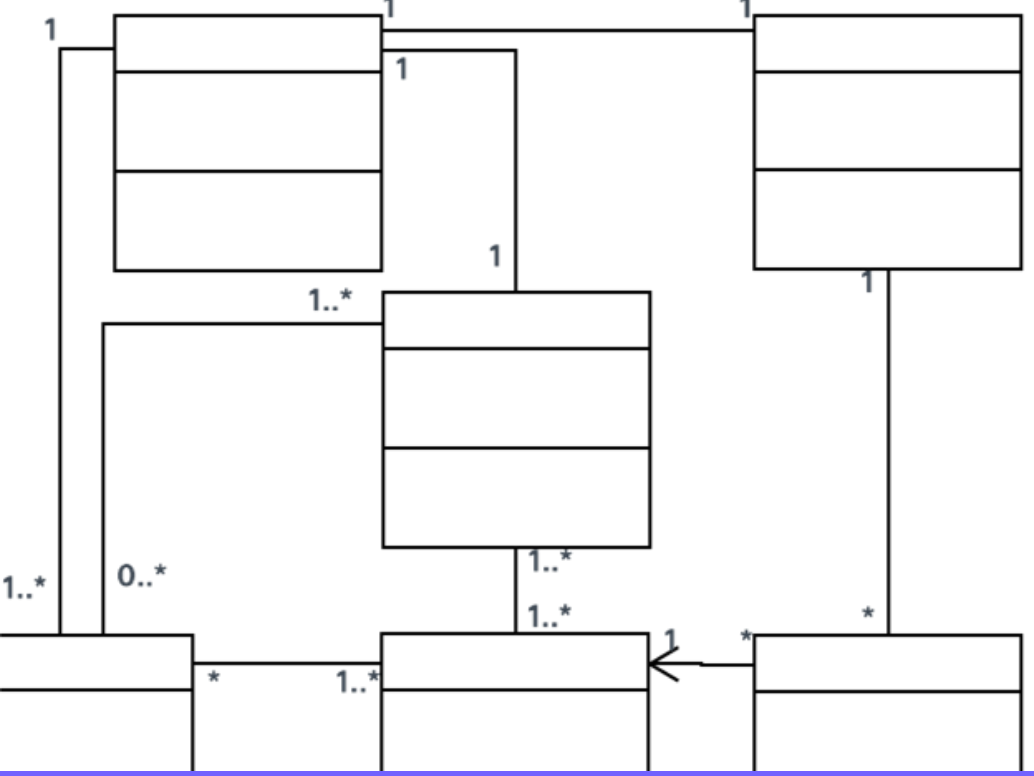
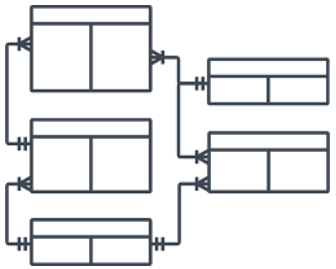


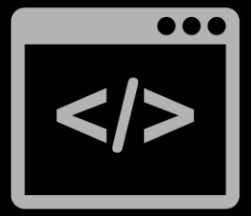


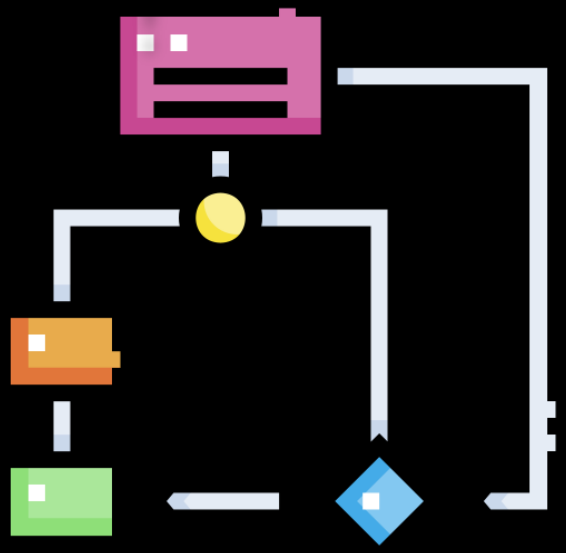
Diseño de Sistemas

Agenda

- Caso 1 – Fuiden
- Caso 2 – SIEEE
- Caso 3 – Fixtures Automágicos
- Caso 4 – Salvando Animales

Fuiden





UNIFIED MODELING LANGUAGE™

Fuiden

Fuiden es una cooperativa que se dedica a la solución integral de logística, es decir que se encarga tanto del almacenamiento de productos de sus clientes como del despacho y entrega de los mismos, a causa las ventas realizadas a través de MELU (Ecommerce donde los clientes de Fuiden ofrecen y venden sus productos al público).
En esta primera etapa se requiere el diseño y desarrollo del módulo de egresos de pedidos, módulo de Productos y Registro de movimientos de Stock.

Situación 1

Considerando que MELU espera que, ante la notificación de un evento, los Sistemas le ***respondan*** con un código de estado ***HTTP 200 en menos de 5 ms***;

¿Cuál cree que sería la mejor forma de obtener los datos del recurso asociado en la notificación para evitar que MELU corte la conexión por no contestar dentro del tiempo esperado?

Situación 1 - Consideraciones

Consideraciones Arquitectónicas:

- Obtener los datos de los recursos asociados de forma ***asincrónica***.

Situación 2

Fuiden nos ha mencionado que solamente utilizarán el Sistema a través del ***navegador de una PC*** (no desde celulares). Además, sabemos que la interfaz debe ser ***fluida***, ***evitando recargar páginas*** mientras sea posible.

- ¿Qué tipo de cliente implementaría (cliente liviano o cliente pesado)? ¿Por qué?
- ¿Cómo manejaría las sesiones de los usuarios?

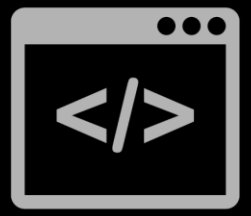
Situación 2 - Consideraciones

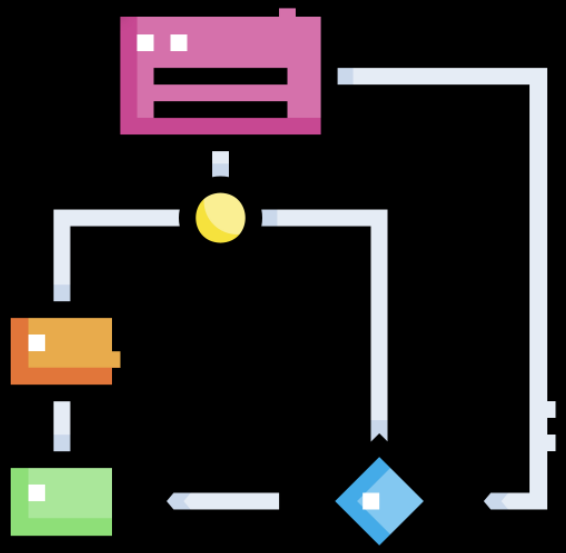
Consideraciones Arquitectónicas:

- Si necesitamos evitar la recarga de páginas entonces podríamos escoger un ***Cliente Pesado*** (Client Side Render)

- Siendo este el caso, las sesiones las manejaríamos por ***Tokens***, siendo nuestro ***Backend Stateless***.

SIEEE





UNIFIED MODELING LANGUAGE™

SIEEE

Las plataformas SIEEE (Si [ocurre] esto, entonces [haz] aquello) o IFTTT (If This, Then That) permiten crear y programar acciones (llamadas recetas) para automatizar diferentes tareas y acciones en Internet. Su nombre deriva de su funcionalidad clave: cuando ocurre un evento específico (esto), la plataforma ejecuta una acción predeterminada en respuesta (aquello). SIEEE simplifica la integración de diferentes servicios y facilita la automatización de tareas diarias.

En su núcleo, IFTTT opera a través de "recetas", que son combinaciones de una condición ("disparador") y una acción ("action"). La condición se establece en una aplicación o dispositivo y actúa como el desencadenante, mientras que la acción se lleva a cabo en otra aplicación o dispositivo como resultado de la condición cumplida. Los usuarios pueden crear y personalizar sus propias recetas o utilizar las creadas por otros miembros de la comunidad IFTTT.

Situación 1

Proponga una arquitectura física para el sistema, mediante un diagrama de despliegue o esquema general:

- Deben figurar los componentes a utilizar, los sistemas externos, con qué protocolos todos se comunican, y cualquier aclaración que considere relevante.
- Tenga en cuenta que ***pueden llegar muchos eventos a la vez*** y que ***no queremos saturar*** nuestros recursos tratándolos todos al mismo tiempo. No apuntamos a manejar información crítica, si se demora “mucho” en realizar una acción o pierde algún paquete, los usuarios están cordialmente advertidos
- Se quiere que el usuario pueda comenzar a usar el sistema rápidamente, con lo cual se deben aceptar ***credenciales de redes sociales*** u otros sitios para registrarse/ingresar al sistema

Situación 1 - Consideraciones

Consideraciones Arquitectónicas:

- Para poder almacenar todos los eventos que llegan de forma simultánea y trabajarlos a medida que “se pueda”, podemos utilizar una ***Cola de Mensajes / de trabajo***. Por ejemplo, RabbitMQ, Apache Kafka, etc.

- Para aceptar credenciales de redes sociales u otros sitios se puede utilizar un ***Single Sign On (SSO)***. Ejemplo: Auth0, Keycloak, etc.

Situación 2

Se observó que muchas de las recetas observan ***exactamente los mismos eventos***, ya sea porque siguen las mismas cuentas en las redes sociales, u observan los mismos cambios en los recursos WEB, con lo cual ***los conectores tienen EXACTAMENTE los mismos datos***.

Teniendo en cuenta esto:

¿Qué se puede implementar para mejorar el rendimiento del sistema? Explique su funcionamiento

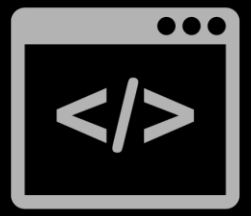
Situación 2 - Consideraciones

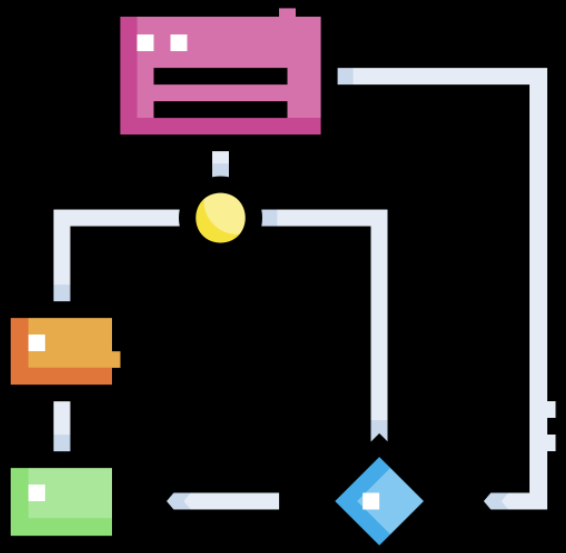
Consideraciones Arquitectónicas:

- Al estar realizando request con exactamente los mismos datos, se puede implementar una ***Caché*** para evitar tener que realizar las mismas solicitudes continuamente.

- Se debe considerar un tiempo adecuado para el ***refresco de los datos*** almacenados en la Caché.

Fixtures Automágicos





Fixtures Automágicos

Se requiere el diseño y el desarrollo de un Sistema que permita realizar el armado automático de fixtures de torneos. En esta oportunidad tendremos que diseñarlo y desarrollarlo para la disciplina de Kayak Polo.

Situación 1

Nos han comentado que se está ***desarrollando, en paralelo***, un Sistema de Resultados y Estadísticas de los torneos. Este Sistema requiere conocer el fixture del torneo que ***nuestro Sistema genera***.

¿Qué tipo de integración escogería? ¿Por qué?

Situación 1 - Consideraciones

Consideraciones Arquitectónicas:

- Puede ser una integración por Web APIs, como API REST; o también una integración por Base de Datos compartida o mediante una Cola de Mensajerías. Para este caso, puede resultar más adecuado una integración por API REST.

- Siguiendo con el caso anterior, nuestro Sistema debería exponer una interfaz REST con el recurso “Fixture”.

Situación 2

El Sistema actual permite la visualización del Fixture a través de una Web. Nos han comentado que la ***carga del motor de base de datos*** (***lecturas***) ***es elevada*** en momentos de alta concurrencia.

¿Qué estrategia aplicaría para reducir la carga en el motor de base de datos?

Situación 2 - Consideraciones

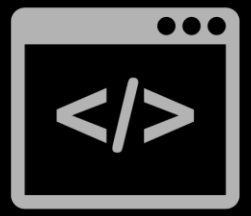
Consideraciones Arquitectónicas:

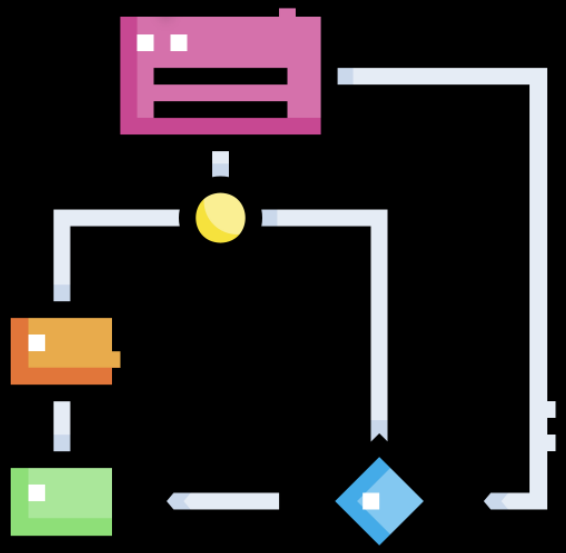
- Se podría implementar una Caché para obtener de forma más rápida los datos más consultados a la Base de Datos, y de esta forma no saturar los motores.

- Otra posibilidad, no excluyente a la anterior, es distribuir la carga entre más instancias de la base de datos. Por ejemplo, se podría tener una instancia que solamente se encargue de las lecturas y otra de las escrituras.

- Si Se propone lo anterior, se debe considerar la implementación de algún mecanismo de réplica de datos.

Salvando Animales





Salvando Animales

Se desea diseñar y desarrollar un Sistema para recolectar, procesar y analizar los datos obtenidos de distintos dispositivos colocados en colonias de diferentes animales en peligro de extinción para su posterior estudio y toma de decisiones.
La plataforma deberá disponer de un espacio en donde los científicos involucrados podrán publicar las conclusiones obtenidas. Actualmente las colonias que se estudiarán serán de aproximadamente entre 1000 y 5000 individuos (animales) dependiendo de cada tipo de especie. En un futuro se seguirán incorporando más individuos y otras especies que no estén en peligro de extinción.

Situación 1

- ¿Cuál sería el mejor ***mecanismo de integración*** entre el Sistema y los Dispositivos/Sensores?.
- Dejar en claro si se prefiere una integración ***sincrónica*** o ***asincrónica***;
- También dejar en claro si el mecanismo que se prefiere es "***pull based***" o "***push based***".

Situación 1 - Consideraciones

Consideraciones Arquitectónicas:

- La integración debería ser desde los sensores hacia el servidor. Podría darse a través del llamado a una API REST, pero también podría ser mediante un Broker de Mensajes como por ejemplo Mosquitto, mediante el protocolo MQTT.

- Si consideramos lo anterior, la integración es ***Push Based***.

- Al ser un mecanismo Push Based, la integración es ***asíncrona***.

Situación 2

Nos han comentado en entrevistas posteriores al relevamiento que algunos científicos además de realizar estudios también son auditores del componente externo a la plataforma y les ***preocupa tener dos usuarios distintos*** para cada tarea.

¿Qué propone para solucionarlo?

Tenga en cuenta que el equipo que está a cargo del desarrollo y mantenimiento del componente externo puede tomar en consideración nuestra propuesta para implementarlo.

Situación 2 - Consideraciones

Consideraciones Arquitectónicas:

- Se podría implementar un SSO.

Situación 3

Además de la plataforma Web, se desea desarrollar una Aplicación Móvil para visualizar en tiempo real los datos de cada uno de los individuos. Considerando ***performance*** y ***mantenibilidad***, indique cuál cree que sería la mejor opción:

- Aplicación nativa
- Aplicación híbrida con cliente liviano
- Aplicación híbrida con cliente pesado

Situación 3 - Consideraciones

Consideraciones Arquitectónicas:

- Deberíamos preguntarnos si buscamos maximizar la performance del lado del Servidor o del Cliente, pues esto cambia bruscamente nuestra decisión.

- En cuanto a mantenibilidad, deberíamos establecer ciertos parámetros para definir entre alguna de las opciones. Un parámetro podría ser "*tiempo aproximado, en horas, que demora implementar un cambio correctivo*".

Gracias