Chapter 02 - Selecting Your API Architecture

API Architectural Styles

One of the most important decisions that you need to make $\xrightarrow{\text{in this case}}$ Consumer-centric design process

- REST
- Webhooks
- GraphQL
- Simple Object Access Protocol (SOAP)
- WebSockets
- gRPC

Representational State Transfer (REST)

- REST by Roy Fielding's Doctoral Dissertation - Architectural Styles and the Design of Network-Based Software Architectures
  - Not very useful in practice
- Useful implementation of REST = Pragmatic REST or RESTful
  Formal definitions and pragmatic practices:
- Client/Server Model: API providers make resources available at individual addresses $\xrightarrow{\text{then}}$ Consumers make requests to these resources using standard HTTP verbs $\xrightarrow{\text{then}}$ Producers provide a response
- The response is defined by the producer.
  - The standard structure of the response is the same for each consumer (Consistency in Response handled by Producer)
- REST response is typically in JSON, sometimes XML
  - Both of them are standard text-based data transfer formats
- The interaction is stateless $\xrightarrow{\text{means}}$ each massage back and forth stands on its own.
  - In conversation of multiple requests and response, each request has to provide information or context from previous responses
- REST APIs are defined by OpenAPI Specification file
- Use API versions to protect existing consumers from changes

Key Terms of REST

- Consider RESTful API to be set of endpoints that are related to the same data source
- API version is a group of endpoints that are consistent for some time so that consumers can count on them
- An API endpoint (operation) is a combination of two fundamental building blocks
  - HTTP verb
  - URL path

Graph Query Language (GraphQL)

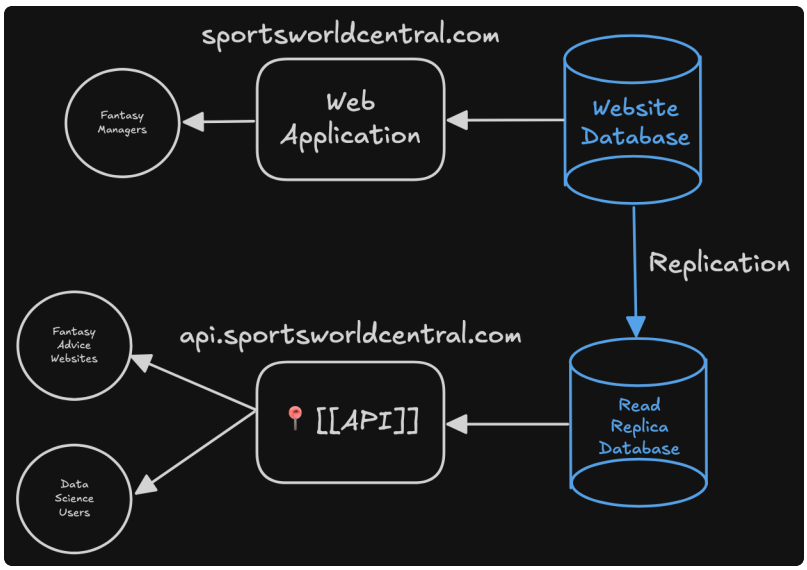
- GraphQL is both query language for API + Query runtime engine
- Developed by Facebook, made open source in 2015
- Big Advantage of GraphQL over REST = fewer API call is needed for the consumer to get what they want $\xrightarrow{\text{so}}$ less network traffic
  Attributes of GraphQL:
- Client/Server model (same as REST)
- Stateless (same as REST)
- Responses are usually JSON (same as REST)
- Instead of only using HTTP verbs, consumer can use the GraphQL query language
- Consumer can specify the contents of response along with query options (unlike REST which the producer decide)
- The producer makes the API available at a single address + consumer passes queries to it via the HTTP POST verb.
- Versioning is not recommended $\xrightarrow{\text{why?}}$ because the consumer defines the contents they are requesting

gRPC

- Deployed by Google, open sourced in 2015
- Very fast, efficient between micro-services
  Attributes of gPRC:
- no more sharing resources $\xrightarrow{\text{instead}}$ remote procedure calls $\approx$ traditional code functions
- Not limited to stateless request-response pattern $\xrightarrow{\text{instead}}$ continues streaming
- Instead of JSON use protocol buffers
  - serializing data that is smaller and faster than JSON & XML
- No more OpenAPI specification file $\xrightarrow{\text{instead}}$ protocol buffers as the specification (in a `.proto` file)
- LLMs using it!

Technology Architecture

- Create the API as part of the web application
- Create the API as a separate application but allow the API to read directly from the website database
  Advantage = Data is always up-to-date with the web application
  Disadvantage = Could slow down the web application if a large number of requests are being made to the API
  * You can physically separate API from the web application to solve this problem



Additional Resources

- The Ten REST Commandments