IUT Paris Descartes
DUT Informatique

2018 - 2019

Développement Full Stack

Mission : Amélioration d'une Application Salesforce

Edifixio

123 rue Jules Guesdes, 93200 Levallois-Perret

Maître de stage : Imed Eddine Iaiche

Tuteur de stage : Mikal Ziane

Arsène Lapostolet

Groupe 202

Résumé / Abstract

Dans le cadre de mon stage de fin d'étude en DUT informatique, j'ai rejoint l'entreprise Edifixio en tant que développeur *full stack*. Je fais partie l'équipe ESIA afin de travailler sur un projet interne, la mise à jour de l'application Tock, permettant aux employés de la société de renseigner leur temps de travail et aux instances de pilotage d'établir des *reportings* à partir de ces données.

Ma mission était donc dans un premier temps de procéder à un *refactoring* du code *back end* de l'application afin de l'optimiser et d'améliorer sa conception en y appliquant des patrons de conception. Ensuite, j'ai été amené à redévelopper l'interface utilisateur afin de la rendre moderne et *responsive*.

Ce projet correspond à la volonté des différents utilisateurs de l'application de disposer d'amélioration concernant l'ergonomie ainsi la modernité de son interface graphique. De plus, pour les utilisateurs en déplacement, le besoin s'est fait sentir de disposer d'une version mobile de l'application. Tock étant une application Salesforce (voir ci-dessous), j'ai découvert cet outil. Salesforce est un progiciel de gestion intégré orienté gestion de la relation client.

Cette mission m'a permis d'acquérir de l'expérience dans le développement sur la plateforme Salesforce : le langage APEX en *back end* et Javascript en *front end*, avec le *Framework Lightning Web Component* et SOQL pour la base de données. J'ai utilisé les IDE (Environnement de Développement Intégré) IntelliJ IDEA Community pour le *back end* et Visual Studio Code pour le *front end*.

Mots clé
Salesforce, Cloud, *Full stack*, PaaS, APEX, Lightning Experience, SOQL, CRM

My end-of-study internship led me to join Edifixio as a *full stack* developer. I am part of the ESIA team in order to work on an internal project, Tock. Tock is an application that allows each employee to fill in his working timesheet and the financial team to pull reportings out of this data.

My mission was first to refactor the app's *back end* in order to optimize its algorithmic and maintainability by applying patron de conceptions. Then, I reworked the *front end* to adapt its look and feel to contemporary expectations.

This project has been initiated because several user expressed the need a more modern and intuitive user interface. In addition, employees who were on business trip required a mobile version of the app. Tock is a Salesforce app, so I had to discover that tool. Salesforce is a customer relationship oriented integrated management software package.

This mission allowed me to acquire experience with Salesforce development : APEX in the *back end*, JavaScript with Lightning Web Component in the *front end* and SOQL for the database. I used several IDE : IntelliJ IDEA for APEX, VS Code for JavaScript.

Remerciements

Je voudrais remercier toutes les personnes qui m'ont permis d'effectuer ce stage et qui ont contribué à son bon déroulement.

J'aimerais tout d'abord remercier l'ensemble des enseignants qui ont eu l'occasion de m'accompagner durant mon cursus à l'IUT Paris Descartes, en particulier Mikal Ziane, dont l'enseignement en conception logiciel et en programmation orientée objet m'a été particulièrement utile pendant ce stage. Je remercie également Monsieur Ziane pour son suivi efficace durant le déroulement du stage.

Je remercie aussi les personnes de Edifixio qui m'ont suivi tout le long du stage, Achraf Dhouib, Imed Eddine Iaiche et Matthieu Cuenin pour leur accompagnement et leurs conseils, mais également toute l'équipe ESIA qui m'a accueilli chaleureusement.

Je remercie également la communauté des développeurs sur le web qui par sa capacité d'entraide notamment au travers de plateformes comme Stackoverflow nous permet de résoudre nos problèmes.

Introduction

L'application Tock existait déjà et fonctionnait depuis 2014. Cependant, le besoin s'est fait sentir par les utilisateurs d'accéder à l'application depuis un mobile, ce qui n'était pas possible dans sa version actuelle, l'application n'ayant pas été pensée avec un design responsive. De plus, le design général de l'application commençait à être daté et avait besoin d'un rafraîchissement. Aussi, le code, à la fois *back end* et *front end*, devenait de plus en plus complexe à maintenir et à mettre à jour, c'est pourquoi il devait être entièrement restructuré. Ainsi, l'objectif de mon stage en tant que développeur *full stack* était de réorganiser le code *back end* et réécrire le *front end* avec des technologies plus récentes.

Dans la première partie du rapport, je détaille l'évolution d'Edifixio ainsi que le fonctionnement de ses différents services. Ensuite, je décris le contenu du stage, le travail réalisé, les difficultés auxquelles je me suis heurté, ainsi que mon contexte de travail.

Sommaire

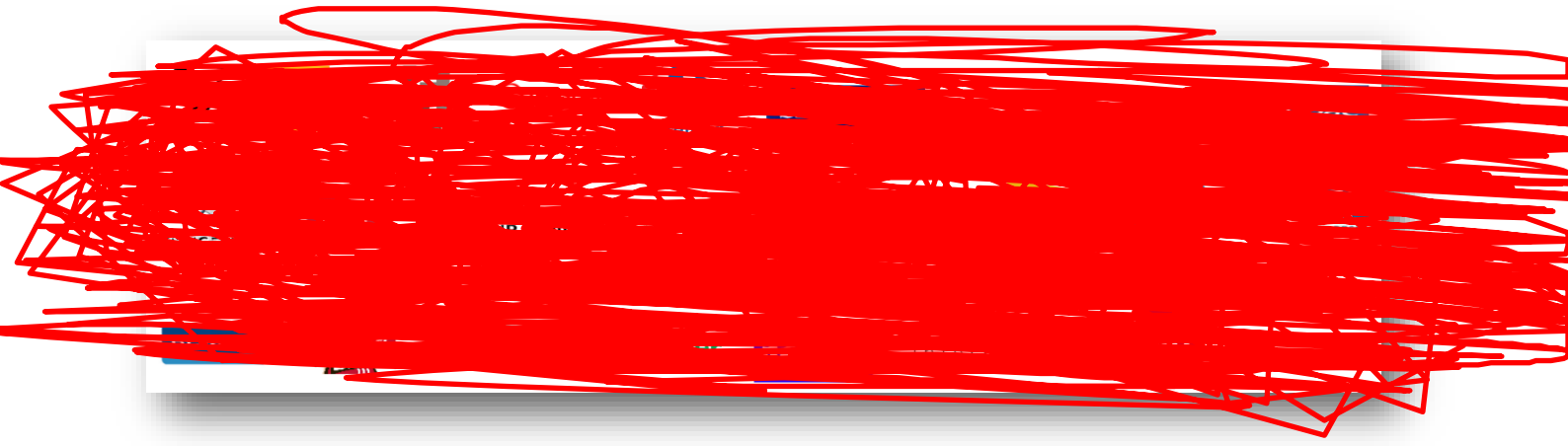
I/ Présentation d'Edifixio

Edifixio est une entreprise de service numérique de taille moyenne. Elle possède environ trois cents collaborateurs dans ses locaux de Levallois et Grenoble mais également des équipes à l'étranger, aux Etats Unis, en Inde et en Tunisie. Le secteur de consulting numérique et du service en Ingénierie Informatique est très important dans l'économie française et son poids croît chaque année. Au sein d'Edifixio, l'équipe ESIA, dirigée par Thierry Humbert, a pour spécialité l'intégration progiciel et le cloud. C'est dans l'équipe Salesforce, au sein de l'équipe ESIA, mon stage se déroule du 8 Avril 2019 au 30 Août 2019. Durant ce stage, j'ai travaillé seul sous la direction de mon tuteur, Imed Eddine Iaiche, et de l'utilisateur principal du projet, Matthieu Cuenin dans les locaux principaux d'EdifXio à Levallois-Perret au Nord-Ouest de Paris.

Edifixio est une SASU (Société par actions simplifiée à associé unique) au capital de 22 000€ fondée en 2000. Le métier d'Edifixio est de concevoir et d'exploiter des applications connectées. Son siège social se situe au 123 rue Jules Guesde à Levallois-Perret.

1.1/ Activités Principales

Avec son chiffre d'affaire de 26M€ en 2018, Edifixio est une entreprise de services numériques très compétitive. Elle a su convaincre de nombreux clients :



Les principaux clients d'Edifixio

L'entreprise intervient sur les grands thèmes que sont :

- Portails, intranets, extranets et sites internet
- CRM (Client Relationship Management), *marketing automation* et service client
- Développement sur plateforme cloud
- Les solutions collaboratives

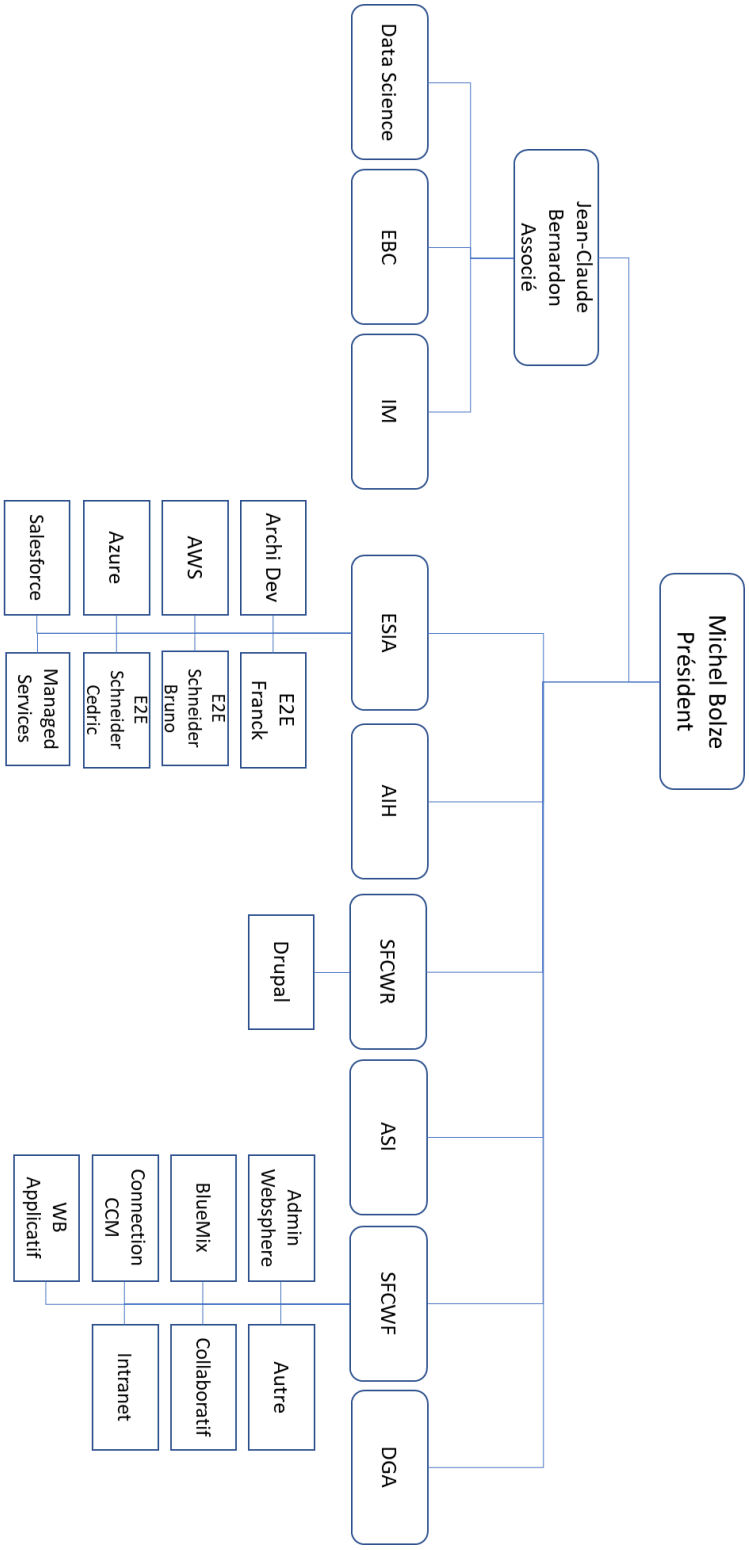
EdfiXio emploie 300 collaborateurs à travers le monde, répartis dans de nombreux locaux à l'international :

La répartition des collaborateurs d'Edifixio à l'international

1.2/ Histoire d'Edifixio

Date	Evenement
2000	Création d'Edifixio, avec le groupe Lafarge comme partenaire industriel.
2001	Partenariat ILOG (Configuration et Optimisation)
2002	Partenariat Autonomy (Moteur de recherche)
	Ouverture d'Edifixio Grenoble
2003	Partenariat IBM (MDM et solutions collaboratives)
2005	Ouverture d'Edifixio Boston
2007	Projet Connexion Danone (Premier réseau d'entreprises connectées en France)
2009	Projet Lafarge (Première migration de masse vers AWS en France)
2011	Ouverture du centre dédié support à Calcutta
2012	Partenariat AWS
	Lancement de l'offre de services managés
	Lancement de l'activité Salesforce
2014	Lancement de l'activité sur Drupal
2015	Partenariat Salesforce
	Nouveau Partenariat AWS
2016	Ouverture d'Edifixio Tunis
2018	Certification Edifixio AWS DevOps

1.3/ Organigramme structurel



Organigramme Structurel

1.4/ Partenaires

Edifixio a également des partenariats avec de nombreux acteurs majeurs du monde numérique et du Cloud :

Salesforce est la solution de référence dans le domaine du CRM communautaire dans le Cloud. Investi dans l'écosystème Salesforce depuis 2010 et devenu *Salesforce Silver* en 2015, Edifixio détient une haute expertise dans la conception et la livraison de programmes basés sur cette plateforme.

IBM Cloud propose des services de transition numérique et d'intelligence artificielle basée dans le Cloud. En tant qu'*IBM Premier Business Partner* depuis 2003, Edifixio est expert en livraison de solutions reposant sur le Cloud IBM.

AWS est la plateforme de cloud à la demande créée par Amazon. *Premier Consulting Partner* de AWS depuis 2010 Edifixio détient une expérience significative dans la livraison d'infrastructures et applications cloud pour ses clients.

Azure est la plateforme cloud créée par Microsoft. EdifXio est expert en technologies Microsoft depuis 2000. Ses titres de *Silver Partner on Cloud Platform and Application Development* depuis 2015 et de *Gold Partner on Cloud Platform* depuis 2016 témoignent de sa capacité à concevoir des solutions basées sur le cloud Azure.

Mulesoft est la référence en matière de gestion du cycle de vie des API. Edifixio est positionné pour devenir un partenaire de Mulesoft afin de bénéficier de la synergie entre Salesforce et Mulesoft pour améliorer encore plus la performance

1.5/ Le pôle ESIA : Intégration et Cloud

Durant mon stage, j'ai intégré le pôle ESIA dont les métiers sont l'intégration et le cloud. Il se spécialise dans plusieurs activités, notamment :

- Full Management Services
- Salesforce
- Amazon Web Services
- Microsoft Azure
- Support exploitation 24/7
- Développement Open Source

Ce pôle est composé de 200 collaborateurs.

Les principaux clients du pôle ESIA sont :

Les principaux clients du pôle ESIA

1.6/ L'équipe Salesforce

L'équipe Salesforce, évoluant au sein du pôle ESIA, développe des solutions basées sur la plateforme Salesforce. Cela comprend l'analyse des besoins, le paramétrage, l'automatisation des processus métiers et de développement de solutions logicielles personnalisées pour les entreprises. L'équipe comprend deux types de collaborateurs. Tout d'abord, les Business Admins, spécialistes en fonctionnement des organisations, utilisent Salesforce par le biais de son interface graphique et créent ainsi tout ce dont une entreprise a besoin pour que son instance de Salesforce lui soit parfaitement adaptée, comme des objets personnalisés, des *flows*, etc … Cependant, certains besoins très précis ne peuvent être accomplis à travers l'interface graphique de Salesforce et nécessitent du développement applicatif sur la plateforme Salesforce. C'est pourquoi l'équipe comprend également un certain nombre de développeurs, qui s'occupent de créer ces applications personnalisées.

II/ Le projet Tock

2.1/ L'application Tock

Tock est une application web qui permet d'automatiser le processus de décompte du temps de travail des employés. Chaque semaine, les employés d'Edifixio renseignent sur Tock leur feuille de temps de la semaine. C'est-à-dire qu'ils vont signaler, pour chaque jour, quelle fraction de leur temps de travail a été consacrée à chaque projet. Il leur est également possible de créer des heures supplémentaires et des astreintes, qui doivent être ensuite validées par un manager.

Ce processus génère une grande quantité de données concernant le coût des projets, la performance des équipes, etc … Tock permet ainsi aux instances de pilotage de la société de consulter ces données importantes pour la stratégie de l'entreprise.

2.2/ Environnement de travail

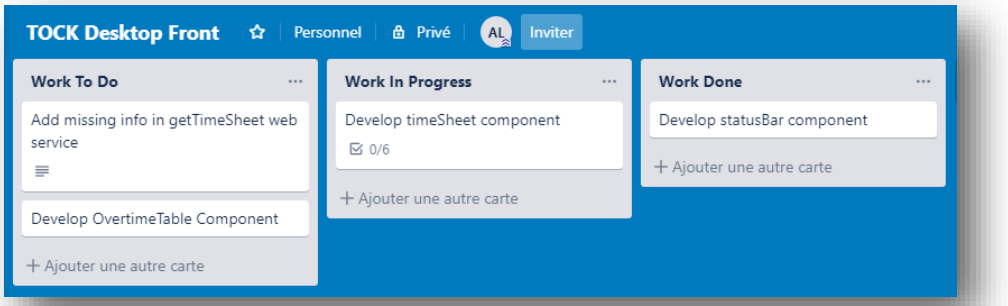
Durant mon stage, j'ai travaillé en autonomie sur le projet Tock. J'avais à ma disposition un ordinateur portable performant ainsi qu'un poste de travail doté d'une connexion internet performante, d'un grand écran ainsi que d'une chaise de bureau confortable. Ce matériel ergonomique m'a permis de me concentrer sur mon travail dans un confort physique et pratique irréprochable.

Au cours de mon travail, j'ai été amené à interagir avec certains membres d'Edifixio. Tout d'abord, Imed Eddine Iaiche, mon tuteur de stage a pris les décisions d'architecture pour le projet. Il était également là pour me guider et répondre à mes interrogations. J'ai également pu bénéficier de l'aide de Clément Tuhault, un web designer, qui a créé les maquettes de l'interface graphique du projet Tock. Enfin, Matthieu Cuenin, en tant qu'administrateur de Tock, a dirigé mon travail en écrivant les spécifications de l'application.

Pour développer sur l'application Tock, je déployais mes changements sur une organisation Salesforce dites « Sandbox », une organisation de test, non critique, permettant de vérifier le bon fonctionnement de ce que je développais, sans risque d'impacter la version de production de l'application. Ensuite, quand un certain nombre de changements avaient été validés, il fallait effectuer la livraison en production des changements. Cela consistait à déployer ces changements sur l'organisation de production afin qu'il profite enfin aux utilisateurs.

2.3/ Organisation du Travail

L'organisation de mon travail à beaucoup évoluée au cours de différentes phases du projet. De manière générale, j'ai moi-même structuré mon travail de façon agile, afin d'organiser au mieux mon travail. J'ai tenu mon *back log* sur Trello.



Tenue du *back log* du projet sur Trello

De plus, afin de bien séparer les différentes fonctionnalités sur lesquelles je travaillais, j'ai, pour chaque ticket ou fonctionnalité, créé une branche sur mon système de gestion de version. J'ai ainsi utilisé un *flow git* très classique, avec une branche master, une branche dev et une branche dédiée pour chaque fonctionnalité.

2.4/ Objectif

L'objectif de stage est de revoir entièrement la conception et l'implémentation de l'application Salesforce Tock. De nombreuses personnes ont contribué à l'ancien code l'application, ce qui le rend très peu lisible, pas documenté et assez fouillis. Ainsi, au niveau *back end* le but de procéder à un *refactoring* du code afin de le rendre propre, lisible, robuste, maintenable et documenté, en lui appliquant différents patrons de conception tels que MCCV (Modèle Contrôleur – Contrôleur Vue) et DTO (Data Transfert Object).

Du point de vue du *front end*, il est prévu de le réécrire entièrement avec des technologies et une interface graphique plus moderne et ergonomique.

2.5/ Technologies utilisées

Salesforce : Salesforce est un progiciel de gestion intégré orienté gestion de la relation client fourni sous la forme d'une plateforme cloud PaaS (Plateform as a Service). Salesforce se présente sous la forme d'une base de données configurée par défaut avec un certain schéma orienté gestion de relation client, avec des tables tels que *Lead, Opportunity, Account,* … Mais il est possible d'étendre ce schéma personnalisé et de l'exploiter grâce à de nombreux outils graphiques (interface dite point & click) créés pour être utilisés par du personnel non technique.

APEX : APEX est un langage de programmation orienté objet, compilé et exécuté pour la plateforme Salesforce. Il est destiné à développer de la logique serveur sur la plateforme pour adapter la plateforme aux processus métiers et de gestion d'une entreprise.

APEX est (comme Salesforce) implémenté en Java. Syntaxiquement, il ressemble beaucoup à ce dernier, à ceci près que la résolution des liens de méthode se fait de façon statique et non dynamique, on retrouve donc le mot clé *virtual* hérité de C++. Il est également possible d'exécuter des instructions dites DML (Data Manipulation Langage), come *insert, update, upsert, delete* permettant d'effectuer des opération CRUD sur la base de données Salesforce via un mécanisme de mapping objet-relationnel. Ce mécanisme permet également d'exécuter des requêtes SOQL directement dans le code APEX, leur résultat étant directement retourné sous forme de liste.

Lightning : Lightning correspond à la version la plus récente de l'interface graphique de Salesforce, par opposition à Salesforce Classic, qui propose un design plus daté et obsolète. Lightning propose également des Frameworks Javascript *front end* permettant de développer des interfaces au « *look and feel* » en accord avec celui de Salesforce Lightning.

SOQL : Salesforce Object Query Language est un langage de requête de base de données ; inspiré du SQL, il permet d'effectuer des requêtes de sélection dans le SGBD de Salesforce. Il ne propose pas d'instructions de modification de donnée étant donné que ces dernières sont effectuées via les mots clés *insert, upsert, update* et *delete* du langage APEX.

LWC : Lightning Web Component est le Framework Lightning le plus récent . Sorti en décembre 2018, il est également orienté composants, mais propose une structure beaucoup plus moderne pour le développement de ces derniers. En effet, la définition du markup se fait dans un fichier HTML natif. Une classe javascript permet, grâce à ses propres attributs, de définir les attributs du composant, ainsi que les gestionnaires d'événement de l'interface graphique. Cela permet de définir des composants de façon beaucoup moins verbeuse qu'avec Aura, plus proche du natif, et donc avec de meilleures performances, en exploitant les capacités de la spécification ECMAScript 6.

Aura : Aura Components est le Framework Lightning le plus ancien. Il est orienté composants. La structure d'un composant comprend un fichier de markup .cmp qui permet de décrire l'aspect graphique du composant et ses attributs grâce à du XML. Ensuite, des fichier javascript stockent des tableaux de gestionnaires d'événement pour les différents événements déclenchés dans le markup.

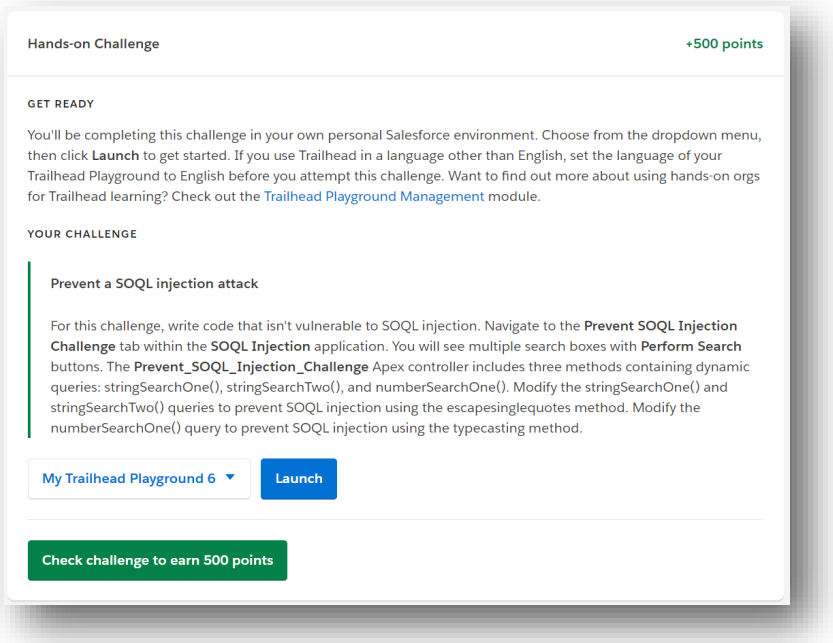
LSDS : Lightning Design System est un Framework CSS qui propose des classes CSS et des icônes permettant d'implémenter facilement la charte graphique de Salesforce dans une application. Il possède toutes les fonctionnalités d'un Framework CSS.

Git : Git est un système de gestion de version. Un système de gestion de version permet de garder un historique de l'ensemble des modifications apportées à un projet. Il permet également grâce à un système de branches, de maintenir plusieurs *workflows* en parallèle. Git permet également via un serveur distant d'assurer le partage des sources entre les différents contributeurs d'un projet.

2.6/ Déroulement du projet

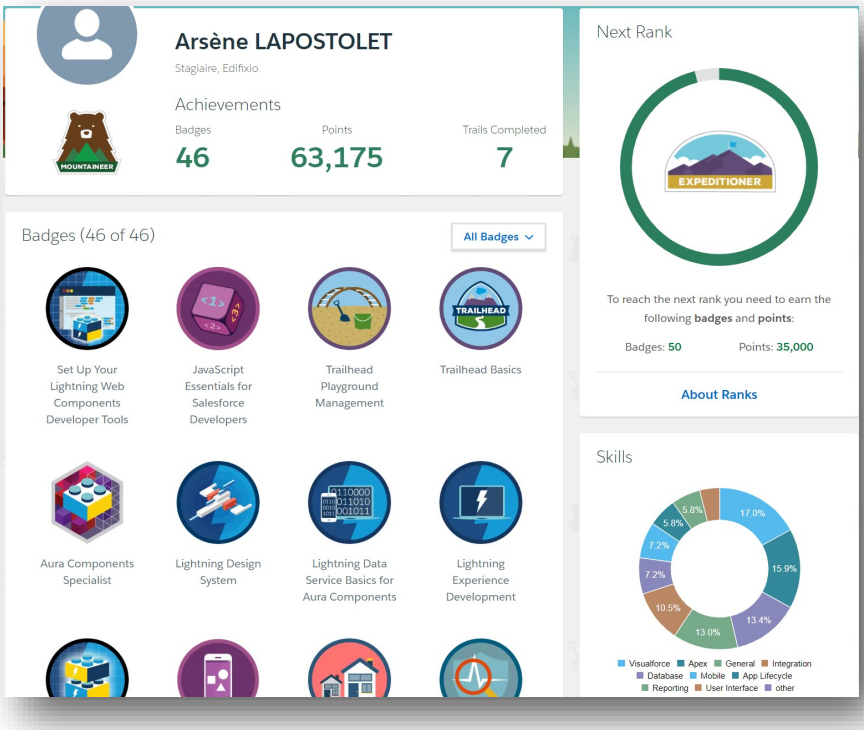
2.6.1 Formation

Avant de commencer ma mission de stage, il me fallait me former sur la plateforme Salesforce afin d'acquérir les compétences nécessaires à la réalisation du projet. J'ai donc effectué des tutoriels sur la plateforme Trailhead. Trailhead est une plateforme web d'e-learning mise à disposition par Salesforce permettant de développer ses compétences avec les différents outils de la plateforme. Cette dernière, très bien pensée, permet d'accéder à des instances salesforce dédiées afin de réaliser les tutoriels. Le système va ensuite vérifier que les étapes de ces derniers sont bien réalisées avant de nous laisser poursuivre.



Défi Trailhead

De plus, la progression de l'apprenant sur la plateforme est rendue plus motivante par l'introduction de gamification. En effet, lorsque l'on termine un défi, on obtient des points, et lorsqu'on termine un module de cours, on obtient un badge correspondant à ce module.



Mon profil sur Trailhead

J'ai eu l'occasion dans un premier temps de terminer les modules dits « admin » mettant en œuvre des outils graphiques afin de découvrir le fonctionnement et les bases de Salesforce. Je me suis ensuite consacré aux modules « developer » qui concernent le développement logiciel sur la plateforme Salesforce. J'ai donc appris la programmation en langage APEX pour le *back end* et Javascript dans le Framework Lightning pour le *front end*. J'ai appris à relier mon environnement de développement à une organisation (instance) Slaesforce via l'interface en ligne de commande *SFDX*.

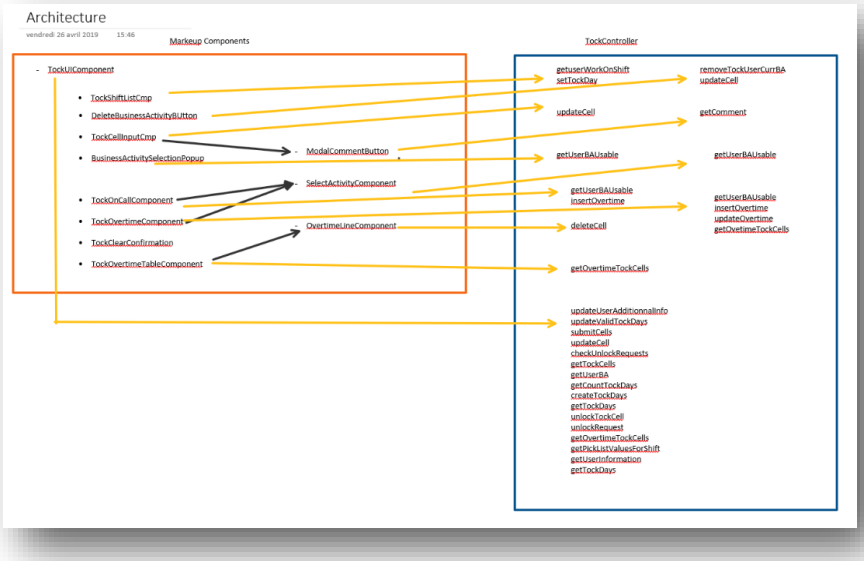
En APEX, j'ai pu développer du code interagissant avec le modèle de donnée de Salesforce via des requêtes SOQL (Salesforce Object Query Language). J'ai appris à utiliser le Framework de test unitaire d'APEX, ainsi que la notion de code *coverage* pour les tests. J'ai également dû développer des *mocks* pour tester notamment le code utilisant des requêtes HTTP. J'ai pu aussi travailler sur le développement d'API REST en APEX ainsi que de services concurrents exploitant le parallélisme.

Côté client, j'ai eu l'occasion de créer des pages VisualForce et leurs contrôleurs APEX de même que des Composants Lightning Web et des Composants Lightning Aura.

Ainsi, au cours de mes deux semaines de formation, j'ai terminé les « Trails » suivant : « Admin Beginner », « Admin Intermediate », « Developer Beginner », « Developer Intermediate ». J'ai aussi complété le super badge « Aura Components Specialist ».

2.6.2 Découverte de Tock

Après avoir fait environ deux semaines de formation, j'ai travaillé à de petites modifications demandées par l'administrateur sur l'application pour me familiariser avec les différents composants du logiciel à la fois sur le *front end* et sur le *back end*. J'ai également fait différents schémas afin de mieux me repérer dans le code.



Ancienne Architecture du *front end* de Tock

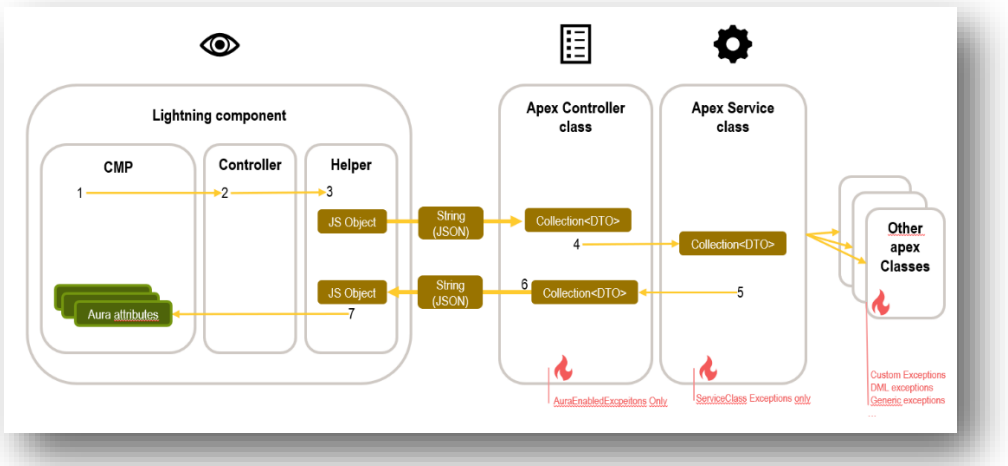
Des tickets étaient donc créés par Matthieu Cuenin sur une feuille Google Sheets, décrivant des bugs ou des demandes de nouvelles fonctionnalités. Les changements que j'ai effectués sont par exemple le changement de la méthode de verrouillage de la feuille de temps de la semaine. En effet, auparavant, l'utilisateur devait, après avoir saisi ses temps, verrouiller sa feuille. Cela était répétitif et contraignant. J'ai donc implémenté un batch (exécutions asynchrones de code sur un grand nombre de lignes de la base de données, paquet par paquet, par défaut des paquets de 200 lignes) ; ce dernier est exécuté tous les dimanches soir, ce qui verrouille les feuilles de temps de la semaine. Afin de créer ce batch, j'ai dû implémenter dans un premier temps l'interface *Database.Batchable* afin de pouvoir traiter par paquet les résultats d'une requête. J'ai aussi dû implémenter l'interface *Schedulable* afin de pouvoir indiquer à la plateforme de l'exécuter périodiquement, un certain jour de la semaine. Il m'a ensuite fallu développer les tests unitaires de ce batch et créer des données de test.

J'ai également réglé quelques bugs et introduit des changements dans le *front end* pour améliorer l'ergonomie.

2.6.3 Refactoring du Back end

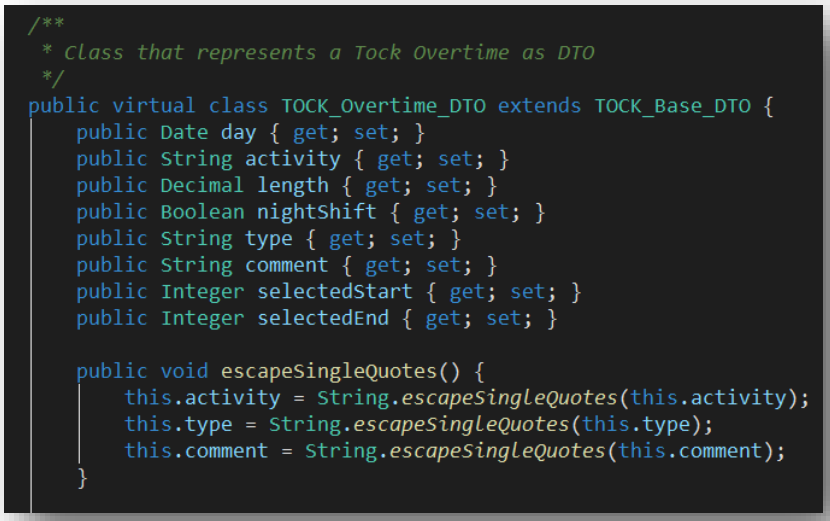
J'ai ensuite travaillé sur le *refactoring* du *Back end* de l'application selon le patron de conception DTO (Data Transfer Object) afin de normaliser les échanges client-serveur dans l'application par le biais du format JSON ainsi que de découpler la logique métier du contrôleur (classe qui fait office d'interface entre la logique d'application et les actions de l'utilisateur).

Ainsi, pour chaque *Webservice* APEX, j'ai dû appliquer le patron de conception Data Transfer Object :



Le patron de conception *Data Transfer Object*

Suivant ce patron de conception, le dialogue entre le *front end* et le *back end* se fait via un *webservice*, acceptant et retournant des objets sous forme de JSON. Le rôle du contrôleur, contenant ces *webservices* est dans un premier temps de définir la structure des différents DTOs par le biais de classes internes.

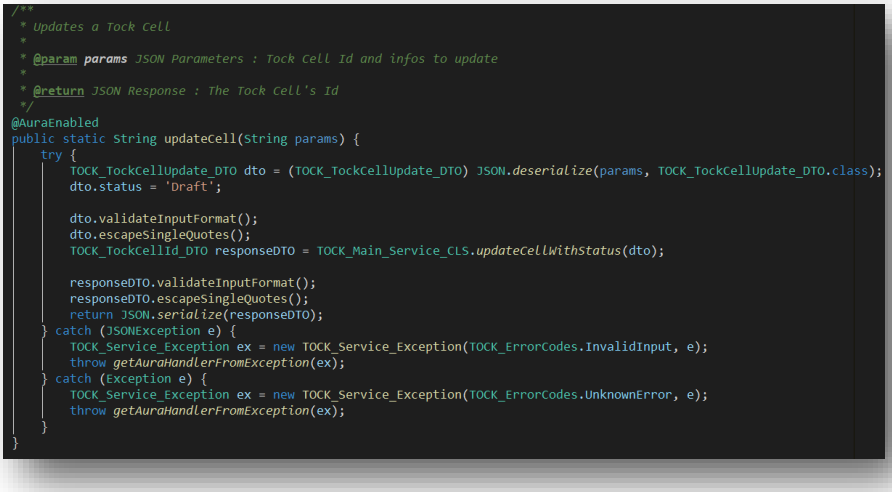


```apex
/**
 * Class that represents a Tock Overtime as DTO
 */
public virtual class TOCK_Overtime_DTO extends TOCK_Base_DTO {
    public Date day { get; set; }
    public String activity { get; set; }
    public Decimal length { get; set; }
    public Boolean nightShift { get; set; }
    public String type { get; set; }
    public String comment { get; set; }
    public Integer selectedStart { get; set; }
    public Integer selectedEnd { get; set; }

    public void escapeSingleQuotes() {
        this.activity = String.escapeSingleQuotes(this.activity);
        this.type = String.escapeSingleQuotes(this.type);
        this.comment = String.escapeSingleQuotes(this.comment);
    }
```

Une classe DTO

Ces classes contiennent des méthodes permettant de sécuriser les entrées mais en aucun cas de la logique afin de limiter le couplage avec cette dernière. Il s'agit ensuite pour chaque webservice correspondant à un point d'entrée de l'API de récupérer le paramètre JSON sous forme de chaîne de caractère, puis de *parser* cette dernière pour obtenir un objet DTO correspondant. Le *webservice* va ensuite appeler la méthode de la classe de service qui lui correspond. Cette dernière contient toute la logique d'application. Le *webservice* s'occupe également de relayer les exceptions de façon normée afin qu'en cas d'erreur, on puisse présenter le problème à l'utilisateur de la façon la plus lisible possible.



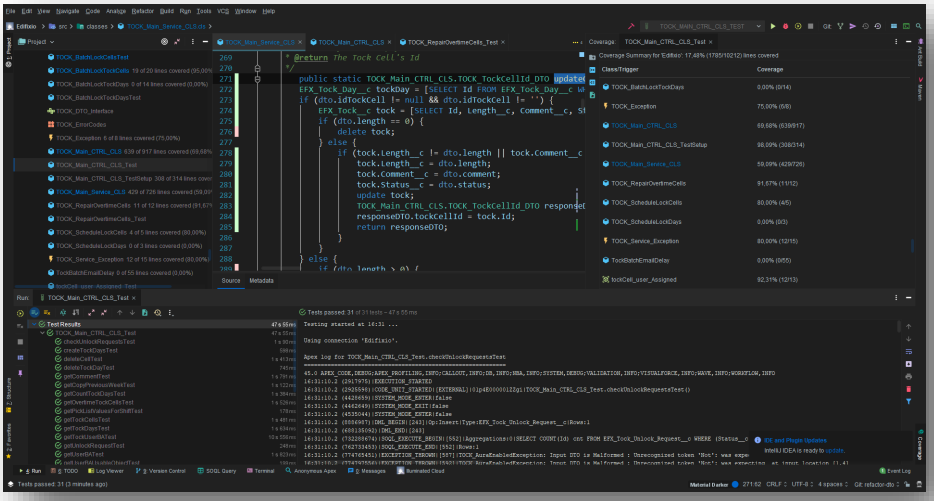
```apex
/**
 * Updates a Tock Cell
 *
 * @param params JSON Parameters : Tock Cell Id and infos to update
 *
 * @return JSON Response : The Tock Cell's Id
 */
@AuraEnabled
public static String updateCell(String params) {
    try {
        TOCK_TockCellUpdate_DTO dto = (TOCK_TockCellUpdate_DTO) JSON.deserialize(params, TOCK_TockCellUpdate_DTO.class);
        dto.status = 'Draft';

        dto.validateInputFormat();
        dto.escapeSingleQuotes();
        TOCK_TockCellId_DTO responseDTO = TOCK_Main_Service_CLS.updateCellWithStatus(dto);

        responseDTO.validateInputFormat();
        responseDTO.escapeSingleQuotes();
        return JSON.serialize(responseDTO);
    } catch (JSONException e) {
        TOCK_Service_Exception ex = new TOCK_Service_Exception(TOCK_ErrorCodes.InvalidInput, e);
        throw getAuraHandlerFromException(ex);
    } catch (Exception e) {
        TOCK_Service_Exception ex = new TOCK_Service_Exception(TOCK_ErrorCodes.UnknownError, e);
        throw getAuraHandlerFromException(ex);
    }
}
```

Un *webservice* APEX

J'ai également dû rédiger des tests unitaires des classes nouvellement créées. Ces tests unitaires devaient pour certains *webservices* être très précis et, par exemple, émuler les droits de profils d'utilisateurs particuliers. De plus, les tests unitaires pour les classes APEX doivent assurer un code coverage d'au moins 75%. C'est-à-dire que 75% des lignes du code de la classe doivent être exécutées par le test.



La vue du mode test d'IntelliJ IDEA

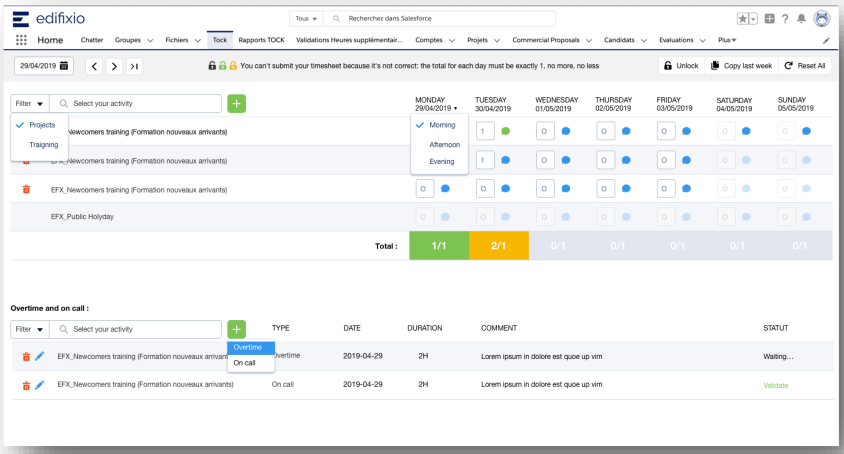
Ainsi, mes tests unitaires sont structurés pour distinguer clairement, pour chaque méthode de tests, la création des données de tests, l'exécution du test, et la vérification des contraintes après son exécution.

2.6.4 Optimisation de l'architecture

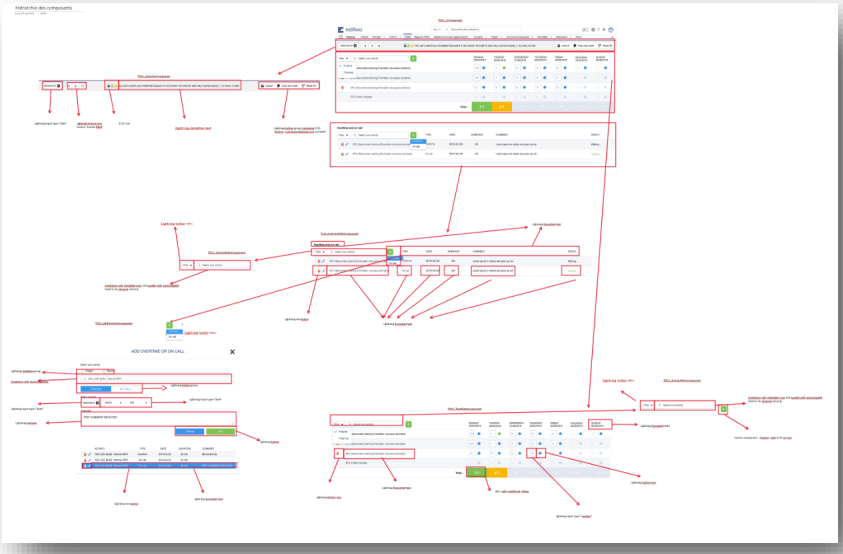
A l'origine, un certain nombre de règles de gestion de la feuille de temps étaient implémentées coté *front end*, dans un unique fonction JavaScript de plus 500 lignes. Ce code était non documenté et donc très difficile à maintenir et peu performant. Ainsi, dans un objectif d'améliorer la lisibilité et la maintenabilité de l'application ainsi que de diminuer le temps de chargement initial de l'application, nous avons décidé de réimplémenter ces règles de gestion en *back end*, dans un web service APEX. Il m'a donc fallu comprendre l'algorithmique de cette méthode et l'implémenter en APEX. J'ai ensuite testé que le JSON renvoyé par ce service Web était cohérent avec les données existantes.

2.6.5 Développement du nouveau *Front end*

Dans un premier temps j'ai prise connaissance de la maquette créé par Clément Tuhault, un webdesigner. J'ai ensuite réfléchis à l'architecture des composants de l'application. Mon objectif était de prototyper cette architecture de la façon le plus précise possible afin d'avoir des objectifs clairs au cours du développement. J'ai également cherché à utiliser au maximum les composants fournis par le biblioth_que standard Lightning.



Nouvelle maquette de l'application



Nouvelle architecture du *front end*

Je me suis ensuite documenté et entrainé sur le nouveau Framework Lightning Web Component. Le développement dans Lightning Web Component m'a rappelé le développement avec Angular.

En effet, chaque composant possède une classe javascript contrôleur permettant de mettre en place le *data binding* et de faire appel à des *webservices*. Chaque composant dispose également d'un *template* HTML dans lequel on définit l'apparence via des balises HTML mais aussi d'autres composants LWC, ainsi que le *binding* des données dans l'interface et les gestionnaires d'événements.

```
import TOCK_Status_NotCorrect from '@salesforce/label/c.TOCK_Status_NotCorrect';
import TOCK_Status_Unlocking from '@salesforce/label/c.TOCK_Status_Unlocking';

export default class StatusBar extends LightningElement {

    // Button Labels
    resetAllLabel = TOCK_ResetAll;
    unlockButtonLabel = TOCK_Unlock_Button;
    copyLastWeekLabel = Tock_CopyLastWeek;

    handlePreviousWeek(){
        this.dispatchEvent(new CustomEvent("previous"));
    }

    handleNextWeek(){
        this.dispatchEvent(new CustomEvent("next"));
    }

    handleCurrentWeek(){
        this.dispatchEvent(new CustomEvent("current"));
    }

    handleUnlock(){
        this.dispatchEvent(new CustomEvent("unlockrequest"));
    }
```

Un *Lightning web component*

```
    @track
    statusMessage = TOCK_Status_Complete;

    @track
    icon = "utility:unlock"

    @track
    iconVariant = "";

    @api
    get status(){
        return this.status;
    }

    set status(value){
        if(value === "Locked"){
            this.statusMessage = TOCK_Status_Locked;
            this.icon = "utility:lock";
            this.iconVariant = "success";
            this.status = value;
        }
        else if(value === "Complete"){
            this.statusMessage = TOCK_Status_Complete;
            this.icon = "utility:unlock";
            this.iconVariant = "";
            this.status = value;
        }
```

Les propriétés réactives

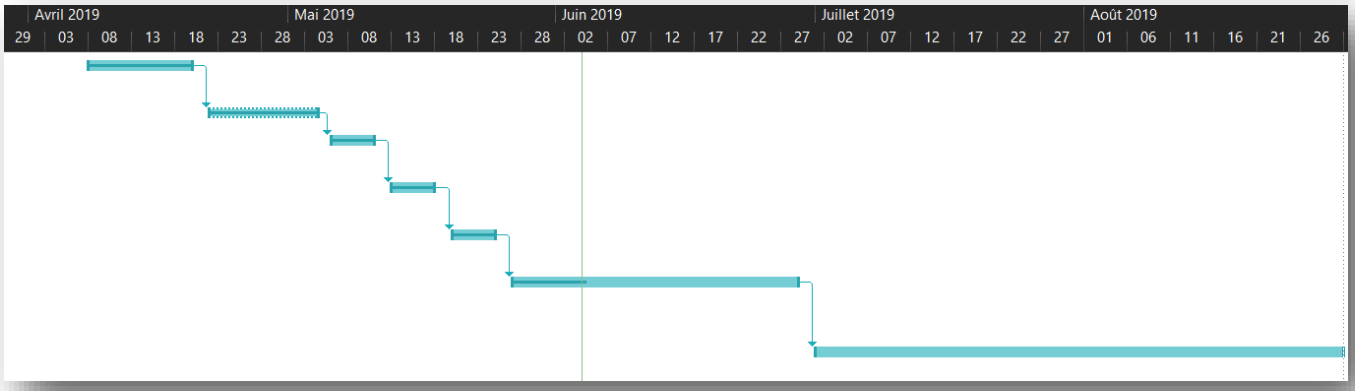
On dispose aussi pour chaque composant d'une feuille de style CSS pour changer les styles. J'ai cependant beaucoup utilisé le Framework CSS Lightning Design System afin que l'application s'intègre au mieux dans le « look and feel » de Salesforce. Pour le data binding, LWC propose deux décorateurs (l'équivalent des annotations en Javascript) qui permettent de définir des propriétés réactives. C'est-à-dire que lorsque la valeur de l'une de ces dernières change, le composant est à nouveau rendu. Le décorateur *@track* permet de définir une propriété réactive privée alors que le décorateur *@api* permet de définir un propriété réactive publique, pouvant être utilisée par les composants parents. Il est également possible d'implémenter de la logique au changement de ces dernières par l'implémentation de *getters* et *setters* personnalisés.

J'ai, dans un premier temps, travaillé à dessiner l'architecture de l'arborescence des composants de l'interface en fonction de la maquette fournie par le web designer via la plateforme sketch.io. Pour chaque partie de l'interface, je l'ai fractionné de la façon la plus atomique possible afin de pouvoir y faire correspondre des composants Lightning présents dans la bibliothèque standard Lightning. Dans l'optique d'ouvrir la possibilité à l'internationalisation de l'application, tous les textes sont affichés via le système de ressource texte de Salesforce.

2.7/ Diagramme de Gantt du Projet

✓		Formation sur la plateforme Trailhead	10 jours	Lun 08/04/19	Ven 19/04/19	
✓		Découverte de Tock	10,5 jours	Lun 22/04/19	Sam 04/05/19	1
✓		Refactoring du Back end	5 jours	Lun 06/05/19	Ven 10/05/19	2
✓		Développement des tests unitaires	5 jours	Lun 13/05/19	Ven 17/05/19	3
✓		Optimisation de l'architecture	5 jours	Lun 20/05/19	Ven 24/05/19	4
		Développement du front end pour ordinateur de Bureau	25 jours	Lun 27/05/19	Ven 28/06/19	5
		Développement du front end pour mobile	45 jours	Lun 01/07/19	Sam 31/08/19	6

Le diagramme de Gantt du projet



Le projet s'est déroulé comme prévu par le planning. Cependant, le stage se poursuivant jusqu'ç fin Août, il est possible que des divergences apparaissent à l'avenir.

Difficultés rencontrées

Au cours de se stage, j'ai pu rencontrer quelques difficultés. Heureusement, grâce à mes compétences acquises tout au long de mon cursus en DUT Informatique, j'ai pu les surmonter pour mener à bien ma tâche.

Tout d'abord, concernant le *versionning* des sur Salesforce, il est sensiblement différent et plus compliqué qu'avec un environnement de développant habituel. En effet, l'environnement d'exécution est situé dans le cloud, celui-ci ne peut pas être pris en compte par le système de gestion de version. C'est-à-dire que même si les fichiers locaux sont *versionnés*, après le déploiement sur l'organisation, les fichiers ne sont plus pris en compte par Git. Cette particularité a pour moi été source de confusion notamment pendant le débogage de certaines fonctionnalités : il m'arrivait de faire des bugs qui avaient déjà été corrigés. Désormais, je redouble d'attention à chaque déploiement pour être certain que la configuration actuelle de mon système de gestion de version est adéquate.

Par ailleurs, à mes débuts dans l'application Tock, j'ai eu un peu de mal à me repérer. Le code étant très peu lisible et non documenté, il m'a fallu persévérer afin de parvenir à identifié le rôle de chaque composant ainsi que ses relations avec les autres. Cependant, après avoir multiplié les relecture ainsi que les schémas, j'ai pu avoir une vue d'ensemble de l'application assez claires.

Bilan technique et personnel

Le bilan de ces neuf semaines de stage chez Edifixio, constituant ma première expérience professionnelle, est très positif, à la fois sur le plan professionnel et sur le plan technique.

Le travail que j'ai accompli, c'est-à-dire moderniser l'application Tock afin d'améliorer l'expérience des utilisateurs aussi bien du point de vue des employés que de celui des instances de pilotage de société, a bien avancé et continue de progresser, chaque jour à un rythme régulier. Mon stage se poursuivant jusqu'en Aout, il me reste encore à conclure le développement du *front end* de la version de bureau de l'application ainsi que celui de la version mobile de l'application. Grâce à l'expérience que j'ai accumulée jusqu'ici sur les technologies utilisées, je compte bien atteindre ces objectifs.

Le bagage technique et scientifique que j'ai eu l'occasion d'acquérir tout au long de ma formation à l'IUT m'a permis de m'adapter rapidement à ce nouvel environnement de travail. L'expérience acquise en programmation et en algorithmique ont été un atout pour moi dans l'apprentissage de nouveaux langages comme APEX et de nouveaux Frameworks comme LWC. De même, mes connaissances en conception et programmation orientée objet, m'ont été d'une grande aide pour le *refactoring* du code. Enfin, j'ai pu mettre à profit mes compétences en création d'interface utilisateur lors du développement du *front end* de Tock.

Les outils de développement que j'ai eu l'occasion d'utiliser pendant mon stage tels que IntelliJ IDEA, Visual Studio Code et Git étaient déjà connus de moi car j'avais déjà eu l'occasion de les utiliser, soit dans le contexte de mon cursus à l'IUT soit dans le contexte de projets personnels. J'ai cependant dû apprendre à les connecter à Salesforce via des *plugins*. La gestion de version est également différente avec le développement sur Salesforce car elle se fait à partir des métadonnées de l'organisation. Cette dernière spécificité m'a d'ailleurs causé quelques problèmes lors du déroulement de mon *flow git*.

Au cours de mes deux semaines de formation, j'ai pu me former et acquérir des compétences qui m'ont été utiles par la suite : la plateforme Salesforce et son fonctionnement, la programmation APEX, Javascript avec Aura, les requêtes SOQL, les bonnes pratiques de développement dans la plateforme Salesforce. J'ai effectué ce travail de formation concernant cet environnement entièrement nouveau pour moi sur la plateforme d'e-learning Trailhead et grâce à la documentation officielle de Salesforce. Pour ce qui est des patrons de conception utilisés au cours du projet, M. Dhouib m'a fourni une documentation de son cru, très bien réalisée et comportant des schémas explicatifs ainsi que des exemples de code. Cette documentation m'a été très utile tout au long de l'étape du *refactoring* de l'application.

En résumé, cette expérience professionnelle m'a permis de mettre en œuvre et d'appliquer en conditions réelles les compétences acquises au cours du DUT telles que le *refactoring* de code, le développement d'interface graphique et le développement objet. J'ai en effet pu acquérir de l'expérience sur ces compétences en les utilisant en conditions réelles dans un environnement et avec des technologies qui m'étaient jusqu'ici totalement inconnues.

Sur le plan humain, cette expérience a tout d'abord contribué à compléter mes connaissances de la communication au sein des entreprises. Par ailleurs, cela a été une occasion pour moi de rencontrer de nombreuses personnes passionnées par l'informatique et avec lesquelles j'ai pu échanger sur de nombreux sujets, ce qui, d'un point de vue à la fois technique et humain, était très enrichissant.

Bilan du travail réalisé

A ce jours j'ai pu réaliser un partie non négligeable du travail planifié. En effet j'ai pu remplir un certain nombre d'objectif.

Tout d'abord, j'ai terminé le complet *refactoring* du *back end* de l'application sous la forme d'un contrôleurs proposant des *webservices* communiquant avec le *front end* via des objets *DTO*. Ces classes ont été scrupuleusement structurées selon le patron de conception correspondant. Le contrôleur ne contient aucune logique et la délègue à une classe de service. La classe contrôleur compte plus de mille sept cent lignes de codes, et la classe de service en compte plus de mille trois cent. Un moteur d'exception personnalisé, doté de codes d'erreurs adaptés à chaque cas été mis place. Des tests unitaires rigoureux ont étés développés afin de vérifier le bon fonctionnement du logiciel. Pour que tous ces tests passent, il a fallu corriger quelques bugs.

Ensuite, au niveau du *front end*, des amélioration mineurs ont été apportées à la version actuelle. Cependant, le développement de la nouvelle version du *front end* avec le *framework* Lightning Web Components a été largement commencée, avec le développement des composants permettant d'assembler la feuille de temps et de la mettre à jours. Il ne reste donc plus qu'à développer la gestion des astreintes et heures supplémentaires.

Conclusion

Au cours de ce stage, j'ai procédé au *refactoring* et à l'optimisation du *back end* de l'application Tock. J'ai également commencé le développement du *front end* pour la version de bureau.

Etant donné que mon stage dure jusqu'au 30 Août, je pense parvenir à terminer la version de bureau ainsi que la version mobile.

Afin de mener à bien mon objectif j'ai dû utiliser de nouveau outils, me familiariser avec un nouvel environnement et appliquer de nouvelles méthodes de travail. J'ai notamment pu approfondir ma capacité à procéder à des *refactorings*, à créer des tests unitaires efficaces et à développer des interfaces graphiques modernes et ergonomiques. J'ai accompli mon travail dans l'objectif d'avoir une qualité de code optimale, afin que ce dernier soit lisible, maintenable, correctement commenté et documenté.

Je vais poursuivre mes études en réalisant un cycle ingénieur en alternance à l'EFREI Paris, dans le parcours « Logiciel et Système d'information » en suivant la majeure « Software Engineering ». J'effectuerai mon alternance chez Edifixio, où une proposition d'embauche m'a été faite afin de poursuivre ma collaboration avec cette entreprise. Ce parcours est dans la continuité de mon stage. Cette expérience m'a permis de conforter mon projet professionnel, en confirmant ma passion pour développement et la conception de logiciels.

Table des figures

Glossaire

Angular : Framework open source de développement d'applications web développé par google utilisant Typescript. Typescript est un sur ensemble fortement typé de Javascript développé par Microsoft.

API REST : une API une interface de programmation applicative, permettant de faire facilement des échanges de donnée entre deux logiciels. La spécification REST (Representational State Transfer) est un style d'architecture logiciel définissant un ensemble de contraintes à utiliser pour créer un webservice.

Contrôleur : Composant du patron de conception Modèle Vue Contrôleur chargé de réagir aux entrées et à les convertir en commandes pour la vue ou le modèle.

CSS : Cascading Style Sheets est un langage de feuille de style utilisé pour décrire la représentation d'un document dans un langage de markup comme HTML.

Data Binding : technique pour lier des sources de données à des fournisseurs de données et de les synchroniser. C'est couramment utilisé dans la conception des interfaces graphiques.

Développeur Full Stack : Un développeur full stack est un développeur capable de créer des logiciels dans le contexte d'une application client ou serveur. Cela correspond à trois capacités principales : programmer un navigateur, programmer un serveur et programmer une base de données.

ECMAScript : Spécification visant à standardiser le JavaScript pour favoriser des implémentations indépendantes du navigateur.

E-learning : Apprentissage de compétences par le biais de cours en ligne.

Framework : En programmation, une Framework est une couche d'abstraction sur un système dans laquelle des fonctionnalités peuvent être développés grâce à du code afin des créer des applications.

Front end et back end : Séparation entre la couche de présentation (interface graphique) correspondant au front end et la couche d'accès aux données correspondant au backend.

HTML : Hyper Text Markup Langage. Langage de markup définissant la structure des pages web.

HTTP : Hyper Text Transfer Protocol. Protocole applicatif d'échange de données sur le Web. HTTP fonctionne comme un protocole de requête et de réponse : un client fait une requête à un serveur qui lui envoie une réponse.

JSON : Javascript Object Notation est un standard de format de fichier qui permet de stocker des données de façon lisible par l'homme mais également de façon structurée.

Logique d'application : Partie d'une application qui vise à transposer les règles du monde réel dans le fonctionnement du programme.

« Look and Feel » : En conception logicielle, terme utilisé pour caractériser l'aspect d'une interface graphique comme ses couleurs, ses formes, etc … mais aussi par sa prise en main, l'expérience de l'utilisateur par rapport aux différents contrôles de l'interface.

Markup : Système d'annotation de document distinguable d'un texte afin d'en structurer les informations. Des langages de makup, comme HTML ont une sémantique de présentation et sont utilisés pour construire des interfaces graphiques.

Mock : en programmation orientée objet, un objet Mock est un objet simulé, imitant le comportement d'un objet réel de façon contrôlée afin de tester d'autres objets qui en dépendent.

Modèle de donnée : Modèle abstrait pour organiser des éléments de donnée et standardiser leurs relations aux entités du monde réel de façon structurer, pour faciliter la résolution de problèmes.

Orienté Objet : Paradigme de programmation basé sur le concept d'objets. Les objets contiennent des données sous la forme d'attributs, ainsi que du code : les méthodes.

Parallélisme : Capacité d'un système informatique à exécuter plusieurs instructions en même temps

Parsing : analyse syntaxique d'une chaine de caractère pour en extraire des données et les stocker de façon structurée.

Patron de conception : En ingénierie logicielle, un patron de conception logicielle est une solution générale, réutilisable à un problème souvent rencontré dans un contexte donné.

Platform As A Service (PaaS) : Catégorie de services cloud qui fournissent une plateforme, permettant aux clients de développer, exécuter et gérer des applications sans construire une infrastructure physique.

Progiciel : Un logiciel utilisé pour satisfaire les besoins d'une entreprise ou d'une organisation.

Responsive Design : Rendu des pages web adaptatif en fonction des dispositifs et de la taille des écrans.

Template : standard de fichier non exécutable utilisés à par des logiciels en tant qu'exemples préformattés pour générer des documents ou des interfaces graphiques.

Test unitaire : Méthode de test qui vise à tester individuellement chaque unité de code afin de vérifier qu'elle fonctionne correctement.

Visualforce : Framework MVC de la plateforme Salesforce pour créer des pages générées dynamiquement des pages web.

Webservice : Service offert par un système à un autre par le biais du web (et donc du protocole HTTP).

Bibliographie et Webographie

- Site web d'Edifixio
https://www.edifixio.com

- Trailhead, la plateforme d'E-learning de salesforce

https://trailhead.com

- Le site web de la documentation technique de Salesforce

https://developer.salesforce.com/

- La documentation du projet Renault NEST de Achraf Dhouib

- Wikipedia

https://fr.wikipedia.org

- Livret d'accueil des stagiaires chez Edifixio