Informe final de cumplimiento – Moval

1. Requisitos del ERS que SÍ se implementan en la entrega

La siguiente tabla recoge los requisitos funcionales (RF) identificados en la plantilla IEEE 830 y su evidencia principal en el código.

ID (ERS)	Nombre	Evidencia en código
RF1.1	Asignar envío	assign_shipment.py
RF1.2	Desasignar envío	unassign_shipment.py
RF1.3	Modificar envío	update_shipment.py
RF1.4	Consultar detalles del envío	get_shipment_details.py
RF1.5	Cálculo del tiempo estimado de envío	calculate_eta.py
RF1.6	Entregar envío	deliver_shipment.py
RF1.7	Asignar paquete a cola	return_shipment_to_q.py
RF1.8	Notificar entrega del paquete	notify_delivery.py
RF1.9	Valorar entrega	rate_delivery.py
RF2.1	Inicio de jornada	start_workday.py
RF2.2	Final de jornada	end_workday.py
RF2.3	Cálculo del tiempo estimado de jornada	calculate_workday_duration.py
RF2.4	Consulta de la jornada	get_active_workday.py
RF2.5	Reportar incidencias	report_incident.py
RF2.6	Valorar jornada	rate_workday.py
RF3.1	Login	login.py
RF3.2	Consultar datos	update_user_data.py
RF3.3	Desplegar ayuda	get_help_content.py
RF3.4	Gestionar usuarios	change_user_role.py
RF3.5	Gestionar ajustes	manage_settings.py
RF3.6	Modificar datos	update_user_data.py
RF3.7	Seguridad y protección de datos	password_hasher.py
RF3.8	Registrar usuario	register_user.py

2. Requisitos del ERS que NO se implementan o que se han modificado

No implementados:

Tras el análisis del sistema desarrollado y su comparación con la especificación inicial de requisitos, se ha verificado que no existe ningún requisito que no haya sido implementado. Todos los requisitos definidos durante la fase de análisis han sido tenidos en cuenta y abordados durante el desarrollo del proyecto, encontrándose reflejados en la implementación final del sistema.

Modificados (detalle):

3. Requisitos NUEVOS implementados (no aparecían en el ERS)

Durante la implementación se añadieron funcionalidades adicionales que no estaban explícitas como RF en el ERS, pero que han sido necesarias para el correcto funcionamiento del programa, así como una mejora en la navegabilidad y uso del mismo; los archivos correspondientes son:

- change_user_role.py / RF3.9 - Cambiar rol de usuario
- errors.py / RF4.1 - Errores
- generate_delivery_route.py / RF1.10 – Generar ruta de entrega
- get_courier_profile.py / RF3.10 – Consultar perfil de repartidor
- list_available_couriers.py / RF1.11 – Obtener lista de repartidores
- list_pending_shipments.py / RF1.12 – Obtener envíos pendientes
- list_ratings.py / RF3.11 – Ver valoraciones
- list_shipments.py / RF1.13 – Ver paquetes
- pop_next_delivery_notification.py / RF4.2 – Activar siguiente notifcación de entrega

Todos estos RFs serán añadidos a los documentos correspondientes para que estén listos el día de la presentación del proyecto.

4. Comentarios sobre cambios/correcciones relevantes

- Se consolidó la lógica de permisos por rol (ADMIN/COURIER/CUSTOMER) en cada caso de uso, con errores consistentes (ValidationError/PermissionError/NotFoundError/ConflictError).
- Se introdujo inyección de dependencias (repositorios y Clock opcional) para facilitar pruebas y desacoplar lógica de negocio de infraestructura.
- Se normalizó el acceso a datos mediante repositorios (UserRepo, ShipmentRepo, CourierRepo, RatingRepo, WorkdayRepo, etc.).

5. Patrones de diseño GoF utilizados

- Strategy: uso de una abstracción de reloj (Clock) inyectable para obtener la hora (permite cambiar la estrategia de obtención de tiempo).
- Facade: el módulo de casos de uso actúa como fachada sobre varias operaciones de repositorio, exponiendo una interfaz simple a la UI.
- Factory (a nivel de wiring): en la inicialización de la app se instancian repositorios y casos de uso centralizando la creación de objetos.