Développement Web - Back-end

Antoine TIREL

Premier trimestre 2025

Table des matières

1 Définitions

1.1 Application web

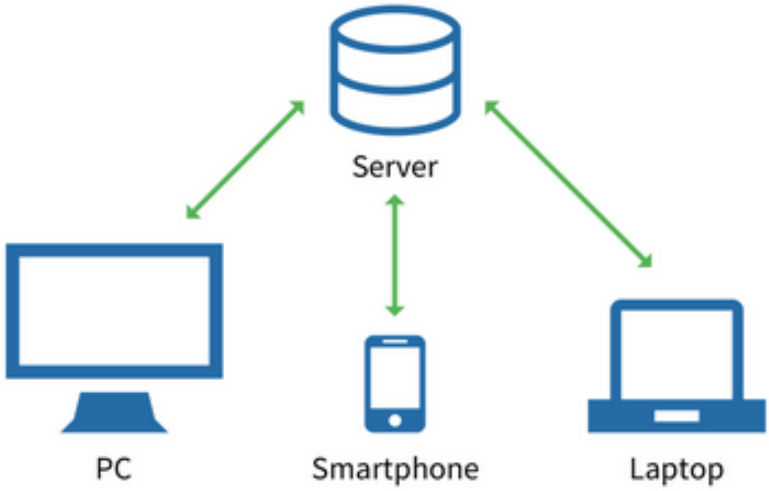
“En informatique, une application web est une application manipulable directement en ligne grâce à un navigateur web et qui ne nécessite donc pas d’installation sur les machines clientes, contrairement aux applications mobiles."
Wikipédia

“Une application Web est un logiciel qui s’exécute dans votre navigateur Web. Les entreprises doivent échanger des informations et fournir des services à distance. Elles utilisent des applications Web pour se connecter aux clients de manière pratique et sécurisée."
Amazon

“Une application web est un logiciel accessible et exécuté sur un site web via le navigateur de l’utilisatrice ou de l’utilisateur. Les entreprises utilisent des applications web pour fournir une large gamme de fonctionnalités dans leur navigateur à leur clientèle. Il n’est donc pas nécessaire de télécharger et d’installer des logiciels."
OVH

Une application web est donc un logiciel applicatif, accessible et traduisible par tous les navigateurs ~~sauf Internet Explorer~~, ne nécessitant aucune installation sur la machine client.
Une application web fonctionne traditionnellement sur le modèle client serveur, bien qu’il puisse exister une quantité de couches intermédiaires plus ou moins manipulables et visibles.

1.1.1 Une petite modélisation



1.2 Histoire

Le **Worl Wide Web** est un système hypertexte public fonctionnant sur Internet, inventé par le CERN en 1990 permettant d'afficher du texte mis en forme sur un navigateur écrit en Objective-C et pouvant être utilisé en C.

1.3 Technologies

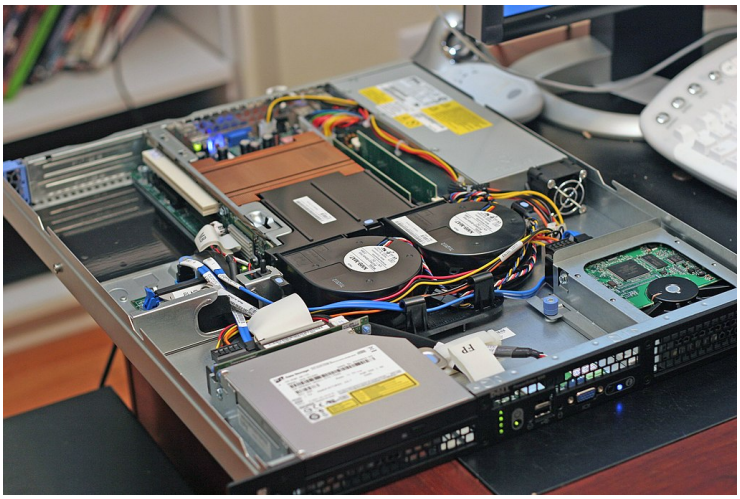
- **HTTP** ou Hyper Text Transfer Protocol pour les intimes
  - Permet de transférer des documents hypertextes
  - Aujourd'hui pratiquement disparue, remplacé par sa version sécurisée **https**
  - Utilise par défaut le port 80, là où le https utilise le port 443
  - http et https utilisent bien le même protocole, et le https est souvent nommé http over TLS pour préciser que le protocole ne change pas, mais qu'il dispose juste d'une surcouche d'encryption des données.
- **HTML** ou Hyper Text Markup Language
  - Permet d'écrire des document hypertexte
  - Ces documents peuvent inclure des images, du texte, du binaire
- **TCP/IP** Transfer Control Protocol/Internet Protocol
  - Protocole de "livraison" des données
  - Basé sur un principe de connexion double
  - Le serveur écoute en permanence, gère les demandes de connexion en les validant et en assurant le résultat
  - Peut dans certains contextes, être remplacé par **UDP**, particulièrement dans les cas où les pertes de données sont négligeables.
- **MIME** standard d'envoi de mail

1.4 Modèle Client Serveur

Le modèle Client Serveur consiste à découper une application en deux couches principales, chacune ayant sa responsabilité, les deux devant fonctionner conjointement pour fournir une application Web fonctionnelle.

1.4.1 Serveur Web

Un serveur web est une machine qui écoute, réceptionne et traite des requêtes. Il est, en général, composé d'un OS, d'un serveur d'écoute, d'une BDD.



[rod24]

1.4.2 Catégories de serveur

- **Statique**
  - Renvoie des données en lecture seule
  - Laisse la responsabilité des modifications du navigateur au client
  - Est, de fait, beaucoup plus sécurisé, les données du serveur ne pouvant pas être modifiées par le client
- **Dynamique**
  - Peut modifier les données
  - Permet de séparer les responsabilités et d'alléger le réseau

1.4.3 Client

Gère l'affichage ou la récupération des données, peut tout aussi bien être un navigateur, qu'une interface CLI qu'un outil dédié.

Exemples :

- **Navigateurs**
  - Chromium
  - Firefox
  - Google Chrome
  - Opera
  - Safari
- **Outils CLI**
  - curl
- **Outils d'envoi de requête**
  - Postman
  - Bruno
  - Wireshark du du dudu

1.4.4 Client VS Serveur

Avec tout ça, on voit émerger deux clans, appelés les fronteux et les backeux, si on excepte les fullstack et les Ops/Architectes.

Dans l'équipe du back

- Le côté caché d'une application
- Assure l'intégrité des données
- Gère la plupart des calculs
- Met à disposition du consommateur
- Est responsable du "pourquoi ça marche"
- Bosse sur la même techno depuis avant ma naissance

Dans l'équipe du front

- Le côté sexy de l'application
- Assure le suivi des contraintes de visuel
- Fait du CSS (et se soigne ensuite)
- Est un artiste abstrait
- Est responsable du "à quoi ça ressemble"
  - Est-ce que la mise en page garde l'œil ?
  - Est-ce qu'un lecteur d'écran peut parcourir la page ?
  - Pourquoi ça ne marche pas ?
- Doit suivre le dernier truc à la mode

Dans l'équipe des full-stack

— Fait du front et du back
— Est de fait meilleur, parce que plus facile à vendre
— Mais est en fait moins bon, parce que pas spécialiste

1.4.5 Où il est un peu question de déploiement

Un code développé sur une machine ne sert pas à grand chose en lui-même, à part pour briller en soirée mondaine. Pour assurer la mise à disposition des applications web, on fait appel aux Ops (IT ops pour les anciens) qui sont responsables du déploiement des applications.
Le déploiement consiste en la mise à jour d'un serveur afin qu'il embarque les dernières modifications réalisées et puisse répondre aux besoins du client final, celui qui paie.
Historiquement, le déploiement était réalisé par des équipes dédiées, ce qui menait souvent au problème du "ça marche sur ma machine".
Ce problème a été résolu avec l'apparition de nouveaux outils de déploiement plus proches du développement qui ont donné naissance aux DévOps, ayant à la fois les notions d'opérations et de développement.

1.5 Langage et frameworks

On l'a vu dans les précédentes sections, une application web ne permet que l'affichage de pages HTML. De fait, un serveur web ne fait qu'envoyer des pages HTML ou du contenu qui sera ensuite traduit pour devenir une page HTML.

Afin de simplifier le travail de développement, et de permettre une grande versatilité, un cadre a été ajouté sur la plupart des langages. Ce cadre (frame en anglais) est la base du concept de framework, qui peut se résumer en une bibliothèque permettant de réaliser l'intégralité des actions attendues par un serveur et un client.

2 Parlons back end

2.1 Pour la culture

2.1.1 PHP

Le langage PHP est un des tous premiers créés et mis à disposition pour la rédaction de pages sites et applications web.
De fait, il est l'un des plus vieux et des plus décriés aujourd'hui, et vous n'aurez aucun mal à en voir du mal partout.
Le langage est particulièrement réputé pour ses failles de sécurité historiques et sa relative non fiabilité.
Toutefois, il représente plus de 75% des sites internet en ligne, et dont 62% sont sur des versions connues pour leur vétusté et leurs failles de sécurité.
Aujourd'hui, le langage n'a pas beaucoup de popularité mais est tout de même maintenu à jour et suivi par une communauté active lui assurant la même pérennité que la plupart des autres langages utilisés pour faire du développement web.

2.1.2 Les dinosaures

Comme vu dans la partie historique du web, le world wide web date de 1990, et est donc sorti après plusieurs langages comme le C, le C++ ou en même temps que le Python.
Si le C et le C++ ont rapidement été dépassés et remplacés par le PHP, qui reprend beaucoup de points de leur syntaxe, Python a mis à disposition deux frameworks : un minimaliste Flask et un lourd Django.

2.1.3 Le monde magique de Microsoft

Dans le monde merveilleux de Bill Gates où tout le monde n'utilise que Windows et où les autres systèmes d'exploitation sont bannis, que tous les serveurs tournent sur Windows Server, un framework permettant de faire du développement web en C# (ou en F# pour les gens bien), un langage inspiré de Java, Dotnet.
Dotnet possède pour lui l'avantage d'être soutenu par Microsoft, ce qui lui assure une vie possiblement aussi longue que Windows, C# étant également utilisé dans la plupart des modules de Windows.

2.2 Java

À l'origine, Java est un langage orienté objet, un des premiers à complètement se détacher du C développé par Sun Microsystems.
Sa facilité d'accès, sa nouveauté, ainsi que la JVM lui assurent une grande popularité, et son apparition peu après le web permet également une prise de décision rapide avec la création de la plateforme Java et de ses nombreuses sous-divisions, principalement Java SE (Standard Edition) et Java EE (Entreprise Edition), deux services ayant accès à des bibliothèques Java, gérant les transaction, les serveurs web applicatifs, la concurrence.
Aujourd'hui, Java est le langage le plus répandu dans l'industrie en France pour du développement back-end, en particulier sur les nouveaux projets et les projets lourds ayant démarré après la grande histoire du PHP.

2.2.1 De la compilation et un peu d'histoire

Java est un langage compilé, dont la compilation produit un fichier .class, qui tourne dans la JVM, une machine virtuelle Java, ce qui permet au code d'être exécuté sur n'importe quel système d'exploitation.
Cette particularité de Java date de la version 1 sortie en 1996.
Historiquement, Java suivait un process de release lent, et ne sortait une nouvelle version que tous les deux ans en moyenne, chaque release étant propriétaire et fermée.
En 2006, Sun crée OpenJDK et ouvre le développement de Java en le rendant open-source, peu de temps avant la sortie de Java 6, ce qui amène de nombreuses personnes à collaborer à la création de Java 7, qui ne sortira que 5 ans plus tard.
Java 7 devient rapidement le langage le plus utilisé pour les projets web, et si vous rencontrez un projet de cette époque ou plus vieux, fuyez. Le langage est également adopté par Android pour le développement de ses applications, et, aujourd'hui encore, un grand nombre d'applications Android sont des fichiers class transformés.
En 2014 sort Java 8, la version immortelle, qui apporte de nombreuses fonctionnalités encore utilisées aujourd'hui et qui est la plus vieille compatible avec la totalité des bibliothèques qui seront utilisées dans ces cours et TD.
Cependant, par soucis de rétro-compatibilité, Java traîne une énorme partie de son historique, ce qui lui vaut de voir apparaître de plus en plus d'alternatives, qui reprennent toutefois sa base.

2.3 Kotlin

Le monde du web évoluant et le langage Java vieillissant, de nombreuses alternatives au langage apparaissent, principalement pour en simplifier l'écriture trop verbeuse.
Parmi ces alternatives, JetBrains décide de proposer Kotlin, un langage de programmation orienté objet dont la compilation produit des fichiers .class, lisibles par les jvm.

Sa facilité et son efficacité, avec notamment une bien meilleure gestion des exceptions que Java au moment de sa sortie conduit Android à privilégier le développement d'applications en Kotlin plutôt qu'en Java, et que Google place officiellement Kotlin comme langage par défaut pour Android Studio et le playstore.

3 Parlons Web

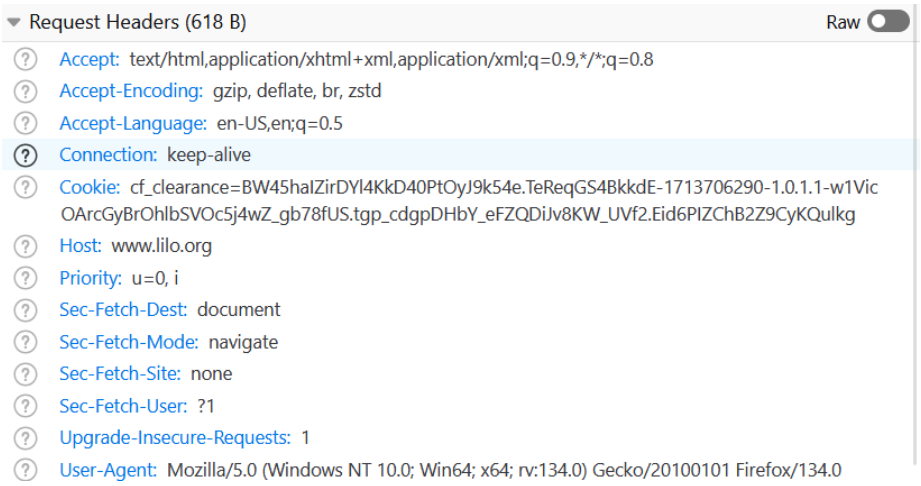
3.1 HTTP

HTTP est le protocole d'échange pour le Web, mais concrètement, comment ça marche ?
HTTP fonctionne selon le principe de requêtes pour communiquer À une URL dédiée sont associées des requêtes, définies par des verbes

- GET, HEAD -> pour récupérer des informations
- POST -> pour créer des informations
- PUT, PATCH -> pour modifier des informations
- DELETE -> pour supprimer de l'information

Une commande est accompagnée d'un contenu divisé en deux parties :

1. HEADERS (en-tête) Contient toujours ou presque l'authentification, ainsi que des informations d'usage, typiquement le navigateur, la cible
2. BODY (corps) contenu réclamé par l'utilisateur



Les commandes donnent lieu à différentes réponses, auxquelles sont associés un code :

- 1XX -> sache que
- 2XX -> succès
- 3XX -> va voir là-bas
- 4XX -> Erreur client
- 5XX -> Erreur serveur

Un HEADER et un BODY

3.2 Un peu de théorie

Soit f une application, on dit que f est idempotente lorsque f(f(a)) = f(a), ce qui ne veut pas pour autant dire que f(a) = a comme par exemple avec la fonction valeur absolue.
Par abus de langage, on dit qu'une requête HTTP est idempotente lorsque son application en série ne modifie pas davantage les données sur le serveur qu'une simple application.
Les requêtes HTTP GET, PUT, HEAD et DELETE sont toutes dites idempotentes : appliquée en série, elles doivent avoir le même effet indéfiniment.
Un exemple :
J'ai un serveur avec les données suivantes :
Liste des prix Nobel de Physique
Je décide de supprimer Albert Einstein de la liste :

DELETE /nobel/albert_einstein

Si j'exécute cette requête en boucle, Albert Einstein n'étant plus dans la liste, il n'est pas supprimé et ma liste contient bien la même chose qu'après mon premier appel.

3.3 URL

Source : MDN

Une URL est composée de plein de blocs détaillés ci-après

- Scheme -> schéma Protocole de transfert permettant d'accéder à la ressource
  - http
  - https
  - mailto
  - ftp
- Authority -> La concaténation du nom de domaine et du numéro de port Le numéro de port est optionnel dans le cas où on utilise les protocoles http ou https.
- Path ou chemin
- Paramètres -> une série de clé/valeurs séparés par une esperluette
- Ancre -> une sous-adresse dans un document

Avec un exemple :
https ://fr.wikipedia.org/wiki/Réseau_Polytech#Écoles

3.4 Il reste un peu de protocole, je vous en remet ?

3.4.1 REST

Une API REST, ou en bon françois académicien Interface de Programmation d'Application à Transfert d'État REprésentationnel.
On a donc un protocole assurant l'interopérabilité lors de la communication. Il n'y a pas d'état, les données sont dynamiques et on peut donc en mettre en cache.
Une API REST communique en format MIME (JSON, XML, HTML, etc.)
Un exemple (qui se trouve être par le plus grand des hasards le TD)

- GET /Pokemon/id
- POST /Pokemon
- PUT /Pokemon/id
- PATCH /Pokemon/id/availability
- DELETE /Pokemon/id

Avantages

- Performant
- Accessible
- Profite du HTTP
- Largement utilisé
- Indépendant du langage

Désavantages

- Une synchronisation entre le client et le serveur est nécessaire

— Risque de surcharge

3.4.2 SOAP

SOAP ou Simple Object Access Protocol
La logique tient ici dans le “simple”, on se limite à la transmission de messages, sous forme d’enveloppe, en envoyant du HTTP et du XML.

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
               xmlns:m="http://www.example.org">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice>
      <m:StockName>T</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

Source : wikipedia

Avantages

— Extensible
— Profite du XML
— Indépendant du langage

Désavantages

— Le XML c’est verbeux
— ~~Mort~~

3.4.3 GraphQL

Le GraphQL, ce n'est pas du SQL, même s'il a le même Q et le même L.
On suit toujours la logique HTTP, mais uniquement avec des POST, et on effectue des queries et des mutations.

```
{
  hero {
    name
    friends {
      name
    }
  }
}
```

Query du client...

```
{
  "data": {
    "hero": {
      "name": "R2-D2",
      "friends": [
        {
          "name": "Luke Skywalker"
        },
        {
          "name": "Han Solo"
        },
        {
          "name": "Leia Organa"
        }
      ]
    }
  }
}
```

...réponse du serveur

Avantages

- Le client choisit ce qu'il requête
- Adapté aux architectures avec contraintes

Désavantages

- Le client DOIT choisir
- Pas de cache côté serveur
- n'utilise pas HTTP en entier

3.5 Langages d'échange

3.5.1 JSON

JavaScript Object Notation Le JSON est un format standard d'échange entre logiciels, il a l'avantage d'être lisible par un humain.
Le JSON est le format le plus répandu pour les échanges REST.

```json
{
    "glossary": {
        "title": "example glossary",
        "GlossDiv": {
            "title": "S",
            "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
                    "SortAs": "SGML",
                    "GlossTerm": "Standard Generalized Markup Language",
                    "Acronym": "SGML",
                    "Abbrev": "ISO 8879:1986",
                    "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
                        "GlossSeeAlso": ["GML", "XML"]
                    },
                    "GlossSee": "markup"
                }
            }
        }
    }
}
```

Le JSON est construit suivant le principe clé/valeur, à chaque clé est associé une valeur qui peut être soit une string, soit un nombre, soit un booléen, soit un tableau de tout ça.

3.5.2 XML

Extensible Markup Language, ou en bon français, langage de balise extensible. Super verbeux et plus présent pour des raisons historiques qu'autre chose.

```xml
<!DOCTYPE glossary PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
 <glossary><title>example glossary</title>
  <GlossDiv><title>S</title>
   <GlossList>
    <GlossEntry ID="SGML" SortAs="SGML">
     <GlossTerm>Standard Generalized Markup Language</GlossTerm>
     <Acronym>SGML</Acronym>
     <Abbrev>ISO 8879:1986</Abbrev>
     <GlossDef>
      <para>A meta-markup language, used to create markup
languages such as DocBook.</para>
      <GlossSeeAlso OtherTerm="GML">
      <GlossSeeAlso OtherTerm="XML">
     </GlossDef>
     <GlossSee OtherTerm="markup">
    </GlossEntry>
   </GlossList>
  </GlossDiv>
 </glossary>
```

3.5.3 YAML

Yet Another Markup Language
Le plus récent, le moins verbeux, le plus sexy, celui qu'on utilise pour nos dépendances en Kotlin

```yaml
glossary:
  title: example glossary
  GlossDiv:
    title: S
    GlossList:
      GlossEntry:
        GlossTerm: Standard Generalized Markup Language
        Acronym: SGML
        Abbrev: ISO 8879:1986
        GlossDef:
          para: |-
            A meta-markup language, used to create markup
            languages such as DocBook.
          GlossSeeAlso:
            GlossSeeAlso: ''
            GlossSee: ''
```

4 Interfaces de communication

4.1 Swagger

Swagger est un outil permettant d'exposer les points d'entrées (endpoints) des REST API d'un serveur.
Il permet, grâce à la spécification OpenAPI de décrire en JSON ou en YAML les retours des endpoints, de se baser sur la documentation, le type de données pour exposer les contrats d'interface et permet ainsi de se libérer visuellement du langage du serveur.

Références

[rod24] RODZILLA. “Serveur démonté”. In : *Wikipédia* (2024).