TP 1 : Objectifs du cours

1. Présentation du Framework .NET :

A. Définition :

Le .Net est un Framework conçu par Microsoft et en qui sont inclus plus de 27 langages différents qui ont été conçu sur une même base d'architecture, qui sont orientés objets et qui bénéficient de plusieurs classes. Microsoft a créé le .Net en partie pour que cette plateforme soit une référence en matière de création des web services.

Les types d'applications qu'on peut développer avec .Net :

- Les applications standard :

C'est à dire les programmes classiques qui sont exécutés sur l'ordinateur on en distingue deux types : celles exécutés en console(terminal) et celles exécutés en interface graphique qui utilisent l'API graphique de .Net.

- Les sites web :

On a premièrement les pages web qui utilisent les composants haut niveaux du Framework (et qui ont des extensions "aspx") et deuxièmement la partie traitement qui utilisent des langages du Framework.

- Web services :
  Permettent d'effectuer des traitements au cours des quels les données sont transmises en XML/SOAP.
- Les Services Windows :
  Les services Windows sont des programmes fonctionnant en arrière-plan (exemple : Le programme de protection des antivirus sous Windows)

B. Principaux langages :

Les 03 principaux langages de programmation avec lesquels on peut programmer une application .NET sont :

- ✓ C#,
- ✓ F#
- ✓ Et Visual Basic.

Ces trois langages sont pris en charge tant par Windows que par MacOs et Linux. Avec C++/CLI, les utilisateurs Windows disposent d'une option supplémentaire grâce à cette variante du langage classique C++ développée par Microsoft.

C. Utilisations :

Qu'est-ce qu'on peut faire avec .Net :

- ✓ Du web : Asp.net, Blazor.
- ✓ Du machine learning : Ml.net.
- ✓ IOT : runtimes optimisés.
- ✓ Apps Windows : WinForms , wpf , winui.
- ✓ Jeux vidéo : Unity.
- ✓ Du mobile : xamarin , maui.

D. Avantages et inconvénients :

- Avantages du Framework .net :
  - ✓ Orienté objets: le module de programmation orienté objet du framework .net permet de réutiliser des composants et du code cce qui permet de gagner en temps et du coût de développement.
  - ✓ Son puissant environnement de développement intégré(visual studio) qui permet de personnaliser l'environnement pour correspondre aux préférences des utilisateurs,aussi une solution peut être utilisée pour des applications basées sur le code écrit dans différentes langues.
  - ✓ Déploiement flexible et entretien facile: La conception modulaire permet d'inclure toutes les dépendances dont vous avez besoin pour le deploiement d'applications
  - ✓ Le noyau .NET prend en charge une large gamme d'applications: permettant de développer des applications dans une multitude de domaines(jeu, mobile,...)
  - ✓ Grande Communauté: on a une grande communauté de developpeur .net, impliquant

que presque n'importe quel problème peut être résolu avec l'aide des membres de la communauté

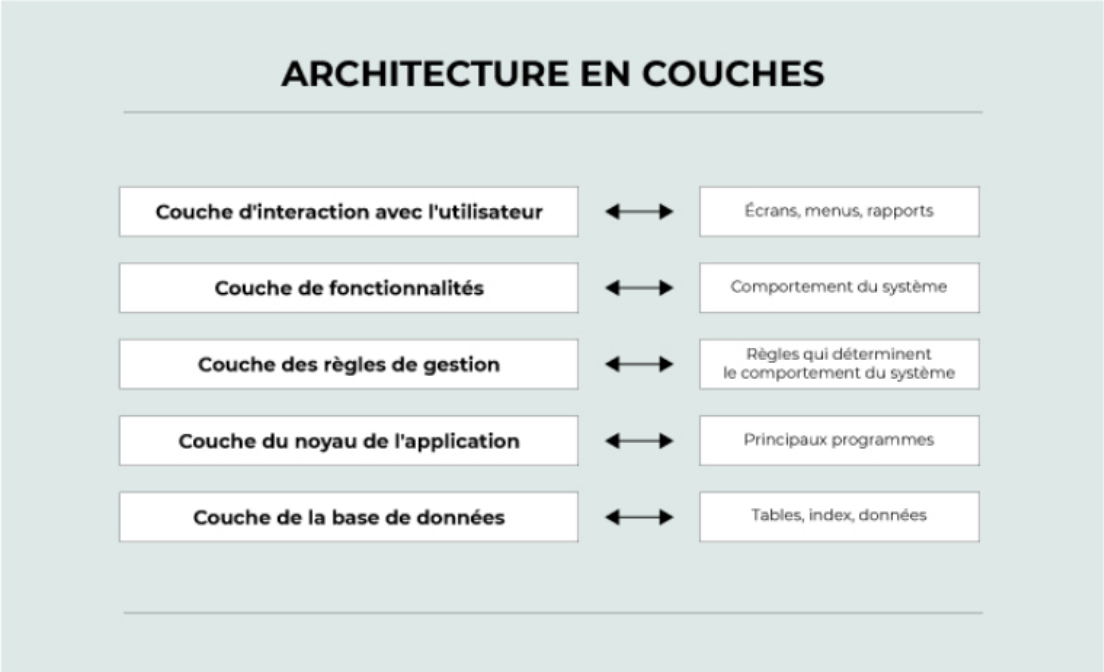
- ✓ Développement rapide: Le .NET Framework offre un large éventail de bibliothèques, de composants et d'outils pré-construits qui accélèrent le processus de développement.

- **Inconvénients du Framework .net :**
  - ✓ Perception communautaire : Dans certains cercles de développement, on peut avoir l'impression que .NET est principalement adapté pour des applications centrées sur Windows.
  - ✓ Plus grandes exigences en matière de mémoire et d'espace disque pour les applications.
  - ✓ Augmentation de la verbosité dans le code par rapport aux langages typés dynamiques.
  - ✓ Soutien limité à certains paradigmes de programmation modernes comme la programmation fonctionnelle.
  - ✓ La complexité peut submerger les développeurs inexpérimentés (les nouveau apprenants).

2- Présentation des différents types d'architectures :

L'architecture en couche :

C'est un modèle d'architecture dans lequel l'application est divisée en plusieurs couches hiérarchiques, sont bien spécifiques et on peut modifier une couche indépendamment des autres.



Architecture en couches

- <u>Explication de chaque couche de l'architecture :</u>
  - ✓ La couche d'interaction avec l'utilisateur (User interaction layer) : C'est la couche visible de l'application ; c'est la couche qui interagit avec les utilisateurs par le biais d'un écran, d'un formulaire de menu, etc.
  - ✓ La couche de fonctionnalités (Functionality Layer) : c'est la couche qui présente les fonctions, les méthodes et les procédures du système.
  - ✓ La couche des règles de gestion (Business rules layer) : Cette couche contient des règles qui déterminent le comportement de l'ensemble de l'application.
  - ✓ La couche du noyau de l'application (Application core layer) : Cette couche contient les principaux programmes, les définitions du code et les fonctions de base de l'application.
  - ✓ La couche de la base de données (Database layer) : Cette couche contient les tables, les index et les données gérées par chacun des modules.

- **<u>Avantages de l'utilisation de l'architecture en couche</u>**
  - ✓ Les couches sont autonomes : les changements effectués sur une couche n'affectent pas les autres.

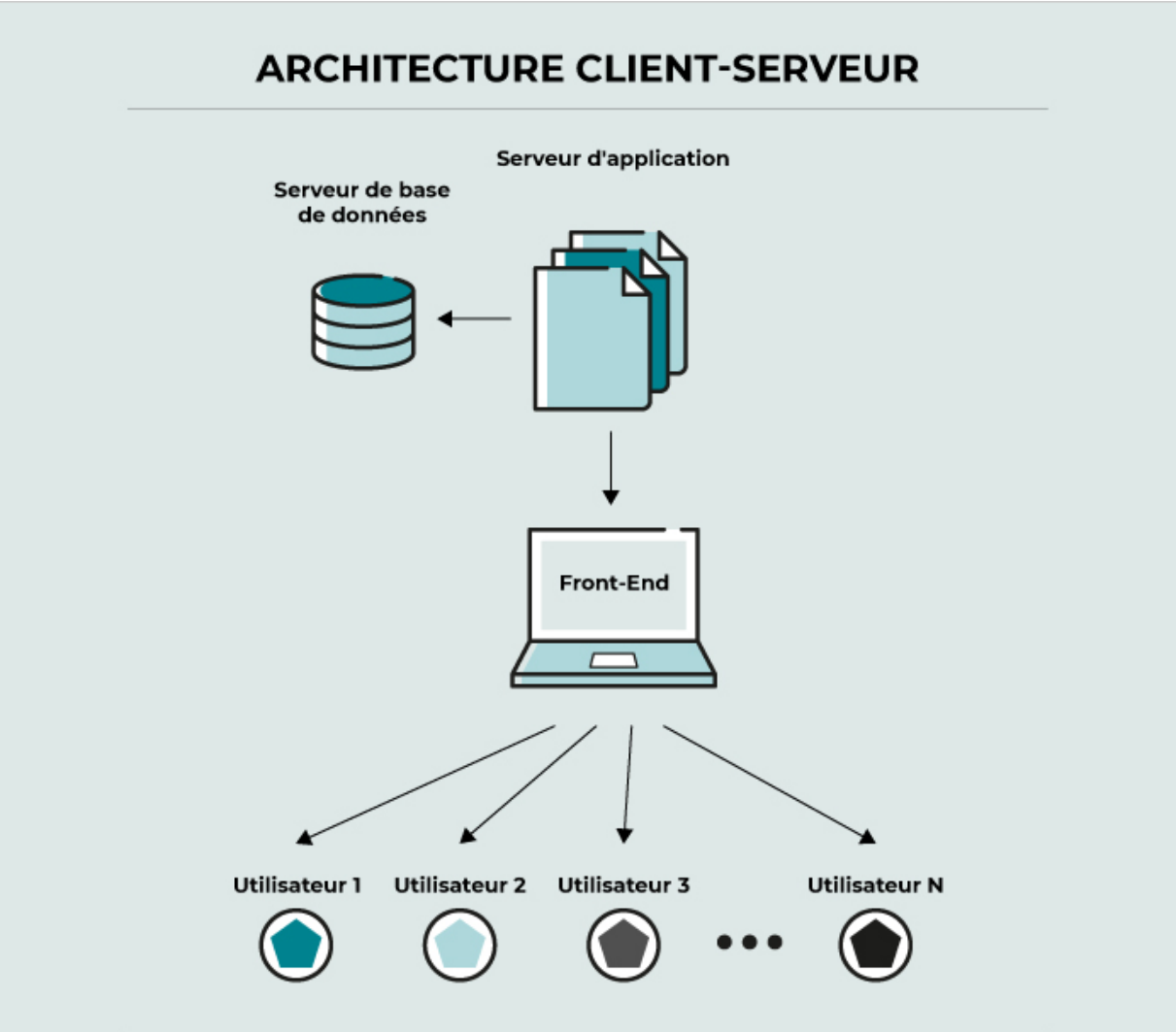
C'est pratique car nous pouvons augmenter les fonctionnalités d'une couche, par exemple faire en sorte qu'une application qui ne fonctionne que sur les PC fonctionne sur les téléphones et les tablettes, sans avoir à réécrire toute l'application.

- ✓ Les couches permettent une meilleure personnalisation du système.

- **Inconvénients de l'utilisation de l'architecture en couche**
  - ✓ Les couches rendent une application plus difficile à maintenir. Chaque changement nécessite une analyse.
  - ✓ Les couches peuvent affecter les performances des applications car elles créent une surcharge dans l'exécution : chaque couche des niveaux supérieurs doit se connecter à celles des niveaux inférieurs à chaque opération du système.

L'architecture client serveur

C'est une architecture dans laquelle les tâches sont repartis entre le client (le consommateur de service) et le serveur (le fournisseur de service).



- **<u>Explication de chaque couche de l'architecture :</u>**
  - ✓ La couche présentation(frontend) : il s'agit de la partie logicielle qui interagit avec les utilisateurs, que ce soit une tablette, un pc un téléphone.
  - ✓ Le serveur d'application : Il se connecte à la base de données, et interagit avec les utilisateurs.
  - ✓ Le serveur de base de données : Il contient les tables, les index et les données gérés par l'application.

- **<u>Avantages de l'utilisation de l'architecture Client-Serveur</u>**
  - ✓ Comme vous pouvez le voir dans notre exemple, l'architecture Client-Serveur sépare le matériel, le

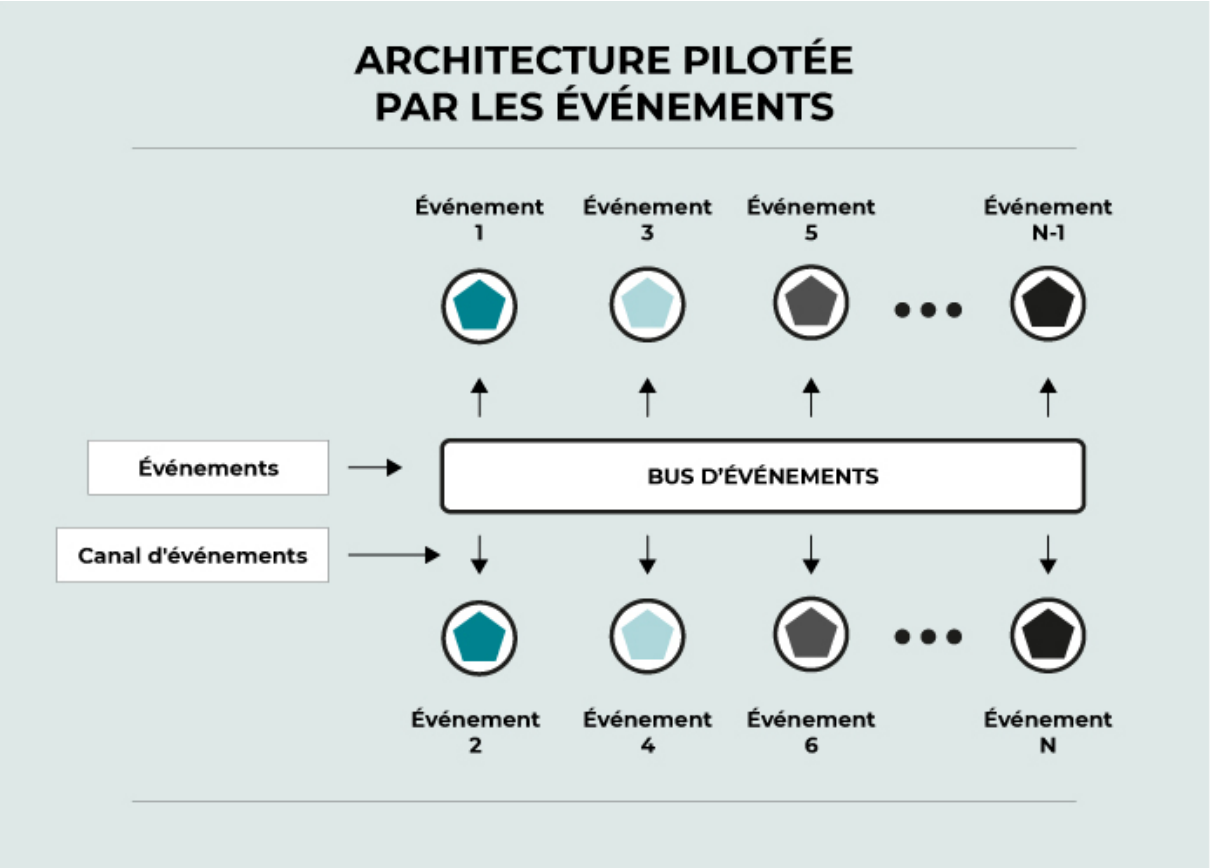
logiciel et la fonctionnalité du système. Par exemple, si une adaptation du logiciel est nécessaire pour un pays spécifique, autrement dit si un changement de fonctionnalité est nécessaire, celui-ci peut être adapté dans le système sans avoir à développer une nouvelle version pour les téléphones, les tablettes ou les ordinateurs portables.

- ✓ Puisque cette architecture sépare le matériel, le logiciel et la fonctionnalité du système, seule la partie front-end doit être adaptée pour communiquer avec différents appareils.

- **Inconvénients de l'utilisation de l'architecture Client-Serveur**
  - ✓ Si tous les Clients demandent simultanément des données au Serveur, celui-ci peut être surchargé.
  - ✓ Si le Serveur tombe en panne pour une raison quelconque, aucun utilisateur ne peut utiliser le système.

L'architecture pilotée par événements

C'est une architecture qui produit, détecte et agit selon les événements du système pertinents pour les utilisateurs. Si aucun événement ne se produit, rien ne se passe dans le système. Cette architecture définit ces événements indispensables comme des déclencheurs.



- Explication de chaque couche de l'architecture :
  - ✓ Le bus d'événements (Event Bus) : Il s'agit d'une ligne de communication qui relie tous les utilisateurs aux événements.
  - ✓ Le canal d'événements (Event Channel) : Un canal d'événements est une étiquette pour le type d'événements auxquels les utilisateurs s'abonnent.
  - ✓ Le traitement des événements (Event Processing) : Toutes les actions entreprises après un événement donné sont exécutées dans le module de traitement des événements. Ce module agit selon un événement donné pour servir l'utilisateur.

- **<u>Avantages de l'utilisation d'une architecture pilotée par les événements :</u>**
  - ✓ Si vous on a des utilisateurs qui ont besoin d'écouter (de s'abonner à) différents messages sur un sujet

spécifique, l'architecture pilotée par les événements est la bonne approche. Cela se produit lorsque le système doit réagir à des événements sans comportement prédictif.

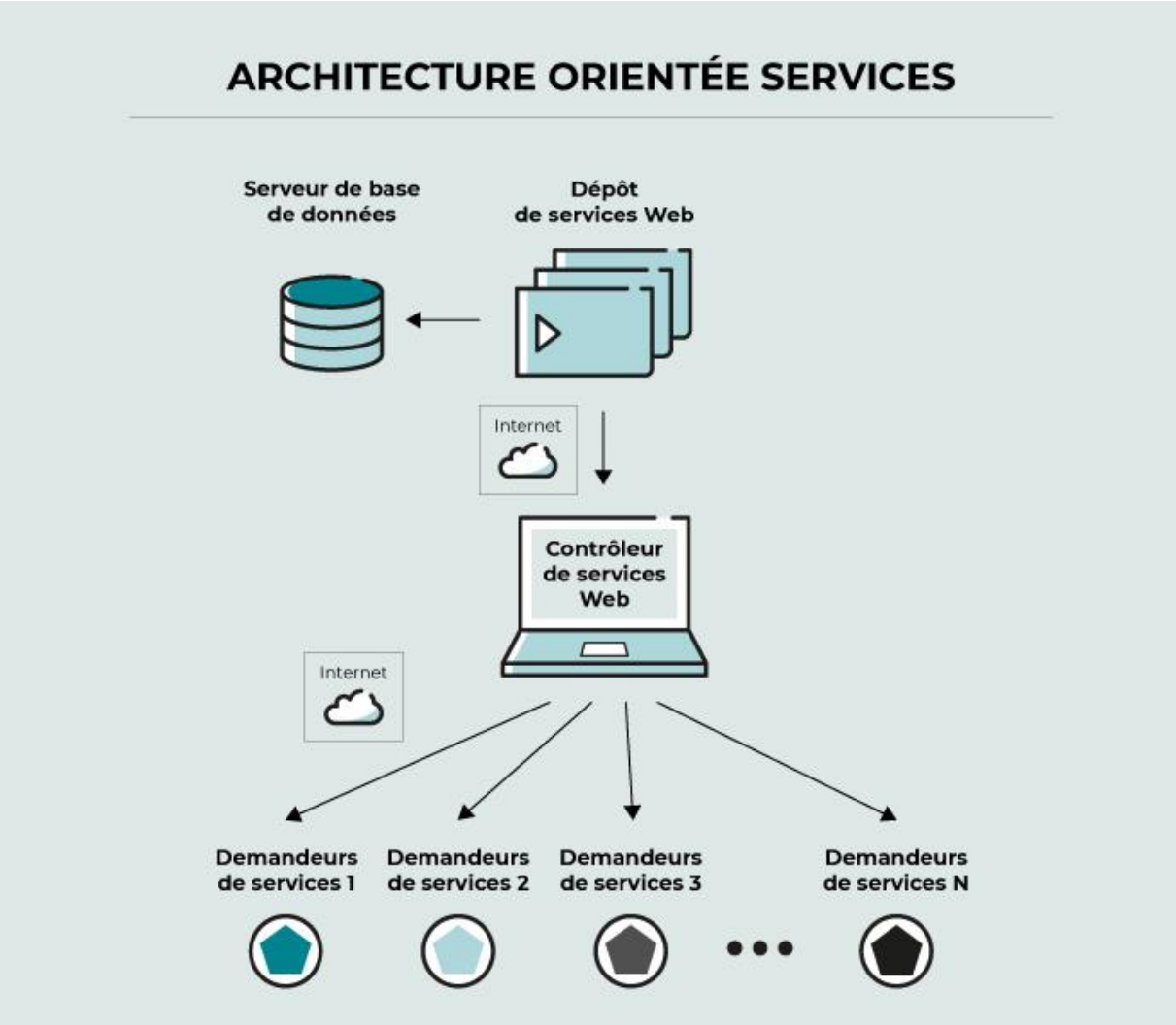
- ✓ Si ces utilisateurs sont en dehors de l'organisation, les événements peuvent être transportés dans un bus d'événements publics et atteindre tous les utilisateurs immédiatement.
- ✓ De nombreux événements peuvent être traités en même temps, voire des millions.

➢ **Il existe également quelques inconvénients majeurs :**

- ✓ Le bus d'événements peut être surchargé.
- ✓ Si le bus d'événements tombe en panne, tout le système tombe en panne.
- ✓ Il n'y a pas de contrôle du flux d'événements : de nombreux événements peuvent se produire en même temps, créant un véritable chaos pour les utilisateurs

Architecture orientée services

Dans cette architecture, les services sont fournis à des consommateurs externes par un processus de communication sur un réseau (Internet).



- **<u>Explications des différentes composantes :</u>**
  - ✓ Le dépôt de services web (Web service repository) : Il s'agit d'une bibliothèque de services web conçue pour répondre à des demandes d'informations externes. L'information fournie est généralement un petit élément, comme un numéro, un mot, quelques variables, etc.
  - ✓ Le contrôleur de services web (Web service controller) : Ce module communique les informations contenues dans le dépôt de services web aux demandeurs de services.
  - ✓ Le serveur de base de données (Database Server) : Ce serveur contient les tables, les index et les données gérés par l'application. Les recherches et les

opérations d'insertion/suppression/mise à jour sont exécutées ici.

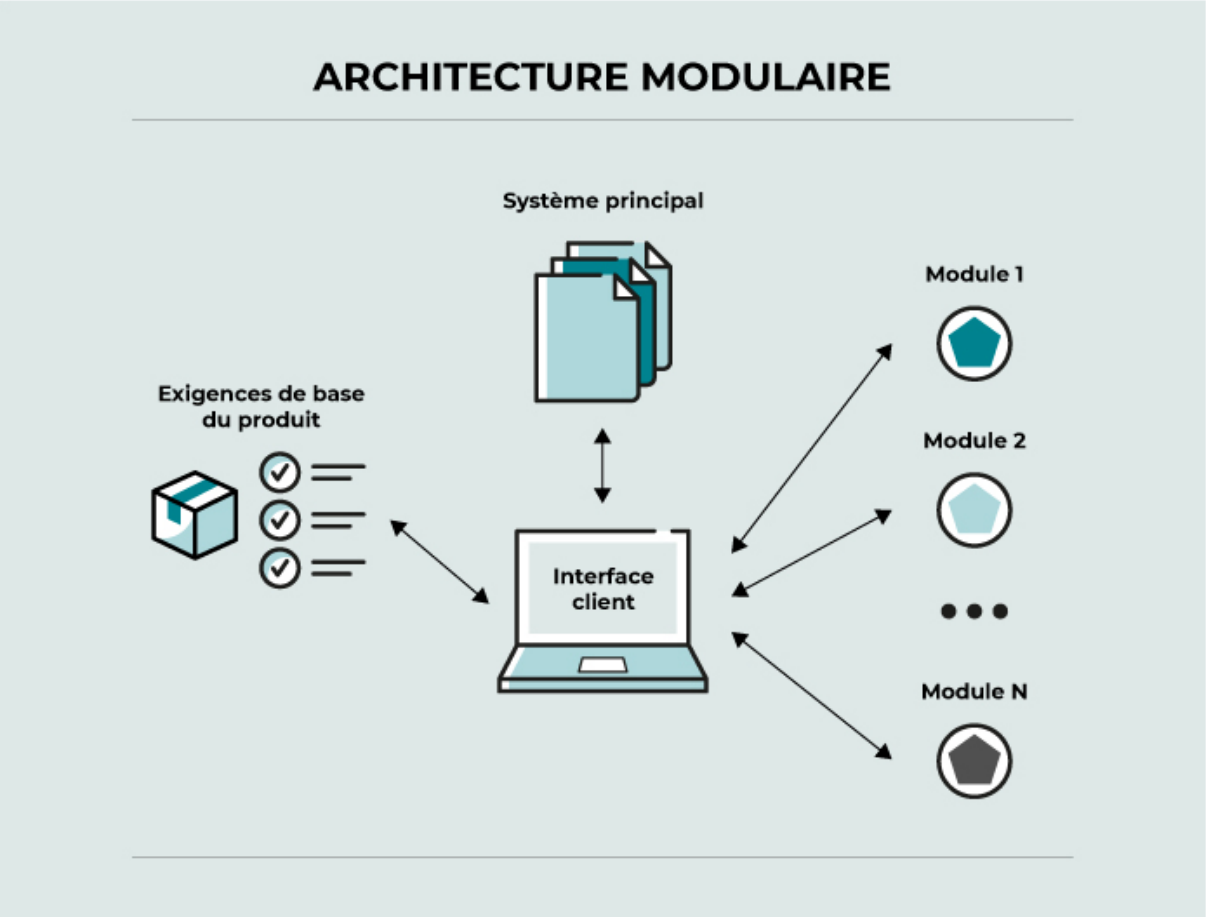
- ✓ Les demandeurs de services (Service Requester) : Il s'agit d'applications externes qui demandent des services au dépôt de services web par l'intermédiaire d'Internet, comme une organisation demandant des informations sur les vols à une compagnie aérienne, ou une autre entreprise demandant à un transporteur la localisation d'un colis à un moment donné.

- ➢ **L'utilisation d'une architecture orientée services présente plusieurs avantages**
  - ✓ Ce modèle permet de collaborer avec des acteurs externes sans les laisser accéder à nos systèmes.
  - ✓ Il permet également de partager avec le monde entier, de manière ordonnée et contrôlée, la sélection des fonctions les plus populaires du site.

- ➢ **Il existe également quelques inconvénients majeurs :**
  - ✓ Les services web peuvent représenter une faiblesse au niveau du site pour les pirates qui veulent engorger le système. Certaines formes d'attaques sont des « dénis de service ». Elles consistent à demander le même service web des millions de fois par seconde, jusqu'à ce que le serveur tombe en panne. Il existe aujourd'hui une technologie permettant de résoudre ce problème, mais c'est toujours un problème à prendre en compte dans les architectures de services web.
  - ✓ Le propriétaire du service web aide d'autres sites, mais reçoit une petite rémunération pour ce faire.

L'architecture modulaire

Une architecture modulaire est une architecture de conception logiciel qui divise un système en modules distincts, chacun responsable d'une fonctionnalité spécifique ou d'une partie délimitée de la logique métier. Les modules aussi peuvent être installés dans certaines de nos applications (des navigateurs web par exemple) on parle de "module additionnel".



- **<u>Explication de la structure :</u>**
  - ✓ Les exigences de base du produit (Baseline product requirements) : Il s'agit de l'ensemble des exigences minimales qui définissent l'application, déterminées au début du processus de développement lorsqu'un ensemble initial de fonctionnalités a été inclus dans le produit.
  - ✓ Le système principal (Main System) : Il s'agit de l'application à laquelle on connecte les modules. Le système principal doit fournir un moyen d'intégrer les

modules et, par conséquent, il modifiera légèrement le produit de base original pour assurer la compatibilité.

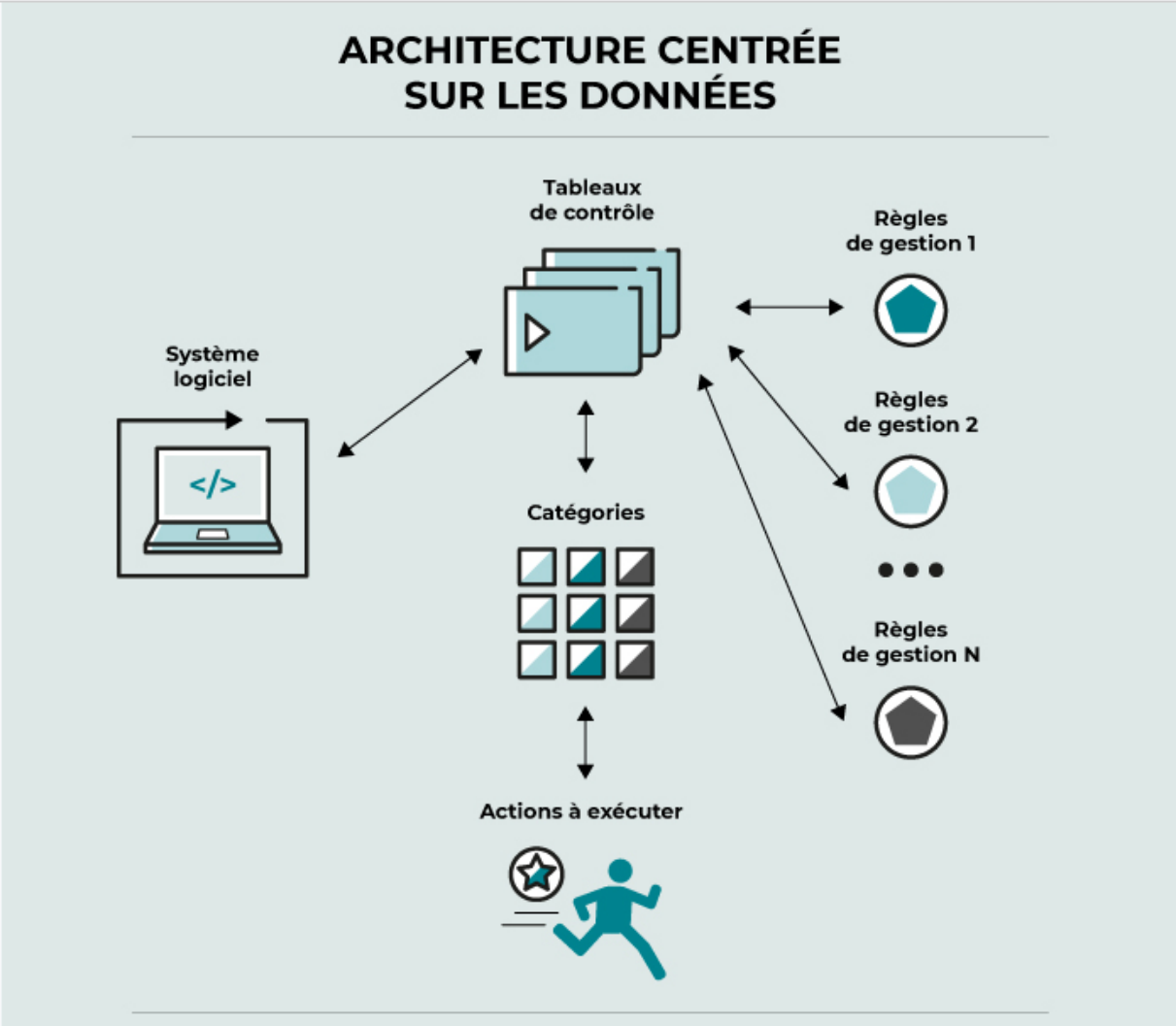
- ✓ L'interface client (Customer Interface) : Il s'agit de la partie qui interagit avec le client, par exemple, un navigateur web (Chrome, Mozilla, etc.).
- ✓ Les modules (Plug-in) : Il s'agit de modules complémentaires qui complètent les exigences minimales de l'application et lui confèrent des fonctionnalités supplémentaires.

- ➢ **L'utilisation d'une architecture modulaire présente plusieurs avantages :**
  - ✓ L'architecture modulaire est le meilleur moyen d'ajouter une fonctionnalité à un système qui n'a pas été conçu initialement pour cela.
  - ✓ Cette architecture supprime les limites de quantité de fonctionnalités qu'une application peut offrir. Nous pouvons ajouter une infinité de modules (le navigateur Chrome dispose de centaines de modules appelés extensions).

- ➢ **Il existe également quelques inconvénients majeurs :**
  - ✓ Les modules peuvent être une source de virus et d'attaques venant d'acteurs externes.
  - ✓ La présence de nombreux modules dans une application peut affecter ses performances.

Architecture centrée sur les données

Cette architecture utilise des règles basées sur des tableaux (par opposition aux règles basées sur des codes) pour gérer les différentes catégories et actions à effectuer après un événement donné. Ces tableaux sont appelés tables de contrôle et permettent aux programmes d'être plus simples et plus flexibles.



- **<u>Explication du fonctionnement :</u>**
  - ✓ Le système logiciel (Software System) : Le système développé en utilisant le modèle d'architecture centré sur les données.
  - ✓ Les tableaux de contrôle (Control Tables) : Un ensemble de tableaux qui définissent les actions que le système doit exécuter pour chaque catégorie.
  - ✓ Les catégories (Categories) : Un ensemble de catégories d'un élément dans le système (types de clients, types de produits, types d'articles, types d'employés) utilisé pour exécuter des actions.
  - ✓ Les actions à exécuter (Action Items) : Un ensemble d'actions à exécuter pour une certaine catégorie.
  - ✓ Les règles de gestion (Business Rules) : Les règles de gestion qui déterminent les actions à exécuter en fonction des catégories.

- **<u>L'utilisation d'une architecture centrée sur les données présente plusieurs avantages :</u>**
  - ✓ Si des catégories changent ou sont ajoutées, il n'est pas nécessaire de modifier le code.
  - ✓ Les règles de gestion sont beaucoup plus faciles à gérer dans un tableau que dans un ensemble de programmes.

- **<u>Il existe également quelques inconvénients majeurs :</u>**
  - ✓ Ce schéma peut poser des problèmes de performance car chaque fois que le code s'exécute, il doit rechercher des données dans une table.
  - ✓ Si la table de contrôle est endommagée, l'ensemble du système ne fonctionne plus.

3- Présentons les différents types de web services :

- ✓ Le premier type de Web Service est le service basé sur SOAP (Simple Object Access Protocol). Il s'agit d'un protocole de communication qui utilise XML pour échanger des informations entre les applications. Ce type de service est souvent utilisé pour les applications métier car il permet une communication fiable et sécurisée entre les différents systèmes.

- ✓ Le deuxième type de Web Service est le service basé sur REST (Representational State Transfer). Il s'agit d'un style d'architecture pour les services Web qui utilise le protocole HTTP pour échanger des données entre les applications. Ce type de service est souvent utilisé pour les applications Web car il est plus léger et plus facile à utiliser que le service basé sur SOAP.

- ✓ Le troisième type de Web Service est le service basé sur XML-RPC (Remote Procedure Call). Il s'agit d'un protocole de communication qui utilise XML pour échanger des données entre les applications. Ce type de

service est souvent utilisé pour les applications Web car il est plus léger et plus facile à utiliser que le service basé sur SOAP.