ESCOLA SENAI

“PROF. DR. EURYCLIDES DE JESUS ZERBINI”

Felipe Carradori de Oliveira

Kauan Silva de Souza

Vinicius Fernando Piantoni

Vinicius Raphael Ferreira Cruz da Silva

**AAMed**

Aplicativo de Atendimento Médico

Campinas SP

2020Felipe Carradori de Oliveira

Kauan Silva de Souza

Vinicius Fernando Piantoni

Vinicius Raphael Ferreira Cruz da Silva

**AAMed**

Aplicativo de Atendimento Médico

Projeto apresentado à Escola SENAI “Prof. Dr. Euryclides de Jesus Zerbini” para obtenção do certificado de conclusão do Curso Técnico de Informática.

Orientador: Paulo Henrique Pansani

Campinas SP

2020

Felipe Carradori de Oliveira

Kauan Silva de Souza

Vinicius Fernando Piantoni

Vinicius Raphael Ferreira Cruz da Silva

**AAMed**

Aplicativo de Atendimento Médico

Trabalho de conclusão de curso aprovado como requisito parcial para obtenção do grau de técnico, do curso Técnico de Informática da Escola SENAI “Prof. Dr. Euryclides de Jesus Zerbini”.

**BANCA EXAMINADORA**

1º Examinador

2º Examinador

3º Examinador

Local e data

**DEDICATÓRIA**

Dedicamos esse trabalho primeiramente para nossa família, que sempre nos ofereceu todo suporte possível para que realizássemos o curso, dedicamos também aos nossos professores que foram responsáveis por gerenciar e aplicar todo conteúdo do curso Técnico de Desenvolvimento de Sistemas e ao grupo que teve grande empenho e disposição para realizar o projeto de maneira respeitosa e agradável.

**AGRADECIMENTOS**

Temos um sentimento intenso de agradecimento pela escola SENAI Prof. Dr. Euryclides de Jesus Zerbini que nos proporcionou uma oportunidade de evoluir e aprender.

Agradecemos aos nossos professores que atribuíram um bom ensino para nosso crescimento profissional, além, de sempre buscar a melhor maneira e forma de tirar nossas dúvidas e aplicar o ensinamento de maneira correta.

E também prestamos nossos agradecimentos aos nossos companheiros de aprendizagem, que sempre mantiveram um ambiente agradável e divertido para aprender.

**EPÍGRAFE**

**“Sonhos determinam o que você quer.**

**Ação determina o que você conquista”.****- Aldo Novak**

**SUMÁRIO**

[1 INTRODUÇÃO 10](#_Toc42209121)

[2 JUSTIFICATIVA 11](#_Toc42209122)

[3 OBJETIVOS 14](#_Toc42209123)

[3.1. Objetivos Gerais 14](#_Toc42209124)

[3.2. Objetivos Específicos 14](#_Toc42209125)

[4 PRODUCT BACKLOG 15](#_Toc42209126)

[5 REQUISITOS NÃO FUNCIONAIS 15](#_Toc42209127)

[6 PREMISSAS 16](#_Toc42209128)

[7 RESTRIÇÕES 17](#_Toc42209129)

[8 ANÁLISE DE RISCOS DE UM PROJETO 18](#_Toc42209130)

[8.1. Nível e Planos de Ação para os Riscos 19](#_Toc42209131)

[8.2. Planos de ação 19](#_Toc42209132)

[9 SPRINTS 21](#_Toc42209133)

[9.1. Primeiro Sprint 21](#_Toc42209134)

[9.1.1. Product Backlog 21](#_Toc42209135)

[9.1.2. Sprint Backlog 21](#_Toc42209136)

[9.1.3. Burn Down Chart 21](#_Toc42209137)

[9.1.4. Diagramas 22](#_Toc42209138)

[9.1.5. Plano de testes 24](#_Toc42209139)

[9.1.5.1. Resultados 25](#_Toc42209140)

[9.1.6. Kanban e Retrospectiva 26](#_Toc42209141)

[9.2. Segundo Sprint 28](#_Toc42209142)

[9.2.1. Product Backlog 28](#_Toc42209143)

[9.2.2. Sprint Backlog 28](#_Toc42209144)

[9.2.3. Burn Down Chart 28](#_Toc42209145)

[9.2.4. Diagramas 29](#_Toc42209146)

[9.2.5. Plano de testes 31](#_Toc42209147)

[9.2.6. Kanban e Retrospectiva 33](#_Toc42209148)

[9.3. Terceiro Sprint 35](#_Toc42209149)

[9.3.1. Product Backlog 35](#_Toc42209150)

[9.3.2. Sprint Backlog 35](#_Toc42209151)

[9.3.3. Burn Down Chart 36](#_Toc42209152)

[9.3.4. Diagramas 36](#_Toc42209153)

[9.3.5. Plano de testes 36](#_Toc42209154)

[9.3.6. Kanban e Retrospectiva 37](#_Toc42209155)

[10 Modelo de Dados 38](#_Toc42209156)

[10.1. Diagrama Conceitual 38](#_Toc42209157)

[10.2. Modelo lógico do banco de dados 39](#_Toc42209158)

[10.3. Dicionário de dados 40](#_Toc42209159)

[11 PRINCIPAIS TELAS DO SISTEMA 41](#_Toc42209160)

[12 CONCLUSÃO 42](#_Toc42209161)

[12.1. Escreva os resultados obtidos 42](#_Toc42209162)

[12.2. Constatações 42](#_Toc42209163)

[12.3. Sugestões de possíveis aperfeiçoamentos técnicos 42](#_Toc42209164)

[13 REFERÊNCIAS 43](#_Toc42209165)

[14 GLOSSÁRIO 45](#_Toc42209166)

[15 ANEXOS 46](#_Toc42209167)

# INTRODUÇÃO

Atualmente as pessoas que precisam de uma determinada ajuda médica passam por diversas dificuldades e barreiras para que o atendimento seja realizado com sucesso e de maneira positiva. Gastam muito tempo indo em vários hospitais por falta de informações sobre disponibilidade de médicos, leitos e equipamentos. Há também a questão de, se o paciente precisar de uma ambulância, a mesma normalmente tende a demorar.

Nossa plataforma online disponibiliza para o usuário que solicite rapidamente a ajuda de um hospital, assim, o hospital recebe uma requisição com os dados do paciente e sua localização, desta forma, eles verificaram se contém disponibilidade de atendê-lo, caso haja, o hospital separa os procedimentos necessários para o atendimento e se necessário solicitam uma ambulância. Economizando tempo para atendimento do paciente e tendo noção dos casos que estão por vir no hospital.

Há aplicativos que podem complementar nosso sistema. Por exemplo, o aplicativo do sus “Meu digiSUS” tem funcionalidades como: agendar uma consulta, lista de vacinas recebidas pelo paciente e poder adicionar aos favoritos os hospitais por perto. Outro exemplo é o “Agenda Fácil” que tem a principal função fazer agendamentos para consultas e exames na rede municipal de saúde de São Paulo. Entretanto, esses aplicativos não tem uma comunicação em tempo real do paciente com o hospital ou ao contrário. Com um sistema que permite essa comunicação em tempo real, diminuiria o tempo de espera dos pacientes e os mesmos não iam precisar fazer tantas transferências em vários hospitais.

# JUSTIFICATIVA

Através de uma pesquisa de campo, no qual foi entrevistado 144 pessoas. Foi detectado um problema de comunicação entre hospitais, que tem como base o alarde com o atendimento ao paciente. Ele se baseia com a demora de verificação de disponibilidade de recursos para executar o devido atendimento.

Figura - Dados da pesquisa de campo

Segundo alguns relatos das pessoas que responderam “sim”, teve como motivo o fato de tentar passar por um determinado atendimento. O principal critério utilizado pelos consultados foi o intenso desgaste que passam no processo de atendimento (a intensa demora, verificação se o hospital tem recursos necessários para atendê-los, se há médicos à disposição).

Figura - Dados da pesquisa de campo

A principal dificuldade relatada pelos entrevistados foi achar os equipamentos necessários para o atendimento. A maioria dos relatos apresentados foi o transtorno de ficar se locomovendo entre os hospitais que tem os recursos necessários.

Na imagem abaixo contém alguns relatos e experiências dos entrevistados:



Figura - Relatos e experiências dos entrevistados

Por conta desses dados, com o intuito de melhorar o atendimento e comunicação (hospitais com outros hospitais e pacientes com hospitais), foi desenvolvido um aplicativo mobile e web, no qual o paciente deixa de enfrentar os problemas de transição entre hospitais e cada hospital sabe se pode ou não suportar o paciente que solicitará o atendimento. Tudo isso por conta de um gerenciamento que os hospitais possuem que é capaz de receber todos os dados necessários do usuário e obter uma prévia do tipo de atendimento que será necessário, com isto, o hospital poderá aceitar a requisição de atendimento do usuário. Com a aceitação positiva da requisição, o paciente será informado ao devido hospital que ele irá e automaticamente ele fica adicionado na lista de espera, ou seja, economizando tempo e facilitando para quem precisa de atendimento.

# OBJETIVOS

A seguir será apresentado os objetivos gerais exibindo resumidamente a ideia central e os objetivos específicos representando as fases para o desenvolvimento do sistema.

# Objetivos Gerais

Desenvolver um sistema mobile e web de conversação entre hospitais e pacientes, melhorando a agilidade no atendimento dos hospitais e uma melhor organização.

# Objetivos Específicos

**Web:**

* Integrar front-end com back-end através de webservice.
* Desenvolver front-end em Javascript utilizando a biblioteca ReactJS.
* Realizar teste unitário.

**Mobile:**

* Desenvolver webservice.
* Desenvolver front-end em Javascript utilizando os frameworks React Native e Expo.
* Integrar front-end com back-end através de webservice.
* Armazenar os dados, se logado, no armazenamento interno do celular.

**Back***-***end:**

* Criar banco de dados através do mongoDB.
* Desenvolver o back-end utilizando JavaScript.
* Criar CRUD do hospital e paciente.
* Realizar autenticação do hospital e paciente.
* Login e logout do hospital e paciente.
* Criar rotas de navegação.
* Realizar a busca de hospitais em um raio de 10Km.

**Testes:**

* Testar todas as funcionalidades do sistema *web*, *mobile* e *back-end.*
* Procurar possíveis erros no sistema.
* Procurar possíveis dificuldades de usabilidade para o usuário

# PRODUCT BACKLOG

O product backlog refere-se as funcionalidades que o software deverá possuir, de forma a atender as necessidades do cliente.

* Realizar cadastro do hospital.
* Realizar cadastro do paciente.
* Realizar login do hospital.
* Realizar login do paciente.
* Realizar logout do hospital.
* Realizar logout do paciente.
* Realizar a visualização dos hospitais cadastrados no mapa.
* Transferência de requisições entre os hospitais.

# REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais se referem às propriedades do sistema, não definido suas funcionalidades, mas suas peculiaridades.

* Desenvolver um layout responsivo.
* Renderização de mapa.
* Listagem de usuários e hospitais.
* Desenvolver um layout autodidata.
* Splash Screen no mobile.

# PREMISSAS

Premissas são os fatores associados ao escopo do projeto que, para fins de planejamento, são assumidos como verdadeiros, reais ou certos sem a necessidade de prova ou demonstração. Ou seja, são hipóteses ou pressupostos.

* O projeto será realizado por quatro pessoas.
* O projeto será realizado em todos os dias úteis da semana, exceto feriados e mudanças de cronograma escolar/pessoal.
* O projeto será realizado no ambiente escolar e fora dele.
* Serão disponibilizados professores para maiores dúvidas e conselhos.
* O projeto tem previsão de termino no fim de maio de do ano de 2020.
* Será disponibilizado um computador no ambiente escolar por integrante do grupo.
* Os sistemas web, mobile e back-end serão programados em ReactJS, React Native e NodeJS, respectivamente.
* O mobile estará disponível tanto para celulares IOS quanto Android.
* Será utilizado o editor de texto Visual Studio Code para a programação em JavaScript.
* Será utilizado o banco de dados não relacional MongoDB para o armazenamento dos dados da plataforma.
* A plataforma depende de conexão à internet.
* O sistema depende de ligação ao banco de dados.

# RESTRIÇÕES

Restrições são limitações impostas interna ou externamente ao projeto. Restrições podem ser a obrigatoriedade de se executar determinadas ferramentas e a forma de trabalho da equipe.

**Restrições Técnicas:**

**Banco de dados:**

* Os modelos do banco de dados devem ser elaborados com o auxílio das ferramentas Visio e brModelo
* Será usado o framework *ExpressJS* para o desenvolvimento do back-end da aplicação.
* O modelo conceitual do projeto deve ser desenvolvido inicialmente.
* O modelo lógico será desenvolvido após a construção do modelo conceitual.
* O modelo físico deve ser iniciado após a finalização dos modelos acima.
* Deverá ser usado o *MongoDB* como banco de dados.
* Os testes serão efetuados todos os dias antes do término da aula.
* O teste unitário será efetuado no final de cada Sprint.

**Plataforma Web:**

* Será usada a biblioteca *ReactJS* para o desenvolvimento da aplicação no front-end web.
* Para requisições HTTP ao servidor, deverá ser usada a biblioteca *axios*.

**Plataforma Mobile:**

* Será usado o framework React Native para o desenvolvimento da aplicação no front-end mobile.
* Deverá ser feito um protótipo navegável com o auxílio da ferramenta *Figma.*
* O aplicativo deverá ter uma interface simples e de fácil navegação.

**Restrições Administrativas:**

* Verificar as tarefas para o desenvolvimento do projeto no *Trello*.
* Compartilhamento e extração de conteúdos através da plataforma GitHub.
* Deverá ser utilizado para a documentação do projeto a ferramenta *Microsoft Word 2016.*

**Restrições Gerais:**

* Todo o código do projeto deve ser desenvolvido com o auxílio do editor Visual Studio Code.
* Todo o código do projeto deverá ser desenvolvido na linguagem de programação *JavaScript*.
* O projeto deverá ser entregue ao final do mês de maio de 2020.

# ANÁLISE DE RISCOS DE UM PROJETO

Na análise de risco são listados todos os prováveis riscos que possam acontecer durante o projeto. Em qual processo, qual é a ameaça, o impacto que o mesmo pode causar no desenvolvimento do projeto, sua probabilidade de acontecer e o risco (produto da multiplicação de “Impacto” com “Probabilidade”)



Figura - Tabela de análise de risco

# Nível e Planos de Ação para os Riscos

Definimos uma hierarquia do nível dos riscos, do mais grave para o menos grave. Assim, damos uma maior atenção às ameaças com maior impacto e probabilidade de acontecer.



Figura - Nível e plano de ação para riscos

# Planos de ação

Planos de ação para os riscos referente à tabela de riscos gerais

* **RISCO 1** – Falta de energia.
* **PLANO 1** – Rever o que foi feito, reorganizar o projeto e continuar em casa.
* **RISCO 2** – Problemas de hardware.
* **PLANO 2** – Comunicar o problema ao responsável pelo hardware e continuar o desenvolvimento do projeto em outro computador.
* **RISCO 3** – Falta de internet.
* **PLANO 3** – Continuar o desenvolvimento do projeto offline e fazer testes estáticos no código.
* **RISCO 4** – Ausência de algum integrante.
* **PLANO 4** – Continuar o cronograma e compensar as horas perdidas fora do ambiente escolar.
* **RISCO 5** – Perda de dados.
* **PLANO 5** – Realizar backup diariamente.
* **RISCO 6** – Conflito entre integrantes do grupo.
* **PLANO 6** – Resolver, de forma pacifica, o conflito internamente.
* **PLANO 7** – Atraso na elaboração do projeto.
* **RISCO 7** – Reorganizar o cronograma e adiantar a atividade fora do horário de aula.
* **RISCO 8** – Ausência de integrante durante a apresentação
* **PLANO 8** – Dividir as funções e apresentar mesmo com o integrante ausente.
* **RISCO 9** – Falha no projetor.
* **PLANO 9** – Se possível, buscar uma nova sala para a apresentação.
* **RISCO 10** – Eventos do SENAI.
* **PLANO 10** – Reorganizar o cronograma e adiantar a atividade fora do horário de aula.
* **RISCO 11** – Problemas familiares.
* **PLANO 11** – Tentar, da melhor maneira possível, ajudar o integrante do grupo que tiver esse problema.
* **RISCO 12** – Feriados.
* **PLANO 12** – Reorganizar o cronograma e adiantar a atividade fora do horário de aula.
* **RISCO 13** – Tecnologias usadas.
* **PLANO 13** – Realizar intensas pesquisas e recorrer dúvidas para pessoas com experiência na devida tecnologia.
* **RISCO 14** – Permissão de acessibilidade.
* **PLANO 14** – Comunicar o responsável que contém os acessos da máquina para que possa legalizar e instalar com sucesso todas as dependências que necessitaremos.

# SPRINTS

Apresenta-se nesta seção a documentação referente ao desenvolvimento do software.

# Primeiro Sprint

Durante o primeiro Sprint, o foco é a conclusão do banco de dados e o back-end, início do front-end web, com páginas tendo como foco o cadastro do hospital e o início do front-end mobile com as telas de login e cadastro do usuário, além do desenvolvimento da documentação.

# Product Backlog

Não houve modificações no Product Backlog.

# Sprint Backlog

* Realizar cadastro do hospital.
* Realizar login do hospital.
* Realizar logout do hospital.

# Burn Down Chart



Figura - Burn Down Chart - Primeiro Sprint

# Diagramas



Figura - Fluxograma (Página inicial da aplicação WEB)



Figura - Fluxograma (Login do hospital)



Figura - Fluxograma (Cadastro do hospital)

# Plano de testes

O plano de teste é uma maneira de encontrar defeitos e bugs no sistema para serem futuramente corrigidos. Mesmo que durante a programação sempre foram realizados alguns, ainda sim no plano são realizados testes finais passo a passo.

# Resultados

Resultados apresentados nos planos de testes



Figura - Caso de teste: Cadastro do hospital



Figura - Caso de teste: Login do hospital



Figura - Caso de teste: Logout do hospital

# Kanban e Retrospectiva

[Apresenta](http://www.devmedia.com.br/curso/introducao-a-uml/128) imagens do quadro de Kanban e as conclusões da retrospectiva.



Figura - Kanban: Primeiro dia do primeiro Sprint



Figura - Kanban: Primeira semana do Primeiro Sprint



Figura - Segunda semana do Primeiro Sprint



Figura - Terceira semana do Primeiro Sprint



Figura - Quarta semana do Primeiro Sprint

Nesse Primeiro Sprint tivemos sucesso em todos os previstos e atividades que estavam programadas para execução, a relação entre o tempo previsto se adaptou muito bem. Poderíamos ser ainda mais rápidos se tivéssemos separado as atividades de uma maneira mais elaborada para todos os membros do grupo.

# Segundo Sprint

### Product Backlog

Não houve modificações no Product Backlog.

### Sprint Backlog

* Realizar cadastro do paciente.
* Realizar login do paciente.
* Realizar logout do paciente.

### Burn Down Chart



Figura - Burn Down Chart – Segundo Sprint

### Diagramas



Figura - Fluxograma (Tela inicial do aplicativo mobile)



Figura - Fluxograma (Login do paciente)



Figura - Fluxograma (Cadastro do paciente)

### Plano de testes

#### Resultados

Resultados apresentados nos planos de testes.

****

Figura - Caso de teste: Cadastro do paciente

****

Figura - Caso de teste: Login do paciente

****

Figura - Caso de teste: Logout do paciente

### Kanban e Retrospectiva



Figura – Kanban: Primeiro dia do segundo Sprint



Figura - Kanban: Primeira semana do segundo Sprint



Figura - Kanban: Segunda semana do segundo Sprint



Figura - Kanban: Terceira semana do segundo Sprint



Figura – Kanban: Quarta semana do segundo Sprint

Todas as atividades que combinamos de realizar nesse Segundo Sprint foi feita com êxito. Mudamos a estratégia da equipe, dividimos dois membros para trabalhar juntos na parte web e os outros dois membros trabalharam juntos na parte mobile (nos ajudou muito em quesito de produtividade).

O Segundo Sprint poderia ser mais aproveitado, mas tivemos o imprevisto da pandemia de COVID-19, com esse fator acabamos tendo um déficit na produção pois tivemos o isolamento social no qual atrapalhou um pouco a nossa comunicação e ritmo.

## Terceiro Sprint

### Product Backlog

* Realizar cadastro do hospital.
* Realizar cadastro do paciente.
* Realizar login do hospital.
* Realizar login do paciente.
* Realizar logout do hospital.
* Realizar logout do paciente.
* Realizar pesquisa de hospitais próximos em um raio de 10Km.
* Transferência de requisições entre os hospitais.
* Possibilitar ao paciente uma forma de fazer a solicitação de atendimento aos hospitais próximos, e que a mesma apareça em tempo real, para os hospitais.
* Permitir que os hospitais que receberem a solicitação, possam fazer o cadastro da mesma no banco de dados.

### Sprint Backlog

* Realizar pesquisa de hospitais próximos em um raio de 10Km.
* Possibilitar ao paciente uma forma de fazer a solicitação de atendimento aos hospitais próximos.
* Permitir que hospitais que receberam a solicitação, possam fazer o cadastro da mesma no banco de dados.

### Burn Down Chart

****

Figura - Burn Down Chart - Terceiro Sprint

### Diagramas



### Plano de testes

#### Resultados

****

Figura - Caso de teste: Visualização dos hospitais no mapa



Figura - Caso de teste: Solicitação do paciente aos hospitais próximos

### Kanban e Retrospectiva

****

Figura - Kanban: Primeiro dia do terceiro Sprint



Figura - Kanban: Primeira semana do terceiro Sprint

****

Figura - Kanban: Segunda semana do terceiro Sprint

Em relação ao que tínhamos planejado não houve sucesso. Existiu uma grande baixa no desenvolvimento e ritmo por conta das mudanças de plano no nosso cotidiano atual causados pela COVID-19, além, dos feriados que surgiram no qual não estava previsto. Em tese, foi o sprint no qual tivemos uma menor produtividade e há fatores que precisam ser revistos para não ocasionar esse déficit em outros sprints.

## Quarto Sprint

### Product Backlog

Não houve modificações no Product Backlog.

### Sprint Backlog

* Transferência de requisições entre os hospitais.
* Permitir que os hospitais que receberem a solicitação, possam fazer o cadastro da mesma no banco de dados

### Burn Down Chart

### Diagramas

### Plano de testes

#### Resultatos

### Kanban e Retrospectiva

# Modelo de Dados

Esta parte do planejamento traz informações necessárias para a construção de um banco de dados para o Sistema de Gerenciamento de Acessos.

# Diagrama Conceitual

Nessa etapa se define: as entidades necessárias para a construção do Banco de Dados;



Figura - Modelo de entidade e relacionamento do banco de dados

# Modelo lógico do banco de dados

Nessa etapa se define: os atributos pertencentes a cada entidade; as chaves primárias e estrangeiras; o tipo de cada campo e valor de determinados campos.



Figura - Modelo lógico do banco de dados

# Dicionário de dados

Nessa etapa é elaborada uma organização básica dos dados do banco. Aqui são informadas as entidades, com seus respectivos campos, tipos e descrições. O banco foi desenvolvido no servidor de banco de dados Robo 3T.



Figura - Dicionário de dados - Hospital



Figura - Dicionário de dados – Paciente

****

Figura - Dicionário de dados – Suporte

# PRINCIPAIS TELAS DO SISTEMA

Descreve de maneira simples as principais telas do sistema

# CONCLUSÃO

# Escreva os resultados obtidos

Resultados obtidos

# Constatações

Constatações

# Sugestões de possíveis aperfeiçoamentos técnicos

Sugestões

# REFERÊNCIAS

NODEJS. **API Reference Documentation**. Disponível em: <<https://nodejs.org/en/docs/>>. Acesso em: 10 fev. 2020

MONGODB. **MongoDB Community Server**. Disponível em: <<https://www.mongodb.com/download-center/community>>. Acesso em: 11 fev. 2020

REACT. **Componentes e Props**. Disponível em: <<https://pt-br.reactjs.org/docs/componewnts-and-props.html>>. Acesso em: 16 fev. 2020.

REACT ROUTER DOM. **Hooks**. Disponível em: <<https://reacttraining.com/react-router/web/api/Hooks>>. Acesso em: 17 fev. 2020.

REACT ROUTER DOM. **Redirect**. Disponível em: < <https://reacttraining.com/react-router/web/api/Redirect>>. Acesso em: 17 fev. 2020.

REACT ICONS. **Font Awesome**. Disponível em: < <https://react-icons.netlify.com/#/>>. Acesso em: 20 fev. 2020.

GITHUB. **Cep-promise**. Disponível em: < <https://github.com/filipedeschamps/cep-promise>>. Acesso em: 25 fev. 2020.

REACT NATIVE. **Getting Started**. Disponível em: <<https://reactnative.dev/docs/getting-started>>. Acesso em: 25 fev. 2020.

REACT NATIVE. **Text Input**. Disponível em: <<https://reactnative.dev/docs/textinput>>. Acesso em: 02 mar. 2020.

ROCKETSEAT. **Como organizar estilos no React Native**. Disponível em: <<https://blog.rocketseat.com.br/como-organizar-estilos-no-react-native/>>. Acesso em: 02 mar. 2020.

PROPROGRAMMING101. **React Native: Custom navigation Transitions**. Disponível em: <<https://www.youtube.com/watch?v=9ajDD3W1JKk>/>. Acesso em: 03 mar. 2020.

REACT NAVIGATION. **Configuring the header bar**. Disponível em: <<https://reactnavigation.org/docs/headers/>>. Acesso em: 03 mar. 2020.

REACT NAVIGATION. **Getting started**. Disponível em: <https://reactnavigation.org/docs/getting-started>. Acesso em: 11 mar. 2020.

REACT NAVIGATION. **Drawer navigation**. Disponível em: <https://reactnavigation.org/docs/drawer-based-navigation>. Acesso em: 11 mar. 2020.

EXPO DOCUMENTATION. **AsyncStorage**. Disponível em:  
<https://docs.expo.io/versions/latest/react-native/asyncstorage/>. Acesso em: 21 mar. 2020.

MEDIUM. **Converting Stateless React Class-Based Components to Pure Functions**. Disponível em: <https://medium.com/@justintulk/converting-stateless-react-components-to-pure-functions-542cd5ad3866>. Acesso em: 18 mar. 2020.

ROCKETSEAT. **Autenticação JWT no React Native com API REST em NodeJS**. Disponível em: <https://blog.rocketseat.com.br/autenticacao-react-native-nodejs/>. Acesso em: 22 mar. 2020.

ROCKETSEAT. **Fluxo de autenticação com Token JWT no React Native**. Disponível em: <https://blog.rocketseat.com.br/fluxo-de-autenticacao-com-react-native/>. Acesso em: 22 mar. 2020.

GITHUB. **Axios-catch-error**. Disponível em: <https://gist.github.com/fgilio/230ccd514e9381fafa51608fcf137253>. Acesso em: 23 mar. 2020.

ROCKETSEAT. **[API NodeJS + Express + Mongo] Autenticação | Diego Fernandes**. Disponível em: <https://www.youtube.com/watch?v=KKTX1l3sZGk>. Acesso em: 22 mar. 2020.

CODARME. **Gerenciando Autenticação com Context API no React Native**. Disponível em: <https://www.youtube.com/watch?v=gsJ6krEJTGM&t=>. Acesso em: 21 mar. 2020.

REACT NAVIGATION: **createDrawerNavigator**. Disponível em: <https://reactnavigation.org/docs/drawer-navigator>. Acesso em: 24 mar.2020.

# GLOSSÁRIO

**AAMed**: Sigla para *Aplicativo de Atendimento Médico,* é um aplicativo que tem como finalidade melhorar o atendimento médico.

**Android**: Sistema operacional baseado no núcleo Linux, desenvolvido por um consorcio de desenvolvedores conhecido como Open Handset Alliance, sendo o principal colaborador o Google.

**Axios**: Cliente HTTP, que funciona tanto no browser quanto em node. É basicamente uma API que sabe interagir tanto com XMLHttpRequest quanto com a interface http do node. Isso significa que o mesmo código utilizado para fazer requisições ajax no browser também funciona no servidor.

**Back-end**: Responsável, em termos gerais, pela implementação da regra de negócio.

**Backup**: Cópia de segurança dos seus dados (informações) de um dispositivo de armazenamento (celulares, tablets, computadores) ou sistema (aplicativos, softwares e jogos) para outro ambiente para que esses mesmos dados possam ser restaurados em caso de perda dos dados originais ou que ocorra um acidente.

**BrModelo**: Ferramenta de código aberto voltada para ensino de modelagem de banco de dados relacional.

**Burn Down Chart**: Representação gráfica do trabalho a ser feito versus tempo.

**CRUD**: Acrônimo da expressão do idioma Inglês, Create (Criação), Read (Consulta), Update (Atualização) e Delete (Destruição). Este acrônimo é comumente utilizado para definir as quatro operações básicas usadas em Banco de Dados Relacionais.

**Expo**: Ferramenta utilizada no desenvolvimento mobile com**[React Native](https://facebook.github.io/react-native/)** que permite o fácil acesso às API’s nativas do dispositivo sem precisar instalar qualquer dependência ou alterar código nativo.

**Express.js**: Projetado para criar aplicativos da Web e APIs. Chamado de estrutura de servidor padrão de fato para o Node.js.

**Figma**: Ferramenta para design de interface.

**Fluxograma**: Representação gráfica da sequência das etapas de um processo, que permite uma análise de limites e fronteiras, fornecendo uma visão global por onde se passa o produto.

**Framework**: Abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

**Front-end**: Conversão de dados em uma interface gráfica, através do uso de HTML, CSS e JavaScript, para que os usuários possam visualizar e interagir com esses dados.

**GitHub**: Plataforma de gerenciamento e operações de software.

**HTTP**: Hypertext Transfer Protocol, é um protocolo de comunicação utilizado para sistemas de informação de hipermídia, distribuídos e colaborativos. Base para a comunicação de dados da World Wide Web.

**Interface**: Referência à conexão física e funcional entre dois sistemas ou dispositivos.

**IOS**: Sistema operacional móvel da Apple Inc. desenvolvido originalmente para o iPhone, também é usado em iPod touch e iPad.

**Javascript**: Linguagem de programação interpretada estruturada, de script em alto nível com tipagem dinâmica fraca e multiparadigma.

**Kanban**: Técnica organizacional que visa aumentar a produtividade e otimizar a realização das tarefas e entregas.

**Microsoft Visio**: Aplicativo para criação de diagramas para o ambiente Windows. Serve para gerar diagramas de diversos tipos, como organogramas, fluxogramas, modelagem de dados, diagramas de redes, plantas baixas, cartazes, etc.

**Microsoft Word**: Processador de textos amplamente equipado, projetado para ajudá-lo a trabalhar de forma mais eficiente quer você passe várias horas por dia diante do computador ou use um software de processamento de textos apenas ocasionalmente.

**MongoDB**: Software de banco de dados orientado a documentos livre, de código aberto e multiplataforma, escrito na linguagem C++.

**Node.js**: Plataforma construída sobre o motor **JavaScript** do Google Chrome para facilmente construir aplicações de rede rápidas e escaláveis. Usa um modelo de I/O direcionada a evento não bloqueante que o torna leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos.

**Product Backlog** Lista priorizada que contém breves descrições de todas as funcionalidades desejadas para um determinado produto.

**Protótipo**: Produto de trabalho da fase de testes e/ou planejamento de um projeto.

**React**: Biblioteca JavaScript de código aberto com foco em criar interfaces de usuário em páginas web.

**React Native**: Framework baseado no React, desenvolvido pela equipe do Facebook, que possibilita o desenvolvimento de aplicações mobile, tanto para Android, como para iOS, utilizando apenas Javascript.

**Robo 3T**: Software gratuito multiplataforma para gerenciamento de banco de dados MongoDB de forma gráfica.

**Servidor**: Computador com alta capacidade de processamento que disponibiliza informações e serviços aos outros computadores ligados a uma mesma rede.

**Software**: Conjunto de componentes lógicos de um computador ou sistema de processamento de dados.

**Splash Screen**: Tela de abertura do aplicativo mobile.

**Sprint**: Conceito utilizado em Scrum. Representa cada uma das etapas de um projeto, determinadas em espações específicos de tempo.

**Sprint Backlog**: Lista em que as tarefas são comprometidas a serem realizadas até um período pré-determinado.

**Teste unitário**: Consiste em validar dados válidos e inválidos via I/O sendo aplicado por desenvolvedores ou analistas de teste. Uma unidade é a menor parte testável de um programa de computador.

**Teste estático**: Tem como objetivo analisar o código sem precisar executá-lo, verificando se as boas práticas adotadas foram efetuadas. O principal objetivo dessa técnica é identificar erros de programação, tais como: Práticas ruins.

**Trello**: Sistema de quadro virtual para gerenciamento de tarefas que segue o método Kanban, muito usado no desenvolvimento com Scrum.

**Visual Studio Code**: Editor de código-fonte desenvolvido pela Microsoft para Windows, Linux e macOS.

**Webservice**: Solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis.

# ANEXOS

Se houver necessidade