Resumen de la Reunión Retrospectiva

Información de la empresa y proyecto:

Empresa / Organización	Duoc UC. Escuela de Informática y Telecomunicaciones
Proyecto	MisViáticos - Plataforma Integral de Gestión de Viáticos y Gastos Empresariales

Información de la reunión:

Lugar	Sala de reuniones de la empresa
Fecha	07/09/2025
Número de iteración / sprint	1
Personas convocadas a la reunión	SCRUM Master - Daniel Iturra Mario Bronchuer Carolina Pérez Jorge Rivas Paula Sandoval
Personas que asistieron a la reunión	Gabriel Muñoz Manuel Gonzales SCRUM Master - Valeria Vidal Daniel Iturra

Instrucciones:

La reunión retrospectiva es una herramienta del marco de trabajo Scrum, que pertenece a la familia de marcos de trabajo de desarrollo ágil, se realiza en cada iteración (denominado Sprint en Scrum), justo después de la reunión de revisión de la iteración (Sprint Review Meeting) con el dueño del Producto (Product Owner). En esta reunión deben revisarse tres aspectos, lo que salió bien durante la iteración (aciertos), lo que no salió tan bien (errores) y las mejoras que pudieran hacerse en la próxima iteración para evitar errores y mantener aciertos.

El dueño del producto (Product Owner) no asiste a la reunión, por lo que es una oportunidad para el equipo para poder hablar sin tapujos de los éxitos y fracasos, siendo importante para el equipo el analizar su propio desempeño e identificar estrategias para mejorar sus procesos. De forma similar, el Scrum Master (quien es el coach del equipo Scrum) puede observar impedimentos comunes que están afectando al equipo y tomar acciones para resolverlos.

La reunión usualmente se restringe a tres horas.

Formulario de reunión retrospectiva

¿Qué salió bien en la iteración? (aciertos)	¿Qué no salió bien en la iteración? (errores)	¿Qué mejoras vamos a implementar en la próxima iteración? (recomendaciones de mejora continua)
Migraciones de base de datos implementadas - Scripts de migración completados con Go Migrate - Seeds iniciales de categorías de gastos (Transporte, Alimentación, Alojamiento, Otros) - Datos de prueba para facilitar desarrollo y testing CRUD de gastos funcional - Endpoints REST implementados: GET, POST, PUT, DELETE - Validaciones de negocio aplicadas correctamente en el backend - Manejo de errores estructurado con códigos HTTP apropiados Arquitectura hexagonal aplicada correctamente	Subestimación del tiempo para EDT y MDI - Requirió horas extras al final del sprint para completar la documentación - Falta de experiencia en descomposición de estructura de trabajo Indefinición inicial del alcance del MVP - Retrabajos en el documento de alcance por falta de claridad sobre límites - Dificultad para priorizar funcionalidades esenciales vs deseables Validación tardía de Google Vision OCR - No validamos tempranamente limitaciones de cuota del servicio - Descubrimiento de restricciones técnicas al final del sprint	Implementar TDD (Test-Driven Development) - Escribir tests antes del código de producción - Garantizar cobertura mínima del 80% en servicios críticos - Usar mocks para aislar dependencias externas Automatizar generación de documentación Swagger - Usar anotaciones en código Go para generar docs automáticamente - Integrar swagger-ui para visualización interactiva - Incluir ejemplos de request/response en cada endpoint Crear archivo .env.example - Documentar todas las variables de entorno necesarias - Incluir valores de ejemplo (no sensibles) para facilitar setup

¿Qué salió bien en la iteración? (aciertos)	¿Qué no salió bien en la iteración? (errores)	¿Qué mejoras vamos a implementar en la próxima iteración? (recomendaciones de mejora continua)
- Separación clara entre domain, ports, adapters y services - Lógica de negocio independiente de infraestructura - Facilita testing unitario y cambios futuros de tecnología Configuración de PostgreSQL optimizada - Connection pooling configurado con parámetros óptimos - Manejo correcto de transacciones para operaciones críticas - Índices creados en campos de búsqueda frecuente Integración AWS S3 completada - Almacenamiento de recibos funcional y probado - Generación de URLs firmadas para descarga segura - Organización de archivos por tenant y fecha	Retraso en endpoint de carga de recibos - Problemas con configuración de CORS en el servidor - Límites de tamaño de archivo no configurados correctamente - Requirió 2 días adicionales de debugging Falta de pruebas unitarias - Servicios desarrollados sin tests debido a priorización de features - Deuda técnica acumulada que deberá abordarse en próximo sprint - Riesgo de regresiones en futuras modificaciones Documentación Swagger incompleta - No se completó la documentación de endpoints desarrollados - Dificultó las pruebas del frontend y comunicación del contrato API - Mario tuvo que leer código fuente para entender payloads	- Agregar validación de variables requeridas al inicio de la aplicación Establecer Definition of Done - Código revisado por peer review - Pruebas unitarias pasando (cobertura mínima 80%) - Documentación actualizada (Swagger + README) - Logs estructurados implementados - Sin warnings de linter o security scanner Setup técnico al inicio del sprint* - Dedicar primeras 2 horas a configuración de entorno y dependencias - Resolver bloqueos técnicos antes de comenzar desarrollo de features - Validar que ambos desarrolladores pueden ejecutar proyecto localmente Pair programming entre backend y frontend - Sesiones de 2 horas al inicio de cada módulo

¿Qué salió bien en la iteración? (aciertos)	¿Qué no salió bien en la iteración? (errores)	¿Qué mejoras vamos a implementar en la próxima iteración? (recomendaciones de mejora continua)
Mejora en comunicación del equipo - Daily standups de 15 minutos funcionando efectivamente - Tablero Kanban en GitHub Projects actualizado diariamente - Bloqueos resueltos con mayor rapidez - Coordinación fluida durante la fase de planificación - Reuniones de sincronización semanales establecidas	Gestión de credenciales AWS insegura - Credenciales inicialmente hardcodeadas en código - Requirió reconfiguración completa de variables de entorno - Tiempo perdido en solucionar vulnerabilidad de seguridad No se inició desarrollo del frontend - Sprint planning sobreestimó capacidad del equipo - Backend tomó más tiempo del estimado - Mario quedó bloqueado esperando endpoints disponibles	- Alinear contratos de API tempranamente - Definir estructura de DTOs y validaciones en conjunto - Evitar malentendidos sobre formato de datos

Nota:

- Se recomienda utilizar viñetas (bullets) para enumerar los aciertos, errores y recomendaciones de mejora continua.
- El formulario se puede extender cuantas páginas sea necesario para registrar todos los aciertos, errores y recomendaciones.