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 paper is about 4 API architecture styles: RPC, SOAP, REST, GRAPHQL, and describes each of them in details, the advantages, disadvantages and some examples of use

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

The serialization and deserialization of parameters is handled by a stub 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 guarantee privacy and integrity inside transactions while also providing for message encryption. Using authentication, integrity, and confidentiality techniques to secure web services at the message level. Authentication verifies the identity of the user and determines if a client is valid in a certain situation. Integrity guarantees that data is not mistakenly modified, altered, or lost. Message encryption is used to ensure that no porty or process may access or leak the information included in the message.

- **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 capabilities that allow a server to transmit data to its clients in response to a specified event. Subscriptions are excellent for informing your clients in real time of changes to back-end data, such as the creation of a new object or modifications to a critical 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.