Rapport du Projet de Techniques de Développemnt Logiciel

Joachim Collin
Tristan Martin

Proffesseur encadrant
Mathieu Bernard
Ecole Nationale des Ponts ParisTech

Champs-Sur-Marne, le 15.01.2022

1 Présentation du projet

Nous avons choisi dans le cadre de notre projet de techniques de développement logiciel de nous orienter vers le développement d'une application/site web pour répondre à un besoin que nous avions identifié lors de notre stage dans le domaine des Bâtiments et Travaux Publics. L'application développée a pour but d'automatiser le pointage des ouvriers de chantier. Nous partions d'un constat simple découvert grâce à notre stage. Actuellement, dans la plupart des chantiers, les heures de chaque ouvrier sont encore notées à la main par les chefs d'équipe, puis entrées dans des tableaux excels pour de nouveau être rentrées dans les logiciels de fiches de paie. Nous voulions donc optimiser ce processus en créant une application/site web ergonomique et facile d'utilisation pour améliorer la productivité et la fiabilité de ce processus. Nos objectifs ont connu de nombreux changements tout au long du projet.

L'objectif initial était de faire une application permettant aux chefs d'équipe de remplir les heures que ses ouvriers ont effectués chaque jour, ainsi que les différentes primes auxquelles ils avaient droit chaque jour en fonction des travaux effectués. Chaque ouvrier pouvait ensuite confirmer sur son compte ce qui a été rempli par son chef. S'il y avait conflit, le problème remontait au chef du chantier. Les heures devaient ensuite pouvoir être exportées sous forme d'un tableau excel par tous les modérateurs du chantier (chef de chantier, directeur financier et rh ...) que l'entreprise utiliserait directement pour créer les fiches de paie. Les ouvriers devaient donc tous avoir un compte personnel qui leur récapitulait également leurs informations et statistiques sur les chantiers où ils travaillaient, (type de travaux effectués, nombre d'heures travaillées...) et leur permettraient de valider les heures rentrées par le chef d'équipe.
Cependant, nous avons décidé de supprimer toute la partie ouvrier car aujourd'hui, les ouvriers sur les chantiers n'ont pas tous un téléphone avec internet et, après en avoir discuté avec un spécialiste du secteur, pas forcément la volonté de vérifier tous les jours leur compte. Le compte ouvrier a donc été abandonné. Pour la création d'un chantier sur l'application, nous avons uniquement besoin d'une base de données répertoriant les ouvriers sur le site et certaines de leurs informations personnelles. Cela pourrait permettre, par exemple, de leur envoyer directement leur fiche de paie tous les mois.

Chaque chantier est géré par un administrateur qui le crée et qui nomme des 'modérateurs' ayant des droits supérieurs à celui des chefs d'équipe (par exemple : ajouter et enlever des ouvriers) comme le chef de chantier, le directeur financier, le RH . Les administrateurs devaient également avoir accès aux informations personnelles de l'ensemble des employés présents sur leur chantier ainsi qu'aux informations sur les travaux réalisés. Cela devait leur permettre d'analyser de façon rapide l'avancement et les coûts sur un chantier et ainsi d'avoir une première approximation des dépenses en personnel réalisées jusqu'alors. En affectant les heures d'un ouvrier à une tâche, nous pensions être en mesure de déceler les activités moins productives que prévues pour chercher à comprendre l'origine des déficits de productivité. Le but étant de comprendre et de

corriger les erreurs faites soit dans la mise en place de l'évaluation des coûts, soit dans la gestion des problèmes rencontrés sur le chantier.
Nous avons commencé notre projet par l'élaboration d'un diagramme UML montrant l'organisation retenue pour la gestion de nos bases de données. Ce diagramme nous a permis de mieux visualiser et réfléchir sur comment nous voulions implémenter tous ces différents objectifs dans notre application. Ce diagramme a évolué tout au long de nos réflexions sur les bases de données que nous allions utiliser. Les deux diagrammes ci-dessous mettent en évidence certains des nombreux changements qui ont été effectués durant notre processus de réflexion.Ils montrent les données et les fonctionnalités dont nous allions avoir besoin :

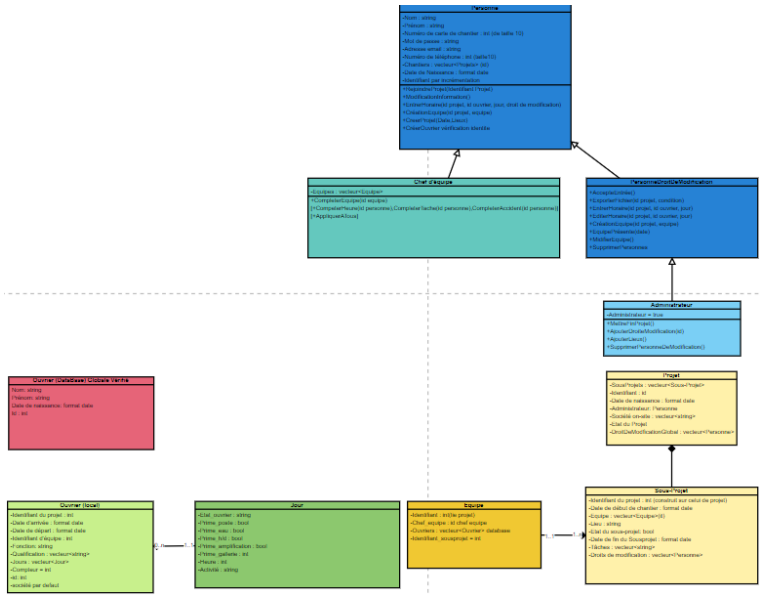

FIGURE 1 – Premier Diagramme ULM que nous avons réalisé

Au début du projet, les diagrammes que nous avons réalisés nous permettait plus de mettre au clair toutes les idées qui circulaient dans nos têtes plutôt que d'avoir un support bien réalisé et strictement correct (au sens UML).

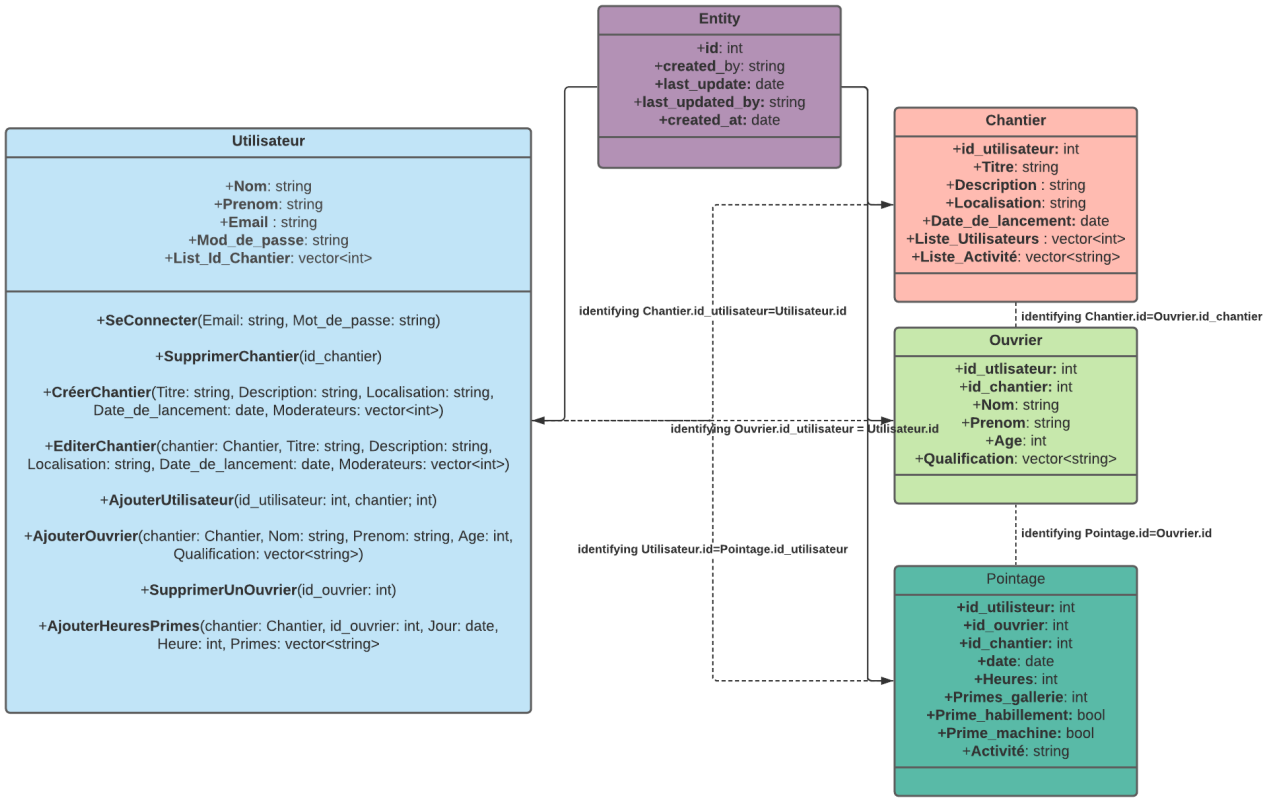

FIGURE 2 – Dernier Diagramme ULM que nous avons réalisé

C'est le dernier diagramme que nous avons réalisé. Il est beaucoup moins compliqué que les précédent et plus juste. Il nous a permis de nous fixer des objectifs à court-moyen terme. Il y a beaucoup d'autres fonctions et classes que nous voulions implémenter (compte Modérateur, compte Chef de chantier, gestion des équipes etc...). Mais en se reposant sur les principes de YAGNI et KISS, on se concentre sur faire une base "simple" et solide sur laquelle grandira l'application.

Après avoir exposé nos envies à notre professeur encadrant, celui-ci nous a alors naturellement orienté vers un projet de type Angular-flask car cela permettait en une fois de faire un code que nous pouvions facilement exporter en application. C'est cette combinaison de frameworks que nous avons utilisé. Notre application s'inscrit dans le cadre des applications de web 2.0. En effet, les utilisateurs ne sont pas seulement des consommateurs de contenu, ils peuvent interagir avec notre code pour que celui-ci répondent à leurs envies. Il s'agit de mettre en place plusieurs solutions pour calculer de multiples solutions avec ce problème.

2 Espérance versus Réalité

2.1 Espérance

Les caractéristiques finales que nous voulions développer pour notre applications étaient les suivantes :

- Système de registration et de connexion efficace permettant d'avoir accès au statut de l'utilisateur sur les différents chantiers.
- Gestion de différents rôles dans l'application avec des administrateurs, des modérateurs et des chefs d'équipe qui n'avaient pas les mêmes droits. Certaines fonctionnalités de l'application devaient être réservées aux responsables des chantiers.
- Création rapide et pertinente pour chaque chantier. Le formulaire de création devait être rapide et facile à prendre en main, tout en contenant l'ensemble des informations essentielles du chantier. Et avec ça, la possibilité de modifier les informations rentrées au préalable.
- Ajout des ouvriers à un chantier.
- Entrée facile et intuitive des heures et primes de pour chaque ouvrier.
- Exportation par le biais de l'application d'un excel qui récapitule l'ensemble des données sur les horaires faites par les ouvriers sur l'ensemble du mois.
- Analyse des heures répertoriées sur l'ensemble des ouvriers d'un chantier pour chaque tâche. Nous avions pour but de repérer les tâches productives et non productives par rapport au devis initial réalisé pour la société.
- Mise en place de test pour vérifier que les modifications ne perturbent pas les mécanismes déjà mis en place.
- Gestion des erreurs avec apparition de pop-up qui affiche un message personnalisé pour chaque erreur

2.2 Réalité

La réalité est bien différente de nos espérances initiales. Nous savions que notre projet allait être difficile pour des débutants en développement frontend et backend. Nous nous attendions à avoir beaucoup de mal à apprendre les langages et savions que nous n'allions pas atteindre nos objectifs avant la fin du projet TDLog. Nous nous ne nous attendions cependant pas à avoir autant de mal à uniquement commencer à pouvoir coder. Nous avons eu ainsi moins d'un mois pour développer réellement notre application, le reste a été de comprendre comment installer les différentes éléments d'angular-flask, les relier, apprendre à utiliser ces différents frameworks etc...

Les caractéristiques présentent dans notre application aujourd'hui sont :

- Système d'enregistrement et de connexion efficace permettant d'avoir accès au statut de l'utilisateur sur les différents chantiers.
- Création rapide et pertinente d'un chantier.
- Possibilité de modification simple des caractéristiques d'un chantier.
- Ajout d'ouvriers associé à un chantier avec un formulaire complet.
- Suppression d'ouvriers possible sur un chantier.

— Ajout d'heures de travail pour les ouvriers des chantiers.
— Mise de test simple pour vérifier le bon fonctionnement de l'application
— Un tout simple à prendre en main et ergonomique.

3 Présentation d'un point technique

Nous avons décidé de vous présenter le problème des API-services dans la relation entre backend et frontend. Il ne s'agit pas du problème le plus dur que nous ayons rencontré durant le développement de notre application. Les problèmes de registration sont sans aucun doute les plus durs mais nous n'avons pas encore totalement réussi à les gérer. C'est pourquoi nous avons décidé de vous présenter la relation entre le backend et le frontend avec les Api-services.
Notre premier objectif, après avoir réalisé la création de chantier, était de pouvoir cliquer dessus et accéder à une page qui dépend du chantier choisi. Il fallait donc pouvoir transmettre un id d'une page à une autre. Sur cette page, on avait besoin d'avoir accès aux caractéristiques de ce chantier..

3.1 Faire passer un id d'une page à l'autre

On a décidé pour faire ça d'utiliser des URL dynamiques. Dans le fichier "app.module.ts" on rajoute alors dans les chemins un path qui dépend un valeur "id" :

```
{path: 'chantier/:id',component : ChantierComponent}
```

Dans le component "InformationComponent", on ajoute dans le constructeur le module "ActivatedRoute" que l'on importe. On ajoute également dans ce component un attribut "id" correspondant à l'id du chantier initial. Lors de l'initialisation de cet attribut dans la fonction "ngOnInit()", on fait en sorte que cet id prenne la valeur transmise dans l'url :

```
ngOnInit(): void {
  this.route.queryParams.subscribe(params => {
    this.id = +this.route.snapshot.paramMap.get('id')});
```

3.2 Récupérer les caractéristiques du chantier. Difficultés.

Dans le component "InformationComponent", nous récupérons l'id du chantier d'origine de la manière expliquée précédement. Maintenant, nous cherchons à récupérer les caractéristiques de ce chantier. Pour cela, nous mettons en place un Api-service : "InformationApiService". Il s'agit d'un Injectable que l'on pourra injecter dans le component "InformationComponent" pour aller récupérer les caractéristiques du chantier. Dans ce service, on met en place une fonction getChantier() qui récupère tous les chantiers.

```
getChantiers()
{
    return this.http
        .get<Chantier[]>(`${API_URL}/chantiers`)
        .pipe(catchError(InformationsApiService.handleError));
}
```

Cette fonction fait une requête http au backend pour récupérer toute la base de données des chantiers et la mettre dans "Chantier[]", une liste de chantiers. L'URL "API-URL/chantiers" nous permet de relier cette requête au backend. Dans le component "InformationComponent", on crée un attribut "chantier : Chantier[]", une liste d'ouvrier que l'on initialise dans le fonction "ngOnInit()" :

```
this.chantiersListSubs = this.infosApi
    .getChantiers()
    .subscribe(res => {this.chantier = res[this.id-1];}, console.error);
```

On récupère ici tous les ouvriers grâce à la fonction "getOuvriers()" et ayant l'id du chantier, on sélectionne la id-ième donnée qui correspond à notre chantier. Au niveau du backend, on identifie la fonction grâce à l'URL, et on dialogue avec la base de données pour aller chercher tous les chantiers :

```
@blueprint.route('/chantiers')
def get_chantiers():
    # fetching from the database
    session = get_session()
    chantier_objects = session.query(Chantier).all()

    # transforming into JSON-serializable objects
    schema = ChantierSchema(many=True)
    chantiers = schema.dump(chantier_objects)

    # serializing as JSON
    session.close()
    return flask.jsonify(chantiers)
```

Le problèmes que l'on a eu avec cette méthode ont été les suivant :

— A chaque fois, on va chercher depuis le frontend tous les chantiers qui ont été créés. Dès que le nombre d'utilisateur de l'application est élevé, cette fonction ralentira inutilement le programme.

— Le gros problème que nous avons eu du mal à identifier est un problème d'id. En effet, si il l'on crée deux chantiers d'id 1 et 2 et que l'on supprime le premier, le chantier n°1 est supprimé de la base de données. Ainsi, si l'on récupère toute la base de données et que l'on sélectionne la deuxième ligne pour avoir le chantier 2, elle n'existe pas et la console nous renvoie une erreur.

3.3 Récupérer les caractéristiques du chantier. Résolution.

Pour régler ces problèmes, il fallait en fait aller chercher seulement le chantier voulu. Pour cela, il a fallu modifier la fonction "getChantier()" qui devient "getChantier(id-chantier)" et qui stock ce qui a été reçu non plus dans un liste de Chantiers mais un Chantier simple. Il faut ainsi faire passer l'id du chantier voulu au backend. Pour faire ça, on modifie l'URL de cette manière :

```
getChantier(id_chantier:number)
{
    return this.http
        .get<Chantier>(`${API_URL}/chantier/${id_chantier}`)
        .pipe(catchError(ChantiersApiService.handleError));
}
```

On modifie ainsi le backend en conséquence :

```
@blueprint.route('/chantier/<id_chantier>',methods=['Get'])
def get_chantier(id_chantier):

    # TODO ensure the chantier_id gives an existing chantier
    db = get_session()
    chantier = db.query(Chantier).filter_by(id=id_chantier).first()
    db.close()

    new_chantier = ChantierSchema().dump(chantier)
    print(new_chantier)

    return flask.jsonify(new_chantier), 201
```

Le problème est donc réglé : l'id présent est passé au backend qui le récupère et filtre la base de données pour renvoyer uniquement le chantier voulu.

4 Retour d'expérience

L'apprentissage de l'utilisation du framework Angular-Flask fut assez difficile pour nous deux, il nous a demandé beaucoup de temps et d'investissement pour découvrir le fonctionnement de ceux-ci. Totalement débutants dans ce domaine, nous attendions à rencontrer des difficultés lors de l'apprentissage de nouveaux langages de programmation. De nombreux concepts sont en effet apparus au fur et à mesure que nous découvrions ces langages ; pour Angular, des concepts comme les Observables, les Subscriptions, les Headers, les Services etc... pour Flask, les Schema de marshmallow, les requêtes aux bases de données, le lancement d'une application etc... Cela a considérablement retardé l'avancement de notre projet mais était une étape nécessaire, à la fois pour essayer de coder une application/site web de la manière dont une application est codée aujourd'hui et à la fois pour apprendre personnellement des bases importantes et

sérieuses que l'on va réutiliser après le module TDLOG. Cependant, nous avons été freinés très largement dans notre apprentissage de ces nouveaux langages et ne pouvions pas avancer normalement. Nous nous sommes confrontés à quelque chose que nous n'avions pas soupçonné : nous pensions qu'installer les différents éléments de l'application et les mettre en relation allait être simple et que nous n'avions qu'à apprendre les langages. En réalité, nous avons passé tellement de temps à régler des problèmes qui n'étaient pas liés à du code que l'apprentissage des langages est passé, durant la majeure partie du module, au second plan.

Le coût d'entrée dans le framework Angular est très important mais aussi très enrichissant. Nous pensons que vous pouvez continuer de recommander cette alliance Flask-Angular l'année prochaine tout en précisant bien que l'acquisition de compétences sur ce logiciel est compliquée et qu'il faut donc que les élèves soient motivés pour leur sujet. Peut-être serait-il intéressant de mettre en place un tutoriel d'installation et d'utilisation pour les élèves de TDLog l'année prochaine car nous trouvons que cette alliance est très puissante et que ce serait un gain de temps énorme pour les futurs élèves et gain de compétence important de savoir utiliser de tels frameworks.
Les choses qui nous ont manqué pour nous permettre d'avancer ce projet un petit peu plus est un encadrement plus important au début car l'installation d'Angular et de tous ses packages associés provoquait de nombreuses erreurs (et nous a empêché de nous concentré sur l'apprentissage des langages) qui pouvait être résolues en un rien de temps par des personnes compétentes et expérimentés. Notre encadrant Mathieu Bernard nous a beaucoup aidé mais nous ne pouvions pas lui envoyer un mail toutes les 5 minutes et une séance de 20 minutes toutes les semaines ne nous a pas semblé suffisante pour faire évoluer le code assez rapidement. Il serait intéressant de rajouter une deuxième séance pour avancer pas à pas, pour perdre moins de temps en étant bloquer.

Au début, nous avons suivi de multiples tutoriels pour apprendre les bases d'Angular. Ils consistaient au départ essentiellement en des tutoriels en anglais de quelques heures sur Youtube bien réalisés mais ne présentant que des bases "trop basiques" (les reste était payant) sur Angular et Flask séparément (et jamais ensemble) que nous n'arrivions pas à relier. Par la suite nous nous sommes tournés vers des tutoriels "Angular-Flask applications" et n'avons trouvé que des blogs relativement anciens et partant du principe que ses lecteurs étaient expérimentés dans ce domaine du développement logiciel et des vidéos indiennes en anglais sur Youtube qui n'étaient malheureusement pas très qualitatives et déjà obsolètes sur nos versions d'Angular et de Flask. Nous avions passé des heures sur Stackoverflow pour tenter de répondre à nos erreurs et demander de l'aide lors que nous en avions besoin. L'accumulation de nos recherches uniquement sur stackoverflow nous amènent à environ 500 de recherches sur ce site (durant tout le module TDLog), sans compter la consultation de plusieurs dizaines d'autres sites du même genre. L'avancée de ce projet fut un véritable parcours du combattant.
Chaque fois, que nous réussissions à faire marcher un tutoriel, la fusion du backend avec

le frontend entraînait un bug de notre code car nous ne comprenions pas ce que nous faisions et ne suivions seulement des listes de directives (que nous avons néanmoins essayé de comprendre à chaque fois). Comme à ce stade du projet, nous ne maîtrisions pas bien Github, nous n'arrivions pas à conserver une base stable sur laquelle se reposer et partir sereinement dans le développement. Il nous a fallu à chaque fois tout recommencer car souvent nous n'avions pas les mêmes erreurs pour le même code sur deux ordinateurs différents (manque de connaissance dans la gestion de l'installation des packages, des environnements virutels etc...).

Notre parcours chaotique mettait à mal notre patience et notre volonté commençait à s'atténuer. Fort heureusement, cette période coïncide parfaitement avec le moment où notre encadrant nous a transmis un projet où Flask/Angular étaient déjà reliés et qui reprenait un tutoriel de Auth0.com. Pour la petite histoire, nous avions déjà essayé de suivre ce tutoriel quelques semaines plus tôt. Cependant ce tutoriel utilisait le logiciel Docker que nous n'avions pas réussi à installer et qui également était assez complexe et complet mais partait du principe que ses lecteurs étaient expérimentés. Nous avions réalisé toutes les parties de ce (long) tutoriel mais cela n'avait fonctionné du tout.
Nous nous trouvions enfin en possession d'un code qui devait être fonctionnel. Il nous faudra malheureusement deux semaines entières pour le faire marcher. En suivant l'exécution comme indiquer dans le ReadMe, deux commandes (que nous ne comprenions pas) se situait sur la même ligne. Le terminal n'exécutant que la première commande, nous ne faisions pas tourné le backend. Avec du recul, nous pensons que la mise en place d'une démonstration pour montrer comment relier Angular-Flask et comment juste lancer un site serait la bienvenue pour les élèves de l'année prochaine.
L'ensemble des vidéos mises à disposition sur Educnet étaient très intéressantes pour nous car elles permettent de comprendre les concepts utilisés dans le développement logiciel sans rentrer dans leur implémentions . Typiquement, nous avons utilisé votre vidéo sur le parallélisme pour notre Projet MOPSI afin d'accélérer les calculs.

5 Conclusion

Nous savions que nous nous engagions dans quelque chose de difficile pour nous et que nous allions rencontrer beaucoup d'obstacles mais nous ne nous étions pas attendu à tel coût d'entrée. Néanmoins, nous ne regrettons pas d'être partie dans cette direction car nous avons gagné beaucoup d'expérience et de compétences importantes. Nous savons maintenant nous débrouiller et régler des problèmes informatiques beaucoup plus efficacement qu'avant et nous avons mis un premier vrai pas dans le monde du développement logiciel, ce qui nous a apporté des compétences cruciales pour les prochaines années. Nos résultats ne sont pas à la hauteur de nos attentes initiales, mais ce projet nous a permis de nous découvrir la complexité du domaine de développement logiciel.