



Conception en UML, Architecture n-tiers, par l'exemple

Utilisation de php 5, Mysql, Html, css, ...

Inspiré de UML2 par la pratique

M. Blay-Fornarino

Les codes sont disponibles sur le site web

Bibliographie

- ❧ «Why MVC is not an application architecture» Stefan Pribsch, the PHP.cc ZendCon 2010
- ❧ Developing Web Applications with PHP, RAD for the World Wide Web,



Approche



Analyse

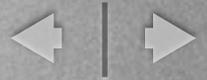
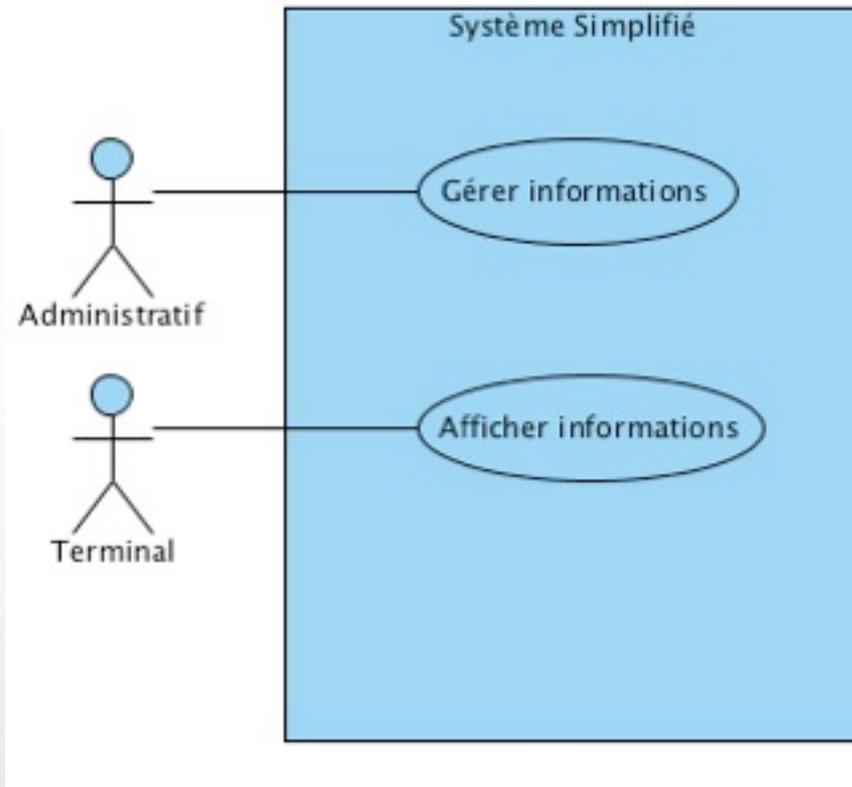
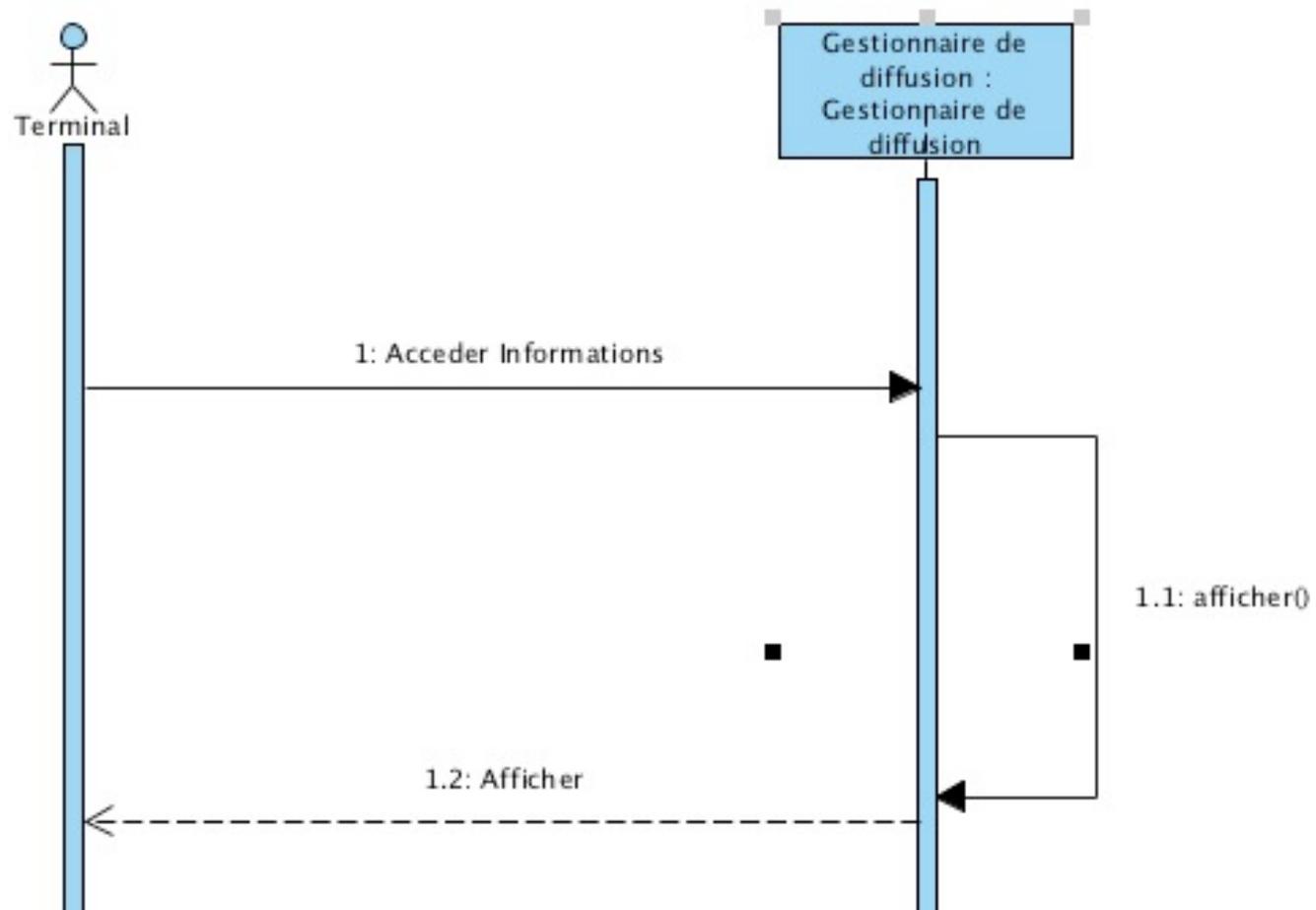


Diagramme de Use-cases



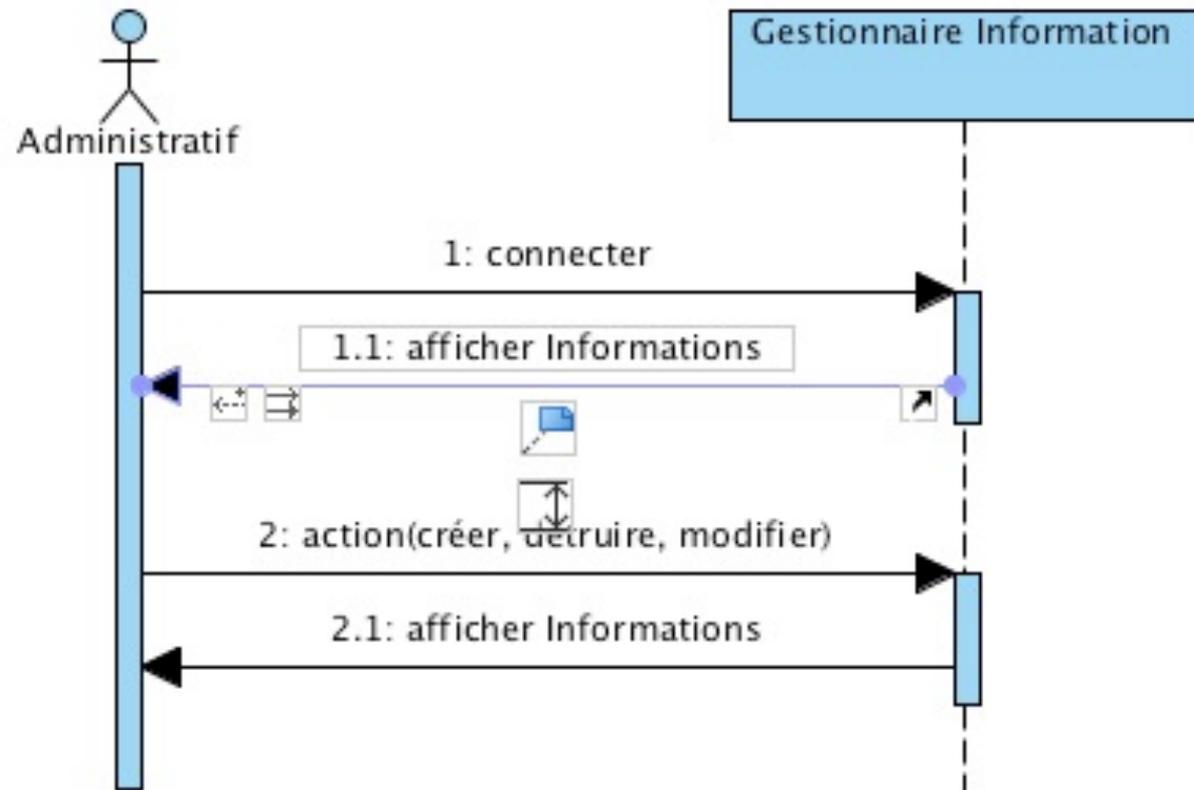


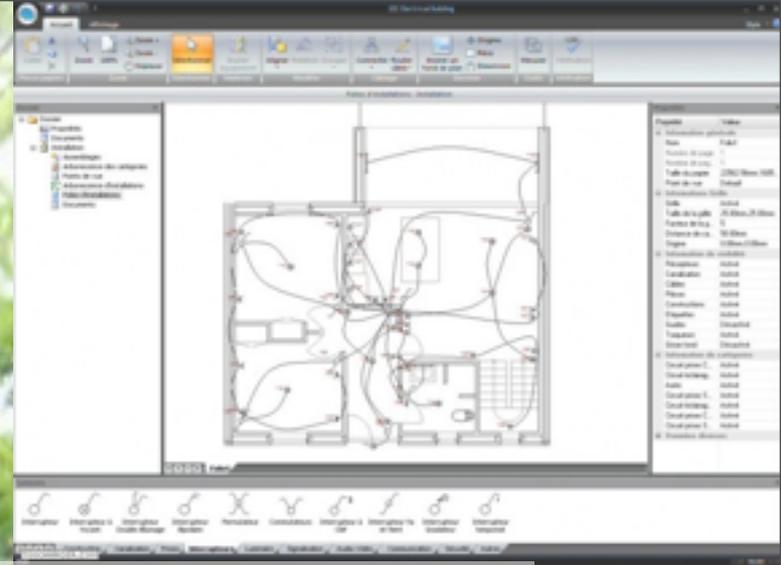
Afficher Informations : niveau Analyse





Gérer Informations : niveau Analyse (0)





Conception *Focus*

- > **Architecture**
- > **Classes**
- > **Données**

8



Conception
Focus

-> **Architecture**

-> **Données** ⁹



Choix d'Architecture

Gestion des Informations

Présentation

Liste des informations

- nuit de l'info [2010-11-12 13:40:18,3]
- Devint [Fri, 26 Nov 2010 22:53,44]
- Rendu Projet ACSI [Sat, 27 Nov 2010 17:55,53]

Logique applicative

Gérer les informations

Stockage

```
CREATE TABLE `information` (  
  `titre` varchar(20) NOT NULL,  
  `date` varchar(22) NOT NULL,  
  `identifiant` int(11) NOT NULL auto_increment,  
  PRIMARY KEY (`identifiant`))
```



Choix d'Architecture

Présentation

Gestion des Informations

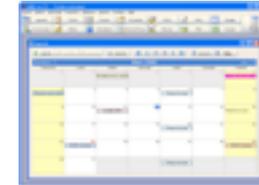
Liste des informations

- nuit de l'info [2010-11-12 13:40:18,3]
- Devint [Fri, 26 Nov 2010 22:53,44]
- Rendu Projet ACSI [Sat, 27 Nov 2010 17:55,53]

Modifier Détruire

Titre de l'information

Créer un nouvelle information



Logique applicative

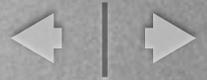
Gérer les informations

Stockage

```
CREATE TABLE `information` (  
  `titre` varchar(20) NOT NULL,  
  `date` varchar(22) NOT NULL,  
  `identifiant` int(11) NOT NULL auto_increment,  
  PRIMARY KEY (`identifiant`))
```



ORACLE



Choix d'Architecture

Gestion des Informations

Liste des informations

- nuit de l'Info [2010-11-12 13:40:18,3]
- Devist [Fri, 26 Nov 2010 22:53,44]
- Rendu Projet ACSI [Sat, 27 Nov 2010 17:55,53]

Titre de l'information

Présentation

Logique applicative

Stockage

clickOn()

IHM

créer Information

Contrôleur

Gérer les informations : métier

```
CREATE TABLE `information` (
  `titre` varchar(20) NOT NULL,
  `date` varchar(22) NOT NULL,
  `identifiant` int(11) NOT NULL auto_increment,
  PRIMARY KEY (`identifiant`))
```

A photograph of a wooden fence with decorative posts, set against a background of trees and a utility pole. The fence is made of vertical wooden planks and has several posts with decorative caps. The ground in front of the fence is covered with green grass and fallen brown leaves. The text is overlaid on a semi-transparent white rectangular box in the center of the image.

SEPARATIONS :
Données, Interactions et
Visualisation, Contrôles

Modèle-Vue-Contrôleur (MVC)

Controller

contrôleur:
chef
d'orchestre

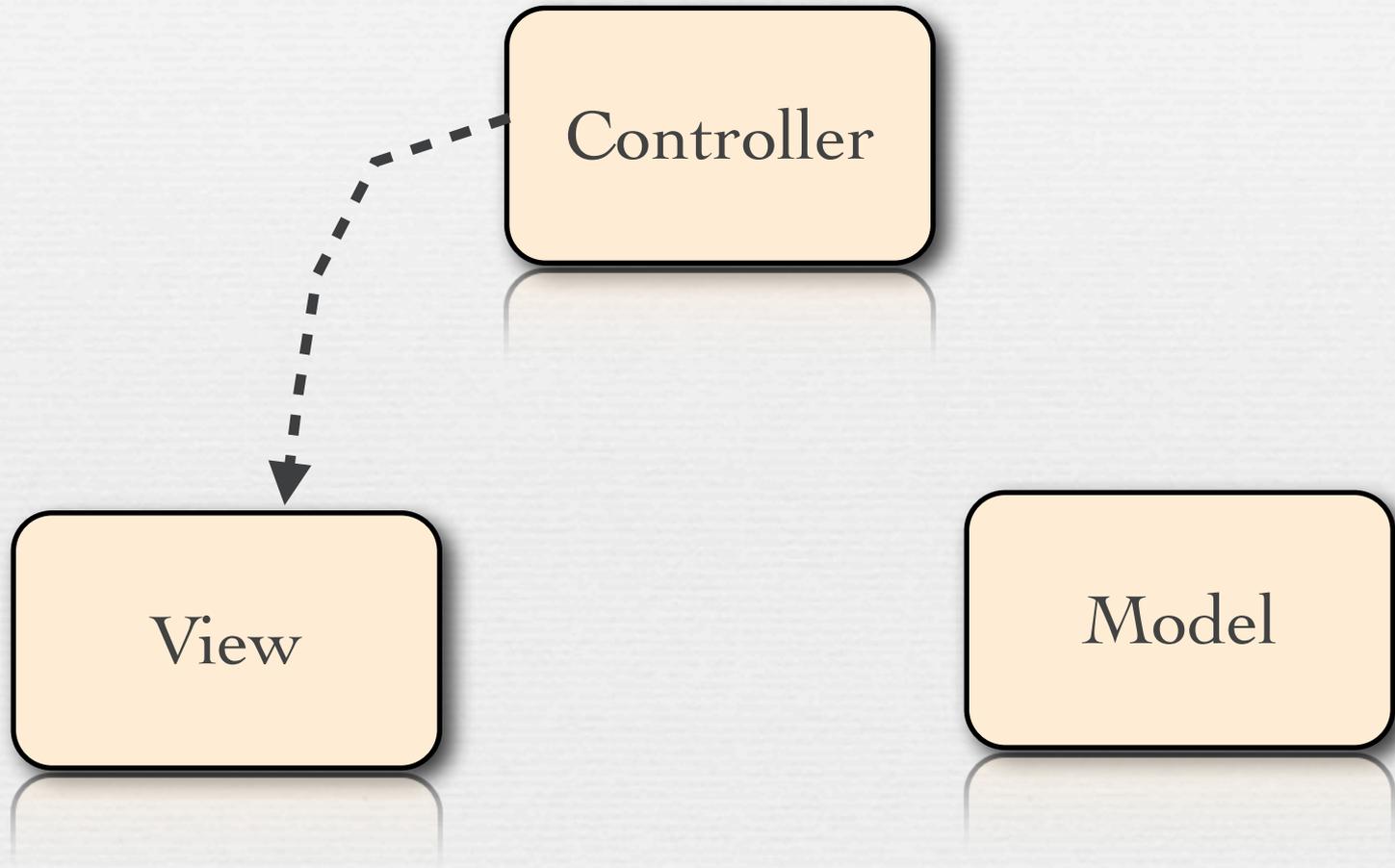
View

La vue: présentée
à l'utilisateur

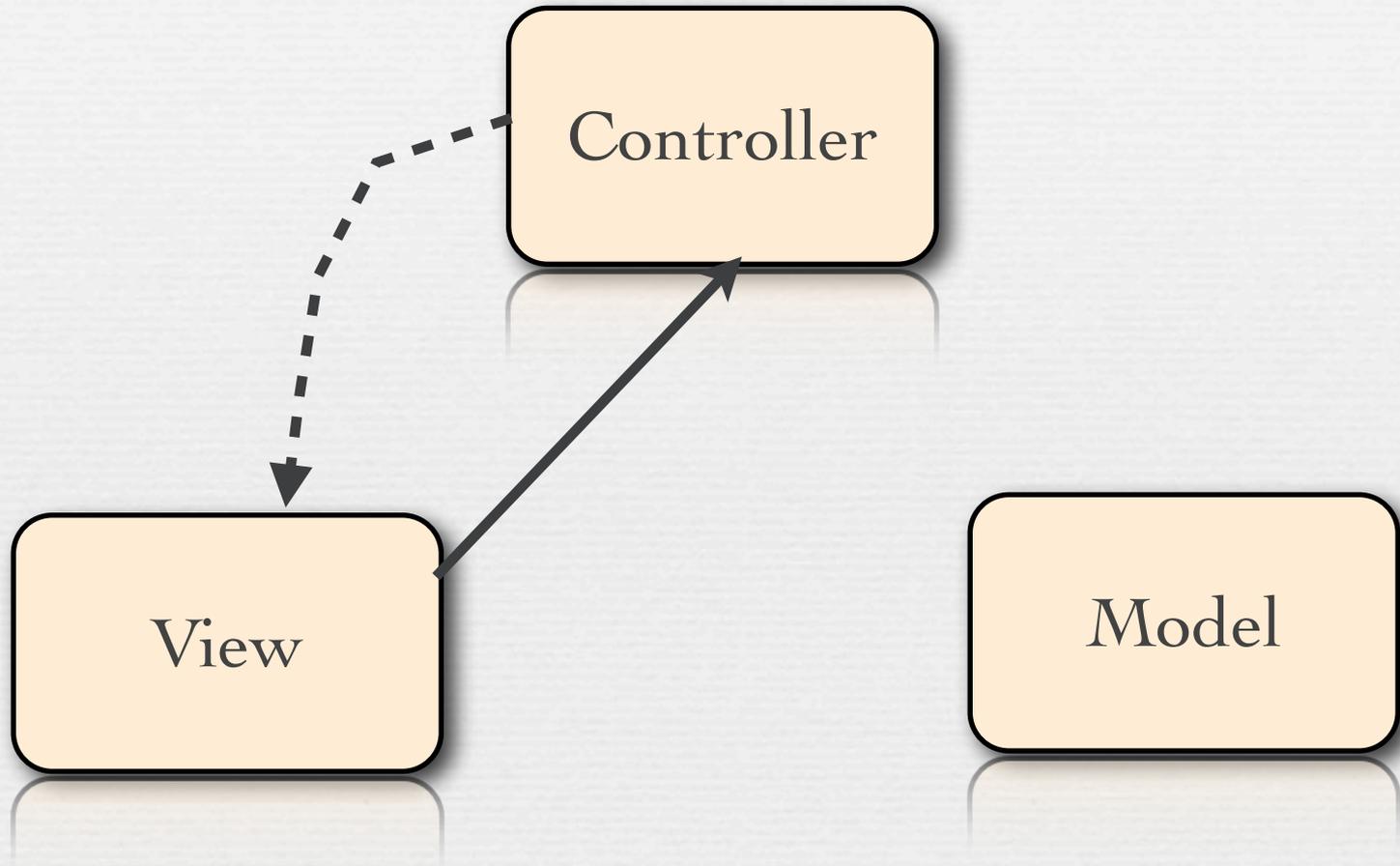
Model

Le modèle: les données
indépendantes

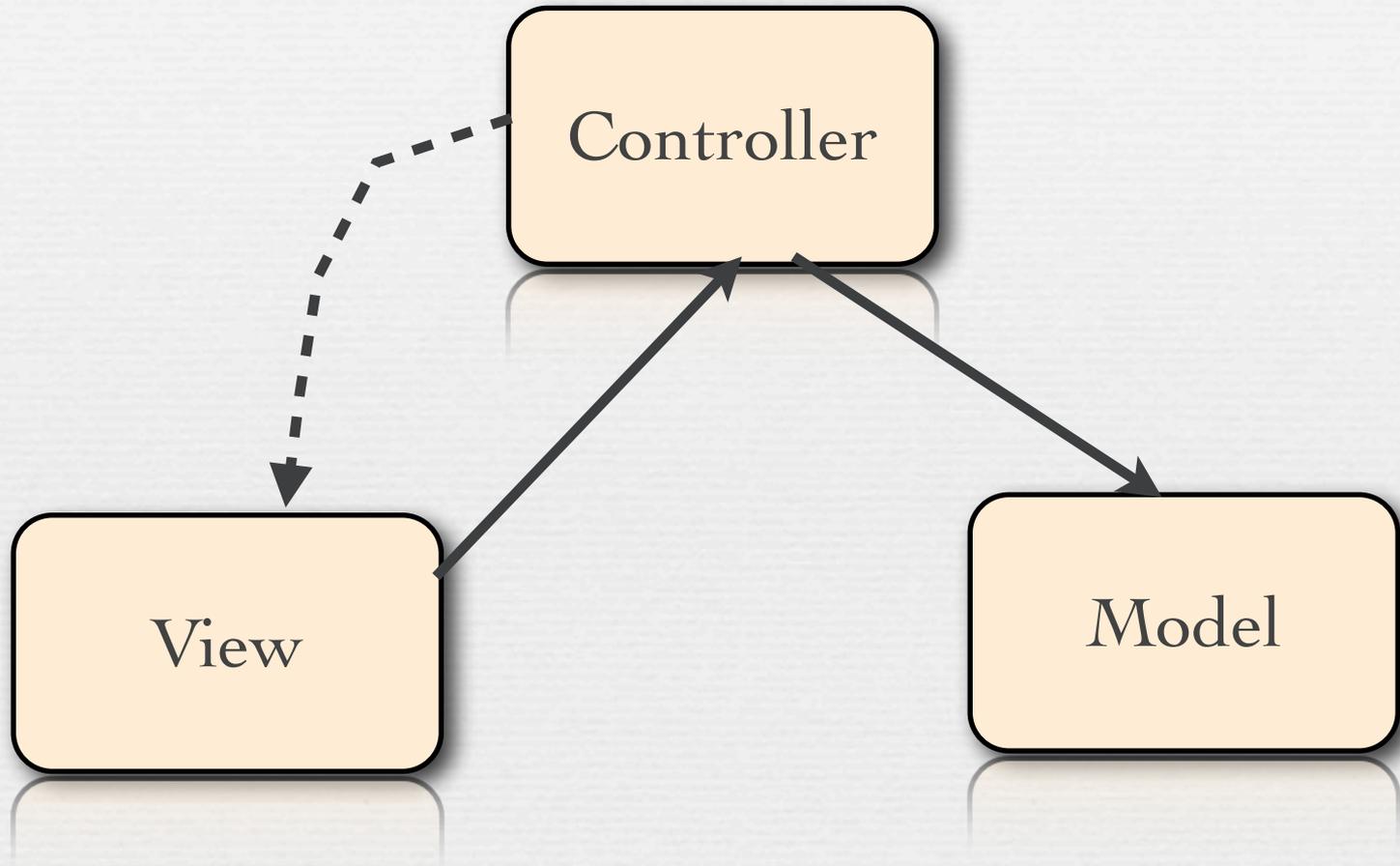
Controlleur observe la vue



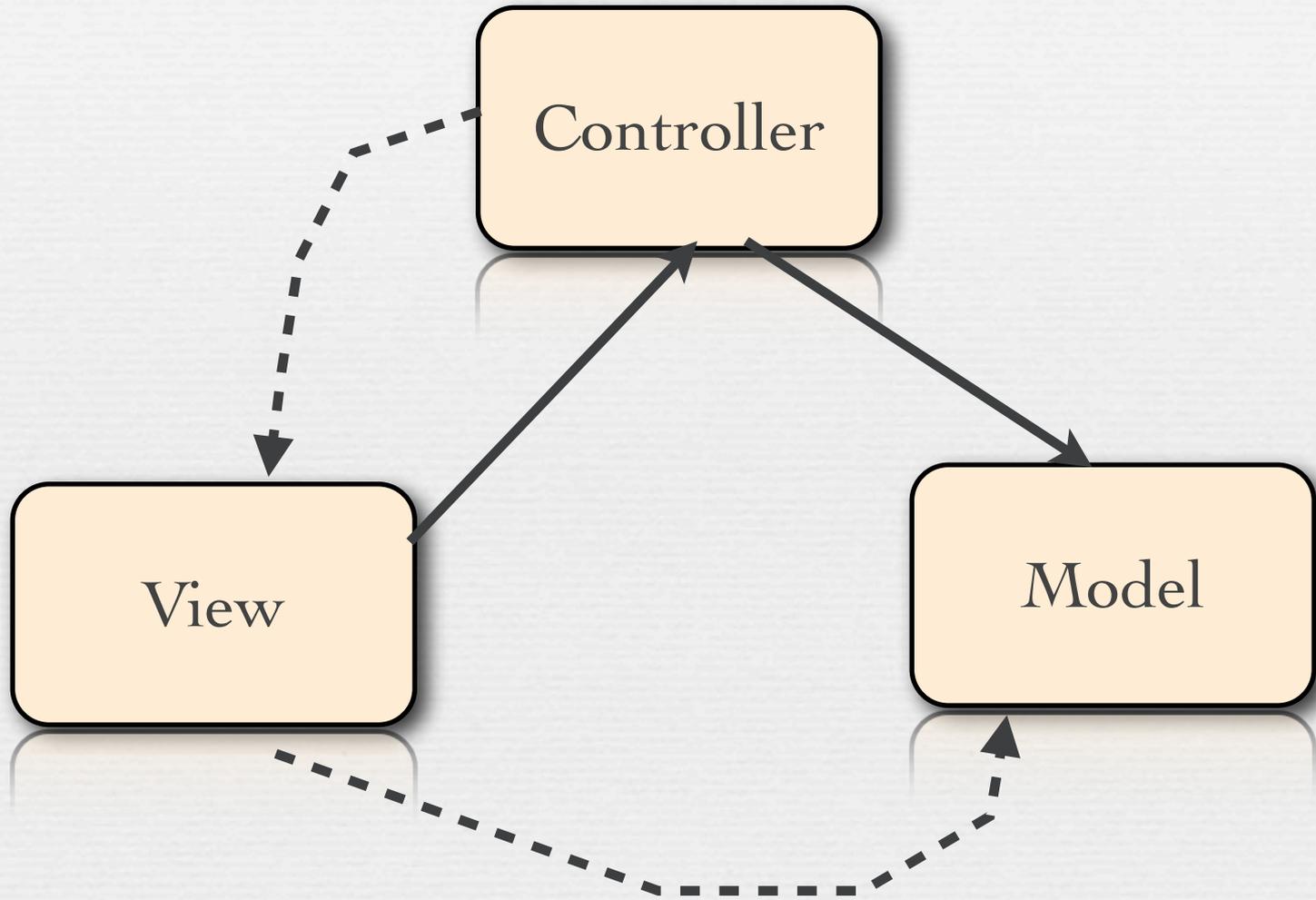
Contrôleur récupère les données de la vue



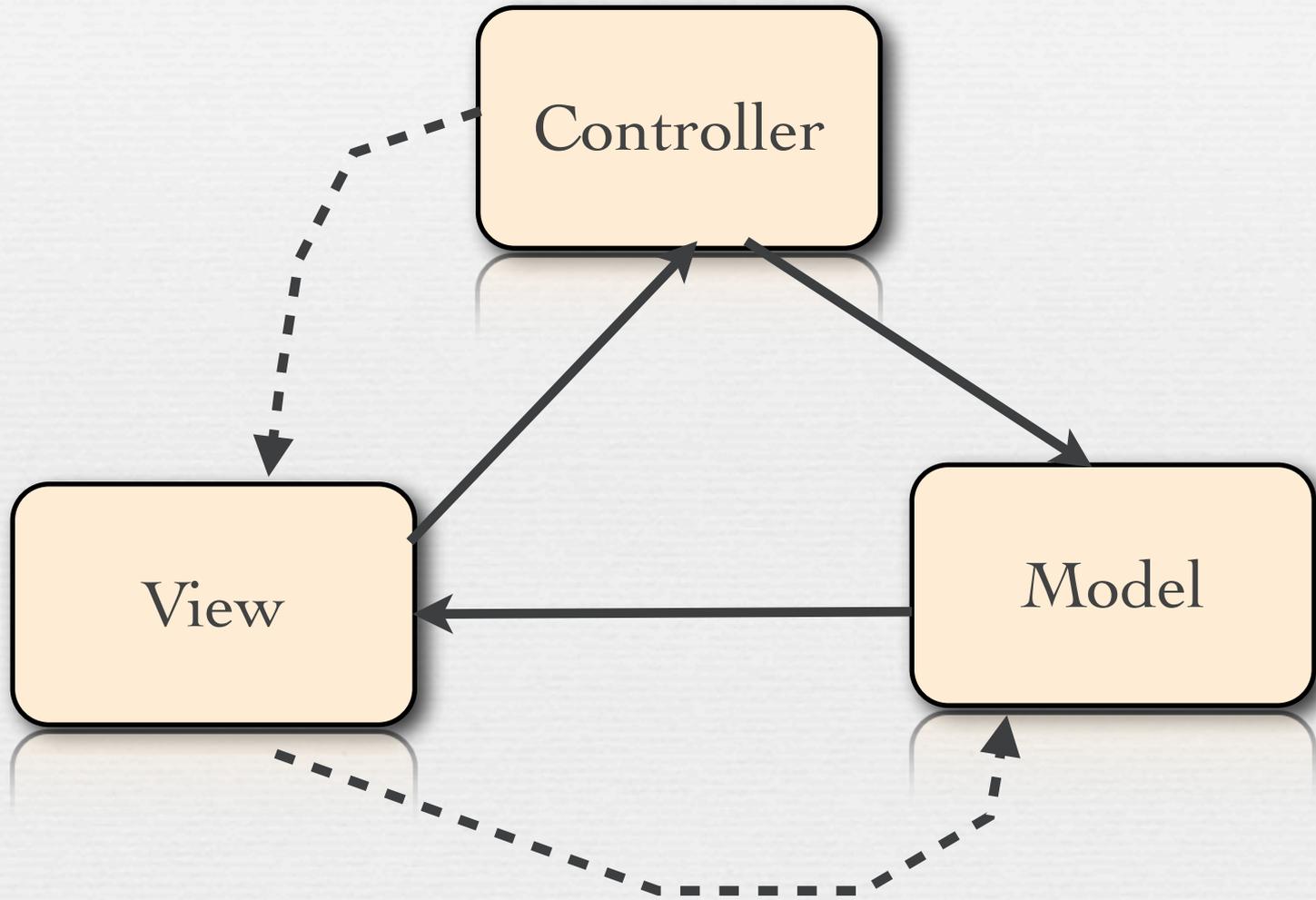
Contrôleur modifie le modèle



Vue observe le modèle

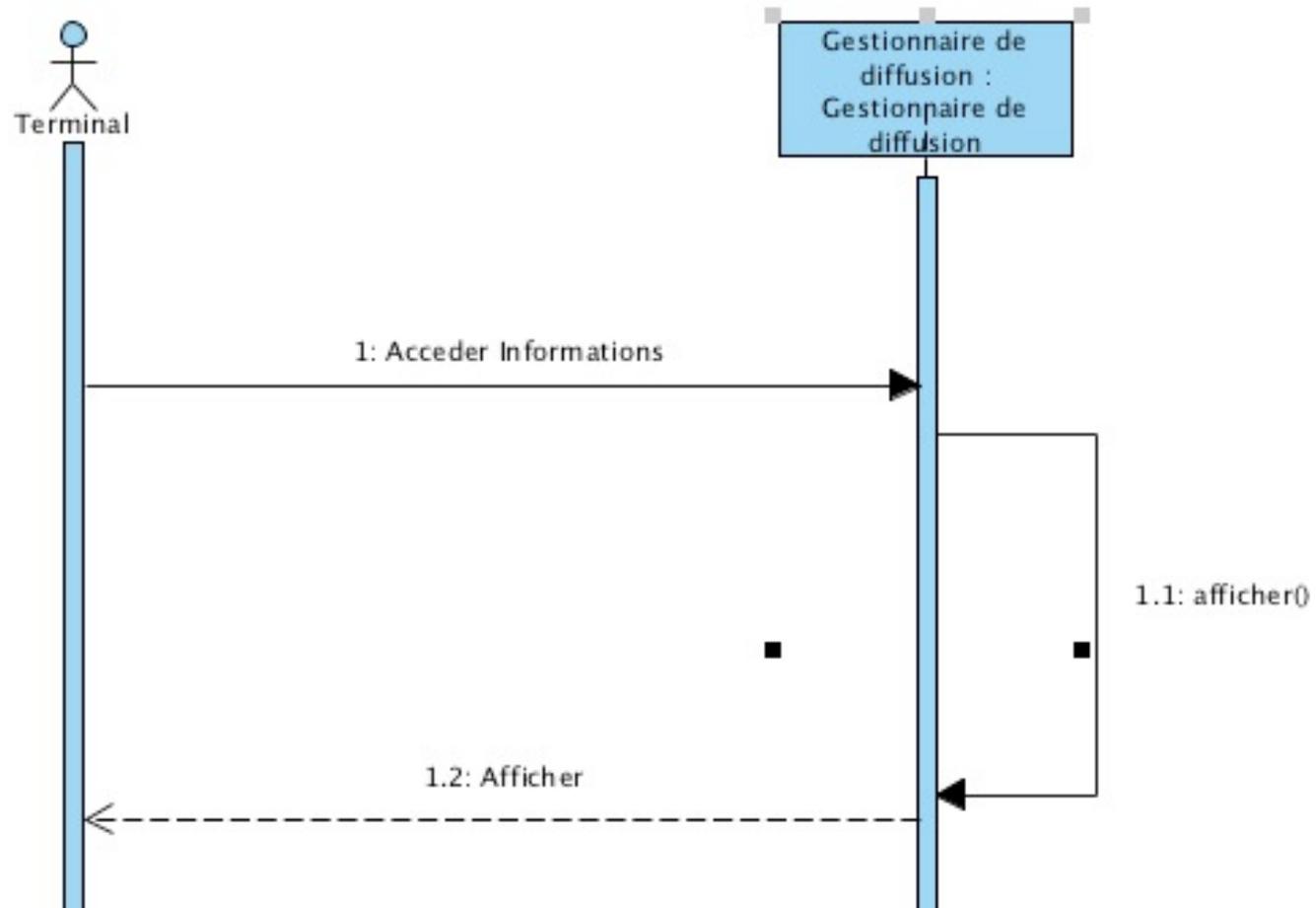


Vue récupère les données du modèle

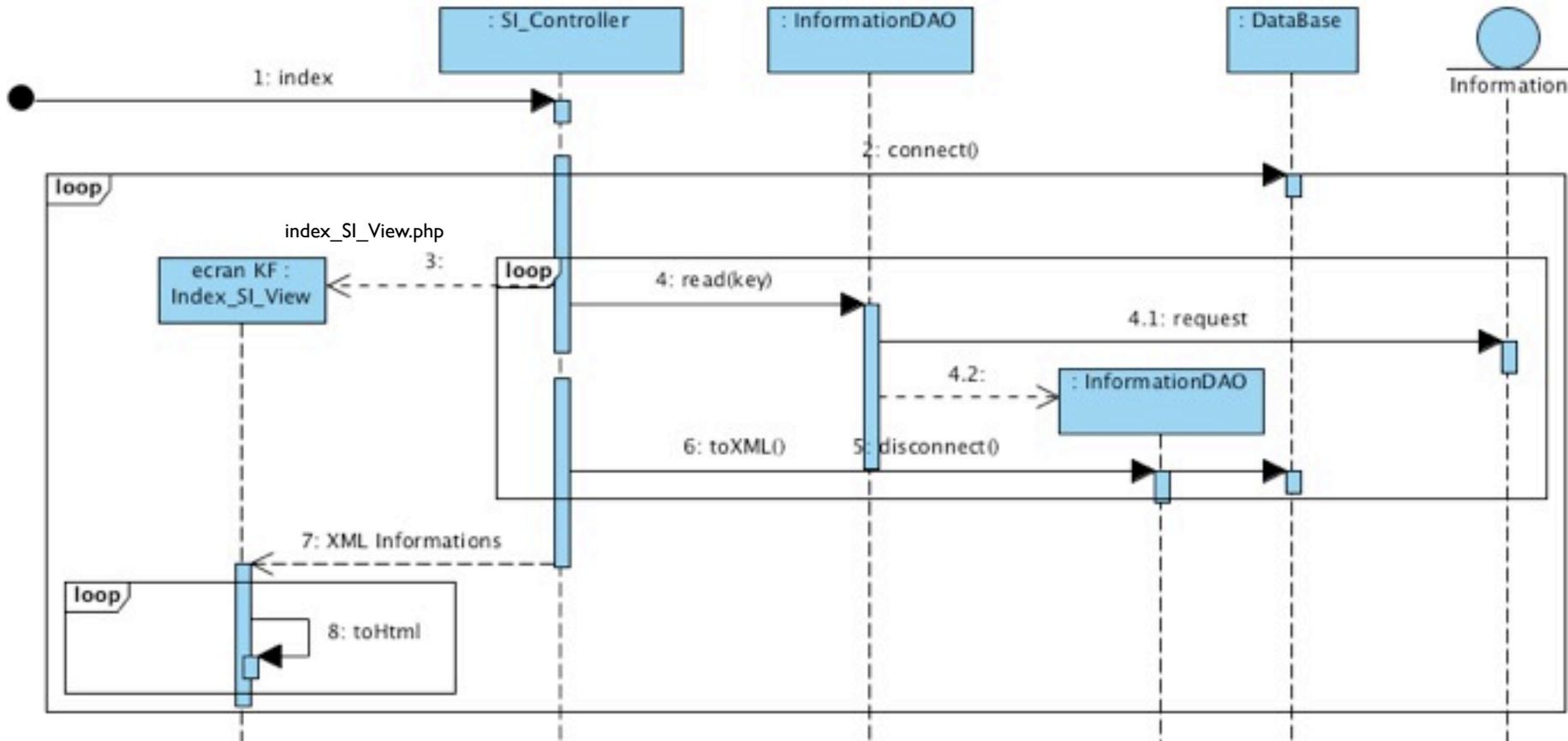




Afficher Informations : niveau Analyse

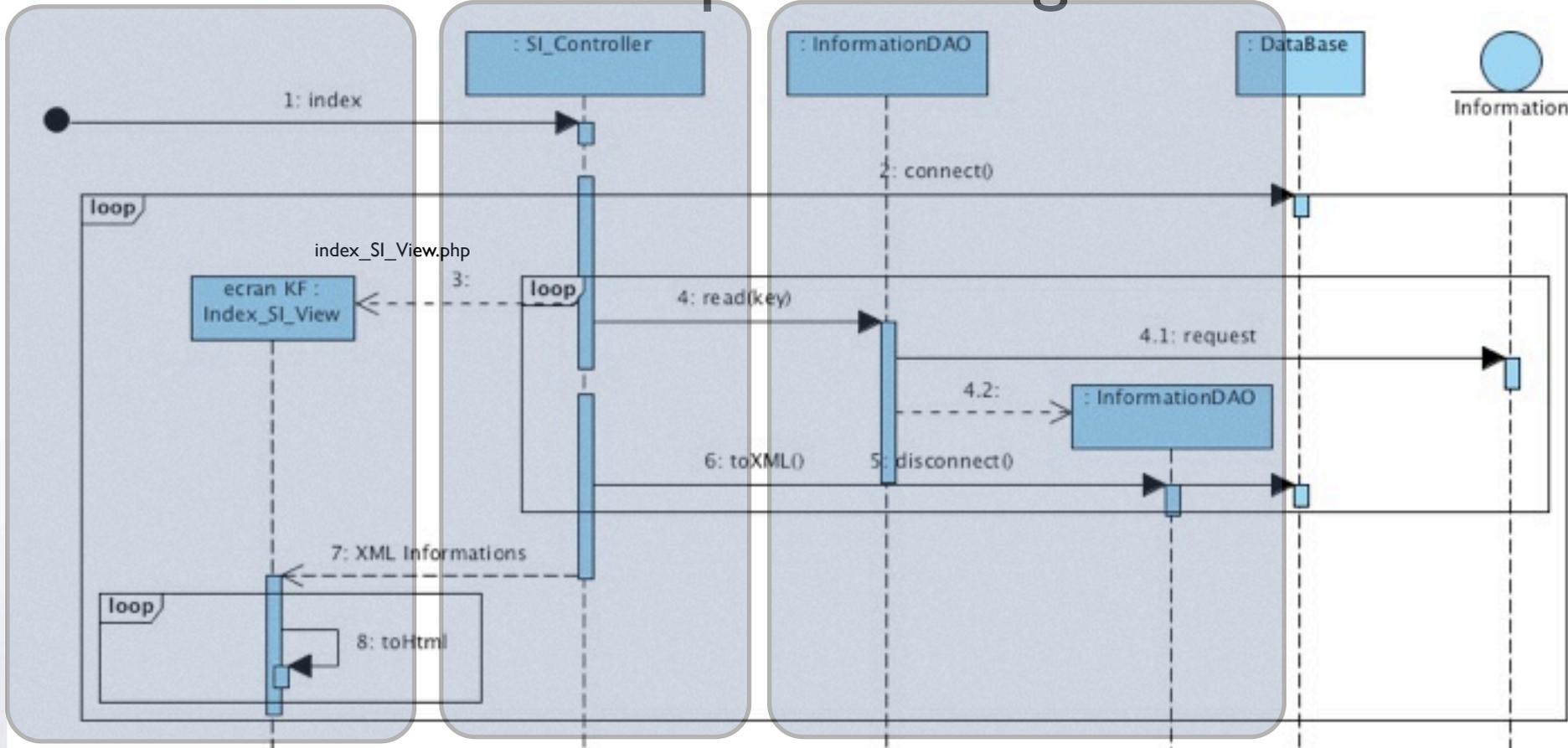


Afficher Informations : niveau Conception



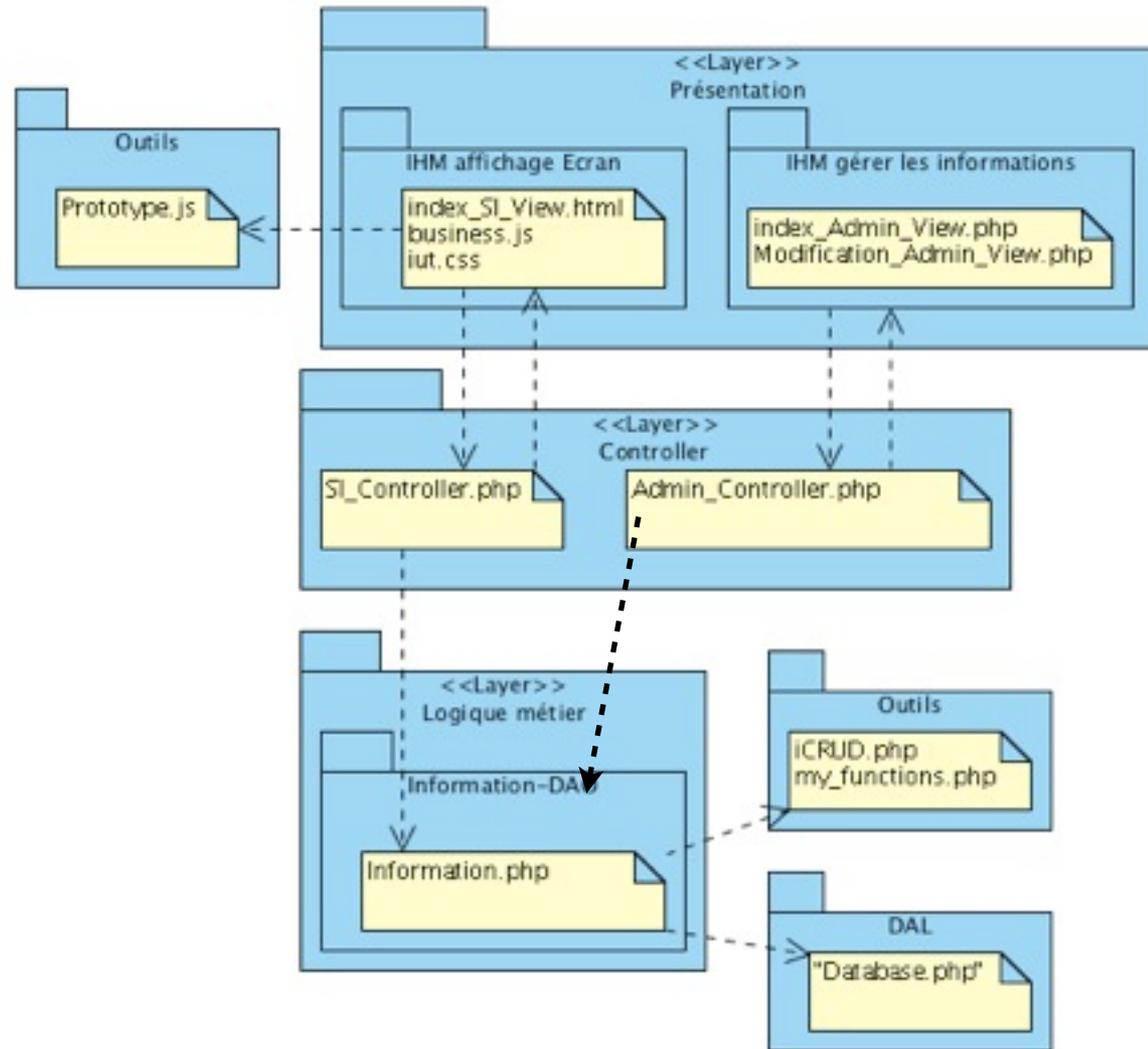


Vers les composants logiciels

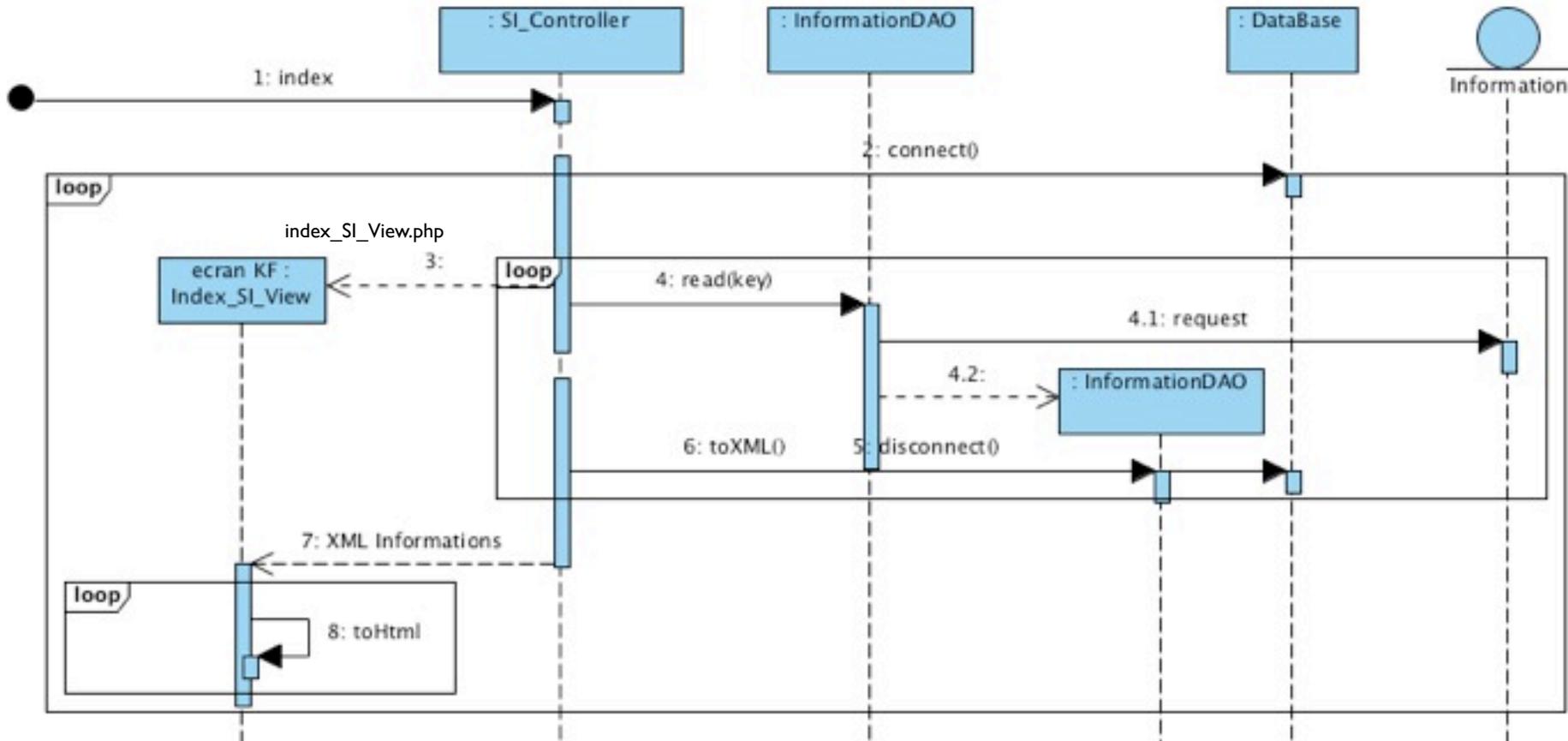




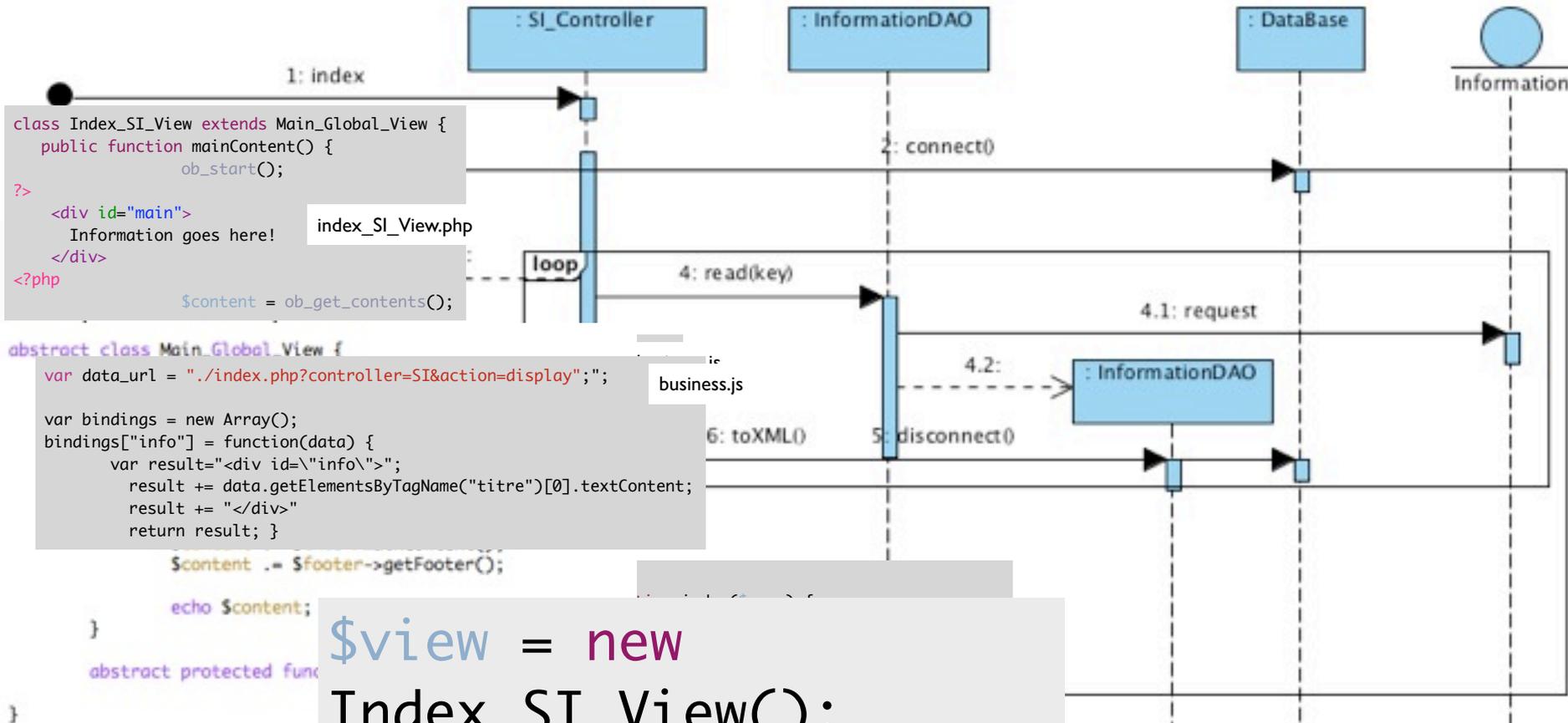
Architecture en couches & Fichiers



Afficher Informations : niveau Conception



Afficher Informations : niveau Conception



```

class Index_SI_View extends Main_Global_View {
    public function mainContent() {
        ob_start();
    }
}
<div id="main">
    Information goes here!
</div>
<?php
    $content = ob_get_contents();
  
```

```

abstract class Main_Global_View {
    var data_url = "./index.php?controller=SI&action=display";
    var bindings = new Array();
    bindings["info"] = function(data) {
        var result="<div id=\"info\">";
        result += data.getElementsByTagName("titre")[0].textContent;
        result += "</div>";
        return result; }
  
```

```

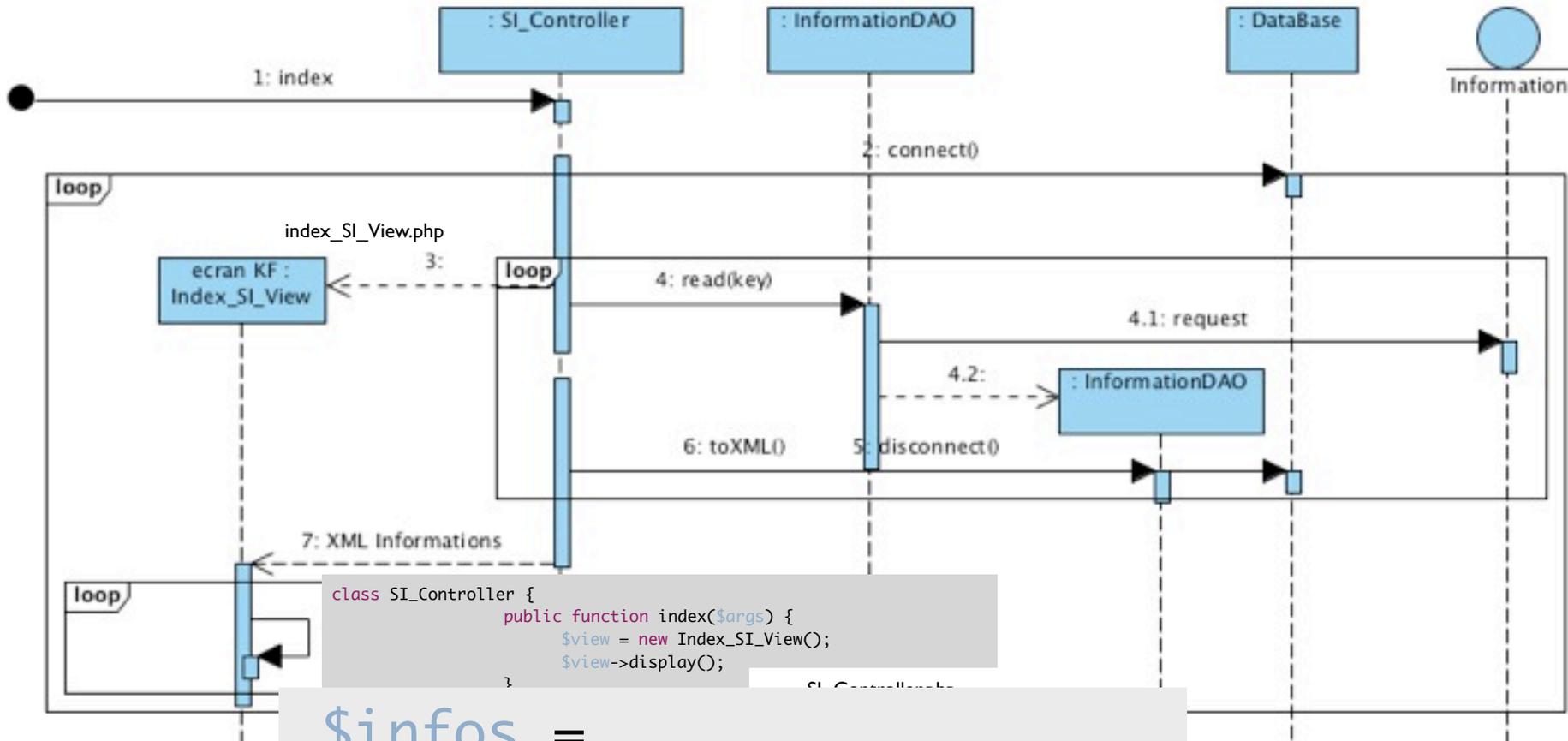
    $content .= $footer->getFooter();
    echo $content;
}
abstract protected func
}
  
```

`$view = new Index_SI_View();`
`$view->display`

```

    }
    $res=$res.'</data>';
    echo $res;
}
  
```

Afficher Informations : niveau Conception



```

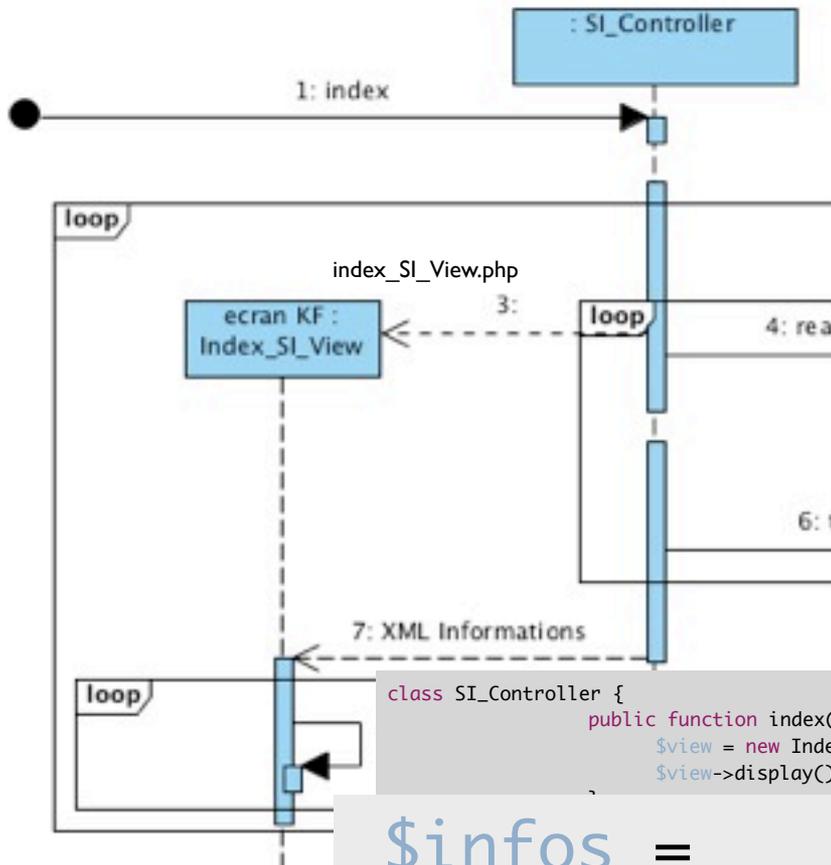
class SI_Controller {
    public function index($args) {
        $view = new Index_SI_View();
        $view->display();
    }
}
  
```

```

$infos =
Information::findAll();
  
```

Information
 <=> InformationDAO

Afficher Informations : niveau Conception



```

class Information implements iCRUD
{
    private $_id;
    private $_titre;
    private $_date;
    public static function findAll() {
        $informations = array();
        Database::connect();
        $query = "SELECT * FROM information";
        $res = mysql_query($query);
        while($line = mysql_fetch_assoc($res))
    }
}
  
```

Information.php

```

class SI_Controller {
    public function index(
        $view = new Inde
        $view->display()
    )
  }
  
```

```

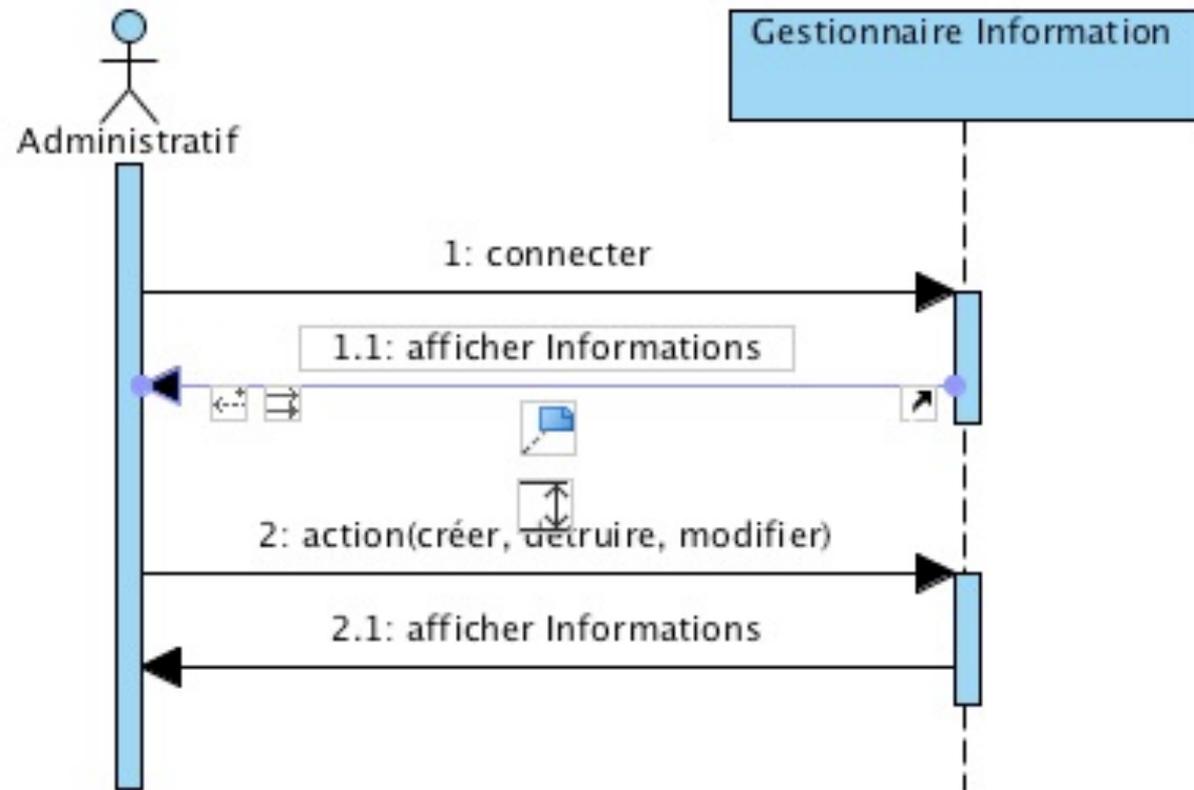
$infos =
Information::findAll();
  
```

```

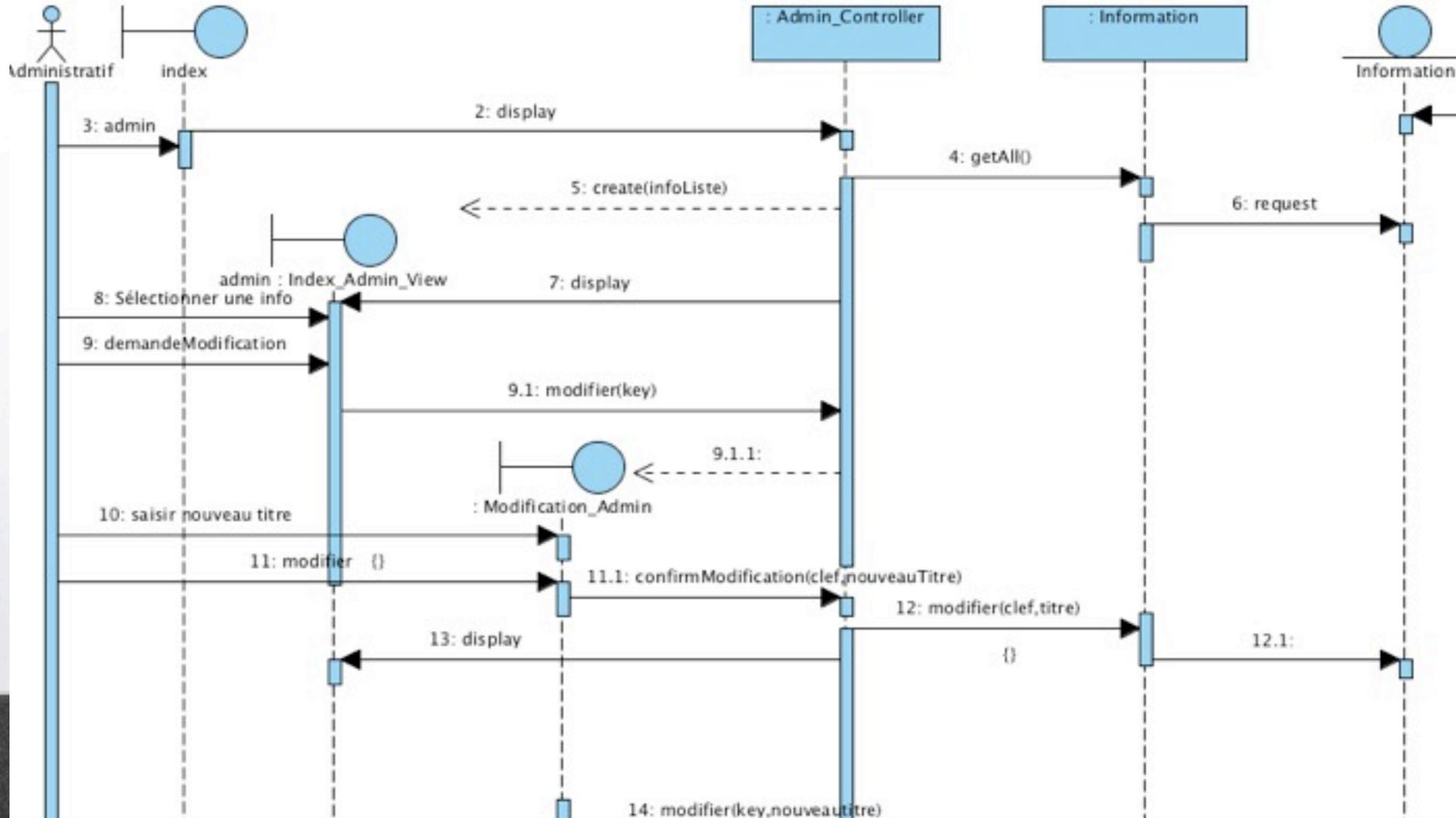
        $res = $res.&#220;mpInformation->toXML() ;
    }
    $res=$res.'</data>';
    echo $res;
}
  
```



Gérer Informations : niveau Analyse (0)

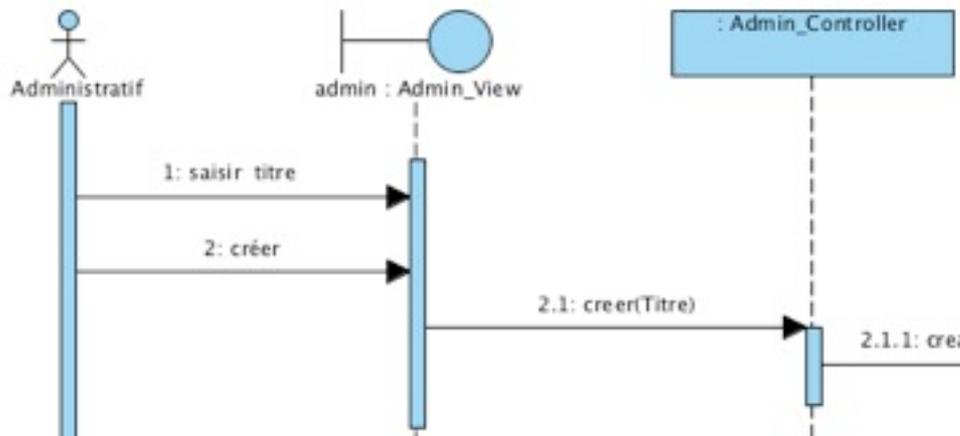


modifier Information : niveau Conception





Créer Information : niveau Conception



```

class Index_Admin_View extends Main_Global_View {
    private $infosListe;

    public function Index_Admin_View($args) {
        $this->infosListe = $args['infosListe'];
    }...
}
<h1>Gestion des Informations</h1>
<h2> Liste des informations </h2>
<form id="infoModifForm" name="infoModifForm" method="post" action="."/ >
<p>
  
```

```

class Admin_Controller {
    public function index($args) {
        $args['infosListe'] = Information::findAll();
        $view = new Index_Admin_View($args);
        $view->display();
    }
}
Admin_Controller.php

public function confirmer_modifier($args) {
    $key = $_POST["key"];
    $newTitre = $_POST["NouveauTitre"];
    $info = Information::read($key);
    $info->setTitre($newTitre);
    $info->update();

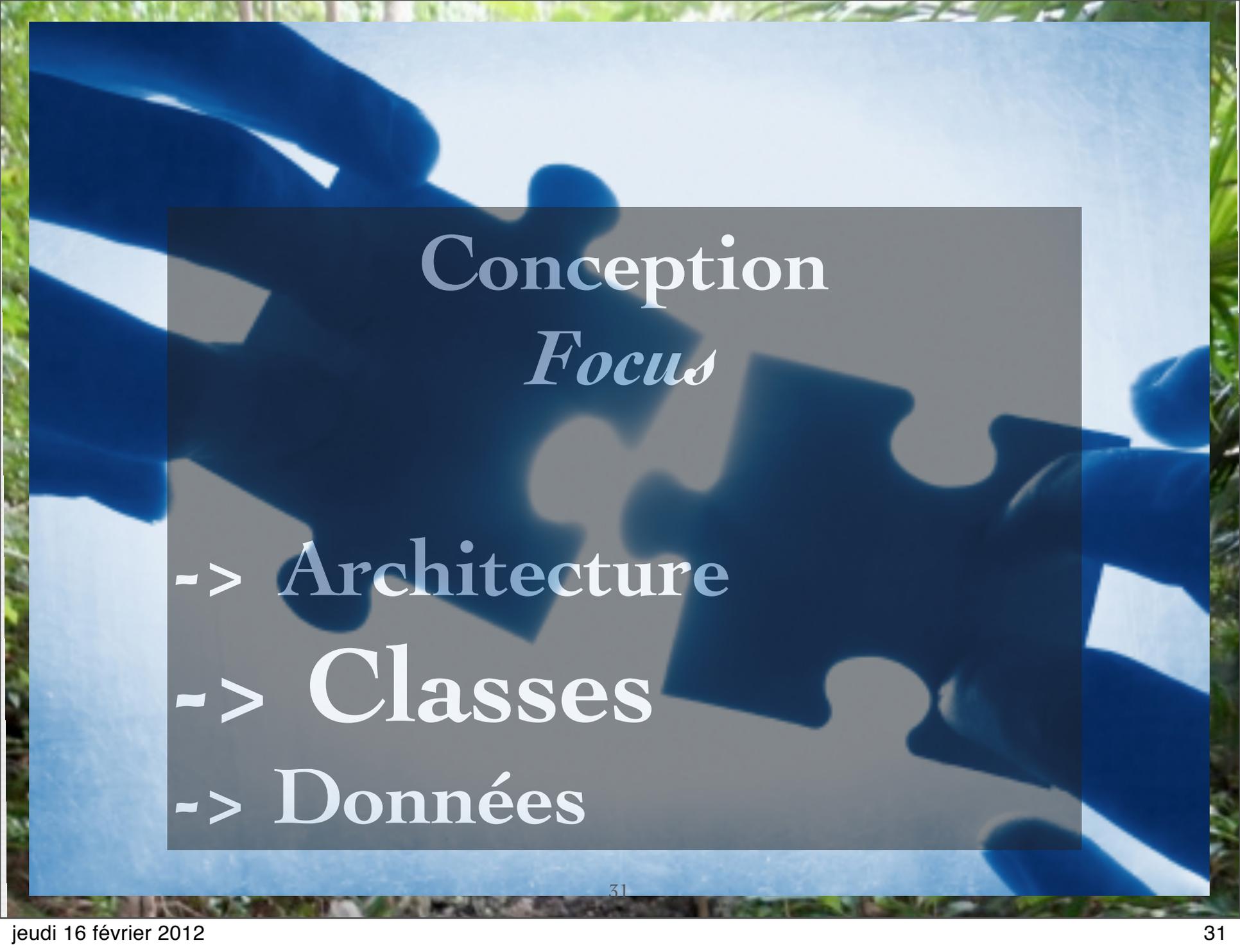
    $args['infosListe'] = Information::findAll();

    $view = new Index_Admin_View($args);
    $view->display();
}

public function create($args) {
    $titre = $_POST["Titre"];
    $info = new Information($titre, $this->today());
    $info->create();

    $args['infosListe'] = Information::findAll();

    $view = new Index_Admin_View($args);
    $view->display();
}
  
```



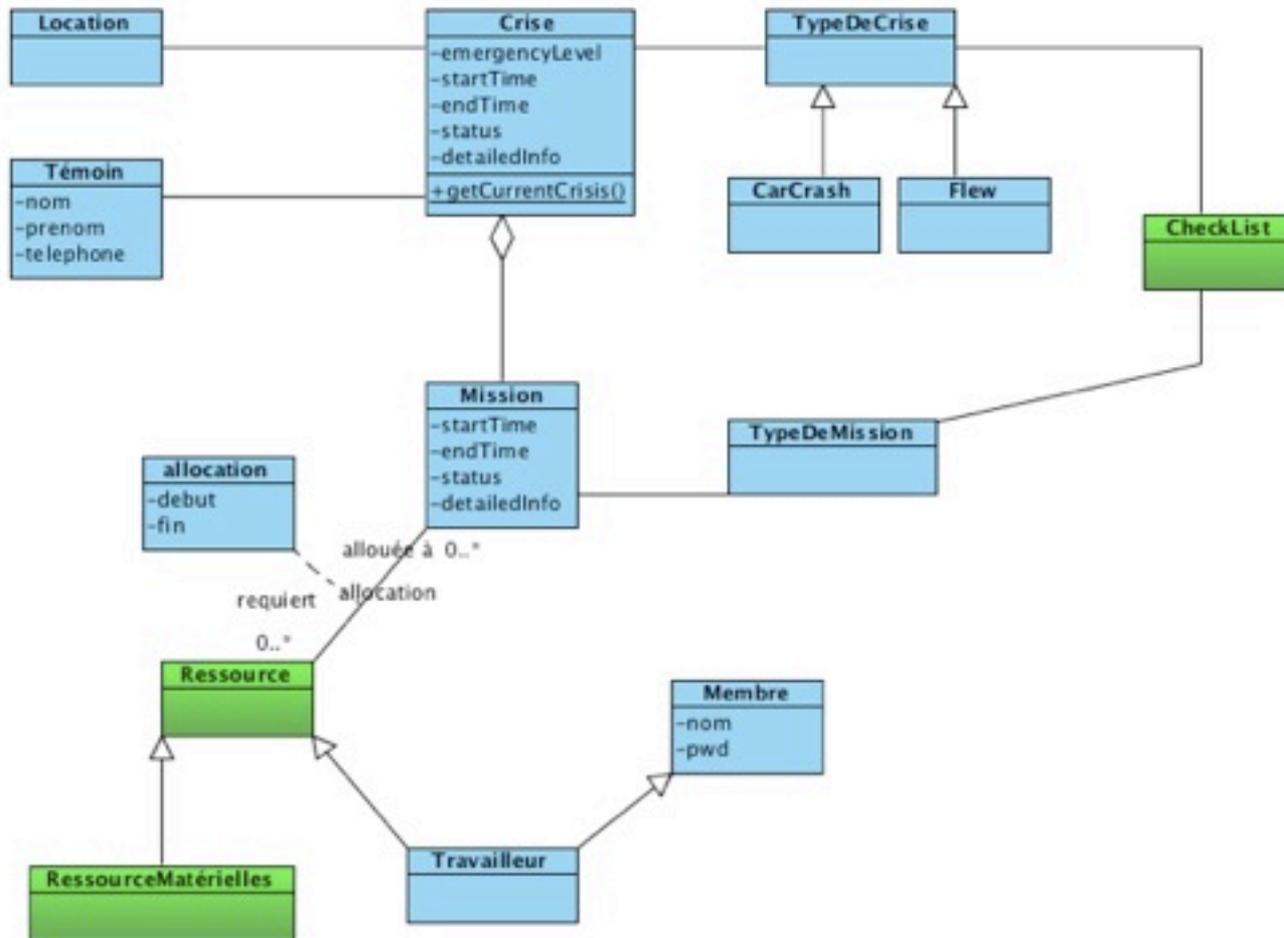
Conception

Focus

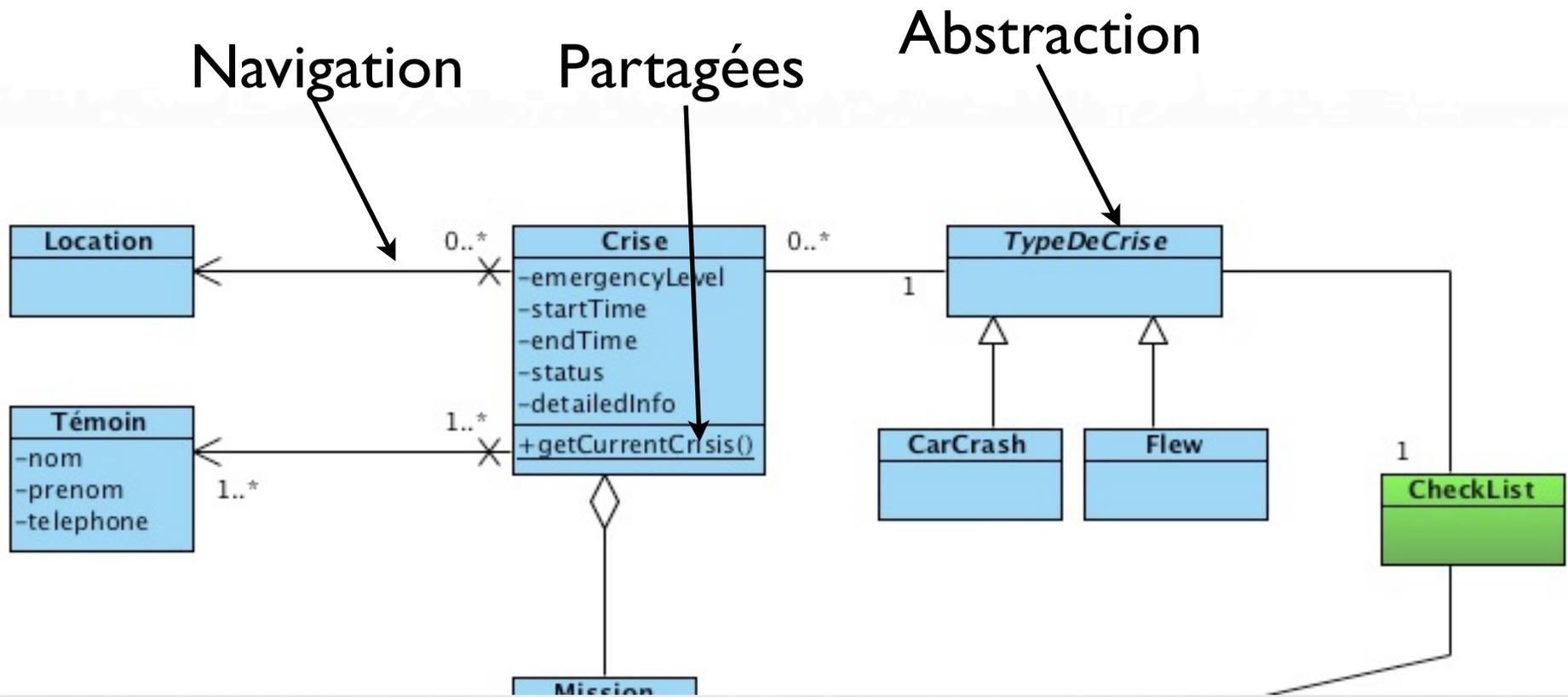
- > Architecture
- > Classes
- > Données



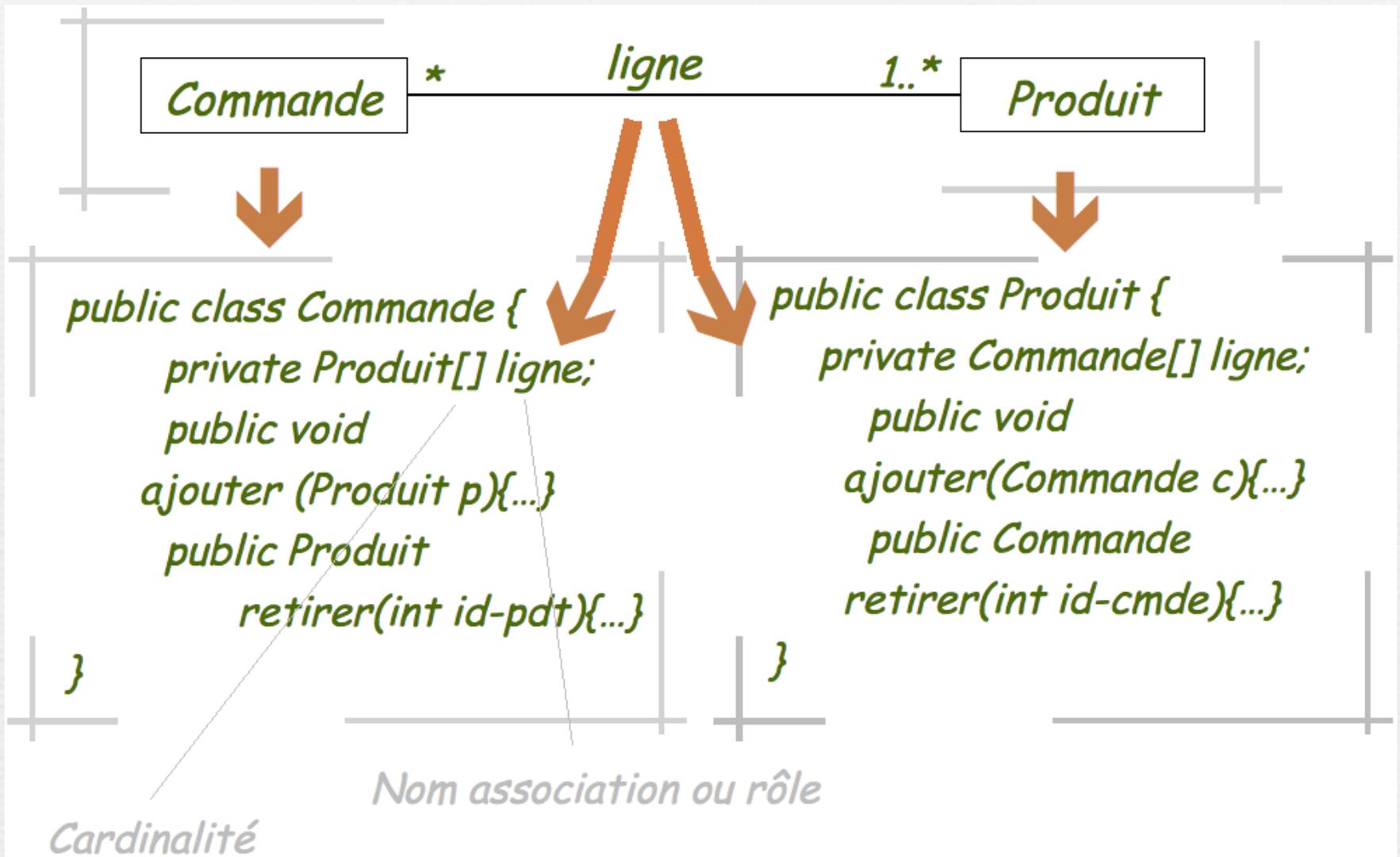
En cours d'analyse, Diagramme de Classes



En conception, Diagramme de Classes



Association...



Association...

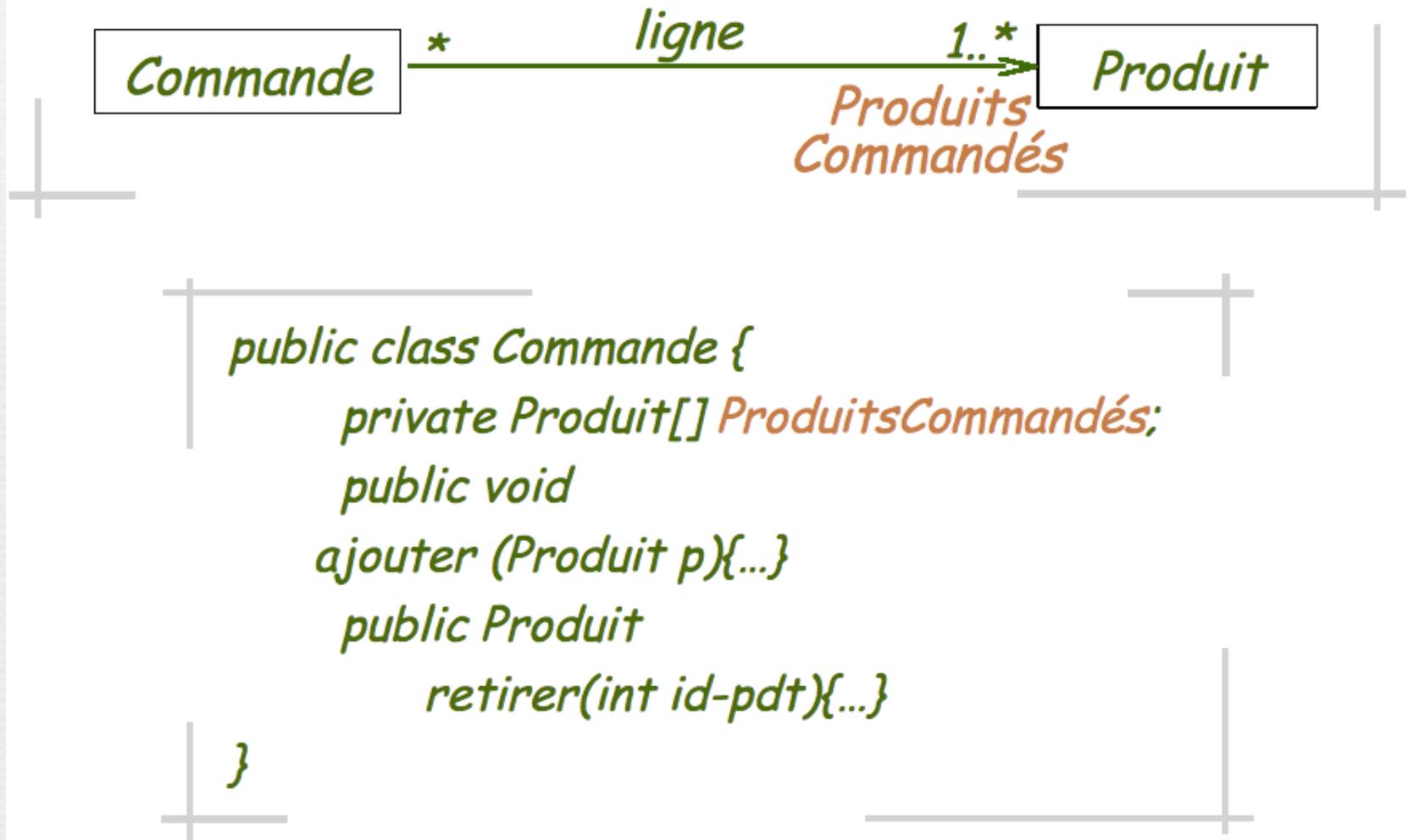


public class Commande {
private Produit[] ligne;
public void
ajouter (Produit p){...}
public Produit
retirer(int id-pdt){...}
}

~~*public class Produit {*~~
~~*private Commande[] ligne;*~~
~~*public void*~~
~~*ajouter (Commande c){...}*~~
~~*public Commande*~~
~~*retirer(int id-cmde){...}*~~
~~*}*~~

Association:

De la conception à l'implémentation



Association:

De la conception à l'implémentation



```
public class Commande {
    private Produit[] ProduitsCommandés;
    public void
    ajouter (Produit p) {
public Commande (Produit[] c) throws Exception {
    if (c.length != 0)
        lignes = c;
    else
        throw new Exception("Un produit au moins est requis");
}
```

Association:

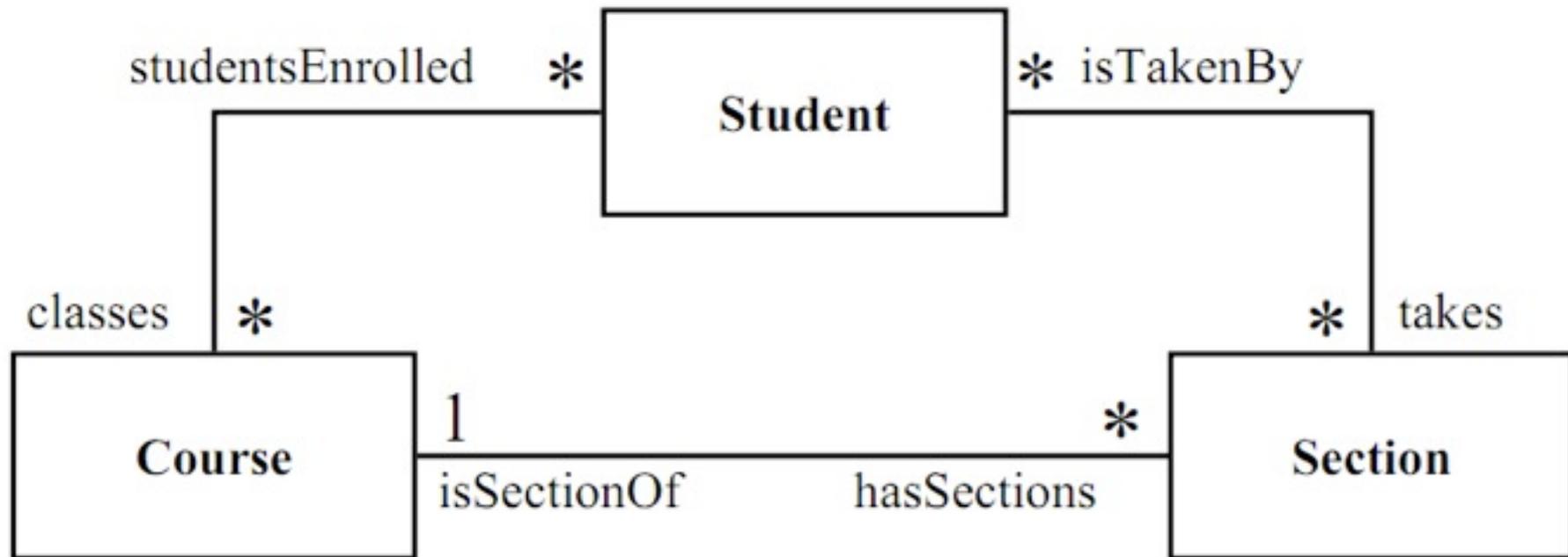
De la conception à l'implémentation



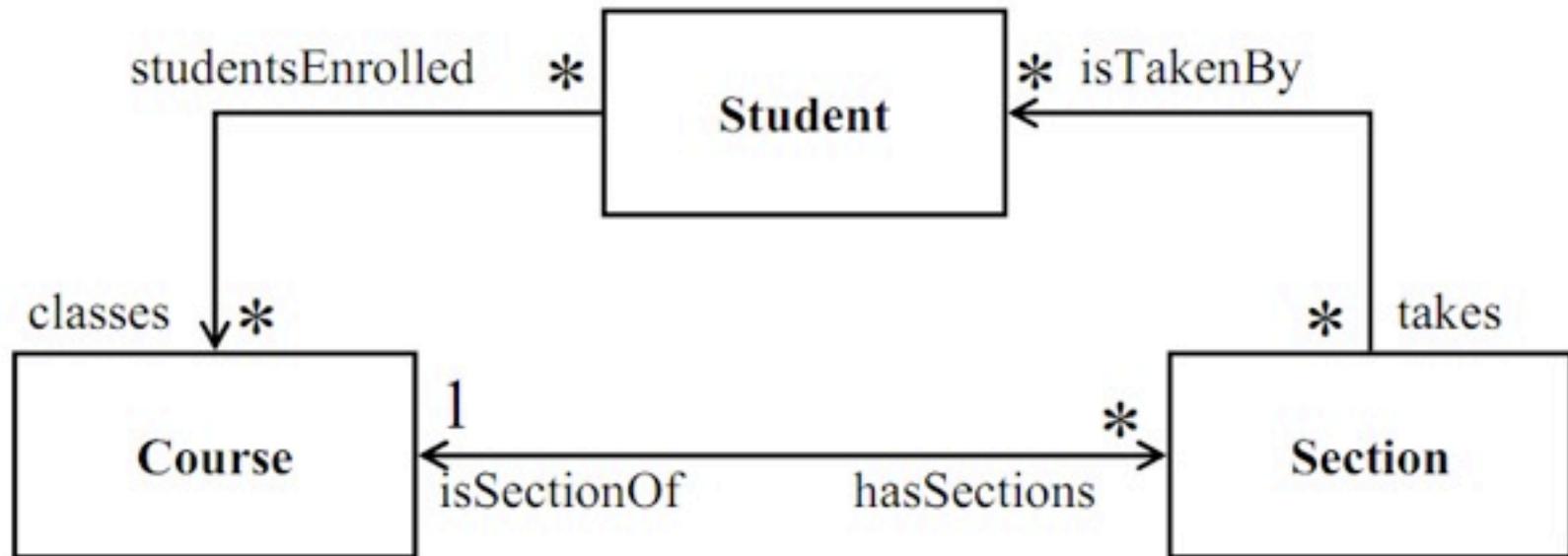
```
public Commande (Produit[] c) throws Exception {
    if (c.length != 0)
        lignes = c;
    else
        throw new Exception("Un produit au moins est
requis");
}
```

```
public boolean oterProduit(Course c) {
    if (lignes.length==1)
        return false;
    ...
}
```

Gestion des associations



Exemple de Raffinement



Ce n'est qu'un exemple, d'autres raffinements sont possibles...

Principes d'implémentation

- Extrémité d'association 1
 - Rôle en Attribut avec type de l'extrémité
 - *Type* `getRole()`
- Extrémité d'association *
 - Rôle (pluriel) en collection
 - Type de l'extrémité en élément de collection
 - Collection `getRoles()`
 - `// Collection<TypeExtrémité>//`

Principes d'implémentation (Suite)

- *Fixer* une extrémité d'association 1
 - `void setRole(Type t)`
- *Fixer* une extrémité d'association *
 - `void setRoles(Collection c)`
 - `void addRole(TypeElement t)`
- *Fixer* une association navigable dans les 2 sens :
 - Définir les responsabilités : un des objets est responsable de la connexion/déconnexion (cf. exemple)

Implémentation

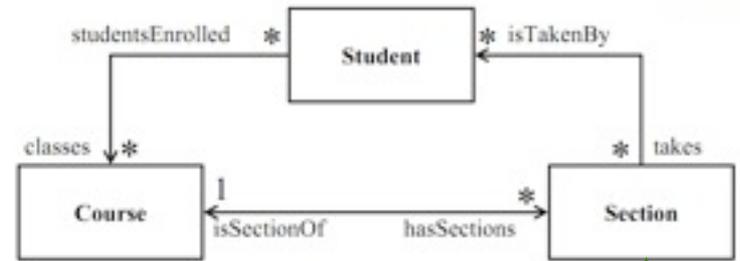
```
public class Section {  
    private String name;  
    private Course isSectionOf;  
    private Collection<Student> isTakenBy = new ArrayL  
    public String toString() {  
    public Section(String name, Course isSectionOf) {  
        this.name = name;  
        //ATTENTION ...  
        setIsSectionOf(isSectionOf);  
    }  
}
```

```
public void setIsSectionOf(Course c) {  
    isSectionOf = c;  
    c.addHasSections(this);  
}
```

```
public Course getIsSectionOf() { return isSectionOf;}
```

```
public Collection<Student> getIsTakenBy() {  
    return isTakenBy;  
}
```

```
public void addStudent(Student s) {  
    isTakenBy.add(s);  
    if (!s.getClasses().contains(isSectionOf)) )  
        s.addClass(isSectionOf);  
}
```



Prise de
responsabilités

Implémentation

```
public class Student {  
    private String name;  
    private Collection<Course> classes;
```

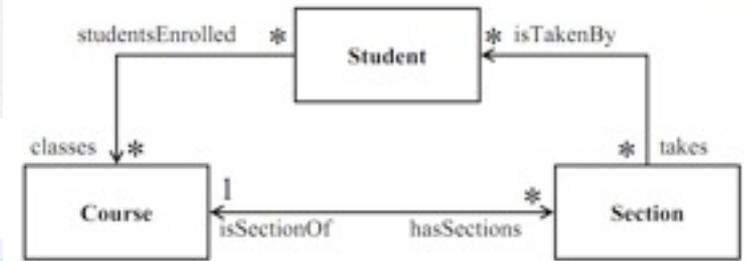
```
    public Student(String name) {..  
    public String toString() {..  
    public Collection<Course> getClasses() {  
        return classes;  
    }
```

```
    protected void addClass(Course c){  
        classes.add(c);
```

```
public class Course {  
    private String name;  
    private Collection<Section> hasSections = new ArrayList<Section>();
```

```
    public Course(String name) {..  
    public String toString() {..  
    public Collection<Section> getHasSections() {  
        return hasSections;
```

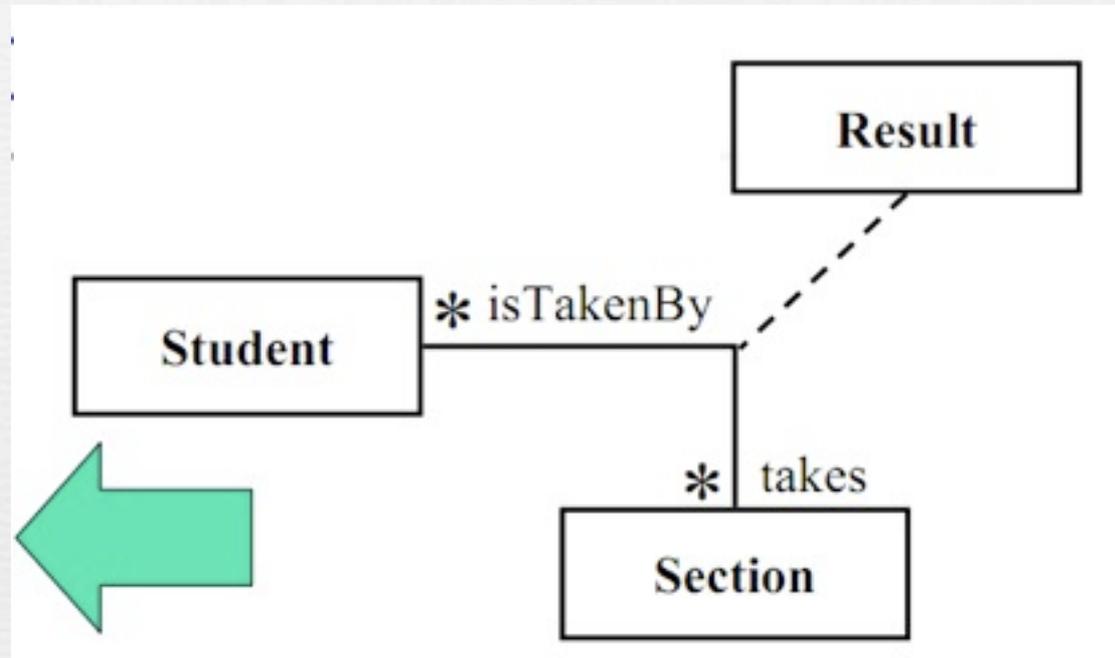
```
    protected void addHasSections(Section s){  
        hasSections.add(s);
```



Définition des
responsabilités
Ne jamais appeler
addHasSections ou
addClass directement !

Implémentation

```
class Student {  
    Result getResult(Section s)  
}  
class Section {  
    Result getResult(Student s)  
}  
class Result {  
    Student getStudent()  
    Section getSection()  
}
```





En résumé : Traduction des associations en attributs

- Autant d'attributs que de classes auxquelles elle est reliée (navigable)
- Association unidirectionnelle = pas d'attribut du côté de la flèche
- Nom de l'attribut = nom du rôle ou forme nominale du nom de l'association
- Attribut du type référence sur un objet de la classe à l'autre extrémité de l'association
 - Référence notée « @ »
- Traduction des multiplicités
 - 1 \Rightarrow @Classe
 - * \Rightarrow Collection @Classe
 - 0..N \Rightarrow Tableau[N] Classe
- Multiplicité avec tri = Collection ordonnée @Classe

Compositions

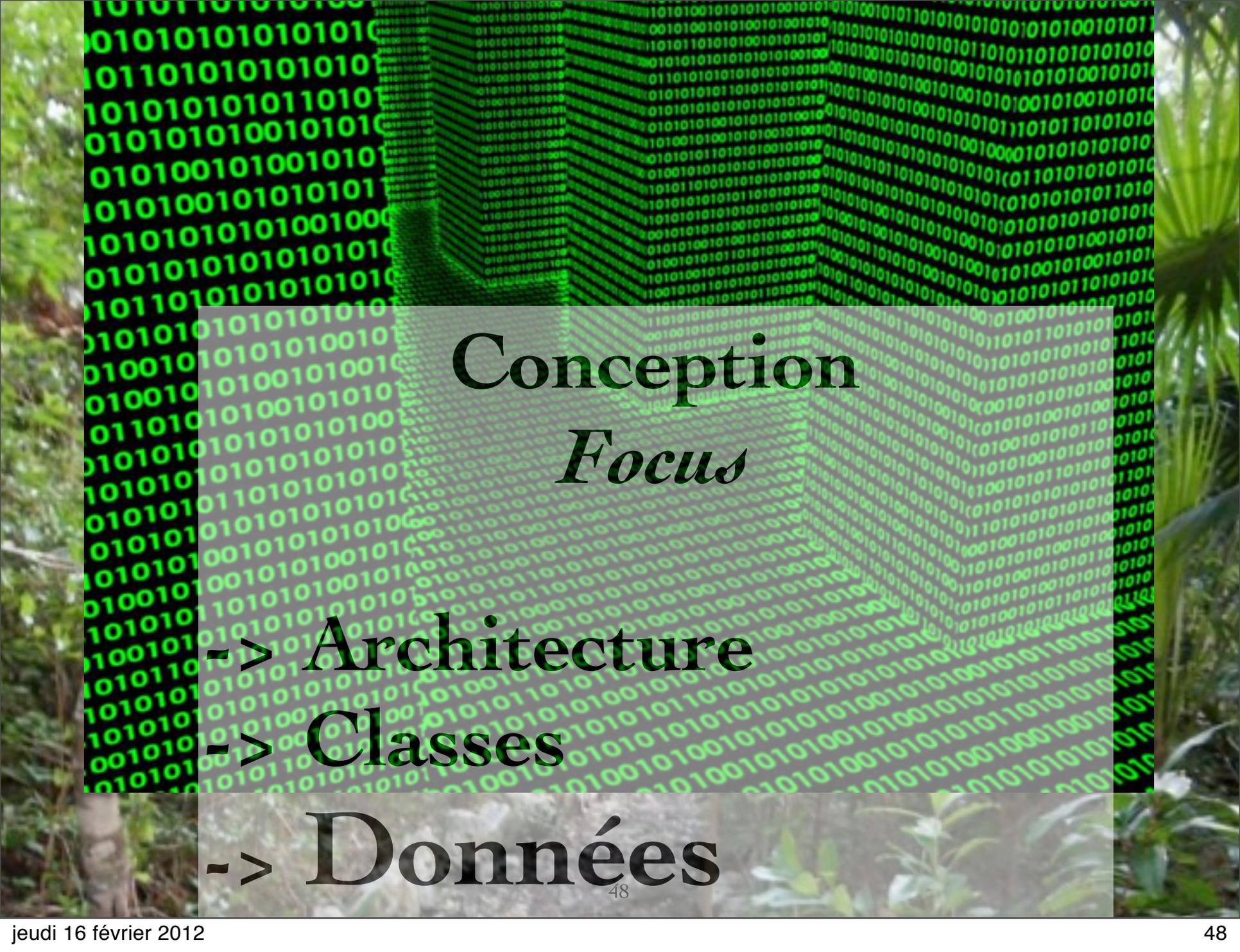


Contrôle du cycle de vie des éléments :

```
constructeur() {
  création objets Roue
  création objet Carrosserie
  création objet Moteur
}
destruteur() {
  destruction objets Roue
  destruction objet Carrosserie
  destruction objet Moteur
}
```

Contrôle du cycle de vie des éléments :

```
constructeur() {
  création objets Porte
  création objet Habitable
}
destruteur() {
  destruction objets Porte
  destruction objet Habitable
}
```



Conception *Focus*

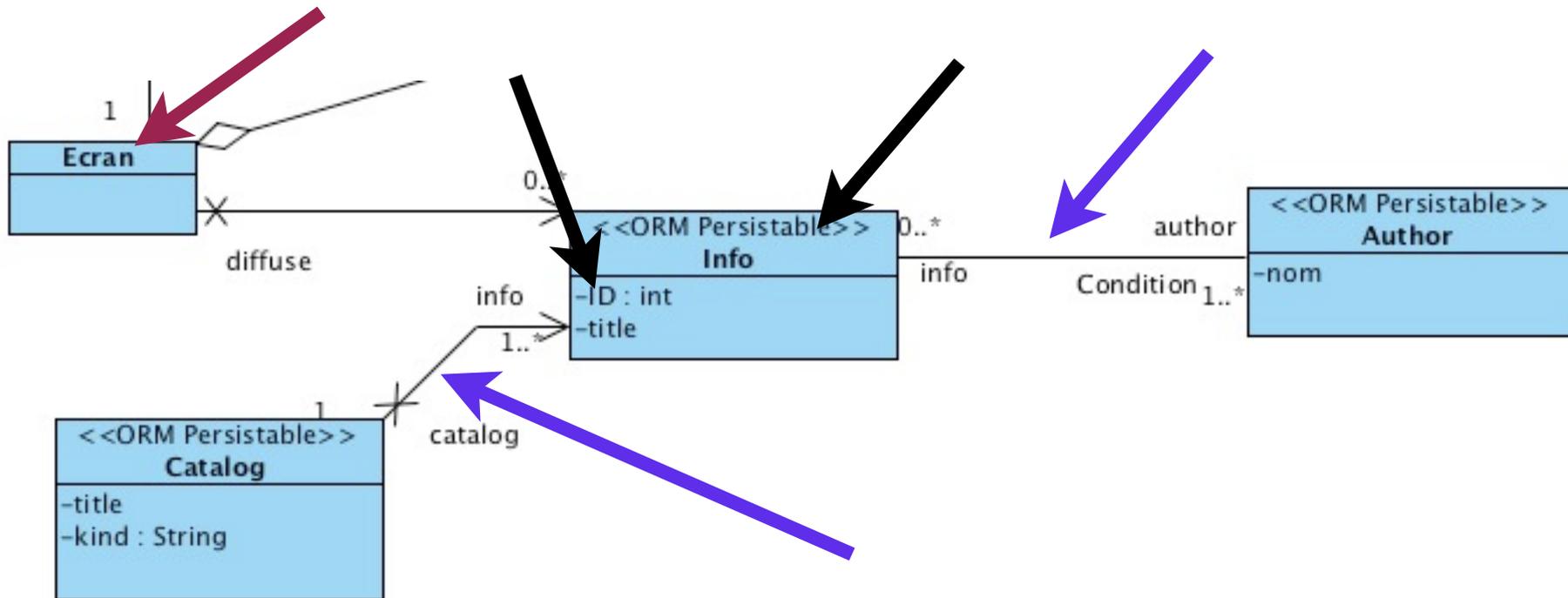
-> Architecture

-> Classes

-> Données

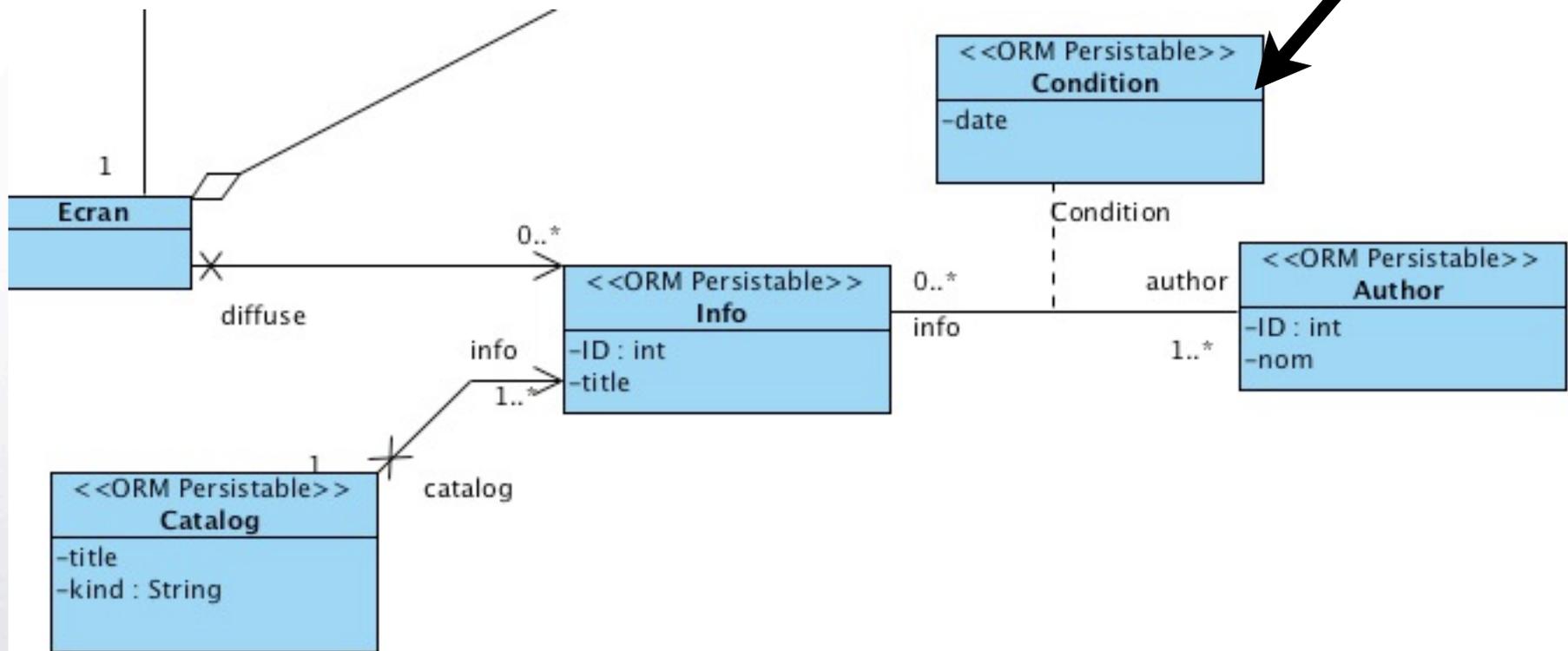


Des classes aux données



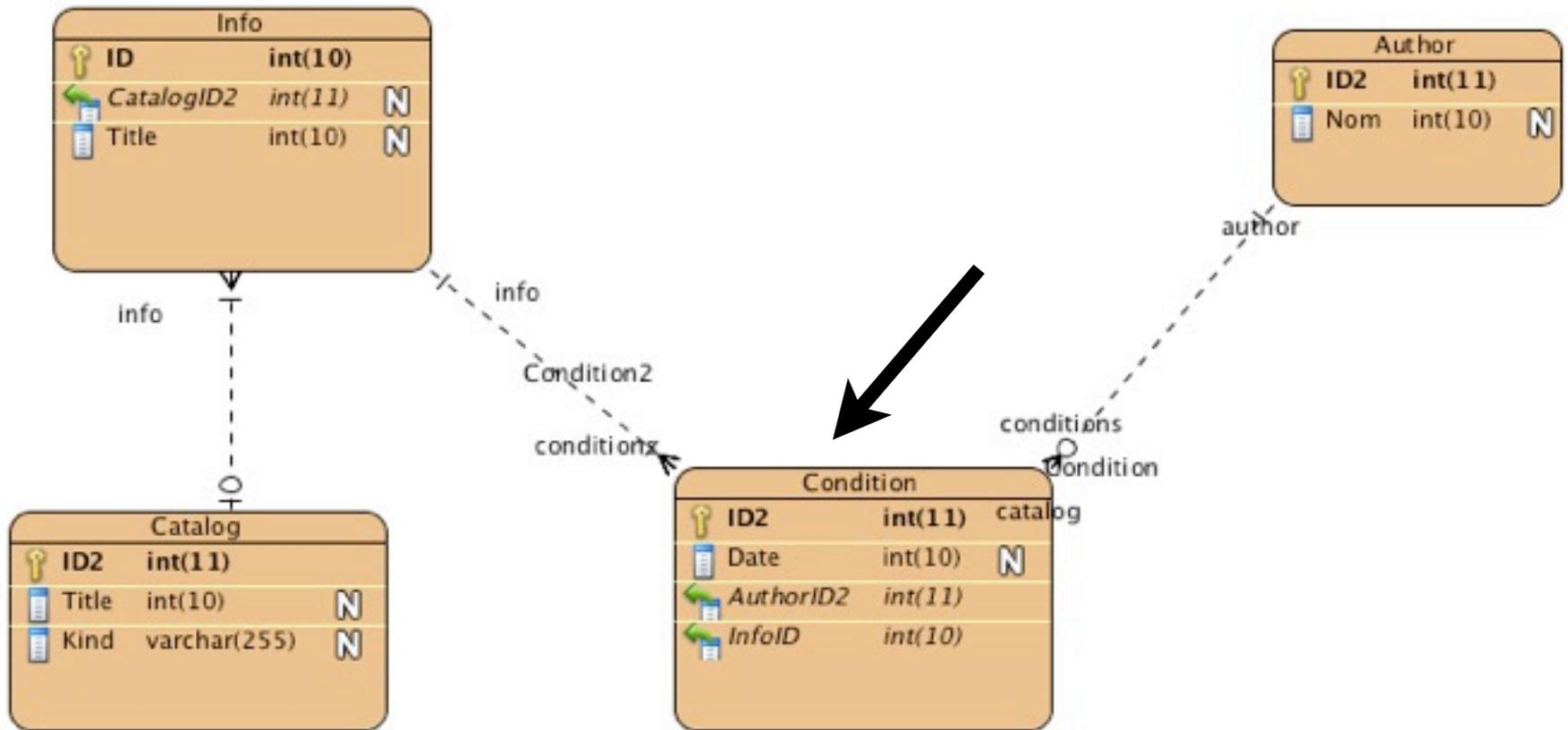


Des classes aux données





Des classes aux données



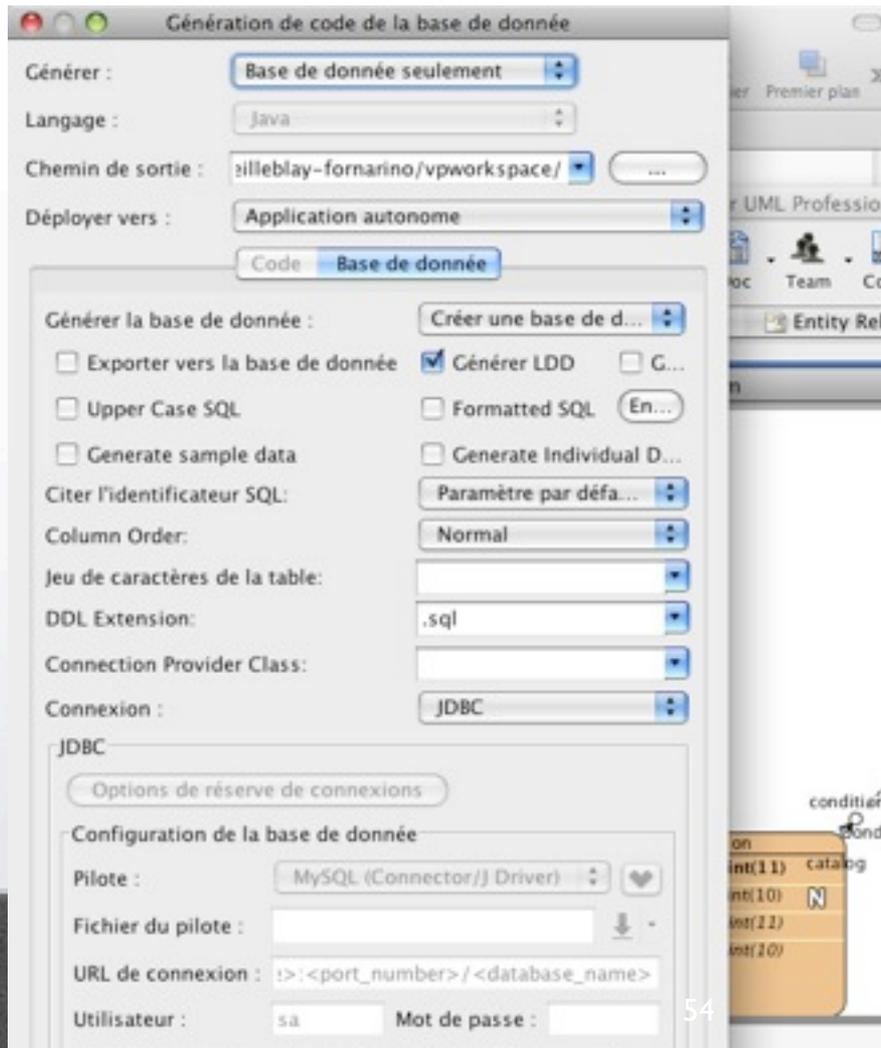


Des classes aux données

The screenshot shows the Visual Paradigm for UML Professional Edition interface. The main window displays a class diagram with three classes: Info, Catalog, and Columns. The Info class has attributes ID (int(10)), CatalogID2 (int(11)), and Title (int(10)). The Catalog class has attributes ID2 (int(11)), Title (int(10)), and Kind (varchar(255)). The Columns class has attributes ID2 (int(11)), Date (int(10)), AuthorID2 (int(11)), and InfoID (int(10)). A context menu is open over the diagram, listing various actions such as 'Assistants...', 'Configuration de la base de donnée...', 'Inverser la base de donnée...', 'Classes Java inverses...', 'Reverse Hibernate...', 'Cadre d'application objet entreprise inverse...', 'Synchroniser vers le diagramme de classe', 'Synchroniser vers le schéma du modèle entité-association', 'Ignore Entities when Synchronizing...', 'Ignore Classes when Synchronizing...', 'Générer la base de donnée...', and 'Générer le code...'. The 'Générer la base de donnée...' option is highlighted.

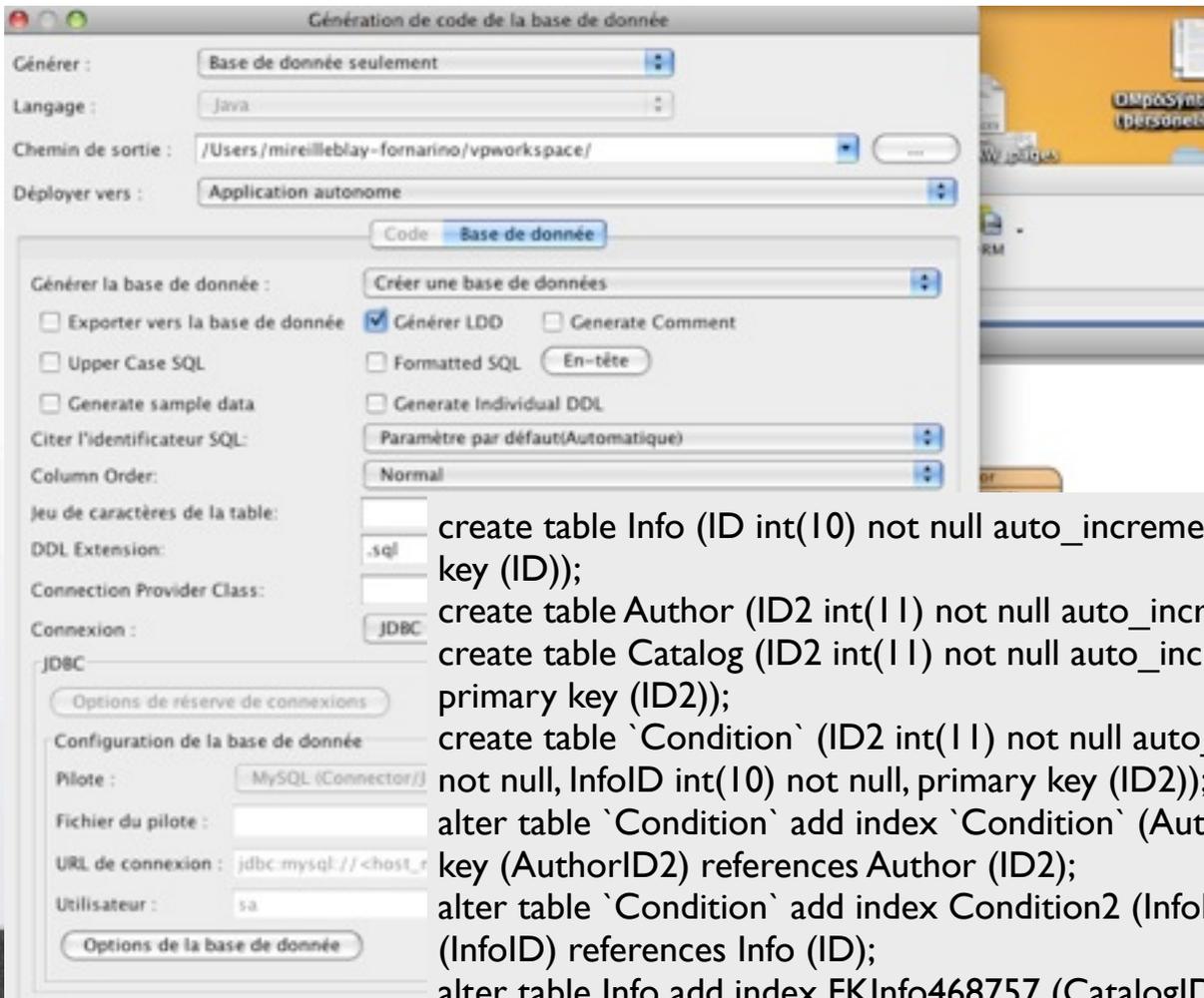


Des classes aux données



54

Des classes aux données



```
create table Info (ID int(10) not null auto_increment, CatalogID2 int(11), Title int(10), primary key (ID));
create table Author (ID2 int(11) not null auto_increment, Nom int(10), primary key (ID2));
create table Catalog (ID2 int(11) not null auto_increment, Title int(10), Kind varchar(255), primary key (ID2));
create table `Condition` (ID2 int(11) not null auto_increment, `Date` int(10), AuthorID2 int(11) not null, Infold int(10) not null, primary key (ID2));
alter table `Condition` add index `Condition` (AuthorID2), add constraint `Condition` foreign key (AuthorID2) references Author (ID2);
alter table `Condition` add index Condition2 (Infold), add constraint Condition2 foreign key (Infold) references Info (ID);
alter table Info add index FKInfo468757 (CatalogID2), add constraint FKInfo468757 foreign key (CatalogID2) references Catalog (ID2);
```



**ApprocheS :
Méthodologies au
prochain cours**