# Introduction

## Généralité

|  |  |  |
| --- | --- | --- |
| **Projet** | | |
| **Nom du projet** | EasyGoing | |
| **Début du projet** | 14.09.2015 | |
| **Fin du projet** | 04.01.2016 | |
| **Durée effective** | 90 heures | |
| **Acteurs** | | |
| **Répondant** | Dr. René Rentsch | rene.rentsch@heig-vd.ch |
| **Développeurs** | Miguel Santamaria (Chef de projet)  Raphaël Racine (Suppléant)  Thibaud Duchoud  Michelle Vanessa Meguep Sakam  Karim Ghozlani | [miguel.santamaria@heig-vd.ch](mailto:miguel.santamaria@heig-vd.ch)  [raphael.racine@heig-vd.ch](mailto:raphael.racine@heig-vd.ch)  [thibaud.duchoud@heig-vd.ch](mailto:thibaud.duchoud@heig-vd.ch)  [michelle.meguepsakam@heig-vd.ch](mailto:michelle.meguepsakam@heig-vd.ch)  karim.ghozlani@heig-vd.ch |

## Objectifs du projet

Ce projet a pour objectif le développement d’un site web permettant la gestion de projets.

Il permet aux membres d’une équipe de gérer un projet autour d’un « scrum board ». Il offre la possibilité de visualiser différentes tâches, qui travaille sur l’une de ces dites tâches et l’avancement de celle-ci.

Des événements ont été mis en place et ceux-ci permettent de suivre les différentes actions effectuées par un utilisateur (par exemple lors du déplacement d’une tâche ou de l’ajout d’un membre au projet). Ce système est conçu comme un fil d’actualité.

Notre projet dispose également d’un tutoriel d’utilisation qui facilite grandement la tâche des nouveaux venus et d’une grande ergonomie

## Technologies utilisées et conventions de codage

### Technologies

Voici la liste des technologies utilisées :

* **Apache v2.4** comme serveur http.
* **PHP v5.6** comme langage coté serveur.
  + **Zend Framework 2** comme framework php.
* **MySQL v5.6** comme moteur de base de données.
  + **PHPMyAdmin v4.4 et MySQL Workbench v6.3** pour gérer la base de données.
* **HTML – CSS** comme langages de structure et design des pages web.
* **Javascript/JQuery v2.1** comme langage/bibliothèque pour dynamiser les pages web.
* **Différentes bibliothèques Javascript** (comme Bootstrap) pour réaliser des actions spécifiques concernant la gestion des boards principalement (cliquer-déposer, animation, gestion d’une grille …).
  + **ContextMenu :** Un menu contextuel en JQuery qui est utilisé pour faire différentes actions sur les tâches qui se trouvent dans le board.
  + **Bootbox :** C’est une bilbliothèque JQuery qui remplace les fenêtres d’alerte classique de Javascript par des fenêtres plus jolies.
  + **Datepicker et Datetimepicker :** Des calendriers JQuery qui permettent de sélectionner respectivement la date et l’heure.

### Conventions de codage

#### Général

Les conventions de codage que nous avons utilisées sont inspirées de celles utilisées dans le framework Zend.

Tout doit être écrit en anglais.

#### Nommage

Les noms de fonctions, de variables et de classes doivent être descriptifs : il faut éviter les abréviations.

Les noms des fichiers PHP correspondent à la classe qu’il contient.

Les variables et fonctions sont écrites en camelCase (première lettre en minuscule et chaque nouveau mot commence par une majuscule).

Pour les variables de classes (membres), si elles sont « protected » ou « private » ont les précède d’un « \_ ».

Les méthodes respectent les mêmes conventions que les variables de classes.

Les constantes sont écrites en majuscule et chaque mot est séparé par un « \_ ».

Les classes sont écrites en PascalCase (chaque mot commence par une majuscule).

#### Autres

En PHP, la première accolade suivant une instruction se place après un retour à la ligne. En Javascript, la première accolade se met directement après l’instruction.

L’indentation se compose de 3 espaces (donc pas de tabulation).

# Rôle du contrôleur « ProjectController »

Le contrôleur « ProjectController » est relatif à un projet en particulier. Il gère toutes les actions qui sont relatives à un projet. Il va, entre autres, afficher les informations relatives à un projet, les détails d’une tâche qui se trouve dans un projet, la liste des membres qui peuvent rejoindre ce projet. Il gère également l’ajout, suppression d’un membre dans le projet et l’ajout, suppression et édition d’une tâche ou sous-tâche dans le projet.

Voici les différentes actions qui le composent :

1. indexAction\* : cette action s’occupe de récupérer les différentes informations relatives au projet courant (le projet, les tâches et sous-tâches qui s’y trouvent, les membres, les événements et d’autres informations utilisées dans la vue). Elle va ensuite demander à la vue d’afficher ces différentes informations.
2. editAction\* : cette action va mettre à jour les informations d’un projet, après une édition de celui-ci. Elle réagit différemment selon si l’utilisateur a effectué une requête POST (c’est-à-dire qu’il a appuyé sur le bouton « Save changes ») ou non :
   1. Dans le premier cas, elle validera les différents champs entrés par l’utilisateur et les droits de celui-ci. Si les champs sont corrects et que l’utilisateur a les droits nécessaires pour modifier les propriétés d’un projet, elle met à jour les valeurs dans la base de données et redirige l’utilisateur sur la page d’index des projets ; s’ils ne sont pas valides ou qu’il n’a pas les droits, elle renvoie une erreur.
   2. Dans le second cas, l’utilisateur n’a pas fait de requête POST et donc elle appelle simplement la vue qui affichera le formulaire d’édition d’un projet.
3. addTaskAction\* : cette action s’occupe d’ajouter une nouvelle tâche au projet. Cette action va, comme la précédente, faire deux choses lorsqu’elle reçoit une requête POST ou non. Dans les 2 cas on va vérifier, si la tâche qu’on veut ajouter est une sous-tâche, que celle-ci est valide.
   1. Dans le cas d’une requête POST, elle vérifie les champs entrés par l’utilisateur. S’ils sont corrects on ajoute la tâche dans la base de données et, si ce n’est pas une sous-tâche, on l’affecte à son créateur. Si les champs sont incorrects elle renvoie une erreur.
   2. Si ce n’est pas une requête POST, elle va afficher le formulaire permettant l’ajout d’une tâche.
4. taskDetailsAction\* : cette action va afficher les détails d’une tâche. Elle récupère les différentes informations à afficher (les informations relatives à une tâche) et les envoie à la vue qui va nous être affichées.
5. editTaskAction\* : cette action s’occupe de l’édition d’une tâche. Elle fait également deux choses lorsqu’elle reçoit une requête POST ou non.
   1. Dans le cas d’une requête POST, elle vérifie les champs entrés par l’utilisateur. S’ils sont corrects elle met à jour les informations dans la base de données et redirige l’utilisateur sur la page d’index d’un projet.
   2. Dans le second cas, elle affiche la vue d’édition d’un projet en passant les informations relatives au projet qui y seront affichées.
6. boardViewMembersAction : cette action va retourner les informations relatives au « board » qui affiche les membres dans la colonne de gauche. Elle va récupérer les membres du projet (et des informations les concernant), leurs spécialisations et les tâches affectées à chaque membres. Elle va ensuite créer une vue en lui envoyant les informations récupérées et elle va enlever le « layout » de cette vue car celle-ci est chargée, du côté client, en ajax (requête « load »). La vue représente un board.
7. boardViewTasksAction : cette action va retourner les informations relatives au « board » qui affiche les tâches dans la colonne de gauche. Elle va récupérer les tâches principales (celles qui ne sont pas des sous-tâches) du projet et les membres affectés à chaque tâche. Elle va ensuite faire la même chose que l’action « boardViewMembersAction » mais c’est une autre vue qui est créée.
8. assignTaskAction\* : cette action gère l’assignement d’une tâche à un membre d’un projet. Elle est appelée lorsqu’un utilisateur « drag-drop » une tâche non assignée pour la mettre dans le « board » (il assigne la tâche à un membre). Les tâches sont assignées en déplaçant les liens qui se trouvent en dessous du « board » dans la liste des tâches du projet.

Elle va vérifier les droits de l’utilisateur et si la tâche est déjà assignée à l’utilisateur cible ou non. Dans le cas où l’utilisateur a les droits nécessaires et la tâche n’est pas déjà assignée à l’utilisateur cible, on ajoute une affectation dans la base de données. Sinon elle affiche une erreur.

1. moveTaskAction\* : cette action gère le déplacement d’une tâche dans le « board ». Lorsqu’un utilisateur déplace une tâche du « board » vers un autre utilisateur ou dans une autre colonne.

Elle va également vérifier les droits de l’utilisateur et, si l’utilisateur déplace la tâche vers un autre utilisateur, que celle-ci ne lui est pas déjà assignée. Dans le cas où tout est ok, elle met à jour les informations dans la base de données. Sinon elle affiche une erreur.

1. unassignTaskAction\* : cette action gère la désaffectation d’une tâche à un utilisateur. Elle est appelée lorsqu’un utilisateur fait un clic droit sur une tâche et sélectionne « Unassign ».

Elle va vérifier les droits de l’utilisateur et s’il a les droits, elle supprime l’affectation correspondante dans la base de données. Sinon elle affiche une erreur.

1. deleteTaskAction\* : cette action gère la suppression d’une tâche. Elle est appelée lorsqu’un utilisateur fait un clic droit sur une tâche et sélectionne « Delete ».

Elle va vérifier les droits de l’utilisateur et si ceux-ci sont suffisants elle supprime la tâche de la base de données et les affectations correspondantes et redirige l’utilisateur à la page d’index du projet. Sinon elle affiche une erreur.

1. addMemberAction\* : cette action gère l’ajout d’un utilisateur dans un projet. Elle a deux comportements différents en fonction de si la requête reçue est de type POST ou non. Quel que soit le type de requête, elle va vérifier les droits de l’utilisateur et si celui-ci n’a pas les bons elle le redirige vers la page d’index du projet.
   1. Dans le premier cas, elle va récupérer les informations postées et celles-ci correspondent au(x) utilisateur(s) à ajouter. Elle va parcourir ses informations et ajouter chaque utilisateur au projet.
   2. Dans le second cas, elle affiche la vue qui permet d’ajouter des utilisateurs au projet (affiche la liste des utilisateurs qui ne sont pas membres du projet et l’utilisateur peut les sélectionner afin de les ajouter).
2. removeMemberAction\* : cette action gère le fait de retirer un membre d’un projet. Elle va vérifier les droits de l’utilisateur et s’ils sont suffisants elle va mettre à jour la base de données en supprimant l’affectation du membre au projet et les spécialisations relatives à celui-ci.
3. detailsAction : cette action retourne, sous format JSON, les informations relatives à un projet. Elle est utilisée lorsqu’un déroule un projet dans la liste des projets.
4. postNewFeedAction\* : cette action gère l’ajout d’une nouvelle « news » correspondant à une tâche. Quand on est sur la page qui affiche le détail d’une tâche l’utilisateur à la possibilité d’ajouter une « news » sur cette tâche. Cette action est appelée lorsqu’un utilisateur ajoute une nouvelle « news » sur une tâche. Elle va récupérer les informations entrées par l’utilisateur et les ajouter dans la base de données. Elle retourne un JSON contenant un champ « success » qui est à « true » s’il n’y a pas d’erreur et à « false » sinon. Lorsque la requête reçue n’est pas de type POST elle retourne se même JSON avec le champ « success » à « false ».

\* Les actions suivies d’un astérisque (\*) contiennent toutes une gestion des événements. Pour chacune de ces tâches, des événements sont créés et insérés dans la base de données. Ces événements correspondent à ce qui est fait dans la tâche en question. Par exemple lorsqu’on retire un membre du projet (*removeMemberTask*) un événement de type utilisateur est créé avec la date courante et contenant le texte : « xxx removed user yyy from project, bye bye ! ». Le texte et le type d’événement change, bien évidemment, en fonction de l’action appelée.

Ces actions envoient, pour chaque événement, une requête au serveur qui gère les événements afin que les autres utilisateurs présent sur le projet en soit informé en temps réel.

(Pour plus d’informations c.f. le chapitre « Serveurs secondaires d’événements »).

# Droits

Sur le board (et donc dans le *ProjectController*), tout le monde ne peut pas tout faire. Certaines actions requièrent certains droits. Il existe 3 types de membres à un projet :

1. Le **créateur** ou le **super administrateur**. C’est celui qui a créé le projet et qui est, de fait, super administrateur du projet. Cet utilisateur à tous les droits sur un projet
2. Le **manager** ou **administrateur**. Cet utilisateur obtient ce rôle lorsqu’il est ajouté comme membre d’un projet. Il a plus de droits qu’un membre lambda mais moins que le créateur.
3. Un membre **lambda**. C’est un utilisateur ajouté au projet sans qu’il soit manager.

Donc par ordre de grandeur il y a **Créateur** > **Manager** > **Lambda**.

Les différentes actions sur les tâches et sous-tâches ne peuvent donc pas être faite par tous.

Voici une liste des droits par action (si l’action n’y apparaît c’est qu’il n’y a pas de gestion des droits, tout le monde peut y accéder) :

* **Affichage d’un projet (*indexAction*) :** Tous les membres d’un projet peuvent accéder au détail de ce projet et donc au board qui lui est associé.
* **Edition d’un projet (editTaskAction) :** Pour pouvoir éditer un projet, le membre doit au moins être un Manager.
* **Création d’une tâche (*addTaskAction*) :** Tous les membres d’un projet peuvent créer des tâches.
* **Affichage du détail d’une tâche (taskDetailsAction) :** Tous les membres d’un projet peuvent accéder aux détails d’une tâche et à son fil d’actualité.
* **Edition d’une tâche (editTaskAction) :** Tous les membres d’un projet peuvent modifier une tâche.
* **Assigner une tâche (assignTaskAction) :** Tous les membres d’un projet peuvent assigner les tâches à n’importe quel autre membre du projet.
* **Déplacer une tâche (moveTaskAction) :** Le créateur peut déplacer les tâches de tout le monde vers tout le monde. Le manager peut déplacer les tâches qui lui sont assignées vers n’importe qui. Il peut également déplacer les tâches des membres lambda vers n’importe qui mais il ne peut pas déplacer les tâches assignées aux autres managers ou au créateur.
* **Désassigner une tâche (unassignTaskAction) :** Le créateur peut désassigner une tâche qui est assignée à n’importe qui (y compris lui-même). Le manager peut désassigner les tâches qui sont assignées aux utilisateurs lambda ou à lui-même. Un utilisateur lambda ne désassigner aucune tâche (même une qui lui est associée).
* **Supprimer une tâche (deleteTaskAction) :** Seul le créateur peut supprimer une tâche.
* **Ajouter un utilisateur au projet (addMemberAction) :** Seul le créateur et les managers d’un projet peuvent ajouter des membres au projet et choisir si l’utilisateur ajouté est manager ou non et quels sont ses droits.
* **Retirer un utilisateur du projet (removeMemberAction) :** Le créateur peut retirer n’importe quel membre du projet (sauf lui-même). Un manager peut retirer les autres utilisateurs qui ne sont pas des managers. Les utilisateurs lambda ne peuvent rien faire.

# Fonctionnement du « Dash board »

## Généralités

Le « Dash board » est d’une grande importance dans notre projet. C’est à cet endroit que les membres d’un projet peuvent voir les différentes tâches qui existent et voir si celles-ci sont assignées à un ou plusieurs autres membres.

Ce « Dash board » se présente sous la forme d’un « Scrum board ».

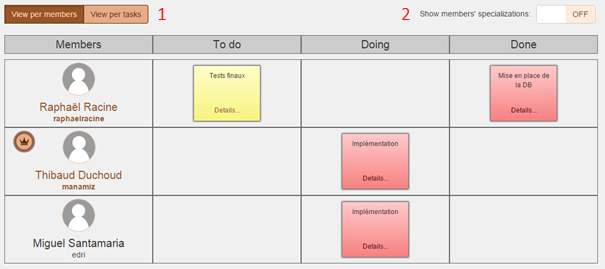


Figure 1 : Exemple d'un "Dash board"

On remarque que dans la colonne de gauche se trouve les membres du projet et dans les colonnes suivantes il y a les tâches assignées à chaque membre et leur avancement.

On voit également une différence de couleurs et une icône pour les membres du projet. Lorsqu’un utilisateur a son nom affiché en brun, il est manager du projet. Lorsqu’il a, en plus, une couronne cela signifie que ce membre est le créateur du projet. Lorsqu’il n’a rien de spécial, il est juste un membre lambda du projet.

Chaque tâche affichée l’est également avec une couleur correspondante à sa priorité. Pour une priorité « High » la tâche est de couleur rouge, pour une priorité « Medium » la tâche est de couleur jaune et pour une priorité « Low » la tâche est de couleur verte.

Allons voir plus en détail la composition de ce board.

(1) Au sommet se trouve un « radio button ». Celui sert à changer l’affichage du board. Lorsque l’on clique sur le bouton « View per members » c’est le board affiché ci-dessus qui affiché et c’est également cette vue là qui permet (et elle seule) de faire des modifications depuis le board (changer l’assignement des tâches …). C’est également cette vue qui est affichée par défaut.

Lorsque l’on clique sur le bouton « View per tasks », c’est un affichage du board par tâches.

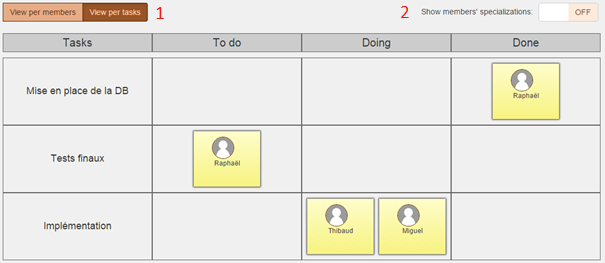


Figure 2 : Exemple d’un « Dash board » en vue par tâches

Comme on peut le voir c’est les tâches qui sont affichées dans la colonne de gauche et les utilisateurs affectés à ces tâches qui se trouvent dans les autres colonnes.

Comme énoncé plus haut, il n’y a que la vue par membres qui permet une quelconque édition. Cette vue n’est utilisée que pour afficher les informations d’une manière différente et plus utile dans certaine situation.

Lorsque l’on clique sur l’un des boutons, on envoie une requête à l’action correspondante à l’affichage du board voulu (*BoardViewMembersAction* ou *BoardViewTasksAction*) et l’on charge la réponse (qui est le code html du board) à l’emplacement voulu.

(2) Dans le coin en haut à droite se trouve un autre bouton. Celui permet, au choix, d’afficher ou non la spécialisation des différents membres du projet. Il n’est utile que dans la vue par membres.

La vue par tâche n’est donc pas du tout affectée par un appui sur ce bouton ou non.

Lorsque l’on choisit d’afficher la spécialisation des membres

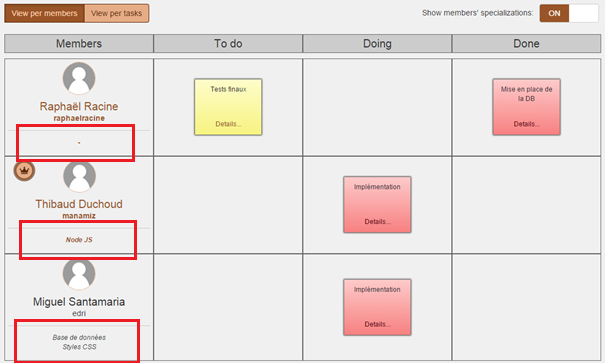


Figure 3 : Board avec l'affichage des spécialisations

On remarque un texte apparaître en dessous du nom des membres un texte désignant la ou les spécialisations. Si un des membres ne possède pas spécialisation, un tirer ( - ) est affiché à la place.

Lors d’un changement d’état du bouton, une animation est lancée et le texte est simplement affiché. Et le dernier état du bouton est également, à chaque changement, sauvegardé dans un cookie afin qu’il reste en état même si l’utilisateur continue de se balader dans les autres pages du site.

## Action sur les tâches

Lorsque l’on fait une clic droit sur une tâche dans le board, un menu contextuel s’ouvre et permet de faire différentes actions. Lorsque l’on clique sur un des choix, les informations sur la tâche correspondante sont récupérées (l’id de la tâche et l’id de l’utilisateur qui y est affecté).

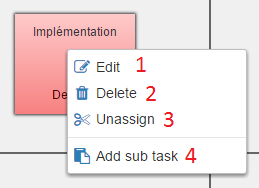


Figure 4 : Menu contextuel d'une tâche

### Edition (1)

L’édition se fait lorsque l’on choisit, dans le menu contextuel, le choix *Edit*.

Tout ce qui sera fait c’est que l’utilisateur sera redirigé sur la page d’édition d’une tâche (…/editTask/ « taskId »). C’est donc à l’action *editTaskAction* qu’une requête sera envoyée en y mettant à la place de *taskId* l’id de la tâche sélectionnée.

Un message sera affiché en cas de réussite ou d’erreur afin d’en informer l’utilisateur.

### Suppression (2)

Pour supprimer une tâche, il faut simplement choisir *Delete* dans le menu contextuel.

Il se passe la même chose que pour l’édition. Lorsque l’on sélectionne *Delete* il va simplement envoyer une requête à l’action *deleteTaskAction* (…/deleteTask/ « taskId »). L’id de la tâche sélectionné sera mise à la place de *taskId* dans l’url.

Un message sera affiché en cas de réussite ou d’erreur afin d’en informer l’utilisateur.

### Création d’une sous-tâche (4)

Pour créer une sous-tâche il suffit de choisir *Add sub task* dans le menu contextuel.

Tout ce qu’il va se passer est que l’utilisateur va être redirigé sur la page d’ajout d’une nouvelle tâche mais passant cette fois l’id de la tâche parent (…/addTask/ « parentId ») qui n’est autre que l’id de la tâche sélectionnée.. C’est donc l’action *addTaskAction* en mode sous-tâche qui est appelée et qui se charge d’ajouter la sous-tâche.

### Désassignation (3)

Pour désassigner une tâche, il suffit de choisir *Unassign* dans le menu contextuel.

Comme pour la suppression, on va envoyer une requête à l’action *unassignTaskAction* (…/unassignTask/ « taskId »). L’id de la tâche sélectionnée sera mise à la place de *taskId* dans l’url.

Un message sera affiché en cas de réussite ou d’erreur afin d’en informer l’utilisateur.

### Assignation

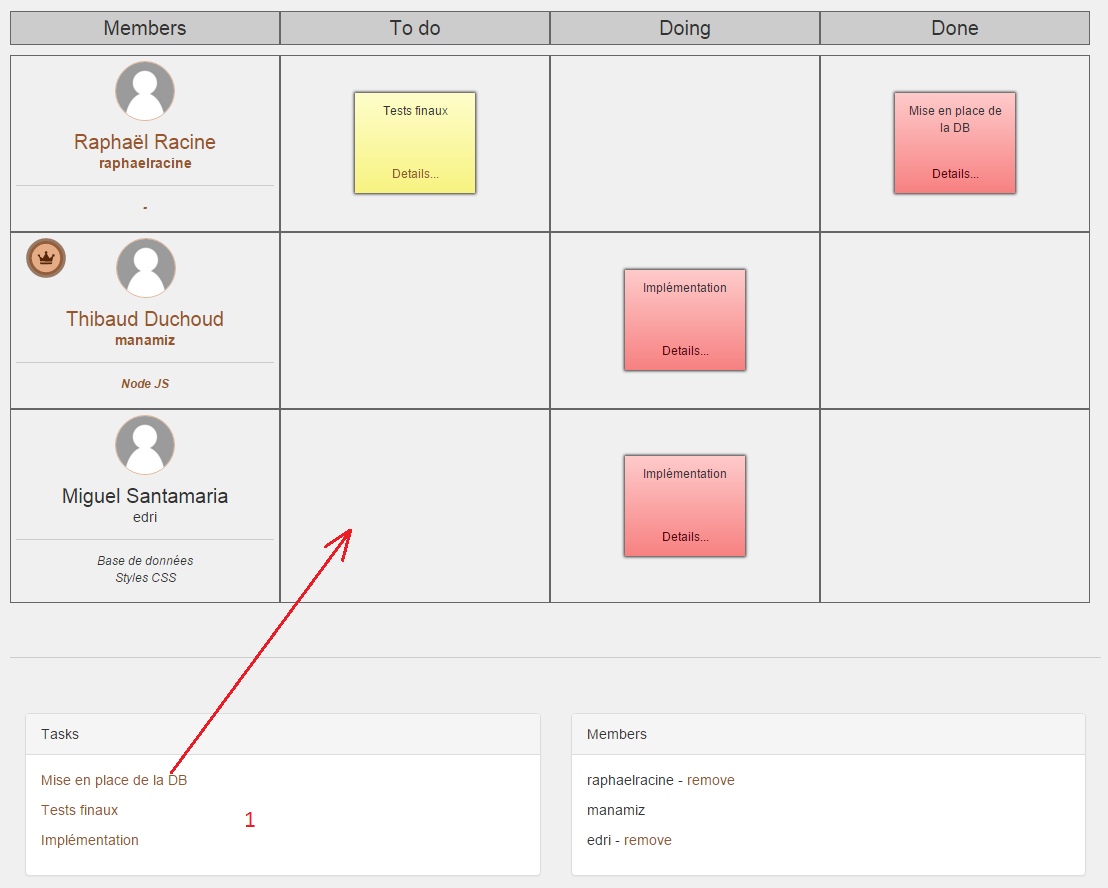


Figure : Assignement d'une tâche

En dessous du board, il y a la liste des tâches du projet qui est affichée (1). Lorsqu’un utilisateur veut assigner une tâche à un autre utilisateur, il fait une cliquer-glisser du lien dans la liste vers la ligne correspondante à l’utilisateur à qui il souhaite assigner la tâche.

Lorsque l’utilisateur lâche la tâche dans le board, l’événement *ondrop* est déclenché. Il va vérifier si ce qui vient d’être lâché est une tâche pour assignation ou une tâche qui est déplacée directement depuis le board. Dans notre cas il s’agit d’une assignation.

Cet événement va récupérer les informations nécessaires à l’affectation : l’id de l’utilisateur cible et l’id de la tâche.

Il va ensuite envoyer une requête au *ProjectController*à l’action *AssignTaskAction*. Il va ensuite récupérer le code renvoyé par la tâche est affiché un feedback. Si l’utilisateur n’a pas les droits ou qu’une erreur s’est produite (déjà assignée …) un message en rouge sera affiché.

### Afficher les détails

Il y a la possibilité d’afficher les détails d’une tâche et donc également le fil d’actualité qu’il lui est associé.

Deux manières sont possibles. Soit en cliquant sur une tâche dans la liste se trouvant en dessous du board sois en cliquant sur le lien *Details…* qui se trouve sur la tâche dans le board.

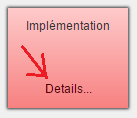
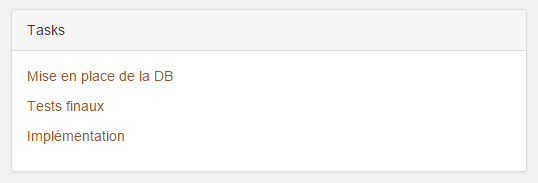


Figure : Liste des tâches d'un projet Figure 7: Détails d'une tâche

Lorsqu’on clique sur un de ces liens, l’utilisateur et redirigé sur la page d’affichage des détails d’une tâche (…/taskDetails/ « taskId ») et l’id de la tâche est passée dans l’url à la place de *taskId*.  
Lorsque l’utilisateur est redirigé sur cette page c’est l’action *taskDetailsAction* qui est appelée et qui s’occupe d’afficher la vue à l’utilisateur.

### Déplacer une tâche

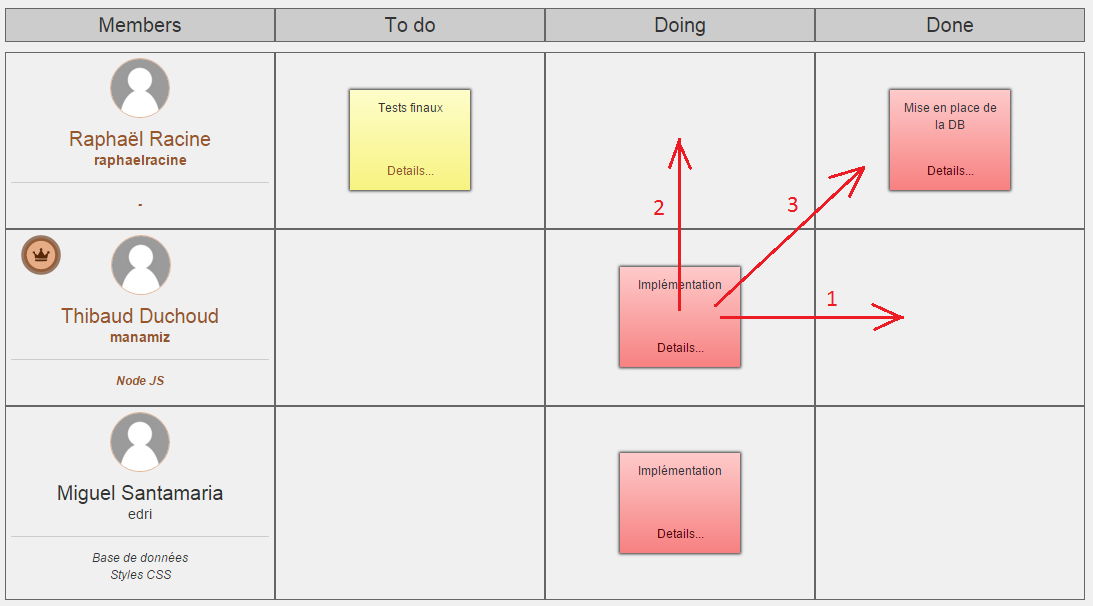


Figure 8: Board avec déplacement d'une tâche

Lors du déplacement d’une tâche dans le board, il y a plusieurs possibilités. Soit on déplace la tâche vers une autre colonne (de *Doing* à *Done* par exemple) (1). Soit on déplace la tâche vers un autre utilisateur (2). Soit les deux (3).

Comme cette action utilise le serveur d’événement, le déplacement s’opère en temps réel chez les autres membres du projet qui se trouvent également sur cette page.

1. Dans le premier cas, il n’y pas de changement de membre et donc pas de réaffectation. C’est juste l’état de la tâche qui change (qui passe de *Doing* à *Done* dans l’image ci-dessus). Ce déplacement va également faire bouger la tâche du fond car un état est lié à une tâche et non à un utilisateur qui travaille sur la tâche.
2. Dans ce second cas, l’état de la tâche ne change pas il y a une réaffectation.
3. Dans ce dernier cas, il y a une réaffectation de la tâche et un changement d’état de la tâche.

Lorsqu’un utilisateur déplace une tâche et qu’il la lâche, il appelle le même événement *ondrop* que pour l’assignation d’une tâche. Mais cette fois l’événement n’est pas en mode affectation.

L’événement va récupérer les informations nécessaires à la vérification, aux droits et aux événements concernant le déplacement.

Il récupère l’id de la tâche, le nom de la colonne de départ, le nom de la colonne d’arrivé, l’id de l’utilisateur de départ et l’id de l’utilisateur d’arrivé.

Ces informations sont ensuite envoyées à l’aide d’une requête POST à l’action *moveTaskAction* (…/moveTask). Cette action va faire les vérifications nécessaires et retourner un code de succès ou d’erreur.

## Action sur les sous-tâches

### Rappel sur les sous-tâches

Lorsqu’un utilisateur crée une sous-tâche, celle-ci est liée à sa tâche parent.

Les sous-tâches ne sont pas affichées dans le board mais dans la liste des tâches se trouvant en dessous du board. Elles sont affichées sous leur parent. Ces sous-tâches ne sont dont pas assignables et on ne peut pas les déplacer.

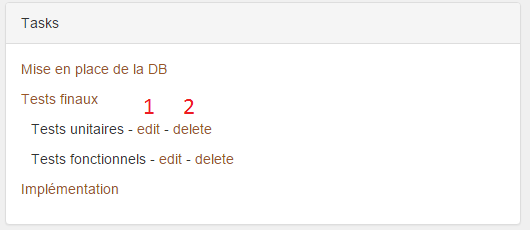


Figure 9: Liste des tâches avec sous-tâches

Les sous-tâches sont là pour pouvoir séparer et/ou différencier les différentes parties d’une tâche principale.

Une sous-tâche ne peut pas changer son parent. Il faut obligatoirement, pour changer le parent d’une sous-tâche, la supprimer et la recréer en sélectionnant le nouveau parent.

### Edition (1)

Pour éditer une sous-tâche, il suffit de cliquer sur le lien *edit* se trouvant à côté de la sous-tâche dans la liste.

Ce lien va rediriger l’utilisateur sur la page d’édition d’une tâche et donc laisser l’action *editTaskAction* gérer la suite. Cette action ne fait pas de différence entre une tâche et une sous-tâche. On lui envoie donc une requête normale comme quoi l’on souhaite éditer une tâche avec l’id qui lui correspond. La requête est envoyée à l’url : …/editTask/  « taskId », où le paramètre taskId est remplacé par l’id de la tâche correspondante.

### Suppression (2)

Pour éditer une sous-tâche, il suffit de cliquer sur le lien *delete* se trouvant à côté de la sous-tâche dans la liste.

Ce lien envoyer une requête à l’action *deleteTaskAction* à l’url : …/deleteTask/ « taskId », où le paramètre taskId est remplacé par l’id de la tâche correspondante.

La page sera alors rafraîchie pour prendre en compte les changements.