|  |
| --- |
| **Webframework Module Utilisateur** |
| Document Technique |
|  |
| Version 2.0 |
|  |
| **AUGUEY Thomas** |
| **23/12/2012** |
|  |

Contenu

[Introduction 3](#_Toc345001348)

[Termes 4](#_Toc345001349)

[Environnement 5](#_Toc345001350)

[Webframework 5](#_Toc345001351)

[Base de données 5](#_Toc345001352)

[Environnement de développement intégré (IDE) 5](#_Toc345001353)

[Suivit de version 5](#_Toc345001354)

[Modèle de données 5](#_Toc345001355)

[Arborescence 6](#_Toc345001356)

[Modèle de données 6](#_Toc345001357)

[Entités 6](#_Toc345001358)

[Domaines 8](#_Toc345001359)

[Formats 9](#_Toc345001360)

[Relations 10](#_Toc345001361)

[Contraintes 10](#_Toc345001362)

[Operations système 11](#_Toc345001363)

[Acteurs 11](#_Toc345001364)

[Visiteur 11](#_Toc345001365)

[Utilisateur 11](#_Toc345001366)

[Administrateur 11](#_Toc345001367)

[Système 11](#_Toc345001368)

[Résultats d’opération 11](#_Toc345001369)

[Détail des codes 11](#_Toc345001370)

[Cas d’utilisation 12](#_Toc345001371)

[userCreateAccount 13](#_Toc345001372)

[userConnectToSession 14](#_Toc345001373)

[userConnectToClient 15](#_Toc345001374)

[Check Connexion 16](#_Toc345001375)

[Déconnexion d’un utilisateur 17](#_Toc345001376)

[Déconnexion de tous les utilisateurs 18](#_Toc345001377)

[Activer un compte utilisateur 19](#_Toc345001378)

[Contrats d’opération 19](#_Toc345001379)

[Create 19](#_Toc345001380)

[Connect 20](#_Toc345001381)

[Développement 21](#_Toc345001382)

[Point d’entrée de l’application 21](#_Toc345001383)

[Modèle MVC 21](#_Toc345001384)

[API Références 21](#_Toc345001385)

[Modèle objet et Classes Managers 21](#_Toc345001386)

# Introduction

Le projet Module Utilisateur permet de gérer dynamiquement des comptes utilisateur pour votre site web.

Fonctionnalités :

* Gestion des comptes utilisateurs
* Gestion des connexions par sessions et adresses IP
* Gestion des informations d’indentification (adresses, identité)

# Termes

Webframework

Le projet principal dont se module est une extesion.

# Environnement

## Webframework

Ce module est développé pour être utilisé avec le modèle minimal MVC du projet Webframework.

Votre application web doit au minimum implémenter l’interface IApplication et définir la variable globale $app pour fonctionner normalement (voir le modèle **/wfw/minimal** du projet **Webframework**).

## Base de données

Le **système de base de données** utilisé est de type **PostgreSQL 8.3** en adéquation avec le projet **Webframework**.

## Environnement de développement intégré (IDE)

Le module est développé sous l’environnement **NetBeans 7.2.1** en langage **PHP/Javascript/XHTML.**

## Suivit de version

L’Utilitaire **GIT** est utilisé pour le suivit et l’évolution du projet.

**Utile** : Le script archive.sh permet de créer une archive compressée du projet ne contenant que les fichiers importants. L’archive est stockée dans le dossier « **../archives »** et nommée automatiquement avec la version du projet en cours.

## Modèle de données

La conception de la **base de données** et du **modèle orienté objet** à été réalisé sur le principe de **l’UML** avec le logiciel **PowerAMC 15.1**.

**Important** :

Pour pouvoir générer correctement le code PHP du modèle orienté objet, veuillez copier le fichier « documents/php.xol » dans le répertoire d’installation de votre programme **PowerAMC** (ex : C:\Program Files (x86)\Sybase\PowerAMC 15\Fichiers de ressources\Langages objet\php.xol).

Ce fichier permet de générer le code **PHP** et les classes **managers** pour l’interaction avec la base de données.

# Arborescence

Voici une vue d’ensemble des dossiers du projet.

* DOCUMENTS : Documents divers relatif à ce projet
* MCD : Fichiers du Modèle conceptuel de données (PowerAMC)
* nbproject : Fichiers du projet **NetBeans**
* SQL : Fichiers SQL de création de la base de données
* wfw-X.X : Librairies et sources du projet à lier avec votre application

# Modèle de données

Le modèle des données persistantes est conçu sur le modèle Merise.

Le document « **mcd/user.mcd** » propose une version schématisé du modèle conceptuel de données, ce modèle est utilisé pour générer le script d’installation SQL. [Vous pouvez ouvrir ce fichier avec le logiciel **PowerAMC** de **Sybase**].

## Entités

Liste les tables ajoutées à la base de données.

Le tableau des entités est définit ainsi :

* Le **type** de donnée est écrit de façon générique (VARCHAR, CHAR, INTEGER, REAL, …)
* La **contrainte** est appliquée sur les champs par la base de données (Domaines, Clés, Check, …)
* Le **format**, définit plus bas, définit la syntaxe pour les chaines de caractères

USER\_ACCOUNT

Stock les comptes utilisateurs actifs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribut | Description | Type | Contrainte | Format |
| USER\_ID | Identifiant | VARCHAR(64) | PRIMARY KEY | Identifier |
| USER\_PWD | Mot-de-passe | VARCHAR(64) | NOT NULL | Password |
| CLIENT\_ID | Identifiant du dossier **Client** | VARCHAR(64) | NOT NULL | Name |
| USER\_MAIL | Adresse électronique | VARCHAR(80) | NOT NULL | UNIQUE | Mail |

USER\_REGISTER

Stock les comptes utilisateurs en attente d’activation.

*Note : Le doublon de nom d’utilisateur (****USER\_ID****) est autorisé, cela pour éviter la saturation en cas d’accumulation d’inscription en attente. Au contraire seulement une inscription par adresse mail (****USER\_MAIL****) est toléré.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribut | Description | Type | Contrainte | Format |
| USER\_REGISTER\_ID | Entier | INT | PRIMARY KEY | Integer |
| USER\_ID | Identifiant | VARCHAR(64) | NOT NULL | Identifier |
| USER\_TOKEN | Mot-de-passe | VARCHAR(64) | NOT NULL | Password |
| USER\_MAIL | Adresse électronique | VARCHAR(80) | NOT NULL | UNIQUE | Mail |

USER\_SESSION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribut | Description | Type | Contrainte | Format |
| SESSION\_ID | Identifiant | CHAR(64) | PRIMARY KEY | Name |
| LOCAL\_PATH | Emplacement de la ressource dans le système de fichier (PRIVE) | CHAR(260) | - | UnixPath |

USER\_CONNECTION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribut | Description | Type | Contrainte | Format |
| CLIENT\_IP | Adresse IP du client | VARCHAR(15) | PRIMARY KEY | IPv4 |
| LAST\_ACCESS | Date de dernier accès à la session | DATE | NOT NULL |  |
| LIFE\_TIME | Temps de vie avant déconnexion | INTEGER | NOT NULL |  |
| LINK\_PATH | Emplacement du lien symbolique dans le système de fichier (PUBLIC) | CHAR(260) | - |  |

USER\_IDENTITY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribut | Description | Type | Contrainte | Format |
| IDENTITY\_ID | Identifiant | INTEGER | PRIMARY KEY |  |
| FIRST\_NAME | Prénom | CHAR(80) | NOT NULL |  |
| LAST\_NAME | Nom | CHAR(80) | NOT NULL |  |
| BIRTH\_DAY | Date de naissance | DATE | NOT NULL |  |
| SEX | Sexe | DOMAIN(SEX) | NOT NULL |  |

USER\_ADDRESS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribut | Description | Type | Contrainte | Format |
| ADDRESS\_ID | Identifiant | INTERGER | PRIMARY KEY |  |
| STREET\_NAME | Nom de rue | VARCHAR(128) | NOT NULL |  |
| STREET\_PREFIX | Préfix de rue | DOMAIN (STREET\_PREFIX) |  |  |
| STREET\_NUMBER | Numéro de rue | NUMBER(4,0) | NOT NULL |  |
| ZIP\_CODE | Code postal | VARCHAR (10) | NOT NULL |  |
| CITY\_NAME | Ville | VARCHAR (80) | NOT NULL |  |
| COUNTRY\_NAME | Pays | VARCHAR (80) | NOT NULL |  |
| BUILD\_NUMBER | Numéro de bâtiment | NUMBER(4,0) |  |  |
| APT\_NUMBER | Numéro appartement | NUMBER(4,0) |  |  |

## Domaines

STREET\_PREFIX

Préfix d’une rue.

|  |  |
| --- | --- |
| Valeur : CHAR(3) | Description |
| BIS | Bis |
| TER | Ter |

SEX

Sexe de la personne.

|  |  |
| --- | --- |
| Valeur : CHAR(1) | Description |
| M | Homme |
| F | Femme |

## Formats

Tous les formats utilisés hérites du projet **Webframework**, reportez-vous au document technique lié au projet pour plus d’informations.

## Relations

1. USER : CONNECTION
   1. Un **utilisateur** peut ouvrir plusieurs **connexions**
   2. Une **connexion** peut être ouverte par un **utilisateur**
2. CONNECTION : SESSION
   1. Une **session** peut être attaché à plusieurs **connexions**
   2. Une **connexion** peut être attachée à une seule **session**
3. USER : IDENTITY
   1. Un **utilisateur** peut renseigner une **identité**
   2. Une **identité** peut être renseignée par plusieurs **utilisateurs**
4. IDENTITY: ADDRESS
   1. Une **identité** peut posséder une **adress**e
   2. Une **adresse** peut être posséder par plusieurs **identités**

## Contraintes

1. Une **connexion** expire après un certain temps d’inactivité, dans ce cas, la connexion est automatiquement supprimée.
2. Pour être utilisé, un **compte utilisateur** doit d’abord être validé avec un code confidentiel envoyé par mail lors de sa création ou par demande du client.
3. Un utilisateur ne peut pas se connecter sous plusieurs adresses IP à la fois, cependant plusieurs utilisateurs peuvent utiliser la même adresse IP. (*Unicité de l’attribut* ***USER\_ID*** *dans la table* ***CONNECTION***).
4. Un lien symbolique est propre à une seule connexion (Unicité de l’attribut **LINK\_PATH** dans la table **CONNECTION).**
   1. Si un lien était partagé entre plusieurs connexions, une connexion fermée (supprimée) pourrait continuer à utiliser ce lien.

# Operations système

Les opérations systèmes sont conçues sur le modèle UML.

Le document « **mcd/operations.moo** » propose une version schématisé des cas d’opération. [Vous pouvez ouvrir ce fichier avec le logiciel **PowerAMC** de **Sybase**].

## Acteurs

### Visiteur

Utilisateur non enregistré.

### Utilisateur

Utilisateur enregistré, ayant créé un compte utilisateur.

### Administrateur

Utilisateur le l’interface de gestion.

### Système

Définition du système avec ses différentes couches de dépendances.

**Système Langage**

1. Application HTML/Javascript
2. Interface Serveur PHP
   1. Système embarqué (système de fichier) Console UNIX
3. Interface de Base de données PG/SQL PL/pgSQL
   1. Base de données SQL SQL

## Résultats d’opération

Les résultats d’opération sont basés sur le modèle du projet **WebframeWork**. Pour plus de détails à ce sujet reportez-vous au document technique concerné.

Noté que ses codes d’erreurs viennent s’ajouter aux codes déjà existant dans le projet **Webframework**. Certains d’entre eux peuvent être utilisés par les opérations présentes ci-dessous.

### Détail des codes

**Contexte Code Description**

ERR\_OK USER\_CREATED L’Utilisateur est créé

ERR\_OK USER\_DELETED L’Utilisateur est supprimé

ERR\_OK USER\_REGISTRED L’Utilisateur est inscrit

ERR\_OK USER\_DISCONNECTED L’Utilisateur est déconnecté

ERR\_FAILED USER\_EXISTS L’Utilisateur existe déjà

ERR\_FAILED USER\_INVALID\_PWD Le mot-de-passe est invalide

ERR\_FAILED USER\_MAIL\_EXISTS L’adresse mail existe déjà

ERR\_FAILED USER\_NAME\_EXISTS Le nom d’utilisateur existe déjà

ERR\_FAILED USER\_MAIL\_REGISTRED Une inscription en attente existe déjà pour cette adresse mail

ERR\_FAILED USER\_REGISTRATIION\_NOT\_EXISTS L’Inscription n’existe pas

## Cas d’utilisation

Les cas d’utilisations ou use cases, liste les procédures et leurs interactions avec le système.

Une procédure doit toujours être terminée pour être validée. Dans le cas contraire et pour maintenir un système cohérant, aucunes des actions précédentes ne doit être conservé.

Implementation

Liste et statut de l’implémentation des cas d’utilisation.

1. **user\_create\_account OK**
2. **user\_register\_account OK**
3. **user\_activate\_account** **OK**
4. **user\_delete KO**
5. **user\_connect\_to\_session** **KO**
6. **user\_connect\_to\_client KO**
7. **user\_check\_connection KO**
8. **user\_check\_connection KO**
9. **user\_disconnect KO**
10. **user\_disconnect\_all KO**
11. **user\_lost\_pwd KO**
12. **user\_identity KO**

### Crée un utilisateur

Crée un nouveau compte utilisateur.

Informations

Identifiant : user\_create\_account

Acteur : Administrateur

Entrée

Type Description

* Mail Adresse du nouvel utilisateur
* Identifier Nom d’utilisateur
* Password Mot de passe
* Name Identifiant client (module Client)

Sortie

Type Description

Procédure

1. Vérifie si le nom d’utilisateur ou le mail existe déjà (USER\_NAME\_EXISTS, USER\_MAIL\_EXISTS)
2. Insert une entrée dans la table USER\_ACCOUNT

### Inscrit un utilisateur

Crée un nouveau compte utilisateur en attente d’activation.

Informations

Identifiant : user\_register\_account

Acteur : Visiteur

Entrée

Type Description

* Mail Adresse du nouvel utilisateur
* Identifier Nom d’utilisateur

Sortie

Type Description

Procédure

1. Vérifie si le nom d’utilisateur ou le mail existe déjà (USER\_NAME\_EXISTS, USER\_MAIL\_EXISTS)
2. Vérifie si l’adresse mail existe déjà dans les inscriptions (USER\_MAIL\_REGISTRED)
3. Génère le code d’activation
4. Insert une entrée dans la table USER\_REGISTER
5. Envoie un mail de confirmation à l’utilisateur avec le code d’activation (module Mail)

### Active un utilisateur

Convertie une inscription en compte utilisateur.

Informations

Identifiant : user\_activate\_account

Acteur : Visiteur

Entrée

Type Description

* Mail Adresse mail
* Identifier Nom d’utilisateur
* Name Jeton d’activation
* Password Mot-de-passe

Sortie

Type Description

Procédure

1. Vérifie si le nom d’utilisateur ou le mail existe déjà (USER\_NAME\_EXISTS, USER\_MAIL\_EXISTS)
2. Vérifie si le nom d’utilisateur et le mail existe dans les demandes (USER\_REGISTRATION\_NOT\_EXISTS)
3. Vérifie le jeton (USER\_REGISTRATION\_NOT\_EXISTS)
4. Appel le cas**[user\_create\_account]**
5. Supprime l’inscription de la table USER\_REGISTER
6. Envoie un mail de confirmation à l’utilisateur avec le code d’activation (module Mail)

### Supprime un utilisateur

Supprime définitivement un compte utilisateur

Informations

Identifiant : user\_delete

Acteur : Administrateur

Entrée

Type Description

* Identifier Nom d’utilisateur

Sortie

Type Description

Identifier Identifiant de connexion

Procédure

1. Vérifie que le nom d’utilisateur existe (USER\_ACCOUNT\_NOT\_EXISTS)
2. Déconnecte l’utilisateur, appel le cas**[user\_disconnect\_account]**
3. Supprime l’entrée de la table USER\_ACCOUNT

### Connexion (session existante)

Crée une connexion depuis une session et un compte existant.

Informations

Identifiant : user\_connect\_to\_session

Acteur : Utilisateur

Entrée

Type Description

* IPv4 Adresse IP du client
* Integer Temps de vie de la session (en seconde)
* Identifier Nom d’utilisateur
* Password Mot de passe
* Identifier Identifiant de session

Sortie

Type Description

Identifier Identifiant de connexion

Procédure

1. Vérifie que le compte utilisateur existe (USER\_ACCOUNT\_NOT\_EXISTS)
2. Vérifie que le compte utilisateur est activé (USER\_ACCOUNT\_UNACTIVATED)
3. Vérifie l’existence de la session dans la table USER\_SESSION (USER\_SESSION\_NOT\_EXISTS)
4. Vérifie si l’entrée existe dans la table USER\_CONNECTION (Adresse IP et Nom d’utilisateur)
   1. Insert une entrée dans la table USER\_CONNECTION
5. Actualise le temps de vie et l’identifiant de session dans la table USER\_CONNECTION
6. Crée la tache de fermeture automatique de connexion (userCloseConnection) à la date d’expiration estimée (date actuelle + temps de vie)

### Connexion (nouvelle session)

Crée une connexion entre un compte utilisateur et une session **Client**.

Informations

Identifiant : user\_connect\_to\_client

Acteur : Utilisateur

Entrée

Type Description

* IPv4 Adresse IP du client
* Integer Temps de vie de la session (en seconde)
* Identifier Nom d’utilisateur
* Password Mot de passe

Sortie

Type Description

* Identifier Identifiant de connexion
* Identifier Identifiant de session

Procédure

1. Vérifie que le compte utilisateur existe (USER\_ACCOUNT\_NOT\_EXISTS)
2. Vérifie que le compte utilisateur est activé (USER\_ACCOUNT\_UNACTIVATED)
3. Vérifie l’existence d’une connexion pour ce compte utilisateur (USER\_CONNECTION\_EXISTS)
4. Vérifie l’existence de la session dans la table USER\_SESSION (USER\_SESSION\_NOT\_EXISTS)
5. Vérifie si l’entrée existe dans la table USER\_CONNECTION (Adresse IP et Nom d’utilisateur)
   1. Insert une entrée dans la table USER\_CONNECTION
6. Actualise le temps de vie et l’identifiant de session dans la table USER\_CONNECTION
7. Crée la session dans la table USER\_SESSION
8. Crée un lien public vers le répertoire du dossier client portant le nom de cet utilisateur (procédure clientLinkData)
9. Crée la tâche de fermeture automatique de connexion (userCloseConnection) à la date d’expiration estimée (date actuelle + temps de vie)

### Actualise l’état d’une connexion

Actualiser une connexion permet de signaler la présence d’un utilisateur et d’obtenir des informations sur la connexion en cours.

Informations

Identifiant : user\_check\_connection

Acteur : Utilisateur

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| UC2 | | | Actualise l’état de la connexion | |  |
| Intentions et objectifs | | | Vérifier l’état de la connexion et renvoie le chemin d’accès aux fichiers. | | |
| Conditions | | | L’identifiant de connexion est connu  ou  Le login, le mot-de-passe et l’IP de l’utilisateur est connue | | |
| Résultat | | | La date d’expiration est actualisée (temps de vie + date en cours).  Le chemin d’accès au système de fichier est retourné. | | |
| Déclenchement | | | L’utilisateur actualise l’état de sa connexion | | |
| Scénario principal | | |  |  | |
|  | **Acteur** | **Etape** | **Action** | | |
|  | Utilisateur | **P1** | Saisie l’identifiant de connexion | | |
|  | Système | **P2** | Vérifie l’existence de la connexion (E1, E2) | | |
|  | Système | **P3** | Actualise la date d’expiration (date system + durée de vie) | | |
|  | Système | **P4** | Actualise la tâche fermeture automatique (E3) | | |
|  |  |  |  | | |
|  |  |  |  | | |
|  |  |  |  | | |
| Scénario alternatif | | |  | | |
|  | **Acteur** | **Etape** | **Action** | | |
|  | Système | **A1** | Si la connexion n’existe pas, informe l’utilisateur que le mot de passe et/ou le nom d’utilisateur est invalide.  [Retour à l’étape P1] | | |
|  | Système | **A2** | Si la session n’existe pas, la session est créée avec le nom de dossier client associé au compte utilisateur et le chemin d’accès. | | |
| Exceptions | | |  | | |
|  | **Code d’erreur** | **Etape** | **Action** | | |
|  | USER\_CONNECTION\_NOT\_FOUND | **E1** | La connexion n’existe pas. | | |
|  | USER\_CONNECTION\_IP\_REFUSED | **E2** | L’Adresse IP ne correspond pas. | | |
|  | SYS\_UPDATE\_TASK | **E3** | La tâche ne peut pas être mise à jour. | | |

### Déconnexion

Termine une connexion entre un compte et une session

Informations

Identifiant : user\_disconnect

Acteur : Utilisateur

Entrée

Type Description

* Identifier Nom d’utilisateur
* IPv4 Adresse IP
* Name Id de connexion

Procédure

1. Vérifie l’existence de la connexion (USER\_CONNECTION\_NOT\_EXISTS)
2. Vérifie l’adresse IP cliente (USER\_IP\_REFUSED)
3. Appel le cas**[user\_disconnect\_account]**

### Déconnecte un utilisateur

Termine une connexion entre un compte et une session

Informations

Identifiant : user\_disconnect\_account

Acteur : Administrateur

Entrée

Type Description

* Identifier Nom d’utilisateur

Procédure

1. Vérifie que le nom d’utilisateur existe (USER\_NOT\_EXISTS)
2. Supprime l’entrée correspondante dans la table USER\_CONNECTION
3. Supprime la session (si automatique)
4. Supprime la tâche de fermeture automatique de connexion

### Déconnecte tous les utilisateurs

Identifiant : user\_disconnect\_all

Acteur : Administrateur

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| UC4 | | | Déconnexion de tous les utilisateur | |  |
| Intentions et objectifs | | | Terminer toutes les connexions utilisateur | | |
| Conditions | | |  | | |
| Résultat | | | Toutes **les connexions** sont supprimées de la base de données.  Toutes **les sessions** **automatiques** sont supprimées de la base de données.  Tous les liens de sessions automatiques sont supprimés du **système de fichiers**. | | |
| Déclenchement | | | L’Administrateur demande la fermeture de toutes les connexions | | |
| Scénario principal | | |  |  | |
|  | **Acteur** | **Etape** | **Action** | | |
|  | Administrateur | **P1** | Confirme la fermeture des connexions | | |
|  | Système | **P2** | Supprime toutes les connexions | | |
|  | Système | **P3** | Supprime toutes les sessions automatiques | | |
|  | Système | **P5** | Retourne le code de résultat ERR\_OK. | | |
| Scénario alternatif | | |  |  | |
|  | **Acteur** | **Etape** | **Action** | | |
|  |  |  |  | | |
| Scénario principal | | |  |  | |
|  | **Acteur** | **Etape** | **Action** | | |
|  |  |  |  | | |
| Exceptions | | |  |  | |
|  |  | **Code** | **Action** | | |
|  |  |  |  | | |

### Obtient un mot-de-passe perdu

Informations

Identifiant : user\_lost\_pwd

Acteur : Utilisateur

Entrée

Type Description

* Identifier Nom d’utilisateur
* Password Mot de passe

Sortie

Type Description

Identifier Identifiant de connexion

Procédure

1. …

### Renseigne une identité

Informations

Identifiant : user\_identity

Acteur : Utilisateur

Entrée

Type Description

* Identifier Nom d’utilisateur
* Password Mot de passe

Sortie

Type Description

Identifier Identifiant de connexion

Procédure

1. …

# Développement

## Point d’entrée de l’application

Conformément au projet **Webframework**, ce projet implémente l’interface IModule.

Vous trouverez le code de la classe UserModule dans le dossier « wfw-1.x/lib » avec l’ensemble des classes du modèle, vous devez inclure le fichier « module.php » à votre application pour pouvoir utiliser les fonctionnalités du module **Utilisateur**.

## Modèle MVC

Le projet est construit sur le modèle <**Modèle-Vue-Contrôleur**>.

* Les classes C++ du modèles sont stockées dans le dossier « wfw-1.x/lib **»**.
* Les classes de vues sont stockées dans le dossier « public/view »
* Les classes contrôleurs sont stockées à la racine du dossier « **/**public**».**

## API Références

La documentation du code source est disponible dans le dossier « doc », il contient une arborescence web (HTML) généré à partir du code source.

Les fichiers de projet **Doxygen** est situé dans le dossier «  doxygen ».

## Modèle objet et Classes Managers

Le modèle de données a son équivalent en classes objets sous **PHP**.