	Titre du document : Livrable 3.3.2 – Module pivot
	Référence : L3.3.2
	Version du 13/10/2011


Livrable 3.3.2

Module pivot

Développement d'un module permettant d'interfacer l'application dashboard avec les forges.


Livrable dû au titre du projet	COCLICO
Lot	3
Tache	3.3
Livrable	L3.3.2 – Module pivot

Rédacteur(s)	Vérificateur(s)	Approbateur(s)
Mathieu Ruhlmann	Séverine Rambaud	

	Titre du document : Livrable 3.3.2 – Module pivot
	Référence : L3.3.2
	Version du 13/10/2011


Documents applicables
Annexe technique au projet COCLICO

Documents de références (pour information)
L3.3.1 – Application dashboard

	Titre du document : Livrable 3.3.2 – Module pivot
	Référence : L3.3.2
	Version du 13/10/2011

Gestion des versions

N° de version	Date	Auteurs	Modification apportées
1	13/10/11	Mathieu Ruhlmann	Version initiale du document

	Titre du document : Livrable 3.3.2 – Module pivot
	Référence : L3.3.2
	Version du 13/10/2011

Sommaire

La réalisation du sous-projet n°3 (« Adaptation des outils et méthodologies ») visait notamment à intégrer dans les forges les dernières méthodologies populaires de l'ingénierie logicielle, en particulier les méthodologies agiles telles que Scrum ou Kanban, décrites dans le livrable L3.3.1. Les différents travaux réalisés ont permis la création de modèles de projets spécifiques permettant d'adresser les spécificités de ces méthodologies, notamment en termes d'artefacts et d'indicateurs projets. Par ailleurs, une application permettant la visualisation des trackers sous la forme de dashboards a été développée, afin de coller à la philosophie de Scrum et Kanban selon laquelle l'avancement d'un projet ou d'un sous-projet (itération) doit être visuellement disponible à tout moment, tout en permettant très facilement la mise à jour régulière de l'avancement de chaque tâche par déplacement de celle-ci dans les différentes étapes d'un workflow de réalisation. L'application dashboard est présentée plus en détails dans le livrable L3.3.1. Le présent document se focalise sur un sous-ensemble de cette application : le module pivot, qui permet d'interfacer une forge avec l'application dashboard en question. Nous aborderons notamment dans ce document la question de la genericité, à savoir comment ce module pivot peut être implémenté pour brancher l'application dashboard sur n'importe quelle forge. Nous décrirons également l'API proposée par ce module pivot.



	Titre du document : Livrable 3.3.2 – Module pivot
	Référence : L3.3.2
	Version du 13/10/2011

Table des matières

1Rappels.....	6
1.1 L'application Dashboard.....	6
1.2 Le Module Pivot.....	6
2Description de l'API.....	8
2.1 TrackerDaoInterface.....	8
2.2 TicketDaoInterface.....	9

	Titre du document : Livrable 3.3.2 – Module pivot
	Référence : L3.3.2
	Version du 13/10/2011

1 Rappels

1.1 L'application Dashboard

Comme nous l'avons vu dans le livrable L3.3.1, l'application dashboard se présente sous la forme d'un plugin intégrable à une forge. Elle permet de répondre au besoin de représentation visuelle de l'avancement d'un projet ou sous-projet, besoin issu des méthodologies agiles qui ont considérablement modifié les pratiques de l'ingénierie logicielle ces dernières années.

L'application dashboard est une application web de type RIA (Rich Internet Application) réalisée en Flex (pour la partie IHM) et en PHP (pour la partie back-end). La communication entre les deux parties fait intervenir le protocole AMF. Celui-ci assure les échanges d'informations entre le client et le serveur via le web.

L'architecture générale de l'application dashboard est rappelée ci-dessous :

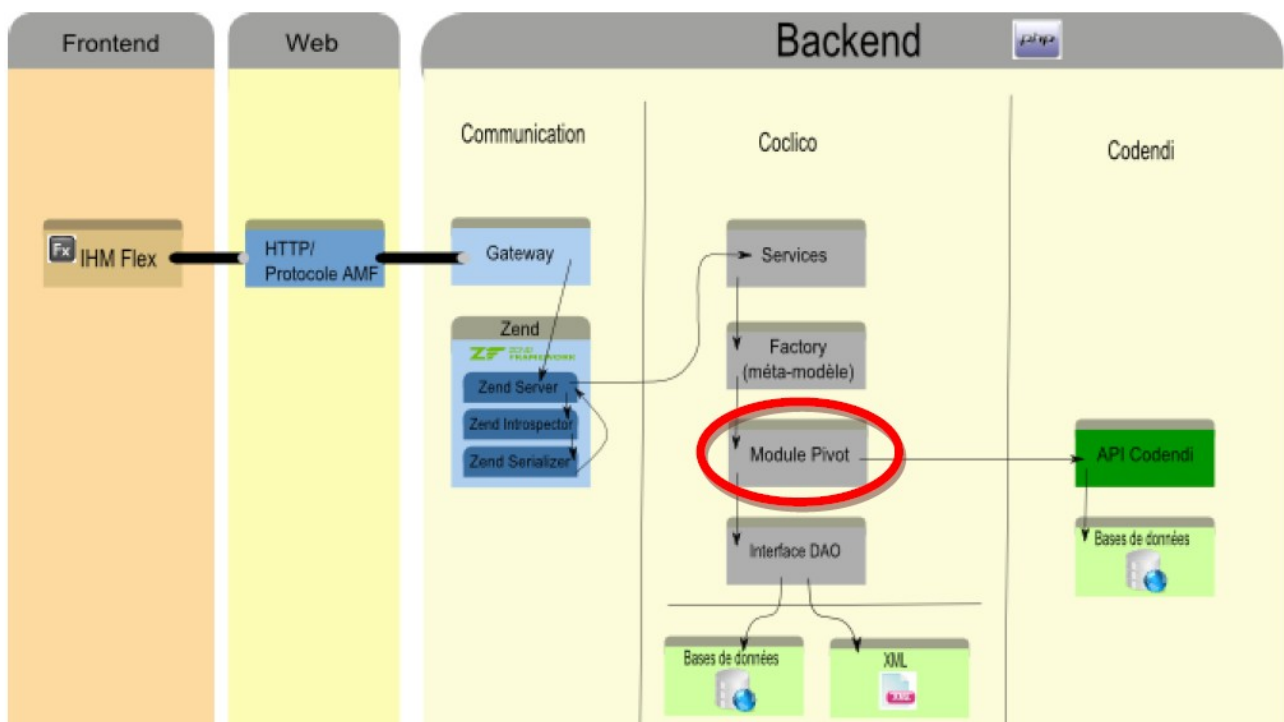



Illustration 1: Architecture générale de l'application Dashboard

1.2 Le Module Pivot

Comme nous pouvons le voir en rouge dans l'illustration ci-dessus, le back-end PHP fait intervenir un **module pivot** qui permet de faire le lien avec la forge sous-jacente. Il assure par ailleurs la généricité de l'application dashboard via l'isolation de la dépendance technique avec la forge sous-


	Titre du document : Livrable 3.3.2 – Module pivot
	Référence : L3.3.2
	Version du 13/10/2011

jacente. Ainsi, pour mettre l'application dashboard à la disposition d'une nouvelle forge, il faut principalement implémenter les fonctions du module pivot pour cette forge.

Ce module à deux fonctionnalités principales : la première est de fournir des objets du métier à partir des objets récupérés directement depuis la forge. La deuxième est à l'opposée : elle est de transformer des objets de notre modèle métier en objets de la forge, pour pouvoir lui transmettre les données modifiées (artefacts) qu'il faut sauvegarder.

En quelques mots, ce module assure le transfert des objets d'un domaine à l'autre, de la forge à l'application dashboard et réciproquement.

Il est à noter que l'utilisation d'un module pivot, interfaçant directement les services de la forge, a été choisie à la place d'OSLC ou de web services pour des raisons de performance.

	Titre du document : Livrable 3.3.2 – Module pivot
	Référence : L3.3.2
	Version du 13/10/2011

2 Description de l'API

L'API du module pivot se résume principalement à deux interfaces de type DAO (Data Access Objects), dont l'implémentation pour une forge donnée doit permettre d'assurer le transfert des objets d'un domaine à l'autre, de la forge en question à l'application dashboard et réciproquement.

2.1 TrackerDaoInterface

L'interface **TrackerDaoInterface** expose toutes les méthodes permettant de charger depuis la forge les informations liées aux trackers. Les deux méthodes principales qui sont exposées permettent :

- De **charger la liste des trackers** associés à un projet donné
- De **charger la liste des champs** associés à un tracker donné pour un projet donné

Le chargement de la liste des trackers intervient lors de la création d'un nouveau dashboard, où l'utilisateur commence par sélectionner le tracker sous-jacent.

Le chargement de la liste des champs associés au tracker intervient à plusieurs moments :

- Lors de la création d'un dashboard, à l'étape de sélection des champs à afficher en lignes et en colonnes, et à l'étape de sélection des champs à afficher sur chaque post-it
- Lors de la création d'un nouveau ticket depuis un dashboard, l'IHM présente à l'utilisateur un dialogue modal lui permettant de fournir toutes les propriétés du post-it créé.
- Lors de l'édition d'un ticket existant depuis un dashboard : l'IHM présente à l'utilisateur un dialogue modal lui permettant de modifier les propriétés du post-it.


Un extrait de l'interface **TrackerDaoInterface** est présenté ci-dessous.

```
interface TrackerDaoInterface {

    /**
     * Load all the trackers for the given project from the forge
     *
     * @param string $projectId
     * @return array of Coclico_Tracker
     */
    public function loadTrackersByProjectId($projectId);

    /**
     * Load all the fields for the given tracker and project.
     *
     * @param string $projectId
     * @param string $trackerId
     * @return array of Coclico_Field
     */
    public function loadAllFieldsByTrackerId($projectId, $trackerId);

}
```

	Titre du document : Livrable 3.3.2 – Module pivot
	Référence : L3.3.2
	Version du 13/10/2011

2.2 TicketDaoInterface

L'interface **TicketDaoInterface** expose toutes les méthodes permettant d'échanger des tickets (artefacts) entre le dashboard et la forge, en lecture et en écriture. Les méthodes principales qui sont exposées permettent :

- De **charger la liste des tickets** associés à un projet donné et à un tracker donné
- De **charger un ticket** à partir de son identifiant
- De **créer un nouveau ticket** pour un projet donné et un tracker donné
- De **modifier un ticket**
- De **déplacer un ticket** d'une cellule à une autre, c'est à dire de modifier certaines propriétés bien précises du ticket
- De **compter le nombre de tickets** qui s'appliquent à un ensemble de filtres
- De **charger la liste des tickets** d'un dashboard donné, en tenant compte de toutes les propriétés du dashboard, notamment les filtres.

Ces méthodes permettent de répondre aux différents besoin fonctionnels en relation directe avec les tickets, besoins qui interviennent à plusieurs niveaux :

- Dans le wizard de création (et de configuration) d'un dashboard , notamment pour ce qui est du comptage des tickets (à l'étape de vérification de la volumétrie et de création des filtres) ;
- Dans l'écran de visualisation d'un dashboard, notamment pour ce qui est du chargement des tickets associés au dashboard et du déplacement d'un ticket ;
- Dans les dialogues modaux de création et modification des tickets du dashboard, notamment pour ce qui est du chargement d'un ticket, de la création d'un nouveau ticket et de la mise à jour d'un ticket.


Un extrait de l'interface **TicketDaoInterface** est présenté ci-dessous.

```
interface TicketDaoInterface {

    /**
     * Get the list of all tickets for the given tracker and project
     *
     * @param string $projectId
     * @param string $trackerId
     * @return array of Coclico_Ticket
     */
    public function loadAllTicketsByTrackerId($projectId, $trackerId);

    /**
     * Load a ticket by its ID, project ID and tracket ID.
     *
     * @param string $projectId
     * @param string $trackerId
     * @param string $ticketId
     * @return Coclico_Ticket
     */
    public function loadTicketById($projectId, $trackerId, $ticketId);

    /**
     * Create a new ticket
     */
}
```

	Titre du document : Livrable 3.3.2 – Module pivot
	Référence : L3.3.2
	Version du 13/10/2011

```

*
* @param string $projectId
* @param string $trackerId
* @param Coclico_Ticket $ticket
* @return Coclico_Ticket
*/
public function createTicket($projectId, $trackerId, $ticket);

/**
* Move a ticket between two cells.
*
* Return NULL if the update was completed successfully.
* Return a String with the error message if an error occurred.
*
* @param string $projectId
* @param string $trackerId
* @param string $ticketId
* @param FieldModifiedContainerVO $fieldModifiesContainer
* @return NULL|string
*/
public function moveTicket($projectId, $trackerId, $ticketId,
$fieldModifiesContainer);

/**
* Update the given ticket
*
* Return NULL if the update was completed successfully.
* Return a String with the error message if an error occurred.
*
* @param string $projectId
* @param string $trackerId
* @param Coclico_Ticket $coclicoTicket
* @return string|NULL
*/
public function updateTicket($projectId, $trackerId,
$coclicoTicket);

/**
* Count the number of tickets that match the given list of
* filters, for the given project and tracker.
*
* @param string $projectId
* @param string $trackerId
* @param FiltrableFieldDataVO $filtrableFieldDataVO
* @return int
*/
public function countTicketsByFiltrableField($projectId,
$trackerId, $filtrableFieldDataVO);

/**
* Get all the tickets for the given dashboard (considering the
* dashboard properties, including filters)
*
* @param Coclico_Dashboard $dashboard
* @return array of Coclico_Ticket
*/
public function getTicketsWithFilters($dashboard);

```