E-BOOK

ALUR TEKNIS APLIKASI KISS

Tabel dan Relasi Database

Dokumentasi Teknis Lengkap

Penulis dan Pengembang:
MUKHSIN HADI, SE, M.Si, CGAA, CPFRM, CSEP, CRP, CPRM, CSCAP, CPABC

DAFTAR ISI

BAB I: ARSITEKTUR SISTEM

1.1 Overview Arsitektur

Aplikasi KISS dibangun menggunakan arsitektur modern yang mengutamakan skalabilitas, maintainability, dan performance. Sistem ini mengadopsi pola arsitektur microservices dengan separation of concerns yang jelas antara frontend, backend, dan database layer.

Arsitektur High-Level KISS

```
┌─────────────────────────────────────────────────────────────┐
│                    PRESENTATION LAYER                       │
├─────────────────────────────────────────────────────────────┤
│  React Frontend     │  Mobile App      │  QR Code Interface │
│  (Port 3000)        │  (React Native)│  (Public Access)   │
└─────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌─────────────────────────────────────────────────────────────┐
│                      API GATEWAY                            │
├─────────────────────────────────────────────────────────────┤
│  • Authentication & Authorization                           │
│  • Rate Limiting & Throttling                              │
│  • Request Routing & Load Balancing                        │
│  • API Versioning & Documentation                          │
└─────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌─────────────────────────────────────────────────────────────┐
│                 BUSINESS LOGIC LAYER                       │
├─────────────────────────────────────────────────────────────┤
│  Auth Service   │ Complaint Svc │ AI Service │ Notification │
│  User Mgmt      │ Survey Svc    │ Analytics  │ Reporting    │
│  Master Data    │ QR Code Svc   │ Escalation │ Integration  │
└─────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌─────────────────────────────────────────────────────────────┐
│                      DATA LAYER                             │
├─────────────────────────────────────────────────────────────┤
│  PostgreSQL      │  File Storage  │  Cache Layer │ Audit Log │
│  (Supabase)      │  (Local/Cloud) │  (Redis)       │ (ELK)        │
└──────────────────────────────────────────────────────────────────┘
```

1.2 Technology Stack

1.2.1 Frontend Technologies

Core Framework: React 18.2.0 + TypeScript 4.9.5

Build Tool: Vite 4.4.5 (Fast HMR, ES modules)

Styling: Tailwind CSS 3.3.0 (Utility-first CSS)

State Management: React Context + useReducer

HTTP Client: Axios 1.4.0 (Promise-based HTTP client)

Routing: React Router 6.14.0 (Client-side routing)

Icons: Lucide React 0.263.1 (Modern icon library)

Real-time: Socket.io Client 4.7.2

Testing: Vitest + React Testing Library

1.2.2 Backend Technologies

Runtime: Node.js 18.17.0 LTS

Framework: Express.js 4.18.2

Language: TypeScript 5.1.6

Database: PostgreSQL 15.3 (via Supabase)

Authentication: JWT + bcryptjs

File Upload: Multer 1.4.5

Real-time: Socket.io 4.7.2

Validation: Joi 17.9.2

Logging: Winston 3.10.0

Testing: Jest + Supertest

1.2.3 Database & Infrastructure

```
Primary Database: PostgreSQL 15.3
Database Service: Supabase (BaaS)
Authentication: Supabase Auth
Real-time: Supabase Realtime
File Storage: Local Storage + Cloud Backup
Caching: Redis (planned)
Monitoring: Supabase Dashboard
Deployment: Vercel (Frontend) + Railway (Backend)
```

1.3 Design Patterns dan Principles

1.3.1 Architectural Patterns

- **MVC (Model-View-Controller):** Separation of concerns dalam backend
- **Repository Pattern:** Abstraksi layer database untuk maintainability
- **Service Layer Pattern:** Business logic terpisah dari controller
- **Observer Pattern:** Event-driven architecture untuk real-time updates
- **Factory Pattern:** Dynamic service instantiation

1.3.2 SOLID Principles Implementation

```
Single Responsibility: Setiap class/module memiliki satu tanggung jawab
Open/Closed: Terbuka untuk extension, tertutup untuk modification
Liskov Substitution: Derived classes dapat menggantikan base classes
Interface Segregation: Interface yang spesifik dan focused
Dependency Inversion: Depend on abstractions, not concretions
```

BAB II: STRUKTUR DATABASE

2.1 Database Schema Overview

Database KISS menggunakan PostgreSQL sebagai primary database dengan struktur yang dinormalisasi untuk memastikan integritas data dan performa optimal. Schema dirancang untuk mendukung skalabilitas horizontal dan vertical seiring pertumbuhan data.

Database Configuration

```
-- Database: kiss_production -- Version: PostgreSQL 15.3 -- Character Set: UTF8 -
- Collation: en_US.UTF-8 -- Connection Limit: 100 -- Shared Buffers: 256MB --
Effective Cache Size: 1GB -- Work Memory: 4MB -- Maintenance Work Memory: 64MB
```

2.2 Schema Organization

2.2.1 Schema Structure

Schema	Purpose	Tables	Description
public	Main application data	25+	Core business entities dan operational data
auth	Authentication (Supabase)	8	User authentication dan session management
audit	Audit trail	3	Logging semua perubahan data untuk compliance
analytics	Data warehouse	5	Aggregated data untuk reporting dan analytics

2.3 Data Types dan Conventions

2.3.1 Standard Data Types

```
-- Primary Keys: UUID (gen_random_uuid()) -- Timestamps: TIMESTAMPTZ (timezone aware)
-- Text Fields: VARCHAR(n) untuk bounded, TEXT untuk unbounded -- JSON Data: JSONB
(binary JSON untuk performance) -- Enums: VARCHAR dengan CHECK constraints -- Boolean:
BOOLEAN dengan DEFAULT values -- Numeric: DECIMAL untuk currency, INTEGER untuk counts
```

2.3.2 Naming Conventions

- **Tables:** snake_case, plural nouns (e.g., service_categories)
- **Columns:** snake_case, descriptive names (e.g., created_at)
- **Indexes:** idx_tablename_columnname (e.g., idx_tickets_status)
- **Foreign Keys:** fk_tablename_referenced_table (e.g., fk_tickets_units)
- **Constraints:** chk_tablename_condition (e.g., chk_tickets_priority)

BAB III: TABEL-TABEL UTAMA

3.1 Core Entity Tables

3.1.1 Tabel ADMINS

Purpose: Authentication dan user management untuk admin

```
CREATE TABLE admins ( id UUID PRIMARY KEY DEFAULT gen_random_uuid(), username
VARCHAR(255) UNIQUE NOT NULL, password_hash TEXT NOT NULL, full_name
VARCHAR(255), email VARCHAR(255) UNIQUE, role VARCHAR(50) DEFAULT 'admin' CHECK
(role IN ('admin', 'superadmin')), is_active BOOLEAN DEFAULT true, last_login
TIMESTAMPTZ, created_at TIMESTAMPTZ DEFAULT now(), updated_at TIMESTAMPTZ DEFAULT
now() ); -- Indexes CREATE INDEX idx_admins_username ON admins(username); CREATE
INDEX idx_admins_email ON admins(email); CREATE INDEX idx_admins_role ON
admins(role); CREATE INDEX idx_admins_is_active ON admins(is_active); -- Comments
COMMENT ON TABLE admins IS 'Tabel untuk menyimpan data admin dan authentication';
COMMENT ON COLUMN admins.username IS 'Username untuk login (unique)'; COMMENT ON
COLUMN admins.password_hash IS 'Password yang sudah di-hash dengan bcrypt';
COMMENT ON COLUMN admins.role IS 'Role admin: admin atau superadmin'; COMMENT ON
COLUMN admins.is_active IS 'Status aktif admin';
```

3.1.2 Tabel UNITS

Purpose: Organizational units/departments untuk routing tiket

```
CREATE TABLE units ( id UUID PRIMARY KEY DEFAULT gen_random_uuid(), name
VARCHAR(255) NOT NULL, code VARCHAR(50) UNIQUE NOT NULL, parent_unit_id UUID
REFERENCES units(id), unit_type_id UUID REFERENCES unit_types(id), description
TEXT, contact_email VARCHAR(255), contact_phone VARCHAR(50), sla_hours INTEGER
DEFAULT 24, is_active BOOLEAN DEFAULT true, created_at TIMESTAMPTZ DEFAULT now(),
updated_at TIMESTAMPTZ DEFAULT now() ); -- Indexes CREATE INDEX idx_units_code ON
units(code); CREATE INDEX idx_units_parent_unit_id ON units(parent_unit_id);
CREATE INDEX idx_units_unit_type_id ON units(unit_type_id); CREATE INDEX
idx_units_is_active ON units(is_active); -- Self-referencing foreign key ALTER
```

```
TABLE units ADD CONSTRAINT fk_units_parent_unit FOREIGN KEY (parent_unit_id)
REFERENCES units(id);
```

3.1.3 Tabel TICKETS

Purpose: Core table untuk semua jenis tiket (keluhan, saran, informasi)

```
CREATE TABLE tickets ( id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
ticket_number VARCHAR(50) UNIQUE NOT NULL, type VARCHAR(50) NOT NULL CHECK (type
IN ( 'information', 'complaint', 'suggestion', 'satisfaction' )), category_id
UUID REFERENCES service_categories(id), title VARCHAR(500) NOT NULL, description
TEXT NOT NULL, -- Submitter information submitter_name VARCHAR(255),
submitter_email VARCHAR(255), submitter_phone VARCHAR(50), submitter_address
TEXT, is_anonymous BOOLEAN DEFAULT false, -- Assignment and routing unit_id UUID
NOT NULL REFERENCES units(id), assigned_to UUID REFERENCES users(id), created_by
UUID REFERENCES users(id), -- Status and priority status VARCHAR(50) DEFAULT
'open' CHECK (status IN ( 'open', 'in_progress', 'escalated', 'resolved',
'closed' )), priority VARCHAR(50) DEFAULT 'medium' CHECK (priority IN ( 'low',
'medium', 'high', 'critical' )), urgency_level INTEGER DEFAULT 3 CHECK
(urgency_level BETWEEN 1 AND 5), -- AI Analysis ai_classification JSONB,
sentiment_score DECIMAL(3,2), confidence_score DECIMAL(3,2), -- SLA Management
sla_deadline TIMESTAMPTZ, first_response_at TIMESTAMPTZ, resolved_at TIMESTAMPTZ,
-- Tracking source VARCHAR(50) DEFAULT 'web' CHECK (source IN ( 'web', 'qr_code',
'mobile', 'email', 'phone' )), qr_code_id UUID REFERENCES qr_codes(id),
ip_address INET, user_agent TEXT, created_at TIMESTAMPTZ DEFAULT now(),
updated_at TIMESTAMPTZ DEFAULT now() ); -- Performance Indexes CREATE INDEX
idx_tickets_ticket_number ON tickets(ticket_number); CREATE INDEX
idx_tickets_type ON tickets(type); CREATE INDEX idx_tickets_status ON
tickets(status); CREATE INDEX idx_tickets_priority ON tickets(priority); CREATE
INDEX idx_tickets_unit_id ON tickets(unit_id); CREATE INDEX
idx_tickets_assigned_to ON tickets(assigned_to); CREATE INDEX
idx_tickets_created_at ON tickets(created_at); CREATE INDEX
idx_tickets_sla_deadline ON tickets(sla_deadline); CREATE INDEX
idx_tickets_source ON tickets(source); -- Composite Indexes untuk query
optimization CREATE INDEX idx_tickets_status_priority ON tickets(status,
priority); CREATE INDEX idx_tickets_unit_status ON tickets(unit_id, status);
CREATE INDEX idx_tickets_assigned_status ON tickets(assigned_to, status);
```

3.2 Supporting Tables

3.2.1 Tabel SERVICE_CATEGORIES

```
CREATE TABLE service_categories ( id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
name VARCHAR(255) NOT NULL, code VARCHAR(50) UNIQUE NOT NULL, description TEXT,
default_sla_hours INTEGER DEFAULT 24, requires_attachment BOOLEAN DEFAULT false,
is_active BOOLEAN DEFAULT true, created_at TIMESTAMPTZ DEFAULT now(), updated_at
TIMESTAMPTZ DEFAULT now() ); -- Sample Data INSERT INTO service_categories (name,
code, description, default_sla_hours) VALUES ('Pelayanan Medis', 'MED', 'Keluhan
terkait pelayanan medis dan klinis', 24), ('Administrasi', 'ADM', 'Keluhan
terkait proses administrasi', 48), ('Fasilitas', 'FAC', 'Keluhan terkait
fasilitas dan infrastruktur', 72), ('SDM', 'HRM', 'Keluhan terkait sumber daya
manusia', 48), ('Keuangan', 'FIN', 'Keluhan terkait pembayaran dan keuangan',
24), ('Farmasi', 'PHR', 'Keluhan terkait pelayanan farmasi', 24),
('Laboratorium', 'LAB', 'Keluhan terkait pelayanan laboratorium', 24);
```

3.2.2 Tabel QR_CODES

```
CREATE TABLE qr_codes ( id UUID PRIMARY KEY DEFAULT gen_random_uuid(), unit_id
UUID NOT NULL REFERENCES units(id), code VARCHAR(255) UNIQUE NOT NULL, token
VARCHAR(500) UNIQUE NOT NULL, name VARCHAR(255) NOT NULL, description TEXT,
is_active BOOLEAN DEFAULT true, usage_count INTEGER DEFAULT 0, created_at
TIMESTAMPTZ DEFAULT now(), updated_at TIMESTAMPTZ DEFAULT now() ); -- Indexes
CREATE INDEX idx_qr_codes_unit_id ON qr_codes(unit_id); CREATE INDEX
idx_qr_codes_code ON qr_codes(code); CREATE INDEX idx_qr_codes_token ON
qr_codes(token); CREATE INDEX idx_qr_codes_is_active ON qr_codes(is_active); --
Function untuk generate QR code CREATE OR REPLACE FUNCTION
generate_qr_code(unit_name VARCHAR) RETURNS VARCHAR AS $$ BEGIN RETURN 'QR-' ||
UPPER(LEFT(unit_name, 3)) || '-' || TO_CHAR(NOW(), 'YYYYMMDD') || '-' ||
LPAD(FLOOR(RANDOM() * 10000)::TEXT, 4, '0'); END; $$ LANGUAGE plpgsql;
```

BAB IV: RELASI ANTAR TABEL

4.1 Entity Relationship Diagram

ERD - Core Entities

```
┌─────────────────┐    ┌─────────────────┐    ┌─────────────────┐
│     ADMINS      │    │     USERS       │    │     UNITS       │
├─────────────────┤    ├─────────────────┤    ├─────────────────┤
│ id (PK)         │    │ id (PK)         │    │ id (PK)         │
│ username        │    │ admin_id (FK)   │────│ name            │
│ password_hash   │    │ full_name       │    │ code            │
│ email           │    │ email           │    │ parent_unit_id  │──┐
│ role            │    │ unit_id (FK)    │────│ unit_type_id    │  │
│ is_active       │    │ role            │    │ sla_hours       │  │
└─────────────────┘    │ is_active       │    │ is_active       │  │
         │              └─────────────────┘    └─────────────────┘   │
         │                       │                       │         │
         └───────────────────────┘                       │         │
                                                         │         │
┌─────────────────┐    ┌─────────────────┐              │         │
│  UNIT_TYPES     │    │   QR_CODES      │              │         │
├─────────────────┤    ├─────────────────┤              │         │
│ id (PK)         │────│ id (PK)         │              │         │
│ name            │    │ unit_id (FK)    │──────────────┘         │
│ code            │    │ code            │                        │
│ icon            │    │ token           │                        │
│ color           │    │ usage_count     │                        │
└─────────────────┘    └─────────────────┘                        │
                                │                                  │
                                │                                  │
┌─────────────────┐    ┌─────────────────┐                        │
│SERVICE_CATEGORIES│   │    TICKETS      │                        │
├─────────────────┤    ├─────────────────┤                        │
│ id (PK)         │────│ id (PK)         │                        │
│ name            │    │ ticket_number   │                        │
│ code            │    │ category_id(FK) │                        │
│ default_sla_hrs │    │ unit_id (FK)    │────────────────────────┘
└─────────────────┘    │ assigned_to(FK) │
                       │ qr_code_id (FK) │──────────────────────────┘
                       │ type            │
                       │ status          │
                       │ priority        │
                       │ ai_classification│
                       └─────────────────┘
                                │
                                │
┌─────────────────┐    ┌─────────────────┐    ┌─────────────────┐
│TICKET_RESPONSES │    │TICKET_ESCALATIONS│   │SATISFACTION_SRVS│
```

```
├─────────────────┤     ├─────────────────┤     ├─────────────────┤
│ id (PK)         │     │ id (PK)         │     │ id (PK)         │
│ ticket_id (FK)  │─────│ ticket_id (FK)  │─────│ ticket_id (FK)  │
│ responder_id(FK)│     │ from_user_id(FK)│     │ overall_score   │
│ message         │     │ to_user_id (FK) │     │ response_time   │
│ is_internal     │     │ reason          │     │ solution_quality│
└─────────────────┘     └─────────────────┘     │ staff_courtesy  │
                                                └─────────────────┘
```

4.2 Foreign Key Relationships

4.2.1 Primary Relationships

Child Table	Foreign Key	Parent Table	Relationship Type	Cascade Rule
users	admin_id	admins	One-to-One	SET NULL
users	unit_id	units	Many-to-One	RESTRICT
units	parent_unit_id	units	Self-Reference	SET NULL
units	unit_type_id	unit_types	Many-to-One	RESTRICT
tickets	unit_id	units	Many-to-One	RESTRICT
tickets	category_id	service_categories	Many-to-One	SET NULL
tickets	assigned_to	users	Many-to-One	SET NULL
qr_codes	unit_id	units	Many-to-One	CASCADE

4.2.2 Referential Integrity Constraints

```
-- Constraint untuk memastikan data integrity ALTER TABLE tickets ADD CONSTRAINT
chk_tickets_priority CHECK (priority IN ('low', 'medium', 'high', 'critical')); ALTER
TABLE tickets ADD CONSTRAINT chk_tickets_status CHECK (status IN ('open',
'in_progress', 'escalated', 'resolved', 'closed')); ALTER TABLE tickets ADD CONSTRAINT
chk_tickets_urgency CHECK (urgency_level BETWEEN 1 AND 5); -- Constraint untuk SLA
deadline ALTER TABLE tickets ADD CONSTRAINT chk_tickets_sla_deadline CHECK
(sla_deadline > created_at); -- Constraint untuk resolved_at ALTER TABLE tickets ADD
CONSTRAINT chk_tickets_resolved_at CHECK (resolved_at IS NULL OR resolved_at >=
created_at); -- Unique constraint untuk ticket_number dengan format ALTER TABLE
tickets ADD CONSTRAINT chk_tickets_number_format CHECK (ticket_number ~ '^TKT-[0-9]
{8}-[0-9]{4}$');
```

4.3 Junction Tables

4.3.1 Tabel USER_ROLES (Many-to-Many)

```
CREATE TABLE user_roles ( id UUID PRIMARY KEY DEFAULT gen_random_uuid(), user_id
UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE, role_id UUID NOT NULL
REFERENCES roles(id) ON DELETE CASCADE, assigned_at TIMESTAMPTZ DEFAULT now(),
assigned_by UUID REFERENCES users(id), is_active BOOLEAN DEFAULT true, created_at
TIMESTAMPTZ DEFAULT now(), updated_at TIMESTAMPTZ DEFAULT now(), -- Prevent
duplicate assignments UNIQUE(user_id, role_id) ); -- Indexes untuk performance
CREATE INDEX idx_user_roles_user_id ON user_roles(user_id); CREATE INDEX
idx_user_roles_role_id ON user_roles(role_id); CREATE INDEX
idx_user_roles_is_active ON user_roles(is_active);
```

4.3.2 Tabel TICKET_ATTACHMENTS

```
CREATE TABLE ticket_attachments ( id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
ticket_id UUID NOT NULL REFERENCES tickets(id) ON DELETE CASCADE, file_name
VARCHAR(255) NOT NULL, file_path TEXT NOT NULL, file_size BIGINT NOT NULL,
mime_type VARCHAR(100) NOT NULL, uploaded_by UUID REFERENCES users(id),
created_at TIMESTAMPTZ DEFAULT now(), -- Constraints CHECK (file_size > 0 AND
file_size <= 104857600), -- Max 100MB CHECK (mime_type IN ( 'image/jpeg',
'image/png', 'image/gif', 'application/pdf', 'application/msword',
'application/vnd.openxmlformats-officedocument.wordprocessingml.document',
'video/mp4', 'video/avi', 'video/quicktime' )) ); -- Indexes CREATE INDEX
idx_ticket_attachments_ticket_id ON ticket_attachments(ticket_id); CREATE INDEX
idx_ticket_attachments_mime_type ON ticket_attachments(mime_type);
```

TENTANG PENULIS

MUKHSIN HADI, SE, M.Si, CGAA, CPFRM, CSEP, CRP, CPRM, CSCAP, CPABC

Praktisi dan akademisi di bidang sistem informasi manajemen dengan pengalaman lebih dari 15 tahun dalam pengembangan aplikasi enterprise untuk sektor publik. Memiliki sertifikasi internasional dalam bidang audit, risk management, dan project management.

Saat ini menjabat sebagai konsultan senior untuk berbagai proyek digitalisasi di sektor kesehatan dan pemerintahan. Aktif dalam penelitian dan pengembangan sistem informasi berbasis AI untuk meningkatkan kualitas pelayanan publik.