

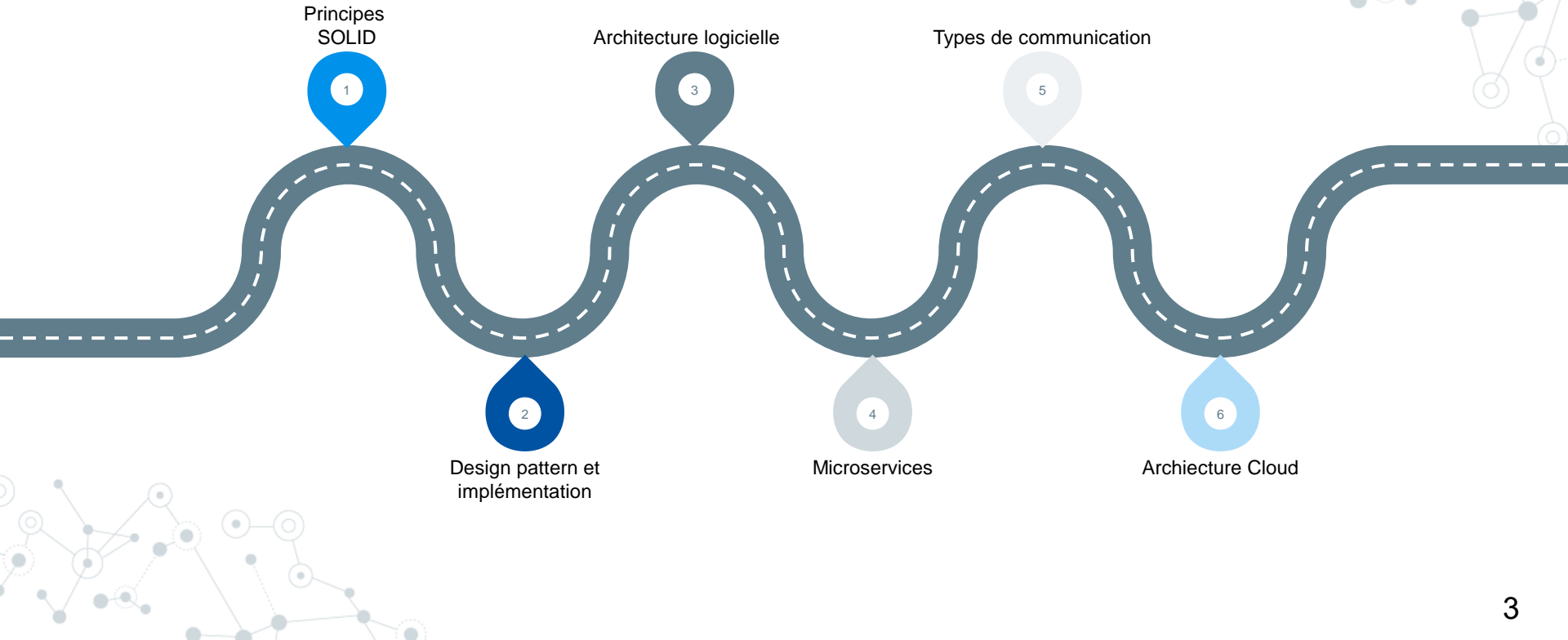




POEC - ARCHITECTURE



L'**architecture logicielle** décrit d'une manière symbolique et schématique les différents éléments d'un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions. Contrairement aux spécifications produites par l'analyse fonctionnelle, le modèle d'architecture, produit lors de la phase de conception, ne décrit pas ce que doit réaliser un système informatique mais plutôt comment il doit être conçu de manière à répondre aux spécifications. L'analyse décrit le « quoi faire » alors que l'architecture décrit le « comment le faire ».

Plan





1. ■ Principes SOLID

SOLID

SOLID est un ensemble de (seulement) 5 bonnes pratiques dont le but est de rendre le code :

- Moins bogué
- Plus facile à lire
- Plus logique
- Maintenable
- Testable
- Extensible





<https://github.com/harrymt/SOLID-principles>

S : **Single Responsibility Principle (SRP)**

Une classe ne doit avoir qu'une seule et unique responsabilité.



Commentaire avec :
Dans ce cas / mais si / sauf quand / ou

O : Open/Closed Principle

Les entités doivent être ouvertes à l'extension et fermées à la modification.



Mots clés :
Instance of / if / switch case



L : Liskov's Substitution Principle (LSP)

Les objets dans un programme doivent être remplaçables par des instances de leur sous-type sans pour autant altérer le bon fonctionnement du programme.



I : Interface Segregation Principle (ISP)

Aucun client ne devrait être forcé d'implémenter des méthodes / fonctions qu'il n'utilise pas.

D : Dependency Inversion Principle (DIP)

Une classe doit dépendre de son abstraction, pas de son implémentation.

FROM STUPID TO SOLID

<https://williamdurand.fr/2013/07/30/from-stupid-to-solid-code/>

2. ■ Patron de conception



Les **patrons de conception** (design patterns) sont des solutions classiques à des problèmes récurrents de la conception de logiciels. Chaque patron est une sorte de plan ou de schéma que vous pouvez personnaliser afin de résoudre un problème récurrent dans votre code.



Classification des patrons de conceptions

Les **Patrons de création** fournissent des mécanismes de création d'objets, ce qui augmente la flexibilité et la réutilisation du code.

Les **Patrons structurels** expliquent comment assembler des objets et des classes en de plus grandes structures, tout en les gardant flexibles et efficaces.

Les **Patrons comportementaux** mettent en place une communication efficace et répartissent les responsabilités entre les objets.



Classification des patrons de conceptions

<https://refactoring.guru/fr>

3. ■ Architecture logicielle



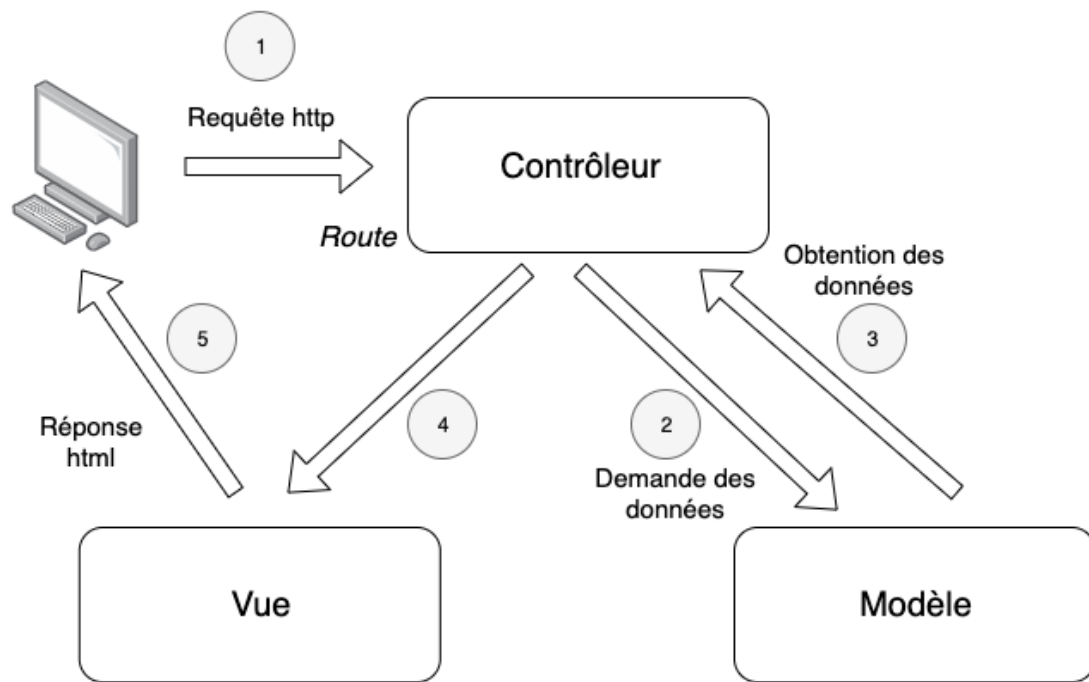
MVC

Model (*modèle*) : gère les données et la logique métier.

View (*vue*) : gère la disposition et l'affichage.

Controller (*contrôleur*) : achemine les commandes des parties "model" et "view".





MVC en dotnet

dotnet new mvc (razor)

dotnet new blazorserver



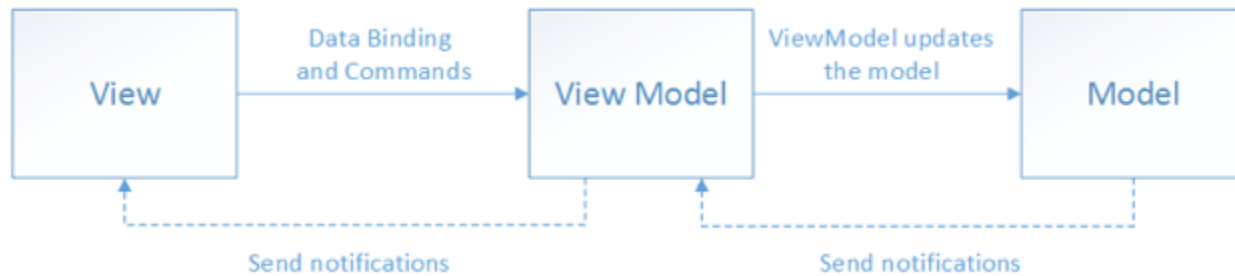
DEMO

MVVM

WPF et MVVM

<https://docs.microsoft.com/fr-fr/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>

MVVM



Binding

Liaison de donnée entre deux propriétés

Commandes

Délègue une action

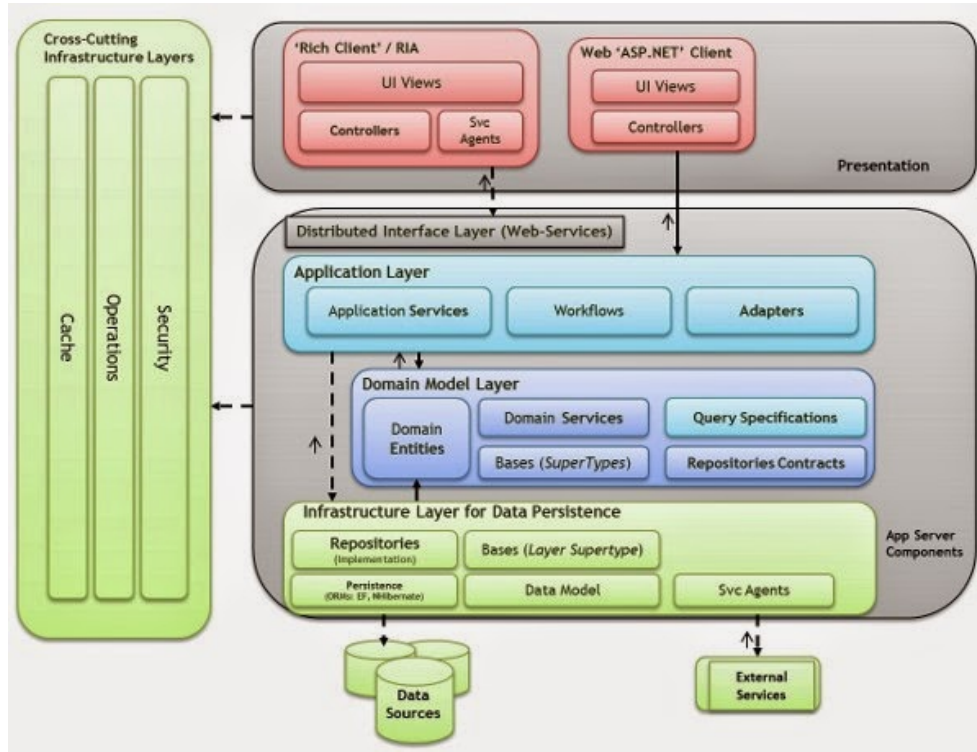
Notifications

La notification permet de dire à la vue qu'un élément change et par conséquent qu'un rafraichissement est nécessaire





DEMO

N layer app

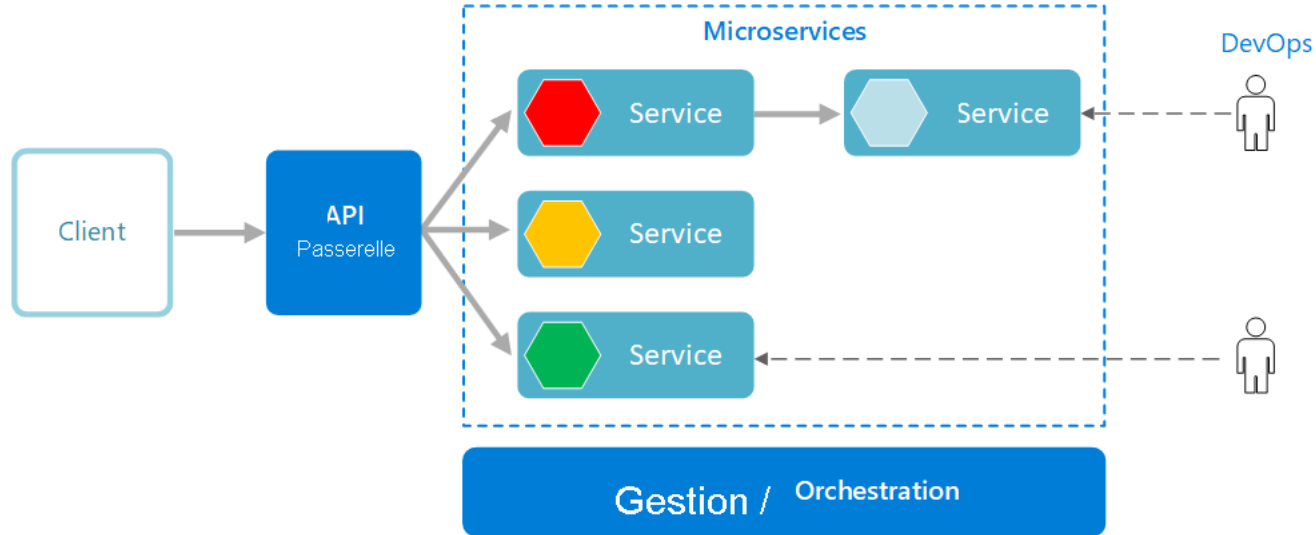


Domain Driven Design



4. ■ Monolithe vs Microservices



Microservices



<https://docs.microsoft.com/fr-fr/azure/architecture/guide/architecture-styles/microservices>

Résilience

Caching



5. ■ Types de communication

Synchrone vs Asynchrone



5. ■ Architecture Cloud

IASS / PASS / SASS

Serverless