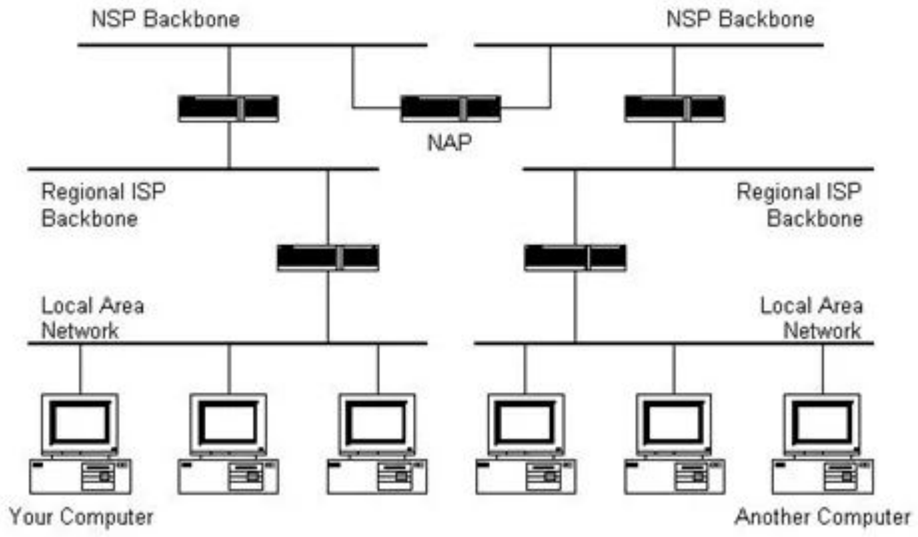
Introduction to Software Engineering

Network Basics Overview

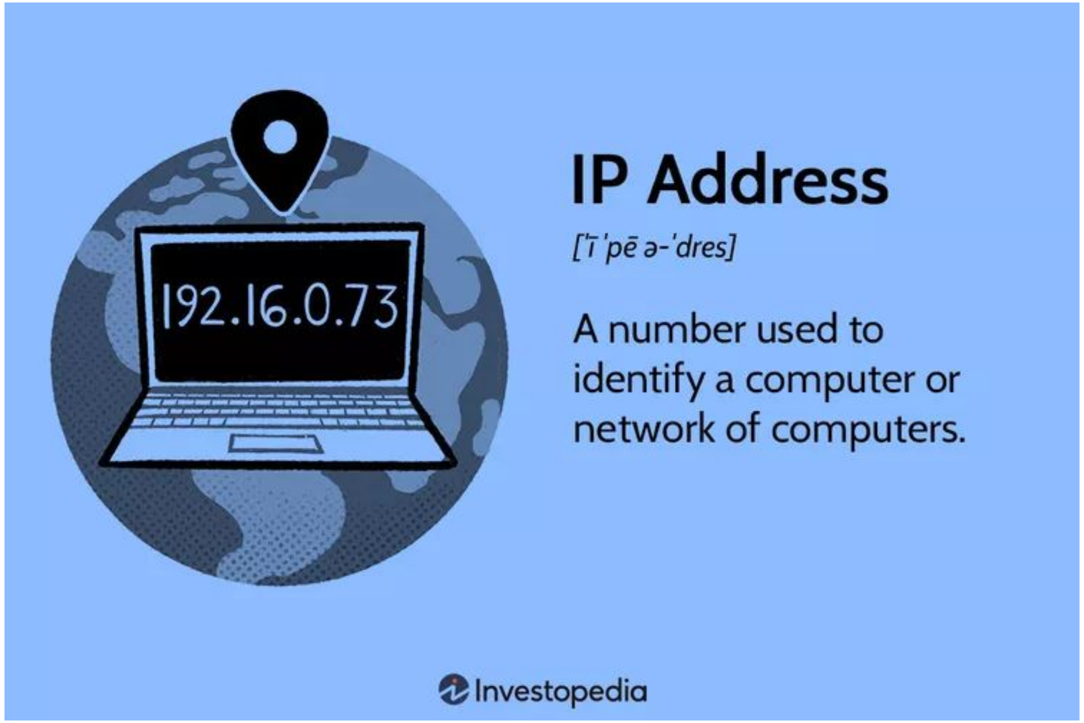
The Structure of the Internet

- The internet is made up of interconnected networks of computers.
- Internet Service Providers (ISPs) provide users with access to the internet.
- Routers route network traffic between different networks.
- Backbones are high-speed networks that carry data traffic around the world.



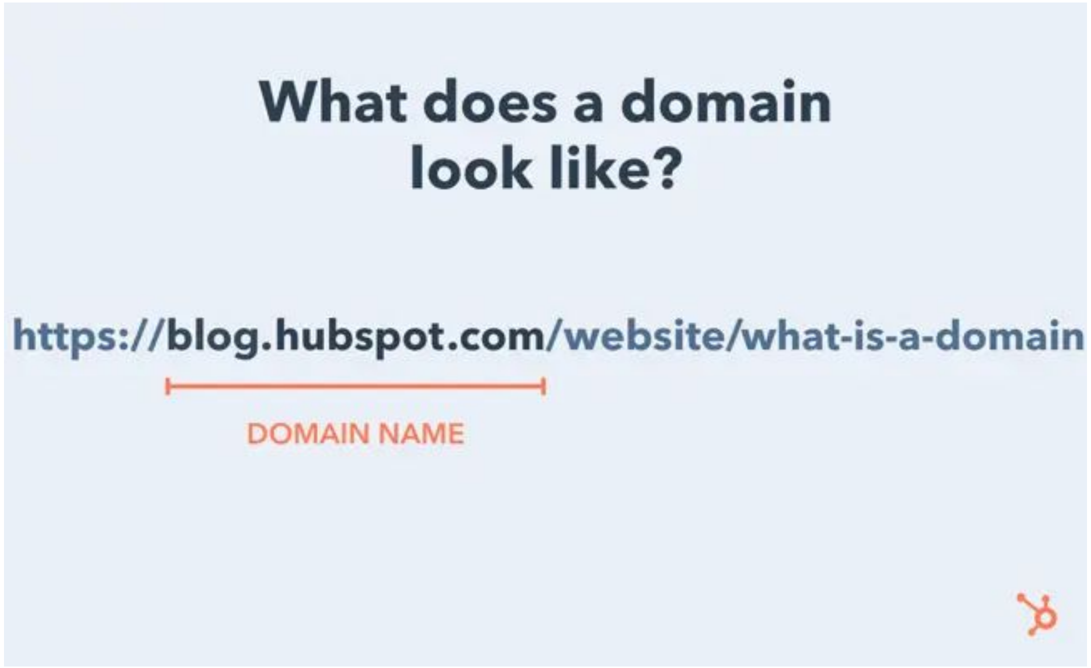
IP

- IP stands for Internet Protocol.
- IP is a unique numerical address assigned to every device connected to the internet.
- IP addresses are used to route network traffic between devices.


192.16.0.73
IP Address
[ˈī ˈpē ə-ˈdres]
A number used to identify a computer or network of computers.
Investopedia

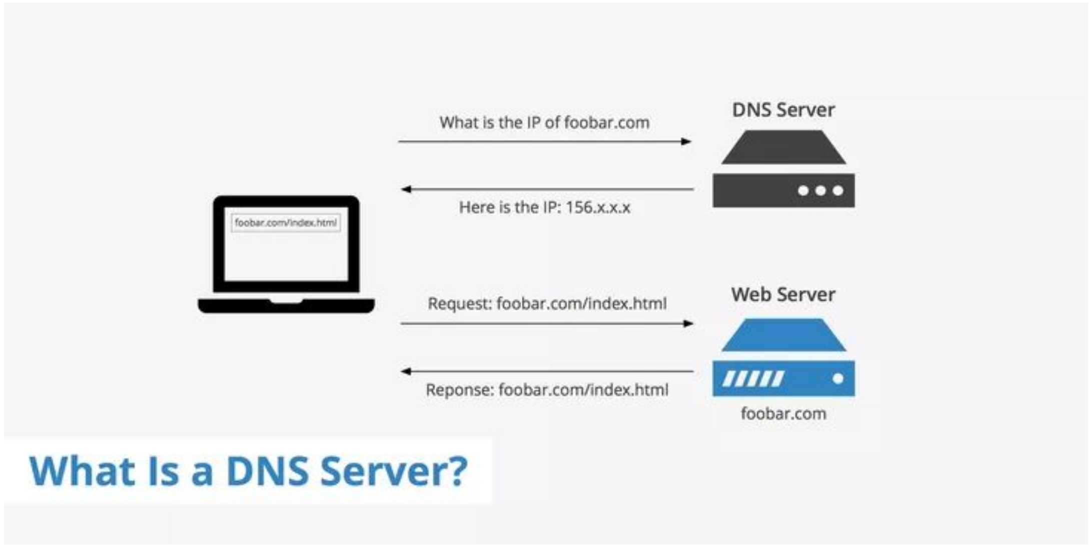
Domain Names

- A domain name is a human-readable address that points to an IP address.
- Domain names make it easier to remember and type IP addresses.
- DNS (Domain Name System) translates domain names to IP addresses.


What does a domain look like?
https://blog.hubspot.com/website/what-is-a-domain
DOMAIN NAME

DNS

- DNS stands for Domain Name System.
- DNS translates domain names to IP addresses.
- DNS servers are like phone directories for the internet.

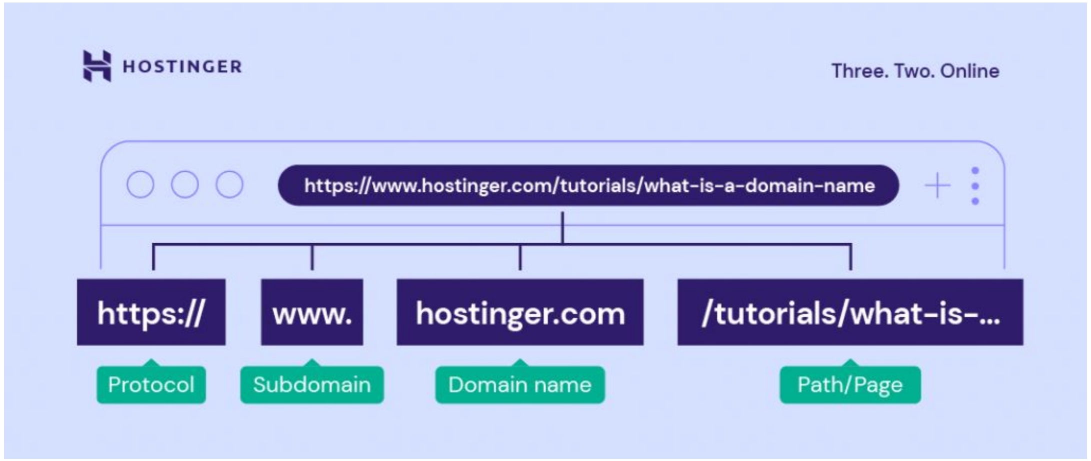

foobar.com/index.html
What is the IP of foobar.com
Here is the IP: 156.x.x.x
DNS Server
Request: foobar.com/index.html
Reponse: foobar.com/index.html
Web Server
foobar.com
What Is a DNS Server?

Hypertext Transfer Protocol (HTTP)

- HTTP is the foundation of data communication for the World Wide Web.
- It defines rules for exchanging messages between web servers and browsers.
- HTTP is a stateless, application-layer protocol.

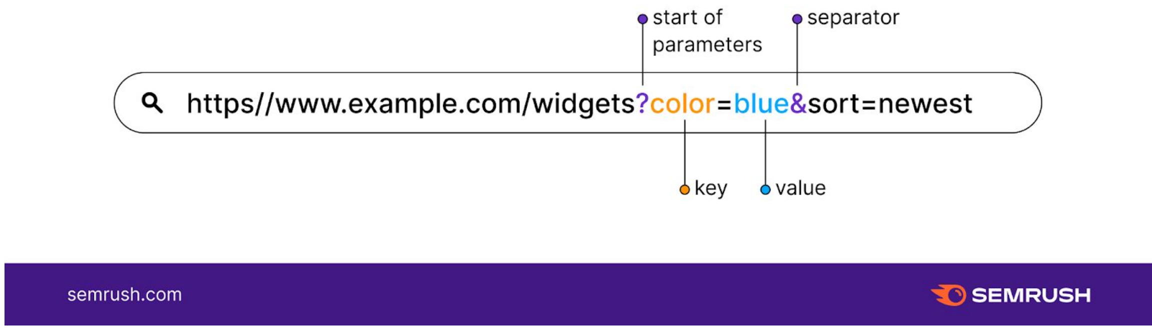
URL

- A URL (Uniform Resource Locator) is an address that identifies a resource on the internet.
- URLs are used to specify the location of web pages, images, videos, and other resources.
- URLs consist of several parts, including the protocol, hostname, port number, path, and query string.



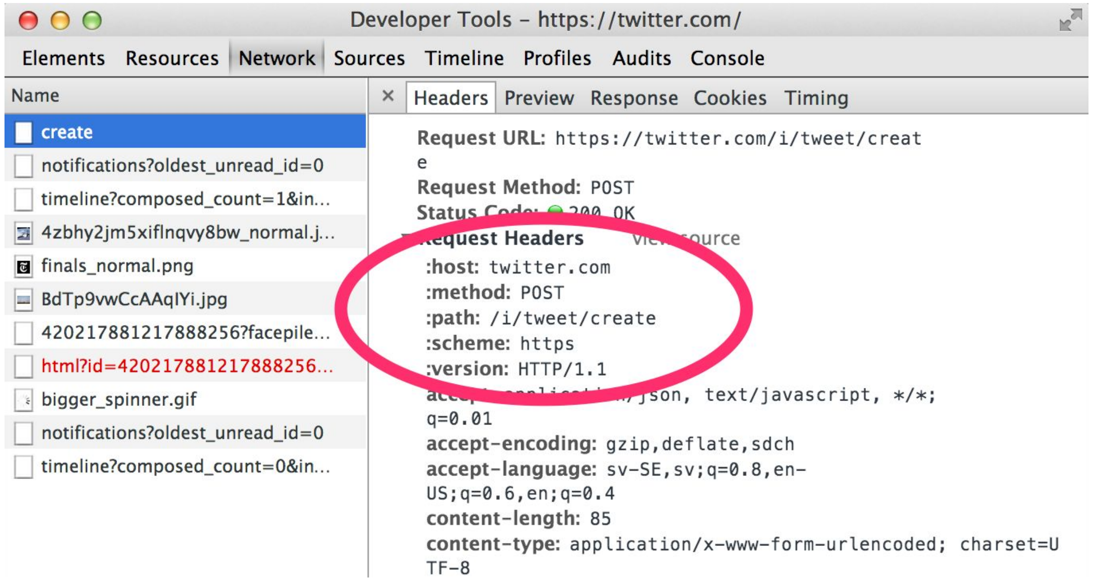
Query String

- A query string is a part of a URL that contains additional information about the requested resource.
- Query strings are often used to pass data to web servers.
- Query strings consist of name-value pairs, which are separated by ampersands (&).



Header

- An HTTP header is a group of lines of text that contain information about an HTTP request or response.
- Common HTTP headers include the Host, Content-Type, and Accept headers.
- HTTP headers are used to control the behavior of web servers and clients.

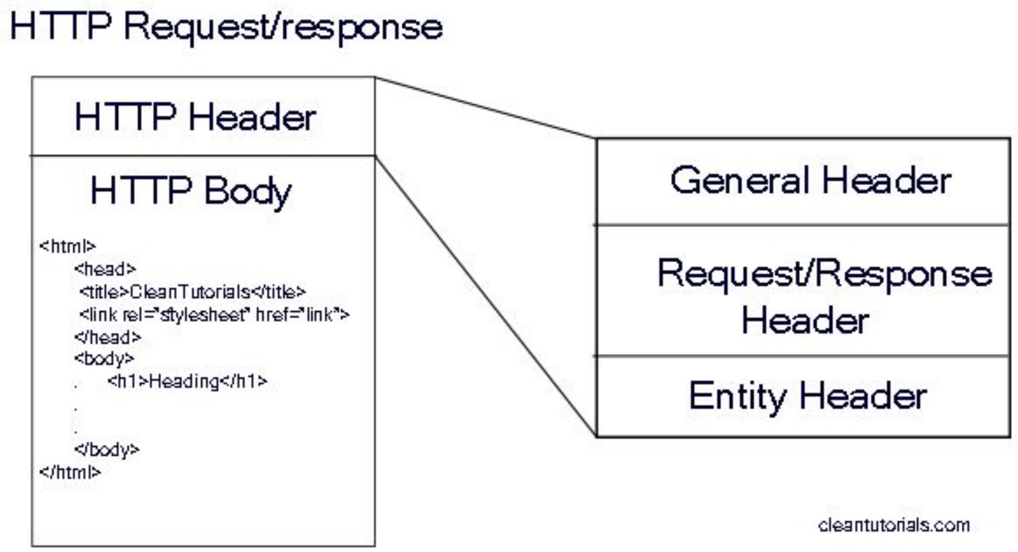


create
notifications?oldest_unread_id=0
timeline?composed_count=1&in...
4zbhy2jm5xiflnqvy8bw_normal.j...
finals_normal.png
BdTp9vwCcAAqIYi.jpg
420217881217888256?facepile...
html?id=420217881217888256...
bigger_spinner.gif
notifications?oldest_unread_id=0
timeline?composed_count=0&in...

```
Request URL: https://twitter.com/i/tweet/creat
e
Request Method: POST
Status Code: 200 OK
Request Headers    view source
  :host: twitter.com
  :method: POST
  :path: /i/tweet/create
  :scheme: https
  :version: HTTP/1.1
  accept: application/json, text/javascript, */*;
  q=0.01
  accept-encoding: gzip,deflate,sdch
  accept-language: sv-SE,sv;q=0.8,en-
  US;q=0.6,en;q=0.4
  content-length: 85
  content-type: application/x-www-form-urlencoded; charset=U
  TF-8
```

Body

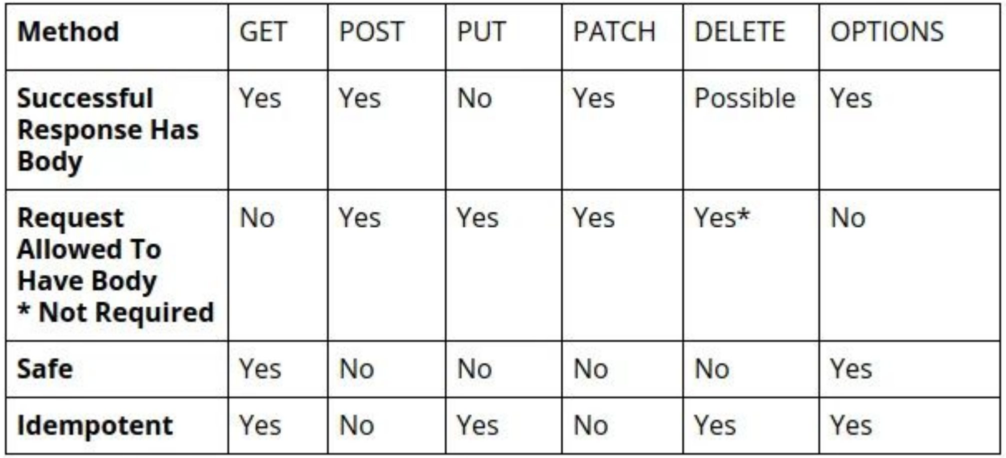
- The HTTP body is the data that is sent in an HTTP request or response.
- The body is typically larger than the header and contains the actual content of the request or response.
- The body can be any type of data, such as HTML, text, images, or videos.



```
<html>
    <head>
     <title>CleanTutorials</title>
     <link rel="stylesheet" href="link">
    </head>
    <body>
        <h1>Heading</h1>
    .
    .
    .
    </body>
</html>
```

HTTP Methods

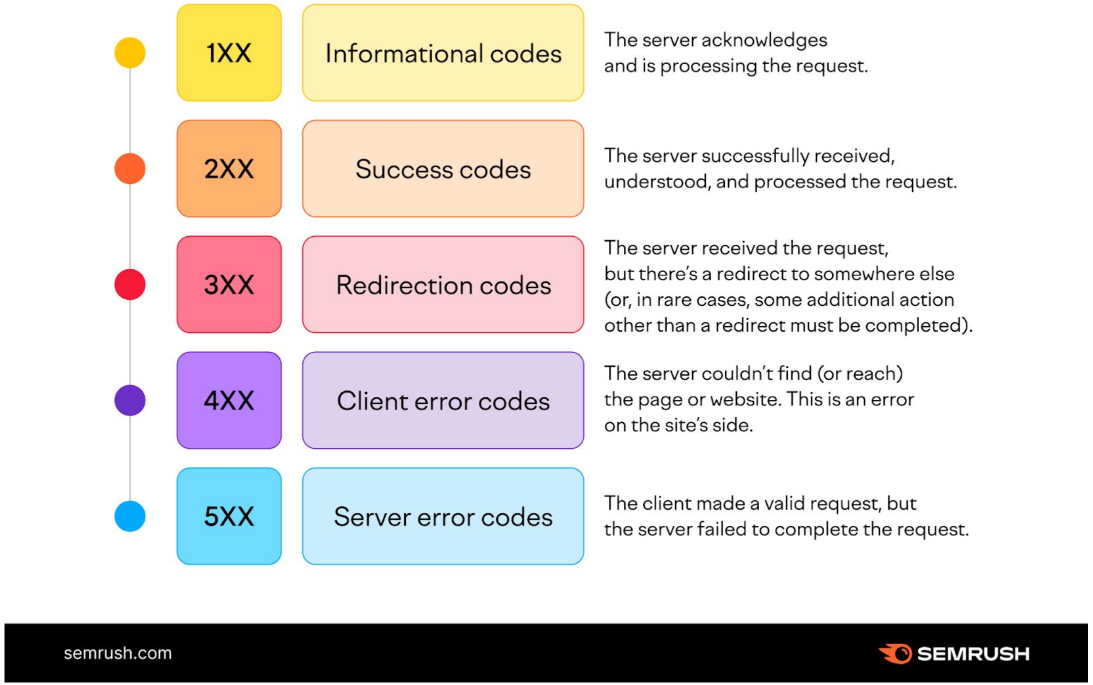
- There are many other HTTP methods besides the ones we have already discussed.
- Some common HTTP methods include PUT, DELETE, HEAD, and OPTIONS.
- Each HTTP method has a specific meaning and is used for a specific purpose.



Method	GET	POST	PUT	PATCH	DELETE	OPTIONS
Successful Response Has Body	Yes	Yes	No	Yes	Possible	Yes
Request Allowed To Have Body * Not Required	No	Yes	Yes	Yes	Yes*	No
Safe	Yes	No	No	No	No	Yes
Idempotent	Yes	No	Yes	No	Yes	Yes

Status Codes

- HTTP status codes are used to indicate the outcome of an HTTP request.
- Common HTTP status codes include 200 OK, 404 Not Found, and 500 Internal Server Error.
- Each HTTP status code has a specific meaning and is used to indicate a specific condition.



1XX	Informational codes	The server acknowledges and is processing the request.
2XX	Success codes	The server successfully received, understood, and processed the request.
3XX	Redirection codes	The server received the request, but there's a redirect to somewhere else (or, in rare cases, some additional action other than a redirect must be completed).
4XX	Client error codes	The server couldn't find (or reach) the page or website. This is an error on the site's side.
5XX	Server error codes	The client made a valid request, but the server failed to complete the request.

What is Backend Development?

- The powerhouse behind the scenes
- Handles data storage, logic, and server-side operations
- Ensures secure and reliable data management
- Lays the foundation for seamless user interactions

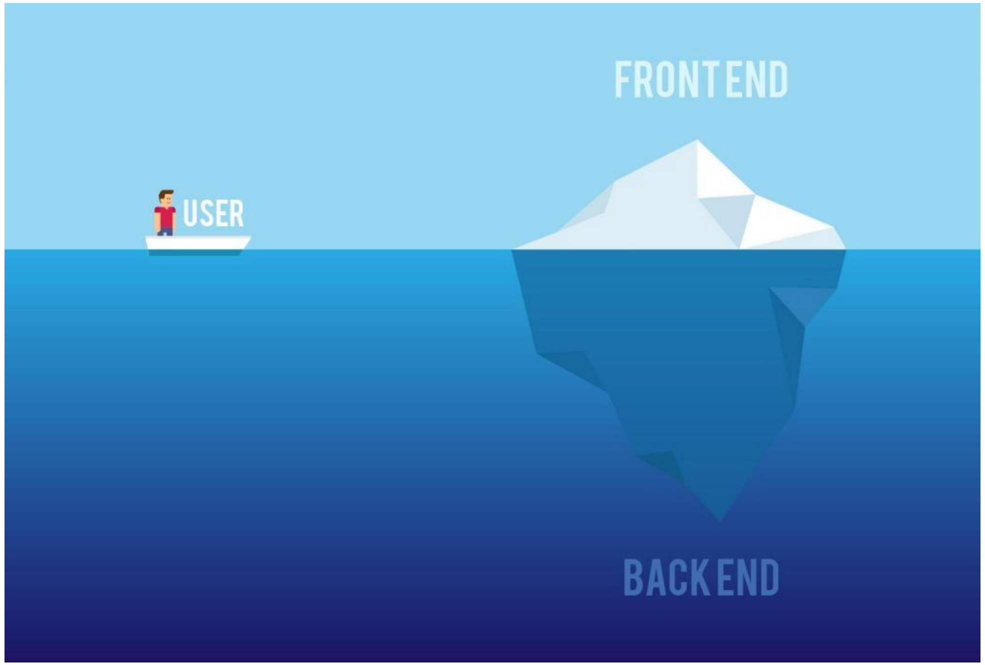
What is Backend Development?

Backend:

- Server-side programming
- Data storage and management
- Application logic and functionality
- Security and authentication

Frontend:

- Client-side programming
- User interface (UI) and user experience (UX)
- Visual elements and interactions
- Responsiveness and accessibility
- JavaScript frameworks and libraries



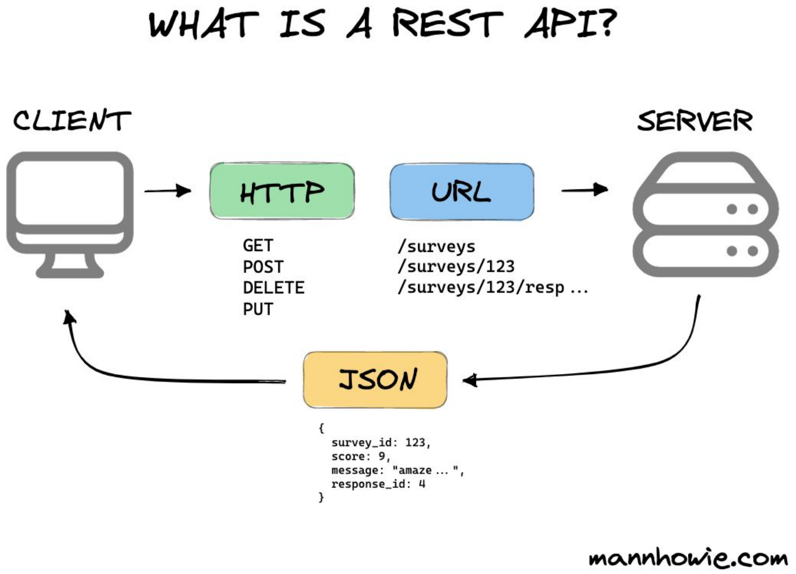
APIs: The Bridges Between Backend and Frontend

API (Application Programming Interface):

- A messenger between backend and frontend
- Defines how applications can communicate and exchange data
- Acts as a standardized interface for accessing backend services

API Types:

- **REST APIs:** Follow a structured format using HTTP methods
- **SOAP APIs:** Use XML-based messaging for structured data exchange



```
{
  survey_id: 123,
  score: 9,
  message: "amaze ... ",
  response_id: 4
}
```

Software Engineering vs. Software Development

- Be the decision-maker and designer. Solve problems.
- Design and implement software for future generations.
- Software engineering is essentially the development of software over time.

Source: Software Engineering at Google

Django

- https://www.djangoproject.com/
- `django-admin startproject <project-name>`
- Project structure
- `python manage.py startapp <app_name>`
- `python manage.py runserver`

Django Models

- A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you’re storing. Generally, each model maps to a single database table.

The basics:

- Each model is a Python class that subclasses `django.db.models.Model`.
- Each attribute of the model represents a database field.

Write

- Model fields
- Migrations
- `.save, .create`
- `id` field
- Foreign key, many to many relation
- `.update, .delete,`
- `.values, .values_list`
- index

Read

- `.all`
- `.filter`
- `.exclude`
- `.get`
- `.limit`
- `.order_by`
- `.count`
- `Lt, lte, gt, gte`
- `(i)exact`
- `(i)contains`
- `Join`
- Querysets are lazy

ACID

Atomicity: The All-or-Nothing Guarantee

- **Atomicity** dictates that a transaction must be treated as a single, indivisible unit.
- Either all operations within the transaction succeed, or all fail.
- No partial completion or intermediate states are allowed.

Consistency: Maintaining Data Integrity

- **Consistency** guarantees that a transaction leaves the database in a valid and consistent state.
- It adheres to the predefined data rules and constraints.
- Transactions must not violate data integrity or introduce inconsistencies.

Consistency ensures that transactions maintain the inherent rules and relationships within the data. Consider a database storing customer orders. A transaction updating an order's status should not introduce inconsistencies, such as an order being marked as both "shipped" and "pending." Consistency safeguards data integrity and prevents invalid or erroneous states from arising.

Isolation: Protecting Concurrent Transactions

- **Isolation** prevents concurrent transactions from interfering with each other's execution.
- Each transaction operates as if it is the only one accessing the database.
- Data is protected from conflicts and unintended modifications.

Isolation ensures that multiple transactions can execute simultaneously without disrupting or invalidating each other's operations. Think of a database managing multiple bank accounts. Isolation prevents withdrawals from one account from affecting the balance of another account simultaneously. This protection prevents data conflicts and ensures the integrity of individual transactions.

Durability: Guaranteeing Data Persistence

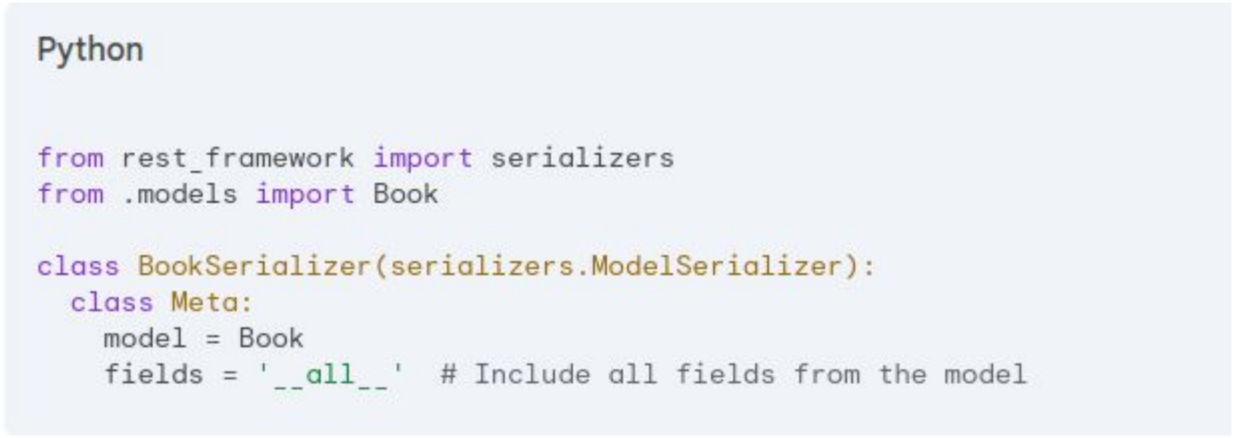
- **Durability** ensures that once a transaction is committed, its changes are permanent and persistent.
- Data is protected from failures, such as power outages or system crashes.
- Committed transactions are not rolled back or lost.

Durability safeguards data by guaranteeing that once a transaction is successfully completed and committed, its changes become permanent and persistent. Even in the event of system failures or power outages, the committed data remains intact. This ensures that transactions are not lost or rolled back, preserving the integrity of the database.

Django Rest Framework

Model Serializer:

- serializing and deserializing the model instances into representations such as json
- Serializable object like json, dict, … <-> Model object


Python

```python
from rest_framework import serializers
from .models import Book

class BookSerializer(serializers.ModelSerializer):
  class Meta:
    model = Book
    fields = '__all__'  # Include all fields from the model
```

Django Rest Framework

```
Model Serializer:

  -   def __init__(self, instance=None, data=empty, **kwargs):

          super().__init__(instance, data, **kwargs)

  -   .data: instance -> data
  -   .create / .update: data -> instance
        -   is_valid
        -   create / update
  -   Many = True
  -   Method fields
  -   To representation
```

Django Rest Framework

View:

- url <-> view <-> serializer <-> model
- Method views: @api_view
- response: rest_framework.response.Response
- path
- include

Django Rest Framework

```
Class based views:

  -   APIView
  -   ListAPIView
  -   CreateAPIView
  -   RetrieveAPIView
  -   DestroyAPIView
  -   UpdateAPIView
```

Principles of API Design