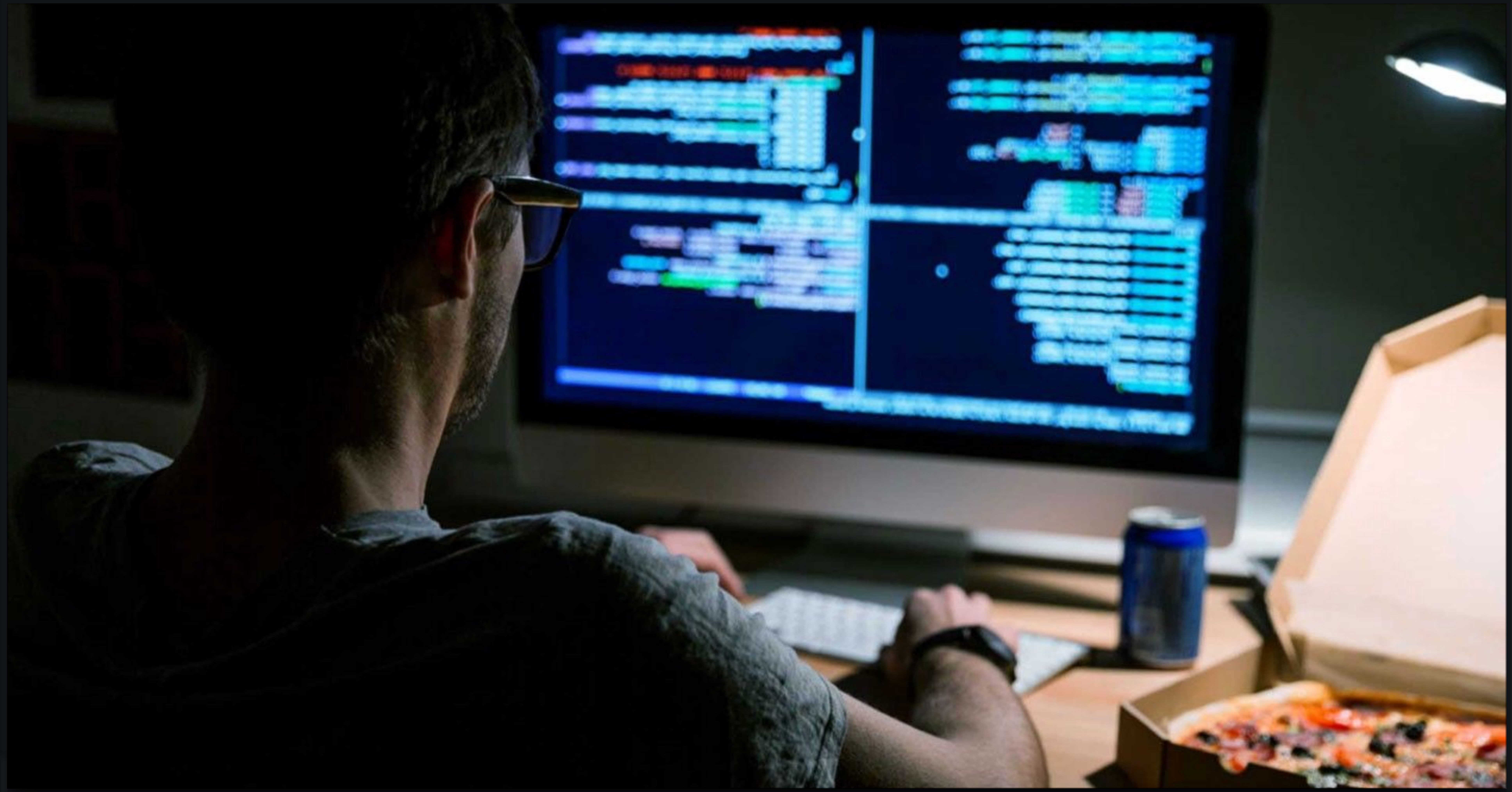
INTRO TO BACKEND DEVELOPMENT

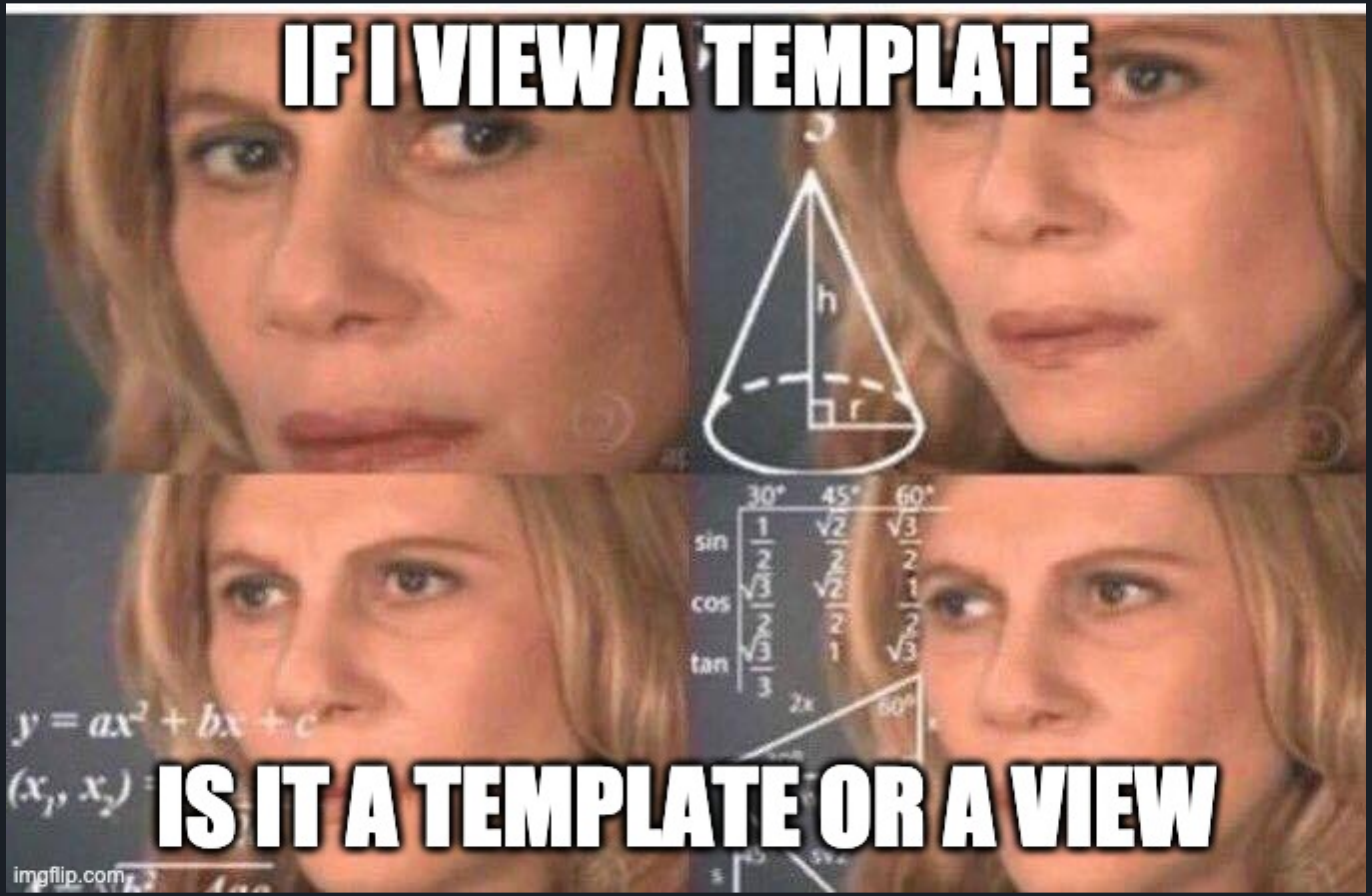
DAY 12

04/21/2021
Instructor - Casey Wilson
TA - Kevin Dublin

Take Home Challenge Review



Check In Time



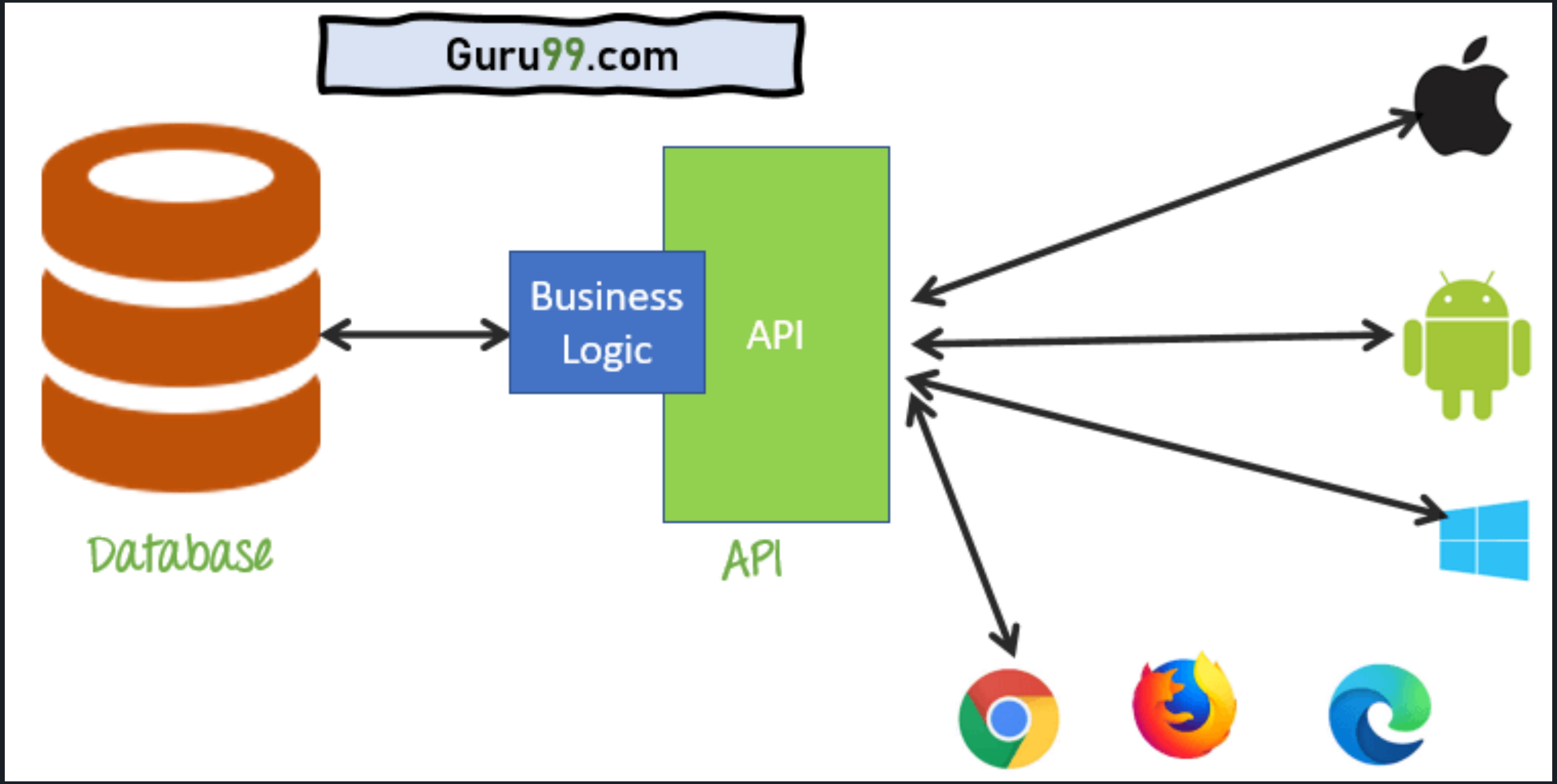
Django Time - Round 4

Django - ORM Shell

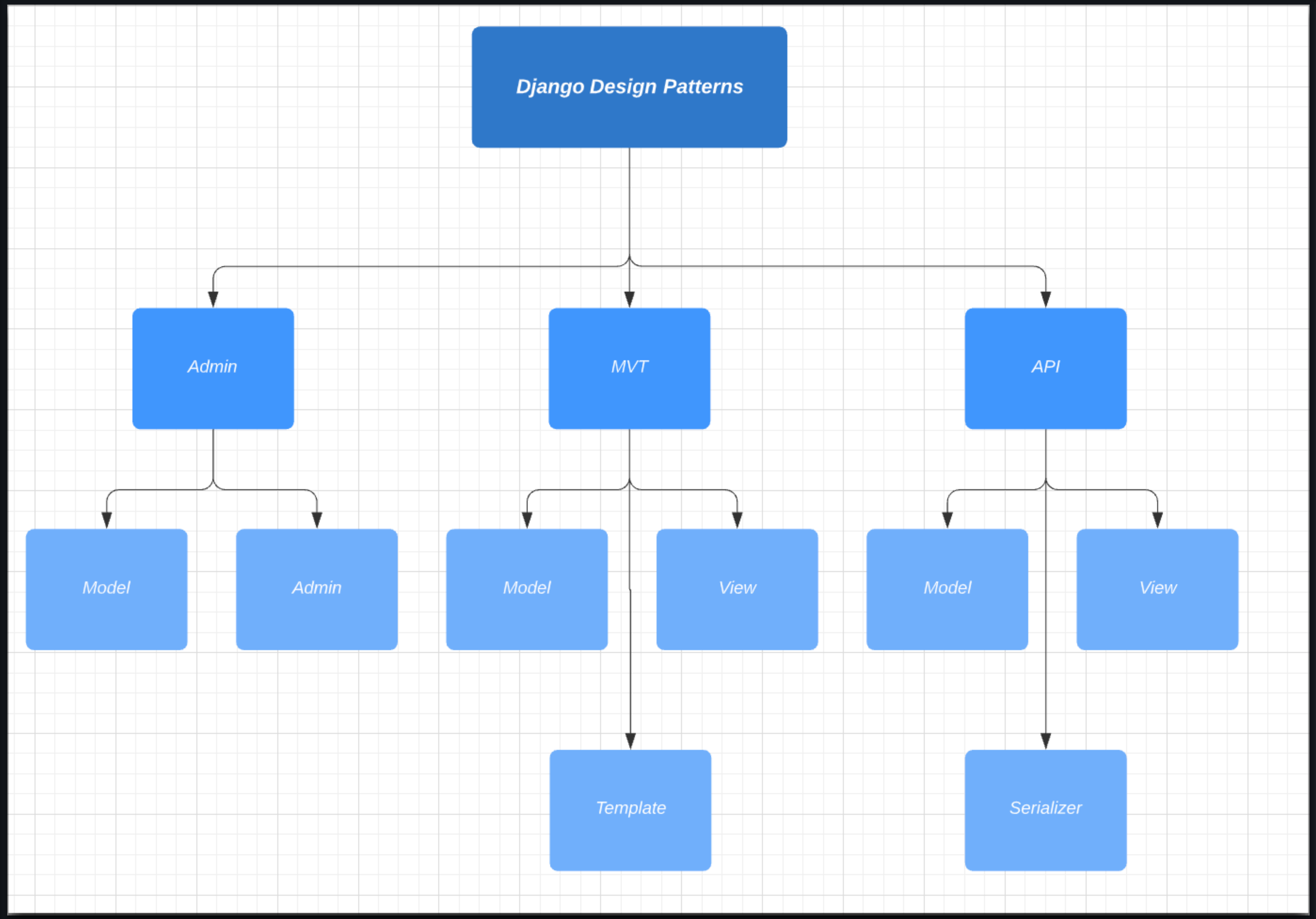
OH, SO YOU LEARNED HOW TO USE COMMAND PROMPT?
TELL ME MORE ABOUT YOUR VAST KNOWLEDGE OF BINARY PROGRAMMING
memegenerator.net

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

What is an API?

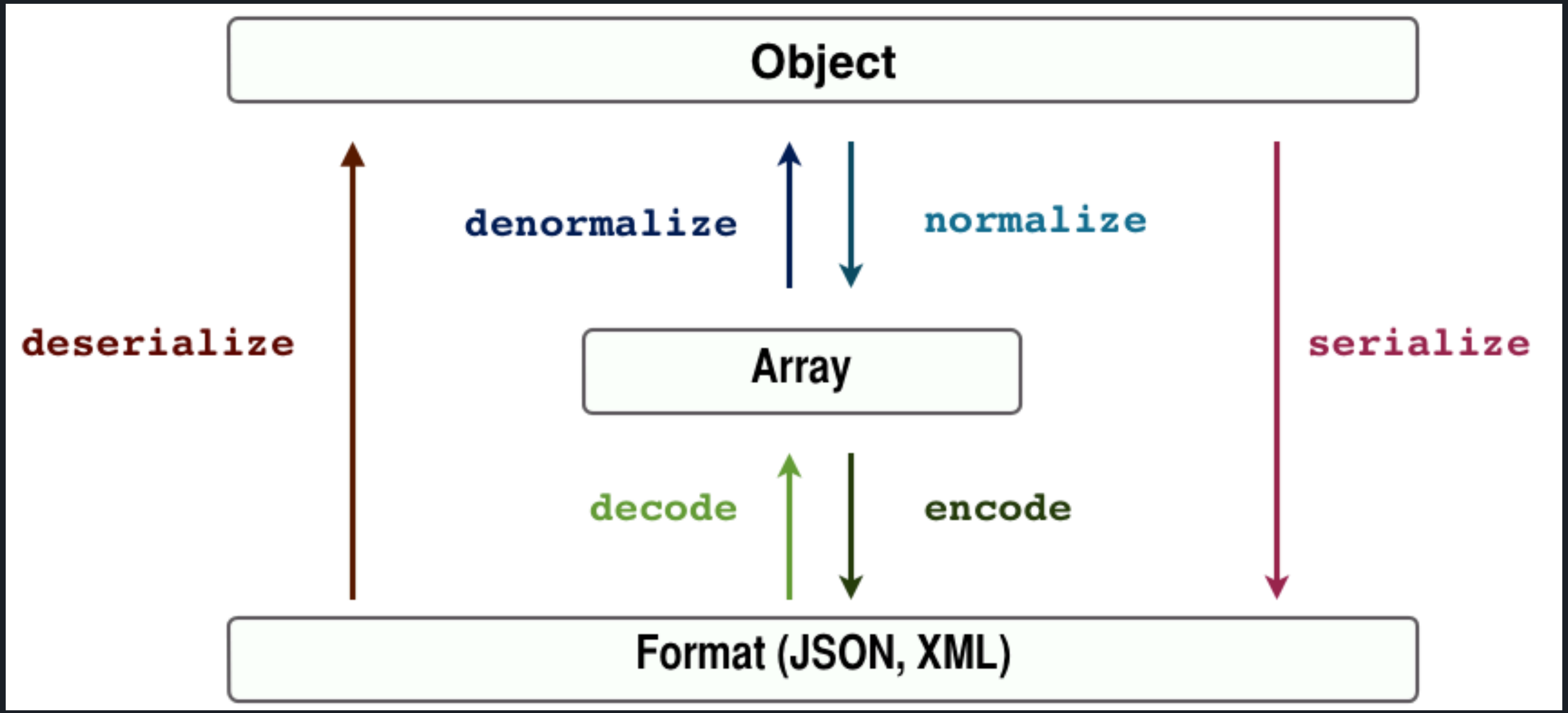


Django - Choose Your Path



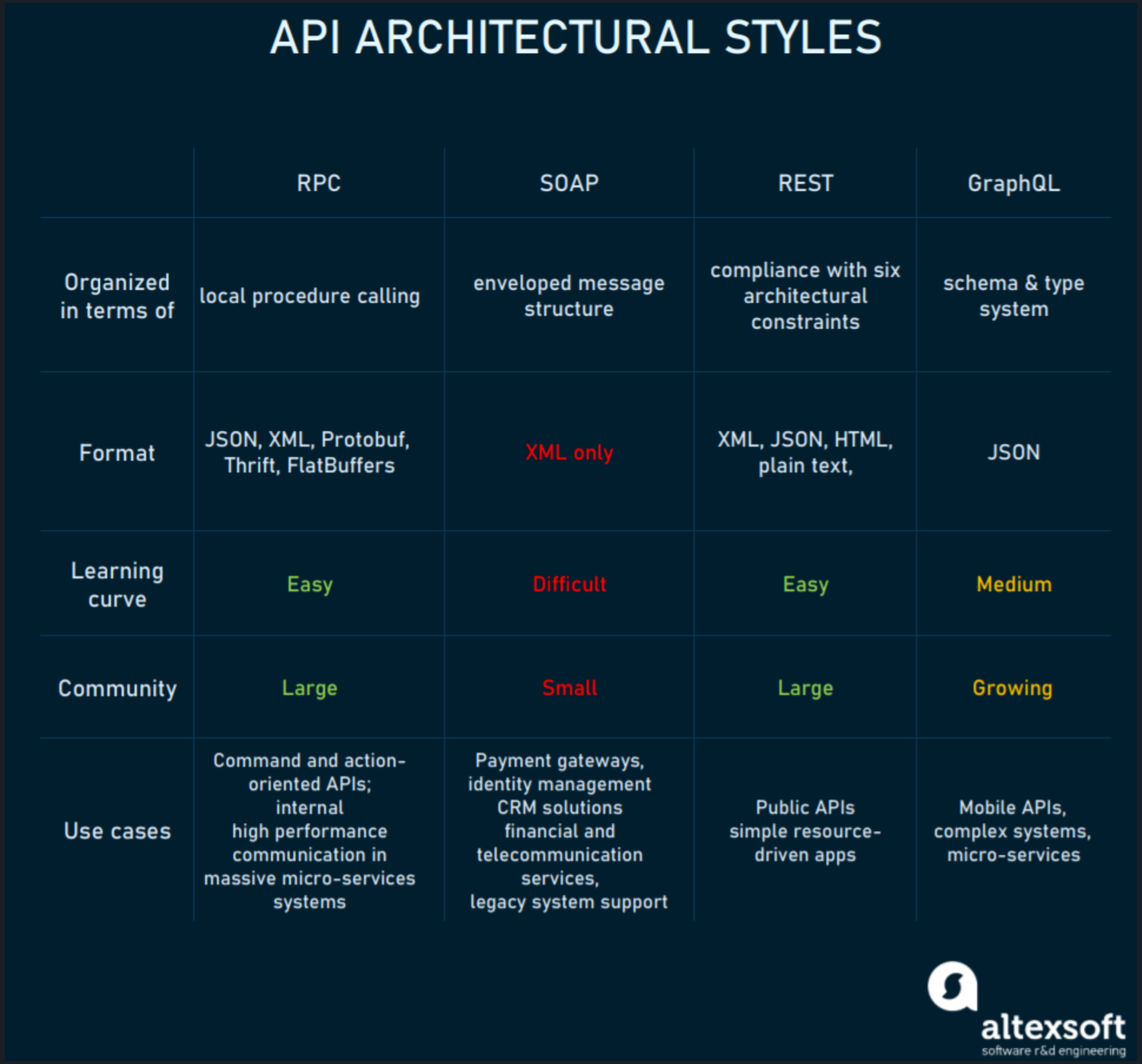
Django - What is a serializer?

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



Django - API Packages

API - Architectures

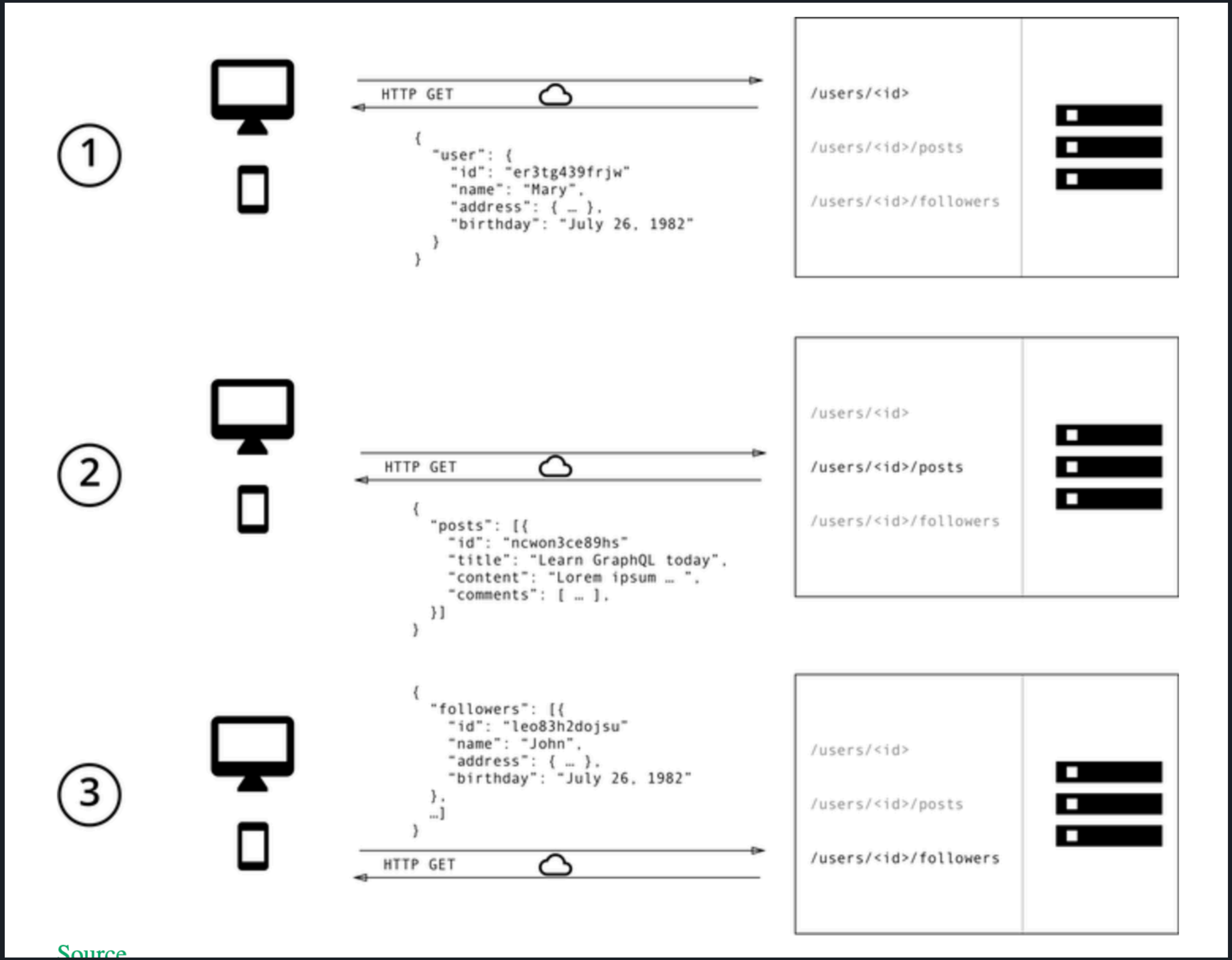


API ARCHITECTURAL STYLES

	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

altexsoft
software r&d engineering

API - Rest Example



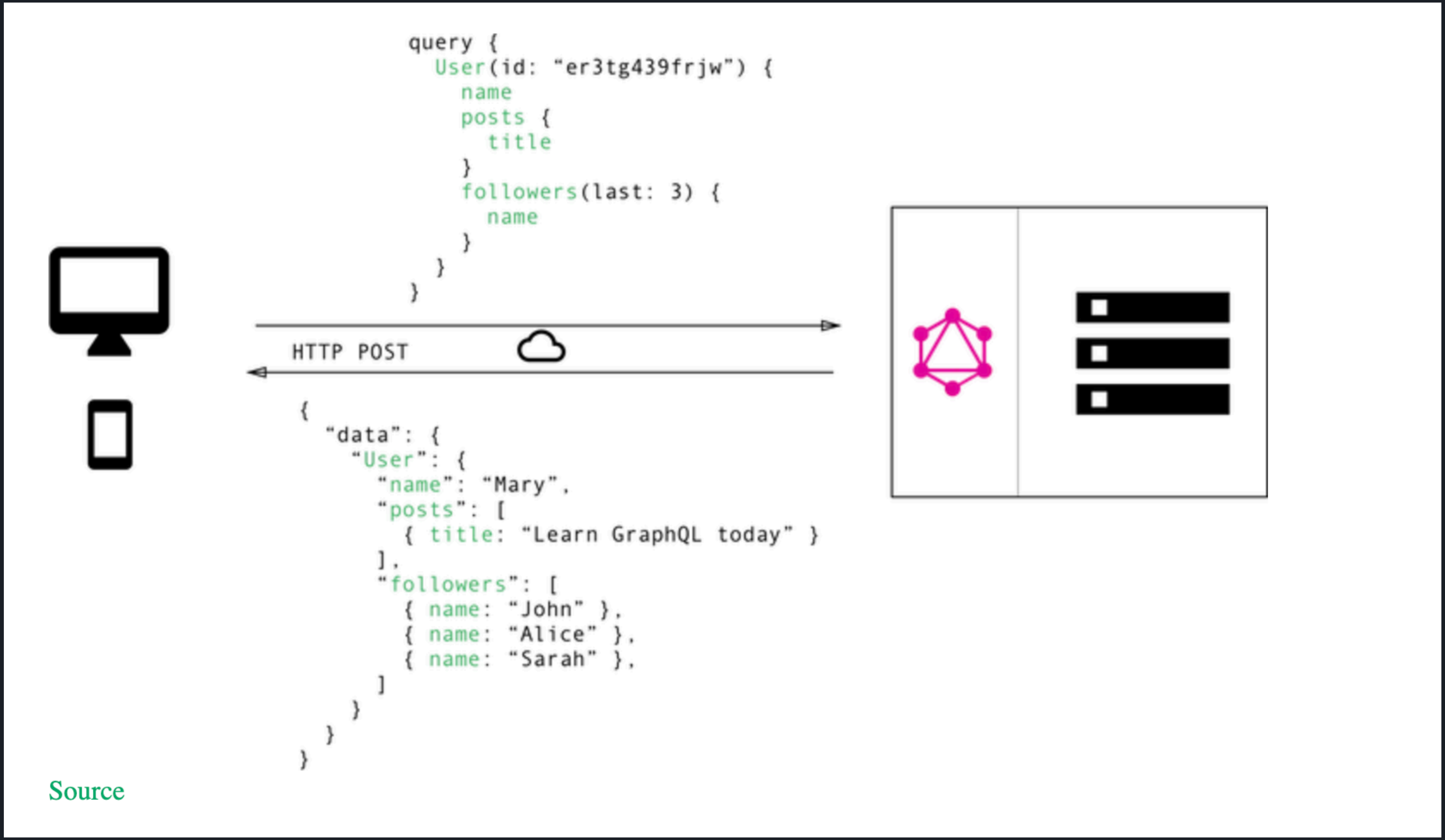
```
{
  "user": {
    "id": "er3tg439frjw"
    "name": "Mary",
    "address": { … },
    "birthday": "July 26, 1982"
  }
}
```

```
{
  "posts": [{
    "id": "ncwon3ce89hs"
    "title": "Learn GraphQL today",
    "content": "Lorem ipsum … ",
    "comments": [ … ],
  }]
}
```

```
{
  "followers": [{
    "id": "leo83h2dojsu"
    "name": "John",
    "address": { … },
    "birthday": "July 26, 1982"
  },
  …]
}
```

Source

API - GraphQL Example

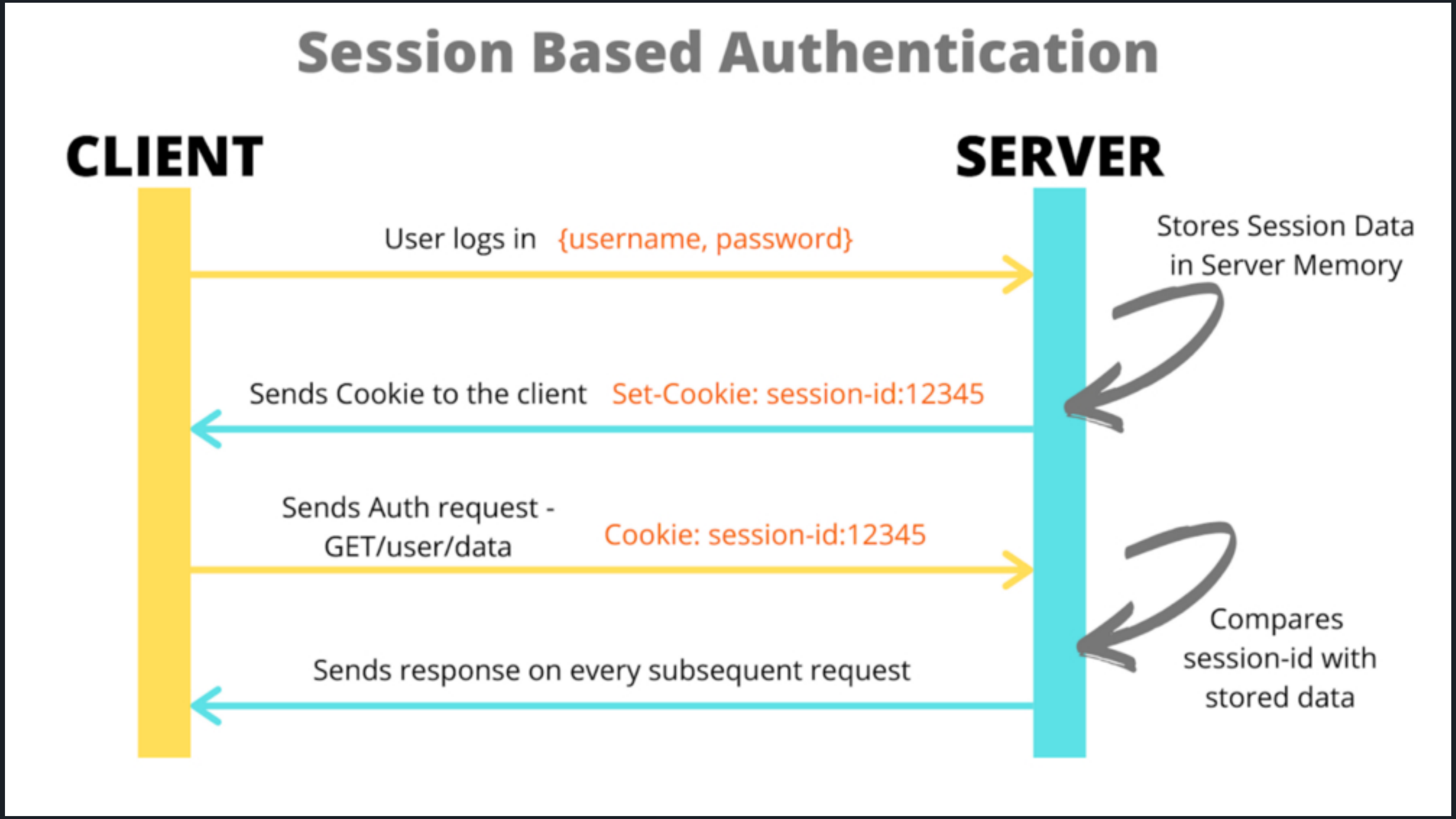


```
query {
  User(id: "er3tg439frjw") {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```

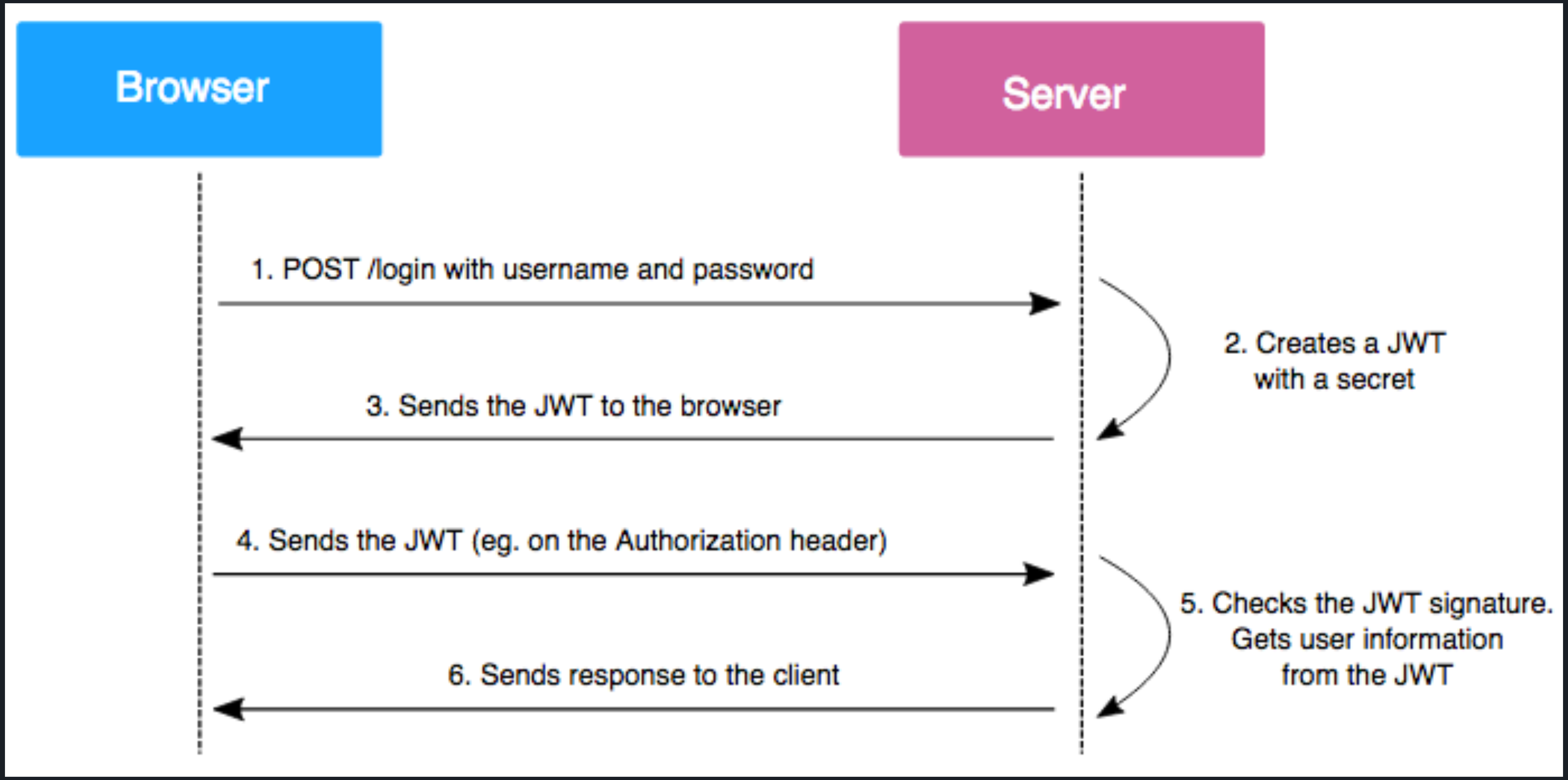
```
{
  "data": {
    "User": {
      "name": "Mary",
      "posts": [
        { title: "Learn GraphQL today" }
      ],
      "followers": [
        { name: "John" },
        { name: "Alice" },
        { name: "Sarah" },
      ]
    }
  }
}
```

Source

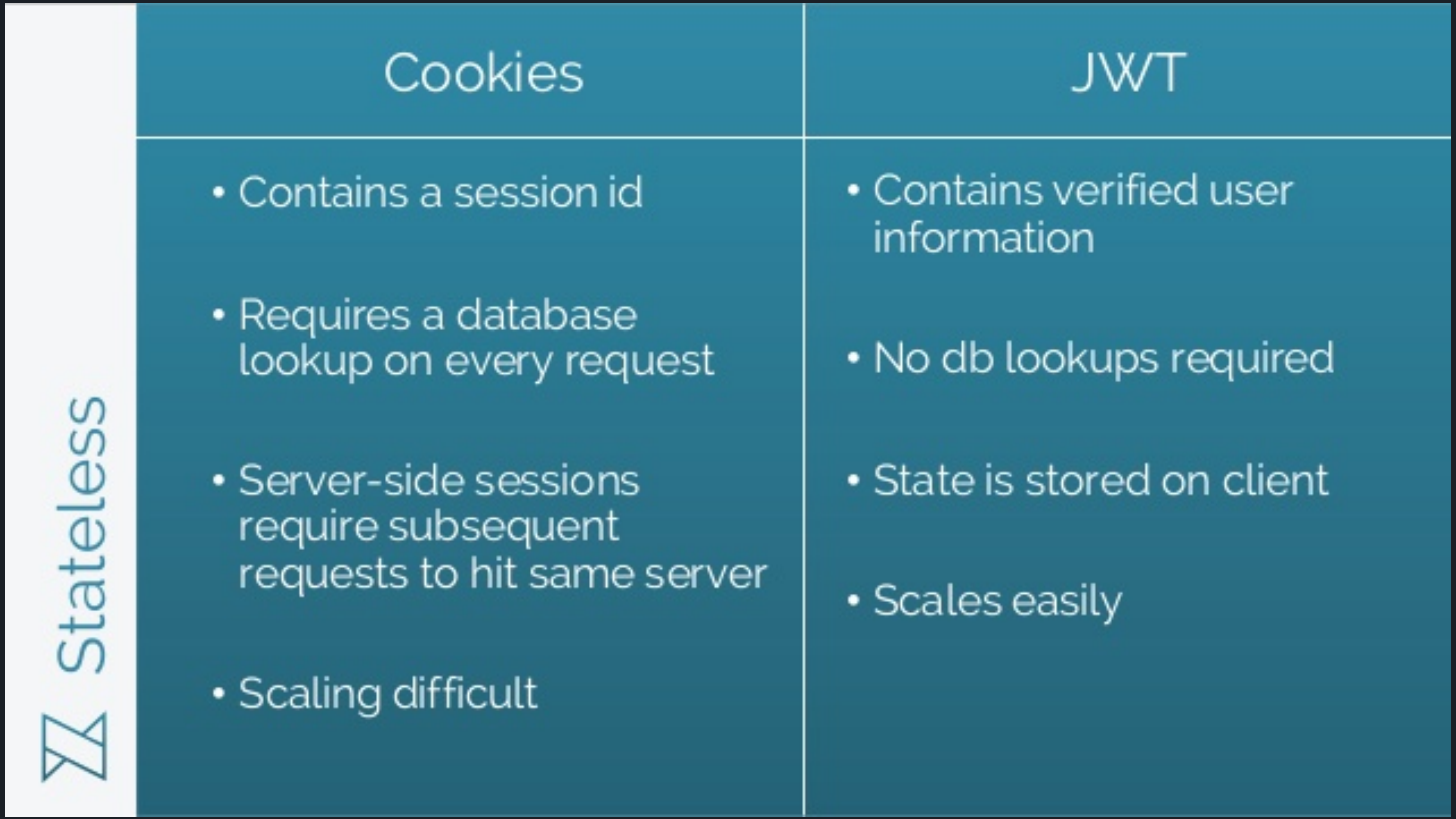
DRF Authentication - Cookie


Session Based Authentication
CLIENT
SERVER
User logs in {username, password}
Stores Session Data in Server Memory
Sends Cookie to the client Set-Cookie: session-id:12345
Sends Auth request - GET/user/data
Cookie: session-id:12345
Compares session-id with stored data
Sends response on every subsequent request

DRF Authentication - JWT

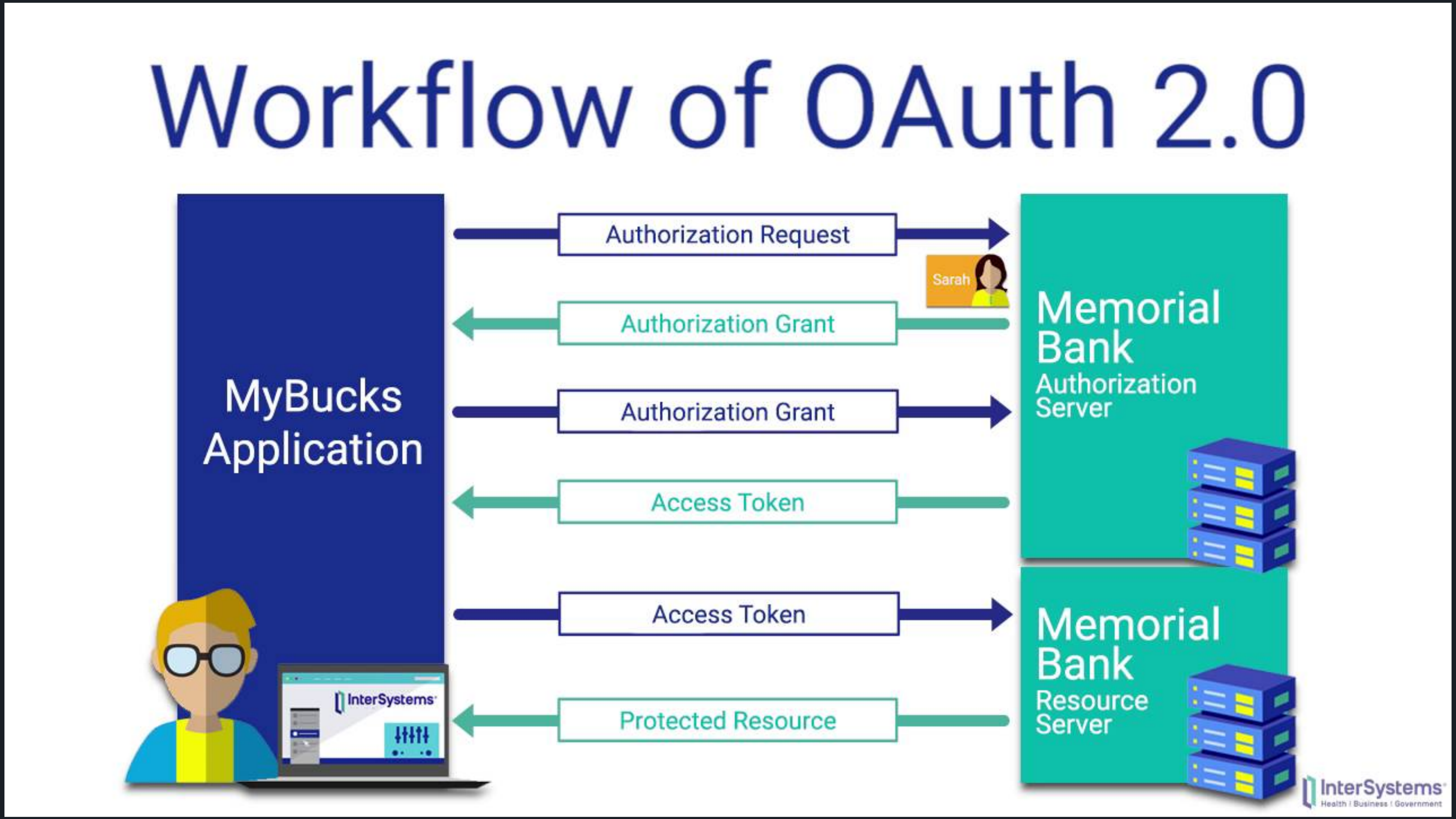

Browser
Server
1. POST /login with username and password
2. Creates a JWT with a secret
3. Sends the JWT to the browser
4. Sends the JWT (eg. on the Authorization header)
5. Checks the JWT signature. Gets user information from the JWT
6. Sends response to the client

DRF Authentication - Cookie vs JWT



Cookies	JWT
• Contains a session id	• Contains verified user information
• Requires a database lookup on every request	• No db lookups required
• Server-side sessions require subsequent requests to hit same server	• State is stored on client
• Scaling difficult	• Scales easily

But what about OAuth(2.0)?



Which One Do You Use?

- **Cookie / Session**
    - Tradiitonal 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 blacklist old tokens

Questions?

Take Home Challenge