PLAN DE MIGRACIÓN Y RESPALDO DE DATOS – CDMI

Sistema de Información:

CDMI – Sistema de Apoyo a la Consejería Distrital de Mujeres Indígenas

# Objetivo General

Realizar la migración del sistema CDMI (frontend en React y backend en Laravel) desde un entorno local (Apache en Windows 10 con PHP 8.0.30 y MySQL) hacia una infraestructura escalable y de alta disponibilidad en AWS, asegurando la continuidad operativa, integridad de datos y capacidad de crecimiento futuro.

# Inventario y dependencias

|  |  |  |
| --- | --- | --- |
| Componente | Versión Actual | Nuevo Entorno AWS |
| PHP | 8.0.30 | 8.2 en EC2 con Apache/Nginx |
| MySQL | mysqlnd 8.0.30 (cliente mysqli) | Amazon RDS – MySQL 8.x |
| Servidor Web | Apache 2.4.58 (Win64) | Apache/Nginx en Ubuntu 22.04 (EC2) |
| Sistema Operativo | Windows 10 | Ubuntu 22.04 LTS en AWS EC2 |
| Frontend | React | React en S3 + CloudFront (opcional) |
| Backend | Laravel | Laravel en EC2 (con balanceo si aplica) |

# Estrategias de Backup

Tipo de Backup

Bases de datos:

Diario (completo):  
mysqldump -u [user] -p[password] --all-databases > /backups/cdmi-full-$(date +%F).sql

Incremental (binlogs):  
mysqlbinlog /var/lib/mysql/mysql-bin.000\* >> /backups/cdmi-incremental-$(date +%F).sql

Aplicación (Laravel + React):

Respaldar código fuente:  
zip -r cdmi-code-$(date +%F).zip /ruta/cdmi

Almacenamiento:

- Subir respaldos automáticamente a AWS S3 (bucket con ciclo de vida de 30 días)  
- Opcional: réplica en Google Cloud Storage

Frecuencia:

- Diario: Backup incremental de MySQL  
- Semanal: Backup completo + código fuente

# Pruebas de restauración

- Simular pérdida de datos  
- Restaurar desde último backup  
- Validar integridad de datos y funcionalidad básica

# Plan de Migración

## 4.1 Preparación

- Realizar backup completo de base de datos y aplicación  
- Configurar:  
 - Amazon RDS (MySQL)  
 - Instancia EC2 Ubuntu 22.04 con PHP, Laravel, Apache  
- Documentar:  
 - Nuevas IPs, credenciales, accesos SSH, tokens, .env

## 4.2 Migración

1. Detener el acceso a la app, informar a usuarios  
2. Usar último backup incremental  
3. Subir y restaurar base de datos en RDS  
4. Subir archivos Laravel y React a EC2/S3  
5. Ajustar configuración Laravel (.env)  
6. Validar funcionamiento de frontend/backend  
7. Ejecutar pruebas funcionales (cliente y admin)

## 4.3 Post-migración

- Actualizar DNS y configuración de red  
- Activar monitoreo con CloudWatch (rendimiento, errores)  
- Notificar a los usuarios el restablecimiento del servicio

# Plan de Contingencia

Si la migración falla:  
- Restaurar último backup en el servidor antiguo  
- Revertir los cambios de DNS  
- Reanudar operación desde entorno local

# Cronograma

|  |  |  |  |
| --- | --- | --- | --- |
| Fase | Duración | Hora | Responsable |
| Backup Inicial | 1 hora | 8:00 | DBA |
| Migración BD | 3 horas | 9:00 | DevOps |
| Pruebas Funcionales | 2 horas | 12:00 | QA (cliente/admin) |
| Corte definitivo | 1 hora | 14:00 | Administrador Sys |

# Documentación Final

- Registro de errores  
- Manual de instalación en AWS  
- Lista de chequeo para futuras migraciones