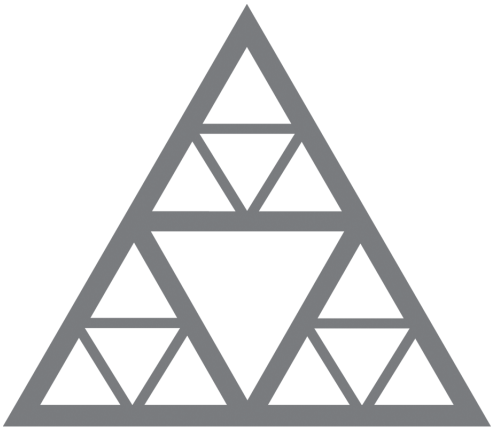
École Nationale des Ponts et Chaussées



École des Ponts

ParisTech

Techniques de développement logiciel
Développement d'une application d'organisation de groupes de travail

Healey Holly
Lecoutre Jérémie
Yang Ping'an
Capgras Justin

1 Présentation de l'application

1.1 Objectif initial

L'objectif de notre projet est de réaliser une application permettant une meilleure organisation de groupes de travail. Les principales fonctions souhaitées sont :

— L'attribution de tâches aux membres d'un groupe de travail

— Un suivi de l'ensemble des tâches au sein de chaque groupe de travail

— Un calendrier représentant l'ensemble des événements relatifs à chaque groupe de travail

Les utilisateurs ont ainsi la possibilité de créer des groupes de travail auxquels sont rattachées des tâches partagées. L'avancement de celles-ci peut être suivi par tous les membres appartenant au groupe.

L'application est aussi dotée d'une page calendrier où s'affichent les deadlines relatives aux différents projets sur lesquels l'utilisateur travaille. La navigation d'une page à l'autre se fait depuis une barre de navigation située sur le bas de l'écran.

Page d'accueil
La page d'accueil prend deux formes en fonction du statut de l'utilisateur (connecté ou déconnecté). S'il est déconnecté, un formulaire de connexion s'affiche, ainsi qu'une option pour récupérer son mot de passe. Il a également la possibilité de créer un compte s'il n'est pas encore enregistré sur l'application. Une fois l'utilisateur connecté, la page d'accueil affiche les différents groupes auxquels appartient l'utilisateur. On y trouve l'option de créer un nouveau groupe de travail, et de modifier ou supprimer un groupe existant. Un groupe est défini par son nom, une brève description, et par les membres qui le composent.

Tâches
La page de tâches affiche la liste des tâches à effectuer ainsi que les tâches déjà effectuées (affichées en fin de liste). Une tâche est définie par un titre, une description et les membres concernés par sa réalisation. On trouve également les options pour ajouter, modifier et supprimer une tâche.

Calendrier
Le calendrier permet de créer des événements. Les événements sont rattachés à un groupe et s'affichent dans le calendrier de tous les utilisateurs concernés. Un événement est défini par son titre, une description, le groupe auquel il est rattaché et la deadline de l'événement. Une fois créé, l'événement s'affiche au niveau de la deadline sur un créneau d'1h avant la fin de sa réalisation. Plusieurs événements peuvent avoir la même deadline et s'affichent en parallèle dans la case. Il existe également une fonction pour supprimer un événement de son calendrier.

Chat
Nous avons aussi envisagé la mise en place d'un chat dans l'application afin de pouvoir discuter entre membres d'un même groupe.

1.2 Réalisation effective

Toutes les pages évoquées précédemment ont pu être créées à l'exception du chat que nous avons décidé de ne pas implémenter. L'interface de connexion liée à la notion d'authentification a été réalisée visuellement en HTML mais nous n'avons pas eu le temps d'implémenter un véritable système d'authentification (fonctions du back).

L'interface visuelle de l'application est la suivante :

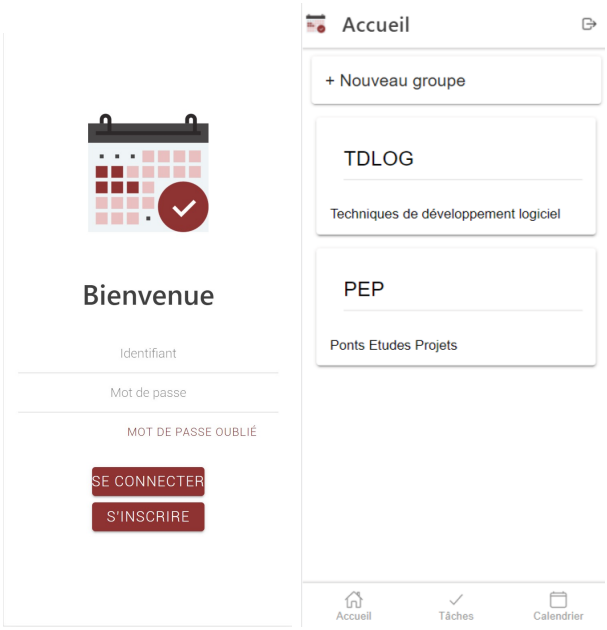


FIGURE 1 – Page d'accueil : affichage déconnecté et connecté

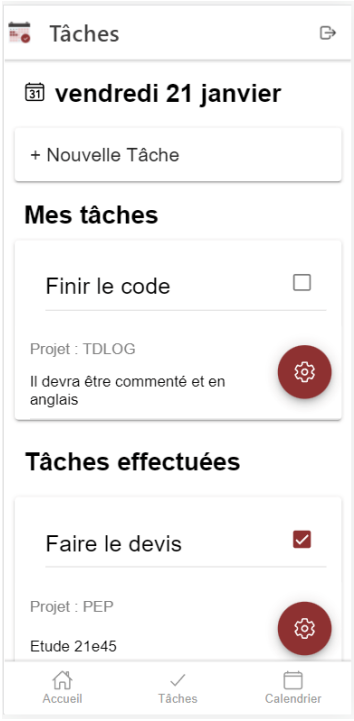


FIGURE 2 – Page de tâches

FIGURE 3 – Calendrier

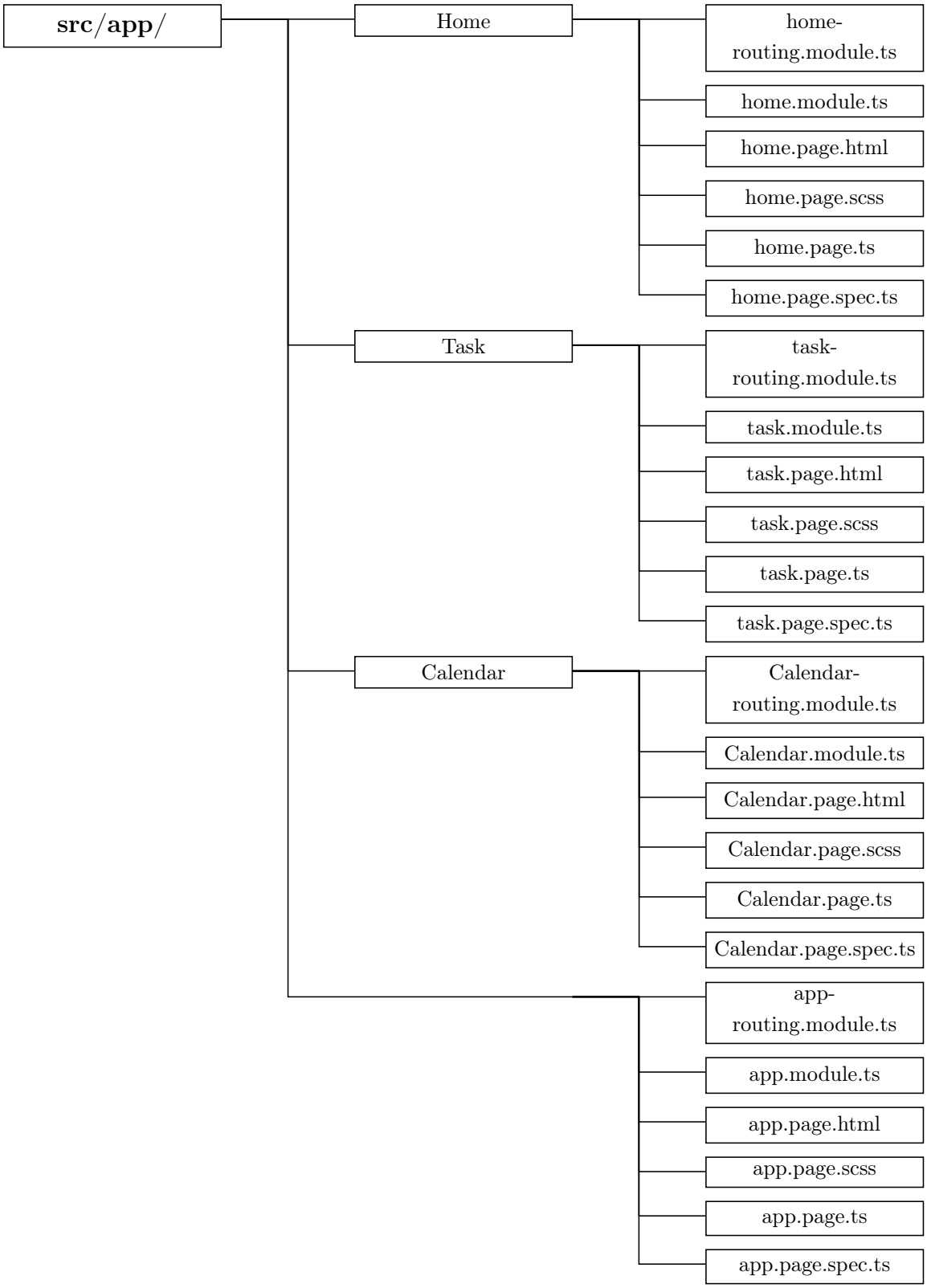
2 Point technique : développement front-end et back-end

La partie front est codée en HTML/CSS. Elle se base sur les composants du framework Ionic avec lequel est développé l'application. Il s'agit d'un framework basé sur des composants UI afin de développer des applications hybrides. La partie back-end est codée en Typescript du fait de l'utilisation d'Ionic et est alimentée par une base de données du service d'hébergement Firebase.

2.1 Structure du code et de la base de données

2.1.1 Structure du code

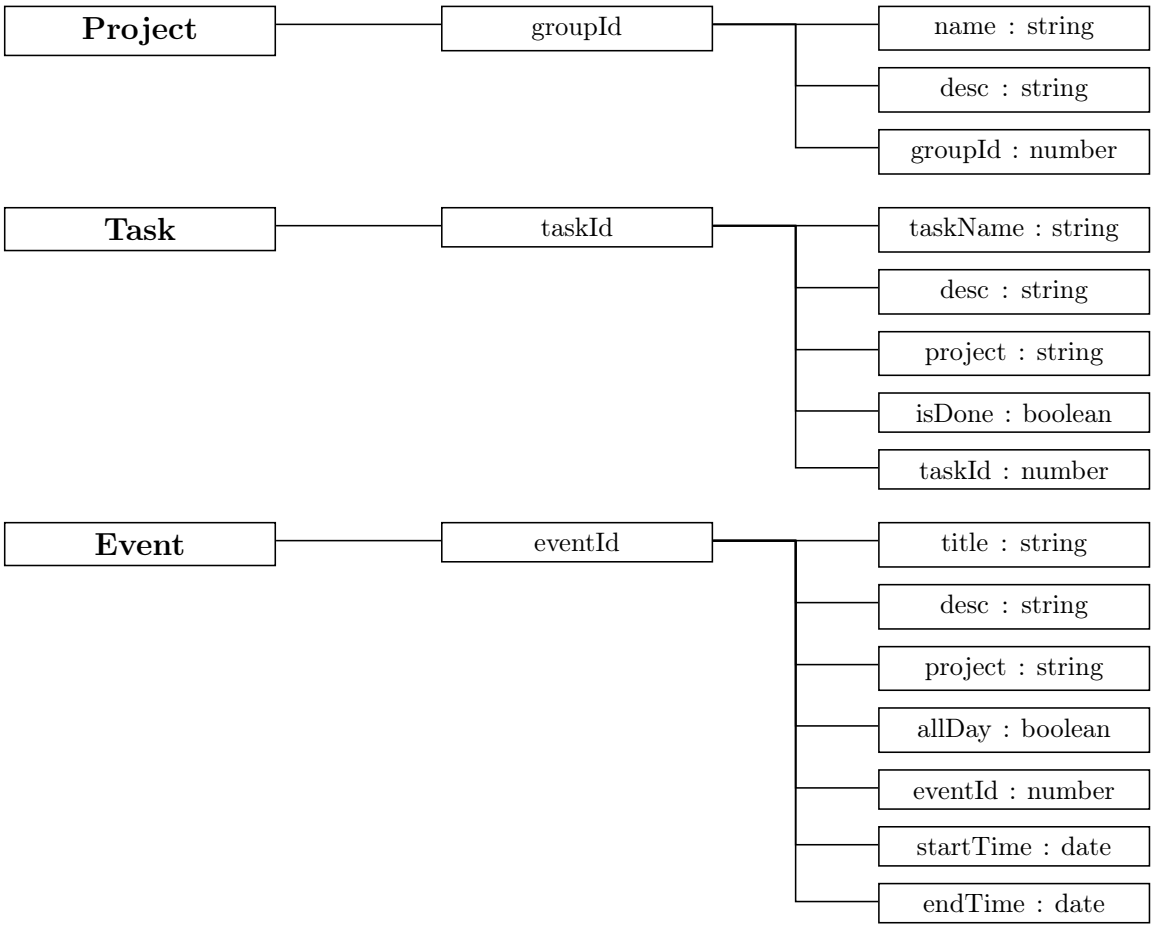
La partie code se situe au chemin *src/app/*. Elle se présente comme ceci :
A chaque page sont associés 6 fichiers. Ceux que nous modifierons sont les fichiers *.html* qui définit la structure de la page, *.sccs* qui en définit le style et *.ts* qui définit la partie dynamique.



2.1.2 Structure de la base de données

On utilise le Cloud Firestore de Firebase, dont la structure est la suivante : chaque tuple est défini par un identifiant qui le relie à sa table. Cet identifiant peut être généré aléatoirement par Firebase mais nous avons décidé de le définir nous même pour avoir plus de facilité dans la manipulation des tables. Comme nous le définissons nous-même, il est répété dans le tuple.

Les trois tables sont indépendantes.



2.2 Focus sur la page des tâches

Comme il serait trop long de rentrer dans tous les détails de l'application, nous allons nous limiter à une présentation détaillée de la page *Task* qui regroupe la plupart des fonctionnalités que nous avons développé. Les autres pages contiennent des fonctions similaires à celles que nous détaillons ici. Tout d'abord, nous récupérons la date du jour, pour cela, on utilise la fonction *toLocaleDateString* dans le constructeur de la page :

```
1 constructor(public fb: AngularFirestore) {
2     const date = new Date();
3     this.currentDate = date.toLocaleDateString('fr-FR',{ weekday: 'long',
          month: 'long', day: 'numeric' });
4     this.taskList = this.fb.collection('Task/').valueChanges();
5     this.projectList = this.fb.collection('Project/').valueChanges();
6   }
```

2.2.1 Ajout de nouvelles tâches

On utilise un « ion-card » pour entrer les informations de notre nouvelle tâche sous forme d'un formulaire. Nous avons donc créé une variable pour piler et dépiler le formulaire lorsque l'on clique sur le bouton d'ajout de tâche.

Après avoir cliqué sur le bouton, une fonction appelée *addTask*() envoie les données saisies vers notre base de données. On peut aussi choisir quel groupe est concerné par cette tâche grâce à une liste à cocher (générée par un « ion-checkbox »). Le titre de la tâche est obligatoire : s'il n'est pas renseigné, il n'est pas possible d'appuyer sur le bouton de création de la tâche et d'ajouter cette tâche dans la base des données.

FIGURE 4 – Ajout de nouvelles tâches

Après avoir ajouté les nouvelles tâches, elles s'affichent chacune dans une carte dans la section « Mes tâches ».

Cette fonction s'appuie sur du code en Typescript et HTML. Le formulaire est notamment créé en HTML tandis que la récupération des données se fait en Typescript. « ngModel » permet de lier ces 2 environnements en permettant le transfert des données. La liste *this.takelist*, définie dans le constructeur, permet de récupérer les tâches depuis la base de données.

```
addTask(){
    console.log("tache ajout e")
    let taskCopy = {
      taskName: this.task.taskName,
      desc: this.task.desc,
      taskId:Math.random(),
      isDone: false
    }
    if(this.isCheckModifier == 1){
      this.isCheckModifier = 0
    }

    this.fb.firestore.doc('Task/'+taskCopy.taskId).set({
      taskName: taskCopy.taskName,
      desc: taskCopy.desc,
      taskId: taskCopy.taskId,
      isDone: taskCopy.isDone,
```

```
18        project: this.projectChecked
19      });
20      this.resetTask();
21    }
```

Ici, $isCheckModifier$ permet de faire passer le formulaire du mode modification de tâche (valeur de 1) au mode création de tâche (valeur 0). On réinitialise les valeurs à la fin en replaçant les données string par ' '.

2.2.2 Classification des tâches

Dans chaque carte on retrouve les informations suivantes : titre, description et projet concerné. Dans le coin supérieur droit, nous avons ajouté une case à cocher représentant l'état de la tâche (réalisée ou non). L'icône est remplie lorsque l'on clique dessus pour signifier la réalisation de la tâche. Lorsqu'une tâche est réalisée, elle est envoyée en fin de liste sur l'affichage pour permettre à l'utilisateur de visualiser en priorité les tâches non terminées.

L'ordre d'affichage des tâches est déterminé par le classement des données dans l'ensemble de tâches que nous avons réalisé en Typescript. Pour classer automatiquement les tâches finies, on utilise une qui utilise la variable booléenne « IsDone ». Si une tâche est finie, sa variable « IsDone » est égale à true et false sinon. Ainsi il suffit de classer les cartes avec cette variable. On utilise pour cela une boucle sur les tâches de la base de données en ne gardant que les tâches pour lesquelle $isDone == true$ puis pour lesquelles $!isDone == true$.

FIGURE 5 – Classification des tâches

2.2.3 Suppression de tâches

Dans le coin inférieur droit nous avons ajouté une roue de réglages qui permet d'effectuer diverses manipulations sur la tâche. En cliquant sur cette icône, 2 possibilités s'offrent à l'utilisateur : la modification de la tâche (crayon) et sa suppression (corbeille).

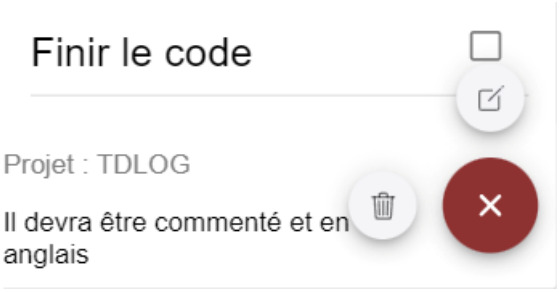


FIGURE 6 – Suppression et modification des tâches

Si l'on appuie sur la corbeille, la fonction $deleteTask(id)$ s'exécute et supprime de la base de donnée la tâche ayant l'ID correspondant à id.

```
deleteTask(id: any){
    this.fb.firestore.doc('Task/'+id).delete()
  }
```

La tâche ne s'affiche donc plus.

2.2.4 Modification de tâches

Cette fonctionnalité s'appuie sur la même questionnaire que celui réalisé pour ajouter une tâche, à la différence que le projet ne peut pas être modifié. Comme expliqué précédemment, on différencie les deux questionnaires avec la variable $isCheckModifier$. Après avoir appuyé sur l'icône de modification, une fonction $beforeUpdatingTask$, qui change la valeur de $isCheckModifier$ et des indications à l'utilisateur, s'exécute. De plus les tâches étant des variables locales aux cartes, il faut faire passer l'identifiant de la tâche en variable globale pour la lier au questionnaire. Cette variable globale est $idForModify$.

```
beforeUpdatingTask(task: any){
    this.isCheckModifier=(this.isCheckModifier+1)%2
    this.titleOfTitre[1] = task.taskName
    this.titleOfDescription[1] = task.desc
    this.idForModify=task.taskId
  }
```

Le formulaire s'ouvre alors pour permettre la modification des informations. Le titre et la description précédents sont affichés dans l'espace d'input des nouvelles valeurs. Lorsque l'on clique sur le bouton « modifier tâche », la fonction $updateTask$ s'exécute est met à jour le titre et la description :

```
updateTask(){
    this.fb.firestore.doc('Task/'+this.idForModify).update({
      taskName: this.task.taskName,
      desc: this.task.desc
    })
    this.isCheckModifier=0
    this.resetTask();
  }
```

FIGURE 7 – Modification des tâches

3 Ouvertures possibles

Nous avons envisagé plusieurs améliorations à apporter à notre application si nous avions disposé de plus de temps.

Options de personnalisation
Nous voudrions permettre davantage d'options de personnalisation (comme la définition d'une couleur par groupe) pour permettre une meilleure lisibilité du calendrier et des tâches. Cette fonctionnalité demanderait un investissement de l'ordre de quelques jours si l'on ajoute également des fonctions de tri (tri des tâches par groupe par exemple).

Chat
La fonctionnalité de chat que nous avions initialement prévue pourrait être implémentée. Cette fonctionnalité demanderait un investissement de l'ordre du mois dû aux technologies utilisées.

Authentification
La fonctionnalité d'authentification a été préparée en front-end mais nous n'avons pas eu le temps de la gérer en back. Il aurait fallu rajouter des identifiants users dans les bases de données et les adresses URL de pages pour rendre leur accès unique à chaque utilisateur. La gestion d'authentification peut se faire très bien avec le module *Auth* de Firebase. Une semaine supplémentaire aurait suffi à implémenter cette fonction et l'on pourrait compter un mois supplémentaire pour pouvoir faire interagir les utilisateurs en groupes de travail plutôt que de passer par des projets individuels.

Gestion des erreurs
Si notre application est fonctionnelle si l'utilisateur la maîtrise, elle peut avoir quelques résultats inattendus. Par exemple pour la sélection du groupe correspondant à une tâche, l'utilisateur ne peut en sélectionner qu'un seul. S'il en sélectionne deux, celui qu'il a coché en dernier prend le dessus. Régler ce problème nécessiterait de décocher automatiquement une case si une autre est sélectionnée.

4 Retour d'expérience

Justin

Le projet que nous avons choisi était à mon sens assez ambitieux. J'avais proposé la technologie et les frameworks parce que j'avais été amené à les utiliser une fois et qu'ils étaient le mieux documentés au niveau des applications mobiles. Malheureusement, nous avons été confrontés à trop de problèmes techniques qui se sont rajoutés à notre manque d'expérience. Cependant, même malgré ces difficultés, je trouve que nous avons tous énormément progressé et que cela nous a permis d'utiliser des concepts (web, base de données) que l'on n'aurait pas pu pratiquer en dehors de ce cours dans notre scolarité à l'École des Ponts.

Holly

Ce cours permettait de se familiariser avec les techniques de développement logiciel. N'ayant pas beaucoup d'expérience en programmation, j'ai trouvé difficile de s'approprier un nouveau langage (Typescript) sans réelle formation. Je pense avoir perdu beaucoup de temps en début de projet à essayer de me former pour ne pas en tirer profit par la suite. J'aurais peut être apprécié avoir une réunion avec l'encadrant en amont du choix du sujet pour mieux évaluer les choix que nous avons fait. Finalement, je trouve le résultat de notre application tout de même satisfaisant et j'ai pu mieux appréhender certains aspects des technologies utilisées.

Jérémie

Ce cours a été une belle occasion d'introduire le développement logiciel dans notre cursus. Si la transition entre les TP et le projet a d'abord paru abrupte du fait de notre inexpérience, ce dernier a été une bonne occasion d'appliquer les cours de SQL suivis en classe préparatoire. Cette voie n'a pas abouti et nous avons finalement privilégié une solution NoSQL, mais les différentes tentatives de gestion d'une base de donnée nous ont permis une belle progression.

Ping'an

Pendant ce cours, nous avons choisi le développement d'une application hybride avec « ionic ». D'abord, il est très difficile pour moi pour avancer car je ne connais pas du tout les langages utilisés par « ionic ». Et j'ai perdu mon temps pour chercher les cours pertinents de « ionic ». Donc je pense qu'il est une bonne idée de créer une base de données pour que les étudiants puissent trouver les manuels facilement.