



# INTRO TO BACKEND DEVELOPMENT

DAY 12

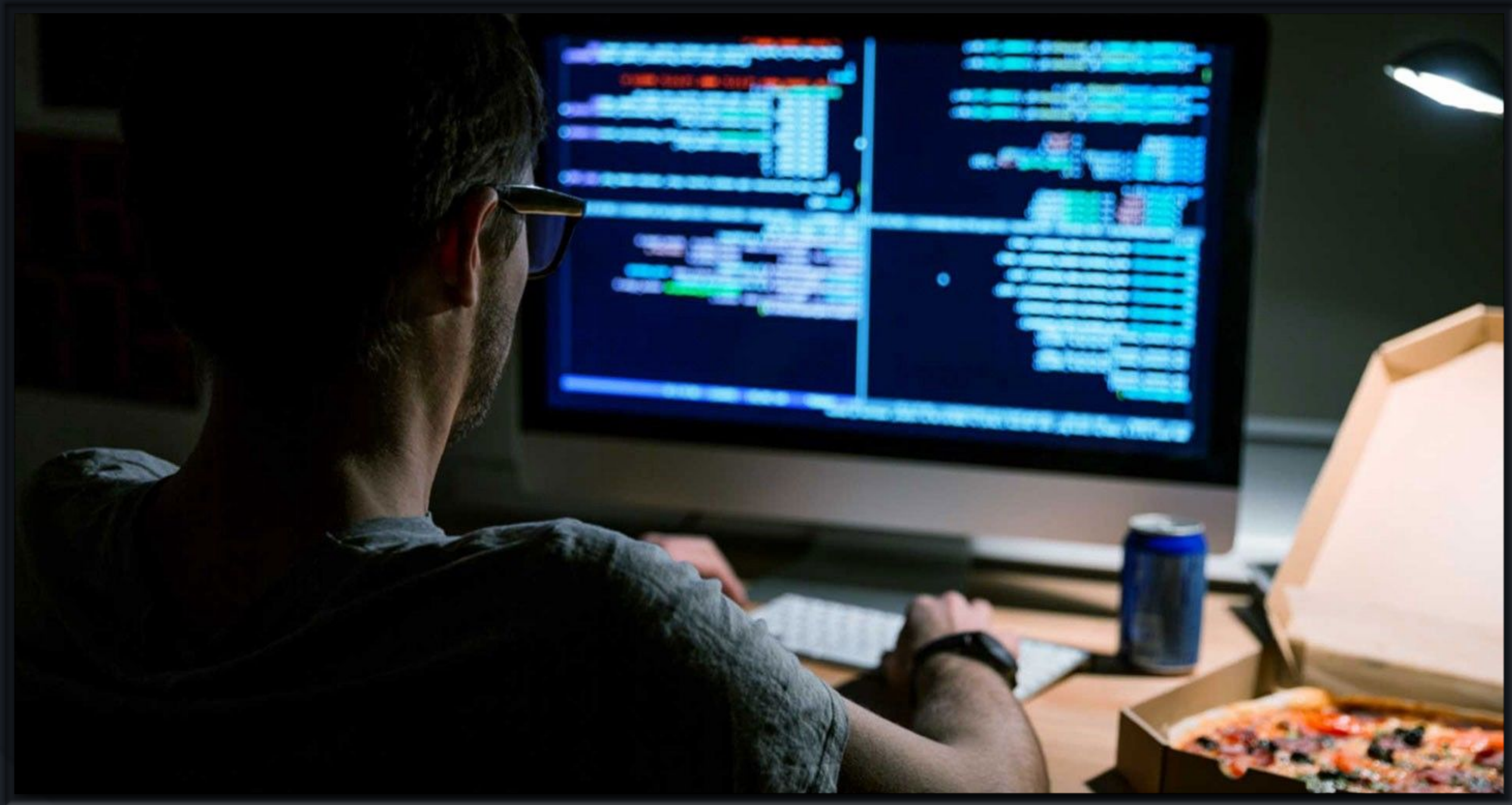
*04/21/2021*

*Instructor - Casey Wilson*

*TA - Kevin Dublin*

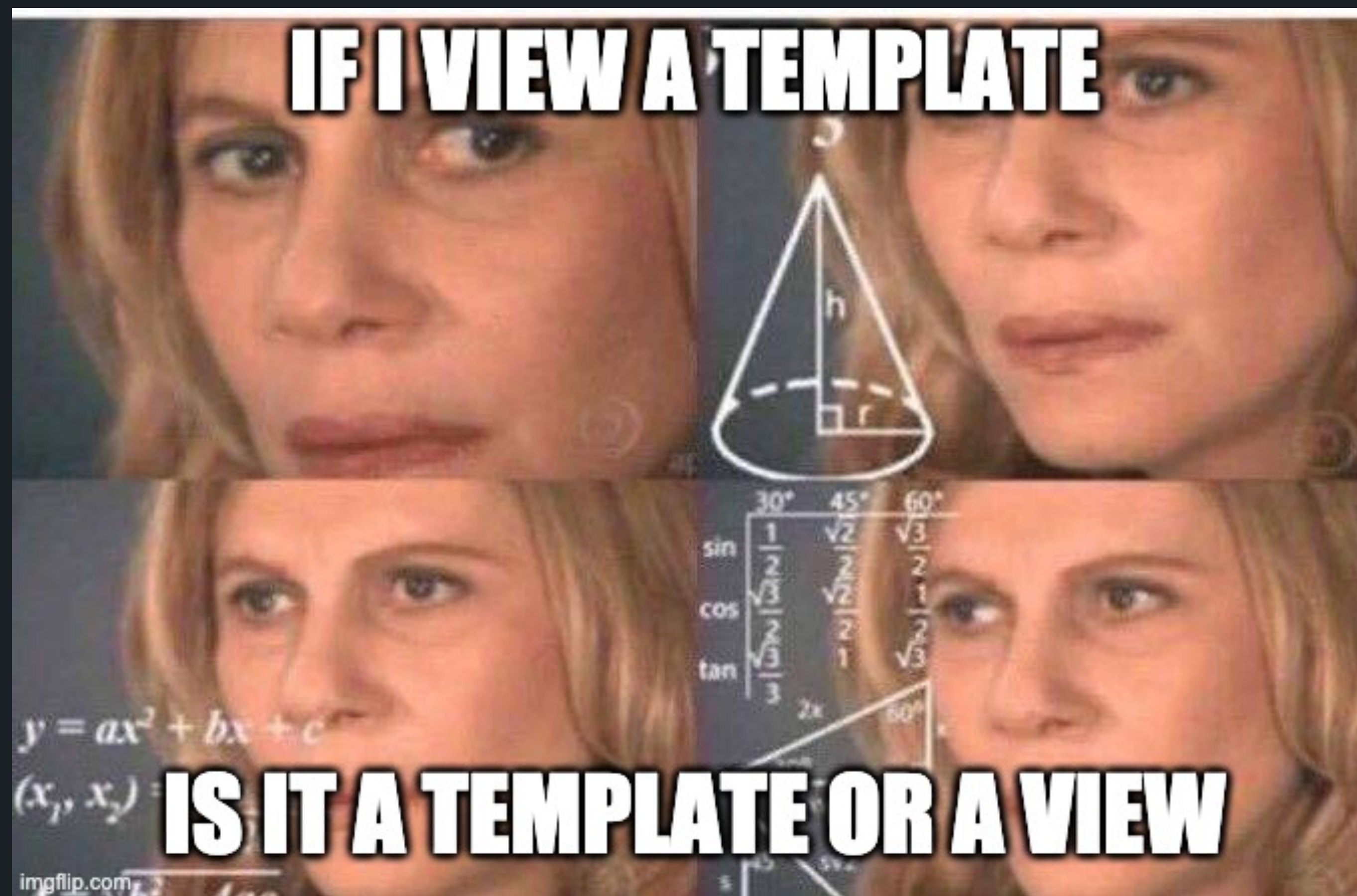


# *Take Home Challenge Review*





# Check In Time





# *Django Time - Round 4*



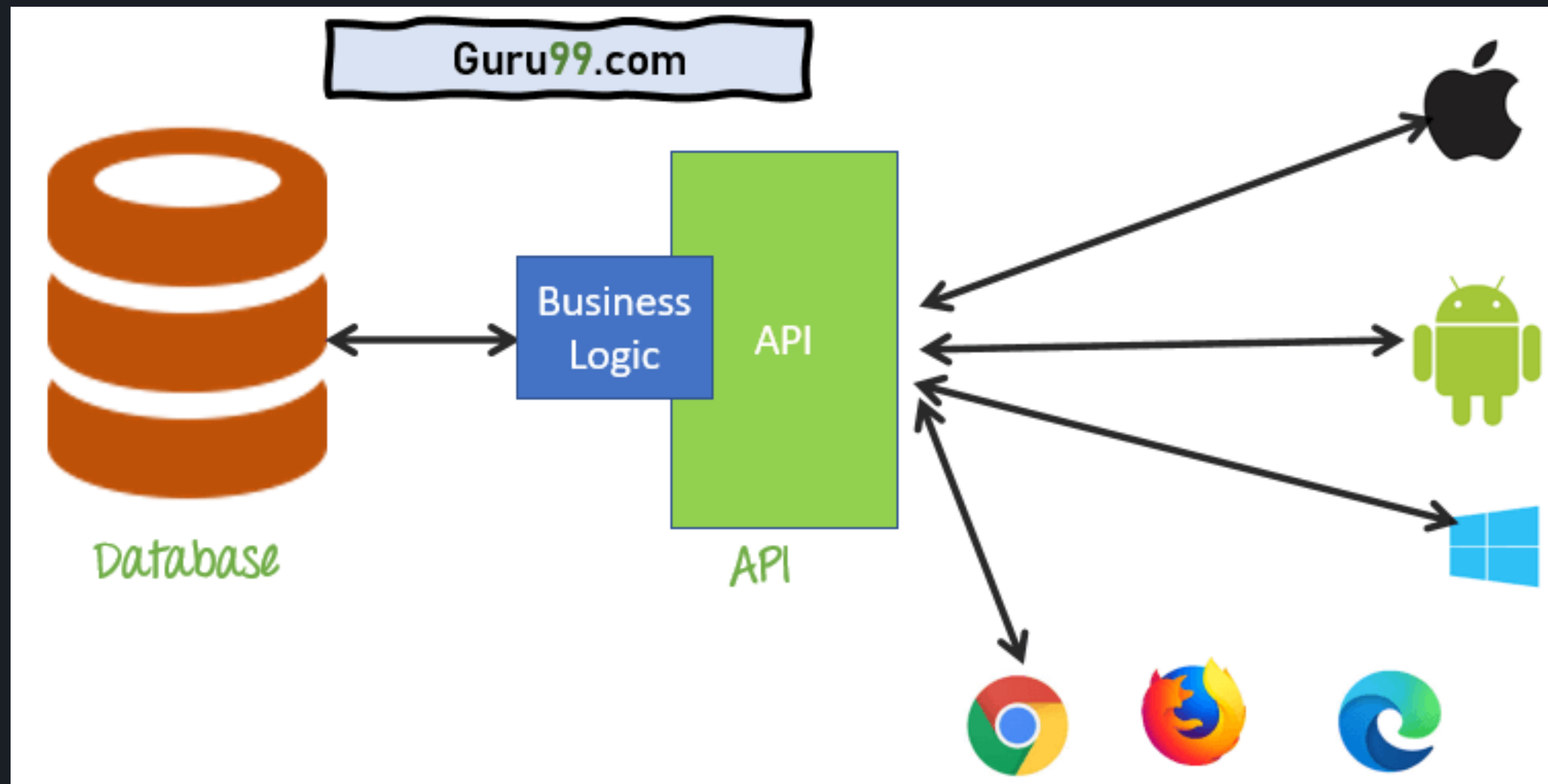


# *Django - ORM Shell*

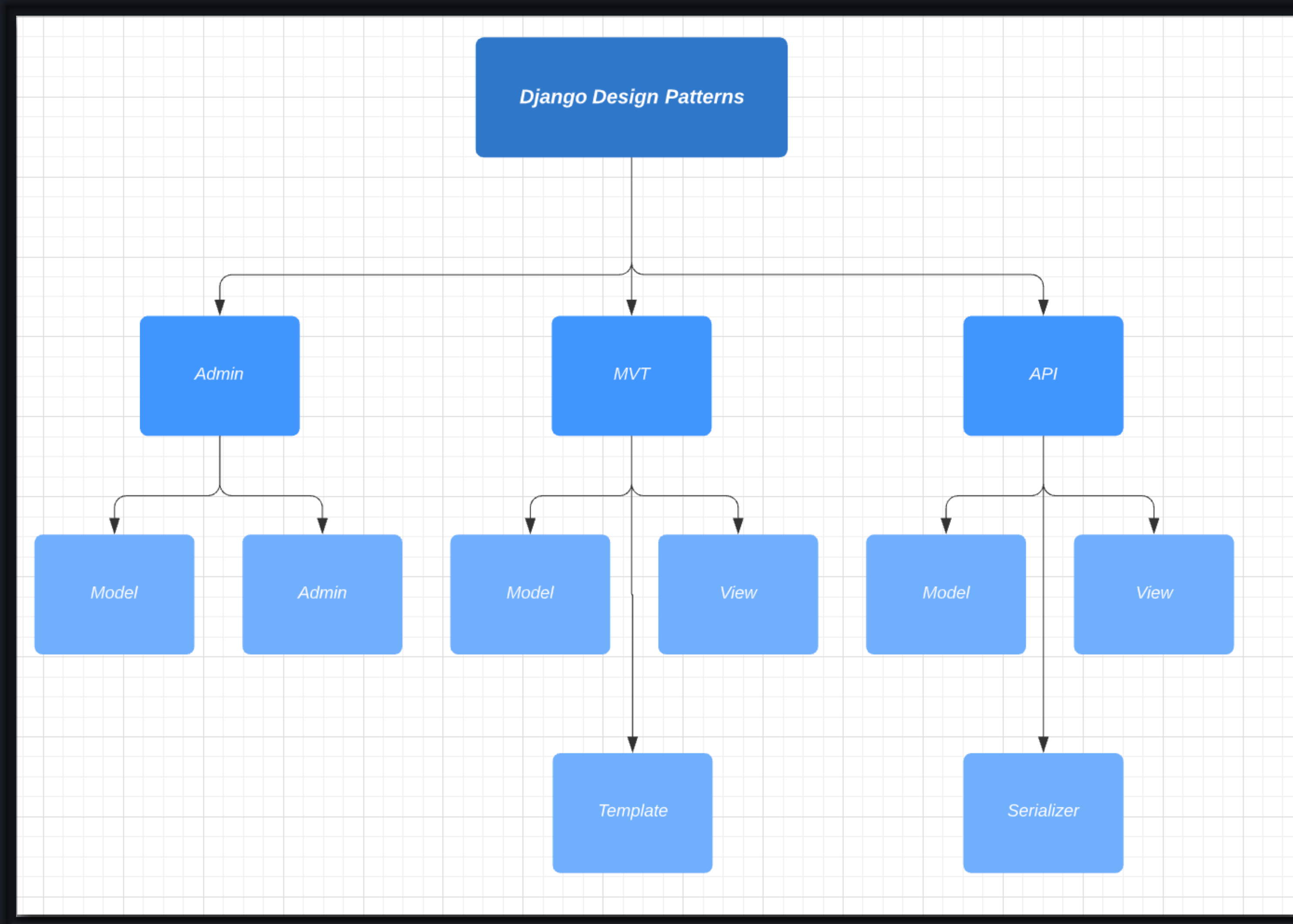


<https://github.com/chrisdl/Django-QuerySet-Cheatsheet>

# *What is an API?*



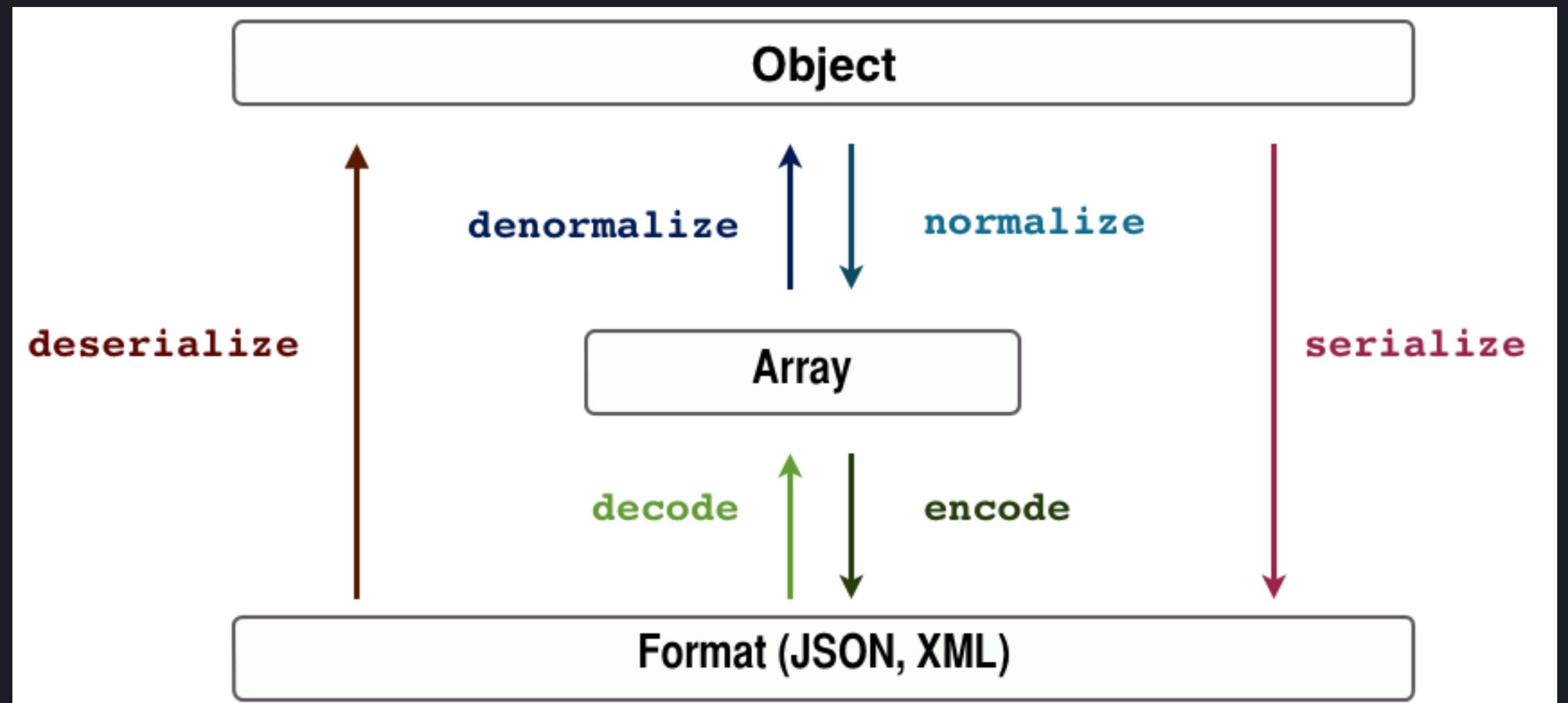
# *Django - Choose Your Path*





# Django - What is a serializer?

- ✧ Serializer
  - ✧ Translates Database Objects into a “common” format
  - ✧ Typically JSON
  - ✧ Can be XML and others





# *Django - API Packages*



**django**

# API - Architectures

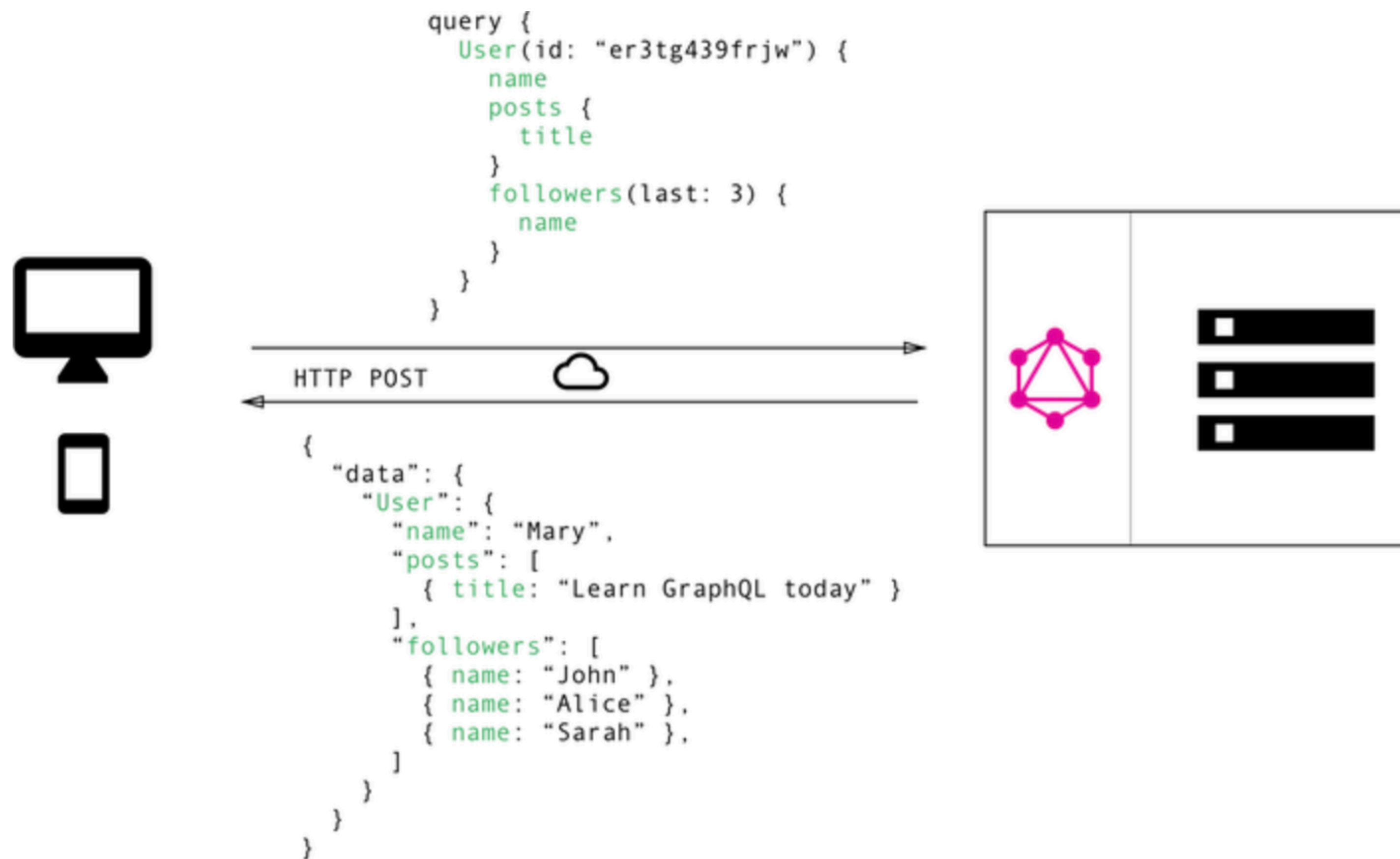
API ARCHITECTURAL STYLES				
	RPC	SOAP	REST	GraphQL
Organized in terms of	local procedure calling	enveloped message structure	compliance with six architectural constraints	schema & type system
Format	JSON, XML, Protobuf, Thrift, FlatBuffers	XML only	XML, JSON, HTML, plain text,	JSON
Learning curve	Easy	Difficult	Easy	Medium
Community	Large	Small	Large	Growing
Use cases	Command and action-oriented APIs; internal high performance communication in massive micro-services systems	Payment gateways, identity management CRM solutions financial and telecommunication services, legacy system support	Public APIs simple resource-driven apps	Mobile APIs, complex systems, micro-services



# API - Rest Example



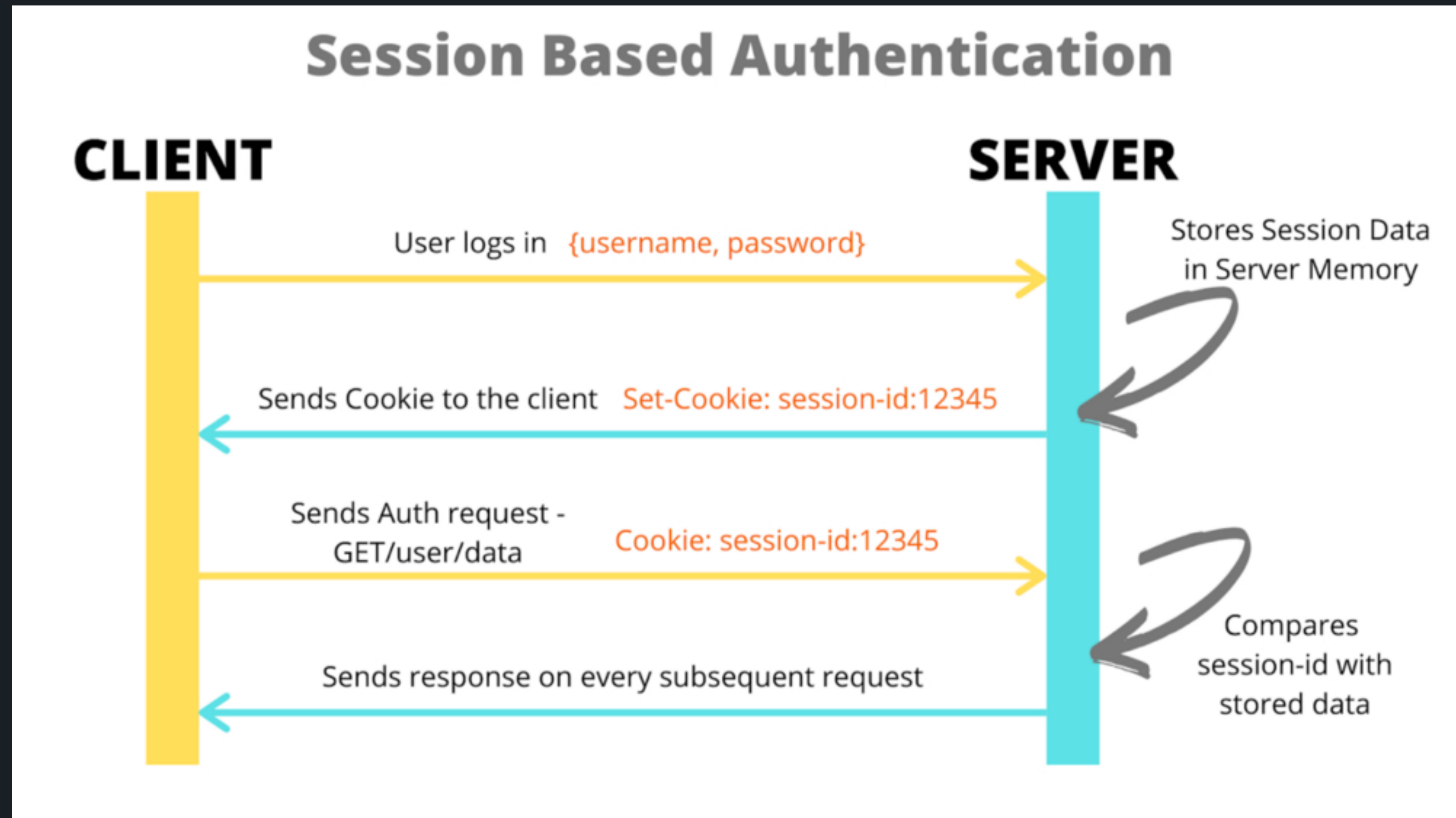
# API - GraphQL Example



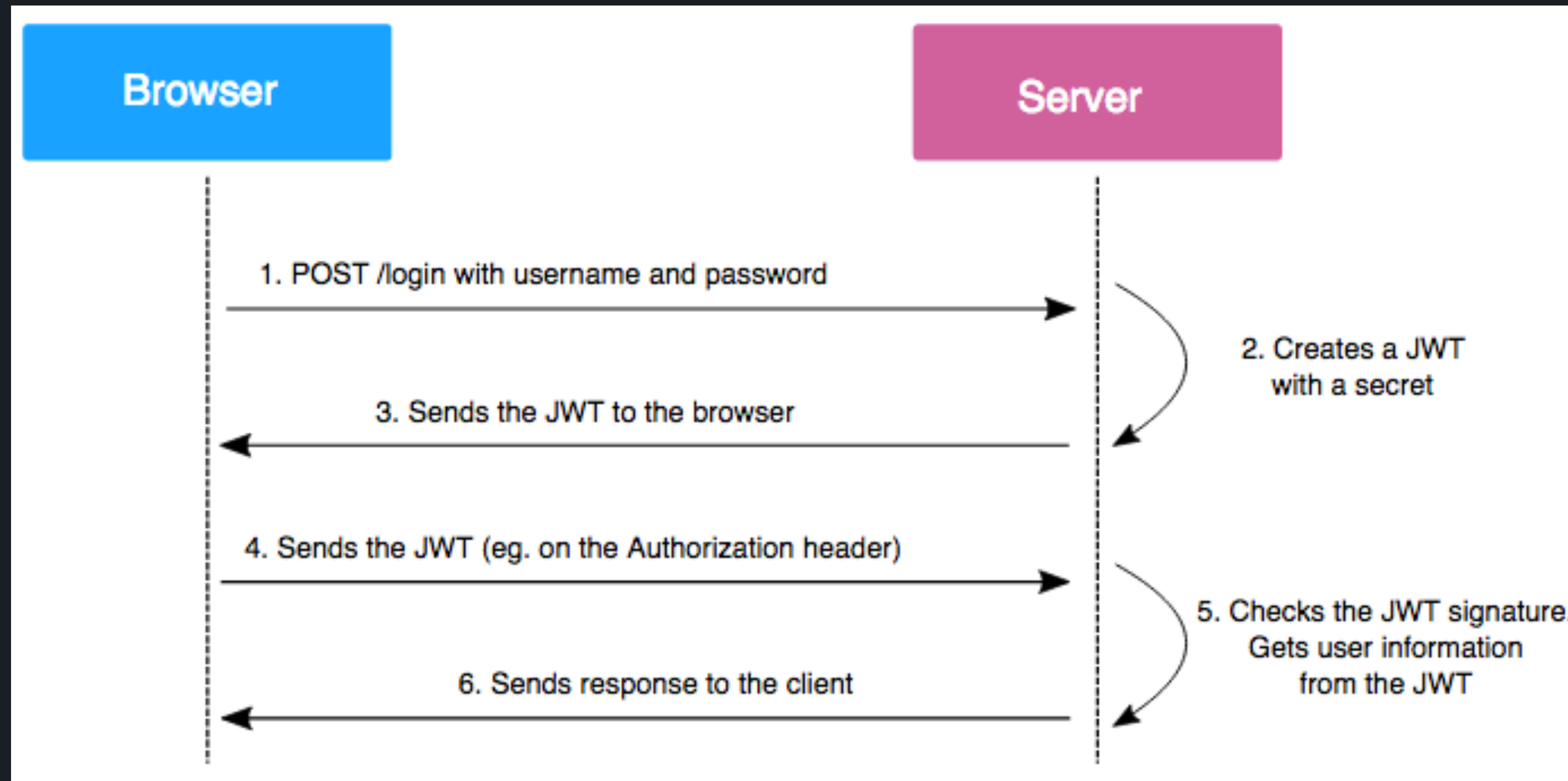
Source



# DRF Authentication - Cookie



# DRF Authentication - JWT



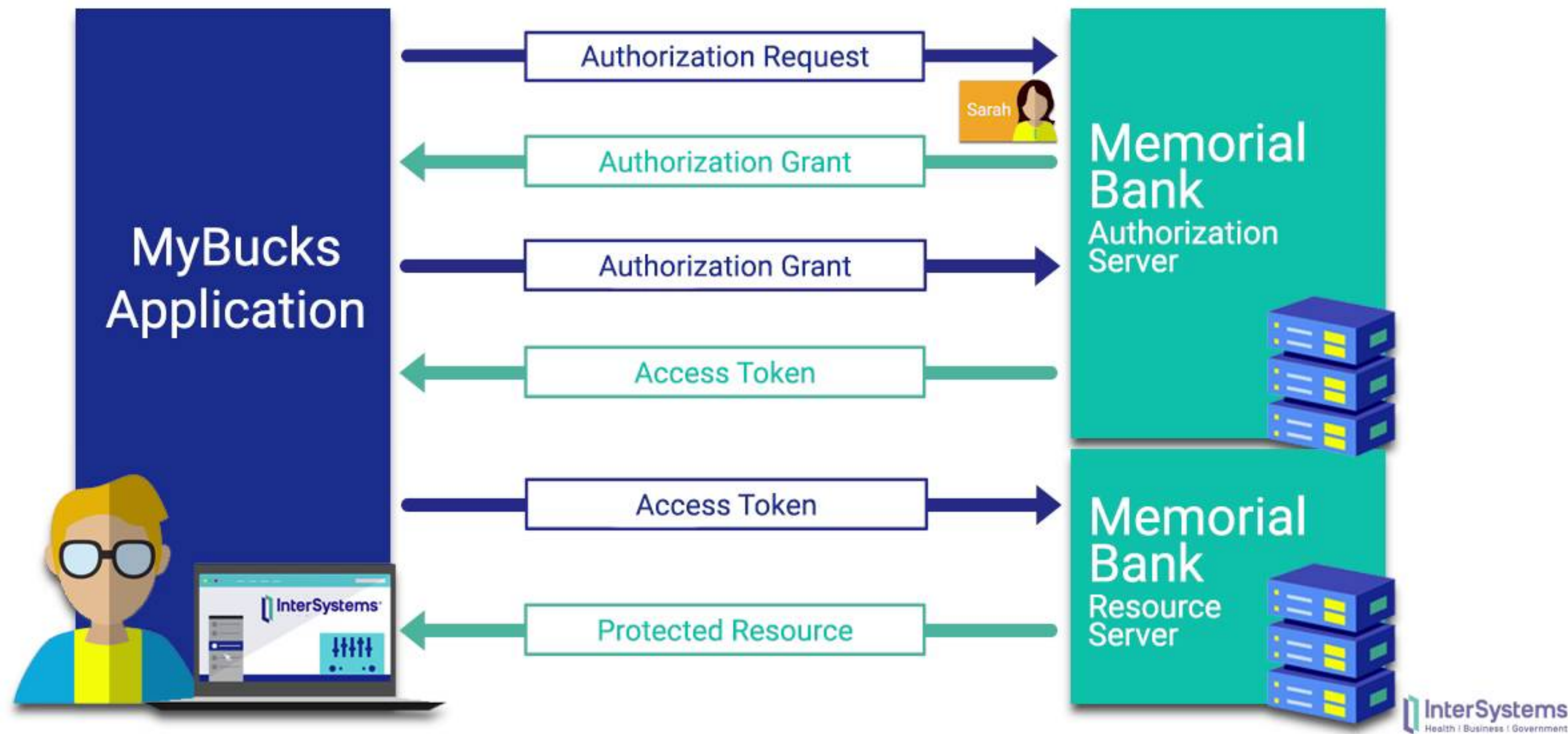


# *DRF Authentication - Cookie vs JWT*

Σ Stateless	Cookies	JWT
	<ul style="list-style-type: none"><li>• Contains a session id</li><li>• Requires a database lookup on every request</li><li>• Server-side sessions require subsequent requests to hit same server</li><li>• Scaling difficult</li></ul>	<ul style="list-style-type: none"><li>• Contains verified user information</li><li>• No db lookups required</li><li>• State is stored on client</li><li>• Scales easily</li></ul>

# *But what about OAuth(2.0)?*

## Workflow of OAuth 2.0





# *Which One Do You Use?*

## ✦ **Cookie / Session**

- ✦ Traditional Web Apps
- ✦ Important Points
  - ✦ Mark cookies as HTTP Only
  - ✦ Only allow Same-Site requests
  - ✦ Give expirations

## ✦ **OAuth (2.0)**

- ✦ Non browser based support (smartwatch, mobile, IoT, etc)
- ✦ Important Points
  - ✦ Route guard you app with redirects to auth server
  - ✦ Register token with client app attributes on server
  - ✦ Give Expirations

## ✦ **JWT**

- ✦ Microservice or millions of users+
- ✦ Important Points
  - ✦ Never store in HTML5 local storage
  - ✦ Only send over secure channels (HTTPS)
  - ✦ Give expirations and exclude old tokens

# *Questions?*





# *Take Home Challenge*

