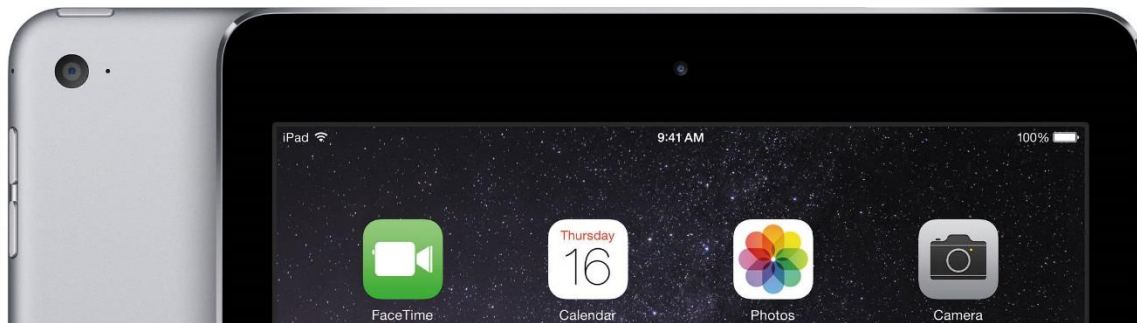


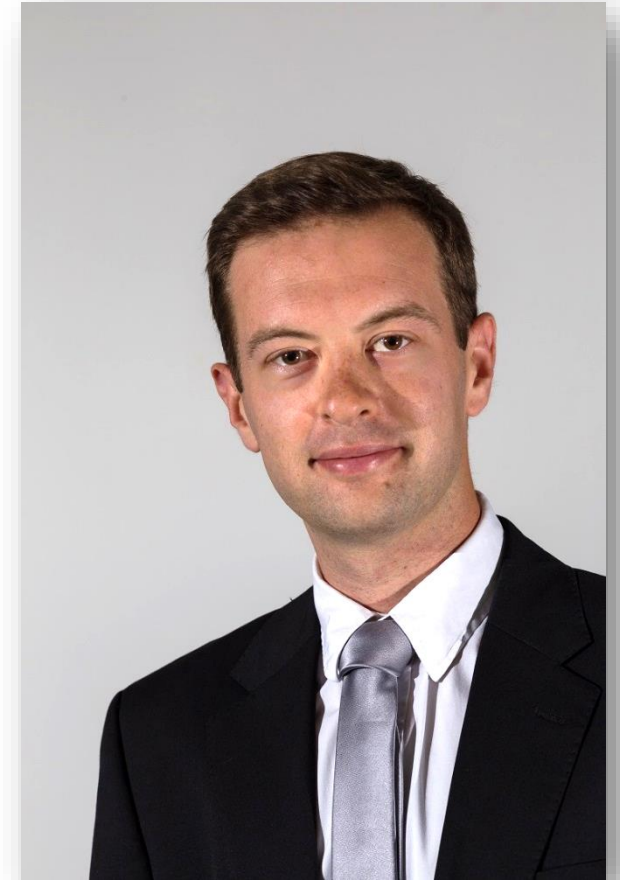
Environnements Logiciels pour l'Informatique Mobile

Patterns Logiciels associés au développement sur terminaux mobiles



Présentation

- Polytech'Nice-Sophia 2012 (IAM)
- 6 ans chez Sopra-Steria
 - Développement
 - Architecture
 - Projets innovants
 - Formation (sécurité)
- gregory.marro@soprasteria.com



Sommaire

1. Particularités du développement mobile
2. Les patterns du développement mobile
3. TD

Particularités d'une application mobile

4 millions d'applications (iOS + Android)



Time to market réduit

Durée de vie d'une application

Nécessité des mises à jour

Concurrence



Coût moyen d'une application :

< 1 \$

Chaque accès doit être validé par
l'utilisateur

Problématique de la vie privée

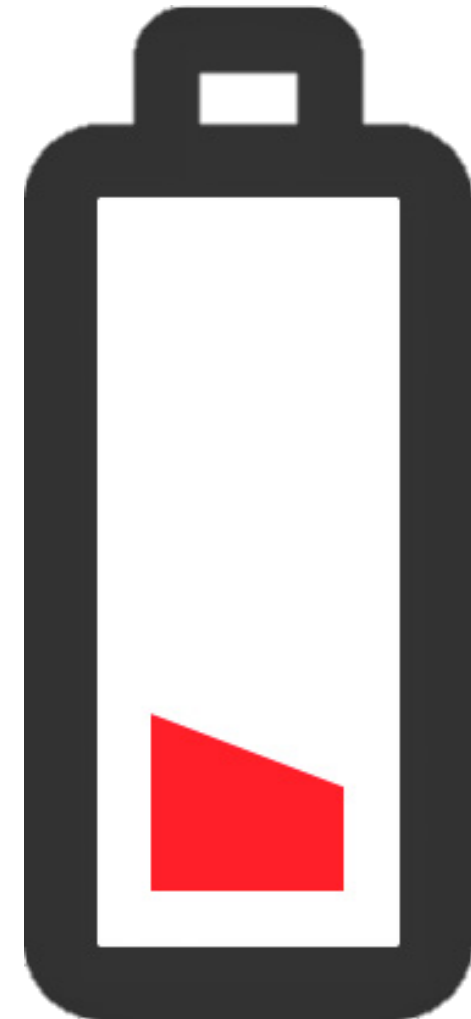
Protection des données



1^{ère} préoccupation des usagers

Cause importante de désinstallation

Principal critère de sélection

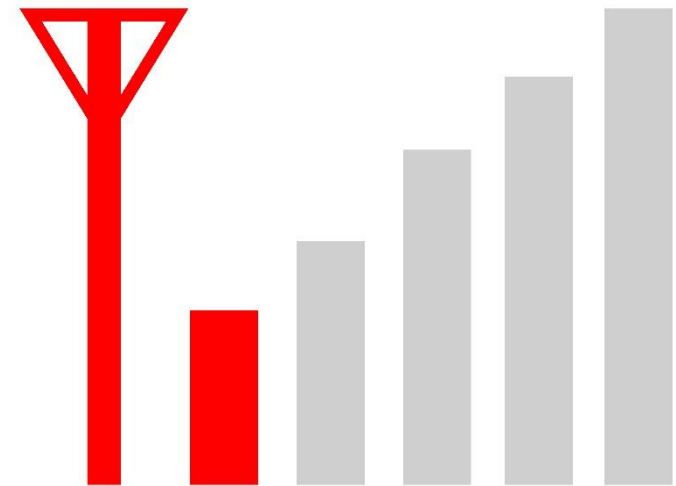


85% du territoire en 3G

25% en territoire en 4G

Mais ...

100% de disponibilité pour une application



Performances limités

Stockage limité

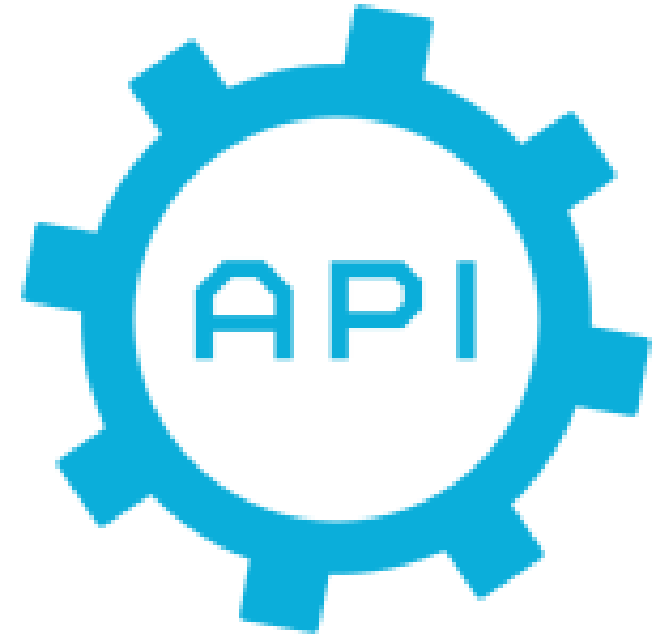
Mais ...

0 latence tolérée



Certains éléments ne sont pas accessibles

Attention particulière aux « wearables »



Les patterns du développement mobile

Obligatoire ?

- Indispensable à long terme
- Garantie de fonctionnement à cours terme
- Ouvert aux évolutions
- Adaptée à la plateforme

Distinguer les types de patterns

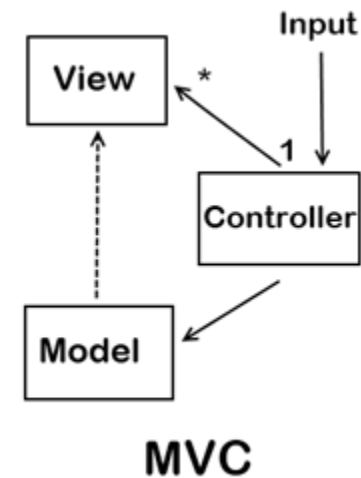
- Patterns dédiés aux interfaces
- Patterns architectures :
 - de construction
 - structuraux
 - comportementaux

TD : Présentation des patterns

- Par groupe de 3
- 45 minutes de préparation (slides)
- 5 minutes de présentation (type, quand, pourquoi, comment)
- Présenter un pattern parmi cette liste :
 - MVC/MVVM/MVP
 - Adapter
 - Factory
 - Observer
 - Decorator
 - Iterator
 - MVP
 - Proxy

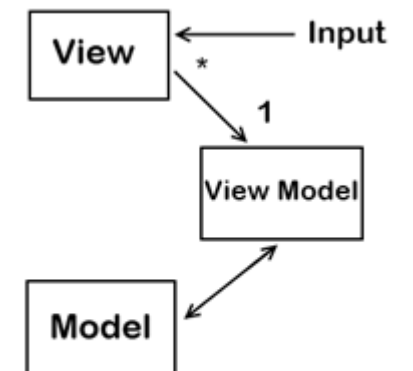
MVC

- Type : Structure
 - Permet de découplage entre les données et la couche de présentation
 - Données = Modèle
 - Présentation = Vue
 - Traitements = Contrôleur
1. la vue envoie une requête qui est analysée par le contrôleur
 2. le contrôleur appelle la méthode voulu du modèle et notifie à la vue que la requête est traitée
 3. la vue notifiée fait une requête au modèle pour se mettre à jour



MVVM

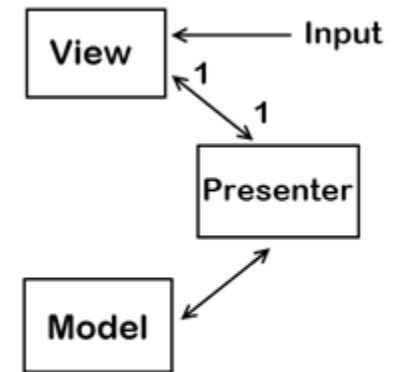
- Type : Structure
- Permet de découplage entre les données et la couche de présentation
- Très proche de MVC/MVP sauf :
 - Le ViewModel peut gérer plusieurs View
 - Aucun lien direct entre la View et le Model



MVVM

MVP

- Type : Structure
- Permet de découplage entre les données et la couche de présentation
- Très proche de MVVM/MVP sauf :
 - Le Presenter ne gère qu'une seule View
 - Aucun lien direct entre la View et le Model

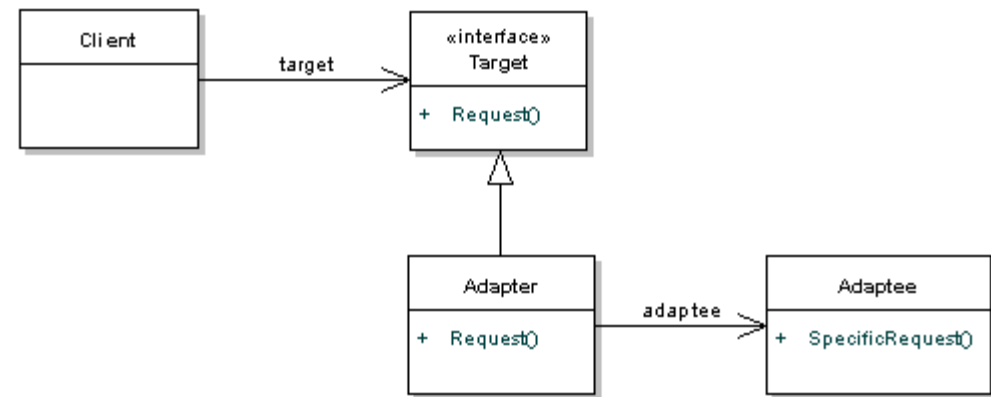


MVP

Adapter

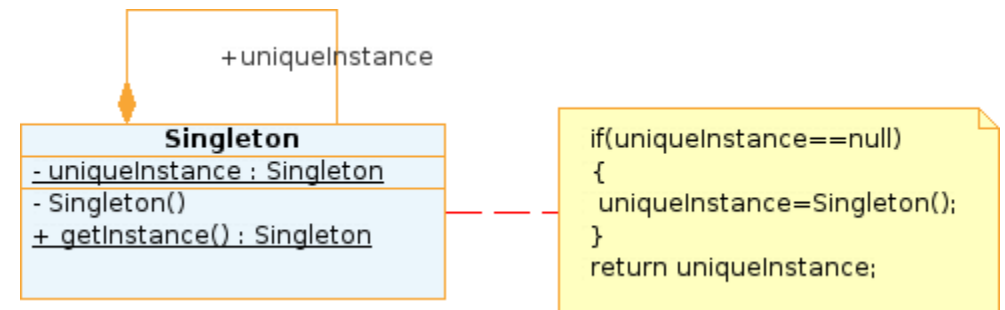
- Type : Structure
- Permet de faire la communication entre des objets qui ne sont pas prévus pour

- Utilisation :
 - Réutiliser d'anciennes API
 - Permettre la communication entre des applications non compatibles



Singleton

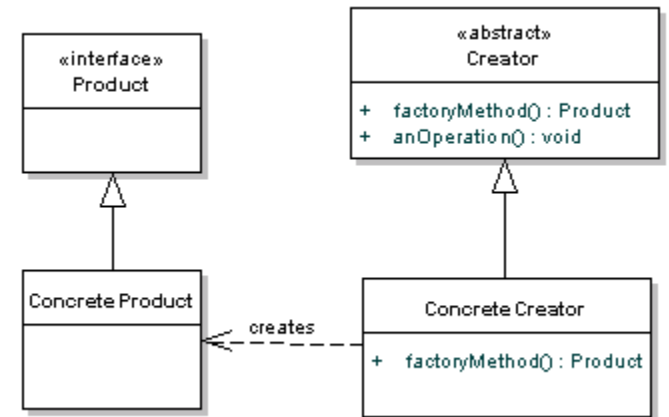
- Type : Construction
- Permet de limiter une classe à une seule et unique instance
- Utilisation :
 - Autre pattern
 - Log
 - ...



- Attention à l'utilisation en environnement multi-thread

Factory

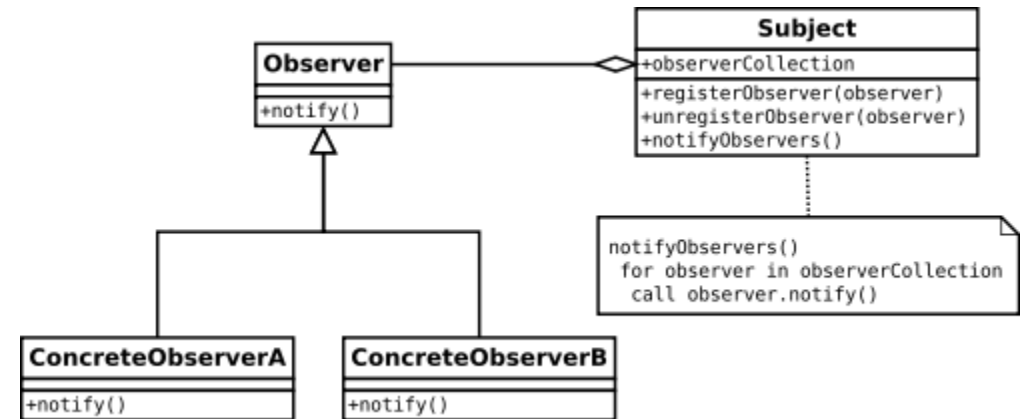
- Type : Construction
- Permet de gérer l'instanciation des objets
- Utilisation :
 - découpler les clients des classes concrètes à instancier
 - Si on ne connaît pas d'avance toutes les classes concrètes à instancier.



Observer

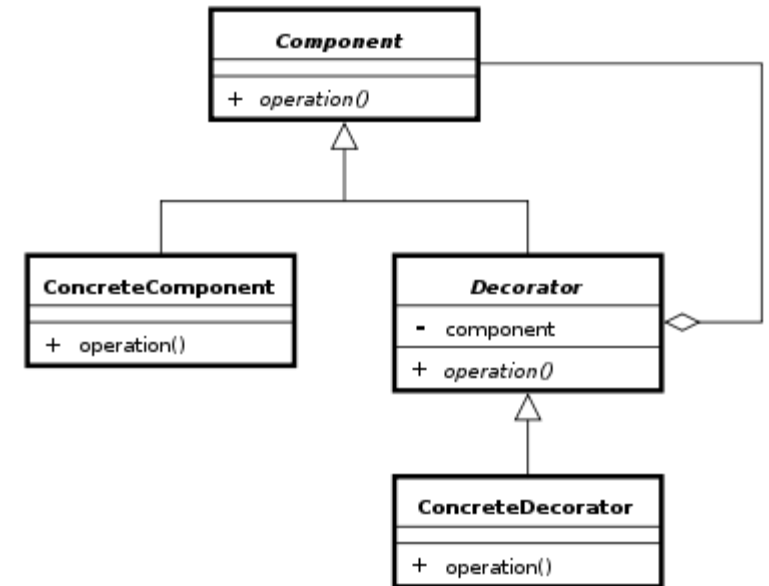
- Type : Comportement
- Permet à une collection de classe d'être notifié au changement d'état

- Utilisation :
 - Autres patterns (MVC)
 - Liaisons dynamiques entre classes



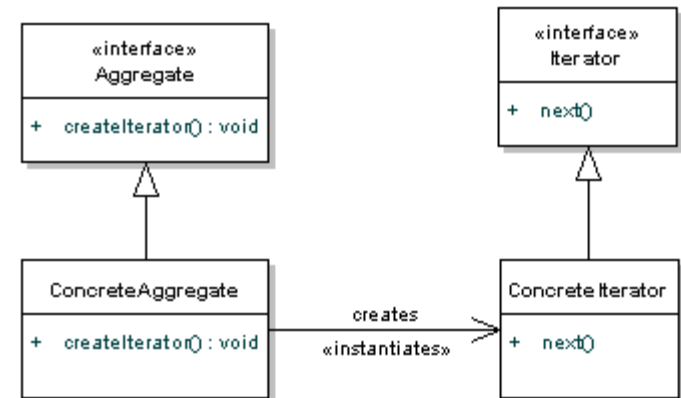
Decorator

- Type : Structure
- Permet d'ajouter des fonctionnalités à une classe sans avoir à faire de l'héritage
- Utilisation :
 - Ajouter des fonctionnalités dynamiquement
 - Limiter l'utilisation de l'héritage



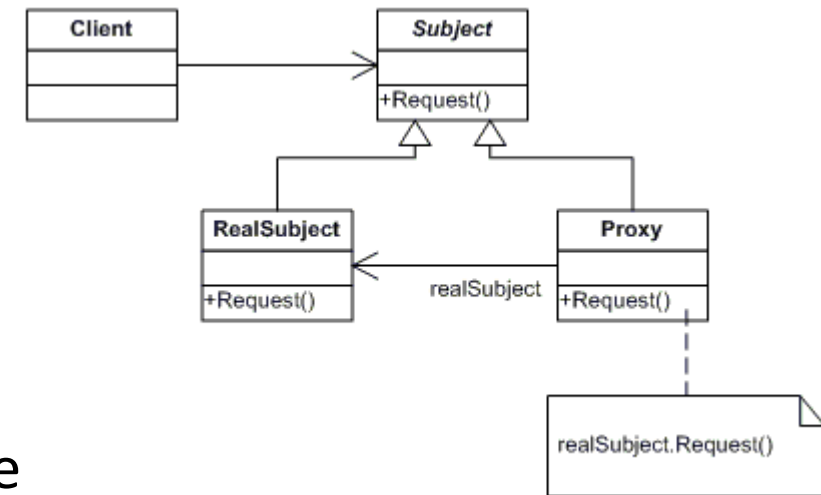
Iterator

- Type : Comportement
- Permet de parcourir une collection d'éléments
- Utilisation :
 - Collections d'objets quelconques
 - Restrictions d'accès aux éléments



Proxy

- Type :
- Permet de cacher une classe par un proxy (chargement lourd)



- Utilisation :
 - Gestion d'accès aux méthodes de la classe substituée
 - Simplification de l'utilisation d'un objet « complexe » (objet à distance ou si l'objet est consommateur de temps)