

Plan de Ejecución Reddit Clone - Desarrollo Integrado

Estrategia de Desarrollo

- **Desarrollo en paralelo:** Backend + Frontend por funcionalidad
 - **Pruebas continuas:** Integración después de cada fase
 - **Iterativo e incremental:** Validar antes de continuar
 - **MVP primero:** Funcionalidad básica funcionando end-to-end
-

FASE 1: Infraestructura Base (Semana 1)

Backend - Fundación

Objetivo: Tener la infraestructura básica funcionando

1.1 Setup Inicial (Días 1-2)

- ☐ **Eureka Server** (Puerto 8761)
 - Configuración básica de Service Discovery
 - Docker compose para bases de datos Oracle
 - Configuración de red entre servicios
- ☐ **API Gateway** (Puerto 8080)
 - Spring Cloud Gateway básico
 - Configuración de CORS
 - Health checks
 - Routing básico preparado

1.2 Base de Datos (Días 3-4)

- ☐ **Docker Compose** para todas las BDs Oracle
 - Scripts de creación de esquemas
 - Configuración de conexiones
 - Flyway setup para migraciones

Frontend - Setup

Objetivo: Estructura base de Angular funcionando

1.3 Proyecto Angular (Días 4-5)

- ☐ **Setup inicial**
 - Angular 17+ con standalone components

- Tailwind CSS configurado
- NgRx store setup básico
- Estructura de carpetas según arquitectura

☐ **Layouts básicos**

- Main layout component
- Auth layout component
- Header component básico
- Routing principal configurado

1.4 Prueba de Integración

- ☐ **Verificación:** Angular puede conectar a API Gateway
 - ☐ **Health check:** Endpoint básico funcionando
 - ☐ **CORS:** Verificar comunicación frontend-backend
-

FASE 2: Autenticación Completa (Semana 2)

Backend - Auth Service

Objetivo: Sistema de autenticación JWT funcionando

2.1 Auth Service (Días 1-3)

- ☐ **Microservicio Auth** (Puerto 8081)
 - Base de datos `auth_db` con tablas de usuarios
 - Endpoints: `/api/auth/login`, `/api/auth/signup`
 - JWT token generation y validation
 - OAuth2 con Google (básico)
 - Password reset functionality
- ☐ **API Gateway Integration**
 - Routing para `/api/auth/**`
 - Security configuration

2.2 User Service Básico (Días 3-4)

- ☐ **Microservicio User** (Puerto 8082)
 - Base de datos `user_db` con perfiles básicos
 - Endpoint: `/api/users/profile` (obtener perfil actual)
 - Comunicación con Auth Service via Feign

Frontend - Auth Module

Objetivo: Login/Signup funcionando con JWT

2.3 Auth Frontend (Días 4-5)

☐ Auth Feature Module

- Login component (según SVG `reddit_login_page`)
- Signup component
- Auth service con JWT handling
- Auth guard para rutas protegidas

☐ NgRx Auth Store

- Actions, reducers, effects para auth
- Manejo de estado de usuario actual
- Token storage en localStorage

2.4 Prueba de Integración

- ☐ **Login/Logout:** Flujo completo funcionando
 - ☐ **JWT:** Token válido y renovación
 - ☐ **Guards:** Protección de rutas funcionando
 - ☐ **User profile:** Datos básicos del usuario
-

FASE 3: Home Feed Básico (Semana 3)

Backend - Posts Service

Objetivo: Crear y listar posts básicos

3.1 Post Service (Días 1-3)

☐ Microservicio Post (Puerto 8084)

- Base de datos `post_db` con posts básicos
- Endpoints:
 - `GET /api/posts` (lista paginada)
 - `POST /api/posts` (crear post)
 - `GET /api/posts/{id}` (post individual)
- Solo posts de texto por ahora

3.2 Vote Service Básico (Días 3-4)

☐ Microservicio Vote (Puerto 8086)

- Base de datos `vote_db`
- Endpoints:

- `POST /api/votes` (votar post)
- `GET /api/votes/posts/{id}` (obtener votos)
- Cálculo básico de score

Frontend - Home & Posts

Objetivo: Feed de posts con voting funcionando

3.3 Home Module (Días 4-5)

☐ Home Feature Module

- Home feed component (según SVG `reddit_home_design`)
- Post card component reutilizable
- Vote buttons component
- Infinite scroll básico

☐ NgRx Post Store

- Actions y reducers para posts
- Effects para cargar posts
- Estado de paginación

3.4 Create Post Básico (Día 5)

☐ Create Post Component

- Formulario básico (solo texto)
- Validaciones
- Navegación después de crear

3.5 Prueba de Integración

- ☐ **CRUD Posts:** Crear y listar posts
- ☐ **Voting:** Upvote/downvote funcionando
- ☐ **Paginación:** Cargar más posts
- ☐ **Real-time:** Votos se actualizan



FASE 4: Comunidades Básicas (Semana 4)

Backend - Community Service

Objetivo: Crear y gestionar comunidades

4.1 Community Service (Días 1-3)

- ☐ **Microservicio Community** (Puerto 8083)

- Base de datos `community_db`
- Endpoints:
 - `GET /api/communities` (listar comunidades)
 - `POST /api/communities` (crear comunidad)
 - `GET /api/communities/{name}` (detalles comunidad)
 - `POST /api/communities/{name}/join` (unirse)

4.2 Posts + Communities Integration (Días 3-4)

☐ Modificar Post Service

- Posts asociados a comunidades
- Endpoint: `GET /api/communities/{name}/posts`
- Validar permisos de posting

Frontend - Community Module

Objetivo: Navegación y gestión de comunidades

4.3 Community Frontend (Días 4-5)

☐ Community Feature Module

- Community page component (según SVG `reddit_community_page`)
- Community header y sidebar
- Community posts list
- Join/Leave functionality

4.4 Navigation Update (Día 5)

☐ Sidebar Component

- Lista de comunidades suscritas
- Navegación entre comunidades
- Crear nueva comunidad

4.5 Prueba de Integración

- ☐ **Communities:** Crear, unirse, ver posts
- ☐ **Posts in communities:** Crear posts en comunidades específicas
- ☐ **Navigation:** Navegar entre home y comunidades

FASE 5: Perfiles de Usuario (Semana 5)

Backend - User Service Completo

Objetivo: Perfiles completos y estadísticas

5.1 User Service Enhancement (Días 1-3)

☐ Expandir User Service

- Endpoints para perfil completo
- User stats (karma, posts count, etc.)
- Edición de perfil
- Avatar upload (integración con Cloudinary)

5.2 User Posts & Stats (Días 3-4)

☐ Posts by User

- Endpoint: `GET /api/users/{username}/posts`
- User karma calculation
- Integration con Vote Service

Frontend - User Profile

Objetivo: Perfiles de usuario completos

5.3 User Module (Días 4-5)

☐ User Feature Module

- User profile component (según SVG `reddit_profile_page`)
- Profile navigation (posts, comments, etc.)
- Profile stats component
- Edit profile functionality

5.4 Prueba de Integración

- ☐ **User profiles:** Ver perfiles completos
 - ☐ **User posts:** Posts del usuario
 - ☐ **Stats:** Karma y estadísticas correctas
-

FASE 6: Sistema de Comentarios (Semana 6)

Backend - Comment Service

Objetivo: Comentarios anidados funcionando

6.1 Comment Service (Días 1-4)

- ☐ **Microservicio Comment** (Puerto 8085)

- Base de datos `comment_db`
- Endpoints:
 - `GET /api/posts/{id}/comments` (comentarios del post)
 - `POST /api/comments` (crear comentario)
 - `POST /api/comments/{id}/reply` (responder comentario)
- Threading system para comentarios anidados

6.2 Comments + Votes Integration (Días 4-5)

☐ Vote Service Update

- Votar comentarios
- Karma por comentarios

Frontend - Comments System

Objetivo: Interfaz de comentarios anidados

6.3 Comment Module (Días 4-5)

☐ Comment Feature Module

- Comment tree component
- Comment item component (anidado)
- Comment form component
- Vote buttons para comentarios

6.4 Post Detail Enhancement (Día 5)

☐ Post Detail Update

- Integrar comment section
- Actualizar según SVG `reddit_post_detail`

6.5 Prueba de Integración

- ☐ **Comments:** Crear y mostrar comentarios
- ☐ **Nested comments:** Threading funcionando
- ☐ **Comment voting:** Votos en comentarios

FASE 7: Notificaciones (Semana 7)

Backend - Notification Service

Objetivo: Sistema de notificaciones básico

7.1 Notification Service (Días 1-4)

☐ **Microservicio Notification** (Puerto 8087)

- Base de datos `notification_db`
- WebSocket configuration
- Email notifications (SendGrid)
- Notification triggers

7.2 Real-time Integration (Días 4-5)

☐ **WebSocket Setup**

- Real-time notifications
- Integration con otros servicios

Frontend - Notifications

Objetivo: Notificaciones en tiempo real

7.3 Notifications Frontend (Días 4-5)

☐ **Notification System**

- WebSocket service
- Notification dropdown
- Real-time updates

7.4 Prueba de Integración

☐ **Real-time:** Notificaciones en tiempo real

☐ **Email:** Notificaciones por email

FASE 8: Polish & Features Avanzadas (Semana 8)

Backend - Features Avanzadas

8.1 Advanced Features (Días 1-3)

- ☐ **Image Upload:** Posts con imágenes
- ☐ **Search:** Elasticsearch integration
- ☐ **Moderation:** Herramientas básicas

Frontend - Polish

8.2 Advanced Frontend (Días 3-5)

- ☐ **Settings Module:** Configuraciones completas
- ☐ **Mobile Responsive:** Optimización móvil
- ☐ **PWA:** Progressive Web App features

- ☐ **Performance:** Lazy loading, optimizaciones

8.3 Testing & Documentation (Día 5)

- ☐ **Testing:** Unit tests críticos
 - ☐ **Documentation:** README y deployment guides
-

Herramientas de Desarrollo

Monitoreo y Testing

- **Postman Collections:** Para testing de APIs
- **Docker Compose:** Desarrollo local completo
- **GitHub Actions:** CI/CD básico
- **Swagger:** Documentación automática de APIs

Base de Datos

- **Flyway:** Migraciones versionadas
- **Oracle DB:** Una instancia por microservicio
- **Redis:** Cache para sesiones y datos frecuentes

Desarrollo

- **Hot Reload:** Angular dev server + Spring Boot DevTools
 - **Logs Centralizados:** ELK Stack básico
 - **Environment Variables:** Configuración por ambiente
-

Métricas de Éxito por Fase

Fase 1-2: ☒ **Auth funcionando end-to-end**

Fase 3: ☒ **Posts con voting funcionando**

Fase 4: ☒ **Comunidades básicas operativas**

Fase 5: ☒ **Perfiles de usuario completos**

Fase 6: ☒ **Sistema de comentarios anidados**

Fase 7: ☒ **Notificaciones en tiempo real**

Fase 8: ☒ **Aplicación completa y optimizada**

¿Te parece bien este plan? ¿Empezamos con la Fase 1 o quieres ajustar algo?

