Documentação de Requisitos - Projeto de Cidade Inteligente (Versão Revisada)

1. Introdução:

Este documento descreve os requisitos funcionais e não funcionais para o sistema de gerenciamento de dados da plataforma de Cidade Inteligente. A solução utiliza uma arquitetura de banco de dados híbrida, com **PostgreSQL** para dados estruturados e transacionais (conforme o esquema fornecido) e **MongoDB** para dados não estruturados, de alta volumetria e em tempo real.

O sistema será o núcleo para monitoramento e gestão de serviços urbanos, integrando dados cadastrais com dados dinâmicos de sensores e eventos.

2. Definições Conceituais:

- **2.1 Requisitos Funcionais:** São as funcionalidades que o sistema deve obrigatoriamente realizar.
- **2.2 Requisitos Não Funcionais:** São as características de qualidade, restrições e condições de operação do sistema.

3. Escopo do Projeto:

A plataforma deverá armazenar e processar:

- **No PostgreSQL (Relacional):**
    - **Cadastros Geográficos:** Gerenciamento de `bairro` e `rua`.
    - **Inventário de Infraestrutura:** Cadastro e status de `camera` e `sensor`, incluindo *`tipo_sensor`* e *`localizacao`* precisa.
    - **Gestão de Ocorrências:** Ciclo de vida completo de uma *`ocorrencia`*, desde a abertura até a resolução, incluindo nível de `criticidade ocorrência` e *`status_ocorrencia`*.
    - **Gestão de Serviços:** Cadastro de *`categoria_servico`*, *`tipo_servico`* e o registro de cada *`servico_executado`*.
    - **Auditoria e Relacionamentos:** Associação entre ocorrências e serviços (`ocorrencia_servico`) e log de auditoria para mudança de status (`log_status_ocorrencia`).
- **No MongoDB (Não Relacional):**
    - **Leituras de Sensores:** Armazenamento de dados de alta frequência da tabela `leitura_sensor`. Cada leitura será um documento no MongoDB para otimizar a performance de escrita e consulta de séries temporais. Cada documento de

leitura no MongoDB conterá o `id_sensor` correspondente, servindo como uma "chave estrangeira" para a tabela `sensor` no PostgreSQL.
    - **Logs de Eventos do Sistema:** Logs detalhados de operação da plataforma, interações de API e "heartbeats" dos dispositivos IoT.
    - **Anexos e Mídias:** Armazenamento de dados não estruturados como fotos e vídeos enviados por cidadãos e associados a uma `ocorrencia`.

4. Requisitos Funcionais (RF):

- **4.1 Gestão de Acessos e Usuários:**
    - **RF01:** O sistema deve permitir o cadastro de novos `usuario`, com nome, email e senha criptografada.
    - **RF02:** O sistema deve diferenciar as permissões de acesso com base no `tipo_usuario` (cidadao, operador, administrador).
- **4.2 Gestão de Cadastros e Infraestrutura:**
    - **RF03:** O sistema deve permitir que administradores cadastrem e gerenciem `bairro`, `rua`, `localizacao`, `tipo_sensor` e `categoria_servico`.
    - **RF04:** O sistema deve permitir o registro de novos dispositivos de `sensor` e `camera`, associando-os a uma `localizacao` e definindo seu `status`.
- **4.3 Gestão de Ocorrências e Serviços:**
    - **RF05:** Um `usuario` do tipo `cidadao` deve poder registrar uma nova `ocorrencia`, fornecendo título, descrição e localização.
    - **RF06:** O sistema deve permitir que uma `ocorrencia` seja gerada automaticamente a partir de dados de um `sensor` ou `camera`.
    - **RF07:** Um `usuario` do tipo `operador` deve poder alterar o `status` de uma `ocorrencia` (de 'aberta' para 'em_analise' ou 'resolvida').
    - **RF08:** O sistema deve registrar automaticamente no `log_status_ocorrencia` toda mudança de status de uma ocorrência, conforme a trigger `trg_ocorrencia_status_update`.
    - **RF09:** O sistema deve atualizar a `data_ultima_modificacao` de uma `ocorrencia` sempre que ela for alterada, conforme a trigger `trg_atualizar_timestamp_ocorrencia`.
    - **RF10:** Operadores devem poder associar um ou mais `servico_executado` a uma `ocorrencia` através da tabela `ocorrencia_servico`.
- **4.4 Ingestão e Processamento de Dados:**
    - **RF11:** O sistema deve receber os dados da `leitura do sensor` dos dispositivos IoT e armazená-los como documentos no MongoDB.
    - **RF12:** O sistema deve permitir o upload de arquivos de mídia (fotos/vídeos) que serão associados a uma `id_ocorrencia` e armazenados no MongoDB (ou em um object storage com referência no MongoDB).
- **4.5 Consultas e Relatórios:**

- **RF13:** O sistema deve gerar relatórios a partir do PostgreSQL sobre a quantidade de ocorrências por bairro, por status ou por nível de criticidade.
- **RF14:** O sistema deve permitir consultas híbridas: ao visualizar uma `ocorrencia` no PostgreSQL, o sistema deve buscar e exibir o histórico de leituras do `sensor` associado que está armazenado no MongoDB.

5. Requisitos Não Funcionais (RNF):

- **5.1 Desempenho:**
  - **RNF01:** Consultas à tabela `ocorrência` devem ter um tempo de resposta máximo de 5000ms.
  - **RNF02:** O MongoDB deve suportar a inserção de, no mínimo, 2000 leituras de sensor por segundo.
- **5.2 Segurança:**
  - **RNF04:** A coluna `senha` da tabela `usuário` deve ser armazenada usando um algoritmo de hash forte (ex: Bcrypt).
  - **RNF05:** A comunicação entre os dispositivos IoT e o sistema deve ser feita através de protocolos seguros (ex: MQTT com TLS).
  - **RNF06:** O sistema deve estar em total conformidade com a Lei Geral de Proteção de Dados (LGPD), especialmente para os dados da tabela `usuário`.
- **5.3 Disponibilidade e Escalabilidade:**
  - **RNF07:** A plataforma deve garantir uma disponibilidade mínima de 99.8%.
  - **RNF08:** A arquitetura do MongoDB deve ser horizontalmente escalável para suportar o aumento do número de sensores.
  - **RNF09:** O PostgreSQL deve operar em um esquema de replicação para garantir alta disponibilidade.
- **5.4 Auditabilidade:**
  - **RNF10:** O sistema deve manter um log auditável de todas as alterações de status nas ocorrências, registrando o status anterior, o novo, o usuário e a data da alteração na tabela `log status ocorrência`.
- **5.5 Portabilidade:**
  - **RNF11:** Toda a aplicação e os bancos de dados devem ser mantidos usando de uma VPS.