

Architecture logicielle

Formalisme des trois vues

Florian Baune - 2026



Objectifs

- Savoir **choisir** la vue qui porte un problème
- Pouvoir **dessiner** une vue avec un formalisme simple
- Faire apparaître ce qu'un élément **n'a pas le droit** de faire

Rappel du chapitre 1

Vue	La question	On y voit
Code	Comment le code est découpé ?	Modules, dépendances
Exécution	Que se passe-t-il quand ça tourne ?	Processus, appels, flux
Déploiement	Où est-ce que ça tourne ?	Machines, réseaux



Vues simplifiées

Vues simplifiées

Toutes les vues ne sont pas utiles

On choisit **celle(s) qui porte(nt) le(s) problème(s)**.

C'est souvent en dessinant une vue que l'on se rend compte de son utilité (ou pas !).

Cas de figure

Ce qu'on a sous les yeux

La vue qui le raconte

Un incident **d'enchaînement**

exécution

Un **chantier** de refonte

code

Un incident **d'infra**

déploiement

Enchaînement : chaque élément répondait. C'est leur **dialogue** qui a échoué.

Vue code — diagramme de paquets

La question : qui a le droit d'appeler qui ?

- une **boîte** = un module
- une **flèche pleine** = « dépend de »
- une **flèche barrée** = dépendance **interdite**

Le piège : dessiner des classes. Une boîte = un fichier → c'est du design.

Vue code — une boîte peut en contenir une autre

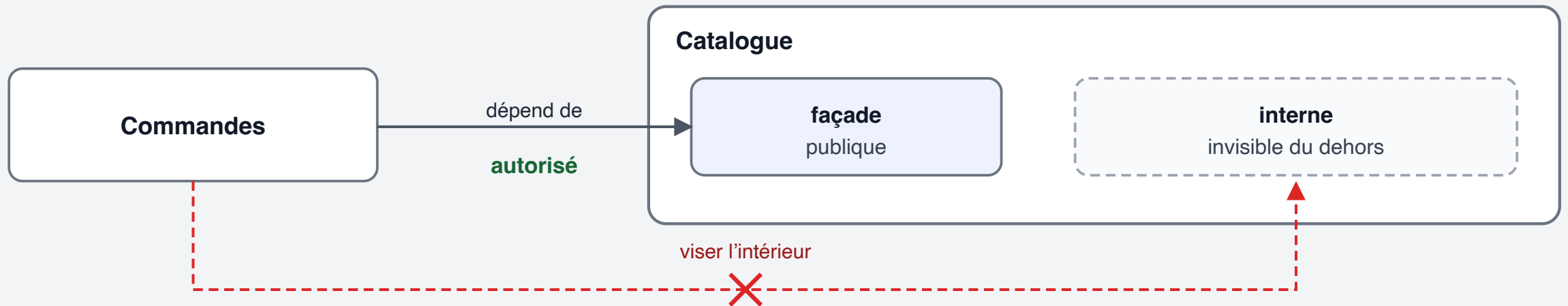
Un module et ses couches. Une interface dans son module.

Une flèche venue du dehors peut **viser une boîte intérieure** — autorisée sur la façade, interdite sur l'intérieur.

Pas un symbole de plus : une **règle d'usage**.

Formalisme — diagramme de paquetages

Vue code (modules)



Une boîte peut en contenir d'autres : un module et ses couches, une interface dans son module.

Une flèche venue du dehors peut viser une boîte intérieure — et c'est là que la règle se joue.

La règle : si tout dépend de tout, l'illisibilité est le diagnostic

Vue exécution — diagramme de séquence

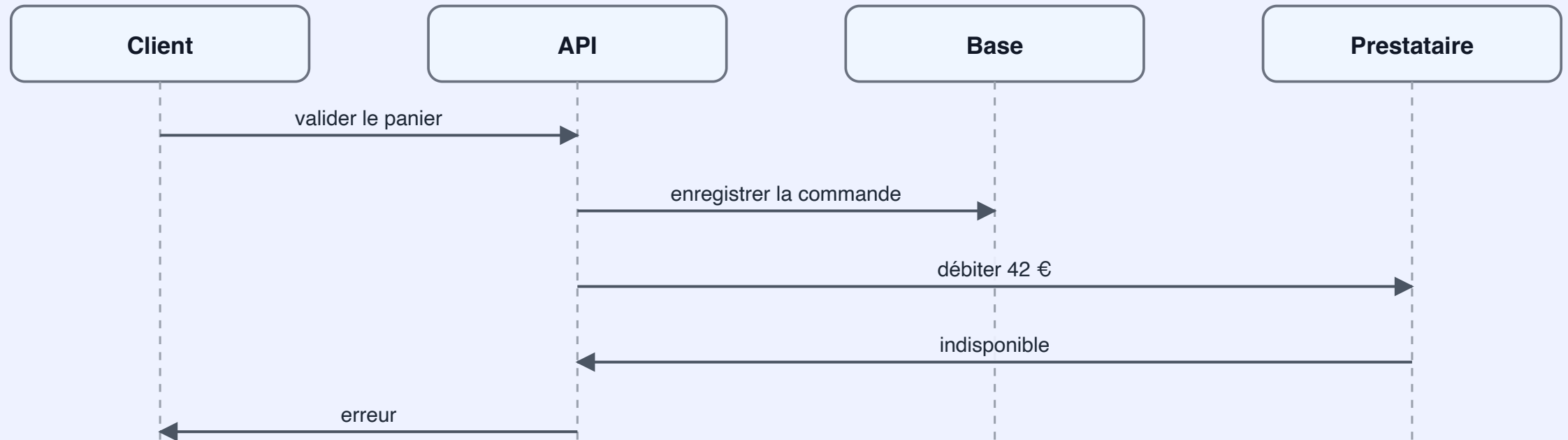
La question : que se passe-t-il, et dans quel ordre ?

- un **participant en colonne** = quelque chose qui **tourne**
- une **flèche horizontale** = un message
- du **premier message au dernier**

Chaque flèche porte une **étiquette**. Une flèche nue ne dit rien.

Formalisme — diagramme de séquence

Vue exécution (composants et connecteurs)



Chaque flèche porte une étiquette : le nom du message. Une flèche nue ne dit rien.

La règle : un seul scénario par diagramme — le temps descend

Vue exécution — Diagramme de flux

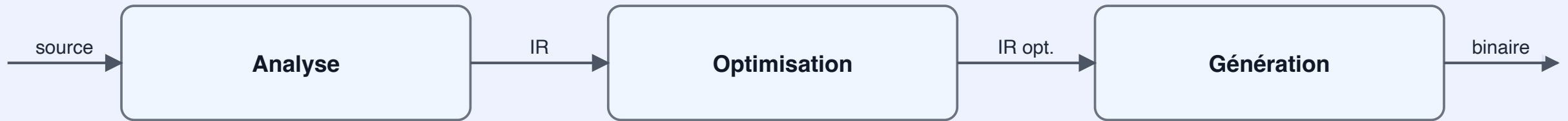
Une **chaîne de traitements** se raconte mal en colonnes :
il n'y a pas de dialogue, seulement des entrées et des sorties.

Second formalisme pour la même vue : le **diagramme de flux**.

- une **boîte** = un traitement
- une **flèche** = une donnée qui circule
- l'**étiquette** = le nom de cette donnée

Formalisme — diagramme de flux (pour une chaîne de traitements)

Vue exécution (composants et connecteurs)



Mêmes symboles que le diagramme de paquetages — mais ici la flèche dit « une donnée circule », et non « dépend de ». C'est l'étiquette de donnée qui distingue les deux schémas à l'œil.

Quand l'employer : une suite d'étapes avec des entrées et des sorties, que la séquence rend mal.

La règle : chaque flèche porte le nom de la donnée qui circule

Flux ou paquetages ?

Les deux schémas ont la même allure : des boîtes et des flèches.

	La flèche dit	On y lit
Paquetages	« dépend de »	qui connaît qui
Flux	« une donnée circule »	ce qui traverse

C'est l'**étiquette de donnée** qui les distingue à l'œil.
Sans elle, le schéma est ambigu.

Vue déploiement — diagramme de déploiement

La question : où est-ce que ça tourne, et sur quoi ?

- une **boîte étiquetée** = une machine, un service hébergé, un lieu
- une **boîte à l'intérieur** = ce qui y tourne
- un **trait** = un lien réseau

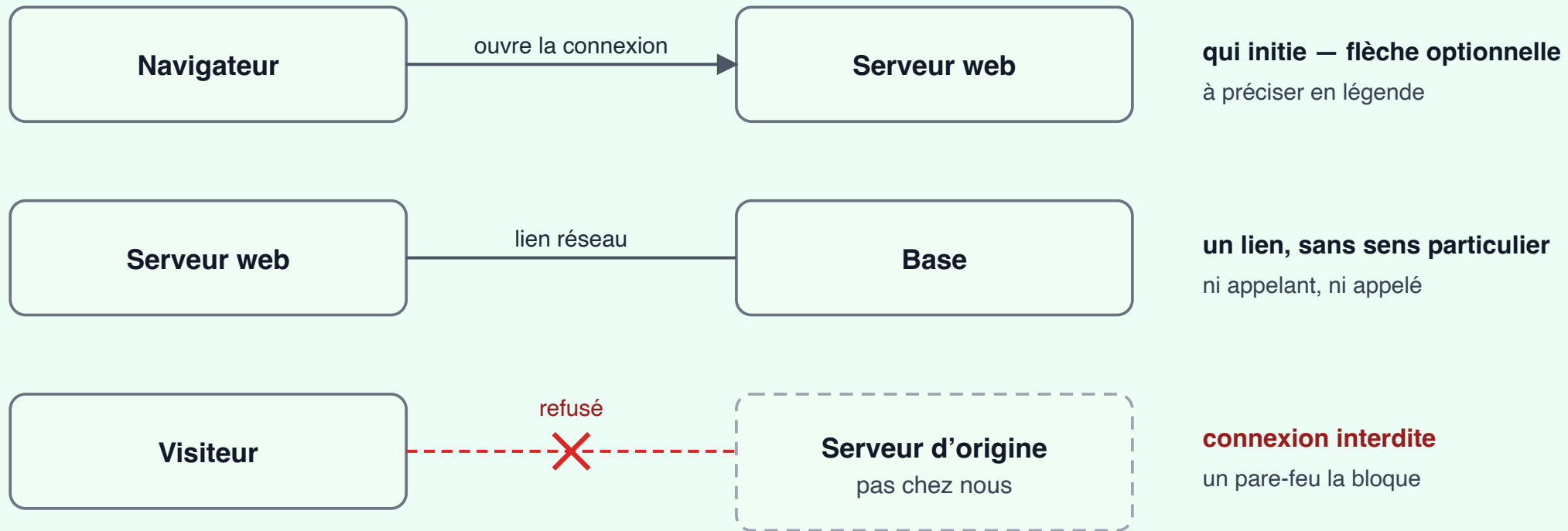
Vue déploiement — trois sortes de trait

- un **trait simple** = un lien réseau, sans sens particulier
- un **trait fléché** = qui **initie** la connexion (*optionnel, à préciser en légende*)
- un **trait barré** = une connexion **interdite**

Pourquoi le trait barré : l'absence de trait ne se distingue pas d'un oubli.

Formalisme — diagramme de déploiement

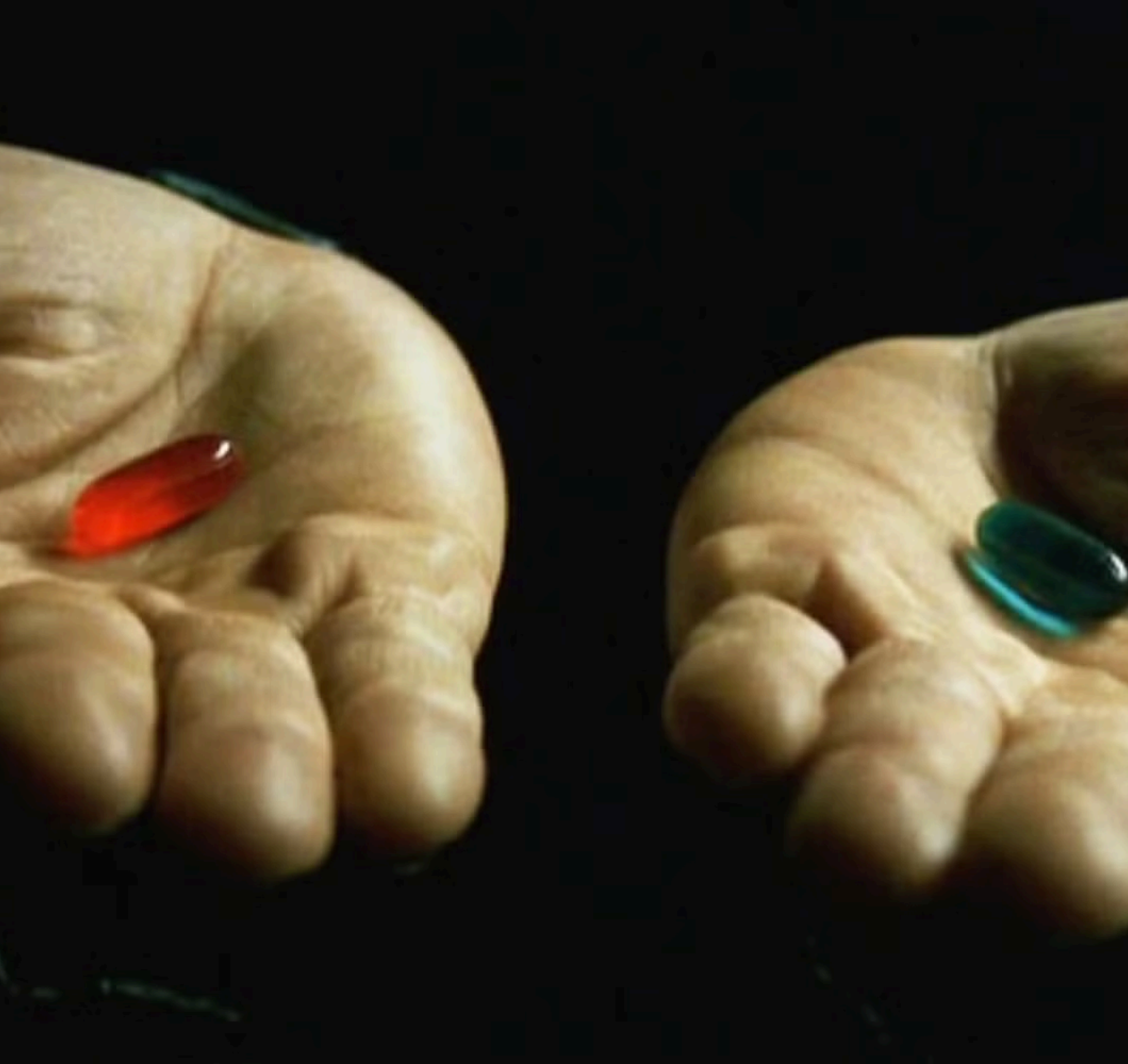
Vue déploiement (allocation)



La règle : ce qui n'est pas chez nous se dessine en pointillés

UML Simplifié

UML propose beaucoup plus **de symboles** par diagramme. On n'approfondira pas ces formalismes dans ce cours.



Choisir sa vue

Une vue est un instrument de diagnostic

Elle aide à répondre à une question posée.

Vue code - questions à se poser

- Qui a le droit d'appeler qui ?
- Où passe la frontière entre deux parties ?
- Qu'est-ce qui m'empêche de faire ce changement ?
- **Indices :**
 - on parle de code, de modules, de paquets
 - difficultés à maintenir ou à faire évoluer

Vue code — un exemple

Une application de réservation, six ans d'existence.

Ajouter un champ « nombre de bagages » au formulaire demande **trois semaines** : il traverse la réservation, la facturation et l'export comptable, et chacun connaît l'intérieur des deux autres.

Pas de crashes. C'est seulement de plus en plus long à faire évoluer.

Vue exécution - questions à se poser

- Que se passe-t-il, et dans quel ordre ?
- Qui attend qui ?
- Que se passe-t-il si celui d'en face ne répond pas ?
- **Indices :**
 - le récit est une chronologie : « d'abord..., puis..., et... »
 - les mots *attendre, appeler, délai, relance, cascade*
 - chaque élément, pris séparément, fonctionnait

Vue exécution — un exemple

Un site de vente. Pour valider un panier, il enregistre la commande, appelle la banque, et **attend** sa réponse avant d'afficher la confirmation.

Un soir, la banque répond en 40 secondes au lieu de 200 millisecondes. Provoquant une erreur type *timeout*.

Rien n'est en panne : les serveurs tournent, la base répond. Et pourtant **plus personne ne peut commander**.

Vue déploiement - questions à se poser

- Où est-ce que ça tourne, et sur quoi ?
- Qu'est-ce qui tombe en même temps que quoi ?
- Si cette machine disparaît ce soir, que reste-t-il ?
- **Indices :**
 - on parle de machines, de régions, de sauvegardes
 - quelque chose a **disparu** physiquement
 - deux choses qu'on croyait séparées étaient au même endroit

Vue déploiement — un exemple

Une entreprise loue **un** serveur. Dessus tournent l'application, la base de données, et la sauvegarde de la nuit, écrite dans un dossier du même disque.

Un matin, le disque lâche.

Le code n'a aucun défaut. Il n'a simplement plus de machine — ni de sauvegarde.

Philosophie

- Une vue doit sert avant tout à **comprendre / clarifier**
- Elle n'a **pas** pour vocation d'être **exhaustive**
- Cela peut aussi servir d'outil de **diagnostique**

Architecture logicielle

Formalisme des
trois vues

Florian Baune - 2026

