This work is licensed under CC BY 4.0

Web Applications and Services

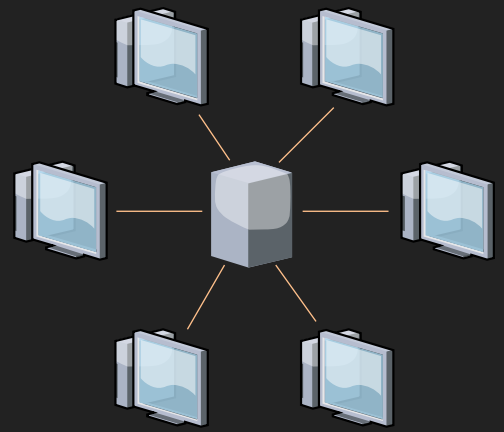
An orientation about web applications and web services
and how they relate to each other.

Some Application Types

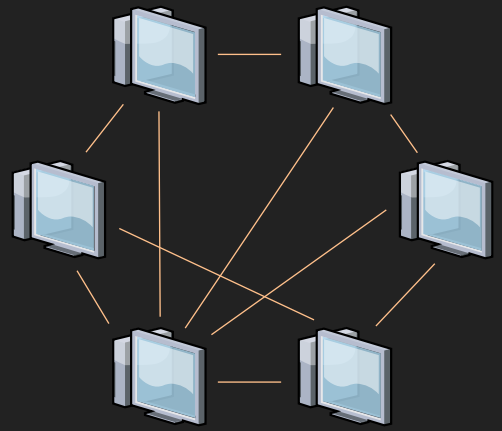
- “Traditional” applications
  - Runs on a single system.



- Client-server applications
  - Is a distributed system made up of both client and server software.
  - Clients and servers communicate by sending messages over a network.
  - When there are only two types of hosts, clients and servers, this type of architecture is sometimes called two-tier architecture.
  - The client makes service requests to the server, the server can accept these requests, process them, and return the requested information to the client.



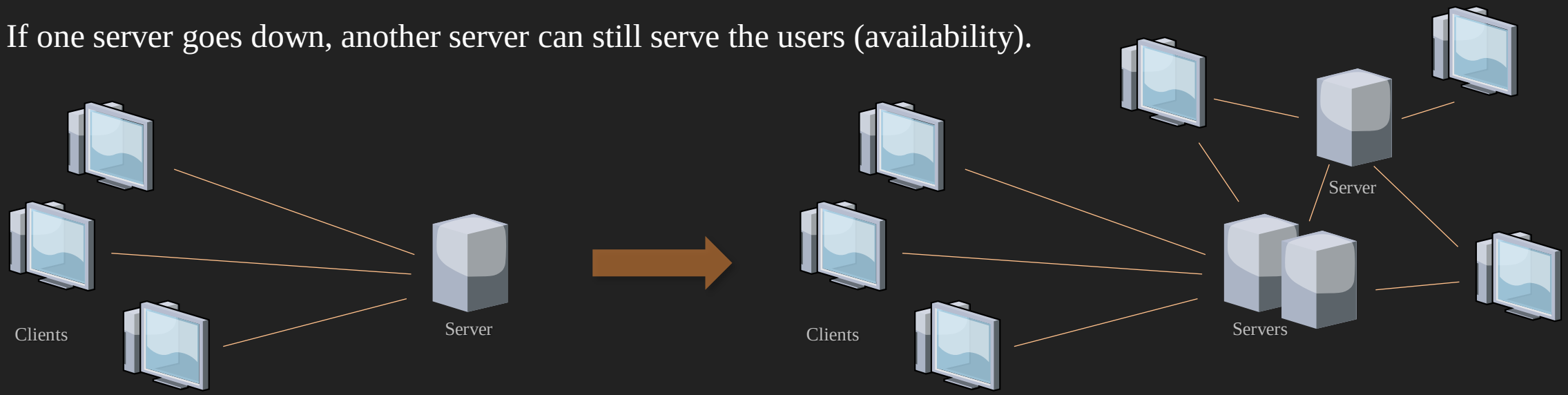
- Peer-to-peer applications (P2P)
  - Each host can simultaneously act as both a client and a server.
    - BitTorrent, communication protocol for P2P file sharing.
    - Bitcoin network, P2P payment network.



Distributed Systems

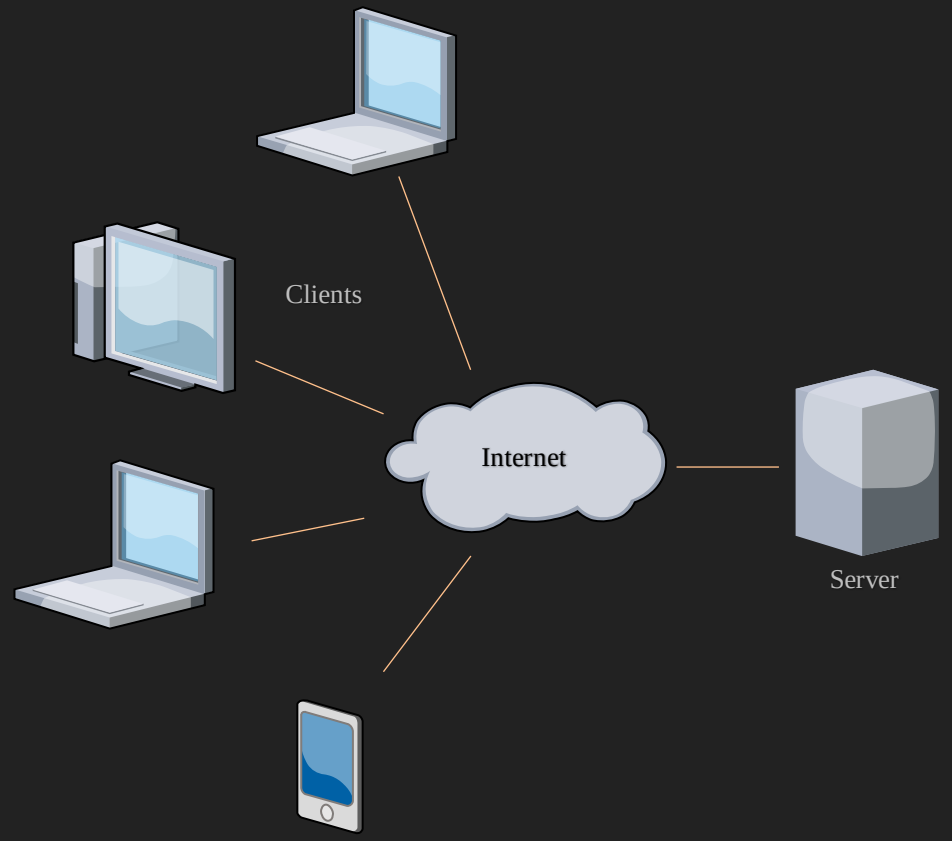
- A distributed system is a system whose components are located on different computers linked by a network which communicate by passing messages to one another.
  - One of many message passing mechanisms is HTTP.
- The goal of a distributed system is to enable scalability, performance and availability.
  - Machines can be added whenever it is needed (horizontal scaling).
  - Locate servers close to the users (reduce time to serve users).
  - If one server goes down, another server can still serve the users (availability).

There are disadvantages such as increased complexity.



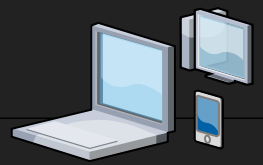
What is the Client-Server Architecture?

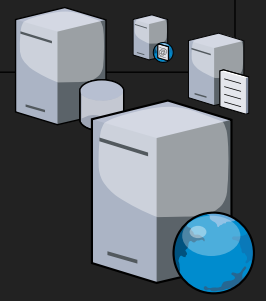
- Nowadays, both clients and servers are considered intelligent, which was not the case in the old mainframe model, where a centralized mainframe computer performed everything on behalf of “dumb” terminals.
- Many clients request and receive service from a centralized server.
- Clients provide an interface to allow a user to request services from the server and view the results that the server returns.
- The server provides a standardized interface to clients, so they don’t need to know how the service is provided under the hood.



Difference Between Client and Server

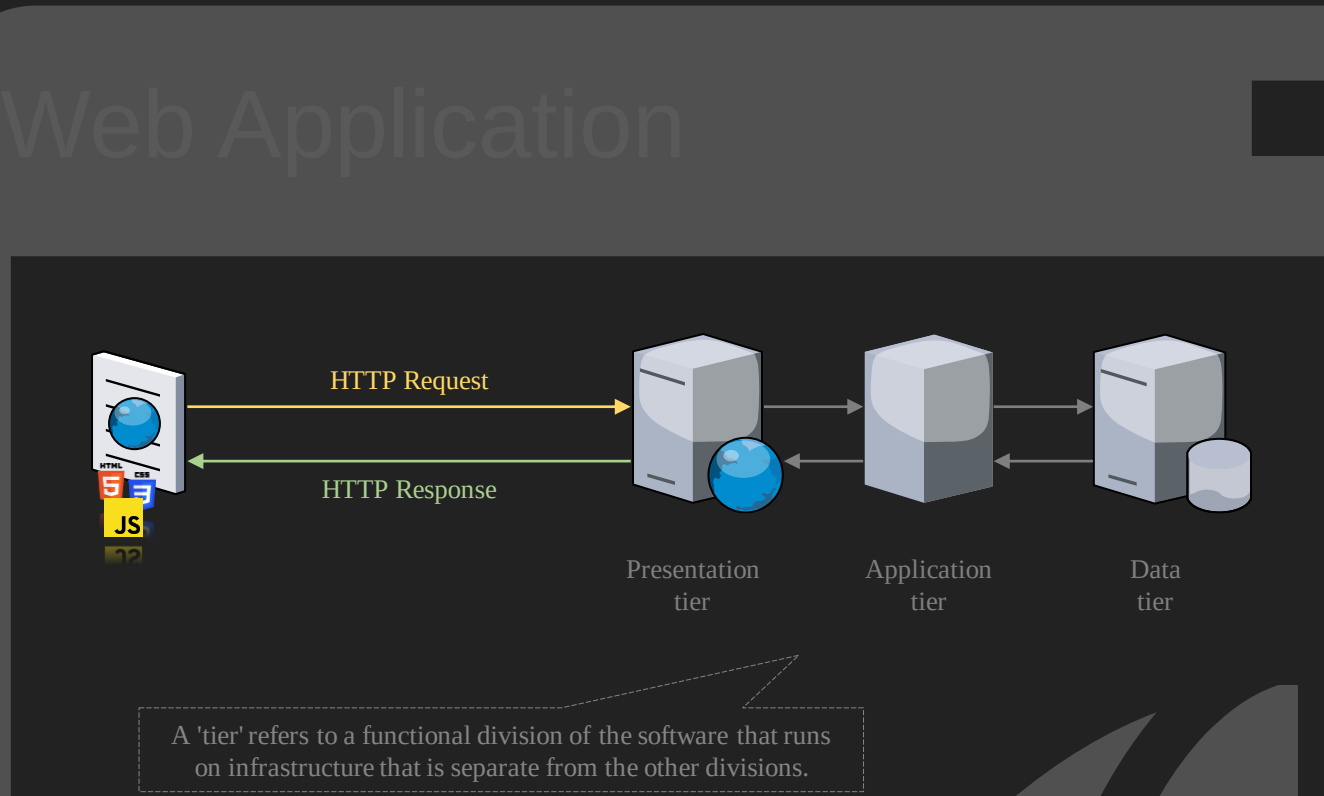
Client	Server
✓ A client is a device or a program that requires services via a network.	✓ A server is a device or a program that responds to the requests of the clients.
✓ Requests the server for content or service function.	✓ Provides functions or services to the clients when clients request so.
✓ Ex: desktops, laptops, smartphones, tablets and web browsers	✓ Ex: mail servers, file servers, database servers, and web servers
✓ **Client-side programming:** Write code that creates a user interface, sends requests to the server and handles responses from the server.	✓ **Server-side programming:** Write code that queries and interacts with databases, accesses files on a server, interacts with other servers, processes user input, and structures web applications.





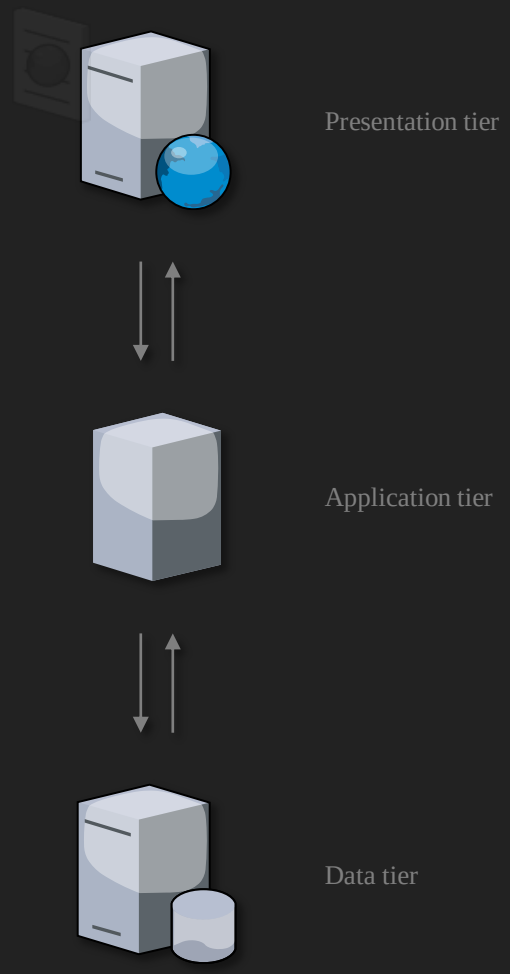
What is a Web Application?

- A web application…
  - …is a client-side and server-side software application in which the client uses a web browser to request the web server over the internet.
  - …use client-side code (HTML, CSS, and JavaScript) to present information to users, and a combination of server-side code (JavaScript, PHP, Ruby, C#, etc.) to handle the storage and retrieval of information.
- The client-side code is downloaded from the server as a result of a client request.
- A common structure of a web application is the three-tier architecture.
  - 1-tier: Web server (presentation tier)
  - 2-tier: Application server (application tier)
  - 3-tier: Database server (data tier)



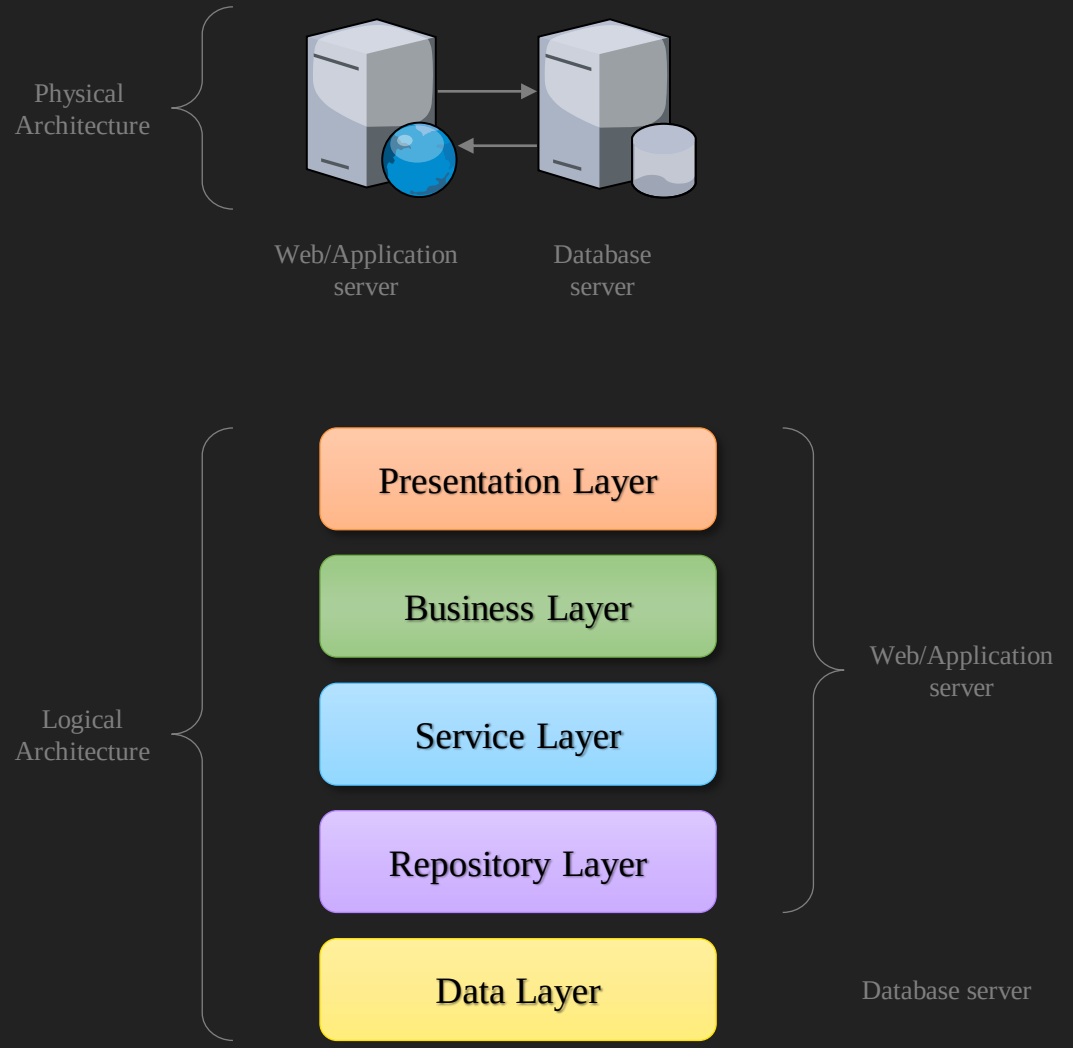
Three-tier Application in Web Development

- ✓ The **web server** is the presentation tier and provides the user interface.
  - The content can be static or dynamic and is usually developed using HTML, CSS, and JavaScript.
- ✓ The **application server** housing the business logic used to process user input.
  - This is the tier that queries the database and returns data to be presented. The logic can be developed using *JavaScript*, Python, Ruby, PHP, or C# and runs a framework such as *Express*, Django, Rails, Symphony, or ASP.NET, for example.
- ✓ The **database server** stores the web application's persistent data.
  - It runs on database management software (DBMS), such as *MongoDB*, MySQL, PostgreSQL, MS SQL Server, and Oracle Database.



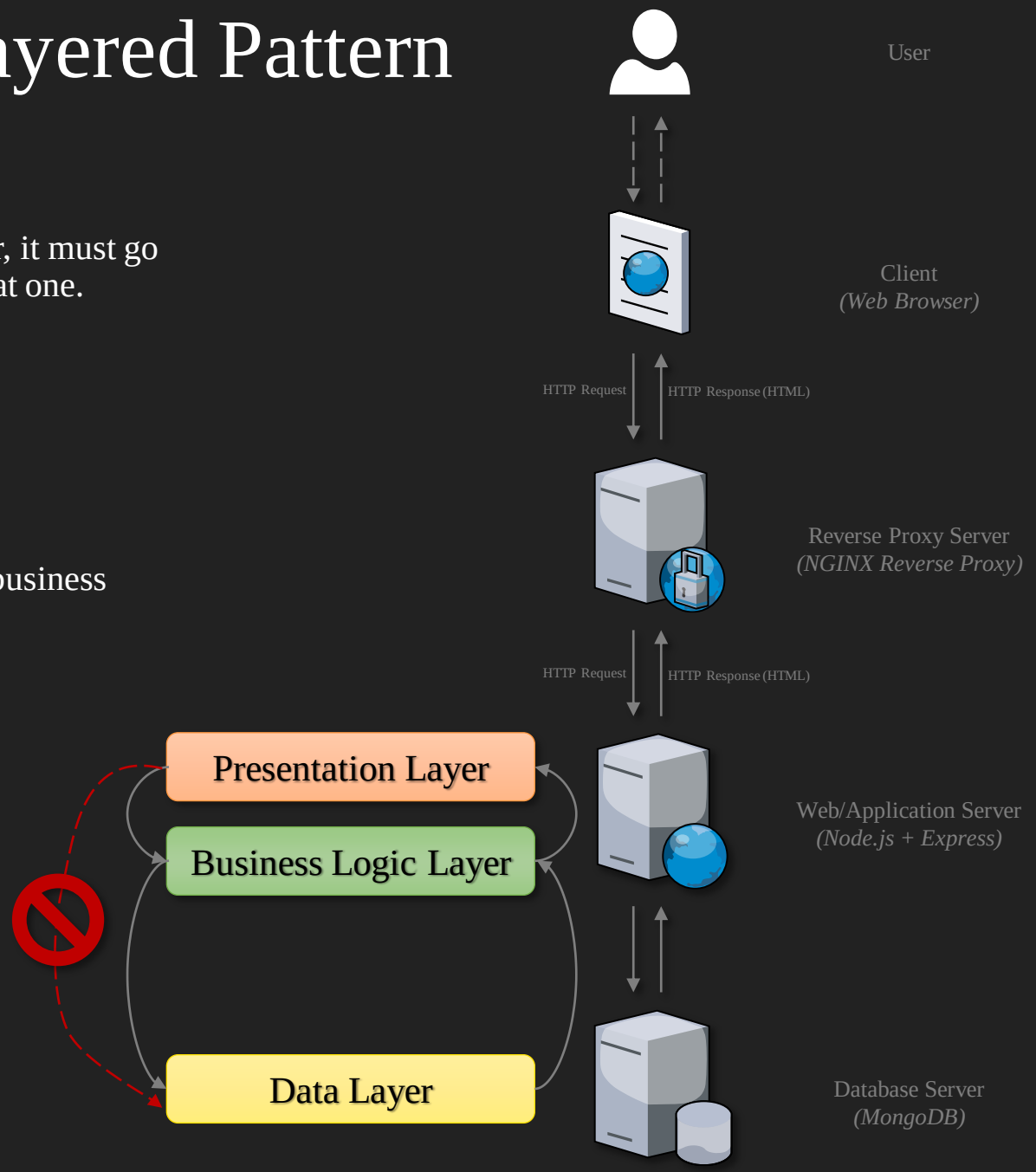
Layered Architecture

- ✓ Good software design says that you should separate concerns.
  - Don’t mix up UI concerns with data access concerns, and separate infrastructure from business logic.
- ✓ …BUT ”separate” means “logically separate” and not “physically separate”.
- ✓ Don’t confuse logical and physical architecture!
  - A tier refers to a physical unit upon which application deployment is done.
  - A layer refers to a logical separation of an application’s code.



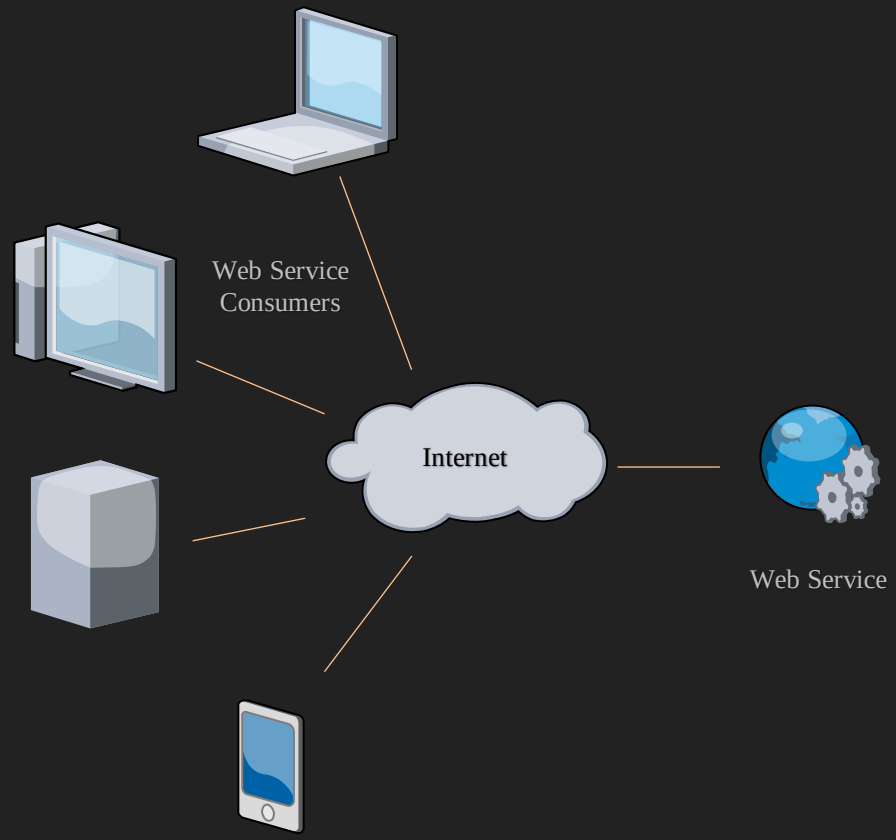
Web Architecture Pattern - Layered Pattern

- **Assume that all layers are closed.**
  - A closed layer means that as a request moves from layer to layer, it must go through the layer right below it to get to the next layer below that one.
- **Presentation Layer**
  - Parses HTTP requests.
  - Validates inputs.
  - Passes information (not any request or response objects) to the business layer below it.
  - Translates results to HTTP status code and HTML.
- **Business Layer**
  - Contains the core of the application logic.
  - Performs all required validations and calculations.
  - Managing data access for the presentation layer.
- **Data Layer**
  - Stores persistent data in a document database.



What is Web Service?

- Web services are responsible for online machine-to-machine communication.
- Web services makes it possible to share application functionality regardless of the source code language, operating system, or hardware used to create that functionality.
- Web services use the HTTP protocol to pass XML or JSON based messages.
- The most common way for a web service to communicate is to use either SOAP, or REST, or both.
  - SOAP is a protocol depending on XML.
  - Representational State Transfer (REST) is an architectural style, which uses exciting HTTP actions and methods to send JSON.



What are RESTful Web Services?

- ✓ RESTful web service is a lightweight, maintainable, and scalable that is built on the REST architecture.
  - The underlying protocol is HTTP and is run on a web server.
  - Expose RESTful API from an application in a secure, uniform, stateless manner to the calling client.
    - Use Uniform Resource Identifiers (URIs) to address resources.



- ✓ Key elements of a RESTful web service:
  - Identification of resources
    - Can be a collection (“cars”) or a singleton (“car”).
      - `https://onest-harrys-cars.com/api/cars` — Identifies a collection of “cars”.
      - `https://onest-harrys-cars.com/api/cars/42` — Identifies a single “car”.
  - Manipulation of resources through representation
    - An identified resource can be returned in formats such as HTML, XML, SVG, JSON, PNG, etc. These formats are representations of the identified resource. Clients indicate through the `Accept` HTTP header their preferred representation.
    - Clients indicate the preferred representation when manipulating an resource using the `Content-Type` HTTP header.
  - Self-descriptive messages
    - Each message contain all information needed to complete the task. The message is said to be stateless.
    - Clients send messages of various types using HTTP methods (`POST`, `GET`, `PUT`, `PATCH`, `DELETE`).

Using HTTP Methods for RESTful Services

- ✓ Used HTTP verbs (or methods) are POST, GET, PUT, PATCH, and DELETE.
  - These correspond to create, read, update, and delete (or CRUD) operations, respectively.
- ✓ Returned status code depends on the combination of HTTP method and resource URI.

HTTP Verb	CRUD	Entire Collection (e.g. /cars)	Specific Item (e.g. /cars/{id})
`POST`	Create	201 (Created), 'Location' header with link to `/cars/{id}` containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists..
`GET`	Read	200 (OK), list of cars. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single car. 404 (Not Found), if ID not found or invalid.
`PUT`	Update/Replace	405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
`PATCH`	Update/Modify	405 (Method Not Allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
`DELETE`	Delete	405 (Method Not Allowed), unless you want to delete the whole collection - not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

Do We Need to Use Another Architecture?

- ✓ The short answer is no!
- ✓ As long as we create a RESTful Web Service, nothing has to be changed.
  - It is the URI that determines if the client is requesting a web service resource or not.
  - The presentation layer still parses the HTTP request, and depending on the requested resource, sends a response containing HTML, JSON, etc., to the client.

