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



Contenu

[Introduction 3](#_Toc347828044)

[Termes 4](#_Toc347828045)

[Environnement logiciel 5](#_Toc347828046)

[Arborescence 6](#_Toc347828047)

[Modèle de données 7](#_Toc347828048)

[Entités 7](#_Toc347828049)

[Domaines 9](#_Toc347828050)

[Formats 10](#_Toc347828051)

[Relations 11](#_Toc347828052)

[Contraintes 11](#_Toc347828053)

[Operations système 12](#_Toc347828054)

[Acteurs 12](#_Toc347828055)

[Visiteur 12](#_Toc347828056)

[Utilisateur 12](#_Toc347828057)

[Administrateur 12](#_Toc347828058)

[Système 12](#_Toc347828059)

[Résultats d’opération 13](#_Toc347828060)

[Détail des codes 13](#_Toc347828061)

[Cas d’utilisation 14](#_Toc347828062)

[Crée un utilisateur 15](#_Toc347828063)

[Inscrit un utilisateur 16](#_Toc347828064)

[Active un utilisateur 17](#_Toc347828065)

[Supprime un utilisateur 18](#_Toc347828066)

[Connexion (session existante) 19](#_Toc347828067)

[Connexion (nouvelle session) 20](#_Toc347828068)

[Maintient une connexion 21](#_Toc347828069)

[Déconnexion 22](#_Toc347828070)

[Déconnecte un utilisateur 23](#_Toc347828071)

[Déconnecte tous les utilisateurs 24](#_Toc347828072)

[Obtient un mot-de-passe perdu 25](#_Toc347828073)

[Renseigne une identité 26](#_Toc347828074)

[Développement 27](#_Toc347828075)

[Point d’entrée de l’application 27](#_Toc347828076)

[Modèle MVC 27](#_Toc347828077)

[API Références 27](#_Toc347828078)

[Modèle objet et Classes Managers 27](#_Toc347828079)

[Paramètres 28](#_Toc347828080)

[Cookies 29](#_Toc347828081)

[Notes d’implémentations 30](#_Toc347828082)

# Introduction

Module Utilisateur est utilisé pour implémenter une gestion des comptes et connexions d’utilisateurs pour votre site web.

Fonctionnalités :

* Gestion des comptes utilisateurs
* Gestion des sessions par jeton de connexion de et adresse IP
* Gestion des informations d’indentification (identité)
* Gestion des informations de location (adresses)

# Termes

Webframework

Projet principal de ce module.

# Environnement logiciel

Webframework

Ce module est développé pour le 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**).

PHP

Ce module est programmé avec **PHP version 5.4.3**.

Les modules suivants doivent être disponibles :

* **php\_pgsql**

Base de données

Le **système de base de données** utilisé est de type **PostgreSQL 8.3**.

Environnement de développement intégré (IDE)

Le module est développé sous l’environnement **NetBeans 7.2.1** avec les langages **PHP/Javascript/XHTML et SQL.**

Suivit de version

L’Utilitaire **GIT** est utilisé pour le suivit de version. La branche master est utilisée pour contenir la dernière version stable 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 en **UML** avec le logiciel **PowerAMC 15.1**.

**Important** :

Pour générer correctement le code **PHP depuis PowerAMC**, 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 modèle orienté objet et les classes **managers** permettant 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-XXX : 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\_ACCOUNT\_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 |
| USER\_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 |
| USER\_CONNECTION\_ID | Identifiant | VARCHAR(16) | PRIMARY KEY | Name |
| 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 |
| USER\_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 |
| USER\_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\_ACCOUNT USER\_CONNECTION
   1. Un **utilisateur** peut ouvrir plusieurs **connexions**
   2. Une **connexion** peut être ouverte par un **utilisateur**
2. USER\_CONNECTION USER\_SESSION
   1. Une **session** peut être attaché à plusieurs **connexions**
   2. Une **connexion** peut être attachée à une seule **session**
3. USER\_ACCOUNT USER\_IDENTITY
   1. Un **utilisateur** peut renseigner une **identité**
   2. Une **identité** peut être renseignée par plusieurs **utilisateurs**
4. USER\_IDENTITY USER\_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\_account\_id *et* user\_client\_ip *dans la table* user\_connection).
4. Un lien symbolique est propre à une seule connexion (Unicité de l’attribut LINK\_PATH dans la table user\_connection**).**

Note : Si un lien est partagé entre plusieurs connexions, une connexion fermée (supprimée) pourrait continuer à utiliser ce lien.

# Tests Unitaires

Les tests fonctionnels sont disponibles dans le répertoire PHP\_UNIT (**PHP**) et Q\_UNIT (**Javascript**).

Ces tests permettent d’informer sur le bon fonctionnement de l’API mais aussi aux développements futures de l’application.

Dans ces tests vous devrez retrouver l’ensemble des cas d’utilisation et des méthodes de l’API.

# Cas d’Utilisation

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**].

## Rôles

### Visiteur

Utilisateur non enregistré.

### Utilisateur

Utilisateur enregistré, possédant un compte utilisateur.

### Administrateur

Super-Utilisateur de l’interface de gestion.

## Résultats

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.

Notez que ces codes d’erreurs viennent s’ajouter aux codes existants du projet **Webframework**. Certains d’entre eux peuvent être réutilisés dans ce projet.

### Détail des codes

**Contexte Code Description**

ERR\_OK USER\_CREATED Utilisateur créé

ERR\_OK USER\_DELETED Utilisateur supprimé

ERR\_OK USER\_REGISTRED Inscription réussite

ERR\_OK USER\_DISCONNECTED Utilisateur déconnecté

ERR\_OK USER\_EXISTS L’Utilisateur existe

ERR\_OK USER\_IDENTITY\_UPDATED Identité mise à jour

ERR\_OK USER\_ADDRESS\_UPDATED Adresse mise à jour

ERR\_FAILED USER\_EXISTS L’Utilisateur existe déjà

ERR\_FAILED USER\_NOT\_EXISTS L’Utilisateur n’existe pas

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

ERR\_FAILED USER\_MAIL\_EXISTS L’adresse e-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 e-mail

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

ERR\_FAILED USER\_CONNECTION\_NOT\_EXISTS La connexion n’existe pas

ERR\_FAILED USER\_CONNECTION\_IP\_REFUSED L’Adresse IP ne correspond pas

ERR\_FAILED USER\_CONNECTION\_EXPIRED La connexion à expirée

ERR\_FAILED USER\_AUTHENTICATION\_FAILED Le nom d'utilisateur ou le mot-de-passe est invalide

ERR\_FAILED USER\_IDENTITY\_NOT\_FOUND Aucune identité n'est définit pour cet utilisateur

ERR\_SYSTEM SYS\_TASK\_CREATE La tâche ne peut pas être créée

ERR\_SYSTEM SYS\_TASK\_UPDATE La tâche ne peut pas être mise à jour

### Détail des messages

|  |  |  |
| --- | --- | --- |
| Contexte | Code | Description |
| USER\_REGISTRED | USER\_MSG\_ACTIVATE\_BY\_MAIL | Merci de consulter vos courriers électroniques pour activer votre compte utilisateur. |
| USER\_REGISTRED | USER\_MSG\_AUTO\_ACTIVATE | Votre compte est maintenant créé.  Votre mot-de-passe de connexion est le suivant: $PWD.  Conservez ce code en lieu sûr, il ne vous sera plus communiqué à l’avenir. |
| ERR\_OK | USER\_PWD\_LOST\_MAIL\_SENT | Merci de consulter vos courriers électroniques, un mail avec votre identifiant et votre mot-de-passe vous a été envoyé. |
| ERR\_OK | USER\_MSG\_TOKEN\_BY\_MAIL | Merci de consulter vos courriers électroniques, un jeton d’activation avec vos identifiants vous a été envoyé. |
| ERR\_OK | USER\_TOKEN\_MAIL | Vous avez demandez le jeton d'activation de votre compte utilisateur.  Voici vos informations d'identification:  Votre nom d'utilisateur : $USER\_ID  Votre adresse eMail : $USER\_MAIL  Votre jeton d'activation: $USER\_TOKEN |

## Contrôleurs

Les cas d’utilisations (ou Use-Cases) liste les procédures et leurs interactions avec le système. Il s’agit des fichiers contrôleurs présent dans l’application.

A retenir pour définir un cas d’utilisation :

1. La procédure écrite est soit validée dans sa totalité, soit annulée. Dans le cas contraire et pour maintenir un système cohérant, aucunes des actions précédentes ne doivent être conservées
2. L’identifiant, les champs d’entrées et de sorties doivent correspondre à la définition du contrôleur écrit dans l’application

Etat de l’implémentation

Cette liste informe sur l’avancement du développement. Un statut OK indique que le cas d’utilisation est rédigé et codé dans l’application, un statut OK indique un cas partiellement implémenté, un statut KO indique un développement en cours

|  |  |  |  |
| --- | --- | --- | --- |
| **Cas d’utilisation (contrôleur)** | **Statut** | **Rôle** | **Priorité** |
| create\_account | OK | Administrateur | Requis |
| register\_account | OK | Visiteur | Requis |
| activate\_account | OK | Visiteur | Requis |
| delete | OK | Administrateur | Requis |
| connect\_to\_session | KO | Utilisateur | Intermédiaire |
| connect | OK | Utilisateur | Requis |
| disconnect | OK | Utilisateur | Requis |
| get\_activation | KO | Visiteur | Intermédiaire |
| create\_session | KO | Administrateur | Optionnel |
| delete\_session | KO | Administrateur | Optionnel |
| check | OK | Utilisateur | Requis |
| disconnect | OK | Utilisateur | Requis |
| disconnect\_user | OK | Administrateur | Requis |
| disconnect\_all | OK | Administrateur | Requis |
| lost\_pwd | OK | Utilisateur | Intermédiaire |
| identity | OK | Utilisateur | Intermédiaire |
| get\_identity | OK | Utilisateur | Intermédiaire |
| get\_address | OK | Utilisateur | Intermédiaire |
| send\_token | OK | Visiteur | Requis |

### 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. Si le module MAIL est présent :
   1. Envoie un mail de confirmation à l’utilisateur avec le code d’activation [mail\_send\_message]
6. Sinon :
   1. Génère un mot de passe
   2. Active le compte avec le mot de passe générique [**user\_activate\_account**]

### Active un compte 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)

### Récupérer un jeton d’activation

L'Utilisateur souhaite récupérer le jeton d’activation suite à une inscription

Informations

Identifiant : send\_token

Acteur : Visiteur

Entrée

Identifiant Description

user\_mail Mail de l’utilisateur

Sortie

Identifiant Description

message Message de confirmation

Procédure

1. Vérifie si l’inscription lié à l’adresse **mail** existe (USER\_REGISTRATION\_NOT\_EXISTS)
2. Envoie un mail contenant : les informations d’activation (USER\_ACCOUNT\_ID, USER\_TOKEN, USER\_MAIL)
3. Retourne un message de confirmation à l’utilisateur (USER\_MSG\_TOKEN\_BY\_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\_IDENTITY *en relation avec le compte*
4. *Supprime l’entrée de la table* USER\_ADDRESS *en relation avec le compte*
5. 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 à un compte utilisateur. La session est créée automatiquement pour cet utilisateur uniquement.

Informations

Identifiant : connect

Acteur : Utilisateur

Entrée

Identifiant Description

user\_account\_id Identifiant de l’utilisateur

client\_ip Adresse IP de l’utilisateur

user\_pwd Mot-de-passe

life\_time Temps de vie de session (minutes)

Sortie

**Nom Type Description**

* CONNECTION\_ID Name Identifiant de connexion

Procédure

1. Vérifie si le compte utilisateur existe (USER\_AUTHENTICATION\_FAILED)
2. Vérifie si le compte utilisateur est inactif (USER\_ACCOUNT\_UNACTIVATED)
3. Si une session existe (table user\_session)
   1. …
4. Sinon
   1. …
5. Si une connexion existe (table user\_connection)
   1. Actualise la connexion (champs life\_time, user\_session\_id, client\_ip)
6. Sinon
   1. Génère un identifiant de connexion (champ user\_connection\_id)
   2. Ajoute la connexion (champs life\_time, user\_connection\_id, user\_account\_id, user\_session\_id, client\_ip)
7. Crée un lien public vers le répertoire client portant le nom de cet utilisateur (client\_link\_data)
8. Crée la tâche de fermeture automatique de connexion (user\_disconnect\_account) à la date d’expiration estimée (date actuelle + temps de vie)
9. (champ user\_connection\_id)

### Maintient une connexion

Actualiser une connexion permet de maintenir l’expiration d’une session utilisateur et d’obtenir des informations sur la connexion en cours.

Informations

Identifiant : check

Acteur : Utilisateur

Entrée

Identifiant Description

user\_connection\_id Identifiant de connexion

client\_ip Adresse IP de l’utilisateur

Sortie

Identifiant Description

expire Date d’expiration de la connexion

uid Identifiant du compte utilisateur (user\_connection\_id)

Procédure

1. Vérifie l’existence de la connexion (table : user\_connection) (USER\_CONNECTION\_NOT\_EXISTS)
2. Vérifie l’adresse IP (USER\_CONNECTION\_IP\_REFUSED)
3. Actualise la date d’accès (table : user\_connection)
4. Calcule la nouvelle date d'expiration (date en cours + life\_time)Déconnexion

Termine une connexion entre un compte et une session

Informations

Identifiant : disconnect

Acteur : Utilisateur

Entrée

Identifiant Description

user\_connection\_id Identifiant de connexion

Procédure

1. Supprime l’entrée correspondante (table : user\_connection)

### Déconnecte une connexion

Termine la connexion spécifiée

Informations

Identifiant : disconnect

Acteur : Administrateur

Entrée

Identifiant Description

user\_connection\_id Identifiant de connexion

Procédure

1. Supprime la connexion (table : user\_connection)
2. Supprime la tâche de fermeture automatique de connexion

### Déconnecte un utilisateur

Termine une connexion entre un compte et une session

Informations

Identifiant : disconnect\_user

Acteur : Administrateur

Entrée

Identifiant Description

user\_account\_id Nom d’utilisateur

Procédure

1. Vérifie si le compte utilisateur existe (user\_not\_exists)
2. Supprime l’entrée correspondante (table : user\_connection)
3. Supprime la session (si automatique)
4. Supprime la tâche de fermeture automatique de connexion

### Déconnecte tous les utilisateurs

Déconnecte tous les utilisateurs

* 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.

Informations

Identifiant : disconnect\_all

Acteur : Administrateur

Procédure

1. Supprime toutes les connexions (table : user\_connection)
2. Supprime toutes les sessions automatiques (table : user\_session)

### Récupérer un mot-de-passe

L'Utilisateur souhaite récupérer son mot-de-passe et son nom d’utilisateur oublié

Informations

Identifiant : lost\_pwd

Acteur : Visiteur

Entrée

Identifiant Description

user\_mail Mail de l’utilisateur

Sortie

Identifiant Description

message Message de confirmation

Procédure

1. Vérifie si le compte utilisateur lié à l’adresse **mail** existe (USER\_NOT\_EXISTS)
2. Envoie un mail contenant : le nom d’utilisateur (USER\_ACCOUNT\_ID) et le mot de passe (USER\_PWD)
3. Retourne un message de confirmation à l’utilisateur (USER\_PWD\_LOST\_MAIL\_SENT)

### Renseigne une identité

Crée ou modifie l'identité (nom, prénom, sexe, ...) lié à un compte utilisateur

Informations

Identifiant : identity

Acteur : Utilisateur

Entrée

Identifiant Description

user\_connection\_id Identifiant de connexion

last\_name Nom de famille

first\_name Prénom

birth\_day Date de naissance

sex Sexe

Procédure

1. Valide les informations de connexion utilisateur (cas : user\_check\_connection)
2. Définit/Crée les données de l'identité en BDD (table : user\_identity)

### Obtient une identité

Affiche l'identité (nom, prénom, sexe, ...) lié à un compte utilisateur

Informations

Identifiant : get\_identity

Acteur : Utilisateur

Entrée

Identifiant Description

user\_connection\_id Identifiant de connexion

Procédure

1. Valide les informations de connexion utilisateur (cas : user\_check\_connection)
2. Obtient l'identité liée à ce compte (table : user\_identity)

### Obtient une adresse

Affiche l'adresse (rue, ville, pays, ...) lié à un compte utilisateur

Informations

Identifiant : get\_address

Acteur : Utilisateur

Entrée

Identifiant Description

user\_connection\_id Identifiant de connexion

Procédure

1. Valide les informations de connexion utilisateur (cas : user\_check\_connection)
2. Obtient l’adresse liée à ce compte (table : user\_address)

# Développement

Ce module suit les recommandations de développement définit par le projet **Webframework**. Pour plus d’informations, consultez la section « **Modules** » disponible dans la documentation technique du projet **Webframework**.

## Paramètres de configuration

Voici les différentes constantes de configuration influant sur le module :

|  |  |  |  |
| --- | --- | --- | --- |
| [catalog\_module] | Requis | Définition | Valeur par défaut |
| ctrl\_path | **OUI** | Chemin d’accès aux contrôleurs | ${catalog\_module\_path}/ctrl/io |
| activation\_mail | **OUI** | Chemin d'accès au Template XML du mail d'activation | ${user\_module\_path}/view/user/templates/activation\_mail.html |
| pwd\_recovery\_mail | **OUI** | Chemin d'accès au Template XML du mail de récupération de mot-de-passe | ${user\_module\_path}/view/user/templates/pwd\_recovery\_mail.html |

## Cookies

Le module utilisateur utilise des cookies du navigateur client pour stocker des informations de connexion.

|  |  |  |
| --- | --- | --- |
| *Nom du cookie* | *Usage* | *Format* |
| cid | Identifiant de connexion. Obligatoire, est utilisé pour maintenir une connexion d’une page à une autre. | Name |
| uid | Nom d’utilisateur. Optionnel, est utilisé pour se reconnecter automatiquement. | Identifier |
| pwd | Identifiant de connexion. Optionnel, est utilisé pour se reconnecter automatiquement. | Password |
| cexpire | Date d’expiration de la connexion | Integer |

## Notes d’implémentations

Ici, sont notées les bugs, les modifications et les améliorations en attente d’implémentation. Cette section est réservée aux développeurs responsables du projet.

Liste des modifications à apporter au projet :

1. Implémenter la gestion des classes de management de tâche système dans la classe **Application**
2. Proposer un paramètre de configuration pour gérer les déconnexions automatiques soit : par utilisateur ou périodiquement pour tous
3. Ajouter une option permettant une obligation de renouvellement des mot-de-passes tous les 3 mois
4. Rappel puis Suppression automatique du compte après une longue période d’inactivité