

**Oswaldo Borges Peres**  
**Bruno Santos Cezário**  
**André Luís Azevedo Guedes**

**Documentação técnica &  
Manual do usuário do IAPCI V3.0**

Oswaldo Borges Peres  
Bruno Santos Cezário  
André Luís Azevedo Guedes

Documentação técnica &  
Manual do usuário do IAPCI V3.0

1ª Edição



Rio de Janeiro – RJ  
2025

Copyright © 2025 by Epitaya Propriedade Intelectual Editora Ltda. Todos os direitos reservados.

Direitos de Edição Reservados à Epitaya Propriedade Intelectual Editora Ltda.

Nenhuma parte desta obra poderá ser utilizada indevidamente, sem estar de acordo com a Lei nº 9.610/98.

Todo o conteúdo, assim como as possíveis correções necessárias dos artigos é de responsabilidade de seus autores.

Dados Internacionais de Catalogação na Publicação (CIP)  
(eDOC BRASIL, Belo Horizonte/MG)

---

P437d

Peres, Oswaldo Borges.

Documentação técnica & manual do usuário do IAPCI V3.0 [livro eletrônico]  
/ Oswaldo Borges Peres, Bruno Santos Cezário, André Luís Azevedo Guedes.  
– Rio de Janeiro, RJ: Epitaya, 2025.

Formato: PDF

Requisitos de sistema: Adobe Acrobat Reader

Modo de acesso: World Wide Web

ISBN 978-85-94431-85-1

1. Cidades inteligentes. 2. Cidades e vilas – Inovações tecnológicas. 3. Desenvolvimento urbano sustentável. I. Cezário, Bruno Santos. II. Guedes, André Luís Azevedo. III. Título.

CDD 307.76

---

Elaborado por Maurício Amormino Júnior – CRB6/2422

Epitaya Propriedade Intelectual Editora Ltda  
Rio de Janeiro / RJ  
contato@epitaya.com.br  
<http://www.epitaya.com.br>



# 1

# MANUAL DO USUÁRIO

# **INSTRUÇÕES PARA INSTALAÇÃO DO APLICATIVO IAPCI NO ANDROID (VIA APK)**

1. Faça o download do aplicativo

- Acesse o link abaixo e baixe o arquivo de instalação (app-debug.apk):

[https://drive.google.com/file/d/1zHXY\\_HefivNbX7zys54jm9zT8P9z2osR/view?usp=sharing](https://drive.google.com/file/d/1zHXY_HefivNbX7zys54jm9zT8P9z2osR/view?usp=sharing)

- Você também pode receber o arquivo por e-mail, WhatsApp ou outros meios autorizados.

2. Inicie a instalação

- Após concluir o download, o Android geralmente exibe uma mensagem na parte inferior da tela perguntando se deseja Instalar.

- Toque em Instalar para iniciar o processo.

3. Permita a instalação de fontes desconhecidas (se necessário)

- Caso seja a primeira vez que você instala um app fora da Play Store, o Android pode exibir um aviso de segurança.

- Siga as instruções na tela:

o Toque em Configurações.

o Ative a opção Permitir desta fonte.

o Volte para a tela de instalação e continue.

4. Aguarde a instalação

- O processo leva apenas alguns segundos.

• Ao final, toque em Abrir para iniciar o aplicativo ou em Concluído para fechar a tela.

5. Acesse o aplicativo

- O ícone do IAPCI estará disponível na tela inicial ou na lista de aplicativos do seu celular.

• Toque no ícone para abrir e siga as instruções para login.

6. Vídeo demonstrativo do processo de instalação

- Veja o passo a passo em vídeo: <https://youtube.com/shorts/S9k6VPBuCc>

# DOCUMENTAÇÃO TÉCNICA - TELA "HOME USUÁRIO"

## Acesso do usuário:

- Veja o passo a passo em vídeo:

<https://youtube.com/shorts/mWf0nKAhGWU>

- Dados para login:

Login: **testeusuario@usuario.com**

Senha: **123456**

## Objetivo da Tela

A tela "Home" permite que o usuário visualize, filtre e analise os dados das suas ocorrências cadastradas no sistema. Essa visualização inclui um mapa interativo, um gráfico de barras com estatísticas por categoria, além de uma tabela com porcentagens de ocorrências por tema.

## Componentes Visuais

1. Cabeçalho Título: Home

Botão de menu lateral no canto direito.

2. Informações do Usuário

Exibição do nome do usuário e foto de perfil. Dados são carregados com base no usuário autenticado via FirebaseAuthentication.

3. Filtros de Pesquisa Cidade

(selectedCity), Município (bairro)

(selectedNeighborhood) Mês

(selectedMonth), Ano (selectedYear)

Os filtros são aplicados em tempo real nos dados exibidos no mapa, gráfico e tabela.



## 4. Mapa (Google Maps)

Exibe marcadores das ocorrências filtradas. Cada marcador representa uma ocorrência e pode ser clicado.

## 5. Gráfico de Barras (Chart.js)

Mostra o número de ocorrências por categoria (como Segurança Pública, Saúde, Mobilidade etc.). O gráfico é atualizado dinamicamente conforme os filtros são alterados.

## 6. Tabela de Porcentagens

Lista todas as categorias com: Quantidade de ocorrências.

Porcentagem com barra de progresso visual.



## Google Maps API

- Mapa centralizado em latitude/longitude padrão.
- Marcadores criados com base nas coordenadas lat/Ing das ocorrências.

## Funcionalidades Técnicas

- Autenticação via Firebase Auth.
- Consulta em tempo real do banco Firebase Realtime Database na coleção /reports.

## Lógica de Filtragem

```
applyFilters() {
  this.filteredOccurrences =
  this.occurrences.filter(...)
}
```

Aplica filtros de cidade, bairro, mês e ano

- Chamado após cada alteração de filtro.
- Atualiza mapa, gráfico e tabela.

## Categorização de Dados

- As ocorrências são agrupadas por occurrences, que indica a categoria escolhida no momento da submissão do relatório.

## Dinâmica de Anos Disponíveis

- Os anos são extraídos dinamicamente com base nas datas das ocorrências:

```
this.years = Array.from(new
Set(data.map(occurrence => new
Date(occurrence.date).getFullYear(
)))
));
```

## Dependências

- Chart.js para renderização dos gráficos.
- @angular/fire para comunicação com Firebase.
- rxjs para manipulação de streams e observables.
- Google Maps API (via <script> no index.html do projeto).

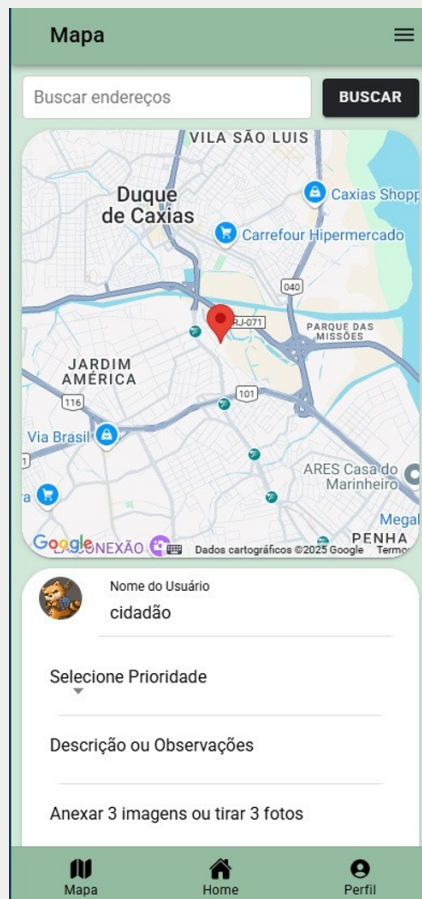
# DOCUMENTAÇÃO TÉCNICA - TELA "TELA DE OCORRÊNCIAS"

## Objetivo

A tela mapa do aplicativo tem como finalidade permitir que o usuário registre uma ocorrência com base em sua localização atual, insira uma descrição, selecione o tipo de ocorrência, tire ou envie fotos, e submeta tudo para o Firebase.

## Estrutura do Componente

- AngularFireDatabase – Para interação com o Firebase RealtimeDatabase.
- FirebaseService, CameraService, UserService, GmapService – Serviços personalizados para lógica de negócio.
- AlertController – Para exibição de alertas no Ionic.
- google.maps – API do Google Maps.



## Mapeamento com Google Maps

loadMap(): carrega o mapa na localização atual do usuário usando navigator.geolocation. Cria um marcador que pode ser arrastado para ajustar a posição. Usa Geocoder para converter coordenadas em endereço (rua, bairro, cidade, estado, CEP). Permite pesquisar endereço via input com ID search-box.

## Fotos

- O usuário pode: Tirar fotos com a câmera (`takePhotoFromCamera`) – até 3 imagens. Selecionar fotos da galeria (`handlePhotoUpload`).
- As fotos são armazenadas em um array `photos[]`, e os URLs são salvos após o upload para o Firebase (`uploadPhotos`).

## Formulário de Ocorrência

- Campos obrigatórios: Nome do usuário (recuperado via `UserService`). Descrição e tipo de ocorrência. Mínimo de 3 fotos. Localização com latitude e longitude.
- Validação e envio: Verifica preenchimento de campos. Exibe alerta se algo estiver faltando. Envia os dados para o Firebase, onde é gerado um `reportId` (rid) único.

## Outros Recursos

- `clearFormFields()`: Limpa todos os campos do formulário após o envio bem-sucedido.
- `showAlert()`: Utilitário para mostrar mensagens de erro ou sucesso ao usuário.
- `fetchAddressDetails(postalCode)`: Recupera cidade e bairro com base no CEP consultado no Firebase

# DOCUMENTAÇÃO TÉCNICA - TELA "TELA DE PERFIL"

## Objetivo

Exibir as informações do usuário logado no sistema, como nome, e-mail, perfil, e endereço (caso disponível), além de oferecer a funcionalidade de logout. A tela também identifica o tipo de perfil do usuário: administrador, gestor, ou cidadão.

## Funcionalidades

### Obtenção de Perfil

- O método `checkUserRole()` é executado no `ngOnInit()`.
- Ele verifica, com base no UID, se o usuário logado é administrador, gestor ou cidadão, definindo os booleans: `isAdmin`, `isGestor` ou `isCidadao`.



### Carregamento de Dados

- O método `getUserProfile()` assina o estado de autenticação e, se houver um usuário logado, recupera:
- `userName`, `email` e o `profile`.  
Esses dados de endereço, como: `cep`, `uf`, `município`, `rua`, `bairro`, `complemento`, etc.

# DOCUMENTAÇÃO TÉCNICA - TELA "HOME GESTOR"

## Acesso do Gestor:

- Veja o passo a passo em vídeo:  
<https://youtube.com/shorts/WUeqtH2z4zI>

- Dados para login:

Login: **testegestor@gestor.com**

Senha: **123456**

## Objetivo

A tela Home Gestor é responsável por exibir um painel analítico com relatórios de ocorrências cadastradas por usuários. A tela permite a visualização de um mapa interativo, gráfico do tipo radar, tabela de distribuição percentual por categorias e filtros para segmentação de dados por cidade, bairro, mês, ano e usuário.

## Funcionalidades

### 1. Carregamento de Ocorrências

- Fonte: Firebase Realtime Database (/reports)
- Após o carregamento:

Atualiza listas de cidades, bairros, anos e nomes de usuários.

Aplica filtros automaticamente.



### 2. Filtragem de Ocorrências

- Filtros disponíveis:

Cidade, Bairro, Mês, Ano

### 3. Gráfico Radar (Chart.js)

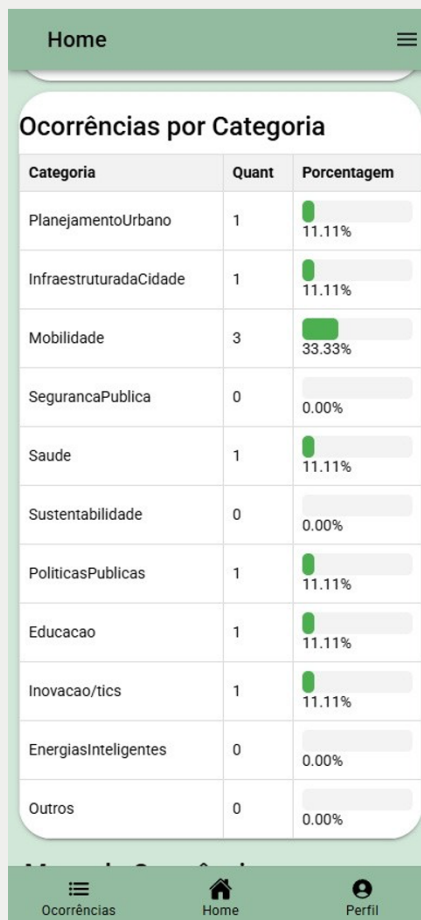
- Exibe a distribuição de ocorrências por categoria.

#### 4. Tabela Percentual com Barra de Progresso

- Lista cada categoria.
- Mostra contagem e percentual relativo ao total de ocorrências filtradas.
- Barra de progresso estilizada com CSS inline.

#### 5. Google Maps

- Exibe as ocorrências filtradas no mapa com marcadores.
- Local padrão: região de RJ.





Estrutura dos Componentes

| Propriedade         | Função    | Descrição                                                              |
|---------------------|-----------|------------------------------------------------------------------------|
| occurrences         | any[]     | Lista original de todas as ocorrências obtidas do Firebase.            |
| filteredOccurrences | any[]     | Lista filtrada com base nos critérios de mês, ano e status.            |
| monthFilter         | string    | Filtro de mês (1 a 12 ou vazio para todos).                            |
| selectedYears       | string    | Filtro de ano (ex: "2023", "2024" ou "all").                           |
| statusFilter        | string    | Filtro de status (fechada, equipe_enviada, nao_encaminhada, ou todos). |
| year                | number [] | Lista de anos únicos extraídos das datas das ocorrências.              |

# DOCUMENTAÇÃO TÉCNICA - TELA "LISTA DE OCORRÊNCIAS"

## Objetivo

A tela Lista de Ocorrências permite ao Gestor visualizar e filtrar uma lista de solicitações cadastradas no sistema. Os filtros disponíveis incluem mês, ano e status. Ao clicar em uma ocorrência, é exibido um modal com os detalhes da solicitação.



| Propriedade         | Função   | Descrição                                                              |
|---------------------|----------|------------------------------------------------------------------------|
| occurrences         | any[]    | Lista original de todas as ocorrências obtidas do Firebase.            |
| filteredOccurrences | any[]    | Lista filtrada com base nos critérios de mês, ano e status.            |
| monthFilter         | string   | Filtro de mês (1 a 12 ou vazio para todos).                            |
| selectedYears       | string   | Filtro de ano (ex: "2023", "2024" ou "all").                           |
| statusFilter        | string   | Filtro de status (fechada, equipe_enviada, nao_encaminhada, ou todos). |
| year                | number[] | Lista de anos únicos extraídos das datas das ocorrências.              |

Detalhes da Ocorrência

FECHAR

Tipo de Ocorrência

Mobilidade



Usuário: cidadão

Descrição

Estação sem trem

 Ver Imagem

 Ver no mapa

status:

nao\_encaminhado

Data:

16/10/2024

Rua:

Rua Onze Unidos

Bairro:

Vigário Geral

Cidade:

Rio de Janeiro

Estado:

Rio de Janeiro

Cep:

21010-610

FECHAR OCORRÊNCIA

ENVIAR EQUIPE PARA O LOCAL

O componente

OccurrenceDetailComponent é responsável por exibir em formato de modal os detalhes de uma ocorrência reportada por um usuário, permitindo visualizar:

- Informações do usuário e da ocorrência
- Localização no mapa
- Imagens associadas à ocorrência
- Ações administrativas como envio de equipe e fechamento da ocorrência

Inputs

| Propriedade                                       | Função   | Descrição                                              |
|---------------------------------------------------|----------|--------------------------------------------------------|
| userName                                          | string   | Nome do usuário que registrou a ocorrência             |
| imageUrl                                          | string   | URL da imagem/avatar do usuário                        |
| occurrences                                       | string   | Tipo de ocorrência reportada                           |
| description                                       | string   | Descrição da ocorrência                                |
| lat / lng                                         | number   | Coordenadas geográficas                                |
| photoUrls                                         | string[] | Lista de URLs de fotos relacionadas                    |
| streetName, neighborhood, city, state, postalCode | string   | Endereço completo da ocorrência                        |
| date                                              | string   | Data da ocorrência                                     |
| status                                            | string   | Status atual da ocorrência (ex: "pendente", "fechada") |
| rid                                               | string   | ID único da ocorrência (report ID)                     |

## Lógica do Componente

### Métodos principais:

- `ngOnInit()`: Inicializa o componente e realiza verificações nos inputs.
- `checkInputs()`: Loga todos os dados recebidos via `@Input` para debug.
- `dismiss()`: Fecha o modal atual.
- `showMap()`: Abre um modal com o mapa usando `MapModalComponent`.
- `openImageModal(index:number)`: Abre um modal para visualização de imagens com `ImageModalComponent`.
- `confirmSendTeam()`: Mostra um alerta para confirmar o envio de equipe e atualiza o status.
- `sendCloseRequest()`: Mostra um alerta para confirmar o fechamento da ocorrência e atualiza o status.
- `updateStatusAndFirebase(newStatus: string)`: Atualiza o status da ocorrência no Firebase e emite evento para atualização externa.
- `showAlert(header, message)`: Alerta genérico.



# DOCUMENTAÇÃO TÉCNICA - TELA " ADMINISTRAÇÃO DE USUÁRIOS"

## Acesso do Administrador:

- Veja o passo a passo em vídeo:

<https://youtube.com/shorts/240KF7XUYHo>

- Dados para login:

Login: **testeadministrador@administrador.com**

Senha: **123456**

Editar Usuário

FECHAR



Imagem Atual

SELECIONAR NOVA IMAGEM

Nome de Usuário

Bruno Cezario

Email

brunoscezarior@souunisuam.com.br

Perfil

Administrador ▾

CEP

21041-020

BUSCAR CEP

UF

RJ

Município

Rio de Janeiro

Rua

Avenida Paris

Bairro

Bonsucesso

Complemento

## Objetivo

A UsuariosPage é uma página de administração responsável por listar, buscar, editar, excluir e cadastrar novos usuários. Ela utiliza serviços do Firebase para gerenciar os dados e a interface Ionic para exibição e interação.

Usuarios

NOVO USUÁRIO

Q Search

|                                                                                     |                           |                  |        |         |
|-------------------------------------------------------------------------------------|---------------------------|------------------|--------|---------|
|    | Usuário: Bruno Cezario    | Perfil: administ | EDITAR | EXCLUIR |
|   | Usuário: Oswaldo          | Perfil: adminis  | EDITAR | EXCLUIR |
|  | Usuário: cidadão          | Perfil: cidadao  | EDITAR | EXCLUIR |
|  | Usuário: Andre            | Perfil: f gestor | EDITAR | EXCLUIR |
|  | Usuário: cidadão 2        | Perfil: cidadao  | EDITAR | EXCLUIR |
|  | Usuário: adm              | Perfil: administ | EDITAR | EXCLUIR |
|  | Usuário: Gestor           | Perfil: f gestor | EDITAR | EXCLUIR |
|  | Usuário: mestrado unisuam | Perfil: admini   | EDITAR | EXCLUIR |

Usuarios

Perfil



# IAPCI

## Crie um Cadastro

Nome de Usuário:

Email:

Senha:

Confirme Senha:

Perfil:

CEP:

UF:

Município:

Rua:

Bairro:

Complemento:

Imagem

Escolher arquivo

Nenhum arquivo escolhido

**CADASTRAR**

**2**

**DOCUMENTAÇÃO  
TÉCNICA**

# FIREBASE

Este serviço (FirebaseService) encapsula toda alógica relacionada à autenticação de usuários, cadastro, armazenamento de dados, upload de imagens, gerenciamento de relatórios e consultas ao banco de dados Firebase Realtime Database e Firebase Storage. É parte de uma aplicação desenvolvida com Angular e Firebase (usando @angular/fire/compat).

## Configuração Inicial

Antes de utilizar o serviço, é necessário instalar os pacotes: npm install firebase @angular/fire

## Configuração no Angular

No environment.ts, adicione sua configuração Firebase:

```
export const environment = {  
  production: false,  
  firebase: { apiKey:  
    "AIzaSyBj3gMju5s8zoTbSQbBi  
    ORTII_Pj LI21Nk",  
    authDomain: "mestrado-  
    unisuam- iapci-  
    63b34.firebaseio.com",  
    projectId: "mestrado-unisuam-  
    iapci- 63b34",  
    storageBucket: "mestrado-  
    unisuam- iapci-  
    63b34.appspot.com",  
    messagingSenderId:  
    "333245454798",  
    appId:  
    "1:333245454798:web:4325aa458  
    28ee cd861739f",  
    measurementId: "G-  
    9BMWCGJKB8"},  
  googleMapsApiKey:  
    "AIzaSyDXgJkiNP4PZD6v6f5x  
    uOPh1M NXJbsZGBM"  
};
```

## Componentes Utilizados

| Componente            | Função                                                               |
|-----------------------|----------------------------------------------------------------------|
| AngularFireAuth       | Autenticação com Firebase (login, registro, logout, update de email) |
| AngularFireDatabase   | Interações com Realtime Database (leitura, escrita, update, delete)  |
| AngularFireStorage    | Upload e leitura de arquivos (imagens) no Storage                    |
| HttpClient            | Requisição externa para consulta de CEP via ViaCEP                   |
| UserService           | Serviço auxiliar para armazenar temporariamente o nome do usuário    |
| rxjs e rxjs/operators | Tratamento reativo de observables                                    |

## Interfaces

```
interface User { uid?: string;
userName:string; email: string;
password:string;
confirmPassword: string; profile:
string; cep?: string;
uf?: string; municipio?: string;
rua?: string; bairro?: string;
complemento?: string;
imageUrl?: string;}
```

```
interface AddressDetails { city:
string; neighborhood: string;}
export interface Report { rid:
string;
body: ReportBody | null; type:
string;
url: string;}
```

## **Autenticação de Usuário**

registerUser(user: User)

Registra um novo usuário com email e senha e salva os dados no Realtime

Database em /users/{uid}.

login(email: string, password: string)

Autentica o usuário com email e senha. logout()

Realiza o logout do usuário atual. getCurrentUserId()

Retorna o uid do usuário autenticado.

## **Perfil de Usuário**

getUserProfile(uid: string)

Obtém o campo profile de um usuário (cidadao, gestor, administrador...).

isGestor, isCidadao,

isAdministrador Funções que verificam o perfil do usuário.

getUserProfileData(uid)

Retorna todos os dados de um usuário específico.

getAllUsers()

Retorna todos os usuários cadastrados.

updateUser(uid, newData)

Atualiza os dados do usuário.

deleteUser(uid)

Remove o usuário do banco.

updateUserEmail(newEmail)

Atualiza o e-mail do usuário autenticado.

## **Endereço (ViaCEP)**

getAddressDetailsByPostalCode (postal Code)

Faz requisição ao ViaCEP para obter cidade e bairro a partir do CEP informado.

Upload de Imagens

## **UploadImage**

(imageFile: File)

Faz upload da imagem para o caminho images/ no Firebase Storage e retorna a URL da imagem.

uploadImageAndGetUrl(image

File: File) Versão alternativa

com exibição da porcentagem de progresso no console.

getImageUrl(imagePath)

Retorna a URL pública de um arquivo no Storage.

## Relatórios (Reports)

`sendData(data)`

Envia um novo relatório para o nó

`/reports.`

`getOccurrences()`

Obtém todos os relatórios.

`getCurrentReportsRid()`

Retorna o rid (ID) do último relatório baseado no timestamp.

`getReportByRid(rid)`

Retorna um relatório específico a partir do seu rid.

`updateReport(rid, newData)`

Atualiza os dados de um relatório.

## Integração com Serviço de Sessão

`getCurrentUserProfile()`

Carrega e define o nome do usuário atual no UserService.

`getUserName(uid)`

Obtém os dados completos de um usuário específico.

## Exemplo de Uso (Registro)

```
this.firebaseService.registerUser(  
  { userName: 'João',  
    email: 'joao@email.com',  
    password: '123456',  
    confirmPassword: '123456',  
    profile: 'cidadao'  
  }).then(() => {  
    console.log('Usuário cadastrado  
com sucesso!');  
  }).catch(err => {  
    console.error(err);  
  });
```

## Estrutura Firebase

```
{  
  "users": {  
    "uid123": { "userName": "João",  
      "email": "joao@email.com",  
      "profile": "cidadao",  
      ...  
    }  
  },  
  "reports": {  
    "rid123": { "uid": "uid123",  
      "description": "...",  
      "occurrences": "...",  
      "photoUrls": [...], "status":  
      "pendente",  
      "timestamp": 1234567890  
    }  
  }  
}
```

# API

O GmapService é um serviço singleton Angular, projetado para carregar dinamicamente o Google Maps JavaScript SDK em tempo de execução. Ele encapsula toda a lógica necessária para verificar a presença do SDK, inserir dinamicamente o `<script>` correspondente no DOM, e resolver uma Promise contendo o módulo `google.maps` após o carregamento.

Esse serviço segue o princípio da responsabilidade única e permite que outros componentes ou serviços possam consumir o SDK do Google Maps de forma segura, desacoplada e controlada.

Permitindo a visualização do mapa e realizando a marcação usando marcadores

- `@angular/core`: Para definição de um serviço injetável.
- `environment`: Usado para acessar a chave da API `googleMapsApiKey`, mantida fora do código-fonte por segurança.

**Oswaldo Borges Peres  
Bruno Santos Cezário  
André Luís Azevedo Guedes**

**Documentação técnica &  
Manual do usuário do IAPCI  
V3.0**

