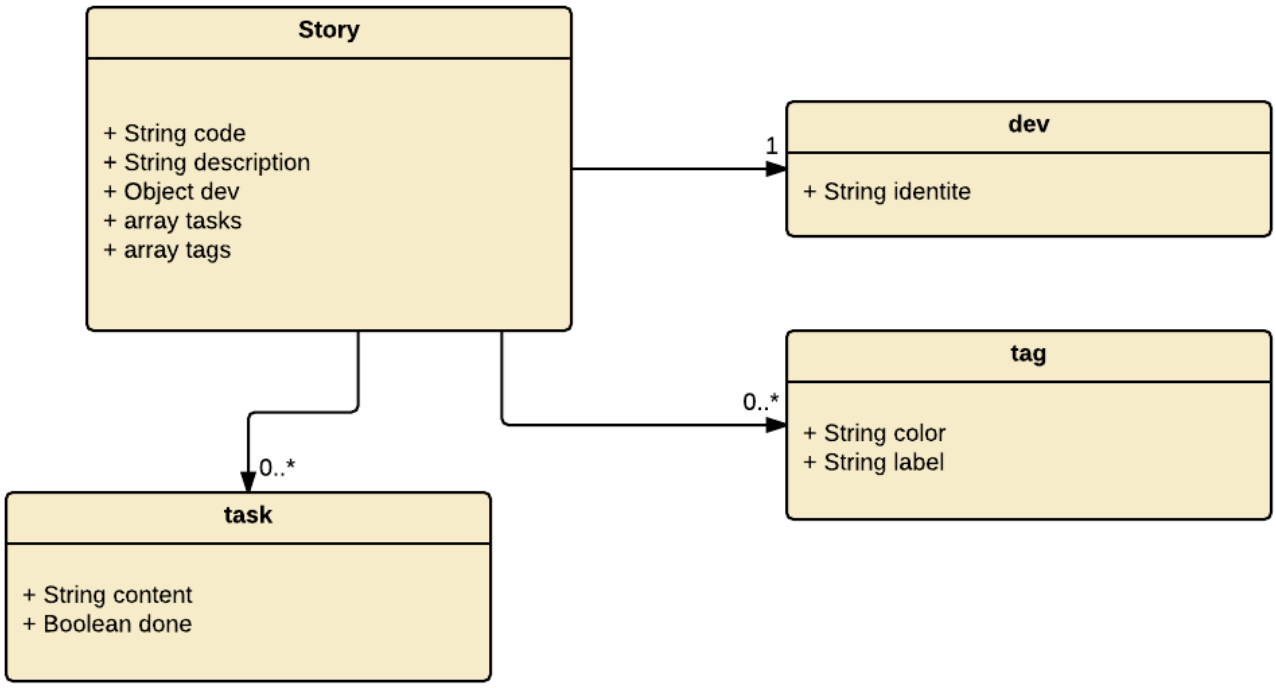

# Boards

## -- Contexte

Dans le cadre de l'évaluation du potentiel d'Angular pour les projets de votre entreprise, vous travaillez sur un projet Exemple permettant de gérer les User stories de projets Scrum.

En voici les principales caractéristiques :

- Chaque User story [**story**] a un code et un descriptif.
- Il est possible de lui apposer des tags [**tags**], composés d'une couleur et d'un label.
- Elle peut être affectée à un développeur[**dev**] (qui a juste une identité).
- Elle peut contenir une liste de tâches [**tasks**], à réaliser ou réalisées.




## -- Eléments à implémenter

- **Strict respect des normes, nommages et consignes** données ⇒ **1 point**
- Usage de **Browserify** pour créer un unique bundle ⇒ **1 point**

## -- Squelette de l'application, routage (3 points)

#### //TODO 2.1.1

Le dossier root de votre application devra être de la forme : **prenom.nom**

Créer la structure suivante :

| **Emplacement** | **Fichier** | **Rôle** |
|---|---|---|
| **/** | index.html | Fichier principal |
| **/js** | | Fichiers js |
| | app.js | Fichier principal de l'application (module) |
| | router.js | Fichier de routage |
| **/js/controllers** | | contrôleurs |
| | story.js | Définit le contrôleur **storyController** |
| | stories.js | Définit le contrôleur **storiesController** |
| **/js/directives** | | directives |
| **/js/services** | | services |
| **/templates** | | templates HTML |
| **/css** | | Feuille(s) de styles |

#### //TODO 2.1.2

Implémenter le routage suivant :

| **URL** | **Composante** | **Valeur** |
|---|---|---|
| **/** | Description | Affiche la liste des user stories |
| | template | templates/stories.html |
| | contrôleur | storiesController |
| | alias | storiesCtrl |
| **/story/:code** | Description | Affiche la user story correspondant à **code** |
| | template | templates/story.html |
| | contrôleur | storyController |
| | alias | storyCtrl |
| / | Route par défaut | |

## -- Service (2 points)

#### //TODO 2.2

Le service **dataService** simule la connexion à un web service ; il est défini de la façon suivante :

**stories**, **tags** et **devs** sont définis “en dur” dans l'équivalent de variables privées :

```
    var stories=[
                    {"code":"B22","description":"En tant que créateur, je veux
ajouter et gérer les réponses d'une question [methods] .",
"tags":[{"color":"red","label":"bug"},{"color":"orange","label":"Admin"}],
                        "dev":{"identite":"James Gosling"},
                        "tasks":[{"content":"get
reponse/all","done":false},{"content":"get
```

```
reponse/:id","done":false},{"content":"put reponse","done":false},{"content":" post
reponse/:id","done":false}]
                    },
                    {"code":"E120","description":"En tant que créateur, je veux
créer / Modifier des quiz [methods]",
                        "tasks":[{"content":"get questionnaire/:id","done":false}],
                        "tags":[]
                    },
                    {"code":"E140","description":"En tant que créateur, je souhaite
gérer les utilisateurs [methods]",
                        "tasks":[{"content":"get
user/all","done":true},{"content":"get user/:id","done":true},{"content":"put
user","done":true}],
                        "dev":{"identite":"Rod Johnson"},
                        "tags":[{"color":"orange","label":"Admin"}]
                    }
    ];
    var
tags=[{"color":"red","label":"bug"},{"color":"#cc317c","label":"question"},{"color"
:"#159818","label":"help
wanted"},{"color":"#cccccc","label":"duplicate"},{"color":"#0052cc","label":"todo"}
];
    var devs=[{"identite":"James Gosling"},{"identite":"Rod
Johnson"},{"identite":"Linus Torvalds"}];
```

| **Service** | **datas.js** (dataService) |
|---|---|
| **Variables privées** | stories<br>Tableau des user stories |
| | tags<br>Tableau des tags disponibles |
| | devs<br>Tableau des développeurs |
| **Méthodes publiques** | getStories()<br>retourne **stories** |
| | getTags()<br>retourne **tags** |
| | getDevs()<br>Retourne **devs** |
| | getStory(code)<br>Retourne la user story correspondant au **code** donné, **null** si le code est inexistant |
| | setStoryAvancement(story)<br>ajoute et valorise le membre **avancement** de la **story** passée en paramétre (avancement=nb tâches réalisées/ nb tâches).<br>Une tâche est réalisée si son membre done=true.<br>A l'issue du passage **story.avancement** est existant et correctement valorisé |

Créez dataService, implémentez les méthodes publiques.
Il faudra ensuite injecter **dataService** aux 2 contrôleurs **storiesController** et **StoryController**

## -- Url /stories (3 points)

#### //TODO 2.3

Affiche la liste des user stories.


Créer le template **/templates/stories.html**, associé au contrôleur **/js/controllers/stories.js**

| **Controller** | **stories.js** (storiesController) |
|---|---|
| **Variables privées** | stories<br>Tableau des user stories |
| **Méthodes publiques** | goto(story)<br>Accède à l'url **story/:code** en utilisant le service **$location** |

## -- Directive storyHeader (2 points)

### //TODO 2.4


On souhaite intégrer l'entête de chaque story affichée à l'url /stories dans une directive, pour la réutiliser plus facilement dans la page suivante, pour présenter le détail de chaque story.

La directive **storyHeader**, accessible en tant qu'élément, ou attribut, permettra d'afficher le contenu suivant d'une story. Il sera nécessaire de lui passer la variable story (qu'elle est en charge d'afficher) dans son **scope**.

- Déclarer la directive **storyHeader**,
- intégrer la dans l'application,
- créer le template correspondant,
- utilisez la à l'url **/stories**

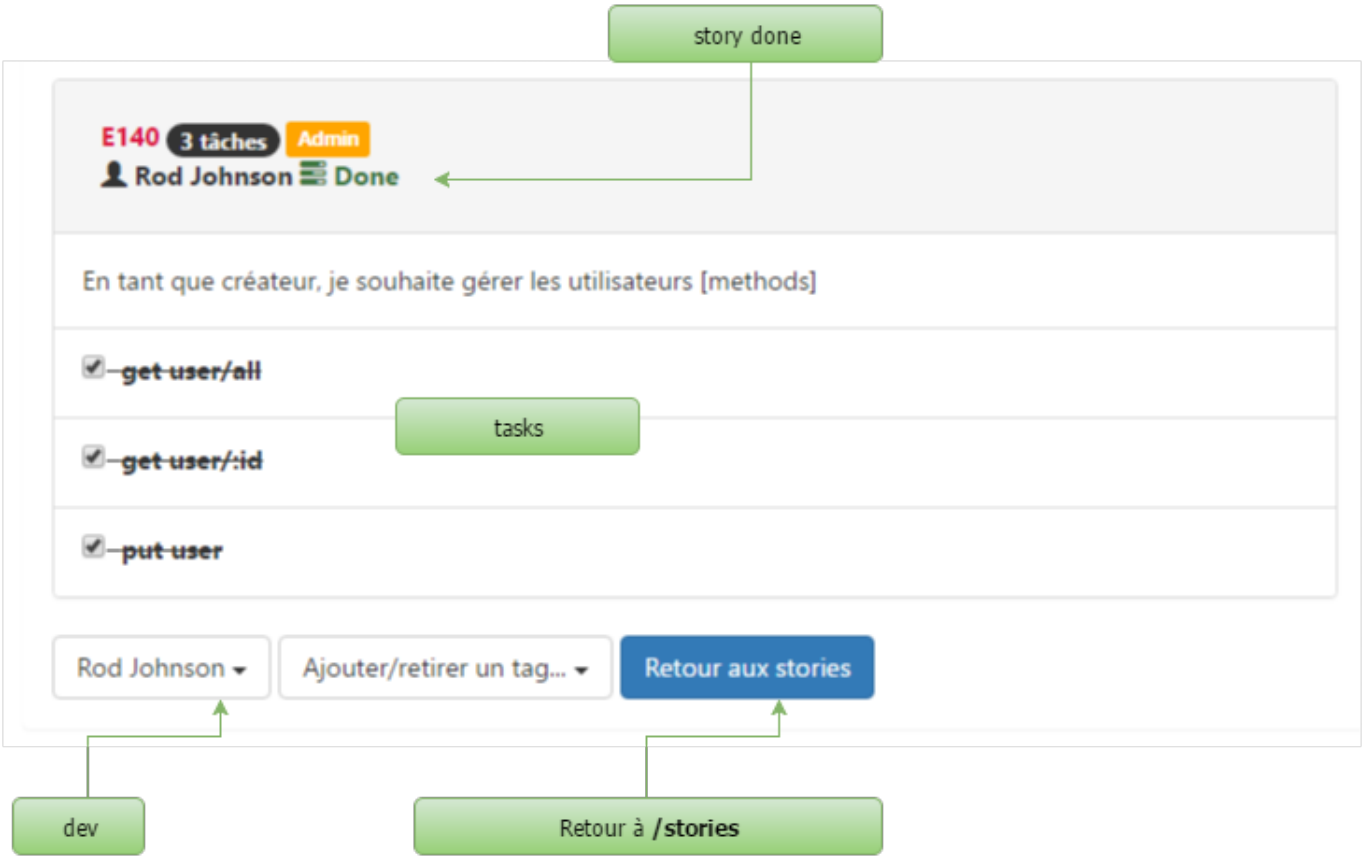
On respectera la structure suivante :

| **Emplacement** | **Fichier** | **Rôle** |
|---|---|---|
| **/js/directives** | | Dossier définissant les directives |
| | storyHeader.js | Fichier définissant la directive storyHeader |
| **/js/directives/templates** | | Fichiers templates |
| | storyHeader.html | Fichier template de la directive storyHeader |

## -- Url /story/:code (5 points)

### //TODO 2.5

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



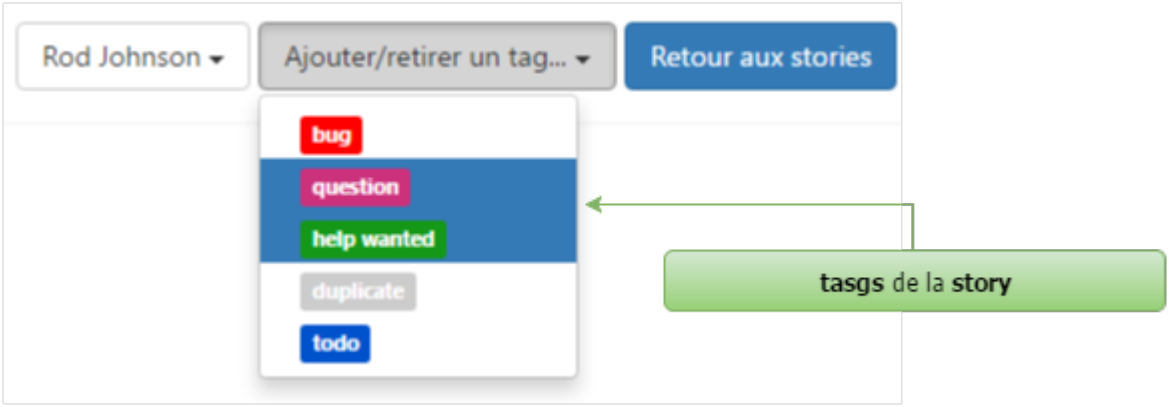

**Comportement de l'interface :**

- 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

**Menu Dev :**


- Le dev actif est sélectionné (classe bs **active**)
- Le clic sur un autre dev change l'affectation
- L'élément **Retirer l'affectation** passe le dev de la story à **null**

**Menu Tags :**




- Les tags présents dans la story sont sélectionnés (classe bs **active**)
- Le clic sur un tag l'ajoute ou le retire à la liste des tags de la story

Créer le template **/templates/story.html**, associé au contrôleur **/js/controllers/story.js**

| **Controller** | **story.js** (storyController) |
|---|---|
| **Variables privées** | story<br>user story à afficher |
| | devs<br>Tableau de tous les développeurs |
| | tags<br>Tableau de tous les tags |
| **Méthodes publiques** | toggleDone(task)<br>Bascule de vrai à faux et inversement le membre **done** de la tâche **task** passée en paramètre |
| | assignDev(dev)<br>Assigne le développeur **dev** à la **story** |
| | indexOfTag(tag)<br>Retourne l'index du **tag** passé en paramètre dans la liste des **tags** de la **story** (-1 s'il n'est pas trouvé) |
| | toggleTag(tag)<br>Ajoute ou retire le **tag** passé en paramètre de la liste des **tags** de la story |

## -- A poursuivre...

#### //TODO 2.6

Fonctionnalités supplémentaires à implémenter :

**Dans la page story/:code :**

- Permettre l'ajout/modification/suppression de tasks (2 points)
- Permettre l'ajout/modification/suppression de nouveaux tags (2 points)

**Dans la page story :**

- Filter les stories à afficher par sélection de devs, par la présence de tags, par la mention **done** (2 points)