for the patient, is tiring and discouraging, as he will be accompanied by an occupational therapist daily and other occasional professionals. Based on this fact, this work offers a web platform to assist the patient and occupational therapist, who will be reported at work only as "therapist", in the use of the prosthesis during the training period. The web platform has as main functionality, information provided for the therapist to closely monitor the training of several patients, without having to be present for consultation. Both patient and therapist can access the system from any location, whether via smartphone or computer with internet access. An interface is directly used to collect data and send results, data collection is done by technologies adapted to both web and mobile mode. The work classes the tools used, the construction of the system, its mode of use and characteristics.

Keywords: Front-End; Back-End; React; Mobile, Occupational Therapist, NoSQL;

Lista de Figuras

Lista de Abreviaturas e Siglas

API Application Programming Interface

CSS Cascading Style Sheets

CRUD Create, Read, Update, Delete

DOM Document Object Model

ES6 ECMAScript 6

ES7 ECMAScript 7

HTML HiperText Markup Language

HTTP Hyper Text Transfer Protocol Secure

JS JavaScript

JSON JavaScript Object Notation

JSX JavaScript XML (Extensible Markup Language)

NOSQL Not Only Structured Query Language

NPM Node Package Manager

REST Representational State Transfer

RFC Request for Comments

SDK Software Development Kit

SQL Structured Query Language

UI User Interface

URL Uniform Resource Locator

Sumário

Capítulo 1

Introdução

1.1 Temática e Motivação

A amputação de um membro afeta quase todos os aspectos da vida de um indivíduo. Os amputados, além de sua deficiência física, sofrem de inúmeros problemas psicológicos e psicossociais (BRADWAY, MALONE e RACY, 1984).

Em alguns casos, a amputação pode levar a complicações, como é o caso de amputados traumáticos, que têm altas taxas de complicações físicas e psicológicas, no qual podem seguir um curso crônico. Se reconhecida, a maioria pode ser tratada e incapacidades desnecessárias seriam evitadas. Os amputados devem, portanto, receber acompanhamento regular do terapeuta no período de reabilitação por pelo menos 2 anos após a lesão e permanecer sob vigilância de longo prazo para evidências de problemas de saúde física e/ou mental (JOHN WILEY & SONS, 2011).

A abordagem fisioterapêutica na fase de pré-protetização utiliza diversos aspectos e técnicas importantes da Fisioterapia, do qual é válido destacar o método utilizado por (NUNES JUNIOR, 2009). Na obtenção dos resultados, foi elaborado um questionário composto por 10 perguntas objetivas, aplicado as condições funcionais relacionadas às atividades da vida diária e avaliação da experiência dolorosa. O paciente relatou importante alívio da sintomatologia álgica e sensação fantasma, e aumento da funcionalidade, melhorando suas atividades cotidianas.

Além disso é importate notar que o desenvolvimento tecnológico altera diversas práticas na área da saúde, abrangendo atividades como diagnóstico, terapia, treinamento, gerenciamento e educação" (NUNES, , *et al.*, 2011). Logo, através de uma aplicação web, os transtornos que o paciente sofre ao se deslocar para o treinamento serão evitados, além é claro, de uma economia de recursos do terapeuta durante o treinamento. A web facilita o treinamento, uma vez que tanto paciente quanto terapeuta podem acessar o sistema, diariamente, a qualquer

momento e em qualquer local, basta que ambos tenham acesso a um smartphone ou computador com acesso à internet.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem como objetivo criar uma plataforma web para auxílio do treinamento de indivíduos com amputação de membros superiores. O sistema tem como propósito ser direto, prático, acessível e com amplo acesso para todos usuários.

1.2.2 Objetivos Específicos

O sistema é voltado para ser utilizado como conteúdo informativo para quem acessá-lo e para os usuários logados, acesso ao sistema de treinamento. A plataforma web é hospedada em um sistema serveless (sem servidor) na nuvem que estará sempre online para uso, o armazenamento de dados será de acordo com a necessidade do terapeuta no decorrer do treinamento.

O acesso é disponível na forma de autenticação para paciente e terapeuta, sendo necessário apenas um dispositivo com acesso à internet que possa utilizar um navegador para entrar no site, realizar a autenticação e ser redirecionado para seu perfil de acordo com a sua finalidade no treinamento, seja paciente ou terapeuta. A plataforma também oferece acesso aos conteúdos informativos do site, sem a necessidade de login, para todos os usuários que acessarem o site.

Capítulo 2

Estado da arte

2.1 Amputações

Amputação refere-se à remoção cirúrgica, espontânea parcial ou completa de um membro ou parte projetada do corpo coberta pela pele. Isso, geralmente, ocorre no plano transversal, mas pode ser no plano longitudinal se parte de um membro for removida. De acordo com (KOHLER, CIEZA, *et al.*, 2009), a incidência e prevalência da amputação são difíceis de determinar com precisão por várias razões, incluindo:

- Múltiplas etiologias patológicas culminando em amputação;
- Várias definições de amputação clinicamente significativa;
- Múltiplas amputações realizadas no mesmo indivíduo no mesmo membro, mas em níveis sequencialmente mais proximais;
- Dificuldade subjacente e incompletude da coleta de dados retrospectivos em muitos dos estudos.

Em relação ao Brasil, o termo amputação é utilizado para definir a retirada total ou parcial de um membro do corpo, sendo considerado um processo reconstrutivo de uma extremidade (CARVALHO, 2003; CAVALCANTE, 2018). Dentre as maiores causas de amputações estão:

- Doenças: 76 por dia, ou seja, 27.800 por ano (MONTIEL, VARGAS e LEAL, 2012);
- Acidentes de trabalho: 12 por dia, ou seja, 4.380 por ano (MACHADO, 2015);
- Acidentes de trânsito: 44 por dia, ou seja, 16.200 por ano (SAÚDE, 2013).

Dados identificados pelo Ministério da Saúde para censo de 2011 detalha a maioria das doenças que causam amputação no país, conforme Tabela 1.

TABELA 1 - Frequência de procedimentos de amputação no SUS por causas.

Nº	Causas	Frequência	%
1	Causas externas (acidentes)	16.294	33,1%
2	Doenças infecciosas e parasitárias	8.808	17,9%
3	Doenças do aparelho circulatório	7,905	16,1%
4	Diabetes	6.672	13,6%
5	Gangrena	5.136	10,4%
6	Doenças do sistema osteomuscular e do tecido conjuntivo	2.961	6,0%
7	Neoplasias	957	0,5
8	Doenças da pele e do tecido subcutâneo	230	0.5%
9	Malformações congênitas e deformidades anomalias cromossômicas	202	0,4%
Total		49.165	100%

Fonte: Brasil. Ministério da Saúde. Secretaria de Atenção à Saúde (2013, p. 08).

Em uma amputação parcial o membro residual de amputação é denominado coto. Este coto é considerado como um novo membro do corpo e geralmente é utilizado para manipulação e controle de uma futura prótese (CARVALHO, 2003).

2.1.1 Níveis de amputações

A extremidade superior é um membro altamente complexo com feixes neurovasculares, vasos linfáticos, músculos e ossos que se unem e formam um apêndice funcional. (MADURI e AKHONDI, 2020).

1- A desarticulação do ombro envolve a remoção completa do úmero da glenóide. Quando possível, a escápula deve ser retida para evitar desfiguração das costas. (MADURI e AKHONDI, 2020).
2- Transumeral pode ocorrer em qualquer comprimento do úmero (MADURI e AKHONDI, 2020).
3- As desarticulações do cotovelo envolvem a remoção do rádio e da ulna completamente do úmero (MADURI e AKHONDI, 2020).
4- Transradial é uma amputação abaixo do cotovelo, entre a articulação do punho e a articulação do cotovelo (BOCCOLINI, 2000).
5- A desarticulação do ombro envolve a remoção completa do úmero da glenóide. Quando possível, a escápula deve ser retida para evitar desfiguração das costas. (MADURI e AKHONDI, 2020).

6- Transumeral pode ocorrer em qualquer comprimento do úmero (MADURI e AKHONDI, 2020).
7- As desarticulações do cotovelo envolvem a remoção do rádio e da ulna completamente do úmero. (MADURI e AKHONDI, 2020).
8- Transradial é uma amputação abaixo do cotovelo, entre a articulação do punho e a articulação do cotovelo (BOCCOLINI, 2000).
9- A desarticulação do punho envolve a remoção dos ossos carpais e de todas as estruturas distalmente.
10- Transcarpiana trata – se dá amputação dos dedos (BOCCOLINI, 2000).

O nível de amputação costuma ser um fator determinante para reintegração social e profissional, um estudo feito por (FERNÁNDEZ, ISUSI e GÓMEZ, 2000) afirma que 91% que voltaram para o trabalho tinham em seu braço amputado, articulação até o cotovelo. Apesar da função vital da mão no membro superior, acredita-se que reter o cotovelo ajuda a realizar certos movimentos diários que são necessários no local de trabalho.

2.2 Tratamentos atuais para pacientes com amputação

O tratamento padrão recomendado para pacientes com amputação é por muitas vezes dispendioso e mal administrado, um estudo feito por (RODRIGUES, 2011) em um Grupo de Prótese, como é denominada a equipe multidisciplinar que atende a pacientes amputados em reabilitação, no qual é composto por ortopedista, fisiatra, fisioterapeuta, assistente social, terapeuta ocupacional e psicólogo, relatou que algum tempo após o início do processo de reabilitação, durante o período de elaboração do material da prótese o paciente aguardava em casa, para só começar o chamado treinamento de marcha, já com a prótese, pelo tempo que fosse necessário. Esse tempo ocioso do paciente poderia ser usado para continuar o tratamento, ele poderia iniciar o treinamento, comunicar possíveis problemas e antecipar ajustes que poderiam serem necessários na prótese.

2.3 Conclusões

Os tratamentos atuais viabilizam a melhor aceitação e adaptação do paciente, porém não é o que costuma acontecer. O paciente, no momento que recebe a prótese, em alguns casos, não encara algo como positivo, pois essa passa a ser a morte da esperança de recuperar o objeto perdido, o uso da prótese torna – se, então, uma questão psíquica, já que em termos físicos ela é perfeitamente capaz de cumprir sua função (RODRIGUES, 2011). Logo, o período de pré-protetização é um ponto importante no tratamento, é necessário utilizá-lo bem para preparar o paciente.

Capítulo 3

Fundamentação Teórica

3.1 Plataforma web

De acordo com (GILLESPIE, 2010) existem diversas definições para o termo plataforma, na qual ele encontra 15 usos diferentes e é destacado 4 categorias: computacional, arquitetônico, figurativo e político. Para o tema que será abordado é válido destacar o significado computacional, no qual, (GILLESPIE, 2010) descreve como uma infraestrutura que suporta o design e uso particular de aplicativos, seja hardware de computador, sistemas operacionais, dispositivos de jogos, dispositivos móveis e formatos de disco digital.

Uma plataforma web é um sistema online hospedado em um servidor web no qual pode ser acessado pelo browser de um computador através de um protocolo HTTP ou localmente. O HTTP (Protocolo de Transferência de Hipertexto), é um protocolo da camada de aplicação da Web, está no coração da Web e é definido no [RFC 1945] e no [RFC 2616], ele é executado em dois programas: um cliente e outro servidor. Os dois, executados em sistemas finais diferentes, conversam entre si por meio da troca de mensagens, o protocolo HTTP defini a estrutura dessas mensagens e o modo como o cliente e o servidor as trocam (KUROSE e ROSS, 2013).

3.2 Arquitetura REST

De acordo com (FIELDING, 2000), uma arquitetura web possui um conjunto de restrições aplicadas internamente, e conforme é adicionado novas restrições(constraints), um novo estilo arquitetônico é formado. REST (Representational State Transfer) é definido por 6 restrições que estabelecem seu modo de operação:

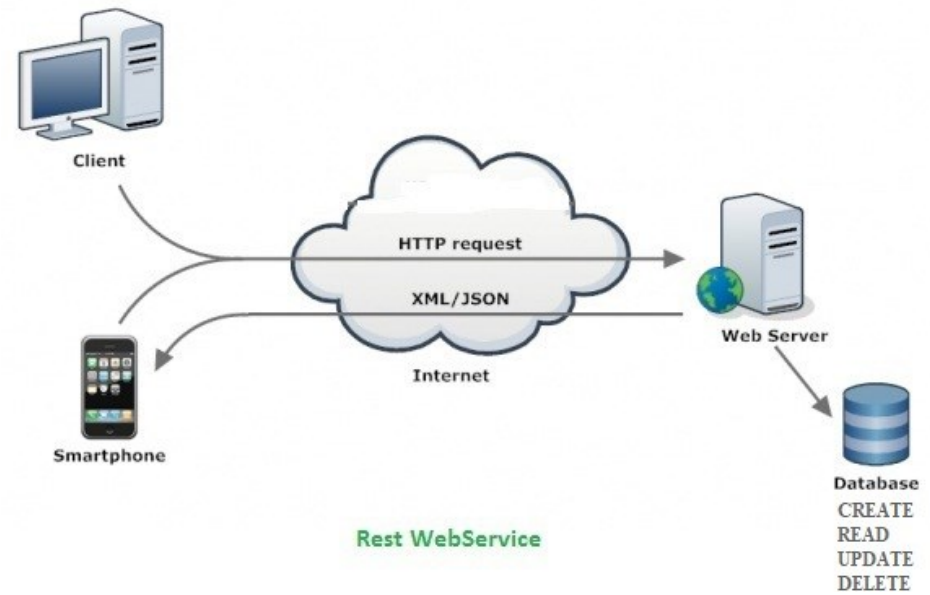
- Cacheable – cache;
- Stateless – sem estado;
- Client-server – cliente-servidor;

- Uniform interface – interface uniforme;
- Code on demand – código sob demanda;
- Layered system – sistema em camadas.

Essas restrições simplificam o modo de utilização do sistema, oferece escalabilidade além de estabelecer uma padronização segura a ser seguida. O modo de atuação web services (sistemas web), foi considerado um estilo arquitetônico que surgiu de uma necessidade em relação ao protocolo HTTP, tal como, usar novos métodos em requisições HTTP, esse modelo uniu as operações CRUD (CREATE, READ, UPDATE, DELETE) aos verbos HTTP para seguir um padrão universal. REST ou RESTFULL, em que este último significa a capacidade de um determinado sistema tem em aplicar os princípios de REST, seguir um padrão de métodos utilizados na maioria das aplicações, o exemplo a seguir será mostrado os principais, no qual é utilizado um recurso chamado Clientes:

Método (requisição HTTP)	URL	Ação	Utilização
GET	/clientes	index	Listar dados de todos clientes.
GET	/clientes/id	show	Mostrar dados de todos clientes do id selecionado.
GET	/clientes/1/edit	edit	Editar dados do cliente com id 1.
POST	/clientes	create	Criar novo cliente.
PATCH	/clientes/id	update	Atualizar os dados de um determinado cliente.
DELETE	/clientes/id	destroy	Excluir um determinado cliente.

Figura 3.1: REST web service



Fonte: (PARVEZ, 2020)

O funcionamento de uma aplicação REST é baseado em requisições (requests) e respostas (responses) realizadas à uma API (Application Programming Interface). As requisições são efetuadas através de URLs (Uniform Resource Locator). Após uma requisição feita ao servidor e seu respectivo processamento, uma resposta é retornada ao cliente que efetuou a requisição (SILVA, 2019).

3.2.1 *API*

Interface de Programação de Aplicação (API), é um conjunto de códigos de programação que permite a transmissão de dados entre um produto de software e outro, ou seja, através de uma API um sistema pode acessar certos tipos de dados que estejam em outro sistema (ATEXSOFT, 2019).

As interfaces de programação de aplicativos consistem em dois componentes:

- Especificação técnica, que descreve as opções de troca de dados entre soluções com a especificação feita na forma de um pedido de protocolos de processamento e entrega de dados.
- Interface de software, que é escrita de acordo com a especificação que a representa.

3.2.2 *Front-End*

Camada que interliga o usuário ao sistema, é uma interface gráfica desenvolvida por código em tecnologias web. Essas tecnologias são divididas em 3 principais linguagens, de marcação, de estilo e de script/programação. A seguir é citado um exemplo de cada linguagem e sua funcionalidade.

1. O HTML (Hypertext Markup Language) não é considerado por muitos uma linguagem de programação, mas sim de marcação, pois atua como organizador de informações de uma página web (MDN WEB DOCS, 2019). "Hipertexto" refere-se aos *links* que conectam páginas da Web entre si, seja dentro de um único site ou entre sites. Links são um aspecto fundamental da web. Ao carregar conteúdo na Internet e vinculá-lo a páginas criadas por outras pessoas, você se torna um participante ativo no world wide web. O HTML usa "Marcação" para anotar texto, imagem e outros conteúdos para exibição em um navegador da Web. 1. A marcação HTML inclui "elementos" especiais, como <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, , , <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, <ul>, <ol>, <li> e muitos outros.

2. CSS (Cascading Style Sheets): foi criado para estilizar elementos escritos em uma linguagem de marcação, o CSS adiciona estilos (por exemplo, fontes, cores, espaçamento) a documentos da web (W3C, 1997-12).

3. JavaScript(JS): é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecida como a linguagem de script para páginas Web, mas usada também em vários outros ambientes sem browser, tais como Apache CouchDB e Adobe Acrobat. O JavaScript é uma linguagem baseada em protótipos, multi-paradigma e dinâmica, suportando estilos de orientação a objetos, imperativos e declarativos (como por exemplo a programação funcional) (MDN, 2020). Devido as mudanças constantes na web, muitos serviços são criados, adaptados, de forma a automatizar conceitos utilizados que seguem certos padrões, a seguir é citado algumas dessas ferramentas.

- Biblioteca: conjunto de funções que são utilizados durante a construção de uma aplicação. Normalmente apresentam soluções de problemas que já foram resolvidos em determinado momento, por isso sua utilização é bem comum, mas não obrigatória. Alguns exemplos de bibliotecas são: JQuery, React, Babel, entre outros...
- Framework: basicamente é um modelo padrão a ser seguido durante a construção de uma aplicação. Quando um determinado conjunto de bibliotecas é relacionado e

interligado é possível criar a partir dessa ligação um framework. Eles são muito utilizados, cada qual com uma vantagem e desvantagem sobre o seu uso, alguns exemplos são: Bootstrap, AngularJS, Vue.js, Next.js, entre outros...

3.2.3 Back-End

Ao contrário do *Front-End* esta camada provê dados, ou seja, é possível manipular os dados que circularão dentro do sistema. No B*ack-End* é possível:

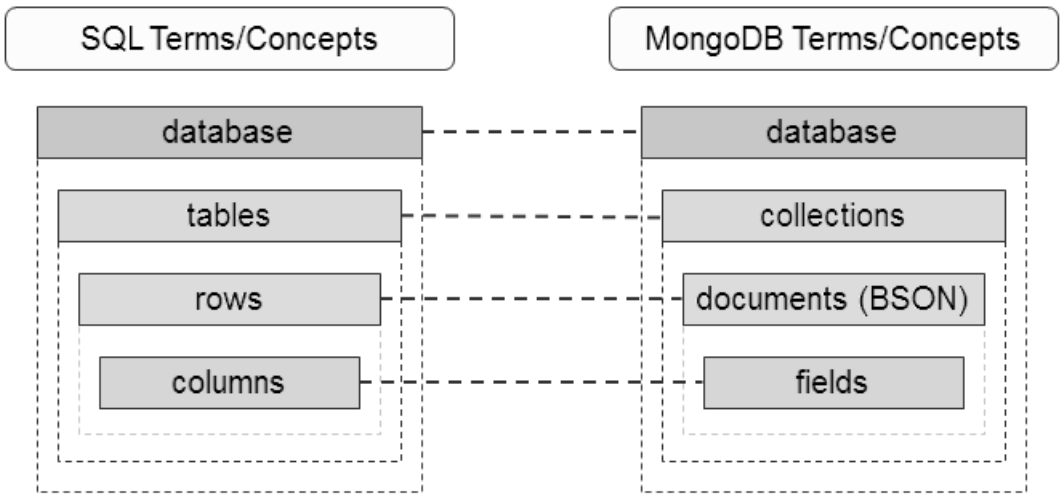
- Criar domínio para sistemas operacionais em servidores;
- Criar regras para o sistema (as técnicas de segurança são feitas aqui);
- É possível analisar informações, dados e obter relatórios de problemas;
- Utilizar Banco de Dados para armazenar dados necessários para aplicação.

O Back-End é o local em que se mapeia todas as informações do sistema e interliga com seu local de armazenamento, que irá fornecê-las quando necessário. Existem diversas linguagens que podem ser utilizadas para operar essa camada do sistema, desde linguagens consideradas “alto nível” por estarem mais próximas da linguagem humana, alguns exemplos são: Python, JavaScript, PHP, Swift, Rails, como as consideradas “baixo nível” por se aproximarem mais da linguagem das máquinas, por exemplo: C e C++.

3.2.4 Banco de Dados

O armazenamento organizado de informações é algo utilizado desde o início da computação, porém os métodos utilizados evoluem conforme as necessidades aumentam. O modo de operação de um Banco de Dados nada mais é que um agrupamento de dados que se relacionam entre si sobre um domínio (SILBERSCHATZ, SUNDARSHAN e KORTH, 2016). Os chamados SGBD (Sistema de Gerenciamento de Banco de Dados) são softwares que respondem a comandos feitos por uma linguagem de programação específica para manipulação, alteração ou recuperação das informações, organizadas de maneira relacional e não relacional. Cada SGBD tem como função controlar o Banco de Dados, agindo como intermediário entre o Banco de Dados e o usuário.

Figura 3.2: SQL vs NoSQL



Fonte: (GUEDES, 2017)

- Banco de dados Relacional: é um Banco de Dados fundamentado no paradigma da orientação a conjuntos, teoria publicada primeiramente por Richard Dedekind e posteriormente complementado por Georg Cantor em 1874 no Journal de Crelle (UNIVERSIDADE D COIMBRA). É o Banco de Dados que armazena suas informações através de estruturas comumente chamadas tables (tabelas) e dependem da integração entre columns (colunas - atributos) e rows (linhas - registros). Sua linguagem padrão de operação é o SQL (Structured Query Language), utilizado para ser de fácil inserção e recuperação em dados tabulares. Alguns exemplos de SGBD são: Oracle, PostgreSQL, MariaDB, MySQL.
- Banco de dados Não-Relacional: Esse tipo de Banco de Dados veio para fornecer soluções para as quais os Bancos de Dados relacionais não operam de forma satisfatória. Nele é possível operar com dados mistos como imagens, mapas e tabelas (não tabuladas, ou seja, um formato não relacional). Sua linguagem de manipulação e controle é o NoSQL (Not Only SQL), sua estrutura é de acordo com o sistema que é implementado, por exemplo o Cloud Firestore, no qual seu sistema é baseado em collections (coleções), documents (documentos) e fields(campos). Alguns exemplos de SGBD são: MongoDB, Redis e Cassandra.

3.3 Visão Geral

O padrão utilizado foi parcialmente baseado no modelo de aplicação REST (Representation State Transfer), em que um dos principais autores do protocolo HTTP, Roy Fielding, ao notar a necessidade de resolver problemas relacionados a semântica de requisições HTTP, criou essa arquitetura ao invar com novos métodos HTTP. (FIELDING, 2000).

O Front-End é a designação para o local onde as tecnologias de interface são interpretadas no navegador da internet, nela ocorre a "ligação entre usuário e servidor", não ocorre regras de segurança, como coleta de informações dos usuários. Já o Back-End, é o responsável por fornecer a segurança, nele estão as regras do sistema, assim como o padrão de operação, sua atuação é diretamente na alocação dos recursos em servers ou serveless.

Existem dois modos de se interligarem o Front-End com o Back-End, pode ser da forma acoplada e desacoplada. Na forma acoplada os sistemas são interligados e totalmente dependentes, viável quando se opera em níveis de baixa escala, no qual não requer muita informação. Já a forma desacoplada ambos ambientes são totalmente independentes, operam de forma separada, por isso em sistemas mais complexos que requer alto nível de informação esse método é o mais viável.

A escolha do tipo de Banco de Dados varia de acordo com as necessidades e características do sistema. Tanto Bancos de Dados relacionais quanto os não-relacionais são bem aplicáveis e estáveis em suas aplicações, porém diferem em seu modo de atuação e gerenciamento dos dados. Atualmente, com a expansão da Cloud Computing, Bancos de Dados não-relacionais (NoSQL) estão ganhando mais espaço, por terem uma melhor performance e uma escalabilidade horizontal superior comparados a Bancos de Dados relacionais (SQL).

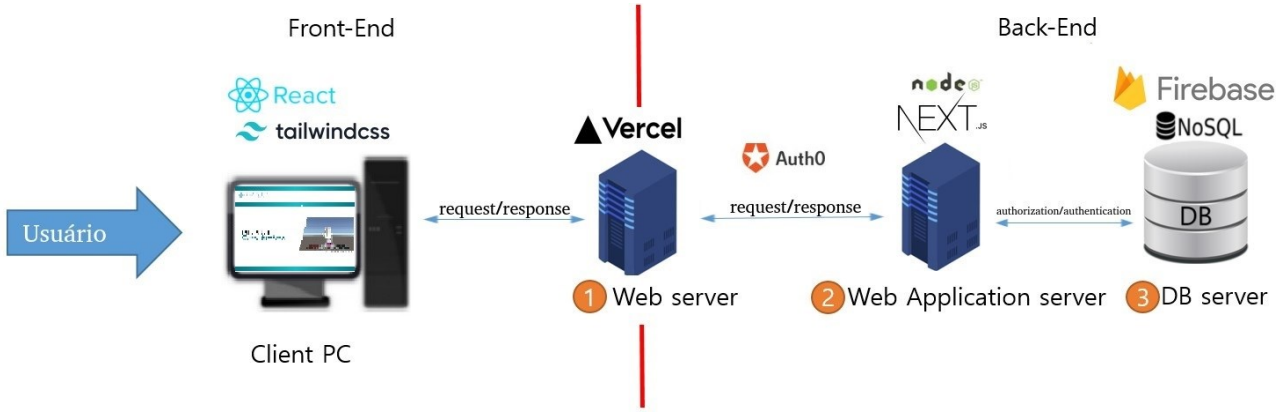
Capítulo 4

Implementação do sistema e tecnologias utilizadas

Nesse último capítulo, é mostrado uma plataforma de modo geral e conceitual. Neste capítulo será detalhado todo o modelo utilizado na implementação do sistema e tecnologias utilizadas na construção da plataforma.

4.1 Estrutura Principal

Figura 4.1: Relacionamento entre usuários e o sistema.



Fonte: adaptado de (HTTPS://DOITNOW-MAN.TISTORY.COM/)

O sistema é composto por uma estrutura principal de Front-End acoplada com o Back-End, um Banco de Dados em cloud computing (computação em nuvem), um serviço de autenticação e autorização. A plataforma tem como base o modelo Front-End e Back-End que contém algumas etapas que se interligam para formar o sistema que o usuário irá ter acesso:

1. Front-End

    a) Interface do sistema que contém as tecnologias utilizadas na parte visual do sistema, é utilizado o padrão HTML + CSS + JS, porém adaptado de acordo com a biblioteca React e o framework tailwindcss.

b) A hospedagem da plataforma será feita no Vercel, nele que serão feitas as requisições(request) do usuário ao Back-End e também o retorno das respostas(response) obtidas por essas requisições.

2. Back-End

a) O servidor de aplicação web (web application server) é o local onde será obtido as informações do lado do servidor e nele é utilizado a linguagem de programação Java Script em um ambiente de execução Node.js, o framework de renderização do servidor é o Next.js.

b) O Banco de Dados utilizado é uma plataforma em cloud computing do Google, o Firebase, seu modo de operação é em NoSQL que utiliza documentos(documents), coleções(collections) e fields(campos) em árvore como estrutura base.

4.2 Requisitos do Sistema

4.2.1 Requisitos Funcionais

Os requisitos funcionais de um sistema referem – se como um sistema deve se comportar, mais precisamente como é o seu funcionamento. Abaixo será detalhado todos os requisitos utilizados.

RF01 Acesso e visualização da plataforma

Descrição: A plataforma deve permitir a visualização das informações estáticas que não necessitam de autenticação para qualquer usuário que acessá-las.

Entrada: Endereço web da plataforma por um browser(navegador).

Processo: O usuário irá colocar o endereço da plataforma em um navegador, após o carregamento poderá navegar no ambiente da plataforma que não requer autenticação de acesso.

RF02 Cadastro de usuários

Descrição: A plataforma deve possuir a opção de cadastro por uma conta Google ou através de um e-mail e senha.

Entrada: Via conta google deve-se ao logar, selecionar a opção logar por conta google e selecionar a conta desejada e o usuário será redirecionado para página inicial do questionário. Via e-mail é preciso colocar um e-mail e uma senha e clicar em “log in” e será redirecionado para página inicial do questionário.

Processo: Caso seja feito através de uma conta Google será necessário selecionar a conta e todas a informações sobre o usuário serão adicionadas automaticamente. Para a opção via e-mail, deve-se colocar o e-mail no campo proposto juntamente com a senha a ser criada e posteriormente fazer a confirmação no e-mail selecionado.

RF03 Login de usuários

Descrição: A plataforma deve possuir a opção de login por uma conta Google ou através de um e-mail e senha.

Entrada: Via conta google deve-se selecionar o perfil. Via e-mail é preciso colocar um e-mail e senha.

Processo: Através de uma conta Google basta selecionar a conta e todas a informações sobre o usuário serão adicionadas automaticamente. Para a opção via e-mail, deve-se colocar o e-mail no campo proposto juntamente com a senha para o carregamento das informações do usuário.

RF04 Resolução do Questionário diário

Descrição: A plataforma deve permitir que os usuários(paciente) possam realizar o questionário diário sobre o treinamento.

Entrada: status do usuário(paciente) mediante ao uso da prótese.

Processo: O usuário seleciona a opção que mais se enquadra ao seu estado atual naquele dia e fazer o salvamento dessa informação.

RF05 Envio da localização

Descrição: A plataforma deve permitir que os usuários (paciente e terapeuta) forneçam sua localização em forma de coordenadas.

Entrada: coordenadas latitudinais e longitudinais.

Processo: O usuário deve clicar sobre o botão que pede sua localização e permitir o envio das coordenadas pelo browser, posteriormente fazer o salvamento dessa informação.

RF06 Edição do status

Descrição: A plataforma deve permitir que os usuários (paciente e terapeuta) possam editar o seu status fornecido durante o dia em que foi feito.

Entrada: Alteração de status e coordenadas.

Processo: O usuário deve selecionar a opção “editar status” no submenu que irá aparecer ao clicar na imagem de perfil ou em seu nome.

RF07 visualização do status

Descrição: A plataforma deve permitir que os usuários (paciente e terapeuta) possam visualizar as informações contidas no seu status de acordo com seu perfil.

Entrada: Visualização do status, da localização e distância (somente para terapeuta).

Processo: O usuário deve selecionar a opção “seu status” no submenu que irá aparecer ao clicar na imagem de perfil ou em seu nome.

4.2.2 Requisitos Não Funcionais

As requisições e/ou restrições do sistema se encontram nos requisitos não funcionais, mediante a essa informação estão a seguir os requisitos não funcionais da plataforma.

RNF01 Somente usuários autenticados (com conta google ou e-mails e senha) terão acesso ao questionário da plataforma.

RNF02 Somente o terapeuta terá acesso aos formulários de todos usuários da plataforma.

RNF03 Somente o terapeuta terá acesso a distância da sua posição em relação aos pacientes que preencherão o formulário.

RNF04 O terapeuta somente terá que fornecer sua localização para cálculo da distância em relação aos pacientes que preencherem o formulário diário.

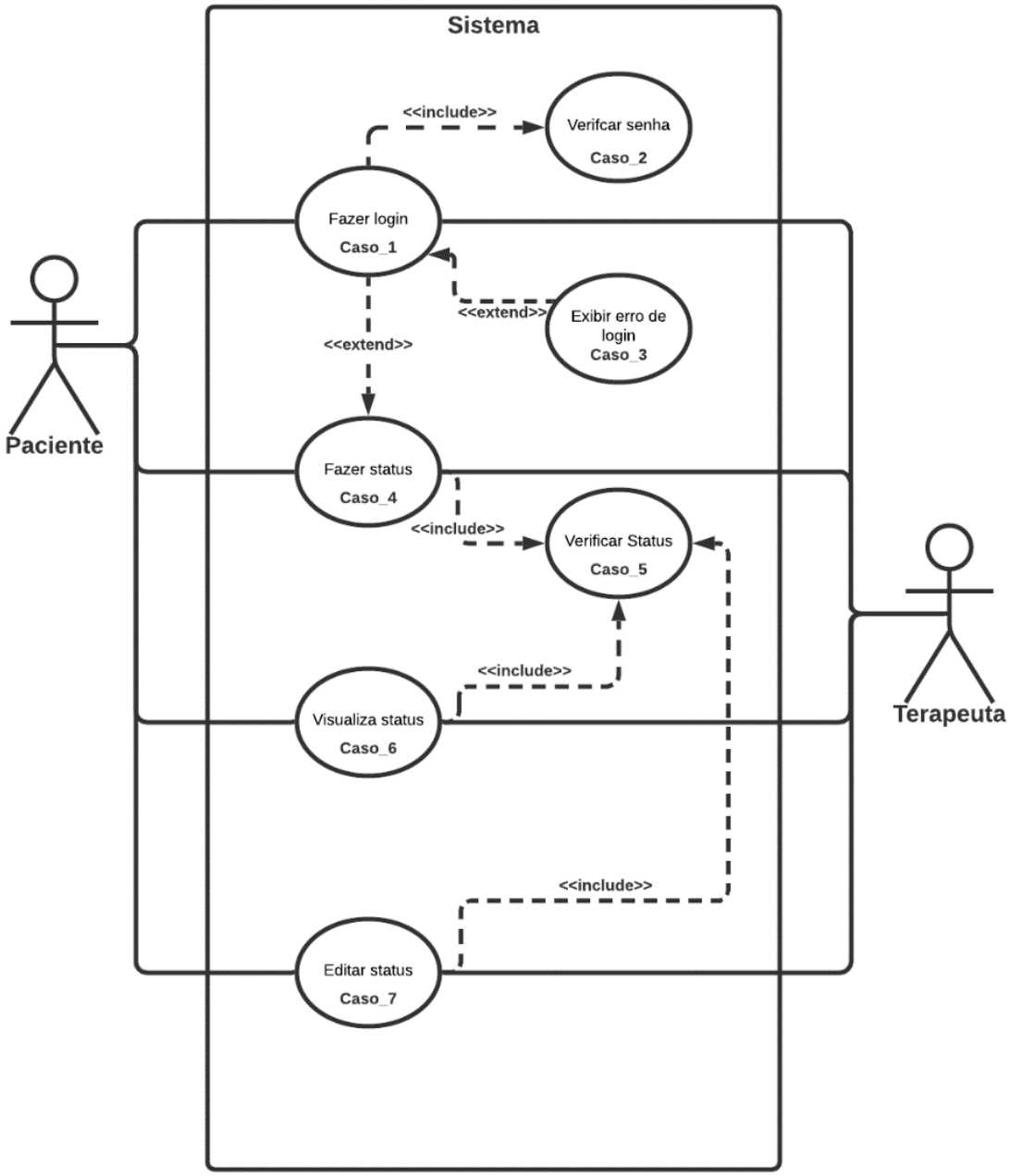
RNF05 A atualização do sistema é de forma diária para todos os usuários, e o conteúdo será salvo de acordo com a data que foi feito.

4.3 Casos de Uso

4.3.1 Casos de Uso do Sistema

O sistema oferece a interação dos pacientes e terapeutas em diferentes perfis. É possível separar o controle da coleta de dados de forma específica para cada usuário. O diagrama a seguir exemplifica o modo de interação do sistema para terapeuta e paciente.

Figura 4.2: Casos de Uso do Sistema.



Fonte: O autor (2020)

Caso_1 – Fazer login

Requisitos Associados

- **RF03**

Atores: Paciente e Terapeuta.

Resumo: O sistema permitirá ao Paciente e Terapeuta logar no sistema.

Cenário Principal

Login:

- O Paciente ou Terapeuta faz login através de uma conta google ou e-mail e senha cadastrado.
- O Paciente ou Terapeuta acessa seu perfil e visualiza seus dados, como foto de perfil e nome de acordo com a conta google ou e-mail utilizado.

Cenário Alternativo

- O Paciente ou Terapeuta faz login, através de uma conta google ou e-mail e senha cadastrado.
- O Paciente ou Terapeuta não acessa seu perfil e é exibido uma mensagem de erro.

Caso_4 – Fazer Status

Requisitos Associados

- **RF04**

Atores: Paciente e Terapeuta.

Resumo: O sistema permitirá que Paciente e Terapeuta preencher o status diário, o terapeuta somente será pedido a localização para uso no cálculo da distância dos pacientes.

Cenário Principal

Paciente:

- O Paciente acessa seu perfil e é redirecionado para preenchimento do status diário, preenche o status, fornece sua localização salva seu status.

Terapeuta:

- O Terapeuta acessa seu perfil e é redirecionado para fornecimento da localização e salva o status.

Cenário Alternativo

Caso Paciente e Terapeuta não preencham seu status diário e tente visualizar ou editar status, será redirecionado para preenchimento do status.

Caso_6 – Visualizar Status

Requisitos Associados

- **RF07**

Atores: Paciente e Terapeuta.

Resumo: O sistema permitirá ao Paciente e Terapeuta visualizar o status diário.

Cenário Principal

Paciente:

- O paciente após finalizar seu status diário, irá acessar a página para visualizar seu status diário.

Terapeuta:

- O terapeuta após informar sua localização, irá acessar a página para visualizar o status de seus pacientes, além disso irá visualizar a distância deles em relação sua localização.

Cenário Alternativo

Caso não seja feito previamente o status diário, não será possível acessar a página de visualização, tanto para o Paciente quanto para o Terapeuta. O usuário será redirecionado para página fazer status.

Caso_7 – Editar Status

Requisitos Associados

- **RF06**

Atores: Paciente e Terapeuta.

Resumo: O sistema permitirá ao Paciente e Terapeuta editar o status diário.

Cenário Principal

Caso o Paciente ou Terapeuta queira alterar o status, basta ir a página editar status e será redirecionado para fornecimento de um novo status diário.

Cenário Alternativo

Caso o Paciente ou Terapeuta não tenha feito previamente o status diário, não será possível editar status diário e com isso será redirecionado para página fazer status.

4.4 Front-End

4.4.1 Ferramentas e Tecnologias

A linguagem de programação utilizada foi JS (Java Script), biblioteca principal React, PostCSS como ferramenta de transformação em conjunto com o framework tailwindcss.

React[1] é uma biblioteca JavaScript para criar UIs (interfaces de usuário). Utiliza uma sintaxe chamada JSX que é uma adaptação do html comum em JavaScript, porém mais voltado para JS. Ao invés de separar tecnologias artificialmente colocando markup e lógica em arquivos separados, o React separa conceitos com unidades pouco acopladas chamadas "componentes" que contém ambos.

- *JSX* previne ataques de injeção, ou seja, previne ataques que aproveitam falhas do sistema. É seguro incorporar entradas de usuário em JSX, por padrão, o React DOM escapa quaisquer valores incorporados no JSX antes de renderizá-los. Assim, assegura que nunca injete algo que não esteja explicitamente escrito na aplicação. Tudo é convertido para string antes de ser renderizado. Isso ajuda a prevenir ataques XSS (cross-site-scripting).

Os componentes React permitem que a UI seja dividida em pedaços independentes e reutilizáveis, para pensar em cada pedaço isoladamente. Os componentes em React podem ser definidos ao estender React.Component ou React.PureComponent.

O pacote react-dom provê métodos específicos para o DOM que podem ser usados no nível superior da aplicação, como uma válvula de escape para sair do modelo do React caso precise.

[1] Documentação disponível em https://pt-br.reactjs.org/

- *Render():* Renderiza um elemento do React no DOM no container fornecido e retorna uma referência ao componente (ou retorna null para componentes sem state). Se o elemento do React foi previamente renderizado no container, isso vai realizar uma atualização nele e só alterar o DOM conforme necessário para refletir o elemento do React mais recente.

Hooks são uma nova adição ao React 16.8. Eles permitem o uso do state e outros recursos do React sem escrever uma classe:

- State hook: *useState* é um Hook chamado dentro de um componente funcional para adicionar alguns states locais a ele. React irá preservar este state entre renderizações, useState retorna um par: o valor do state atual e uma função que permite atualizá-lo. É possível chamar essa função a partir de um manipulador de evento ou de qualquer outro lugar, o único argumento para useState é o state inicial. O argumento de state inicial é utilizado apenas durante a primeira renderização.

- Hook de efeito: o Hook de Efeito, useEffect, adiciona a funcionalidade de executar efeitos colaterais através de um componente funcional. Quando useEffect é chamado, é como dizer ao React para executar a sua função de "efeito" após liberar as mudanças para o DOM. Efeitos são declarados dentro do componente, para que eles tenham acesso as suas props e state. Por padrão, React executa os efeitos após cada renderização — incluindo a primeira renderização.

Tailwindcss[2] é um famework de estrutura CSS que pode ser composta para construir qualquer design, diretamente em sua marcação. Utiliza um plugin PostCSS como pré-processador. O tailwind UI uma interface de usuário do tailwindcss que possui uma coleção snippets HTML pré-construídos e totalmente responsivos.

4.5 Back-End

4.5.1 Ferramentas e Tecnologias

[2] Documentação disponível em https://tailwindcss.com/docs

A linguagem de programação utilizada no Back-End foi o Java Script em um ambiente de execução Node.js. Além disso, para agilizar o processo da criação do sistema e pensando nas tecnologias utilizadas no Front-End foi utilizado o framework Next.js, uma vez que ele oferece suporte SSR (server side rendering – renderização do lado do servidor, além de webpack já configurado para React e transpilação de ES6 e ES7 (ECMAScript 6 e ECMAScript 7), o Next.js também mantém o início simples e flexível o bastante para deixar o projeto escalável para o tamanho necessário.

Next.js: é um framework web que trata grande parte dos protocolos, como o HTTP. Nele é possível fazer um request(requisição) e obter uma response(reposta) do servidor, ou seja, ele é o principal responsável por gerenciar grande parte da segurança durante a entrada e saída de dados do sistema. No Next.js é possível:

- Usar SSR (Server Side Rendering, quando um site é gerado é possível utilizar a renderização de frameworks Front-End, por exemplo quando uma informação é encaminhada para o React Dom (responsável pela renderização dos elementos React) ele renderiza a página com essa informação, porém com o Next.js é possível enviar essa página inteira construída a partir do servidor para o cliente/usuário de forma otimizável, somente quando necessário.

- Fazer APIs, ou seja, provê uma fonte de dados para aplicação.

- Criar sites estáticos, renderizando diretamente no navegador.

- Configurar o ambiente de uma forma limpa, sem precisa importar o React, uma vez que é configurado para operar com React e suas páginas aos componentes em React, por isso a importação é opcional.

O Next.js foi criado pelos mesmos criadores da plataforma que é utilizada para hospedagem do sistema, Vercel.

O Vercel [3] é uma plataforma em nuvem para sites estáticos e funções sem servidor (serveless) que se adapta perfeitamente ao seu fluxo de trabalho. Ele permite que os desenvolvedores hospedem sites Jamstack e serviços da Web que são implantados instantaneamente, escalonados automaticamente e não requerem supervisão, tudo sem

[3] Documentação disponível em https://vercel.com/docs

configuração. Para implantar um projeto existente basta se inscrever no Vercel utilizando o GithHub, GitLab ou Bitbucket – então com uma implantação para cada push – selecionando um repositório de sua conta. Nele é possível adicionar variáveis de ambiente, utilizadas para armazenar chaves e operações relacionadas à segurança da aplicação.

4.5.2 Sistema de autenticação/autorização

O Auth0[4] é uma plataforma de acesso seguro. O Auth0 permite personalizar totalmente qualquer estágio do pipeline de autenticação e autorização para se adaptar a novas políticas, novos aplicativos e novas plataformas. A seguir é listado algumas de suas funcionalidades:

- O Auth0 permite autenticação e autorização de aplicativos e APIs com qualquer provedor de identidade em execução em qualquer stack(pilha), em qualquer dispositivo ou nuvem.

- O Auth0 torna mais fácil fornecer aos usuários a capacidade de autenticação com as credenciais com as quais estão mais familiarizados, seja corporativo, social ou específico da aplicação. O widget do Auth0 oferece uma caixa de login / registro totalmente personalizável e pronta utilizando JavaScript.

- Auth0 oferece suporte aos padrões da indústria, como SAML, OpenID Connect, JSON Web Token, OAuth 2.0, OAuth 1.0a, WS-Federation e OpenID.

- Auth0 fornece SDKs para todas as plataformas populares da web.

- O Auth0 oferece acesso a relatórios e análises eficientes para que o desenvolvedor possa ver facilmente o que está acontecendo.

- O Auth0 oferece autenticação multifator, em que fornece segurança aprimorada para aplicações e ativar a autenticação multifator em minutos.

- O Auth0 permite que os usuários façam login em vários aplicativos apenas uma vez, seguindo algumas etapas simples. Além de ser executado de forma local e na nuvem.

[4] Documentação disponível em https://auth0.com/

- O Auth0 simplifica os registros e logins para usuários finais, permitindo que eles usem informações de login existentes de seu provedor de rede social.

4.5.3 Banco de Dados

O Banco de Dados utilizado é o Cloud Firestore do Firebase[5], ele possibilita armazenar e sincronizar dados entre usuários e dispositivos em escala global por meio de um Banco de Dados NoSQL hospedado na nuvem. O Cloud Firestore oferece sincronização ao vivo e suporte off-line, além de consultas eficientes a dados. A integração com outros produtos do Firebase possibilita a criação de aplicativos que de fato não precisam de servidor. O Cloud Firestore é fornecido com SDKs para dispositivos móveis e Web, além de um conjunto abrangente de regras de segurança no qual é possível acessar o Banco de Dados sem precisar manter servidor próprio. Com o Cloud Functions, um produto de computação sem servidor, é possível executar um código de Back-End hospedado que responde a alterações no Banco de Dados. Além de acessar ao Cloud Firestore com bibliotecas de cliente tradicionais (por exemplo, Node, Python, Go e Java). Além disso é possível:

- Estruturar dados com coleções e documentos. Criar hierarquias para armazenar dados relacionados e recuperar os dados necessários usando consultas expressivas. O escalonamento das consultas é feito de acordo com o tamanho do conjunto de resultados (e não do conjunto de dados). Assim, a aplicação pode ser escalonada desde o início.

- Sincronizar automaticamente os dados da aplicação entre dispositivos. Será enviado notificações sobre as alterações de dados conforme elas ocorrerem. Dessa maneira, será criado experiências colaborativas e aplicações em tempo real com facilidade. Os usuários podem acessar e alterar os próprios dados a qualquer momento, mesmo quando estiverem off-line. O modo off-line está disponível para iOS, Android e Web.

Com base na infraestrutura de armazenamento do Google, o Cloud Firestore foi criado para crescer com a empresa e por isso pode se concentrar em criar a aplicação em vez de gerenciar

[5] Documentação disponível em https://firebase.google.com/docs?authuser=0

servidores ou se preocupar com a consistência. Com a sua linguagem de segurança declarativa, pode restringir o acesso aos dados com base nos dados de identidade do usuário, na correspondência de padrões nos dados predefinidos pelo desenvolvedor. O Cloud Firestore também pode ser integrado ao Firebase Authentication para oferecer autenticação de usuário simples e intuitiva.

4.6 Estrutura

4.6.1 Desenvolvimento Front-End e back-end

A plataforma foi desenvolvida com o Front-End acoplada ao Back-End e por isso foram necessárias algumas configurações e instalações de ferramentas padrões de operação:

1. vercel-json: arquivo de utilizado para deploy(implantação) do sistema no Vercel, contém as informações sobre as variáveis de ambiente do sistema para funcionamento.

2. tailwind.config.js: Arquivo de configuração do tailwind, contém as configurações dos temas utilizados do tailwind, como os plugins utilizados no layout do sistema.

3. postcss.config.js: arquivo de configuração do Postcss, contêm os plugins que serão utilizados no Tailwind.

4. package.json: arquivo com as todas dependências do sistema, um repositório central de configurações das ferramentas utilizadas no sistema. Um pacote é um arquivo ou diretório descrito por um arquivo package.json. Um pacote deve conter um arquivo package.json para ser publicado no registro npm. Os pacotes podem ser sem escopo ou com escopo para um usuário ou organização, e os pacotes com escopo podem ser privados ou públicos

5. package-lock.json[6]: Ele descreve a árvore exata que foi gerada, de modo que as instalações subsequentes possam gerar árvores idênticas, independentemente das

[6] Documentação disponível em https://docs.npmjs.com/cli/v6/configuring-npm/package-lock-json

atualizações de dependência intermediárias. Este arquivo deve ser confirmado em repositórios de origem e serve a vários propósitos:

5.1. Descrever uma representação única de uma árvore de dependências de forma que colegas de equipe, implantações e integração contínua tenham a garantia de instalar exatamente as mesmas dependências.

5.2. Fornecer um recurso para os usuários "viajarem no tempo" para estados anteriores de node_modules sem ter que confirmar o próprio diretório.

5.3. Facilitar uma maior visibilidade das mudanças na árvore por meio de diffs de controle de origem legíveis.

5.4. Otimizar o processo de instalação permitindo que o npm ignore resoluções repetidas de metadados para pacotes instalados anteriormente.

6. next.config.js[7] é um módulo Node.js regular, não um arquivo JSON. Ele é usado pelo servidor Next.js e pelas fases de construção e não está incluído na construção do navegador.

7. Firebase-secret.json[8]: é um SDK (Software Development Kit) O SDK Admin permite que você interaja com o Firebase em ambientes privilegiados para executar ações como:

7.1. Ler e gravar dados do Realtime Database com privilégios de administrador totais;

7.2. Enviar de maneira programática mensagens do Firebase Cloud Messaging usando uma abordagem simples e alternativa aos protocolos de servidor do Firebase Cloud Messaging;

7.3. Gerar e verificar os tokens do Auth do Firebase;

[7] Documentação disponível em https://nextjs.org/docs/api-reference/next.config.js/introduction
[8] Documentação disponível em https://firebase.google.com/docs/admin/setup?authuser=0

7.4. Acessar os recursos do Google Cloud Platform, como os buckets do Cloud Storage e os Bancos de Dados do Cloud Firestore associados aos seus projetos do Firebase.

7.5. Criar seu próprio console de administração simplificado para realizar ações como procurar dados do usuário ou alterar o endereço de e-mail de um usuário para fazer a autenticação.

8. node_modules[9]: Um módulo é qualquer arquivo ou diretório dentro do diretório node_modules que pode ser carregado pela função require () do Node.js. Para ser carregado pela função require () do Node.js, um módulo deve ser um dos seguintes:

   8.1. Uma pasta com um arquivo package.json contendo um campo "principal".

   8.2. Um arquivo JavaScript.

9. styles: diretório de estilos css, estão a base, componentes e serviços gerais do tailwind.

10. public: diretório contendo todos os assets utilizados, no sistema em questão estão todas a imagens e ícones utilizados.

11. pages: diretório padrão do Next.js. Esse diretório contém as páginas do sistema, vale destacar o arquivo _app que é responsável pela replicação do layout para todas páginas da plataforma. Como a plataforma está acoplada, esse diretório contém tanto códigos relacionados a camada Front-End quanto códigos da camada Back-End, por isso o diretório api está presente internamente no diretório pages. O diretório api contém configurações de retorno, login, logout, configurações da plataforma Auth0 além do salvamento dos dados no Firebase.

12. model: diretório no qual contém o arquivo marker.js que possui algumas configurações do Firebase para o modo de como será armazenado as informações coletadas no questionário da plataforma.

13. Lib: diretório que contém os arquivos:

[9] Documentação disponível em https://docs.npmjs.com/about-packages-and-modules

13.1. geo.js: arquivo de configuração da geolocalização, ou seja, calcula a localização do usuário para ser usada posteriormente.

13.2. geo.js: arquivo de configuração da geolocalização, ou seja, calcula a localização do usuário para ser usada posteriormente.

13.3. db.js: arquivo de inicialização e conexão da plataforma com o Firebase.

13.4. datatime.js: arquivo que obtenção da data atual em que sistema está sendo utilizado pelo usuário.

13.5. Auth0: arquivo de configuração para inicialização do Auth0 no sistema.

13.6. AuthContext.js: arquivo de verificação para saber se o usuário está logado.

14. Componentes: diretório de configuração geral das páginas, nele estão o footer (rodapé), header (cabeçalho), padrões das páginas. Está contido dentro desse diretório as configurações da navbar (barra de ferramentas), para modo de atuar na versão web e versão mobile.

4.6.2 Armazenagem de dados no Firebase

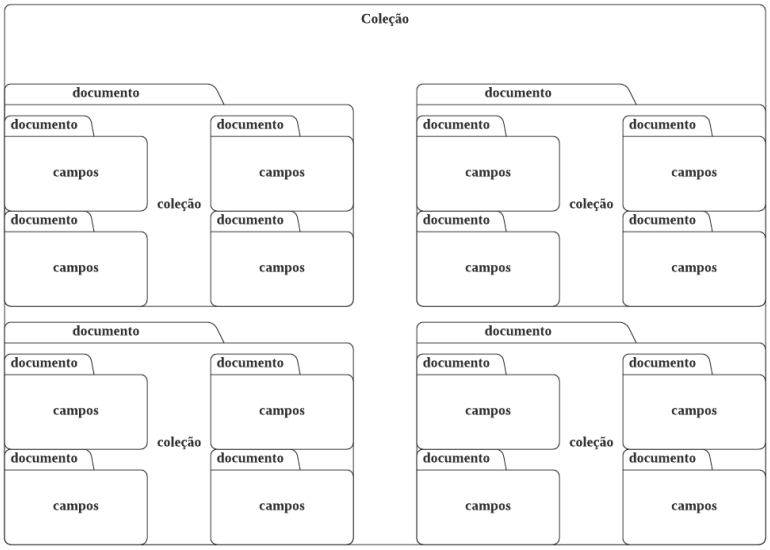
O Cloud Firestore é um Banco de Dados NoSQL orientado a documentos. Ao contrário de um Banco de Dados SQL, não há tabelas nem linhas. Em vez disso, os dados são armazenados em documentos, que são organizados em coleções.

Cada documento contém um conjunto de pares chave-valor. O Cloud Firestore é otimizado para armazenar grandes coleções de documentos pequenos.

É necessário que todos os documentos sejam armazenados em coleções. Os documentos podem conter subcoleções e objetos aninhados, que podem incluir campos primitivos, como strings, ou objetos complexos, como listas.

Coleções e documentos são criados implicitamente no Cloud Firestore. Basta atribuir dados a um documento dentro de uma coleção. Se a coleção ou o documento não existir, o Cloud Firestore o criará.

Figura 4.3: Armazenamento de dados no Firebase.



Fonte: O autor (2020).

A plataforma está definida em / coleção / documento / coleção / documento | campo.

Capítulo 5

Resultados

5.1 Plataforma Web

5.1.1 Páginas informativas

O sistema possui 3 páginas estáticas representadas pelas Figuras 5.1, 5.2, 5.3, com conteúdos relacionados ao tema proposto inicialmente, suas informações são baseadas na cartilha de orientação a paciente (INTO, 2016) e na tese de mestrado de (LIMA, 2019).

Figura 5.1: Tela inicial da plataforma web.

Fonte: O autor (2020).

Figura 5.2: Tela Sobre da plataforma web.

Fonte: O autor (2020).

Figura 5.3: Tela Terapeuta da plataforma web.

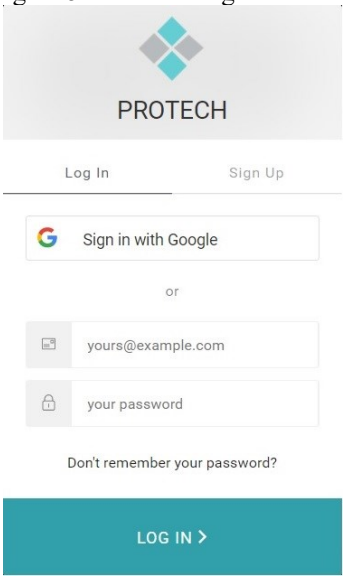
Fonte: O autor (2020).

Essas páginas tem como função informar e educar o paciente ou qualquer outro usuário que tenha acessado o site naquele momento. São páginas estáticas que não requerem autenticação ou autorização para sua visualização.

5.1.2 Páginas de acesso restrito

São páginas que requerem autenticação para visualização do seu conteúdo. Seu acesso é feito após o login que se encontra na navbar (barra de navegação) do site. É válido notar que o sistema é adaptado para acesso mobile, no qual não há perdas de informações independente do tamanho da tela do aparelho utilizado pelo usuário, para seu acesso.

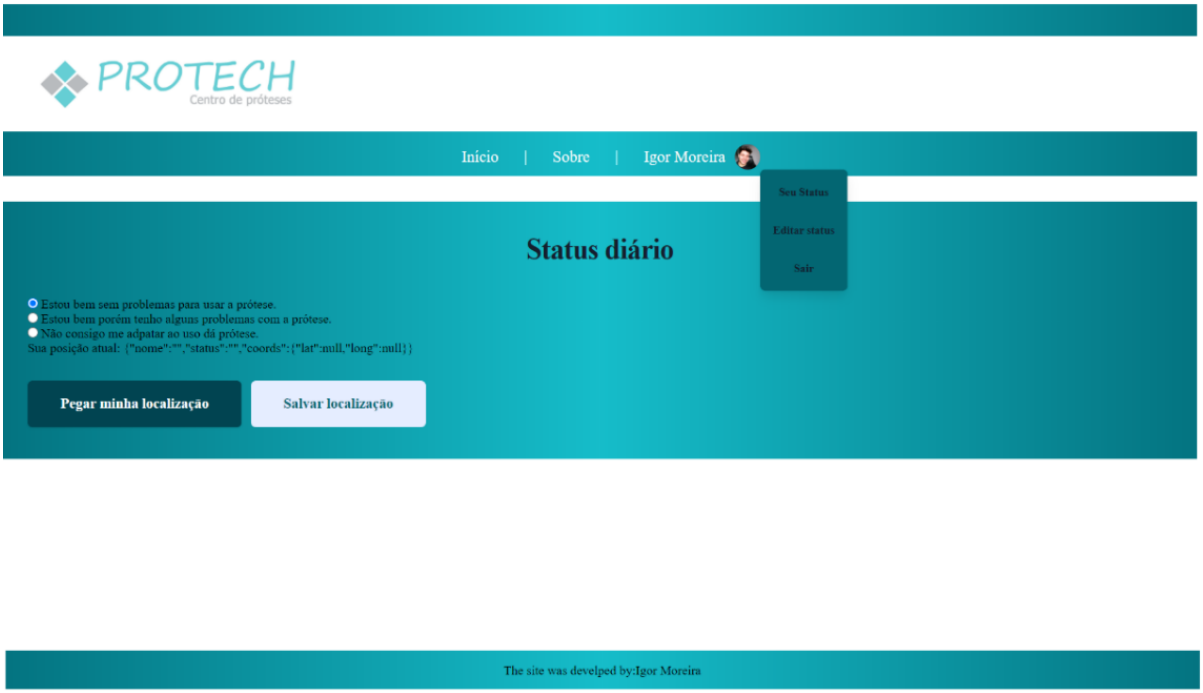
Figura 5.4: Tela de login/cadastro.



Fonte: O autor (2020)

A Figura 5.4 é apresentada após o redirecionamento de login, no momento que é clicado pelo usuário. Seu design é projetado para ser adaptado para a versão mobile, as figuras a 5.5, 5.6, 5.7, 5.8 possuem duas versões, a primeira é o padrão web convencional, feito para um computador, já a segunda versão é o padrão mobile feito para um smartphone.

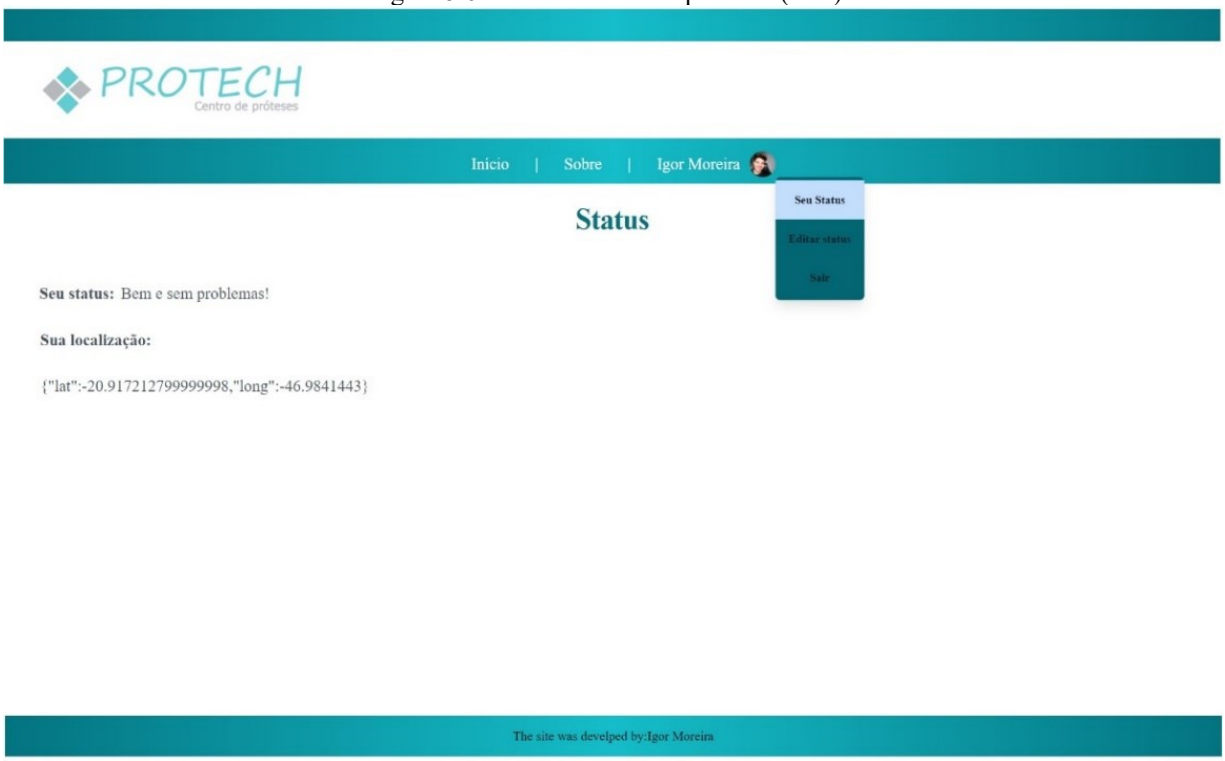
Figura 5.5: Tela do questionário do paciente (web).



Fonte: O autor (2020)

A Figura 5.5 representa o questionário de avaliação do paciente sobre sua condição ao utilizar a prótese, nela o paciente seleciona sua condição atual, posteriormente envia sua localização e finaliza salvando no Firebase.

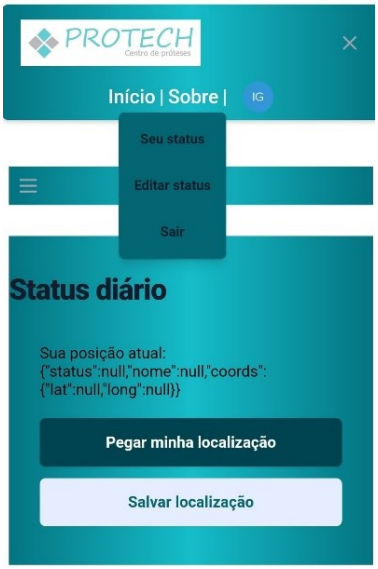
Figura 5.6: Tela do Status do paciente (web).



Fonte: O autor (2020)

A Figura 5.6 representa o status da autoavaliação(web) feita pelo paciente sobre sua condição, no questionário anterior. Nela estão as informações enviadas pelo paciente na página questionário.

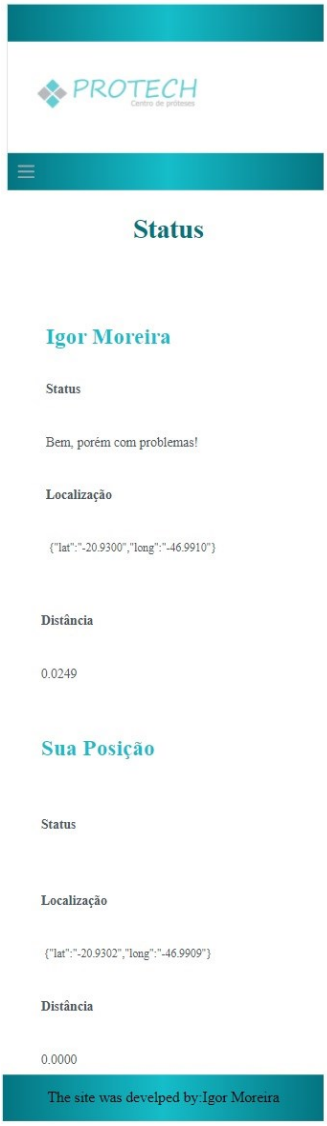
Figura 5.7: Tela do questionário do paciente (mobile).



Fonte: O autor (2020)

A Figura 5.7 representa a coleta da localização para fins comparativos do terapeuta (versão mobile).

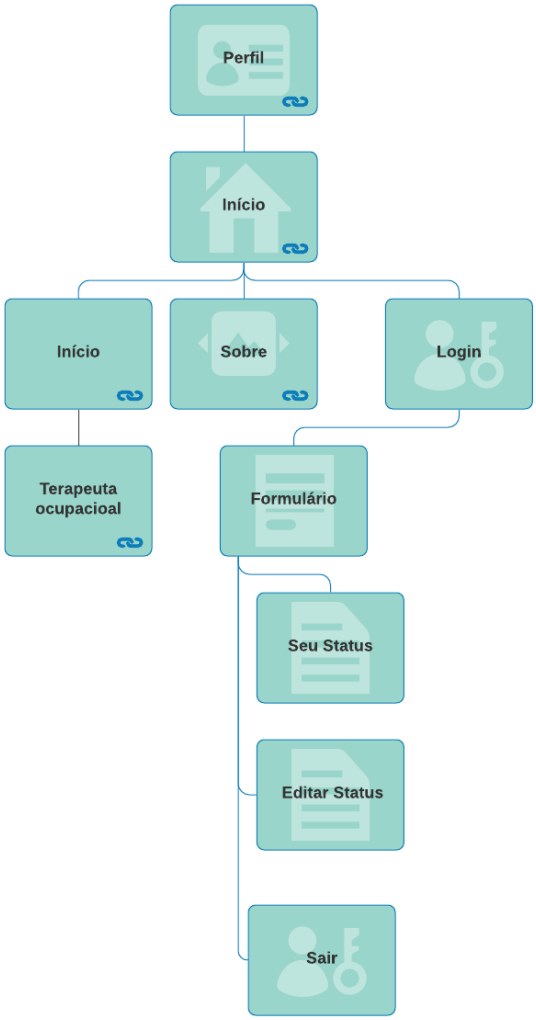
Figura 5.8: Tela do Status do terapeuta (mobile).



Fonte: O autor (2020)

A Figura 5.8 representa a visualização dos status de todos pacientes pelo terapeuta (versão mobile), as informações contidas são: status, localização e distância que os pacientes se encontram do terapeuta.

Figura 5.9: Mapa do site



Fonte: O autor 2020

A Figura 5.9, contém as informações da plataforma através do sitmap (mapa do site), nela estão os caminhos para as páginas e tipo de conteúdo a ser acessado de forma ilustrativa.

Capítulo 6

Conclusões e Trabalho Futuros

Este trabalho apresentou a construção de um sistema para auxiliar no processo de treinamento de pessoas com amputação de membros superiores. Todo o sistema foi projetado para ser portátil e acessível. Por isso, sua hospedagem foi feita numa plataforma em nuvem sem servidor físico (serveless), que fez o desenvolvimento do sistema ser mais rápido e prático.

A Internet, atualmente, está cada vez mais acessível e os aparelhos utilizados para seu acesso como, computadores, smartphones, tablets, estão cada vez mais comuns e com preços reduzidos, isso porque o desenvolvimento e a melhora constante dos hardwares barateiam o custo final desses produtos. Pensando nisso, todo o desenvolvimento do sistema foi projeto com foco na compatibilidade com esses equipamentos.

O sistema conta com a interação direta do paciente e terapeuta, nele contém a coleta de informações de ambos usuários, no caso do paciente, informações sobre a rotina diária do treinamento na utilização da prótese, já em relação ao perfil do terapeuta é possível observar esses resultados e também a localização do local que foi feita essa autoavaliação, para que ele possa ter maior conhecimento sobre o caso em questão. Além disso, é possível acessar a plataforma para obter informações sobre como funciona o treinamento e o local onde o paciente será acompanhado, de forma irrestrita para todos que acessarem a plataforma.

Como trabalho futuro, pretende-se aumentar o questionário atual, estender para auxiliar também durante o tratamento com os demais profissionais que acompanham o paciente nesse processo de adaptação. Por isso será necessário separar as camadas de visualização (Front-End) e coleta de dados (Back-End), uma vez que com o aumento da plataforma, a complexidade do sistema também aumentará e isso faz com que um sistema acoplado (Front-End e Back-End dependentes) tenha mais riscos a falhas em sua execução e ser mais trabalhoso a manutenção do código, caso queira modificar ou reparar o sistema. Por isso é viável esse desacoplamento à medida que o sistema cresce e demanda mais recursos.

Referências Bibliográficas

ATEXSOFT. What is API: Definition, Types, Specifications, Documentation. **AltexSoft**, 2019. Disponivel em: <https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/>. Acesso em: 01 nov. 2020.

BOCCOLINI, F. Reabilitação: amputados, amputações e próteses. **Robe**, São Paulo, n. 2, 2000.

BRADWAY, J. K.; MALONE, J. M.; RACY, J. Psychological adaptation to amputation: an overview. **Orthotics and Prosthetics**, v. 38, p. 46–50, 1984.

CARVALHO, J. A. Amputações de membros inferiores - Em busca da plena reabilitação, São Paulo, n. 2, 2003.

CAVALCANTE, R. S. **Desenvolvimento de um jogo sério para treinamento de amputados de membros superiores**. Universidade Federal de Uberlândia. [S.l.]. 2018.

FERNÁNDEZ, A.; ISUSI, I.; GÓMEZ, M. Factors conditioning the return to work of upper limb amputees in Austrias, Spain. **Prosthetics and Orthotics International**, p. 143-147, 2000.

FIELDING, R. T. **Architectural Styles and the design of network-based software architectures**. University of California. Irvine. 2000.

GILLESPIE, T. **The Politics of 'Platforms'**. Cornell University. [S.l.], p. 347-364. 2010.

GUEDES, M. NOSQLSQL vs NoSQL, qual usar? **trinaweb**, 2017. Disponivel em: <https://www.treinaweb.com.br/blog/sql-vs-nosql-qual-usar/#:~:text=Resumindo%3A%20o%20conceito%20de%20modelo,conjunto%20de%20colunas%20ou%20documentos.>. Acesso em: 03 nov. 2020.

HTTPS://DOITNOW-MAN.TISTORY.COM/. [Desenvolvimento Web] 1. Conceito de conexão de front-end e back-end. Disponivel em: <https://doitnow-man.tistory.com/248>.

INTO. Centro de Amputados. **Orientação a Paciente - Membro Inferio -**, Rio de Janeiro, 28 jan. 2016. 1-17. Disponivel em: <https://www.into.saude.gov.br/folhetos-e-cartilhas-para-o-paciente/cartilhas/353-centro-de-amputados-orientacao-a-pacientes-membro-inferior>.

JOHN WILEY & SONS, L. Factors affecting outcome after traumatic limb amputation. **British Journal of Surgery Society Ltd**, v. 99, 2011.

KOHLER, et al. Developing Core Sets for Persons Following Amputation Based on the International Classification of Functioning, Disability and Health as a Way to Specify Functioning. **Prosthetics and Orthotics International**, v. 33, p. 117-129, 2009.

KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a internet uma abordagem top-down**. Tradução de Daniel Vieira. 6ª. ed. [S.l.]: Pearson Education do Brasil, 2013. 72 p.

LIMA, D. A. C. D. **Uma arquitetura baseada em técnicas de Realidade Misturada para o treinamento do uso de próteses acionadas por tirantes para indivíduos com amputação de membros superiores**. Universidade Federal de Uberlândia. Uberlândia. 2019.

MACHADO, R. Acidentes com máquinas causam 12 amputações por dia no país, 2015. Disponivel em: <https://revistacipa.com.br/acidentes-com-maquinas-causam-12-amputacoes-por-dia-no-pais/>. Acesso em: 20 out. 2020.

MADURI, P.; AKHONDI, H. Upper Limb Amputation. **StatPearls [Internet]**, 2020. Disponivel em: <https://www.ncbi.nlm.nih.gov/books/NBK540962/>. Acesso em: 26 out. 2020.

MDN. JavaScript. **MDN web docs moz:** //a, 2020. Disponivel em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 02 nov. 2020.

MDN WEB DOCS. HTML: Linguagem de Marcação de Hipertexto. **MDN web docs moz:** //a, 2019. Disponivel em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. Acesso em: 01 nov. 2020.

MONTIEL, A.; VARGAS, M. A. D. O.; LEAL, S. M. C. Caracterização de pessoas submetidas à amputação. **Enfermagem em Foco**, v. 3, p. 169-173, 2012. ISSN 4.

NUNES JUNIOR, C. Tratamento fisioterapêutico na fase pré-protetização em pacientes com amputação transtibial unilateral. **Fisioterapia Brasil**, v. 10, n. 4, 2009.

NUNES, F. D. L. D. S. et al. Realidade Virtual para saúde no Brasil: conceitos, desafios e oportunidades. **Revista Brasileira de Engenharia Biomédica**, v. 27, p. 243-258, 2011. ISSN 4.

PARVEZ. Types Of Web Services SOAP,XML-RPC And Restful. **Phpflow.com**, 2020. Disponivel em: <https://www.phpflow.com/php/web-service-types-soapxml-rpcrestful/>. Acesso em: 01 nov. 2020.

REACTJS. Uma biblioteca JavaScript para criar interfaces de usuário. **React**, 2020. Disponivel em: <https://pt-br.reactjs.org/>. Acesso em: 04 nov. 2020.

RODRIGUES, L.. **Uma psicanalista em uma equipe multidisciplinar: atendimento a pacientes com amputação em reabilitação com prótese**. Universidade de São Paulo - Instituto de Psicologia. São Paulo. 2011.

SAÚDE, M. D. Diretrizes de atenção à pessoa amputada. **Biblioteca Virtual em Saúde**, 2013. Disponivel em: <https://bvsms.saude.gov.br/bvs/publicacoes/diretrizes_atencao_pessoa_amputada.pdf>. Acesso em: 25 out. 2020.

SILBERSCHATZ, A.; SUNDARSHAN, S.; KORTH, H. F. **Sistema de Banco de Dados**. [S.l.]: Elsevier, 2016.

SILVA, P.. **Plataforma web de realidade aumentada não-imersiva para tratamento de aracnofobia**. Universide Federal de Uberlândia. Uberlândia, p. 6-12. 2019.

SILVA, R. A. D. **Avaliação e Certificação de Dispositivos Protéticos e Ortéticos para Membro Inferior**. UNIVERSIDADE DO PORTO. Porto, p. 16. 2014.

UNIVERSIDADE D COIMBRA. Bibiloteca Matemática - George Cantor. **Universidade de Coimbra**. Disponivel em: <https://www.uc.pt/fctuc/dmat/departamento/bibliomat/servicos/matematicos/Cantor-G>. Acesso em: 03 nov. 2020.

W3C. W3C. **W3C**, 1997-12. Disponivel em: <https://www.w3.org/Style/CSS/learning>. Acesso em: 02 nov. 2020.