**Produto 2**

**Banco de dados GIP**

**Cooperação Alemã para o Desenvolvimento**

**Agência da GIZ em Brasília**

Contrato 83463063

Sumário

[**1. CONTRATADA 1**](#_heading=h.1ksv4uv)

[**2. CONTRATANTE 1**](#_heading=h.2jxsxqh)

[**3. CONTRATO 2**](#_heading=h.z337ya)

[**3.1 Objeto do contrato 2**](#_heading=h.3j2qqm3)

[**3.2 Termos de referência 2**](#_heading=h.1y810tw)

[**3.3 Período de atuação 2**](#_heading=h.4i7ojhp)

[**4. PRODUTO 2 - OBJETO 2**](#_heading=h.2xcytpi)

[**5. CONTEÚDO ENTREGÁVEL 4**](#_heading=h.1ci93xb)

[5.1 ENTREGÁVEL 1 - OBTENÇÃO DOS DADOS 4](#_heading=h.keivnxqckg6)

[Pedir os dados das outras camadas principais  
Imóveis Privados Cadastrados no Sigef 5](#_heading=h.xworaljm1h7k)

[5.2 ENTREGÁVEL 2 - PROCESSO DE EXTRAÇÃO DE DADOS 6](#_heading=h.p070wzrxkmbk)

[Pegar os dados e os significados no QGIS e trazer pra essa tabela 7](#_heading=h.1ys0brsb1trd)

[**5.3 ENTREGÁVEL 3 - CONSTRUÇÃO DO BANCO DE DADOS 8**](#_heading=h.2bn6wsx)

[Atualizar essas tabelas para os novos nomes de campos e novas tabelas 9](#_heading=h.ybnu3hyrt6sg)

[5.4 ENTREGÁVEL 4 - DADOS INICIAIS 12](#_heading=h.dfy3zw85iyga)

# 

# 1. CONTRATADA

**M42 – Tecnologias WEB**

**Responsável pelo Projeto:**

Heygon de Andrade do Lago

E-mail: heygon@m42.com.br

**Pessoal Técnico**

Jacson Pertussatti - Gerente de projetos e Desenvolvedor Frontend

Lucas Borges - Administrador de banco de dados

Heygon de Andrade do Lago - Desenvolvedor backend e frontend

# 2. CONTRATANTE

**Cooperação Alemã para o Desenvolvimento - Agência da GIZ em Brasília**

**Papeis e responsabilidades.**

**GIZ**

Responsável Administrativo/ Contrato: Daya Rodrigues

E-mail: daya.rodrigues@giz.de

Responsável Técnica do Projeto: Katia Castro de Matteo

e-mail: [katia.matteo@giz.de](mailto:katia.matteo@giz.de)

**Serviço Florestal Brasileiro**

Coordenadora - SFB/MMA: Lilianna Mendes Latini Gomes

E-mail: lilianna.gomes@florestal.gov.br

# 3. CONTRATO

Contrato para Peritos e outros Prestadores de Serviços - 83463063

Prazo Contratual: 14/06/2024 à 31/01/2025

## 3.1 Objeto do contrato

Construção de uma Página Web permitindo a visualização interativa de mapas, camadas de dados geoespaciais e informações relacionadas, facilitando a análise geográfica e consultas personalizadas de dados geoespaciais.

## 3.2 Termos de referência

O(a) Contratado(a) compromete-se a prestar os serviços especificados no Acordo Especial (Anexo 1).

## 3.3 Período de atuação

Com vistas a prestar os serviços especificados o(a) Contratado(a) colocará à disposição, no período previsto de 14.06.2024 a 31.01.2025.

# 4. PRODUTO 2 - OBJETO

O objetivo do produto 2, a construção do banco de dados da GIP, envolve primeiramente a coleta dos dados dos imóveis e proprietários presentes em arquivos de Excel. Esses dados, que podem incluir uma variedade de informações como endereços, tamanhos de propriedades, valores de mercado e detalhes de propriedade, são a base fundamental para o funcionamento do sistema. A tarefa inicial é garantir que esses dados sejam extraídos de forma precisa e completa, evitando a perda ou corrupção de informações vitais durante o processo de transferência. Este processo requer uma análise meticulosa dos arquivos de Excel para identificar e classificar todas as variáveis relevantes que precisam ser armazenadas no banco de dados.

Além disso, a coleta de dados envolve a limpeza e normalização dos mesmos, assegurando que qualquer dado inconsistente ou duplicado seja corrigido ou removido antes da inserção no banco de dados. A normalização dos dados é crucial para evitar redundâncias e garantir que cada peça de informação esteja única e claramente definida. Esta etapa preparatória estabelece as bases para um banco de dados eficiente e funcional, facilitando o acesso e a análise posterior dos dados. Todo esse processo de coleta e organização de dados é essencial para criar uma base sólida para o desenvolvimento subsequente do banco de dados relacional, garantindo a integridade e a qualidade dos dados que serão utilizados.

Com os dados devidamente coletados e organizados, a próxima fase do projeto envolve a construção do banco de dados relacional utilizando o PostgreSQL. O PostgreSQL foi escolhido por sua robustez, flexibilidade e capacidade de lidar com grandes volumes de dados de maneira eficiente. A construção do banco de dados envolve a criação de uma estrutura relacional que permita o armazenamento organizado e a interconexão de dados dos imóveis e proprietários. Isso é alcançado através da definição de tabelas e a implementação de relações entre essas tabelas, garantindo que os dados possam ser acessados e manipulados de maneira lógica e eficiente.

A modelagem do banco de dados inclui a definição de chaves primárias e estrangeiras, índices, e outros mecanismos de integridade referencial que asseguram a consistência e a integridade dos dados. Além disso, a utilização do PostgreSQL oferece recursos avançados como suporte a dados geoespaciais, que podem ser particularmente úteis para aplicações que envolvem a localização e a análise de propriedades. A construção do banco de dados também envolve a implementação de procedimentos armazenados e gatilhos que ajudam a automatizar tarefas comuns e garantir que as operações de manipulação de dados sejam realizadas de maneira segura e eficiente.

A última etapa do objetivo do produto 2 é garantir que o backend possa acessar e manipular os dados armazenados no banco de dados de forma eficiente. O backend é responsável por tratar as solicitações do usuário final, realizando operações como consultas, atualizações e geração de relatórios. Para alcançar isso, o backend deve ser desenvolvido com a capacidade de se comunicar eficazmente com o banco de dados PostgreSQL, utilizando linguagens de programação adequadas e bibliotecas que suportem interações com bancos de dados relacionais.

Essa integração é crucial para assegurar que o usuário final tenha acesso rápido e consistente aos dados sempre que realizar alguma requisição. O backend deve ser capaz de otimizar consultas e gerenciar a carga de trabalho para garantir que o desempenho do sistema seja mantido, mesmo quando o número de usuários ou a quantidade de dados crescer. Além disso, a integração com o backend deve considerar aspectos de segurança, garantindo que apenas usuários autorizados possam acessar ou modificar os dados. A combinação de uma estrutura de banco de dados bem projetada e um backend eficiente resultará em um sistema robusto, capaz de atender às necessidades dos usuários finais de forma confiável e eficaz.

# 5. CONTEÚDO ENTREGÁVEL

## 5.1 ENTREGÁVEL 1 - OBTENÇÃO DOS DADOS

Os dados iniciais do projeto foram fornecidos em arquivos no formato de shapefile. Esse formato é amplamente reconhecido e utilizado em diversas plataformas de Sistemas de Informação Geográfica (SIG), incluindo PostGIS, ArcGIS, QGIS e TerraView. A escolha de manter os dados no formato shapefile será estratégica, visando garantir a compatibilidade e a facilidade de uso em diferentes programas de georreferenciamento. A universalidade do formato shapefile assegura que os dados possam ser facilmente exportados e importados entre esses sistemas, minimizando problemas de interoperabilidade.

Os arquivos shapefile fornecem uma forma eficiente de armazenar e manipular dados geoespaciais, permitindo que sejam utilizados para diversas análises e visualizações geográficas. Essa abordagem também permite que os dados sejam compartilhados entre diferentes plataformas e equipes, garantindo que todas as partes envolvidas possam acessar e trabalhar com as mesmas informações de maneira eficiente e sem complicações.

## (Imagem 1 - Imóveis representados de forma georreferenciadas no arquivo shapefile)

Os dados contidos nos arquivos shapefile incluem várias informações cruciais. Entre elas, estão os limites das propriedades, dados sobre a topografia, uso da terra, vegetação e outros atributos relevantes para a análise geográfica. Essas informações são fundamentais para a criação de um banco de dados georreferenciado, que permitirá a visualização e a análise espacial de maneira integrada. A riqueza de detalhes presentes nos arquivos shapefile é aproveitada para construir um sistema robusto e preciso, capaz de atender às necessidades dos usuários finais e fornecer insights valiosos.

Os arquivos shapefile fornecidos pela GIP contêm informações básicas sobre os imóveis registrados. Esses dados incluem atributos essenciais e comuns que aparecem em várias camadas, como o nome do imóvel, o código do imóvel, o município, o estado e a área total. Essas informações são fundamentais para identificar e categorizar os imóveis de maneira eficaz.

Cada camada de dados possui um conjunto de características e dados específicos, que são armazenados em tabelas dedicadas dentro do banco de dados. Esses atributos menos comuns, que podem incluir detalhes mais granulares ou específicos sobre os imóveis, são tratados de forma separada para garantir uma organização clara e evitar redundâncias desnecessárias. Por exemplo, dados sobre o tipo de uso do solo, características ambientais ou infra-estruturas podem ser exclusivos de uma camada específica e, portanto, são registrados em tabelas distintas.

Essa abordagem permite uma gestão mais eficiente dos dados, facilitando a consulta e a análise detalhada de informações específicas. Além disso, ao armazenar dados em tabelas especializadas, o sistema pode oferecer funcionalidades avançadas de consulta e análise, permitindo aos usuários combinar informações de diferentes camadas para realizar estudos comparativos ou avaliações detalhadas.

Essa estrutura de armazenamento e organização de dados foi projetada para manter a integridade e a consistência das informações, além de permitir uma fácil expansão do banco de dados conforme novos tipos de dados ou atributos sejam identificados e necessitam ser incorporados. Com essa organização, o sistema é capaz de fornecer uma visão abrangente e detalhada dos imóveis, suportando uma ampla gama de necessidades analíticas e operacionais.

## (Imagem 2 - Exemplo de dados contidos no arquivo shapefile da GIP)

Durante a fase de processamento, os dados são organizados para garantir a sua precisão e consistência. Essa etapa pode incluir a correção de erros de georreferenciamento, a eliminação de dados duplicados e a padronização dos atributos.

Para implementar a leitura e processamento dos arquivos shapefile, a solução proposta utiliza o Node.js como a principal tecnologia de backend. Entre as bibliotecas escolhidas para essa tarefa, destaca-se a "Streaming Shapefile Parser". Esta biblioteca é especializada na leitura de arquivos shapefile, que são amplamente utilizados para armazenar dados geoespaciais. A função principal da "Streaming Shapefile Parser" é transformar o conteúdo desses arquivos em objetos de dados estruturados. Esses objetos de dados são, então, armazenados em um banco de dados, facilitando o acesso e a manipulação pelo sistema posteriormente.

| const shapefile = require('shapefile');  shapefile.open("path/to/your/shapefile.shp")  .then(*source* => *source*.read()  .then(function log(*result*) {  if (*result*.done) return;  console.log(*result*.value); // Output the data object  return *source*.read().then(log);  }))  .catch(*error* => console.error(*error*.stack)); |
| --- |

(Exemplo de código utilizando a biblioteca "Streaming Shapefile Parser" para ler os arquivos no formato shapefile )

Além disso, outra biblioteca essencial no processo é a "dbf-reader". Esta ferramenta é particularmente importante para lidar com os dados de geolocalização contidos nos arquivos shapefile. Especificamente, a "dbf-reader" extrai informações sobre longitude e latitude dos imóveis. Estes dados são fundamentais para a criação de marcações georreferenciadas, que podem ser visualizadas e manipuladas em mapas digitais. A capacidade de projetar essas marcações de maneira precisa é crucial para análises geoespaciais detalhadas e para a visualização eficaz das informações espaciais relacionadas aos imóveis.

| const DBFFile = require('dbf-reader');  // Especificar o caminho para o arquivo DBF  const dbPath = "path/to/your/shapefile.dbf";  // Ler o arquivo DBF  DBFFile.open(dbfPath)  .then(*dbf* => {  console.log(`DBF contém ${*dbf*.recordCount} registros.`);  return *dbf*.readRecords();  })  .then(*records* => {  *records*.forEach(*record* => {  console.log(*record*); // Output the record data  });  })  .catch(*error* => console.error(*error*.stack)); |
| --- |

(Exemplo de código utilizando a biblioteca "dbf-reader" para ler os arquivos no formato dbf para coletar os dados de georreferenciamento )

A escolha dessas bibliotecas foi motivada pela necessidade de uma solução eficiente e escalável para lidar com grandes volumes de dados geoespaciais. O Node.js, com sua arquitetura de eventos não bloqueantes, é particularmente adequado para este tipo de processamento de dados, pois pode gerenciar múltiplas operações de leitura e escrita simultaneamente, sem comprometer o desempenho do sistema.

## 5.2 ENTREGÁVEL 2 - PROCESSO DE EXTRAÇÃO DE DADOS

O uso dessas bibliotecas e tecnologias citadas acima, permite uma integração fluida entre os dados geoespaciais e o banco de dados, garantindo que as informações sejam armazenadas de forma organizada e que possam ser acessadas e manipuladas conforme necessário. Além disso, a precisão na conversão e armazenamento dos dados geoespaciais é crucial para garantir que as análises realizadas posteriormente sejam baseadas em informações corretas e confiáveis.

| const shapefile = require('shapefile');  const DBFFile = require('dbf-reader');  const { Client } = require('pg');  // Configuração do PostgreSQL  const client = **new** Client({  user: 'seu\_usuario',  host: 'localhost',  database: 'seu\_banco\_de\_dados',  password: 'sua\_senha',  port: 5432,  });  client.connect();  const shaPath = "path/to/your/shapefile.shp";  const dbPath = "path/to/your/shapefile.dbf";  // Função para inserir dados no banco de dados  async function insertData(*data*) {  const query = `  INSERT INTO geospatial\_data (id, name, longitude, latitude)  VALUES ($1, $2, $3, $4)  RETURNING id`;  try {  const res = await client.query(query, *data*);  console.log(`Inserted row with ID: ${res.rows[0].id}`);  } catch (err) {  console.error(err.stack);  }  }  // Ler o shapefile  shapefile.open(shaPath)  .then(*source* => *source*.read()  .then(async function processShape(*result*) {  if (*result*.done) return;  const shapeData = *result*.value;  console.log("Shape:", shapeData);  // Extraia os dados necessários do shapefile para o banco de dados aqui  // Por exemplo, usando shapeData.properties ou outros atributos  await insertData([shapeData.id, shapeData.properties.name, shapeData.properties.longitude, shapeData.properties.latitude]);  return *source*.read().then(processShape);  }))  .catch(*error* => console.error(*error*.stack));  // Ler o arquivo DBF  DBFFile.open(dbfPath)  .then(*dbf* => *dbf*.readRecords())  .then(*records* => {  *records*.forEach(*record* => {  console.log("DBF Record:", *record*);  // Extraia os dados necessários do DBF para o banco de dados aqui  // Por exemplo:  insertData([*record*.id, *record*.name, *record*.longitude, *record*.latitude]);  });  })  .catch(*error* => console.error(*error*.stack));  // Feche a conexão com o banco de dados após a conclusão  process.on('exit', () => {  client.end();  }); |
| --- |

(Exemplo de código utilizando ambas as bibliotecas para demonstrar a integração entre as duas de forma a ler os arquivos combinados para coletar os dados e salvá-los em um banco de dados)

Portanto, a combinação do Node.js com as bibliotecas "Streaming Shapefile Parser" e "dbf-reader" representa uma solução robusta e eficiente para o processamento de dados geoespaciais. Isso não só facilita o armazenamento e a leitura dos dados pelo sistema, mas também garante que os usuários tenham acesso a informações precisas e geograficamente corretas para análise e tomada de decisão. Este processo de integração e manipulação de dados é uma parte essencial do sistema, suportando tanto as operações diárias quanto as análises estratégicas.

A limpeza e a normalização dos dados são essenciais para garantir que as análises subsequentes sejam precisas e confiáveis. Uma vez processados, os dados são integrados ao sistema de informações geográficas, onde podem ser visualizados e manipulados pelos usuários finais.

A utilização dos dados processados permite a realização de diversas análises geográficas, incluindo a identificação de padrões espaciais, a avaliação de mudanças no uso da terra e a geração de relatórios detalhados. Os usuários podem interagir com os dados através de uma interface web intuitiva, desenvolvida em React, que facilita a navegação e a visualização das informações geoespaciais. A interface permite que os usuários consultem e cruzem dados de diferentes fontes, gerando novas análises e insights.

## 5.3 ENTREGÁVEL 3 - CONSTRUÇÃO DO BANCO DE DADOS

A criação de um banco de dados relacional visa receber e classificar informações em diversas tabelas, cada uma projetada para um propósito e tipo de informação específica. O objetivo principal é evitar a duplicidade de dados e assegurar a integridade e organização das informações armazenadas.

Para atingir esse objetivo, cada tabela no banco de dados é projetada com um identificador de referência exclusivo. Esse identificador permite a ligação entre tabelas por meio de campos comuns, facilitando a integração e a relação entre diferentes conjuntos de dados. Isso garante que os dados possam ser correlacionados de maneira eficiente, permitindo uma análise mais abrangente e detalhada.

A estrutura do banco de dados é desenvolvida para manter a consistência e integridade dos dados, assegurando que todas as informações relevantes possam ser acessadas e gerenciadas de forma eficaz. A organização cuidadosa das tabelas e a utilização de identificadores de referência contribuem para uma base de dados robusta e confiável, essencial para a gestão de informações complexas e inter-relacionadas.

A construção do banco de dados também envolve a definição de esquemas e a implementação de regras de validação para assegurar que os dados inseridos atendam aos requisitos do projeto. Isso inclui a normalização das tabelas para eliminar redundâncias e otimizar o desempenho das consultas. Além disso, são aplicadas técnicas de indexação para melhorar a velocidade de acesso e a recuperação dos dados.

A seguir vamos detalhar as tabelas e os dados que serão armazenados em cada uma delas e como os campos em comum entre as tabelas servirão de elo e responsáveis por interligar as tabelas evitando assim a duplicidade de dados.

## (Imagem 3 - Modelo de tabelas que compõem o banco de dados desenvolvido para armazenar as propriedades e camadas)

## 5.3.1 - Detalhamento das tabelas do banco de dados

Para uma melhor compreensão da entrega do produto 2, as tabelas criadas serão descritas de duas maneiras distintas. Primeiramente, será apresentada uma lista dos dados que serão armazenados em cada tabela. Em seguida, será exibido o código utilizado para a criação de tabelas, especificando os tipos de dados que podem ser armazenados. Por exemplo, o campo **"nome"** na tabela de **usuários** é responsável por armazenar o nome dos usuários, e esse campo pode receber uma sequência de caracteres com um limite de até 255 caracteres, representado pelo tipo de dado **`varchar(255)`**.

Essa apresentação detalhada dos dados e da estrutura das tabelas é essencial para permitir que técnicos com conhecimento em banco de dados possam reproduzir o modelo em outros ambientes ou realizar a recuperação de tabelas em caso de emergências. A documentação clara do esquema de banco de dados facilita o entendimento e a manutenção do sistema, assegurando que todos os dados sejam armazenados e gerenciados de maneira eficiente e consistente, como demonstrado na figura 3 referente ao modelo de dados.

Além disso, foram aplicadas algumas técnicas de normalização para evitar a duplicação desnecessária de dados. Uma dessas técnicas é a utilização de **chaves estrangeiras**, que interligam diferentes tabelas por meio de campos comuns. Por exemplo, no banco de dados, existe uma tabela denominada **"Imóveis"** e outra chamada **"Proprietários"**. A tabela **"Proprietários"** possui um campo chamado **"ID"**, que é um número sequencial único para cada registro. Este campo **"ID"** pode ser referenciado na tabela **"Imóveis"** como **"IdProprietario"**, estabelecendo uma relação entre as duas tabelas.

Com essa estrutura de chaves estrangeiras, quando for necessário consultar todas as propriedades associadas a um determinado proprietário, basta procurar pelo número de **"ID"** contido no campo **"IdProprietario"** na tabela **"Imóveis"**. Essa abordagem permite que as consultas sejam realizadas de maneira rápida e eficiente, além de garantir que os dados não sejam duplicados, mantendo a integridade e consistência do banco de dados.

A implementação de chaves estrangeiras é uma prática comum em bancos de dados relacionais, pois elas ajudam a manter a integridade referencial. Isso significa que cada valor de chave estrangeira em uma tabela corresponde a um valor válido na tabela referenciada. Por exemplo, todos os valores no campo **"IdProprietario"** na tabela **"Imóveis"** devem corresponder a valores existentes no campo **"ID"** da tabela **"Proprietários"**. Essa integridade referencial é crucial para garantir que os dados em diferentes tabelas estejam sincronizados e consistentes, evitando problemas como referências inválidas ou dados órfãos.

Além das chaves estrangeiras, a definição de tipos de dados adequados para cada campo é outra prática importante. Tipos de dados como **`varchar`**, **`integer`**, **`date`**, entre outros, são usados para definir que tipo de informação pode ser armazenada em cada campo. Isso não só ajuda a garantir que os dados sejam inseridos corretamente, mas também otimiza o armazenamento e a consulta desses dados. Por exemplo, usar **`varchar(255)`** para campos de texto garante que o armazenamento seja eficiente, ao mesmo tempo em que impõe um limite razoável no tamanho dos dados.

Em resumo, a criação e documentação detalhada das tabelas do banco de dados do sistema GIP não só facilitam a manutenção e a expansão do sistema, mas também garantem que os dados sejam gerenciados de maneira eficiente e segura. A utilização de chaves estrangeiras e a definição de tipos de dados são práticas essenciais para manter a integridade e consistência dos dados, permitindo que o sistema funcione de maneira eficaz e sem problemas. Isso também facilita a vida dos técnicos e administradores de banco de dados, que podem facilmente entender, reproduzir e manter o esquema do banco de dados conforme necessário.

**Tabelas de detalhes da área**

| Dados a serem armazenados   | id | | --- | | idimovel | | idareafip | | nomearea | | dataregistro | | idtipoarea | | idtipoclassificacaoarea | | idestrategia | | CREATE TABLE areadetalhe(  id SERIAL NOT NULL,  idimovel integer,  idareafip integer,  nomearea text,  dataregistro date,  idtipoarea integer,  idtipoclassificacaoarea integer,  idestrategia integer,  PRIMARY KEY(id),  CONSTRAINT areadetalhe\_idareafip\_fkey FOREIGN key(idareafip) REFERENCES tipoclassificacaoarea(id),  CONSTRAINT areadetalhe\_idestrategia\_fkey FOREIGN key(idestrategia) REFERENCES estrategiafip(id),  CONSTRAINT areadetalhe\_idimovel\_fkey FOREIGN key(idimovel) REFERENCES imoveis(id),  CONSTRAINT areadetalhe\_idtipoarea\_fkey FOREIGN key(idtipoarea) REFERENCES tipoarea(id),  CONSTRAINT areadetalhe\_idtipoclassificacaoarea\_fkey FOREIGN key(idtipoclassificacaoarea) REFERENCES classificacaofip(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Tabela classificação fip**

| Dados a serem armazenados   | id | | --- | | descclassificacao | | CREATE TABLE classificacaofip(  id SERIAL NOT NULL,  descclassificacao text,  PRIMARY KEY(id)  ); |
| --- | --- | --- | --- |

**Tabela comentários**

| Dados a serem armazenados   | id | | --- | | comentario | | data | | status | | likes | | quemcomentou | | quemaprovou | | idnoticia | | CREATE TABLE comentarios(  id SERIAL NOT NULL,  comentario text,  "data" varchar(255),  status varchar(255),  likes varchar(255),  quemcomentou integer,  quemaprovou integer,  idnoticia integer,  PRIMARY KEY(id),  CONSTRAINT comentarios\_idnoticia\_fkey FOREIGN key(idnoticia) REFERENCES noticias(id),  CONSTRAINT comentarios\_quemaprovou\_fkey FOREIGN key(quemaprovou) REFERENCES usuarios(id),  CONSTRAINT comentarios\_quemcomentou\_fkey FOREIGN key(quemcomentou) REFERENCES usuarios(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Tabela contatos**

| Dados a serem armazenados   | id | | --- | | nome | | foto | | telefone | | celular | | email | | status | | quemcriou | | CREATE TABLE contatos(  id SERIAL NOT NULL,  nome varchar(255),  foto varchar(255),  telefone varchar(255),  celular varchar(255),  email varchar(255),  status varchar(255),  quemcriou integer,  PRIMARY KEY(id),  CONSTRAINT contatos\_quemcriou\_fkey FOREIGN key(quemcriou) REFERENCES usuarios(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Tabela descrição da recomendação**

| Dados a serem armazenados   | id | | --- | | descrecomendacao | | shortrecomendacao | | recomendacaoid | | CREATE TABLE descrecomendacao(  id SERIAL NOT NULL,  descrecomendacao text,  shortrecomendacao text,  recomendacaoid integer,  PRIMARY KEY(id),  CONSTRAINT descrecomendacao\_recomendacaoid\_fkey FOREIGN key(recomendacaoid) REFERENCES recomendacoes(id)  ); |
| --- | --- | --- | --- | --- | --- |

**Tabela documentos**

| Dados a serem armazenados   | id | | --- | | titulo | | descricao | | url | | downloads | | status | | quemcriou | | CREATE TABLE documentos(  id SERIAL NOT NULL,  titulo varchar(255),  descricao text,  url text,  downloads varchar(255),  status varchar(255),  quemcriou integer,  PRIMARY KEY(id),  CONSTRAINT documentos\_quemcriou\_fkey FOREIGN key(quemcriou) REFERENCES usuarios(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Tabela estratégia fip**

| Dados a serem armazenados   | id | | --- | | descestrategia | | CREATE TABLE estrategiafip(  id SERIAL NOT NULL,  descestrategia text,  PRIMARY KEY(id)  ); |
| --- | --- | --- | --- |

**Tabela geolocalização**

| Dados a serem armazenados   | id | | --- | | idimovel | | idarea | | geom | | nomeprodutor | | nomepropriedade | | tipo | | recomendacao | | estrategia | | visitas | | cpfprodutor | | telefoneprodutor | | latitude | | longitude | | shortrecomendation | | cercamentodearea | | plantiodemudas | | construcaodeaceiros | | remocaodepragas | | subsolagem | | controlequimico | | analisedesolo | | na | | outras | | status | | CREATE TABLE geolocalizacao(  id SERIAL NOT NULL,  idimovel integer,  idarea integer,  geom text,  nomeprodutor varchar(255),  nomepropriedade varchar(255),  regiaopropriedade varchar(255),  tipo varchar(255),  recomendacao text,  estrategia text,  visitas integer,  cpfprodutor varchar(255),  telefoneprodutor varchar,  latitude varchar,  longitude varchar,  shortrecomendation text,  cercamentodearea varchar,  plantiodemudas varchar,  construcaodeaceiros varchar,  remocaodepragas varchar,  subsolagem varchar,  controlequimico varchar,  analisedesolo varchar,  na varchar,  outras varchar,  status varchar,  PRIMARY KEY(id),  CONSTRAINT geolocalizacao\_idimovel\_fkey FOREIGN key(idimovel) REFERENCES imoveis(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Tabela grupo recomendação**

| Dados a serem armazenados   | id | | --- | | descgroup | | CREATE TABLE grouprecomendation(  id SERIAL NOT NULL,  descgroup text,  PRIMARY KEY(id)  ); |
| --- | --- | --- | --- |

**Tabela imóveis**

| Dados a serem armazenados   | id | | --- | | idpropriedade | | nome | | area | | areaprodutiva | | areaapp | | arearl | | regiao | | bacia | | statusimovel | | tecnicoresponsavel | | supervisorresponsavel | | dataultimavisita | | situacaoimovel | | status | | longitude | | latitude | | idproprietario | | quemcadastrou | | quantidadevisitas | | areatecnologiasabc | | areaplanejamentoisa | | CREATE TABLE imoveis(  id SERIAL NOT NULL,  idpropriedade integer,  nome varchar(255),  area varchar(255),  areaprodutiva varchar(255),  areaapp varchar(255),  arearl varchar(255),  regiao varchar(255),  bacia varchar(255),  statusimovel varchar(255),  tecnicoresponsavel varchar(255),  supervisorresponsavel varchar(255),  dataultimavisita date,  situacaoimovel varchar(255),  status varchar(255),  latitude double precision,  longitude double precision,  idproprietario integer,  quemcadastrou integer,  quantiadevisitas integer,  areatecnologiasabc varchar(255),  areaplanejamentoisa varchar(255),  PRIMARY KEY(id),  CONSTRAINT imoveis\_idproprietario\_fkey FOREIGN key(idproprietario) REFERENCES proprietarios(id),  CONSTRAINT imoveis\_usuarios\_fk FOREIGN key(id) REFERENCES usuarios(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Tabela mapas temáticos**

| Dados a serem armazenados   | id | | --- | | create\_time | | nome | | descricao | | imoveis | | coordenadas | | status | | quemcriou | | views | | CREATE TABLE mapastematicos(  id SERIAL NOT NULL,  create\_time date,  nome varchar(255),  descricao text,  imoveis json,  coordenadas json,  status varchar(255),  quemcriou integer,  views integer,  PRIMARY KEY(id),  CONSTRAINT mapastematicos\_quemcriou\_fkey FOREIGN key(quemcriou) REFERENCES usuarios(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Tabela notícias**

| Dados a serem armazenados   | id | | --- | | titulo | | conteudo | | data | | publicado | | ativado | | linkoriginal | | anexo | | quemaprovou | | quemescreveu | | escritor | | numerodeviews | | CREATE TABLE noticias(  id SERIAL NOT NULL,  titulo varchar(255),  conteudo text,  "data" varchar(255),  publicado varchar(255),  ativado varchar(255),  linkoriginal text,  anexo text,  quemaprovou integer,  quemescreveu integer,  escritor integer,  numerodeviews integer,  PRIMARY KEY(id),  CONSTRAINT noticias\_quemcriou\_fkey FOREIGN key(quemaprovou) REFERENCES usuarios(id),  CONSTRAINT noticias\_quemescreveu\_fkey FOREIGN key(quemescreveu) REFERENCES usuarios(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Tabela parceiros**

| Dados a serem armazenados   | id | | --- | | nome | | descricao | | link | | logo | | status | | quemcriou | | CREATE TABLE parceiros(  id SERIAL NOT NULL,  nome varchar(255),  descricao text,  link text,  logo varchar(255),  status varchar(255),  quemcriou integer,  PRIMARY KEY(id),  CONSTRAINT parceiros\_quemcriou\_fkey FOREIGN key(quemcriou) REFERENCES usuarios(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Tabela proprietários**

| Dados a serem armazenados   | id | | --- | | nome | | cpf | | telefone | | celular | | sexo | | uf | | email | | status | | quemcadastrou | | CREATE TABLE proprietarios(  id SERIAL NOT NULL,  nome varchar(255),  cpf varchar(255),  telefone varchar(255),  celular varchar(255),  sexo varchar(255),  uf varchar(255),  email varchar(255),  status varchar(255),  quemcadastrou integer,  PRIMARY KEY(id),  CONSTRAINT proprietarios\_quemcadastrou\_fkey FOREIGN key(quemcadastrou) REFERENCES usuarios(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Tabela recomendações**

| Dados a serem armazenados   | id | | --- | | idarea | | idpropriedade | | codigorecomendacao | | subgruporecomendacao | | gruporecomendacao | | statusrecomendacao | | CREATE TABLE recomendacoes(  id SERIAL NOT NULL,  idarea integer,  idpropriedade integer,  codigorecomendacao integer,  subgrouporecomendacao integer,  grouporecomendacao integer,  statusrecomendacao integer,  PRIMARY KEY(id),  CONSTRAINT recomendacoes\_grouporecomendacao\_fkey FOREIGN key(grouporecomendacao) REFERENCES grouprecomendation(id),  CONSTRAINT recomendacoes\_idarea\_fkey FOREIGN key(idarea) REFERENCES areadetalhe(id),  CONSTRAINT recomendacoes\_idpropriedade\_fkey FOREIGN key(idpropriedade) REFERENCES imoveis(id),  CONSTRAINT recomendacoes\_statusrecomendacao\_fkey FOREIGN key(statusrecomendacao) REFERENCES statusrecomendacao(id),  CONSTRAINT recomendacoes\_subgrouporecomendacao\_fkey FOREIGN key(subgrouporecomendacao) REFERENCES subgruporecomendacao(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Tabela status recomendação**

| Dados a serem armazenados   | id | | --- | | descstatus | | CREATE TABLE statusrecomendacao(  id SERIAL NOT NULL,  descstatus varchar(255),  PRIMARY KEY(id)  ); |
| --- | --- | --- | --- |

**Tabela sub grupo recomendação**

| Dados a serem armazenados   | id | | --- | | descsubgrupo | | CREATE TABLE subgruporecomendacao(  id SERIAL NOT NULL,  descsubgrupo varchar(255),  PRIMARY KEY(id)  ); |
| --- | --- | --- | --- |

**Tabela tipo de área**

| Dados a serem armazenados   | id | | --- | | desctipoarea | | CREATE TABLE tipoarea(  id SERIAL NOT NULL,  desctipoarea varchar(255),  PRIMARY KEY(id)  ); |
| --- | --- | --- | --- |

**Tabela tipo de classificação de área**

| Dados a serem armazenados   | id | | --- | | desctipoclassificacaoarea | | CREATE TABLE tipoclassificacaoarea(  id SERIAL NOT NULL,  desctipoclassificacaoarea text,  PRIMARY KEY(id)  ); |
| --- | --- | --- | --- |

**Tabela Usuários**

| Dados a serem armazenados   | id | | --- | | nome | | email | | senha | | instituicao | | cidade | | tipodeparceiro | | estrategiagip | | sexo | | avatar | | status | | perfil | | CREATE TABLE usuarios(  id SERIAL NOT NULL,  nome varchar(255),  email varchar(255),  senha varchar(255),  instituicao varchar(255),  cidade varchar(255),  tipoparceiro varchar(255),  estrategiagip varchar(255),  sexo varchar(255),  avatar varchar(255),  status varchar(255),  perfil integer,  PRIMARY KEY(id)  ); |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Como um exemplo, para facilitar o entendimento dessa parte, teremos uma tabela chamada de **Propriedades** e outra chamada **Proprietários,** a tabela de **Proprietários** teria um campo chamado **ID** que será responsável por armazenar um número sequencial que será único para cada um dos registros, esse campo **ID** pode ser inserido na tabela de **Propriedades** com o nome **IdProprietario,** que terá como objetivo interligar as duas tabelas. Uma vez que o usuário quiser consultar todas as propriedades de um dono, basta procurar por correspondências no campo **IdProprietario,** e dessa forma podemos recuperar uma lista de dados garantindo a velocidade dos dados e que eles não sejam duplicados.

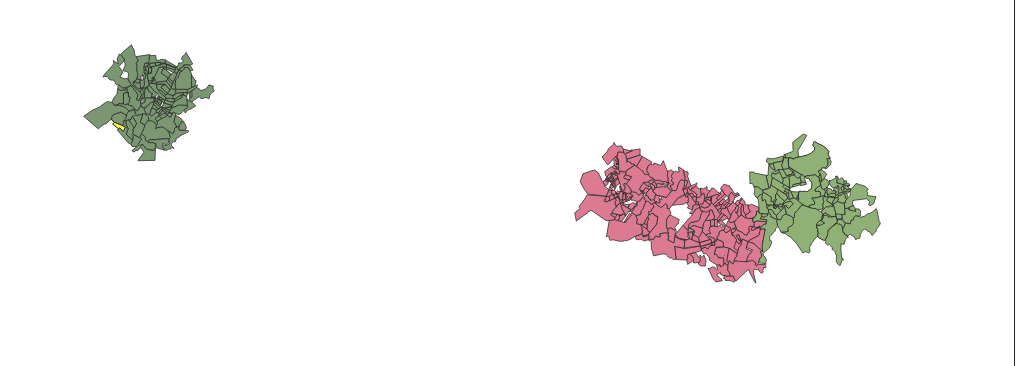
Esse processo de construção do banco de dados é fundamental para o sucesso do projeto, pois fornece a base necessária para o armazenamento e a manipulação eficiente das informações. A utilização de identificadores de referência e a ligação entre tabelas através de campos comuns permitem uma organização lógica e coesa dos dados, facilitando futuras análises e consultas personalizadas.

Em resumo, a tarefa de construção do banco de dados é um passo crucial para garantir que o sistema de gerenciamento de informações funcione de maneira eficiente e confiável, evitando a duplicidade de dados e promovendo a integridade e a acessibilidade das informações armazenadas.

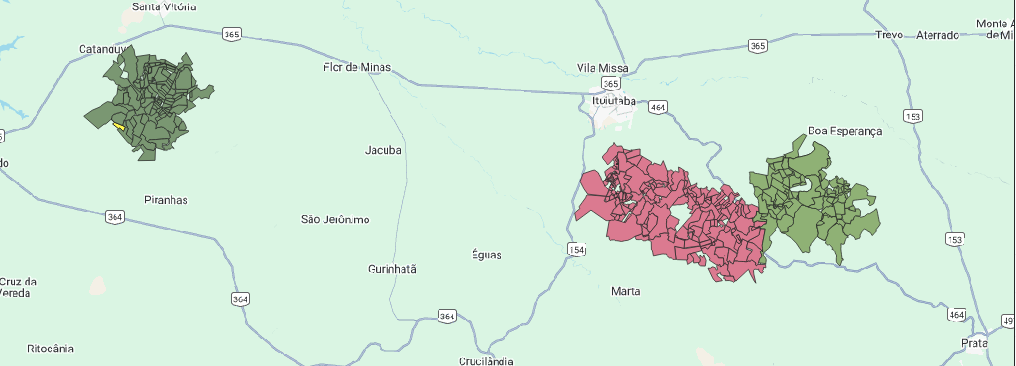
É importante ressaltar que essa modelagem inicial, pode sofrer alterações e evoluções para comportar os requisitos e as necessidades do projeto, caso essas modificações e evoluções venham a acontecer, elas serão informadas no documento de relatório de entrega do projeto.

## 5.4 ENTREGÁVEL 4 - DADOS INICIAIS

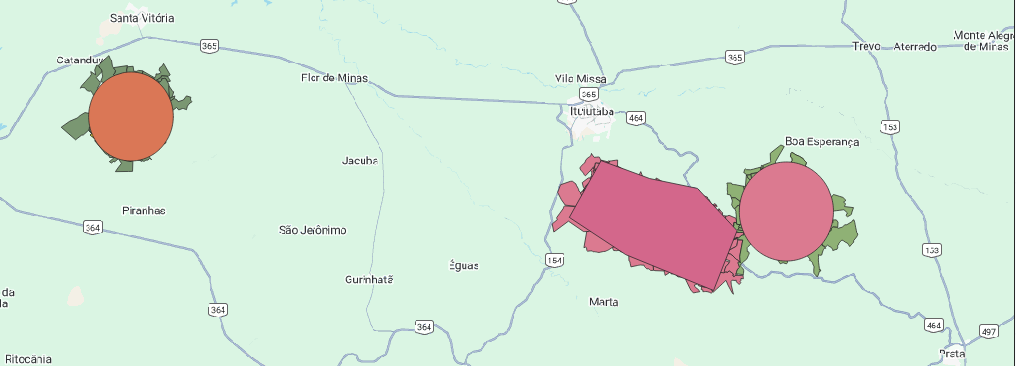
Os dados iniciais que farão parte das tabelas consistem nos dados dos imóveis, como citados acima, mas além deles, os dados das camadas citadas no TDR, deverão fazer parte dessa carga inicial no banco de dados.

Para deixar claro o que são as camadas, elas consistem em informações que são organizadas de forma visual e georreferenciadas a fim de combinar com outras informações resultando em análises mais profundas e completas sobre os dados que estão sendo analisados. Nas imagens abaixo temos um exemplo de como podemos perceber o uso de camadas para facilitar essa análise de dados.

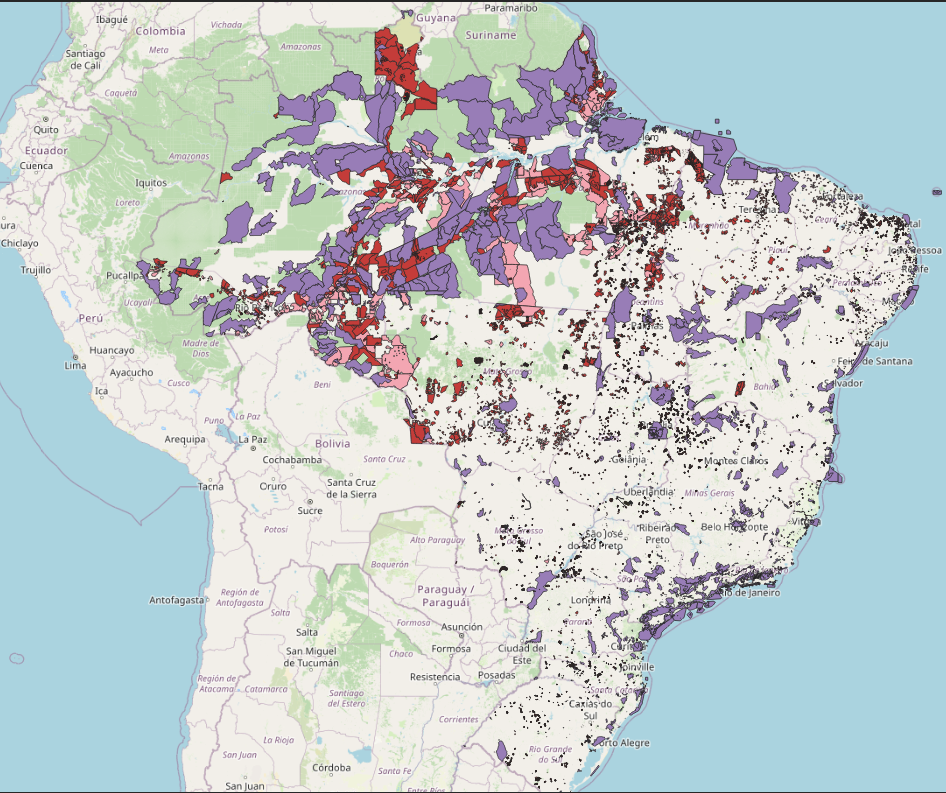
## (Imagem 4 - Imóveis apenas com suas delimitações sem contexto)



## (Imagem 5 - Imóveis com a camada contendo as delimitações e posições de cidades e rodovias fornecidas pelo google maps)



## (Imagem 6 - Imóveis com a camada de limite delimitando os limites da região estudadas nas cidades de São Lourenço do Leste(círculo localizada a direita da imagem), Invernada do Sul (círculo localizada a esquerda da imagem) e Área Central do Estado (retângulo ao centro) )



## (Imagem 7 - Representa todo o território Brasileiro com todas as camadas aplicadas ao mesmo tempo

Além da navegação e da organização dos dados, o uso de camadas integrará funcionalidades avançadas de análise de dados. Os usuários poderão combinar diferentes conjuntos de dados para realizar análises complexas, permitindo uma compreensão mais profunda das informações geoespaciais disponíveis. Essa capacidade de análise é fundamental para maximizar o valor dos dados coletados e fornecer insights significativos que podem informar a tomada de decisões.

Com base no termo de referência inicial, e considerando os desafios de articular, convencer, implementar e gerar resultados que promovam transformações no território e apoiem de forma efetiva, algumas camadas e informações foram sugeridas para que já venham nos modelos iniciais do projeto. As camadas serão fornecidas pela contratante e inseridas no sistema. São elas:

* Bacias Hidrográficas de Atuação do Projeto;
* Áreas prioritárias de intervenção da GIP;
* Áreas iniciais de atuação da GIP (Córrego São Lourenço, Área Central e Córrego da Invernada;
* Propriedades Rurais (CAR) com destaque para as áreas iniciais de intervenção da GIP
* Mapas de conectividade da paisagem;
* Mapas da Fragilidade Ambiental;
* Mapas temáticos utilizados na construção do Mapa da Fragilidade Ambiental;
* Mapas temáticos resultantes das análises realizadas pelo Projeto FIP Paisagens Rurais;
* Mapas das áreas de intervenção fornecidas pelos parceiros potenciais – iniciativa privada, ONGs, universidades etc.
* Limites Político-Administrativos - Estados e Municípios IBGE, sedes municipais e outros dados do IBGE
* Camadas do INCRA – exemplo: Assentamentos Rurais
* Imóveis Privados Cadastrados no Sigef
* Imóveis Privados Cadastrados no SNCI
* Imóveis Públicos Cadastrados no Sigef
* Imóveis Públicos Cadastrados no SNCI
* Limites oficiais Unidades de Conservação (Federais, Estaduais e Municipais)
* Corredor Ecológico proposto no âmbito do Projeto para Implementação da GIP
* Camada CAR
* Camada TAUS
* Sistema Hidrográfico e Bacias e sub-bacias Hidrográficas DNAEE
* Sistema Viário (Rodovias Federais, Estaduais e Municipais - rodovia pavimentada, em pavimentação, sem pavimentação) DNIT/IBGE/SEINFRA/AGETRANA
* Portos e Aeroportos DNIT
* Dados TerraClass - INPE
* Desmatamento – Prodes 2022 / MAPBiomas
* Linhas de Transmissão/ PCHs/ usinas hidrelétricas – ANEEL Vulnerabilidade à Inundações (fonte ANA)
* Limites dos Biomas (IBGE)
* Processos sobre conflitos fundiários (sirenejud)
* Imóveis rurais atendidos pelos Projetos
* Resultados das UAIS
* Modelagens Espaciais do INPE para a Área
* Produtos do Senar e Brasplan
* Outras camadas poderão ser solicitadas pelo SFB e GIZ.

Essas camadas de imóveis foram selecionadas para integrar a carga inicial do banco de dados da GIP e estão sob monitoramento contínuo. O acompanhamento constante visa executar análises detalhadas e promover melhorias na região. O monitoramento regular permite a coleta de dados precisos e atualizados, essenciais para avaliar o desenvolvimento e identificar áreas que necessitam de intervenção, como demonstrado na **Imagem 7**.

Além disso, essa vigilância sistemática facilita a implementação de estratégias de gestão e conservação, garantindo que os imóveis sejam mantidos em conformidade com as normas e padrões estabelecidos. As informações coletadas durante o monitoramento são analisadas para identificar tendências e padrões, possibilitando a tomada de decisões informadas e a elaboração de planos de ação eficazes.

O processo de monitoramento também envolve a atualização periódica dos dados no sistema da GIP, assegurando que todas as informações estejam sempre precisas e atualizadas. Dessa forma, é possível realizar um acompanhamento detalhado e contínuo das propriedades, promovendo a eficiência na gestão dos imóveis e contribuindo para o desenvolvimento sustentável da região. O objetivo final é utilizar os dados coletados para aprimorar a qualidade de vida dos habitantes e garantir a preservação do meio ambiente local, através de uma gestão integrada e bem-informada.

As camadas serão apresentadas em formato de mapa, permitindo ao usuário acessar menus exclusivos para consulta e cruzamento de dados. Essa funcionalidade facilitará a geração de novas análises. Além disso, os dados poderão ser exportados conforme a necessidade, garantindo flexibilidade no uso das informações. Cada camada georreferenciada contribuirá para uma visualização detalhada e integrada, aprimorando a capacidade de interpretação dos dados espaciais.

Brasília - DF, 01 de Agosto de 2024

Atenciosamente,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Heygon de Andrade do Lago M42 – Tecnologias WEB