**FASE 1: LEITOR DE QR CODE PARA CHAVES DE ACESSO**

**Projeto:** Mercado em Números: Seu Primeiro Dashboard Profissional - Fase 1 **Componente Curricular:** Business Intelligence **Curso:** Superior de Inteligência Artificial

**Turma:** 2º Período **Docente:** Ms. Ujeverson Tavares Sampaio **Data:** 14/10/2025 (Simulação)

**Grupo:** Frederico Lemes Rosa e Maria Clara Bragança

**1. Descrição Detalhada das Funcionalidades Implementadas**

O projeto implementa uma aplicação web robusta em Python para a **captura automatizada da Chave de Acesso (44 dígitos) de cupons fiscais**, cumprindo integralmente os requisitos da Fase 1.

| Requisito do Projeto | Funcionalidade Implementada |
| --- | --- |
|  | **Ler o QR Code** de um cupom fiscal e extrair apenas a Chave de Acesso. | A aplicação oferece duas abas de leitura: **Câmera em Tempo Real** (via streamlit-webrtc) e **Upload de Imagem** (para análise de fotos estáticas). O módulo extrair\_chave() garante que apenas a chave de 44 dígitos seja isolada do link completo do QR Code. |
|  | **Interface web intuitiva.** | Utilização do Streamlit para criar uma interface limpa, com feedback visual (cores e mensagens de sucesso/erro) em tempo real, travamento da câmera após detecção para evitar múltiplas leituras e abas de navegação claras. |
|  | **Armazenar a chave de acesso** (em .txt, .csv, .xlsx ou BD Relacional). | O armazenamento é feito em um **Arquivo .csv (chaves.csv)**, que é criado e atualizado automaticamente. A coluna Chave é forçada ao tipo string (dtype=str) no *Pandas* para garantir a preservação dos 44 dígitos sem arredondamentos ou notação científica. |
|  | **Evitar duplicidades** e exibir um alerta: "Cupom já lido". | A função salvar\_dados() verifica a existência da chave no arquivo .csv antes de adicioná-la. Em caso de duplicidade, a adição é impedida e um alerta visual é acionado na interface. |
|  | **Entrada por leitura direta do QR Code e/ou upload de imagem.** | Ambas as funcionalidades foram implementadas com algoritmos de **Visão Computacional** para aumentar a taxa de sucesso da leitura (abordagem de **Força Bruta/Inteligência**). |

**2. Justificativa da Escolha das Bibliotecas e Ferramentas**

| Ferramenta | Justificativa de Escolha |
| --- | --- |
| **Python** | Linguagem central para o Curso de IA e a linguagem demandada para construção do Dashboard/BI. |
| **Streamlit** | Escolhido por ser **"mais simples e direto"** , permitindo a criação rápida de uma interface web funcional (Interface web intuitiva ) e um ótimo protótipo de BI. |
| **OpenCV (cv2)** e **NumPy (np)** | Essenciais para implementar a lógica de **Visão Computacional e IA**, tanto para o pré-processamento de imagens estáticas (processar\_imagem) quanto para a detecção em tempo real (QRReader), garantindo a leitura de QR Codes de baixa qualidade. |
| **PyZBar (pyzbar)** | Biblioteca especializada em decodificação de códigos de barra e QR Code, servindo como o **núcleo do processo de leitura** e *fallback* nos algoritmos de detecção. |
| **Streamlit-WebRTC** | Necessário para acessar e processar o *feed* da câmera/webcam diretamente no navegador, permitindo a **leitura em tempo real** do QR Code do cupom fiscal. |
| **Pandas (pd)** | Fundamental para a **Manipulação e Armazenamento Estruturado dos Dados** no formato .csv, garantindo o controle de tipos de dados (44 dígitos como string) e a checagem de duplicidades. |

**3. Explicação da Estrutura de Dados e das Decisões de Modelagem**

**Estrutura de Dados:** A estrutura de dados para o armazenamento das chaves de acesso é um **arquivo CSV (chaves.csv)**, que simula um banco de dados simples para a Fase 1. A tabela possui uma única coluna essencial:

* **Coluna:** Chave
* **Tipo de Dado:** String (Forçada via dtype=str no Pandas)
* **Decisão de Modelagem:** O tipo String é crucial. A Chave de Acesso possui 44 dígitos. Se fosse armazenada como um tipo numérico (e.g., int64), haveria truncamento e perda de precisão, inviabilizando a identificação única do documento fiscal.

**Decisões de Modelagem (Lógica de Detecção Inteligente):**

1. **Extração de Chave (Regex):** A função extrair\_chave(texto) usa expressões regulares e *split* para isolar a chave de 44 dígitos, mesmo que o QR Code retorne a URL completa (exemplo de link ).
2. **Abordagem de Força Bruta (Upload):** O leitor estático (ler\_qr\_code) aplica mais de 100 variações de pré-processamento (filtros, rotações e escalas) para **aumentar a assertividade** na leitura de imagens com problemas de foco ou iluminação, simulando a robustez de um sistema profissional.
3. **Controle de Fluxo em Tempo Real:** A classe QRReader introduz um sistema de **"trava" (qr\_lock\_success)** com um contador. Após uma leitura bem-sucedida, o sistema é pausado por alguns segundos para: a) evitar a leitura múltipla do mesmo QR Code no mesmo frame; b) dar tempo para o usuário apresentar o próximo cupom, cumprindo o requisito de *Controle para impedir leitura duplicada*.