

IBDW : Projet Web

Application de gestion d'une chorale

**Barrand
Briquet
Diallo
Monier**

**Paume
Sabatier
Sarboni**

**Printemps
2014**

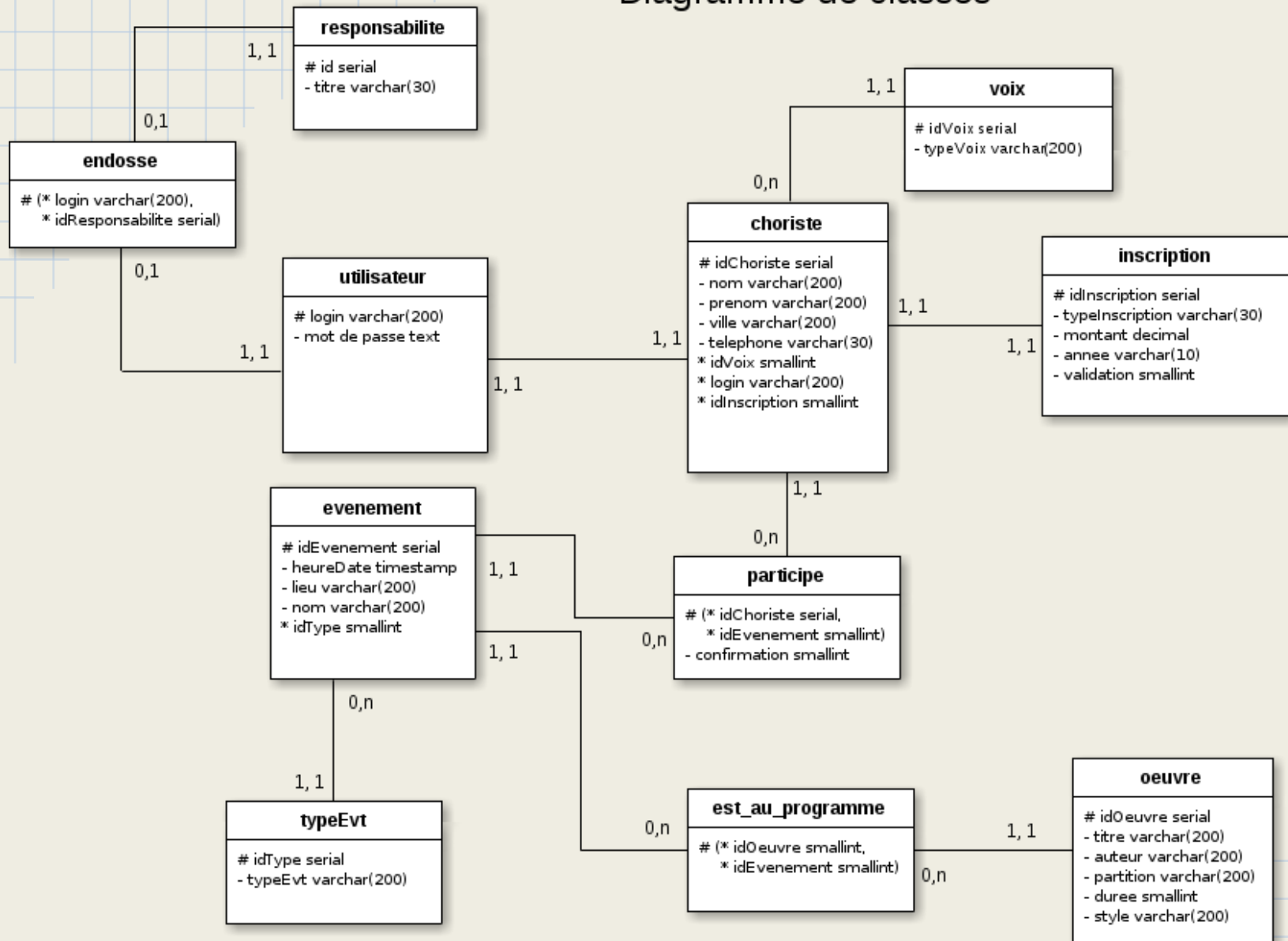
Plan de la présentation

- Modélisation des données
- Modélisation des traitements
- Implémentation
- Bilan



Modélisation des données

Diagramme de classes



Améliorations du diagramme de classes



Modélisation des traitements

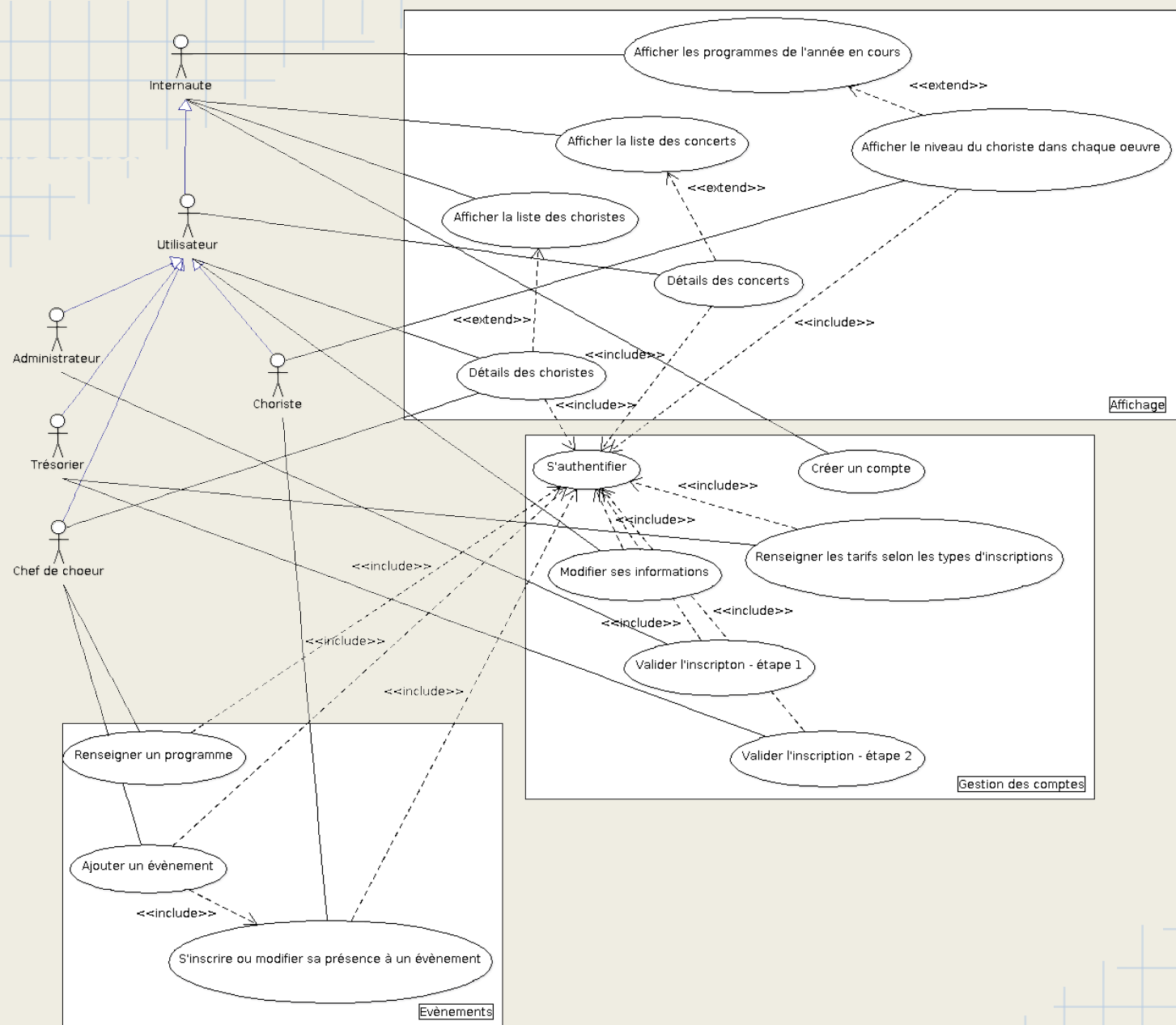


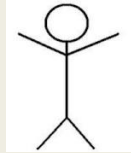
Diagramme de cas d'utilisation

Procédures d'affichage

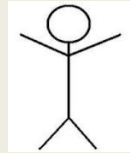
- Statistique des présences
- Affichage des concerts passés et à venir
- Affichage des choristes
- Affichage du programme

Cas d'utilisation "Affichage des concerts"

Acteurs :



Internaute



Utilisateur

Objectif : Permettre à l'internaute de visualiser les événements passés et à venir, ordonnés par ordre chronologique

Si l'internaute est un utilisateur authentifié, pour chaque concert, un voyant s'affiche :

- vert si suffisamment de choristes sont inscrits pour cet événement ;
- rouge sinon.

Maquette “Affichage des concerts passés et à venir”

Liste des concerts

Concerts à venir (30 résultats maximum)

date	heure	lieu	présence
30/03/2014	16h	<u>Evry</u>	✓
29/03/2014	16:00	<u>Evry</u>	✓
28/03/2014	16:00	<u>Evry</u>	✓

Dates précédentes

Dates suivantes

Concerts passés (30 résultats maximum)

Date	Heure	Lieu	Présence
24/03/2014	16:00	<u>Evry</u>	✓
23/03/2014	16:00	<u>Paris</u>	✓
22/03/2014	16:00	<u>Corbeilles</u>	✓
21/03/2014	16:00	<u>Evry</u>	✓
20/03/2014	16:00	<u>Evry</u>	✓

Dates précédentes

Dates suivantes

Maquette “Affichage des concerts passés et à venir”

Liste des concerts

Concerts passés(30 résultats maximum)

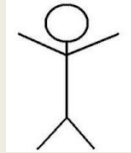
Il n'y a pas de concerts passés

Concerts à venir (30 résultats maximum)

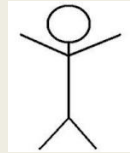
Il n'y a pas de concerts à venir

Cas d'utilisation "Affichage de la liste des choristes"

Acteurs :



Internaute



Utilisateur

Objectif : Visualiser les choristes (nom, prénom et voix) inscrits sur le site

Si l'internaute est un utilisateur authentifié, pour chaque choriste, les données suivantes sont affichées :

- nombre de répétitions auxquelles il a assisté et qu'il a manqué ;
- taux de présence aux répétitions.

La liste regroupe les choristes par type de voix et affiche le taux moyen de présence par type de voix.

Maquette “Affichage de la liste des choristes”

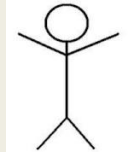
Liste des choristes							
Choristes (30 résultats au maximum)							
Voix	Nom	Prenom	Responsabilités	Nombre de répétitions assistées	Nombre de répétitions manquées	Taux de présence	Coordonnées
Alto	Dupond	Henry	Chef de choeur	5	5	100	Marseille / 0752222222
Alto	Morand	Bertrand	Webmaster	3	2	175	Paris / 0633333333
Total alto						114	
Basse	Dupont	Martin	Tresorier	2	2	100	Toulouse / 0500000000
Basse	gree	Thomas	Tresorier	1	4	25	Bordeaux / 0199999999
Total Basse						50	
Soprano	row	Quentin		10	1	100	Caen / 0333322222
Soprano	green	Sebastien		4	1	40	Campiegne / 0235632145
Total Soprano						70	
Tenor	dio	Alphonse		2	3	75	Paris / 0603050409
Tenor	thi	thake		2	3	75	Paris / 0709080203
Total Tenor						75	
Choristes précédents		Choristes suivants					

Maquette “Affichage de la liste des choristes”

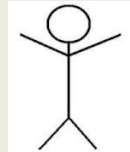
Affichage des choristes
Choristes (30 résultats au maximum)
Il n'y a pas de choristes

Cas d'utilisation “Affichage des programmes”

Acteurs :



Internaute



Utilisateur

Objectif :

Permettre à l'internaute de visualiser les œuvres étudiées durant l'année courante. A la fin de chaque groupe d'œuvre

Si l'internaute est un utilisateur authentifié, pour chaque œuvre, un voyant s'affiche :

- vert si l'œuvre a été suffisamment étudiée pour être jouée en concert
- jaune si l'œuvre est en cours d'apprentissage ;
- rouge sinon.

Maquette “Affichage des programmes”

Programme de l'année (1/2)					
Oeuvres(30 au maximum)					
Style	Titre	Auteur	Partition	Durée	Niveau de la chorale
Baroque	Seul	Mozart	http://www.exemple.fr	02:00	✓
Classique	Toto	Titi	http://www.exemple.fr	03:00	✓
Rock	Seul	Tata	http://www.exemple.fr	01:00	✓
Baroque	Seul	Mozart	http://www.exemple.fr	01:00	✓
Classique	Seul	Mozart	http://www.exemple.fr	01:00	✓
Classique	Seul	Mozart	http://www.exemple.fr	01:00	⚠
Classique	Seul	Mozart	http://www.exemple.fr	01:00	✓
Classique	Seul	Mozart	http://www.exemple.fr	01:00	✓
Classique	Seul	Mozart	http://www.exemple.fr	01:00	✓

Maquette “Affichage des programmes”

Programme de l'année (2/2)

Durée totale par style

Style	Durée totale
Baroque	1h
Classique	2h05
Rock	30 min

Maquette “Affichage de la liste des choristes”

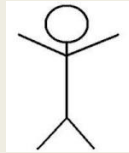
Affichage des choristes
Choristes (30 résultats au maximum)
Il n'y a pas de choristes

Procédures interactives

- Enregistrement des présences aux répétitions
- Création des évènements
- Ajout / modification d'un choriste
- Création du programme

Cas d'utilisation "S'inscrire ou modifier sa présence à un événement"

Acteurs :



Choristes

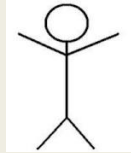
Objectif : Permettre aux choristes d'indiquer ou de modifier leurs présences à une répétition

Maquette :

La maquette est identique de celle qui est dans la procédure d'affichage, seul une liste déroulante avec un bouton « Valider » permettra de modifier ou d'indiquer la présence des choristes lors d'une répétition.

Cas d'utilisation "Ajouter un évènement"

Acteurs :



Chef de Choeur

Objectif :

Ajouter un évènement (répétition ou concert ou saison) avec un programme


Maquette “Ajouter un évènement” (1 / 2)

Création d'une saison

Nom :

Date :

Lieu :

Programme : 

Ajouter une oeuvre

Titre :

Auteur :

Partition :

Durée :

Style :

Maquette “Ajouter un évènement” (2/2)

Création d'un concert ou d'une répétition

Type d'évènement :

Nom :

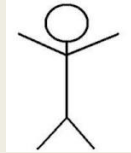
Date :

Lieu :

Programme :

Cas d'utilisation "Ajout/Modification d'un choriste"

Acteurs :



Internaute



Utilisateur

Objectif de l'ajout :

Créer un compte pour pouvoir accéder à des informations personnalisées

Objectif de la modification :

Permettre à l'utilisateur de modifier ses informations tels que sa voix, sa ville, son numéro de téléphone et son mot de passe.

Maquettes “Ajout / Modification d’un choriste”

Inscription

Nom :

Prénom :

Login :

Mot de passe :

Confirmation de mot de passe :

Téléphone :

Ville :

Voix :

Valider

Mon compte

Nom : Dupont

Prénom : Henry

Login : d.henry

Ancien mot de passe :

Nouveau mot de passe :

Téléphone :

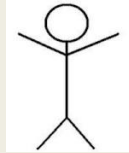
Ville :

Voix :

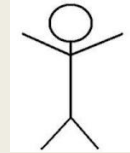
Valider

Cas d'utilisation "Validation de l'inscription"

Acteurs :



Administrateur



Trésorier

Objectif :

- Permettre à l'administrateur de vérifier si l'inscription est bien celle d'un choriste.
- Permettre au trésorier d'indiquer que l'inscription est bien réglée par le choriste.

Maquette “Validation de l’inscription”

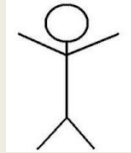
Validation des nouveaux inscrits

Nom	Prénom	Login	Validation de l'administrateur	Validation du trésorier
Dupont	Henry	d.henry	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Morand	Bertrand	m_bertrand	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Dupont	Martin	dmartin		

Valider

Cas d'utilisation "Ajouter un programme"

Acteurs :



Chef de Choeur

Objectif : Définir en début d'année l'ensemble des oeuvres qui vont être étudiées dans l'année

Maquette “Ajouter un programme”

Création d'un programme

Titre :	<input type="text"/>
Compositeur :	<input type="text"/>
Année courante :	<input type="text"/>
Style :	<input type="text"/>
Type d'évènement :	<input type="text" value="concert"/>



Implémentation

Modèle, Vue, Contrôleur

- La structure **MVC** est très utilisée dans les projets de génie logiciel comportant une interface
- Décomposer le développement en trois items indépendants :
 - Les **Modèles** gèrent les interactions avec les source de données et les envoient aux Contrôleurs
 - Les **Contrôleurs** effectuent tous les traitements nécessaires sur ces données, puis les envoient aux Vues
 - Les **Vues** formatent les données pour les afficher d'une façon compréhensible pour l'utilisateur
- Répartition des tâches très aisée
- Forte structuration, haute maintenabilité
- S'inscrit dans notre démarche de **qualité**

Cycle de vie d'une requête

Par exemple, `GET /choristes`

1. La requête est reçue par le serveur web et redirigée vers le fichier `index.php`.
2. La **route** (URL) demandée est comparée à toutes les routes définies dans le fichier. Lorsqu'elle en **match** une, la fonction associée (le **contrôleur**) est exécutée.
3. La fonction récupère les données via un **modèle** et effectue un traitement (ou pas) dessus. Elle renvoie ces données à une **vue** via des variables PHP.
4. La **vue** s'occupe d'intégrer les données brutes issues du contrôleur dans la structure HTML des pages, par exemple entre des balises `` pour une liste.

Pourquoi utiliser un framework côté serveur ?

- Standardiser les développements en adoptant une structure et des conventions uniques
- Éviter les choix de frameworks complets mais contraignants : utiliser un micro-framework
- Faciliter au maximum l'adoption du framework par les membres de l'équipe qui ne connaissent que le PHP "from scratch"
- Nous avons choisi Flight PHP
 - souple, y compris dans le moteur de templates
 - facile à apprendre
 - extensible
 - compatible PHP 5.2

Flight

Pourquoi utiliser un framework côté client ?

- **Bootstrap** : un framework CSS et Javascript
 - Développé par Twitter
 - Inclut jQuery
- Beaucoup d'éléments intégrés :
 - Barre de menus
 - Système puissant de grilles pour positionner les `<div>`
 - Menus déroulants
 - Boutons
 - Tables
 - Étiquettes de couleur
 - Validation dynamique des formulaires
 - Beaucoup d'autres...
- Un style graphique propre dès le début sans avoir écrit aucune ligne de CSS !



Présentation d'un traitement

Liste des choristes (1/5)

L'utilisateur arrive sur une page du site via un lien, par exemple :

- <http://mychorus.com/choristes>

Le fichier `.htaccess` procède alors à une réécriture d'Url basée sur les expressions régulières. Ainsi, le lien ci-dessus pointe en réalité vers `index.php`.

Fichier `.htaccess`

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php [QSA,L]
```

Le fichier `index.php` endosse alors le rôle de contrôleur principal de l'application. Il initialise l'application (chargement et configuration des fichiers) et surtout il définit les "routes" (il associe des URL à des fonctions).

```
Flight::route('/choristes', function() { Choristes::get(); });
```

Présentation d'un traitement

Liste des choristes (2/5)

Ce morceau de code permet d'indiquer à l'application qu'elle doit faire appel à la fonction `get()` de la classe `Choristes` lorsqu'un utilisateur demande l'accès à la page `/choristes`.

```
<?php
require_once '../model/Choristes.php'; // Modèle (accès aux données)

class Choristes {
    // GET /choristes
    function get() {
        [...]
    }
    [...]
}
```

Une fois dans la fonction, on a besoin de récupérer les données à afficher, il faut alors interroger la base de données.

Présentation d'un traitement

Liste des choristes (3/5)

On appelle donc le modèle associé à la classe Choristes, il contient les fonctions d'interactions avec les données liés aux choristes (`select` / `update` / `insert`).

Dans notre cas, la fonction `get()` fait appel à `getChoristes()` du modèle :

```
<?php
require_once 'Generic.php'; // Fonctions liés à l'exécution de requêtes

class Queries {
    // Liste des choristes (nom, responsabilité, participations...)
    function getChoristes() {
        $fetchall = True;
        $sql = 'SELECT nom [...] FROM Choriste [...]';
        list($success, $result) = Query::execute($sql, $fetchall);
        return array($success, $result);
    }
    [...]
}
```

Présentation d'un traitement

Liste des choristes (4/5)

On a récupéré nos données , on effectue des traitements si nécessaire.
Ensuite, il ne nous reste plus qu'à les envoyer à la vue pour qu'elle les affiche.

```
// On modifie une propriété de la barre de navigation (la page active)
Flight::render('navbar.php', array('activePage' => 'choristes'),'navbar');

[...]

// Enfin on transmet les données à la vue
if($data['success'])
    Flight::render('ChoristesLayout.php', array('data' => $data));
else
    Flight::render('ErrorLayout.php', array('data' => $data));
```

La vue récupère alors les données et les affiche avec des instructions basiques de PHP (conditions, boucles).

Présentation d'un traitement

Liste des choristes (5/5)

```
<?php
echo($header);
echo($navbar);

echo '<h1>Liste des choristes</h1>';

// Liste des choristes
echo '<table class="table table-striped">';
[...]
if(count($data['content']) > 0) {

    foreach($data['content'] as $row) {
        echo '<tr>';
        echo '<td>' . $row['prenom'] . ' ' . $row['nom'] . '</td>';
        [...]
        echo '</tr>';
    }
}

echo '</table>';
echo $footer;
```

Tests & Qualité

- Jeux de tests
 - Tests sur des cas particuliers ou éventuels bugs d'affichage
 - Concerne l'affichage, les fonctionnalités du site
- Quelques exemples
 - Affichage d'un message lorsqu'il n'y a pas de données dans la base. Par exemple, "Aucune donnée à afficher".
 - L'accès au site est contrôlé : seuls les comptes utilisateurs ayant été validés au moins par le webmaster peuvent se connecter.
 - Les champs du formulaire d'inscription d'un choriste sont vérifiés avant l'envoi.



Bilan

Difficultés rencontrées

- Les outils
 - L'utilisation d'un framework ou non ?
 - L'utilisation du gestionnaire de versions Git, pas évident pour tous
 - Perte de temps dû à des problèmes de logiciel sur la création des maquettes
 - Problèmes de mise en place de l'environnement de travail.
- Le développement
 - L'implémentation de la base de données a été revue
- Gestion de projets
 - Difficultés à découper le travail au sein du groupe
 - Contrainte de temps à prendre en compte

Apports pédagogiques

- Utilisation du gestionnaire de versions **Git**
 - Service web d'hébergement de projets **GitHub**
 - Gestionnaire de versions
 - Logiciel Open Source
 - Puissant mais complexe
- Configuration d'un serveur web
 - Configuration d'Apache et de PHP
 - Installation de PostgreSQL
 - Installation du framework Flight PHP
- Mise en place d'une architecture MVC



Apports pédagogiques

Structure du projet

Sur [notre dépôt Git](#),

- les fichiers du framework [Flight](#) sont inclus dans le répertoire `lib`. Il est inutile d'essayer comprendre le contenu de ces fichiers;
- les contrôleurs, qui font les requêtes dans la base de données et les traitements sur ces données, sont dans `controllers`;
- les vues (templates HTML qui contiennent les données envoyées par les contrôleurs) sont dans `views`;
- les feuilles de style CSS sont dans `css` et les scripts JavaScript dans `js`;
- la documentation est dans `doc`.

Quel est le cycle de vie d'une requête ?

L'étude du cycle de vie d'une requête va nous permettre de comprendre à peu près l'ensemble des fonctionnalités de Flight que nous utilisons.

Réécriture d'URL

On prend l'exemple du cas où l'application est installée à la racine du serveur web (typiquement `/var/www/` sur Ubuntu ou Debian comme on a à l'école).

Sans entrer dans les détails, il faut savoir que toute requête du genre `http://localhost/choristes/new` sera redirigée par le serveur HTTP Apache vers `http://localhost/index.php` grâce à une fonctionnalité appelée "URL rewriting" (ou réécriture d'URL en français).

En opérant cette redirection, Apache fournit aussi à `index.php` l'URL qui était demandée initialement (`/choristes/new`) dans une variable. Pour comprendre comment cette variable est utilisée, il faut étudier `index.php`.

Merci de votre attention

**Nous sommes disponibles
pour vos questions**