Universidade Tecnológica Federal do Paraná – UTFPR
Departamento Acadêmico de Eletrônica – DAELN
ELE64 Oficina de Integração

IdentifierFlow

Ramon M. P. Mariano
reversmon@gmail.com (41) 99796-3433
Alejandro Klein
alejandroklein@alunos.utfpr.edu.br (41) 99692-4882
Talita Ribeiro Santos
tsantos@alunos.utfpr.edu.br (41)99935-3627

11/2024

O projeto IdentifierFlow visa criar um sistema de controle de acesso por reconhecimento facial, aplicável em ambientes que demandam segurança e eficiência no gerenciamento de entradas e saídas, como residências, empresas, laboratórios, órgãos públicos e academias.

Curitiba
2024

Conteúdo

1 Introdução

O **IdentifierFlow** é um sistema de controle de acesso que utiliza *reconhecimento facial* para aumentar a segurança e eficiência em diversos tipos de ambientes que demandam controle rigoroso de entradas e saídas, incluindo não apenas residências, mas também empresas, laboratórios, órgãos públicos e academias.

Esse sistema inclui:

- Um **aplicativo móvel** para captura de imagens faciais e comunicação com o servidor;
- Um **servidor de processamento de imagens** para realizar o reconhecimento facial e verificar a identidade dos usuários;
- Uma **interface web** para administradores gerenciarem acessos, cadastrar usuários, e visualizar registros;
- **Integração com uma catraca controlada por um ESP32**, para efetuar o bloqueio ou liberação de acesso com base nas autorizações recebidas do servidor.

O IdentifierFlow, portanto, oferece uma **solução versátil** para diversas aplicações, fornecendo um controle de acesso seguro e em tempo real para diferentes tipos de ambientes.

Para mais detalhes, acesse o projeto completo no Notion: IdentifierFlow - Oficinas 2.
Futuramente disponivel em: www.IdenfierFlow.com.

2 Escopo e Requisitos

2.1 Escopo de Alto Nível

- **Objetivo do Projeto**: Desenvolver um sistema de automação residencial com controle de acesso via reconhecimento facial, visando proporcionar maior segurança e conveniência para os moradores.
- **Problema a Ser Solucionado**: Em um ambiente residencial, a segurança é uma prioridade. Muitas vezes, os sistemas de segurança tradicionais não oferecem um controle de acesso preciso e em tempo real. O sistema IdentifierFlow visa resolver este problema ao permitir o acesso apenas a pessoas autorizadas, utilizando reconhecimento facial.
- **Escopo do Projeto**:
  - Desenvolver um aplicativo móvel em Flutter para capturar imagens faciais e comunicar-se com o servidor.
  - Criar um sistema web em Django para cadastro, gerenciamento de acessos e visualização de registros.
  - Implementar um sistema de processamento de imagens em Flask utilizando TensorFlow e YOLO para autenticação facial em tempo real.
  - Integrar o sistema com uma catraca física, controlada via ESP32, para permitir ou restringir o acesso físico.
  - Documentar o sistema com informações técnicas e manuais de operação.
- **Delimitações do Escopo**:
  - **Inclusões**:
    - Implementação de banco de dados para armazenamento seguro dos dados de moradores.
    - Configuração de containers Docker para modularidade e escalabilidade do sistema.
    - Autenticação via reconhecimento facial para controle de acesso.
  - **Exclusões**:
    - Suporte contínuo para manutenção após a instalação inicial.
    - Integração com sistemas de segurança externos ao edifício.
- **Diagrama em Blocos**: O diagrama a seguir apresenta uma visão geral do sistema proposto, incluindo seus componentes principais e fluxos de dados.

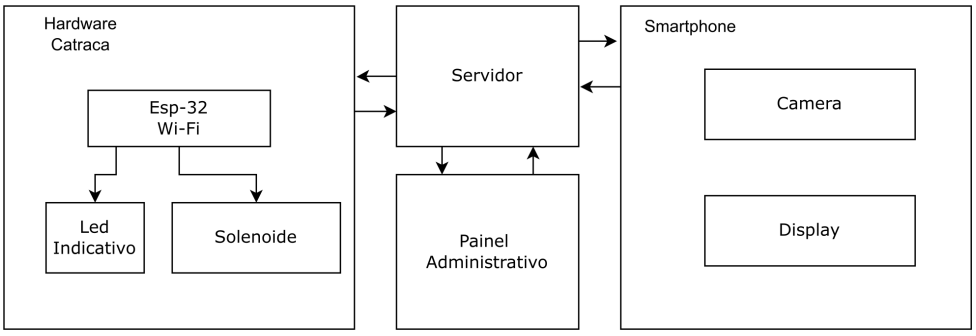

Figura 1: Diagrama de Blocos do Sistema IdentifierFlow

2.2 Requisitos Funcionais e Não Funcionais do Projeto IdentifierFlow

2.2.1 Sistema Flutter: App de Reconhecimento

Requisitos Funcionais

- **[RF-FLUTTER-001]**: O aplicativo deve capturar imagens em tempo real da câmera do dispositivo e transmiti-las para o servidor para processamento de reconhecimento facial.
- **[RF-FLUTTER-002]**: O aplicativo deve exibir o retorno do servidor com a imagem processada, indicando se o acesso foi aprovado (quadrado verde) ou negado (quadrado vermelho).
- **[RF-FLUTTER-003]**: O aplicativo deve permitir que o morador visualize seu histórico de acessos, incluindo data, hora e status de cada tentativa.
- **[RF-FLUTTER-004]**: O aplicativo deve enviar notificações ao morador sobre tentativas de acesso autorizadas ou não autorizadas em tempo real.

Requisitos Não Funcionais

- **[RNF-FLUTTER-001]**: O aplicativo deve ser desenvolvido em Flutter para garantir compatibilidade com dispositivos Android e iOS.
- **[RNF-FLUTTER-002]**: A comunicação entre o aplicativo e o servidor deve ser realizada através de uma conexão segura (UDP) para proteger as imagens e dados dos usuários.
- **[RNF-FLUTTER-003]**: O sistema de captura e transmissão de imagens deve funcionar com baixa latência para permitir o processamento em tempo real e resposta rápida ao usuário.
- **[RNF-FLUTTER-004]**: O aplicativo deve ser responsivo e se adaptar a diferentes tamanhos de tela de dispositivos móveis.

2.2.2 Sistema de Processamento de Imagens

Requisitos Funcionais

- **[RF-FLASK-001]**: O sistema de processamento de imagens deve receber as imagens capturadas pelo aplicativo e enviadas pelo servidor para realizar a análise de reconhecimento facial.
- **[RF-FLASK-002]**: O sistema deve processar as imagens usando modelos de reconhecimento facial treinados com TensorFlow e YOLO para identificar se o acesso deve ser autorizado ou negado.
- **[RF-FLASK-003]**: O sistema deve retornar ao servidor o status da análise, incluindo uma imagem processada com um quadrado verde (acesso aprovado) ou vermelho (acesso negado).
- **[RF-FLASK-004]**: O sistema deve registrar localmente os logs de cada tentativa de processamento para auditoria e monitoramento.

Requisitos Não Funcionais

- **[RNF-FLASK-001]**: O sistema de processamento de imagens deve ser implementado em Flask para fornecer uma API REST que permita comunicação com o sistema web.
- **[RNF-FLASK-002]**: O sistema deve usar TensorFlow e YOLO para garantir um reconhecimento facial eficiente e em tempo real.

- **[RNF-FLASK-003]**: A resposta do processamento de imagem deve ser enviada em menos de 1 segundo para garantir baixa latência no sistema de reconhecimento.
- **[RNF-FLASK-004]**: A comunicação entre o sistema de processamento de imagens e o servidor Django deve ser realizada em uma rede segura, utilizando HTTPS para proteção de dados sensíveis.
- **[RNF-FLASK-005]**: O sistema de processamento de imagens deve ser escalável para suportar múltiplas requisições simultâneas sem comprometer o desempenho.

2.2.3 Sistema Web (Django)

Requisitos Funcionais

- **[RF-DJANGO-001]**: O sistema web deve permitir o cadastro de novos moradores, incluindo informações como nome, e-mail, número de telefone, andar e apartamento.
- **[RF-DJANGO-002]**: O sistema web deve permitir que o administrador visualize e filtre os logs de acessos para monitoramento de entradas e saídas dos moradores.
- **[RF-DJANGO-003]**: O sistema web deve permitir que o administrador visualize o histórico de acessos, incluindo data, hora de entrada e status (aprovado/negado).
- **[RF-DJANGO-004]**: O sistema web deve possibilitar a configuração de permissões e horários de acesso dos moradores.
- **[RF-DJANGO-005]**: O sistema web deve comunicar-se com o serviço de processamento de imagens para verificar e armazenar o status das tentativas de acesso.

Requisitos Não Funcionais

- **[RNF-DJANGO-001]**: O sistema web deve ser implementado em Django para proporcionar uma interface segura e de fácil uso.
- **[RNF-DJANGO-002]**: A comunicação com o serviço de processamento de imagens deve ser realizada via API REST para garantir uma integração eficiente e segura.
- **[RNF-DJANGO-003]**: O sistema web deve ser seguro, com autenticação de usuários e autorização para acesso a dados sensíveis.
- **[RNF-DJANGO-004]**: O sistema web deve ter uma interface intuitiva para facilitar o gerenciamento por parte dos administradores.
- **[RNF-DJANGO-005]**: A transmissão de dados sensíveis entre o sistema web e outros serviços deve ser feita usando HTTPS para garantir segurança.

2.2.4 Configuração do Servidor com Docker

Para facilitar a configuração e o gerenciamento do ambiente de execução, o servidor utilizará o Docker para hospedar os serviços em containers. Cada aplicação (sistema web em Django, sistema de processamento de imagens em Flask com TensorFlow e YOLO) será executada em um container separado, permitindo maior modularidade e escalabilidade.

Requisitos Funcionais

- **[RF-DOCKER-001]**: O servidor deve utilizar Docker para orquestrar os containers das diferentes aplicações.
- **[RF-DOCKER-002]**: Cada aplicação (Django, Flask com TensorFlow e YOLO) deve ser executada em um container separado, permitindo fácil manutenção e escalabilidade.

- **[RF-DOCKER-003]**: O sistema deve permitir que os containers se comuniquem através de uma rede interna do Docker, mantendo a segurança dos dados.

Requisitos Não Funcionais

- **[RNF-DOCKER-001]**: A utilização de Docker deve garantir que o ambiente seja facilmente reproduzível em outros servidores.
- **[RNF-DOCKER-002]**: A configuração dos containers deve seguir as melhores práticas de segurança para evitar acessos não autorizados.
- **[RNF-DOCKER-003]**: A estrutura com containers deve suportar escalabilidade horizontal, permitindo o aumento de instâncias de cada serviço conforme a demanda.

2.2.5 Sistema da Catraca (C++ com ESP32)

Requisitos Funcionais

- **[RF-CATRACA-001]**: A catraca deve receber o sinal do servidor para liberar a passagem do usuário após o reconhecimento facial.
- **[RF-CATRACA-002]**: A catraca, implementada em C++ com ESP32, deve registrar localmente os logs de acesso, incluindo o nome do usuário e o horário de liberação ou bloqueio.
- **[RF-CATRACA-003]**: A catraca deve enviar os logs de acesso periodicamente ao servidor para sincronização.

Requisitos Não Funcionais

- **[RNF-CATRACA-001]**: O sistema da catraca deve ser implementado em C++ para garantir um desempenho robusto e rápido no controle de acesso.
- **[RNF-CATRACA-002]**: A comunicação entre a catraca e o servidor deve ser estável e segura, utilizando ESP32 para integração com o servidor.
- **[RNF-CATRACA-003]**: A catraca deve ter capacidade de registrar logs localmente para redundância e segurança dos dados de acesso, mesmo em caso de falha de comunicação com o servidor.
- **[RNF-CATRACA-004]**: O sistema deve ser tolerante a falhas, assegurando que, em caso de perda de conexão com o servidor, os logs sejam mantidos localmente até o restabelecimento da conexão.

3 Análise de Riscos

Para o projeto IdentifierFlow, identificamos riscos técnicos que afetam diretamente o funcionamento e a segurança do sistema. Usando a metodologia GUT (Gravidade, Urgência e Tendência), avaliamos cada risco para entender sua prioridade e definimos estratégias de mitigação. A Tabela 2 mostra esses riscos classificados e suas respectivas ações de mitigação, evidenciando os pontos que precisam de maior atenção para assegurar a eficiência e robustez do projeto.

Risco Técnico	Descrição	Gravidade (G)	Urgência (U)	Tendência (T)	Total (G x U x T)	Mitigação
Precisão do reconhecimento facial	O sistema pode não reconhecer corretamente o morador, resultando em falhas de autenticação.	5	5	4	100	Aperfeiçoar o treinamento do modelo de IA com mais dados e ajustar parâmetros de detecção para melhorar a precisão.
Segurança dos dados de imagem e autenticação	Dados sensíveis podem estar vulneráveis a interceptação durante a transmissão.	5	5	4	100	Implementar criptografia de ponta a ponta e protocolos seguros (HTTPS, SSL).
Delay no tempo de transmissão	Atrasos na transmissão das imagens para o servidor podem comprometer a experiência do usuário e a segurança.	4	4	4	64	Implementar otimização de rede e compressão de imagem para reduzir o tempo de transmissão.
Falhas de conectividade entre componentes	Problemas de conectividade entre o app, servidor e catraca podem impedir o funcionamento integrado do sistema.	5	4	3	60	Monitoramento contínuo da conexão e protocolos de reconexão automática.
Capacidade de processamento do servidor	O servidor pode ter dificuldade para processar múltiplas requisições simultâneas, afetando o tempo de resposta.	4	4	3	48	Utilizar containers Docker para escalabilidade horizontal e aumentar a capacidade conforme necessário.
Robustez do hardware da catraca	Falhas no hardware da catraca podem impedir a entrada/saída dos moradores.	4	3	3	36	Testar durabilidade do hardware e realizar manutenções preventivas.
Consumo de energia e autonomia do sistema	O alto consumo de energia pelo sistema pode resultar em problemas de operação contínua.	3	3	4	36	Usar componentes de baixo consumo e otimizar o sistema para economizar energia.
Compatibilidade entre diferentes dispositivos	Problemas de compatibilidade podem surgir ao conectar diferentes versões de hardware ou sistemas operacionais.	3	3	3	27	Testar o sistema em múltiplos dispositivos e versões para garantir compatibilidade.

Figura 2: Riscos do projeto

Os riscos com maior pontuação, como a precisão do reconhecimento facial e a segurança dos dados de imagem, exigem atenção prioritária. O reconhecimento facial precisa ser preciso para evitar acessos indevidos, o que implica no aperfeiçoamento contínuo do modelo de IA. A segurança dos dados também é crítica, pois envolve a proteção de informações sensíveis, demandando criptografia e protocolos seguros de comunicação.

Além disso, fatores como o delay no tempo de transmissão e falhas de conectividade entre os componentes (aplicativo, servidor e catraca) podem comprometer a experiência do usuário e a integridade do sistema. Para mitigar esses problemas, recomenda-se a implementação de otimizações de rede, técnicas de compressão de dados e monitoramento constante da conexão.

Por fim, ao avaliar e priorizar esses riscos técnicos, o projeto IdentifierFlow assegura uma abordagem estruturada para lidar com possíveis problemas operacionais, direcionando esforços para áreas vulneráveis e promovendo um ambiente seguro e confiável para usuários e administradores.

4 Integração

Para a confecção deste projeto, será utilizado conhecimento das seguintes matérias:

- Fundamentos de Programação 1
- Técnicas de Programação
- Introdução à prática de laboratório em eletricidade e eletrônica
- Química
- Eletricidade
- Circuitos Elétricos
- Introdução a Banco de Dados
- Eletrônica Geral 1
- Sistemas Microcontrolados
- Estrutura de Dados

5 Cronograma

O cronograma detalhado incluirá prazos definidos para cada uma das fases do projeto, abordando desde o desenvolvimento inicial até a implantação.

5.1 Cronograma Detalhado

A Tabela apresenta o cronograma detalhado das atividades do projeto, com datas de início e término para cada tarefa. Este cronograma foi extraído e preparado a partir do arquivo `Cronograma`.

Código	Descrição da Atividade	Responsável Principal	Backup/Auxílio	Status da Atividade	Data de Início	Data de Término	Horas Previstas	Horas Extras	Horas Trabalhadas
AT01	Configuração do ambiente Docker para containers Django, Flask e banco de dados	Ramon	Alejandro	A fazer	18/11/2024	23/11/2024	10	3	-
AT02	Projeto do PCB em KiCad para catraca	Talita	Ramon	A fazer	18/11/2024	23/11/2024	6	1.8	-
AT03	Desenvolvimento inicial da API REST em Flask para processamento de imagens	Alejandro	Ramon	A fazer	18/11/2024	23/11/2024	12	3.6	-
AT04	Configuração de comunicação entre o servidor Django e Flask	Ramon	Talita	A fazer	18/11/2024	23/11/2024	5	1.5	-
AT05	Desenvolvimento do sistema de reconhecimento facial com TensorFlow e YOLO	Alejandro	Talita	A fazer	25/11/2024	30/11/2024	14	4.2	-
AT06	Desenvolvimento da interface do app Flutter para captura de imagens e exibição de status	Talita	Alejandro	A fazer	25/11/2024	30/11/2024	8	2.4	-
AT07	Arquitetura do banco de dados PostGreeSQL	Ramon	Alejandro	A fazer	25/11/2024	30/11/2024	7	2.1	-
AT08	Testes iniciais de integração para garantir a comunicação entre Flask e Django	Ramon	Talita	A fazer	25/11/2024	30/11/2024	5	1.5	-
AT09	Integração do app Flutter com a API Flask para envio e recebimento de imagens	Talita	Ramon	A fazer	02/12/2024	07/12/2024	8	2.4	-
AT10	Implementação de Login no app Flutter para Administrador	Talita	Alejandro	A fazer	02/12/2024	07/12/2024	6	1.8	-
AT11	Desenvolvimento do de APIS em C++ com MQTT para liberar a catraca (ESP-32)	Alejandro	Ramon	A fazer	02/12/2024	07/12/2024	10	3	-
AT12	Implementação da lógica de validação e envio de logs do ESP32 para o servidor	Alejandro	Talita	A fazer	02/12/2024	07/12/2024	8	2.4	-
AT13	Teste e validação da integração completa entre Django, Flask, Flutter e ESP32	Todos	-	A fazer	09/12/2024	14/12/2024	10	3	-
AT14	Teste final do aplicativo Flutter e sistema de comunicação com ESP32	Talita	Alejandro	A fazer	09/12/2024	14/12/2024	5	1.5	-
AT15	Impressao em fenolite do modulo PCB e Montagem da placa	Ramon	Alejandro	A fazer	09/12/2024	14/12/2024	6	1.8	-
AT16	Otimização de latência para processamento de imagens no Flask	Alejandro	Talita	A fazer	09/12/2024	14/12/2024	7	2.1	-
AT17	Documentação do sistema e preparação para testes finais	Todos	-	A fazer	16/12/2024	21/12/2024	8	2.4	-
AT18	Últimos ajustes no reconhecimento facial e teste de precisão	Alejandro	Talita	A fazer	16/12/2024	21/12/2024	5	1.5	-
AT19	Finalização dos logs de acesso e validação no sistema da catraca	Talita	Alejandro	A fazer	23/12/2024	28/12/2024	7	2.1	-
AT20	Revisão de todo o sistema para identificar melhorias	Ramon	Todos	A fazer	23/12/2024	28/12/2024	6	1.8	-
AT21	Integração completa e testes de comunicação entre sistemas	Todos	-	A fazer	03/02/2025	07/02/2025	10	3	-
AT22	Preparação da documentação final para a apresentação	Ramon	Todos	A fazer	03/02/2025	07/02/2025	8	2.4	-
AT23	Testes de validação finais do sistema completo	Todos	-	A fazer	10/02/2025	14/02/2025	12	3.6	-
AT24	Otimizações finais e ajustes com foco em desempenho	Alejandro	Talita	A fazer	10/02/2025	14/02/2025	5	1.5	-
AT25	Revisão e preparação para a apresentação final	Todos	-	A fazer	17/02/2025	17/02/2025	4	1.2	-

Figura 3: Cronograma Detalhado do Projeto IdentifierFlow

5.2 Diagrama de Gantt

A Figura 4 exibe o diagrama de Gantt do projeto, ilustrando visualmente o cronograma das atividades, suas durações e dependências.

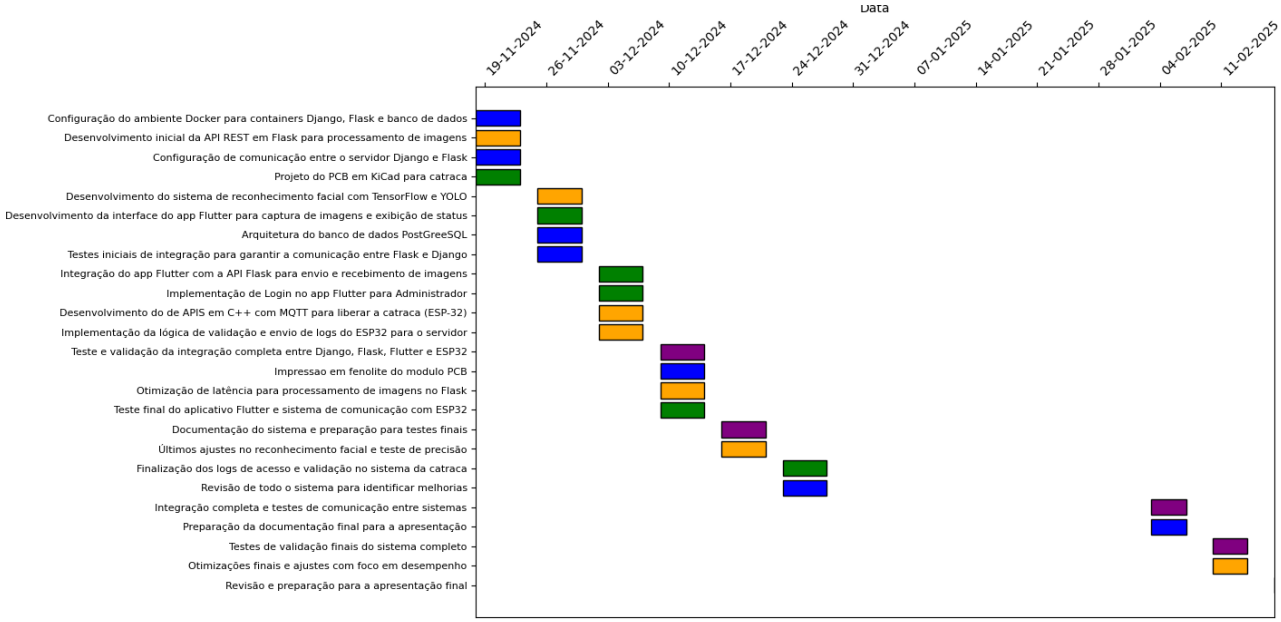

Figura 4: Diagrama de Gantt do Projeto IdentifierFlow

6 Materiais e Métodos

O sistema de controle de acesso utilizará uma catraca controlada por um microcontrolador **ESP32**, que será responsável por receber os comandos de liberação e bloqueio de acesso a partir do servidor central. O ESP32 é uma escolha ideal devido à sua capacidade de comunicação sem fio (Wi-Fi e Bluetooth), que permitirá a integração fácil com o servidor.

- **Microcontrolador ESP32**: Gerencia os sensores e atuadores da catraca e realiza a comunicação com o servidor através de uma rede WLAN segura.
- **Solenoide de Travamento**: Controla a trava da catraca, acionando o bloqueio ou a liberação de passagem com base nas autorizações recebidas do servidor.
- **Sensores de Presença e Contagem**: Detectam a passagem de pessoas, registrando cada entrada e saída para monitoramento em tempo real.
- **LED Indicativo**: Indica visualmente o status da catraca (verde para acesso autorizado e vermelho para acesso negado).
- **Fonte de Alimentação**: Fornece energia contínua e estável ao sistema da catraca.

Materiais	Quantidade	Custo Unitário (R$)	Custo Total (R$)
ESP32 WiFi	1	55,00	R$ 55.00
LED Indicador	2	1,09	R$ 2.18
Ácido para Corrosão	1 litro	20,00	R$ 20.00
Placa Fenolite Lisa	1	15,00	R$ 15.00
Pinos de Conexão	10	0,50	R$ 5.00
Smartphone	1	800,00	R$ 800.00
Servidor	1	1.500,00	R$ 1.500.00
Fonte de alimentação 5v	1	15	R$ 15.00
Total	-	-	**R$ 2.412.18**

Figura 5: Tabela de recursos no projeto IdentifierFlow