# 8. Gestión de la Comunicación

Plan de comunicación

Para mantener el flujo de información técnica y administrativa, se ha establecido la siguiente matriz de comunicación:

| Qué se comunica | Medio / Canal | De quién a quién | Cuándo (Frecuencia) |
|---|---|---|---|
| **Nuevas Tareas y Bugs** | GitHub Issues (Tablero) | Desarrollador a Líder | Al identificar o iniciar una tarea |
| **Revisión de Código** | Pull Requests | Desarrollador a Admin | Al finalizar una funcionalidad |
| **Estado del Servidor** | Docker Logs / Dashboard | Sistema a Admin | Tiempo real (Monitor 24/7) |
| **Alertas Críticas** | WhatsApp / SMTP | Sistema a Usuario/Admin | Inmediato (Automático al fallar) |

Herramientas y medios

El proyecto se apoya en GitHub como plataforma central para el control de versiones, la gestión de tareas (Issues) y la revisión colaborativa de código. Para la parte operativa, se utiliza Docker Desktop que permite comunicar el estado de los servicios (Apache, MySQL) mediante registros en tiempo real, y la API de Twilio/WhatsApp para el envío automatizado de notificaciones críticas y códigos de verificación.

Ejemplo de informes

Los avances se reportan mediante el Historial de Commits, utilizando un formato estandarizado (ej. feat: Agrega validación (#42)) que sirve como bitácora de progreso. Asimismo, se genera un Reporte de Validación Automática mediante el script validate-docker.ps1, el cual confirma técnicamente que todos los servicios están "OK" antes de cualquier entrega.

# 9. Monitoreo, Control e Integración

**Plan de integración de áreas**

Para integrar las áreas de Frontend, Backend (PHP) y Base de Datos, se sigue estrictamente el flujo de trabajo **Git Flow**. Este proceso dicta que el desarrollo se realice de forma aislada en ramas específicas (ej. feature/#123) y que la integración final se haga únicamente mediante **Pull Requests** obligatorios. Además, el despliegue se gestiona de forma contenerizada con **Docker Compose**, lo que asegura que todos los módulos funcionen juntos de manera idéntica en cualquier entorno de desarrollo o producción.

**Indicadores (KPIs)**

El control del proyecto se mide mediante los siguientes indicadores técnicos:

- **Estado de Servicios:** Se busca mantener el 100% de los contenedores en estado "Up", verificado mediante el comando docker-compose ps.
- **Conectividad Backend:** Se monitorea la respuesta exitosa ("pong") en el endpoint de prueba test_ping.php.
- **Tiempo Real:** Se verifica la latencia de conexión del servicio WebSocket en test_websocket.html para asegurar el funcionamiento fluido del módulo de asistencia.

**Gestión de cambios**

La rama main se encuentra protegida para evitar errores en producción. Cualquier cambio en el sistema requiere seguir un protocolo estricto: primero, crear un **Issue** en GitHub describiendo el requerimiento; segundo, crear una rama derivada para trabajar el cambio (ej. fix/login-error); y tercero, aprobar una revisión de código obligatoria antes de poder fusionar los cambios. No se permiten modificaciones directas sobre la versión estable del proyecto.

**Gestión de aceptación**

Para que un módulo o funcionalidad se considere "Aceptado" y listo para producción, debe cumplir tres criterios:

1. El script de validación validate-docker.ps1 debe ejecutarse completamente sin reportar errores.
2. Las pruebas funcionales realizadas en las URLs de prueba (como login_test.html) deben ser exitosas.
3. El Pull Request correspondiente debe haber sido revisado y aprobado formalmente por el administrador del proyecto.

**Evidencia de seguimiento**

El seguimiento del proyecto queda documentado a través de:

- **Trazabilidad Git:** Un historial inmutable de todos los cambios realizados, identificando autor y fecha en el repositorio.
- **Logs de Contenedores:** Los registros de acceso y errores de los servicios Apache y PHP, accesibles mediante el comando docker-compose logs.
- **Respaldos de Base de Datos:** Los archivos JSON y SQL que se generan automáticamente en la carpeta /backups mediante el script cron_backup.php.