

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**  
**PUC Minas Virtual**

**Pós-graduação *Lato Sensu* em Arquitetura de *Software Distribuído***

**Projeto Integrado**

**Relatório Técnico**

**Aisofware Tracker**

Sistema de controle de frotas

Igor Araújo

Belo Horizonte, Brasil  
Agosto, 2022

## Projeto Integrado – Arquitetura de Software Distribuído

### *Sumário*

<b>PROJETO INTEGRADO – ARQUITETURA DE SOFTWARE DISTRIBUÍDO</b>	<b>2</b>
<b>1. Introdução</b>	<b>4</b>
<b>2. Cronograma do Trabalho</b>	<b>5</b>
<b>3. Especificação Arquitetural da solução</b>	<b>6</b>
<b>3.1 Restrições Arquiteturais</b>	<b>6</b>
<b>3.2 Requisitos Funcionais</b>	<b>7</b>
<b>3.3 Requisitos Não Funcionais</b>	<b>8</b>
<b>3.4 Mecanismos Arquiteturais</b>	<b>9</b>
<b>4. Modelagem Arquitetural</b>	<b>10</b>
<b>4.1 Diagrama de Contexto</b>	<b>10</b>
<b>4.2 Diagrama de Container</b>	<b>11</b>
<b>4.3 Diagrama de Componentes</b>	<b>12</b>
<b>4.4 UML do Banco de Dados</b>	<b>14</b>
<b>5. POC (Prova de Conceito)</b>	<b>15</b>
<b>5.1 Wireframe</b>	<b>15</b>
<b>5.2 Código da aplicação</b>	<b>15</b>
Menu:	16
• Código da aplicação	16
<b>Etapa 3</b>	<b>17</b>
<b>6. Avaliação da Arquitetura (ATAM)</b>	<b>17</b>
<b>6.1. Análise das abordagens arquiteturais</b>	<b>17</b>
<b>6.2. Cenários</b>	<b>17</b>

<b>6.3.</b>	<b>Evidencia da Avaliação</b>	<b>19</b>
<b>6.4.</b>	<b>Resultados Obtidos</b>	<b>29</b>
<b>7.</b>	<b>Avaliação Crítica do Resultado</b>	<b>30</b>
<b>8.</b>	<b>Conclusão</b>	<b>31</b>
	<b>Referências</b>	<b>32</b>

## ***1. Introdução***

Locomover-se e transportar cargas sempre foi uma necessidade humana, seja por terra, água ou ar. Na ultima década, principalmente no Brasil a frota de veículos terrestres vem aumentando vertiginosamente. Segundo dados do IBGE a frota de veículos terrestres no Brasil somando todas as categorias cresceu aproximadamente 35 % comparando os números absolutos entre 2010 e 2018, mostrando como é grande e promissor este mercado.

Em 2010 o número de veículos era de 64,8 milhões, já em 2018 este número subiu para mais de 100,7 milhões, em 2021, neste cenário adverso vivido pelo mundo em plena pandemia este número cresceu para 111,4 milhões de veículos.

Focando mais nos veículos de transporte terrestre como carros, caminhões, vans, e etc, temos números consideráveis também. Em 2010 havia cerca de 37,2 milhões de automóveis e 2,1 milhões de caminhões, e em 2021 estes números vão para 59,2 milhões e 2,9 milhões respectivamente.

Com esta crescente vários novos tipos de negócios são desenvolvidos, não só pela possibilidade de levar seus produtos para os mais diversos locais do país, mas também com serviços, além dos bens de consumo. Dado a este crescimento, mecanismos para gerenciar e cuidar destes veículos são cada vez mais necessários, tanto para pessoas físicas, com sistemas de rastreamentos e geolocalização, como principalmente para empresas. Onde podem ter uma visão geral de toda a sua frota, além de trazer mais segurança e informação logística.

O intuito deste projeto de rastreamento controle de frotas para empresas, ajudando na logística e localização em casos de roubo ou mal uso do veículo. Este serviço estará disponível, inicialmente, no mercado nacional, trazendo também um sistema de fácil operação, bonito e de interface limpa. Contendo o mínimo para realizar todas as operações necessárias, de forma direta e simples.,Dado objetivo de ser um sistema simples e direto com suas funcionalidades básicas, trás para o mercado uma maior competitividade e menor custo, refletindo também no preço final para o cliente. E neste documento serão apresentados os requisitos arquiteturais, funcionais e não funcionais e as diagramações da solução para o desenvolvimento da plataforma.

## 2. Cronograma do Trabalho

A seguir é apresentado o cronograma proposto para as etapas deste trabalho.

Datas		Atividade / Tarefa	Produto / Resultado
De	Até		
01/07/2022	01/07/2022	1. Cronograma do Trabalho	Construção desta tabela
02/07/2022	02/07/2022	2. Contextualização do trabalho	Construção da contextualização deste projeto
03/07/2022	05/07/2022	3. Definição dos requisitos Arquiteturais	Lista dos requisitos Arquiteturais identificados
03/07/2022	05/07/2022	4. Definição dos requisitos Funcionais	Lista dos requisitos funcionais identificados
03/07/2022	05/07/2022	5. Definição dos requisitos Não-funcionais	Lista dos requisitos Não-funcionais identificados
03/07/2022	05/07/2022	6. Definição dos Mecanismos Arquiteturais	Lista dos Mecanismos Arquiteturais identificados
06/07/2022	08/07/2022	7. Construção dos Diagramas de Contextos – Modelo C4	Diagrama de contexto criado no Draw.io e documentado
10/08/2022	10/08/2022	8. Revisão da Etapa 1	Documento Etapa 1 revisado
10/08/2022	10/08/2022	9. Construção do vídeo de apresentação da Etapa 1	Vídeo criado da Etapa 1
10/08/2022	10/08/2022	10. Apresentação em PPT da Etapa 1	PPT
10/08/2022	14/08/2022	11. Publicação no repositório Github Etapa 1	Arquivos produzidos no Github disponíveis abertamente
11/08/2022	15/08/2022	12. Construção dos Diagramas de Contêineres	Diagramas de contêineres
13/08/2022	15/08/2022	13. Construção dos Diagramas de Componentes	Diagramas de componentes
16/08/2022	18/08/2022	14. Desenho dos Wireframes da POC	Protótipos de telas de baixa fidelidade
19/08/2022	30/09/2022	15. Código da aplicação	Aplicação com 3 requisitos implementados
30/09/2022	30/09/2022	16. Publicação no repositório Github	Arquivos produzidos no

		Etapa 2	Github disponíveis abertamente
01/10/2022	03/10/2022	17. Análise das abordagens arquiteturais	Seção do documento produzido
04/10/2022	05/10/2022	18. Cenários	Seção do documento produzido
06/10/2022	07/10/2022	19. Evidências da avaliação	Seção do documento produzido
08/10/2022	09/10/2022	20. Resultados obtidos	Seção do documento produzido
10/10/2022	11/10/2022	21. Avaliação crítica dos resultados	Seção do documento produzido
12/10/2022	13/10/2022	22. Conclusão	Seção do documento produzido
14/10/2022	19/10/2022	23. Construção do vídeo de apresentação da Etapa 3	Vídeo da etapa 3 disponível
20/10/2022	20/10/2022	24. Publicação no repositório Github Etapa 3	Arquivos produzidos no Github disponíveis abertamente

### 3. Especificação Arquitetural da solução

Esta seção apresenta a especificação básica da arquitetura da solução a ser desenvolvida, incluindo diagramas, restrições e requisitos definidos pelo autor, tal que permitem visualizar a macroarquitetura da solução.

#### 3.1 Restrições Arquiteturais

Requisitos Arquiteturais são todos os aqueles, funcionais ou não funcionais, que têm impacto direto na arquitetura do sistema. A lista a seguir mostra os requisitos de arquitetura identificados para a implementação inicial.

- [RA01] - Deve ser usado tecnologias abertas (Open source) para o desenvolvimento de toda a plataforma.
- [RA02] - Deve ser usado o serviço de nuvem da Heroku como provedora da infraestrutura necessária para a plataforma.
- [RA03] - Deve ser usado o serviço oAuth0 JWT para autenticação, com a possibilidade inicial de criação de conta diretamente na plataforma por outro usuário que tenha acesso de ADM.

- [RA04] - Deve-se utilizar uma API de um terceiro, Trackmax, para ter as informações de geolocalização e informações da frota.

### **3.2 Requisitos Funcionais**

Requisitos funcionais são todos os requisitos, recursos ou funções que são esperados no processo que o software pode satisfazer.

Em geral, requisitos funcionais representam ações que devem ser executadas pelo sistema, ou seja, requisitos funcionais são “o que o sistema deve fazer”

- [RF01] - Manter Usuários
- [RF02] - Manter Motoristas
- [RF03] - Manter Dispositivos (Veículos)
- [RF04] - Manter Grupos (Clientes)
- [RF05] - Consultar posicionamento de todos os dispositivos no Mapa por latitude e longitude
- [RF05] - Exibir veículos no mapa
- [RF05] - Exibir rota de veículo no mapa
- [RF05] - Exibir lista de veículos com informações básicas, posicionamento e último endereço de posicionamento.
- [RF06] - Consultar e Exportar Relatórios de Rotas
- [RF07] - Consultar e Exportar Relatórios de Eventos
- [RF08] - Consultar e Exportar Relatórios de Resumo

### 3.3 Requisitos Não Funcionais

Os Requisitos Não Funcionais estão associados às restrições de funcionalidades que **ditam como** o sistema deve fazer. A lista a seguir apresenta os requisitos não funcionais identificados para o desenvolvimento inicial da plataforma.

- [RNF01] - A plataforma deve habilitar a autenticação baseado no modelo JWT diretamente no sistema.
- [RNF02] - O sistema deve ser responsivo web.
- [RNF03] - O sistema deve ser responsivo para todos os tipos de tela (Computadores, Notebooks, Tablets e Smartphones).
- [RNF04] - O sistema deverá utilizar uma API de terceiro para realizar as persistências de informações dos dispositivos rastreadores.
- [RNF05] - O sistema deve operar 24x7.
- [RNF06] - Apenas usuários com permissão de Admin, tem permissão de criar e editar dados do sistema
- [RNF07] - Usuários operacionais, podem apenas acessar dados de leitura.

### 3.4 Mecanismos Arquiteturais

Os mecanismos arquiteturais representam conceitos técnicos fundamentais que serão padronizados por toda a solução. Eles são refinados durante o projeto em três estados, representados pelas três categorias de Mecanismos Arquiteturais:

- **Mecanismo de Análise**, que dá ao mecanismo um nome, uma descrição resumida e alguns atributos básicos derivados dos requisitos do projeto.
- **Mecanismo de Design**, que são mais concretos e assumem alguns detalhes do ambiente de implementação.
- **Mecanismo de Implementação**, que especifica a exata implementação de cada mecanismo.

Análise	Design	Implementação
<b>Persistência</b>	Micro ORM	Dapper
<b>Persistência</b>	Banco de Relacional	Postgres
<b>Front end</b>	Interface de comunicação com o usuário do painel	ASP.NET, Razor, html5, javascript, css
<b>Front end</b>	Navegador Web	Mozilla Firefox / Google Chrome
<b>Teste de Software</b>	Testes Unitários	XUnit
<b>Autenticação</b>	OAuth0 JWT	Microsoft.AspNetCore.Authentication.JwtBearer
<b>Tratamento de exceções</b>	Camada para tratar as exceções criando interações diferentes para usuários e técnicos	ASP.NET e C#
<b>Build</b>	Linha de comando	Terminal utilizando CLI ASP.NET
<b>Build</b>	Via Pipeline	Heroku pipeline
<b>Deploy</b>	Pipeline	Heroku / Github

## 4. Modelagem Arquitetural

Esta seção apresenta a modelagem arquitetural da solução proposta, de forma a permitir seu completo entendimento visando à implementação da Prova de Conceito (PoC) da plataforma Aisoftware Tracker na seção 5.

Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para documentação de arquitetura de software. Mais informações a respeito podem ser encontradas aqui: <https://c4model.com/> e aqui: <https://www.infoq.com/br/articles/C4-architecture-model/>. Dos quatro níveis que compõem o modelo C4 três serão apresentados aqui e somente o Código será apresentado na próxima seção (5).

### 4.1 Diagrama de Contexto

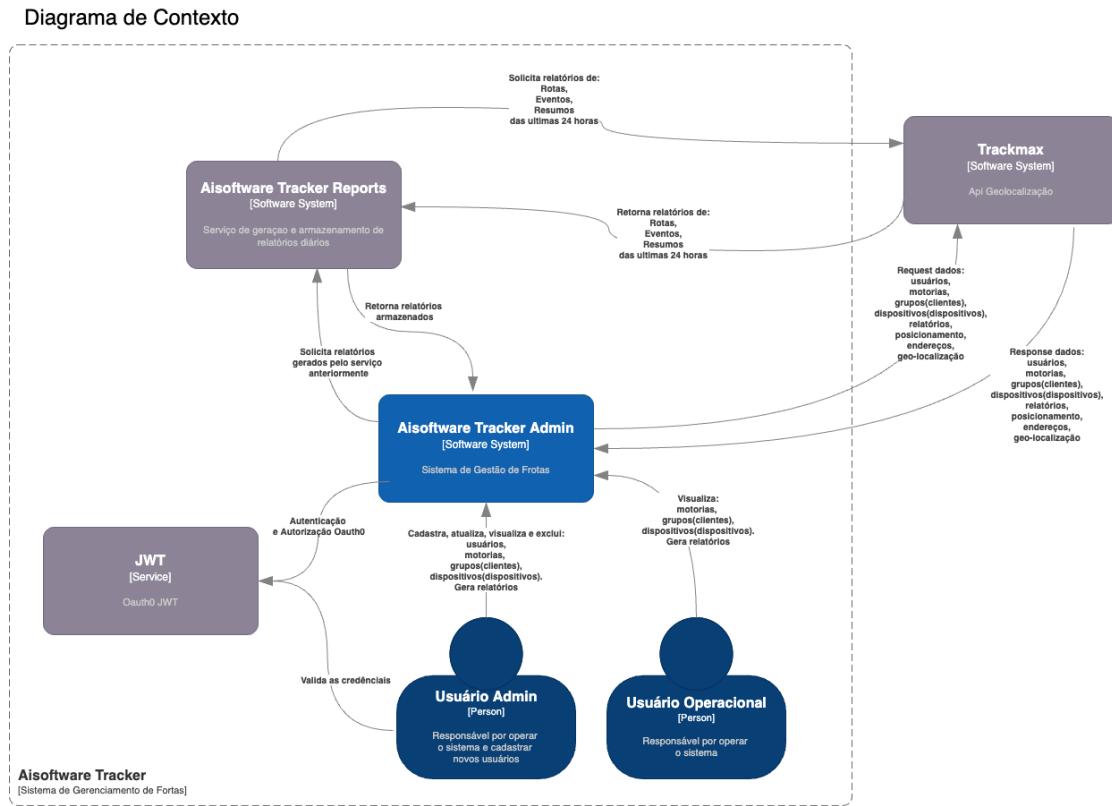


Figura 1 - Visão Geral da Solução Aisoftware Trecker

A figura 1 mostra a especificação o diagrama geral da solução proposta, com todos seus principais sistemas e pessoas envolvidas.

## 4.2 Diagrama de Container

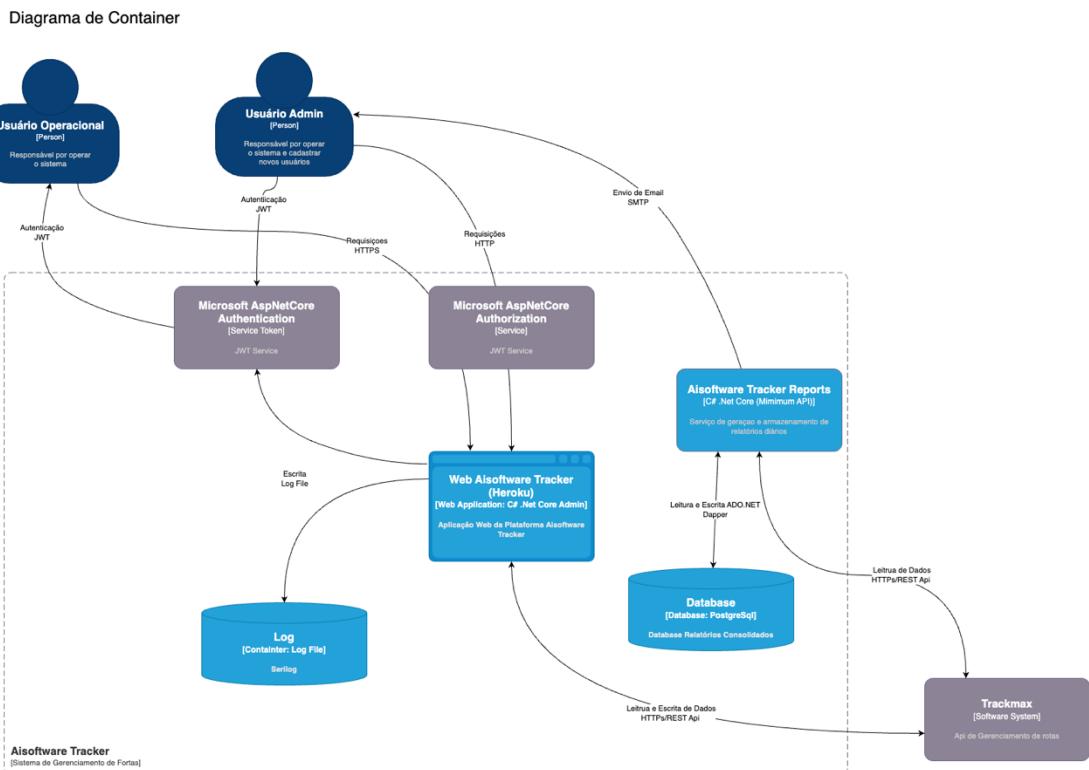


Figura 2 – Diagrama de Containter Aisoftware Trecker

A Figura 2 apresenta os containers da aplicação e a forma como estão organizados, a arquitetura visa manter somente uma fonte de verdade tratando-se dos dados da aplicação como um todo, a fonte de dados oficial é terceira API Trackmax.

O serviço de “Report” consome da API terceira e consolida relatórios no banco de dados para que posteriormente estes relatórios seja enviados via e-mail para os devidos clientes, dado ao seu volume de informações.

Toda manipulação de autenticação e autorização é feito pelo serviços JWT token alocado dentro da aplicação principal que também gerencia os cookies de comunicação com a API terceira.

A rastreabilidade de erros e inconsistências são monitoradas pelos logs do Serilog em arquivos no server e pelo Heroku Metrics.

### 4.3 Diagrama de Componentes

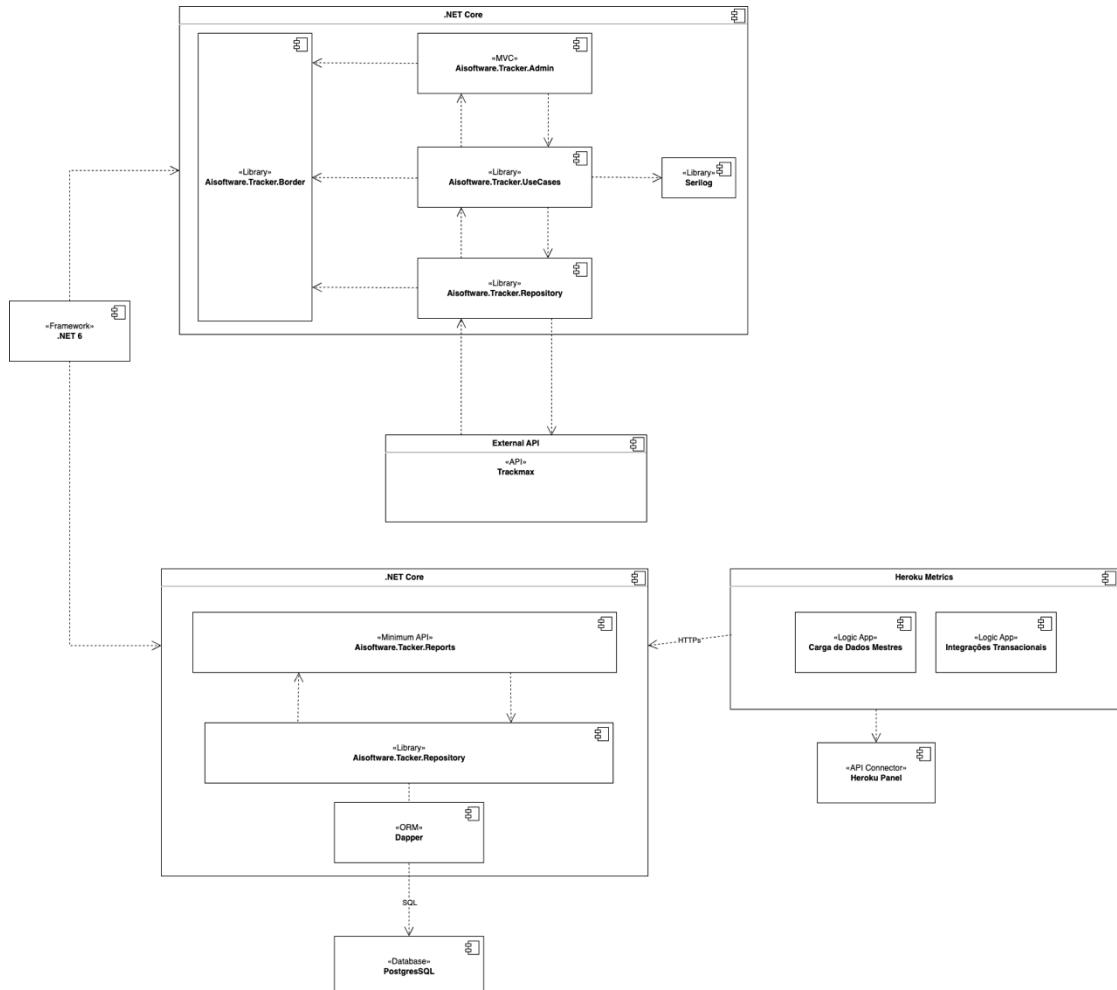


Figura 3 – Diagrama de Componentes Aisoftware Trecker

A Figura 3 apresenta os componentes da aplicação e estes podem ser detalhados como:

- Net 6 – Framework de desenvolvimento de sistemas WEB.
- Aisoftware.Tracker.Admin – Aplicação que disponibiliza a UI para utilização dos usuários.
- Aisoftware.Tracker.UseCases – Lib responsável por conter as regras de negócio da aplicação.
- Aisoftware.Tracker.Repository – Lib responsável pela comunicação externa com as APIs de terceiros.
- Aisoftware.Tracker.Reports – Aplicação responsável por gerar relatórios e os envia para determinados usuários em horários específicos via SMTP
- Dapper – Micro ORM responsável pela comunicação e interações com o banco de dados.
- Database – Banco de dados responsável por armazenar dados de dos relatórios.

- External Api – Api externa que fornece informações estratégicas para o funcionamento da aplicação.
- Serilog – Serviço responsável por gerar e escrever logs da aplicação.
- Heroku Metrics – Serviço externo que traz informações importantes da vida do container.
- Heroku Panel – Painel de demonstração das métricas coletadas do Heroku.

#### 4.4 UML do Banco de Dados

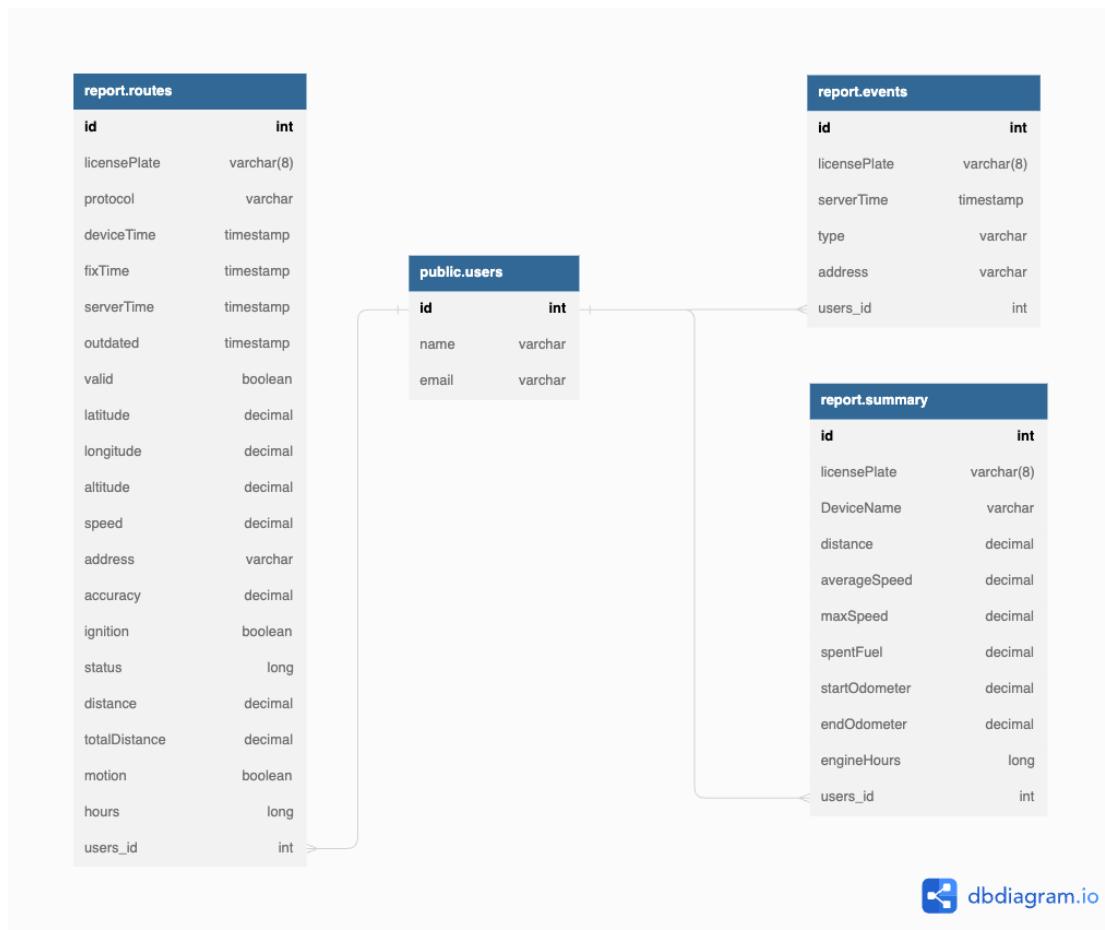


Figura 4 – UML do banco de dados

A figura 4 tem como objetivo mostrar o UML do banco de dados relacionado ao serviço de relatórios, onde o objetivo do mesmo é armazenar o consolidado dos relatórios gerados.

## 5. POC (Prova de Conceito)

Para validar algumas implementações propostas para aplicação foram utilizadas provas de conceito com alguns pontos importantes como SOLID, para um código mais limpo e coeso. E a implementação de Design Patterns que ajudam nesta construção como Singleton, Repository, Injeção de Dependência, AbstractFactory.

### 5.1 Wireframe

Para apresentação da aplicação foi criado um Wireframe navegável com a ferramenta Miro, onde o mesmo está disponível no seguinte link:

[https://miro.com/app/board/uXjVPeGQ-7Y=/?share\\_link\\_id=352327279950](https://miro.com/app/board/uXjVPeGQ-7Y=/?share_link_id=352327279950)

Onde, para navegar basta clicar nas setas (↗) dentro dos componentes da tela.

### 5.2 Código da aplicação

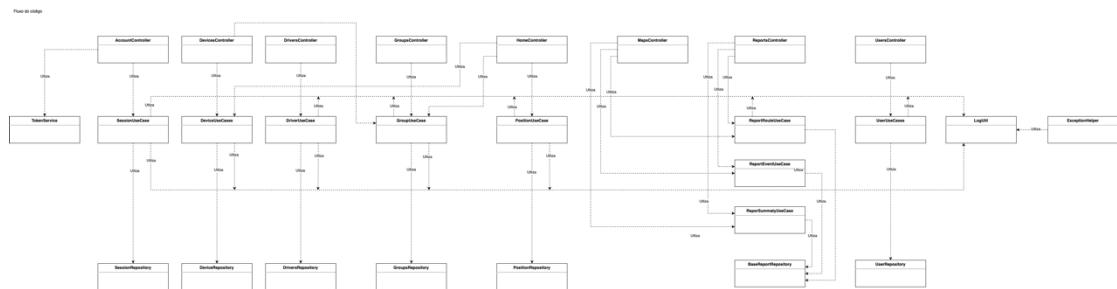


Figura 5 – Código da aplicação

A figura 5 demonstra como estrutura da aplicação foi organizada e como os componentes implementados se relacionam.

Os componentes implementados são divididos em:

Controllers: Responsável por intermediar as requisições enviadas pelas Views com as respostas fornecidas pelas models (dentro do Border)

UseCases: Responsável por concentrar toda a regra de negócio do sistema.

Repositories: Responsável por realizar a consulta e persistência de dados consumindo uma API

Borders: Responsável por ter os objetos comuns entre as camadas, sendo a única camada conhecida por todas as outras. Onde neste modelo está os objetos de manipulação de exception e log.

Link aplicação: <https://aisoftware-tracker-admin.herokuapp.com/>

O código da aplicação está no seguinte repositório no github:

<https://github.com/igoraujo/tcc-pos-graduacao#etapa-02>

**Menu:**

- Código da aplicação

## ***Etapa 3***

### ***6. Avaliação da Arquitetura (ATAM)***

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção visando avaliar se ela atende ao que foi solicitado pelo cliente, segundo o método ATAM.

#### ***6.1. Análise das abordagens arquiteturais***

Atributos de Qualidade	Cenários	Importância	Complexidade
Disponibilidade	Cenário 1: O sistema deve estar disponível 7/7, com enfase no horário comercial.	A	M
Usabilidade	Cenário 2: O sistema deve prover boa usabilidade e compatibilidade browsers distintos.	M	B
	Cenário 3: O sistema deve possuir responsividade em sua interface de usuário.	M	B
Desempenho	Cenário 4: O tempo de resposta das requisições deve ser inferior a 2 segundos em 90% do tempo.	M	M
	Cenário 5: O sistema deve suportar picos de tráfego com escalonamento automático.	A	A
Segurança	Cenário 6: O sistema deve exigir autenticação para navegar no mesmo e autorização para acessar determinados tipos de acessos restritos a diferentes perfis.	A	M

#### ***6.2. Cenários***

Cenário 1 - Disponibilidade: Ao acessar o sistema durante qualquer horário do dia em qualquer dia da semana, deve ser possível logar e utilizar dos recursos do sistema, principalmente gerar relatórios e consultar o mapa. O ideal é que o sistema sempre esteja disponível, podendo ficar no máximo 48hs anuais indisponível.

Cenário 2 - Usabilidade: A navegação pela interface do sistema deve ser simples e intuitiva, com nomes e ícones de fácil percepção para ajudar no entendimento da navegação independente do browser utilizado: Google Chrome, Mozilla Firefox, Microsoft Edge, entre outros.

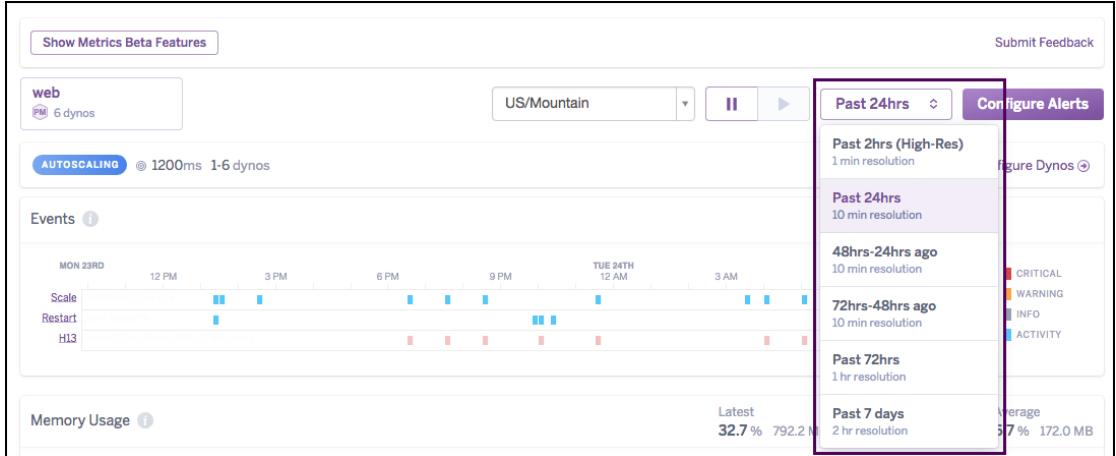
Cenário 3 – Usabilidade: Ao navegar pela interface utilizando um navegador móvel, deve ser possível executar todas as funções do sistema, de maneira responsável.

Cenário 4 – Desempenho: O tempo de resposta da operação de cadastro de iniciativas de distribuição deve ser igual ou inferior ao tempo normal de 2 segundos, para as atividades normais. Podendo ter este tempo extendido ao gerar relatórios com grande volume de dados.

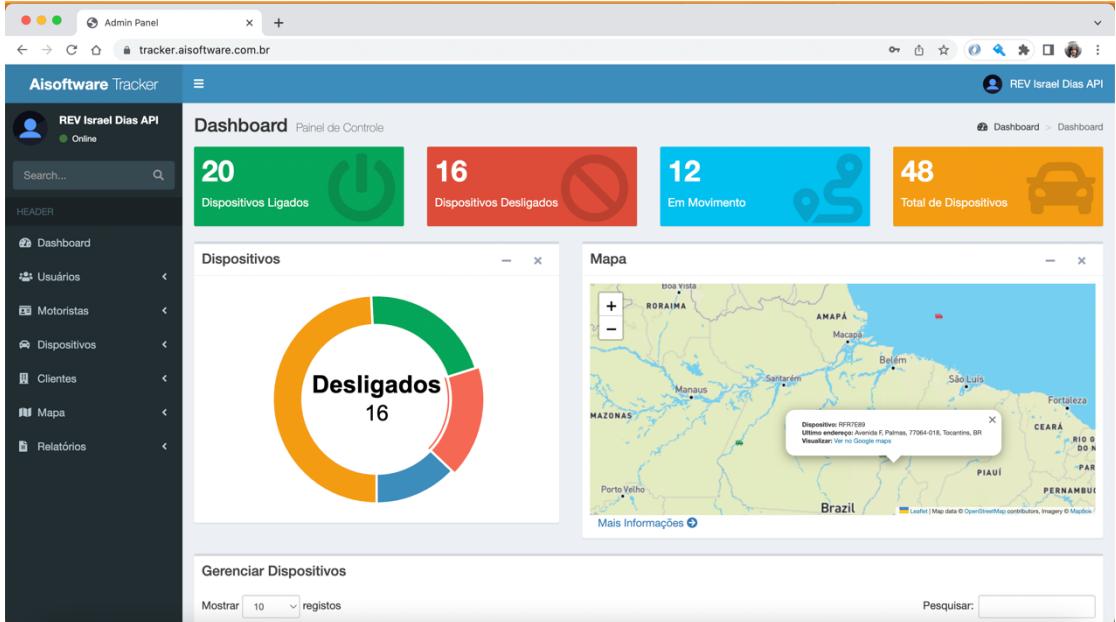
Cenário 5 – Desempenho: O Sistema deve suportar picos de acesso, escalando automaticamente e alocando mais Dynos (Pods) caso necessário.

Cenário 6 – Segurança: O sistema deve exigir autenticação para acessar o mesmo e sua navegação é baseada em autorização de acesso, de acordo com privilégios do perfil.

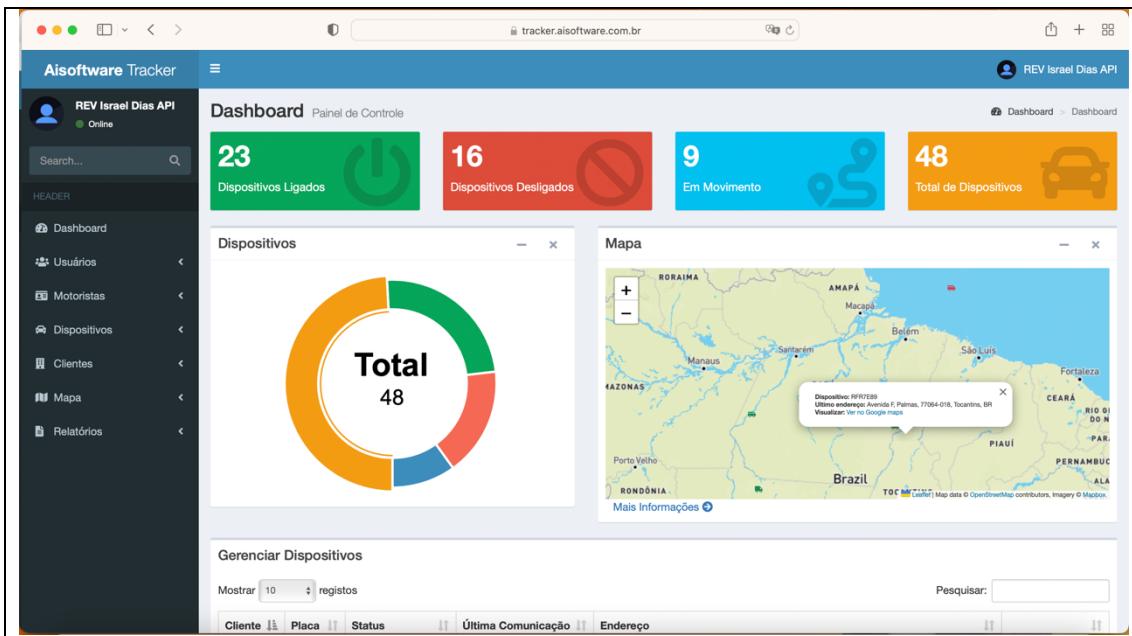
### 6.3. Evidencia da Avaliação

Atributo de Qualidade:	Confiabilidade
Requisito de Qualidade:	O sistema deve estar disponível 7/7 em qualquer hora do dia.
Preocupação:	Garantir que o sistema esteja disponível principalmente durante o horário comercial e, que seus respectivos relatórios estejam sendo gerados conforme o esperado.
Cenários(s):	<p>Cenário 1</p> <p>Ambiente:</p> <p>Sistema em operação normal</p>
Estímulo:	Acesso ao sistema 7/7, porém, principalmente nos dias de semana e em horário comercial.
Mecanismo:	Utilização normal do sistema e monitoramento através do Heroku Metrics e Arquivos de Log.
Medida de Resposta:	Utilização dos alarmes do Heroku Metrics.
	
Consideração sobre a arquitetura:	
Riscos:	Alguma instabilidade ou indisponibilidade de um fornecedor terceiro pode interferir no tempo de resposta ou disponibilidade de recursos da aplicação.
Pontos de Sensibilidade:	Utilização de API de terceiro para alguns recursos.

## Projeto Integrado - Aisoftware Tracker

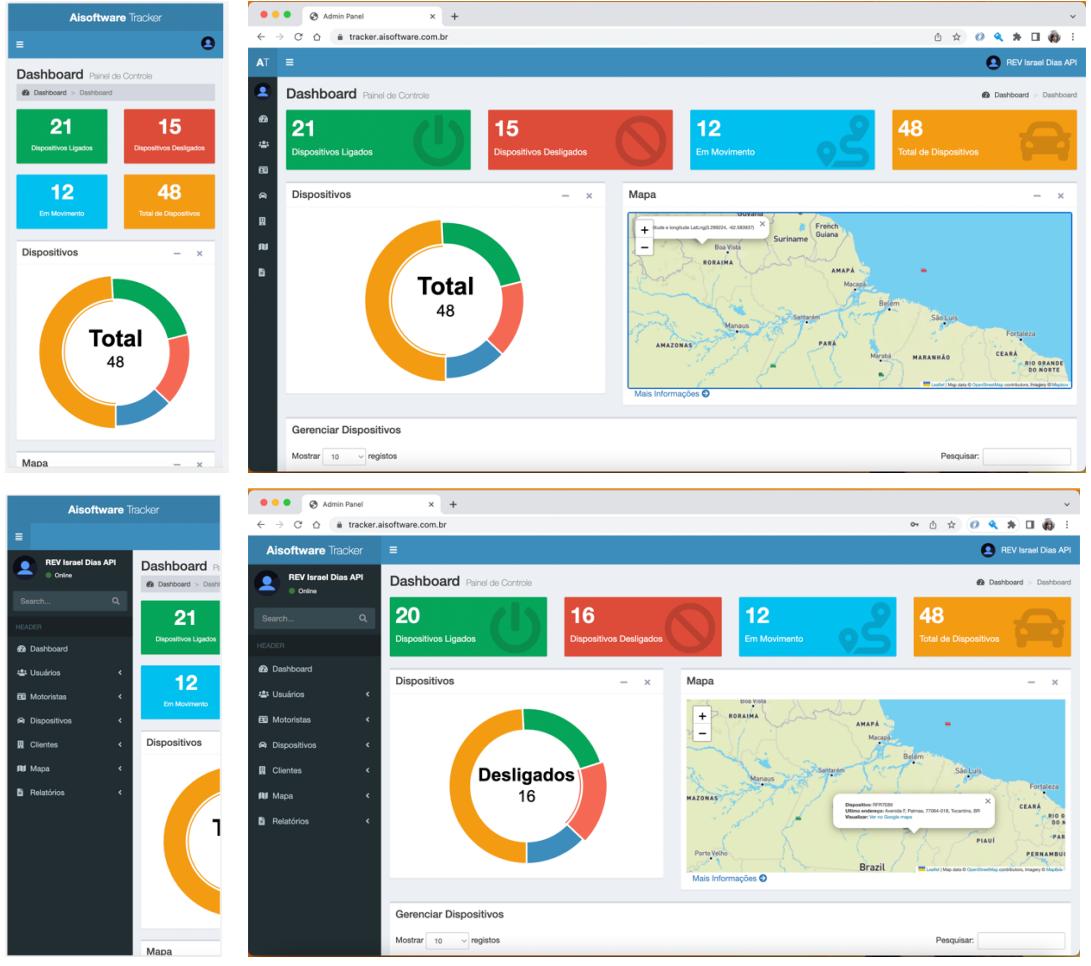
<b>Tradeoff</b>	Custo de internalizar os recursos que a API do terceiro fornece.
<b>Atributo de Qualidade:</b>	Confiabilidade
<b>Requisito de Qualidade:</b>	O sistema web deve ser compatível com os principais browsers do mercado atual.
<b>Preocupação:</b>	O sistema deve prover boa usabilidade e compatibilidade browsers distintos.
<b>Cenários(s):</b>	Cenário 2
<b>Ambiente:</b>	Ambiente:
<b>Sistema em operação normal</b>	Sistema em operação normal
<b>Estímulo:</b>	Acesso a URL principal da aplicação
<b>Mecanismo:</b>	Utilização de ferramentas e recursos de front-end aceita pela maioria dos navegadores, assim como web-kits compatíveis com os mesmos.
<b>Medida de Resposta:</b>	As telas e componentes devem de ser de fácil abstração pelo usuário apresentando coesão entre as telas da aplicação.
<b>Google Chome:</b>	 <p>The screenshot shows the Aisoftware Tracker Admin Panel in Google Chrome. The dashboard displays key statistics: 20 Dispositivos Ligados (green), 16 Dispositivos Desligados (red), 12 Em Movimento (blue), and 48 Total de Dispositivos (orange). Below these, a circular chart indicates 16 devices are off. On the right, a map of Brazil highlights the location of device RFR7690 in Tocantins, BR. The left sidebar lists navigation options like Dashboard, Usuários, Motoristas, Dispositivos, Clients, Mapa, and Relatórios.</p>
<b>Safari:</b>	

## Projeto Integrado - Aisoftware Tracker



### Consideração sobre a arquitetura:

Riscos:	Alguns recursos podem apresentar comportamentos visuais distintos em navegadores muito antigos, por exemplo, o Internet Explorer.
Pontos de Sensibilidade:	Utilização de biblioteca de mapas de terceiros
Tradeoff	N/A

<b>Atributo de Qualidade:</b>	Confiabilidade
<b>Requisito de Qualidade:</b>	O sistema deve possuir responsividade em sua interface de usuário.
<b>Preocupação:</b>	
O sistema deve possuir responsividade para dispositivos desktop e mobile.	
<b>Cenários(s):</b>	
Cenário 3	
Ambiente:	
Sistema em operação normal	
<b>Estímulo:</b>	
Acesso a URL principal da aplicação	
<b>Mecanismo:</b>	
Utilização de bibliotecas, kits e técnicas de responsividade como, o Bootstrap, e estilização manual com CSS, e Javascript.	
<b>Medida de Resposta:</b>	
O layout do sistema deve se adaptar conforme o tipo de dispositivo utilizado para acesso.	
 <p>The figure consists of three side-by-side screenshots of the Aisoftware Tracker dashboard. The top screenshot shows the desktop version with a header, sidebar, and main content area. The middle screenshot shows the tablet version with a simplified layout. The bottom screenshot shows the mobile version with a very compact layout. All versions show the same data: 21 Dispositivos Ligados, 15 Dispositivos Desligados, 12 Em Movimento, 48 Total de Dispositivos, and a circular chart labeled 'Total' with a value of 48. A map of Brazil is also visible in the background of the desktop and tablet versions.</p>	

Celular
 Computador

Mostrando resultados para o URL: <https://tracker.aisoftware.com.br/Account/Login>  
[Executar com URL original](#)

Entender a experiência dos seus usuários       Nenhum dado

Diagnosticar problemas de desempenho

Desempenho

Acessibilidade

Práticas recomendadas

SEO

Celular
 Computador

Mostrando resultados para o URL: <https://tracker.aisoftware.com.br/Account/Login>  
[Executar com URL original](#)

Entender a experiência dos seus usuários       Nenhum dado

Diagnosticar problemas de desempenho

Desempenho

Acessibilidade

Práticas recomendadas

SEO

**Consideração sobre a arquitetura:**

<b>Riscos:</b>	Dispositivos onde a biblioteca do Bootstrap eventualmente não consiga manter a responsividade por falta de mapeamento.
<b>Pontos de Sensibilidade:</b>	A incapacidade de testar em todos os tipos e resoluções de telas existentes no mercado.
<b>Tradeoff</b>	Custo de testes de layouts em uma grande quantidade de dispositivos do mercado.

<b>Atributo de Qualidade:</b>	Confiabilidade
<b>Requisito de Qualidade:</b>	O tempo de resposta das requisições deve ser inferior a 2 segundos em 90% do tempo. O Sistema deve suportar picos de acesso, escalando automaticamente e alocando mais Dynos (Pods) caso necessário
<b>Preocupação:</b>	Garantir que o sistema tenha um tempo de resposta e load menor ou igual a 2 segundos em requisições e navegações comuns. E o sistema deve suportar picos de acessos.
<b>Cenários(s):</b>	Cenário 4 e 5
Ambiente:	Navegações e requisições das páginas
<b>Estímulo:</b>	Loads de telas e requisições HTTP.
<b>Mecanismo:</b>	Utilização de métricas do browser (Chrome) e o Page Speed Insights do Google, e o Heroku Metrics
<p>The screenshot shows the Heroku Metrics dashboard. At the top, there are two tabs: "Celular" and "Computador", with "Computador" being the active tab. Below the tabs is a large green circle with the number "94" inside, labeled "Desempenho". A legend below the circle indicates performance ranges: red for 0-49, orange for 50-89, and green for 90-100. To the right of the circle is a screenshot of a login page for "Aisoftware Tracker" with fields for "Email" and "Password" and a "Login" button. Below the dashboard title, it says "MÉTRICAS" and lists several metrics with their values: First Contentful Paint (0,3 s), Speed Index (1,8 s), Largest Contentful Paint (0,5 s), Time to Interactive (1,6 s), Total Blocking Time (130 ms), and Cumulative Layout Shift (0). There is also a link "Abrir visualização".</p>	

# Projeto Integrado - Aisoftware Tracker

**Dispositivos**

**Total**

12 / 38 requests | 0 B / 10.4 kB transferred | 525 kB / 1.3 MB resources | Finish: 1.94 s | DOMContentLoaded: 1.77 s | Load: 1.83 s

Network

Web Vitals

LONG TASKS

Timings

Main — https://tracker.aisoftware.com.br/

Worker — https://tracker.aisoftware.com.br/cdn-cgi/challenge-platform/h/g/scripts/pica.js

Raster

GPU

Chrome\_ChildIOThread

Response Time ( 95TH PERCENTILE MEDIAN )

04 PM 07 PM 10 PM 01 AM 04 AM 07 AM 10 AM 01 PM

250 ms 200 ms 150 ms 100 ms 50 ms 0

**Medida de Resposta:**

O tempo de resposta e load da tela igual ou inferior a 2 segundos. E alteração na quantidade de Dynos instanciados.

**Consideração sobre a arquitetura:**

<b>Riscos:</b>	Alguma instabilidade ou indisponibilidade de um fornecedor terceiro pode interferir no tempo de resposta ou disponibilidade de recursos da aplicação.
<b>Pontos de Sensibilidade:</b>	Utilização de API de terceiro para alguns recursos.

## Projeto Integrado - Aisoftware Tracker

Tradeoff	Custo de internalizar os recursos que a API do terceiro fornece. Custo de Pods adicionais.
----------	--

<b>Atributo de Qualidade:</b>	Confiabilidade
<b>Requisito de Qualidade:</b>	O sistema deve exigir autenticação para navegar no mesmo e autorização para acessar determinados tipos de acessos restritos a diferentes perfis
<b>Preocupação:</b>	Que usuários sem as devidas permissões realizem atividades dentro do sistema as quais os mesmos não tenham permissão.
<b>Cenários(s):</b>	
Cenário 6	
Ambiente:	
Sistema em operação normal	
<b>Estímulo:</b>	
Acesso a telas e recursos específicos aos quais necessitam de permissão para tipo específico de usuário	
<b>Mecanismo:</b>	
Controle de acesso de usuário por Roles e ClaimIdentity no token JWT.	
<b>Medida de Resposta:</b>	
O Sistema deve permitir ou não acessar determinados recursos de acordo com a autenticação e/ou autorização que o usuário detém.	

## Projeto Integrado - Aisoftware Tracker

### Consideração sobre a arquitetura:

<b>Riscos:</b>	A área de segurança sempre possui um nível de risco crítico, qualquer vulnerabilidade pode comprometer a aplicação.
<b>Pontos de Sensibilidade:</b>	Todas as requisições externas para a aplicação devem possuir certificado HTTPS.
<b>Tradeoff</b>	N/A

#### **6.4. Resultados Obtidos**

Após a avaliação dos critérios de qualidade propostos, foi constatado que o objetivo foi alcançado. Os resultados obtidos para os atributos de disponibilidade, usabilidade, desempenho, rastreabilidade, segurança e acessibilidade estão apresentados em uma tabela a seguir:

Requisitos Não Funcionais	Teste	Homologação
RNF01: A plataforma deve habilitar a autenticação baseado no modelo JWT diretamente no sistema	OK	OK
RNF02: O sistema deve ser responsivo web	OK	OK
RNF03: O sistema deve ser responsivo para todos os tipos de tela (Computadores, Notebooks, Tablets e Smartphones).	OK	OK
RNF04: O sistema deverá utilizar uma API de terceiro para realizar as persistências de informações dos dispositivos rastreadores	OK	OK
RNF05: O sistema deverá utilizar uma API de terceiro para realizar as persistências de informações dos dispositivos rastreadores	OK	OK
RNF06: O sistema deve operar 24x7	OK	OK
RNF07: Apenas usuários com permissão de Admin, tem permissão de criar e editar dados do sistema	OK	OK
RNF08: Usuários operacionais, podem apenas acessar dados de leitura.	OK	OK

## 7. Avaliação Crítica do Resultado

Após avaliar todos os aspectos arquitetônicos deste projeto, dois se destacam como os mais importantes: segurança e facilidade de uso. Levando em conta esses critérios e o orçamento disponível, a arquitetura proposta atende adequadamente aos requisitos estabelecidos.

A decisão de construir a interface gráfica com ASPNET Core 6.0 trouxe uma grande vantagem a nível de desenvolvimento, por não trazer muita complexidade de implementação e toda robustez e segurança da linguagem (C#) e facilidades que o framework traz. Com módulos de independentes por se utilizar o MVC do e outras decisões de segregação de responsabilidades tomadas durante a construção do projeto.

Com isto a decisão trouxe uma facilidade para implementar a segurança de acesso, autenticação e autorização, por ter *libs* simples de se implementar o JWT e bastante material e documentação.

A abordagem de camadas também permite futuramente trocar a camada de apresentação para outra tecnologia, como o React, Angular e outras libs e frameworks de front-end.

O uso de bibliotecas como Bootstrap para desenvolver os layouts também se mostrou assertiva, por ter vários recursos e classes nativos à biblioteca que deixaram o design limpo, simples e de fácil utilização.

A utilização de uma plataforma PaaS como o Heroku, removeu toda complexidade de infraestrutura, como a criação de estrutura de CI/CD, Builds, configuração de métricas, alertas e configurações de distribuição DNS e etc. Trazendo com isto uma maior velocidade no desenvolvimento e publicação da aplicação.

## ***8. Conclusão***

É possível afirmar que este projeto foi bem-sucedido, mostrando a possibilidade de criar uma plataforma de conexão entre os participantes dos programas de doação de cestas básicas.

A realização da prova de conceito foi essencial para entender o efeito que a priorização da segurança teria sobre o desempenho do sistema e determinar quais elementos de um software seguro poderiam ser implementados sem prejudicar a facilidade de uso. Foi possível descobrir que não é possível criar um sistema que satisfaça todas as necessidades ao mesmo tempo, exigindo que algumas sejam priorizadas em detrimento de outras.

Foi possível notar que aumentar a complexidade do sistema, usando um modelo de microserviços, pode ter um efeito negativo na manutenção, especialmente se as fronteiras entre os serviços não estiverem bem definidas.

Um ponto importante da arquitetura proposta foi a possibilidade de reproduzir o ambiente de produção em um computador com recursos limitados. Esta decisão foi tomada com base na escolha de usar ferramentas de código aberto que não estão ligadas a um provedor de serviços de nuvem. O sistema tornou-se mais manutenível, pois foi possível testar ideias e questões durante o processo de criação, sem precisar configurar recursos de infraestrutura.

Uma equipe distribuída poderia executar este projeto de forma que ele fosse criado sob uma licença de código aberto.

Futuras expansões do projeto devem ser adotados APIs com BFF e a internalização de processos os quais são oferecidos por meio de terceiros.

Além disso, podem ampliar a variedade de interfaces visuais disponíveis criando aplicativos nativos para melhorar a eficiência e a facilidade de uso em dispositivos móveis.

## ***Referências***

**LOCALIZA. Evolução da frota de veículos no Brasil.**

Disponível em: <https://frotas.localiza.com/blog/frota-de-veiculo>.

Acesso em: 10 de agosto de 2022.

**IBGE. Frota de Veículos.**

Disponível em:

<https://cidades.ibge.gov.br/brasil/pesquisa/22/28120?indicador=28122&tipo=grafico>.

Acesso em: 10 de agosto de 2022.

**UNISEPE. O transporte rodoviário movimentando o Brasil.**

Disponível em: <https://portal.unisepe.com.br/blog-unisepe/o-transporte-rodoviario-movimentando-o-brasil/>.

Acesso em: 10 de agosto de 2022.

**SUSEP. IVR - Índice de veículos roubados.**

Disponível em:

[http://www2.susep.gov.br/menuestatistica/RankRoubo/resp\\_menu1.asp](http://www2.susep.gov.br/menuestatistica/RankRoubo/resp_menu1.asp).

Acesso em: 10 de agosto de 2022.

**MINISTÉRIO DA INFRAESTRUTURA. Frota de veículos - 2021.**

Disponível em: <https://www.gov.br/infraestrutura/pt-br/assuntos/transito/conteudo-Senatran/frota-de-veiculos-2021>.

Acesso em: 10 de agosto de 2022.