

Introduction au Système d'Informations

Notions d'architectures des systèmes d'information

- Le Système d'Information
- Clients
- Serveurs
- Datastores
- Communication au sein du SI
- Exemple Récupitulatif

Le Système d'Information

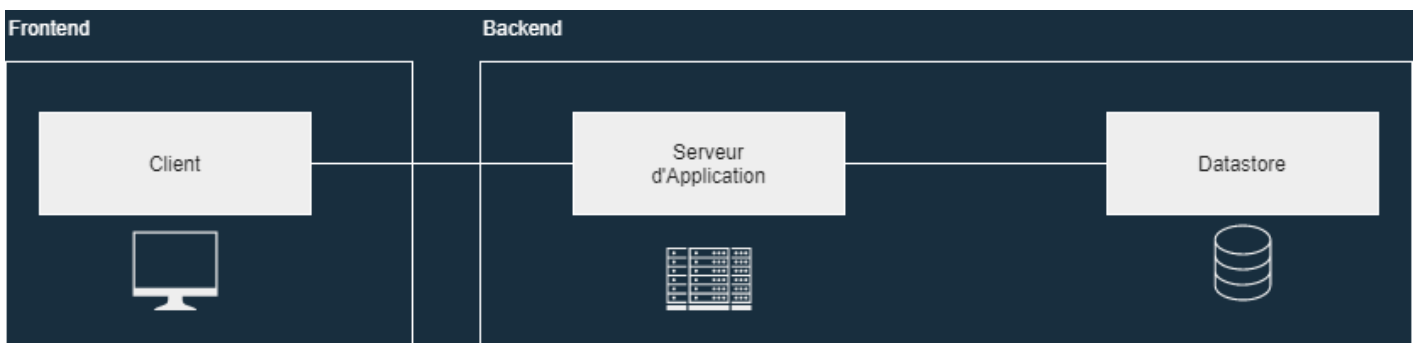
Un système d'information est une solution technique qui répond à une **problématique métier**. Une problématique métier c'est n'importe quelle situation de la vie réel, qui peut être assistée, accélérée, facilitée voir automatiser grâce à un SI. Les composantes d'un système d'information sont :

- Des données : informations gérées et manipulées par le SI
- Des acteurs : les personnes qui interagissent avec le SI
- Des processus : les actions des acteurs qui vont produire des traitements sur le données

A chacune de ces composantes correspondent des supports :

- Les datastores contiennent les données
- Les actions sont mis à dispositions des acteurs par les applications clients
- Les processus sont implémentés sur les serveurs

Ces différents supports sont interfacés par les réseaux, des protocoles, et des composants logiciels pour gérer ces protocoles.



Nous pouvons déjà établir une délimitation claire entre les composants du système qui appartiennent au backend et ceux qui appartiennent au frontend.

Clients

Les applications clients sont les applications avec lesquels les acteurs vont interagir directement. Elle disposent dans la majorité des cas d'une interface graphique pour en faciliter l'utilisation et pour les rendre accessibles aux utilisateurs non techniques. Nous allons voir les types les plus communs d'application clientes.

Client de bureau

Les clients de bureau sont des applications graphiques qui vont s'exécuter nativement sur le poste de l'utilisateur. Elles peuvent être développées avec presque tous les langages de programmation imaginables.

Exemples de technologies

- C# : Windows Form, WPF, Avalonia, UWP
- Java : Swing, JavaFX
- C++ : Win32 API, Qt

Clients Mobile

Les clients mobiles sont des applications graphiques qui vont s'exécuter nativement sur le smartphone de l'utilisateur. Elles peuvent être développées avec presque tous les langages de programmation imaginables.

Exemples de technologies

- C# : Xamarin
- Java/Kotlin : Android SDK
- C++ : Android NDK

Client Web Riche

Les clients web riches sont des applications développées ou compilées en Javascript qui s'exécute dans le navigateur du client. Elles sont principalement développées en Javascript, mais certains autres langages, comme par exemple Typescript, qui ont Javascript comme cible de compilation peuvent aussi être utilisé.

Certains client webs comme Discord peuvent se faire passer pour des clients de bureau, mais ne sont en fait des navigateurs dédiés qui affiche un client web, en utilisant par exemple le framework Electron.

Exemples de technologies

- Angular
- Vue
- React

Clients web "à pages"

Les clients web à pages sont des applications dont l'interface graphique est affichée sous forme de pages HTML rendues dans le navigateur de l'utilisateur. La logique de l'interface est majoritairement exécutée coté serveur, provoquant un chargement du navigateur à chaque interaction.

Serveurs

Les applications serveurs sont les applications qui forment l'épine dorsale du système d'information ; ils forment la partie la plus importante du back-end. Ils sont composés d'un certain nombre de couches qui vont chacune avoir un objectif bien précis. À noter qu'un même SI peut disposer de plusieurs serveurs qui peuvent se partager ces responsabilités. Les applications serveur peuvent être développées avec tous les langages de programmation.

Couche API

La couche API a pour rôle d'assurer l'interface avec le client. Les serveurs sont interrogés par les applications clientes afin d'accéder à des services. Ils doivent donc exposer ces services via des interfaces qui implémentent des protocoles de communication ainsi que des standards de communication au-dessus de ces protocoles. Les serveurs peuvent également être amenés à exposer des services qui auront pour vocation d'être utilisés non pas par des clients, mais d'autres serveurs, soit locaux au SI soit externes au SI.

Couches Services

La couche service a pour objectif d'implémenter les règles de gestion de l'application. Ce sont les règles propres à l'application, qui permettent de faire le lien entre la couche API, la couche persistance et la couche métier. Cela consiste notamment d'orchestrer sous quelle forme les données seront reçues et envoyées, ainsi que comment elles seront traitées avant d'être transmises à la couche métier et à la couche persistance.

Couche Métier

Logique métier

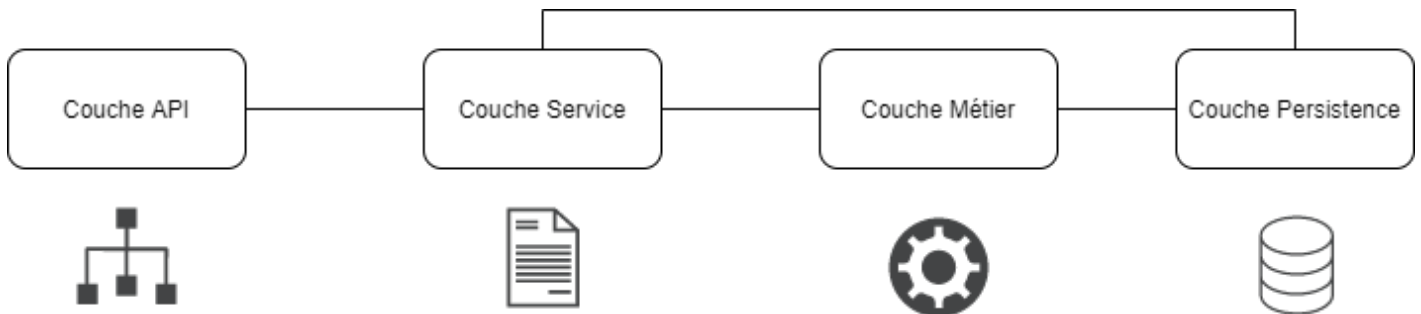
La couche métier implémente les règles de gestion du métier ; le code de logique métier, ou de logique du domaine, est l'implémentation des règles du monde réel. C'est à dire modéliser les interactions entre les acteurs et les données puis calculer les résultats de ces actions. L'objectif est de coder les règles de gestion, les processus métier de façon à répondre aux besoins du domaine du SI.

Modèle de donnée

Le modèle de donnée contient toutes les informations nécessaires et produites par la logique métier. Ce sont les données qui sont persistées dans les datastores. Elles représentent l'état de l'application. Ces informations sont souvent stockées sous la forme d'objets appelées Entités.

Couche Persistance

Le rôle de la couche persistance est de lire et écrire les données du modèle (les entités) dans un Datastore, afin de les stocker de façon durable.



Datastores

Les datastores sont des serveurs dont le but est uniquement de stocker des données. Dans la majeure partie des cas, ils s'agit de systèmes de gestion de base de donnée.

Bases de données relationnelles

Les bases de données relationnelles sont des base de données très strcturées composées de tables qui contiennent des lignes. On peut établir des contraintes sur les valeurs, et des relation entre les tables. Les SGBDR utilisent un moteur transactionnel strict, dit ACID (Atomicité Cohérence Isolation Durabilité).

Exemple : MySQL, Oracle, PostgreSQL, SQLite

Base de données NoSQL

Les bases de données NoSQL stockent les données sous la forme de documents JSON. Les relations ne sont pas supportées et ont un moteur transactionnel beaucoup plus léger n'implémentant pas le standard ACID. Cela permet au SGBD NoSQL d'être facilement distribuables sur plusieurs machines.

Exemple : MongoDB, Cassandra, CouchDB

Communication au sein du SI

Nous avons vu que le SI est constitué de différents éléments. Nous allons maintenant aborder la façon dont communiquent ces éléments.

Communication client serveur

Protocoles de communication

Le protocole de communication le plus massivement utilisé pour la communication client-serveur est le protocole HTTP. C'est un protocole synchrone (requête - réponse) en mode connecté (il utilise le protocole de transport TCP).

Vous pouvez consulter plus en détail le protocole HTTP dans ce cours : [Le Protocole HTTP](#)

Il existe aussi le protocole WebSocket, qui n'est pas exactement du HTTP mais est interopérable avec. Le protocole WebSocket permet d'établir une connexion full-duplex, c'est à dire que le client peut envoyer des messages au serveur et le serveur peut envoyer des messages au client.

Standards de services HTTP

Pour exposer des services sur le protocole HTTP, il existe plusieurs standards.

Representational State Transfer (ReST)

Le standard ReST est un standard sans état qui vise à faciliter l'interopérabilité en donnant plus de contrôle au client. Le concept principal de ReST est le concept de ressource. Une ressource est une donnée présente sur le serveur, sur laquelle on va effectuer des opérations en utilisant la sémantique des méthodes HTTP. On peut interagir avec des collections ou des entités individuelles de la ressource.

Le transport des données en Rest se fait soit au format XML, soit au format JSON, mais JSON est beaucoup plus utilisé.

[Exemple d'API ReST](#)

SOAP

SOAP (Simple Object Access Protocol) est un standard de webservice qui expose des fonctions qui peuvent être appelées par le client. La spécification se fait à travers d'un fichier WSDL (Web Service Description Language) au format XML. Ce standard est un peu désuet car il est assez lourd à utiliser.

GRPC

gRPC est un système d'invocation de procédure distance (remote procedure call) très moderne développé par Google. Les données sont transportées en binaires pour augmenter les performances et la version 2 du protocole HTTP est utilisée. Les services sont décrits par un langage dédié (Protobuf), avec support de génération de code pour la plupart des langages. Il supporte nativement des fonctionnalités comme l'authentification et le streaming bidirectionnel. Le protocole gRPC est très utilisé dans les infrastructures cloud pour la communication entre les conteneurs applications.

Exemple Récupitulatif