FURSA HUB - SYSTEM ARCHITECTURE

Version: 1.0
Date: December 2024
Prepared by: Josh

---

1. EXECUTIVE SUMMARY

Fursa Hub is an opportunity marketplace platform connecting opportunity seekers with opportunity providers across funding, startup programs, tenders, and jobs within a social ecosystem designed for the East African market.

The platform consists of mobile applications (iOS and Android), a web management dashboard, and a microservices backend architecture.

Core Value Proposition: Post → Apply model where organizations post opportunities and individuals/businesses apply for them.

---

2. AUTHENTICATION ARCHITECTURE

2.1 Overview

The authentication system supports multiple authentication methods to accommodate diverse user preferences and security requirements.

2.2 Authentication Methods

Method	Description
Email + Password	Traditional email-based authentication
Phone + Password	Phone number with password

Method	Description
Username + Password	Username-based login
Passwordless (Email OTP)	One-time password sent via email
Passwordless (Phone OTP)	One-time password sent via SMS
OAuth2 (Google)	Social login via Google
OAuth2 (Apple)	Social login via Apple

2.3 Token Strategy

- **Type:** JWT (JSON Web Tokens)
- **Access Token:** Short-lived (15-30 minutes)
- **Refresh Token:** Long-lived (7-30 days)
- **Storage:** Access token in memory, refresh token in secure HTTP-only cookie

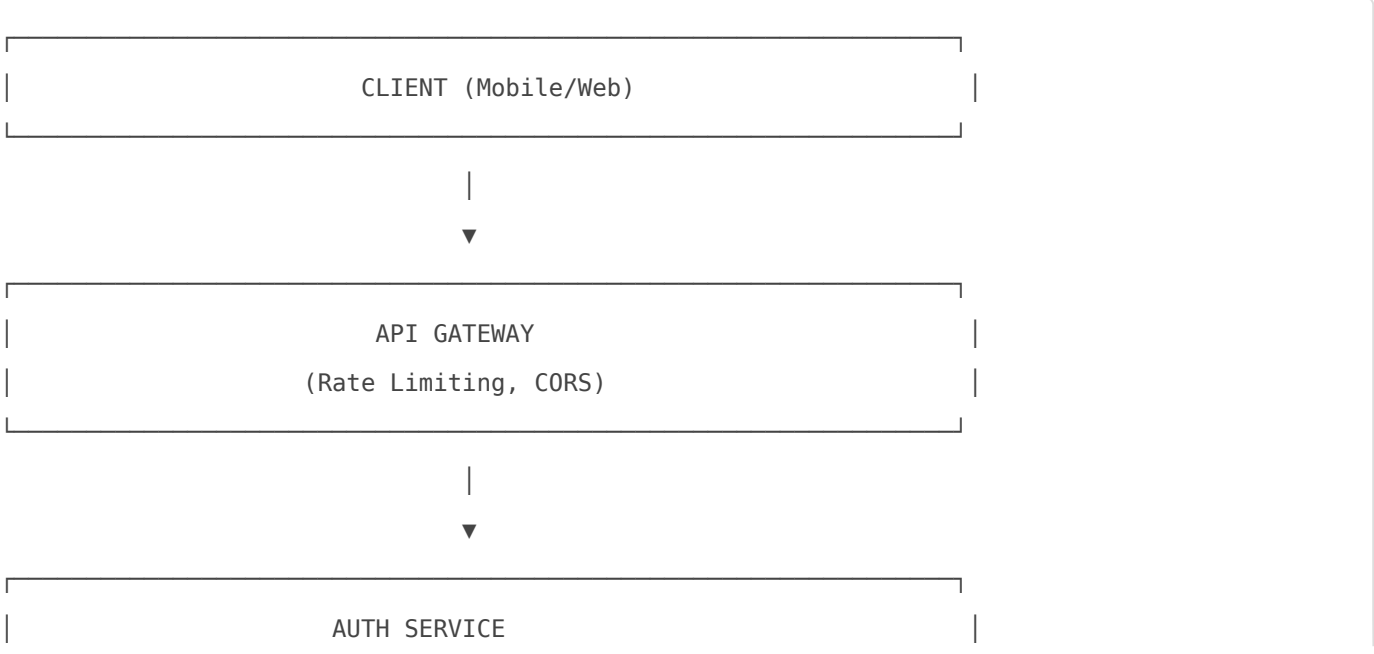
2.4 Phone Verification

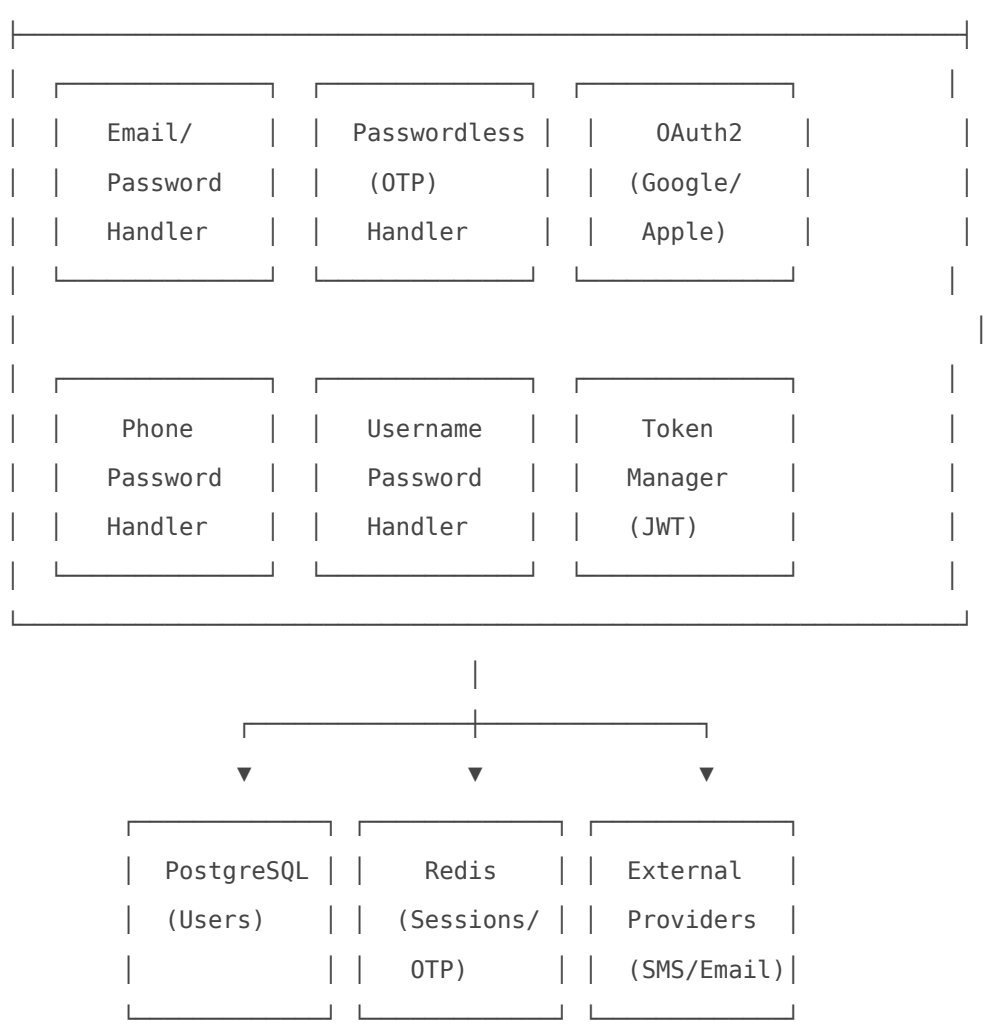
Phone number is **required** for all users and must be verified via OTP before account activation.

2.5 Account Linking

When a user signs up with one method (e.g., email) and later attempts OAuth2 with the same email, accounts will be linked automatically.

2.6 Authentication Flow Diagram





2.7 OTP Flow

```
User Request → Generate OTP → Store in Redis (5 min TTL) → Send via SMS/Email
                                                                    |
User Submit OTP → Validate against Redis → Success → Issue JWT Tokens
                                         → Failure → Increment attempt counter
```

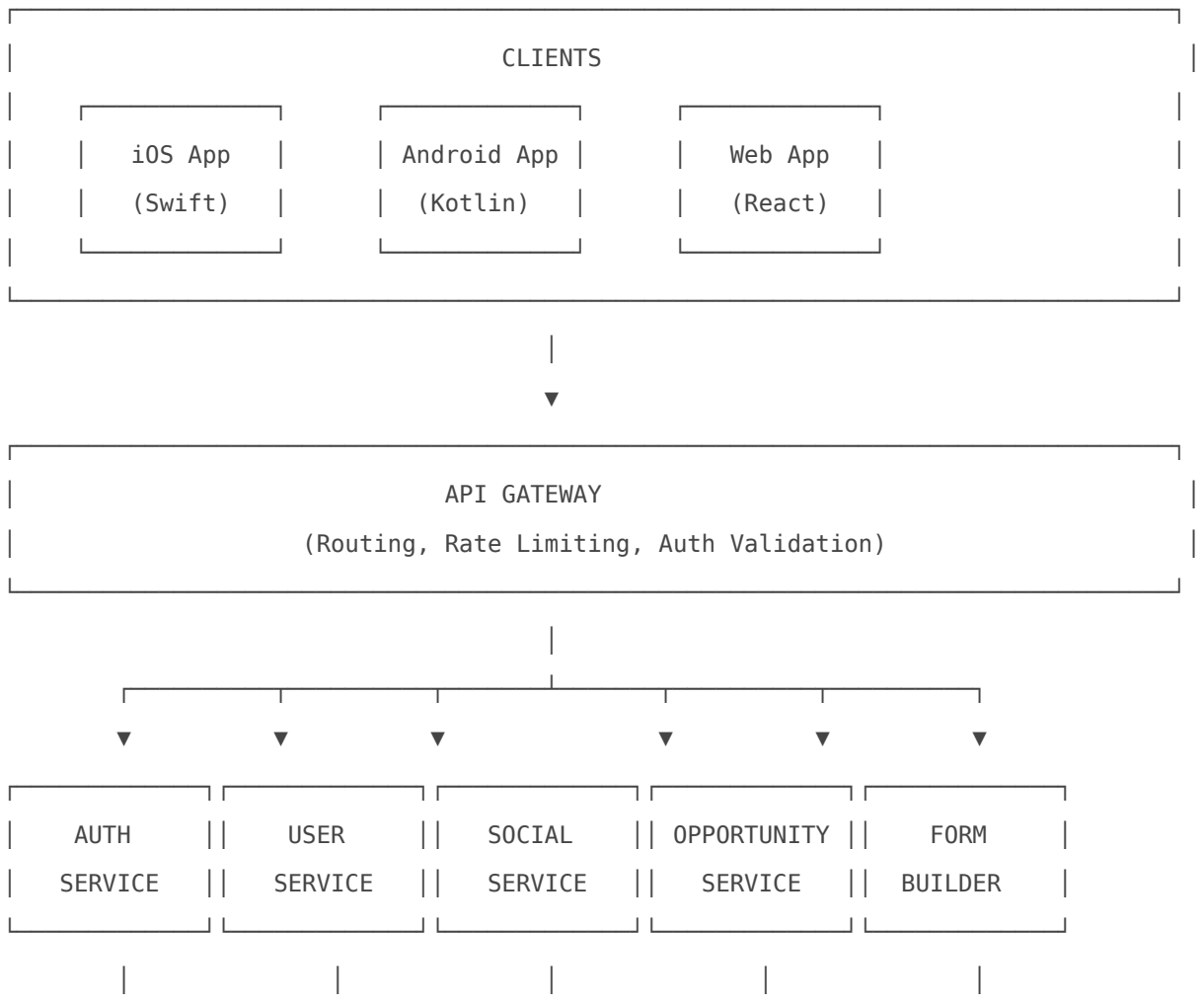
2.8 OAuth2 Flow

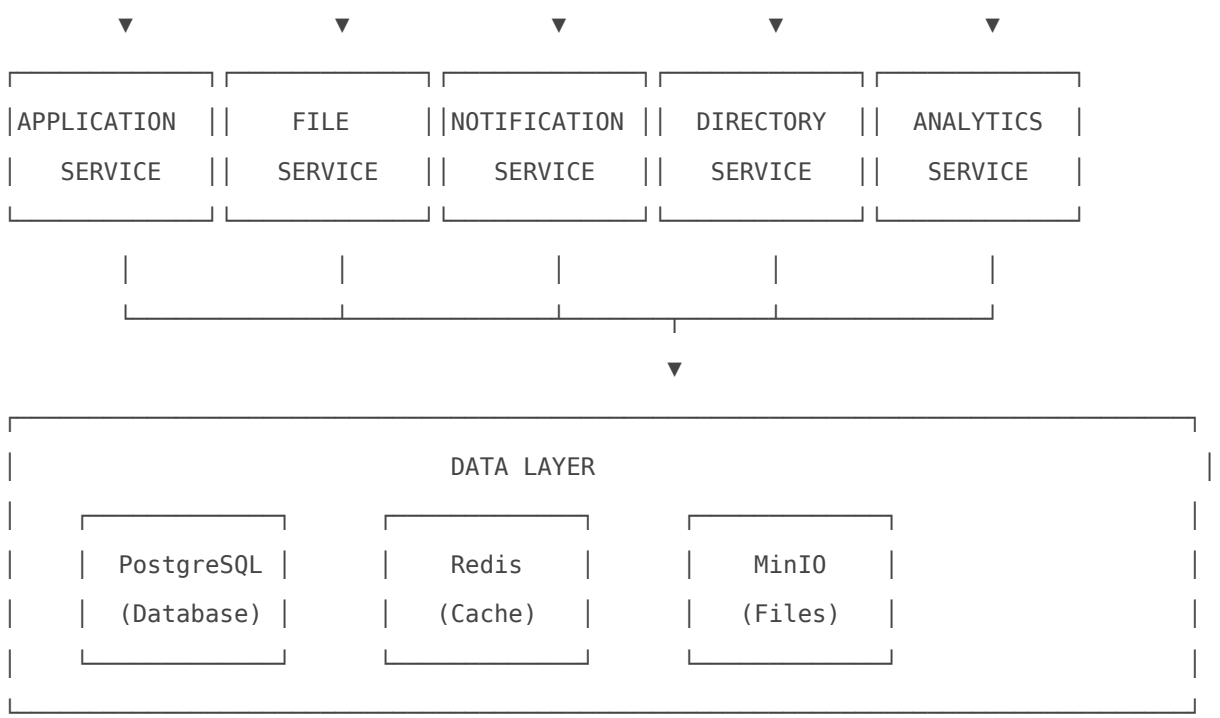
```
User clicks "Sign in with Google/Apple"
        |
        ▼
Redirect to Provider Authorization URL
        |
        ▼
User Authenticates with Provider
        |
```

```
        ▼
Provider redirects back with Authorization Code
        |
        ▼
Backend exchanges Code for Access Token
        |
        ▼
Fetch user profile from Provider
        |
        ▼
Create/Link user account → Issue JWT Tokens
```

3. MICROSERVICES ARCHITECTURE

3.1 High-Level System Diagram





3.2 Services Breakdown

AUTH SERVICE

- User registration and login
- Multi-method authentication (email, phone, username, passwordless, OAuth2)
- JWT token generation and validation
- OTP generation and verification
- Password reset flow
- Account linking

USER SERVICE

- User profile management
- Role management (User, Staff, Admin)
- Account settings
- Privacy settings

SOCIAL SERVICE

- Post creation (text, images, videos, links)
- Likes, comments, shares
- Feed generation
- Content moderation

OPPORTUNITY SERVICE

- Funds management (create, list, search)

- Calls management (startup calls, proposals)
- Business/Tenders management
- Jobs management
- Events management
- Innovation showcase

FORM BUILDER SERVICE

- Dynamic form creation (like Google Forms)
- Field types: text, textarea, dropdown, checkbox, radio, file upload, date, etc.
- Validation rules
- Form templates

APPLICATION SERVICE

- Application submission
- Custom status management (defined by opportunity owner)
- Application data storage
- Export functionality (CSV/Excel)

FILE SERVICE

- File upload/download
- Image processing and optimization
- MinIO integration
- CDN management

NOTIFICATION SERVICE

- Push notifications
- Email notifications (via Glue Email)
- SMS notifications (via Textfy)
- In-app notifications

DIRECTORY SERVICE

- People directory
- Business directory
- Search and filtering
- Profile verification

ANALYTICS SERVICE

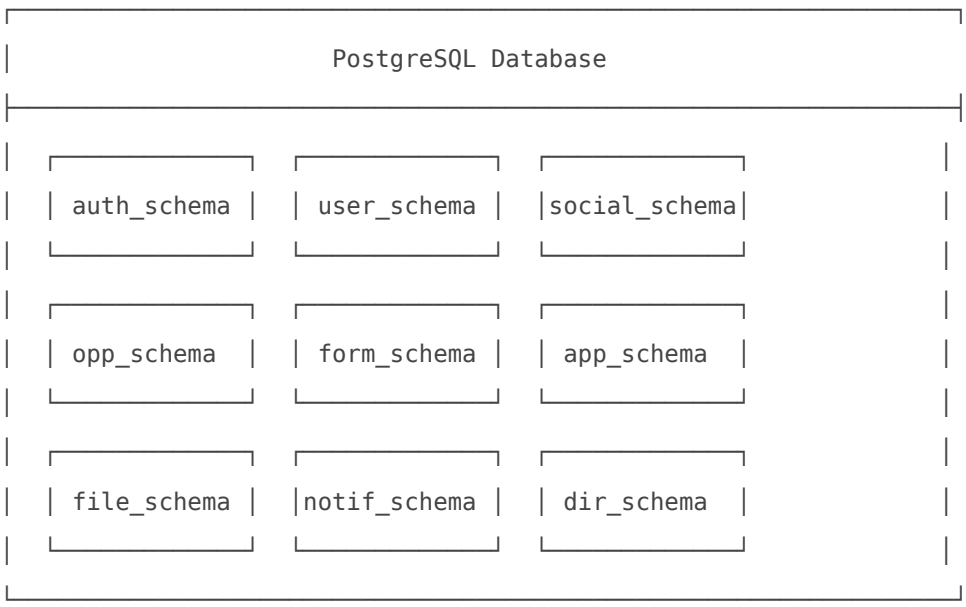
- View counts
- Application statistics
- Basic analytics for opportunity owners

---

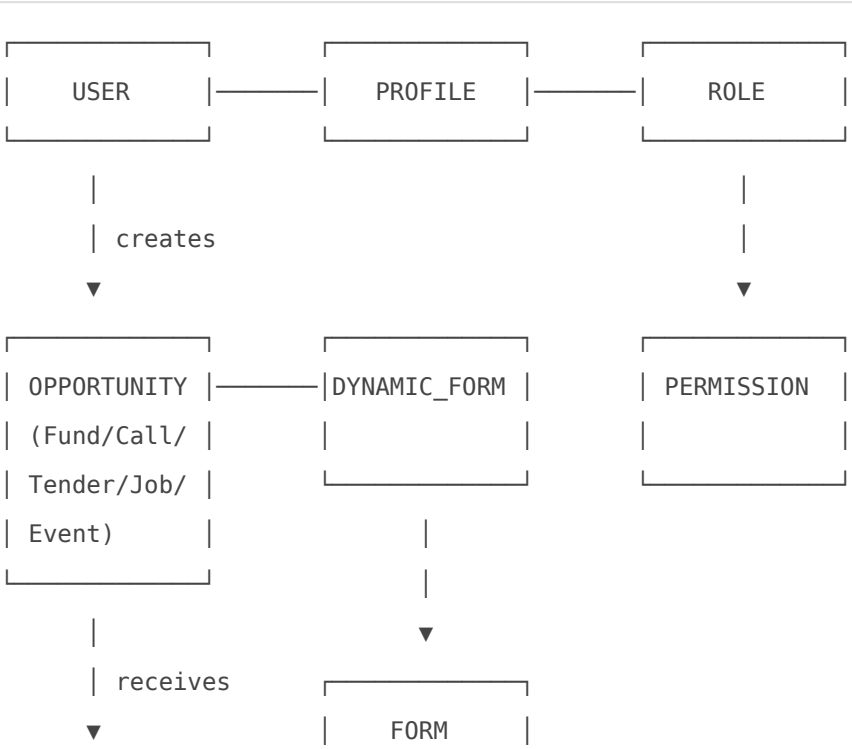
4. DATABASE ARCHITECTURE

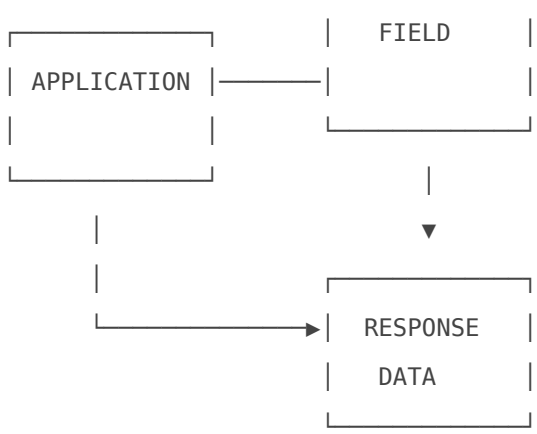
4.1 Database Strategy

For MVP, all microservices share a **single PostgreSQL database** with schema separation per service.



4.2 Core Entity Relationships





4.3 Key Tables

Auth Schema

- `users` - Core user credentials
- `auth_providers` - OAuth2 linked accounts
- `refresh_tokens` - Active refresh tokens
- `otp_requests` - OTP verification tracking

User Schema

- `profiles` - User profile information
- `roles` - System roles (User, Staff, Admin)
- `permissions` - Role permissions

Social Schema

- `posts` - Social media posts
- `comments` - Post comments
- `likes` - Post/comment likes
- `shares` - Post shares

Opportunity Schema

- `opportunities` - Base opportunity table (polymorphic)
- `funds` - Funding opportunities
- `calls` - Startup/proposal calls
- `tenders` - Business tenders
- `jobs` - Job listings
- `events` - Event listings
- `innovations` - Innovation showcase
- `opportunity_statuses` - Custom statuses per opportunity

Form Schema

- `forms` - Form definitions
- `form_fields` - Field definitions
- `field_options` - Dropdown/radio options
- `field_validations` - Validation rules

Application Schema

- `applications` - Submitted applications
- `application_responses` - Form field responses
- `application_files` - Uploaded files
- `application_status_history` - Status change log

Directory Schema

- `businesses` - Business listings
- `business_categories` - Business categorization

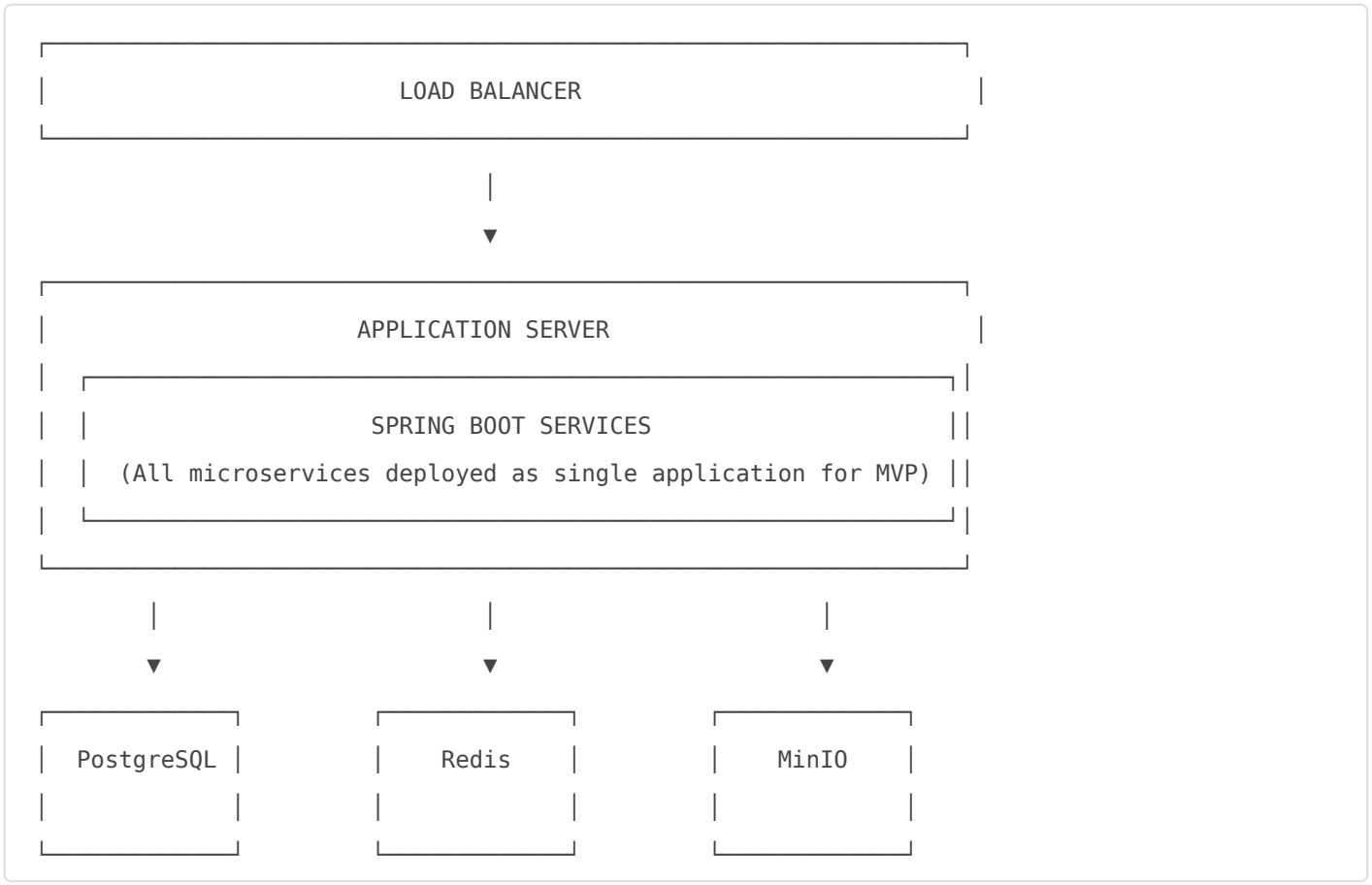
---

5. INFRASTRUCTURE ARCHITECTURE

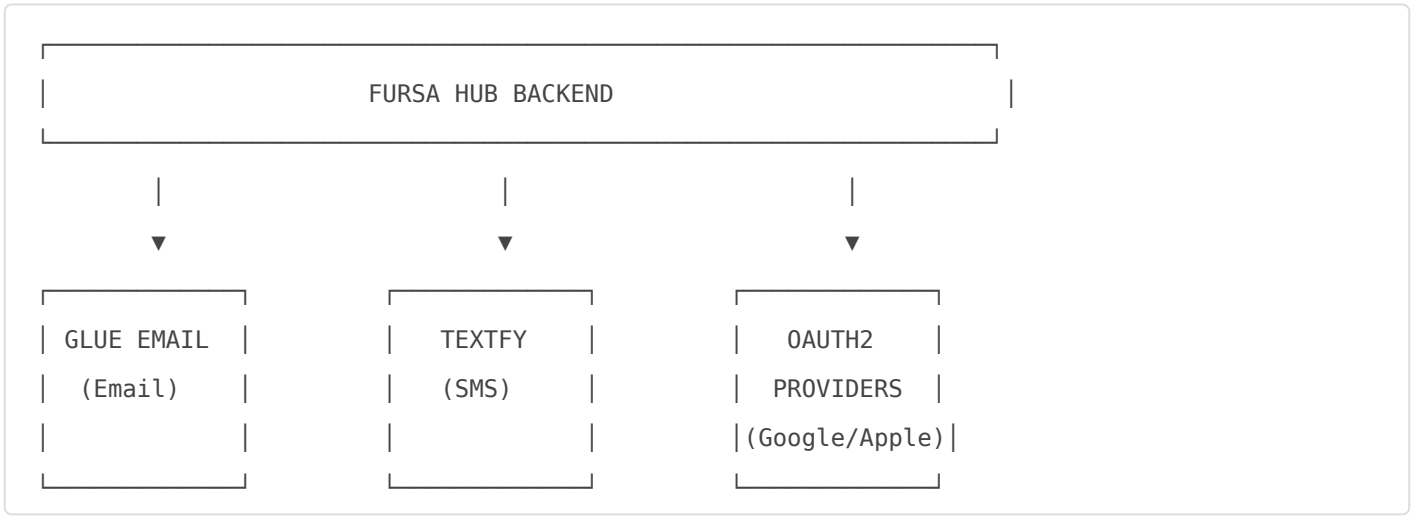
5.1 Technology Stack

Layer	Technology
Backend	Spring Boot (Java)
Database	PostgreSQL
Cache	Redis
File Storage	MinIO
Email	Glue Email
SMS	Textfy
Web Frontend	React
iOS	Swift
Android	Kotlin

5.2 Deployment Architecture (MVP)



5.3 External Service Integration



---

6. API DESIGN

6.1 API Standards

- **Protocol:** REST over HTTPS

- **Format:** JSON
- **Versioning:** URL-based (e.g., `/api/v1/`)
- **Authentication:** Bearer token (JWT)

6.2 API Structure

```
/api/v1
├── /auth
│   ├── POST   /register              # Register new user
│   ├── POST   /login                 # Email/phone/username login
│   ├── POST   /login/otp/request     # Request OTP
│   ├── POST   /login/otp/verify      # Verify OTP
│   ├── POST   /oauth/google          # Google OAuth2
│   ├── POST   /oauth/apple           # Apple OAuth2
│   ├── POST   /token/refresh         # Refresh access token
│   ├── POST   /logout                # Logout
│   └── POST   /password/reset        # Password reset
│
├── /users
│   ├── GET    /me                    # Current user profile
│   ├── PUT    /me                    # Update profile
│   └── GET    /:id                   # Get user by ID
│
├── /social
│   ├── GET    /feed                  # Get feed
│   ├── POST   /posts                 # Create post
│   ├── GET    /posts/:id             # Get post
│   ├── PUT    /posts/:id             # Update post
│   ├── DELETE /posts/:id             # Delete post
│   ├── POST   /posts/:id/like        # Like post
│   ├── DELETE /posts/:id/like        # Unlike post
│   ├── POST   /posts/:id/comments    # Add comment
│   ├── GET    /posts/:id/comments    # Get comments
│   └── POST   /posts/:id/share       # Share post
│
├── /opportunities
│   ├── GET    /funds                 # List funds
│   ├── POST   /funds                 # Create fund
│   ├── GET    /funds/:id             # Get fund details
│   ├── PUT    /funds/:id             # Update fund
```

```
│   ├── DELETE /funds/:id            # Delete fund
│   │
│   ├── GET    /calls                # List calls
│   ├── POST   /calls                # Create call
│   ├── GET    /calls/:id            # Get call details
│   │
│   ├── GET    /tenders              # List tenders
│   ├── POST   /tenders              # Create tender
│   ├── GET    /tenders/:id          # Get tender details
│   │
│   ├── GET    /jobs                 # List jobs
│   ├── POST   /jobs                 # Create job
│   ├── GET    /jobs/:id             # Get job details
│   │
│   ├── GET    /events               # List events
│   ├── POST   /events               # Create event
│   ├── GET    /events/:id           # Get event details
│   │
│   └── GET    /innovations          # List innovations
│
├── /forms
│   ├── POST   /                     # Create dynamic form
│   ├── GET    /:id                  # Get form structure
│   ├── PUT    /:id                  # Update form
│   └── DELETE /:id                  # Delete form
│
├── /applications
│   ├── POST   /                     # Submit application
│   ├── GET    /my                   # My applications
│   ├── GET    /opportunity/:id      # Applications for opportunity
│   ├── GET    /:id                  # Get application details
│   ├── PUT    /:id/status           # Update application status
│   └── GET    /opportunity/:id/export # Export applications (CSV)
│
├── /directory
│   ├── GET    /people               # Search people
│   ├── GET    /businesses           # Search businesses
│   └── GET    /businesses/:id       # Get business details
│
├── /files
```

```
│   ├── POST   /upload               # Upload file
│   └── GET    /:id                  # Get file
│
└── /notifications
    ├── GET    /                     # Get notifications
    ├── PUT    /:id/read             # Mark as read
    └── PUT    /read-all             # Mark all as read
```

6.3 Response Format

Success Response:

```
{
  "success": true,
  "data": { },
  "message": "Operation successful"
}
```

Error Response:

```
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "Human readable message"
  }
}
```

Paginated Response:

```
{
  "success": true,
  "data": [ ],
  "pagination": {
    "page": 1,
    "limit": 20,
    "total": 100,
    "totalPages": 5
  }
}
```

7. DYNAMIC FORM SYSTEM

7.1 Overview

The form builder allows opportunity owners to create custom application forms similar to Google Forms.

7.2 Form Structure

```
{
  "id": "uuid",
  "opportunityId": "uuid",
  "title": "Application Form",
  "description": "Please fill out this form",
  "fields": [
    {
      "id": "uuid",
      "type": "text",
      "label": "Full Name",
      "placeholder": "Enter your full name",
      "required": true,
      "order": 1
    },
    {
      "id": "uuid",
      "type": "dropdown",
      "label": "Experience Level",
      "required": true,
      "options": ["Junior", "Mid-Level", "Senior"],
      "order": 2
    },
    {
      "id": "uuid",
      "type": "file",
      "label": "Upload CV",
      "required": true,
      "allowedTypes": ["pdf", "doc", "docx"],
```

```
      "maxSize": 5242880,
      "order": 3
    }
  ]
}
```

7.3 Supported Field Types

Type	Description
text	Single line text input
textarea	Multi-line text input
number	Numeric input
email	Email input with validation
phone	Phone number input
date	Date picker
datetime	Date and time picker
dropdown	Single select dropdown
multiselect	Multiple select
radio	Radio button group
checkbox	Checkbox group
file	File upload
url	URL input with validation

---

8. APPLICATION STATUS SYSTEM

8.1 Custom Status Flow

Each opportunity owner defines their own status workflow:

```
Owner creates opportunity
        |
        ▼
Owner defines custom statuses
Example: ["Received", "Under Review", "Shortlisted", "Interview", "Accepted", "Rejected"]
```

```
        |
        ▼
Applicants submit applications (default status: first in list)
        |
        ▼
Owner manually updates status as needed
        |
        ▼
Applicant receives notification on status change
```

8.2 Status Configuration

```
{
  "opportunityId": "uuid",
  "statuses": [
    { "id": 1, "name": "Received", "color": "#gray", "isDefault": true },
    { "id": 2, "name": "Under Review", "color": "#blue" },
    { "id": 3, "name": "Shortlisted", "color": "#yellow" },
    { "id": 4, "name": "Accepted", "color": "#green", "isFinal": true },
    { "id": 5, "name": "Rejected", "color": "#red", "isFinal": true }
  ]
}
```

---

9. USER ROLES & PERMISSIONS

9.1 System Roles

Role	Description
User	Regular platform user - can apply and create opportunities
Staff	Fursa Hub team member - helps with platform management
Admin	Full platform access - manages users, content, and settings

9.2 Permission Matrix

Action	User	Staff	Admin
Create opportunities	✓	✓	✓
Apply to opportunities	✓	✓	✓
Manage own content	✓	✓	✓
View all applications	✗	✓	✓
Moderate content	✗	✓	✓
Manage users	✗	✗	✓
System settings	✗	✗	✓

---

10. NOTIFICATION SYSTEM

10.1 Notification Channels

Channel	Provider	Use Case
Push	FCM/APNs	Real-time mobile alerts
Email	Glue Email	Formal communications
SMS	Textfy	OTP, critical alerts
In-App	Internal	Activity feed

10.2 Notification Events

Event	Push	Email	SMS	In-App
New application received	✓	✓	✗	✓
Application status changed	✓	✓	✗	✓
Application submitted (confirmation)	✗	✓	✗	✓
New comment on post	✓	✗	✗	✓
New like on post	✗	✗	✗	✓
OTP verification	✗	✓	✓	✗
Password reset	✗	✓	✗	✗

11. SECURITY CONSIDERATIONS

11.1 Authentication Security

- Password hashing using BCrypt (strength 12)
- JWT tokens with short expiry
- Refresh token rotation
- Rate limiting on auth endpoints
- Account lockout after failed attempts
- OTP expiry (5 minutes)
- OTP attempt limits

11.2 API Security

- HTTPS only
- CORS configuration
- Request validation
- SQL injection prevention (parameterized queries)
- XSS prevention
- Rate limiting per user/IP

11.3 Data Security

- Encryption at rest (database)
- Encryption in transit (TLS)
- File upload validation
- Sensitive data masking in logs

---

12. SCALABILITY PATH

12.1 MVP Phase (Current)

- Monolithic deployment (all services in one application)
- Single PostgreSQL database
- Single Redis instance
- Single MinIO instance

12.2 Growth Phase (Future)

- Split into separate microservices
- Database per service
- Redis cluster
- MinIO cluster or cloud storage (S3)
- Message queue (RabbitMQ/Kafka)
- Kubernetes orchestration

---

14. APPENDIX

14.1 Glossary

Term	Definition
Opportunity	Any postable item (Fund, Call, Tender, Job, Event, Innovation)
Owner	User who creates an opportunity
Applicant	User who applies to an opportunity
Dynamic Form	Custom form created by opportunity owner

14.2 Contact

For technical questions regarding this architecture, please contact the development team.

Revision #5
Created 21 December 2025 13:29:31 by Admin Qbit
Updated 21 December 2025 13:40:06 by Admin Qbit