



## **Projeto Final de Programação INF2102**

# Sistema para previsões de séries temporais

Aluno: Claudio Vieira Escudero  
Matrícula: 1912702

Orientador: Prof. Marcus Vinicius Soledade Poggi de Aragão

22 de Junho de 2020

## Conteúdo

Introdução .....	3
Especificação dos Requisitos .....	3
Arquitetura .....	4
Módulo Front-End .....	4
Módulo Back-End .....	5
Ferramenta .....	6
Layout .....	6
Inserindo os dados .....	7
Selecionando o modelo e parâmetros .....	7
Visualizando o resultado .....	8
Exportando resultado .....	9
Instalação da ferramenta .....	9
Como adicionar um novo modelo estatístico .....	10
Editando o Front-End .....	10
Editando o Back-End .....	11
Testes .....	12
Conclusão .....	12
Referências .....	13

## Introdução

Não é novidade para ninguém que o mundo está cada vez mais globalizado e a competição do mercado se apresenta como uma necessidade fundamental para ter visibilidade. A estratégia mais comum para tentar prever os próximos passos dos concorrentes ou até mesmo obter previsões da sua própria empresa para um apoio à decisão é fazer prognóstico com os seus dados, para tentar se antecipar e ter um diferencial.

Atualmente, muitas empresas utilizam planilhas ou métodos ultrapassados para analisar previsões e obter, assim, resultados futuros de negócio, como atividades financeiras, recursos, demandas ou potencialização de campanhas publicitárias.

A proposta deste trabalho é desenvolver e apresentar uma ferramenta simples e barata para calcular previsões utilizando dados de séries históricas de dados, tecnicamente chamado de séries temporais.

Após a implementação da ferramenta, um desenvolvedor, com pouca experiência de estatística, poderá implantar novos modelos de previsões, além da ferramenta já apresentar 2 modelos (auto-ARIMA [14] e Prophet [15]). Desta forma, a ferramenta poderá ser adaptável e reutilizável com pouco custo para qualquer empresa.

## Especificação dos Requisitos

A especificação dos requisitos tem como objetivo satisfazer os usuários, atendendo as suas reais necessidades.

- Ferramenta deve ser web;
- Acessível com os browsers atuais do mercado;
- Ser responsivo (funcionar em desktop ou mobiles);
- Usuário deve inserir seus dados de uma série temporal;
- Usuário deve escolher um modelo estatístico;
- Usuário deve preencher os parâmetros dos modelos pré-cadastrados;
- Ferramenta deve exibir a previsão junto com os dados preenchidos pelo usuário;
- Usuário pode salvar o resultado em arquivo csv ou excel;
- Ferramenta não deve armazenar nenhuma das informações dos dados do usuário.

## Arquitetura

A ferramenta foi criada em 2 (dois) módulos, front-end e back-end. A comunicação entre os dois módulos é feita por requisições REST [1] como protocolo de comunicação e com JSON [2] como formato de mensagem. Este tipo de protocolo é o mais utilizado no mercado.

Na fig 2, o diagrama de sequência [3] demonstra o funcionamento da comunicação entre os módulos:

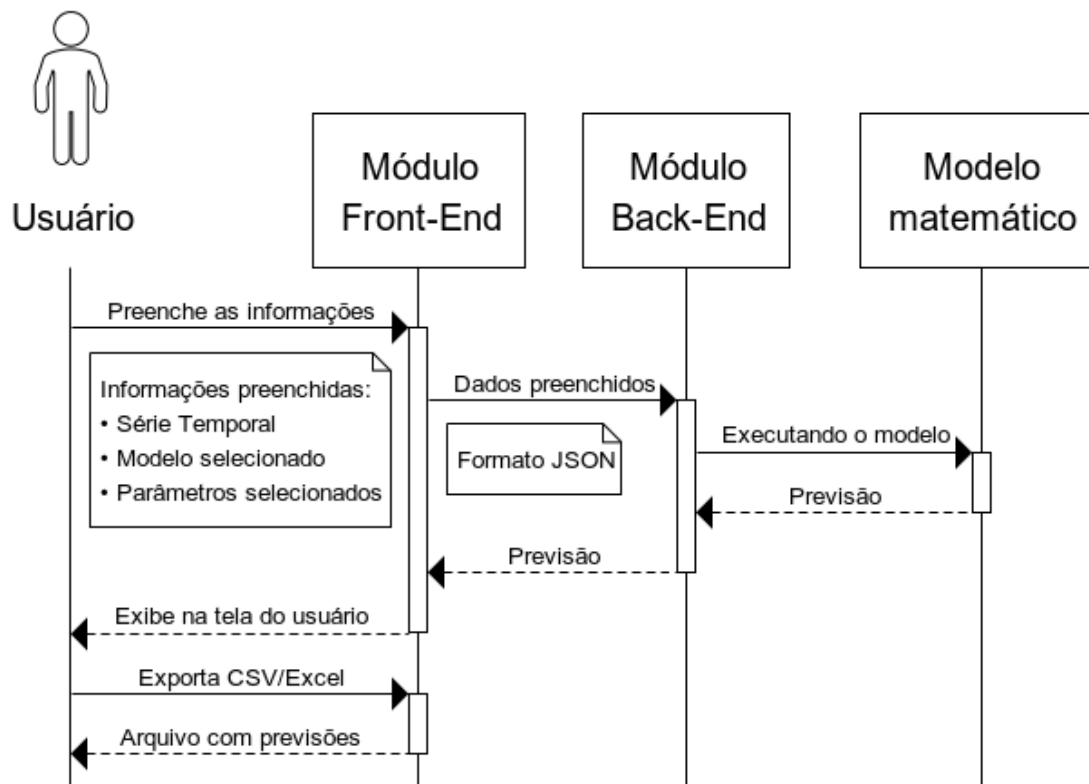


Fig 1. Diagrama de sequência de comunicação entre os módulos

## Módulo Front-End

O módulo Front-End é responsável pela interação direta com o usuário.

Tecnologias e plataformas utilizadas:

- **Container Docker com imagem para o server NGINX [4].**

Container Docker [5] é uma camada de abstração e automação para virtualização de sistema operacional no Linux, isolando por completo a configuração do servidor principal.

NGINX [6] é um servidor leve de HTTP.

- **Linguagem TypeScript com interpretador Node.js**

TypeScript [7] é um superconjunto de JavaScript desenvolvido pela Microsoft que adiciona tipagem e alguns outros recursos a linguagem.

Node.js [8] é um interpretador de javascript e serve como base para o TypeScript.

- **Framework Angular [9]**

Framework desenvolvido em TypeScript [7] para criação de websites complexos e dinâmicos.

- **Bootstrap para o framework visual**

Bootstrap [10] é um framework web para criação de elementos de interface para sites e páginas web usando HTML, CSS e JavaScript.

## Módulo Back-End

O módulo Back-End é responsável por receber as informações preenchidas na interface do usuário e, também, por executar de modelos estatísticos para gerar previsões.

Tecnologias e plataformas utilizadas:

- **Container Docker com imagem para o server NGINX [4]**

Container Docker [5] é uma camada de abstração e automação para virtualização de sistema operacional no Linux, isolando por completo a configuração do servidor principal.

NGINX [6] é um servidor leve de HTTP.

- **Linguagem Python com ambiente Anaconda**

Anaconda [11] é um ambiente para linguagem de programação Python muito utilizada na comunidade de computação científica com um gerenciador de implementações de pacotes. Este ambiente já inclui os principais pacotes para ciência de dados.

- **Framework Web Flask [12]**

Um micro-framework web multiplataforma escrito em Python simplifica o desenvolvimento web ou de microserviços.

- **Pandas**

Pandas [13] é biblioteca em Python que fornece estruturas de dados de alto desempenho, fáceis para carregar dados de diversos tipos de DataFrames, além de poder manipular essas estruturas de forma otimizada.

- **Pmdarima**

Pmdarima [14] é biblioteca em Python de estatística projetada para análise de previsões em séries temporais.

- **Prophet**

Prophet [15] é biblioteca em Python e R de estatística projetada para análise de previsões em séries temporais desenvolvida pelo Facebook.

## Ferramenta

Neste capítulo será apresentada a utilização da ferramenta e seus resultados.

### Layout

Assim que o usuário acessar a ferramenta pelo navegador, visualizará três blocos, conforme a Fig 2:

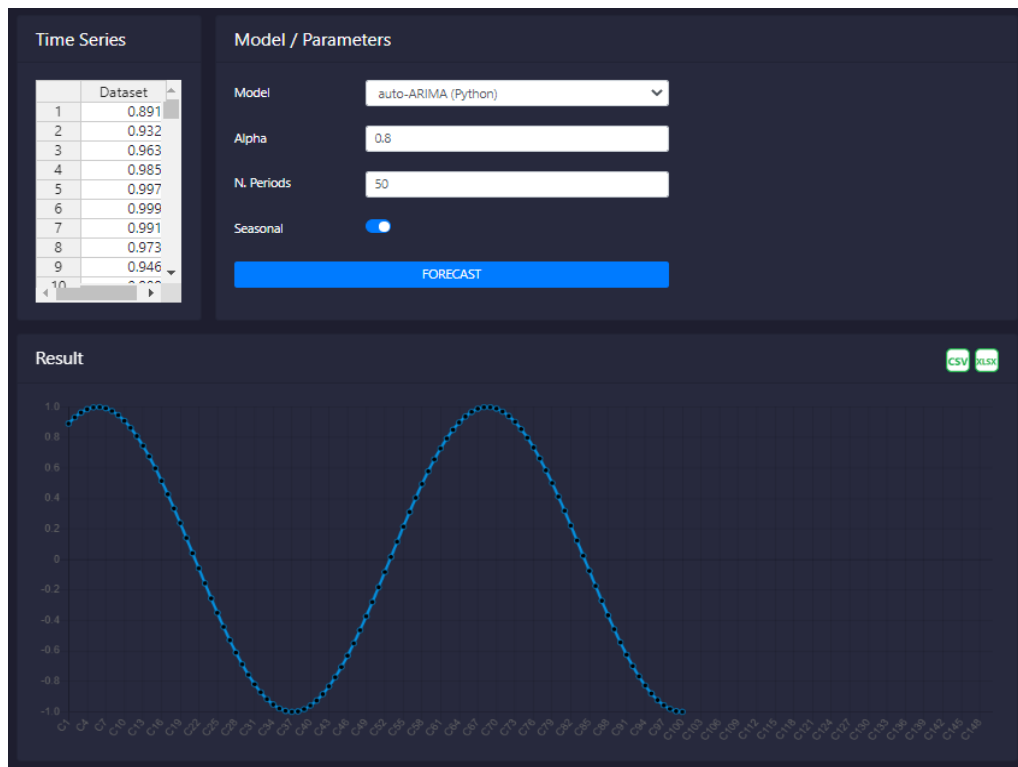


Fig 2. Tela principal da ferramenta

Cada bloco é encarregado pelas seguintes tarefas:

- **Time Series:** Responsável por coletar os dados (série temporal) de entrada preenchidos pelo usuário.
- **Model / Parameters:** Responsável por parametrização do modelo estatístico, que será executado.
- **Result:** Responsável por exibir o resultado para o usuário.

## Inserindo os dados

O bloco "Time Series" contém uma tabela que é editável pelo usuário. Nela, os dados inseridos irão alimentar a tabela com as informações necessárias para que a ferramenta possa fazer uma previsão, como por exemplo: Dados de demanda, estoque, sensores de equipamentos, etc.

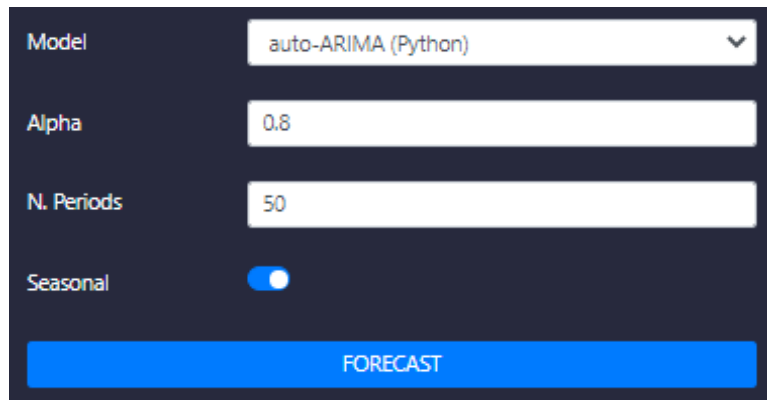
É importante que o usuário possa inserir a maior quantidade que puder de períodos de séries, assim a ferramenta terá mais exemplos e poderá detectar padrões com mais acurácia. Para facilitar a inserção dos dados, o usuário pode copiar as linhas do excel e colar na tabela que se encontra no bloco "Time Series".

Ao atualizar a tabela, o usuário pode verificar o comportamento da série no bloco "Result", porque a ferramenta tem uma atualização automática para visualização da série que será enviada.

## Selecionando o modelo e parâmetros

O bloco "Model / Parameters" tem o objetivo de permitir que o usuário informe o modelo que deseja utilizar na previsão, utilizando alguns modelos já pré-cadastrados (o capítulo "Como adicionar um novo modelo estatístico" explica como adicionar novos modelos). Com isso, assim que o modelo for selecionado, os parâmetros do modelo deverão ser preenchidos de acordo com a Fig 3, segue abaixo os parâmetros de exemplo:

- **Alpha:** Margem para o intervalo de confiança para a previsão.
- **N. Periods:** Quantidade de períodos para a previsão.
- **Seasonal:** Informando se a série é sazonal.

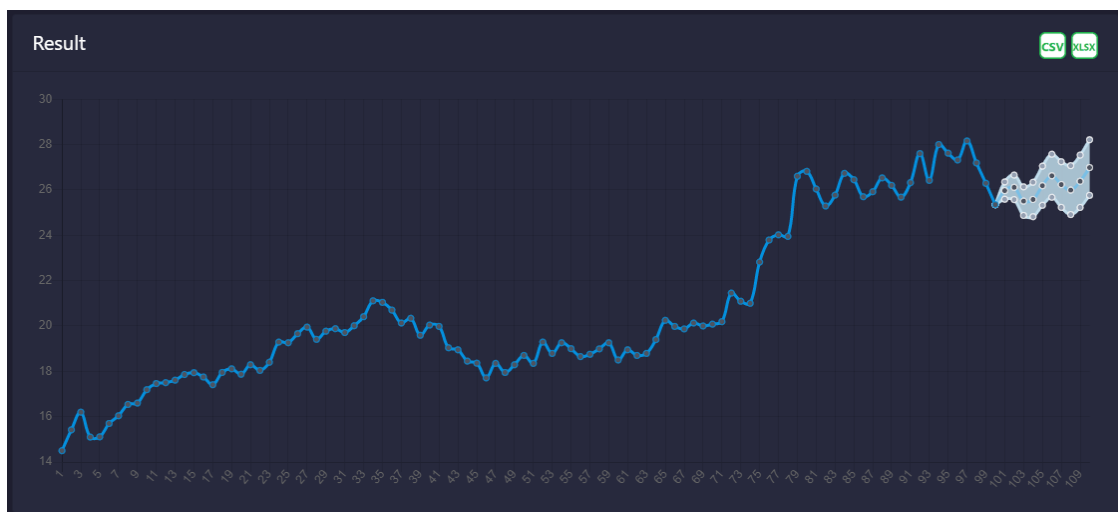


The image shows a configuration form for a forecasting model. It has a dark background with white text and input fields. The fields are: 'Model' with a dropdown menu showing 'auto-ARIMA (Python)'; 'Alpha' with a text input field containing '0.8'; 'N. Periods' with a text input field containing '50'; and 'Seasonal' with a toggle switch that is currently turned on. At the bottom of the form is a large blue button labeled 'FORECAST'.

*Fig 3. Parâmetros padrões*

Assim que o usuário informar os parâmetros, o mesmo deverá apertar o botão “FORECAST” para realizar a previsão no bloco “Result”.

## Visualizando o resultado



*Fig 4. Série de entrada e previsão*

O bloco "Result" contém duas informações: os dados históricos preenchidos pelo usuário e a série prevista, sendo representados respectivamente pela cor azul escuro e azul claro.

Os dados históricos preenchidos pelo usuário possuem informações acerca de demandas, estoque, sensores de equipamentos, etc. Esses dados foram preenchidos na tabela do bloco "Time Series".



A série prevista é o que o modelo estatístico gerou, acompanhado da seleção do bloco "Model / Parameters". Observa-se na Fig 4, que a previsão contém 3 linhas (3 valores para eixo y), que são os valores de intervalo de confiança superior, inferior e o valor sugerido de previsão.

Intervalo de confiança é um grau de certeza, quando mais próximo aos extremos, menor a probabilidade de acurácia. Este intervalo de confiança é configurável com o parâmetro "Alpha" do bloco "Model / Parameters".

## Exportando resultado

A ferramenta possui suporte de exportar em até 2 (dois) formatos de arquivos, csv e xlsx (Excel). Esta opção é habilitada e visível assim que o usuário faz a previsão. Os botões são em formato de ícones ao lado direito do bloco, como apresentado na Fig 5.



*Fig 5. Botões para exportar*

## Instalação da ferramenta

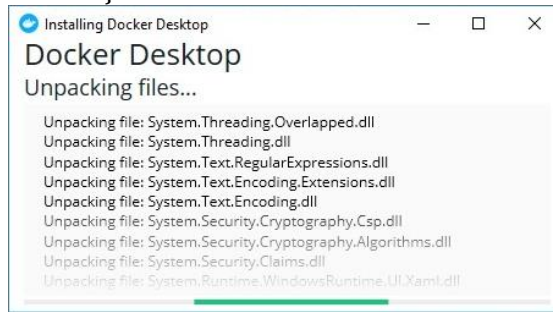
Neste capítulo será apresentada a instalação da ferramenta no windows.

Como foi descrito no capítulo de arquitetura, a ferramenta foi construída em containers do Docker com Docker-Compose para a orquestração entre os módulos. Com isso, a máquina na qual será instalada a ferramenta necessita apenas do Docker instalado com o "Hyper-V Manager" habilitado.

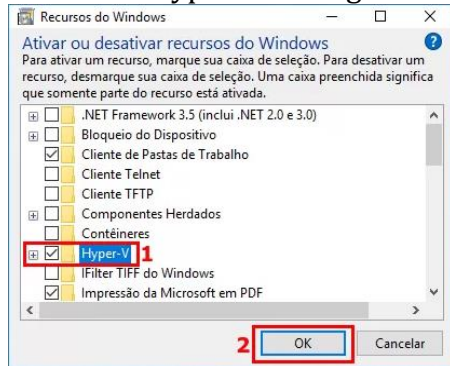
### Passos de instalação

1. Download do Docker  
<https://hub.docker.com/editions/community/docker-ce-desktop-windows/>

## 2. Instalação do Docker



## 3. Habilitar “Hyper-V Manager” em Recursos do windows



4. No diretório dos arquivos da ferramenta, abra o terminal e execute o comando abaixo para rodar o Docker Compose.

```
docker-compose.exe -f docker-compose-local.yml up --build
```

5. Finalizando, abra o endereço abaixo pelo navegador:

<http://localhost:4200/>

6. Pronto! O capítulo Ferramenta mostra a utilização.

## Como adicionar um novo modelo estatístico

Este capítulo mostrará como adicionar o modelo estatístico ARIMA, além dos 2 (dois) modelos pré-cadastrados: auto-ARIMA e Prophet.

Para adicionar um novo modelo, é obrigatório editar os 2 (dois) módulos, o front-end para o usuário poder selecionar o novo modelo e o back-end para poder realizar as previsões.

## Editando o Front-End

O arquivo “/app/forecast-form/forecast-form.component.ts” tem a principal funcionalidade de exibir as opções de modelos que o usuário pode selecionar, para

isso é necessário adicionar uma nova opção na variável “models”, como no exemplo da Fig 6.

```
30 public models: any[] = [  
31   {value: 'autoarima_python', viewValue: 'auto-ARIMA'},  
32   {value: 'prophet_python', viewValue: 'Prophet'},  
33   {value: 'arima_python', viewValue: 'ARIMA'}  
34   ];
```

*Fig 6. Arquivo forecast-form.component.ts*

Para adicionar uma nova opção é necessário adicionar o nome (viewValue) do modelo e um código (value) que será utilizado.

Feito isso, a interface já mostrará essa opção para o usuário. Porém, é fundamental adicionar o novo modelo no back-end.

## Editando o Back-End

O arquivo “/src/app.py” tem como principal funcionalidade receber a requisição do módulo front-end e executá-lo. Abaixo os 2 (dois) passos para edição:

**Primeiro passo:** Criar a função de execução do modelo

Seguindo o exemplo da imagem Fig 7, na linha 91 uma função foi criada com o nome “execute\_arima\_python”. Essa função obrigatoriamente possui 2 (dois) tipos de retornos: array de previsão (forecast) e array com intervalo de confiança.

```
91 def execute_arima_python(data, params):  
92     alpha = params['alpha']  
93     n_periods = params['n_periods']  
94     n_periods = n_periods if n_periods <= 1000 else 1000  
95  
96     model = sm.tsa.ARIMA(data, order=(1, 1, 1))  
97     forecast, _, conf_int = model.fit().forecast(steps=n_periods, alpha=alpha)  
98  
99     return forecast, conf_int
```

*Fig 7. app.py*

**Segundo passo:** Chamada da função do primeiro passo

Seguindo o exemplo da imagem Fig 8, nas linhas 44 e 45, uma nova opção do modelo (arima) foi adicionada para a chamada da função “execute\_arima\_python” que foi criada no primeiro passo.

```
40 if content['model'] == 'autoarima_python':  
41     forecast, conf_int = execute_autoarima_python(data, params)  
42 elif content['model'] == 'prophet_python':  
43     forecast, conf_int = execute_prophet_python(data, params)  
44 elif content['model'] == 'arima_python':  
45     forecast, conf_int = execute_arima_python(data, params)
```

*Fig 8. app.py*

Após essas alterações, o usuário já pode realizar os testes como descrito no capítulo “Ferramenta”.

## Testes

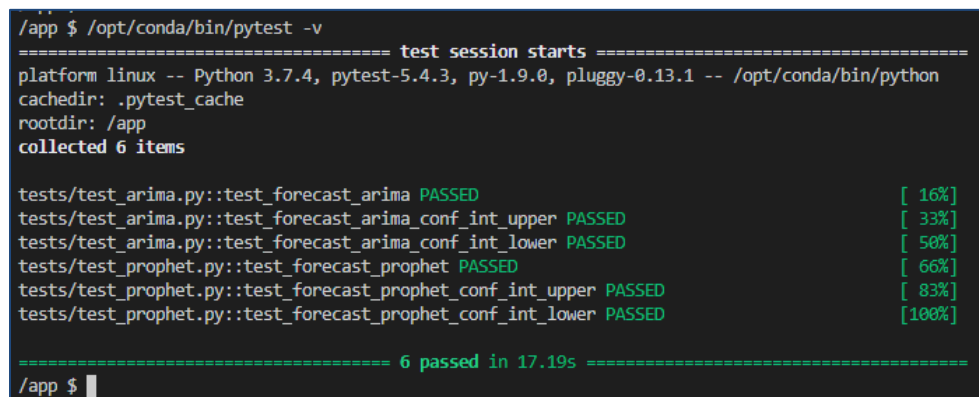
Testes unitários foram implementados em seis principais funcionalidades de previsões utilizando o *framework* Pytest [17].

Funcionalidades:

- Checar a previsão do auto-ARIMA
- Checar o intervalo de confiança superior do auto-ARIMA
- Checar o intervalo de confiança inferior do auto-ARIMA
- Checar a previsão do Prophet
- Checar o intervalo de confiança superior do Prophet
- Checar o intervalo de confiança inferior do Prophet

A Fig 9 mostra um exemplo da execução do script de teste unitário utilizando o comando:

```
$ /opt/conda/bin/pytest -v
```



```
/app $ /opt/conda/bin/pytest -v
===== test session starts =====
platform linux -- Python 3.7.4, pytest-5.4.3, py-1.9.0, pluggy-0.13.1 -- /opt/conda/bin/python
cachedir: .pytest_cache
rootdir: /app
collected 6 items

tests/test_arma.py::test_forecast_arma PASSED [ 16%]
tests/test_arma.py::test_forecast_arma_conf_int_upper PASSED [ 33%]
tests/test_arma.py::test_forecast_arma_conf_int_lower PASSED [ 50%]
tests/test_prophet.py::test_forecast_prophet PASSED [ 66%]
tests/test_prophet.py::test_forecast_prophet_conf_int_upper PASSED [ 83%]
tests/test_prophet.py::test_forecast_prophet_conf_int_lower PASSED [100%]

===== 6 passed in 17.19s =====
/app $
```

Fig 9. Teste unitário

## Conclusão

Este trabalho apresenta uma ferramenta simples e barata para calcular previsões utilizando dados de séries históricas de informações, tecnicamente conhecidas como séries temporais. Além disso, a ferramenta pode ser adaptada com modelos estatísticos personalizados para os usuários.

Todos os códigos estão disponíveis no github [16] para possíveis colaboradores com a finalidade de contribuição para o aperfeiçoamento da ferramenta.

## Referências

1. <https://restfulapi.net/http-methods/>
2. <https://www.json.org/>
3. <https://online.visual-paradigm.com/diagrams/tutorials/sequence-diagram-tutorial/>
4. [https://hub.docker.com/\\_/nginx](https://hub.docker.com/_/nginx)
5. <https://www.docker.com/>
6. <https://www.nginx.com/>
7. <https://www.typescriptlang.org/>
8. <https://nodejs.org/en/>
9. <https://angular.io/>
10. <https://getbootstrap.com/>
11. <https://anaconda.org/>
12. <https://flask.palletsprojects.com/en/1.1.x/>
13. <https://pandas.pydata.org/>
14. <https://pypi.org/project/pmdarima/>
15. <https://facebook.github.io/prophet/>
16. <https://github.com/escudero/AppForecast>
17. <https://flask.palletsprojects.com/en/1.1.x/testing/>