University of Batna 2
Faculty of Mathematics and Computer Science
Computer Science Department
Master ISIDS
Semester 2

Advanced Web Development

Dr O.MESSAOUDI

2022/2023

Chapter 1
Introduction to Web Apps

- Apps Web: Introduction
- Features and components of web apps
- Concept of Client-Server
- Emergence of Apps Web
- History and evolution
- Web apps types
- Design patterns

Web Apps: Introduction

- **A web application (web app)** is a software application that runs on a web server and can be accessed over the Internet through a web browser.

- **A web app** allows users to interact with the application through a web interface, without the need to download and install any software.

- **Examples:** online shopping, email, social media, and project management tools.

Web Apps: Common features

Using web apps, users usually encounter features such as:

- User registration and login
- Dynamic content
- Real-time updates
- Notifications
- Online transactions

Web Apps: Core components

Building web apps require:

- Client-side (front-end): HTML, CSS, JavaScript
- Server-side (back-end): Database, Web Server, Application Server
- Server-side programming language: PHP, Ruby, Python, Java, etc.
- Web Protocols: HTTP, HTTPS
- Framework: Ruby on Rails, Django, ExpressJS, etc.
- API for server-to-server communication.

These components interact with each other to deliver dynamic, interactive web experiences to users.

Basic concepts of client-server Architecture

Client-server architecture refers to the organization of a system in which one or more clients request and receive services from one or more servers.

In the context of web applications, the client is typically a web browser running on a user's device, and the server is a remote computer that provides services to the client.

Basic concepts of client-server Architecture

The basic components of a client-server system are:

- **Client:** The client is the device or software application that requests services from the server. In the case of a web application, the client is typically a web browser.

- **Server:** The server is the device or software application that provides services to the client. In the case of a web application, the server is typically a computer that is running a web server software, such as Apache or Nginx.

Basic concepts of client-server Architecture

The basic components of a client-server system are:

- **Protocol:** defines how clients and servers communicate with each other. In the case of web applications, the most common protocol is the Hypertext Transfer Protocol (HTTP).
- **Request and response:** In a client-server system, the client sends a request for services to the server, and the server returns a response.
- **Statelessness:** The server does not retain any information about previous requests or responses. This allows for greater scalability, as the server does not have to keep track of the state of each client, but it also means that the client must provide all of the information needed for each request.

Web Apps: History and evolution

The first web apps were simple, static websites that provided information and resources, but as web technology advanced, they became more dynamic and interactive.

Web Apps: History and evolution

The introduction of Ajax (Asynchronous JavaScript and XML) in the early 2000s enabled web apps to update content in real-time without reloading the page.

The rise of mobile devices and the increasing availability of high-speed Internet have also driven the growth of web apps.

Today, web apps are a major part of the software landscape, and they are continuing to evolve, incorporating new technologies such as artificial intelligence, machine learning, and virtual and augmented reality.

Web Apps: History and evolution

- **Web 1.0**
  - Static web pages
  - First business model based on web technology
  - Mosaic : The first GUI web browser
  - Netscape, Mozilla, Microsoft Internet explorer
- **Web 2.0**
  - Introduction of AJAX and dynamic content
  - Examples : Social networks, e-commerce, wikis, etc.
  - Apps web vs Apps desktop
- **Web 3.0**
  - Web intelligent (ubiquity, meta information)
  - Internet of Things
  - Machine learning

Web Apps: History and evolution

Web 2.0 and 3.0 Enablers

- JavaScript, XML, JSON (Ajax)
- Communication asynchrone
- Possibility to user web services (REST)
- Cloud computing (SaaS)

Everything will be Web enabled

Web Apps types

Types of web apps and their architecture:

- Server-side web apps
- Client-side web apps
- Hybrid web apps

Web Apps types

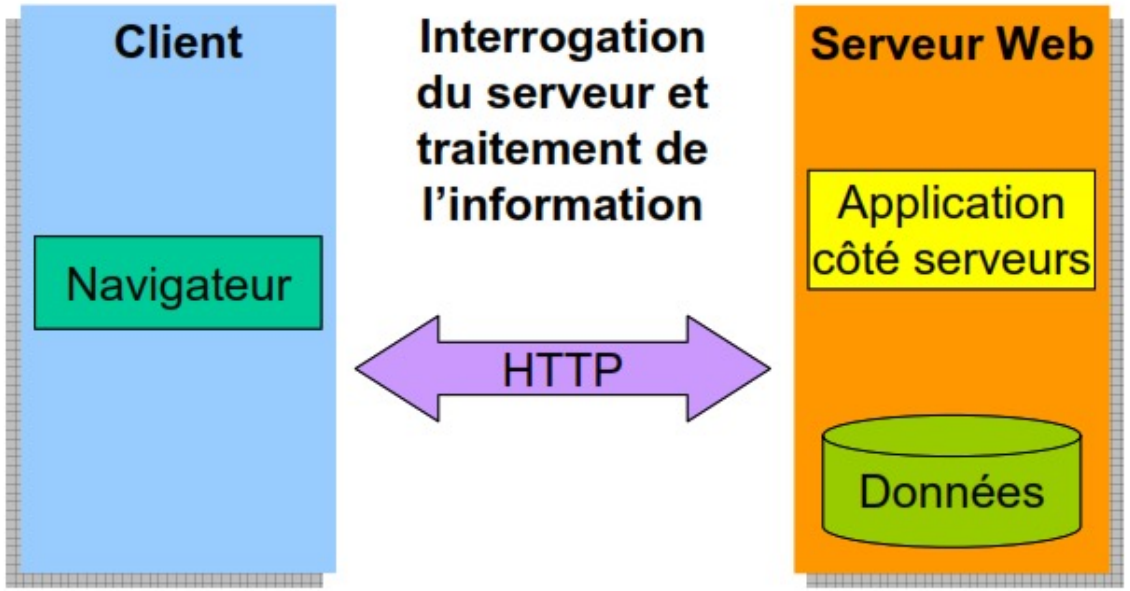
1 - Server-side web apps

In a server-side web app, all processing is performed on the server, and the results are sent to the client in the form of HTML, CSS, and JavaScript.

Web Apps types

1 - Server-side web apps

- The client sends an HTTP request from his browser
- The server responds and sends a response back
- The client receives the response and renders an HTML document



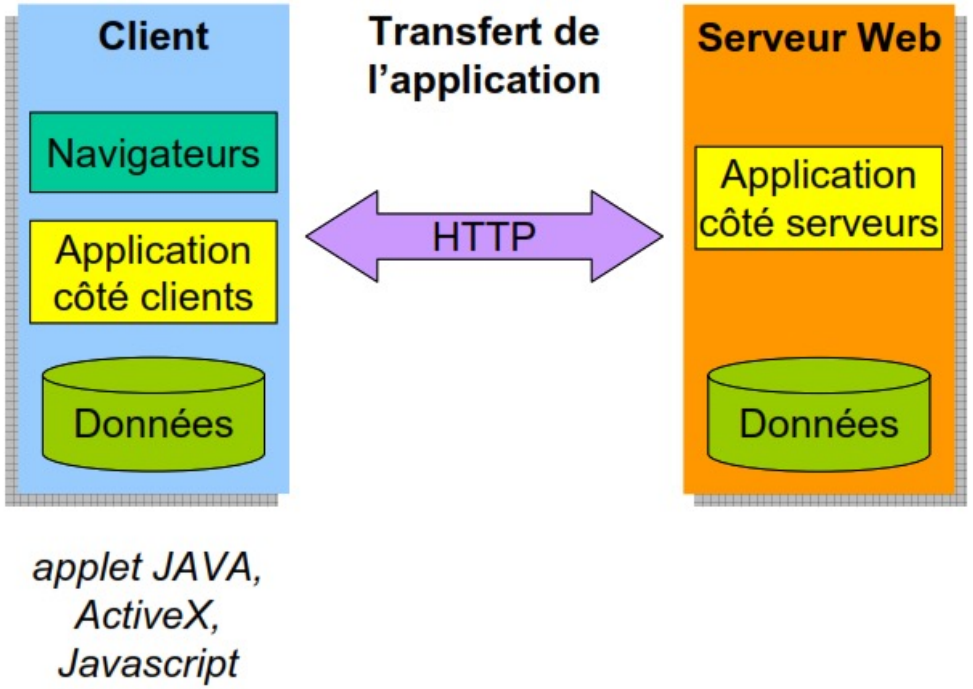
Web Apps types

2 - Client-side web apps

In a client-side web app, all processing is performed on the client (typically in the web browser) and the server is only used to provide data.

2 - Client-side web apps

- The client sends an HTTP request from his browser
- The server responds by sending data and code
- The client receives the response and renders an HTML document by executing the received code using server data.



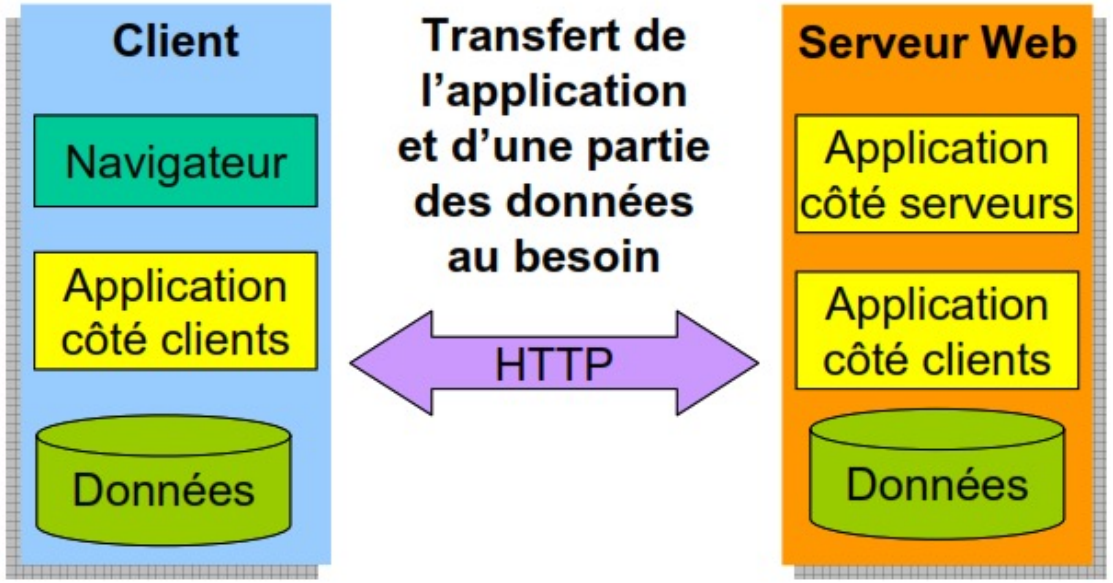
Web Apps types

3 - Hybrid web apps

A hybrid web app is a combination of server-side and client-side processing, with some processing performed on the server and some performed on the client.

3 - Hybrid web apps

- The client sends an HTTP request from his browser
- To respond, the server executes part of the code to generate and send an HTML document, along side data and/or code.
- The client receives the response and renders an HTML document. And uses data and code to update the interface without needing to go back to the server.



Web Apps types

Impact of web app architecture on:

It's important to note that the architecture of a web app can affect its ***performance***, ***security***, and ***scalability***.

It's important to choose the appropriate architecture for a web app based on its specific requirements. In many cases, a hybrid approach can provide the best balance of these factors.

Web Apps types

1 - Performance

- Server-side web apps can provide good performance for large and complex applications.

- Client-side web apps can provide responsive user interfaces.

- Hybrid web apps can provide a good balance, with some processing performed on the server for efficiency, and some performed on the client for responsiveness.

Web Apps types

2 - Security

- Server-side web apps can provide better security, as sensitive data is processed and stored on the server.

- Client-side web apps can be more vulnerable to security threats, as sensitive data is processed and stored in the client's web browser.

- Hybrid web apps can provide a balance of security, with some sensitive data processed on the server for security and some processed on the client for performance.

Web Apps types

3 - Scalability

- Server-side web apps can be more scalable, as the processing load can be distributed across multiple servers and the database can be easily scaled to handle larger amounts of data.

- Client-side web apps can be more limited in terms of scalability, as there is no easy way to distribute the processing load on the client side.

- Hybrid web apps can provide a balance of scalability, with some processing performed on the server for scalability and some performed on the client for performance.

Design patterns

Design patterns are reusable solutions to common problems in software development, and they can be applied to web applications as well.

Problems include:

- Complexity
- Code Duplication
- Tight Coupling
- Scalability
- Testability
- Maintainability

Design patterns

Commonly used in web apps:

- Model-View-Controller (MVC)
- Model-View-ViewModel (MVVM)
- N-tier

Design patterns

Model-View-Controller (MVC): *MVC is a design pattern that separates an application's data, presentation, and control into three separate components. In web Apps:*

- *Model:* represents the data
- *View:* is the HTML template that is rendered in the browser
- *Controller:* is the code that manages the communication between the model and the view

Design patterns

Model-View-ViewModel (MVVM): *MVVM is a variation of MVC that is commonly used in web applications built with JavaScript frameworks, such as Angular and Vue.js.*

- *View and model are the same as the MVC.*
- *View Model:* The view model serves as a mediator between the model and the view, and it is responsible for updating the view whenever the model changes.

N-tier architecture

The ***N-tier architecture*** is a design pattern that separates a web application into multiple tiers or layers, each with a specific responsibility. It is also known as multi-tier architecture.

Each tier can be physically separated, meaning that it can run on a separate server, or it can be logically separated, meaning that it can run on the same server but be separated in the application code.

N-tier architecture

The most common tiers in a N-tier architecture are:

- ***Presentation tier:*** This tier is responsible for the user interface and presentation of data to the user. It typically includes HTML, CSS, and JavaScript, as well as any client-side frameworks or libraries.

- ***Application tier:*** This tier is responsible for the application logic and processing. It typically includes a web server, middleware, and any server-side frameworks or libraries.

- ***Data tier:*** This tier is responsible for storing and retrieving data. It typically includes a database server and the database itself.

Web Apps vs other apps

Advantages	Disadvantages
Accessibility	**Limited Functionality**
Compatibility	**Slow Performance**
Easy Deployment	**Security**
Easy Updating	**Offline Access**
Cost-effective	**Dependence on Internet**
	Connectivity *(fast and reliable)*