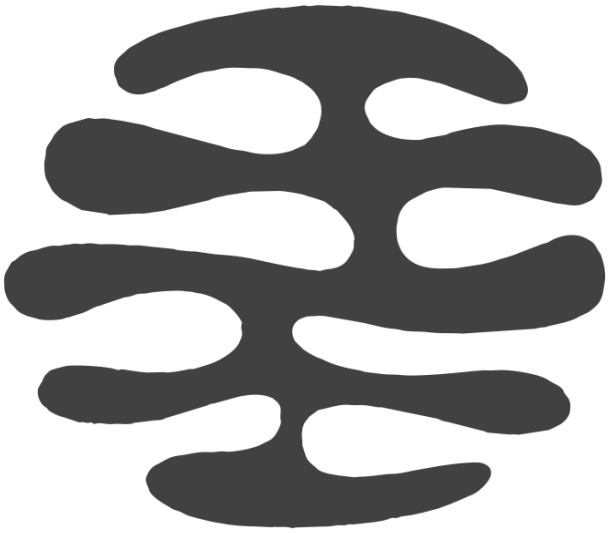

CARONTE

Sprint 3 Retrospective

Sprint 3

Grupo 9

Realizado por:

García Rivero, Andrés Francisco

Chico Castellano, Álvaro

Revisado por:

García Escudero, Ángel

Índice

Dificultades a las que nos enfrentamos

1. Autogestión en grupos grandes

Primera experiencia en la autogestión de un grupo de gran tamaño.

2. Desafíos técnicos en el desarrollo

Posibles dificultades técnicas durante el desarrollo del proyecto.

3. Limitaciones de disponibilidad

Disponibilidad limitada de los miembros del equipo debido a otras responsabilidades (prácticas, asignaturas, etc.).

4. Falta de experiencia en el nicho del proyecto

Falta de familiaridad con el nicho de negocio en el que se desarrolla el proyecto.

Puntos fuertes del equipo

1. Buen desempeño general y resolución ágil de problemas

Buen trabajo general por parte del equipo y rápida resolución de los problemas que han ido surgiendo.

2. Mejora en la estimación y planificación de tareas

Se ha notado una mejora en la estimación de tareas, lo que ha permitido una planificación más realista y eficiente.

3. Colaboración constante y actitud de ayuda mutua

Excelente colaboración entre los miembros del equipo, con una actitud constante de ayuda mutua.

4. Gestión eficaz ante la baja de un compañero

Se ha logrado cubrir con éxito la baja de un compañero, asumiendo su carga de trabajo sin afectar al ritmo del equipo.

5. Resolución rápida de bugs críticos

Los bugs críticos han sido resueltos con agilidad, minimizando su impacto en el desarrollo.

6. Actitud positiva frente a críticas constructivas

El equipo ha mostrado una actitud positiva frente a las críticas, enfocándose en la mejora continua.

7. Mejora notable del frontend reflejada en Codacy

Se ha mejorado significativamente el frontend, lo que se ha reflejado en una subida de nota en Codacy.

8. Aumento general en la calidad del código

También se ha notado una mejora general en la calidad del código entregado.

9. Creación de un anuncio claro y atractivo del producto

Se ha grabado un anuncio efectivo que muestra de forma clara y atractiva la funcionalidad del producto.

10. Trabajo en equipo y compromiso continuo

Los miembros demuestran una implicación alta, colaborando de forma activa y apoyándose mutuamente, incluso ante situaciones de desgaste.

11. Reparto y actualización constante de tareas

La distribución de tareas se ajusta en función de la carga de cada uno y se actualiza a lo largo del sprint, facilitando la organización.

12. Cobertura de pruebas en backend y optimización del testing

Se ha avanzado notablemente en la cobertura de test en el backend, ayudando a detectar y corregir errores de forma eficiente.

13. Gestión oportuna de incidencias y corrección de errores

Tanto en el testing como en el despliegue, la detección y solución rápida de errores ha contribuido a mantener la calidad del proyecto.

14. Retroalimentación positiva y mejora continua

Se ha incorporado de manera efectiva el feedback de usuarios piloto, permitiendo ajustes rápidos y mejoras en la UX.

Puntos del equipo que se pueden mejorar

1. Optimizar la dinámica de las reuniones de coordinación

Mejorar la dinámica de las reuniones de coordinadores para poder tratar todos los temas dentro del tiempo previsto, asegurando mayor eficacia en la toma de decisiones.

2. Corregir el fallo en el envío de encuestas internas

Se debe garantizar el correcto envío de las encuestas internas, ya que los fallos en este proceso afectan negativamente la calidad y utilidad de las respuestas.

3. Recuperar la productividad del equipo

Identificar y abordar los factores que han contribuido a la disminución de la productividad, como la menor participación en programación y el aumento del estrés.

4. Avanzar con la implementación de tests en frontend

Es necesario avanzar en la creación y ejecución de pruebas para el frontend, asegurando la estabilidad y calidad del desarrollo.

5. Mantener actualizadas las tareas para un mejor seguimiento

Asegurar que el estado de las tareas se actualice de forma regular, lo que facilitará un seguimiento claro y preciso del avance del proyecto.

6. Cierre oportuno de pull requests

Fomentar el cierre de las pull requests en tiempos razonables, evitando que queden abiertas demasiado tiempo y afectando el flujo de trabajo.

7. Optimizar la gestión de la información en los canales de comunicación

Reducir el exceso de información y filtrar contenidos relevantes para evitar saturaciones y la pérdida de datos importantes.

8. Realizar anuncios más claros y relevantes

Aumentar la frecuencia y claridad de los anuncios para mantener al equipo informado de forma oportuna y precisa.

9. Evitar prisas en la implementación previa al despliegue

Planificar y ejecutar nuevas funcionalidades con el tiempo necesario para realizar un despliegue controlado, evitando así errores y retrasos.

10. Consultar al equipo antes de tomar decisiones no establecidas

Fomentar la participación y alineación consultando a todos los miembros antes de realizar cambios o acciones no planificadas.

11. Coordinar la refactorización del código antes del despliegue

No llevar a cabo refactorizaciones sin informar previamente a la persona responsable, permitiendo su revisión y asegurando el correcto funcionamiento del sistema.

12. Evitar errores al mover tareas entre sprints en GitHub

Revisar y mejorar el proceso de traslado de tareas entre sprints, de forma que se eviten errores en las gráficas o en la organización de las mismas.

13. Mejorar la comunicación entre frontend y backend

Establecer una coordinación más estrecha entre ambos equipos; por ejemplo, completando las tareas de backend con antelación para no bloquear el desarrollo en frontend.

14. Incrementar la cobertura de tests en el backend

Abordar la insuficiencia actual de pruebas automatizadas en el backend, lo que ayudará a detectar errores de forma temprana y a mejorar la calidad general del código.

15. Terminar de eliminar errores que afectan los despliegues

Identificar y resolver de forma definitiva aquellos errores que generen problemas durante los despliegues.

16. Priorizar tareas de mayor impacto

Asignar más tiempo y recursos a las tareas críticas para el éxito del proyecto, asegurando su correcta ejecución.

17. Verificar exhaustivamente las funcionalidades antes de finalizar implementaciones

Comprobar de forma rigurosa todas las funcionalidades, especialmente en el frontend, antes de dar por finalizado el desarrollo de una tarea.

18. Uso responsable de la inteligencia artificial

Revisar y moderar el uso intensivo de la IA para complementar el trabajo humano sin generar dependencia excesiva.

19. Aumentar la presencia en redes sociales

Incrementar la frecuencia de publicación de posts en redes sociales para mejorar la comunicación externa y la visibilidad del proyecto.

Puntos débiles del equipo

1. Deficiencias en testing y calidad del frontend

Se han evidenciado problemas en la realización y sincronización de pruebas del frontend (incluido el manejo del token en distintos dispositivos), lo que ha derivado en funcionalidades vulnerables y lanzamientos con errores o incompletos.

2. Gestión inadecuada de refactorizaciones y merges

La ausencia de validación funcional durante las refactorizaciones y el merge de pull requests en etapas finales han provocado duplicidad de código, acumulación de issues y dificultades en la integración de cambios.

3. Planificación y seguimiento deficiente de tareas

La falta de control y priorización en el avance de las tareas, sumada a una asignación desigual de la carga laboral, afecta la productividad general del equipo y complica la detección oportuna de problemas.

4. Comunicación ineficiente en cambios críticos

No se han transmitido de forma oportuna y adecuada las modificaciones importantes (por ejemplo, actualizaciones del archivo .env), lo que ha afectado el funcionamiento y la coordinación en el proyecto.

5. Errores críticos en despliegues y sobrecarga en el equipo

Los despliegues de última hora han presentado fallos graves que generan inestabilidad en la aplicación, mientras que la sobrecarga de trabajo en algunos miembros repercute negativamente en el rendimiento global del equipo.

Aprendizaje con respecto a nuestros puntos fuertes

Durante esta semana, el equipo ha demostrado un sólido rendimiento general, resolviendo de forma ágil los problemas que han ido surgiendo. Esta capacidad de respuesta ha sido clave para mantener el ritmo del desarrollo sin interrupciones.

Se ha observado una mejora notable en la estimación de tareas, lo que ha permitido una planificación más precisa y realista. Esto, sumado a la excelente colaboración entre los miembros del equipo, ha contribuido a una dinámica de trabajo muy positiva, con una actitud constante de ayuda mutua.

Ante la baja de un compañero, el equipo supo adaptarse rápidamente, asumiendo su carga de trabajo sin que se viera afectado el progreso general. Asimismo, los bugs críticos han sido resueltos con rapidez, minimizando su impacto en el desarrollo del producto.

Otro aprendizaje relevante ha sido la actitud constructiva del equipo frente a las críticas, utilizándose como una oportunidad para mejorar. Esto ha ido acompañado de una mejora significativa en el frontend, que se ha reflejado directamente en una mejor puntuación en Codacy.

En general, la calidad del código entregado ha mejorado respecto a semanas anteriores, lo que indica un avance técnico constante. Además, se ha realizado un gran trabajo con grabación de un anuncio que muestra de forma clara y atractiva la funcionalidad del producto, reforzando la imagen del equipo y del proyecto.

Posibles mejoras del equipo

Para mejorar la estabilidad del equipo y optimizar el desarrollo, es necesario establecer una regla clara que prohíba refactorizar el código justo antes del despliegue sin notificar previamente a quien lo desarrolló. Esto permitirá revisar los cambios y evitar errores imprevistos que afecten al sistema.

También se debe mejorar la organización del proyecto en herramientas como GitHub, asegurando que la gestión de las tareas sprints se haga correctamente para evitar errores como la rotura de gráficas por un mal traslado de tareas.

Es prioritario reforzar la comunicación entre frontend y backend. Las tareas del backend deben completarse con antelación para evitar bloqueos en el desarrollo del frontend. Además, se debe avanzar en la incorporación de tests tanto en backend como en frontend, para mejorar la cobertura y la detección de errores.

En cuanto a la dinámica interna, se debe mejorar la eficiencia de las reuniones de coordinación, asegurando que se aborden todos los temas dentro del tiempo previsto. También es clave actualizar permanentemente el estado de las tareas y cerrar las pull requests en tiempos razonables para mantener un flujo de trabajo ordenado.

Se debe reducir el exceso de mensajes en los canales de comunicación y aumentar la frecuencia y claridad de los anuncios importantes, evitando que se pierda información relevante.

Para garantizar la calidad del desarrollo, es importante no apurar la implementación de nuevas funcionalidades sin tener en cuenta el tiempo necesario para el despliegue. Finalmente, se debe fomentar la consulta y alineación del equipo antes de realizar cualquier cambio no establecido previamente, con el fin de evitar malentendidos y asegurar una ejecución más coordinada.

"Sprint 3 Retrospective". Se ha utilizado la IA para mejorar la redacción de algunas secciones del documento, así como para corregir errores ortográficos que hubieran pasado desapercibidos. Toda la información generada por IA ha sido revisada por el equipo Caronte.