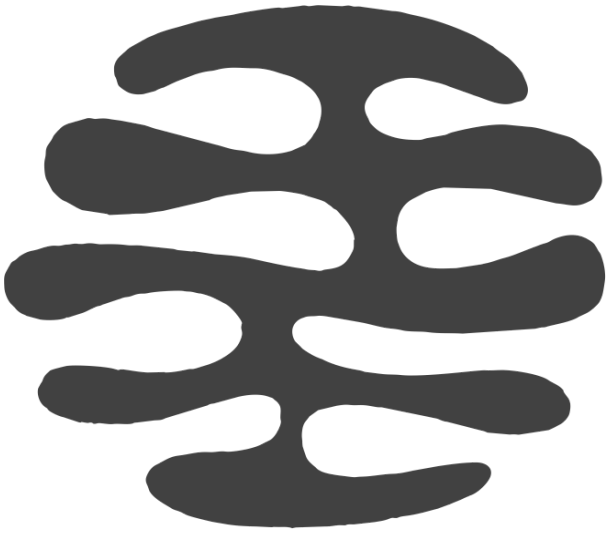

CARONTE

Sprint 2 Retrospective

Sprint 2

Grupo 9

Realizado por:

Chicho Castellano, Álvaro

Revisado por:

Galván Cancio, Daniel

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. Trabajo de alta calidad y productividad

Buena calidad y cantidad de trabajo realizado.

2. Ambiente positivo y colaborativo

Excelente ambiente de trabajo y actitud positiva de todos los miembros del equipo.

3. Apoyo entre compañeros

El equipo brinda apoyo a los compañeros cuando lo necesitan, fomentando la colaboración.

4. Mejor cumplimiento de deadlines

Mejora en el cumplimiento de deadlines, aunque aún hay margen de mejora.

5. Impacto positivo del commitment agreement

Cambio significativo en el commitment agreement, añadiendo penalización en cas de no responder encuestas, lo que ha permitido una mejora notable en la recopilación de feedback y críticas internas.

6. Progreso en la calidad de las presentaciones

Progreso en la calidad de las presentaciones, logrando mayor claridad y efectividad en la comunicación.

7. Automatización y control de calidad del código

Uso eficiente de workflows para la automatización y de Codacy para garantizar la calidad del código.

8. Aplicación del feedback de los usuarios piloto

Implementación efectiva del feedback proporcionado por los usuarios piloto.

9. Evaluación objetiva del rendimiento

Desarrollo de una métrica objetiva para evaluar el rendimiento de cada miembro del equipo.

10. Alta disponibilidad y compromiso del equipo

Alta disponibilidad y compromiso por parte del equipo durante toda la semana.

11. Revisiones de código efectivas

Code reviews constructivos y resolución eficiente de bugs críticos en tiempos adecuados.

12. Distribución equilibrada del tiempo de trabajo

Distribución equilibrada del tiempo de trabajo, con una dedicación adecuada de horas por parte de todos los compañeros.

13. Aumento en la frecuencia de reuniones semanales

Aumento en la frecuencia de reuniones semanales.

14. Mayor calidad y concisión en las reuniones

Las reuniones semanales han mejorado en términos de claridad y precisión, enfocándose en los puntos clave para optimizar el tiempo.

15. Actitud positiva ante la retroalimentación.

Todos los miembros del equipo han demostrado una buena comprensión cuando se han detectado problemas o se les ha señalado algún aspecto a mejorar, solucionándolo rápidamente sin tomárselo de manera negativa.

16. Trabajo constante por parte del equipo

Se ha mantenido un esfuerzo continuo y comprometido por parte de todos los integrantes.

17.Mejora en el conocimiento y eficiencia del equipo

Se ha observado un mayor nivel de experiencia, permitiendo realizar más trabajo en menos tiempo y con mejores resultados.

Puntos del equipo que se pueden mejorar

1. Reducción de mensajes en los canales de comunicación

Reducir la cantidad de mensajes en los canales de comunicación para evitar saturación y mejorar la claridad en la información.

2. Minimizar la introducción de bad smells en el código

Minimizar en la medida de lo posible la introducción de bad smells en el código, asegurando una mayor calidad desde la base.

3. Mejor priorización de tareas y gestión del tiempo

Mejorar la priorización de tareas, enfocándose en lo realmente importante y optimizando los tiempos de desarrollo.

4. Refinar la estimación y reasignación de tareas

Refinar la estimación de tareas y, en caso de desviaciones, reasignarlas de manera más eficiente, ajustando la cantidad de personas involucradas según sea necesario.

5. Fomentar la comunicación y cohesión del equipo

Fomentar una mayor comunicación y cohesión dentro del equipo, evitando la formación de subgrupos independientes. Se debe abordar el proyecto como un todo y no como la suma de varias partes aisladas.

6. Mejorar la lectura y uso del canal de avisos

Mejorar la lectura y uso del canal de avisos, asegurando que se consulte con la frecuencia necesaria y que su contenido sea relevante y bien gestionado.

7. Anticipar la carga de trabajo de cada miembro

Es importante prever la disponibilidad de cada integrante para la semana siguiente, de modo que si alguien tendrá menos tiempo, pueda adelantar trabajo el fin de semana y evitar dejarlo para el último momento.

8. Precisión en el registro de horas

Algunas tareas tienen un conteo de horas irreal en Clockify. Se debe ser más preciso y honesto al registrar el tiempo dedicado a cada tarea para reflejar mejor el esfuerzo real.

9. Falta de publicaciones en redes sociales

Es necesario comenzar a activar las redes para dar a conocer el producto y aumentar su visibilidad.

10. Revisión de pull requests dentro del mismo departamento

Las PR deben ser analizadas y testeadas por miembros del mismo equipo, asegurando un análisis completo y validando todos los casos posibles antes de su aprobación.

11. Aumento de la frecuencia de reuniones.

Se ha mejorado el número de reuniones semanales, pero sería recomendable realizarlas con mayor frecuencia para una mejor coordinación.

12. Cuadrante de exámenes y obligaciones

Se podría establecer un calendario con las fechas de exámenes y otras obligaciones puntuales de los miembros, permitiendo una mejor distribución del trabajo cuando haya sobrecarga personal.

13. Aumentar la frecuencia de publicaciones en redes sociales

Aunque se ha avanzado en la activación de redes sociales, es necesario aumentar la frecuencia de publicación para mejorar la visibilidad del proyecto.

14. Implementación de tests en el backend y mejora de la calidad del código

Iniciar la implementación de tests en el backend y enfocarse en la calidad del código, evitando soluciones rápidas sin una validación adecuada.

15. Mayor insistencia a los usuarios piloto para obtener feedback

Insistir más a los usuarios piloto para que respondan y proporcionen feedback, asegurando una mejor retroalimentación para la mejora del producto.

16. Mejora en la participación en el feedback

Se ha incrementado la cantidad de respuestas en los formularios de feedback, pero todavía es necesario insistir en algunos casos para lograr una participación completa.

17. Implementación de test unitarios en el backend

La revisión del código no debería basarse solo en prueba y error; se deberían añadir test unitarios para validar cada funcionalidad de manera más eficiente y estructurada

Puntos del equipo que se deben cambiar

1. Mejorar las pruebas antes de fusionar código en develop

Probar mejor el código antes de fusionarlo en *develop*, ya que los errores no detectados pueden generar problemas en el desarrollo y afectar la estabilidad del proyecto.

2. Iniciar antes el desarrollo en backend para evitar bloqueos en frontend

Comenzar antes el desarrollo en backend para garantizar que esté listo a tiempo y permitir que el frontend avance sin bloqueos.

3. Establecer deadlines más tempranas para backend que para frontend

Establecer deadlines más tempranas para el backend en comparación con el frontend, ya que este último depende del backend para poder desarrollar correctamente sus funcionalidades.

4. Controlar el uso de IA con revisión adecuada

Evitar el uso excesivo de la IA sin una revisión adecuada por parte del miembro responsable, especialmente en ciertas secciones de código y documentación.

5. Evitar comentarios o actitudes que afecten la moral del equipo

Prestar atención a ciertos comentarios o actitudes que pueden afectar negativamente la moral del equipo y el ambiente de trabajo.

6. Optimizar la planificación de deadlines con fechas personalizadas

Mejorar la planificación de deadlines, asignando fechas personalizadas según la complejidad y necesidad de cada tarea.

7. Falta de coordinación en tareas compartidas

Cuando una tarea involucre a varias personas, es esencial que haya comunicación constante, manteniendo a todos informados sobre el progreso y evitando malentendidos o bloqueos.

8. Retrasos en el cumplimiento de deadlines

Se han producido incumplimientos en algunos plazos de entrega, lo que afecta la planificación y el progreso del proyecto.

Aprendizaje con respecto a nuestros puntos fuertes

El equipo ha mantenido una alta calidad y cantidad de trabajo, reflejando un compromiso constante con los objetivos del proyecto. El ambiente de trabajo sigue siendo excelente, con una actitud positiva de todos los miembros y una fuerte disposición a colaborar y apoyarse

mutuamente con una alta disponibilidad y compromiso a lo largo de la semana y una buena distribución del tiempo de trabajo

Se ha observado una mejora en el cumplimiento de deadlines, aunque aún hay margen para optimizar la gestión del tiempo y evitar retrasos. En este sentido, el commitment agreement ha tenido un impacto positivo, permitiendo una recopilación más efectiva de feedback y críticas internas, lo que favorece la mejora continua del equipo.

En términos de comunicación y presentación, se ha logrado un avance significativo en la calidad de las exposiciones, mejorando la claridad y efectividad en la transmisión de información. También se ha fortalecido el uso de workflows y herramientas como Codacy, lo que ha permitido una mejor automatización y un control de calidad más eficiente en el código. Además las code reviews han sido constructivas y la resolución de bugs críticos se ha realizado en tiempos adecuados, garantizando un desarrollo más estable y eficiente.

La implementación del feedback de los usuarios piloto ha sido efectiva, contribuyendo a la mejora del producto en función de sus necesidades. Además, la introducción de una métrica objetiva para evaluar el rendimiento de cada miembro ha ayudado a mantener una visión más clara del desempeño individual y colectivo.

El equipo ha logrado mejorar la organización con un aumento en la frecuencia de reuniones semanales. Además, se ha optimizado la calidad de estas reuniones, haciéndolas más concisas y enfocadas en los puntos clave, lo que ha favorecido un uso más eficiente del tiempo.

Otro aspecto positivo ha sido la actitud proactiva del equipo ante la retroalimentación. Se ha demostrado una gran capacidad para aceptar y solucionar rápidamente los problemas detectados sin generar conflictos, lo que contribuye a un ambiente de trabajo más productivo y colaborativo.

El esfuerzo y compromiso de todos los integrantes se ha mantenido constante, reflejándose en un progreso continuo y en una mayor eficiencia en la ejecución de las tareas. Asimismo, se ha notado una mejora en el conocimiento y desempeño del equipo, permitiendo realizar más trabajo en menos tiempo sin comprometer la calidad.

Estos aprendizajes refuerzan la importancia de la comunicación, la actitud constructiva ante el feedback y la constancia en el trabajo para el correcto desarrollo del sprint

Posibles mejoras del equipo

Para mejorar la comunicación dentro del equipo, se debe reducir la cantidad de mensajes en los canales, evitando la saturación y asegurando que la información sea clara y relevante. También es importante mejorar la lectura y el uso del canal de avisos, garantizando que todos los miembros lo consulten con la frecuencia necesaria.

Desde el punto de vista técnico, se debe minimizar la introducción de bad smells en el código, asegurando una mayor calidad desde el inicio. Para ello, se recomienda iniciar la implementación de tests en el backend, lo que permitirá validar las funcionalidades de manera estructurada y evitar errores. Asimismo, se debe mejorar la revisión del código antes de fusionarlo en develop, reduciendo los problemas en el desarrollo y asegurando la estabilidad del proyecto.

Para optimizar el desarrollo y la planificación del trabajo, es fundamental mejorar la priorización de tareas, enfocándose en lo realmente importante. Además, se deben asignar deadlines más tempranas para el backend en comparación con el frontend, ya que este último depende del primero para avanzar. En caso de desviaciones en la estimación de tareas, se deben reasignar los recursos de manera eficiente, ajustando la cantidad de personas involucradas según la necesidad de cada tarea.

También es esencial fomentar una mayor cohesión dentro del equipo, evitando la creación de subgrupos y viendo el proyecto como un todo. Es importante prestar atención a ciertos comentarios o actitudes que puedan afectar la moral del equipo y generar un ambiente menos productivo.

Por otro lado, en el ámbito externo, se debe aumentar la frecuencia de publicaciones en redes sociales para mejorar la visibilidad del proyecto. Además, se debe insistir más a los usuarios piloto para que proporcionen feedback, asegurando una retroalimentación más completa y útil para las mejoras del producto.

Finalmente, se recomienda evitar el uso excesivo de la IA sin una revisión adecuada del miembro responsable, especialmente en secciones críticas del código y la documentación. Mantener un control sobre estas implementaciones garantizará la calidad y coherencia del trabajo realizado.

Para optimizar la planificación del trabajo, es fundamental anticipar la carga de cada miembro del equipo, permitiendo que aquellos con menor disponibilidad durante la semana puedan

adelantar tareas el fin de semana previo. Asimismo, establecer un cuadrante con los exámenes y otras obligaciones personales ayudará a equilibrar mejor la distribución del trabajo.

Es importante mejorar la presencia en redes sociales, iniciando publicaciones constantes para aumentar la visibilidad del proyecto. Además, se debe registrar con mayor precisión las horas en Clockify para reflejar el esfuerzo real.

En términos de coordinación, se debería incrementar la frecuencia de reuniones, ya que esto facilitará la sincronización del equipo. Para mejorar la calidad del código, las pull requests deben ser revisadas por miembros del mismo equipo, asegurando un análisis más detallado y completo. Además, se debe seguir fomentando la participación activa en los formularios de feedback para recopilar información clave sobre el desempeño del equipo.

Se debe comenzar con la implementación de test unitarios en el backend, reduciendo la dependencia de la prueba y error, asegurando mayor estabilidad en el desarrollo. También se debe reforzar la comunicación entre frontend y backend para evitar retrasos, asegurando que los endpoints estén listos antes de que el frontend los requiera. De igual manera, cualquier modificación que pueda romper la funcionalidad de la aplicación, debe notificarse con suficiente antelación para prevenir y corregir los problemas.

Finalmente, es esencial fortalecer la comunicación en tareas compartidas, garantizando que todos los involucrados se mantengan informados y coordinados. También se debe trabajar en una mejor gestión de los plazos de entrega, reduciendo retrasos y asegurando una ejecución más eficiente del proyecto.

"Sprint 2 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.