Universidad del Valle de Guatemala Ingeniería en Software 2 **Estudiantes:** 22272, Bianca Renata Calderón Caravantes 22473, Madeline Nahomy Castro Morales 22716, Aroldo Xavier López Osoy 22233, Daniel Eduardo Dubon Ortiz 22386, Flavio André Galán Donis 18020, Ángel Martín Ortega Yung	**Fecha:** Viernes, 05 de julio 2024 **Sección**: 20 **Docente**: Cristián Muralles

TAREA 1

Reflexión de Aplicación de Lean Software Development

Principio	Porcentaje	Explicación	Mejoras
Eliminar desperdicios	80%	Hemos trabajado en eliminar código innecesario y mejorar la calidad del software con pruebas exhaustivas y revisiones de código frecuentes. Sin embargo, aún enfrentamos desafíos que han limitado nuestro progreso completo. Por ejemplo, hemos experimentado algunos errores durante el registro de usuarios y dificultades en los despliegues en github pages.	Para mejorar es importante que nosotros como equipo abordemos de manera prioritaria los errores detectados durante el registro de usuarios y los problemas recurrentes en los despliegues. Se deben implementar controles más rigurosos durante el proceso de registro para garantizar que toda la información se capture correctamente. Además, es necesario revisar y fortalecer los procedimientos de despliegue para minimizar las interrupciones y asegurar actualizaciones sin contratiempos.

Amplificar el aprendizaje	100%	En nuestro equipo de desarrollo fomentamos activamente una mentalidad de aprendizaje continuo, en este caso JavaScript, React, CSS inline, etc. Nos centramos en el intercambio de conocimientos y experiencias entre colegas a través de diversas estrategias. Estas incluyen la programación en pareja, que nos permite enfrentar retos en equipo; revisiones de código, que fomentan un diálogo constructivo sobre nuestras prácticas; la creación de documentación exhaustiva; el uso de una wiki que acumula progresivamente nuestro conocimiento colectivo; la implementación de código completamente anotado para facilitar su comprensión; y sesiones de intercambio y formación continua, que enriquecen nuestro entendimiento y habilidades de manera colectiva.	Actualmente, consideramos que nuestro enfoque para amplificar el aprendizaje es efectivo y robusto, ya que incorpora múltiples facetas y técnicas que promueven el desarrollo profesional y personal continuo. No identificamos áreas específicas de mejora en este momento porque siempre buscamos apoyarnos mutuamente y adaptarnos a nuevas oportunidades de aprendizaje. Sin embargo, permanecemos abiertos a la retroalimentación y dispuestos a incorporar nuevas estrategias que puedan surgir en el futuro, asegurando que nuestro ambiente de aprendizaje continúe siendo dinámico y enriquecedor.

Tomar decisiones lo más tarde posible	90%	El inicio del proyecto fue lento porque ya teníamos un plan, pero lo fuimos cambiando para que este fuera más eficiente. Nos tomamos nuestro tiempo en la toma de decisiones porque buscamos la mejor manera en la que se puede implementar el código. Nos enfocamos principalmente en el funcionamiento antes de empezar a diseñar la interfaz. Sin embargo, esto también llevó a que el código de cada integrante fuera distinto, por lo que ahora tomamos el tiempo para que haya más coherencia entre el código.	El 10% restante representa cuando fallas en aportar coherencia en un primer punto y en opinar grupalmente en cuanto funcionamiento, botones, vista, etc. De lo contrario, no hay muchas mejoras a implementar pues ya hemos estado usando prototipos y pruebas de concepto para validar ideas antes de tomar decisiones definitivas, realizamos un figma de lo que esperamos sea el funcionamiento y el visual y al aceptar el mismo procedemos con el código.
Entregar lo antes posible	90%	Como bien se redactó en el punto anterior al principio no tuvimos muchas entregas de software debido a los cambios que se tuvieron al principio del proyecto. Pero en entregas más recientes siempre le hemos dado prioridad a las user stories para siempre tener 1 o 2 features funcionales para las entregas de software.	Consideramos que deberíamos realizar pruebas live, al momento antes de realizar el deploy para que no se nos escapen bugs en medio de una entrega de software (basado en historias de usuario).
Potenciar el equipo	100%	Consideramos que con lo que a	No consideramos que haya nada que

		“Potenciar al equipo” se refiere, todos los miembros del equipo en la toma de decisiones importantes, como la estimación de tiempo y la priorización de tareas en los sprints de SCRUM. Utilizamos Plan It Poker para que cada integrante aporte su perspectiva, lo cual aumenta el compromiso y la sensación de responsabilidad individual.	mejorar en este aspecto, debido a que esta modalidad nos ha ayudado a avanzar bastante con las recientes entregas de software y también dicha técnica para puntuar las tareas fue brindada por los ingenieros que nos guiaron durante el primer sprint.
Crear la integridad	75%	Nuestro software cuenta con pruebas unitarias y de integración. Además cuenta con CD para publicarse de forma automática a github pages y a hacer deploy del backend en AWS.	El 25% restante se debe a que las pruebas de integración del backend no se pueden correr de forma automatizada dentro de github actions y tampoco se tienen end to end tests.
Visualizar todo el conjunto	100%	Cada sprint nos reunimos con los ingenieros y las doctoras para reevaluar prioridades y cómo encajan dentro de la visión del software que estamos desarrollando. En caso se encuentre una mejora, se agrega al backlog para ser retomado en los siguientes sprints.	No consideramos que haya que mejorar nada en este aspecto ya que los ingenieros nos han ayudado de gran manera a lo largo de la planeación del proyecto y las doctoras se han comunicado con nosotros de forma abierta.