



**UNIVERSIDADE FEDERAL DO CEARÁ  
CAMPUS QUIXADÁ**

**Projeto de Software Web  
Estação controladora**

**Equipe:**

Anny Caroline, Camila Stéfany, Carlos Alberto, Deyvisson Souza, Francisca Beatriz, Gilgleison Paulino, João Mateus, Johnny Marcos, José Robertty, Marcelo Martins, Raynara Lima, Ruan Derlan

**Professor:** André Braga

Outubro, 2019

## SUMÁRIO

<b>JUSTIFICATIVA</b>	<b>3</b>
<b>FERRAMENTAS UTILIZADOS</b>	<b>3</b>
<b>DESCRIÇÃO DOS COMPONENTES DE SOFTWARE</b>	<b>3</b>
<b>TELAS DA ESTAÇÃO CONTROLADORA</b>	<b>5</b>
<b>ESCALA DO MAPA E DO MAILCAR</b>	<b>7</b>
<b>TESTES UNITÁRIOS REALIZADOS</b>	<b>8</b>

## 1. JUSTIFICATIVA

Este documento tem a finalidade de descrever os componentes de código do *software web* desenvolvidos no atual projeto, requisitado para a disciplina de Sistemas Embarcados do curso de Engenharia de Computação da Universidade Federal do Ceará. Além disso, este documento tem a intenção de descrever todos os testes unitários realizados com as funções do sistema, utilizando os componentes de software.

Todos os códigos desenvolvidos pela equipe de Software Web encontra-se na pasta “Códigos desenvolvidos” no mesmo diretório onde encontra-se este mesmo documento.

## 2. FERRAMENTAS UTILIZADAS

Componentes
Node.js
React.js
Socket.IO
Pubsub Publisher
Pubsub Subscriber
ImagerMapper
Bootstrap
Google Compute Engine

## 3. DESCRIÇÃO DOS COMPONENTES DE SOFTWARE

Segue abaixo a descrição de cada uma das ferramentas de software desenvolvidas:

- **Node.js**
  - Node.js não é uma linguagem de programação. Você programa utilizando a linguagem JavaScript;
  - Node.js não é um framework Javascript. Ele está mais para uma plataforma de aplicação, na qual você escreve seus programas com Javascript que serão compilados, otimizados e interpretados pela máquina virtual V8;
  - Funções:
    - Comunicação: Receber, publicar e salvar as mensagens no banco de dados de comunicação.
- **React.js**

- O React.js é uma biblioteca e possui a finalidade de configurar as interface de usuário nas páginas web.
- Funções:
  - Tratar as informações recebidas da plataforma móvel;
  - Plotar os dados recebidos dos sensores;
  - Atributos(Botões e selecionáveis) para seleção e configuração de ações;
  - Com base nos dados recebidos, posicionar o MailCar no mapa.
- **Socket.IO**
  - Socket.IO é uma biblioteca JavaScript para aplicativos da web em tempo real. Permite comunicação bidirecional em tempo real entre clientes e servidores da Web.
  - Funções:
    - Captar, através das funções *get* as mensagens do Google Cloud Pub/Sub.
- **Pubsub Publisher**
  - Publicar mensagens na plataforma do Google Cloud Pub/Sub.
  - Funções:
    - Enviar as solicitações de modo para o MailCar.
- **Pubsub Subscriber**
  - Ler mensagens na plataforma do Google Cloud Pub/Sub.
  - Funções:
    - Receber mensagens contendo o estado do MailCar.
- **Image Mapper**
  - Permite mapear a imagem referente a planta do local e desenhar na interface a trajetória do MailCar, ilustrando sua localização.
  - Funções:
    - Fazer a movimentação do MailCar, seguindo os dados recebidos pelo Pub/Sub.
- **Google Compute Engine**
  - Compute Engine é uma aplicação do Google para criar virtual machine.
  - A máquina virtual feita pelo Compute Engine será a responsável pela hospedagem da página web
  - Funções:
    - Hospedar a página com um link global para ser acessado pelo controlador

## 4. TELAS DA ESTAÇÃO CONTROLADORA

### 4.1. Tela inicial

Na tela inicial temos o botão iniciar que com ele conseguimos acessar a tela principal.



localhost:3000/#home

Figura 1 - Tela inicial

### 4.2. Tela principal

Na tela principal podemos visualizar o modo de operação do MailCar, o botão para realizar a solicitação do modo expresso e o botão para visualizar os dados dos sensores; a bateria do MailCar e o mapa contendo a trajetória em operação e a posição atual do MailCar.

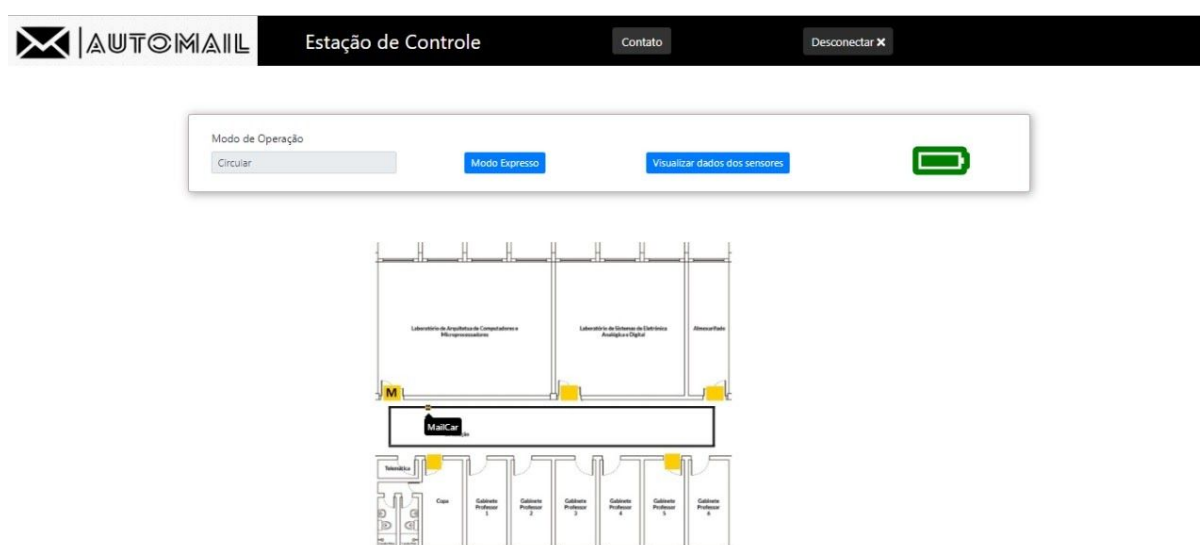


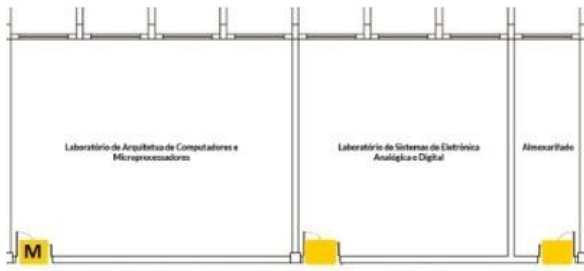
Figura 2 - Tela Principal

#### 4.3. Tela de solicitação do modo expresso

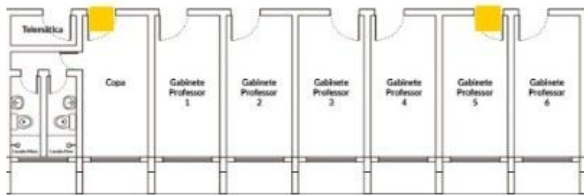
Nesta tela bgo MailBox desejado, o sentido da entrega e ao confirmar a solicitação, será enviada ao banco de mensagens do Google Cloud Pub/Sub, onde será lida pelo MailCar.

**Modo Expresso** ✕

Selecionar Mailbox:



Circulação



Sentido:

Selecione o sentido ▼

**Cancelar** **Confirmar**

Figura 3 - Tela do modo expresso

#### 4.4. Tela de visualização do estado dos sensores

Ao selecionar o botão de visualização dos sensores, direcionamos para essa tela onde a variação dos sensores é demonstrada. Na figura abaixo temos apenas duas informações, porém quando rolamos na tela, há os gráficos com as outras informações.



Figura 4 - Tela de visualização dos sensores

## 5. ESCALA DO MAPA E DO MAILCAR

A escala do mapa foi feita a partir das medidas do ambiente real. O valor resultante da medição do ambiente real foi de 1,879.8 cm o valor em pixel que representa esta área é de 481 pixel, logo o valor de 1 cm é de 0,255 pixels em "x". O valor resultante da medição do ambiente real foi de 270 cm o valor em pixels que representa esta área é de 77 pixels, logo o valor de 1 cm é de 0,285 em "y".

Levando em consideração todo o desenho do mapa contido na tela de controle e que os pontos iniciais da rota do Mailcar serão 40 cm afastada em "x" e 30 cm afastadas em "y", temos que os pontos iniciais e finais respectivamente em "x" são 19,2 pixels e 479,8 e que os pontos iniciais e finais em "y" são 229,55 e 289,45.

Para escala do Mailcar também usamos seu tamanho real de 26 cm por 15 cm, porém foi identificado que o Mailcar tinha um tamanho em relação ao mapa que era muito pequeno por esse motivo foi usado um tooltip que acompanha o carinho informando sua localização.

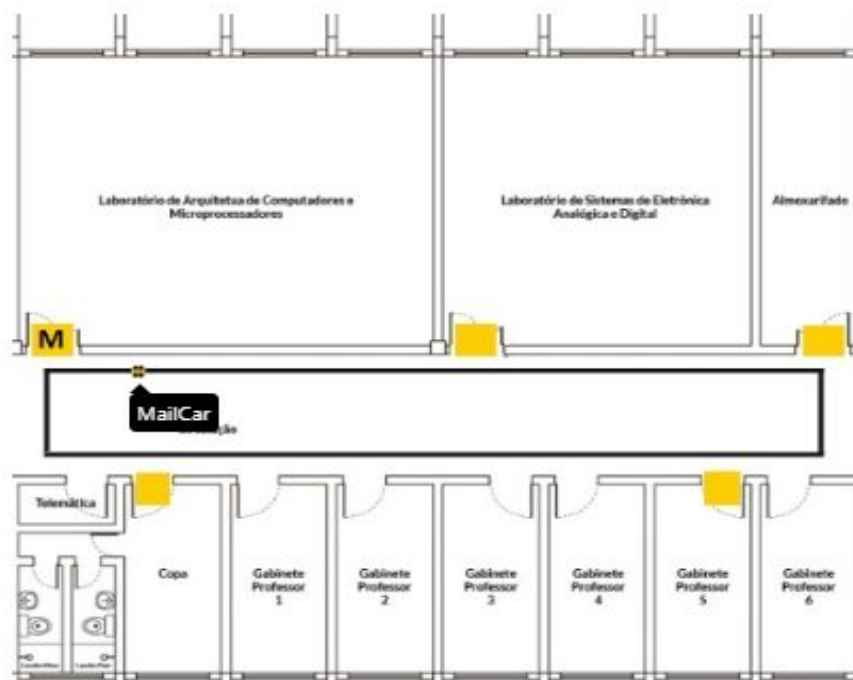


Figura 5 - Mapa de visualização do MailCar

## 6. TESTES UNITÁRIOS REALIZADOS

Os testes aqui descritos estão relacionados com os testes presentes no documento AutoMail: Plano de Testes, onde todos os testes do sistema são descritos e melhor abordados.

Foram realizados doze testes unitários, visando, principalmente, testar a precisão das medições feitas pelos sensores, validar os processos de comunicação e verificar as funcionalidades da estação.

- Teste 1

O primeiro teste unitário foi feito com a intenção de validar a modelagem proposta do grafo do sistema. Foi descrito os nós e as arestas do grafo adaptado (com distâncias fictícias) utilizando as classes Node e Edge. Após isso foi criado o suposto grafo do sistema, chamamos a função dijkstra passando como parâmetro o nó do Mailbox Master (que coincide com o nó da estação do sistema). Após isso testamos uma série de pontos e verificamos se o caminho gerado pela função travel corresponde ao menor caminho entre o nó passado como parâmetro até o nós da estação do sistema.

Foi possível verificar que, para os nós testados, a função travel sempre retornou a menor caminho entre o nó e a estação.

- Teste 2

O segundo teste foi para verificar se o modo de operação está sendo informado de acordo com o resultado dos dados da plataforma móvel informado ao controlador através da interface. Foi verificado através da seleção do módulo expresso.



Através deste teste foi possível observar que o sistema mostra os modos ao controlador de forma correta e com boa visualização baseado nos dados.

- Teste 3

O terceiro teste verifica se qualquer máquina tem acesso ao sistema. Para isso, acessamos o URL da estação em outras máquinas e constatamos que a sua visualização foi um sucesso.

- Teste 4

O quarto teste é referente ao bom funcionamento das transições de telas da estação, sua coerência e fluidez na aplicação. Utilizamos os botões que geram transições para outras telas e se apresentaram fluidas e coerentes.

- Teste 5

O quinto teste está ligado ao teste anterior, uma vez em que testamos o funcionamento dos botões, mas não o efeito de transição das telas. Utilizamos os botões e vimos que seu funcionamento está coerente com o desejado.

- Teste 6

O referente teste verifica a conexão e a leitura de mensagens através do Google Cloud Pub/Sub. Executamos um script de verificação de mensagem e constatamos o seu recebimento, como visto na figura a seguir:

```
Received message 787492909509132:
  Data: {
    "instru": {
      "accx": " ",
      "accy": " ",
      "vg": " ",
      "velox": " ",
      "veloy": " ",
      "distan": " ",
      "angulo": " ",
      "bate": "100",
      "enco1": " ",
      "enco2": " ",
      "ultra1": " ",
      "ultra2": " ",
      "ultra3": " "},
      "status": {
        "modo": "1"
      },
      "coord": {
        "x": "0",
        "y": "0"
      },
      "erros": []
    }
  }
1 message(s) received.
```

Figura 6 - Json recebido do Google Cloud Pub/Sub

- Teste 7

Para a realização do teste 7 foi preciso a conexão com a internet e a utilização da função fornecida pelo próprio Google Cloud especificada na figura a seguir.

```
router.post('/setData' , (req, res) => {  
  const topicName = 'Car_input';  
  
  let json = JSON.parse(req.query[0])  
  const data = JSON.stringify( json);  
  
  const dataBuffer = Buffer.from(data);  
  publishMessage(topicName, dataBuffer)  
  
})
```

Figura 7 - Função envio de dados para o Google Cloud

Com a resolução do teste foi possível observar a mensagem enviada pelo sistema no repositório do Google Cloud.

- Teste 8

Para o Teste 8 recebemos uma mensagem em formato de JSON do Google Cloud e verificamos como a mensagem autêntica sobre o sistema após a decodificação.

Foi verificado que todos os dados que foram decodificados de forma correta.

- Teste 9

O nono teste é referente a definir a movimentação do MailCar de acordo com o tratamento de mensagens. Como no teste 6, observamos o tratamento da mensagem recebida, onde separamos o eixo x e y, que serão usados para a localização do MailCar.

- Teste 10

No Teste 10 foi necessário enviar dados pelo tópico do Google Cloud para validar a interface da bateria, a partir desse momento foi verificado que o interface se comporta de maneira esperada, então foi variado os dados para todos os valores previstos para suprir todos os estados das interfaces da bateria.

Observou-se que a interface da bateria se comporta de maneira adequada para todos os estados da bateria.

- Teste 11

Esse teste tem como objetivo verificar a solicitação do modo expresso. Para isso, selecionamos o mailbox desejado, o seu sentido e o solicitamos. A solicitação é vista na figura a seguir:

```
{"modo":"3","destino":1,"sentido":"1"}  
Message 787543499643518 published.
```

Figura 8 - Json publicado no tópico do Google Cloud Pub/Sub

- Teste 12

Esse teste tem com objetivo de verificar a visualização dos dados sensores, é necessário clicar no botão de visualização do sensores. Com isso é possível visualizar a tela de dados do sensores que possui todas a características necessárias para cumprimento do teste.