Plan de Migración: Portal RRHH Zabala Gailetak

📋 Resumen Ejecutivo

Proyecto: Sistema interno de gestión de recursos humanos para Zabala Gailetak
Scope: Portal web + App Android para gestión de empleados
Tecnología actual: Node.js/Express + React + React Native + MongoDB
Tecnología objetivo: PHP Vanilla + Kotlin Android + PostgreSQL
Deadline: Diciembre 2026 (aproximadamente 11 meses)
Versión: 1.0
Fecha: 14 de Enero de 2026

🎯 Alcance del Proyecto

1.1 Contextualización del Cambio

El proyecto originally diseñado como una plataforma de comercio electrónico (e-commerce) para la venta de productos Zabala Gailetak ha cambiado completamente de dirección. El nuevo sistema será un **portal interno de gestión de recursos humanos** destinado exclusivamente a los trabajadores de la empresa, permitiendo la gestión completa del ciclo de vida del empleado dentro de la organización.

Este cambio de scope implica una reorganización completa de las prioridades de desarrollo, donde las funcionalidades orientadas al cliente externo se reemplazan por herramientas de gestión interna, comunicación empresarial y administración de personal.

1.2 Funcionalidades Principales

Módulo	Descripción	Prioridad	Web	Android
Gestión de Empleados	Altas, bajas, modificaciones, consulta de datos	Alta	✅	✅
Autenticación	Credenciales + MFA + Passkey	Alta	✅	✅
Nóminas	Historial de nóminas, consultas de pagos	Alta	✅	✅
Vacaciones	Días disponibles, solicitudes, calendario	Alta	✅	✅

Módulo	Descripción	Prioridad	Web	Android
Documentos	Solicitudes de documentación, subida/bajada de archivos	Alta	✅	✅
Quejas/Sugerencias	Sistema de feedback interno	Media	✅	✅
Chat RRHH	Comunicación con departamento de RRHH	Media	✅	✅
Chat por Sección	Chat entre compañeros del mismo departamento	Media	✅	✅
Gestión de Credenciales	Gestión de accesos y permisos por rol	Media	✅	✅
Auditoría	Logs de todas las acciones para compliance	Media	✅	❌

1.3 Funcionalidades Adicionales Recomendadas

Módulo	Descripción	Impacto	Fase
Roles y Permisos	Sistema RBAC: Admin, RRHH, Jefe de Sección, Empleado	Alto	3
Perfil de Empleado	Datos personales, contacto de emergencia	Medio	3
Políticas Corporativas	Documentos de empresa, normativas internas	Bajo	9
Sistema de Notificaciones	Alertas de solicitudes, recordatorios	Medio	4
Evaluaciones de Desempeño	Evaluaciones periódicas de empleados	Bajo	9
Ticketing IT	Sistema de soporte técnico interno	Bajo	9

1.4 Funcionalidades Eliminadas del Scope Original

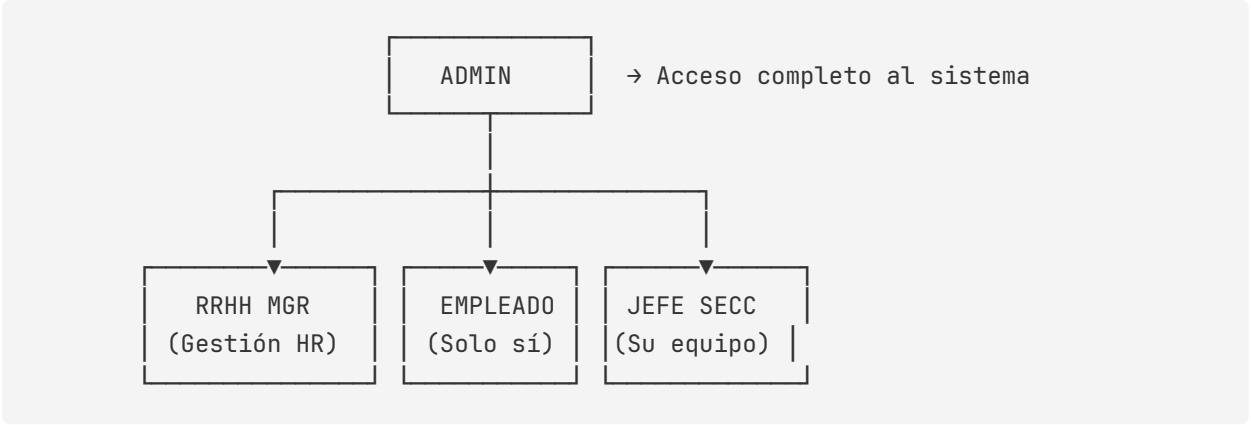
Dado el cambio de orientación de e-commerce a portal RRHH interno, las siguientes funcionalidades quedan fuera del scope del proyecto:

- Catálogo de productos
- Carrito de compras
- Procesamiento de pagos externos
- Sistema de valoraciones y reseñas de clientes
- Gestión de inventario para venta

- Seguimiento de pedidos de clientes
- Descuentos y promociones
- Integración con pasarelas de pago externas

👥 Roles de Usuario y Permisos

2.1 Jerarquía de Roles



2.2 Descripción de Roles

ADMIN (Administrador del Sistema)

- Acceso completo a todas las funcionalidades del sistema
- Gestión de usuarios y roles
- Configuración del sistema
- Visualización de auditoría completa
- Creación y eliminación de cualquier registro

RRHH MGR (Responsable de Recursos Humanos)

- Gestión completa de empleados (alta, baja, modificación)
- Aprobación de vacaciones de todos los departamentos
- Visualización de nóminas de todos los empleados
- Gestión documental completa
- Acceso a quejas y sugerencias
- Chat con cualquier empleado
- Generación de reportes y estadísticas

JEFE DE SECCIÓN (Departamento)

- Visualización de empleados de su sección

- Aprobación de vacaciones de su equipo
- Chat con su sección
- Acceso a datos básicos de su equipo
- Solicitud de documentos a empleados

EMPLEADO (Usuario Estándar)

- Acceso únicamente a sus propios datos
- Solicitud de vacaciones
- Consulta de sus nóminas
- Subida y descarga de sus documentos
- Participación en chat de sección y chat con RRHH
- Creación de quejas y sugerencias

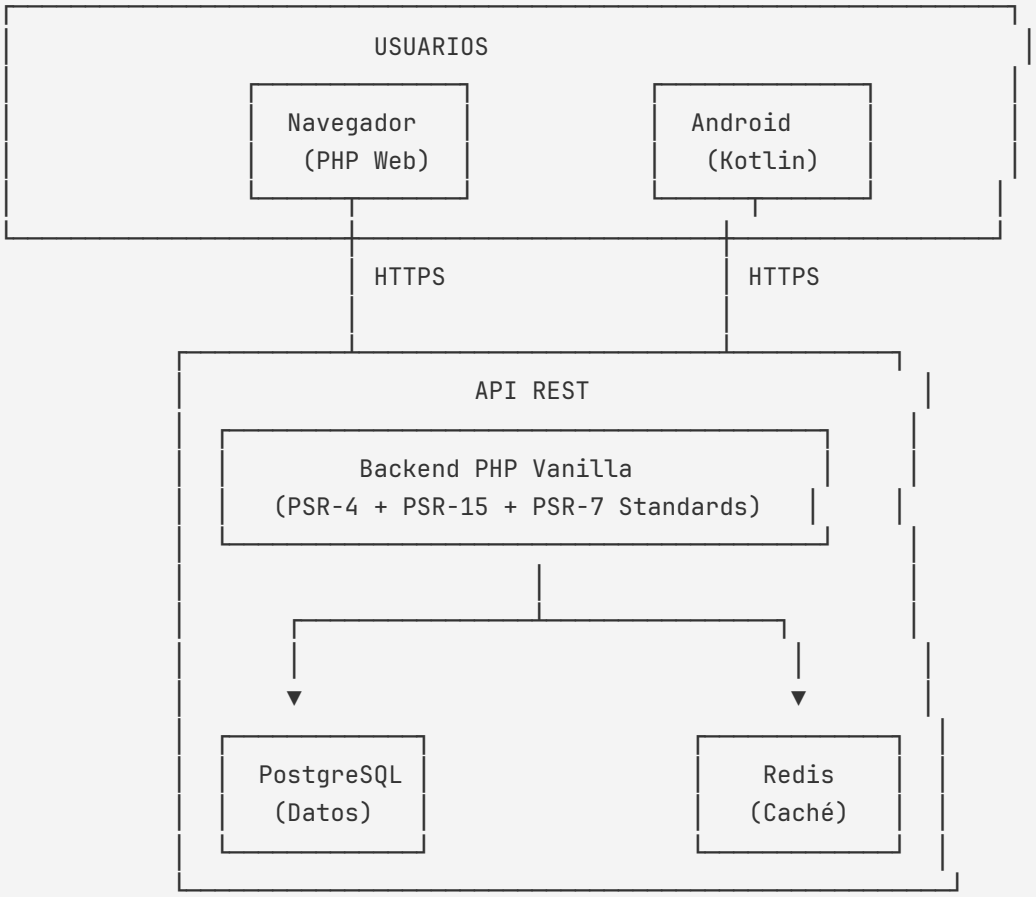
2.3 Matriz de Permisos

Permiso	Admin	RRHR Mgr	Jefe Secc	Empleado
Gestión de Empleados				
Crear empleado	✅	✅	❌	❌
Editar empleado	✅	✅	❌	❌
Eliminar empleado	✅	❌	❌	❌
Ver todos los empleados	✅	✅	❌	❌
Ver empleados de sección	✅	✅	✅	❌
Ver propio perfil	✅	✅	✅	✅
Editar propio perfil	✅	✅	✅	✅
Gestión de Nóminas				
Ver nóminas de todos	✅	✅	❌	❌
Ver nóminas de sección	✅	✅	✅	❌
Ver propia nómina	✅	✅	✅	✅
Descargar nóminas PDF	✅	✅	✅	✅
Vacaciones				
Solicitar vacaciones	✅	✅	✅	✅

Permiso	Admin	RRHR Mgr	Jefe Secc	Empleado
Aprobar vacaciones (todos)	✅	✅	❌	❌
Aprobar vacaciones (sección)	✅	✅	✅	❌
Ver calendario global	✅	✅	✅	❌
Ver propio historial	✅	✅	✅	✅
Documentos				
Subir documento (todos)	✅	✅	✅	❌
Subir documento (propio)	✅	✅	✅	✅
Solicitar documento (empleado)	✅	✅	✅	✅
Solicitar documento (RRHH→empleado)	✅	✅	✅	❌
Ver documentos de todos	✅	✅	❌	❌
Ver propios documentos	✅	✅	✅	✅
Comunicación				
Chat con RRHH	✅	✅	✅	✅
Chat con sección	✅	✅	✅	✅
Chat global	✅	❌	❌	❌
Quejas				
Crear queja	✅	✅	✅	✅
Ver todas las quejas	✅	✅	❌	❌
Resolver queja	✅	✅	❌	❌
Administración				
Gestión de usuarios	✅	❌	❌	❌
Configuración sistema	✅	❌	❌	❌
Ver auditoría completa	✅	❌	❌	❌
Reportes y estadísticas	✅	✅	❌	❌

🏗️ Arquitectura del Sistema

3.1 Vista General de Arquitectura



3.2 Stack Tecnológico

Backend PHP Vanilla

Versión mínima: PHP 8.4
Estándares: PSR-1, PSR-4, PSR-7, PSR-11, PSR-15, PSR-17

Extensiones requeridas:

```
- pdo_pgsql     → Conexión PostgreSQL
- pdo           → abstracción base de datos
- session       → gestión de sesiones
- json          → respuestas JSON
- openssl       → encriptación
- hash          → hashing de passwords
- mbstring      → manejo de caracteres
- tokenizer     → parsing PHP
- fileinfo      → detección tipos MIME
- redis         → conexión Redis (opcional)
```

```
- gd            → procesamiento imágenes
- zip           → manejo archivos comprimidos
```

Dependencias Composer:

```
{
    "require": {
        "php": "^8.4",
        "psr/http-message": "^2.0",
        "psr/http-server-handler": "^1.0",
        "psr/http-server-middleware": "^1.0",
        "psr/log": "^3.0",
        "psr/container": "^2.0",
        "psr/cache": "^3.0",
        "monolog/monolog": "^3.0",
        "otphp/totp": "^3.0",
        "web-auth/webauthn-lib": "^5.0",
        "symfony/polyfill-php84": "^1.0"
    },
    "require-dev": {
        "phpunit/phpunit": "^11.0",
        "phpstan/phpstan": "^2.0",
        "squizlabs/php_codesniffer": "^3.10"
    }
}
```

Android (Kotlin)

Versión mínima: Kotlin 2.0 / API 26 (Android 8.0)
Versión target: API 35 (Android 15)

Dependencias principales:

```
// Core
implementation("org.jetbrains.kotlin:kotlin-stdlib:2.0.0")

// Compose
implementation(platform("androidx.compose:compose-bom:2024.02.00"))
implementation("androidx.compose.ui:ui")
implementation("androidx.compose.ui:ui-graphics")
implementation("androidx.compose.ui:ui-tooling-preview")
implementation("androidx.compose.material3:material3")
implementation("androidx.compose.material:material-icons-extended")

// Navigation
implementation("androidx.navigation:navigation-compose:2.7.7")

// Lifecycle
implementation("androidx.lifecycle:lifecycle-runtime-compose:2.7.0")
implementation("androidx.lifecycle:lifecycle-viewmodel-compose:2.7.0")
```

```
// DI
implementation("com.google.dagger:hilt-android:2.50")
kapt("com.google.dagger:hilt-android-compiler:2.50")

// Networking
implementation("com.squareup.retrofit2:retrofit:2.9.0")
implementation("com.squareup.retrofit2:converter-gson:2.9.0")
implementation("com.squareup.okhttp3:okhttp:4.12.0")
implementation("com.squareup.okhttp3:logging-interceptor:4.12.0")

// Local Storage
implementation("androidx.room:room-runtime:2.6.1")
implementation("androidx.room:room-ktx:2.6.1")
kapt("androidx.room:room-compiler:2.6.1")
implementation("androidx.datastore:datastore-preferences:1.0.0")

// Security
implementation("androidx.credentials:credentials:1.5.0")
implementation("androidx.credentials:credentials-play-services-auth:1.5.0")
implementation("androidx.security:security-crypto:1.1.0-alpha06")
implementation("androidx.biometric:biometric:1.1.0")

// Image Loading
implementation("io.coil-kt:coil-compose:2.5.0")

// Coroutines
implementation("org.jetbrains.kotlinx:kotlinx-coroutines-android:1.7.3")
implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core:1.7.3")
```

Base de Datos

PostgreSQL 16

```
- Extensiones requeridas:
  ├── pgcrypto       → Encriptación a nivel BD
  ├── pg_trgm        → Búsqueda por similitud textual
  ├── uuid-ossp      → Generación de UUIDs
  └── citext         → Case-insensitive text
```

Redis 7 (Caché)

```
- Usos principales:
  ├── Caché de sesiones
  ├── Caché de consultas frecuentes
  ├── Rate limiting
  └── Cola de WebSocket (opcional)
```

3.3 Estructura de Directorios PHP

```
hr-portal/
├── config/
│   ├── config.php              # Configuración principal (dev/prod)
│   ├── database.php            # Conexión PostgreSQL
│   ├── security.php            # Configuración de seguridad
│   ├── routes.php              # Definición de rutas API
│   ├── auth.php                # Configuración autenticación
│   ├── mail.php                # Configuración SMTP
│   └── cache.php               # Configuración Redis
│
├── public/
│   ├── index.php               # Front controller
│   ├── .htaccess               # Configuración Apache
│   ├── api.php                 # Entry point API REST
│   ├── assets/
│   │   ├── css/
│   │   │   ├── app.css
│   │   │   ├── auth.css
│   │   │   └── dashboard.css
│   │   ├── js/
│   │   │   ├── app.js
│   │   │   ├── auth.js
│   │   │   └── dashboard.js
│   │   └── images/
│   │       ├── logo.svg
│   │       └── icons/
│   └── uploads/                # Documentos subidos (temporal)
│       ├── temp/
│       └── documents/
│
├── src/
│   ├── App.php                 # Clase principal aplicación
│   │
│   ├── Auth/
│   │   ├── SessionManager.php  # Gestión de sesiones
│   │   ├── AccessControl.php   # Control de acceso RBAC
│   │   ├── TokenManager.php    # JWT tokens
│   │   ├── MFA/
│   │   │   ├── MFAManager.php  # Gestor MFA
│   │   │   ├── TOTPService.php # TOTP implementation
│   │   │   └── BackupCodes.php # Códigos de respaldo
│   │   └── Passkey/
│   │       ├── PasskeyAuth.php # WebAuthn implementation
│   │       └── CredentialRepo.php # Repositorio credenciales
│   │
│   ├── Database/
│   │   ├── Database.php        # Conexión PDO
│   │   ├── Connection.php      # Pool de conexiones
│   │   ├── QueryBuilder.php    # Builder consultas seguras
│   │   ├── Migrations.php      # Migraciones BDD
│   │   └── Seeder.php          # Datos iniciales
│   │
```

```
│   ├── Http/
│   │   ├── Request.php         # PSR-7 Request
│   │   ├── Response.php        # PSR-7 Response
│   │   ├── ServerRequest.php   # ServerRequest
│   │   └── ResponseEmitter.php # Emisor respuestas
│   │
│   ├── Routing/
│   │   ├── Router.php          # Router principal
│   │   ├── RouteCollection.php # Colección rutas
│   │   ├── UrlGenerator.php    # Generador URLs
│   │   └── RouteHandler.php    # Manejador rutas
│   │
│   ├── Security/
│   │   ├── XSSProtector.php    # Protección XSS
│   │   ├── CSRFProtection.php  # Protección CSRF
│   │   ├── SecurityHeaders.php # Headers seguridad
│   │   ├── Validator.php       # Validación inputs
│   │   ├── PasswordHasher.php  # Hashing passwords
│   │   └── RateLimiter.php     # Rate limiting
│   │
│   ├── Models/
│   │   ├── BaseModel.php       # Modelo base
│   │   ├── User.php            # Modelo usuario
│   │   ├── Employee.php        # Modelo empleado
│   │   ├── Department.php      # Modelo departamento
│   │   ├── Vacation.php        # Modelo vacaciones
│   │   ├── Document.php        # Modelo documento
│   │   ├── Payroll.php         # Modelo nómina
│   │   ├── Complaint.php       # Modelo queja
│   │   ├── ChatMessage.php     # Modelo mensaje chat
│   │   ├── AuditLog.php        # Modelo auditoría
│   │   └── Notification.php    # Modelo notificación
│   │
│   ├── Repositories/
│   │   ├── UserRepository.php
│   │   ├── EmployeeRepository.php
│   │   ├── DepartmentRepository.php
│   │   ├── VacationRepository.php
│   │   ├── DocumentRepository.php
│   │   ├── PayrollRepository.php
│   │   ├── ComplaintRepository.php
│   │   ├── ChatRepository.php
│   │   ├── AuditRepository.php
│   │   └── NotificationRepository.php
│   │
│   ├── Services/
│   │   ├── EmployeeService.php
│   │   ├── VacationService.php
│   │   ├── DocumentService.php
│   │   ├── ChatService.php
│   │   ├── NotificationService.php
│   │   ├── PayrollService.php
```

```
│   │   ├── ComplaintService.php
│   │   ├── AuditService.php
│   │   ├── EmailService.php
│   │   └── FileStorageService.php
│   │
│   ├── Middleware/
│   │   ├── AuthenticationMiddleware.php
│   │   ├── AuthorizationMiddleware.php
│   │   ├── CSRFMiddleware.php
│   │   ├── RateLimitMiddleware.php
│   │   ├── SecurityHeadersMiddleware.php
│   │   ├── JsonBodyParserMiddleware.php
│   │   └── ErrorHandlerMiddleware.php
│   │
│   ├── View/
│   │   ├── View.php            # Motor de templates
│   │   ├── Layout.php          # Layout base
│   │   ├── Renderer.php        # Renderizador HTML
│   │   └── helpers.php         # Funciones helper globales
│   │
│   └── Utils/
│       ├── DateUtils.php
│       ├── StringUtils.php
│       ├── ArrayUtils.php
│       ├── FileUtils.php
│       ├── Logger.php
│       └── Exceptions.php
│
├── templates/
│   ├── layouts/
│   │   ├── base.php            # Layout base HTML
│   │   ├── auth.php            # Layout autenticación
│   │   ├── dashboard.php       # Layout panel principal
│   │   └── error.php           # Layout errores
│   │
│   ├── auth/
│   │   ├── login.php           # Formulario login
│   │   ├── mfa.php             # Verificación MFA
│   │   ├── passkey.php         # Registro Passkey
│   │   ├── register.php        # Registro (si aplica)
│   │   ├── logout.php          # Logout
│   │   ├── recovery.php        # Recuperación cuenta
│   │   └── recovery-code.php   # Código respaldo
│   │
│   ├── employees/
│   │   ├── list.php            # Lista empleados
│   │   ├── view.php            # Detalle empleado
│   │   ├── form.php            # Alta/edición
│   │   ├── profile.php         # Perfil propio
│   │   └── search.php          # Búsqueda avanzada
│   │
│   ├── vacations/
```

```
│   │   ├── dashboard.php       # Panel vacaciones
│   │   ├── request.php         # Solicitud nueva
│   │   ├── calendar.php        # Calendario visual
│   │   ├── approvals.php       # Aprobaciones pendientes
│   │   └── history.php         # Historial solicitudes
│   │
│   ├── documents/
│   │   ├── list.php            # Lista documentos
│   │   ├── upload.php          # Subir documento
│   │   ├── requests.php        # Solicitudes documentos
│   │   ├── request-form.php    # Crear solicitud
│   │   └── view.php            # Ver documento
│   │
│   ├── payroll/
│   │   ├── list.php            # Lista nóminas
│   │   ├── detail.php          # Detalle nómina
│   │   └── download.php        # Descarga PDF
│   │
│   ├── complaints/
│   │   ├── form.php            # Crear queja
│   │   ├── list.php            # Lista quejas
│   │   ├── view.php            # Detalle queja
│   │   └── status.php          # Seguimiento estado
│   │
│   ├── chat/
│   │   ├── layout.php          # Layout chat
│   │   ├── hr.php              # Chat RRHH
│   │   ├── department.php      # Chat por departamento
│   │   ├── conversation.php    # Conversación individual
│   │   └── new.php             # Nueva conversación
│   │
│   ├── admin/
│   │   ├── dashboard.php       # Panel admin
│   │   ├── users.php           # Gestión usuarios
│   │   ├── settings.php        # Configuración sistema
│   │   ├── audit.php           # Visor auditoría
│   │   └── reports.php         # Reportes
│   │
│   ├── components/
│   │   ├── header.php
│   │   ├── sidebar.php
│   │   ├── footer.php
│   │   ├── modal.php
│   │   ├── alert.php
│   │   ├── table.php
│   │   ├── form.php
│   │   ├── pagination.php
│   │   └── card.php
│   │
│   └── errors/
│       ├── 400.php
│       ├── 401.php
```

```
│       ├── 403.php
│       ├── 404.php
│       ├── 405.php
│       └── 500.php
│
├── tests/
│   ├── bootstrap.php
│   ├── Unit/
│   │   ├── Auth/
│   │   │   ├── SessionManagerTest.php
│   │   │   ├── AccessControlTest.php
│   │   │   └── MFAManagerTest.php
│   │   ├── Database/
│   │   │   ├── DatabaseTest.php
│   │   │   └── QueryBuilderTest.php
│   │   ├── Security/
│   │   │   ├── XSSProtectorTest.php
│   │   │   ├── ValidatorTest.php
│   │   │   └── PasswordHasherTest.php
│   │   ├── Models/
│   │   │   ├── UserTest.php
│   │   │   └── EmployeeTest.php
│   │   └── Services/
│   │       ├── VacationServiceTest.php
│   │       └── DocumentServiceTest.php
│   │
│   ├── Integration/
│   │   ├── Api/
│   │   │   ├── AuthApiTest.php
│   │   │   ├── EmployeeApiTest.php
│   │   │   └── VacationApiTest.php
│   │   └── Database/
│   │       └── MigrationTest.php
│   │
│   └── E2E/
│       ├── LoginCest.php
│       ├── EmployeeCest.php
│       └── VacationCest.php
│
├── logs/
│   ├── application.log
│   ├── security.log
│   ├── audit.log
│   ├── error.log
│   └── access.log
│
├── migrations/
│   ├── 001_init_schema.sql
│   ├── 002_seed_roles.sql
│   ├── 003_seed_admin_user.sql
│   ├── 004_add_employees.sql
│   ├── 005_add_vacations.sql
```

```
│   ├── 006_add_documents.sql
│   ├── 007_add_payroll.sql
│   ├── 008_add_complaints.sql
│   ├── 009_add_chat.sql
│   ├── 010_add_notifications.sql
│   ├── 011_add_audit_indexes.sql
│   └── ... (migraciones incrementales)
│
├── docs/
│   ├── API.md              # Documentación API REST
│   ├── SECURITY.md         # Guía seguridad
│   ├── DEPLOYMENT.md       # Guía despliegue
│   ├── DATABASE.md         # Schema y procedimientos
│   ├── CONTRIBUTING.md     # Guía contribuciones
│   └── CHANGELOG.md        # Historial cambios
│
├── scripts/
│   ├── migrate.sh          # Script migraciones
│   ├── seed.sh             # Script seed datos
│   ├── backup.sh           # Script backup BDD
│   └── cron-jobs.sh        # Tareas cron
│
├── vendor/                         # Composer dependencies
├── composer.json
├── composer.lock
├── phpunit.xml
├── phpstan.neon
├── .php-cs-fixer.php
├── .env.example
├── .env.testing
├── nginx.conf
├── Dockerfile
├── docker-compose.yml
├── Makefile
└── README.md
```

3.4 Estructura Android (Kotlin)

```
app/
├── src/main/
│   ├── java/com/zabalagailetak/hrapp/
│   │   ├── HrApplication.kt
│   │   │
│   │   ├── data/
│   │   │   ├── local/
│   │   │   │   ├── AppDatabase.kt
│   │   │   │   ├── dao/
│   │   │   │   │   ├── UserDao.kt
│   │   │   │   │   ├── EmployeeDao.kt
│   │   │   │   │   ├── VacationDao.kt
│   │   │   │   │   ├── DocumentDao.kt
```

```
│   │   │   │   │   ├── PayrollDao.kt
│   │   │   │   │   ├── ComplaintDao.kt
│   │   │   │   │   └── MessageDao.kt
│   │   │   │   └── entity/
│   │   │   │       ├── UserEntity.kt
│   │   │   │       ├── EmployeeEntity.kt
│   │   │   │       ├── VacationEntity.kt
│   │   │   │       ├── DocumentEntity.kt
│   │   │   │       ├── PayrollEntity.kt
│   │   │   │       ├── ComplaintEntity.kt
│   │   │   │       └── MessageEntity.kt
│   │   │   │
│   │   │   ├── remote/
│   │   │   │   ├── api/
│   │   │   │   │   ├── AuthApiService.kt
│   │   │   │   │   ├── EmployeeApiService.kt
│   │   │   │   │   ├── VacationApiService.kt
│   │   │   │   │   ├── DocumentApiService.kt
│   │   │   │   │   ├── PayrollApiService.kt
│   │   │   │   │   ├── ComplaintApiService.kt
│   │   │   │   │   └── ChatApiService.kt
│   │   │   │   ├── dto/
│   │   │   │   │   ├── AuthDtos.kt
│   │   │   │   │   ├── EmployeeDtos.kt
│   │   │   │   │   ├── VacationDtos.kt
│   │   │   │   │   ├── DocumentDtos.kt
│   │   │   │   │   ├── PayrollDtos.kt
│   │   │   │   │   ├── ComplaintDtos.kt
│   │   │   │   │   └── MessageDtos.kt
│   │   │   │   ├── interceptor/
│   │   │   │   │   ├── AuthInterceptor.kt
│   │   │   │   │   ├── ErrorInterceptor.kt
│   │   │   │   │   └── LoggingInterceptor.kt
│   │   │   │   └── WebSocketClient.kt
│   │   │   │
│   │   │   └── repository/
│   │   │       ├── AuthRepository.kt
│   │   │       ├── EmployeeRepository.kt
│   │   │       ├── VacationRepository.kt
│   │   │       ├── DocumentRepository.kt
│   │   │       ├── PayrollRepository.kt
│   │   │       ├── ComplaintRepository.kt
│   │   │       └── ChatRepository.kt
│   │   │
│   │   ├── di/
│   │   │   ├── AppModule.kt
│   │   │   ├── NetworkModule.kt
│   │   │   ├── DatabaseModule.kt
│   │   │   ├── RepositoryModule.kt
│   │   │   └── UseCaseModule.kt
│   │   │
│   │   ├── domain/
```

```
│   │   │   ├── model/
│   │   │   │   ├── User.kt
│   │   │   │   ├── Employee.kt
│   │   │   │   ├── Vacation.kt
│   │   │   │   ├── Document.kt
│   │   │   │   ├── Payroll.kt
│   │   │   │   ├── Complaint.kt
│   │   │   │   ├── Message.kt
│   │   │   │   └── Notification.kt
│   │   │   │
│   │   │   ├── repository/
│   │   │   │   ├── AuthRepository.kt
│   │   │   │   ├── EmployeeRepository.kt
│   │   │   │   └── VacationRepository.kt
│   │   │   │
│   │   │   └── usecase/
│   │   │       ├── auth/
│   │   │       │   ├── LoginUseCase.kt
│   │   │       │   ├── LogoutUseCase.kt
│   │   │       │   ├── RefreshTokenUseCase.kt
│   │   │       │   └── SetupMFAUseCase.kt
│   │   │       ├── employees/
│   │   │       │   ├── GetEmployeesUseCase.kt
│   │   │       │   ├── GetEmployeeUseCase.kt
│   │   │       │   └── UpdateEmployeeUseCase.kt
│   │   │       ├── vacations/
│   │   │       │   ├── GetVacationsUseCase.kt
│   │   │       │   ├── RequestVacationUseCase.kt
│   │   │       │   └── ApproveVacationUseCase.kt
│   │   │       └── ... (más casos de uso)
│   │   │
│   │   ├── presentation/
│   │   │   ├── navigation/
│   │   │   │   ├── NavGraph.kt
│   │   │   │   ├── Screen.kt
│   │   │   │   └── NavHost.kt
│   │   │   │
│   │   │   ├── ui/
│   │   │   │   ├── theme/
│   │   │   │   │   ├── Color.kt
│   │   │   │   │   ├── Theme.kt
│   │   │   │   │   ├── Typography.kt
│   │   │   │   │   └── Shape.kt
│   │   │   │   │
│   │   │   │   ├── components/
│   │   │   │   │   ├── auth/
│   │   │   │   │   │   ├── AuthTextField.kt
│   │   │   │   │   │   ├── PasswordTextField.kt
│   │   │   │   │   │   └── MFACodeField.kt
│   │   │   │   │   ├── common/
│   │   │   │   │   │   ├── LoadingIndicator.kt
│   │   │   │   │   │   ├── ErrorMessage.kt
```

```
│   │   │   │   │   │   ├── EmptyState.kt
│   │   │   │   │   │   └── ConfirmationDialog.kt
│   │   │   │   │   ├── employees/
│   │   │   │   │   │   ├── EmployeeCard.kt
│   │   │   │   │   │   └── EmployeeListItem.kt
│   │   │   │   │   ├── vacation/
│   │   │   │   │   │   ├── VacationBalanceCard.kt
│   │   │   │   │   │   └── VacationStatusBadge.kt
│   │   │   │   │   ├── chat/
│   │   │   │   │   │   ├── MessageBubble.kt
│   │   │   │   │   │   └── ChatInputField.kt
│   │   │   │   │   └── ... (más componentes)
│   │   │   │   │
│   │   │   │   ├── screens/
│   │   │   │   │   ├── auth/
│   │   │   │   │   │   ├── LoginScreen.kt
│   │   │   │   │   │   ├── MFAScreen.kt
│   │   │   │   │   │   ├── PasskeySetupScreen.kt
│   │   │   │   │   │   └── SplashScreen.kt
│   │   │   │   │   ├── home/
│   │   │   │   │   │   ├── HomeScreen.kt
│   │   │   │   │   │   └── DashboardScreen.kt
│   │   │   │   │   ├── employees/
│   │   │   │   │   │   ├── EmployeeListScreen.kt
│   │   │   │   │   │   ├── EmployeeDetailScreen.kt
│   │   │   │   │   │   └── EmployeeFormScreen.kt
│   │   │   │   │   ├── vacation/
│   │   │   │   │   │   ├── VacationDashboardScreen.kt
│   │   │   │   │   │   ├── VacationRequestScreen.kt
│   │   │   │   │   │   ├── VacationCalendarScreen.kt
│   │   │   │   │   │   └── VacationApprovalScreen.kt
│   │   │   │   │   ├── documents/
│   │   │   │   │   │   ├── DocumentListScreen.kt
│   │   │   │   │   │   ├── DocumentUploadScreen.kt
│   │   │   │   │   │   └── DocumentRequestScreen.kt
│   │   │   │   │   ├── payroll/
│   │   │   │   │   │   ├── PayrollListScreen.kt
│   │   │   │   │   │   └── PayrollDetailScreen.kt
│   │   │   │   │   ├── complaints/
│   │   │   │   │   │   ├── ComplaintFormScreen.kt
│   │   │   │   │   │   └── ComplaintListScreen.kt
│   │   │   │   │   ├── chat/
│   │   │   │   │   │   ├── HRChatScreen.kt
│   │   │   │   │   │   ├── DepartmentChatScreen.kt
│   │   │   │   │   │   └── ChatConversationScreen.kt
│   │   │   │   │   └── profile/
│   │   │   │   │       ├── ProfileScreen.kt
│   │   │   │   │       └── SettingsScreen.kt
│   │   │   │   │
│   │   │   │   └── widgets/
│   │   │   │       ├── vacation/
│   │   │   │       │   ├── VacationCalendar.kt
```

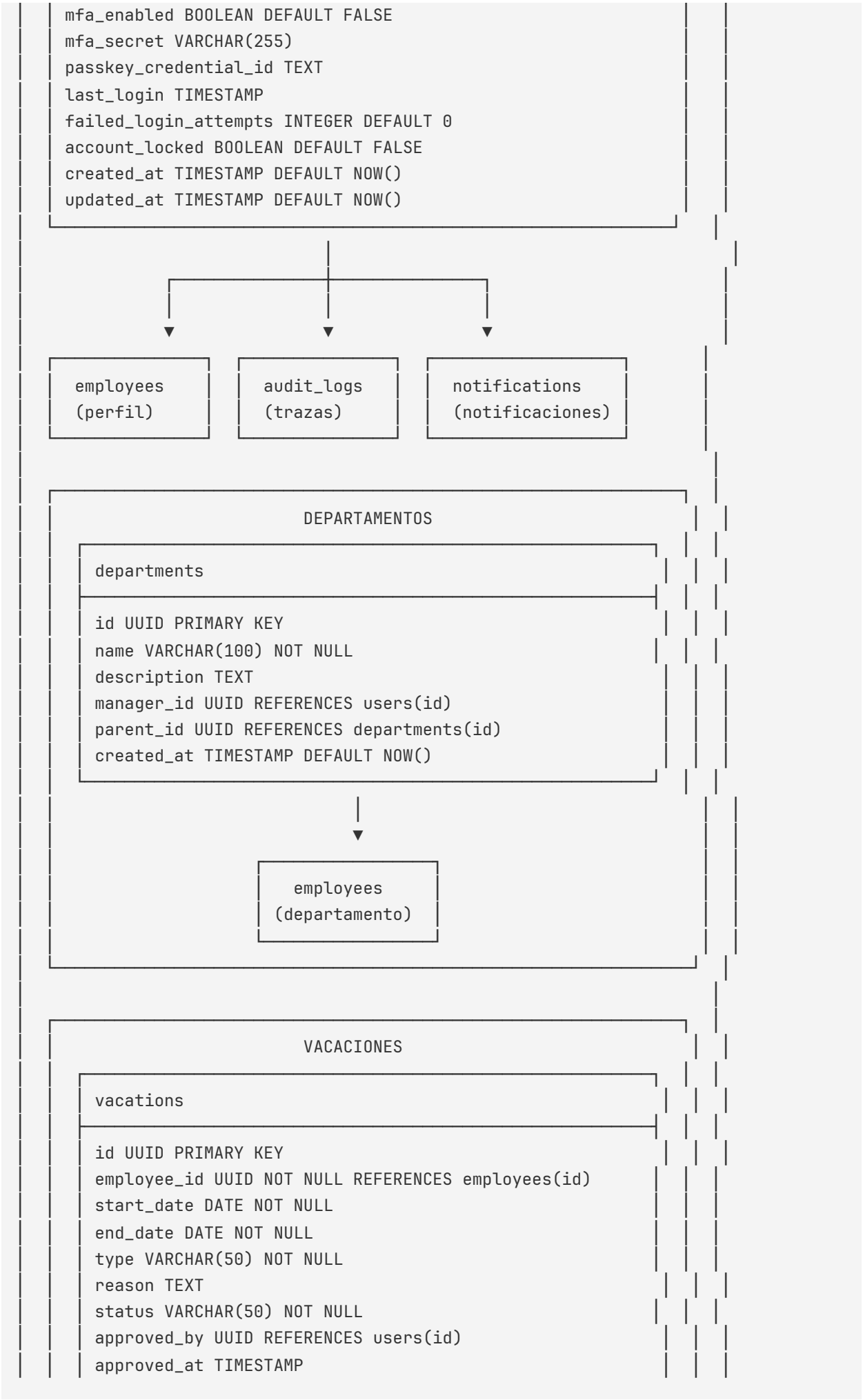
```
│   │   │   │   │   └── VacationBalanceBar.kt
│   │   │   │   ├── chat/
│   │   │   │   │   ├── ChatListItem.kt
│   │   │   │   │   └── TypingIndicator.kt
│   │   │   │   └── ... (más widgets)
│   │   │   │
│   │   │   └── viewmodel/
│   │   │       ├── auth/
│   │   │       │   ├── AuthViewModel.kt
│   │   │       │   ├── MFAViewModel.kt
│   │   │       │   └── PasskeyViewModel.kt
│   │   │       ├── home/
│   │   │       │   ├── HomeViewModel.kt
│   │   │       │   └── DashboardViewModel.kt
│   │   │       ├── employees/
│   │   │       │   ├── EmployeeListViewModel.kt
│   │   │       │   ├── EmployeeDetailViewModel.kt
│   │   │       │   └── EmployeeFormViewModel.kt
│   │   │       ├── vacation/
│   │   │       │   ├── VacationDashboardViewModel.kt
│   │   │       │   ├── VacationRequestViewModel.kt
│   │   │       │   └── VacationCalendarViewModel.kt
│   │   │       ├── documents/
│   │   │       │   ├── DocumentListViewModel.kt
│   │   │       │   └── DocumentUploadViewModel.kt
│   │   │       ├── payroll/
│   │   │       │   ├── PayrollListViewModel.kt
│   │   │       │   └── PayrollDetailViewModel.kt
│   │   │       ├── chat/
│   │   │       │   ├── ChatListViewModel.kt
│   │   │       │   └── ChatConversationViewModel.kt
│   │   │       └── profile/
│   │   │           ├── ProfileViewModel.kt
│   │   │           └── SettingsViewModel.kt
│   │   │
│   │   ├── security/
│   │   │   ├── TokenManager.kt
│   │   │   ├── PasskeyManager.kt
│   │   │   ├── BiometricManager.kt
│   │   │   ├── SecureStorage.kt
│   │   │   └── CertificatePinner.kt
│   │   │
│   │   └── util/
│   │       ├── Result.kt
│   │       ├── Extensions.kt
│   │       ├── DateUtils.kt
│   │       ├── CurrencyUtils.kt
│   │       ├── Resource.kt
│   │       └── Constants.kt
│   │
│   ├── res/
│   │   ├── values/
```

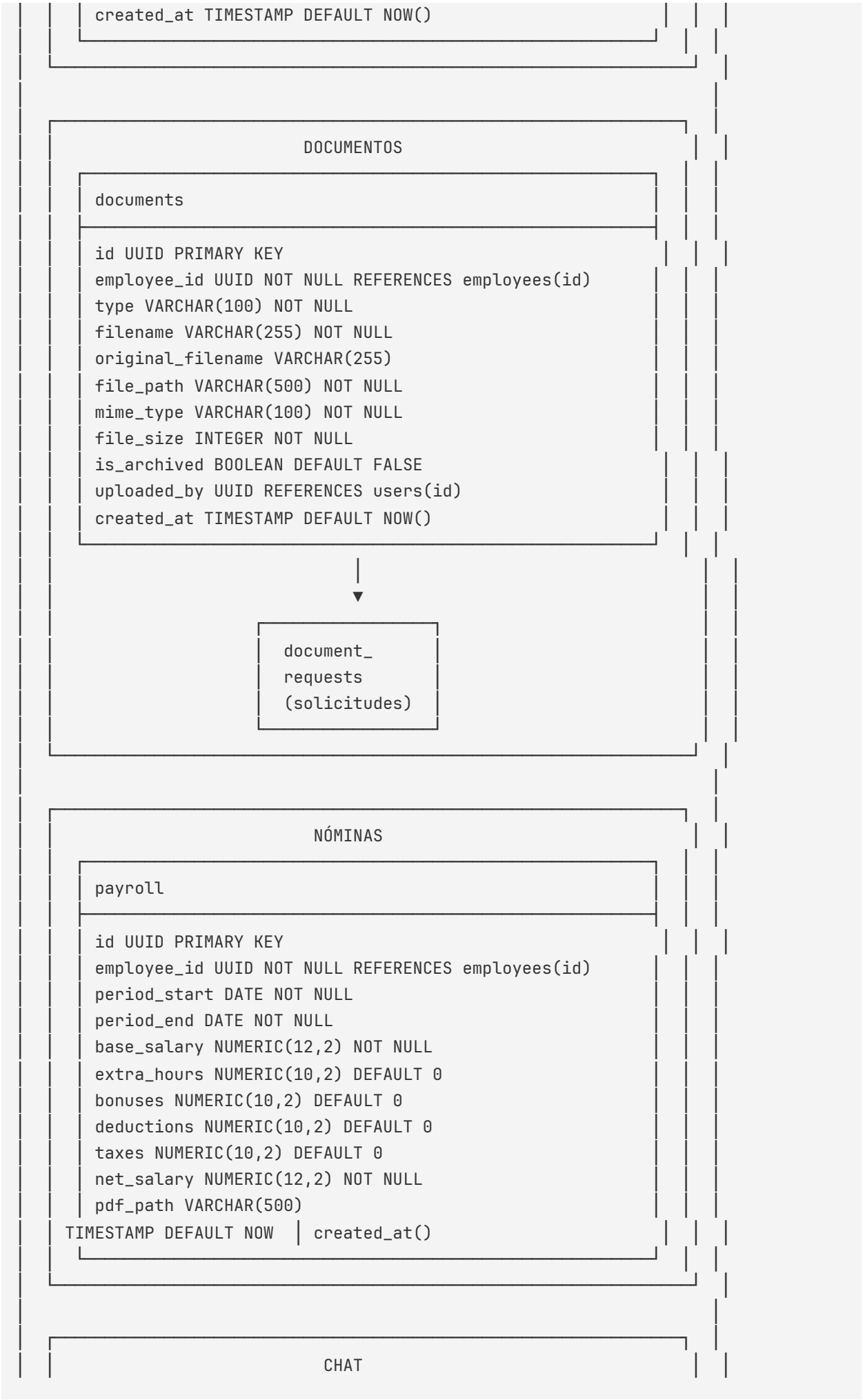
```
│   │   │   ├── strings.xml
│   │   │   ├── colors.xml
│   │   │   ├── themes.xml
│   │   │   └── dimens.xml
│   │   ├── drawable/
│   │   │   ├── ic_logo.xml
│   │   │   ├── ic_visibility.xml
│   │   │   └── ... (iconos)
│   │   ├── mipmap-hdpi/
│   │   ├── mipmap-mdpi/
│   │   ├── mipmap-xhdpi/
│   │   ├── mipmap-xxhdpi/
│   │   └── mipmap-xxxhdpi/
│   ├── assets/
│   └── xml/
│       └── network_security_config.xml
│
│   └── AndroidManifest.xml
│
├── src/test/
│   ├── java/com/zabalagailetak/hrapp/
│   │   ├── UnitTests/
│   │   │   ├── AuthViewModelTest.kt
│   │   │   ├── EmployeeUseCaseTest.kt
│   │   │   └── VacationUseCaseTest.kt
│   │   ├── IntegrationTests/
│   │   │   └── RepositoryTest.kt
│   │   └── PresentationTests/
│   │       ├── LoginScreenTest.kt
│   │       └── DashboardScreenTest.kt
│   │
│   └── java/com/zabalagailetak/hrapp/UiTests/
│       ├── LoginFlowTest.kt
│       ├── VacationRequestTest.kt
│       └── ChatFlowTest.kt
│
└── build.gradle.kts
```

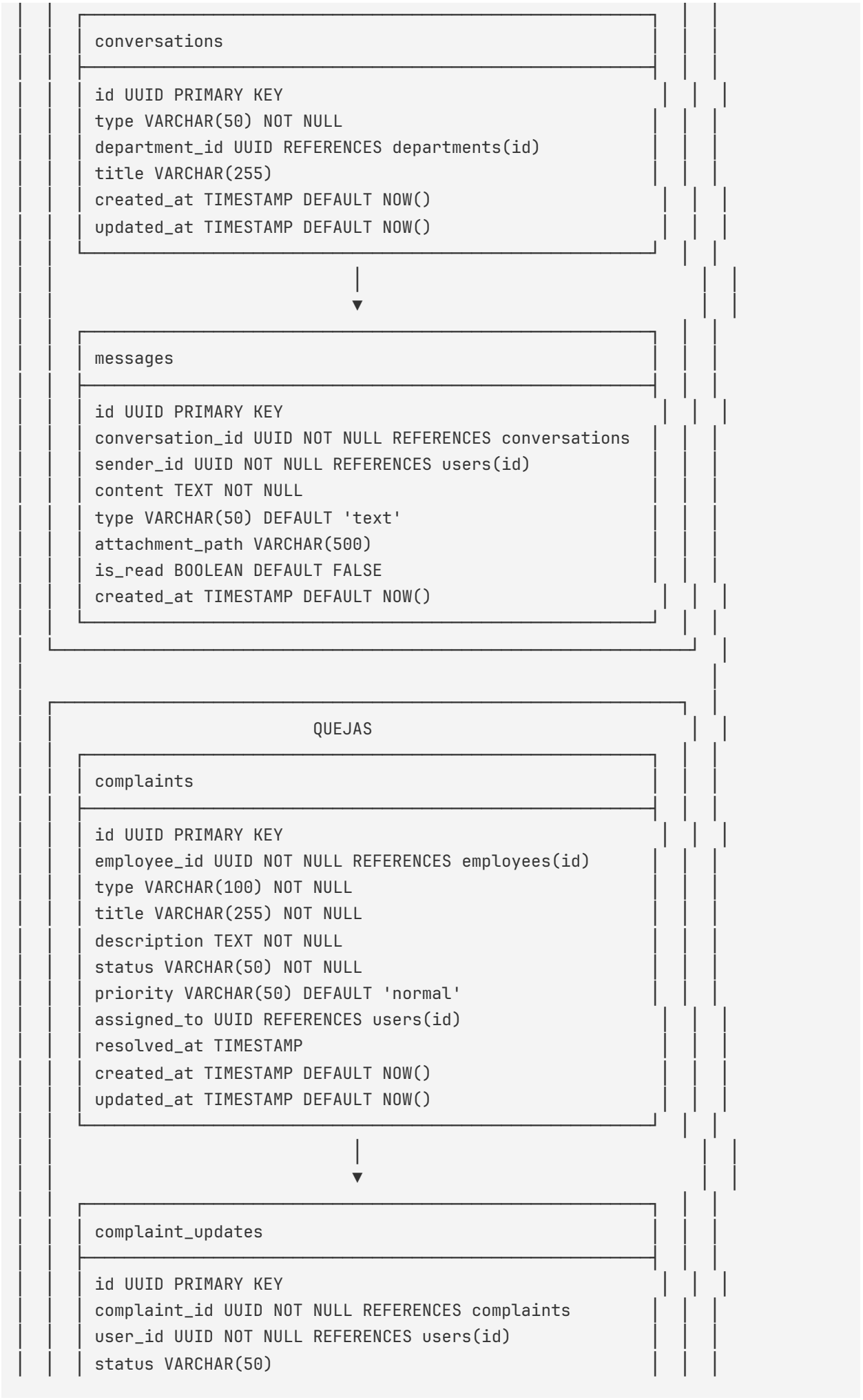
3.5 Modelo de Datos (PostgreSQL)

Diagrama de Entidades Principales

```
┌──────────────────────────────────────────────────────────────────────┐
│                          USUARIOS                                    │
│  ┌──────────────────────────────────────────────────────────┐       │
│  │ users                                                    │       │
│  ├──────────────────────────────────────────────────────────┤       │
│  │ id UUID PRIMARY KEY                                      │       │
│  │ email VARCHAR(255) UNIQUE NOT NULL                       │       │
│  │ password_hash VARCHAR(255) NOT NULL                      │       │
│  │ role VARCHAR(50) NOT NULL                                │       │
```


mfa_enabled BOOLEAN DEFAULT FALSE
mfa_secret VARCHAR(255)
passkey_credential_id TEXT
last_login TIMESTAMP
failed_login_attempts INTEGER DEFAULT 0
account_locked BOOLEAN DEFAULT FALSE
created_at TIMESTAMP DEFAULT NOW()
updated_at TIMESTAMP DEFAULT NOW()
employees (perfil)
audit_logs (trazas)
notifications (notificaciones)
DEPARTAMENTOS
departments
id UUID PRIMARY KEY
name VARCHAR(100) NOT NULL
description TEXT
manager_id UUID REFERENCES users(id)
parent_id UUID REFERENCES departments(id)
created_at TIMESTAMP DEFAULT NOW()
employees (departamento)
VACACIONES
vacations
id UUID PRIMARY KEY
employee_id UUID NOT NULL REFERENCES employees(id)
start_date DATE NOT NULL
end_date DATE NOT NULL
type VARCHAR(50) NOT NULL
reason TEXT
status VARCHAR(50) NOT NULL
approved_by UUID REFERENCES users(id)
approved_at TIMESTAMP


created_at TIMESTAMP DEFAULT NOW()
DOCUMENTOS
documents
id UUID PRIMARY KEY
employee_id UUID NOT NULL REFERENCES employees(id)
type VARCHAR(100) NOT NULL
filename VARCHAR(255) NOT NULL
original_filename VARCHAR(255)
file_path VARCHAR(500) NOT NULL
mime_type VARCHAR(100) NOT NULL
file_size INTEGER NOT NULL
is_archived BOOLEAN DEFAULT FALSE
uploaded_by UUID REFERENCES users(id)
created_at TIMESTAMP DEFAULT NOW()
document_
requests
(solicitudes)
NÓMINAS
payroll
id UUID PRIMARY KEY
employee_id UUID NOT NULL REFERENCES employees(id)
period_start DATE NOT NULL
period_end DATE NOT NULL
base_salary NUMERIC(12,2) NOT NULL
extra_hours NUMERIC(10,2) DEFAULT 0
bonuses NUMERIC(10,2) DEFAULT 0
deductions NUMERIC(10,2) DEFAULT 0
taxes NUMERIC(10,2) DEFAULT 0
net_salary NUMERIC(12,2) NOT NULL
pdf_path VARCHAR(500)
TIMESTAMP DEFAULT NOW
created_at()
CHAT


conversations
id UUID PRIMARY KEY
type VARCHAR(50) NOT NULL
department_id UUID REFERENCES departments(id)
title VARCHAR(255)
created_at TIMESTAMP DEFAULT NOW()
updated_at TIMESTAMP DEFAULT NOW()
messages
id UUID PRIMARY KEY
conversation_id UUID NOT NULL REFERENCES conversations
sender_id UUID NOT NULL REFERENCES users(id)
content TEXT NOT NULL
type VARCHAR(50) DEFAULT 'text'
attachment_path VARCHAR(500)
is_read BOOLEAN DEFAULT FALSE
created_at TIMESTAMP DEFAULT NOW()
QUEJAS
complaints
id UUID PRIMARY KEY
employee_id UUID NOT NULL REFERENCES employees(id)
type VARCHAR(100) NOT NULL
title VARCHAR(255) NOT NULL
description TEXT NOT NULL
status VARCHAR(50) NOT NULL
priority VARCHAR(50) DEFAULT 'normal'
assigned_to UUID REFERENCES users(id)
resolved_at TIMESTAMP
created_at TIMESTAMP DEFAULT NOW()
updated_at TIMESTAMP DEFAULT NOW()
complaint_updates
id UUID PRIMARY KEY
complaint_id UUID NOT NULL REFERENCES complaints
user_id UUID NOT NULL REFERENCES users(id)
status VARCHAR(50)

```
│  │  │ comment TEXT                                                  │  │  │
│  │  │ created_at TIMESTAMP DEFAULT NOW()                            │  │  │
│  │  └──────────────────────────────────────────────────────────────┘  │  │
│  └──────────────────────────────────────────────────────────────────┘  │
└──────────────────────────────────────────────────────────────────────┘
```

Script de Schema Inicial

```
-- ============================================================================
-- SCHEMA: HR Portal Database
-- Version: 1.0
-- Date: 2026-01-14
-- ============================================================================

-- Create extension for UUID generation
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION IF NOT EXISTS "pgcrypto";

-- Create custom types
DO $$ BEGIN
    CREATE TYPE user_role AS ENUM ('admin', 'hr_manager', 'department_head', 'employe
EXCEPTION
    WHEN duplicate_object THEN null;
END $$;

DO $$ BEGIN
    CREATE TYPE vacation_status AS ENUM ('pending', 'approved', 'rejected', 'cancelle
EXCEPTION
    WHEN duplicate_object THEN null;
END $$;

DO $$ BEGIN
    CREATE TYPE vacation_type AS ENUM ('annual', 'sick', 'personal', 'maternity', 'pa
EXCEPTION
    WHEN duplicate_object THEN null;
END $$;

DO $$ BEGIN
    CREATE TYPE complaint_status AS ENUM ('open', 'in_progress', 'resolved', 'closed'
EXCEPTION
    WHEN duplicate_object THEN null;
END $$;

DO $$ BEGIN
    CREATE TYPE complaint_priority AS ENUM ('low', 'normal', 'high', 'urgent');
EXCEPTION
    WHEN duplicate_object THEN null;
END $$;

DO $$ BEGIN
    CREATE TYPE chat_type AS ENUM ('hr', 'department', 'individual');
```

```
EXCEPTION
    WHEN duplicate_object THEN null;
END $$;

DO $$ BEGIN
    CREATE TYPE message_type AS ENUM ('text', 'image', 'file', 'system');
EXCEPTION
    WHEN duplicate_object THEN null;
END $$;

DO $$ BEGIN
    CREATE TYPE document_type AS ENUM ('contract', 'nif', 'payroll', 'certificate', '
EXCEPTION
    WHEN duplicate_object THEN null;
END $$;

DO $$ BEGIN
    CREATE TYPE document_request_status AS ENUM ('pending', 'submitted', 'rejected',
EXCEPTION
    WHEN duplicate_object THEN null;
END $$;

-- ============================================================================
-- TABLE: users
-- ============================================================================
CREATE TABLE IF NOT EXISTS users (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    email VARCHAR(255) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    role user_role NOT NULL DEFAULT 'employee',
    mfa_enabled BOOLEAN NOT NULL DEFAULT FALSE,
    mfa_secret VARCHAR(255),
    mfa_backup_codes TEXT[],
    passkey_credential_id TEXT,
    passkey_public_key TEXT,
    passkey_counter INTEGER DEFAULT 0,
    last_login TIMESTAMP,
    failed_login_attempts INTEGER NOT NULL DEFAULT 0,
    account_locked BOOLEAN NOT NULL DEFAULT FALSE,
    lock_until TIMESTAMP,
    password_changed_at TIMESTAMP NOT NULL DEFAULT NOW(),
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_users_email ON users(email);
CREATE INDEX IF NOT EXISTS idx_users_role ON users(role);
CREATE INDEX IF NOT EXISTS idx_users_locked ON users(account_locked);

-- ============================================================================
-- TABLE: departments
-- ============================================================================
```

```
CREATE TABLE IF NOT EXISTS departments (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    name VARCHAR(100) NOT NULL,
    description TEXT,
    manager_id UUID REFERENCES users(id),
    parent_id UUID REFERENCES departments(id),
    is_active BOOLEAN NOT NULL DEFAULT TRUE,
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_departments_manager ON departments(manager_id);
CREATE INDEX IF NOT EXISTS idx_departments_parent ON departments(parent_id);

-- ============================================================================
-- TABLE: employees
-- ============================================================================
CREATE TABLE IF NOT EXISTS employees (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id UUID NOT NULL UNIQUE REFERENCES users(id) ON DELETE CASCADE,
    employee_number VARCHAR(50) NOT NULL UNIQUE,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    nif VARCHAR(20) NOT NULL UNIQUE,
    birth_date DATE,
    gender VARCHAR(20),
    phone VARCHAR(50),
    personal_email VARCHAR(255),
    address TEXT,
    department_id UUID REFERENCES departments(id),
    position VARCHAR(100),
    hire_date DATE NOT NULL,
    termination_date DATE,
    employment_type VARCHAR(50) DEFAULT 'full_time',
    contract_type VARCHAR(50),
    salary NUMERIC(12,2),
    vacation_days INTEGER NOT NULL DEFAULT 22,
    vacation_days_used INTEGER NOT NULL DEFAULT 0,
    emergency_contact_name VARCHAR(200),
    emergency_contact_phone VARCHAR(50),
    emergency_contact_relation VARCHAR(100),
    profile_photo_path VARCHAR(500),
    notes TEXT,
    is_active BOOLEAN NOT NULL DEFAULT TRUE,
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_employees_user ON employees(user_id);
CREATE INDEX IF NOT EXISTS idx_employees_department ON employees(department_id);
CREATE INDEX IF NOT EXISTS idx_employees_number ON employees(employee_number);
CREATE INDEX IF NOT EXISTS idx_employees_name ON employees(last_name, first_name);
```

```
CREATE INDEX IF NOT EXISTS idx_employees_active ON employees(is_active);

-- ============================================================================
-- TABLE: vacation_balances
-- ============================================================================
CREATE TABLE IF NOT EXISTS vacation_balances (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    employee_id UUID NOT NULL REFERENCES employees(id) ON DELETE CASCADE,
    year INTEGER NOT NULL,
    total_days INTEGER NOT NULL DEFAULT 22,
    used_days INTEGER NOT NULL DEFAULT 0,
    pending_days INTEGER GENERATED ALWAYS AS (total_days - used_days) STORED,
    carried_over_days INTEGER DEFAULT 0,
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
    UNIQUE(employee_id, year)
);

CREATE INDEX IF NOT EXISTS idx_vacation_balances_employee ON vacation_balances(employ
CREATE INDEX IF NOT EXISTS idx_vacation_balances_year ON vacation_balances(year);

-- ============================================================================
-- TABLE: vacations
-- ============================================================================
CREATE TABLE IF NOT EXISTS vacations (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    employee_id UUID NOT NULL REFERENCES employees(id) ON DELETE CASCADE,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    type vacation_type NOT NULL,
    reason TEXT,
    status vacation_status NOT NULL DEFAULT 'pending',
    approved_by UUID REFERENCES users(id),
    approved_at TIMESTAMP,
    rejection_reason TEXT,
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_vacations_employee ON vacations(employee_id);
CREATE INDEX IF NOT EXISTS idx_vacations_status ON vacations(status);
CREATE INDEX IF NOT EXISTS idx_vacations_dates ON vacations(start_date, end_date);
CREATE INDEX IF NOT EXISTS idx_vacations_approver ON vacations(approved_by);

-- ============================================================================
-- TABLE: documents
-- ============================================================================
CREATE TABLE IF NOT EXISTS documents (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    employee_id UUID NOT NULL REFERENCES employees(id) ON DELETE CASCADE,
    type document_type NOT NULL,
    filename VARCHAR(255) NOT NULL,
```

```
    original_filename VARCHAR(255),
    file_path VARCHAR(500) NOT NULL,
    mime_type VARCHAR(100) NOT NULL,
    file_size BIGINT NOT NULL,
    checksum VARCHAR(64),
    description TEXT,
    is_archived BOOLEAN NOT NULL DEFAULT FALSE,
    archived_at TIMESTAMP,
    uploaded_by UUID NOT NULL REFERENCES users(id),
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_documents_employee ON documents(employee_id);
CREATE INDEX IF NOT EXISTS idx_documents_type ON documents(type);
CREATE INDEX IF NOT EXISTS idx_documents_uploaded_by ON documents(uploaded_by);

-- ============================================================================
-- TABLE: document_requests
-- ============================================================================
CREATE TABLE IF NOT EXISTS document_requests (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    employee_id UUID NOT NULL REFERENCES employees(id) ON DELETE CASCADE,
    requested_by UUID NOT NULL REFERENCES users(id),
    requested_type document_type NOT NULL,
    description TEXT,
    deadline DATE,
    status document_request_status NOT NULL DEFAULT 'pending',
    submitted_document_id UUID REFERENCES documents(id),
    submitted_at TIMESTAMP,
    rejection_reason TEXT,
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_doc_requests_employee ON document_requests(employee_id
CREATE INDEX IF NOT EXISTS idx_doc_requests_status ON document_requests(status);
CREATE INDEX IF NOT EXISTS idx_doc_requests_requested_by ON document_requests(request

-- ============================================================================
-- TABLE: payroll
-- ============================================================================
CREATE TABLE IF NOT EXISTS payroll (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    employee_id UUID NOT NULL REFERENCES employees(id) ON DELETE CASCADE,
    period_start DATE NOT NULL,
    period_end DATE NOT NULL,
    base_salary NUMERIC(12,2) NOT NULL,
    extra_hours NUMERIC(10,2) DEFAULT 0,
    bonuses NUMERIC(10,2) DEFAULT 0,
    commissions NUMERIC(10,2) DEFAULT 0,
    deductions NUMERIC(10,2) DEFAULT 0,
```

```
    taxes NUMERIC(10,2) DEFAULT 0,
    social_security NUMERIC(10,2) DEFAULT 0,
    other_deductions NUMERIC(10,2) DEFAULT 0,
    gross_salary NUMERIC(12,2) GENERATED ALWAYS AS (
        base_salary + COALESCE(extra_hours, 0) + COALESCE(bonuses, 0) + COALESCE(comm
    ) STORED,
    net_salary NUMERIC(12,2) NOT NULL,
    pdf_path VARCHAR(500),
    pdf_filename VARCHAR(255),
    notes TEXT,
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW(),
    UNIQUE(employee_id, period_start, period_end)
);

CREATE INDEX IF NOT EXISTS idx_payroll_employee ON payroll(employee_id);
CREATE INDEX IF NOT EXISTS idx_payroll_period ON payroll(period_start, period_end);

-- ============================================================================
-- TABLE: conversations
-- ============================================================================
CREATE TABLE IF NOT EXISTS conversations (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    type chat_type NOT NULL,
    department_id UUID REFERENCES departments(id),
    title VARCHAR(255),
    last_message_at TIMESTAMP,
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_conversations_type ON conversations(type);
CREATE INDEX IF NOT EXISTS idx_conversations_department ON conversations(department_i
CREATE INDEX IF NOT EXISTS idx_conversations_last_message ON conversations(last_messa

-- ============================================================================
-- TABLE: conversation_participants
-- ============================================================================
CREATE TABLE IF NOT EXISTS conversation_participants (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    conversation_id UUID NOT NULL REFERENCES conversations(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    joined_at TIMESTAMP NOT NULL DEFAULT NOW(),
    left_at TIMESTAMP,
    is_muted BOOLEAN DEFAULT FALSE,
    last_read_at TIMESTAMP,
    UNIQUE(conversation_id, user_id)
);

CREATE INDEX IF NOT EXISTS idx_conv_participants_user ON conversation_participants(us
CREATE INDEX IF NOT EXISTS idx_conv_participants_conversation ON conversation_partici
```

```
-- ============================================================================
-- TABLE: messages
-- ============================================================================
CREATE TABLE IF NOT EXISTS messages (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    conversation_id UUID NOT NULL REFERENCES conversations(id) ON DELETE CASCADE,
    sender_id UUID NOT NULL REFERENCES users(id),
    content TEXT NOT NULL,
    type message_type NOT NULL DEFAULT 'text',
    attachment_path VARCHAR(500),
    attachment_name VARCHAR(255),
    attachment_size BIGINT,
    reply_to_id UUID REFERENCES messages(id),
    is_edited BOOLEAN DEFAULT FALSE,
    is_deleted BOOLEAN DEFAULT FALSE,
    deleted_at TIMESTAMP,
    created_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_messages_conversation ON messages(conversation_id);
CREATE INDEX IF NOT EXISTS idx_messages_sender ON messages(sender_id);
CREATE INDEX IF NOT EXISTS idx_messages_created ON messages(created_at DESC);
CREATE INDEX IF NOT EXISTS idx_messages_reply ON messages(reply_to_id);

-- ============================================================================
-- TABLE: complaints
-- ============================================================================
CREATE TABLE IF NOT EXISTS complaints (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    employee_id UUID NOT NULL REFERENCES employees(id) ON DELETE CASCADE,
    type VARCHAR(100) NOT NULL,
    title VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    status complaint_status NOT NULL DEFAULT 'open',
    priority complaint_priority NOT NULL DEFAULT 'normal',
    assigned_to UUID REFERENCES users(id),
    resolved_at TIMESTAMP,
    resolution_summary TEXT,
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_complaints_employee ON complaints(employee_id);
CREATE INDEX IF NOT EXISTS idx_complaints_status ON complaints(status);
CREATE INDEX IF NOT EXISTS idx_complaints_priority ON complaints(priority);
CREATE INDEX IF NOT EXISTS idx_complaints_assigned ON complaints(assigned_to);

-- ============================================================================
-- TABLE: complaint_updates
-- ============================================================================
CREATE TABLE IF NOT EXISTS complaint_updates (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
```

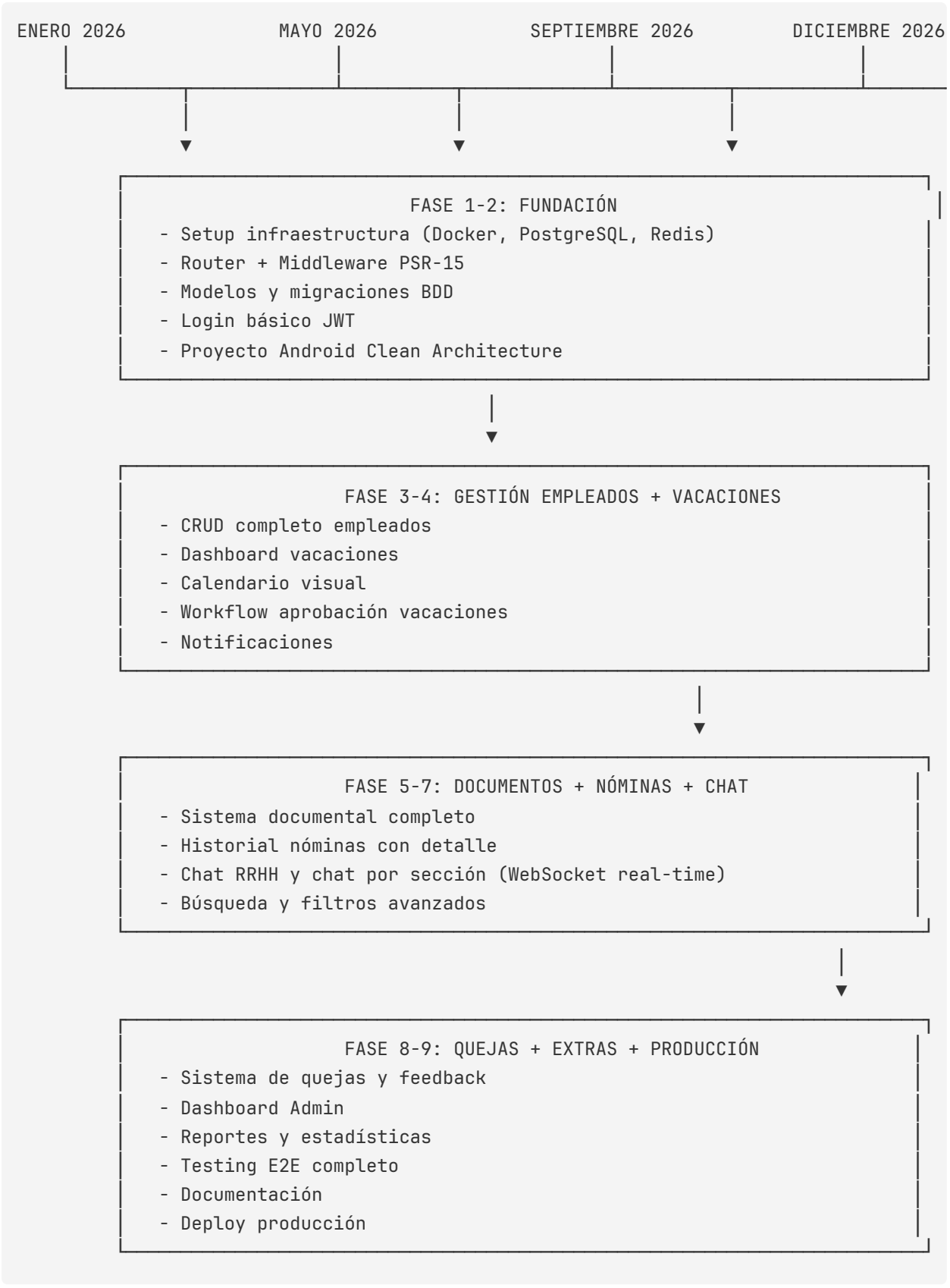
```
    complaint_id UUID NOT NULL REFERENCES complaints(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id),
    status complaint_status,
    comment TEXT,
    created_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_complaint_updates_complaint ON complaint_updates(compl

-- ============================================================================
-- TABLE: notifications
-- ============================================================================
CREATE TABLE IF NOT EXISTS notifications (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    type VARCHAR(100) NOT NULL,
    title VARCHAR(255) NOT NULL,
    message TEXT NOT NULL,
    link VARCHAR(500),
    is_read BOOLEAN NOT NULL DEFAULT FALSE,
    read_at TIMESTAMP,
    created_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_notifications_user ON notifications(user_id);
CREATE INDEX IF NOT EXISTS idx_notifications_unread ON notifications(user_id, is_read
CREATE INDEX IF NOT EXISTS idx_notifications_created ON notifications(created_at DESC

-- ============================================================================
-- TABLE: audit_logs
-- ============================================================================
CREATE TABLE IF NOT EXISTS audit_logs (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id UUID REFERENCES users(id),
    action VARCHAR(100) NOT NULL,
    entity_type VARCHAR(100),
    entity_id UUID,
    old_values JSONB,
    new_values JSONB,
    ip_address INET,
    user_agent TEXT,
    request_path VARCHAR(500),
    request_method VARCHAR(10),
    status_code INTEGER,
    duration_ms INTEGER,
    created_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_audit_logs_user ON audit_logs(user_id);
CREATE INDEX IF NOT EXISTS idx_audit_logs_action ON audit_logs(action);
CREATE INDEX IF NOT EXISTS idx_audit_logs_entity ON audit_logs(entity_type, entity_id
CREATE INDEX IF NOT EXISTS idx_audit_logs_created ON audit_logs(created_at DESC);
```

```
-- ============================================================================
-- TABLE: sessions
-- ============================================================================
CREATE TABLE IF NOT EXISTS sessions (
    id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    token VARCHAR(500) NOT NULL,
    ip_address INET,
    user_agent TEXT,
    expires_at TIMESTAMP NOT NULL,
    created_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE INDEX IF NOT EXISTS idx_sessions_user ON sessions(user_id);
CREATE INDEX IF NOT EXISTS idx_sessions_token ON sessions(token);
CREATE INDEX IF NOT EXISTS idx_sessions_expires ON sessions(expires_at);

-- ============================================================================
-- FUNCTIONS AND TRIGGERS
-- ============================================================================

-- Function to update updated_at timestamp
CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = NOW();
    RETURN NEW;
END;
$$ language 'plpgsql';

-- Apply updated_at trigger to all tables with this column
DO $$
DECLARE
    t text;
BEGIN
    FOR t IN
        SELECT table_name
        FROM information_schema.columns
        WHERE column_name = 'updated_at'
        AND table_schema = 'public'
    LOOP
        EXECUTE format('
            CREATE TRIGGER update_%I_updated_at
            BEFORE UPDATE ON %I
            FOR EACH ROW
            EXECUTE FUNCTION update_updated_at_column()
        ', t, t);
    END LOOP;
END;
$$ language 'plpgsql';
```

```sql
-- Function to generate employee number
CREATE OR REPLACE FUNCTION generate_employee_number()
RETURNS TRIGGER AS $$
DECLARE
    year_prefix TEXT;
    next_seq INTEGER;
BEGIN
    year_prefix := EXTRACT(YEAR FROM NOW())::TEXT;

    SELECT COALESCE(MAX(CAST(SUBSTRING(employee_number FROM 5) AS INTEGER)), 0) + 1
    INTO next_seq
    FROM employees
    WHERE employee_number LIKE year_prefix || '-%';

    NEW.employee_number := year_prefix || '-' || LPAD(next_seq::TEXT, 4, '0');
    RETURN NEW;
END;
$$ language 'plpgsql';

-- ============================================================================
-- SEED DATA
-- ============================================================================

-- Insert default roles (handled by ENUM)

-- Insert admin user (password: Admin123!)
INSERT INTO users (id, email, password_hash, role, mfa_enabled)
VALUES (
    uuid_generate_v4(),
    'admin@zabalagailetak.com',
    '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', -- password
    'admin',
    FALSE
);

-- Insert sample departments
INSERT INTO departments (id, name, description) VALUES
(uuid_generate_v4(), 'Administración', 'Departamento de administración y gestión'),
(uuid_generate_v4(), 'Producción', 'Departamento de producción'),
(uuid_generate_v4(), 'Calidad', 'Departamento de control de calidad'),
(uuid_generate_v4(), 'Mantenimiento', 'Departamento de mantenimiento'),
(uuid_generate_v4(), 'Recursos Humanos', 'Departamento de recursos humanos'),
(uuid_generate_v4(), 'IT', 'Departamento de tecnologías de la información');

-- Insert sample users for departments (employees)
-- Los passwords son: Employee123!
```

📅 Plan de Implementación por Fases

Visión General del Timeline



Fase 1: Fundación (Semanas 1-4)

Objetivo: Configurar infraestructura base del proyecto

Semana	Tarea Principal	Entregable	Estado
1	Configurar PostgreSQL + Redis + Docker	Entorno dev funcionando	⏳
1	Setup proyecto PHP vanilla (PSR-4, Composer)	Estructura base PHP	⏳
1	Configurar nginx + SSL para desarrollo	Servidor web configurado	⏳
2	Implementar router + middleware PSR-15	Sistema routing funcional	⏳
2	Configurar seguridad (headers, CSRF, XSS)	Base de seguridad	⏳
2	Implementar gestión de sesiones seguras	SessionManager	⏳
3	Crear modelos base y migraciones BDD	Schema DB completo	⏳
3	Implementar autenticación básica (JWT)	Login básico funcional	⏳
3	Crear servicios core (Employee, Auth)	Servicios base	⏳
4	Setup Android Studio + Gradle + Kotlin	Proyecto Android base	⏳
4	Implementar arquitectura Clean + MVI	Estructura Android	⏳
4	Implementar API Auth para Android	Endpoints auth Android	⏳

Entregable fase 1: Sistema de login básico funcional en web y Android

Criterios de aceptación:

- [ ] Usuario puede registrarse/login con email/password
- [ ] Sesión se mantiene entre requests
- [ ] API REST responde con JSON correcto
- [ ] Android puede autenticarse y recibir token
- [ ] Headers de seguridad aplicados

Fase 2: Autenticación Avanzada (Semanas 5-8)

Objetivo: Sistema de autenticación robusto con MFA y Passkey

Semana	Tarea Principal	Entregable	Estado
5	Implementar MFA (TOTP) en PHP	MFA funcional	⏳
5	Generar QR codes para setup MFA	Interfaz setup MFA	⏳
5	Implementar validación código MFA	Verificación 6 dígitos	⏳

Semana	Tarea Principal	Entregable	Estado
5	Implementar códigos de respaldo	Backup codes funcional	⏳
6	Integrar biblioteca Passkey/WebAuthn PHP	Backend Passkey	⏳
6	Implementar registro credential Passkey	Registration flow	⏳
6	Implementar autenticación Passkey	Login Passkey	⏳
6	Integrar CredentialManager Android	Android Passkey	⏳
7	Implementar política de passwords seguros	Password hashing robusto	⏳
7	Implementar rate limiting por IP	Protección brute force	⏳
7	Sistema de lockout tras intentos fallidos	Account lockout	⏳
8	Auditoría de seguridad completa	Informe seguridad	⏳
8	Tests unitarios autenticación	Cobertura > 80%	⏳
8	Documentación API autenticación	Docs completas	⏳

Entregable fase 2: Login con MFA y Passkey funcionando en ambas plataformas

Criterios de aceptación:

- [ ] Usuario puede activar MFA con Google Authenticator
- [ ] Login requiere código MFA tras password
- [ ] Códigos de respaldo funcionan si se pierde MFA
- [ ] Usuario puede registrar Passkey desde Android
- [ ] Login Passkey funciona con biometría
- [ ] Cuenta se bloquea tras 5 intentos fallidos
- [ ] Passwords deben cumplir política de seguridad

Fase 3: Gestión de Empleados (Semanas 9-14)

Objetivo: CRUD completo de empleados con todos sus datos

Semana	Tarea Principal	Entregable	Estado
9	API CRUD empleados PHP	Endpoints funcionales	⏳
9	Web: Lista empleados + paginación	Vista lista empleados	⏳

Semana	Tarea Principal	Entregable	Estado
9	Web: Filtros por departamento/búsqueda	Filtros avanzados	⏳
10	Web: Formulario alta empleado	Formulario completo	⏳
10	Web: Formulario edición empleado	Editar datos	⏳
10	Validación datos empleados	Validación server-side	⏳
10	Android: Lista empleados	Pantalla lista Android	⏳
11	Android: Detalle empleado	Pantalla detalle Android	⏳
11	Android: Búsqueda empleados	Búsqueda en app	⏳
11	Web: Perfil empleado completo	Vista perfil detallada	⏳
11	Sistema de secciones/departamentos	Gestión departamentos	⏳
12	Asignación jefe de sección	Jerarquía departamentos	⏳
12	Búsqueda avanzada (nombre, NIF, etc.)	Múltiples criterios	⏳
12	Exportación lista empleados (CSV/PDF)	Reporte empleados	⏳
13	Web: Baja de empleados (lógica)	Soft delete funcional	⏳
13	Web: Reactivación empleado	Restaurar registro	⏳
13	Historial cambios empleado	Trazabilidad datos	⏳
13	Android: Operaciones básicas empleado	App CRUD básico	⏳
14	Tests unitarios empleados	Cobertura > 70%	⏳
14	Tests integración API empleados	Tests API	⏳
14	Documentación completa empleados	Docs API	⏳

Entregable fase 3: Gestión completa de empleados en web y Android

Criterios de aceptación:

- [ ] Admin puede crear/editar/eliminar empleados
- [ ] RRHH puede crear/editar empleados
- [ ] Jefe de sección ve solo su equipo
- [ ] Empleado ve solo su propio perfil
- [ ] Búsqueda funciona por múltiples campos

- [ ] Paginación y filtros funcionan correctamente
- [ ] Android muestra lista y detalle correctamente

Fase 4: Vacaciones y Calendario (Semanas 15-20)

Objetivo: Sistema completo de gestión de vacaciones

Semana	Tarea Principal	Entregable	Estado
15	API: Balance vacaciones empleado	Endpoints balance	⏳
15	API: CRUD solicitudes vacaciones	Endpoints solicitudes	⏳
15	Web: Dashboard vacaciones usuario	Panel principal	⏳
15	Web: Balance días disponibles	Visualización clara	⏳
16	Web: Calendario visual anual	Componente calendario	⏳
16	Web: Calendario por departamento	Vista global	⏳
16	Web: Formulario solicitudvacaciones	Formulario intuitivo	⏳
16	Selector rango fechas	Date range picker	⏳
17	Flujo aprobación (jefe → RRHH)	Workflow aprobación	⏳
17	Notificaciones email approval	Alertas automáticas	⏳
17	Android: Dashboard vacaciones	UI Android vacaciones	⏳
17	Android: Selector fechas + calendario	Selector fechas	⏳
18	Android: Lista solicitudes propias	Lista Android	⏳
18	Android: Estado solicitudes	Seguimiento estado	⏳
18	Web: Aprobaciones pendientes (jefes)	Panel aprobación	⏳
18	Web: Aprobaciones RRHH	Panel RRHH	⏳
19	Notificaciones push Android	Push notifications	⏳
19	Centro notificaciones web	Panel notificaciones	⏳
19	Reglas de negocio vacaciones	Lógica completa	⏳
20	Web: Calendario empresa (todos)	Vista global completa	⏳
20	Tests unitarios vacaciones	Cobertura > 70%	⏳

Semana	Tarea Principal	Entregable	Estado
20	Documentación completa	Docs API	⏳

Entregable fase 4: Sistema de vacaciones completo

Criterios de aceptación:

- [ ] Usuario puede solicitar vacaciones
- [ ] Días disponibles se calculan correctamente
- [ ] Jefe puede aprobar/rechazar de su equipo
- [ ] RRHH puede aprobar/rechazar globalmente
- [ ] Calendario muestra vacaciones de todos
- [ ] Notificaciones llegan por email y push
- [ ] Android permite solicitar y ver estado

Fase 5: Documentos (Semanas 21-24)

Objetivo: Sistema de gestión documental completo

Semana	Tarea Principal	Entregable	Estado
21	API: Subida archivos segura	Endpoint upload	⏳
21	API: Descarga archivos	Endpoint download	⏳
21	Validación tipos MIME	Solo archivos permitidos	⏳
21	Almacenamiento seguro (fuera webroot)	Sistema almacenamiento	⏳
22	Web: Lista documentos usuario	Vista documentos	⏳
22	Sistema tipos documentos	Taxonomía documentos	⏳
22	Web: Subida con validación	Formulario subida	⏳
22	Previsualización PDFs/imágenes	Viewer integrado	⏳
23	Solicitud de documentos (RRHH→empleado)	Workflow solicitudes	⏳
23	Notificación empleado solicitud	Alerta documento	⏳
23	Android: Lista documentos	UI Android documentos	⏳
23	Android: Cámara + subida archivos	Subida desde móvil	⏳

Semana	Tarea Principal	Entregable	Estado
24	Historial versiones documentos	Control versiones	⏳
24	Búsqueda documentos	Por tipo, fecha, nombre	⏳
24	Android: Detalle documento	Preview Android	⏳
24	Tests documentos	Cobertura > 70%	⏳

Entregable fase 5: Sistema documental completo

Criterios de aceptación:

- [ ] Usuario puede subir documentos
- [ ] Solo tipos de archivo permitidos
- [ ] RRHH puede solicitar documentos a empleados
- [ ] Empleado recibe notificación de solicitud
- [ ] Android permite subir desde cámara/archivos
- [ ] Documentos se almacenan de forma segura

Fase 6: Nóminas (Semanas 25-28)

Objetivo: Consulta de historial de nóminas

Semana	Tarea Principal	Entregable	Estado
25	API: Historial nóminas por empleado	Endpoints nóminas	⏳
25	API: Detalle nómina (desglose conceptos)	Detalle completo	⏳
25	Web: Lista nóminas por empleado	Vista nóminas web	⏳
25	Filtros por año/mes	Búsqueda temporal	⏳
26	Web: Detalle nómina (desglose)	Desglose completo	⏳
26	Web: Cálculo automático conceptos	Cálculos correctos	⏳
26	Web: Descarga PDFs nóminas	Generación PDFs	⏳
26	Android: Lista nóminas	UI Android nóminas	⏳
27	Android: Detalle + descarga	Pantalla detalle Android	⏳
27	Filtros por fecha/rango	Búsqueda nóminas	⏳

Semana	Tarea Principal	Entregable	Estado
27	Resumen anual nóminas	Vista anual	⏳
28	Comparativa nóminas	Gráfico evolución	⏳
28	Tests nóminas	Cobertura > 70%	⏳
28	Documentación API nóminas	Docs completas	⏳

Entregable fase 6: Consulta de nóminas completa

Criterios de aceptación:

- [ ] Empleado puede ver sus nóminas
- [ ] RRHH puede ver todas las nóminas
- [ ] Desglose de conceptos es correcto
- [ ] Descarga de PDF funciona
- [ ] Android permite ver nóminas

Fase 7: Chat (Semanas 29-34)

Objetivo: Sistema de mensajería interno real-time

Semana	Tarea Principal	Entregable	Estado
29	Servidor WebSocket para chat	Servidor WS funcional	⏳
29	API: Gestión conversaciones	Endpoints conversaciones	⏳
29	Web: Chat RRHH (público)	UI chat RRHH web	⏳
29	Web: Chat por sección	UI chat sección web	⏳
30	Web: Lista conversaciones	Vista conversaciones	⏳
30	Web: Interfaz mensaje individual	Chat completo web	⏳
30	Historial mensajes + persistencia	Almacenamiento mensajes	⏳
30	Búsqueda en chat	Buscar mensajes	⏳
31	Android: Chat RRHH	UI Android chat RRHH	⏳
31	Android: Chat por sección	UI Android chat sección	⏳
31	Android: Lista conversaciones	Lista Android	⏳

Semana	Tarea Principal	Entregable	Estado
31	Android: Chat conversación	Pantalla chat Android	⏳
32	Notificaciones push chat	Alertas nuevos msgs	⏳
32	Indicadores de lectura	Mensajes leídos	⏳
32	Indicador escribiendo…	Typing indicator	⏳
33	Adjuntos en chat (imágenes)	Compartir archivos	⏳
33	Emojis y formato texto	Rich text básico	⏳
33	Web: Mensajes automáticos sistema	Notifications sistema	⏳
34	Tests chat	Cobertura > 70%	⏳
34	Documentación API chat	Docs completas	⏳

Entregable fase 7: Sistema de chat completo

Criterios de aceptación:

- [ ] Chat RRHH funciona con WebSocket
- [ ] Chat por sección funciona correctamente
- [ ] Mensajes se persisten en BD
- [ ] Notificaciones push llegan
- [ ] Android sincroniza mensajes en tiempo real
- [ ] Se ven indicadores de lectura

Fase 8: Quejas y Feedback (Semanas 35-38)

Objetivo: Sistema de complaints interno

Semana	Tarea Principal	Entregable	Estado
35	API: CRUD quejas/sugerencias	Endpoints quejas	⏳
35	Web: Formulario queja/anónimo	Formulario web	⏳
35	Categorías quejas	Clasificación quejas	⏳
35	Priorización quejas	Niveles prioridad	⏳
36	Web: Lista quejas (RRHH)	Vista RRHH quejas	⏳

Semana	Tarea Principal	Entregable	Estado
36	Web: Detalle queja	Vista completa	⏳
36	Flujo resolución quejas	Workflow resolución	⏳
36	Web: Seguimiento estado quejas	Timeline estado	⏳
37	Android: Formulario queja	UI Android queja	⏳
37	Android: Estado seguimiento	Tracking Android	⏳
37	Notificaciones actualizaciones quejas	Alertas cambios	⏳
38	Reporte estadísticas quejas	Dashboard quejas	⏳
38	Tests quejas	Cobertura > 70%	⏳
38	Documentación completa	Docs API	⏳

Entregable fase 8: Sistema de complaints funcional

Criterios de aceptación:

- [ ] Empleado puede crear queja
- [ ] Opciones de anonimato disponibles
- [ ] RRHH ve y gestiona quejas
- [ ] Workflow de resolución funciona
- [ ] Empleado ve estado de su queja

Fase 9: Extras y Polish (Semanas 39-44)

Objetivo: Funcionalidades adicionales y refinamiento para producción

Semana	Tarea Principal	Entregable	Estado
39	Sistema notificaciones global	Centro notificaciones	⏳
39	Preferencias usuario	Configuración personal	⏳
39	Notificaciones email configurables	Preferencias email	⏳
40	Dashboard Admin completo	Panel administración	⏳
40	Gestión usuarios admin	CRUD usuarios admin	⏳
40	Reportes y estadísticas	Gráficos/estadísticas	⏳

Semana	Tarea Principal	Entregable	Estado
40	Exportar datos (GDPR)	Exportación datos	⏳
41	Perfil empleado editable	Modificación datos propios	⏳
41	Contactos emergencia	Datos contacto	⏳
41	Photo perfil	Subir foto perfil	⏳
42	Testing E2E completo	Tests E2E web	⏳
42	Testing E2E Android	Tests E2E app	⏳
42	Optimización rendimiento	Performance tuning	⏳
43	Documentación técnica	Docs completas	⏳
43	Manual de usuario	Guías usuario	⏳
43	Documentación API actualizada	Docs finales	⏳
44	Deploy producción	Sistema en prod	⏳
44	Monitorización (logs, errores)	Sistema monitoreo	⏳
44	Backup y recovery	Procedimientos backup	⏳
44	Training usuarios	Formación inicial	⏳

Entregable fase 9: Producto listo para producción

Criterios de aceptación:

- [ ] Todas las funcionalidades funcionan
- [ ] Tests E2E pasan
- [ ] Documentación completa
- [ ] Sistema desplegado en producción
- [ ] Usuarios formados

📊 Estimación de Esfuerzo

Resumen por Fase

Fase	Semanas	Complejidad	Esfuerzo (personas)	Dependencias
Fase 1: Fundación	4	Media	2 desarrolladores	-
Fase 2: Autenticación	4	Alta	2 desarrolladores	Fase 1
Fase 3: Empleados	6	Media	2 desarrolladores	Fase 1
Fase 4: Vacaciones	6	Media	2 desarrolladores	Fase 1
Fase 5: Documentos	4	Media	2 desarrolladores	Fase 3
Fase 6: Nóminas	4	Baja	1-2 desarrolladores	Fase 3
Fase 7: Chat	6	Alta	2 desarrolladores	Fase 1
Fase 8: Quejas	4	Baja	1 desarrollador	Fase 3
Fase 9: Extras	6	Media	2 desarrolladores	Fases 1-8
Total	**44**	-	**~8-10 meses-hombre**	-

Recursos Necesarios

Rol	Cantidad	Responsabilidad Principal	Tipo
Lead Developer PHP	1	Arquitectura PHP, seguridad, code review	Full-time
Developer PHP	1-2	Implementación features backend	Full-time
Lead Developer Android	1	Arquitectura Android, UI/UX, integración	Full-time
Developer Android	1	Implementación pantallas y lógica	Full-time
DevOps	0.5 (compartido)	Docker, CI/CD, infraestructura	Parcial
QA	0.5 (opcional)	Testing E2E, documentación	Parcial

Estimación de Coste (aproximada)

Recurso	Meses	Coste/mes	Total
Lead PHP	11	5.000€	55.000€
Dev PHP	11	3.500€	38.500€
Lead Android	11	5.000€	55.000€
Dev Android	11	3.500€	38.500€

Recurso	Meses	Coste/mes	Total
DevOps (50%)	11	4.000€	22.000€
QA (50%)	6	2.500€	15.000€
Total	-	-	**~224.000€**

Nota: Estos costes son estimaciones orientativas. Pueden variar según el mercado y la disponibilidad de recursos.

⚠️ Riesgos Identificados y Mitigaciones

Matriz de Riesgos

ID	Riesgo	Probabilidad	Impacto	Nivel	Mitigación
R1	Retraso por complejidad Passkey	Media	Alto	🟡 Amarillo	Plan B: solo MFA si Passkey es muy complejo
R2	Integración Android más compleja de lo esperado	Media	Medio	🟡 Amarillo	Prototipo temprano para validar arquitectura
R3	Rendimiento chat WebSocket bajo carga	Baja	Medio	🟢 Verde	Optimizar WebSocket desde el inicio, escalar horizontal
R4	Cambios requisitos durante desarrollo	Media	Medio	🟡 Amarillo	Metodología ágil, sprints cortos, revisiones frecuentes
R5	Disponibilidad recursos (enfermedades, rotación)	Media	Alto	🟡 Amarillo	Contratos claros, documentación exhaustiva, knowledge sharing
R6	Seguridad datos sensibles (nóminas)	Baja	Alto	🟢 Verde	Auditoría seguridad externa, encriptación BD
R7	Compatible browsers antiguos	Baja	Bajo	🟢 Verde	Polyfills solo si es necesario, soporte browsers modernos
R8	Tiempo desarrollo mayor al estimado	Media	Alto	🟡 Amarillo	Buffer de tiempo en planning, scope flexible

Plan de Contingencia

Si Passkey es demasiado complejo:

- Reducir a MFA obligatorio (TOTP)
- Passkey como opción adicional (no requisito)
- Implementación Passkey en Fase 9 (nice to have)

Si Android se retrasa:

- Priorizar funcionalidades web
- Android con funcionalidades básicas en launch
- Actualizaciones posteriores para Android

Si hay cambios de requisitos:

- Requerir sign-off de PO para cambios scope
- Trade-off analysis para cada cambio
- Ajustar timeline si es necesario

🔧 Métricas de Éxito

Métricas de Código

Métrica	Objetivo	Medición
Cobertura tests unitarios	> 70%	PHPUnit coverage report
Cobertura tests integración	> 50%	PHPUnit integration tests
Tiempo respuesta API (p95)	< 200ms	APM/New Relic
Complejidad ciclomática promedio	< 10	PHPStan/CodeClimate
Duplicación de código	< 3%	PHPMD/Copy/Paste Detector
Score seguridad	Grade A	Security headers + OWASP ZAP

Métricas de Seguridad

Métrica	Objetivo	Medición
Vulnerabilidades críticas conocidas	0	Scanner automatizado
Tiempo exposición sesión inactiva	< 8 horas	Configuración sesión
Logs de auditoría acciones sensibles	100%	Revisión logs
Autenticación 2FA usuarios	100%	Reporte usage

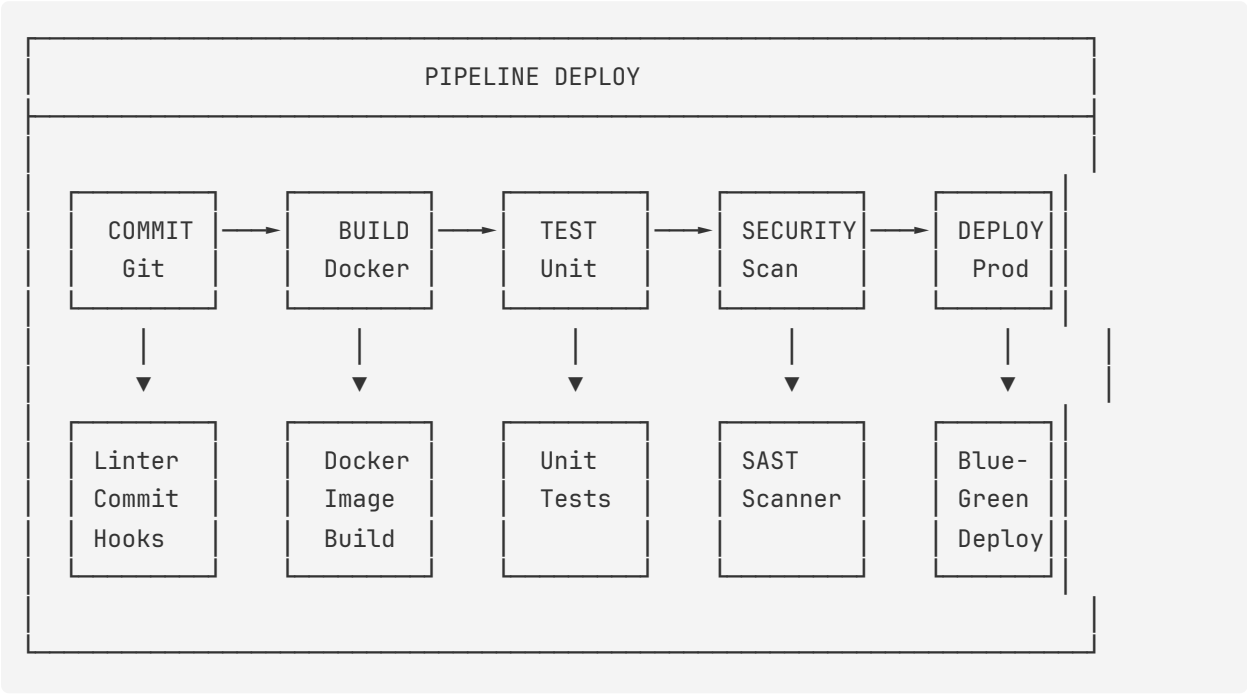
Métrica	Objetivo	Medición
Passwords débiles	0%	Password policy enforcement
Puerto expuesto incorrectamente	0	Security audit

Métricas de Usuario

Métrica	Objetivo	Medición
Tiempo carga página (primer paint)	< 2 segundos	Lighthouse
App startup time (Android)	< 3 segundos	Android Profiler
Tasa de error login	< 5%	Analytics
Satisfacción usuarios (NPS)	> 40	Encuestas
Adoption rate (a 3 meses)	> 80%	Usage analytics
Accesibilidad WCAG 2.1	Level AA	Audit accesibilidad

🚀 Métricas de Deploy

Pipeline CI/CD



Pasos del Pipeline

1. **Commit Validation**
    - PHP CodeSniffer (PSR-12)
    - PHPStan (Static Analysis)
    - Pre-commit hooks
2. **Build**
    - Composer install
    - Build Docker image
    - Build APK Android
3. **Test**
    - PHPUnit Unit Tests
    - PHPUnit Integration Tests
    - Android Lint
4. **Security Scan**
    - PHP Security Advisories
    - OWASP ZAP Scan
    - Dependency Check
5. **Deploy**
    - Deploy a staging
    - Tests E2E en staging
    - Deploy a producción

📋 Siguientes Pasos Inmediatos

Esta Semana (Días 1-7)

Configuración inicial:

- [ ] Confirmar equipo de desarrollo
- [ ] Configurar repositorios Git (PHP y Android)
- [ ] Setup entorno desarrollo Docker
- [ ] Instalar IDEs y herramientas necesarias
- [ ] Configurar comunicación equipo (Slack/Discord)

Documentación:

- [ ] Crear wiki del proyecto
- [ ] Documentar arquitectura inicial
- [ ] Setup sistema de tickets/issues

Próximas 2 Semanas (Días 8-14)

Análisis técnico:

- [ ] Decidir estructura detallada Base de Datos
- [ ] Crear mocks/wireframes UI (Figma/balsamiq)
- [ ] Definir contratos API detallados
- [ ] Planificar sprint 1 (Fase 1)

Desarrollo:

- [ ] Iniciar Fase 1: Fundación
- [ ] Setup PostgreSQL + Redis
- [ ] Implementar estructura PHP base
- [ ] Configurar Android Studio

Mes 1

Objetivos:

- [ ] MVP login funcional
- [ ] Documentación técnica inicial
- [ ] Revisión seguridad básica
- [ ] Primera demo al equipo

📚 Documentos Relacionados

Documento	Descripción	Estado
API.md	Documentación detallada de endpoints REST	⏳ Pendiente
SECURITY.md	Guía de seguridad y best practices	⏳ Pendiente
DEPLOYMENT.md	Guía de despliegue a producción	⏳ Pendiente
DATABASE.md	Schema detallado y procedimientos	⏳ Pendiente
CONTRIBUTING.md	Guía de contribuciones al código	⏳ Pendiente

Documento	Descripción	Estado
CHANGELOG.md	Historial de cambios	⏳ Pendiente

📝 Control de Versiones

Versión	Fecha	Autor	Cambios
1.0	2026-01-14	AI Assistant	Versión inicial del plan

✅ Aprobaciones

Rol	Nombre	Firma	Fecha
Product Owner			
Tech Lead			
Project Manager			

Documento creado: 14 de Enero de 2026
Próxima revisión programada: 28 de Enero de 2026
Próxima actualización automática: 11 de Febrero de 2026