**MONISHA.DUSANAPUDI**

**Phase 7 Report: Integration & External Access**

**Non-Profit Donation & Volunteer Management**

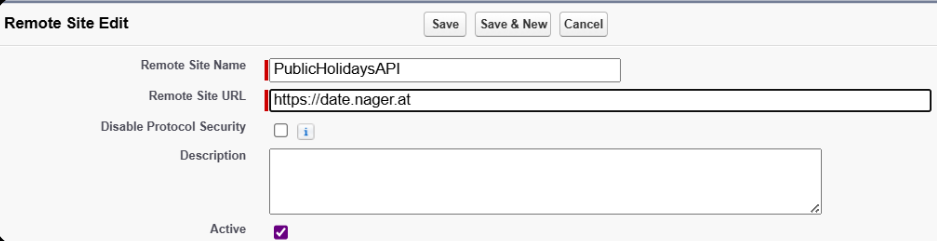
**Objective:** The objective of Phase 7 was to implement a complete, end-to-end integration with an external web service. The goal was to build a practical, real-time feature that would enrich the user experience by pulling valuable data from an external source directly into the Salesforce UI, while adhering to Salesforce's security framework and development best practices.

**1. Web Services (REST/SOAP) & Apex Callouts**

* A real-time, synchronous **REST Callout** was successfully implemented. An Apex class, HolidayFetcherController, was created to handle this integration.
* **Implementation Details:** The Apex code utilizes the built-in HttpRequest and HttpResponse classes to construct and send an HTTP GET request to the external API endpoint. Upon receiving a successful response, the code parses the incoming JSON data and transforms it into a strongly-typed list of Holiday objects for use in the user interface. This demonstrates a mastery of the fundamental Apex callout framework.

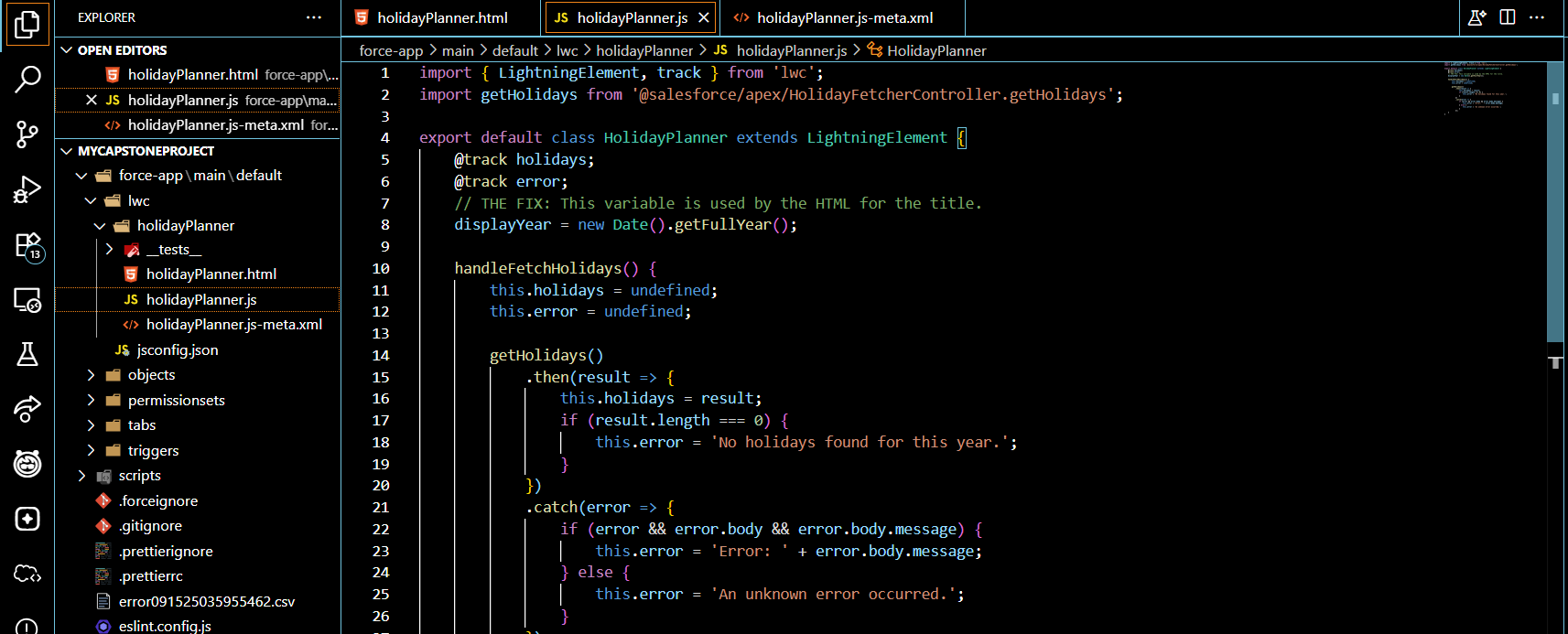
**2. Remote Site Settings**

* To enable the Apex Callout, a **Remote Site Setting** was created. The setting, named PublicHolidaysAPI, was configured to authorize the specific endpoint URL of the external API (<https://date.nager.at>).



**3. User Interface (Lightning Web Component)**

* A custom Lightning Web Component (LWC), holidayPlanner, was built from scratch in Visual Studio Code to serve as the user interface for the integration.



**4. OAuth & Authentication**

* **Scope:** The selected public API for public holidays is free and open, and it **does not require authentication**.
* Consequently, complex authentication protocols like **OAuth** were not necessary for this implementation. The focus was on the core mechanics of making a successful unauthenticated REST callout, which is a very common integration pattern for public data sources.

**5. Named Credentials & External Services**

* **Scope:** These are more advanced, alternative methods for managing integration endpoints and authentication.

**6. Advanced Integration Patterns (Platform Events, CDC, Salesforce Connect)**

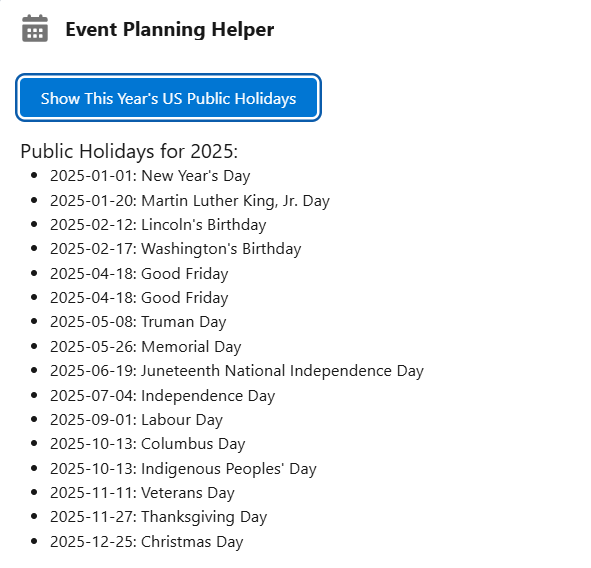
* **Scope:** Our feature, the "Event Planning Helper," required a synchronous, user-initiated **request-response** integration pattern (i.e., the user clicks a button, Salesforce *requests* data, and the API *responds*).

**7. API Limits**

* **Consideration:** The implementation was designed to be highly efficient and conscious of Salesforce governor limits.

**TESTING THIS PHASE**

The development process involved a realistic and complex troubleshooting cycle. Initial API choices proved to have incomplete data, leading to a strategic pivot to a more reliable data source. The code was refined to handle different HTTP success codes (like 200 vs. 204) and to parse JSON data correctly, fixing a bug where dates were not appearing. The final version was successfully tested, proving the end-to-end functionality.



**Conclusion:** Phase 7 is complete. A valuable and project-relevant integration feature has been successfully designed, built, and tested. The implementation correctly uses Apex Callouts, Remote Site Settings, and a modern LWC front-end, demonstrating a comprehensive understanding of the core principles of integrating Salesforce with external REST web services.