Rapport de Stage

Nom du stagiaire : Soumia B.Azza (étudiante en 2ème année BTS SIO SLAM).

Formation : BTS SIO option SLAM avec le CNED.

Maître de stage : Monsieur Nordine Azza — Développeur Java auto-entrepreneur (freelance).

Durée du stage : 10 semaines en 2023 + 1 mois en 2024.

Remerciements

Je tiens à remercier sincèrement mon maître de stage Monsieur Azza pour sa disponibilité, sa patience et tous les conseils qu'il m'a donnés pendant ces semaines.
Il m'a non seulement aidé à progresser sur le plan technique, mais il m'a aussi transmis des valeurs importantes pour réussir dans ce métier : rigueur, curiosité, persévérance.
Merci pour sa confiance et pour son soutien dans tous les moments où j'en av.

Sommaire

Introduction

Pour mon stage de BTS SIO, que j'ai réalisé chez un auto-entrepreneur (freelance), Monsieur Nordine Azza, développeur freelance spécialisé en Java.

Ce choix n'était pas fait au hasard : devenir freelance après mon diplôme est un objectif qui me tient à cœur.

Je voulais donc découvrir de près ce que signifie réellement travailler en indépendant, comment gérer les projets, les clients, l'organisation quotidienne...

Être freelance, surtout dans le domaine du développement, demande beaucoup d'autonomie, une rigueur professionnelle et une vraie passion pour le code.

Ce stage m'a permis de vivre une vraie immersion dans cet univers, tout en renforçant mes compétences techniques sur des outils modernes que je souhaitais maîtriser.

Présentation de l'Entreprise

Monsieur Nordine Azza est un développeur freelance expérimenté, spécialisé dans le développement d'applications backend en Java/Spring Boot et frontend avec Angular.

Il travaille pour différents types de clients, notamment IPPON Technologie (un cabinet de conseil en transformation numérique reconnu pour son expertise dans les architectures logicielles, le cloud et le développement agile.) son principal client, et intervient aussi comme formateur, partageant ses connaissances avec d'autres développeurs.

Pendant mon stage, j'ai pu profiter de ses conseils précieux, basés sur son expérience concrète du terrain.

Missions Réalisées

Refactoring et optimisation de code

Au début de mon stage, mon maître de stage m'a confié la mission de faire du refactoring sur du code existant, développé par d'autres personnes avant moi.

Ce travail m'a beaucoup appris : lire un code qu'on n'a pas écrit soi-même n'est pas évident au début.

J'ai dû comprendre la logique, détecter les faiblesses, améliorer la structure et appliquer de bonnes pratiques pour le rendre plus clair et plus efficace.

Rédaction de Tests Unitaires

Ensuite, j'ai été chargé d'écrire des tests unitaires pour plusieurs parties du projet.

Même si ce n'était pas la partie la plus « fun » du travail, c'était vraiment formateur.

J'ai appris à prévoir tous les cas possibles, à m'assurer que le code fonctionnait comme prévu, et à anticiper d'éventuelles erreurs pour éviter les mauvaises surprises en production.

Développement d'une Application Web : Générateur de Compte-Rendu d'Activités (CRAGenerator)

L'une des principales missions durant ce stage a été le développement d'une application web permettant aux freelances de générer des comptes-rendus d'activités (CRAGenerator) sous forme de PDF.

Cette application devait prendre en charge plusieurs fonctionnalités, comme la gestion des tâches, la génération automatique de PDF à partir de données saisies par l'utilisateur, et l'intégration des jours fériés et des week-ends dans le calcul des heures travaillées.

1. Conception de l'application :

L'application a été conçue avec une architecture full-stack, utilisant les technologies suivantes :

- **Back-end :**
    - **Java avec Spring Boot** pour la logique métier et la gestion des API REST.
    - **Maven** pour la gestion des dépendances et de la configuration du projet.
    - **Swagger** pour la documentation des API REST.
    - **JUnit** pour les tests unitaires du back-end.
    - **Spring Security** pour la gestion de l'authentification et des autorisations.

- **Front-end :**
    - **Angular** pour la partie front-end, utilisant **TypeScript** pour le langage de développement et **Visual Studio Code** comme IDE.
    - **Bootstrap** pour la gestion de la mise en page et des composants d'interface.
    - **Axios** pour la gestion des requêtes HTTP entre le front-end et le back-end.

L'application permet aux utilisateurs (les freelances) de renseigner leurs heures travaillées et de générer automatiquement un **compte-rendu d'activités** (CRA) au format PDF.

Ce PDF inclut des informations sur le nombre d'heures travaillées, les jours spécifiques, ainsi que des calculs précis sur le **Taux Journalier Moyen (TJM)**, tout en prenant en compte les jours fériés et les week-ends.

2. Développement de l'application :

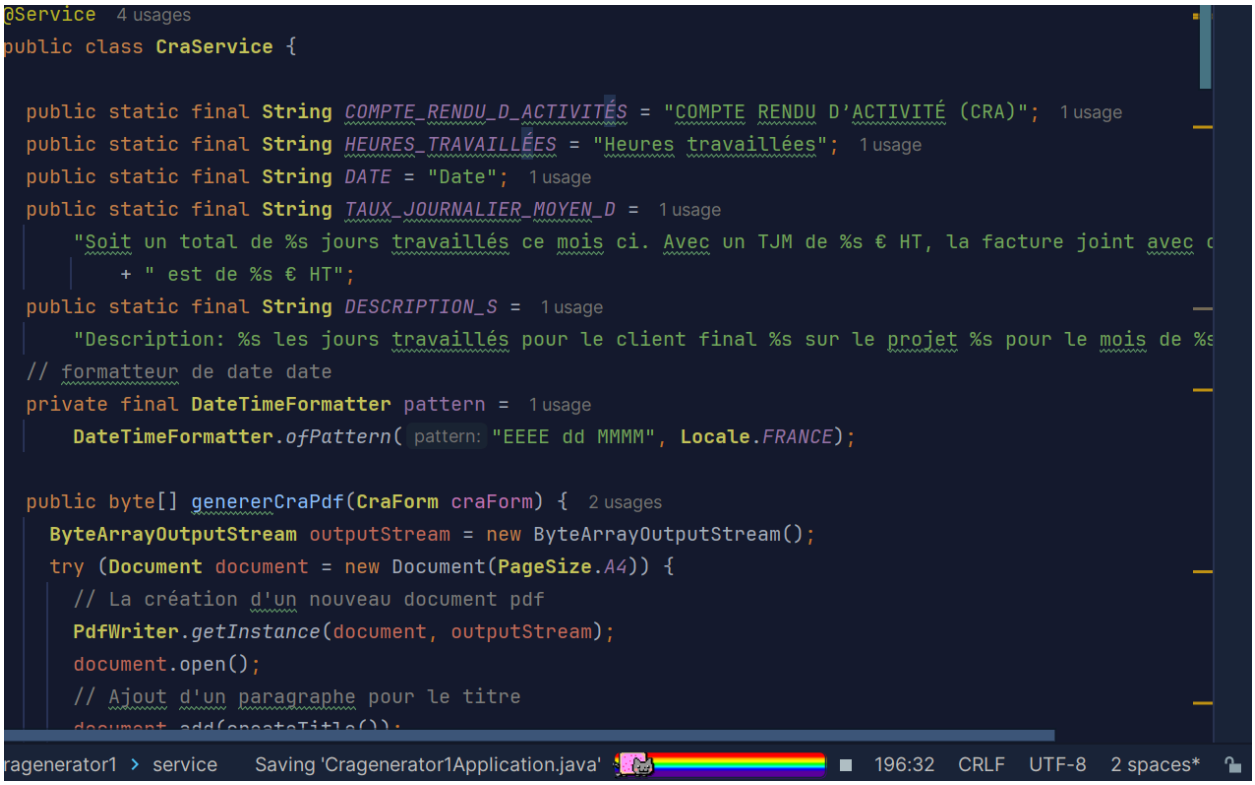
Le développement de l'application a été divisé en plusieurs étapes clés, chacune ayant son propre ensemble de défis et de solutions :

Back-end (Spring Boot) :

- Création de l'API REST permettant à l'utilisateur d'envoyer les données du CRA (ex : description, client, dates, heures travaillées, etc.).
- Mise en place d'un **service de génération de PDF** à l'aide de la bibliothèque **iText**.
- Intégration de la gestion des **jours fériés** via une API externe de jours fériés fournie par le gouvernement français (https://calendrier.api.gouv.fr).

- Calcul dynamique des heures travaillées en prenant en compte les jours fériés et week-ends.

Voici une partie du code du service de génération du PDF :



```
// Récupérer le premier jour du mois
var startMonth = Ligne.getDayMonthYear(craForm.lignes().get(0).dateDebut());
// Le compteur AtomicInteger pour compter le nb de jours travaillés
AtomicInteger numJrTravaill = new AtomicInteger();
startMonth
    .datesUntil(startMonth.plusMonths( monthsToAdd: 1))
    .forEach(
        (localDate -> {
          var heureTravaille = craForm.heuresTravaillByDate(localDate);
          jourTravail(localDate, heureTravaille, numJrTravaill);
          String nbJrTravail =
              heureTravaille > 0
                  ? String.format(" %.1f (%d) ", heureTravaille, numJrTravaill.get() / 2)
                  : "";

          PdfPCell dateCell = createPdfCell(localDate.format(pattern));
          PdfPCell htCell = createPdfCell(nbJrTravail);
          updateCellsDependingOfDay(isWeekEnd(localDate), Color.LIGHT_GRAY, dateCell, htCell);
          updateCellsDependingOfDay(isJourFerie(localDate), Color.PINK, dateCell, htCell);
          updateCellsDependingOfDay(
              isJourConge(localDate, heureTravaille), Color.PINK, dateCell, htCell);
          // L'ajout des cellules à la table
          table.addCell(dateCell);
          table.addCell(htCell);
```

```
 public static PdfPCell createPdfCell(String datePdf) {  2 usages
font.setStyle(Font.BOLD);
PdfPCell dateCell = new PdfPCell(new Phrase(datePdf, font));
dateCell.setPaddingLeft(1);
dateCell.setVerticalAlignment(Element.ALIGN_MIDDLE);
dateCell.setHorizontalAlignment(Element.ALIGN_LEFT);
dateCell.setBorderWidth(1);
dateCell.setBorderColor(Color.GRAY);
dateCell.setFixedHeight(30);
return dateCell;


blic static boolean isWeekEnd(LocalDate date) {  4 usages
return date.getDayOfWeek() == DayOfWeek.SATURDAY || date.getDayOfWeek() == DayOfWeek.SUNDAY;


blic Map<LocalDate, String> lesJourFeries() {  1 usage
String uri = "https://calendrier.api.gouv.fr/jours-feries/metropole/2024.json";
RestTemplate restTemplate = new RestTemplate();
ResponseEntity<Map<LocalDate, String>> responseEntity =
    restTemplate.exchange(uri, HttpMethod.GET,  requestEntity: null, new ParameterizedTypeReference<>() {
Map<LocalDate, String> joursFeries = responseEntity.getBody();
return joursFeries;
```

Front-end (Angular) :

- Création d'une interface utilisateur permettant à l'utilisateur de saisir les informations nécessaires pour générer son CRA.
- Mise en place de **formulaires dynamiques** pour gérer les entrées de l'utilisateur, et affichage des résultats sous forme de tableau.
- Communication avec l'API REST en utilisant **Axios** pour envoyer les données et récupérer le PDF généré.

Gestion des jours fériés et week-ends :

- Le calcul des heures travaillées exclut les jours fériés et week-ends.
- J'ai utilisé l'API **jours-feries** du gouvernement pour récupérer les jours fériés de l'année en cours.

- Implémentation de la logique pour calculer les heures travaillées uniquement sur les jours ouvrés.

Sécurité et authentification :

- Le back-end a été sécurisé avec **Spring Security** pour limiter l'accès à l'application en fonction des rôles utilisateur (admin, freelance).

- Mise en place d'une authentification basée sur **JWT** (JSON Web Tokens) pour sécuriser les endpoints de l'API.

3. Tests de l'Application :

J'ai également contribué à l'écriture de tests unitaires pour vérifier la fiabilité de l'application et m'assurer qu'elle fonctionnait correctement avant sa mise en production.

Voici un exemple de test unitaire :

```java
@Test
void doitRetournerLePremierJourDuMois() {
    Assertions.assertThat(Ligne.getDayMonthYear(LocalDate.of( year: 2024,  month: 12,  dayOfMonth: 15)))
        .isEqualTo(LocalDate.of( year: 2024,  month: 12,  dayOfMonth: 01));
}

@Test
void doitRetournerLeMemeDateSi1erJourDuMoisSaisi() {
    Assertions.assertThat(Ligne.getDayMonthYear(LocalDate.of( year: 2024,  month: 12,  dayOfMonth: 01)))
        .isEqualTo(LocalDate.of( year: 2024,  month: 12,  dayOfMonth: 01));
}
```

4. Déploiement de l'application :

Une fois le développement terminé, l'application a été déployée sur **AWS** pour assurer sa disponibilité et sa scalabilité.

Voici les étapes clés du déploiement :

1. ***Back-end sur Amazon EC2*** *: L'instance* ***EC2*** *a été configurée pour héberger le back-end de l'application. Un serveur* ***Ubuntu*** *a été utilisé, et l'application a été lancée avec* ***Spring Boot*** *en utilisant* ***Java 17****.*

2. ***Front-end sur Amazon S3 et CloudFront*** *: Le front-end développé avec* ***Angular*** *a été compilé et déployé dans un* ***bucket S3****, qui sert ensuite les fichiers statiques aux utilisateurs via* ***CloudFront*** *pour optimiser la distribution des fichiers.*

3. ***Configuration du domaine avec Route 53*** *: Le domaine de l'application a été configuré avec* ***Route 53*** *pour gérer les DNS et associer le domaine personnalisé à l'application déployée sur AWS.*

4. ***Sécurisation avec SSL*** *: Un certificat SSL a été configuré pour sécuriser les communications entre l'utilisateur et l'application.*

5. Résultat Final :

L'application est maintenant opérationnelle, permettant aux freelances de générer leurs comptes-rendus d'activités de manière rapide et automatisée.

Le déploiement sur AWS garantit une haute disponibilité, une performance optimale et une gestion facile de l'infrastructure.

Compétences Développées

- **Développement full-stack (Spring Boot + Angular)** : J'ai développé des compétences en intégrant des technologies Java pour le back-end et Angular pour le front-end.

- **Gestion d’une API REST** : J'ai appris à concevoir et à exposer des API REST sécurisées et performantes.

- **Génération de PDF** : J'ai travaillé avec la bibliothèque **iText** pour générer des rapports au format PDF, une fonctionnalité clé de l'application.

- **Déploiement sur AWS** : J'ai déployé l'application sur **AWS** en utilisant EC2 pour le back-end et S3 + CloudFront pour le front-end.

- **Tests unitaires et gestion des erreurs** : J'ai écrit des tests pour m'assurer de la fiabilité du code, notamment pour la gestion des heures travaillées et des jours fériés.

Conclusion

Le projet de développement du **Générateur de CRA** a été très formateur, me permettant de mettre en pratique mes connaissances en développement web full-stack et en gestion d'une application de A à Z.

Ce projet m’a aussi permis de découvrir les aspects techniques du déploiement d’une application sur AWS, en comprenant les étapes de mise en production et la gestion d’une infrastructure cloud.

Ce stage m'a conforté dans ma volonté de poursuivre une carrière en freelance, avec une maîtrise plus approfondie des technologies modernes de développement et de déploiement.