


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 Priebsch, the PHP.cc ZendCon 2010
- Developing Web Applications with PHP, RAD for the World Wide Web,


Approche

Analyse

Diagramme de Use-cases

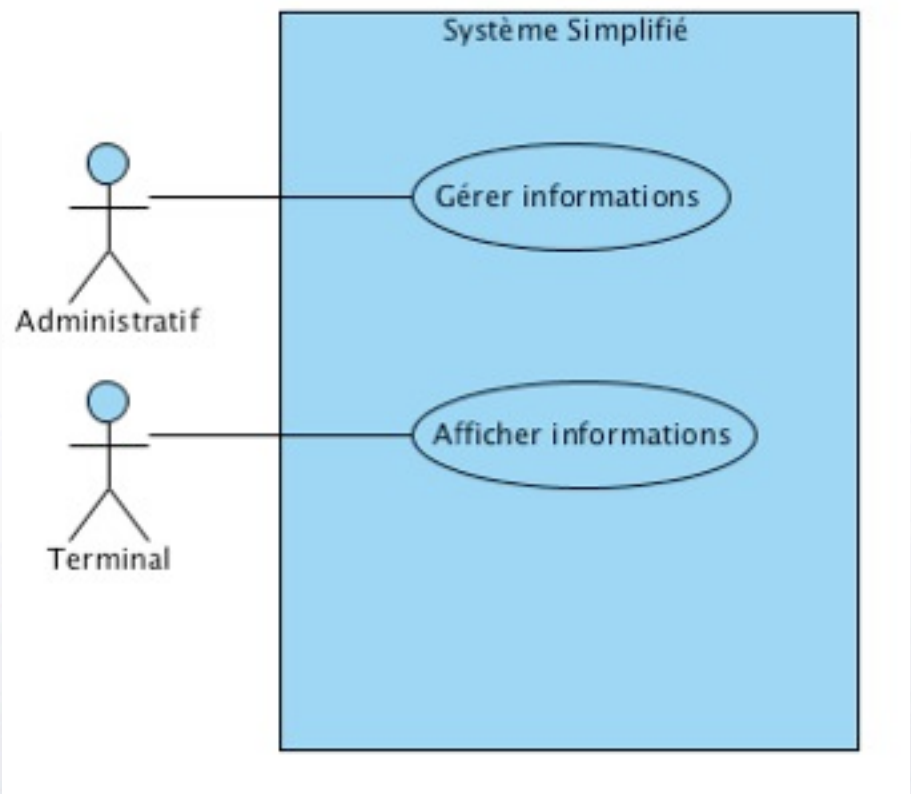
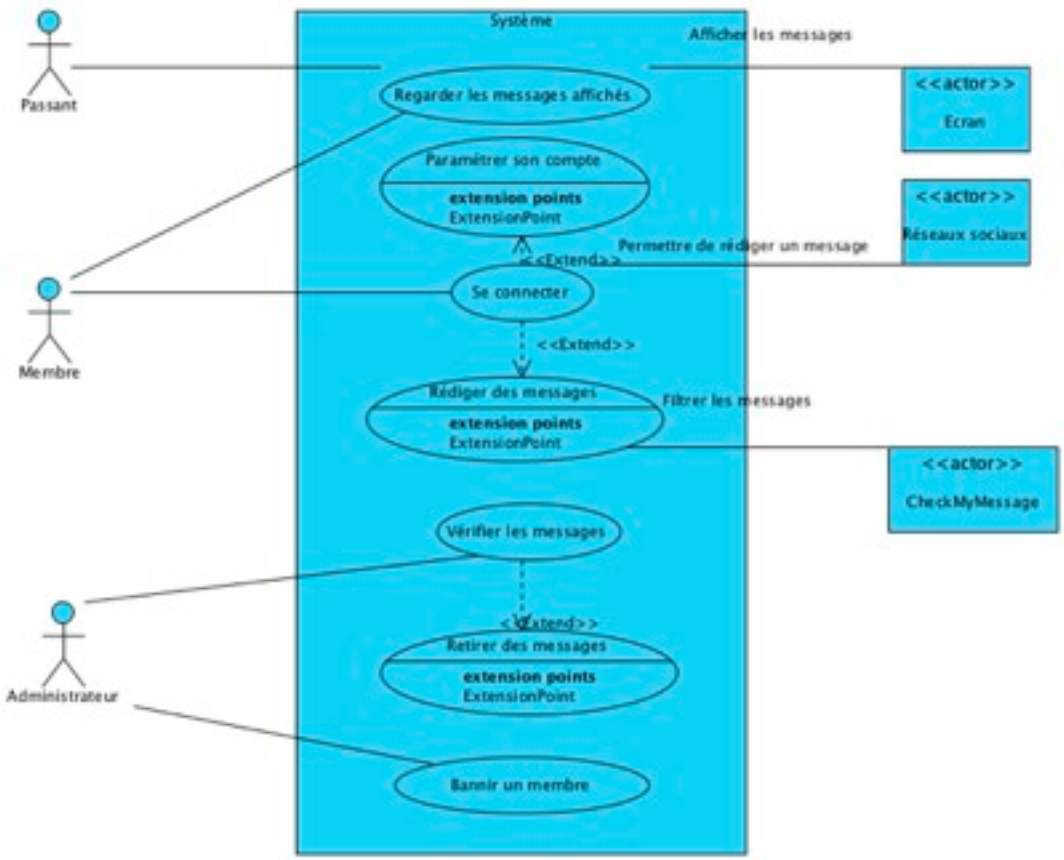
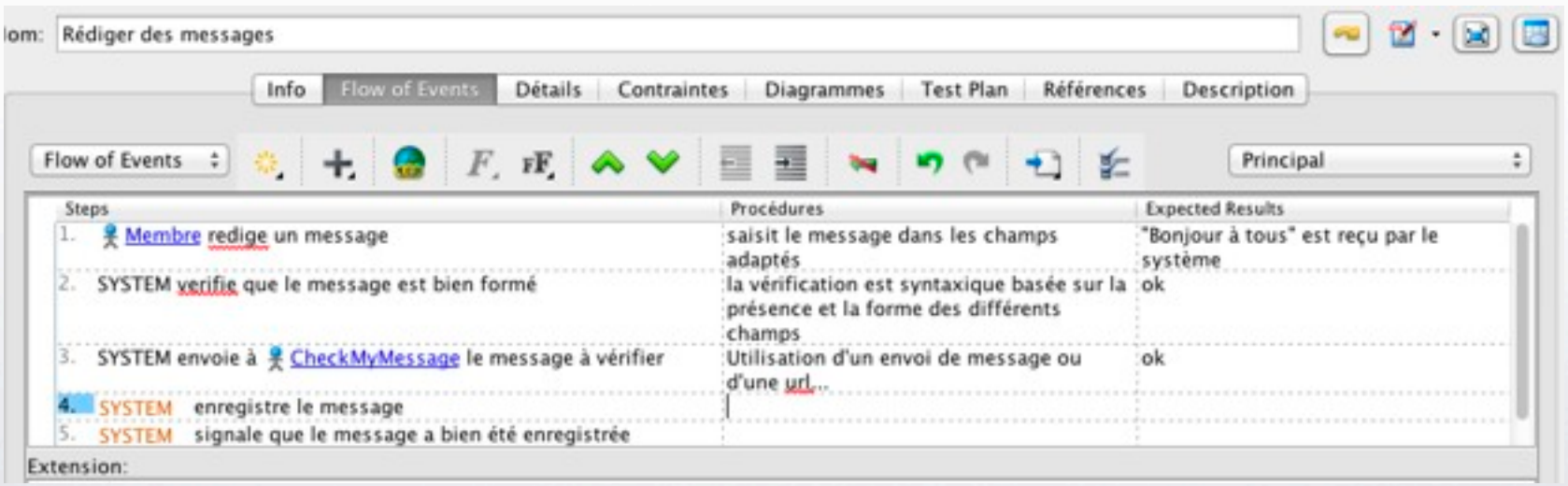


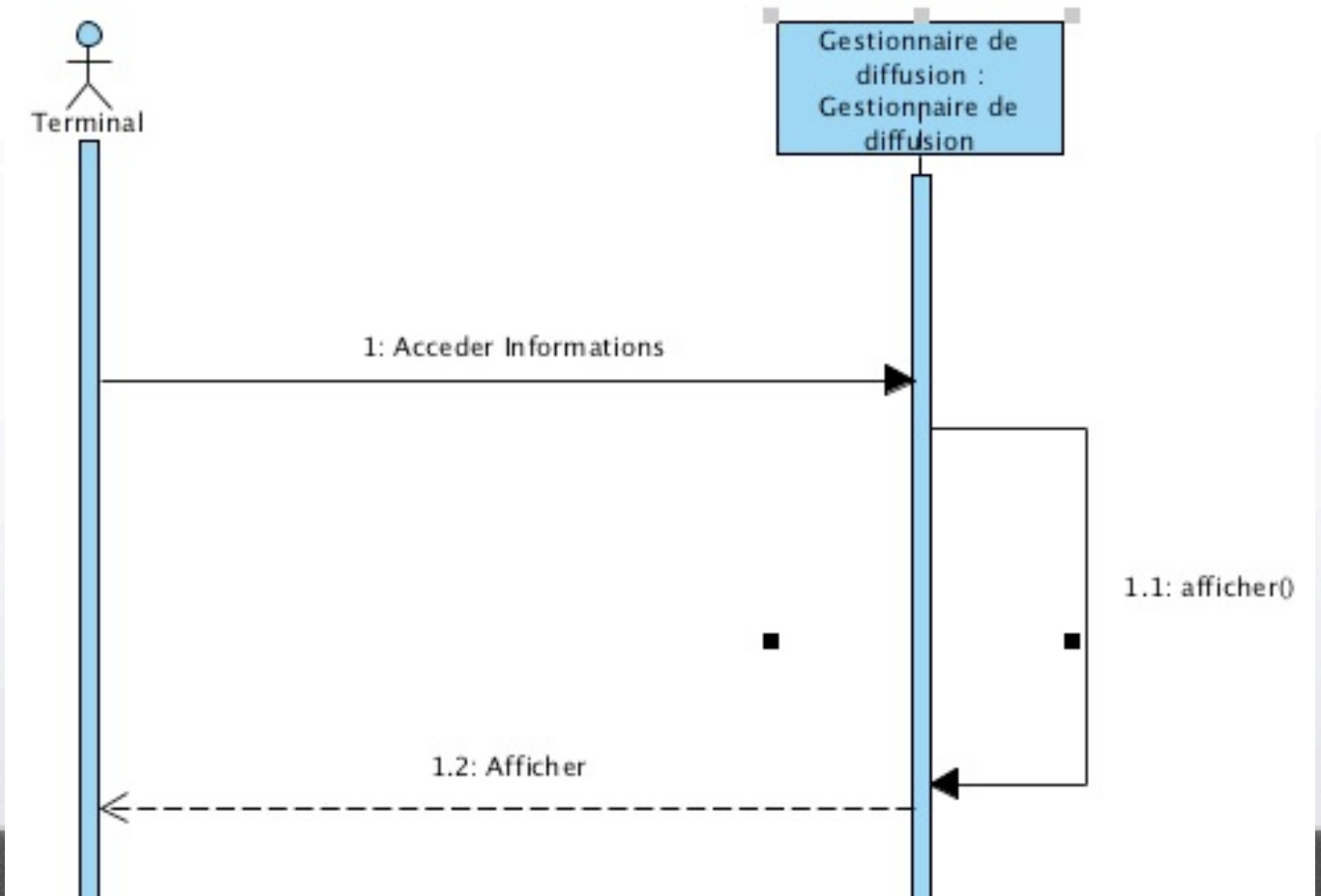
Diagramme de Use-cases



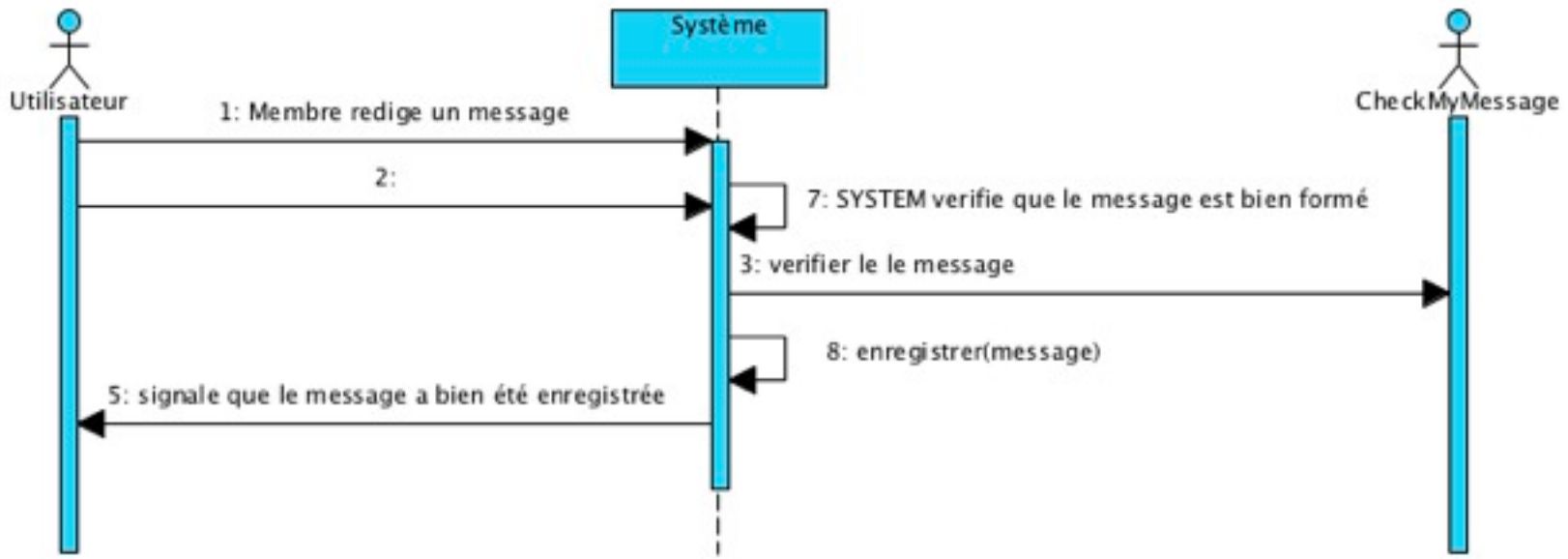
Description des flots d'évènements et préparation des procédures de tests



Visualisation sous la forme de diagramme de séquences : Afficher Informations

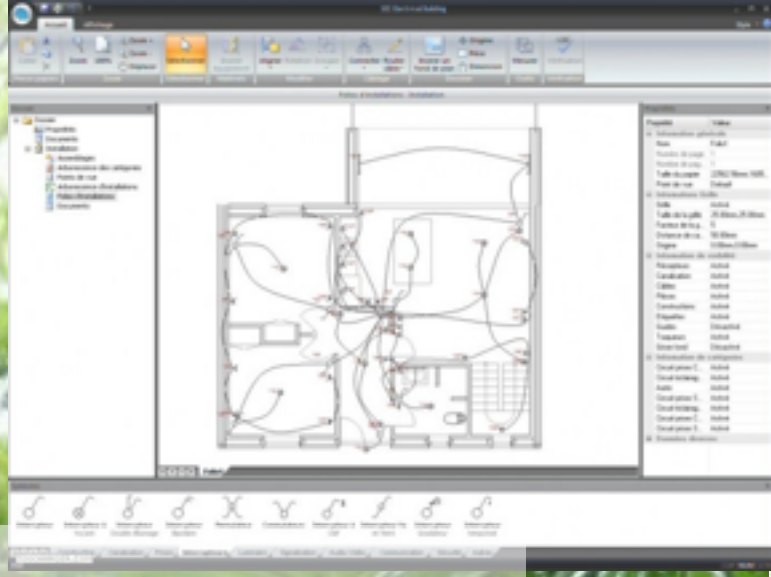


Visualisation sous la forme de diagramme de séquences : rédiger message



L'analyse produit :

- Au moins
  - Les uses cases de plus haut niveau
  - les flots d'évènements associés
  - les premières procédure de tests de validation
  - Des diagrammes de séquences de niveau analyse
- Mais aussi
  - un glossaire initial



Conception

Focus

-> Architecture

-> Classes

-> Données

Conception

Focus

-> Architecture

-> Classes

-> Données

Choix d'Architecture

Présentation

Logique applicative

Gérer les informations

Stockage

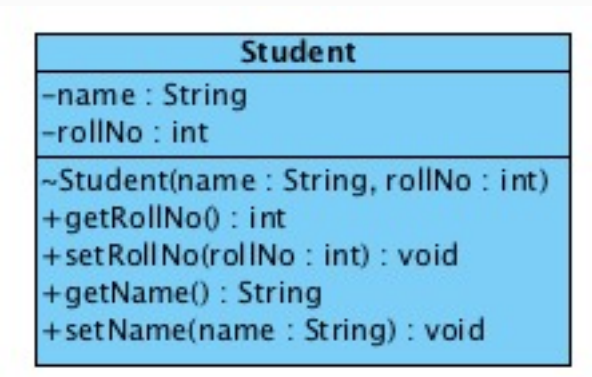
```
CREATE TABLE `information` (
 `titre` varchar(20) NOT NULL,
 `date` varchar(22) NOT NULL,
`identifiant` int(11) NOT NULL auto_increment,
 PRIMARY KEY  (`identifiant`))
```

SEPARATIONS : Persistance et accès aux données : DAO (Data Access Object Pattern)

http://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm

Séparer la logique métier de l'accès aux données

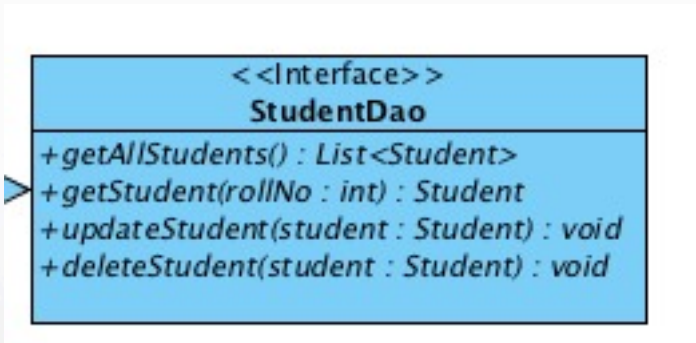
Un étudiant




```java
public class Student {
   private String name;
   private int rollNo;

   Student(String name, int rollNo){
      this.name = name;
      this.rollNo = rollNo;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public int getRollNo() {
      return rollNo;
   }

   public void setRollNo(int rollNo) {
      this.rollNo = rollNo;
   }
}
```

Séparer la logique métier de l'accès aux données

Des étudiants




```java
import java.util.List;

public interface StudentDao {
   public List<Student> getAllStudents();
   public Student getStudent(int rollNo);
   public void updateStudent(Student student);
   public void deleteStudent(Student student);
}
```

Séparer la logique métier de l'accès aux données

USAGE

```
StudentDao studentDao = ....

//print all students
for (Student student : studentDao.getAllStudents()) {
   System.out.println("Student: [RollNo : "
      +student.getRollNo()+", Name : "+student.getName()+" ]");
}

//update student
Student student =studentDao.getAllStudents().get(0);
student.setName("Michael");
studentDao.updateStudent(student);

//get the student
studentDao.getStudent(0);
System.out.println("Student: [RollNo : "
   +student.getRollNo()+", Name : "+student.getName()+" ]");
   }
}
```

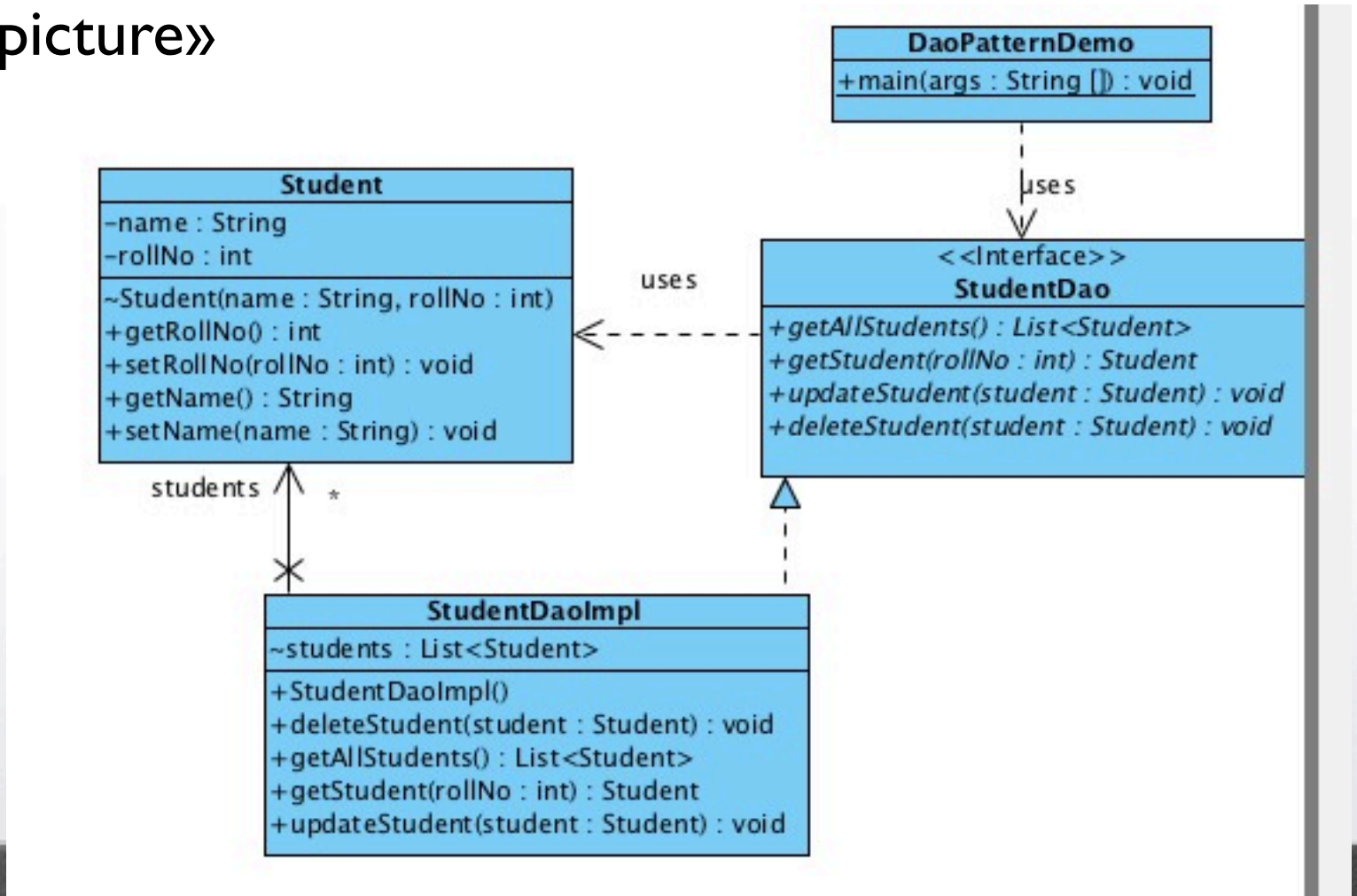
Séparer la logique métier de l'accès aux données

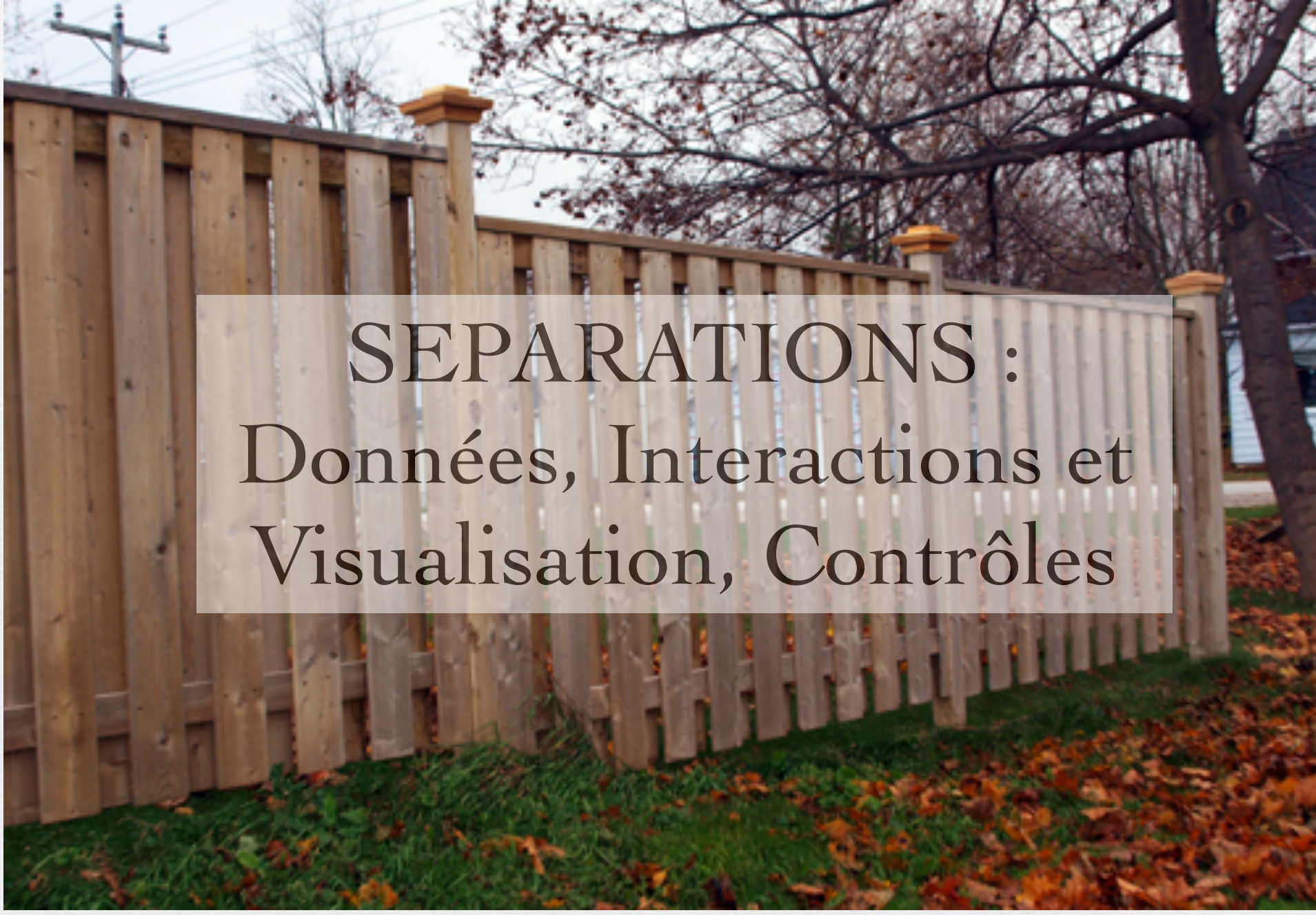
Connexion à la BD

```
import java.util.ArrayList;
import java.util.List;

public class StudentDaoImpl implements
StudentDao {

   //list is working as a database
   List<Student> students;

   public StudentDaoImpl(){
      students = new ArrayList<Student>();
      Student student1 = new
Student("Robert",0);
      Student student2 = new Student("John",
1);
      students.add(student1);
      students.add(student2);
   }
   @Override
   public void deleteStudent(Student
student) {
      students.remove(student.getRollNo());
      System.out.println("Student: Roll No "
+ student.getRollNo()
         +", deleted from database");
   }

   //retrive list of students from the
database
   @Override
   public List<Student> getAllStudents() {
      return students;
   }

   @Override
   public Student getStudent(int rollNo) {
      return students.get(rollNo);
   }

   @Override
   public void updateStudent(Student
student) {
```

Séparer la logique métier de l'accès aux données

«Big picture»




SEPARATIONS :
Données, Interactions et
Visualisation, Contrôles

Modèle-Vue-Controleur (MVC)

Controller

contrôleur: chef d'orchestre

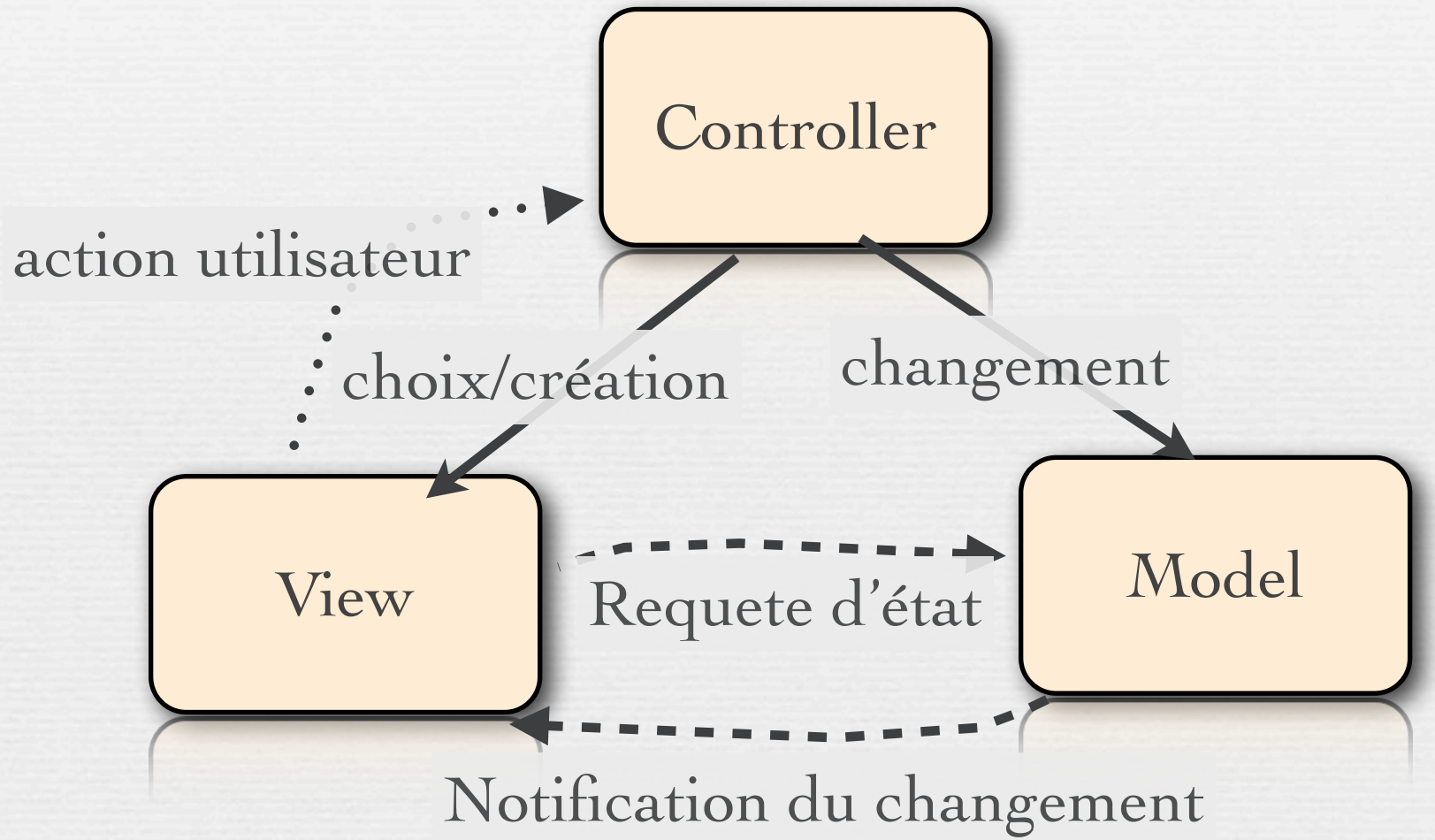
View

La vue: présentée à l'utilisateur

Model

Le modèle: les données indépendantes

MVC


Controller
action utilisateur
choix/création
changement
View
Requete d'état
Model
Notification du changement

Exemple simpliste en php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//FR"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en_US" xml:lang=
"en_US">
  <head>
    <title>
      Exemple
    </title>
  </head>
  <body>
  <?php
    $db=mysql_connect($dbhost,$dbuser,$dbpass);
    mysql_select_db($dbname,$db);
    $query="SELECT product.name, product.price FROM product where product.type=42";
   $req=mysql_query($query)or die('Erreur SQL !<br>'.$sql.'<br>'.mysql_error());
   while ($row = mysql_fetch_array($req)) {
   echo "Nom :".$row[0]." Prix :".$row[1];
}
    mysql_close();
  ?>
  </body>
</html>
```

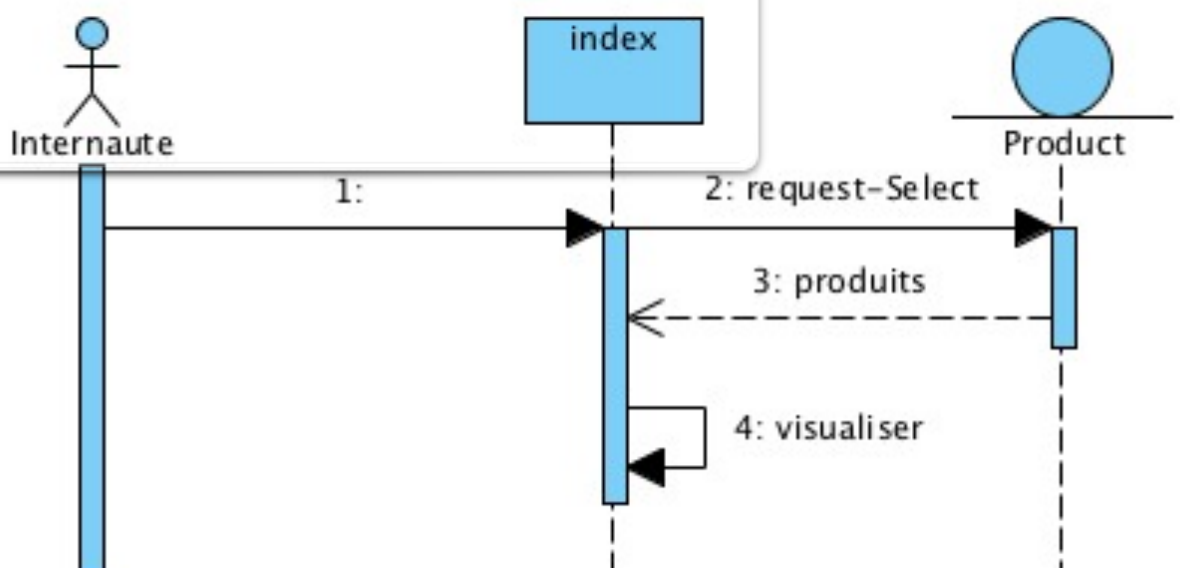
Mise en page

Accès aux données

http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

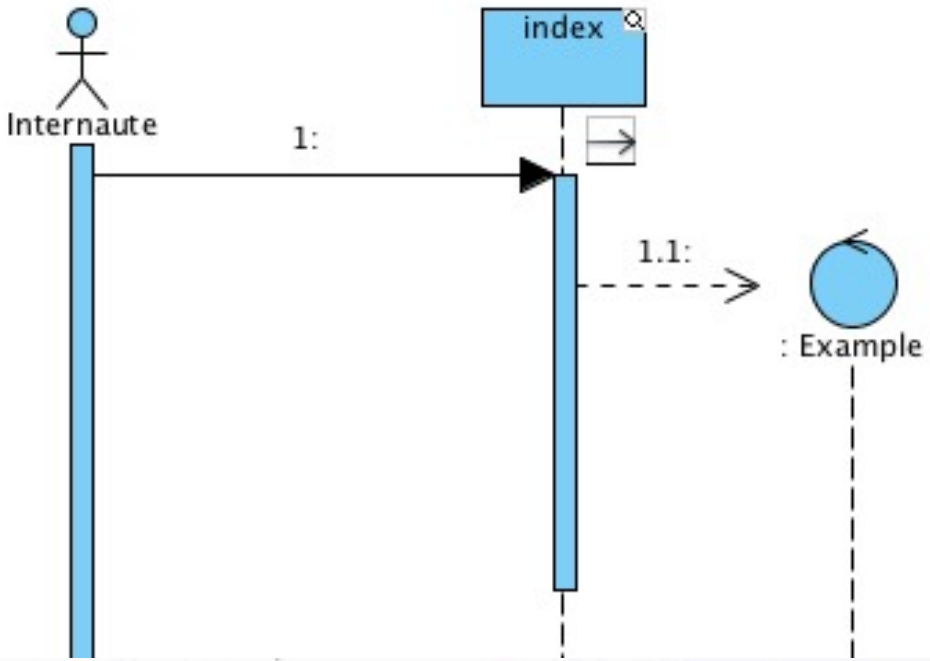
Exemple simpliste en php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//FR"
...
  <head>
...
  </head>
  <body>
  <?php
    $db=mysql_connect($dbhost,$dbuser,$dbpass);
    mysql_select_db($dbname,$db);
    $query="SELECT product.name, product.price FROM product where
product.type=42";
   $req=mysql_query($query)or die('Erreur SQL!<br>'.
$sql.'<br>'.mysql_error());
   while ($row = mysql_fetch_array($req)) {
   echo "Nom :".$row[0]." Prix :".$row[1];
}
    mysql_close();
  ?>
  </body>
</html>
```



Version MVC : entrée

index.php:



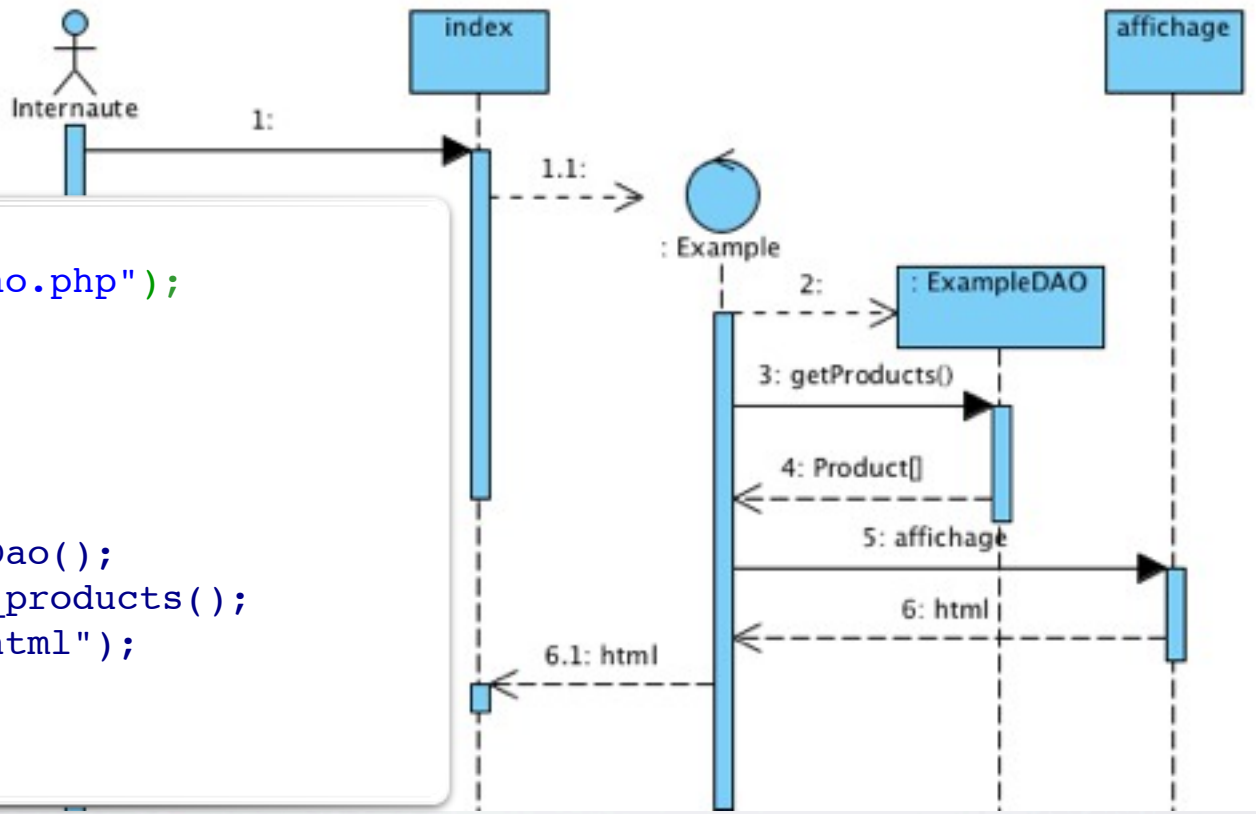

```php
<?php
if  ($_GET['do']=="")
{
  require ("default.html");
}
else
{
  if($_GET['do']=="affichage")
  {
    require("action/class_example.php");
    new Example();
  }
}
?>
```

http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

Contrôleur

class_example.php:

```php
<?php
 require ("dao/dao_example_dao.php");

  class Example
 {
  public function example()
  {
    $exampleDao= new ExampleDao();
    $data = $exampleDao->get_products();
    require("web/affichage.phtml");
  }
}
?>
```



Il ne fait pas grand chose...

http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

Visualisation

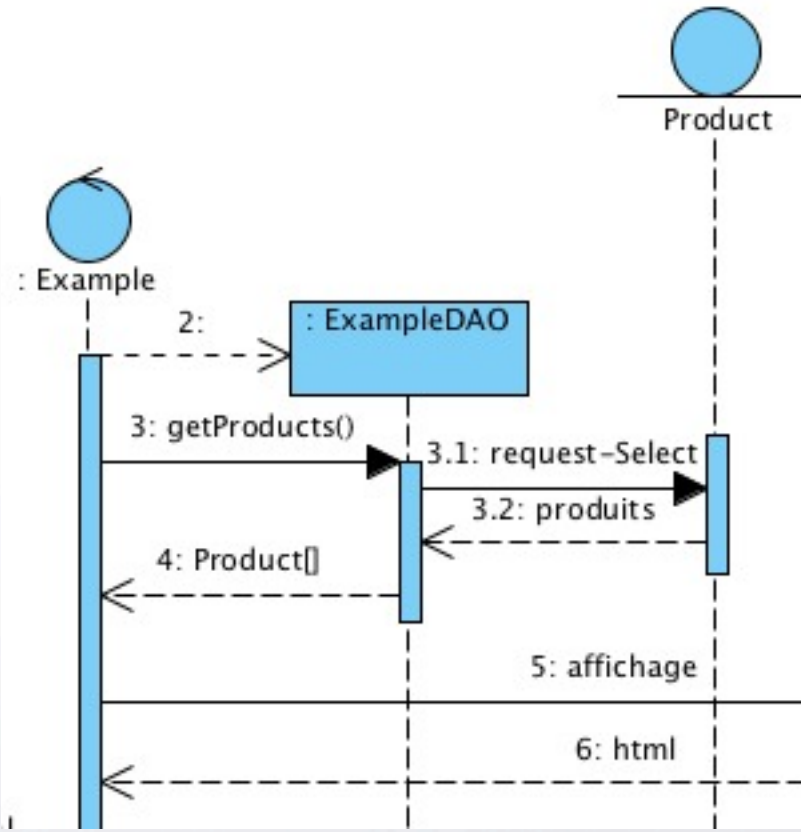
affichage.phtml:

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="en_US" xml:lang="en_US">
  <head>
    <title>
      Exemple
    </title>
  </head>
  <body>
    <?
foreach ($data AS $row )
{
?>
   Nom :
<? echo $row["name"]; ?>
 Prix :
<? echo $row["price"]; ?> <br/>
<?
}
?>
  </body>
</html>
```

http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

Accès aux données : DAO (Data Access Object)

```php
<?php
 require ("class_connect_bd.php");
 class ExampleDao
 {
   public  function get_products()
   {
    $sql="SELECT product.name,
                  product.price
          FROM product where product.type=42";
    $query = mysql_query($sql) or
          die('Erreur SQL !<br>'.mysql_error());
    $all =  mysql_fetch_all($query);
    return $all;
   }
 }
?>
```

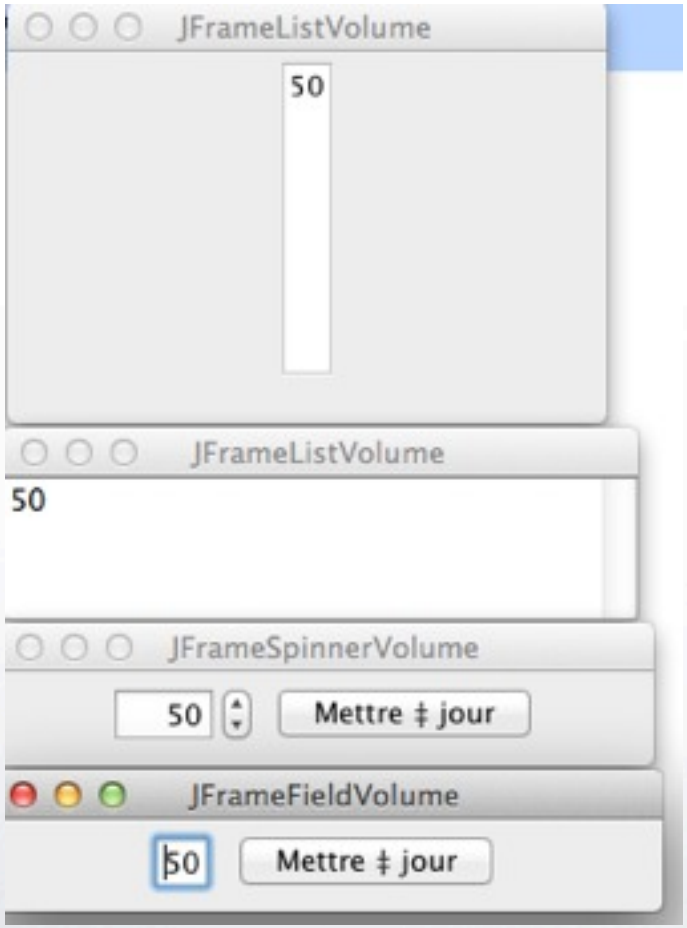


dao_example_dao.php

C'est tellement simple qu'il est assimilé au modèle dans cet exemple.

http://blog.nalis.fr/index.php?post/2009/10/19/Architecture-%3A-Le-Design-Pattern-MVC-en-PHP

Exemple en java



- Une vue listant les différents volumes et qui ajoutera chaque nouveau volume dans une liste déroulante : JFrameListVolume
- Une vue permettant de modifier le volume à l'aide d'un spinner avec un bouton permettant de valider le nouveau volume : JFrameSpinnerVolume
- Une vue permettant de modifier le volume avec un champ texte avec un bouton permettant de valider le nouveau volume : JFrameFieldVolume

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Modèle

```
public class VolumeModel {
    private int volume;

    public VolumeModel(){
        super();
        volume = 0;
    }

    public int getVolume() {
        return volume;
    }

    public void setVolume(int volume)
{
        this.volume = volume;
    }
}
```

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Modèle «observable*»

```
import
javax.swing.event.EventListenerList;

public class VolumeModel {
  private int volume;
  private EventListenerList listeners;

  public VolumeModel(){
    this(0);
  }
  public VolumeModel(int volume){
    super();
    this.volume = volume;
    listeners = new EventListenerList();
  }

  public void setVolume(int volume) {
        this.volume = volume;
        fireVolumeChanged();
    }
```

```
public void
addVolumeListener(VolumeListener
listener){
 listeners.add(
     VolumeListener.class,listener);
}
```

```
public void fireVolumeChanged(){
  VolumeListener[] listenerList =
(VolumeListener[])listeners.getListener
s(VolumeListener.class);
  for(VolumeListener listener :
listenerList){
    listener.volumeChanged(new
VolumeChangedEvent(this, getVolume()));
        }
    }
}
```

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Evénements ...

```
import java.util.EventListener;

public interface VolumeListener extends EventListener {
    public void volumeChanged(VolumeChangedEvent event);
}

import java.util.EventObject;

public class VolumeChangedEvent extends EventObject{
    private int newVolume;

    public VolumeChangedEvent(Object source, int newVolume){
        super(source);

        this.newVolume = newVolume;
    }

    public int getNewVolume(){
        return newVolume;
    }
}
```

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Vue «abstraite» à l'écoute*

```
public abstract class VolumeView implements VolumeListener{
    private VolumeController controller = null;

    public VolumeView(VolumeController controller){
        super();
        this.controller = controller;
    }

    public final VolumeController getController(){
        return controller;
    }

    public abstract void display();
    public abstract void close();
}
```

Contrôleur

```
public class VolumeController {
  public VolumeView fieldView = null;
  public VolumeView spinnerView = null;
  public VolumeView listView = null;

  private VolumeModel model = null;

  public VolumeController (VolumeModel
model){
    this.model = model;

    fieldView = new
JFrameFieldVolume(this,model.getVolume());
    spinnerView = new
JFrameSpinnerVolume(this,model.getVolume()
);
    listView = new JFrameListVolume(this,
model.getVolume());

    addListenersToModel();
  }
```

```
  private void addListenersToModel() {
    model.addVolumeListener(fieldView);
    model.addVolumeListener(spinnerView);
    model.addVolumeListener(listView);
  }

  public void displayViews(){
    fieldView.display();
    spinnerView.display();
    listView.display();
  }

  public void closeViews(){
    fieldView.close();
    spinnerView.close();
    listView.close();
  }

  public void notifyVolumeChanged(int
volume){
    model.setVolume(volume);
  }
```

Vue (réactive)

```
public class JFrameListVolume
extends VolumeView {
     ....
  public
JFrameListVolume(VolumeController
controller) {
    this(controller, 0);
  }

  public
JFrameListVolume(VolumeController
controller, int volume){
    super(controller);
    buildFrame(volume);
  }

  private void buildFrame(int
volume) {
    ...
    jListModel.addElement(volume);
        list....
  }
```

```
  @Override
  public void close() {
    frame.dispose();
  }

  @Override
  public void display() {
    frame.setVisible(true);
  }

  public void
volumeChanged(VolumeChangedEvent
event) {
jListModel.addElement(event.getNewVo
lume());
  }
}
```

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Vue «réactive et active»

```
public class JFrameFieldVolume extends
VolumeView impl  ements ActionListener{
  ........

  public
JFrameFieldVolume(VolumeController
controller) {
    this(controller, 0);
  }

  public
JFrameFieldVolume(VolumeController
controller, int volume){
    super(controller);
    buildFrame(volume);
  }

  private void buildFrame(int volume) {
    frame = new JFrame();
    .....
    field.setValue(volume);
    .....
  }
```

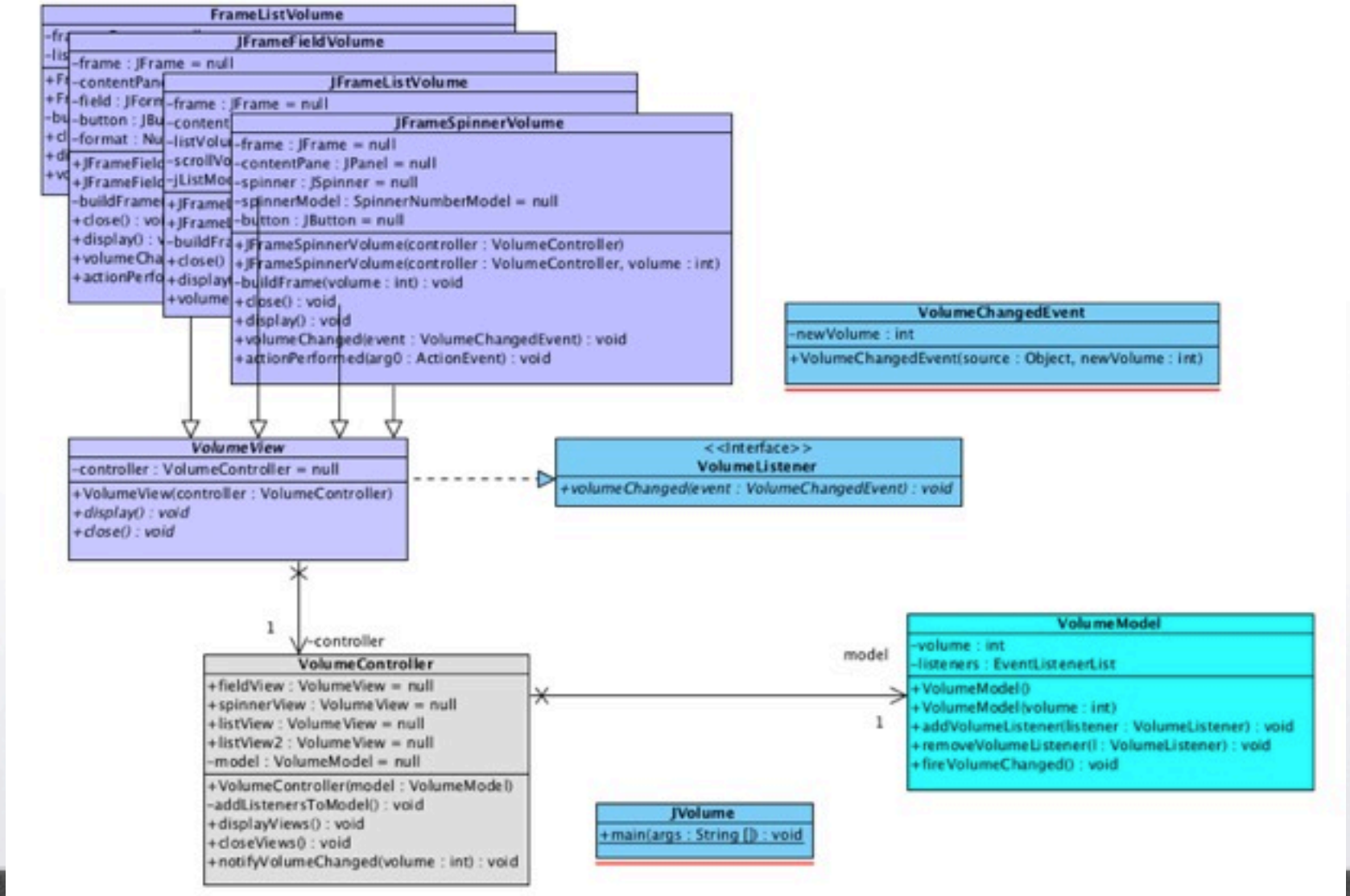
```
  @Override
  public void close() {
    frame.dispose();
  }

  @Override
  public void display() {
    frame.setVisible(true);
  }

  public void
volumeChanged(VolumeChangedEvent event)
{
field.setValue(event.getNewVolume());
  }

  public void
actionPerformed(ActionEvent arg0) {
   getController().
     notifyVolumeChanged(
Integer.parseInt(field.getValue().toStr
ing()));
  }
```

Lanceur

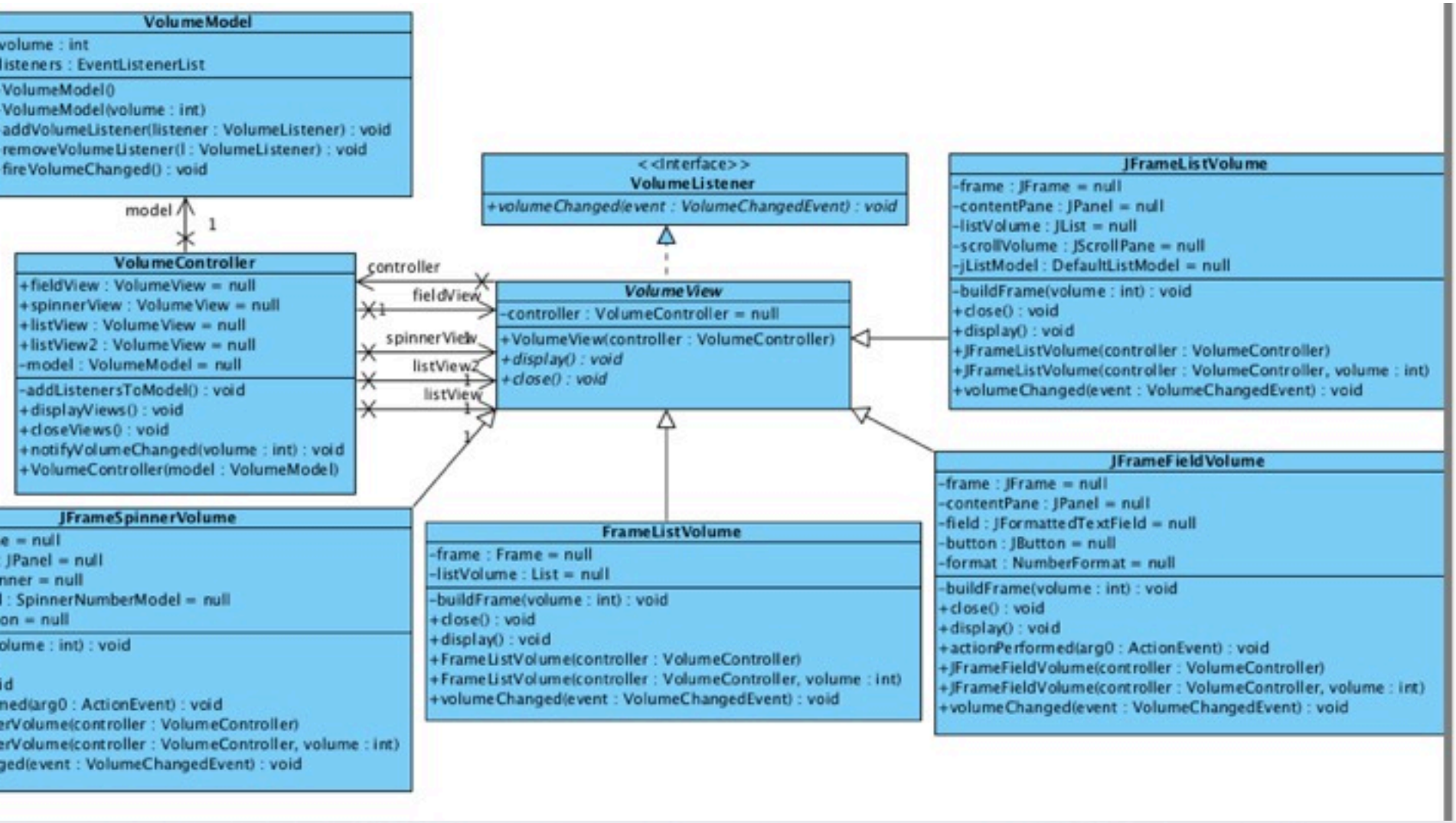
```
package volume;

public class JVolume {
  public static void main(String[] args) {
    VolumeModel model = new VolumeModel(50);
    VolumeController controller = new VolumeController(model);
    controller.displayViews();
  }
}
```

http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

Vue UML des classes

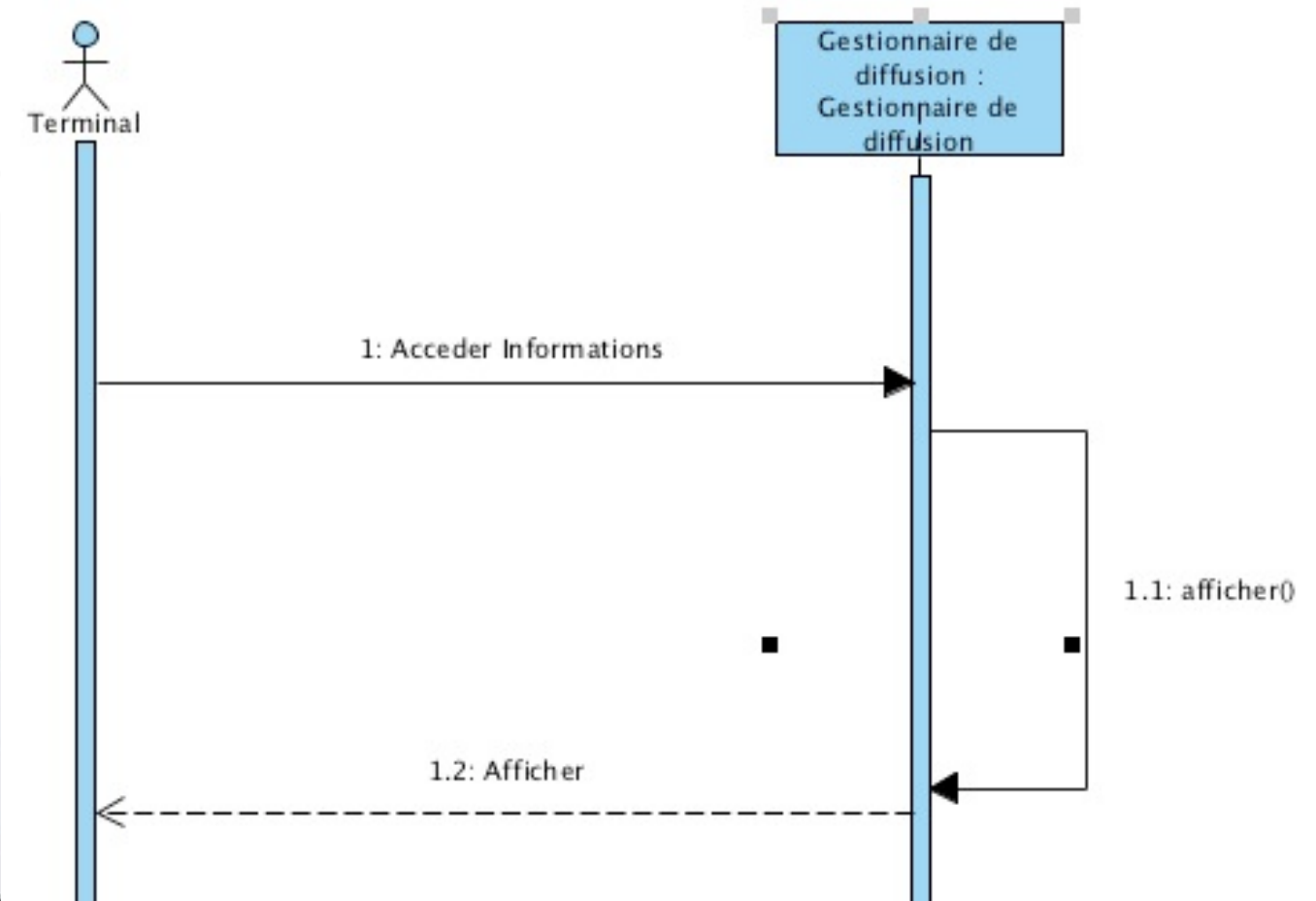


Reverse-Engineering

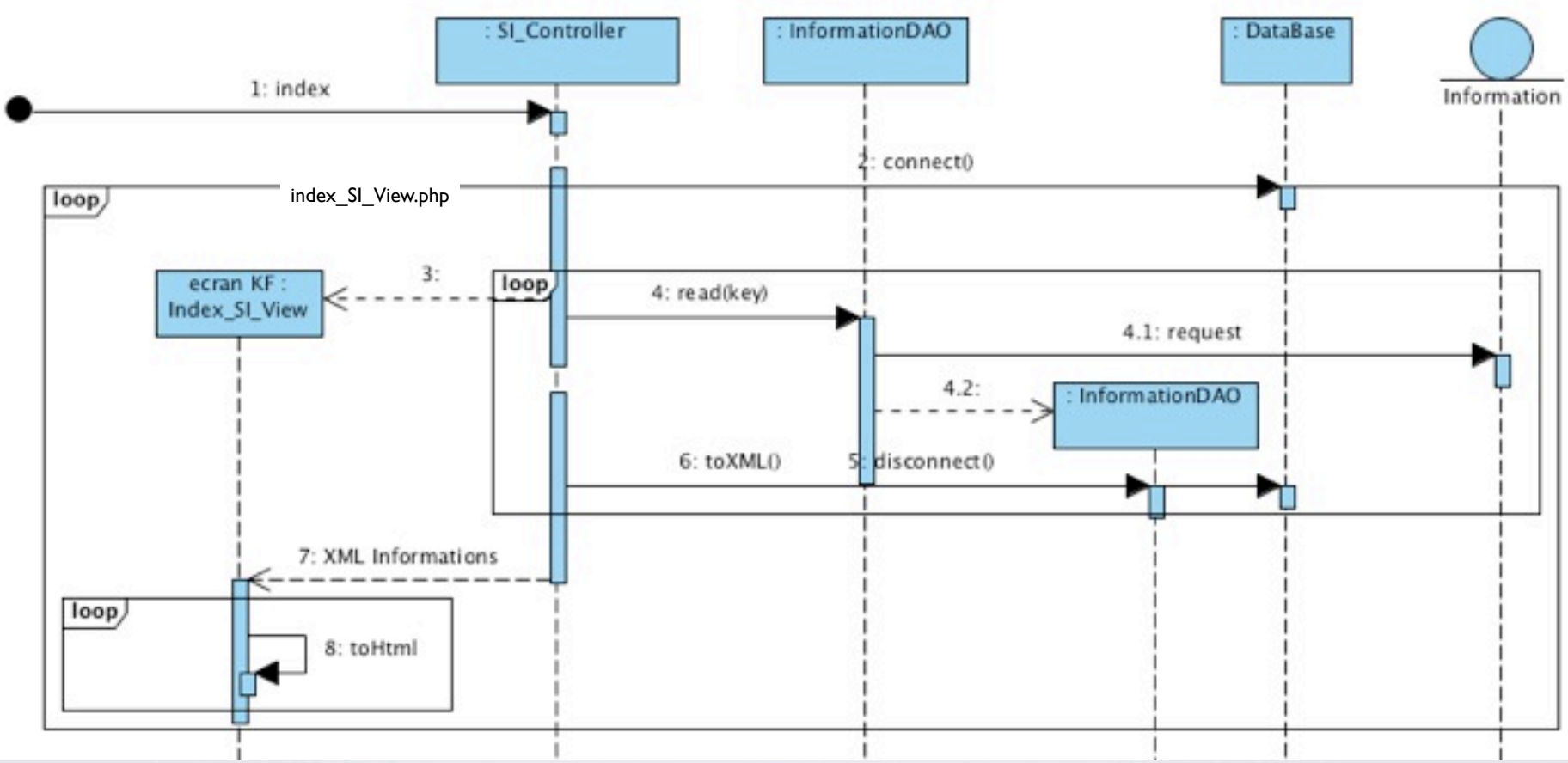


http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

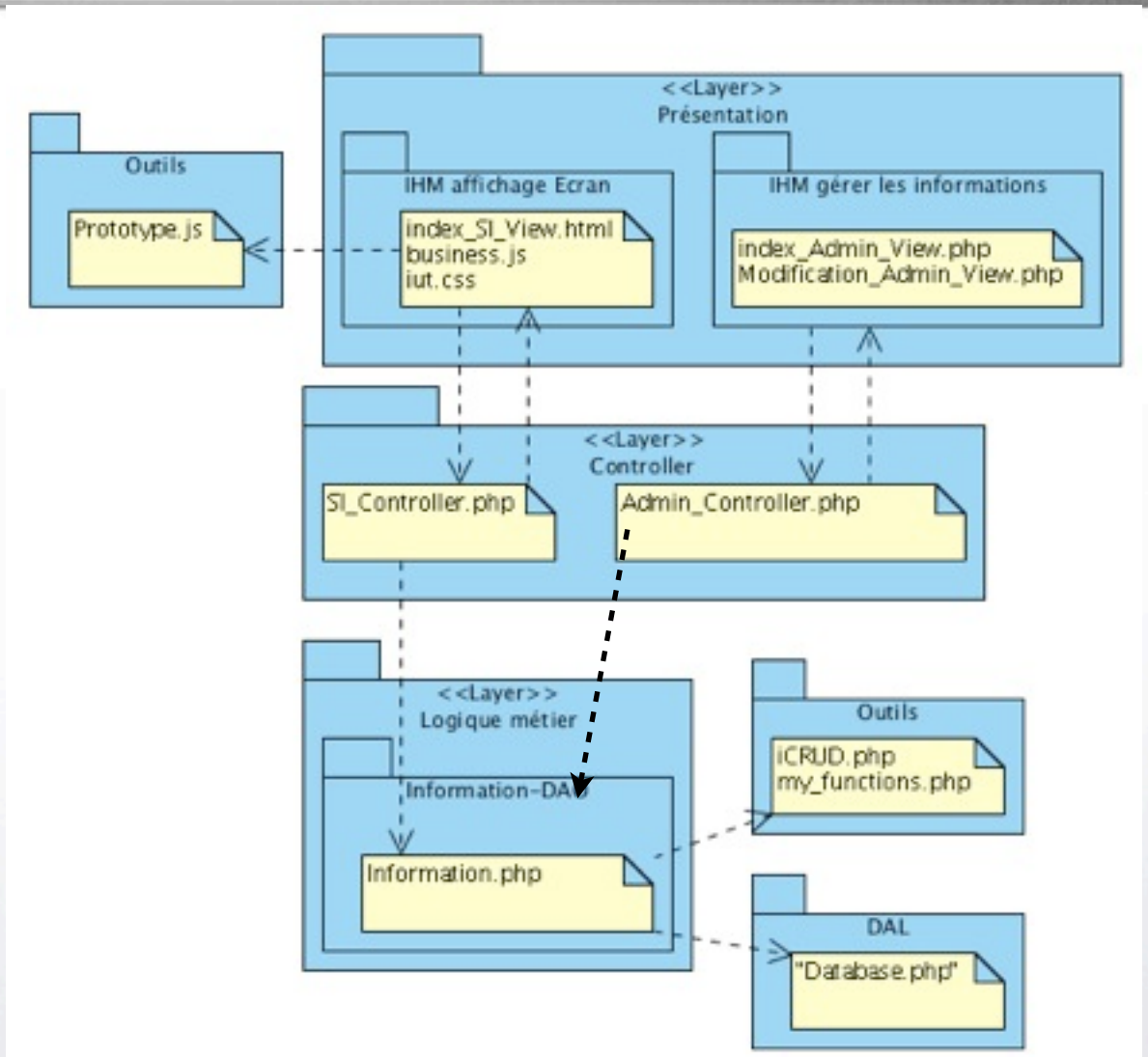
Afficher Informations : niveau Analyse



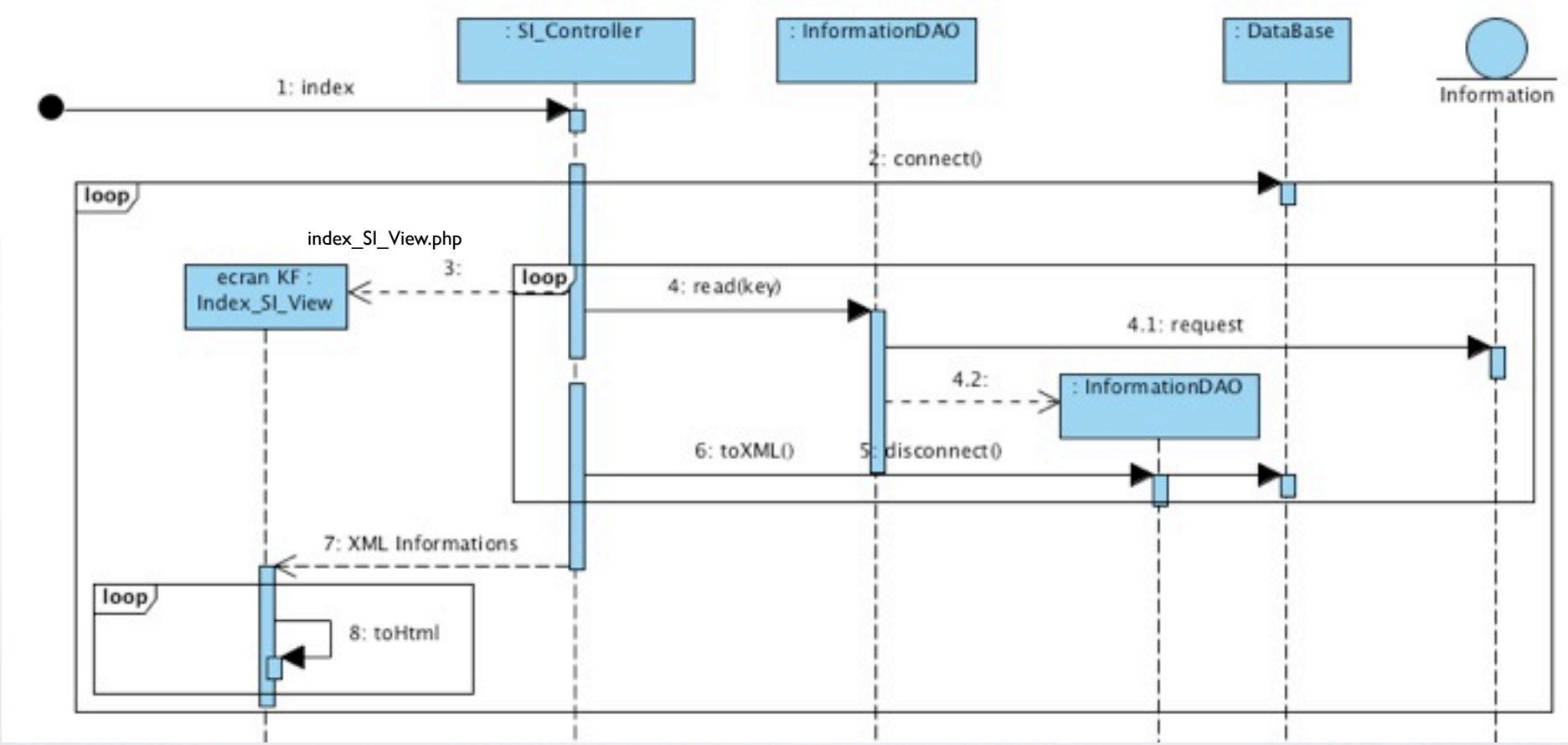
Afficher Informations : niveau Conception



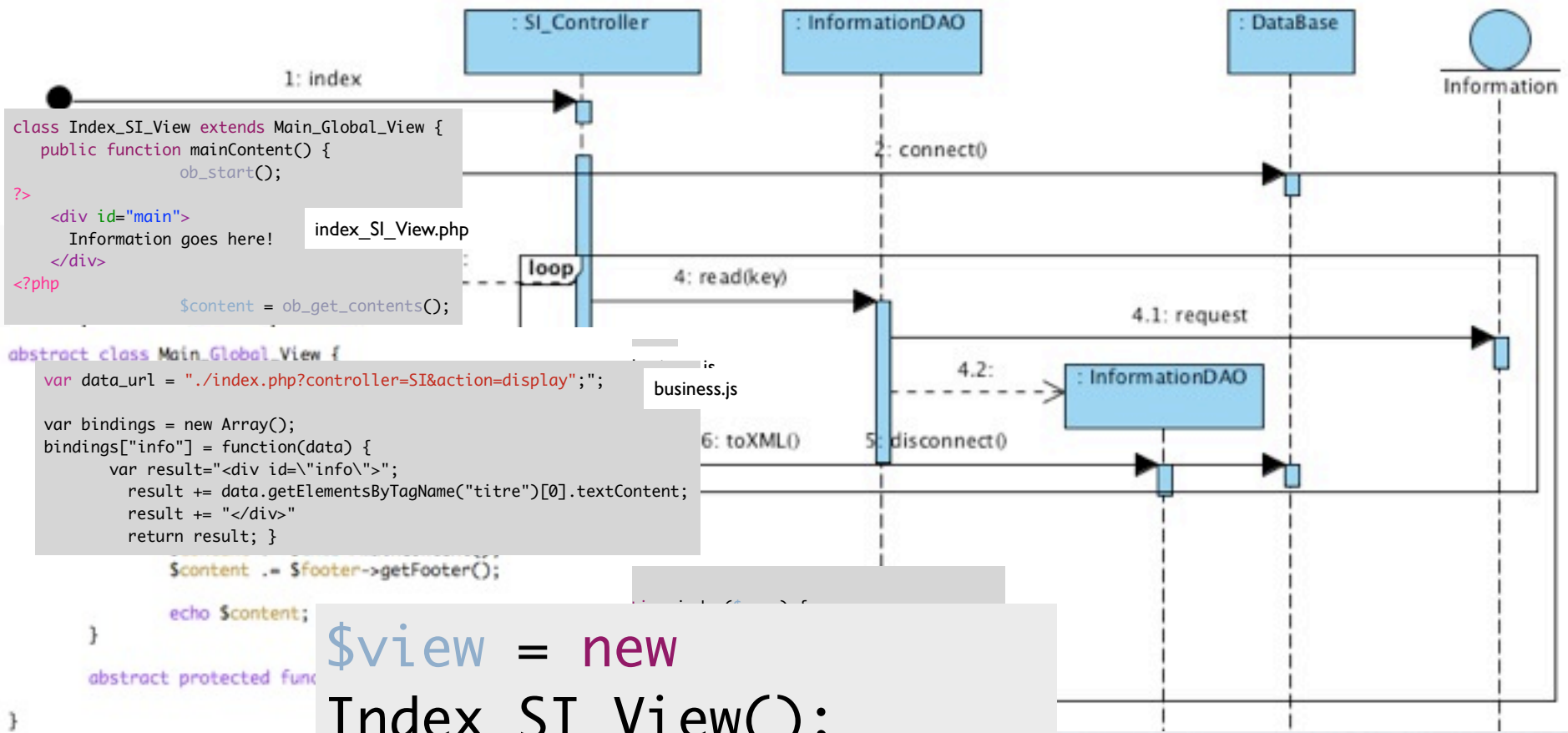
Architecture en couches & Fichiers



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();
```

index_SI_View.php

```
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; }
```

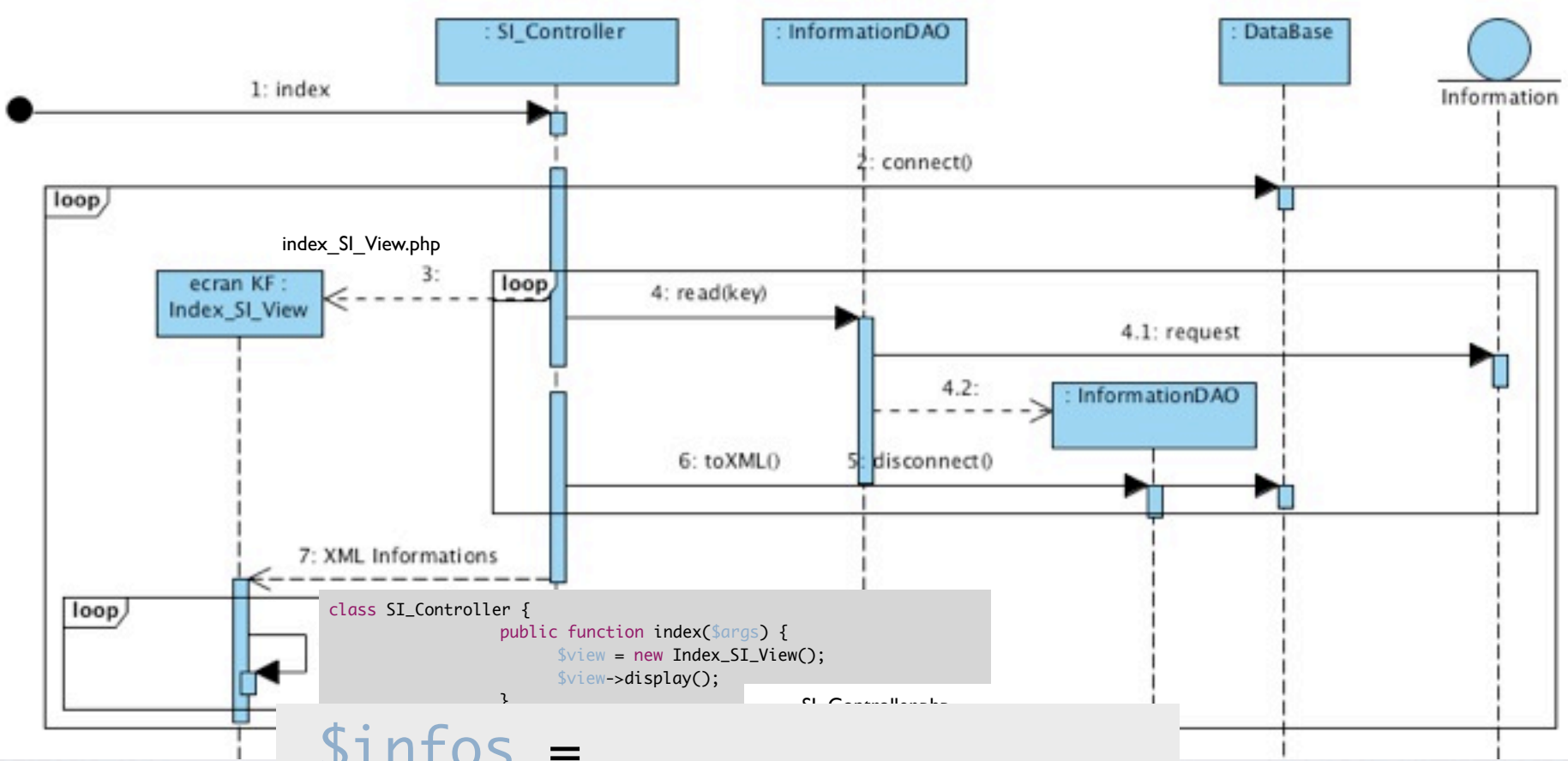
business.js

```
            $content .= $footer->getFooter();

            echo $content;
    }

    abstract protected func
}
```

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

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

    echo $res;
}
```

Afficher Informations : niveau Conception



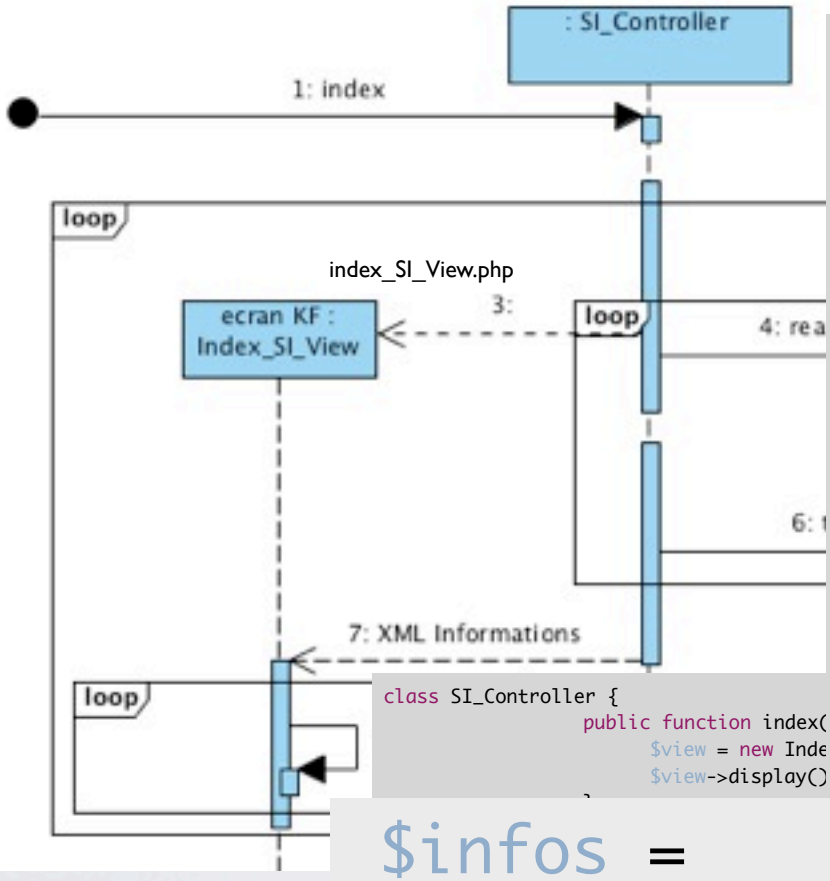

```
class SI_Controller {
                public function index($args) {
                      $view = new Index_SI_View();
                      $view->display();
                }
```

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

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

                    echo $res;
                }
```

Information
<=>InformationDAO

Afficher Informations : niveau Conception



Information.php

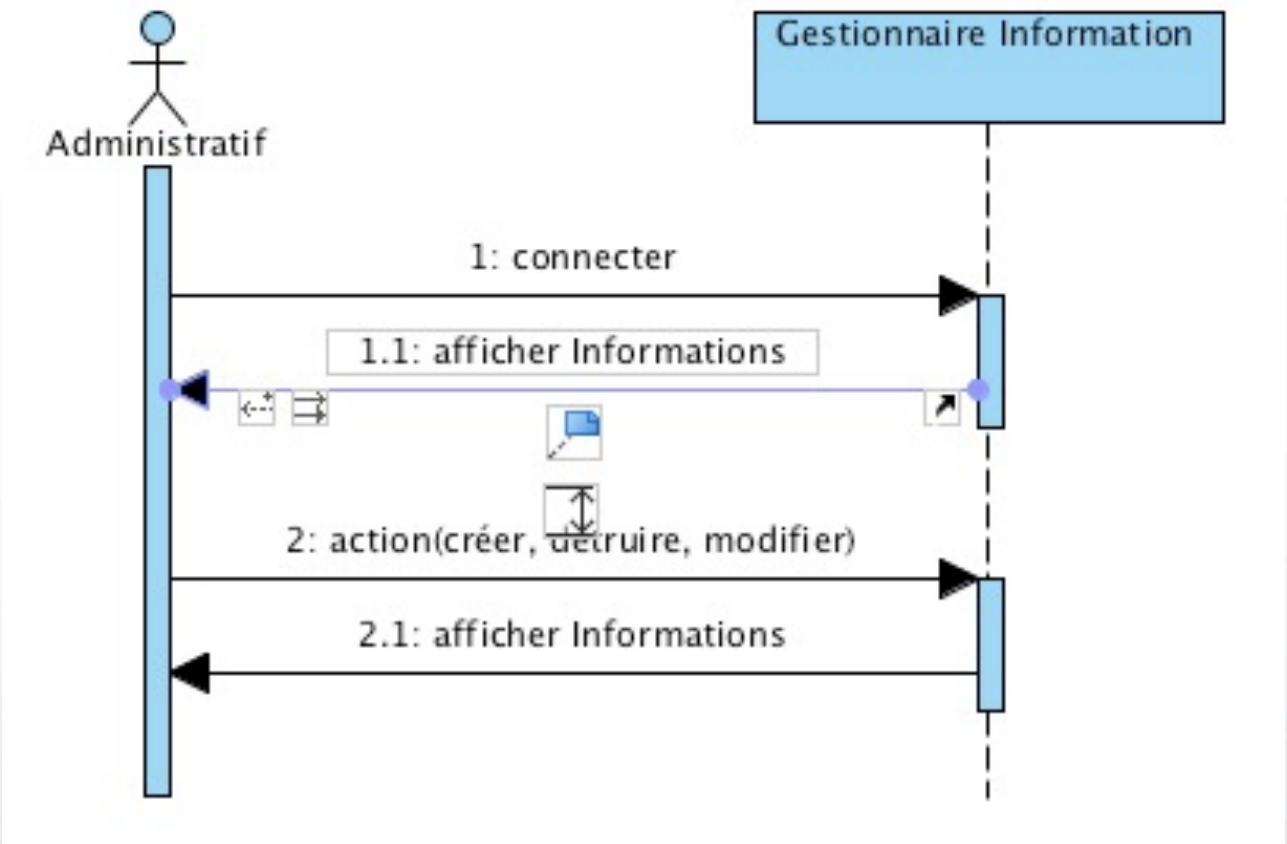
```
class Information implements iCRUD
{
  private $_id;
  private $_titre;
  private $_date;
  public static function findAll() {
   $informations = array();
   Database::connect();
    $query = "SELECT * FROM information";
    $res = mysql_query($query);
    while($line = mysql_fetch_assoc($res))
{
        $titre = $line["titre"];
       $date = $line["date"];
       $key = $line["identifiant"];
                                  nformation($titre,
```

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

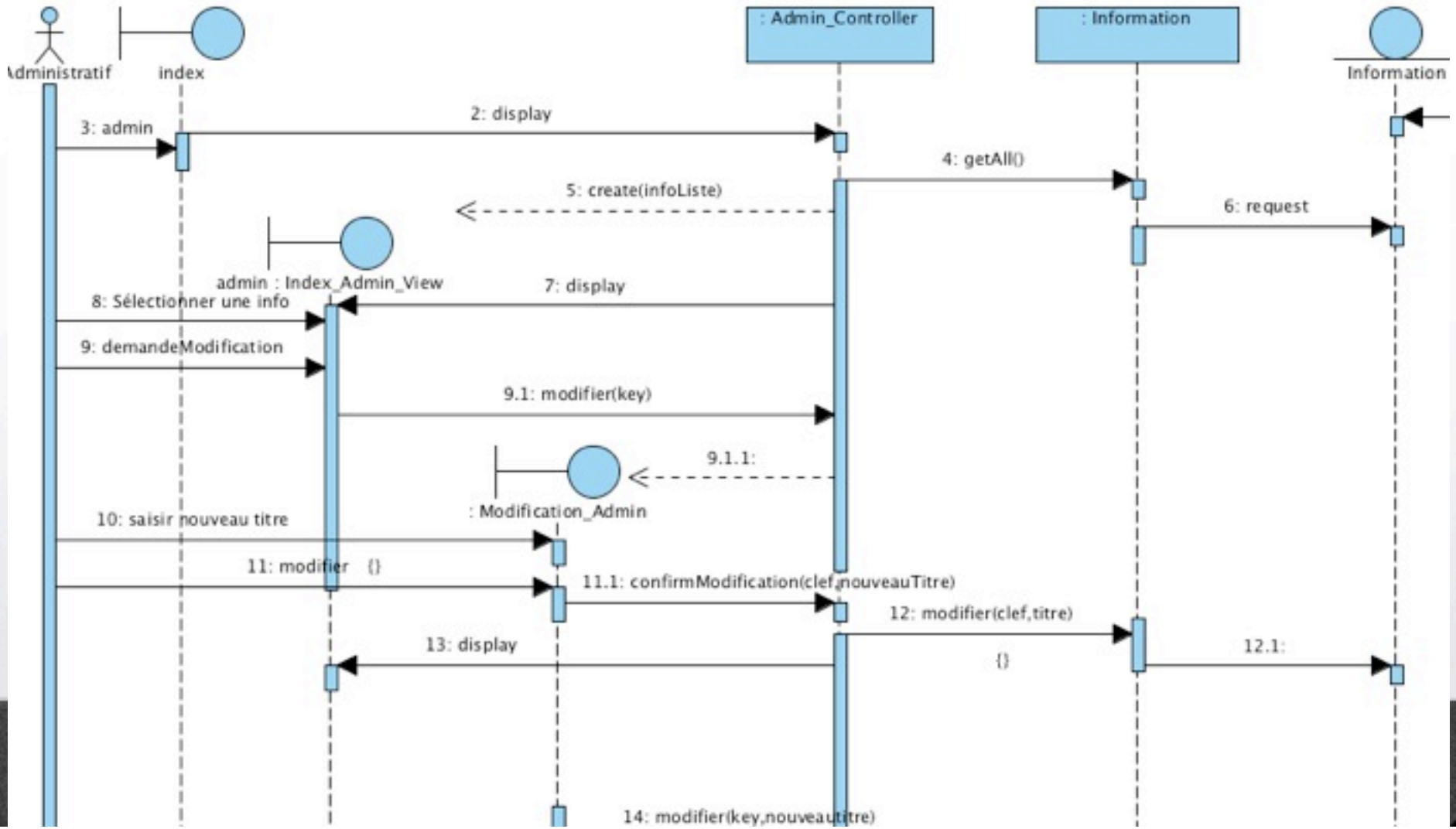
```
$infos =
Information::findAll();
```

```
                    $res = $res.$tmpInformation->toXML() ;
                }
                $res=$res.'</data>';

                    echo $res;
                }
```

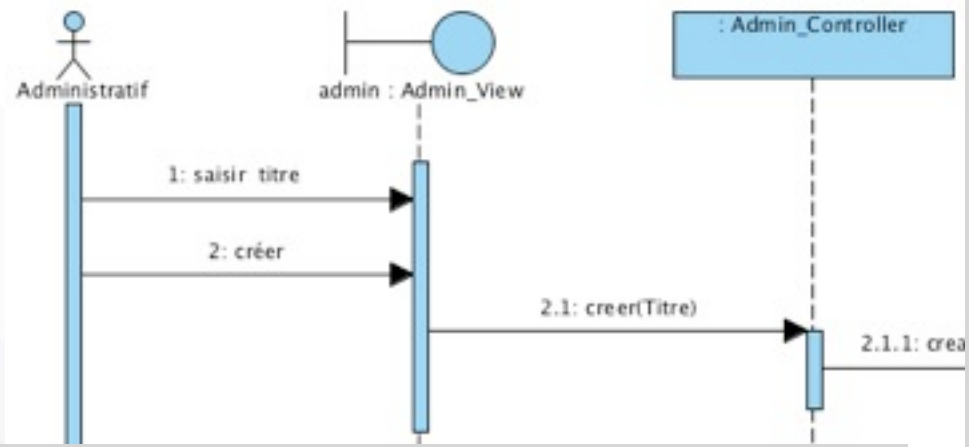
Gérer Informations : niveau Analyse (0)



modifier Information : niveau Conception



Créer Information : niveau Conception

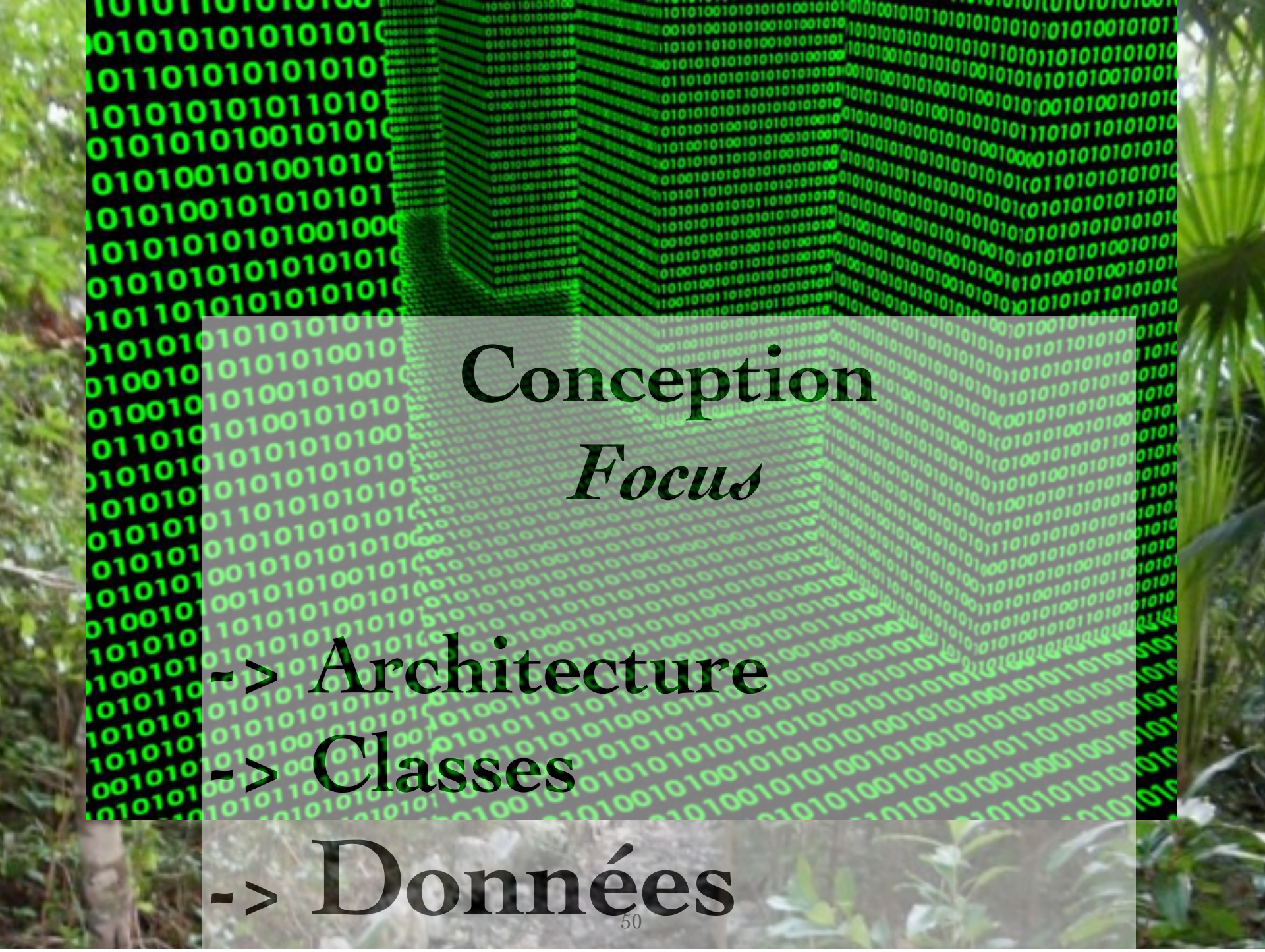


admin_View

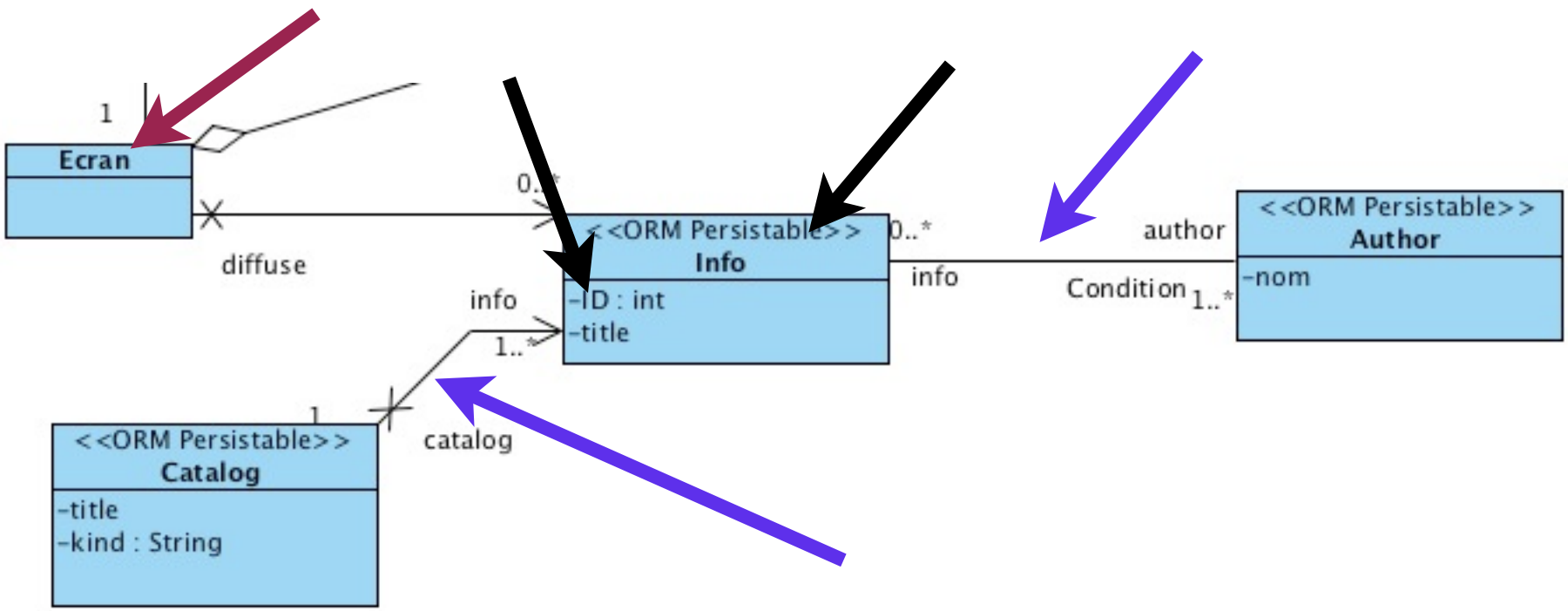
```
class Index_Admin_View extends Main_Global_View {

      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>
```

Admin_Controller.php

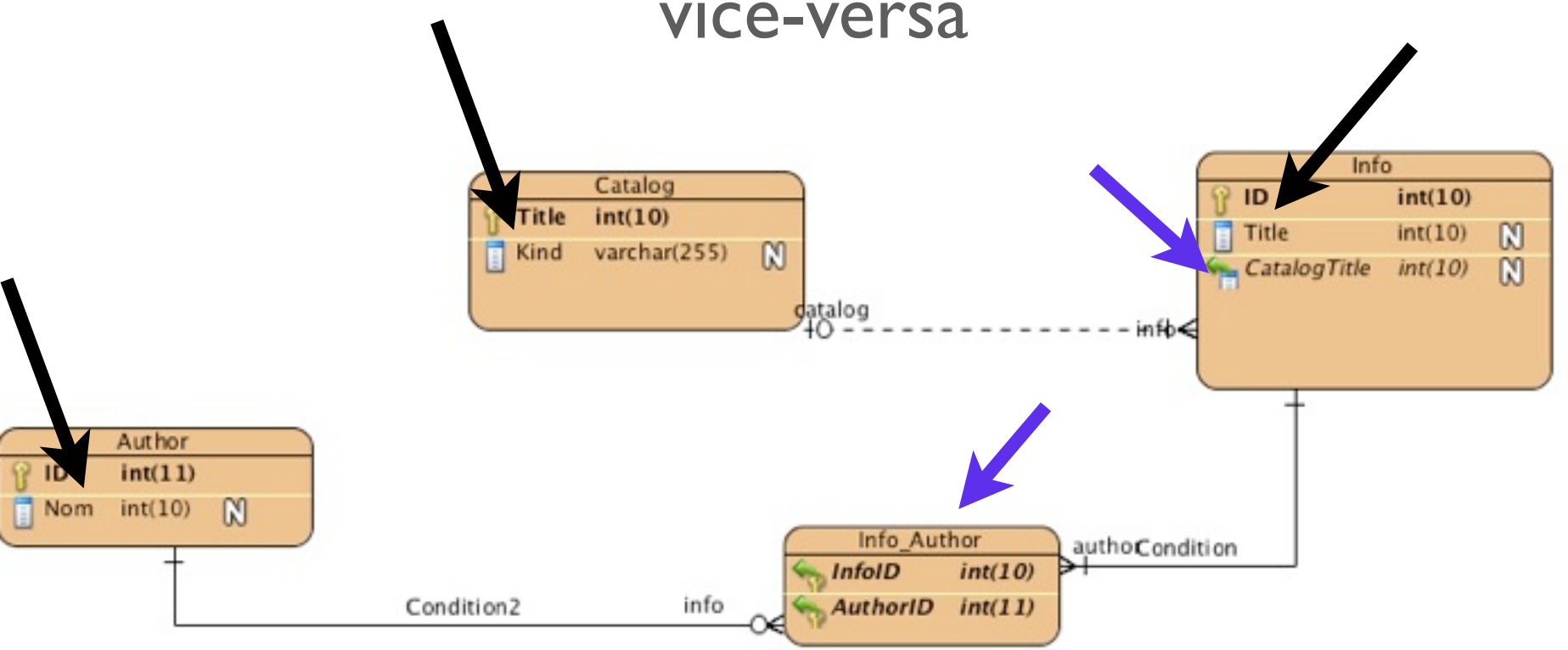
```
class Admin_Controller {
   public function index($args) {
                        $args['infosListe'] = Information::findAll();
                        $view = new Index_Admin_View($args);
                        $view->display();
                }

   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();
                }
```

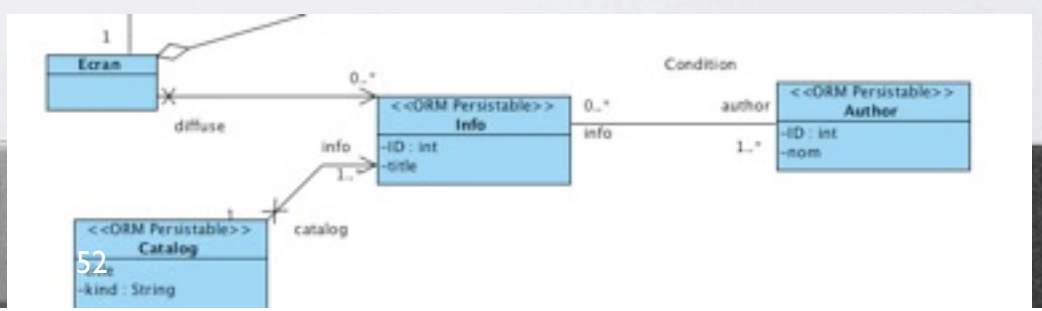

Conception
Focus
-> Architecture
-> Classes
-> Données

Des classes aux données

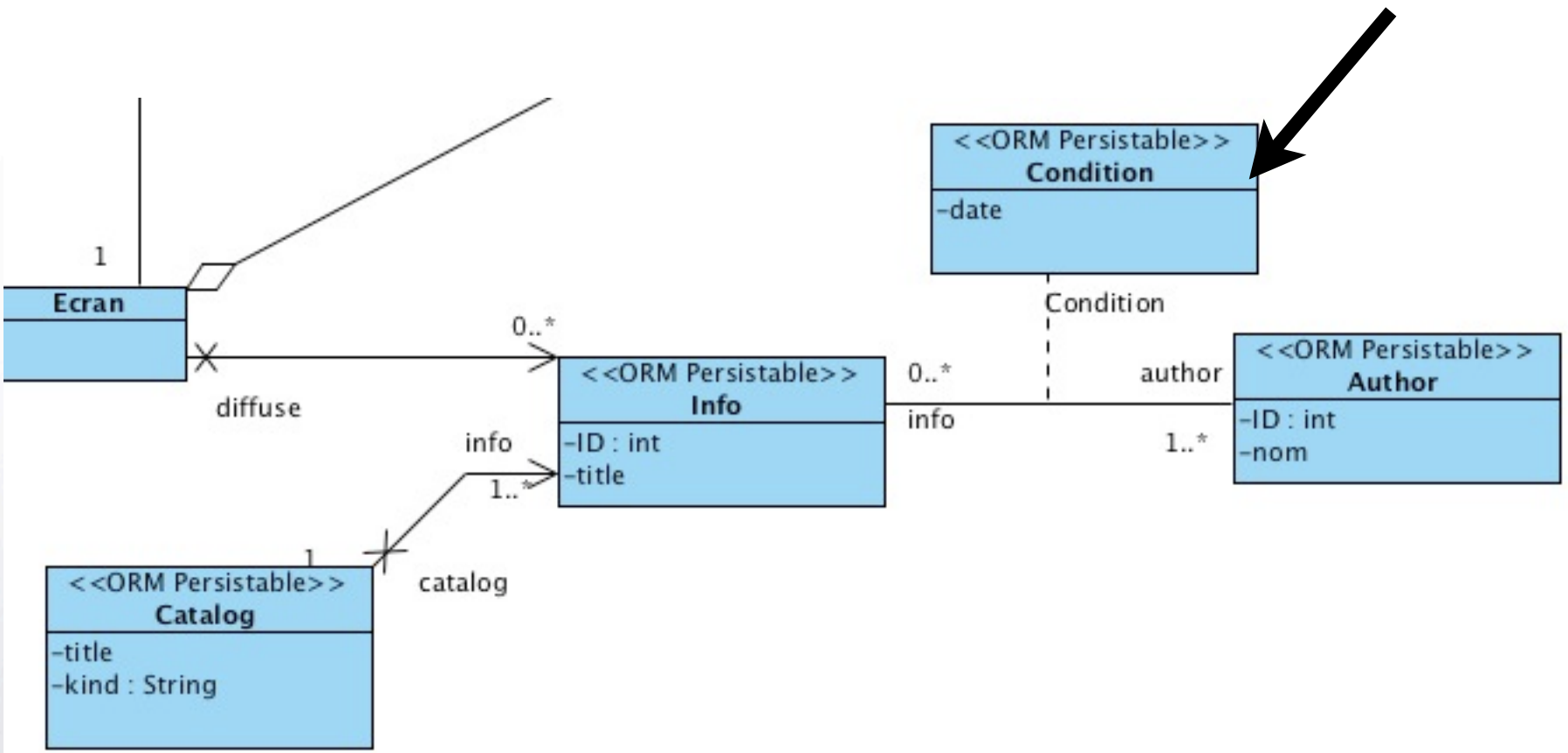


Des classes aux données (entité-relation) et vice-versa

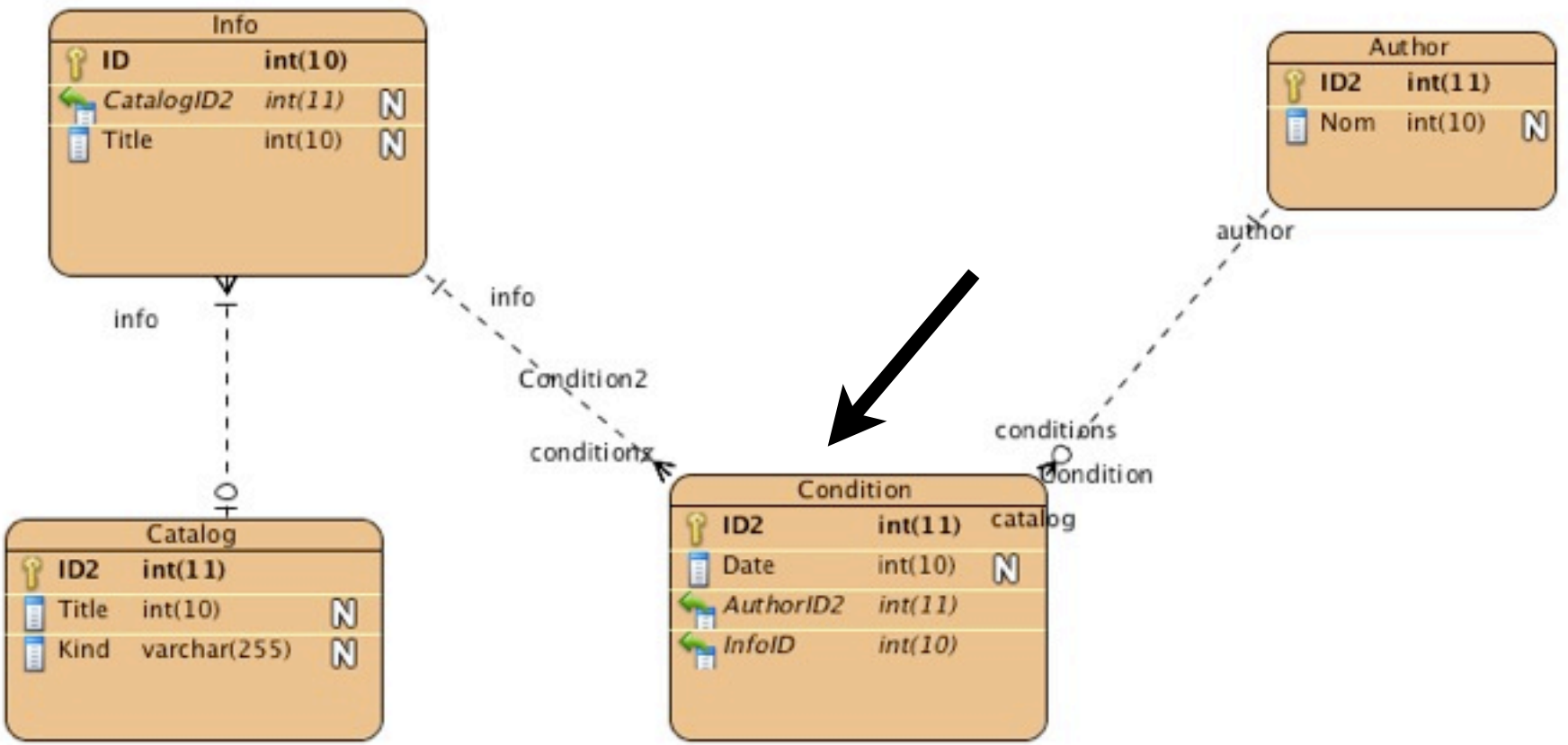




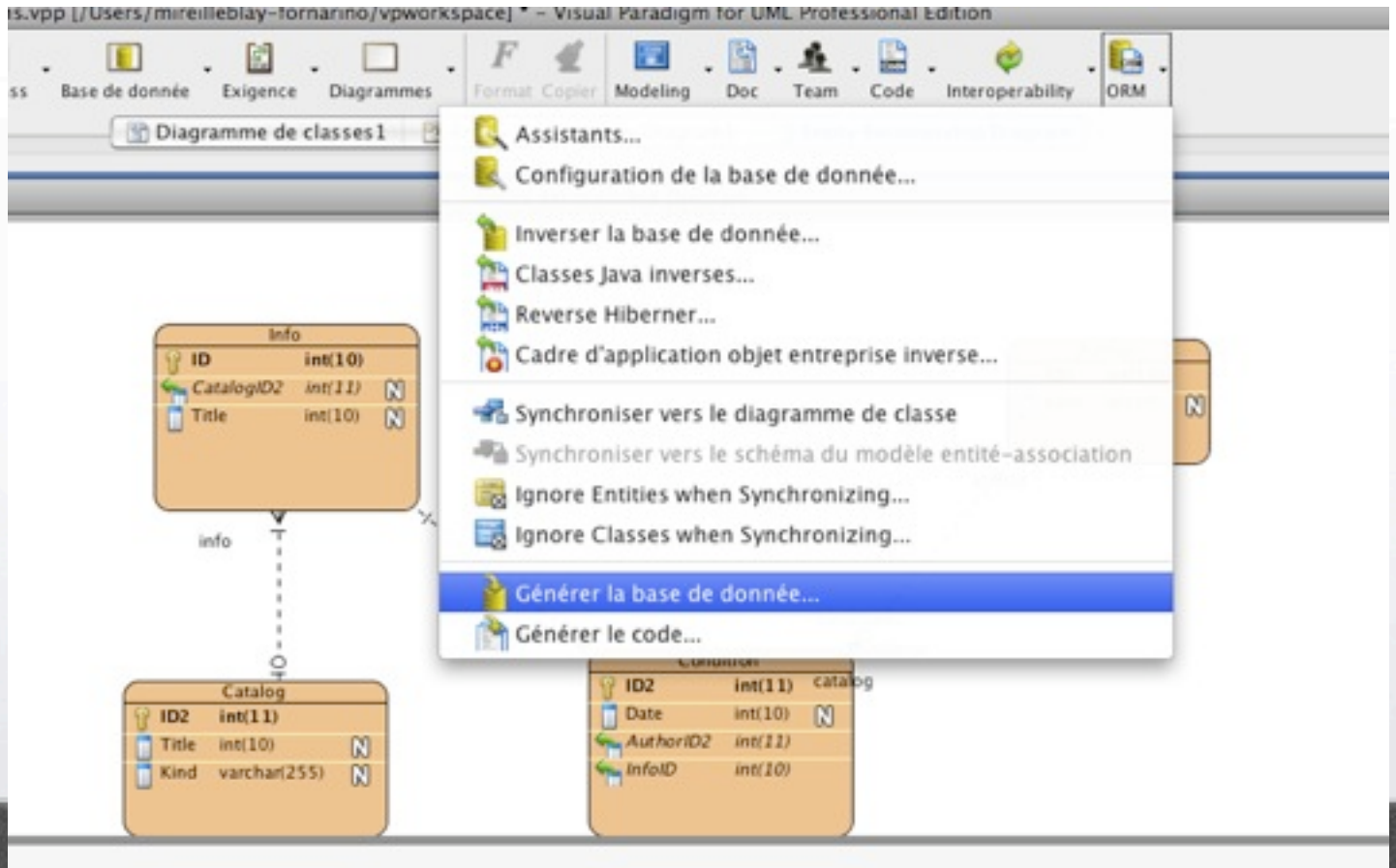
Des classes aux données



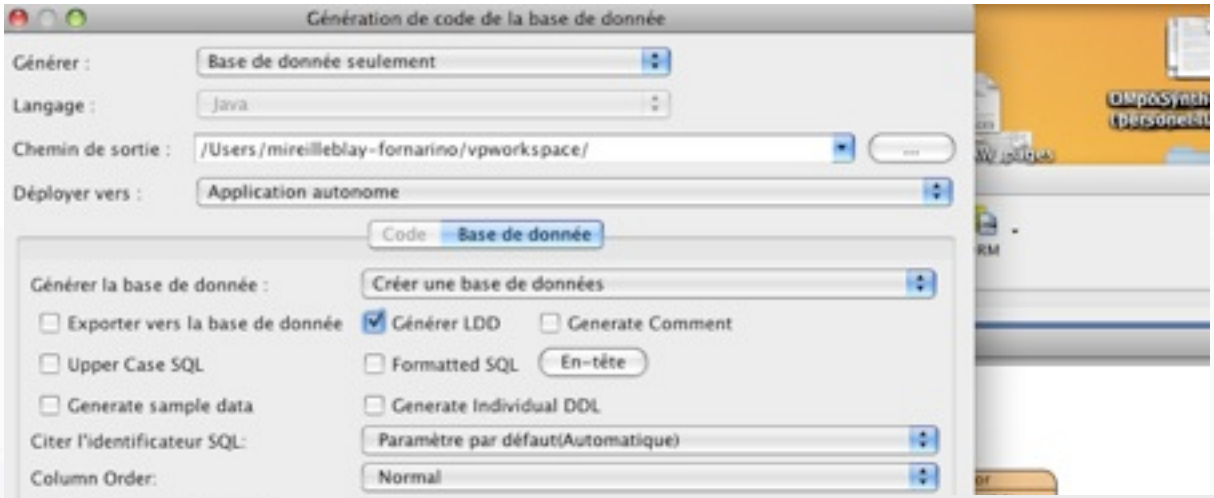
Des classes aux données



Des classes aux données



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, InfoID 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 (InfoID), add constraint Condition2 foreign key
(InfoID) references Info (ID);
alter table Info add index FKInfo468757 (CatalogID2), add constraint FKInfo468757 foreign key
(CatalogID2) references Catalog (ID2);
```

ApprocheS : Méthodologies

Le modèle - DAO

- Data Access Object
  - Le Dao a pour but de transformer les données contenues dans une bases de données en objets et inversement
- Correspondance bijective (SGBD / paradigme objet)
  - une table (appelée aussi relation) à une liste d'objets
  - une ligne d'une table (appelée aussi tuple) à un objet
  - un champs de base de données à un attribut d'objet
  - une valeur d'un champs à une valeur d'attribut d'un objet
- Si un objet est modifié, sa ligne associée dans la table l'est aussi