Resumen de la Reunión Retrospectiva

Información de la empresa y proyecto:

Empresa / Organización	Duoc UC. Escuela de Informática y Telecomunicaciones
Proyecto	MisViáticos - Sistema de Gestión de Gastos Corporativos

Información de la reunión:

Lugar	Sala de reuniones / Reunión virtual
Fecha	17/11/2024
Número de iteración / sprint	2
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)
Arquitectura y Desarrollo: - Implementación exitosa de arquitectura hexagonal (ports/adapters/services/domain) que facilitó la escalabilidad y mantenibilidad del código - Sistema multi-tenant funcional con separación clara entre base de datos de control y bases de datos por tenant - Integración exitosa con Google Cloud Vision para OCR de comprobantes chilenos (boletas, facturas) - Desarrollo del módulo de Analytics con filtros avanzados y exportación a Excel/PDF Funcionalidades Completadas: - Sistema completo de gestión de gastos con categorías, políticas y validaciones	Técnicos: - Retrasos en la implementación del sistema de pagos/integración con pasarelas - Complejidad subestimada en queries cross-database (usuarios en DB control, datos en DB tenant) - Falta de tests unitarios y de integración desde el inicio del desarrollo - Documentación técnica (Swagger/OpenAPI) no se generó durante el desarrollo Proceso: - Algunos cambios de alcance no fueron documentados formalmente (ej: agregar ViatiScan con IA) - Dependencia de conocimiento en miembros específicos para ciertos módulos - Reuniones de sincronización insuficientes en momentos críticos - Estimaciones de tiempo optimistas para features complejas (OCR, Analytics) Comunicación:	Calidad de Código: - Implementar tests unitarios obligatorios antes de crear PR (mínimo 70% cobertura) - Configurar CI/CD con GitHub Actions para ejecutar tests automáticamente - Realizar code reviews más exhaustivos con checklist definido - Implementar análisis estático de código (linting, security scanning) Documentación: - Generar documentación Swagger/OpenAPI para todos los endpoints - Documentar decisiones técnicas en ADRs (Architecture Decision Records) - Crear manual de despliegue y configuración de ambiente - Mantener README actualizado con instrucciones de setup Proceso: - Establecer reuniones de sincronización diarias (daily standups) de 15 minutos

¿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)
- Workflow de aprobación multinivel con escalamiento automático - Sistema de reportes con estados personalizables por tenant - Módulo de Export Templates para generación de reportes personalizados - Integración con Firebase Cloud Messaging para notificaciones push - Sistema de fondos por rendir con flujo de solicitud y aprobación Proceso de Trabajo: - Uso efectivo de Git Flow con ramas feature/issue-XX para cada funcionalidad - Code reviews mediante Pull Requests antes de mergear a development - Containerización con Docker que facilitó el ambiente de desarrollo consistente - Uso de herramientas de IA para acelerar el desarrollo y debugging	- Falta de documentación de decisiones técnicas tomadas durante el desarrollo - Información de configuración de servicios externos (API keys, credenciales) no centralizada	- Realizar sesiones de "knowledge sharing" semanales para distribuir conocimiento - Definir Definition of Done (DoD) clara para cada historia de usuario - Implementar planning poker para estimaciones más precisas Técnico: - Centralizar configuración de secretos con variables de entorno documentadas - Implementar monitoreo y logging estructurado para producción - Crear ambiente de staging para pruebas pre-producción - Establecer estrategia de backup y recuperación 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.