**🎯 Guia Prático - Prompts Replit AI para Sistema FELKA**

**Desenvolvimento Acelerado com Inteligência Artificial**

**🚀 INTRODUÇÃO - ESTRATÉGIA REPLIT AI**

**Por que Replit AI para o Sistema FELKA?**

* **Aceleração**: 70% redução no tempo de desenvolvimento
* **Qualidade**: AI code review contínuo reduz bugs em 60%
* **Especialização**: Prompts otimizados para reconhecimento facial
* **Produtividade**: De 10 semanas para 6 semanas de desenvolvimento

**Como Usar Este Guia**

1. **Copy & Paste**: Prompts prontos para usar no Replit AI
2. **Sequencial**: Execute os prompts na ordem apresentada
3. **Iterativo**: Refine os resultados conforme necessário
4. **Validação**: Teste cada módulo antes de prosseguir

**📋 FASE 1: SETUP INTELIGENTE (Dias 1-3)**

**🏗️ PROMPT 1.1 - Estrutura Base do Projeto**

Crie a estrutura completa de um projeto Node.js para sistema de controle de acesso com reconhecimento facial. O projeto deve ter:

ESTRUTURA DE PASTAS:

- src/controllers (auth, facial, employee, visitor, vehicle, badge)

- src/services (facial, integration, notification)

- src/models (prisma models)

- src/middleware (auth, validation, security)

- src/routes (api routes modulares)

- src/utils (helpers, constants)

- public/assets (css, js, imagens)

- tests (unit, integration, e2e)

PACKAGE.JSON com dependências:

- Express.js, Prisma, PostgreSQL

- face-api.js, sharp, multer

- bcryptjs, jsonwebtoken, joi

- socket.io, cors, helmet

- winston, dotenv, nodemon

Configure scripts de desenvolvimento, build e teste.

**🏗️ PROMPT 1.2 - Schema Prisma Completo**

Implemente schema Prisma completo para sistema de controle de acesso baseado nestes requisitos:

TABELAS NECESSÁRIAS:

1. User (porteiros e admins) - id, username, password\_hash, role, created\_at

2. Employee - id, registration, cpf, name, position, sector, is\_active, created\_at

3. FacialEncoding - id, employee\_id, visitor\_id, encoding\_data, confidence, created\_at

4. Visitor - id, cpf, name, company, visit\_type, responsible, created\_at

5. AccessLog - id, person\_type, person\_id, direction, method, gatekeeper\_id, timestamp

6. Vehicle - id, license\_plate, driver\_name, checklist\_approved, departure\_time, return\_time

7. SystemConfig - id, key, value, description

RELACIONAMENTOS:

- Employee 1:N FacialEncoding

- Visitor 1:N FacialEncoding

- User 1:N AccessLog

- Vehicle N:1 Employee (driver)

ÍNDICES para performance:

- cpf único em Employee e Visitor

- composite index em AccessLog (timestamp, person\_type)

- index em FacialEncoding.encoding\_data para busca rápida

Inclua seeders para usuário admin inicial e configurações básicas.

**🏗️ PROMPT 1.3 - Configuração Servidor Express**

Configure servidor Express completo com todas as funcionalidades de segurança:

MIDDLEWARE DE SEGURANÇA:

- CORS configurado para frontend específico

- Helmet com política CSP para câmeras

- Rate limiting (100 req/min por IP)

- Morgan para logs de requisições

- Express.json com limite 10MB para imagens

ESTRUTURA DE ROTAS:

- /api/auth (login, logout, refresh)

- /api/facial (encode, recognize, compare)

- /api/employees (CRUD + search)

- /api/visitors (CRUD + quick registration)

- /api/vehicles (list, departure, return)

- /api/badges (generate PDF)

- /api/logs (access history + reports)

WEBSOCKET SETUP:

- Socket.io para atualizações tempo real

- Rooms por portaria

- Eventos: access\_granted, vehicle\_update, system\_alert

ERROR HANDLING:

- Middleware global de erros

- Status codes apropriados

- Logs detalhados com Winston

- Responses padronizados

Configure para rodar na porta 3000 com hot reload.

**🏗️ PROMPT 1.4 - Sistema de Logs e Auditoria**

Implemente sistema completo de logs e auditoria para rastreabilidade:

WINSTON CONFIGURATION:

- File transport para logs permanentes

- Console transport para desenvolvimento

- Levels: error, warn, info, debug

- Formato JSON estruturado

- Rotação diária de arquivos

TIPOS DE LOGS:

1. Security Logs - tentativas de login, acessos negados

2. Access Logs - entradas/saídas de pessoas

3. System Logs - erros, warnings, performance

4. Facial Logs - tentativas de reconhecimento, falhas

5. Integration Logs - comunicação com sistemas externos

ESTRUTURA DO LOG:

- timestamp ISO

- level (error/warn/info)

- module (auth/facial/access)

- action (login/recognize/entry)

- user\_id, ip\_address, user\_agent

- details (objeto com dados específicos)

- duration\_ms (para performance)

AUDITORIA COMPLIANCE:

- Retenção de logs por 2 anos

- Logs imutáveis após criação

- Hash de integridade para logs críticos

- Backup automático diário

Inclua middleware de logging automático para todas as rotas API.

**🔐 FASE 2: AUTENTICAÇÃO INTELIGENTE (Dias 4-6)**

**🔑 PROMPT 2.1 - Sistema de Autenticação JWT**

Implemente sistema completo de autenticação JWT com as seguintes especificações:

AUTH SERVICE:

- Hash passwords com bcrypt (salt rounds: 12)

- Generate JWT tokens (access: 1h, refresh: 7 dias)

- Validate tokens com verificação de expiração

- Role-based authorization (ADMIN, GATEKEEPER)

- Password reset com tokens temporários

AUTH CONTROLLER:

- POST /login - email/username + password

- POST /logout - invalidate refresh token

- POST /refresh - gerar novo access token

- GET /profile - dados do usuário autenticado

- POST /reset-password - reset com email

MIDDLEWARE DE AUTORIZAÇÃO:

- authenticateToken - verificar JWT válido

- requireRole(['ADMIN']) - verificar permissões

- rateLimitAuth - 5 tentativas por 15 minutos

- auditLog - registrar todas tentativas

VALIDAÇÕES JOI:

- Login: email required, password min 6 chars

- Password: complexity rules (upper, lower, number)

- Sanitização de inputs para prevenir injection

SECURITY FEATURES:

- IP tracking para tentativas suspeitas

- Account lockout após 5 tentativas falhas

- Secure httpOnly cookies para refresh tokens

- CSRF protection com tokens

Inclua tratamento de erros detalhado e logs de auditoria completos.

**🔑 PROMPT 2.2 - Middleware de Segurança**

Crie middleware completo de segurança para proteger todas as rotas:

AUTH MIDDLEWARE:

```javascript

// authenticateToken - verificar JWT em headers

// extractUser - adicionar user data ao req

// requireAuth - rejeitar requests não autenticados

AUTHORIZATION MIDDLEWARE:

// requireRole - verificar permissões específicas

// requireAdmin - apenas administradores

// requireGatekeeper - porteiros e admins

SECURITY MIDDLEWARE:

// rateLimitByIP - limitar por endereço IP

// rateLimitByUser - limitar por usuário autenticado

// validateInput - sanitizar dados de entrada

// auditRequest - log completo da requisição

VALIDATION MIDDLEWARE:

// validateBody - validar JSON body com Joi

// validateParams - validar URL parameters

// validateQuery - validar query strings

// validateFiles - validar uploads de imagem

ERROR HANDLING:

* Capturar erros de autenticação
* Responses padronizados para falhas
* Não exposer dados sensíveis em erros
* Log detalhado para debugging

Todos os middlewares devem ser modulares e reutilizáveis.

### 🔑 PROMPT 2.3 - Testes de Autenticação

Gere suite completa de testes para módulo de autenticação:

TESTES UNITÁRIOS (Jest):

* Auth Service: hash/verify passwords, generate/validate tokens
* Auth Controller: todas as rotas com casos de sucesso/falha
* Middleware: authenticateToken, requireRole, rateLimiting
* Validations: schemas Joi com dados válidos/inválidos

TESTES DE INTEGRAÇÃO (Supertest):

* Login com credenciais válidas/inválidas
* Logout e invalidação de tokens
* Refresh token workflow completo
* Rate limiting e lockout de conta
* Autorização por roles em rotas protegidas

CASOS DE TESTE SEGURANÇA:

* Tentativas de brute force
* Tokens expirados ou inválidos
* Injection attacks (SQL, NoSQL, XSS)
* CSRF attacks
* Privilege escalation attempts

MOCKS E FIXTURES:

* Mock do banco de dados
* Usuários de teste (admin, gatekeeper)
* Tokens válidos/inválidos para teste
* Requests maliciosos para security tests

CONFIGURAÇÃO TESTE:

* Database em memória ou test database
* Cleanup após cada teste
* Timeouts apropriados
* Coverage report com 95% mínimo

Organize testes em describe blocks lógicos e inclua setup/teardown.

---

## 🧠 FASE 3: RECONHECIMENTO FACIAL IA (Dias 7-12)

### 👁️ PROMPT 3.1 - Integração Face-api.js

Implemente integração completa com Face-api.js para reconhecimento facial:

CONFIGURAÇÃO FACE-API:

* Carregamento de modelos: ssd\_mobilenetv1, face\_landmark\_68, face\_recognition
* Configuração para dispositivos mobile/tablet
* Otimização de performance com WebGL
* Cache de modelos em localStorage

CAPTURA DE VÍDEO:

* Access webcam com constraints: 640x480 mínimo
* Preview em tempo real no canvas
* Captura automática quando face detectada
* Feedback visual para posicionamento correto

PROCESSAMENTO FACIAL:

* Detecção de face única por frame
* Extração de landmarks para qualidade
* Geração de descriptor de 128 dimensões
* Validação de qualidade da captura (confiança > 0.8)

ENCODING SERVICE:

// encodeFace(imageData) -> descriptor array

// validateFaceQuality(detection) -> boolean

// preprocessImage(canvas) -> optimized data

// saveFaceEncoding(personId, descriptor) -> database

INTERFACE DE CAPTURA:

* Canvas overlay com guias visuais
* Botão manual para captura
* Contador de tentativas (máx 5)
* Feedback de qualidade em tempo real
* Loading states durante processamento

Optimize para tablets Android com camera frontal.

### 👁️ PROMPT 3.2 - Sistema de Matching Facial

Desenvolva algoritmo robusto de matching facial:

COMPARISON ENGINE:

// compareFaces(descriptor1, descriptor2) -> similarity score

// findBestMatch(inputDescriptor, database) -> person + confidence

// bulkCompare(descriptor, encodings[]) -> ranked results

MATCHING CONFIGURATION:

* Threshold padrão: 0.6 (ajustável via admin)
* Máximo comparações por busca: 1000
* Timeout por comparação: 5 segundos
* Cache de encodings ativos em memória

OPTIMIZATION STRATEGIES:

* Index espacial para descriptors similares
* Batch processing para múltiplas comparações
* Worker threads para processamento pesado
* Progressive loading de encodings

FALLBACK HANDLING:

* Múltiplas tentativas de captura
* Diferentes ângulos e condições de luz
* Degradação graceful para entrada manual
* Logs detalhados de falhas

REAL-TIME PROCESSING:

* WebSocket para feedback instantâneo
* Queue system para processamento assíncrono
* Priority queue para funcionários vs visitantes
* Load balancing para múltiplas câmeras

PERFORMANCE METRICS:

* Tempo médio de reconhecimento < 3s
* Taxa de sucesso > 95%
* Taxa de falsos positivos < 1%
* Memory usage otimizado para dispositivos móveis

Inclua sistema de métricas e monitoramento de performance.

### 👁️ PROMPT 3.3 - Interface Mobile de Reconhecimento

Crie interface mobile responsiva para controle de acesso em tablet:

LAYOUT PRINCIPAL (Três Seções):

<!-- Seção Superior: Reconhecimento Facial -->

<section id="facial-recognition" class="h-1/2">

<!-- Vídeo preview com overlay de guias -->

<!-- Status indicator (vermelho/verde/amarelo) -->

<!-- Resultado: nome + cargo/status -->

<!-- Botões: Entrada Manual Funcionário | Entrada Manual Visitante -->

</section>

<!-- Seção Inferior Esquerda: Saída Veículos -->

<section id="vehicle-departure" class="h-1/2 w-1/2">

<!-- Lista veículos com checklist aprovado -->

<!-- Botão "Dar Saída" por veículo -->

</section>

<!-- Seção Inferior Direita: Retorno Veículos -->

<section id="vehicle-return" class="h-1/2 w-1/2">

<!-- Lista veículos em trânsito -->

<!-- Botão "Registrar Retorno" por veículo -->

</section>

COMPONENTES REACT:

* FacialCapture: câmera + processamento
* StatusIndicator: feedback visual claro
* PersonResult: exibição de dados reconhecidos
* VehicleList: lista com ações
* ManualEntry: modals para entrada manual

CSS RESPONSIVO:

* Otimizado para tablets 10-12 polegadas
* Breakpoints: 768px, 1024px, 1280px
* Touch-friendly (botões mín 44px)
* Alto contraste para visibilidade
* Animações suaves para feedback

STATES MANAGEMENT:

* useState para status da câmera
* useEffect para lifecycle da detecção
* useContext para dados globais
* Custom hooks para facial recognition

WebSocket para atualizações tempo real de veículos e status do sistema.

### 👁️ PROMPT 3.4 - Otimização Performance Mobile

Otimize o sistema de reconhecimento facial para performance em dispositivos móveis:

PERFORMANCE OPTIMIZATIONS:

* Reduzir tamanho dos modelos Face-api.js
* Lazy loading de modelos conforme necessário
* Canvas resizing para otimizar processamento
* Frame skipping para reduzir CPU usage
* Memory management com garbage collection

MOBILE-SPECIFIC ADAPTATIONS:

* Detecção de orientação do dispositivo
* Ajuste automático de qualidade baseado em hardware
* Fallback para dispositivos com baixa performance
* Progressive enhancement

CACHING STRATEGIES:

* Service Worker para cache de modelos
* IndexedDB para encodings frequentes
* Memory cache para sessão ativa
* Network-first para dados críticos

BACKGROUND PROCESSING:

* Web Workers para processamento pesado
* Async/await para não bloquear UI
* Queue system para múltiplas detecções
* Priority handling para ações urgentes

BATTERY OPTIMIZATION:

* Pausar detecção quando inativo
* Reduzir frame rate em modo economia
* Throttling automático baseado em bateria
* Sleep mode após inatividade

ERROR HANDLING:

* Graceful degradation para câmeras com problemas
* Fallback automático para entrada manual
* Retry logic inteligente
* User feedback claro sobre problemas

Inclua métricas de performance e monitoramento em tempo real.

---

## 👥 FASE 4: GESTÃO DE PESSOAS (Dias 13-15)

### 👨‍💼 PROMPT 4.1 - Módulo Funcionários

Implemente módulo completo de gestão de funcionários integrado ao reconhecimento facial:

EMPLOYEE SERVICE:

// getActiveEmployees() -> funcionários ativos do RH

// searchEmployee(query) -> busca por nome/cpf/matrícula

// registerFacialData(employeeId, descriptor) -> salvar encoding

// syncWithHR() -> sincronizar base de funcionários

// getEmployeeHistory(employeeId) -> histórico de acessos

EMPLOYEE CONTROLLER:

* GET /employees - lista com paginação e filtros
* GET /employees/search?q= - busca inteligente
* POST /employees/:id/facial - cadastrar face
* GET /employees/:id/access-history - histórico
* PUT /employees/:id - atualizar dados
* DELETE /employees/:id/facial - remover encoding facial

INTEGRAÇÃO COM RH:

* API endpoint para buscar funcionários ativos
* Sincronização automática diária às 6h
* Cache local com TTL de 24 horas
* Fallback para base local quando RH indisponível
* Webhook para mudanças em tempo real

FACIAL REGISTRATION FLOW:

1. Buscar funcionário por matrícula/CPF
2. Validar se está ativo no RH
3. Capturar múltiplas fotos (3-5 ângulos)
4. Processar e gerar encoding médio
5. Salvar com timestamp e qualidade
6. Confirmar sucesso do cadastro

SEARCH INTELLIGENCE:

* Autocomplete com Fuse.js
* Busca fuzzy por nome/sobrenome
* Filtros: setor, cargo, status
* Ordenação por relevância
* Highlights nos resultados

Inclua validações robustas e tratamento de erros específicos.

### 👥 PROMPT 4.2 - Módulo Visitantes

Desenvolva sistema inteligente de gestão de visitantes:

VISITOR SERVICE:

// quickRegister(basicData) -> cadastro rápido

// searchVisitor(cpf, name) -> busca em base histórica

// recordVisit(visitorId, visitData) -> nova visita

// generateVisitorBadge(visitorId) -> crachá PDF

// getVisitorHistory(cpf) -> histórico completo

VISITOR CONTROLLER:

* POST /visitors/quick-register - cadastro rápido na portaria
* GET /visitors/search - busca por CPF/nome
* POST /visitors/:id/visit - registrar nova visita
* GET /visitors/:id/history - histórico de visitas
* POST /visitors/:id/facial - cadastrar reconhecimento facial
* PUT /visitors/:id - atualizar dados básicos

QUICK REGISTRATION FLOW:

1. Campos obrigatórios mínimos: nome, CPF, empresa
2. Campos da visita: tipo, responsável, duração estimada
3. Captura facial opcional (recomendada)
4. Geração automática de crachá
5. Notificação ao responsável interno

VISITOR TYPES:

* Prestador de Serviço
* Cliente/Fornecedor
* Candidato RH
* Visitante Pessoal
* Entrega/Coleta
* Manutenção

SMART FEATURES:

* Detecção de visitantes recorrentes
* Auto-preenchimento de dados históricos
* Sugestão de responsáveis por empresa
* Validação de CPF em tempo real
* Integração com agenda corporativa

COMPLIANCE LGPD:

* Termo de consentimento para dados biométricos
* Opção de entrada sem reconhecimento facial
* Política de retenção configurável
* Anonização automática após prazo

Inclua sistema de notificações e alertas para responsáveis.

### 👥 PROMPT 4.3 - Sistema de Entrada Manual (Fallback)

Implemente sistema robusto de entrada manual como fallback:

FALLBACK SCENARIOS:

* Falha no reconhecimento facial (após 3 tentativas)
* Problema técnico com câmera
* Pessoa não cadastrada no sistema
* Emergência ou situação especial

MANUAL ENTRY INTERFACE:

// Modal overlay com busca inteligente

// Autocomplete com highlight de resultados

// Seleção rápida com dados pré-carregados

// Botões grandes para touch interaction

EMPLOYEE MANUAL ENTRY:

1. Campo de busca: nome, CPF, matrícula
2. Resultados ordenados por relevância
3. Seleção com preview de dados
4. Confirmação de entrada
5. Tag "ENTRADA\_MANUAL" no log
6. Opção para registrar motivo da falha

VISITOR MANUAL ENTRY:

1. Busca em base histórica primeiro
2. Se encontrado: carregar dados + nova visita
3. Se não encontrado: formulário completo
4. Campos obrigatórios mínimos
5. Geração de crachá temporário
6. Registro com tipo "MANUAL"

SEARCH OPTIMIZATION:

* Debounce de 300ms para performance
* Cache de resultados frequentes
* Indexação otimizada para busca
* Fuzzy search com Fuse.js
* Filtros por categoria (funcionário/visitante)

EMERGENCY FEATURES:

* Modo emergência para acesso irrestrito
* Log especial para situações de emergência
* Notificação automática para administradores
* Override manual para administradores

Inclua validação de dados e feedback visual claro.

---

## 🚛 FASE 5: CONTROLE DE VEÍCULOS (Dias 16-18)

### 🚗 PROMPT 5.1 - Módulo Veículos Integrado

Implemente controle completo de veículos integrado ao sistema de checklist:

VEHICLE SERVICE:

// getApprovedVehicles() -> veículos com checklist OK

// registerDeparture(vehicleId, gatekeeperId) -> saída

// registerReturn(vehicleId, baseId) -> retorno

// getInTransitVehicles() -> veículos em trânsito

// syncWithChecklist() -> sincronizar dados de checklist

VEHICLE CONTROLLER:

* GET /vehicles/approved - lista veículos liberados
* POST /vehicles/:id/departure - registrar saída
* POST /vehicles/:id/return - registrar retorno
* GET /vehicles/in-transit - veículos em viagem
* GET /vehicles/:id/tracking - rastreamento completo

INTEGRATION WITH CHECKLIST:

* Buscar veículos com status "CHECKLIST\_APPROVED"
* Validar se todos itens obrigatórios foram verificados
* Cache local com sincronização a cada 5 minutos
* Fallback para lista manual em caso de falha

DEPARTURE WORKFLOW:

1. Listar veículos com checklist aprovado
2. Selecionar veículo na interface tablet
3. Confirmar dados: placa, motorista, destino
4. Registrar porteiro responsável pela liberação
5. Timestamp de saída
6. Atualizar status para "EM\_TRANSITO"
7. WebSocket update para todas interfaces

RETURN WORKFLOW:

1. Listar veículos em trânsito
2. Selecionar veículo retornando
3. Selecionar base de retorno
4. Registrar timestamp de chegada
5. Calcular tempo total de viagem
6. Atualizar status para "RETORNADO"
7. Notificar gestores se necessário

REAL-TIME UPDATES:

* WebSocket events para mudanças de status
* Sincronização automática entre bases
* Notifications para atrasos > tempo estimado
* Dashboard em tempo real para gestores

Inclua validações de negócio e tratamento de edge cases.

### 🎫 PROMPT 5.2 - Gerador de Crachás

Desenvolva sistema automático de geração de crachás em PDF:

BADGE GENERATOR SERVICE:

// generateEmployeeBadge(employeeId) -> PDF buffer

// generateVisitorBadge(visitorId, visitData) -> PDF buffer

// generateQRCode(accessData) -> QR code image

// printBadge(pdfBuffer) -> send to thermal printer

PDF GENERATION (PDF-lib):

* Template base para funcionários e visitantes
* Layout responsivo para diferentes tamanhos
* Qualidade de impressão: 300 DPI
* Formato: A6 ou tamanho personalizado

EMPLOYEE BADGE TEMPLATE:

* Header: Logo empresa + "FUNCIONÁRIO"
* Foto: placeholder ou imagem do RH
* Dados: Nome, Cargo, Setor, Matrícula
* QR Code: dados de acesso codificados
* Footer: Data/hora de geração
* Cores: esquema corporativo

VISITOR BADGE TEMPLATE:

* Header: Logo empresa + "VISITANTE"
* Foto: capturada ou placeholder
* Dados: Nome, Empresa, Responsável
* Tipo de visita e duração estimada
* QR Code: dados de acesso temporário
* Footer: "Válido até [data/hora]"
* Cores: diferenciadas (ex: laranja)

QR CODE INTEGRATION:

* Dados codificados: ID, tipo, timestamp, hash
* Formato: JSON compacto
* Verificação de integridade
* Expiração automática

PRINTER INTEGRATION:

* Suporte para impressoras térmicas
* Configuração de tamanho de papel
* Quality settings automáticos
* Queue de impressão
* Fallback para impressora padrão

AUTO-GENERATION TRIGGERS:

* Funcionário: primeira entrada do dia
* Visitante: a cada nova visita
* Reposição: quando crachá perdido
* Emergência: geração manual por admin

Inclua preview antes da impressão e logs de geração.

---

## 📊 FASE 6: INTERFACES E RELATÓRIOS (Dias 19-21)

### 🖥️ PROMPT 6.1 - Interface Administrativa Web

Desenvolva interface web completa para administração do sistema:

DASHBOARD PRINCIPAL:

* Métricas em tempo real: acessos hoje, pessoas presentes, veículos em trânsito
* Gráficos: Chart.js para acessos por hora, top funcionários, tipos de entrada
* Alertas: falhas de reconhecimento, tentativas suspeitas, sistema offline
* Quick actions: cadastro facial, relatórios, configurações

GESTÃO DE FUNCIONÁRIOS:

* Tabela com busca, filtros e paginação
* Ações: visualizar, editar, cadastrar facial, histórico
* Bulk operations: exportar, importar, sincronizar RH
* Modal para cadastro/edição facial
* Preview da qualidade do encoding

GESTÃO DE VISITANTES:

* Histórico completo de visitas
* Filtros avançados: período, empresa, tipo, responsável
* Estatísticas: visitantes únicos, empresas mais frequentes
* Exportação de relatórios
* Gestão de dados LGPD

CONFIGURAÇÕES DO SISTEMA:

* Threshold de reconhecimento facial
* Horários de funcionamento
* Integrações ativas/inativas
* Configurações de impressora
* Backup e manutenção

MÓDULO DE USUÁRIOS:

* CRUD completo para porteiros e admins
* Gestão de permissões e roles
* Logs de atividades por usuário
* Configurações pessoais
* Reset de senhas

RESPONSIVE DESIGN:

* Desktop-first com adaptação mobile
* Sidebar colapsível
* Tables responsivas com scroll horizontal
* Modals mobile-friendly
* Dark mode opcional

Use React + Tailwind CSS com componentes reutilizáveis.

### 📈 PROMPT 6.2 - Sistema de Relatórios Analytics

Crie sistema completo de relatórios e analytics:

RELATÓRIOS OPERACIONAIS:

1. Relatório de Acessos
   * Período configurável
   * Filtros: pessoa, tipo entrada, porteiro
   * Dados: timestamp, método, duração permanência
   * Exportação: PDF, Excel, CSV
2. Relatório de Visitantes
   * Visitantes únicos por período
   * Empresas mais frequentes
   * Tipos de visita mais comuns
   * Responsáveis mais requisitados
3. Relatório de Veículos
   * Saídas e retornos por período
   * Tempo médio de viagem por rota
   * Veículos mais utilizados
   * Performance de checklists
4. Relatório de Performance
   * Taxa de sucesso reconhecimento facial
   * Tempo médio de processamento
   * Picos de uso do sistema
   * Falhas e downtimes

ANALYTICS DASHBOARD:

// Componentes Chart.js:

// - Line chart: acessos por hora/dia

// - Bar chart: top funcionários, empresas visitantes

// - Pie chart: tipos de entrada, métodos de acesso

// - Heat map: horários de pico

FILTROS INTELIGENTES:

* Date range picker com presets
* Multi-select para pessoas/empresas
* Filtros salvos para relatórios frequentes
* Quick filters: hoje, semana, mês
* Comparação entre períodos

EXPORTAÇÃO AVANÇADA:

* PDF com gráficos incluídos
* Excel com múltiplas abas
* CSV para análises externas
* Email automático de relatórios
* Agendamento de relatórios recorrentes

REAL-TIME MONITORING:

* Live dashboard com auto-refresh
* WebSocket para métricas em tempo real
* Alertas configuráveis por threshold
* Sistema de notificações push
* Status health check do sistema

KPI TRACKING:

* Taxa de reconhecimento facial
* Tempo médio de entrada
* Satisfação dos usuários
* Performance do hardware
* Uptime do sistema

Inclua cache inteligente para performance e exportação assíncrona.

---

## 🔧 FASE 7: TESTES E VALIDAÇÃO (Dias 22-25)

### 🧪 PROMPT 7.1 - Testes Automatizados Completos

Gere suite completa de testes automatizados para todo o sistema:

UNIT TESTS (Jest):

// Facial Recognition:

// - Face encoding accuracy

// - Comparison algorithm precision

// - Threshold validation

// - Error handling para casos edge

// Authentication:

// - JWT generation/validation

// - Password hashing/verification

// - Role-based authorization

// - Rate limiting functionality

// Business Logic:

// - Employee registration flow

// - Visitor management

// - Vehicle control logic

// - Badge generation

INTEGRATION TESTS (Supertest):

// API Endpoints:

// - Auth workflow completo

// - CRUD operations para todas entidades

// - File upload para imagens faciais

// - WebSocket connections

// Database Integration:

// - Prisma operations

// - Transaction handling

// - Data consistency

// - Performance queries

E2E TESTS (Puppeteer):

// User Journeys:

// - Complete employee registration

// - Facial recognition success/failure