TD n°7

Dépôt eCampus : parcours 2

- Suite Projet **boards**
- Application gestion de projets SCRUM

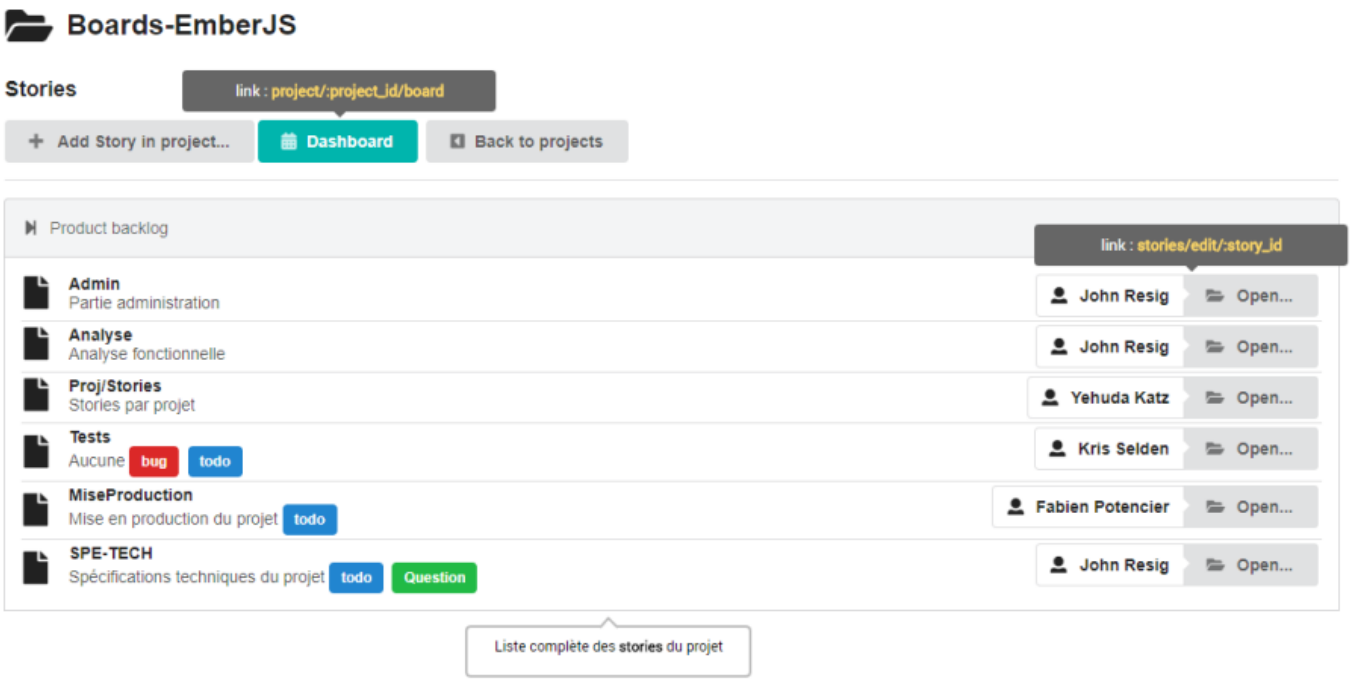
Objectifs

- Conception d'interfaces Web client riche
- Manipulations datas
- Utilisation de composants

-- Eléments à implémenter

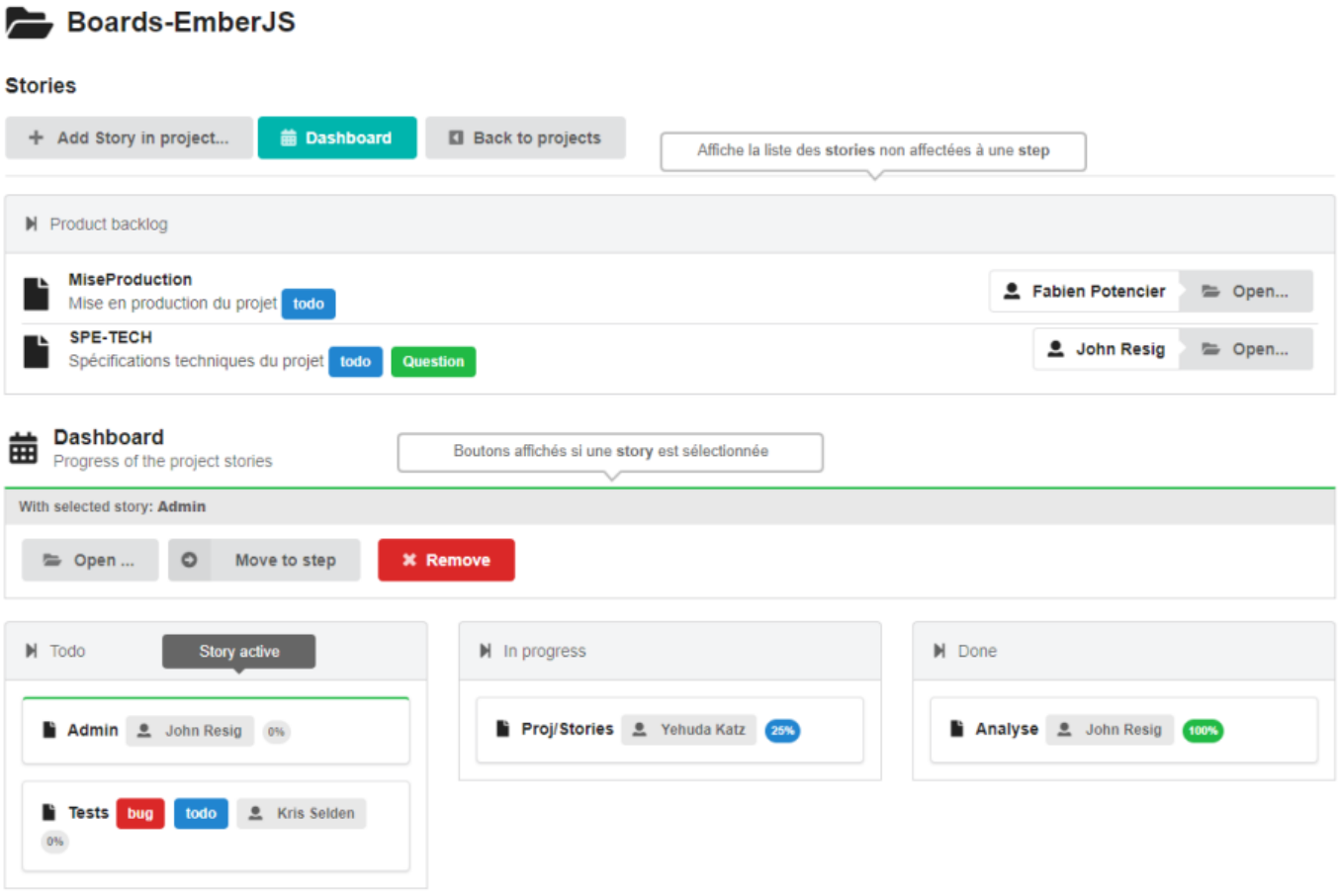
-- Route project/{idProject}/board

//TODO 2.1



Elle affiche la liste complète des stories du projet sélectionné.

on lui ajoute la partie tableau de bord, permettant d'afficher par steps (en colonnes), les user stories du projet sélectionné.



Comportement/conception de l'interface

Steps

Les Steps s'affichent de manière complète dans l'interface, même si elles ne contiennent aucune story.
On pourra utiliser :

- le composant htmlGrid de phpMv-UI pour leur disposition.
- Chaque **step** est composée d'un **ui top attached segment** pour son titre, suivi d'un **ui attached segment** pour son contenu.

Stories

Lorsque la route **project/{idProject}** est active, toutes les stories sont affichées dans le product backlog.
Lorsque la route **project/{idProject}/board** est active, toutes les stories non affectées à une step sont affichées dans le product backlog, les autres sont réparties entre leurs steps respectives.

- Le click sur une story présente dans le board la rend active et affiche la barre de 3 boutons (Open, MoveTo -dropDown des steps- et Remove).
- Les boutons **Open..** renvoient vers l'url **story/{idStory}**
- Il est possible de déplacer les stories entre les steps par un drap and drop

Si le composant HtmlGrid n'est pas utilisé, le tableau suivant pourra servir a élaborer la grille des steps :

```
['one', 'two', 'three', 'four',
        'five', 'six', 'seven', 'eight', 'nine',
        'ten', 'eleven', 'twelve', 'thirteen', 'fourteen',
        'fifteen', 'sixteen']
```

Exemple de mise en oeuvre du drag and drop

Soit les éléments HTML suivants définis dans un template:

```
<div class="drag-item" draggable="true" data-ajax="donnée du transfert">
Contenu draggable
</div>
```

Ajoute la possibilité de déplacer tous les éléments ayant la classe css **drag-item**, et définit leur propriété data à **data-ajax**

```
        $this->gui->setDraggable(".drag-item",["attr"=>"data-ajax"]);
```

Définit une ou plusieurs zones potentielles cible du drag and drop :

```
<div class="drop-zone">
Zone de drop
</div>
```

Définit les éléments ayant pour classe **drop-zone** comme zone de drop :

```
        $this->gui->asDropZone(".drop-zone");
```

Appel de code js après drop et récupération des données de l'élément déplacé et de la zone de drop :

```
        $this->gui->asDropZone(".drop-zone",'console.log("data du drag :
"+data);console.log("id du drop : "+$(event.target).attr("id"));');
```

Code de requête ajax

Pour obtenir le code d'une requête Ajax, mais sans l'exécuter :
Code d'une requête GET vers l'url **project/settings** dont le résultat sera affiché dans l'élément HTML d'id **settings-elm**

```
$js=$this->gui->getDeferred("project/settings","#settings-elm");
```

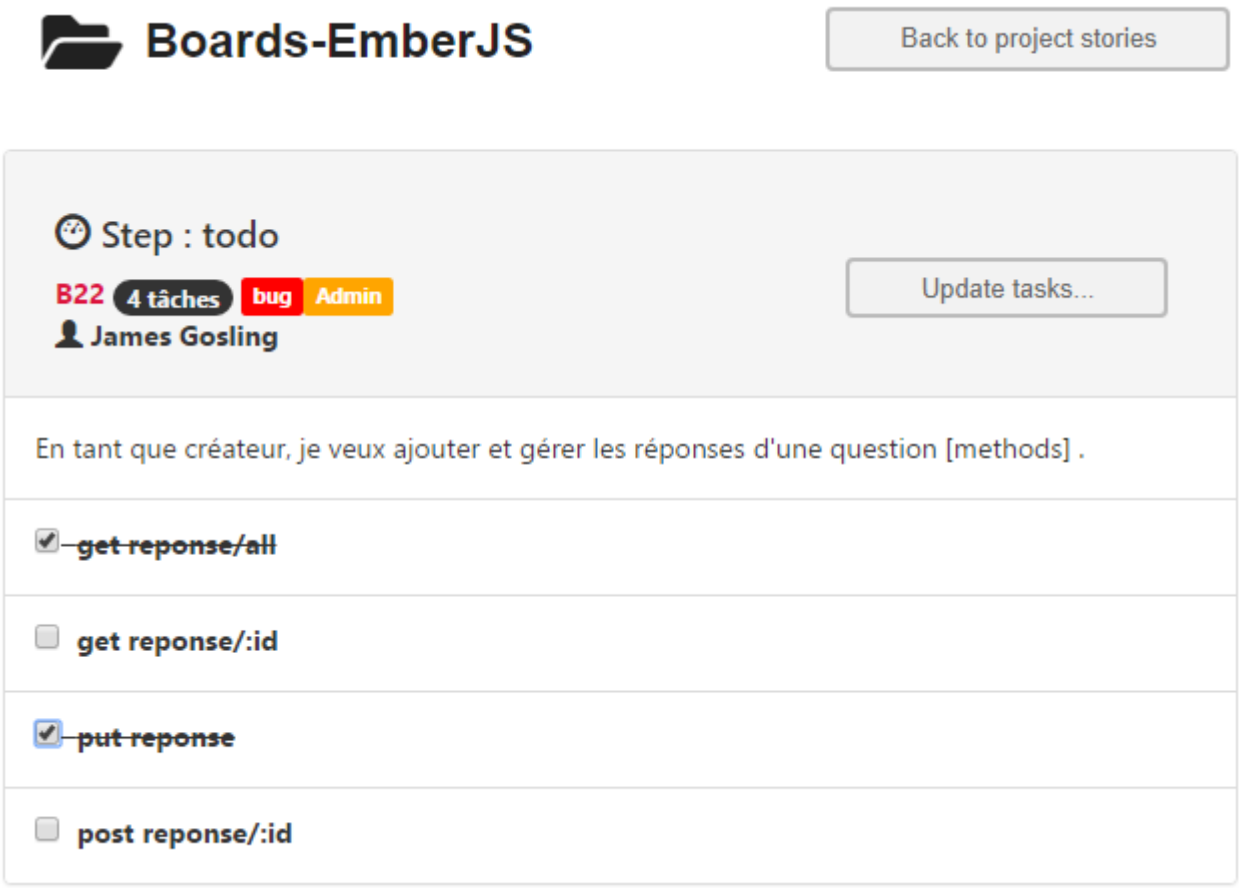
Exemple avec utilisation de javascript pour compléter l'url :

```
$js=$this->gui->getDeferred("project/settings","#settings-
elm",["attr"=>'js:data+"/5"']);
```

-- Route Stories/edit/{idStory}

//TODO 2.2

Affiche la **story** correspondant au **code** passé dans l'url.



Comportement de l'interface :

- Une task **done** est barrée
- Les cases à cocher permettent de faire passer une tâche de non réalisée (done=false) à réalisée (done=true) et inversement