# Funcionalidade: Captura e Identificação de Veículos

**História de Usuário**: Como usuário, quero que o sistema seja capaz de capturar o fluxo de veículos que passam por um ponto específico para que eu possa monitorar o tráfego em tempo real.

**Tarefas**:

**1-** **Configuração do Hardware**: Configurar a câmera e conectá-la ao Arduino para capturar imagens de uma via pública.

**2-Desenvolvimento de Software**: Criar um script em Python utilizando a biblioteca OpenCV para:

Processar imagens em tempo real.

Identificar e contar veículos na via.

**3-Testes e Validação**:

Realizar testes de identificação de veículos para garantir precisão.

Garantir que os dados processados não contenham informações pessoais ou identificáveis dos veículos, mantendo o foco apenas na contagem total.

#### **Critérios de Aceitação:**

**1-Precisão de Identificação**: O sistema deve identificar veículos com uma margem de erro mínima (definir o percentual aceitável).

**2-Privacidade dos Dados**:

Não armazenar dados pessoais ou identificadores específicos dos veículos (ex.: placas).

Armazenar apenas o número total de veículos identificados.

**3-Disponibilidade no Sistema**: A funcionalidade deve ser acessível para os atores designados, permitindo que visualizem:

A contagem de veículos em tempo real.

Relatórios com os dados agregados do fluxo em diferentes horários.

* **Prioridade** : 1
* **Duração** : 7 dias

# Funcionalidade: Registro de Fluxo em Diferentes Horários

**História de Usuário**: O usuário precisa registrar no sistema o fluxo de veículos em horários variados do dia.

**Tarefas**:

**1-Implementação do Armazenamento de Dados**: Configurar um banco de dados MongoDB para armazenar os dados de contagem de veículos.

Estruturar o banco de dados para incluir:

Data e hora exatas da contagem.

Total de veículos detectados no período.

**2-Desenvolvimento da Persistência de Dados**:

Garantir que o sistema armazene os dados de forma contínua, sem interrupções ou duplicações.

Implementar um mecanismo de verificação para evitar a inserção repetida de dados.

**3-Validação e Testes**:

Testar o fluxo contínuo de dados para garantir que não haja falhas ou perdas durante a captura e armazenamento.

Simular diferentes horários do dia para validar a consistência e precisão dos registros.

#### **Critérios de Aceitação:**

**1-Precisão Temporal**: Os dados de contagem devem ser armazenados com a data e o horário exatos em que foram capturados.

**2-Dados Agregados Recuperáveis**: O sistema deve permitir a recuperação fácil e rápida dos dados agregados (ex.: contagem total por hora, dia ou intervalo personalizado).

**3-Integridade dos Dados**: Os registros devem ser livres de duplicações e inconsistências.

* **Prioridade** : 3
* **Duração** : 4 dias

# Funcionalidade: Exibição Intuitiva de Dados

**História de Usuário**: O usuário deve acessar a página de gráficos no sistema, selecionar os filtros disponíveis, como intervalos de datas e horários, e visualizar os dados ajustados diretamente nos gráficos. Ele pode interagir com os gráficos para explorar informações detalhadas e exportar os resultados, caso necessário.

**Tarefas**:

**1-Desenvolvimento da Página de Exibição**:

Criar uma interface web intuitiva e responsiva para exibição de gráficos dinâmicos.

tilizar bibliotecas como **Plotly** ou **Matplotlib** em Python para a construção dos gráficos.

**2-Implementação de Gráficos Dinâmicos**:

Integrar os gráficos com o banco de dados MongoDB para exibição em tempo real.

Garantir que os gráficos sejam claros e personalizáveis.

**3-Desenvolvimento de Filtros**:

Criar filtros para permitir que o usuário selecione intervalos de datas e horários.

Garantir a funcionalidade de atualização automática dos gráficos ao aplicar os filtros.

**4-Testes e Ajustes**:

Testar a usabilidade dos filtros para garantir que são intuitivos e eficazes.

Validar a precisão e a clareza dos gráficos em diferentes cenários de uso.

**Critérios de Aceitação**:

**1-Filtros Funcionais**:

A página deve permitir ao usuário filtrar os dados exibidos com base em intervalos de datas e horários.

**2-Clareza e Intuitividade**:

Os gráficos devem ser fáceis de interpretar, com legendas claras, títulos e eixos devidamente rotulados.

Não deve haver dúvidas ou ambiguidades sobre os valores mostrados nos gráficos.

**3-Atualização em Tempo Real**:

A aplicação dos filtros deve atualizar os gráficos de forma imediata e sem erros.

* **Prioridade** : 2
* **Duração** : 10 dias

# Funcionalidade: Análise de Dados e Histórico

**História de Usuário**: O usuário deve acessar a seção de análise de dados no sistema, aplicar os filtros disponíveis para selecionar períodos específicos e visualizar os gráficos com os dados históricos de fluxo de veículos. Ele pode explorar os padrões de tráfego apresentados e utilizar as informações para embasar decisões de planejamento urbano.

**Tarefas**:

**1-Recuperação de Dados Históricos**:

Implementar a funcionalidade para puxar dados históricos armazenados no s sistema.

**2-Desenvolvimento de Filtros**:

Criar filtros que permitam a seleção de períodos específicos, como datas e intervalos de tempo, para análise detalhada.

**3-Exibição de Gráficos**:

Desenvolver gráficos claros e intuitivos que apresentem os dados históricos selecionados pelo usuário.

**4-Testes e Validação**:

Testar a exibição dos dados históricos em diferentes cenários para garantir precisão e usabilidade.

Validar a funcionalidade dos filtros para assegurar que os dados exibidos correspondem aos parâmetros definidos pelo usuário.

**Critérios de Aceitação**:

**1-Recuperação de Dados**:

O sistema deve permitir a recuperação de dados históricos completos e consistentes.

**2-Filtros Funcionais**:

O usuário deve ser capaz de aplicar filtros para especificar os períodos de análise.

**3-Visualização Clara**:

Os gráficos devem exibir os dados históricos de forma clara e ajustada aos filtros aplicados.

* + Exibir gráfico com dados histórico especificados pelo usuário
* **Prioridade** : 4
* **Duração** : 4 dias