# 🏗️ Reddit Clone - Arquitectura de Microservicios

## 🎯 Decisión Arquitectónica: Microservicios

**Justificación**: Para portafolio profesional que demuestre conocimientos avanzados de arquitectura distribuida.

---

## 🗄️ Base de Datos: Oracle Database

```sql
-- Cada microservicio tendrá su propia base de datos Oracle
- Oracle Database 21c XE por microservicio
- Spring Data JPA + Hibernate
- HikariCP Connection Pool
- Flyway para migraciones por servicio
```

---

## 🚀 Microservicios Identificados (8 servicios)

### 1. API Gateway Service 🌐

```
Port: 8080
Database: Ninguna (stateless)
Responsabilidades:
- Punto de entrada único
- Enrutamiento de requests
- Rate limiting
- Load balancing
- Cors configuration
- Request/Response logging
```

### 2. Auth Service 🔐

```
Port: 8081
Database: auth_db (Oracle)
Responsabilidades:
- Autenticación (login/signup)
- Autorización (JWT tokens)
- OAuth2 (Google)
- Password reset
- Refresh tokens
- User roles y permissions
```

**Estructura Auth Service:**

```
auth-service/
├── src/main/java/com/redditclone/auth/
│   ├── AuthServiceApplication.java
│   ├── config/
│   │   ├── SecurityConfig.java
│   │   ├── JwtConfig.java
│   │   └── OAuth2Config.java
│   ├── controller/
│   │   ├── AuthController.java
│   │   └── OAuth2Controller.java
│   ├── service/
│   │   ├── AuthService.java
│   │   ├── JwtService.java
│   │   ├── RefreshTokenService.java
│   │   └── OAuth2Service.java
│   ├── model/
│   │   ├── User.java
│   │   ├── RefreshToken.java
│   │   ├── UserRole.java
│   │   └── OAuth2UserInfo.java
│   ├── dto/
│   │   ├── LoginRequest.java
│   │   ├── SignupRequest.java
│   │   ├── AuthResponse.java
│   │   ├── RefreshTokenRequest.java
│   │   └── PasswordResetRequest.java
│   ├── repository/
│   │   ├── UserRepository.java
│   │   └── RefreshTokenRepository.java
│   └── security/
│       ├── JwtAuthenticationFilter.java
│       ├── JwtAuthenticationEntryPoint.java
│       └── CustomUserDetailsService.java
│
```

## 3. User Service 🧑🏿

Port: 8082
Database: user_db (Oracle)
Responsabilidades:
- Gestión de perfiles de usuario
- Configuraciones de usuario
- Avatar y banner upload
- User stats (karma, awards)
- Followers/Following
- User preferences

**Estructura User Service:**

```
user-service/
├── src/main/java/com/redditclone/user/
│   ├── UserServiceApplication.java
│   ├── controller/
│   │   ├── UserController.java
│   │   ├── UserProfileController.java
│   │   ├── UserSettingsController.java
│   │   └── UserStatsController.java
│   ├── service/
│   │   ├── UserService.java
│   │   ├── UserProfileService.java
│   │   ├── UserSettingsService.java
│   │   ├── AvatarService.java
│   │   └── UserStatsService.java
│   ├── model/
│   │   ├── User.java
│   │   ├── UserProfile.java
│   │   ├── UserSettings.java
│   │   ├── NotificationSettings.java
│   │   ├── UserStats.java
│   │   └── UserFollow.java
│   ├── dto/
│   │   ├── UserDTO.java
│   │   ├── UserProfileDTO.java
│   │   ├── UserStatsDTO.java
│   │   ├── UpdateProfileRequest.java
│   │   ├── UserSettingsDTO.java
│   │   └── AvatarUploadResponse.java
│   ├── repository/
│   │   ├── UserRepository.java
│   │   ├── UserProfileRepository.java
│   │   ├── UserSettingsRepository.java
│   │   ├── UserStatsRepository.java
│   │   └── UserFollowRepository.java
│   └── client/
│       └── AuthServiceClient.java
│
```

## 4. Community Service 🏘️

```
Port: 8083
Database: community_db (Oracle)
Responsabilidades:
- Gestión de comunidades/subreddits
- Memberships y roles
- Community settings y rules
- Moderator management
- Community stats
- Community discovery
```

**Estructura Community Service:**

```
community-service/
├── src/main/java/com/redditclone/community/
│   ├── CommunityServiceApplication.java
│   ├── controller/
│   │   ├── CommunityController.java
│   │   ├── CommunityMembershipController.java
│   │   ├── CommunityModerationController.java
│   │   └── CommunityRulesController.java
│   ├── service/
│   │   ├── CommunityService.java
│   │   ├── CommunityMembershipService.java
│   │   ├── CommunityModerationService.java
│   │   ├── CommunityRulesService.java
│   │   └── CommunityStatsService.java
│   ├── model/
│   │   ├── Community.java
│   │   ├── CommunityMembership.java
│   │   ├── CommunityRule.java
│   │   ├── CommunityModerator.java
│   │   ├── CommunityStats.java
│   │   └── CommunitySettings.java
│   ├── dto/
│   │   ├── CommunityDTO.java
│   │   ├── CreateCommunityRequest.java
│   │   ├── CommunityStatsDTO.java
│   │   ├── CommunityRuleDTO.java
│   │   ├── CommunityMembershipDTO.java
│   │   └── ModeratorDTO.java
│   ├── repository/
│   │   ├── CommunityRepository.java
│   │   ├── CommunityMembershipRepository.java
│   │   ├── CommunityRuleRepository.java
│   │   ├── CommunityModeratorRepository.java
│   │   └── CommunityStatsRepository.java
│   └── client/
│       ├── UserServiceClient.java
│       └── AuthServiceClient.java
│
```

## 5. Post Service 📝

```
Port: 8084
Database: post_db (Oracle)
Responsabilidades:
- Gestión de posts
- Post content (text, image, link)
- Post metadata
- Post categories/flairs
- Draft management
- Post search
```

**Estructura Post Service:**

```
post-service/
├── src/main/java/com/redditclone/post/
│   ├── PostServiceApplication.java
│   ├── controller/
│   │   ├── PostController.java
│   │   ├── PostDraftController.java
│   │   ├── PostSearchController.java
│   │   └── PostFlairController.java
│   ├── service/
│   │   ├── PostService.java
│   │   ├── PostDraftService.java
│   │   ├── PostSearchService.java
│   │   ├── PostFlairService.java
│   │   └── ImageUploadService.java
│   ├── model/
│   │   ├── Post.java
│   │   ├── PostContent.java
│   │   ├── PostFlair.java
│   │   ├── PostDraft.java
│   │   ├── PostImage.java
│   │   └── PostType.java (enum)
│   ├── dto/
│   │   ├── PostDTO.java
│   │   ├── CreatePostRequest.java
│   │   ├── UpdatePostRequest.java
│   │   ├── PostDraftDTO.java
│   │   ├── PostFlairDTO.java
│   │   └── ImageUploadResponse.java
│   ├── repository/
│   │   ├── PostRepository.java
│   │   ├── PostDraftRepository.java
│   │   ├── PostFlairRepository.java
│   │   └── PostImageRepository.java
│   └── client/
│       ├── UserServiceClient.java
│       ├── CommunityServiceClient.java
│       └── AuthServiceClient.java
│
```

## 6. Comment Service 💬

```
Port: 8085
Database: comment_db (Oracle)
Responsabilidades:
- Gestión de comentarios
- Sistema de comentarios anidados
- Reply management
- Comment threading
- Comment moderation
```

**Estructura Comment Service:**

```
comment-service/
├── src/main/java/com/redditclone/comment/
│   ├── CommentServiceApplication.java
│   ├── controller/
│   │   ├── CommentController.java
│   │   ├── CommentThreadController.java
│   │   └── CommentModerationController.java
│   ├── service/
│   │   ├── CommentService.java
│   │   ├── CommentThreadService.java
│   │   ├── CommentModerationService.java
│   │   └── CommentTreeService.java
│   ├── model/
│   │   ├── Comment.java
│   │   ├── CommentThread.java
│   │   ├── CommentModeration.java
│   │   └── CommentStatus.java (enum)
│   ├── dto/
│   │   ├── CommentDTO.java
│   │   ├── CreateCommentRequest.java
│   │   ├── UpdateCommentRequest.java
│   │   ├── CommentThreadDTO.java
│   │   └── CommentTreeDTO.java
│   ├── repository/
│   │   ├── CommentRepository.java
│   │   ├── CommentThreadRepository.java
│   │   └── CommentModerationRepository.java
│   └── client/
│       ├── PostServiceClient.java
│       ├── UserServiceClient.java
│       └── AuthServiceClient.java
│
```

## 7. Vote Service ⬆️ ⬇️

```
Port: 8086
Database: vote_db (Oracle)
Responsabilidades:
- Sistema de voting (upvote/downvote)
- Vote calculations
- Karma management
- Vote history
- Vote analytics
```

**Estructura Vote Service:**

```
vote-service/
├── src/main/java/com/redditclone/vote/
│   ├── VoteServiceApplication.java
│   ├── controller/
│   │   ├── VoteController.java
│   │   ├── KarmaController.java
│   │   └── VoteAnalyticsController.java
│   ├── service/
│   │   ├── VoteService.java
│   │   ├── KarmaService.java
│   │   ├── VoteCalculationService.java
│   │   └── VoteAnalyticsService.java
│   ├── model/
│   │   ├── Vote.java
│   │   ├── VoteType.java (enum)
│   │   ├── Karma.java
│   │   └── VoteHistory.java
│   ├── dto/
│   │   ├── VoteDTO.java
│   │   ├── VoteRequest.java
│   │   ├── KarmaDTO.java
│   │   └── VoteStatsDTO.java
│   ├── repository/
│   │   ├── VoteRepository.java
│   │   ├── KarmaRepository.java
│   │   └── VoteHistoryRepository.java
│   └── client/
│       ├── PostServiceClient.java
│       ├── CommentServiceClient.java
│       ├── UserServiceClient.java
│       └── AuthServiceClient.java
│
```

## 8. Notification Service 🔔

Port: 8087
Database: notification_db (Oracle)
Responsabilidades:
- Sistema de notificaciones
- Email notifications
- Push notifications
- Notification preferences
- Real-time notifications (WebSocket)
- Notification history

**Estructura Notification Service:**

```
notification-service/
├── src/main/java/com/redditclone/notification/
│   ├── NotificationServiceApplication.java
│   ├── controller/
│   │   ├── NotificationController.java
│   │   ├── NotificationPreferencesController.java
│   │   └── WebSocketController.java
│   ├── service/
│   │   ├── NotificationService.java
│   │   ├── EmailNotificationService.java
│   │   ├── PushNotificationService.java
│   │   ├── WebSocketNotificationService.java
│   │   └── NotificationPreferencesService.java
│   ├── model/
│   │   ├── Notification.java
│   │   ├── NotificationType.java (enum)
│   │   ├── NotificationPreferences.java
│   │   ├── EmailTemplate.java
│   │   └── NotificationStatus.java (enum)
│   ├── dto/
│   │   ├── NotificationDTO.java
│   │   ├── CreateNotificationRequest.java
│   │   ├── NotificationPreferencesDTO.java
│   │   └── EmailNotificationDTO.java
│   ├── repository/
│   │   ├── NotificationRepository.java
│   │   ├── NotificationPreferencesRepository.java
│   │   └── EmailTemplateRepository.java
│   ├── websocket/
│   │   ├── WebSocketConfig.java
│   │   ├── NotificationWebSocketHandler.java
│   │   └── WebSocketSessionManager.java
│   └── client/
│       ├── UserServiceClient.java
│       ├── PostServiceClient.java
│       ├── CommentServiceClient.java
│       └── AuthServiceClient.java
│
```

---

## 🔄 Comunicación Entre Microservicios

### Service Discovery

yaml

```yaml
# Eureka Server
eureka-server:
  port: 8761

# Cada microservicio se registra en Eureka
spring:
  application:
    name: auth-service
  eureka:
    client:
      service-url:
        defaultZone: http://localhost:8761/eureka
```

## API Gateway Routes

yaml

```yaml
spring:
  cloud:
    gateway:
      routes:
        - id: auth-service
          uri: lb://auth-service
          predicates:
            - Path=/api/auth/**
        - id: user-service
          uri: lb://user-service
          predicates:
            - Path=/api/users/**
        - id: community-service
          uri: lb://community-service
          predicates:
            - Path=/api/communities/**
        - id: post-service
          uri: lb://post-service
          predicates:
            - Path=/api/posts/**
        - id: comment-service
          uri: lb://comment-service
          predicates:
            - Path=/api/comments/**
        - id: vote-service
          uri: lb://vote-service
          predicates:
            - Path=/api/votes/**
        - id: notification-service
          uri: lb://notification-service
          predicates:
            - Path=/api/notifications/**
```

---

## 🗄️ Esquema de Base de Datos por Microservicio

### Auth DB (auth_db)

sql

```sql
TABLES:
- users (id, username, email, password_hash, created_at, updated_at)
- refresh_tokens (id, user_id, token_hash, expires_at, created_at)
- oauth2_users (id, user_id, provider, provider_id, created_at)
- user_roles (id, user_id, role_name, created_at)
```

## User DB (user_db)

sql

```sql
TABLES:
- user_profiles (id, user_id, display_name, bio, location, avatar_url, banner_url, created_at, updated_at)
- user_settings (id, user_id, theme, language, timezone, privacy_settings, created_at, updated_at)
- notification_settings (id, user_id, email_notifications, push_notifications, frequency, dnd_start, dnd_end)
- user_stats (id, user_id, post_karma, comment_karma, total_posts, total_comments, awards_received)
- user_follows (id, follower_id, following_id, created_at)
```

## Community DB (community_db)

sql

```sql
TABLES:
- communities (id, name, display_name, description, creator_id, member_count, created_at, updated_at)
- community_memberships (id, community_id, user_id, role, joined_at)
- community_rules (id, community_id, rule_number, title, description, created_at)
- community_moderators (id, community_id, user_id, permissions, appointed_at)
- community_stats (id, community_id, total_posts, total_comments, active_users, created_at)
- community_settings (id, community_id, is_private, allow_images, allow_videos, require_approval)
```

## Post DB (post_db)

sql

```sql
TABLES:
- posts (id, title, content, post_type, author_id, community_id, vote_count, comment_count, created_at, updated_at)
- post_content (id, post_id, content_type, content_url, content_text)
- post_flairs (id, community_id, name, color, background_color, created_at)
- post_flair_assignments (id, post_id, flair_id, assigned_at)
- post_drafts (id, user_id, title, content, community_id, created_at, updated_at)
- post_images (id, post_id, image_url, image_order, uploaded_at)
```

## Comment DB (comment_db)

sql

```sql
TABLES:
- comments (id, content, author_id, post_id, parent_comment_id, vote_count, depth, created_at, updated_at)
- comment_threads (id, post_id, root_comment_id, total_comments, created_at)
- comment_moderation (id, comment_id, moderator_id, action, reason, created_at)
```

## Vote DB (vote_db)

```sql
TABLES:
- votes (id, user_id, target_id, target_type, vote_type, created_at, updated_at)
- karma_history (id, user_id, karma_type, points_change, reason, created_at)
- vote_aggregations (id, target_id, target_type, upvotes, downvotes, score, updated_at)
```

### Notification DB (notification_db)

```sql
TABLES:
- notifications (id, user_id, type, title, message, read_status, created_at)
- notification_preferences (id, user_id, email_enabled, push_enabled, frequency, dnd_start, dnd_end)
- email_templates (id, template_name, subject, html_content, text_content, created_at)
- notification_history (id, notification_id, delivery_method, status, sent_at)
```

---

## 🚀 Tecnologías por Microservicio

### Comunes a todos:

- **Spring Boot 3.2**
- **Spring Data JPA**
- **Oracle Database 21c XE**
- **Flyway** (migraciones)
- **Docker** (containerización)
- **Maven** (build tool)

### Específicas:

- **API Gateway**: Spring Cloud Gateway, Eureka Client
- **Auth Service**: Spring Security, JWT, OAuth2
- **User Service**: Cloudinary (images), Redis (cache)
- **Community Service**: Redis (cache)
- **Post Service**: Cloudinary (images), Elasticsearch (search)
- **Comment Service**: Redis (cache para threading)
- **Vote Service**: Redis (cache para calculations)
- **Notification Service**: Spring WebSocket, SendGrid (email)

---

## 📋 Plan de Desarrollo por Fases

### Fase 1: Core Services (Semanas 1-4)

1. **API Gateway + Eureka Server**
2. **Auth Service** (login/signup/JWT)
3. **User Service** (basic profile)
4. **Configuración Oracle DB** para todos

### Fase 2: Content Services (Semanas 5-8)

5. **Community Service** (crear/unirse comunidades)
6. **Post Service** (crear/listar posts)
7. **Comment Service** (comentarios básicos)

### Fase 3: Engagement Services (Semanas 9-12)

8. **Vote Service** (upvote/downvote)
9. **Notification Service** (notificaciones básicas)
10. **Integración completa entre servicios**

### Fase 4: Advanced Features (Semanas 13-16)

11. **Real-time features** (WebSocket)
12. **Search functionality** (Elasticsearch)
13. **Image upload** (Cloudinary)
14. **Testing & Documentation**

¿Te parece bien esta arquitectura de microservicios? ¿Quieres que detalle algún servicio específico o ajuste algo?