Seminar 3: Comparing API Architectural Styles - https://levelup.gitconnected.com/comparing-api-architectural-styles-soap-vs-rest-vs-graphql-vs-rpc-84a3720adefa
An article about the different types of APIs and how they compare.
Keywords: API, REST, RPC, SOAP, GraphQL, *whatever word you didn't understand* etc.

- **What is the paper about?**

The study examines four API architectural styles: RPC, SOAP, REST, and GRAPHQL, and discusses their benefits, drawbacks, and application examples.

- **What is a server / client stub, in the context of RPC?**

A stub handles argument serialization and deserialization in the context of RPC.

- **What does it mean to be integrated with WS-security protocols? Exemplify some of these protocols and what they protect against.**

The WS-Security protocols provide transactional privacy and integrity while also providing for message encryption. At the message level, the authentication, integrity, and confidentiality techniques for safeguarding web services are used. Authentication verifies a user's identity and evaluates whether or not a client is legitimate in a given situation. Information integrity guarantees that data isn't unintentionally modified, updated, or lost. Confidentiality ensures that no porty or process may read or expose the contents of a communication by encrypting it.

- **How do you understand HATEOAS?**

With HATEOS, responses to REST requests return not only data, but also actions that can be performed on the resources. This helps make applications loosely coupled.

- **"GraphQL has *subscriptions*" - What are subscriptions? Why would we need them?**

Subscriptions are GraphQL features that allows a server to send data to its clients when a specific event happens. Subscriptions are useful for notifying your client in real-time about changes to back-end data, such as the creation of a new object or updates to an important field.

- **Order the API patterns by message size.**
    1. RPC
    2. GRAPHQL
    3. REST
    4. SOAP

- **Which API pattern would best fit your laboratory work? Why?**

Suitable will be the use of REST, because it has the highest level of abstraction and best modeling of the API. It permits to have decoupled client and server, achieving flexibility and remaining stable which is great for distributed systems.