UNIVERSIDAD DEL VALLE DE GUATEMALA

Ingeniería de Software – Sección 21

Ing. Pablo Barreno

Tarea 1 - Reflexión sobre Lean SW

Osman Emanuel de León García – 23428

Milton Giovanni Polanco Serrano - 23471

Gadiel Amir Ocaña Véliz - 231270

Guatemala, 04 de abril de 2025

Tarea 1 - Reflexión de Aplicación de Lean Software Development

Tabla de Principios Lean y su Aplicación en el Proyecto

Principio	**Porcentaje**	**Explicación**	**Mejoras**
Eliminar desperdicios	70%	- Código no utilizado: Eliminamos funciones obsoletas durante la migración a Supabase. Sin embargo, quedan algunas variables sin uso en el frontend. - Retrasos: Tuvimos retrasos en la integración con Microsoft OAuth debido a complejidades técnicas no anticipadas. - Documentación: La documentación técnica la actualizamos tarde en algunos casos, lo que generó confusión. - Trabajo parcial: La implementación de Microsoft OAuth no la completamos.	- Refactorización: Usar herramientas como ESLint para identificar y eliminar variables/funciones no utilizadas. - Buffer de tiempo: Asignar un 20% adicional para integraciones externas. - Documentación ágil: Actualizar la documentación en paralelo al desarrollo. - Priorización: Usar técnicas MoSCoW para evitar trabajo parcial.

Amplificar el aprendizaje	85%	- Programación en pareja: Lo aplicamos en tareas críticas como la migración a Supabase. - Revisiones de código: Las implementamos en GitHub con pull requests. - Sesiones de conocimiento: Realizamos consistentemente sesiones de mejora para el proyecto de corta duración. - Falta de pruebas unitarias: Componentes como la encriptación no tenían cobertura suficiente.	- Spikes técnicos: Investigar tecnologías complejas (ej: Azure AD) antes del sprint. - Cobertura de pruebas: Aumentar al 90% con Jest (backend) y Cypress (frontend). - Wiki interna: Crear una base de conocimiento en Notion para lecciones aprendidas (ej: configuración de Supabase).
Tomar decisiones tarde	60%	- Tecnologías: Decidimos usar Supabase tarde en el proyecto, lo que requirió migración costosa. - Diseño de DB: Optamos por esquemas desnormalizados después de identificar cuellos de botella. - Proveedores OAuth: Añadimos Microsoft en	- Prototipado rápido: Validar opciones técnicas (ej: MongoDB vs Supabase) con POCs. - Decisiones reversibles: Usar feature flags para integraciones. - Matriz de evaluación: Comparar opciones con criterios como costo,

		Sprint #4 sin evaluar alternativas.	rendimiento y seguridad.
Entregar lo antes posible	90%	- Sprints cortos: Entregas cada 2 semanas con features funcionales (ej: JWT en Sprint #3). - MVP: Se priorizaron historias críticas (registro, gráficos). - Feedback temprano: Testeo con usuarios en Fase 3 de Design Studio. - Retraso en HU19: Microsoft OAuth no se entregó en Sprint #4.	- Continuous Delivery: Automatizar despliegues con GitHub Actions. - Feature toggles: Liberar funcionalidades incompletas pero deshabilitadas. - Priorización dinámica: Revisar el backlog semanalmente con el cliente.
Potenciar el equipo	80%	- Autonomía: Por cada miembro elegimos tareas según expertise (ej: Gadiel en DevOps). - Carga desigual: Gadiel asumió más trabajo en la migración. - Retrospectivas: Realizamos, pero sin seguimiento de acciones.	- Pair programming rotativo: Distribuir tareas complejas (ej: Milton y Osman en OAuth). - Planificación colaborativa: Usar Planning Poker para estimaciones. - Tablero de progreso: Visualizar carga de

		- Comunicación: Daily meetings efectivos, pero tuvimos cuellos de botella.	trabajo con herramientas como Jira. - Empoderamiento: Delegar decisiones técnicas a dueños de módulos.
Crear integridad	75%	- CI/CD: Implementado con GitHub Actions para backend. - Pruebas: Pruebas E2E con Cypress (85% cobertura). - Seguridad: Vulnerabilidad en /accounts por falta de validación de roles. - Deuda técnica: Algunos componentes no tienen tests unitarios.	- Pipeline robusto: Añadir steps de seguridad (SAST con SonarQube). - Pruebas unitarias: Cubrir módulos críticos (ej: encriptación) con Jest. - Monitorización: Usar New Relic para detectar bugs en producción. - Automatización: Escaneo diario de vulnerabilidades.

Visualizar el conjunto	65%	- Arquitectura: Diseñamos un diagrama inicial pero no se actualizó. - Dependencias: No anticipamos el impacto de Supabase en el frontend. - Lean UX Canvas: Las usamos para hipótesis (ej: rendimiento de Supabase), pero no se aplicó a todas las features. - Interoperabilidad: Nos faltó análisis de integración con otros sistemas (ej: bancos).	- Diagramas actualizados: Usar C4 Model para documentar arquitectura. - Impact mapping: Visualizar relaciones entre features y objetivos. - Workshops: Sesiones mensuales para alinear visión técnica y de negocio. - Herramientas: Miro para mapear dependencias y riesgos.

Conclusión

El equipo ha aplicado los principios Lean en un 75% general, destacando en entregas tempranas (90%) y aprendizaje continuo (85%), pero con oportunidades en toma de decisiones tardía (60%) y visualización del conjunto (65%). Los principales desafíos fueron la gestión de tiempo en integraciones externas y la falta de pruebas automatizadas. Para mejorar:

1. Automatización: Incrementar cobertura de pruebas y CI/CD (meta: 95%).
2. Planificación: Introducir spikes técnicos y buffers para tareas complejas.
3. Colaboración: Implementar pair programming rotativo y retrospectivas accionables.
4. Documentación: Mantener diagramas y wikis actualizados en tiempo real.

Con estas acciones, se proyecta alcanzar un 85-90% de adherencia a Lean en este semestre, priorizando calidad, aprendizaje y entrega de valor.