
Dossier de Projet

MEETING-APPS

R-numérique

Desrues Lucas

Diplôme Développeur Logiciel IMIE

Table des matières

Remerciements	p.3
Compétences mise en oeuvres	p.4
Projet Meeting-Apps	p.5
L'agence	p.6
Cahier des charges	p.7
Acteurs du projet	p.7
Contexte	p.7
Enjeu	p.7
Objectif:	p.7
Fonctionnalités attendues	p.8
Contraintes	p.8
Planification du projet	p.9
Wbs	p.9
Diagramme de Gantt	p.10
Spécifications fonctionnelles	p.11
Wireframe	p.11, p.12

Remerciements

Mon tuteur Mr Nicolas Ramel, qui est le patron de R-numérique pour sa confiance.

Mon collègue Thomas pour sa patience indéfectible et les longues minutes passées sur le débogage.

Mme Eliza Bourbousson pour m'avoir donné la chance de participer à cette formation.

Mon formateur référent Gabriel Block pour avoir su nous partager son savoir avec pédagogie et pour ses encouragements.

Compétences mises en oeuvre

Maquette une Application :

- Utiliser un outil de maquettage (p.12,p.13)
- Connaissance du formalisme des cas d'utilisation et du diagramme d'état de la notation du langage modélisation unifié UML (p.11)

Concevoir une base de données :

- Connaissance du Modèle relationnel (p.14)
- Construire le schéma entités association (p.14)

Mettre en place une base de données :

- Connaissance du langage de requête structurée SQL (p.15)
- Connaissance des différents types de codage des données (p.15)

Développer une interface utilisateur :

- Connaissance des concepts de la programmation objet (p.14 à p.)
- Développer dans un langage objet (p.14 à p.)
- Utiliser les normes de codage du langage et auto-documenter le code au moyen du nommage (p.18, p.19)
- Utiliser les bibliothèques de composants graphiques (p.27, p.30)

Développer des composants d'accès aux données :

- Connaissance du langage de requête structurée SQL
- Coder les accès aux données, la consultation, la création et la mis à jour, à partir de requêtes natives ou de procédures stockées

Développer des pages web en lien avec une base de donnée :

- Connaissance des langages du développement web, tel que langage de balise, feuilles de styles et langage de script client
- Connaissances des composants serveurs, pages web dynamiques
- Développer la partie dynamique de l'application avec de composants serveurs
- Utiliser un cadre (framework) de persistance de données
- Utiliser un jeu de tests de l'application web en priorisant les tests ou en appliquant une stratégie de tests

Utiliser l'anglais dans son activité professionnelle en informatique

- Identifier les différents types de documents techniques et leur structure
- Lire et exploiter les différents documents techniques

Project Meeting-Apps

« Meeting-Apps » is a module developed for « Ze-Apps »

The project « Ze-Apps » is an « ERP » (Environment Relation Partner).

The objective is to have an open-source core containing the main modules usually found in a classic « ERP » and to improve it at will with customs modules developed by the community (either free or pay to use).

The economic plan is similar to « Prestashop » with pluggable contents.

The advantage for the clients to have a software that fits their needs perfectly.

« Meeting-Apps » is one of those modules.

It simplifies note taking, helps transform tasks into action items, and enables users to include colleagues so projects are not limited to one person's device.

This module automates note-taking, structures these notes, and then synchronizes the result into a collaborative platform. It allows its users to analyze activity based on multiple criteria like the type of task or the meeting they are attending.

It also allows you to track the advancement of a project through a progress bar that updates automatically whenever you have validated a task.

The main goal is to save time on taking notes and planning during appointments or meetings and to share more easily with concerned people without sending mails or meeting reports.

L'agence



Qui sont ils ?

La société R-Numérique est une agence web proposant ses services depuis maintenant 10 ans.

Le personnel se compose d'un développeur et du patron de l'entreprise qui est responsable du développement commercial.

Que font ils ?

Des sites internet, développement de bases de données, ERP, CRM, référencement, publication numérique, banque d'images, Emailing, Hébergement de sites internet et d'applications, Gestion de serveurs.

Ils proposent une expertise complète du projet et adaptent le développement au besoin du client.

Pour qui ?

Pour toutes les entreprises quelque soit leurs tailles ou leurs budgets.

Cahier des charges

Acteurs du projet

MOA (Maitrise d'ouvrage)

- Lucas Desrues
- Nicolas Ramel
- Thomas Sauques

MOE (Maitrise d'oeuvre)

- Lucas Desrues

Contexte

Afin de faciliter les prises de notes lors de réunions, il semble judicieux d'agrémenter notre ERP « ZE-APPS» avec un module répondant à ce besoin. En effet , la prise de notes lors de rendez-vous ou réunion est une perte de temps considérable. L'objectif est de faciliter la prise de notes et les actions qui en découlent pour gagner du temps comparé à la prise de note standard sur papier.

Enjeu

Gagner 15 minutes à chaque réunions ou rendez-vous avec une prise de notes plus rapide

Objectifs

Augmenter l'efficacité des prises de notes en réunion et permettre un tri naturel des actions en découlant.

Fonctionnalités attendues

Les utilisateurs peuvent:

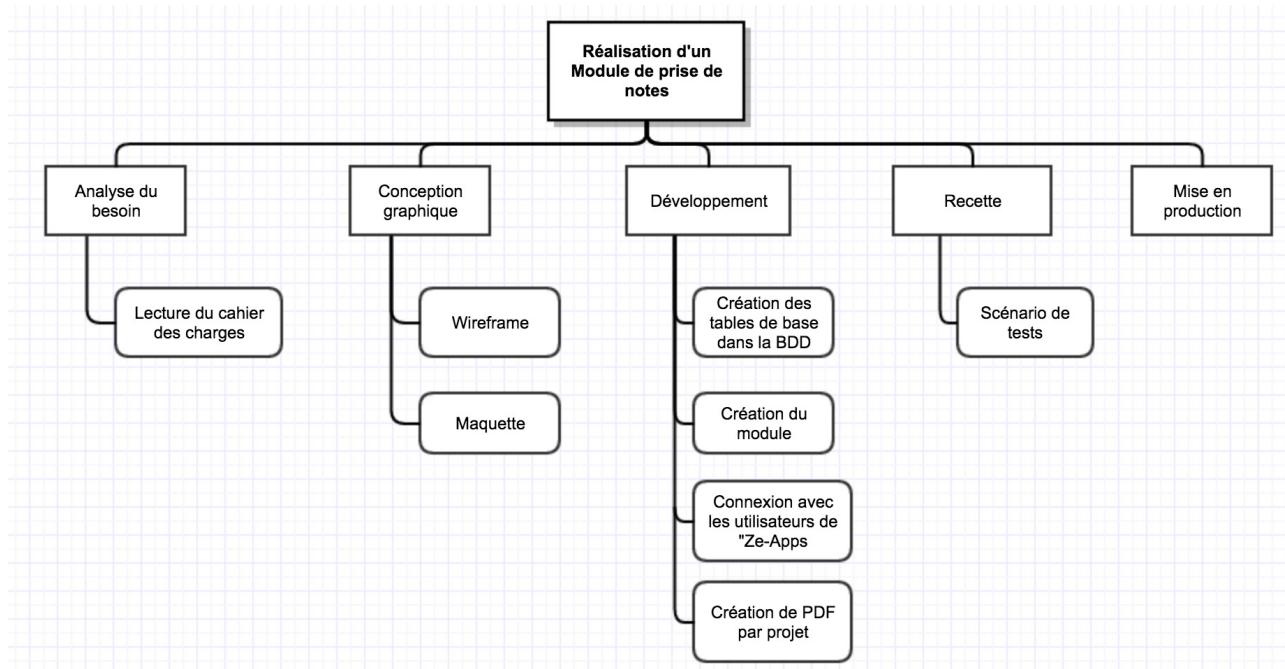
- Créer un Projet
- Créer une ou plusieurs Réunions propre à ce projet (Réunion, Rendez-vous, ...)
- Créer un ou plusieurs Ordre du jour à ces Réunions (Ordre du jour de la réunion)
- Créer des Notes pour les Ordre du jour ou directement pour la réunion, agrémentées d'un texte décrivant l'action :
 - Tache
 - Date d'échéance
 - Remarque
 - Appel
 - Réunion
 - Document
 - Question
 - Idée
 - Client
 - Fichier
 - Mail
 - Risque
 - Décision
- Attribuer des taches à d'autres utilisateurs, les utilisateurs en questions doivent avoir une notification les en informant.
- Les Notes doivent pouvoir être changées de positions pour une meilleure classification.

Contraintes

- Le module doit être intégré dans l'ERP « Ze-Apps »
- Il doit utiliser la base Utilisateurs de « Ze-Apps »
- Utiliser du JavaScript en front avec AngularJS et du PHP du coté serveur
- « Ze-Apps » étant développé avec le framework CodeIgniter (Modifié pour la gestion de module « Ze-apps »), Meeting-Apps devra également utiliser CodeIgniter
- Meeting-Apps doit être multi-langues (Français et Anglais)
- Les Actions doivent pouvoir être attribuées aux Projets, Ordre du jour et Participants avec un système de « Drag and Drop »

Planification du Projet

WBS



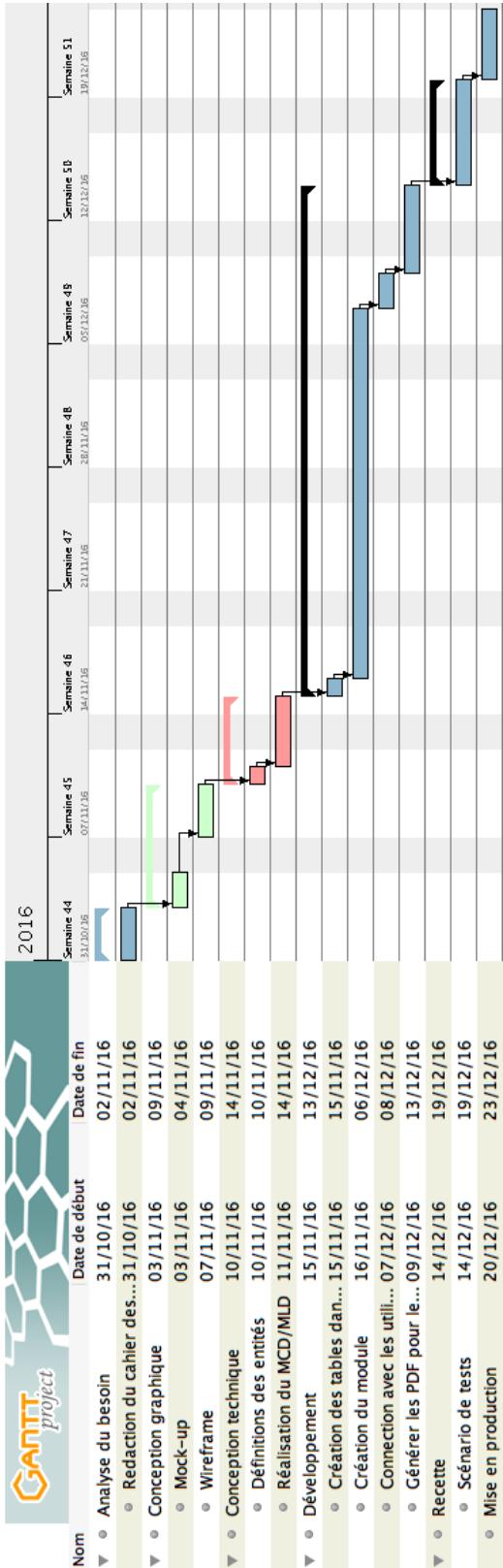
Ci-dessus le « WBS »(Work Breakdown Structure)

On voit ici le découpage de toutes les actions de ce projet en parties distinctes.

Le fait de bien distinguer l'architecture du projet a permis une bien meilleure organisation pour le travail à accomplir

Il a également permis de réaliser un diagramme de Gantt pour estimer le temps de travail pour chaque parties.

DIAGRAMME DE GANTT



Ici, le diagramme de Gantt où est estimé le temps nécessaire pour chaque taches précédemment décrites dans le « WBS ».

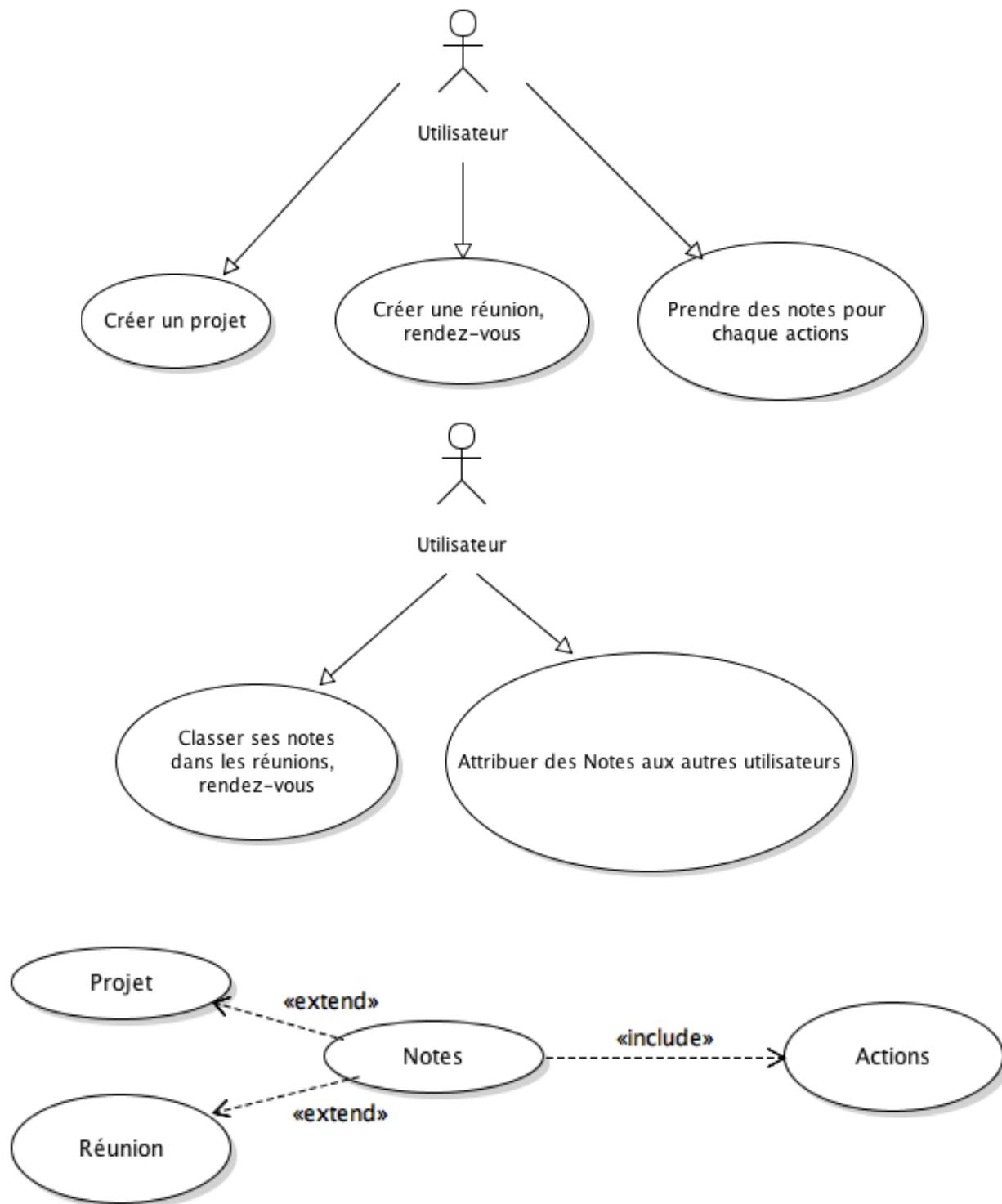
On voit maintenant l'ordre des taches à réaliser.

(On ne peut pas voir le chemin critique étant donné que l'ensemble du projet est réalisé par une unique personne).

L'organisation aurait été différente s'il y aurait eu plusieurs personnes affectées à ce projet.

Exemple: La conception technique et la conception graphique auraient pu démarrées en même temps.

Spécifications Fonctionnelles



Ci-dessus les « use case ».

Il permettent une compréhension des fonctionnalités du module pour toutes personnes extérieures au développement du projet.

WIREFRAME

The wireframe shows the main interface of the application. At the top, there is a header with the logo "Zeapps", a "Menu" button, a search icon, and a user profile section with a bell icon and "Nom d'utilisateur". Below the header is the title "Liste des Projets" and a "Nouveau Projet" button with a plus sign. The main content area displays a list of five projects in cards:

Project Name	Status	Progress (%)	Actions
Ze-apps	Statut	100%	Editer Supprimer
QuiltMania	Statut	100%	Editer Supprimer
La boite noire	Statut	75%	Editer Supprimer
Square-enix	Statut	50%	Editer Supprimer
Riot	Statut	0%	Editer Supprimer

Ci-dessus la page d'accueil de l'application.

Elle permet d'ajouter de nouveaux projets et de visualiser les projets déjà existant.

On voit également une barre de progression permettant de voir l'avancement du projet suite à la cloture des tâches validées.



Sujets

- Ze-Apps. 1er sujet
- Ze-Apps. 2eme sujet
- Ze-Apps. 3eme sujet
- Ze-Apps. 4eme sujet

Ze-Apps

	Ze-Apps. 1er sujet Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent bibendum, nisl nec ultrices scelerisque, lectus magna dignissim lectus, at hendrerit diam mauris eget erat.	✓ ✗
	Rappeler Mme Michon 0656768976	✓ ✗
	Why not ?	✓ ✗
	plop pop plop 23/12/2016	✓ ✗

Ci-dessus la page de prise de note.

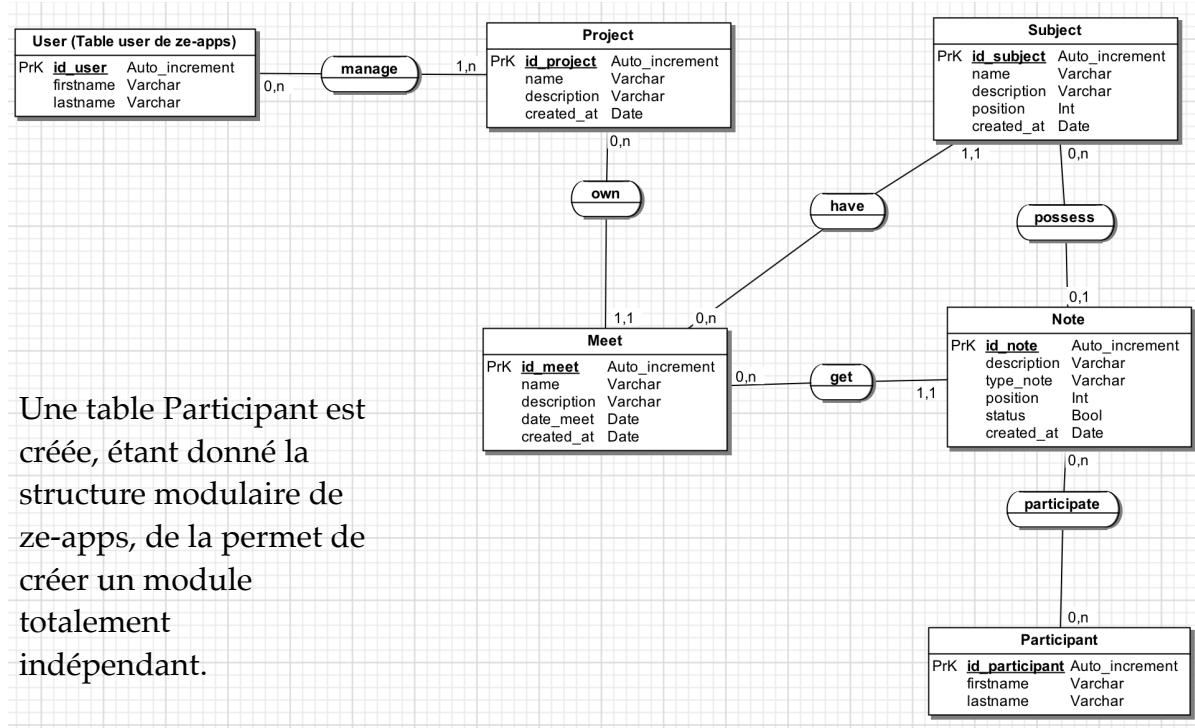
Les icônes en haut de la page sont cliquables.

Lorsque une action est cliquée, cela ouvre un nouveau champs au milieu de la page où l'on peut prendre sa note.

Toutes ces notes peuvent être affiliées à un participant et à un sujet (réunion) et nous pourrons donc trier les notes via un système de filtre.

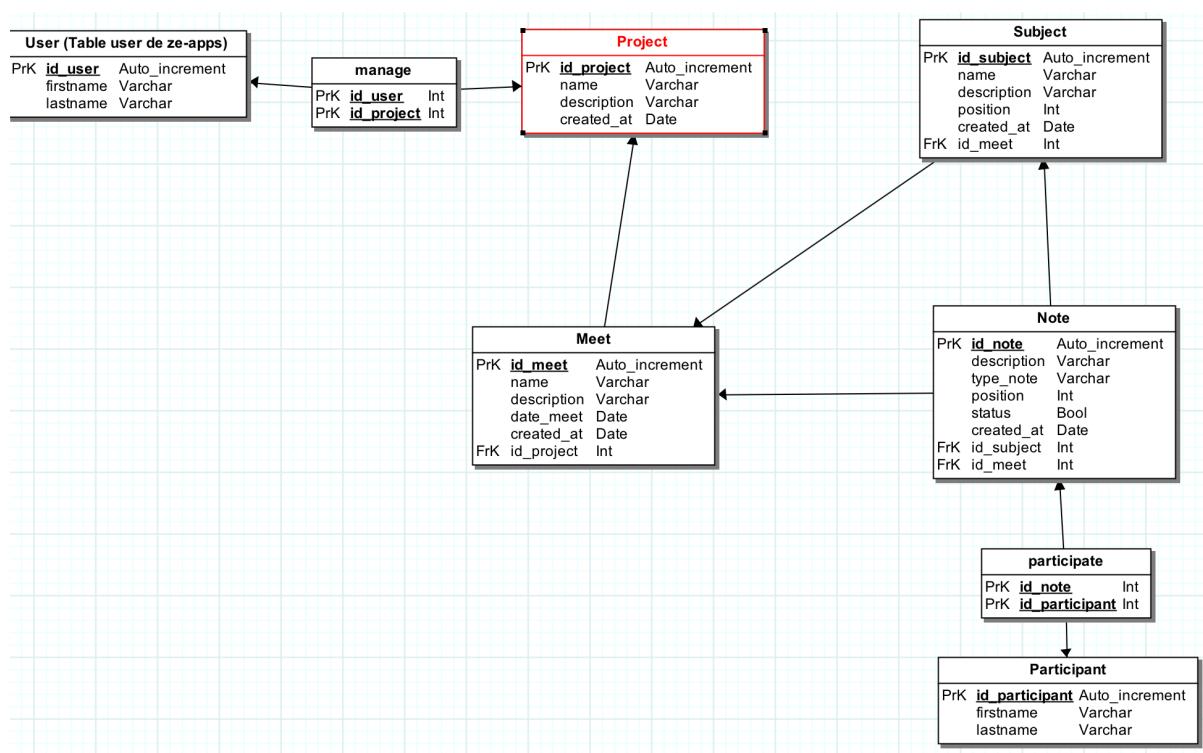
Spécifications Techniques

MCD



Une table Participant est créée, étant donné la structure modulaire de ze-apps, de la permet de créer un module totalement indépendant.

MLD



Mise en place de la base de données

Afin de mettre en place une base de données dans « PHP my admin », il existe différentes manières de procéder.

Il est possible de créer manuellement des tables et de les remplir manuellement en utilisant l'interface de « PHP my admin ».

Il est aussi possible de créer un script au format SQL et de l'importer directement dans « PHP my admin ».

On peut aussi directement écrire ses requêtes dans l'onglet SQL .

Voici le script permettant la création de la table « Projet » dans la base de données:

```
--  
-- Structure de la table `meeting_app_projects`  
--  
  
CREATE TABLE `meeting_app_projects` (  
    `id` int(11) NOT NULL,  
    `id_user` int(11) NOT NULL,  

```

Les champs « **created_at** », « **updated_at** » et « **deleted_at** » sont là pour permettre le soft delete.

Le soft delete permet de garder en base les données supprimées au cas où des données seraient effacées malencontreusement.

Fonctionnement général

Le module est réalisé dans l'ERP « Ze-Apps ».

Les langages utilisés sont AngularJS 1.5 et Bootstrap pour le coté front-end et PHP 5.6 pour le coté back-end. Le tout couplé dans le framework CodeIgniter 3. Le framework CodeIgniter est basé sur le pattern Modèle-Vue-Contrôleur.

Pour les chargement des vues

-L'utilisateur fait une requête d'accès à une URL (Boutons , liens, changement de vues ...)

-Le routeur Angular lit l'URL et charge le contrôleur, le template associé et les éventuelles actions à exécuter avant le chargement du template dans la vue

```
app.config(['$routeProvider',
  function ($routeProvider) {
    $routeProvider
      .when('/ng/meeting_app/project/plan', {
        templateUrl: '/meeting_app/project/plan',
        controller: 'MeetingAppPlanCtrl'
      })

      .when('/ng/meeting_app/project/new', {
        templateUrl: '/meeting_app/project/form',
        controller: 'MeetingAppProjectsFormCtrl'
      })

      .when('/ng/meeting_app/project/:id', {
        templateUrl: '/meeting_app/project/form',
        controller: 'MeetingAppProjectsFormCtrl'
      })
  }]);
;
```

-Angular appelle l'URL du template.

```
class Project extends CI_Controller
{
    //***** Load view called by the http request *****/
    public function form()
    {
        $data = [];
        $this->load->view('project/form', $data);
    }

    public function plan()
    {
        $data = [];
        $this->load->view('project/plan', $data);
    }
}
```

Contrôleur PHP de la classe Project

-Codeigniter réceptionne l'appel et charge la fonction du contrôleur passée dans l'URL qui renvoi le fichier PHP de la vue à Angular.

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');
?>
<div id="breadcrumb"><span i18n="Projet"></span></div>

<div id = "content" ng-click="project_detail">
<div class="row">
    <div class="col-md-12 clearfix ">
        <h2 class="meetingapp-title">Liste des Projets
            <a href="/ng/meeting_app/project/new" class="pull-right">
                <span class="little">nouveau projet</span> <i class="fa fa-plus" aria-hidden="true"></i>
            </a>
        </h2>
    </div>
</div>
<div ng-repeat="project in projects">
    <div class="meetingapp-cell row" >
        <b class="col-md-12" >{{project.name | uppercase}}</b>
        <div class="col-md-6 col-md-offset-6" style="...">
            Statut
            <br/>
        </div>
        <div class="pull-right meetingapp-edit-trash-icon">
            <i class="fa fa-pencil fa-2x" aria-hidden="true" ng-click="edit_project(project.id)"></i>
            <i class="fa fa-trash fa-2x" aria-hidden="true" ng-click="delete_project(project.id)"></i>
        </div>
        <div class="col-md-4">
            <a href="ng/meeting_app/meet/plan/{{project.id}}" class="btn btn-default">
                <strong>{{project.nbMeets}} Réunion<span ng-if="project.nbMeets > 1"></span></strong>
            </a><br>
        </div>
        <div class="col-md-5">
            <div class="progress">
                <div class="progressbar" role="progressbar" aria-valuenow="{{project.avancement}}" aria-valuemax="100" aria-valuemin="0" data-aos="zoom-in" data-aos-duration="2000">
                    {{project.avancement}}%
                </div>
            </div>
            <div class="col-md-12">
                <button class="pointer btn btn-default" ng-click="addMeet(project.id)">
                    Ajouter une réunion
                </button>
            </div>
        </div>
    </div>
</div>
<br/>
</div>
```

Pour les fonctionnalités

- L'utilisateur effectue une action dans la vue.
- Angular cherche dans le contrôleur la fonction appelé par l'action.
- La fonction récupère les nouvelles données, les stocks dans une variable « **data** » et les envoie au contrôleur PHP via une requête « **http** » de type Ajax (asynchronous Js and xml).

La requête contient le nom du module -> l'entité correspondante -> la fonction .
Les données sont formatées au format Json.

```
$scope.form = [];
/**
 * Persist in database
 */
$scope.save = function () {
    var $data = {};

    //***** Check if fields(name and description) are completed *****/
    if ($scope.form.name != undefined && $scope.form.description != undefined) {
        if ($routeParams.id != 0) {
            $data.id = $routeParams.id;
        }

        $data.name = $scope.form.name;
        $data.description = $scope.form.description;

        $http.post('/meeting_app/project/save', $data).then(function (obj) {
            // pour que la page puisse être redirigé
            console.log(obj.data);
            $location.path("/ng/meeting_app/project/plan");
        });
    }
}
```

.
Sauvegarde
d'un
Projet côté
Angular

- Codeigniter récupère l'URL, la parse et charge le contrôleur du module appelé puis exécute la fonction. Le controller PHP réceptionne la requête ainsi que les données envoyées, cherche la fonction demandée (« save » en l'occurrence) et l'exécute .
- La fonction charge les modèles nécessaires à l'action et envoie les données à la BDD .

```

public function save() {
    $this->load->model("meeting_app_projects", "project");
    $this->load->library('session');
    $this->load->model("zeapps_users", "user");

    $user = $this->user->getUserByToken($this->session->userdata('token'));

    // constitution du tableau
    $data = [];

    if (strcasecmp($_SERVER['REQUEST_METHOD'], 'post') === 0 && strpos($_SERVER['CONTENT_TYPE'], 'application/json') !== FALSE) {
        // POST is actually in json format, do an internal translation
        $data = json_decode(file_get_contents('php://input'), true);
    }

    $data["id_user"] = $user->id;

    if (isset($data["id"]) && is_numeric($data["id"])) {
        $this->project->update($data, $data["id"]);
    } else {
        $this->project->insert($data);
    }

    echo json_encode("OK");
}

```

Cf. Sauvegarde d'un projet côté PHP

- Les données sont persistées et le contrôleur PHP renvoie un objet Json générique pour forcer une réponse « http 200 » du serveur et déclencher la promesse de la requête Angular.

En l'occurrence , on redirige vers une autre page.

```

$http.post('/meeting_app/project/save', $data).then(function (obj) {
    // pour que la page puisse être redirigé
    console.log(obj.data);
    $location.path("/ng/meeting_app/project/plan");

});

```

Cf. Promesse et redirection

Réalisation

La contrôleur « Project »

La vue « Project » est la vue d'accueil du module.

Le gestionnaire de projets :

Voici la page d'accueil du module.

Ici nous pouvons créer, éditer, supprimer et voir l'avancement d'un projet.

The screenshot displays a dashboard titled "Liste des Projets". It lists three projects:

- RÉSEAU ELECTRIQUE**: 3 Réunions, Statut 100%, with edit and delete icons.
- RÉSEAU INTERNET**: 1 Réunion, Statut 60%, with edit and delete icons.
- AGRANDISSEMENT DE LA SOCIÉTÉ**: 0 Réunion, Statut 0%, with edit and delete icons.

A "nouveau projet +" button is located in the top right corner of the list area.

L'affichage des projets :

- Dans le contrôleur JS correspondant à la vue , une fonction permet de charger dans la vue tous les projets déjà existants.

```
/****** Load all projects saved in BDD *****/
var loadList = function () {
    $http.get('/meeting_app/project/getAll').then(function (response) {
        if (response.status == 200) {
            $scope.projects = response.data ;
        }
    });
loadList() ;
```

Cf. Chargement
des projets coté
Angular

- La fonction fait un appel http au contrôleur PHP et assigne les données contenues dans la réponse à l'attribut « **projects** » de l'objet « **\$scope** ».
- La fonction « **loadList()** » est exécuté .
- La vue peut maintenant travailler avec l'objet « **projects** »

```
</div>
<div ng-repeat="project in projects">
  <div class="meetingapp-cell row" >
    <b class="col-md-12" >{{project.name | uppercase}}</b>
    <div class="col-md-6 col-md-offset-6" style="...">
      Statut
      <br/>
    </div>
    <div class="pull-right meetingapp-edit-trash-icon">
      <i class="fa fa-pencil fa-2x" aria-hidden="true" ng-click="edit_project(project.id)" ></i>
      <i class="fa fa-trash fa-2x" aria-hidden="true" ng-click="delete_project(project.id)"></i>
    </div>
    <div class="col-md-4">
      <a href="ng/meeting_app/meet/plan/{{project.id}}" class="btn btn-default">
        <strong>{{project.nbMeets}} Réunion<span ng-if="project.nbMeets > 1"></span></strong>
      </a><br>
    </div>
    <div class="col-md-5">
      <div class="progress">
        <div class="progress-bar" role="progressbar" aria-valuenow="{{project.advancement}}" aria-valuemin="0" aria-valuemax="100" style="width: {{project.advancement}}%">
          {{project.advancement}}%
        </div>
      </div>
    </div>
    <div class="col-md-12">
      <button class="pointer btn btn-default" ng-click="addMeet(project.id)">
        Ajouter une réunion
      </button>
    </div>
  </div>
  <br/>
</div>
```

- La directive « **ng-repeat** » permet de faire une boucle directement depuis le DOM.

L'ajout de réunion à un projet:

Une fois un projet ajouté, nous pouvons créer une réunion (ou plusieurs) pour le projet en question. L'ajout d'une réunion se fait via une modale pour éviter la redirection vers une autre page et donc améliorer l'expérience utilisateur.

- L'utilisateur clique sur le champs « Ajouter une réunion ».

```
<div class="col-md-12">
  <button class="pointer btn btn-default" ng-click="addMeet(project.id)">Ajouter une réunion'</button>
</div>
```

- La fonction permettant l'ajout est passée dans une directive « **ng-click** » (propre à Angular) en lui passant l'id du projet en paramètre.

```

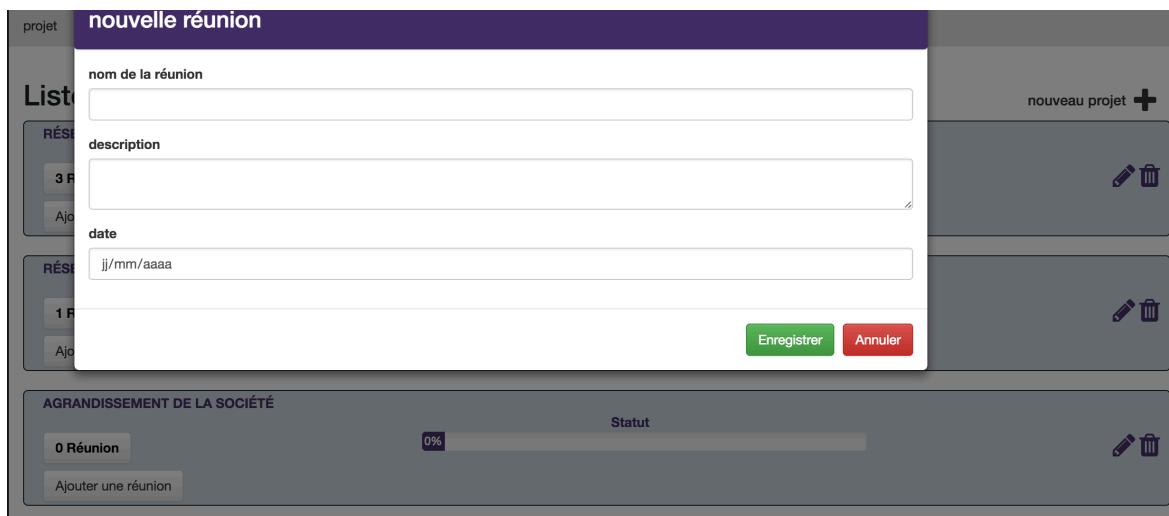
/***** Call addMeet Modal *****/
$scope.addMeet = function (id_project) {
    zeapps_modal.loadModule("meeting_app", "addMeet", {id_project:id_project}, function(objReturn) {
        });
};

}

```

La directive « **ng-click** » nous permet d'appeler une fonction du contrôleur au clic sur l'élément.

- Le contrôleur Angular de la vue exécute la fonction et charge le template de la modal correspondante.
- La fonction « **loadList()** » est rappelé pour recharger les projets et ainsi mettre à jour le compteur de réunions sans recharger la page.



- Une fois le formulaire rempli et validé, la modal se referme et la page est mise à jour à la volée.

Le contrôleur « Meet »

La vue « meet » permet de gérer les réunions ou rendez-vous assignés à un projet.

Le gestionnaire de réunions :

Sur cette page il est possible d'éditer, supprimer et accéder au détail d'une réunion ce qui est le cœur de l'application.

The screenshot shows a web application interface for managing meetings. At the top, there is a dark header bar with a 'menu' button, a search icon, a 'cherche' button, and a user profile 'N. Ramel'. Below the header, a breadcrumb navigation shows 'projet > réunion'. A backlink '← Retour aux projets' is present. The main title is 'liste des réunions'. A table lists three meetings with columns for 'libellé', 'description', 'date prévu', and 'actions'. Each meeting has edit and delete icons in the 'actions' column.

libellé	description	date prévu	actions
Lotissement Aigrefeuille	Obtention du marché sur le raccordement électrique d'Aigrefeuille sur Maisne	10/12/2016	
Extension Accenture	Agrandissement du bâtiment Accenture	17/12/2016	
Câblage nouveau site école IMIE	Préparation du câblage pour le déménagement de l'IMIE à St Herblain	10/12/2016	

La fonction « Edit » :

La fonction « edit » permet de modifier une réunion préalablement entrée.

Pour ce faire, la modale de formulaire est appelée et surchargée puis renvoyée dans la base de donnée.

The screenshot shows a modal window titled 'nouvelle réunion' (new meeting). It contains fields for 'nom de la réunion' (name of the meeting) with the value 'Lotissement Aigrefeuille', 'description' (description) with the value 'Obtention du marché sur le raccordement électrique d'Aigrefeuille sur Maisne', and 'date' (date) with the value '10/12/2016'. At the bottom of the modal are 'Enregistrer' (Register) and 'Annuler' (Cancel) buttons. In the background, the main 'Meet' controller table is visible, showing the same three meetings listed earlier.

- L'utilisateur clique sur le crayon se trouvant sur la ligne de la réunion.

	date prévu	actions
	10/12/2016	  

- Une directive « **ng-click** » est passée dans la balise englobant le logo. L'id de la réunion est passé en argument pour que le contrôleur Angular puisse la récupérer.

```
<td>
  <div class="pull-right meetingapp-edit-trash-icon">
    <a href="/ng/meeting_app/subject/plan/{{meet.id}}"/></a>
    <i class="fa fa-pencil fa-2x" aria-hidden="true" ng-click="edit_meet(meet.id)" ></i>
    <i class="fa fa-trash fa-2x" aria-hidden="true" ng-click="delete_meet(meet.id)"></i>
  </div>
</td>
```

- Le directive « **ng-click** » appelle la fonction du contrôleur « **edit_meet** ».
- Le contrôleur rappelle la même modale servant à ajouter une réunion et lui passe l'argument reçu en option. L'argument « **id** » est passé sous la forme d'un objet **{id:id}**.

```
$scope.edit_meet = function (id) {
  zeapps_modal.loadModule("meeting_app", "addMeet", {id:id}, function(objReturn) {
    loadList();
  });
};
```

- Le contrôleur de la modale grâce à une condition vérifie que l' « **id** » est bien passé en argument et qu'il est différent de zéro.

```
// charge la fiche

if (option.id && option.id != 0) {
    $http.get('/meeting_app/meet/get/' + option.id).then(function (response) {
        if (response.status == 200) {
            $scope.form = response.data;
            $scope.form.date_meet = new Date($scope.form.date_meet);
        }
    });
}
```

- Il fait une requête http au serveur PHP en passant l' « id » dans le chemin pour récupérer les données dans la BDD correspondant à cet « id ».

```
public function get($id) {
    $this->load->model("meeting_app_meets", "meet");
    echo json_encode($this->meet->get($id));
}
```

- Grace à la promesse faite dans le contrôleur Angular, les données sont récupérées et assignées à « \$scope.form ».
- Une fois la correction effectuée , l'utilisateur clique sur le bouton enregistrer possédant lui aussi une directive « ng-click » servant à enregistrer les modifications en base de données.

Grace à cette méthode, l'édition d'une donnée existante est facilitée car l'utilisateur n'a pas à remplir l'intégralité des champs mais simplement le champs à corriger.

Pour rentrer dans le détail de cette réunion il suffit de cliquer sur n'importe quel champs dans le tableau correspondant à cette réunion,

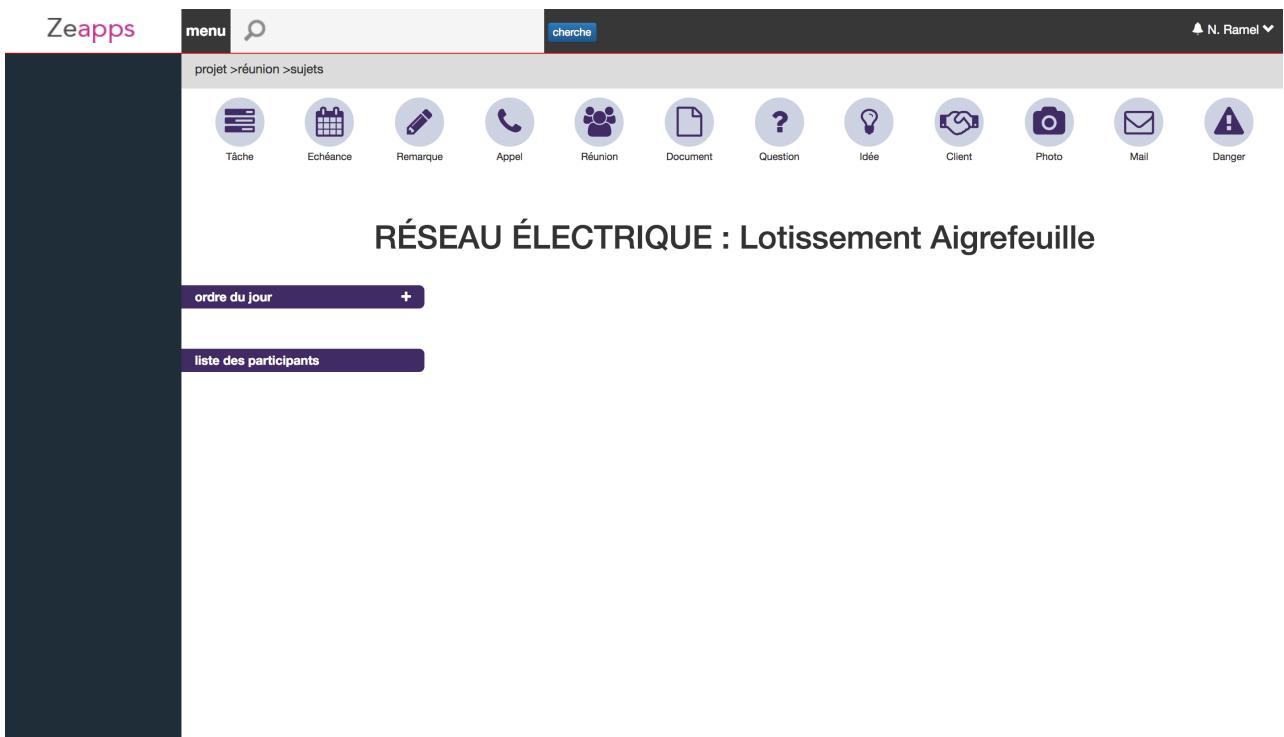
```
<td><b> <a href="/ng/meeting_app/subject/plan/{{meet.id_project}}/{{meet.id}}">{{meet.name}}</a></b></td>
<td> <a href="/ng/meeting_app/subject/plan/{{meet.id_project}}/{{meet.id}}">{{meet.description}}</a></td>
```

En effet, les champs « libellé » et « description » sont placés dans une balise « <a> » avec le lien vers la page suivante dans lequel nous passons l'id du projet ainsi que l'id de la réunion sur lequel nous avons cliqué

Le contrôleur « Subject »

La vue « subject » est le cœur de l'application, c'est elle qui permet d'attribuer des sujets puis des notes aux réunions définies précédemment.

Le gestionnaire de réunions :



Pour créer une note il suffit simplement de cliquer sur un des icônes les symbolisant en haut de l'écran.

```
<div class="col-md-1">
  <i class="fa fa-pencil fa-3x meetingapp-circle pointer" ng-model="remark" ng-click="createNote('remark')" aria-hidden="true"></i><br/>
  Remarque
</div>
<div class="col-md-1">
  <i class="fa fa-phone fa-3x meetingapp-circle pointer" ng-model="call" ng-click="createNote('call')" aria-hidden="true"></i><br/>
  Appel
</div>
```

Pour chaque type de notes , le type est passé dans une directive « **ng-model** » qui va permettre de lier un l'élément du DOM au **\$scope**.

Les icônes ont étées faites avec l'outil “Font Awesome” et du CSS pour permettre de faire le cercle et la couleur entourant le logo et gérer sa taille.

```
.meetingapp-circle{  
    display: block;  
    width:55px;  
    height:55px;  
    line-height: 55px;  
    color: #452F66;  
    background:#D0D6E1;  
    border-radius: 50%;  
}
```

Un carré est simplement initié en mettant les propriétés “width“ et “height“ à la même valeur ensuite les bords sont arrondies avec le “border-radius“ a 50%. Les couleurs sont ensuite définies par “color“ et “background“ avec des références couleurs en hexadécimal.

Une classe « col-md-1 » est attribué à la « div » entourant chaque note. C'est le framework « **Bootstrap** » qui permet de l'utiliser. Le framework divise la page en 12 colonnes égales, le fait de d'attribuer la valeur « 1 » à chaque note sachant qu'il y 12 types de notes au total permet de toutes les placés idéalement sur la page.

La création des notes:

Pour l'ajout des notes , lorsque le logo est cliqué , une note vide se crée.



La page se met en mode édition, ce qui fait qu'il est désormais impossible de créer une nouvelle note tant que la note n'est pas validée ou annulée.

```

***** Create an empty field for note *****

$scope.createNote = function (type_note) {
    if ($scope.edition_encours_global == false) {
        var data = {};

        data.name = "";
        data.description = "";
        data.description_edit = "";
        data.type_note = type_note;
        data.edition_encours = true ;
        data.position = 0;
        data.id_subject = 0;

        data.participants = [];

        if ( !$scope.arrayNotes[0]){
            $scope.arrayNotes[0]= [];
        }
        $scope.arrayNotes[0].unshift(data);

        $scope.edition_encours_global = true;
    }
}

```

Le contrôleur Angular exécute la fonction , instancie un objet vide « data » et le remplit :

- **name** -> Chaîne de caractère vide

- **description** -> Chaîne de caractère vide

- **description_edit** -> Chaîne de caractère vide (Cela va servir plus tard en attribuant la valeur à **description** ou non au cas où l'utilisateur annulerait l'édition)

- **type_note** = L'argument passé dans la fonction

- **position** = Elle est passée à 0 (La variable servira plus tard pour enregistrer la position lors d'un drag and drop)

- **id_subject** = Il est passé à 0 car pour l'instant la note est juste affiliée à un rendez vous, l'**id_subject** sera modifié lorsque l'on attribuera cette note à un ordre du jour ultérieurement.

- **participants** = on crée un tableau qui servira à stocker des participants pour chaque notes.

Une condition vérifie si l'index 0 du tableau **arrayNotes[]** existe, sinon il l'initialise avec un tableau vide . Ensuite il place l'objet **data** au début de la pile (index 0).

L'ajout d'ordre du jour:



RÉSEAU ÉLECTRIQUE : Lotissement Aigrefeuille

ordre du jour +

CABLAGE
MATÉRIAUX
VOIRIE

liste des participants

Nicolas Rameil
Thomas Sauques
Lucas Desrues

CABLAGE

erertz
Participants: Thomas Sauques ,

Calcul du cable disponible au dépôt

Un cable de meilleur qualité pourrait réduire le coût de maintenance
Participants: Nicolas Rameil ,

MATÉRIAUX

Appeler BP matériaux
Participants: Lucas Desrues ,

Prendre tout les matériaux chez BP ?

Lorsque l'utilisateur arrive sur cette vue, elle ne contient que le nom du projet ainsi que le nom de la réunion .

L'utilisateur peut s'il le veut créer des Ordres du jour (Sujets) à sa réunion. Pour se faire, il lui suffit de cliquer sur le `+`dans le menu de gauche à coté de « ordre du jour ».

Une modale de formulaire s'ouvre de la même manière que présenté précédemment à la page 22.

Maintenant des barres symbolisant les ordres du jour apparaissent dans la zone principale de la vue grâce à la directive « **ng-repeat** ».

```
<div ng-repeat="subject in subjects" class="col-md-12 ">
    <h4 class="subheader col-md-12">{{subject.name|uppercase}}</h4>
```

Un filtre « **uppercase** » est appliqué pour automatiquement affiché en majuscules.

Ces barres vont permettre de trier naturellement les notes que nous avons initiées avec un système de Drag and Drop.

Le Drag And Drop des notes coté HTML:

Les notes peuvent maintenant être triées avec un système de Drag and Drop.
Pour mettre en place cette fonctionnalité, la librairie « JQuery ui-sortable » sera utilisée.

Elle est disponible sur GitHub à l'URL : <http://angular-ui.github.io/ui-sortable/>.

Pour mettre en place cette fonctionnalité, deux tableaux quasi identiques sont créés. La première contient les notes créées par défaut, auxquels aucun ordre du jour (sujet) n'a été donné. C'est ces notes qui appartiennent pour l'instant qu'à une réunion.
Et le deuxième est placé dans la <div> (p.29) qui contient l'affichage des ordres du jour.

1er tableau:

```
iv class="col-md-12">


||
||
||


```

La balise <tbody>:

- Dans la balise <tbody> la directive « **ui-sortable** » est appelée, c'est la fonction permettant le Drag and Drop et « **sortableNote** » est son nom.

- Le sélecteur « **noteContainer** » est indiqué dans la « **class=" "** », c'est ce sélecteur qui va permettre à la fonction de voir quels sont les tableaux connectés.
- « **ng-model** » indique que nous travaillerons avec l'index 0 de « **arrayNotes** ».
- « **data-type** », ici il est paramétré à 0 d'office, c'est le tableau par défaut.

La balise <tr>:

- « **ng-repeat** » qui va permettre d'afficher toutes les notes de « **noteArray[]** » ayant l'index 0.
- « **data-id** » où est récupéré l'id de la note.
- « **class=" "** », l'id de la note est récupéré et stocké avec une chaîne de caractère pour une meilleure compréhension. C'est ce qui va permettre de voir qu'elle ligne (note) du tableau a été sélectionnée.

2ème tableau:

```
<table class="table pointer">
  <tbody ui-sortable="sortableNote" class="noteContainer defaut" ng-model="arrayNotes[subject.id]" data-type="{{ subject.id }}>
    <tr ng-repeat="note in arrayNotes[subject.id] | orderBy:'position'" class="col-md-12 ligne_tableau_{{note.id}}" ng-click="edit_note(note)" data-id="{{note.id}}>
```

Le deuxième tableau est pratiquement identique, sauf :

- « **ng-model** » indique que nous travaillerons avec l'index correspondant à l'id du sujet de « **arrayNotes** ».
- « **data-type** », le numéro du tableau est celui correspondant à l'id du sujet.

Voici donc la manière de procéder du côté HTML avec cette fonctionnalité, passons maintenant au cœur de la fonction du côté Angular.

(J'espère que cette explication était suffisante car il m'est impossible de faire un screen shot de toute la page qui est assez longue)

Le Drag And Drop des notes coté Angular:

```
// Drag and Drop From one table to another

$scope.sortableNote = {
  connectWith: ".noteContainer",
  placeholder: "app",
  delay: 300,
  axis: "y",
  stop: function( event, ui ) {

    //Id of dragged note
    var idObj = $(ui.item[0]).attr("data-id") ;
    console.log(idObj);
    //Select the table line and his parent ("tr")
    var selectedLine = $(".ligne_tableau_" + idObj) ;
    var subject_id = selectedLine.parent().attr("data-type") ;

    var position = -1 ;
    var positionDefinitive = 0 ;

    //Select tbody and go through each row
    $("tr", selectedLine.parent()).each(function () {
      position++ ;
      if (idObj == $(this).attr("data-id")) {
        positionDefinitive = position ;
      }
    }) ;
    var data = {} ;
    data.idObj = idObj ;
    data.id_subject = subject_id ;
    data.position = positionDefinitive ;

    $http.post('/meeting_app/subject/saveNotePosition', data);
  }
}
```

Dans la fonction, les options sont configurées :

- **connectWith** : Sert à connecter différentes listes , en l'occurrence « **noteContainer** » qui a précédemment mis dans les « **class=" "** » des deux tableaux à connecter.

- **delay** : Sert à définir le temps où il faut rester cliquer sur l'élément avant de déclencher la fonction.

- **axis** : Sert à limiter le déplacement du drag and drop à l'horizontale ou à la verticale.

- **stop** : Pour décrire quoi faire lorsque on relâche l'élément.

Ensuite, le cœur de la fonction se trouvant dans **stop** :

- Une variable « **idObj** » est instanciée, l'id de l'object ciblé est récupéré grâce à JQuery et l'attribut « **data-id** » qui a été défini (p.31).
- Une variable « **selectedLine** » est instanciée, qui va servir à savoir quelle ligne du tableau est impactée.
 - Une variable « **subject_id** » est instanciée, l'attribut parent « **data-type** » est récupéré et stocké dans la variable.
Celle ci permet de savoir dans quel tableau nous sommes.
- Tout les « **<tr>** » du tableau sont parcourus et lorsque « **idObj** » est identique au « **data-id** » , nous savons où a été lâché la note et la position de la note est stocké dans une variable.
 - Toutes les variables nécessaire à la mis à jour de la position des notes sont passées dans un objet « **data** » et envoyé au contrôleur PHP via une requête **\$http** pour être traitées.

Maintenant le contrôleur PHP va devoir mettre à jour toutes les positions de chaque notes pour chaque tableaux dans la BDD.

Le Drag And Drop des notes côté PHP:

```
public function saveNotePosition() {
    $this->load->model("meeting_app_notes", "notes");

    $this->load->library('session');
    $this->load->model("zeapps_users", "user");

    // constitution du tableau
    $data = array();

    if (strcasecmp($_SERVER['REQUEST_METHOD'], 'post') === 0 && stripos($_SERVER['CONTENT_TYPE'], 'application/json') !== FALSE) {
        // POST is actually in json format, do an internal translation
        $data = json_decode(file_get_contents('php://input'), true);
    }

    //echo json_encode($data["idObj"]);

    $note = $this->notes->get($data["idObj"]);

    $this->notes->updateOldTable( $note->position);

    $note->position = $data["position"] ;
    $note->id_subject = $data["id_subject"];

    $this->notes->updateNewTable($data["position"]);

    //give good position value to dragged note
    $this->notes->update($note, $note->id);
}
```

Comme à chaque fonction où une connexion à la base de donnée est nécessaire, les modèles requis sont chargés.

Tout les données envoyées par le contrôleur Angular sont stockées dans un tableau « **\$data** » ce qui permet d'accéder facilement aux données.

Grace au modèle “**meeting_app_notes**“ et à la variable « **idObj** » envoyé par Angular, la note qui a bougé est récupérée et stockée dans un objet « **\$note** ».

Ensuite une requête **SQL** est faite , cette requête est stocké dans le modèle “**meeting_app_notes**“ et a été placé dans une fonction « **updateOldTable** » où est placé en argument « **\$note->position** » (méthode pour accéder à l'attribut position de l'objet **\$note**).

```

?php

class Meeting_app_notes extends MY_Model
{
    public function __construct()
    {
        parent::__construct();
        $this->soft_deletes = TRUE;
    }

    public function updatePositions($position, $id_meet, $id_project, $id_subject) {
        return $this->db->query("UPDATE Meeting_app_notes SET position = (position+1) WHERE position >= ".
            ".$position." AND id_meet = ".$id_meet." AND id_project = ".$id_project." AND id_subject = ".$id_subject);
    }

    public function updateOldTable($position) {
        return $this->db->query("UPDATE Meeting_app_notes SET position = (position-1) WHERE position > ".$position);
    }

    public function updateNewTable($position) {
        return $this->db->query("UPDATE Meeting_app_notes SET position = (position+1) WHERE position >= ".$position);
    }
}

```

Nous voyons donc ici la fonction « **updateOldTable** » placé dans le modèle des notes contenant la requête.

La requête accède à la table « **meeting_app_notes** » dans la base de données et décrémente de 1 les notes où la position actuelle est strictement inférieure à la position de la note “draggé”.

« **updateNewTable** » va elle incrémenté de 1 les notes qui ont une position supérieur ou égale à la position de la note “draggé”.

La fonction du contrôleur PHP change l’id du sujet (ordre du jour) si nous avons changé de tableaux.

Elle attribue désormais la bonne position à la note dragué grâce à son id.

Désormais toutes les notes de chaque tableaux ont une position mise à jour ce qui permettra qu’au rechargement de la page ou à la prochaine utilisation les notes soient toujours affichées dans l’ordre établi.