**ACCESO DEV ORG.**

Correo: [ramos16.drak819@agentforce.com](mailto:ramos16.drak819@agentforce.com)

Contraseña: EfZGg94hKjVxkJU

**CÓDIGO FUENTE DE LWC Y TRIGGERS**

<https://github.com/Cskill0/SalesForce_v2>

**DOCUMENTACIÓN TÉCNICA**

**🔹 ¿Qué aprendí del ecosistema Salesforce?**

Durante el desarrollo del proyecto, comprendí la importancia del **modelo declarativo** de Salesforce para construir aplicaciones rápidamente sin necesidad de mucho código. Exploré y apliqué recursos como:

* **Lightning Web Components (LWC)** para construir interfaces modernas y reactivas.
* **Objetos personalizados** (como Seguimiento\_\_c) para modelar datos de negocio.
* **Triggers y Apex** para lógica avanzada que no puede resolverse de forma declarativa.
* **Named Credentials y External Credentials** para realizar integraciones seguras sin exponer secretos.

También adquirí experiencia usando herramientas como **Developer Console**, **registros de depuración**, y el **modo Lightning** para administrar configuraciones y permisos con mayor eficiencia.

**🔹 ¿Cómo investigué e implementé la API?**

El objetivo era obtener información climática desde [OpenWeatherMap](https://openweathermap.org/) e integrarla dentro de registros Seguimiento\_\_c.

Pasos realizados:

1. **Lectura de la documentación oficial** de OpenWeather API para conocer la estructura del endpoint, autenticación por API Key y formato de respuesta JSON.
2. **Implementación de Named Credential + External Credential** para consumir la API de forma segura desde Salesforce (usando callout:OpenWeather).
3. Creación de una clase WeatherServicio con método @future(callout=true) para consumir los datos en segundo plano, actualizar el campo Clima\_Actual\_\_c y emitir un evento Clima\_Actualizado\_\_e vía plataforma de eventos.
4. Integración del llamado en un **trigger** de Seguimiento\_\_c, detectando cuándo cambia el campo Ubicacion\_\_c.

Además, realicé pruebas con registros reales, visualización del clima desde un componente LWC, y ajustes en permisos para asegurar que el callout se ejecutara correctamente.

**🔹 ¿Qué decisiones tomé entre arquitectura declarativa y programática?**

**Declarativa:**

* Uso de **objetos personalizados** (Seguimiento\_\_c) y campos estándar.
* Configuración de **Named Credential** y **Permission Set** sin código.
* Reglas de seguridad (perfil y conjunto de permisos) gestionadas desde UI.

**Programática:**

* Se eligió usar **Apex con @future(callout=true)** porque:
  + El consumo de API requiere lógica de red no disponible en flujos.
  + Se necesitaba procesar y transformar los datos de la API (JSON).
* Uso de **eventos de plataforma (Clima\_Actualizado\_\_e)** para mantener componentes LWC actualizados en tiempo real.
* Validaciones adicionales implementadas vía **Triggers**, como el límite de 5 seguimientos pendientes por contacto.

Esta combinación permitió aprovechar la **flexibilidad del código donde era necesario**, sin comprometer la **seguridad ni la mantenibilidad declarativa** del sistema.