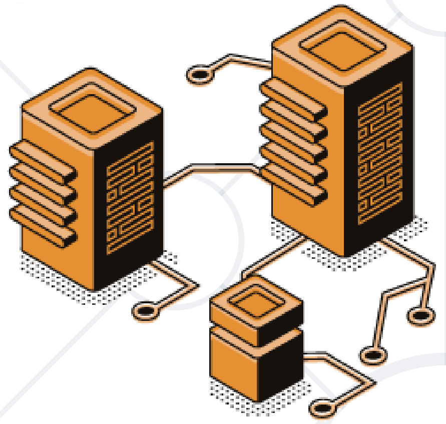
Software Architectures and Containers

Front-End, Back-End, APIs, Microservices

Virtualization, Containers, Docker, Cloud



SoftUni Team

Technical Trainers

Software University

https://about.softuni.bg

Have a Question?

sli.do

#qa-fund

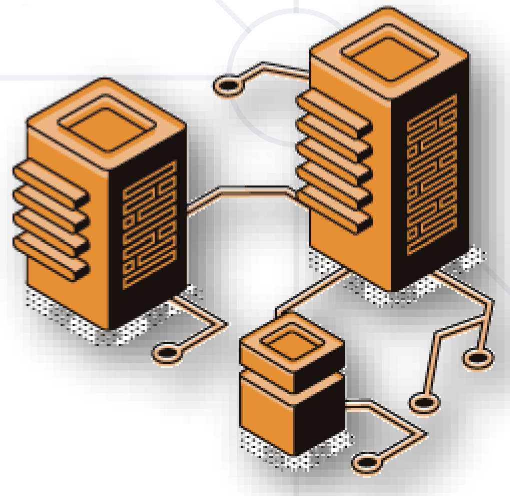
Table of Contents

Software Architectures

Monolith, Client-Server, 3-Tier, Microservices

Software Architectures

- Software systems consist of **interconnected components** organized in certain structure called an **architecture**
- Concepts related to **software architectures**:
  - Monolith apps
  - Client-server model
  - Front-end and back-end
  - 3-tier and multi-tier architecture
  - SOA and microservices

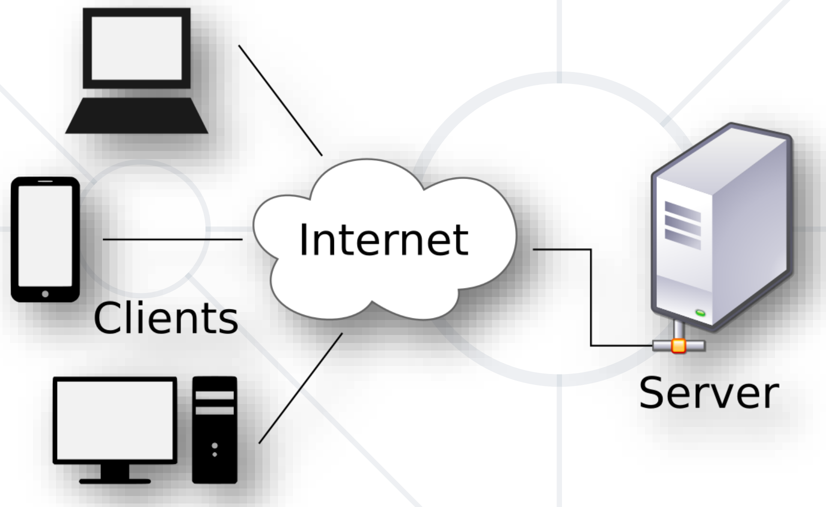


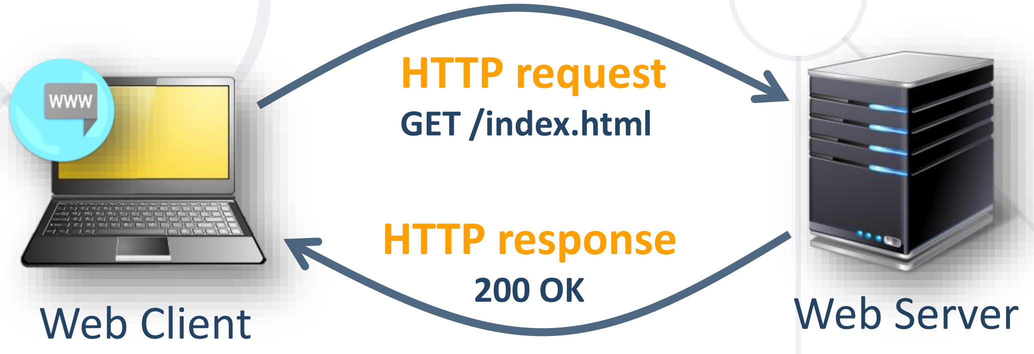
Monolith Apps

- **Monolith** apps
  - A **single application** holds its data, logic and user interface (UI)
  - **Single user** (no shared data access)
  - **Disconnected** from the Internet
  - App data is stored on the **local machine**
  - Examples
    - A simple smartphone **game**
    - The **Notepad** text editor

The "Client-Server" Model

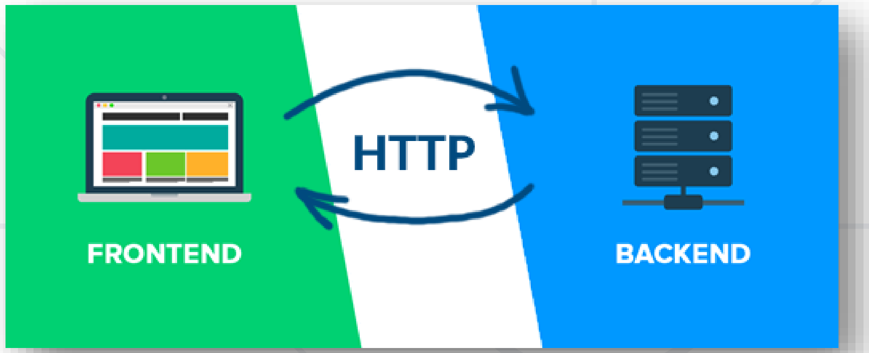
- The **client-server** architectural model
  - The **server** holds app data and logic and provides APIs to clients
  - The **clients** implement the UI (the **user interface**) and consume the server APIs
- Examples:
  - Web browser ⟷ Web site
  - Email client ⟷ Email server
  - Chat client ⟷ Chat server





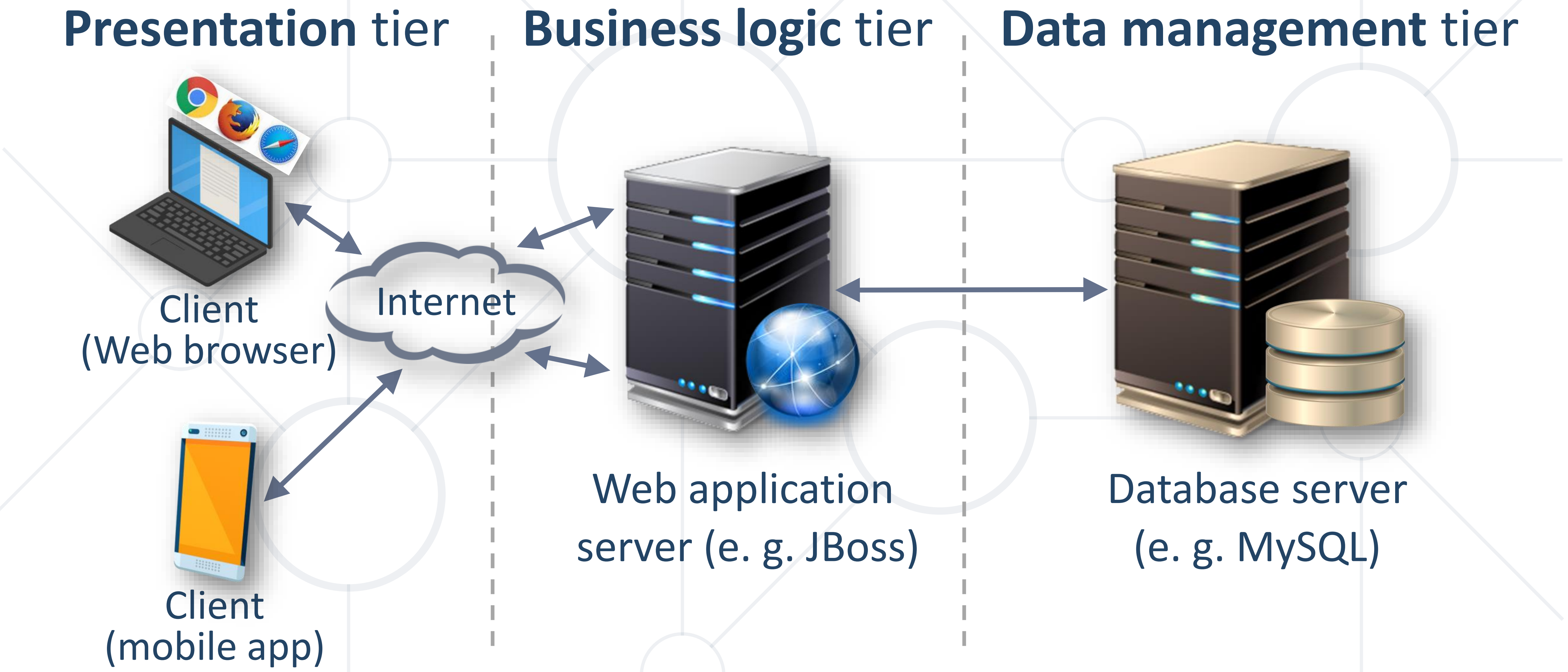
Front-End and Back-End

- **Front-end** and **back-end** separate the modern apps into **client-side** (UI) and **server-side** (data) components
- **Front-end** == client-side components (presentation layer)
  - Implement the **user interface** (UI)
- **Back-end** == server-side components (data and business logic APIs)
  - Implements **data storage and processing**

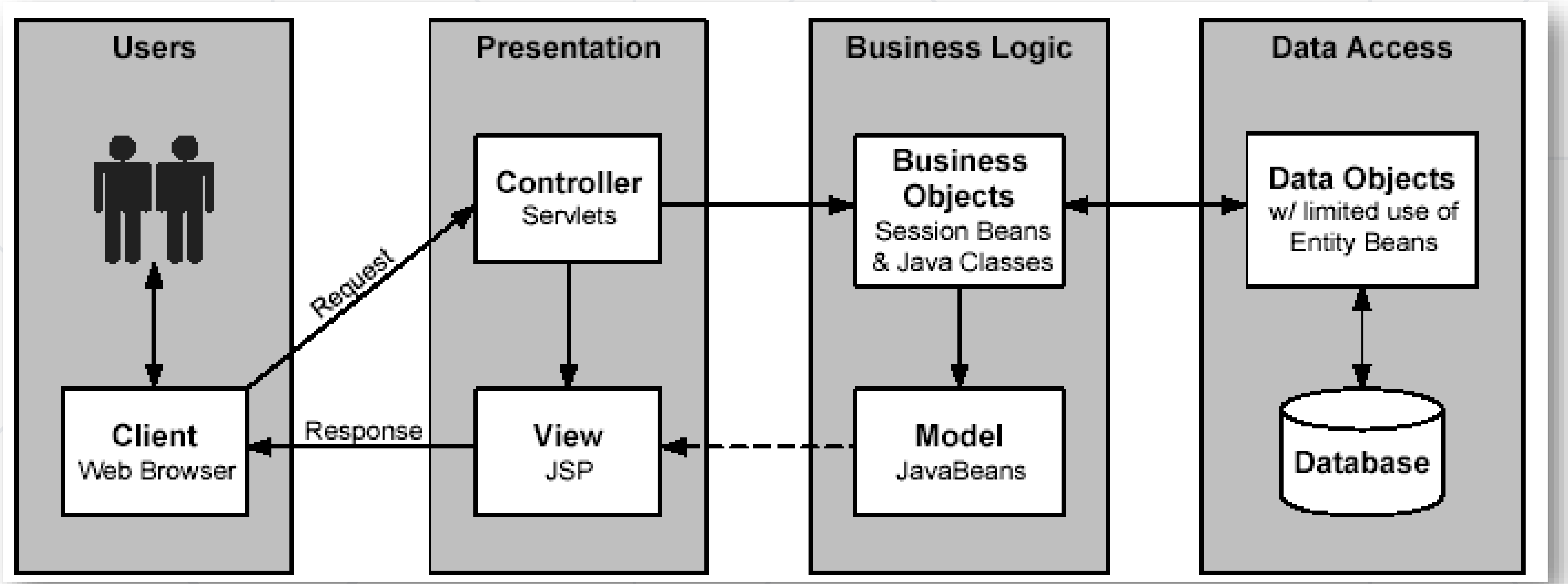


- **HTTP** connects front-end with back-end

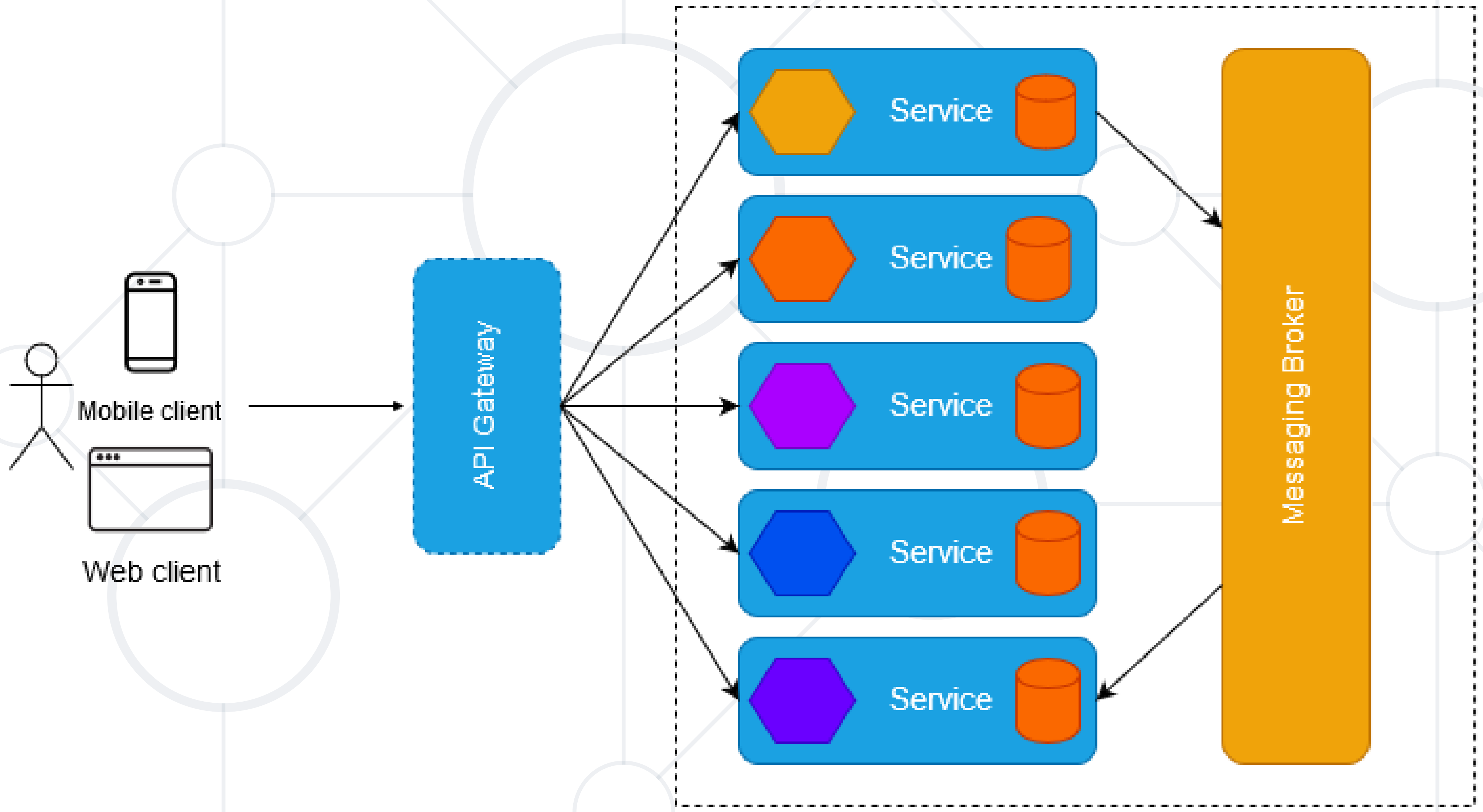
3-Tier Architecture / Multi Tier Architecture



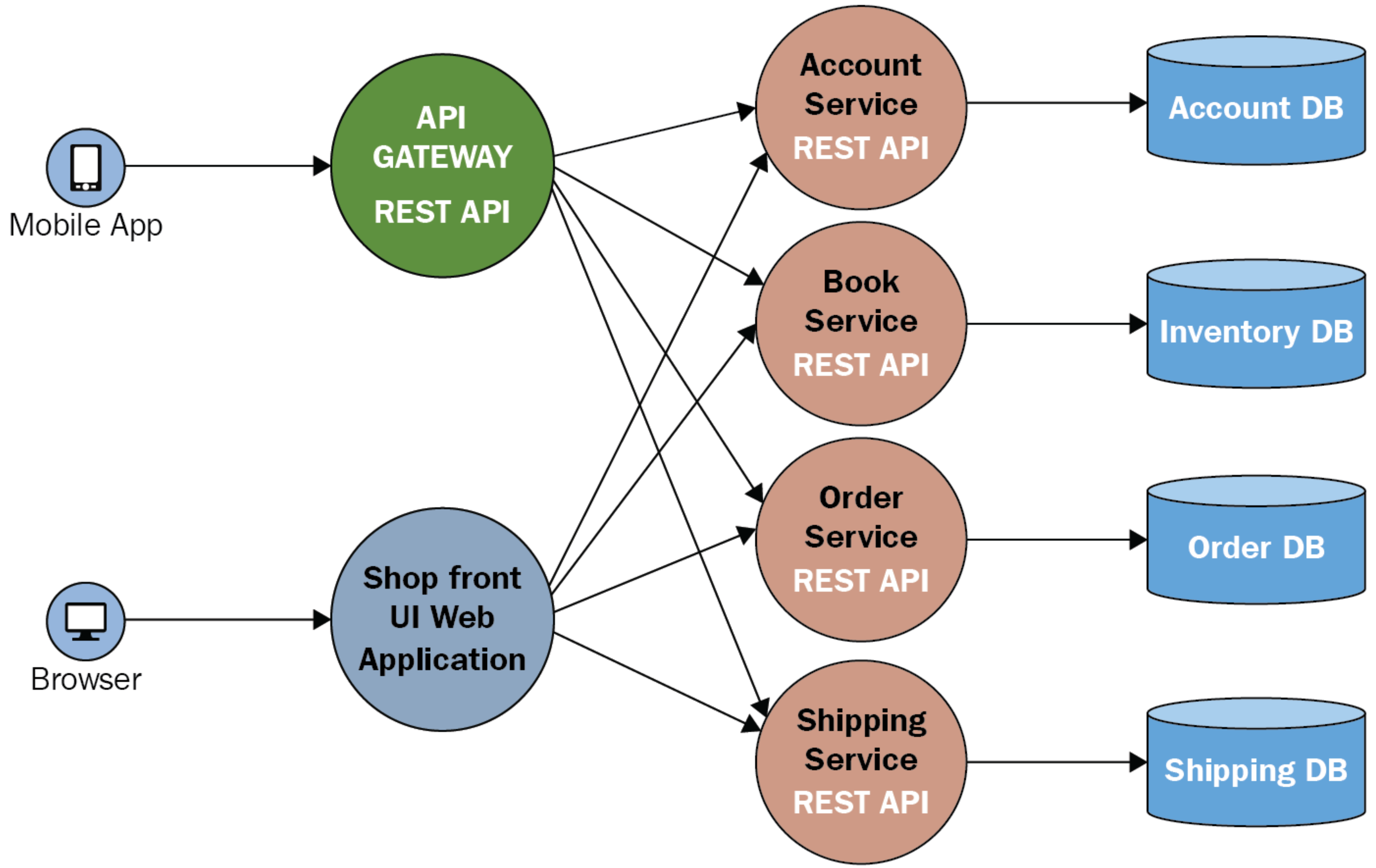
Multi-Tier Architecture – Example



Microservice Architecture



Microservice Architecture – Example



Front-End Concepts

HTML + CSS + JavaScript + JS Libraries

Front-End Technologies

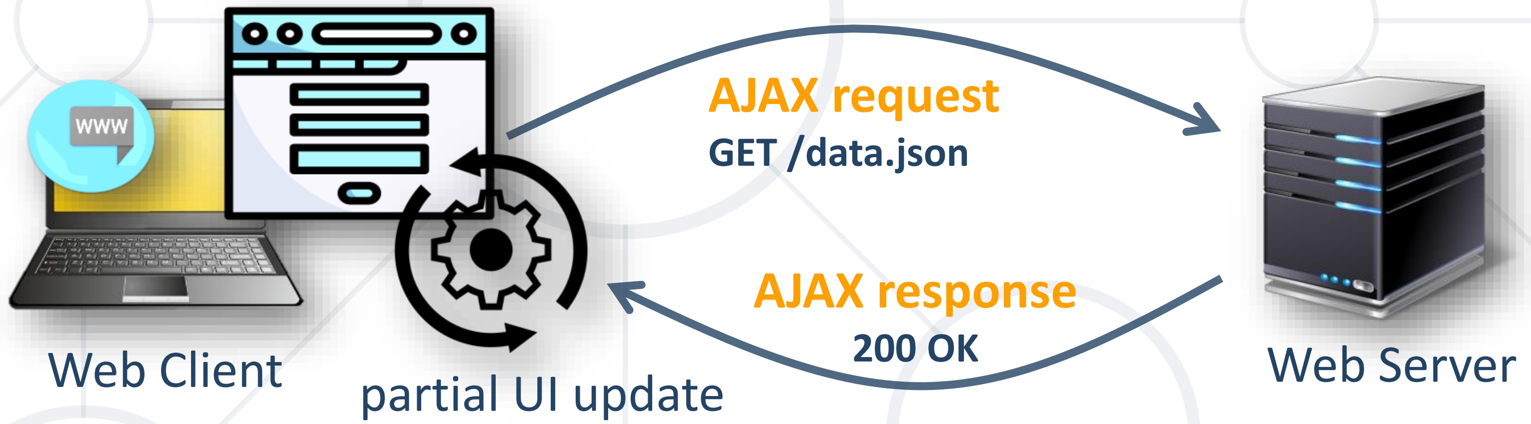
- Front-end **technologies**
  - **Web front-end**: HTML + CSS + JavaScript + JS libraries
  - **Web front-end frameworks**: React, Angular, Vue, Flutter
  - **Desktop front-end**: XAML (Microsoft), UIKit (Apple)
  - **Mobile front-end**: Android UI, SwiftUI
  - **Hybrid mobile front-end**: React Native, Ionic
- **Front-end developers** deal with UI, UX and front-end technologies and frameworks

Web Front-End and DOM

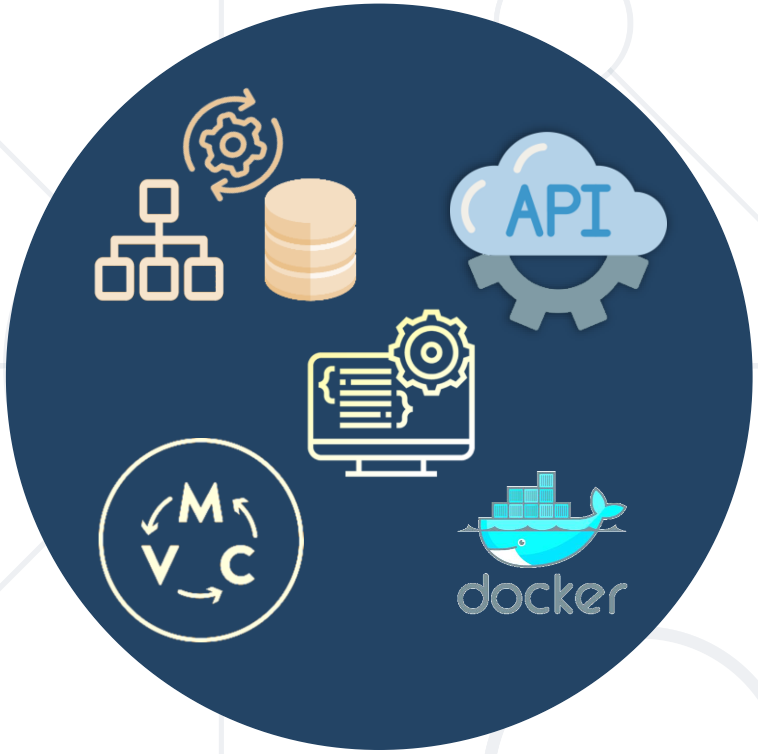
- **Web front-end technologies** (see https://platform.html5.org)
  - HTML, CSS, JavaScript, DOM, AJAX
  - JS front-end frameworks (e.g. React, Angular, Vue)
- **DOM** (the Document Object Model)
  - **DOM** == a tree of UI and other elements
  - Documents in the Web browserare represented by a **DOM tree**
  - The **DOM API** allows changing the DOM from JS
  - DOM Interaction https://repl.it/@nakov/summator-js-dom

AJAX and RESTful APIs

- **AJAX** is a technology for **asynchronous** execution of **HTTP requests** from client-side JavaScript with **dynamic UI updates**



- **RESTful APIs** are HTTP-based Web services
  - The HTTP methods **GET**, **POST**, **PUT** and **DELETE** retrieve, create, modify and delete data



Back-End

Concepts and Technologies

Back-End Technologies

- **Back-end technologies** are about server-side programming
  - **Data management** technologies and **ORM frameworks**
  - Backend **Web frameworks** and **MVC** frameworks
  - **REST API** frameworks, **reactive** APIs, other services and APIs
  - **Microservices**, **containers** and **cloud**
- **Back-end developers** work on the server-side
  - They deal with the business logic, data processing, data storage, cloud services, APIs

Back-End Languages and Platforms

- Back-end **technologies**: server-side frameworks and libraries
  - **C# / .NET back-end**: ASP.NET MVC, Web API, Entity Framework, …
  - **Java back-end**: Java EE, Spring MVC, Spring Data, Hibernate, …
  - **JavaScript back-end**: Node.js, Express.js / Meteor, MongoDB, …
  - **Python back-end**: Django / Flask, Django ORM / SQLAlchemy, …
  - **PHP back-end**: Apache, Laravel / Symfony, …

Databases

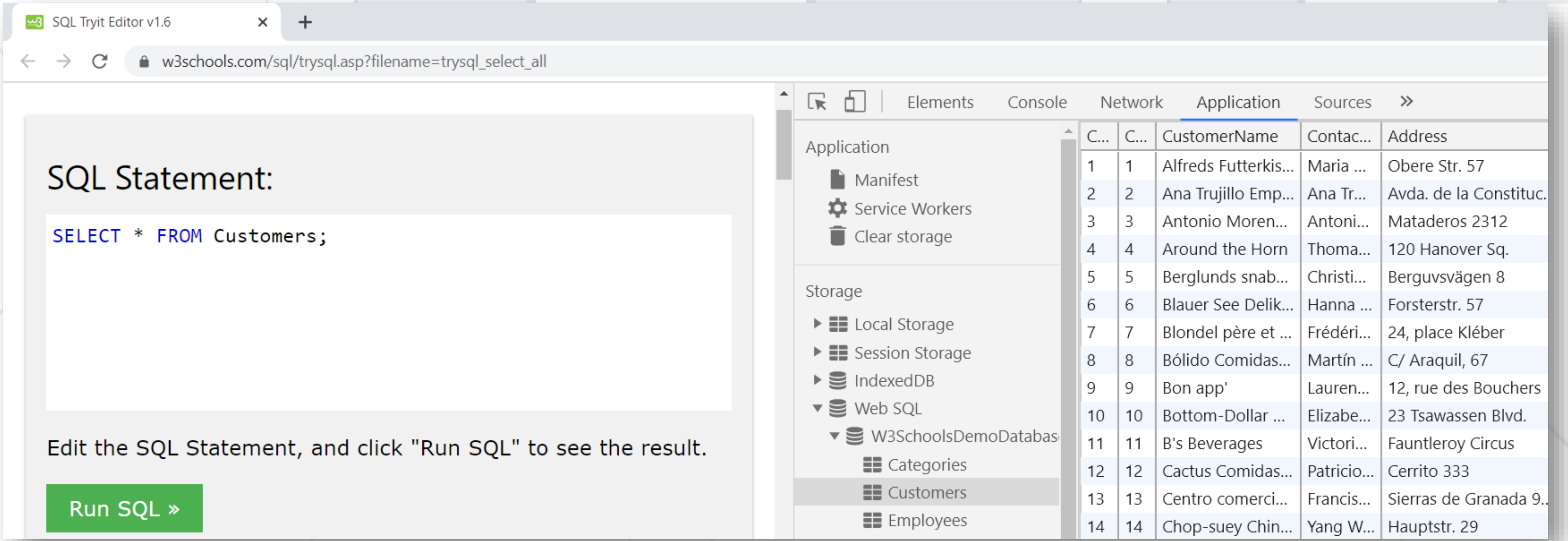
Relational Databases, SQL, NoSQL

Databases

- **Databases** hold and manage data in the back-end systems
- **Relational databases** (**RDBMS**)
  - Hold data in **tables** + **relationships**
  - Use the **SQL** language to query / modify data
  - Examples: MySQL, PostgreSQL, Web SQL in HTML5
- **NoSQL databases**
  - Hold collections of documents or key-value pairs
  - Examples: MongoDB, IndexedDB in HTML5

Web SQL – Example

- Web SQL is a relational database, embedded the Web browsers
  - It is fully functional **RDBMS system**, runs at the **client-side**



SQL Statement:

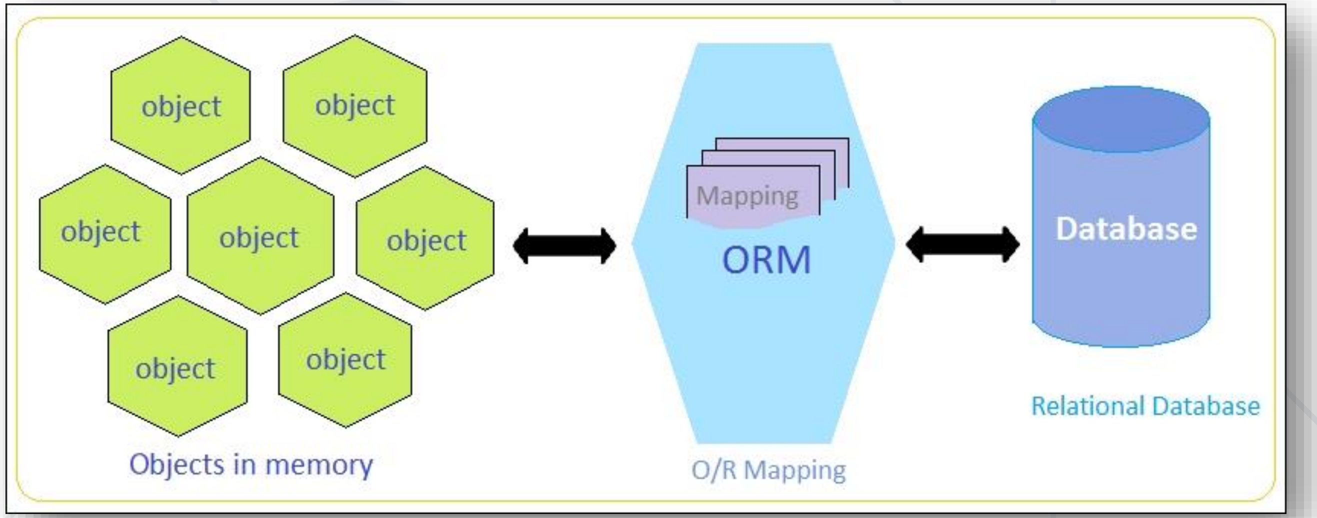
```
SELECT * FROM Customers;
```

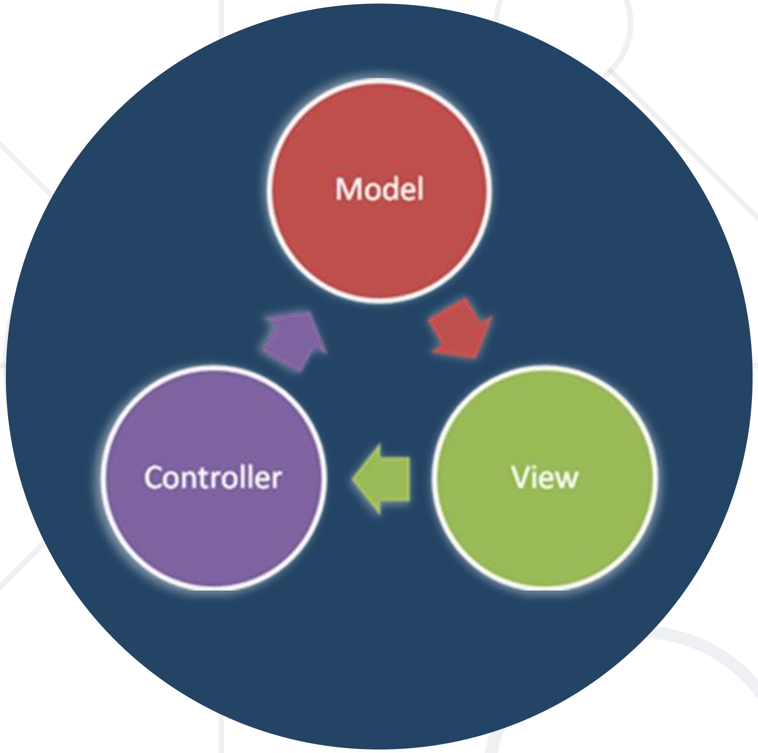
Edit the SQL Statement, and click "Run SQL" to see the result.

C...	C...	CustomerName	Contac...	Address
1	1	Alfreds Futterkis...	Maria ...	Obere Str. 57
2	2	Ana Trujillo Emp...	Ana Tr...	Avda. de la Constituc.
3	3	Antonio Moren...	Antoni...	Mataderos 2312
4	4	Around the Horn	Thoma...	120 Hanover Sq.
5	5	Berglunds snab...	Christi...	Berguvsvägen 8
6	6	Blauer See Delik...	Hanna ...	Forsterstr. 57
7	7	Blondel père et ...	Frédéri...	24, place Kléber
8	8	Bólido Comidas...	Martín ...	C/ Araquil, 67
9	9	Bon app'	Lauren...	12, rue des Bouchers
10	10	Bottom-Dollar ...	Elizabe...	23 Tsawassen Blvd.
11	11	B's Beverages	Victori...	Fauntleroy Circus
12	12	Cactus Comidas...	Patricio...	Cerrito 333
13	13	Centro comerci...	Francis...	Sierras de Granada 9..
14	14	Chop-suey Chin...	Yang W...	Hauptstr. 29

ORM Frameworks

- **ORM frameworks** (object-relational mapping) allow persisting objects in relational database (by mapping classes to tables)
  - E.g. store JS objects in MySQL database
- Popular ORM frameworks:
  - **Entity Framework** (C#)
  - **Hibernate** (Java)
  - **Sequelize** (JavaScript)
  - **SQLAlchemy** (Python)



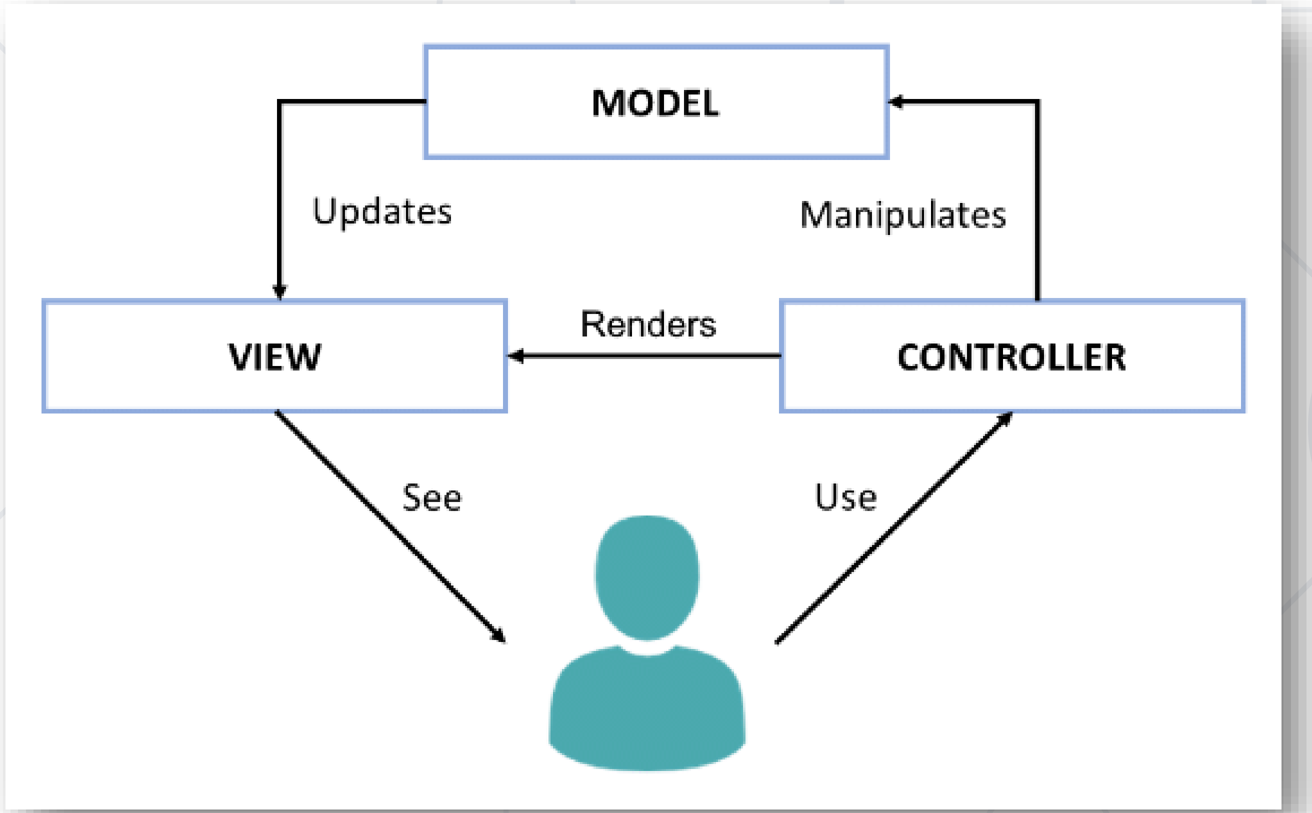


Back-End Frameworks

Model-View-Controller and MVC Frameworks

The Model-View-Controller (MVC) Pattern

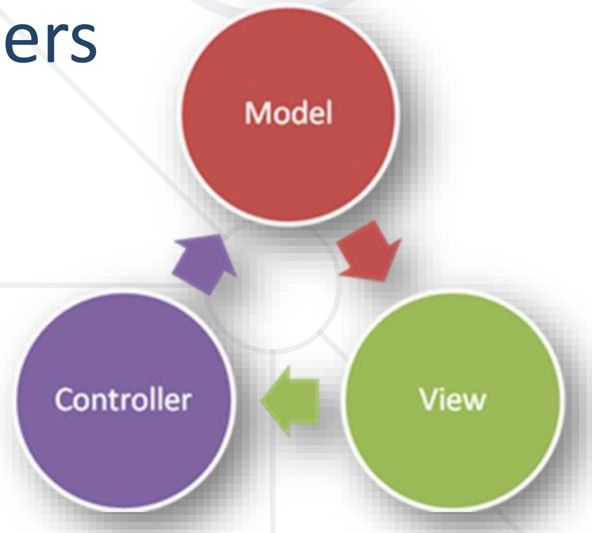
- The **Model-View-Controller (MVC)** pattern



- **Controller**
  - Handles user actions
  - Updates the model
  - Renders the view (UI)
- **Model**
  - Holds app data
- **View**
  - Displays the UI, based on the model data

Web MVC Frameworks

- **Web MVC frameworks** used to build Web applications
  - **Controllers** handle HTTP GET / POST and render a view
  - **Views** display HTML + CSS, based on the models
  - **Models** hold app data for views, prepared by controllers
- Examples of Web MVC frameworks:
  - **ASP.NET MVC** (C#), **Spring MVC** (Java), **Express** (JS), **Django** (Python), **Laravel** (PHP), **Ruby on Rails** (Ruby), **Revel** (Go), …
  - https://repl.it/@nakov/MVC-express-pug-example



Web Services

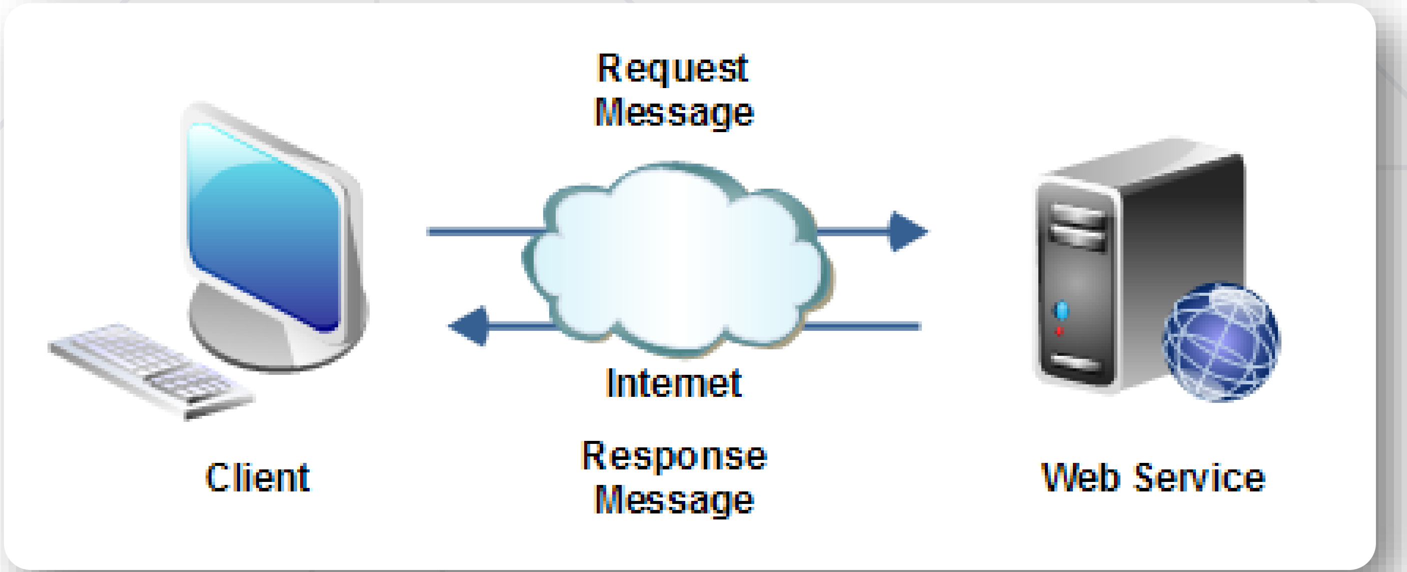
Communication between Systems and Components

What is API?

- **API** == **A**pplication **P**rogramming **I**nterface
  - Programming interface, designed for communication between system components
  - Set of **functions** and **specifications** that software programs and components follow to talk to each other
- API **examples**:
  - **JDBC** – Java API for apps to talk with database servers
  - **Windows API** – Windows apps talk with Windows OS
  - **Web Audio API** – play audio in the Web browser with JS

What is Web Service?

- **Web services** implement **communication** between software **systems** or **components** of over the **network**
  - Using standard **protocols**, such as HTTP, JSON and XML
  - Exchanging **messages**, holding data and operations



Web Services and APIs

- **Web services** expose **back-end APIs** over the **network**
  - May use different **protocols** and **data formats**: **HTTP**, **REST**, **GraphQL**, **gRPC**, **SOAP**, JSON-RPC, JSON, BSON, XML, YML, …
- **Web services** are hosted on a Web server (HTTP server)
  - Provide a set of functions, invokable from the Web (Web API)
- **RESTful APIs** is the most popular Web service standard
  - Uses **HTTP requests** (GET, POST, PUT, DELETE, …) to invoke remote functionals at the server-side
  - https://replit.com/@nakov/shorturl

Containers, Docker, Cloud

Virtualization, Cloud, Containers, Docker

Virtualization and Cloud

- **Virtualization** == running a **virtual machine** (VM) / virtual environment inside a physical hardware system
  - E.g. run Android VM or Linux inside a Windows host
  - Storage, memory, networking, desktops can also be virtual
- **Cloud** == computing resources, virtual machines, storage, platforms and software instances, available on demand
  - **IaaS** (infrastructure as a service) – virtual machines on demand
  - **PaaS** (platform as a service) – app deployment environments
  - **SaaS** (software as a service) – software instances, e.g. Office 365

Containers and Docker

- **Container image** == software, packaged with its dependencies, designed to run in a virtual environment (like Docker)
  - E.g. WordPress instance (Linux + PHP + Apache + WordPress)
  - Simplified installation, configuration and deployment
- **Docker** is the most popular containerization platform
  - Runs **containers** from local **image** or downloaded from the **Docker Hub** online repository
  - Open-source, runs on Linux, Windows, Mac

Docker – Example

- Install **Docker** on your local computer
  - Or use the Docker online playground: https://labs.play-with-docker.com (with a free Docker Hub registration)
- Download and **run a Docker image** in a new container:

```
docker run -it -p 80:80 alexwhen/docker-2048:latest
```

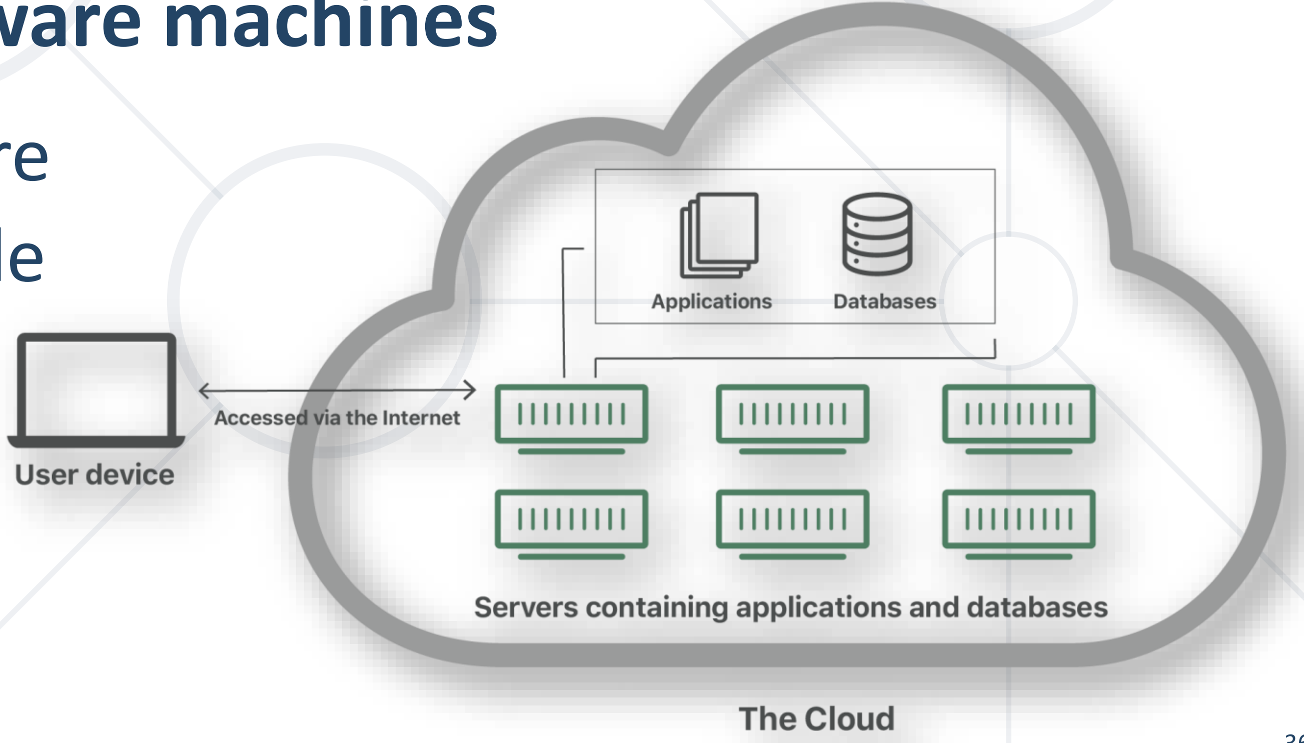
- Open the exposed URL: http://localhost:8080
- View currently running Docker containers

```
docker ps
```

Cloud

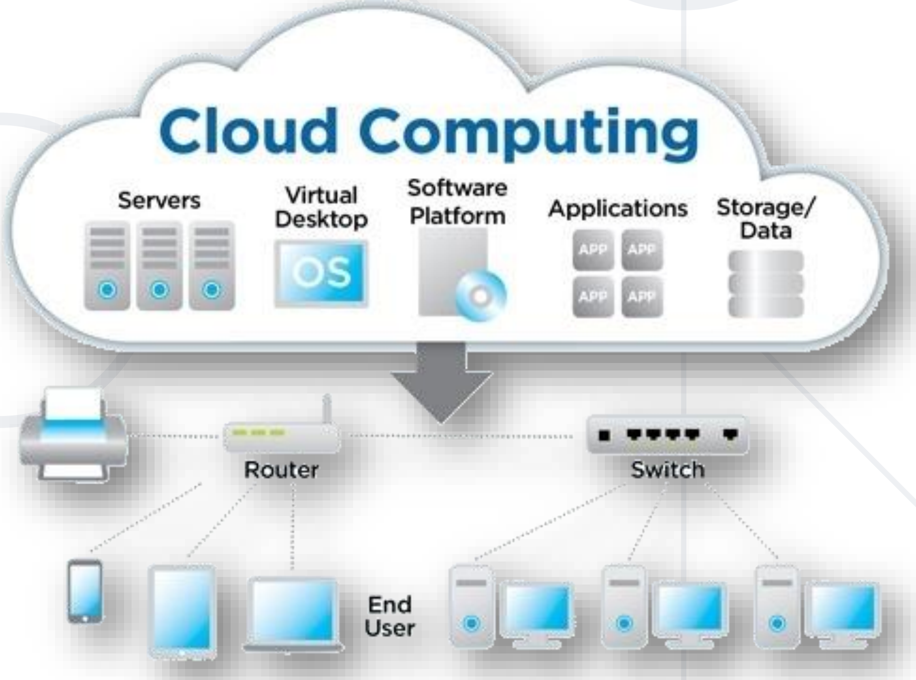
What is Cloud?

- **Cloud** is a **virtual space** (**software** and **services**) that runs on the **Internet**, instead of locally on your computer
- **Clouds** combine the computing power and resources of **multiple hardware machines**
  - **Share cloud resources** more efficiently between multiple users and apps
  - Save **costs**
  - Better **service**


Applications
Databases
Accessed via the Internet
User device
Servers containing applications and databases
The Cloud

How the Cloud Works?

- In the cloud everyone consumes a **portion** of the **shared computing resources**
  - CPU, memory, storage, IO, networking, etc.
- If your **business is small**, you consume **less cloud resources**
- If your business is growing, you consume more resources
- **Pay as you go**
  - Start for **free**, **pay** when you grow and need more resources



Microsoft Azure

- Fast-growing **public cloud** from **Microsoft**
- Provides **rich PaaS platform**
  - Mainly for **.NET developers**
  - Provides also Java, PHP, Python, and Node.js APIs
  - **Databases, storage, mobile back-ends, CDN**, …
- Provides **IaaS cloud** (Windows and Linux VMs)
- **Azure for Students** https://azure.microsoft.com/free/students/

Summary

- **Front-End**: client-side apps
  - HTML + CSS + JavaScript + AJAX
- **Back-End** == server-side apps and APIs
  - Back-end logic: databases, data processing, ORMs, APIs, Web APIs, MVC frameworks
- **Containers** and **Docker**
  - Run OS with preinstalled apps in a container
- **Cloud** == rental of computing resources

Software University

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни БУДИТЕЛ

Finance Academy

SoftUni Diamond Partners

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank
Решения за твоето утре

DRAFT KINGS

VIVACOM

AMBITIONED

PHAR VISION

INDEAVR
Serving the high achievers

SOFTWARE GROUP

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License