**ATAM: Sistema de Gestión de Tareas**

**Contexto del Proyecto:** Crear una plataforma en línea que permita a los usuarios organizar, administrar y realizar seguimiento de sus tareas y proyectos de manera eficiente. La aplicación deberá contar con funcionalidades como crear tareas, asignarlas a miembros del equipo, establecer fechas de vencimiento, prioridades y categorías, así como generar informes de avance y completitud de las tareas.

1. **Definición de Objetivos de Calidad:** Identificamos los siguientes objetivos de calidad:

* **Seguridad:** Las tareas solo deberán ser mostradas y protegidas para cada usuario, de manera que el usuario que ingrese en la plataforma podrá ver exclusivamente sus tareas creadas.
* **Rendimiento:** La visualización y actualización del estado de las tareas deberá ser rápida incluso en situaciones de carga alta.
* **Usabilidad:** Las tareas serán gestionadas fácilmente desde la plataforma.

**2. Identificación de Escenarios:** Definimos escenarios específicos para evaluar cada objetivo:

* **Escenario de Seguridad**: "Prevención de acceso a las tareas para usuarios no autorizados".
* **Escenario de Rendimiento:** "Visualización rápida de tareas durante horas pico".
* **Usabilidad:** “Visualización y gestión de tareas intuitivamente según el estado correspondiente”.

**3. Creación del Modelo de Arquitectura Inicial:** Desarrollamos un modelo arquitectónico inicial que incluye componentes para visualización, creación, actualización, eliminado de tareas.

**Iteración 1:**

Hasta el momento, solo hemos tenido una iteración, pero creemos que esta única interacción refleja de manera bastante completa las actividades que un usuario puede realizar en el sistema

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

**Descripción de los componentes:**

* **Frontend UI Container**: Este componente proporciona la interfaz de usuario para la aplicación. Incluye los paneles, botones y otros controles que los usuarios interactúan para administrar sus tareas.
* **Backend API Container**: Este componente proporciona la lógica de la aplicación. Incluye el código que maneja tareas como crear, actualizar, eliminar y recuperar tareas.
* **Memory Database**: Este componente almacena los datos en memoria de la aplicación, como tareas y usuarios.

**Interacciones:**

* + - **El Frontend UI Container** llama a los servicios en el Backend API Container para realizar operaciones como crear, actualizar, eliminar y recuperar tareas.
    - **El Backend API Container** lee y escribe datos de la base de datos para realizar las operaciones solicitadas por el Frontend UI Container.

**4. Análisis Preliminar:**

* Evaluamos cómo la arquitectura aborda cada escenario. Por ejemplo, para el escenario de seguridad, revisamos cómo se manejan los controles de acceso para el usuario y que a su vez este pueda ver solo las tareas que está gestionando. También por ejemplo el escenario de rendimiento donde analizamos que el usuario pueda ver sus tareas sin ninguna inconsistencia, y a su vez las observe independientemente de en qué estado las esté poniendo.

**5. Identificación de Trade-offs:** Se hizo un diseño de la aplicación fácil de manejar y agradable con la principal característica de que pueda crear y ajustar sus tareas arrastrándolas a los diferentes estados que desee el usuario.

Descubrimos que, para mejorar el rendimiento del sistema, este va de la mano con la usabilidad ya que, si el sistema puede gestionar eficientemente el estado de las tareas, el rendimiento se verá mejor reflejado. Se identifica un trade-off entre rendimiento y usabilidad.

**6. Iteración y Refinamiento:** Realizamos ajustes en el diseño para abordar los trade-offs identificados. En un principio se tenía pensado cambiar el estado de las tareas con un botón, pero esto es poco agradable y funcional para el usuario. Por ende, se implementó la técnica del drag and drop que usan usualmente en los diseños UX para poner la aplicación más intuitiva y mejorar la usabilidad de la aplicación.

**7. Resultados:**

El análisis detallado de la arquitectura del Sistema de Gestión de Tareas nos ha proporcionado información valiosa sobre cómo nuestras decisiones arquitectónicas impactan en los objetivos de calidad previamente definidos.

Se logró hacer identificar que la aplicación cumpla con los objetivos de calidad, así mismo con los trade-offs y el respectivo refinamiento para que el sistema de gestión de tareas sea lo más intuitivo y fácil de manejar para el usuario final.

**QAW: Sistema de Gestión de Tareas**

**Contexto del Proyecto**: Continuando con el mismo sistema de gestión de tareas, ahora aplicaremos Quality Attribute Workshop (QAW) para identificar y priorizar atributos de calidad en colaboración con algunas personas conocidas dispuestas a opinar sobre el sistema.

**1. Identificación de Atributos de Calidad:** En una reunión conjunto, reunimos a partes interesadas, como conocidos, familiares y desarrolladores del sistema, para identificar atributos de calidad. Algunos de los atributos identificados son:

* **Funcionalidad**
* **Usabilidad**
* **Fiabilidad**
* **Rendimiento**
* **Seguridad**

**2. Definición de Escenarios de Calidad:** Con la participación de las partes interesadas, definimos escenarios detallados para cada atributo:

* **Funcionalidad:** La aplicación debe permitir a los usuarios crear tareas con una descripción en el estado correspondiente, con la opción de mover dicha tarea a cualquier otro estado y eliminarlas.
* **Usabilidad**: La aplicación debe proporcionar una interfaz de usuario clara y concisa.
* **Fiabilidad**: La aplicación no debe fallar al crear, actualizar, eliminar o recuperar tareas.
* **Rendimiento**: La aplicación debe poder crear, actualizar, eliminar o recuperar tareas instantáneamente sin ningún retraso.
* **Seguridad**: La aplicación debe mostrar únicamente las tareas que le corresponden al usuario que ingresó al sistema.

**3. Priorización de Atributos de Calidad:**

Utilizando técnicas de votación y discusiones colaborativas, las partes interesadas priorizan los atributos de calidad. Descubrimos que la funcionalidad, la usabilidad, la fiabilidad y el rendimiento son críticos para el éxito del producto. La seguridad es también importante, pero tiene una menor prioridad.

**4. Definición de Métricas:** Para cada atributo de calidad, definimos métricas específicas que nos ayudarán a evaluar el rendimiento del sistema en esos aspectos. Por ejemplo:

* Para la **funcionalidad**, podríamos medir el tiempo en que se crean las tareas y las respectivas asignaciones a sus estados.
* Para la **usabilidad**, podríamos medir el tiempo promedio para completar tareas comunes.
* Para la **fiabilidad**, podríamos tomar en cuenta las razones por las cuales el aplicativo podría fallar al ejecutar cualquiera de las funciones en la gestión de tareas.
* Para el **rendimiento**, podemos evaluar el tiempo de respuesta que se toma la aplicación para validar las funciones de visualización, creación, modificación y eliminación de la tarea.
* Para la **seguridad**, podremos evaluar las formas en que el usuario se puede ver vulnerado al tener sus tareas en manos ajenas.

**5. Desarrollo de Escenarios Detallados:** En sesiones adicionales, trabajamos en desarrollar escenarios detallados para comprender mejor cómo se manifestarían los atributos de calidad en situaciones específicas.

**Escenario de Funcionalidad: Creación de Tareas y Cambio de Estado**

* **Descripción:** Un usuario registrado inicia sesión en el sistema y desea crear una nueva tarea. El usuario proporciona una descripción detallada de la tarea y elige un estado específico. Luego, el usuario intenta cambiar el estado de la tarea.
* **Resultado Esperado:** La tarea se crea con éxito, y el cambio de estado se realiza de manera instantánea. No se producen errores ni retrasos en estas acciones.

**Escenario de Usabilidad: Interfaz de Usuario Intuitiva**

* **Descripción:** Un usuario sin experiencia previa con el sistema intenta realizar una serie de tareas comunes, como crear una tarea, marcar una tarea como completada.
* **Resultado Esperado:** El usuario puede completar estas acciones de manera intuitiva y sin confusiones. La interfaz de usuario es clara y fácil de navegar, lo que permite a los usuarios realizar estas tareas de manera eficiente.

**Escenario de Fiabilidad: Gestión de Tareas sin Errores**

* **Descripción:** El usuario utiliza el sistema concurridamente para crear, actualizar, eliminar y recuperar tareas de manera continua durante un período de carga elevada.
* **Resultado Esperado:** El sistema demuestra una alta confiabilidad al ejecutar estas operaciones, sin errores ni caídas del sistema, incluso bajo carga intensiva.

**Escenario de Rendimiento: Carga de la Página de Inicio**

* **Descripción:** Un usuario abre la aplicación web del sistema de gestión de tareas e inicia sesión. El usuario espera que la página de inicio del sistema se cargue de manera rápida y sin demoras notables.
* **Resultado Esperado:** La página de inicio del sistema debe cargar de manera rápida y eficiente después de que el usuario inicie sesión, proporcionando acceso inmediato a las tareas y la funcionalidad principal. No debe haber retrasos significativos en la carga de la página.

**Escenario de Seguridad: Restricción de Acceso a Datos Privados**

* **Descripción:** Un usuario está autenticado en el sistema y desea acceder a los detalles de una tarea que no le pertenece. Intenta acceder directamente a la URL de la tarea de otro usuario.
* **Resultado Esperado:** El sistema debe verificar las credenciales del usuario y asegurarse de que solo tenga acceso a los datos y tareas que le corresponden. Debe denegar el acceso a la tarea de otro usuario y mostrar un mensaje de error o redirigir al usuario a la página de inicio.

**Resultados:**

Hemos realizado un proceso detallado de Quality Attribute Workshop (QAW) para el Sistema de Gestión de Tareas, trabajando en conjunto con diversas partes interesadas, como conocidos, familiares y desarrolladores del sistema. Este proceso nos ha permitido avanzar significativamente en la comprensión y mejora de la calidad del sistema, logrando los siguientes resultados clave

1. **Atributos de Calidad Identificados y Priorizados:** Se ha identificado y priorizado una lista de atributos de calidad críticos para el éxito del sistema. Estos atributos incluyen Funcionalidad, Usabilidad, Fiabilidad, Rendimiento y Seguridad. Esta priorización nos ha proporcionado una visión clara de las áreas en las que debemos enfocar nuestros esfuerzos de desarrollo y evaluación.
2. **Escenarios Detallados Definidos:** Para cada uno de estos atributos de calidad, hemos definido escenarios detallados que describen situaciones específicas de uso del sistema. Estos escenarios son esenciales para comprender cómo los atributos de calidad se manifiestan en la práctica y servirán como base sólida para las pruebas y evaluaciones posteriores.
3. **Métricas Establecidas para Evaluación:** Para cada atributo de calidad, hemos definido métricas específicas que nos permitirán evaluar y medir el rendimiento del sistema en esas dimensiones. Estas métricas nos proporcionarán datos cuantitativos para determinar si el sistema cumple con los estándares de calidad establecidos.
4. **Escenarios Detallados Desarrollados:** Se han desarrollado escenarios detallados adicionales para comprender mejor cómo se manifiestan los atributos de calidad en situaciones específicas de uso. Estos escenarios proporcionan una descripción concreta de las interacciones de los usuarios con el sistema y nos ayudarán a evaluar su rendimiento de manera más precisa.