Chapitre 1

Introduction

Chapitre 2

Le contexte du travail

2.1 Le cahier des charges ?

2.2 L'architecture de la plateforme cible

2.3 Le framework Angularjs ?

AngularJS est un framework écrit en javascript par Google libre et open-source qui permet d'améliorer, au même titre que JQUERY, la syntaxe de javascript ainsi que la productivité du développeur. Il étend le HTML pour le rendre dynamique, et permet de développer ses propres balises et attributs HTML. C'est un framework qui se veut extensible et qui pousse vers un développement structuré, en couches, le but n'étant pas d'ajouter de simples animations au DOM, mais bien d'apporter un aspect applicatif au front-end.

2.3.1 Concept

Angular est construit autour de concepts et de bonnes pratiques incontournables dans le monde du développement web.

- Architecture `Modèle-Vue-Contrôller` (MVC) : si vous connaissez le développement, vous avez sûrement entendu parler de ce type d'architecture incontournable qui consiste à avoir une stricte séparation entre les données (Modèle), la présentation des données (Vue), et les actions que l'on peut effectuer sur ces données (Contrôleur).
- Le `data-binding bidirectionnel` est un moyen de lier la partie vue à la partie logique. En d'autres termes, grâce à cela, les éléments de votre code HTML seront liés à votre contrôleur JavaScript est rendu possible grâce au scope. Le scope est le "liant" d'une application AngularJS, c'est lui qui contient les variables et fonctions qui font la liaison entre vues et contrôleurs (ou autres). Il permet donc aux données de pouvoir être mises à jour par les vues et par le modèle.
- `L'injection de dépendances` permet de charger certaines parties de l'application seulement quand c'est nécessaire.
- La manipulation du DOM au moyen de directives : AngularJS dispose d'un grand nombre de directives permettant de manipuler le DOM et offre aussi la possibilité aux dévelopeurs d'écrire leurs propres directives.
- Le Routage, le module de routage de Angularjs est l'un des composants clés lui permettant de créer des applications `Single Page Application` (SPA). En effet, si le développeur veut pouvoir naviguer sur plusieurs pages et en même temps en faire une application monopage, il peut utiliser le module `ui-router`.

Angularjs permet de réaliser des applications web en mode Single Page Application. C'est à dire une seule page qui ne se recharge pas. L'idée de base est d'augmenter le langage HTML pour permettre la représentation des données métiers, qui sont elles traitées et gérées avec le langage Javascript. Depuis son avennement, plusieurs projets communautaires ont vu le jour dans le but de permettre la ficilitation de son utilisation et d'étendre ses fonctionnalités. C'est le cas de `angular-UI` qui a pour but de fournir des outils classiques dans le développement web en reprennant leur implémentation sans JQUERY. Restangular est un projet communautaire permettant d'utiliser une REST API dans AngularJS sans avoir à faire soit même l'ensemble des opérations HTTP (GET/POST/PUT/DELETE). Ce projet est très utilisé et très pratique. Il permet d'éviter la création de services pour manipuler ce genre d'API.

2.4 Les ressources REST

2.5 Le contrôle d'accès en utilisant des rôles (RBAC)

Chapitre 3

La Conception de notre générateur de module

3.1 Le concept général

3.1.1 Définition d'un module

Un module est une application métier [1] basée sur l'architecture client-serveur. Nous nous référons au terme module pour indiquer le caractère modulaire que peut prendre une application. Par exemple,la gestion des ressources humaines est un assemblage de plusieurs sous-applications telles que la gestion des employés, la gestion des carrières, la gestion de la paie et la gestion des embaûches [2].

L'architecture client-serveur, implique qu'un module est composé d'une application *client* et d'une application *serveur*. L'application client est une interface graphique [3] permettant à aux utilisateur d'interagir avec un système d'information. L'application serveur est celle qui gère les données métiers dans un système d'information. La gestion des données est faite essentiellement via une base de données et les processus métiers.

3.1.2 Le portail applicatif des modules

C'est l'interface principale [4] permettant de présenter les différentes fonctionnalités des modules. Comme le montre la figure **??**, elle est divisée en six parties à savoir :

– La barre d'entête (cf. le chiffre 1 sur la figure **??**). Cette partie indique le nom du module courant, la liste des tâches à réaliser et la liste des modules.

1. A expliquer
2. donner la liste complète si possible
3. spécifier pourquoi c'est une interface graphique
4. parler du PAE

– Le bloc de menus (cf. le chiffre 2 sur la figure **??**). Il présente les différents éléments du menu d'un module.
– Le bloc des modules liées (cf. le chiffre 3 sur la figure **??**). Présente la liste des modules en lien avec le module courant.
– L'espace de travail (cf. le chiffre 4 sur la figure **??**).
– La barre d'actions (cf. le chiffre 5 sur la figure **??**).

L'nterface est basée sur la notion d'applications monopages [5]. De ce fait, les parties une, deux, trois et cinq sont statiques alors que la partie 4 est dynamique. Les éléments des parties statiques changent avec l'installation d'un nouveau module.

Actions versus menu. Les éléments du menu permettent de naviguer à travers les différentes vues d'un module alors que les actions permettent de faire des opérations sur les éléments de la vue courante. D'un point de vue technique, la différence se trouve au niveau des composants avec lesquels ils sont implmentés : les menus sont implémentés sous forme de liens et les actions boutons.

Liste des modules versus modules liés La liste des modules de la barre d'entête représente l'ensemble des modules installés sur le système tandisque la liste des modules liés représente les dépendances entre différents modules. Les modules de gestion de stocks et d'achats sont liés ou dépendent du module de la gestion des articles. Dans ce cas de figure, le module gestion des articles aura dans sa liste de modules liés les modules stocks et achats. L'intérêt principal des modules liés est d'éviter à l'utilisateur, d'aller chercher un module parmi une liste de module qui peut vite être conséquent.

L'installation d'un module. Elle consiste, pour un module donné, à fournir les éléments de parties statiques. ces dernières sont ensuite positionnées dans leur bloc respectif.

3.1.3 Le principe générale

L'idée du générateur est d'utiliser un environnement graphique permettant de spécifier les différents éléments nécessaires à l'implémentation des applications client et serveur, en vue d'augmenter la productivité générale [6] d'un développeur. En effet, une fois les éléments spécifiés, nous passons à la génération automatique des codes des applications client serveur. Ceci permettra aux développeurs de se focaliser sur la conception de leurs modules.

5. Aussi nommée Single Page Application (SPA)
6. listing des productivités

Nous regroupons les éléments du module en deux catégories : ceux *côté client* et ceux *côté serveur*. Les éléments côté client sont ceux qui sont nécessaires pour la mise en place de l'interface principale. Ils permettent de définir les différentes pages ou vues html ainsi que les services qui permettent de recupérer les données du serveur. Les éléments côté serveur permettent de définir les bases de données, les processus métiers et ressources permettant d'interagir avec le client.

Productivité.

3.2 La présentation des éléments graphiques côté client

3.2.1 Overview

Présentation de la vue générale

La vue générale se concentre sur les éléments côté client d'un module. Elle est l'étape principale dans le processus de réalisation d'un module. Aussi simple qu'elle parait, elle rassemble tous les outils nécessaires pour concevoir chacun des éléments de la partie cliente d'un module. Elle se résume à l'interface de la figure suivante. Sous les onglets `components` et `treeview`, nous disposons respectivement de la palette de composants html et de leur organisation sur la vue en forme d'arborescence. sous les onglets `variables` et `functions`, on définit les variables et les fonctions qui sont utilisées pour écrire le contrôleur de la vue. Sous les onglets `actions` et `configs` on définit respectivement les actions du module et les paramètres de configuration de la vue.

Les composants

Ils constituent un outil graphique permettant de manipuler les éléments qui entrent dans l'élaboration d'une interface web. Les manipulations sont basées sur un certains nombres de propriétés que nous leur avons associé afin de controller leur mise en forme, leurs attributs html ainsi que les données qu'ils manipulent.

Nous avons resencé les composants les plus couremment utilisés dans les applications web et nous les avons regroupé en trois grandes catégories :

- *Les composants simples* ; ce sont des éléments HTML dont la définition ne nécéssite pas l'imbrication d'un autre composant. exemple :
- *Les composants conteneurs* Ce sont des éléments qui peuvent contenir des composants de n'importe quelle catégorie, par exemple l'élément html `form` permet de regrouper les composants simples tels que les `input` et des `button` pour la réalisation d'un formulaire.

– Les composants widgets Ce sont des éléments non standards que nous avons entièrement écris, en proposant leur design et leur conception.

L'arbre des composants

Le concept de l'arbre des composants est d'augmenter la productivité des dévéloppeurs en limitant le perte de temps due aux changements récurents dans le développement des interfaces [7].

L'arbre des composants est un outil permettant de représenter les composants d'une interface. Ses composants sont organisés sous forme d'arbre afin de mieux schématiser la hiérarchie induite par les composants conteneurs. La figure **??** représente l'arbre des composants de l'interface de la figure **??**.

Notre concept se décline via l'interface de la figure **??** dans laquelle on constate qu'un certains nombres d'actions sont associées à chaque composant. En effet pour permettre la réutilisation de codes nous proposons les cinq actions suivantes :

– Monter - descendre : elles permettent les déplacements de composants vers le haut ou vers le bas, dans le même conteneur. Ceci permet de réorganiser les composants d'un même conteneur.

– couper : elle permet les déplacements de composants d'un conteneur à un autre. Ceci permet la réorganisation globale des composants.

– copier : permet de cloner un composant, soit dans un même conteneur ou non. Ceci permet de générer facilement une interface avec des composants semblables.

– Supprimer : elle permet d'enlever un composant de l'arborescence.

Les fonctions

Le concept ici est d'améliorer la productivité du dévloppeur en :

– évitant les *erreurs d'écriture* dans les noms des fonctions dans les vues et dans les contrôleurs.

– proposant une *vue d'ensemble* du travail à réaliser (isolation ?).

Le concept se décline via l'inteface de la figure **??**

Erreur d'écriture Le lien « generate functions », génère les noms des fonctions, à partir de l'arbre des composants. Les composants tels que button, ... et ... font appel à des fonctions qu'ils définissent dans leurs propriétés. à partir des propriétés des composants cités plus haut, nous extrayons le nom des variables à déclarer.

7. explication détaillée

Vue d'ensemble Nous pouvons dinstinguer parmi les variables à définir, celles qui sont dejà définies ou non. La définition d'une variable se fait en l'isolant des autres variables. Ceci permet au développeur de se focaliser sur le reste des variables et surtout de ne pas perdre de temps à chercher une information dans les codes qui peuvent rapideent devenir illisibles.

Les variables

Le concept ici est d'améliorer la productivité du dévloppeur en :

- évitant les *erreurs d'écriture* dans les noms des fonctions dans les vues et dans les contrôleurs.
- proposant une *vue d'ensemble* du travail à réaliser.
- simplifiant la déclaration des variables.

Le concept se décline via l'inteface de la figure **??**

Erreur d'écriture. Le lien « generate vars », génère les noms des variables, à partir de l'arbre des composants. Les composants manipulent des variables. à partir des propriétés, nous extrayons le nom des variables à implémenter.

Vue d'ensemble. Nous pouvons dinstinguer parmi les fonctions à implémenter, celles qui sont dejà définies ou non. La définition d'une fonction se fait en l'isolant [8] des autres fonctions. Ceci permet au développeur de se focaliser sur le reste des fonctions et surtout de ne pas perdre de temps à chercher une information dans les codes qui peuvent rapideent devenir illisibles.

Déclaration. Les variables peuvent être de type *resolve*, *async* et *cache*. les variables de type *resolve* sont celles dont les valeurs sont présentes avant le chargement de la page qui les utilise. les variables de type *async* sont celles dont les valeurs proviennent d'un serveur de manière asynchrone ; dans ce cas, le service permettant d'obtenir la donnée doit être préciser. Les variables de type *cache* sont celles dont les valeurs sont mises ou proviennent d'un cache lorsque l'attribut cache a pour valeur `to` ou `from`.

Les actions

Ils constituent un outil graphique permettant de spécifier les actions associées à une vue donnée.

représentent une partie des fonctionnalités du module faisables à partir de la vue encours d'édition. Pour ajouter une fonction, il faut :

- Préciser le nom de l'action ;

8. collapsible

– Préciser le rôle de l'utilisateur qui peut y avoir accès ;
– Préciser l'icône de l'action, c'est sur cette dernière qu'il faut cliquer pour exécuter l'action ;
– Préciser les dépendances ; awesome file input
– Enfin écrire le corps de la fonction de l'action

La configuration

Le concept ici est d'améliorer la productivité du développeur en simplifiant la mise en place des menus, des droits et de la navigation. Ce concept se décline via l'interface de la figure suivante **??**.

Navigation. Elle consiste à construire l'état/route auquel(le) la vue est associée tout en précisant le nom de l'état, l'url associé et le chemin static ou dynamic du template. Ainsi, en parcourant la liste des vues, on peu les connaître les différents états du module.

Menus. La case à cocher « module menu »permet de spécifier si une vue a une entrée ou non dans la liste des menus d'un module. De ce fait, il est possible en parcourant la liste des vues de la figure **??**, d'obtenir les éléments du menu. On pourra non seulement identifier les icônes associées aux menus mais aussi changer l'ordre d'apparition des éléments dans le menu par un simple déplacement dans la liste des vues.

Droits d'accès

3.3 La présentation des éléments graphiques côté serveur

Les resources

Le concept ici est de faciliter l'implémentation des APIs REST. Ceci se décline dans l'interface de la figure **??**. Elle est divisée en trois parties à savoir : la liste des ressources, les détails d'une ressource et les paramètres des ressources.

La liste des ressources . Elle permet d'avoir une vue d'ensemble sur les ressources d'un module. Son organisation en arborescence facilite la détection d'incohérences liées aux URIs

Les Détails d'une ressource. Au delà de l'URI visible dans l'arborescence, on distingue dans cette partie les informations liées à une ressource. Ces dernières peuvent être mise à jour en cliquant sur l'action `update`.

Les paramètres d'une ressource Cette partie donne une vue d'ensemble des `méthodes` d'une ressources et permet aussi d'en créer de nouvelles. Une méthode est la définition d'une opération sur la ressource.

Les opérations

Les opérations sont étroitement liées aux ressources

Les données Externes

Chapitre 4

L'implémentation du générateur

4.1 Les élément côté client

4.1.1 L'arbre des composants

4.1.2 Le générateur de code

Après la conception d'un module sous le format json, vient l'étape de sa génération. la génération des différents codes du module consiste à matérialiser les fichiers des parties client et serveur du module concerné.

génération des vues. Avant de procéder à la génération des codes html, nous avons préalablement défini un formalisme pour chaque élément html.on a donc :

- `input`

```
<div  style="style"  class="class">
  <label for="input1">input1</label>
  <input class="␣form-control" name="input1" id="input1" type="text" ng-
</div>
```

- `radio`

```
<div  >
  <label for="radio2">radio2</label>
    <div  style="style"  class="class">
      <label class="␣radio-inline">
        <input id="radio2" type="radio" value="toto" ng-model="genre"> t
      </label>
      <label class="␣radio-inline">
        <input id="radio2" type="radio" value="tata" ng-model="genre"> t
      </label>
    </div>
```

```
</div>
```

- textarea

```
<div class="form-group␣"  style="style"  class="class">
  <label for="textarea4">textarea</label>
  <textarea id="textarea4" class="form-control" name="textarea4" ng-mode
</div>
```

- select

```
<div  class="form-group">
  <select name="select5" id="select5" ng-model="null" class="form-contro
    <option ng-repeat='item in items' value="{{item.rvalue}}"> {{item.dv
    </option>
  </select>
</div>
```

- label

```
<label style="css␣style" > Label </label>

<label> {{ label }} </label>
```

4.1.3 Les vues générées

4.1.4 Les configs

4.2 Les éléments côté serveur

4.2.1 La représentation d'une ressource (basée sur le WADL)

4.2.2 La liste des ressources

Chapitre 5

conclusion