// Architecture Logicielle

ARCHITECTURE LOGICIELLE

Construisez des systèmes logiciels puissants avec une architecture intelligente

Térence FERUT

Plan Global

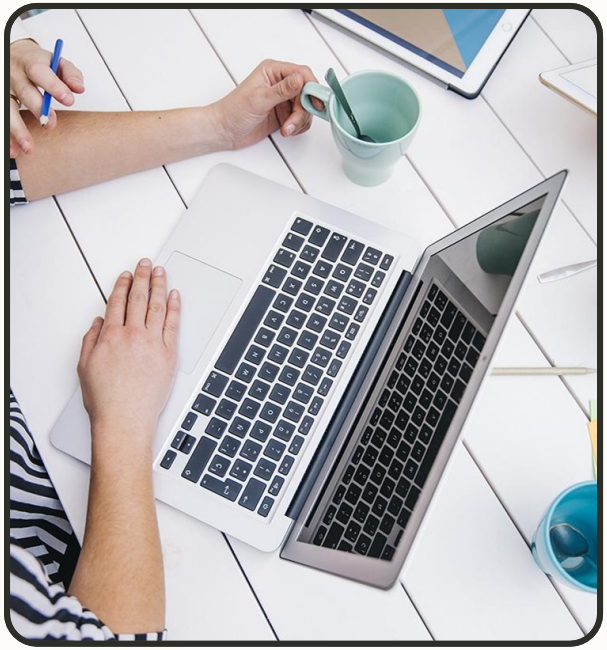
01 Introduction à l'architecture logicielle

02 Styles d'architecture logicielle

03 Design Patterns dans l'architecture logicielle

04 Autres approches et grands principes

Plan Global



Introduction à l'Architecture Logicielle

La clé d'un logiciel performant : une architecture logicielle bien pensée

Objectifs du cours

Savoir proposer une architecture logicielle répondant à la demande du client

Savoir mesurer la qualité d'un logiciel selon des critères définis

Comprendre la mise en place de modèles d'architecture logicielle et des styles architecturaux

Comprendre les principaux Design Pattern et les mettre en place

Evaluations

Coef 1 x4

QCM

4 QCM répartis tout au long du cours, en début de séances.

Coef 2

Projet final

Le rendu de votre projet à faire en équipe de 4 à 5

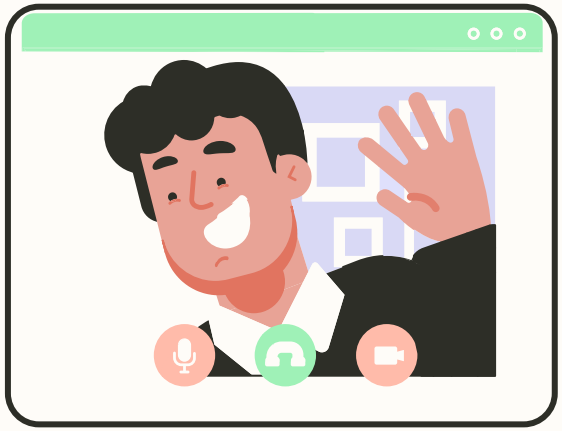
Evaluations

Coef 1 x???

Notes de TP

Un ou plusieurs TP peuvent être notés durant le semestre. Vous devrez donc tous les rendre en temps et en heure.

DISCLAIMER



Tout au long de ce cours, nous verrons énormément d'exemples tirés de ces dernières décennies. Nous ne ferons pas de focus sur un langage en particulier mais au contraire nous discuterons de nombreux langages et technologies différents.

Aperçu du cours

- Ce premier cours a vocation à vous donner un aperçu de ce qu'est l'architecture logicielle et à quels besoins elle répond.
- **Il ne s'agit donc que d'un survol de points sur lesquels nous reviendrons plus longuement dans les cours à venir.**

#01

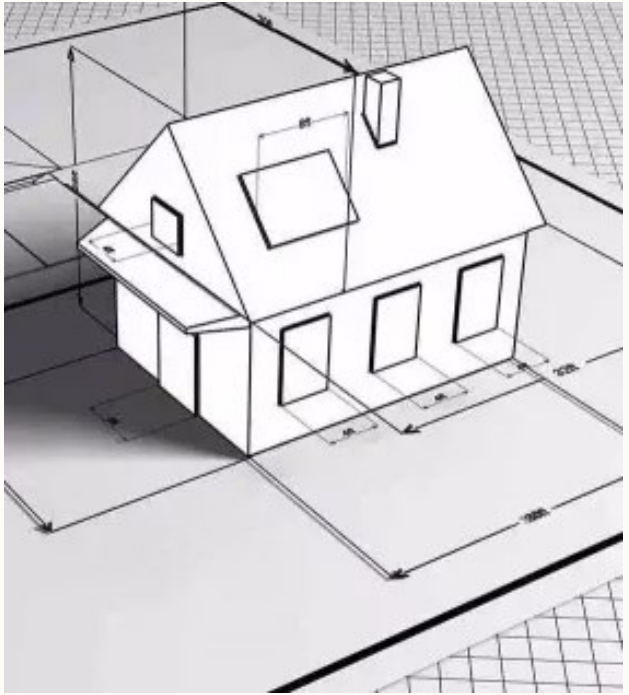
Introduction à l'architecture logicielle

1.1.1 Définition de l'architecture logicielle ✖

- L'architecture logicielle est une représentation organisationnelle de la structure d'un système informatique.
- Son rôle est central dans le développement de logiciels, puisqu'elle définit les composants du système, leurs interactions et leur coordination pour répondre aux exigences du logiciel.

1.1.1 Définition de l'architecture logicielle ✖

- *Pensez aux plans d'un architecte pour une maison, précisant où chaque élément (comme les fenêtres, les portes, les escaliers) doit aller et comment ils interagissent entre eux.*



1.1.1 Définition de l'architecture logicielle ✖

✖

- Les précurseurs de l'architecture logicielle remontent aux années 1960 et 1970, lorsque des chercheurs comme Edsger Dijkstra et David Parnas ont commencé à définir les principes de la structuration des systèmes logiciels.
- Ils ont été les pionniers de concepts tels que la modularité et l'encapsulation qui sont devenus les piliers de la POO

1.1.1 Définition de l'architecture logicielle ✖

✖

- L'architecture logicielle est essentielle pour assurer la qualité, la maintenabilité et l'évolutivité d'un système.
- Une bonne architecture facilite la compréhension du système, sa modification et son évolution.
- Elle contribue à prévenir les bugs et facilite les tests et la validation du logiciel.

1.1.2 Évolution de l'architecture logicielle

- L'architecture logicielle a beaucoup évolué depuis ses débuts.
- Au cours des années 1980, l'approche orientée objet a introduit de nouvelles façons de structurer les systèmes.
- Plus récemment, des approches comme les architectures orientées services (*SOA*) et les microservices ont été développées pour gérer la complexité croissante des systèmes logiciels.

1.1.2 Évolution de l'architecture logicielle✖

✖

- Plusieurs approches architecturales ont influencé le domaine, comme les architectures en couches, les architectures centrées sur les événements, ou encore l'architecture *clean*.
- Chacune a apporté de nouvelles façons de penser la structuration et l'organisation des systèmes.

1.1.3 Objectifs de l'architecture logicielle

- Une bonne architecture logicielle vise à faciliter la réalisation des exigences fonctionnelles et non fonctionnelles d'un système, tout en assurant sa qualité, sa réutilisabilité, sa maintenabilité, sa scalabilité et sa sécurité.
- Elle aide à prévenir les problèmes et facilite l'évolution du logiciel.

1.1.3 Objectifs de l'architecture logicielle ✖

✖

- L'architecture logicielle joue un rôle clé dans l'amélioration de la qualité des logiciels en favorisant la cohérence, en facilitant les tests et en réduisant les risques d'erreurs.
- Elle contribue également à la réutilisabilité du code, à la maintenabilité du système, à sa capacité à évoluer (*scalabilité*) et à sa sécurité.

1.1.4 Rôles et responsabilités de l'architecte

- L'architecte logiciel joue un rôle clé dans le processus de développement.
- Il définit l'architecture du système, s'assure que les choix architecturaux sont bien compris et suivis par l'équipe de développement, et vérifie que l'architecture répond aux exigences du système.

1.1.4 Rôles et responsabilités de l'architecte

Exercice - Réflexion

- *Selon vous, quelles sont les missions de l'architecte logiciel dans une équipe de développement ?*

1.1.4 Rôles et responsabilités de l'architecte

- L'architecte logiciel doit posséder de solides compétences techniques, une bonne compréhension des exigences du système et une vision globale de la structure du système.
- Il doit aussi être capable de communiquer efficacement avec les autres membres de l'équipe.

1.1.4 Rôles et responsabilités de l'architecte

- L'architecte logiciel doit en effet travailler en étroite collaboration avec les autres membres de l'équipe de développement, mais aussi avec les utilisateurs, les testeurs, et les autres parties prenantes du projet.

1.1.5 Approches et méthodologies

- Chaque approche, que nous nommerons par la suite "*style architectural*" a ses propres avantages et inconvénients.
- *Nous verrons en détail les principaux styles architecturaux dans la suite du cours.*

1.1.5 Approches et méthodologies

Exemples

- *Par exemple, l'approche orientée services (SOA) favorise la réutilisabilité et l'interoperabilité, mais peut être complexe à mettre en œuvre.*
- *L'architecture en microservices offre une grande flexibilité et scalabilité, mais nécessite une gestion attentive des interactions entre services.*

1.1.6 Outils et technologies

- De nombreux outils et technologies sont disponibles pour concevoir, documenter et communiquer une architecture logicielle.
- Ils vont des outils de modélisation comme UML (*Unified Modeling Language*) aux outils de visualisation et de documentation.

1.1.6 Outils et technologies

Exemples

- *Par exemple, UML est largement utilisé pour modéliser l'architecture d'un système.*
- *Des diagrammes d'architecture peuvent être utilisés pour visualiser la structure du système et documenter les choix architecturaux.*


#02
Principes fondamentaux de l'architecture logicielle
15, 2021 - 18H
1 - 15H
JU
JULY 11, 2
AUGUST 8, 2021 -
RIL 15, 2021 - 15H
ting with Company A

1.2.1 Séparation des préoccupations

- La séparation des préoccupations est un principe qui vise à organiser un logiciel en parties distinctes, chacune ayant une responsabilité unique.
- Cela favorise une architecture claire et permet de réduire les risques d'erreurs.

1.2.1 Séparation des préoccupations

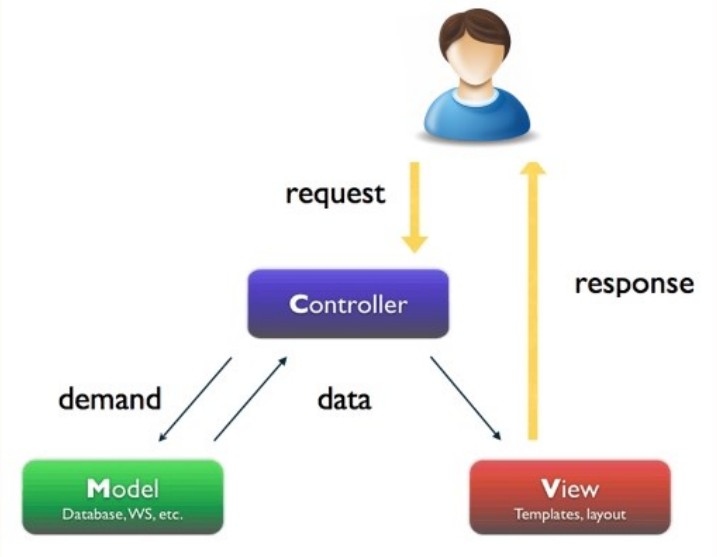
Exemples

- *Par exemple, dans une application web typique, la présentation (le front-end) est séparée de la logique métier (le back-end).*
- *Dans l'histoire du développement web, cette séparation a pris de l'importance avec l'émergence du modèle MVC (Modèle-Vue-Contrôleur) dans les années 1970 par Trygve Reenskaug au Palo Alto Research Center (PARC) dans la Silicon Valley.*

1.2.1 Séparation des préoccupations

Exercice - Réflexion

- *A l'aide de quelques mots, indiquez comment le MVC illustre le principe de séparation des préoccupations.*



1.2.2 Modularité

- La modularité est la capacité à diviser un système en modules ou composants distincts.
- Chaque module réalise une tâche spécifique et peut fonctionner de manière indépendante des autres.
- Cela améliore la maintenabilité, la testabilité et la réutilisabilité du code.

1.2.2 Modularité

Exemples

- *Par exemple, dans le système d'exploitation Unix (créé en 1969), chaque programme est conçu pour réaliser une tâche simple et bien définie, illustrant une forte modularité.*

1.2.2 Modularité

Exercice - Réflexion

- *Décrivez à l'aide d'un exemple inventé comment la modularité peut aider à gérer la complexité d'un grand projet logiciel.*

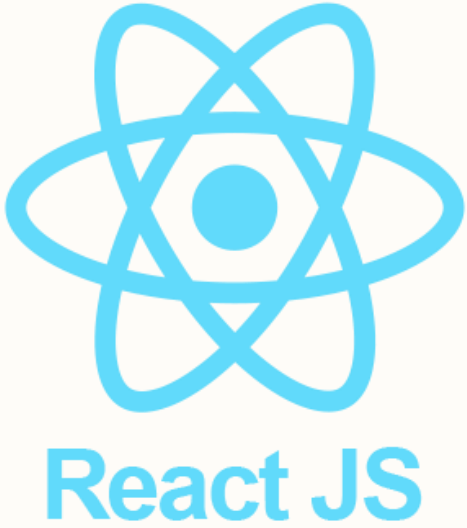
1.2.3 Réutilisabilité

- La réutilisabilité est la capacité à utiliser un même composant logiciel dans plusieurs contextes.
- Elle permet de gagner du temps et de réduire les coûts de développement.

1.2.3 Réutilisabilité

Exemples

- *Par exemple, les bibliothèques de fonctions ou les frameworks, comme React (créé par Facebook en 2013), permettent de réutiliser des composants pour accélérer le développement.*



1.2.3 Réutilisabilité

Exercice - Réflexion

- *Trouvez un exemple de composant que vous pourriez réutiliser dans différents projets et expliquez comment cela pourrait vous aider.*

1.2.4 Extensibilité

- L'extensibilité est la capacité d'un système à ajouter de nouvelles fonctionnalités ou à modifier les existantes avec facilité.
- Cela facilite l'évolution du logiciel pour répondre aux nouveaux besoins.

1.2.4 Extensibilité

Exemples

- *Par exemple, le navigateur web Chrome de Google, lancé en 2008, a été conçu pour être extensible grâce à son système de modules complémentaires ou extensions.*

1.2.5 Cohérence

- La cohérence en architecture logicielle se réfère à l'uniformité et à la conformité à une structure ou à des conventions communes.
- Cela facilite la lecture et la compréhension du code.

1.2.5 Cohérence

Exemples

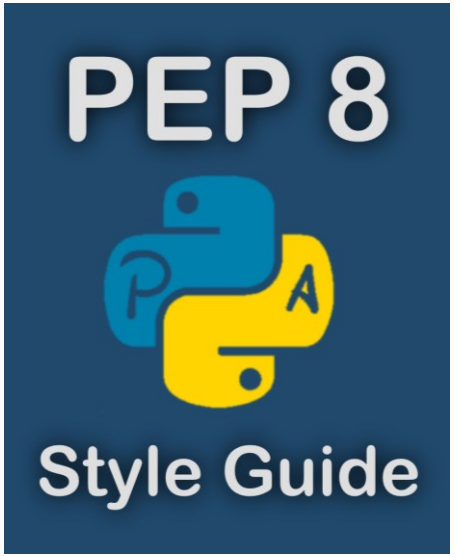
- *Par exemple, l'utilisation de conventions de codage et de normes de nommage, comme le PEP8 pour Python, introduit en 2001, contribue à la cohérence du code.*

1.2.5 Cohérence

Exercice - Réflexion

Revenons sur l'exemple de Python et PEP8.

Pourquoi, selon vous, ces conventions sont-elles importantes pour maintenir la cohérence ?



Aperçu du cours suivant

- Cette introduction terminée, nous sommes prêts à passer à l'étape suivante – celle de la découverte des principaux styles d'architecture.
- **Nous passerons en revue près d'une dizaine de styles d'architectures et analyserons à chaque fois leurs points forts et leurs faiblesses.**

Merci pour votre attention !

Avez-vous des questions ?

contact@astroware-conception.com

www.astroware-conception.com

www.linkedin.com/in/terence-ferut/