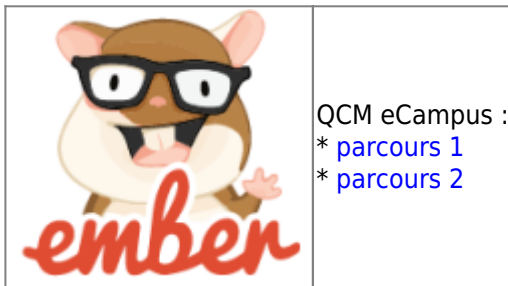


TD n°5



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

Objectifs

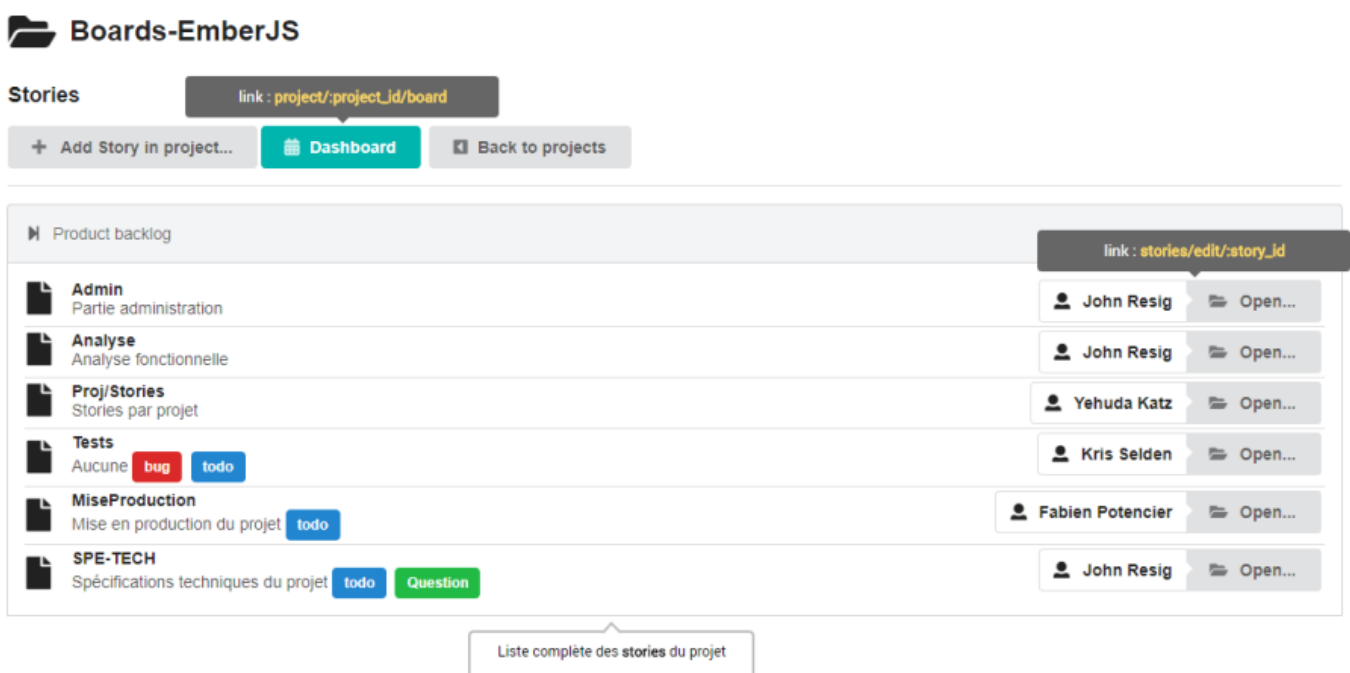
- Conception d'interfaces Web client riche
- Manipulations datas
- Utilisation composants

- Eléments à implémenter

- Route `project/:project_id/board`

//TODO 2.1

Cette route est déjà partiellement implémentée (`project/:project_id`):



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.

Boards-EmberJS

Stories

+ Add Story in project...
 Dashboard
 Back to projects
 Affiche la liste des stories non affectées à une step

Product backlog

- **MiseProduction**
 Mise en production du projet todo

 Fabien Potencier Open...
- **SPE-TECH**
 Spécifications techniques du projet todo Question

 John Resig Open...

Dashboard

Progress of the project stories

Boutons affichés si une story est sélectionnée

With selected story: **Admin**

Open ...
Move to step
✕ Remove

Todo Story active

- **Admin** John Resig 0%
- **Tests** bug todo Kris Selden 0%

In progress

- **Proj/Stories** Yehuda Katz 25%

Done

- **Analyse** John Resig 100%

Comportement/conception de l'interface

Steps

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

- le composant HTML ui-grid de semantic 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:project_id** est active, toutes les stories sont affichées dans le product backlog. Lorsque la route **project:project_id/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:story_id**
- Il est possible de déplacer les stories entre les steps par un drag and drop

Le tableau suivant pourra servir à élaborer la grille des steps :

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

Models

Ajouts de membres non persistants aux models :

Model	propriété	Rôle
project	boardVisible	booléen déterminant si board est visible
	backlog	computed property* des stories à afficher dans le backlog (* sur boardVisible et stories.@each.step)
story	active	booléen déterminant si la story est sélectionnée
	progress	computed property* donnant l'avancement de la story (% des tâches réalisées) (* sur tasks@each.done)

Component

Pour le drag and drop, on utilise les fonctionnalités HTML5, au travers de 2 composants (fortement inspirés de cet [article](#)) :

draggable-dropzone correspond à la zone de drop (step d'accueil)

```
import Component from '@ember/component';
import {set,get} from '@ember/object';

export default Component.extend({
  classNames      : [ 'draggableDropzone' ],
  classNameBindings : [ 'dragClass' ],
  dragClass       : 'deactivated',

  dragLeave(event) {
    event.preventDefault();
    set(this, 'dragClass', 'deactivated');
  },

  dragOver(event) {
    event.preventDefault();
    set(this, 'dragClass', 'activated');
  },

  drop(event) {
    var data = event.dataTransfer.getData('text/data');
    this.sendAction('dropped', data,get(this,'content'));
    set(this, 'dragClass', 'deactivated');
  }
});
```

draggable-item correspond à l'élément déplacé (drag) (story)

```
import Component from '@ember/component';
import {get} from '@ember/object';

export default Component.extend({
  classNameBindings: ['isActive'],
  isActive: Ember.computed('content.active', function() {
    let content=get(this,'content');
    if(get(content,'active'))
      return this.get('activeClass');
    return '';
  }),
  attributeBindings : [ 'draggable' ],
  draggable          : 'true',

  dragStart(event) {
    return event.dataTransfer.setData('text/data', get(this,
'content').get('id'));
  },
  doubleClick(){
    this.sendAction("onDbClick",get(this,'content'));
  },
  click(){
    this.sendAction("onClick",get(this,'content'));
  }
});
```

Exemple d'utilisation **draggable-dropzone**:

```
{{#draggable-dropzone content=step dropped="addToStep" class="column"}}
...
{{/draggable-dropzone}}
```

Attribut	Rôle
content	objet associé
dropped	action déclenchée sur le drop dont le prototype est (dragContent.id,dropZoneContent)

Exemple d'utilisation **draggable-dropzone**:

```
{{#draggable-item class="ui segment" activeClass="green inverted" content=story
onClick="activate"}}
...
{{/draggable-item}}
```

Attribut	Rôle
content	objet associé
activeClass	Classe utilisée si content.active==true
onClick	action déclenchée sur le click de l'élément

- Créer impérativement une **page d'accueil** permettant de naviguer entre les différents TODOs
- Alimenter la base de données avec des enregistrements réalistes en nombre suffisant (BDD à mettre dans le dossier **DB** de votre repository)
- Mettez en valeur votre projet et présentez son contenu dans le **README.md**

From:
<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**

Permanent link:
<http://slamwiki2.kobject.net/riclient/emberjs/td5>

Last update: **2019/08/31 14:21**

