Select **one** of the following REpresentational State Transfer (RESTful) topics below. In your writing, be sure to explain the “what,” “how,” and “why” of the selected topic. Provide a simple code or code snippet example to further illustrate your thoughts.

* Uniform Interface
* Client-Server
* Stateless
* Cacheable
* Layered System
* Code on Demand (Optional)

**“What,” “How,” and “Why” Questions**

* **What:** ~~What are you writing about~~? Give the audience a brief overview of the topic by providing them with foundational information (history, background information, etc.).
* **How:** How is the information relevant? Apply personal knowledge (this can be through research or actual practiced knowledge) to build trust with the audience.
* **Why:** Justify your position and/or course of action. The audience needs proof the information you are presenting is creditable and actionable.

Many topics regarding REpresentational State Transfer (RESTful) include Uniform Interface, client-server, stateless, cacheable, layered system, and code on demand. The focus of this discussion will be on the uniform interface. REST is an example of a software architectural style that sets rules when developing web services (GeeksforGeeks, 2018). Systems with request and approval can access and manipulate the web services by following the rules (GeeksforGeeks, 2018). The Uniform Interface follows four guidelines of resource-based, manipulation of resources through representations, self-descriptive messages, and hypermedia as the engine of application state (HATEOAS)" (GeeksforGeeks, 2018). The Uniform Interface "simplifies and decouples client and server interactions" (Bhagwan Sahane, 2024). This helps improve the predictability of the interface and increases consistency (Bhagwan Sahane, 2024). The core principles include standardized methods, resource identification, representation of resources, and stateless interactions.

I do not have any personal experience with Uniform Interface, but the resources I found provided further insight. Uniform Interface is extremely relevant to the development process. RESTful APIs provide interoperability and decoupling by implementing "standardized HTTP methods, resource URIs, and uniform resource representations" (Bhagwan Sahane, 2024). It also improves performance through cacheability and scalability (GeeksforGeeks, 2018).

We should implement the Uniform Interface for a few key reasons. The Uniform Interface simplifies the interface, making it more user-friendly (Bhagwan Sahane, 2024). Uniform Interface allows for interoperability, allowing clients and servers to work flawlessly together (Bhagwan Sahane, 2024). Another reason to use the Uniform Interface is that clients and servers are decoupled from each other (Bhagwan Sahane, 2024).

Here is a code example of uniform interface in practice (Bhagwan Sahane, 2024):

Retrieve Product Details:

GET /products/123 HTTP/1.1  
Host: api.example.com  
Accept: application/json

Response:

{  
 "id": 123,  
 "name": "Wireless Mouse",  
 "price": 25.99,  
 "inventory": "In Stock"  
}

Create a New Product:

POST /products HTTP/1.1  
Host: api.example.com  
Content-Type: application/json  
  
{  
 "name": "Keyboard",  
 "price": 45.99  
}

Response:

HTTP/1.1 201 Created  
Location: /products/124

Update a Product:

PUT /products/123 HTTP/1.1  
Host: api.example.com  
Content-Type: application/json  
  
{  
 "price": 22.99  
}

Response:

HTTP/1.1 200 OK

Delete a Product:

DELETE /products/123 HTTP/1.1  
Host: api.example.com

Response:

HTTP/1.1 204 No Content

**References**

Bhagwan Sahane. (2024, August 13). *Mastering REST APIs: Uniform Interface in REST - Bhagwan Sahane - Medium*. Medium. https://medium.com/@bvsahane89/mastering-rest-apis-uniform-interface-in-rest-50b23062156f

GeeksforGeeks. (2018, December 9). *REST API Architectural Constraints*. GeeksforGeeks. https://www.geeksforgeeks.org/javascript/rest-api-architectural-constraints/

**Assignment Requirements and Grading:**

* An initial post of approximately 250 words is due by **Thursday, 11:59 p.m., CST**.
* Submit your post by clicking on the assignment link above, then Create Thread. You must create a thread in order to view your peers' posts. Tip: Create your post in a Word document and then copy and paste your work into the thread.
* A minimum of three (3) responses, to the original threads of other students, of 100-200 words each are due by **Sunday, 11:59 p.m., CST**.
* This discussion board is worth **25 Points**.
* To view the rubric grading criteria, click on the following link: [Discussion Board Grading Rubric.](https://content.bellevue.edu/cst/csd/rubricdbv3.pdf)

Hey, Megan! I think you did a fantastic job of covering the what, how, and why behind the layered system. The code example is a great way to further elaborate your thoughts and show how it can be directly implemented. You are spot on that the layered system is a fundamental part of the REST API. There are so many working parts of the API structure, so breaking each down to fully examine it provides a lot of insight. The uniform interface also provides similar benefits like flexibility. The uniform interface and layered system work in tandem to make a stronger, more improved system.

Hi, Brian! You did an excellent job on your post for this week. You thoughtfully answered the what, how, and why behind the uniform interface. The code snippet from the Medium website did a nice job of adding to your explanations. RESTful APIs are definitely a beneficial system to implement when needing to exchange information digitally. It is especially useful when designing web services. The user should always remain at the forefront of our minds when developing. It should drive our actions as developers. RESTful APIs also make a better experience for developers, which is a huge plus when coding.

Hello there, Arely! I really enjoyed reading your post on stateless in RESTful APIs. You perfectly captured the background history of stateless, how the information is relevant, and how this information is actionable. RESTful APIs have many moving parts, so breaking each of the six down helps to fully comprehend the process and understand how each part is important. A big part of the RESTful APIs is that they are scalable, reliable, flexible, user-friendly, and easily integrable. You are spot on that stateless does not store data. This means with stateless, if a user logs in, their login credentials are not saved.