INSTITUTO TECNOLÓGICO DE COSTA RICA
Escuela de Ingeniería en Computación
IC-4810: Administración de Proyectos

# Proyecto: Dog Lovers

## Lecciones Aprendidas

Profesora:
Adriana Álvarez Figueroa

Integrantes:

Daniela Alvarado Andrade - 2021004342
Tamara Nicole Rodríguez Luna - 2021077818
Jonathan Andrey Porras Sandi - 2021069645

13 de noviembre del 2023

# Tabla de contenidos:

# Introducción:

Este documento tiene como objetivo proveer una compilación estructurada y directa de las experiencias vividas por estudiantes de Ingeniería en Computación durante la realización del proyecto "Dog Lovers". Se presenta un conjunto de lecciones que emergieron de los retos y aciertos en el transcurso del proyecto, ofreciendo así un recurso práctico para futuros proyectos.

Los estudiantes se enfrentaron a varios errores típicos en la gestión y ejecución del desarrollo de software, y cada uno de estos ha sido cuidadosamente desglosado en este documento. Se han definido categorías que engloban áreas críticas como la planificación, la comunicación, las pruebas y el mantenimiento, proporcionando en cada una exploración detallada del problema, el impacto que tuvo y las recomendaciones para prevenir o reducir dichos problemas en adelante.

No obstante, este informe no se detiene únicamente en los obstáculos encontrados, sino que también pone de manifiesto los éxitos y prácticas efectivas que contribuyeron al progreso del equipo. Estos aspectos positivos son presentados con la intención de que se reconozcan y adopten en proyectos futuros.

# Lecciones Aprendidas:

| Categoría | Problema/Éxito | Impacto | Recomendación |
|---|---|---|---|
| Planificación y estimación | Estimación de plazos inexacto | Retrasos en el proyecto, mala planificación de recursos, entregas finales afectadas, y aumenta el estrés en el equipo. | Emplear técnicas de estimación, utilizar herramientas de gestión de proyectos, y revisar regularmente las estimaciones con base en el progreso del proyecto y hacer los ajustes necesarios. |
| Comunicación y colaboración | No dar avances periódicos del prototipo a la profesora | Producto final que no cumple con las expectativas o requerimientos, y la pérdida de oportunidades de retroalimentación | Implementar reuniones de avance del prototipo con la profesora y el equipo. |
| Calidad y Pruebas | No agregar datos reales en algunas ventanas del prototipo | Falta de realismo en pruebas, malentendidos sobre la funcionalidad y el rendimiento del sistema. | Utilizar datos simulados que emulen la realidad y realizar pruebas para validar la precisión y relevancia. |
| Calidad y Pruebas | No realizar pruebas de usabilidad | Menor calidad en la experiencia del usuario, lo que puede llevar a una menor adopción y satisfacción del usuario final. | Incluir sesiones de pruebas de usabilidad como wireframe testing y first-click testing en las etapas tempranas y a lo largo del desarrollo del proyecto. |
| Desarrollo y | No implementar la | Aumenta el riesgo | Utilizar |

| | | | |
|---|---|---|---|
| Mantenimiento | entrega continua del trabajo | de errores en la integración y retrasa la detección de problemas. | herramientas de integración y entrega continua y mantener un enfoque de desarrollo ágil. |
| Calidad y Pruebas | Ignorar ciertos aspectos de la accesibilidad del software, como la opción de alto contraste, aumento de letra, navegación con teclado, etc. | El sistema excluye a usuarios con discapacidades y puede violar los estándares de accesibilidad requeridos. | Seguir las pautas de accesibilidad como WCAG y realizar pruebas de accesibilidad durante el desarrollo. |
| Planificación y estimación | Subestimar la complejidad del proyecto | Mala asignación de recursos y tiempos, resultando en una sobrecarga del equipo y en la posible entrega de un producto de baja calidad. | Realizar un análisis de riesgos y complejidad al principio del proyecto y ajustar la planificación en consecuencia. |
| Planificación y estimación | No realizar un seguimiento adecuado del progreso del proyecto | Dificultad para identificar desviaciones del plan y corregirlas a tiempo, resultando en retrasos y problemas de calidad. | Utilizar metodologías ágiles y herramientas de seguimiento de proyectos como Jira o Trello. |
| Planificación y estimación | Dividir el grupo equitativamente en un grupo de codificación y en otro de documentación | Puede crear silos de información y falta de comprensión integral del proyecto por parte de todos los miembros del equipo. | Fomentar un ambiente de equipo cruzado donde todos los miembros participen en diferentes tareas y aprendan unas de otras. |
| Comunicación y colaboración | Conflictos internos en el equipo | Disminución de la moral, colaboración deficiente y potencial | Establecer reglas claras de trabajo en equipo y resolver |

| | | | |
|---|---|---|---|
| | | fracaso en la entrega del proyecto. | conflictos a través de mediación y establecimiento de metas comunes. |
| Calidad y Pruebas | Falta de revisión de código entre pares. | Mayor probabilidad de errores, menor calidad del código y falta de transferencia de conocimiento entre los miembros del equipo. | Implementar un proceso de revisión de código y fomentar la cultura de aprendizaje y mejora continua. |
| Desarrollo y Mantenimiento | No identificar claramente los requisitos no funcionales del sistema | Problemas de rendimiento, seguridad y escalabilidad en etapas posteriores del proyecto. | Realizar sesiones para definir y documentar los requisitos no funcionales desde el inicio. |
| Desarrollo y Mantenimiento | No asignar tareas adecuadamente entre los 14 participantes desde el principio del proyecto | Desbalance en la carga de trabajo y potencial subutilización o sobrecarga de los miembros del equipo. | Usar una matriz de asignación de responsabilidades como RACI y asegurarse de que las tareas estén equilibradas y claras para todos. |
| Calidad y pruebas | No escribir pruebas unitarias | Aumenta el riesgo de defectos y problemas de calidad en el software, dificultando las fases de mantenimiento. | Incorporar la escritura de pruebas unitarias como una práctica estándar en el ciclo de vida del desarrollo |
| Calidad y pruebas | No tener un entorno de pruebas dedicado | Aumento del riesgo de errores en producción debido a la falta de validación en un entorno similar. | Configurar un entorno de pruebas que imite el entorno de producción lo más cercanamente posible. |
| Desarrollo y | No tener un proceso | Dificultades para | Establecer y |

| | | | |
|---|---|---|---|
| Mantenimiento | de rollback claro para las versiones. | revertir cambios en caso de que una nueva versión cause problemas. | documentar un plan de rollback antes de desplegar nuevas versiones. |
| Comunicación y colaboración | Falta de comunicación en el equipo | Malentendidos, reducción de eficiencia y productividad, falta de cohesión en el equipo y toma de decisiones inadecuadas | Establecer rutinas de comunicación regulares, asegurándose que todo el equipo esté enterado de cambios o situaciones que estén sucediendo. Además, usar canales de comunicación efectivos y herramientas colaborativas |
| Calidad y pruebas | No realizar pruebas de carga y estrés. | Incertidumbre sobre cómo se comportará el sistema bajo condiciones de uso elevadas, lo que podría resultar en caídas y pérdida de datos. | Incluir pruebas de carga y estrés como parte del ciclo regular de pruebas. |
| Calidad y pruebas | No documentar los resultados de las pruebas | Falta de referencia para futuras iteraciones, dificultades para validar la cobertura de pruebas y para cumplir con estándares de calidad. | Crear plantillas de documentación para resultados de pruebas y mantener un registro actualizado |
| Calidad y pruebas | No tener una persona o equipo de pruebas dedicado. | Menor calidad del software, ya que las pruebas pueden ser inconsistentes y no tan rigurosas. | Asignar roles específicos para pruebas o establecer un equipo pequeño dedicado a esta |

| | | | |
|---|---|---|---|
| | | | tarea. |
| Comunicación y colaboración | No tener un registro de decisiones y cambios del proyecto. | Pérdida de rastro de la justificación de decisiones, dificultades en la gestión de cambios y confusión en el equipo. | Mantener un registro de decisiones y cambios con las fechas, razones y personas involucradas |
| Gestión de Riesgos | No medir el progreso con KPIs (Indicadores Clave de Rendimiento) relevantes. | Dificultad para evaluar la eficiencia del proceso de desarrollo, identificar problemas y medir el éxito del proyecto. | Definir KPIs claros al inicio del proyecto, como porcentaje de tareas completadas, bugs por semana, tiempo de entrega de características, etc., y hacer seguimiento regularmente. |
| Comunicación y colaboración | No darle seguimiento a la herramienta de gestión de proyecto seleccionada (Trello) | Pérdida de visibilidad del progreso del proyecto, tareas olvidadas o desatendidas y mala gestión de prioridades. | Designar a un miembro del equipo como responsable de la actualización y monitoreo de la herramienta, y realizar revisiones periódicas de su estado. |
| Comunicación y colaboración | No celebrar el éxito y los hitos alcanzados. | Desmotivación del equipo y falta de reconocimiento al esfuerzo y logros, lo que puede afectar el rendimiento general | Establecer hitos claros y celebrarlos con el equipo al alcanzarlos, ya sea mediante reconocimientos, reuniones de celebración o menciones especiales. |
| Gestión de riesgos | No analizar y actualizar la matriz de riesgos | Reacción lenta ante los problemas emergentes, y | Realizar reuniones semanales para revisar y actualizar |

| | | | |
|---|---|---|---|
| | semanalmente | potencial fallo en la mitigación de riesgos. | la matriz de riesgos, involucrando a todo el equipo en la identificación y manejo de nuevos riesgos |
| Planificación y estimación | No planificar una estrategia de backup y restauración de la base de datos. | Riesgo significativo de pérdida de datos y tiempo de inactividad prolongado en caso de fallas. | Definir y documentar una política de backup y restauración, y realizar pruebas periódicas de los backups. |
| Planificación y estimación | No planificar la capacidad del servidor y los recursos de almacenamiento | Problemas de rendimiento y escalabilidad, y potencialmente costos excesivos por sobre o infra dimensionamiento. | Realizar un análisis de la demanda esperada y planificar los recursos del servidor de acuerdo a esta, con posibilidad de escalamiento. |
| Gestión de riesgos | No considerar la posibilidad de tener que migrar a otro framework o tecnología. | Dificultad y alto costo en caso de que sea necesario cambiar de tecnología por requerimientos no previstos | Diseñar el sistema con un alto grado de modularidad y documentar las decisiones de arquitectura para facilitar posibles migraciones. |
| Desarrollo y mantenimiento | No considerar el mantenimiento post-lanzamiento del sistema | Problemas de soporte y actualización después del lanzamiento, afectando la satisfacción del usuario. | Planificar una fase de mantenimiento, incluyendo recursos y procesos para el manejo de bugs y nuevas características. |
| Desarrollo y mantenimiento | No tener un proceso claro de gestión de versiones. | Confusión y posibles conflictos en el código fuente, | Implementar un sistema de control de versiones como |

| | | | |
|---|---|---|---|
| | | complicando el desarrollo y la implementación. | Git y definir una estrategia de ramificación y etiquetado. |
| Planificación y estimación | Incorrecta estimación en la curva de aprendizaje de nuevas tecnologías | Retrasos en el proyecto debido a que los estudiantes tardan más en familiarizarse con las herramientas y tecnologías. | Realizar un análisis realista de las habilidades del equipo y proporcionar recursos para acelerar el aprendizaje. |
| Desarrollo y mantenimiento | No considerar la internacionalización y localización del software | Limitación en la expansión a otros mercados y falta de adaptabilidad cultural y lingüística del producto. | Diseñar el software desde el principio para soportar la internacionalización y localización |
| Desarrollo y mantenimiento | No establecer una biblioteca de componentes reutilizables. | Redundancia en el trabajo de desarrollo y mayor esfuerzo para mantener el código. | Crear y mantener una biblioteca de componentes y servicios comunes que puedan ser reutilizados en diferentes partes del proyecto. |
| Calidad y pruebas | No automatizar las pruebas cuando es posible. | Incremento en el tiempo y recursos necesarios para la realización de pruebas manuales y mayor probabilidad de errores humanos. | Invertir tiempo en la creación de pruebas automáticas para las partes críticas y repetitivas del software. |
| Diseño y retroalimentación | No utilizar diagramas para explicar la arquitectura del sistema. | Falta de entendimiento común de la estructura del sistema y dificultades en la incorporación de | Crear diagramas arquitectónicos como parte de la documentación y usarlos para comunicar las decisiones de diseño |

| | | | |
|---|---|---|---|
| | | nuevos miembros al equipo | |
| Desarrollo y mantenimiento | No utilizar inyección de dependencias para mejorar la mantenibilidad | Código más acoplado y difícil de mantener, testear y extender. | Aplicar principios de diseño de software como la inyección de dependencias para desacoplar componentes y facilitar el testing. |
| Gestión de riesgos | No manejar todas las excepciones del proyecto de manera coherente | Comportamiento impredecible de la aplicación y posibles fallos en el tiempo de ejecución. | Definir una estrategia de manejo de excepciones y asegurarse de que se aplique consistentemente en todo el proyecto. |
| Desarrollo y mantenimiento | No considerar la implementación de funcionalidades offline. | Disminución de la usabilidad del software en situaciones de conectividad limitada o inexistente | Evaluar las necesidades del usuario y considerar la implementación de características que puedan funcionar sin conexión. |
| Comunicación y colaboración | Hacer reuniones con el cliente (profesora) para tener claros los requerimientos del sistema y no hacer suposiciones | Asegura que el producto final cumpla con las expectativas del cliente, minimizando los malentendidos y los cambios de última hora | Mantener un calendario regular de reuniones y preparar agendas claras. Documentar y compartir los puntos discutidos y las decisiones tomadas. |
| Comunicación y colaboración | Definir criterios de aceptación claros | Facilita la evaluación de la calidad del software y la comprobación del cumplimiento de los requisitos. | Trabajar con el cliente para definir criterios medibles y revisarlos a lo largo del proyecto para garantizar su relevancia continua. |

| | | | |
|---|---|---|---|
| Comunicación y colaboración | Revisar múltiples veces los documentos del proyecto, asegurándose que todo el equipo estuviera satisfecho con ellos | Mejora la calidad de los entregables y asegura la alineación del equipo con el plan del proyecto. | Establecer revisiones periódicas de la documentación y utilizar herramientas de revisión colaborativa. |
| Comunicación y colaboración | Coordinar la integración del proyecto entre los diferentes equipos | Permite una colaboración efectiva entre los equipos y reduce los problemas de integración. | Utilizar prácticas de integración continua y herramientas de gestión de proyectos para coordinar las tareas. |
| Planificación y estimación | Realizar un análisis comparativo con aplicaciones similares antes de empezar el desarrollo del proyecto | Proporciona una comprensión más profunda del mercado y del espacio del problema, permitiendo diseñar un producto mejor informado. | Documentar las características de las aplicaciones competidoras y realizar análisis SWOT para identificar oportunidades y amenazas |
| Diseño y retroalimentación | Considerar el feedback del cliente (profesora) en el prototipo para mejorarlo y consecuentemente mejorar el sistema final | Aumenta la calidad y relevancia del producto final, alineándolo con las necesidades reales del usuario | Implementar un proceso formal para recoger, analizar y actuar sobre el feedback. |
| Comunicación y colaboración | Realizar reuniones antes de entregas importantes o en momentos de tomas de decisiones importantes | Asegura que el equipo esté informado y en acuerdo antes de momentos críticos, promoviendo la toma de decisiones efectiva. | Planificar estas reuniones con antelación y asegurar que todos los miembros tengan la información necesaria para contribuir. |
| Diseño y | Implementación | Mejora la usabilidad | Adoptar un enfoque |

| | | | |
|---|---|---|---|
| retroalimentación | exitosa del diseño responsive | y accesibilidad del sistema en diferentes dispositivos y tamaños de pantalla. | 'mobile-first' en el diseño y realizar pruebas continuas en múltiples dispositivos. |
| Diseño y retroalimentación | Creación de un prototipo inicial funcional | Permite validar ideas y obtener feedback temprano, reduciendo los riesgos y orientando el desarrollo posterior | Utilizar herramientas de prototipado rápido y fomentar la iteración basada en las pruebas de usabilidad y el feedback recibido. |

# ¿Para qué sirve documentar las lecciones aprendidas de un proyecto?

Documentar las lecciones aprendidas de un proyecto sirve como un recurso vital para la reflexión y la mejora continua tanto a nivel individual como organizacional. Este proceso consiste en identificar y analizar los éxitos y errores ocurridos durante la vida de un proyecto con el fin de capitalizar esa experiencia de manera constructiva en el futuro.

De esta manera, documentar lo que funcionó bien y lo que permite que los equipos apliquen esos conocimientos a proyectos similares en el futuro. Es una forma de consolidar la experiencia adquirida en una base de conocimiento accesible que puede servir como guía o punto de referencia. Además, reconocer las estrategias efectivas y las ineficiencias previas puede ayudar a evitar repetir errores y, por ende, ahorrar tiempo y recursos en esfuerzos futuros. Así, los equipos pueden ser más ágiles y eficientes al no tener que “reinventar la rueda”.

Las lecciones aprendidas también son un recurso de aprendizaje para el desarrollo de habilidades y la formación del equipo. El personal nuevo o menos experimentado puede beneficiarse enormemente del conocimiento acumulado, acelerando su curva de aprendizaje y su capacidad para contribuir efectivamente. En empresas con una alta rotación de personal o con equipos que cambian constantemente, documentar las lecciones aprendidas ayuda a retener el conocimiento crítico dentro de la organización, asegurando que la experiencia valiosa no se pierda.

Finalmente, al revisar lo que ha funcionado en el pasado, los líderes y administradores de proyecto pueden tomar decisiones más informadas y estratégicas, lo cual es especialmente valioso en situaciones complejas o bajo condiciones inciertas.