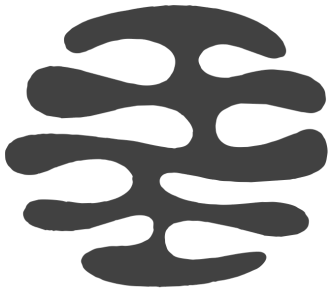

CARONTE

Lecciones aprendidas

Sprint 3

Grupo 9

Realizado por:

Chico Castellanos, Álvaro

Revisado por:

García Escudero, Ángel

Índice

Índice 2
Introducción 3
Tabla de lecciones aprendidas: 4

Introducción

La gestión del conocimiento y la mejora continua son elementos clave en la administración de proyectos, ya que permiten optimizar procesos, minimizar errores y fortalecer el desempeño del equipo. En este sentido, la identificación y documentación de las lecciones aprendidas resulta fundamental para consolidar buenas prácticas y corregir aquellos aspectos que han representado desafíos durante la ejecución del proyecto.

Para ello, se han identificado y analizado tanto las fortalezas como las áreas de mejora a lo largo del desarrollo. A partir de este análisis, se ha documentado cada problema encontrado junto con la solución implementada, con el objetivo de disponer de un registro que facilite su resolución en caso de que se presente nuevamente en este u otros proyectos. Esto permitirá optimizar la toma de decisiones y mejorar la eficiencia en futuras ocasiones.

Tabla de lecciones aprendidas:

ÁREA	INCIDENTE / PROBLEMA	LECCIÓN APRENDIDA	COMENTARIOS
Despliegue	La implementación de múltiples versiones en cortos periodos genera un sobrecoste, ya que el proceso de despliegue consume más recursos que la ejecución habitual de la aplicación.	Es más eficiente desplegar la base de datos por separado en Google Cloud, manteniendo el backend y frontend en Railway. Además, se han implementado configuraciones para reducir el consumo de RAM.	N/A
Comunicación	Falta de una estrategia clara para la gestión de la información y la comunicación en un equipo de gran tamaño.	Se han creado diferentes departamentos y roles para asegurar que la información llegue de manera equitativa a todos los miembros sin necesidad de reuniones constantes. Asimismo, se han establecido canales de comunicación específicos para cada área, evitando la dispersión de información.	N/A
Comunicación	La falta de definición precisa de los requisitos ha llevado a comenzar tareas sin un criterio claro, lo que ha generado cambios posteriores y trabajo adicional.	Es fundamental definir con exactitud todos los requisitos antes de comenzar cualquier tarea. Aunque esto implique un esfuerzo inicial mayor, a largo plazo evita correcciones y optimiza el tiempo de desarrollo.	N/A
Tecnología	Posibles incompatibilidades entre las versiones de las tecnologías instaladas en los equipos de los desarrolladores y las utilizadas en el proyecto.	Se recomienda el uso de un entorno virtual que contenga las versiones exactas de las tecnologías requeridas para el proyecto, asegurando coherencia y evitando conflictos de compatibilidad.	N/A
Tecnología	La versión gratuita de la herramienta de despliegue de Vercell solo permite asociar un único usuario de GitHub para gestionar el despliegue del repositorio. Como resultado, solo esa persona puede realizar commits en la rama de despliegue para actualizar la web.	Se ha implementado una metodología de trabajo basada en ramas colaborativas dentro del equipo. De este modo, todos los miembros pueden contribuir mediante pull requests, mientras que el usuario autorizado se encarga de fusionarlas en la rama principal (main), garantizando así una colaboración eficiente incluso con la versión gratuita de la herramienta.	N/A

Comunicación	Resolución desorganizada de problemas y errores en el código o documentación por parte de la persona que los detecta, sin tener un conocimiento preciso del contexto ni del propósito original, lo que puede generar inconsistencias o complicaciones adicionales.	Implementación de una metodología de comunicación en la que los errores sean reportados directamente a la persona responsable de esa parte del código o documentación, permitiéndole resolverlos de manera más eficiente, dado su mayor conocimiento y contexto sobre el trabajo realizado.	N/A
Documentación	Posibles conflictos en la política de ramas en GitHub debido a la falta de conocimiento preciso por parte del equipo, olvidos o interpretaciones erróneas sobre su funcionamiento, lo que puede generar problemas en la gestión del código.	Crear un documento detallado que explique con precisión el funcionamiento de la política de ramas, aclarando todas las posibles dudas y sirviendo como referencia para el equipo.	N/A
Documentación	Los miembros del equipo no responden de manera oportuna y efectiva a las lecciones aprendidas y al feedback semanal, lo que limita la mejora continua y la optimización del trabajo en equipo.	Establecer una política de penalizaciones y sanciones para quienes no respondan al feedback semanal, garantizando su cumplimiento. Además, se deberá hacer obligatorio que cada miembro del equipo proponga al menos una lección aprendida por semana para fomentar la mejora continua.	N/A
Despliegue	Los problemas en el despliegue se están volviendo frecuentes debido a que los cambios se suben a última hora, lo que obliga al encargado a realizarlo bajo presión y fuera del horario adecuado. Esta situación aumenta el riesgo de errores que no pueden ser solucionados a tiempo, afectando la estabilidad y eficiencia del proceso.	Establecer una política de penalizaciones y sanciones para aquellos miembros que, sin previo aviso, no cumplan con la fecha de entrega establecida. En caso de tener dificultades con el tiempo, se podrá coordinar con el encargado del despliegue para posponer esa parte de la funcionalidad y programarla para la siguiente semana, evitando así problemas en la estabilidad del sistema.	N/A

Commits	Algunos miembros del equipo no han seguido la política establecida de commits, utilizando nombres poco descriptivos o un formato inadecuado, lo que dificulta el seguimiento y la organización del código.	Se ha implementado un workflow que analiza los nombres de los commits para verificar el cumplimiento de la política definida. En caso de que no se sigan las directrices, la pull request que contenga esos commits no podrá fusionarse en la rama develop.	N/A
Backend/ frontend	El manejo de archivos grandes en base64 (como videos MP4) generó cadenas extremadamente largas, dificultando su prueba y procesamiento en herramientas como Postman	Comprimir el archivo antes de convertirlo a base64 reduce significativamente su tamaño, facilitando la transferencia de grandes archivos y mejorando la eficiencia del sistema.	N/A
Tecnología	Falta de conocimiento en la implementación de tecnologías para métodos de pago y la asociación de pagos o suscripciones a un usuario.	Se ha integrado Stripe en la aplicación para gestionar los métodos de pago y la asociación de suscripciones de manera eficiente.	N/A
Frontend	Problemas al probar funciones en dispositivos móviles han generado discrepancias con la versión desplegada, lo que puede provocar variaciones en la capa visual y afectar la funcionalidad.	Se ha implementado una metodología de versiones de pre-despliegue que permitan verificar el correcto funcionamiento de todas las funcionalidades antes del despliegue oficial.	N/A
Frontend	El desarrollo de un código excesivamente complejo en un solo archivo dificulta su posterior modularización, haciendo que la división de funcionalidades resulte complicada y poco eficiente.	Se ha priorizado la creación y reutilización de componentes, manteniendo el archivo principal lo más limpio y estructurado posible.	N/A

Documentación	Al redactar un acuerdo de usuario, existe el riesgo de incluir cláusulas abusivas de manera inadvertida, lo que puede generar desconfianza en los usuarios y disuadirlos de utilizar la aplicación.	Uso de Claudette, una IA en fase beta que permite identificar cláusulas abusivas en acuerdos de usuario redactados en inglés. Si bien su funcionamiento es eficiente, su análisis se centra únicamente en las oraciones individuales y no en el contexto general, por lo que sus resultados deben interpretarse con cautela.	N/A
Organización	La falta de respuesta por parte del equipo a las tareas de feedback y lecciones aprendidas ha limitado la mejora continua y la optimización del trabajo en equipo, dificultando la identificación y resolución de problemas recurrentes.	Se han incorporado cláusulas en el Commitment Agreement que establecen penalizaciones para aquellos miembros que no respondan a las tareas de feedback y lecciones aprendidas, con el objetivo de fomentar la responsabilidad y evitar la repetición de esta situación.	N/A
Frontend	Retraso de las tareas porque backend no tenía terminada la funcionalidad	Puede dejarse diseñado prácticamente entero las vistas de frontend con datos de prueba y una vez este hecho en backend cambiar los datos de prueba por las llamadas. En mi caso si hubiera esperado a que la gente de backend hubiera terminado la tarea hubiera sido imposible que la vista se hubiera terminado a tiempo para entregarse.	N/A
Backend	Integrar Codacy con spring-boot es complicado si no se tiene en cuenta la base de datos y el .env.	Debemos añadir el mismo ecosistema en el workflow ya que de lo contrario el contexto de la aplicación falla al no compilar del todo y se carga el análisis.	N/A
Documentación	Los README.md no eran profesionales.	Añadiendo badges a los README.md de los repos todo luce más profesional y además tenemos información de coverage sin tener que entrar en Codacy.	N/A

Backend	Hacer tests de carga era algo que no teníamos contemplado.	Usar un proyecto a parte con Locust para el tema de los endpoints ha ayudado mucho a hacerlos rápido y bien de manera efectiva.	N/A
Documentación	Conocer los endpoints de la API era tedioso.	Usar Swagger para hacer las Open API Specifications ayuda a conocer todos los endpoints en detalle.	N/A
Frontend	Cambio de componentes core de la aplicación, al modificarse, se han descuadrado muchas vistas	Revisar detenidamente todas las vistas, avisar repetidamente para que el resto de compañeros también puedan revisar su parte.	N/A
Frontend	Al registrar mal la tarjeta por primera vez y luego de poner bien la tarjeta en un navegador nuevo no dejaba aceptar el pago. Al realizar el pago por primera vez de la esquela ocurría un error en el pago, y no se realizaba y este error solo saltaba la primera vez que pagabas con ese navegador.(si pagabas creando otra cuenta incluso modo incógnito no volvía a saltar por lo que fue difícil replicar este error también).	Proporciona una explicación detallada acerca de los conocimientos adquiridos o acciones llevadas a cabo para la resolución del incidente o problema mencionado anteriormente. El problema detectado de momento era porque al abrir un nuevo navegador sin ningún tipo de datos de la página, el token no se creaba correctamente y con ello no se podía acceder a esa información para después realizar el pago. Era un fallo que ocurría solo la primera vez que usaba ese navegador, por lo que fue difícil replicar el error, como lección aprendida es que porque algo funcione en local no quiere decir que funcione en otro dispositivo, por lo que he aprendido que testear en otros dispositivos y navegadores y no conformarme probando solo el modo incógnito en un solo navegador es suficiente para saber si algo no falla.	N/A

		En cuanto al error del pago de la esquela aplicando lo mismo de antes, este caso era porque el payment_Id estaba declarado como un estado y por eso fallaba solamente la primera vez que se intentaba realizar el pago con un navegador nuevo, ya que después de este como que el estado ya se quedaba en true y no volvía a saltar ni en modo incógnito, fue difícil replicar el error porque era un fallo a nivel que implicaba el token también, pero en este caso la solución fue quitar la propiedad de pago como estado para que no comprobara que estuviera en el token antes que se generase siquiera la id del pago en el inicio.	
Despliegue	Nuevamente a la hora de desplegar había cambios sin mergear en develop, incluso errores, lo que nos ha traído a tener errores en producción.	Debemos comprobar mejor el correcto funcionamiento de develop antes de desplegar. Además se deben de cumplir las deadlines de Code Freeze.	N/A
Backend	Dime cómo veo la cantidad de cobertura que tiene el authController.	Usando mvn clean verify y después mvn jacoco:report.	N/A
Backend	Cuando se termina la implementación de una funcionalidad que tenga bastantes casos, es difícil que una persona externa pueda revisar con facilidad y rápidamente dicha funcionalidad.	Proporciona una explicación detallada acerca de los conocimientos adquiridos o acciones llevadas a cabo para la resolución del incidente o problema mencionado anteriormente. Lo más sencillo ha sido compartir la colección de Postman que ha utilizado el desarrollador mientras estaba realizando todo el código, además es una colección que puede editar todas las personas que	N/A

		tengan acceso, por lo que si otro desarrollador encuentra algún error puede ayudar a mantener todo funcionando correctamente. Esto también ayuda cuando hay dos personas trabajando a pares en una misma funcionalidad.	
Backend	Al intentar intentar mockear algunas funciones, a veces daba error a pesar de que el mockeo parecía estar correctamente formado.	Proporciona una explicación detallada acerca de los conocimientos adquiridos o acciones llevadas a cabo para la resolución del incidente o problema mencionado anteriormente. Resulta ser que al mockear es importante tener en cuenta el orden en el que se realizan los mocks y el uso de los matches, ya que en caso de utilizar matches en los mocks de forma incorrecta suceden errores imprevistos.	N/A
Backend	Problema con los test de backend, ya que han llevado mucho tiempo y son un poco complejos.	Cuando no sepamos hacer algo, usamos las tecnologías que tenemos a nuestro alcance para que nos ayuden y nos proporcionen diferentes soluciones.	N/A
Backend	Se ha roto la aplicación al refactorizar de manera descuidada.	Necesidad de test unitarios o un método eficiente para comprobar que no se rompe la funcionalidad.	N/A
Backend	Debido a la refactorización se retrasó una tarea en desarrollo y otra que dependía de esta primera.	Hay que identificar bien las batallas y cuando lucharlas. Creo que el mejor momento para refactorizar es el viernes después de la entrega de un sprint. Los cambios deben ser lo más rápidos posibles, controlados y posteriormente testeados. No vale que surjan problemas no previstos y que se retrase o tenga que rehacer el trabajo.	N/A

Backend/Frontend	Por no contactar con la persona creadora de un código o responsable de una tarea, se realizó un trabajo que ya se había resuelto en otra rama. Si el que cambia el código no lo hubiera hecho por su cuenta y sin avisar no habría trabajado en balde.	Contactar siempre y asegurarse que el trabajo que se vaya a realizar no esté hecho ya. No nos creamos superman, porque puede ser que estés trabajando para nada. No modificar código que no es tuyo sin avisar. Es posible que el fallo ya esté resuelto o que esa persona tarde 5 veces menos que si te tienes que poner a entender el código. Hay que interesarse por la tarea y el que la realizó.	N/A
Documentación	Retraso en la entrega de la documentación.	Pese a saber de los posibles problemas o retrasos que pueden tener algunas tareas, en concreto, debido a su complejidad o a diversos factores, se han establecido períodos de entrega más tempranos para poder hacer frente a posibles retrasos o problemas por desconocimiento. Aún así, ha quedado en evidencia que si las cosas no se hacen bien (aunque sea en las versiones de prueba) la entrega de estos documentos pueden llegar incluso a suspender la entrega.	N/A
Frontend	Hemos dejado para el final el testing en frontend sin tener un plan claro definido anteriormente.	El no haber tenido un plan/estrategia definida para el Testing de frontend desde el comienzo del proyecto ha causado que a la hora de implementarlo hayan surgido muchos problemas y falte tiempo. La lección aprendida sería realizar un plan de pruebas al comienzo del proyecto donde se detalle cuando se van a hacer los test, quién y sobre todo con que herramienta. En el caso de ser una herramienta nueva nos daría tiempo a estudiarla de antemano para cuando llegue la hora de trabajar los test, tener menos problemas	N/A

Marketing	La necesidad de una herramienta para la edición de vídeo gratuita.	Davinci Resolve es un editor de video muy fácil de usar, completo y gratis. Ha sido un gran descubrimiento.	N/A
Backend	Al probar en despliegue entrar a la información de ciertos usuarios resultaba saltar una excepción relacionada a comparar que la id del usuario. Esto sucedía de igual forma en ciertos otros casos, haciendo parecer que el error obtenido era por una mala obtención o parseo en backend de la información del JWT en los nuevos usuarios, ya que en los antiguos no daba problema, provocando con esto dudas de donde estaba en verdad el problema, ya que en el entorno local no sucedía dicha excepción.	Viendo en qué métodos fallaba ese error pude observar que en todos ellos se realizaba la comparación del id utilizando ==. Esta forma solo sirve en determinados casos en java, como pueden ser en los tipos primitivos y en algunos tipos envolventes (Long, Integer, etc.) hasta ciertos valores. En este caso lo que sucedía es que al tener ids de valor mayor a 127 provocaba error. La solución aplicada ha sido cambiar estos == por el método equals.	N/A

"Lecciones aprendidas". Se ha utilizado la IA para corregir errores ortográficos que hubieran pasado desapercibidos. Toda la información generada por IA ha sido revisada por el equipo Caronte.