|  |
| --- |
| SMILE - EPSI |
| Dossier professionnel RNCP 35584 - Expert en Informatique et en Système d’Information |
| Promotion 2021-2023 |

|  |
| --- |
| Ryan Murati  08/01/2023 |

Table des matières

[I- Remerciements 2](#_Toc138091367)

[II- Introduction 3](#_Toc138091368)

[III- Environnement professionnel 4](#_Toc138091369)

[A) Présentation de l’entreprise SMILE 4](#_Toc138091370)

[i) Organisation de la structure 4](#_Toc138091371)

[ii) Présentation du SI, acteurs, informations et flux 4](#_Toc138091372)

[B) Activités de la structure 4](#_Toc138091373)

[i) Les différentes équipes de SMILE 4](#_Toc138091374)

[ii) Les objectifs des activités 5](#_Toc138091375)

[iii) Les clients réguliers 5](#_Toc138091376)

[C) Mon activité à SMILE 7](#_Toc138091377)

[i) Place dans les équipes 7](#_Toc138091378)

[ii) Missions confiées 7](#_Toc138091379)

[IV- Valorisation des compétences 8](#_Toc138091380)

[A) Projet WSO2 8](#_Toc138091381)

[i) Explication en détail du projet 8](#_Toc138091382)

[ii) Quelles compétences validées par le projet ? 12](#_Toc138091383)

[iii) Renseignement sur la compétence et bonnes pratiques 12](#_Toc138091384)

[B) Projet INRAE 13](#_Toc138091385)

[i) Explication en détail du projet 13](#_Toc138091386)

[ii) Contexte de réalisation, pourquoi avoir besoin de cette compétence dans l’activité? Comment ça a été réalisé et assimilé ? Quelles ressources utilisées ? As-t’elle été traversée par des évolutions majeures ces 10 dernières années ? 13](#_Toc138091387)

[iii) Renseignement sur la compétence et bonnes pratique 13](#_Toc138091388)

[C) Projet Java 13](#_Toc138091389)

[i) Explication en détail du projet 13](#_Toc138091390)

[ii) Contexte de réalisation, pourquoi avoir besoin de cette compétence dans l’activité? Comment ça a été réalisé et assimilé ? Quelles ressources utilisées ? As-t’elle été traversée par des évolutions majeures ces 10 dernières années ? 13](#_Toc138091391)

[iii) Renseignement sur la compétence et bonnes pratique 13](#_Toc138091392)

[V- Contexte de réalisation 14](#_Toc138091393)

[VI- Conclusion 15](#_Toc138091394)

[A) Résumé 15](#_Toc138091395)

[B) Ouverture sur l’avenir 15](#_Toc138091396)

[VII- Annexe 15](#_Toc138091397)

# Remerciements

Je tiens à remercier toutes les personnes qui m’ont accompagné lors de mes deux années en alternance à SMILE, ainsi que toutes celles qui m’ont aidé lors de la rédaction de ce rapport. En premier lieu, j’adresse mes sincères remerciements à **Sophie DEMOIS**, ainsi que toute l’équipe pédagogique de l’EPSI, dont le suivi et l’aide apportée a été d’une grande ressource lors de la rédaction de ce rapport. J’aimerai ensuite remercier mes deux manageuses de SMILE, **Cécile WALKENER** et **Thi** **thuy Linh LE**, qui m’ont suivi tout au long de mon alternance, et m’ont présenté les collaborateurs avec lesquels j’allais travailler.

Je remercie également toutes les personnes du pôle GED de SMILE, notamment **Nicolas Vezian**, qui ont su m’intégrer à l’entreprise lors de mon arrivée, et pour leur disponibilité et leur aide précieuse à chacune de mes questions ou difficultés.

Un grand merci à mes anciens équipiers de l’INRAE, particulièrement **Nicolas Brun** avec qui j’ai eu plaisir de travailler pendant 8 mois, et qui m’ont formé avec patience et bienveillance.

# Introduction

SMILE développe et apporte des solutions open source pour divers clients ; j'ai donc, tout au long de mon apprentissage, collaboré auprès des clients issus de secteurs différents, pour leur apporter des solutions digitales et développer leur projet de transformation numérique.

Après une présentation de l’entreprise SMILE, de son SI, de ses équipes et des missions qu’elle propose en général, je décrirai les compétences j’ai pu acquérir lors des projets que j’ai mené dans l’entreprise, en décrivant les différentes tâches effectuées, les difficultés que j’ai pu rencontrer et comment je les ai surmontées, tout en gardant un regard critique sur les possibles axes d’améliorations de mon travail.

Lors de mon arrivée à SMILE, j’ai été placé dans une équipe de deux personnes, afin que je sois formé sur une technologie que je ne connaissais pas, WSO2 API Manager. Passés quelques mois à acquérir les compétences nécessaires sur cette technologie, j’ai été à même de proposer de développer un logiciel de démonstration, permettant de présenter aux potentiels clients les possibilités que la technologie offrait.

Après le succès de ce projet solo, j’ai rejoint l’INRAE (Institut national de la recherche agronomique), en régie, avec l'idée d’une formation sur leurs technologies et leurs projets pendant 6 mois suivi d’une autonomie projet les 6 mois suivants

Après mes 6 mois de formation, je me suis rendu compte que je projet ne me stimulais pas, j’ai donc fait une demande a l’INRAE et a SMILE afin de changer de projet plus tôt, ce qu’ils ont accepté après délibération. J’ai donc continué deux mois à l’INRAE avant de rejoindre un projet le 01/03/2023 pour le gouvernement, dans une équipe de 7 personnes en Agilité SCRUM.

Je parlerais aussi, pour démontrer l’acquisition de certaines compétences, d’anciens projets effectués lors de mon cursus d’apprentissage.

# Environnement professionnel

## Présentation de l’entreprise SMILE

### Organisation de la structure

SMILE est une société spécialisée dans le développement de solutions open source. Crée en 1991 à Paris, elle est maintenant implantée dans 17 agences réparties dans 9 pays différents. Elle offre des solutions à de nombreux grands comptes français et européens, comme la préfecture de Lyon, la SNCF ou encore Total et beIN sport.

La société emploie plus de 2000 collaborateurs, et l’agence de Lyon est composée de 150 collaborateurs, dispersés en différentes équipes que nous retrouverons plus tard dans le dossier.

Voici l’organigramme de la structure lyonnaise lors de mon arrivée, **[montrant les différentes équipes et les personnes dirigeant ces équipes] :**

**ORGANIGRAMME**

Le groupe SMILE m’a embauché en octobre 2021 en tant que développeur java apprenti, et m’a confié différentes missions, que je décrirai tout au long de ce rapport.

### Présentation du SI, acteurs, informations et flux

## Activités de la structure

### Les différentes équipes de SMILE

Comme présenté précédemment, SMILE compte plusieurs équipes qui travaillent sur plusieurs projets différents. Cette partie va détailler les objectifs et les missions des équipes proches de mon travail.

Les équipes peuvent-être séparées en trois pôles :

**Data**

Les équipes du pôle data suivent des projets dont le but est de mettre en œuvre un système de gestion de données. Cela se traduit par :

* De la relation client afin de trouver quels sont les bonnes technologies pour implémenter ce système de gestion.
* Une création de systèmes qui correspondent aux demandes clients, cherchant à convenir au mieux aux besoins, en restant le plus optimisé possible.
* Un accompagnement client lors de la restitution du projet.

Les collaborateurs de ces équipes peuvent être amenés à utiliser diverses technologies comme *Talend*, *Power BI*, *Jedox* ou encore des API Manager tel que ***WSO2 API Manager.***

**Java/Backend**

Le pôle Backend est composé de multiples équipes, travaillant sur de nombreux projets, pour des clients ou en interne. Les collaborateurs de ces équipes fonctionnent en Agilité (SCRUM) dans des équipes d’environ 6 à 8 personnes constituées de développeurs backend, frontend, de testeurs, et d’un ScrumMaster.

Leur travail consiste à développer de nouvelles features sur un projet donné en Java, de participer aux créations des US (User Stories), faire de la relecture de code ainsi que de participer aux temps d’équipe agile.

**Régies**

Il est possible que des clients fassent appel à SMILE pour qu'un collaborateur ou une équipe, soit détaché dans leur propre structure pour aider au développement de leur projet. On appelle cela un projet en régie. C’est une coutume assez fréquente chez SMILE ; c'est ainsi que j'ai rejoint l’INRAE (**I**nstitut **N**ational de **R**echerche pour l’**A**griculture l’alimentation etl’**E**nvironnement).

Les autres pôles de SMILE (pour lesquels je n'ai pas collaboré) sont les suivants :

Javascript,Drupal, PHP, E-commerce, GED/ECM, Fonctionnel et testing, Project Management

### Les objectifs des activités

Gagner de l’argent, fidélité client, utiliser les compétences des collaborateurs à bon escient

### Les clients réguliers

Grâce à son expertise et à son ancienneté, SMILE a pu obtenir la confiance de nombreuses entreprises et intervient donc chez de nombreux clients différents. Certains font régulièrement appel aux services de SMILE, voici une présentation non exhaustive de ces clients :

* beIN SPORTS

Le défi de beIN SPORTS était de créer un site web flexible capable d’assurer la publication de contenus multilingues et multisites, permettant à 60 journalistes et éditorialistes, répartis dans 4 salles de rédaction, de pouvoir facilement contribuer sur le site.

Après 2 ans de développement SMILE produit une plateforme open source développé dans 4 pays et 7 langues, permettant l'intégration de nombreux prestataires.

Avec ce site SMILE remporte en 2015 en Symphony Awards du « *best B2C website of the year* »

* Suez

Lors de leur affiliation, pendant 2 ans SMILE intervient chez Suez sur les technologies *Symphony* et *eZ publish.*

Dans cette période Suez recherche des experts afin de concevoir une plateforme web optimisée en seulement un mois. Ce projet sera donné à une équipe Smilienne, qui développera la solution en seulement 30 jours.

Un an après ce projet Suède décide d'accorder sa confiance à SMILE en lui confiant la conception et la maintenance de ces 3 sites web.

* EDF

Depuis 2016 SMILE occupe la fonction d'experts référent au sein du département SPENCER – ESA (département centralisé des besoins informatiques du groupe).

Actuellement plus de 60 tickets sont gérés chaque mois sur des technologies diverses et variées.

* Total

SMILE et total sont en lien depuis de nombreuses années. En effet 4 équipes sont mobilisées en parallèle contenant 20 collaborateurs, 7 projets sont réalisés par année avec total et plus de 500 tickets sont traités chaque année.

* Vitam

Vitam est un projet demandé par l'État français afin de créer un socle commun d'archivage open source permettant la facilité d'accès et la documentation. Missionnés sur le projet, SMILE se lance dans la conception d'un projet hautement sécurisé réutilisable et pérenne.

Engagé sur 20 ans ce fait le choix de la flexibilité technologique c'est-à-dire que la solution vitam est conçue de façon modulaire permettant l'optimisation ou la modification de certains pans de l'application

## Mon activité à SMILE

### Place dans les équipes

Chez SMILE, j’ai été placé dans 3 équipes différentes que j’expliquerais plus en détail dans la partie IV. Voici malgré tout une courte introduction à ma place dans ces-dites équipes :

*Equipe WSO2*

Lors de mon projet sur *WSO2 API Manager (APIM ou AM)* j’étais apprenti **« API Constructer »** dans une équipe composée de 2 personnes à mon arrivée.

*Régie INRAE*

Pendant ma mission en régie a l’INRAE, j’ai rejoint une équipe de 7 personnes afin de développer en Agilité une solution de gestion de rôles et habilitations utilisant la solution *OpenIDM* de forgerock.

*Equipe Java*

J’ai rejoint un projet java en tant que second développeur Backend. Ce projet consiste en un développement d’une application web permettant la création de formulaires lors d'accidents entre véhicules contenant des matières dangereuses. Ce projet est destiné au ministre de l’Écologie et a été développé par une équipe de 8 personnes : 2 développeurs backend, 4 développeurs front-end un Scrum Master et notre Product Owner.

### Missions confiées

Création d’API, apprentissage WSO2, Développement solution démo client en java WSO2

INRAE = OpenIDM forgerock, création d’API, gestion de rôles et habilitation, contact client

Java : création de formulaire pour le gvt recensant les accidents de transports contenant des matières chimiques

Puis autre projets (CGI MCO puis dev projet Xénon – Canada – MSPR)

# Valorisation des compétences

## Projet WSO2

### Explication en détail du projet

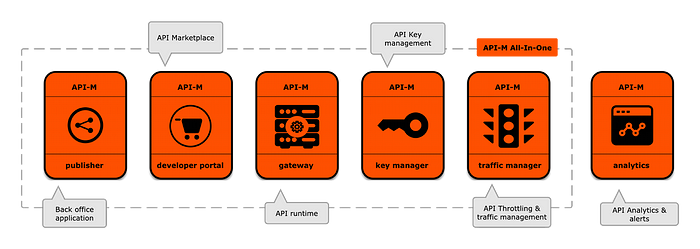
Lors de mon arrivée chez SMILE, contrairement à ma fiche de poste (développeur java junior), je n’ai pas rejoint un projet java. En effet, j’ai été placé dans une équipe composée de deux personnes (Stéphane et Lihn), dont l’expertise était le logiciel *WSO2 AM*.

Le but de cette équipe « WSO2 » était d’installer, chez des clients demandant, la solution WSO2 AM, de former les collaborateurs de ces clients afin de leur permettre de comprendre le logiciel ainsi que de modifier les APIs de ces clients pour qu’elles soient utilisables par WSO2 AM ; c’est-à-dire en format swagger API.

**Qu’est-ce qu’est *WSO2 API Manager* ?**

*WSO2 AM* est un gestionnaire d'API, c’est-à-dire un logiciel permettant aux entreprises de créer, publier et gérer des APIs. Il prend en charge la gestion du cycle de vie des API, le développement d'applications, le contrôle d'accès, la limitation de débit et l'analyse dans un seul système. L’exposition des APIs est gérée via un outil d’édition et de publication ; et la consommation via une plateforme sur laquelle l’utilisateur pourra s’inscrire aux APIs souhaitées. Le transfert d’information est réalisé via une passerelle (Gateway) permettant l’interopérabilité des SI des deux parties (éditeur et souscripteur).

Voici un schéma d’explication :



Comme dit précédemment, j’ai rejoint ce projet dès mon arrivée à SMILE malgré mon titre de développeur java junior. J’ai donc dû me former sur cette solution avant de pouvoir moi aussi entamer des processus d’installation chez les clients.

Pour cela, j’ai suivi une auto-formation, interne a SMILE, pendant 6 semaines, afin d’apprendre au mieux le fonctionnement de la solution. Pendant cette formation, j’ai appris à créer et utiliser des APIs en swagger, les ajouter a WSO2 AM, créer des « applications » grâce à ces APIs, créer des rôles, des gateways et à sécuriser les APIs.

**[PARLER DE LA FORMATION, QUOI D’APPRIS, QUOI DE COMPLEXE, OVERCAME SMTH ?]**

J’ai par ailleurs rédigé aux alentours de décembre 2021 un article interne pour SMILE, expliquant comment configurer Auth0 comme « Key Manager ». Vous pouvez le retrouver en Annexe.

Cette présentation m’a permis d’affirmer la fin de mon autoformation et de lancer mon autonomie sur les différents projets qui pourraient nécessiter ma présence.

Malheureusement, après quelques peaufinages de l’article en lien avec le responsable WSO2 de SMILE, Gregory Eve, les clients demandant une installation WSO2 APIM se fîtes de plus en plus rare, et les projets que mes collègues avaient nécessitaient des compétences dans des technos que je ne maitrisais pas, comme Talend ou Elasticsearch.

J’ai donc demandé un point avec mon manager, mes coéquipiers ainsi que Gregory, afin de leur proposer le développement d’une application en **Java**, utilisant l’**api manager** de **WSO2**, qui permettrai de faire des démonstrations des possibilités d’utilisation de WSO2 APIM dans un contexte de projet.

Cette proposition a été très bien reçue par mon manager et Gregory, puis après quelques délibérations nous sommes venus à un objectif plus précis : développer une application en java, permettant de démontrer l’utilisation de 3 APIs intégrés sur une gateway WSO2 APIM.

J’ai ainsi pu commencer le développement de mon projet, Interne à SMILE et en solo. Pour commencer j’ai effectué un benchmark des API que je pouvais utiliser, voulant rester dans un contexte professionnel et en lien avec l’essence de SMILE, je me suis dirigé vers des APIs OpenSource, des *Open APIs.*

Lors de ce benchmark, 3 API sont ressorties par leur praticité et le fait qu'elle soit open source.

La première API que j'ai choisie est une API de découpage administratif appelée **API Géo**, provenant du site api.gouv.fr, qui référence des informations sur toutes les villes françaises (comme son code INSEE, sa superficie, son nombre d'habitants, etc.) ; Ensuite, j’ai inclus au projet **CoronavirusTracker**, une API postée sur github, qui référence l'évolution de l'épidémie de COVID-19 dans le monde ou dans un pays demandé ; Enfin, pour ajouter des informations à la ville choisie par l'utilisateur, j'ai intégré **WeatherAPI**, qui permet d'avoir des informations sur la météo sur une ville donnée grâce aux retours de l’API Géo.

Ces 3 API me semblaient être un choix adapté et professionnels pour une démonstration client, et après présentation à mon équipe, ce choix a été validé. J'ai malgré tout dû abandonner une API lors de mes recherches, en effet, la première appli que j'avais choisie pour l'épidémie de COVID-19 ne correspondais pas aux besoins du projet.

Cette étape de maquettage et de benchmark m’a pris environ deux semaines, et, dès l'accomplissement de la maquette, j'ai réuni mes 2 équipiers ainsi que mon manager afin d'avoir un retour avant de commencer le développement.

Voici à quoi ressemblait la maquette acceptée par mon équipe :



J’avais donc prévu que l’application soit constituée de deux écrans :

-Le premier contenant une liste déroulantes des villes de France, permettant à l’utilisateur de choisir sur quelle ville il voudrait un focus d’informations.

-Le second serait constitué de 3 parties distinctes, la première, en vert, contiendrait les informations « basiques » de la ville choisie, c’est-à-dire son nombre d’habitant, sa surface, sa région et ses code postaux (car une ville de France peut avoir plusieurs codes postaux, par exemple Metz en a trois). La deuxième partie, en bleu, contiendrait des informations météorologiques de la ville, les températures maximum et minimum, la vitesse des vents, l’humidité, etc. Et la troisième partie, en rouge, contiendrait les informations relatives aux Covid-19, l’épidémie encore assez présente dans la période de réalisation du projet.

Après l'étape de maquettage de benchmark j'ai dû intégrer les API que j'avais choisies dans une instance locale de WSO2 APIM, afin d'utiliser une option proposée par le logiciel permettant d'exporter les API en différents SDK. Cette fonction permet de générer les APIs en 4 formats différents utilisable au gré du développeur : en *Android*, en *Java*, en *javascript* ainsi qu'en *Jmeter*, comme présenté ci-dessous.

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Une fois les ressources des API exportées en classes Java, j'ai pu commencer le développement de mon logiciel en *Spring MVC*, une technologie que ma collègue Lihn m’avait conseillé pour ce genre de projet.

Voici une brève explication du fonctionnement de *Spring MVC*.

Comme son nom l'indique Spring MVC utilise le design pattern MVC (modèle-vue-contrôleur) afin de créer des applications web en Java Avec le Framework Spring.

Le design pattern MVC est constitué de trois éléments : le modèle qui définit la structure de la donnée, la vue qui est l’élément que nous allons présenter à l'utilisateur (l’UI), et le contrôleur qui contient les fonctions qui modifie les données de la vue et du modèle.

Au centre de *Spring MVC* se trouve la dispatcher servlet, qui gère toutes les requêtes et réponses http, c’est le contrôleur principal de notre projet. Dans les modèles, nous allons retrouver Les données qui seront transmises vers les vues. Les vues, dans *Spring MVC*, sont implémentées en Java ServerPages (JSP), conçues pour être implémenté de manière optimisée.

Lors de la création de mon projet, je n'avais jamais utilisé la solution *Spring MVC*, j'ai donc en premier lieu suivi un tutoriel sur le site W3 School afin de me familiariser avec la technologie et le lien entre Spring et le design pattern MVC que j'avais déjà utilisé auparavant.

En premier lieu j'ai créé un projet Spring sur start.spring.io, en choisissant un projet Maven en Java et en ajoutant les dépendances nécessaires un Spring mvc.

J'ai ensuite créé mon premier JSP, un index qui ne contenait qu'un texte en hyperlien qui renvoyait sur la 2nde page que je comptais créer. J'ai ensuite modifié mon web.xml afin qu'il contienne les données nécessaires au bon fonctionnement de mon projet j'ai créé un dispatcher servlet qui se chargeait au lancement de mon application. Avec cela j'ai créé une servlet nommée HelloWorldservlet, qui allait agir comme ma deuxième servlet, et à cela j'ai ajouté un « helloWorld.jsp » qui ne contenait qu'un titre servant d'exemple. En dernier lieu j'ai créé un contrôleur « HelloController » qui faisait la jonction entre le lien hypertexte et la vue « helloWorld » lorsque l’utilisateur clique sur le lien.

Ce tutoriel était très basique mais couvrait l'entièreté des nécessités principales afin de réaliser un projet Spring MVC. J'ai donc pu débuter la réalisation de mon projet de démonstration dès la fin de ce tutoriel.

J'ai donc commencé par créer mon projet Maven sur start.spring.io comme effectué précédemment. J'ai aussi généré 3 JDK (Java Development Kit) à partir de la fenêtre « développeur » de *WSO2 APIM*, que j'ai build en utilisant Maven afin de pouvoir inclure ces 3 *JDK* dans mon projet précédemment créé, me permettant d’utiliser les fonctions liées à ces kits de développement dans les contrôleurs de mon projet.

J’ai ensuite créé 3 classes différentes, InfoCommunes.java, InfoCorona.java et InfoMeteo.java, en suivant un tutoriel dans la documentation de *WSO2 APIM*, expliquant comment lier les JDK créés par le logiciel avec les 3 API précédemment implémentés sur mon instance locale de *WSO2 APIM*.

Voici un exemple suivi d'explications de la création d'une des classes afin de pouvoir utiliser l'API avec Maven :



Dans cette classe nous pouvons remarquer 3 blocs distincts, un bloc de déclaration, un bloc de configuration pour l'API Commune ainsi qu'un 3e bloc instanciation d'une liste, suivi d'un appel à une fonction de l'API Commune.

Dans la 1ere partie, j’ai déclaré « statcommune », une instance de l’objet « CommunesAPI » crée dans le jdk de l’API commune ; ainsi que « client », de l’ApiClient. L'objet ApiClient permet à notre objet statcommune d'appeler l’API Communes après la configuration.

La 2e partie concerne justement la configuration de l'objet client qui contient 3 fonctions de configuration pour la réception des données de l'API ainsi que la fonction « setBasePath » qui va contenir l'URL d'appel à l'API. Cette url est stockée dans une classe «Constants.java », créé auparavant, contenant toutes les constantes du projet.

Enfin, la 3e partie de notre fonction concerne la demande d'informations spécifiques à l'API commune. En effet, la classe « Communes API » contient plusieurs fonctions d'appel vers l'API commune, demandant plusieurs informations en fonction de la réponse attendue. Ici j'avais besoin de certaines données spécifiées dans la liste « needInfo ». Je fais ensuite appel à la fonction communeCodeGet, qui demande, en argument, un code insee ainsi qu'une liste de retours demandé, suivi d’un format de retour et de la « geometry », concernant les coordonnées GPS de la ville.

Cette fonction « getNeededInfoCommune » nous permet donc, lors de son appel avec un code Insee, d'appeler l'API Communes et d'avoir, en retour, un objet « commune » contenant le nom, les codes postaux, la surface, le département, la région ainsi que le nombre d'habitants d'une ville de France.

Comme dit précédemment, j'ai créé les classes InfoCorona.java et InfoMeteo.java, qui contiennent similairement la même fonction que InfoCommune.java, c'est-à-dire un appel à l'API en l'attente d'une réponse contenant des données demandées. Vous pourrez retrouver ces fonctions en annexe.

Après avoir créé les trois contrôleurs des APIs, et les avoir testés avec Postman, j’ai créé en parallèle mes modèles, vues et contrôleurs permettant l’utilisation de l’application.

Comme expliqué précédemment, il ne me fallait que deux écrans, le premier permettant à l’utilisateur de choisir la ville dont il voulait les informations ; et le deuxième affichant les informations de la ville choisie.

Ayant choisi développer mon projet en *springMVC*, j’ai créé mes vues au format JSP, format permettant de contenir des balises HTML ainsi que des variables du modèle lié a la vue. J’ai donc débuté le développement du projet par la création des modèles, permettant d’avoir une vision sur les données nécessaires aux créations des vues.

Le premier modèle créé est nommé « SelectVille.java », et sera utilisé pour les données du 1er écran de sélection des villes, et ne contient que deux attributs, une liste de Communes «listeDonnéeesCommunes», que nous remplissons avec un fichier JSON contenant toutes les villes de France et un attribut « preferences » contenant le nom et le code Insee de la ville.

La fonction permettant de récupérer une liste de Communes provient d’une classe « ParseData.java » dont je parlerais ultérieurement.

Parallèlement, j’ai développé mon deuxième modèle, nommé InfoAPIsFront, contenant les données des 3 APIs. Voici le modèle, suivi d’une explication de sa structure :

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

On peut tout d’abord observer l’appel aux trois contrôleurs précédemment développés avec, comme paramètre, le code Insee de la ville choisie par l’utilisateur afin de récupérer les données des APIs que nous souhaitons afficher.

Ensuite nous ajoutons toutes les données que nous souhaitons afficher sur la vue dans le modèle, ces ajouts sont décomposés en 4 blocs, référençant les APIs distinctes.

Les 3 premiers blocs sont assez rudimentaires, ne s’agissant que de simples initialisations de variables, mais le 4e bloc est assez intéressant, car il contient des calculs permettant d’afficher le nombre de cas et décès journaliers de l’épidémie de coronavirus en France, donnée dont j’avais prévu l’affichage dans les maquettes.

L’api des cas de covid-19 ne permettait pas un retour des cas/décès journaliers mais totaux de toute l’épidémie, afin donc d’avoir ces valeurs journalières, j’ai soustrait le nombre de cas totaux du jour précédent la requête aux nombres de cas totaux du jour de la requête.

Ainsi, mes deux modèles étaient complétés, il me fallait maintenant créer mes vues et mes contrôleurs.

J’ai commencé par créer ma vue « SelectVille.jsp », constituée, en plus d’un header et d’un footer classique, d’un formulaire permettant de sélectionner une ville dans une liste déroulante contenant toutes les villes de France, ainsi qu’un bouton de validation, qui envoyait la donnée du formulaire (le nom de la ville que l’utilisateur sélectionne et son code Insee).

La ville choisie était donc envoyée lors du clic sur le bouton de validation vers le contrôleur « C

**[MANQUE CONTROLLEURS+2E VUE+PARSING JSON]**

**[ENSUITE : RENDU COMPTE DU PB DES VILLES ET DU CODE INSEE]**

Pour faire un appel le plus précis possible, j'avais l'idée de tout d’abord chercher la ville que l'utilisateur choisissait en fonction des données qu'il mettrait dans le formulaire de la première page. Afin que cela soit possible, Il me fallait un fichier contenant toutes les villes de France ainsi que certaines informations les concernant, permettant d’effectuer une recherche dans ce-dit fichier en fonction de ce que l'utilisateur entrais comme valeur. En cherchant brièvement sur github, j'ai trouvé un fichier Json d'environ 200000 lignes, composé de l'entièreté des villes françaises ainsi que des informations relatives à ces villes comme le code INSEE, le code postal, le nom de la commune, etc. (le lien du repo github est en annexe).

Lorsque j'ai trouvé ce fichier il me fallait faire un choix : fallait-il que je crée une base de données à partir du Json, contenant toutes les villes de France et leurs informations, ou était-il possible de garder ce fichier tel quel, malgré la probabilité d'un temps de latence lors de la recherche à cause du parsing d’un fichier très lourd.

Je n'ai pas voulu prendre ce choix seul, j'ai donc demandé à mes collègues quelle solution leur semblait adéquate dans le contexte de mon projet. Après délibération nous avons convenu que de garder le fichier tel quel ne serait pas une bonne idée mais qu'une base de données serait peut-être excessive pour un projet de démonstration. La solution que nous avons envisagée est donc la suivante : modifier le fichier json afin de garder les 10000 plus grandes villes de France, réduisant grandement le temps de charment lors du parsing du fichier, tout en gardant la possibilité d’effectuer des recherches des plus grandes villes de France.

Nous avions aussi pensé à stocker le fichier Json en cache dans le navigateur, mais cette solution n'était pas envisageable à cause du fait que l'application ne serait lancée que très rarement pour les démonstrations (c'est à dire que le stockage en cache devra se faire trop régulièrement pour que ce soit efficace).

Afin donc de modifier le fichier automatiquement, j'ai créé un script dont le but était de supprimer toutes les lignes après les dix millièmes plus grandes villes. À noter que par « grande » j'entends ici contenant le plus grand nombre d'habitants. Le script est retrouvable en annexe.

Le fichier ainsi créé me permet donc de récupérer les informations précise d'une ville demandée par l'utilisateur dans le premier formulaire de choix. Cette récupération s'organise grâce à une fonction de recherche via un parsing json en fonction des attributs que l'utilisateur à rempli dans le formulaire.

Pendant ce projet malgré le fait que je sois seul, mes collègues, Lihn et Stéphane, m’ont été d'une grande aide et m’ont donné un regard objectif sur mes idées et mes états d’avancement.

### Quelles compétences validées par le projet ?

### Renseignement sur la compétence et bonnes pratiques

## Projet INRAE

### Explication en détail du projet

### Contexte de réalisation, pourquoi avoir besoin de cette compétence dans l’activité? Comment ça a été réalisé et assimilé ? Quelles ressources utilisées ? As-t’elle été traversée par des évolutions majeures ces 10 dernières années ?

### Renseignement sur la compétence et bonnes pratique

## Projet Java

### Explication en détail du projet

### Contexte de réalisation, pourquoi avoir besoin de cette compétence dans l’activité? Comment ça a été réalisé et assimilé ? Quelles ressources utilisées ? As-t’elle été traversée par des évolutions majeures ces 10 dernières années ?

### Renseignement sur la compétence et bonnes pratique

# Contexte de réalisation

# Conclusion

## Résumé

## Ouverture sur l’avenir

# Annexe

Lien de l’article Auth0 : <https://docs.google.com/document/d/1ix73uuw6ou-B6POphNXcAZHEMKA5nOYnNBFuzg0x0R8/edit#heading=h.rozdwj5vcws4>

Lien repo github : <https://github.com/high54/Communes-France-JSON>

InfoCorona.java :

Une image contenant texte, capture d’écran

Description générée automatiquement

InfoMeteo.java :

Une image contenant texte, capture d’écran, violet, violette

Description générée automatiquement